Systems Analysis and Design 11th Edition

Managing Systems Implementation

Chapter Objectives

- Explain the importance of software quality assurance and software engineering
- Describe application development using structured, object-oriented, and agile methods
- Draw a structure chart showing top-down design, modular design, cohesion, and coupling
- Explain the coding process
- Explain unit, integration, and system testing

Chapter Objectives (Cont.)

- Differentiate between program, system, operations, and user documentation
- List the main steps in system installation and evaluation
- Develop training plans for various user groups, compare in-house and vendor training options, and describe effective training techniques
- Describe data conversion and changeover methods
- Explain post-implementation evaluation and the final report to management

Structured Application Development

- **Structure Charts**
  - Display program modules and the relationships among them
  - Module – Represented by a rectangle
  - Types of modules
    - **Control**: Higher-level module
    - **Subordinate**: Lower-level modules
    - **Library**: Reusable code that can be invoked from more than one point chart

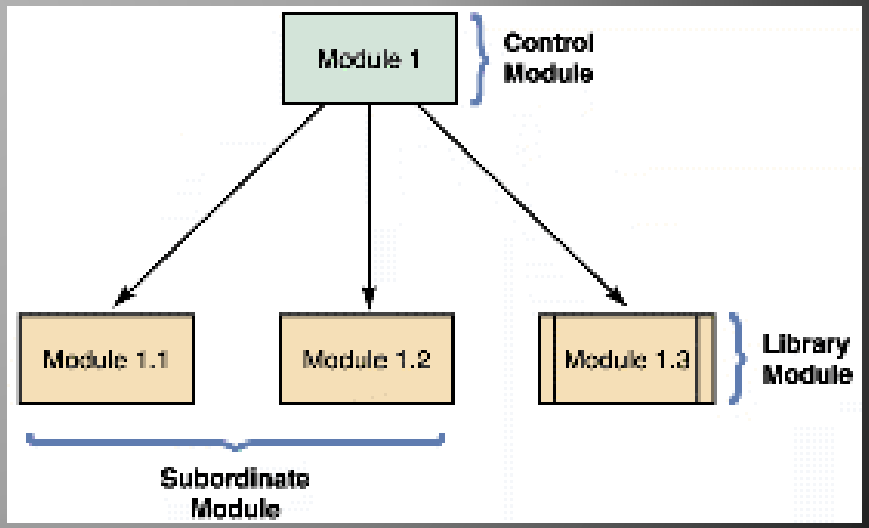


FIGURE 11-8 An example of structure chart modules.

Structured Application Development (Cont. 1)

- **Data couple**: Represented by an arrow with an empty circle
  - Shows data that one module passes to another
- **Control couple**
  - Represented by an arrow with a filled circle
  - Shows a **status flag**

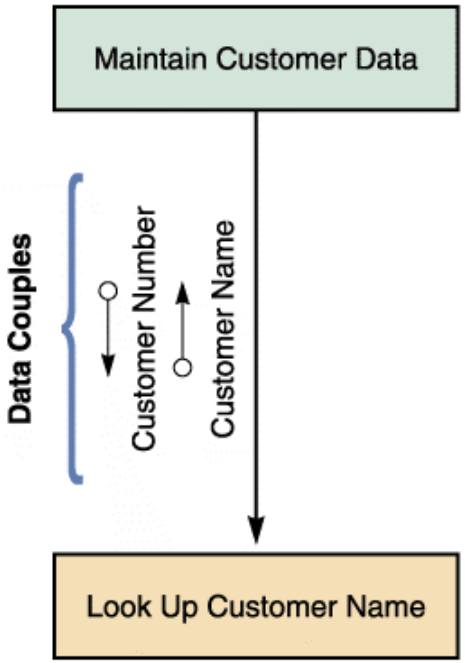


FIGURE 11-9 An example of a structure chart data.

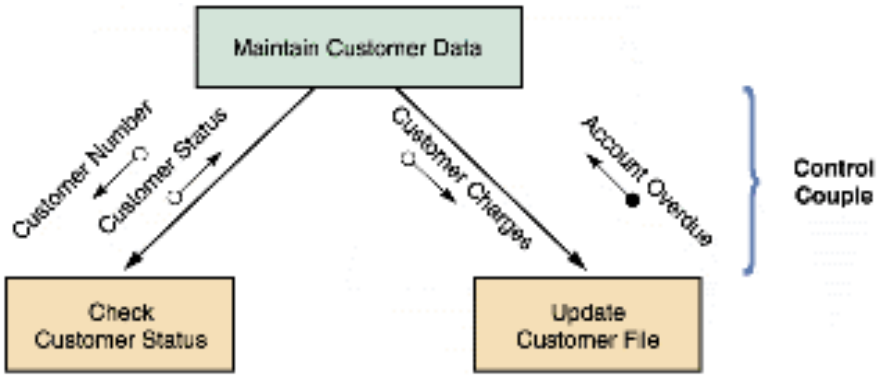


FIGURE 11-10 An example of a structure chart control couple.

Structured Application Development (Cont. 2)

- **Condition**: Indicates that a control module determines which subordinate modules will be invoked
  - Represented by a line with a diamond on one end
- **Loop**: Indicates that one or more modules are repeated
  - Represented by a curved arrow

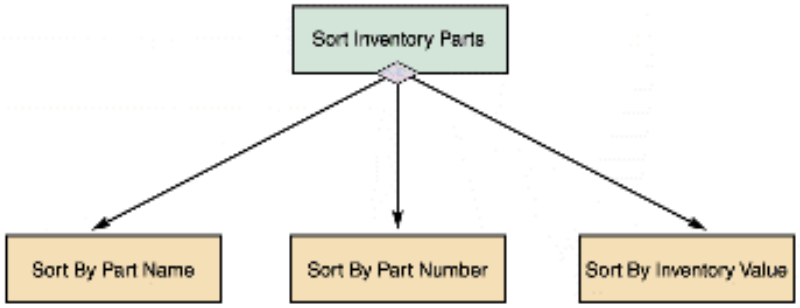


FIGURE 11-11 The diagram shows a control module that triggers three subordinate modules.

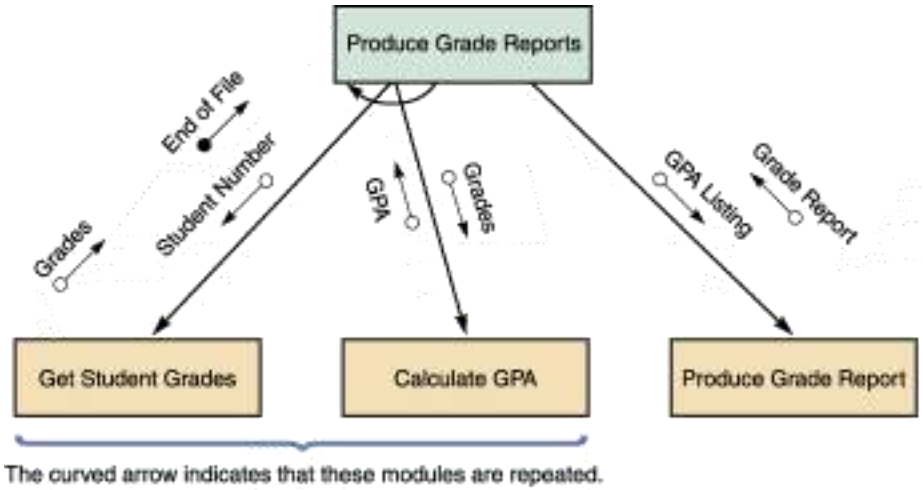


FIGURE 11-12 The diagram shows a structure chart loop with two repeating modules.

