Quality Assurance Through Software Engineering

16

Systems Analysis and Design, 7e
Kendall & Kendall

Learning Objectives

- Recognize the importance of users and analysts taking a total quality approach to the entire SDLC
- Create structure charts to design modular, top-down systems
- Use a variety of techniques to improve the quality of software design and maintenance
- Understand the importance of running a variety of tests during systems development to identify unknown problems

Approaches to Quality Assurance

- Securing total quality assurance through designing systems and software with a top-down and modular approach
- Documenting software with appropriate tools
- Testing, maintaining, and auditing software

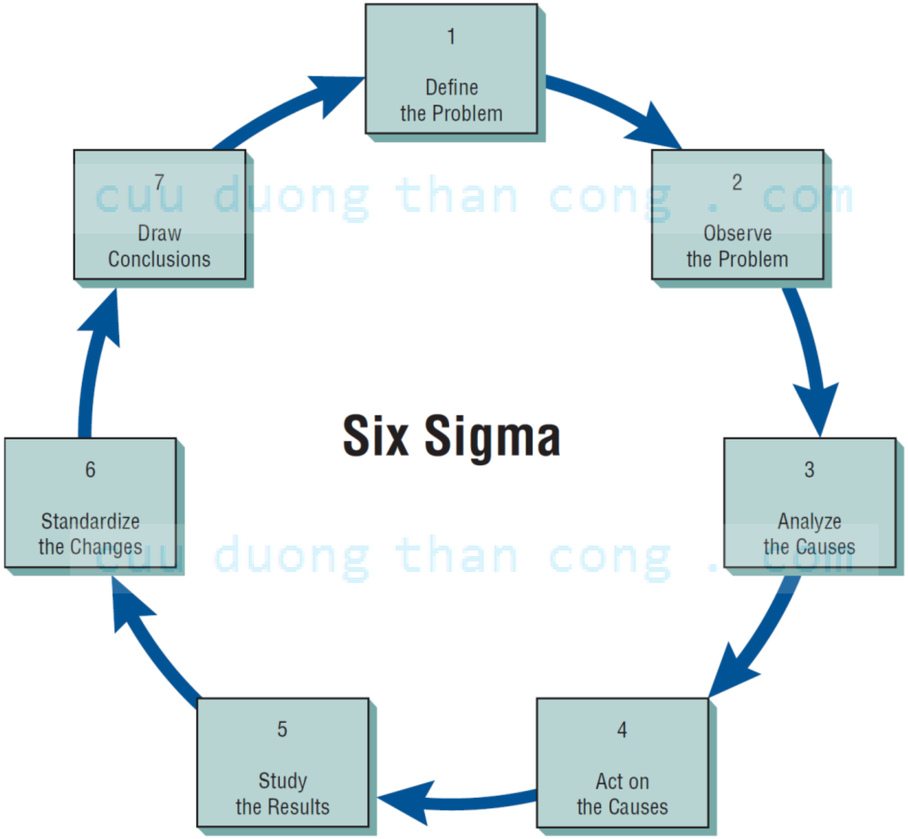
Major Topics

- Six Sigma
- Quality assurance
- Walkthroughs
- Structure charts
- Modules
- Data and control passing
- Documentation
- Testing

Six Sigma

- A culture built on quality
- Uses a top-down approach
- Project leader is called a Black Belt
- Project members are called Green Belts
- Master Black Belts have worked on many projects and are available as a resource to project teams

Figure 16.1 Every systems analyst should understand the methodology and philosophy of Six Sigma



Responsibility for Total Quality Management

- Full organizational support of management must exist
- Early commitment to quality from the analyst and business users

Structured Walkthroughs

- One of the strongest quality management actions is to do structured walkthroughs routinely
- Use peer reviewers to monitor the system's programming and overall development
- Point out problems
- Allow the programmer or analyst to make suitable changes

Involved in Structured Walkthroughs

- The person responsible for the part of the system being reviewed
- A walkthrough coordinator
- A programmer or analyst peer
- A peer who takes notes about suggestions

Systems Design and Development

- Bottom-up
- Top-down
- Modular

Bottom-Up Design

- Identifying the processes that need computerization as they arise
- Analyzing them as systems
- Either coding or purchasing packaged software to meet the immediate problem

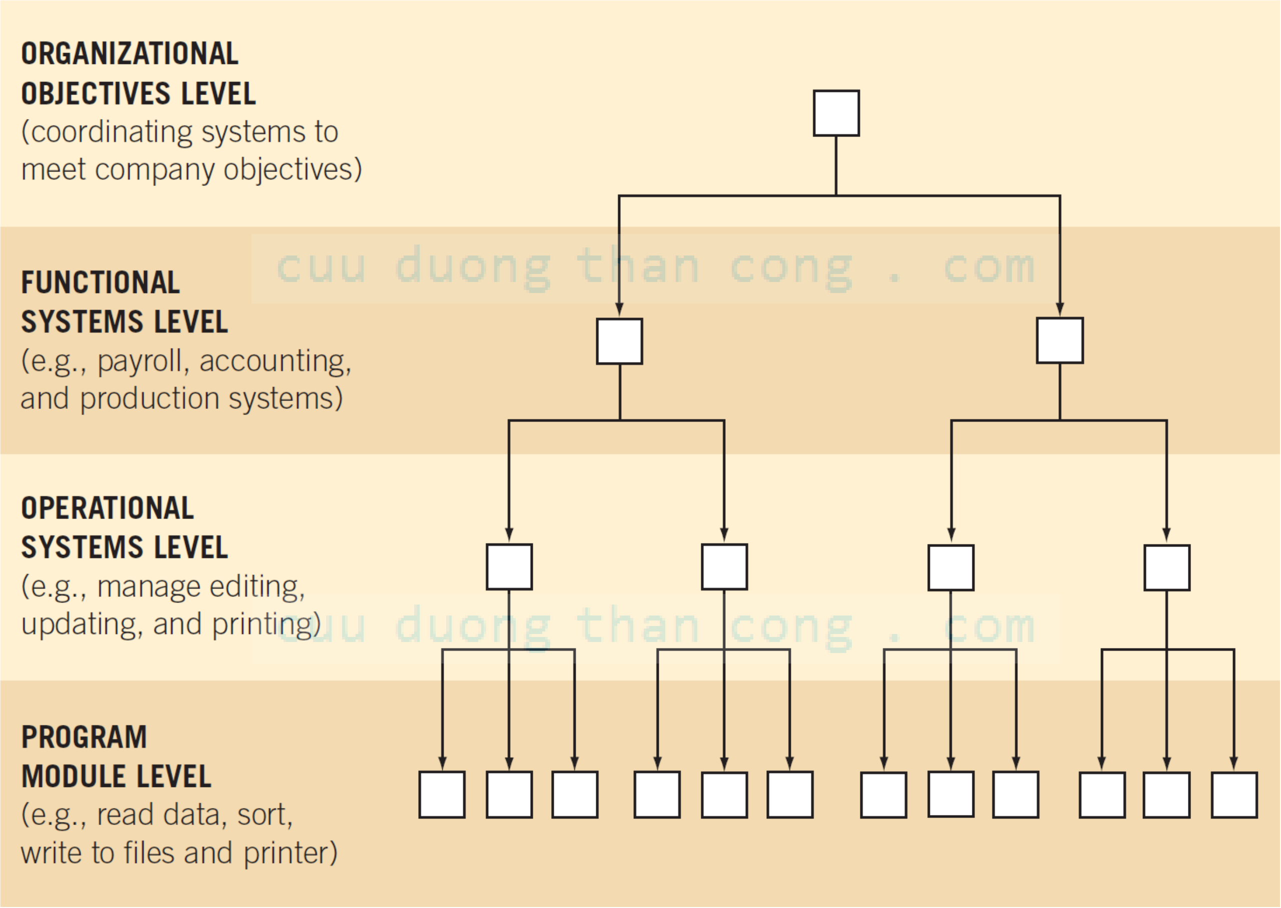
Disadvantages of a Bottom-Up Approach

- There is a duplication of effort in purchasing software, and entering data
- Worthless data are entered into the system
- Overall organizational objectives are not considered and hence cannot be met

The Top-Down Approach

- Top-down design allows the systems analyst to ascertain overall organizational objectives and how they are best met in an overall system
- The system is divided into subsystems and their requirements

Figure 16.3 Using the top-down approach to first ascertain overall organizational objectives


ORGANIZATIONAL OBJECTIVES LEVEL
(coordinating systems to meet company objectives)
FUNCTIONAL SYSTEMS LEVEL
(e.g., payroll, accounting, and production systems)
OPERATIONAL SYSTEMS LEVEL
(e.g., manage editing, updating, and printing)
PROGRAM MODULE LEVEL
(e.g., read data, sort, write to files and printer)

Advantages of the Top-Down Approach

- Avoiding the chaos of attempting to design a system all at once
- Enables separate systems analysis teams to work in parallel on different but necessary subsystems
- Prevents losing sight of what the system is suppose to do

Disadvantages of the Top-Down Approach

- There is a danger that the system will be divided into the wrong subsystems
- Once subsystem divisions are made, their interfaces may be neglected or ignored
- The subsystems must be eventually reintegrated

Modular Development

- Breaking the programming into logical, manageable portions or modules
- Works well with top-down design
- Each individual module should be functionally cohesive, accomplishing only one function

Advantages of Modular Programming

- Modules are easier to write and debug
- Modules are easier to maintain
- Modules are easier to grasp because they are self-contained subsystems

Guidelines for Modular Programming

- Keep each module to a manageable size
- Pay particular attention to the critical interfaces
- Minimize the number of modules the user must modify when making changes
- Maintain the hierarchical relationships set up in the top-down phases