Structured Application Development (Cont. 3)

- **Cohesion and Coupling**
  - **Cohesion** measures a module's scope and processing characteristics
    - A module that performs a single function or task has a high degree of cohesion

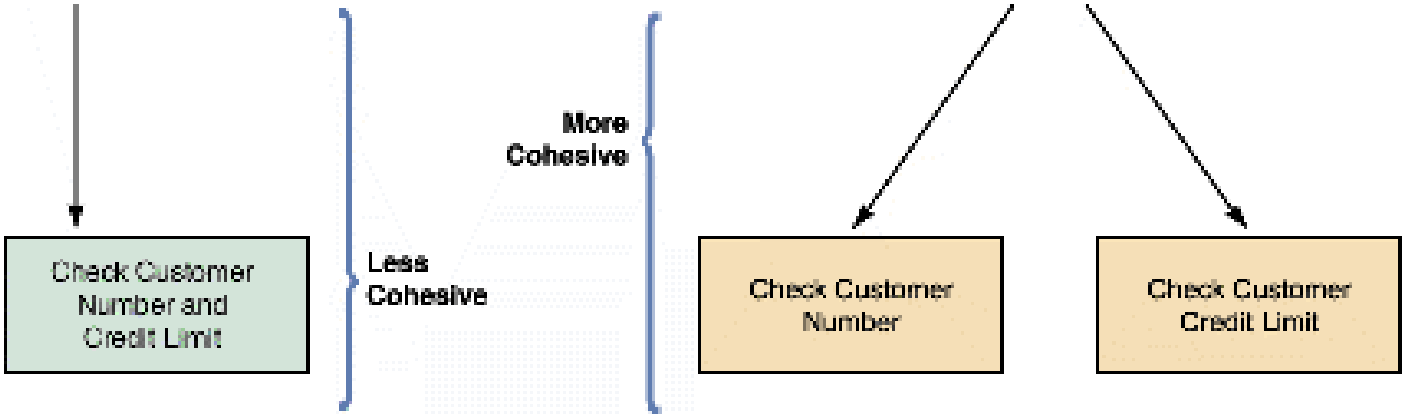


FIGURE 11-13 Two examples of cohesion. Notice that the single module on the left is less cohesive than the two modules on the right.

Structured Application Development (Cont. 4)

- **Coupling** describes the degree of interdependence among modules
  - Modules that are independent are **loosely coupled**
    - Loosely coupled modules are easier to maintain and modify
  - In **tightly coupled** modules, one module is linked to internal logic contained in another module

Structured Application Development (Cont. 5)

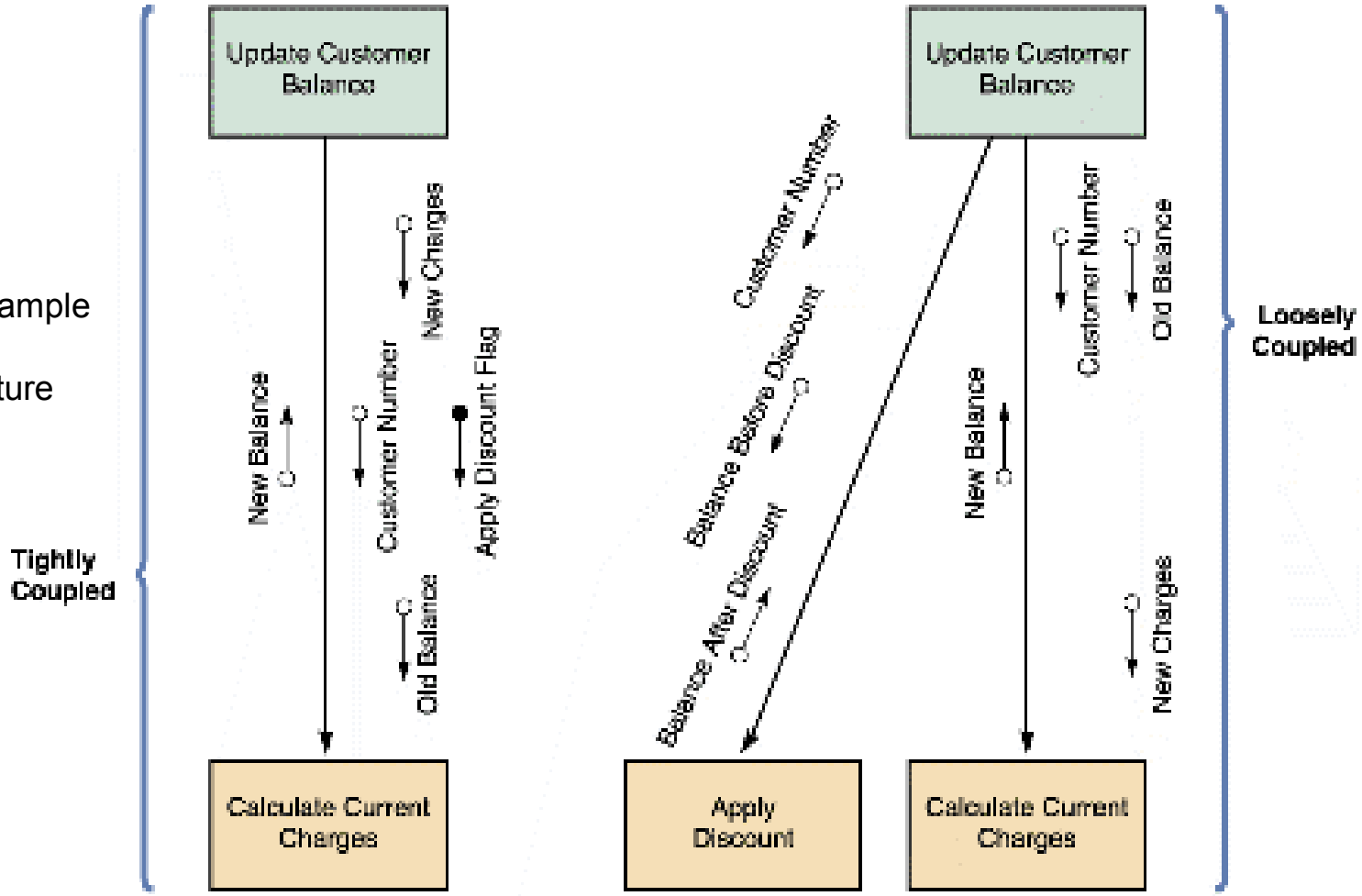


FIGURE 11-14 An example of tightly coupled and loosely coupled structure charts.