Modularity in the Windows Environment

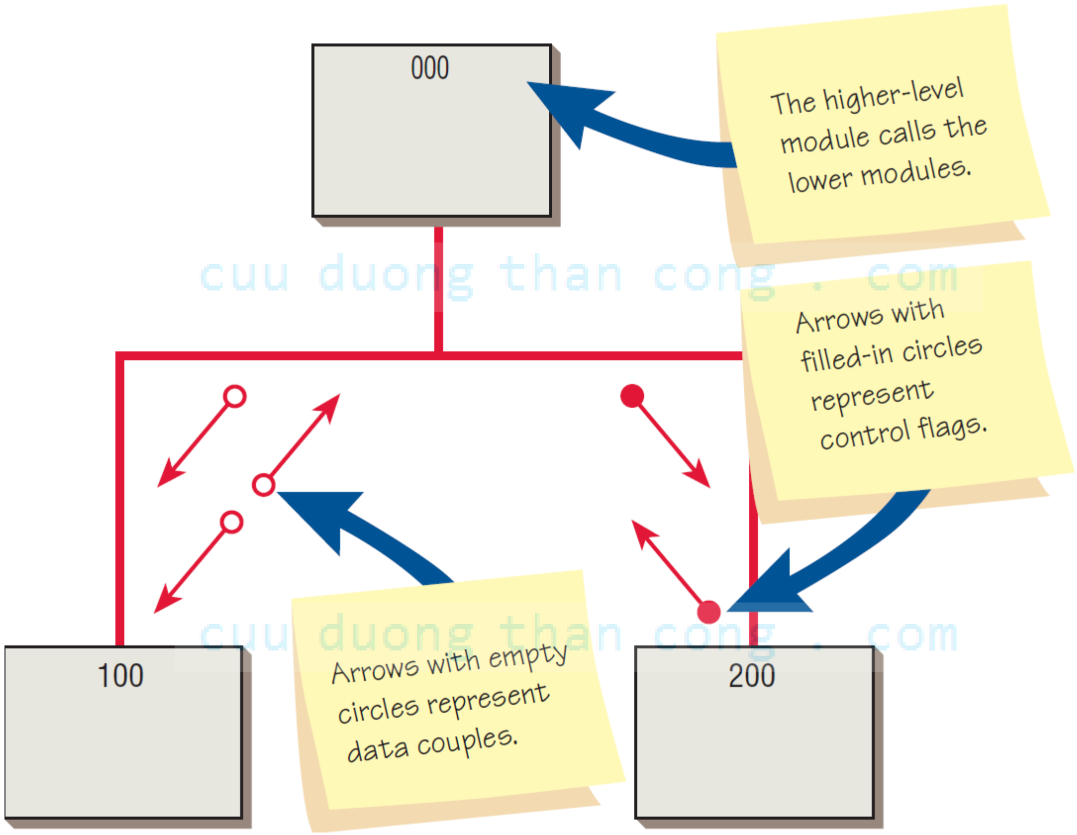
There are two systems to link programs in Microsoft Windows:

- Dynamic Data Exchange (DDE) shares code by using Dynamic Link Library (DLL) files
- Object Linking and Embedding (OLE) ties in application data and graphics

Using Structure Charts to Design Systems

- The recommended tool for designing a modular, top-down system is a structure chart
- A structure chart is simply a diagram consisting of rectangular boxes, representing the modules, and connecting lines

Figure 16.4 A structure diagram encourages top-down design using modules



The higher-level module calls the lower modules.

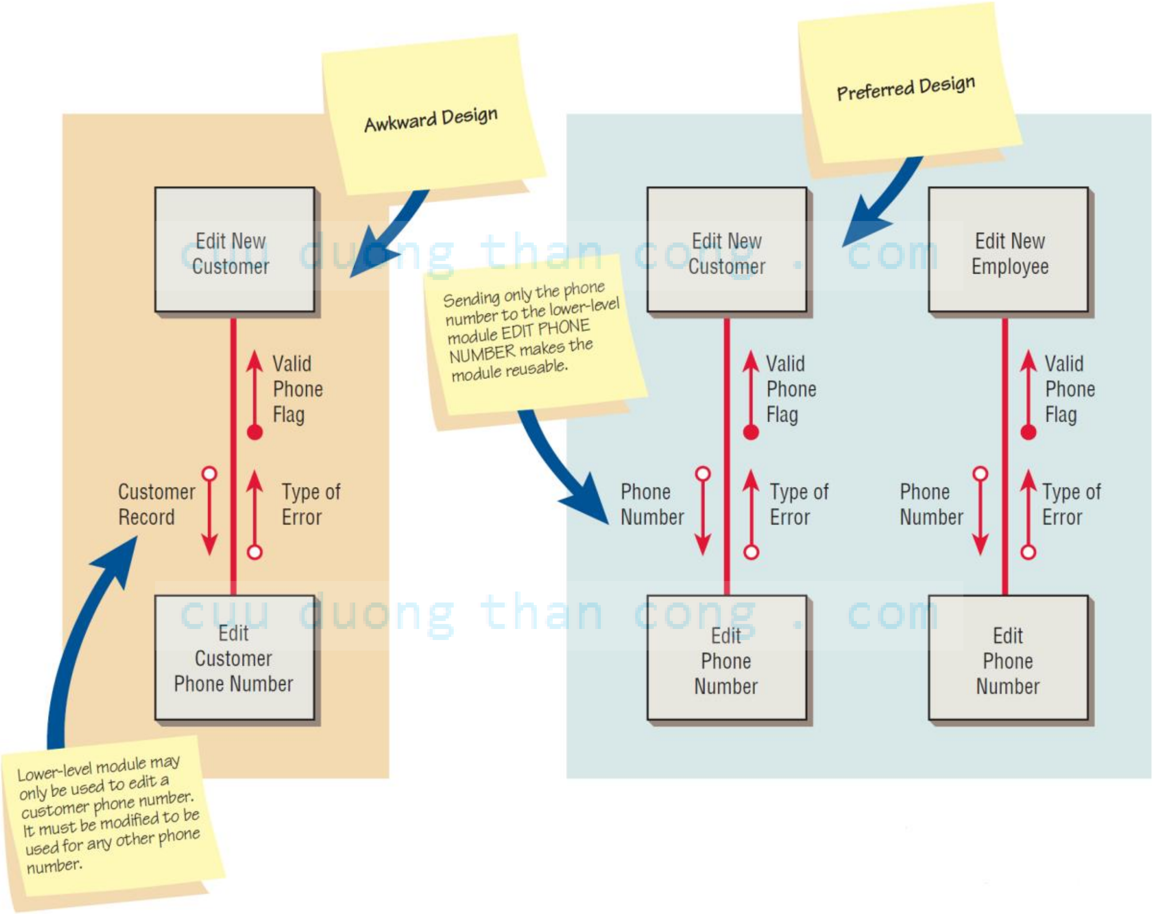
Arrows with filled-in circles represent control flags.