Structured Application Development (Cont. 6)

- **Drawing a Structure Chart**
  - Step 1 – Review DFDs for accuracy and completeness
  - Step 2 – Identify the program modules and determine control-subordinate relationships
  - Step 3 – Add couples, loops, and conditions
  - Step 4 – Analyze the structure chart and ensure that it is consistent with all previous documentation

Structured Application Development (Cont. 7)

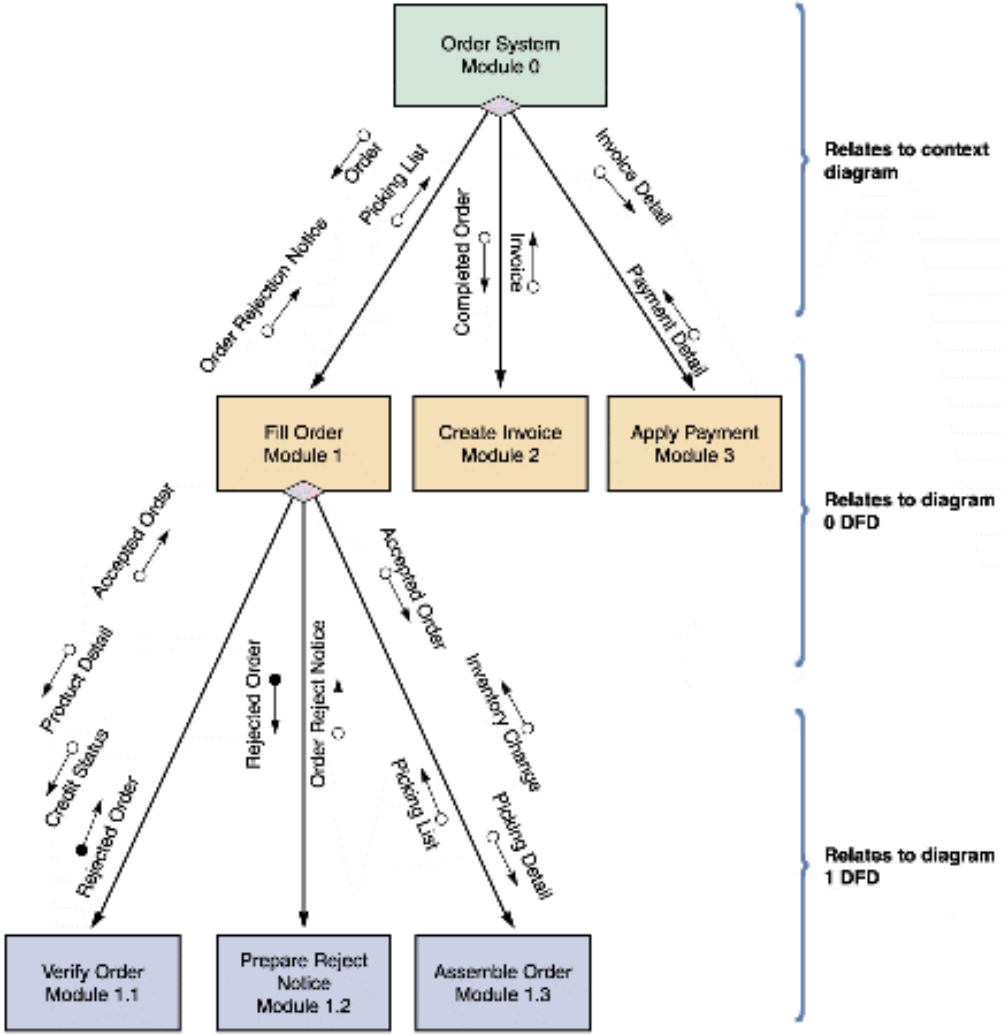


FIGURE 11-15 A structure chart based on the order system DFDs in Chapter 5. The three-level structure chart relates to the three DFD levels.

Object-Oriented Application Development

- **Characteristics of Object-Oriented Application Development**
  - Individual object instances belong to classes of objects with similar characteristics
  - The relationship and interaction among classes are described using a class diagram
    - Class diagrams include class attributes and **methods**
    - Object relationship diagrams provide an overview of object interaction

Object-Oriented Application Development (Cont. 1)

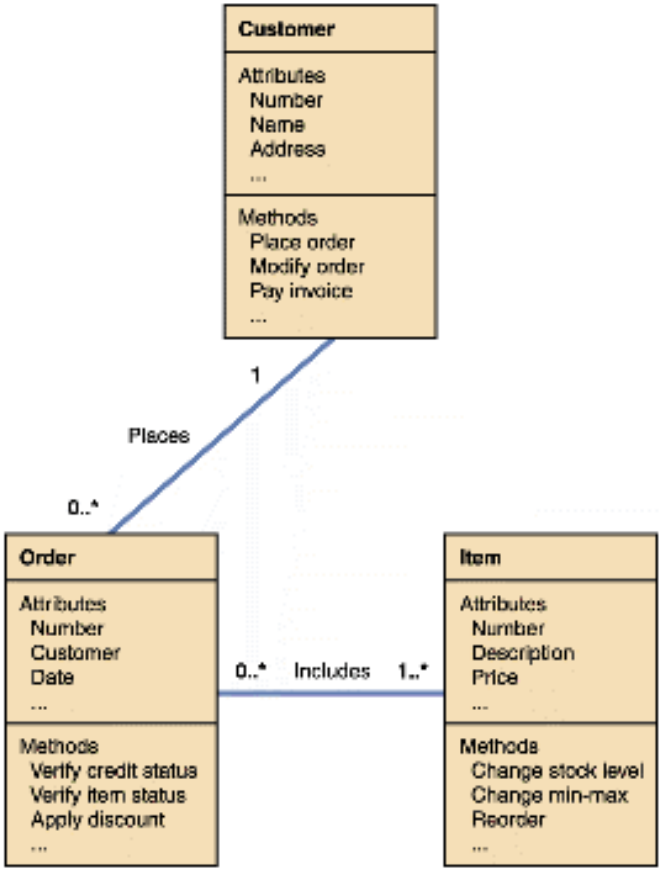
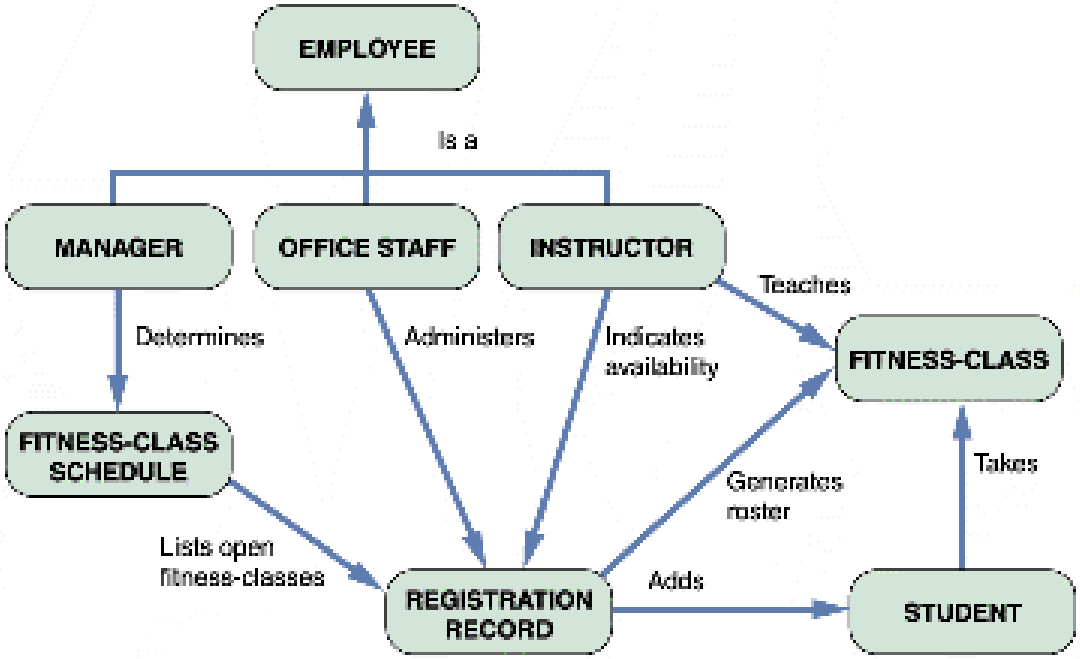


FIGURE 11-16 A simplified class diagram for a customer order processing system.

FIGURE 11-17 An object-relationship diagram for a fitness center.



Object-Oriented Application Development (Cont. 2)

- **Implementation of Object-Oriented Designs**
  - Objective – To translate object methods into program code modules and determine what event or message will trigger the execution of each module
- **Object-Oriented Cohesion and Coupling**
  - Classes should be as loosely coupled as possible
  - An object's methods should also be loosely coupled and highly cohesive

Agile Application Development

- Uses a highly iterative process
- Development team is in constant communication with the primary user (**customer**)
- Based on a quick and nimble development process that easily adapts to change
- Focuses on small teams, intense communication, and rapid development iterations

Agile Application Development

(Cont. 1)

The Values of Extreme Programming

Extreme Programming (XP) is based on values. The rules we just examined are the natural extension and consequence of maximizing our values. XP isn't really a set of rules but rather a way to work in harmony with your personal and corporate values. Start with XP's values listed here then add your own by reflecting them in the changes you make to the rules.

Simplicity: We will do what is needed and asked for, but no more. This will maximize the value created for the investment made to date. We will take small simple steps to our goal and mitigate failures as they happen. We will create something we are proud of and maintain it long term for reasonable costs.

Communication: Everyone is part of the team and we communicate face to face daily. We will work together on everything from requirements to code. We will create the best solution to our problem that we can together.

Feedback: We will take every iteration commitment seriously by delivering working software. We demonstrate our software early and often then listen carefully and make any changes needed. We will talk about the project and adapt our process to it, not the other way around.

Respect: Everyone gives and feels the respect they deserve as a valued team member. Everyone contributes value even if it's simply enthusiasm. Developers respect the expertise of the customers and vice versa. Management respects our right to accept responsibility and receive authority over our own work.

Courage: We will tell the truth about progress and estimates. We don't document excuses for failure because we plan to succeed. We don't fear anything because no one ever works alone. We will adapt to changes when ever they happen.

FIGURE 11-19 The five core values of extreme programming (XP).
Source: ©2009, Don Wells

Agile Application Development
(Cont. 2)

- **Extreme Programming (XP)**
  - Uses pair programming
    - **Pair programming**: Involves two programmers working on the same task on the same computer, where one programs while the other watches
  - **Test-driven development (TDD)**
    - Focuses on end results from the beginning
    - Prevents programmers from straying from their goals
- **User Stories**
  - Short, simple requirements definitions
    - Programmers review user stories to determine the project's requirements, priorities, and scope

Agile Application Development
(Cont. 3)

- **Iterations and Releases**
  - **Release plans** are to be developed by teams
  - User stories are implemented in a series of **iteration cycles**
    - An **iteration planning meeting** is held at the beginning of each cycle
    - Iteration cycles continue until all user stories have been implemented, tested, and accepted

Agile Application Development
(Cont. 4)

- **The Future of Agile Development**
  - Agile methodology is becoming popular for software projects
    - Supporters claim that it speeds up software development and delivers precisely what the customer wants
    - Critics claim that it lacks discipline and produces systems of questionable quality
  - Agile methodology does not work as well for larger projects because of their complexity and the lack of focus on a well-defined end product

Testing The System

- Each program is tested to ensure that it functions correctly
  - Program is compiled using a CASE tool or language compiler, which helps detect **syntax errors**
  - **Desk checking** helps spot **logic errors**
  - Organizations require a **structured walkthrough** or **code review**
    - Groups of three to five IT staff members participate
    - A **design walkthrough** is held to review the interface along with people who will work with the new system and ensure that all necessary features have been included

Testing The System (Cont. 1)

- **Unit testing**
  - Testing of an individual program or module
  - Aims to identify and eliminate:
    - Execution errors
    - Logic errors
  - **Test data** should contain both accurate and erroneous data

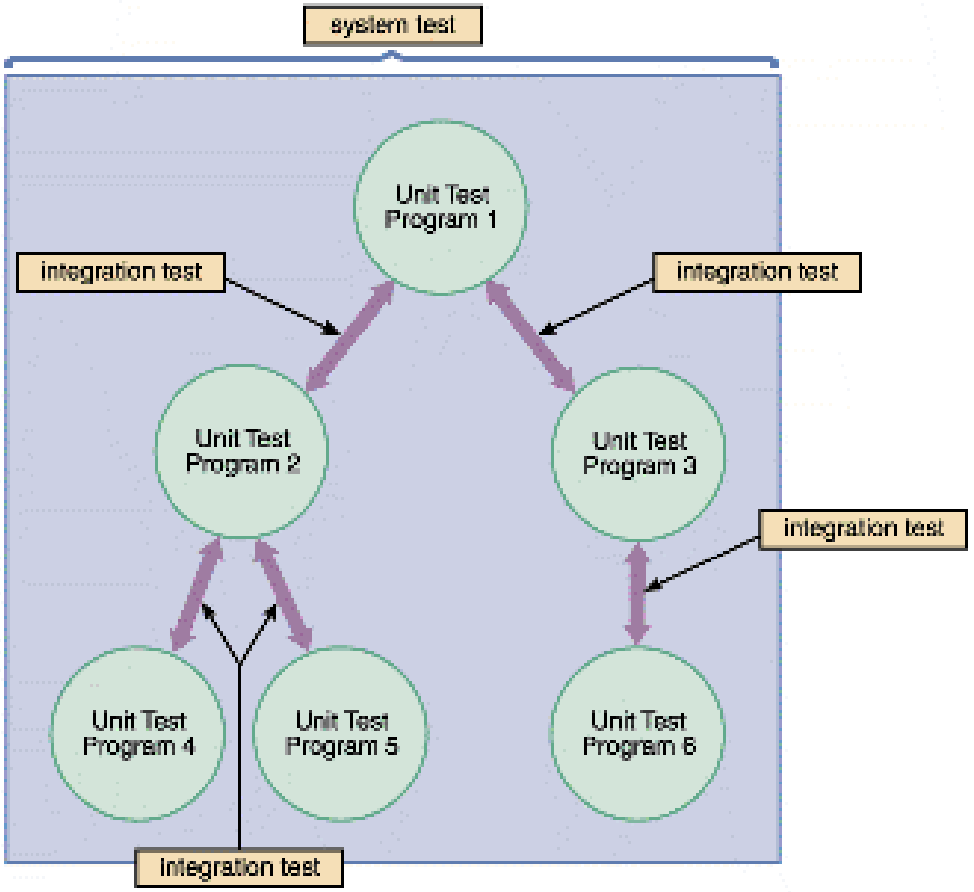


FIGURE 11-20 The first step in testing is unit testing, followed by integration testing, and then system testing.