Arrows with empty circles represent data couples.

Data Couples and Control Flags

- The fewer control flags and data couples in the system, the easier it is to change the system
- Control flags govern which portion of a module is to be executed and associated with IF…THEN…ELSE…and other similar statements
- Data coupling is when only data required is passed through the data couple
- Stamp coupling is when excessive data is passed through the data couple

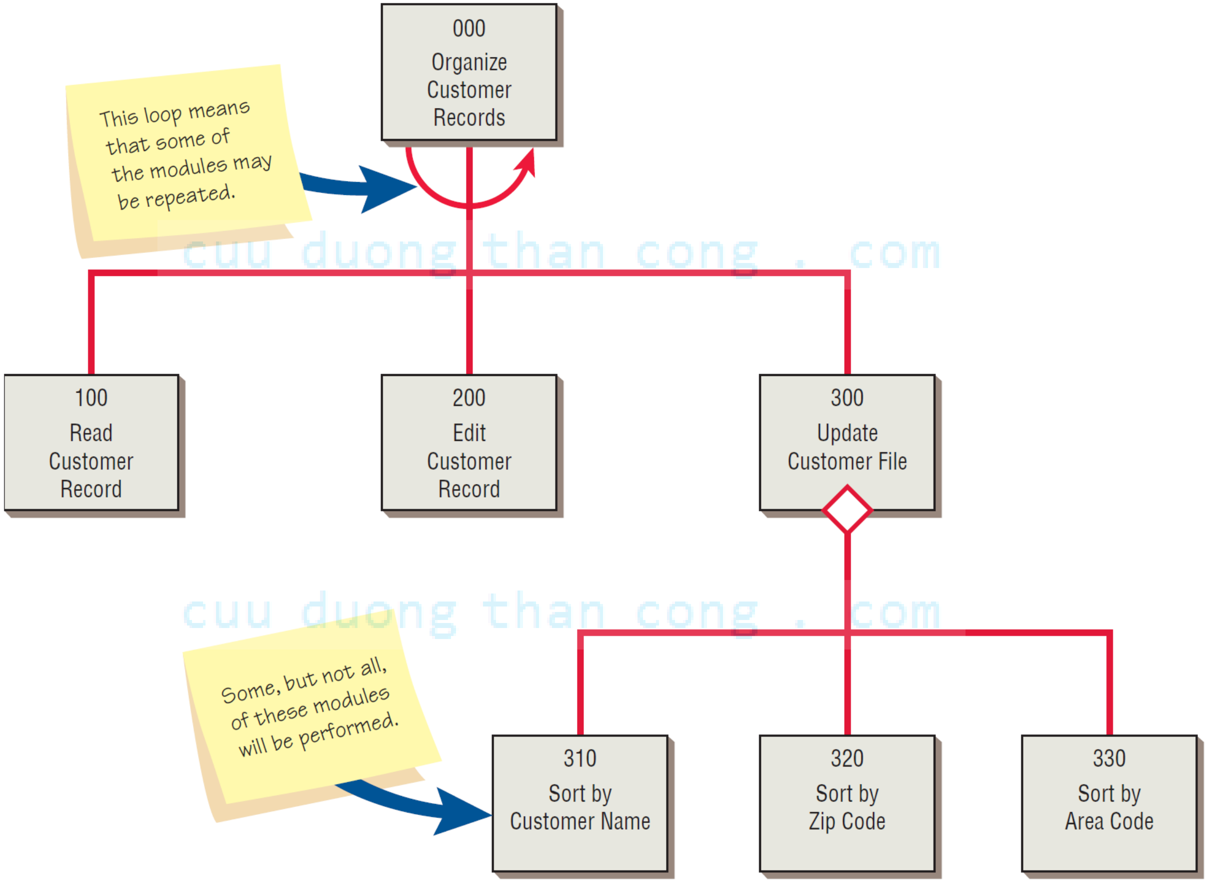
Figure 16.8 Creating reusable modules



Sending only the phone number to the lower-level module EDIT PHONE NUMBER makes the module reusable.

Lower-level module may only be used to edit a customer phone number. It must be modified to be used for any other phone number.

Figure 16.9 The loop and diamond are two symbols that indicate special action in a structure chart


000
Organize Customer Records
This loop means that some of the modules may be repeated.
100
Read Customer Record
200
Edit Customer Record
300
Update Customer File
Some, but not all, of these modules will be performed.
310
Sort by Customer Name
320
Sort by Zip Code
330
Sort by Area Code

Drawing a Structure Chart

- A data flow diagram may be used to create a structure chart
    - Indicates the sequence of the modules
    - Indicates modules subordinate to a higher module

Types of Modules

- Control modules
- Transformational modules
- Functional modules

Control Modules

- Found near the top of the structure chart and contain the logic for performing the lower-level modules
- May or may not be represented on the data flow diagram
- Usually contains IF, Perform, and DO statements
- Should not be very large in size

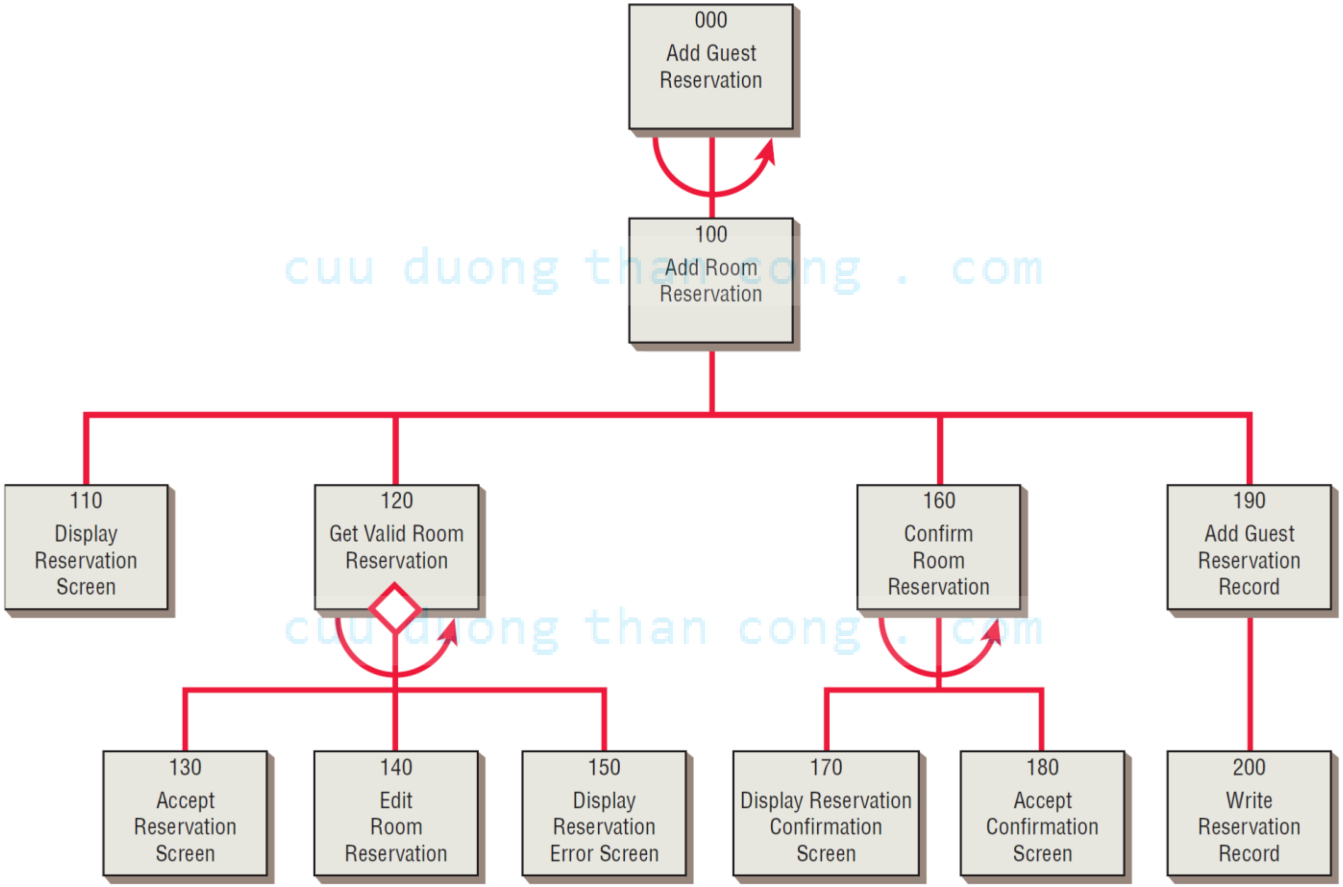
Transformational Modules

- Created from a data flow diagram
- Usually perform only one task
- Have mixed statements, IF and PERFORM or DO statements and many detailed statements such as MOVE and ADD