Testing The System (Cont. 2)

- **Stub testing** is required
  - The programmer:
    - Simulates each program outcome or result
    - Displays a message to indicate whether the program has executed successfully
  - A **test plan** is created during the systems design phase
    - Contains details regarding testing procedures
- **Integration Testing**
  - Testing two or more programs that depend on each other to make sure that the programs work together properly

Testing The System (Cont. 3)

- **System Testing (Acceptance Tests)**
  - Objectives
    - Perform a final test of all programs
    - Verify that the system will handle all data properly
    - Ensure that the IT staff has the documentation and instructions needed to operate the system properly
    - Demonstrate that users can interact with the system successfully
    - Verify that all system components are integrated properly
    - Confirm that the information system can handle predicted volumes of data in a timely and efficient manner

Documentation

- **Program Documentation**
  - Describes the inputs, outputs, and processing logic for all program modules
  - Process starts in the systems analysis phase and continues during systems implementation
  - Overall documentation is prepared early in the SDLC
  - **Defect tracking software** or **bug tracking software**
    - Used to document and track program defects, code changes, and **patches**

Documentation (Cont. 1)

- **System Documentation**
  - Describes the system's functions and how they are implemented
- **Operations Documentation**
  - Contains the information needed for processing and distributing online and printed output
- **User Documentation**
  - Consists of instructions and information for users who will interact with the system

Documentation (Cont. 2)

- **Online Documentation**
  - Provides immediate help when users have questions or encounter problems

FIGURE 11-23 The Cisco Support Community invites users to contribute valuable experience and documentation using social media.
Source: Cisco Support Community

Documentation (Cont. 3)

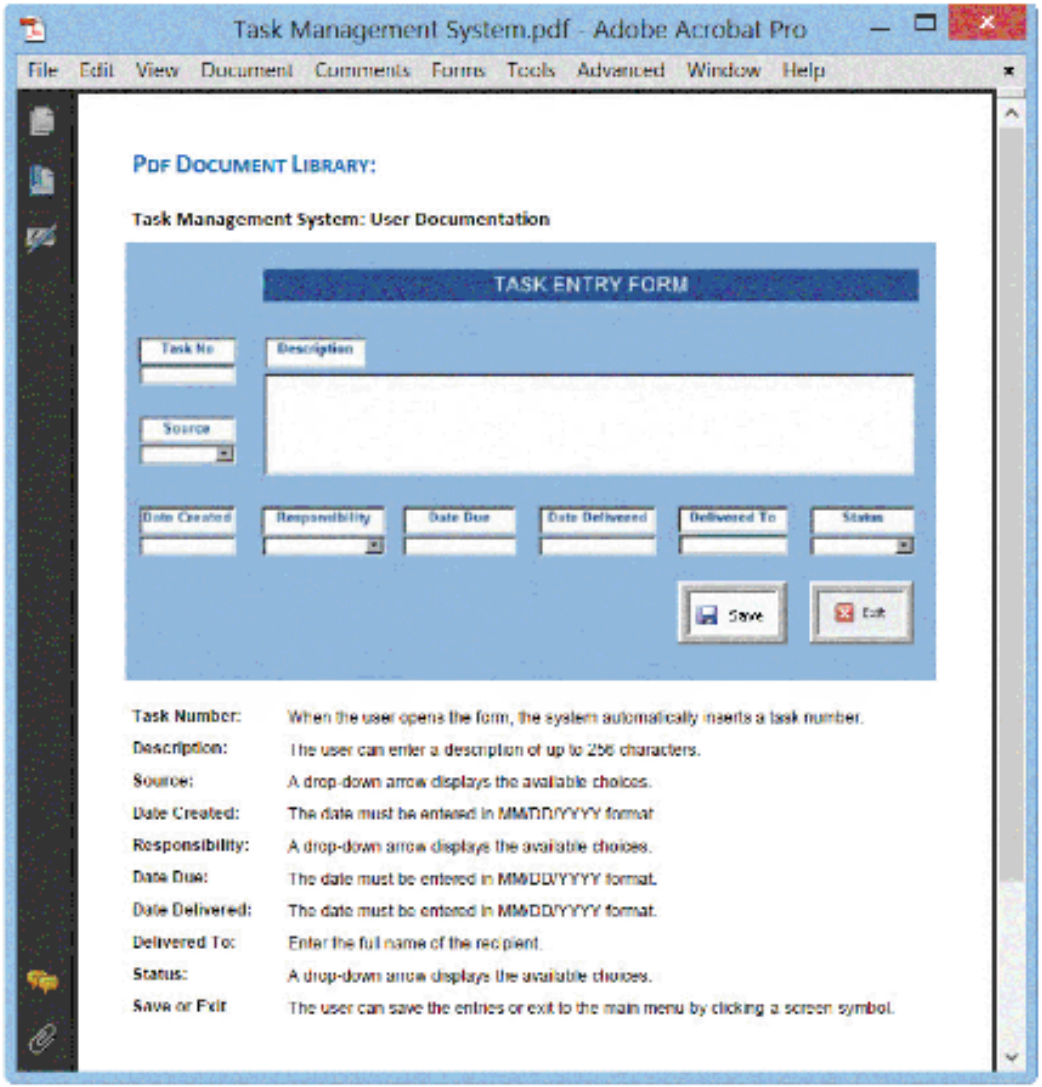


FIGURE 11-24 A sample page from a user manual. The instructions explain how to add a new task to the system.

System Installation and Evaluation

- Prepare a separate operational and test environment
- Provide training for users, managers, and IT staff
- Perform data conversion and system changeover
- Carry out a post-implementation evaluation of the system
- Present a final report to management

Operational and Test Environments

- **Operational** or **production environment**
  - Environment for the actual system operation
- **Test environment**
  - Environment that analysts and programmers use to develop and maintain programs
  - A separate test environment is necessary to maintain system security and integrity and protect the operational environment

Operational and Test Environments

(Cont.)