Functional Modules

- Perform only one task
- The easiest to code, debug, and maintain

Figure 16-13 A structure chart for adding hotel guest reservations online


000
Add Guest Reservation
100
Add Room Reservation
110
Display Reservation Screen
120
Get Valid Room Reservation
160
Confirm Room Reservation
190
Add Guest Reservation Record
130
Accept Reservation Screen
140
Edit Room Reservation
150
Display Reservation Error Screen
170
Display Reservation Confirmation Screen
180
Accept Confirmation Screen
200
Write Reservation Record

Module Subordination

- A subordinate module is one lower on the structure chart called by another module higher in the structure
- Allowing a lower-level module to perform a task not required by the calling module is called improper subordination

Software Engineering and Documentation

- The total quality assurance effort requires that programs be documented properly
- Documentation
  - Allows you to "see" the system without having to interact with it
  - Provides an overview of the system itself
  - Shortens time to perform maintenance

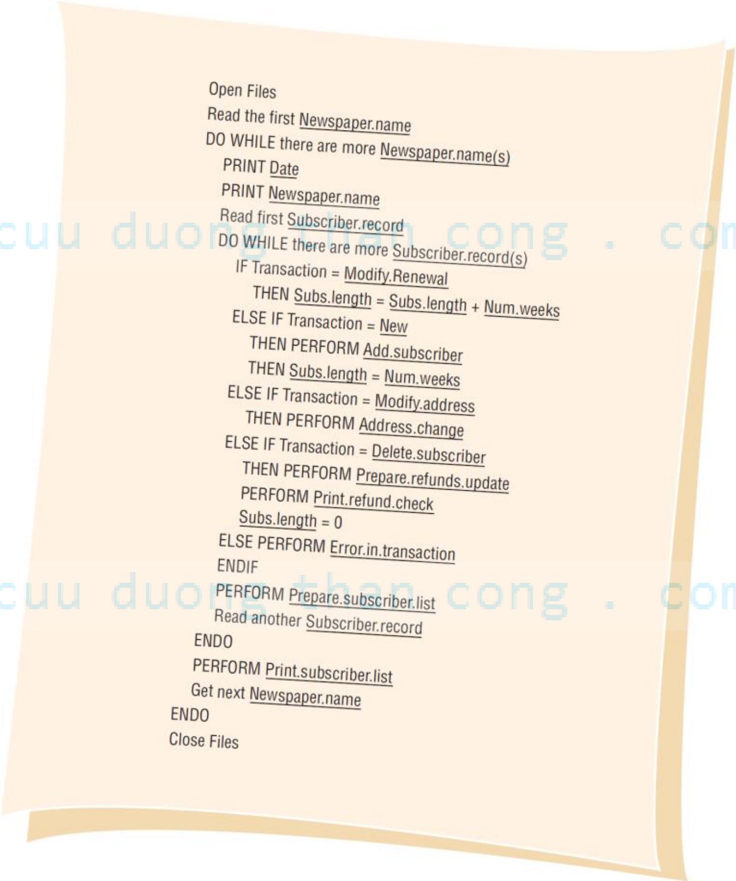
System Documentation

- Pseudocode
- Procedure manuals
- The FOLKLORE method

Pseudocode

- Similar to structured English
- It is not a particular type of programming code, but it can be used as an intermediate step for developing program code

Figure 16.16 Using pseudocode to depict a subscription update service for a newspaper conglomerate



```
Open Files
Read the first Newspaper.name
DO WHILE there are more Newspaper.name(s)
    PRINT Date
    PRINT Newspaper.name
    Read first Subscriber.record
    DO WHILE there are more Subscriber.record(s)
        IF Transaction = Modify.Renewal
            THEN Subs.length = Subs.length + Num.weeks
        ELSE IF Transaction = New
            THEN PERFORM Add.subscriber
            THEN Subs.length = Num.weeks
        ELSE IF Transaction = Modify.address
            THEN PERFORM Address.change
        ELSE IF Transaction = Delete.subscriber
            THEN PERFORM Prepare.refunds.update
            PERFORM Print.refund.check
            Subs.length = 0
        ELSE PERFORM Error.in.transaction
        ENDIF
        PERFORM Prepare.subscriber.list
        Read another Subscriber.record
    ENDO
    PERFORM Print.subscriber.list
    Get next Newspaper.name
ENDO
Close Files
```

Procedure Manuals

- The English-language component of documentation
- Key sections
  - Introduction
  - How to use the software
  - What to do if things go wrong
  - A technical reference section
  - An index
  - Information on how to contact the manufacturer

Procedure Manuals (Continued)

- Procedure manual complaints
  - They are poorly organized
  - It is hard to find needed information
  - The specific case in question does not appear in the manual
  - The manual is not written in plain English