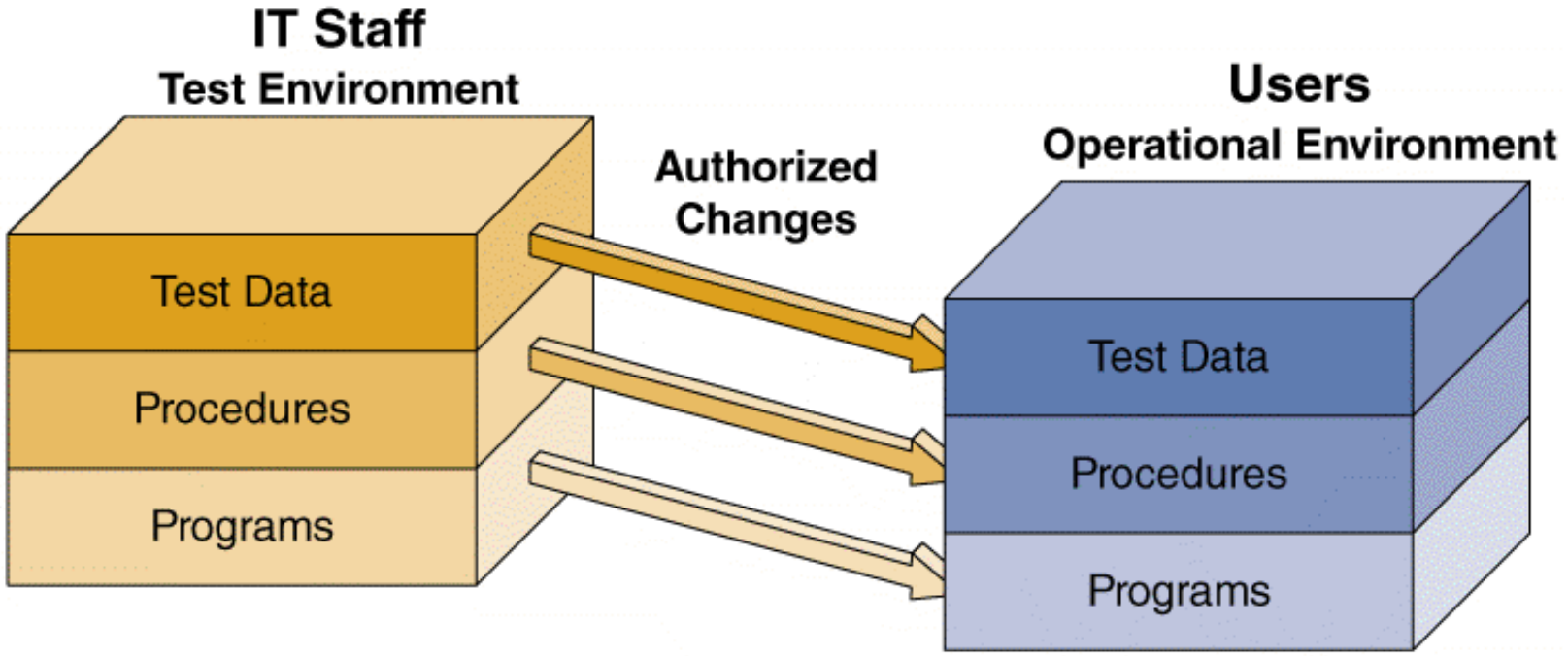


FIGURE 11-25 The test environment versus the operational environment. Notice that access to the test environment is limited to IT staff, while the operational environment is restricted to users.

Training

- **Training Plan**
  - Identify who should receive training and what training is required
    - Organization should be observed carefully to determine how the system will support business operations, and who will be involved or affected
  - Main groups for training
    - Users
    - Managers
    - IT staff

Training (Cont. 1)

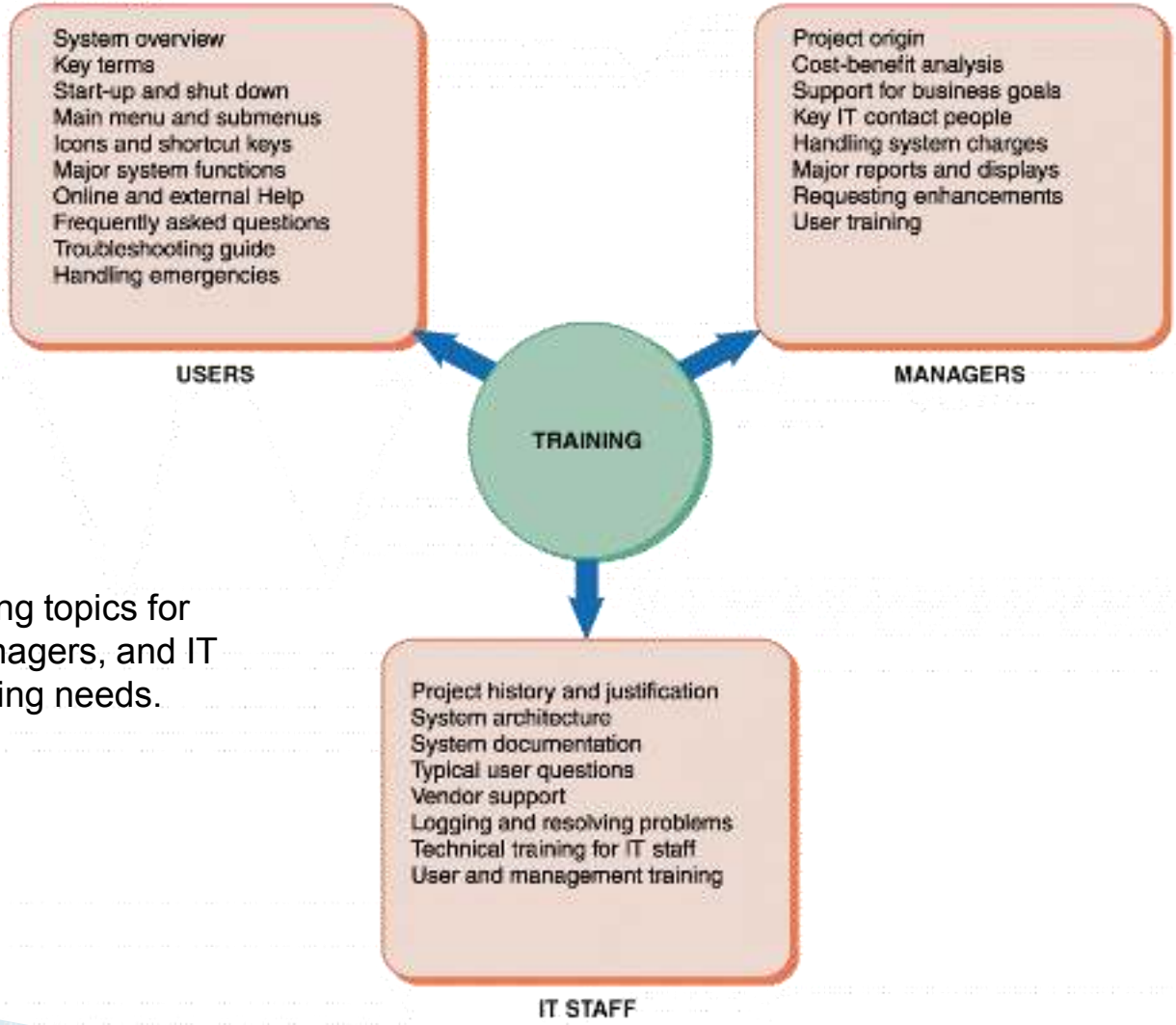


FIGURE 11-26 Examples of training topics for three different groups. Users, managers, and IT staff members have different training needs.

Training (Cont. 2)

- **Vendor Training**
  - Required if the system includes the purchase of hardware or software
    - Scope is limited to a standard version of the vendor's software or hardware
- **Webinars, Podcasts, and Tutorials**
  - Web-based training options
- **Outside Training**
  - Viable alternative if vendor training is not practical

Figure 11-27 Udemy is one of many sources of online training.
Source: Udemy.com

Training (Cont. 3)

- **Training Tips**
  - Train people in groups, with separate training programs for distinct groups
  - Select the most effective place to conduct the training
  - Provide for learning by hearing, seeing, and doing
  - Rely on previous trainees
- **Interactive Training**
  - Effective training is interactive, self-paced, and multimedia-based

Training (Cont. 4)

- Online training
  - Effective when it is more realistic
  - Sophisticated training systems offer interactive sessions
  - Training material should include a reference section that summarizes all options and commands, lists all possible error messages, and what actions the user should take when a problem arises
  - A full-scale test, or **simulation** must be conducted after the training is complete

Training (Cont. 5)