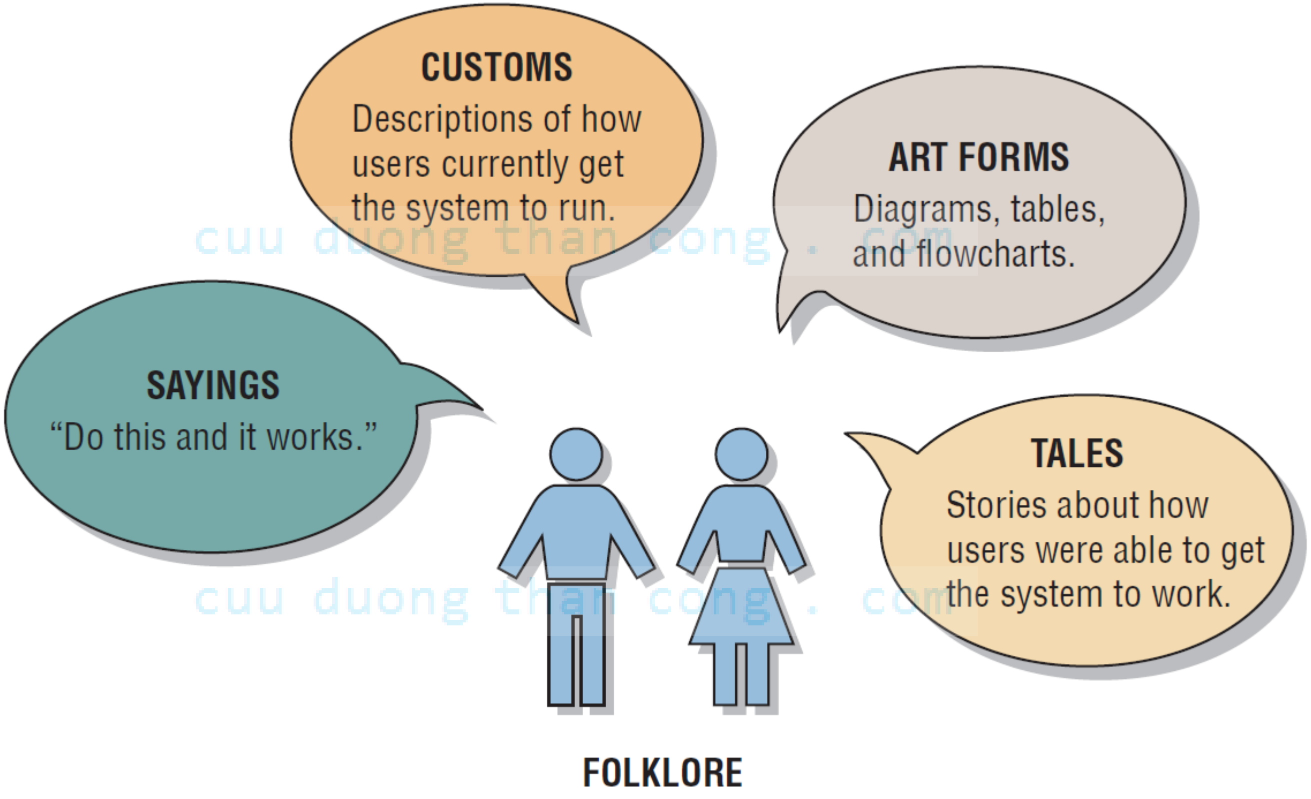
Web Documentation

- FAQ (Frequently Asked Questions)
- Help desks
- Technical support
- Fax-back services
- Downloading updates

The FOLKLORE Method

- Collects information in the categories
  - Customs
  - Tales
  - Sayings
  - Art forms

Figure 16.18 Customs, tales, sayings, and art forms used in the FOLKLORE method of documentation apply to information systems


CUSTOMS
Descriptions of how users currently get the system to run.
ART FORMS
Diagrams, tables, and flowcharts.
SAYINGS
"Do this and it works."
TALES
Stories about how users were able to get the system to work.
FOLKLORE

Choosing a Design and Documentation Technique

- Is it compatible with existing documentation
- Is it understood by others in the organization
- Does it allow you to return to working on the system after you have been away from it for a period of time

Choosing a Design and Documentation Technique (Continued)

- Is it suitable for the size of the system you are working on
- Does it allow for a structured design approach if that is considered to be more important than other factors
- Does it allow for easy modification