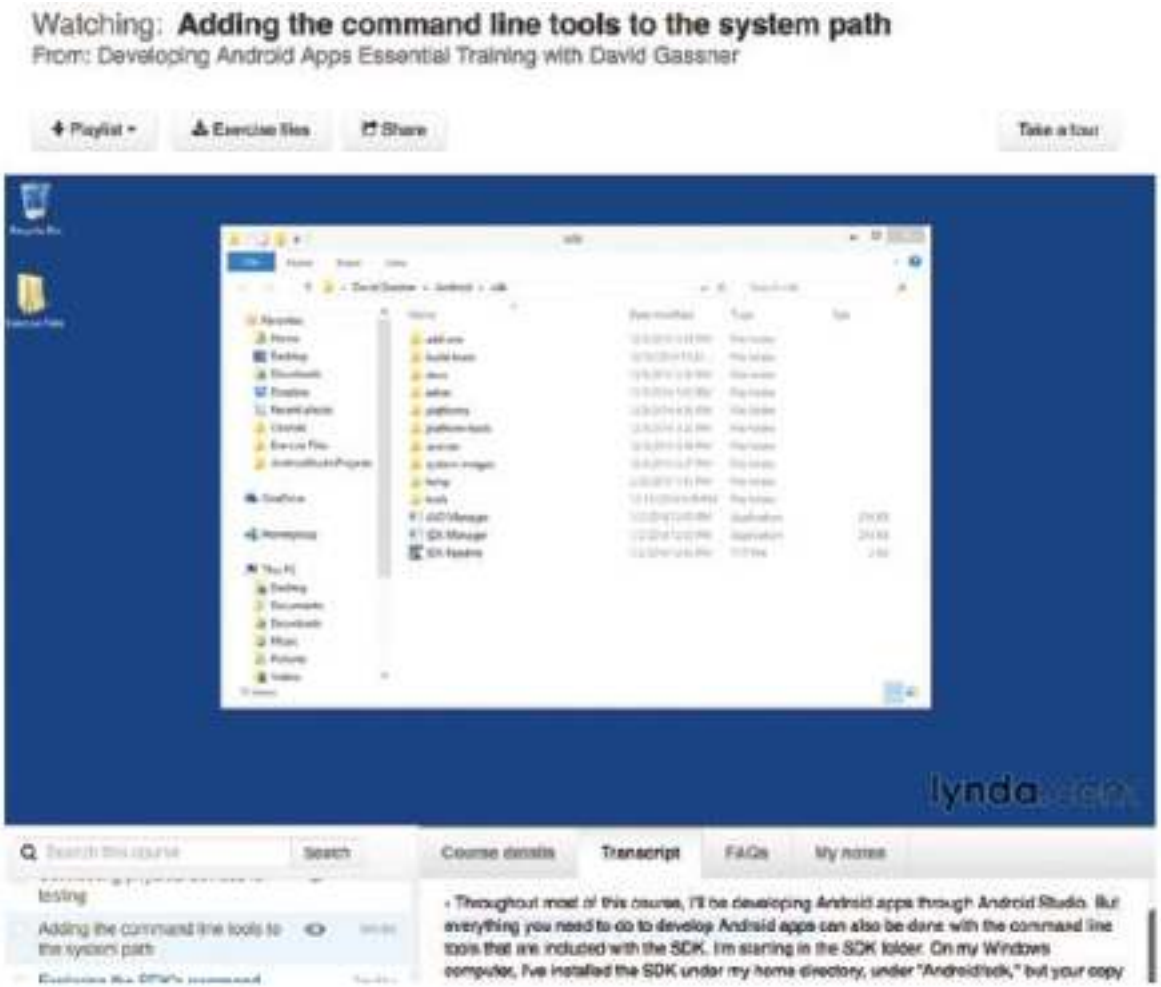


FIGURE 11-29 A sample lesson on Android app development in an online tutorial from lynda.com.
Source: lynda.com, Inc.

Data Conversion

- Process in which existing data is loaded on to a new system
- **Data Conversion Strategies**
  - Automate the data conversion process if possible
  - Old systems may be able to export data in ASCII or ODBC (Open Database Connectivity) formats
    - ODBC drivers are a form of middleware
  - If a standard format is not available, develop a program to extract the data and convert it to an acceptable format

Data Conversion (Cont.)

- **Data Conversion Security and Controls**
  - Maintain strict input controls during the conversion process
  - Ensure that all system control measures are in place and operational to protect data from unauthorized access and to help prevent erroneous input
  - Ensure that error-free data is fed into the new system

System Changeover

- Process of putting the new information system online and retiring the old system
- Can be rapid or slow, depending on the method
- Types
  - Direct cutover
  - Parallel operation
  - Pilot operation
  - Phased operation

System Changeover (Cont. 1)

- **Direct Cutover**
  - Enables changeover when the new system becomes operational
  - Least expensive changeover method
    - IT group has to operate and maintain only one system at a time

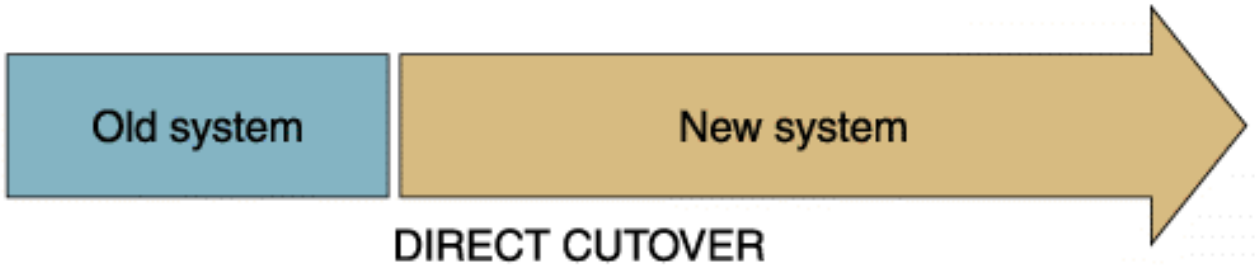


FIGURE 11-30 The four system changeover methods.

System Changeover (Cont. 2)

- **Parallel Operation**
  - Both the old and the new information systems operate fully for a specified period
  - Advantage – Lower level of risk
    - The old system can be used as a backup
  - Not practical if the old and new systems are incompatible

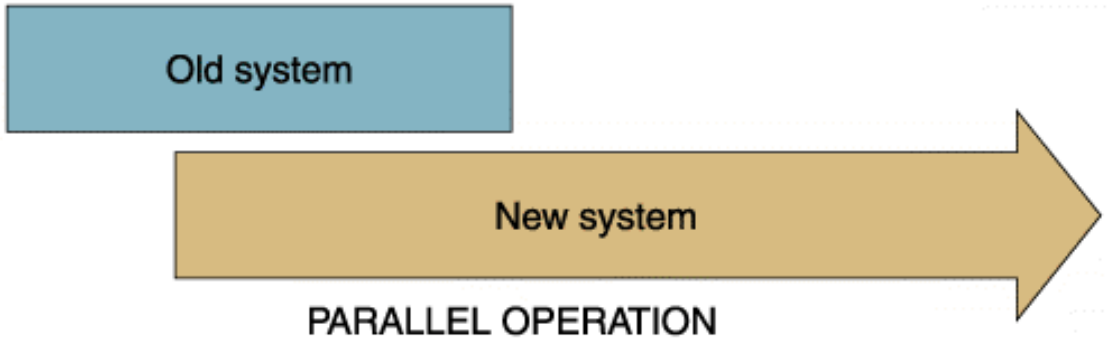


FIGURE 11-30 The four system changeover methods.