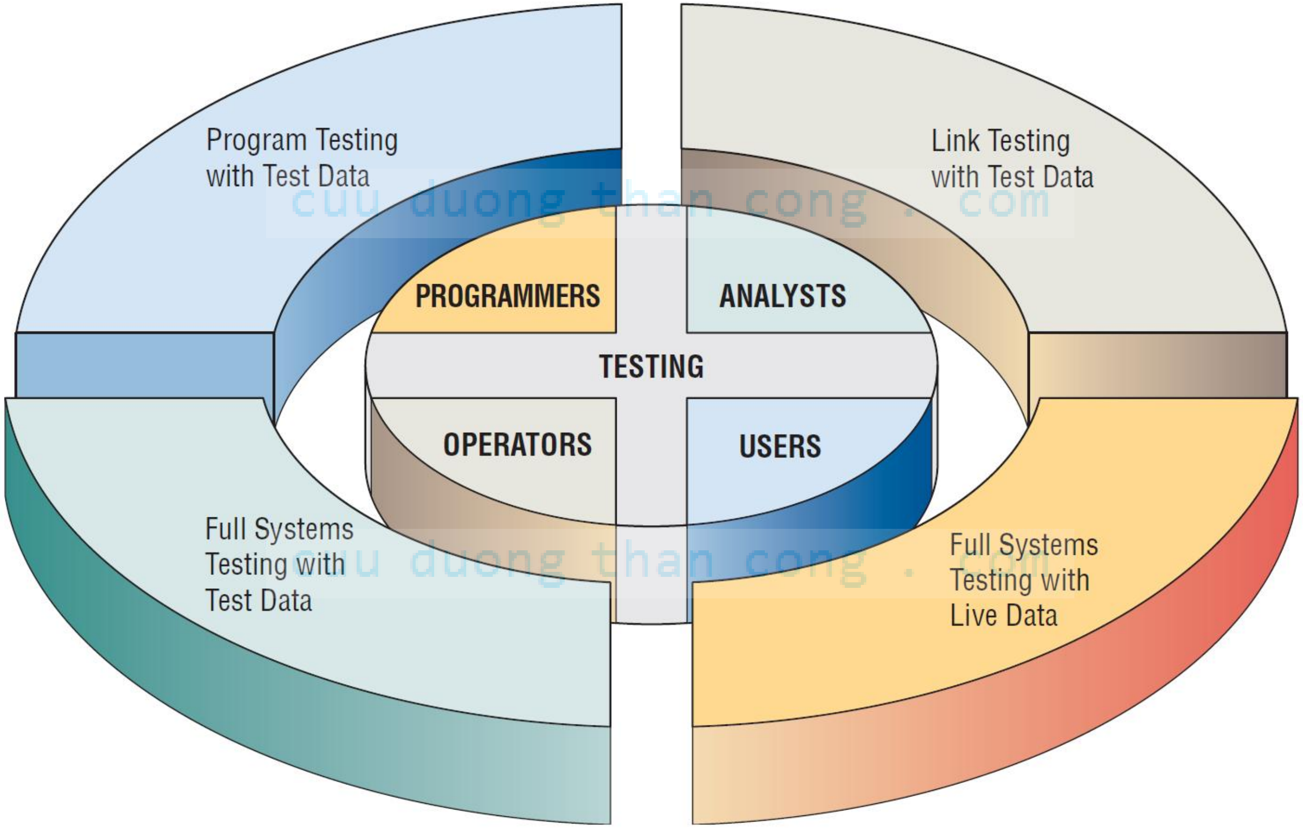
Testing, Maintenance, and Auditing

- The testing process
- Maintenance practices
- Auditing

The Testing Process

- Program testing with test data
- Link testing with test data
- Full system testing with test data
- Full system testing with live data

Figure 16.19 Programmers, analysts, operators, and users all play different roles in testing software and systems


Program Testing with Test Data
Link Testing with Test Data
PROGRAMMERS
ANALYSTS
TESTING
OPERATORS
USERS
Full Systems Testing with Test Data
Full Systems Testing with Live Data

Program Testing with Test Data

- Desk check programs
- Test with both valid and invalid data
- Check output for errors and make any needed corrections

Link Testing with Test Data

- Also referred to as string testing
- Checks to see if programs that are interdependent actually work together as planned
- Test for normal transactions
- Test with invalid data

Full System Testing with Test Data

- Adequate documentation in procedure manuals
- Are procedure manuals clear enough
- Do work flows actually “flow”
- Is output correct and do users understand this output

Full System Testing with Live Data

- Comparison of the new system’s output with what you know to be correctly processed output
- Only small amounts of live data are used

Maintenance Practices

- Reduce maintenance costs
- Improve the existing software
- Update software in response to the changing organization
- Ensure channels for feedback
- Classification scheme

Auditing

- Having an expert who is not involved in setting up or using the system examine information in order to ascertain its reliability
- There are internal and external auditors
- Internal auditors study the controls used in the information system to make sure that they are adequate
- External auditors are used when the information system processes data that influences a company’s financial statements

Summary

- TQM
  - Designing systems and software with a top-down, modular approach
  - Designing and documenting systems and software using systematic methods
  - Testing systems and software so that they can be easily maintained and audited

Summary (Continued)

- Six Sigma
  - Define the problem
  - Observe the problem
  - Analyze the causes
  - Act on the causes
  - Study the results
  - Standardize the changes
  - Draw conclusions

Summary (Continued)

- Structure charts
- Structure chart modules
  - Control
  - Transformational
  - Functional
- Documentation
  - Pseudocode
  - Procedure manuals
  - FOLKLORE

Summary (Continued)

- Testing
- System Maintenance
- Auditing