System Changeover (Cont. 3)

- **Pilot Operation**
  - Implements the complete new system at a selected location of the company
  - **Pilot site**: Group that uses the new system first
  - Reduces the risk of system failure
  - Less expensive than a parallel operation

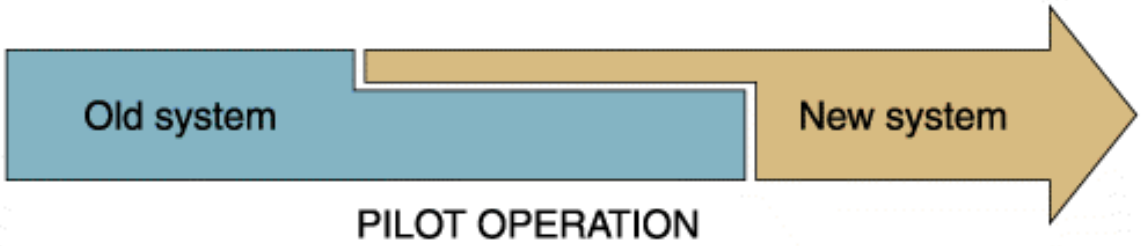


FIGURE 11-30 The four system changeover methods.

System Changeover (Cont. 4)

- **Phased Operation**
  - Implements the new system in stages, or modules
  - Combines direct cutover and parallel operation to reduce risks and costs
    - Only a part of the system is given to all users
  - Risk of errors or failures is limited to the implemented module only
  - Less expensive than full parallel operation

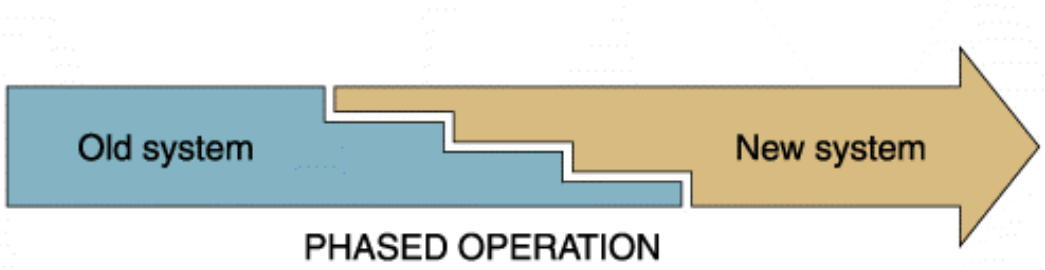


FIGURE 11-30 The four system changeover methods.

System Changeover (Cont. 5)

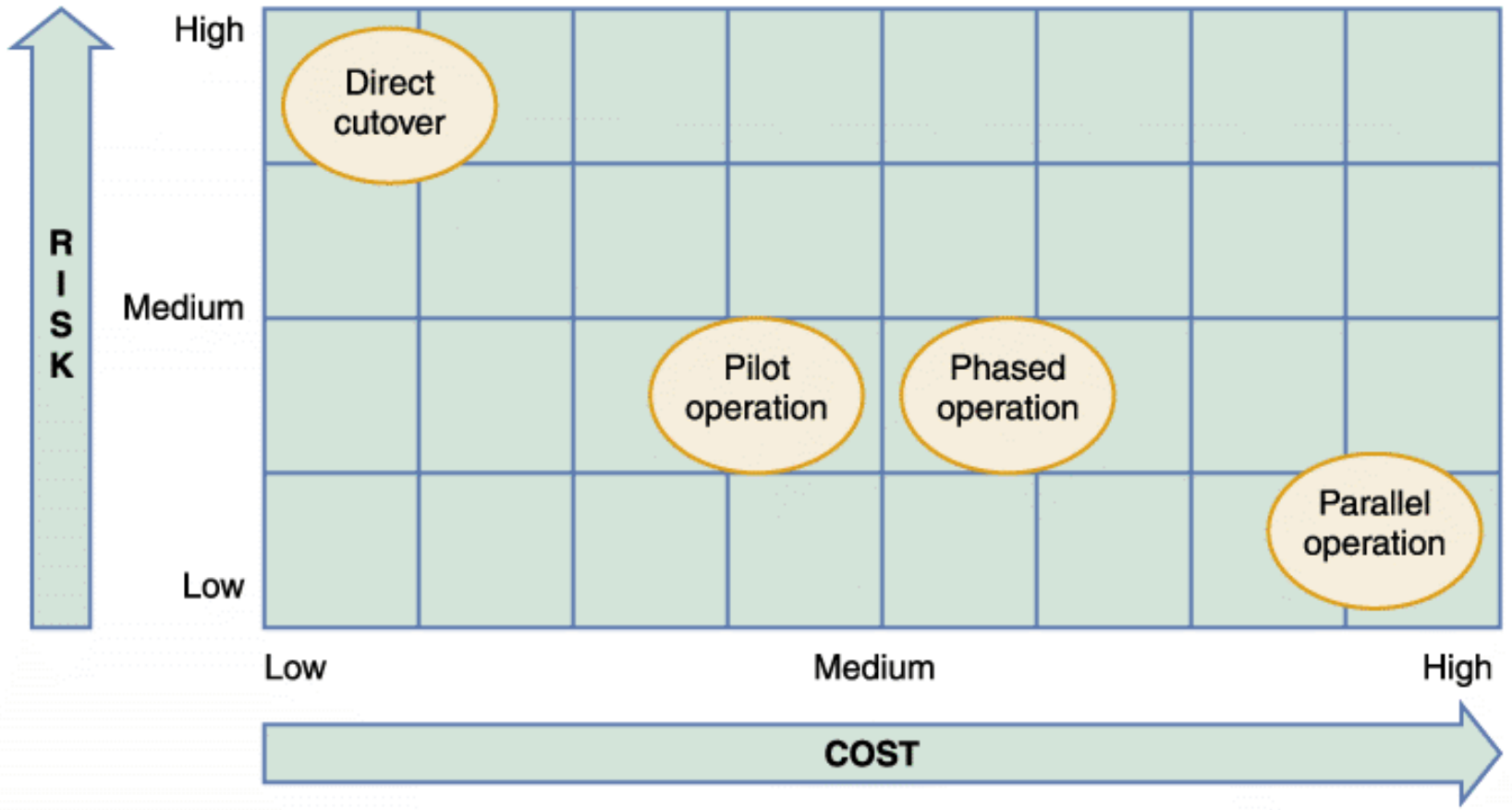


FIGURE 11-31 Relative risk and cost characteristics of the four changeover methods.

Chapter Summary

- Systems implementation phase consists of application development, testing, installation, and evaluation of the new system
- Each systems development approach has its own set of tools
- System developers can use more generic tools to help them translate the system logic into properly functioning program modules

Chapter Summary (Cont. 1)

- In agile development, the customer creates user stories describing required features and priority levels
- Cohesion measures a module's scope and processing characteristics
- Coupling measures relationships and interdependence among modules
- There are four steps used to create a structure chart

Chapter Summary (Cont. 2)

- Programmers perform desk checking, code review, and unit testing tasks during application development
- In addition to system documentation, analysts and technical writers also prepare operations documentation and user documentation
- During the installation process, an operational environment for the new information system is established separate from the test environment

Chapter Summary (Cont. 3)

- Everyone who interacts with the new information system should receive training appropriate to their skills
- Data conversion often is necessary when installing a new information system
- System changeover is the process of putting the new system into operation

Chapter Summary (Cont. 4)

- A post-implementation evaluation assesses and reports on the quality of the new system and the work done by the project team
- The final report to management includes the final system documentation, describes any future system enhancements that already have been identified, and details the project costs