

# *Systems Analysis and Design in a Changing World, Fourth Edition*

## CHAPTER

# 2

## APPROACHES TO SYSTEM DEVELOPMENT



# Learning Objectives

- ◆ Explain the purpose and various phases of the systems development life cycle (SDLC)
- ◆ Explain when to use an adaptive approach to the SDLC in place of a more predictive traditional SDLC
- ◆ Explain the differences between a model, a tool, a technique, and a methodology
- ◆ Describe the two overall approaches used to develop information systems: the traditional method and the object-oriented method

## Learning Objectives (continued)

- ◆ Describe some of the variations of the systems development life cycle (SDLC)
- ◆ Describe the key features of current trends in systems development: the Unified Process (UP), Extreme Programming (XP), Agile Modeling, and Scrum
- ◆ Explain how automated tools are used in system development

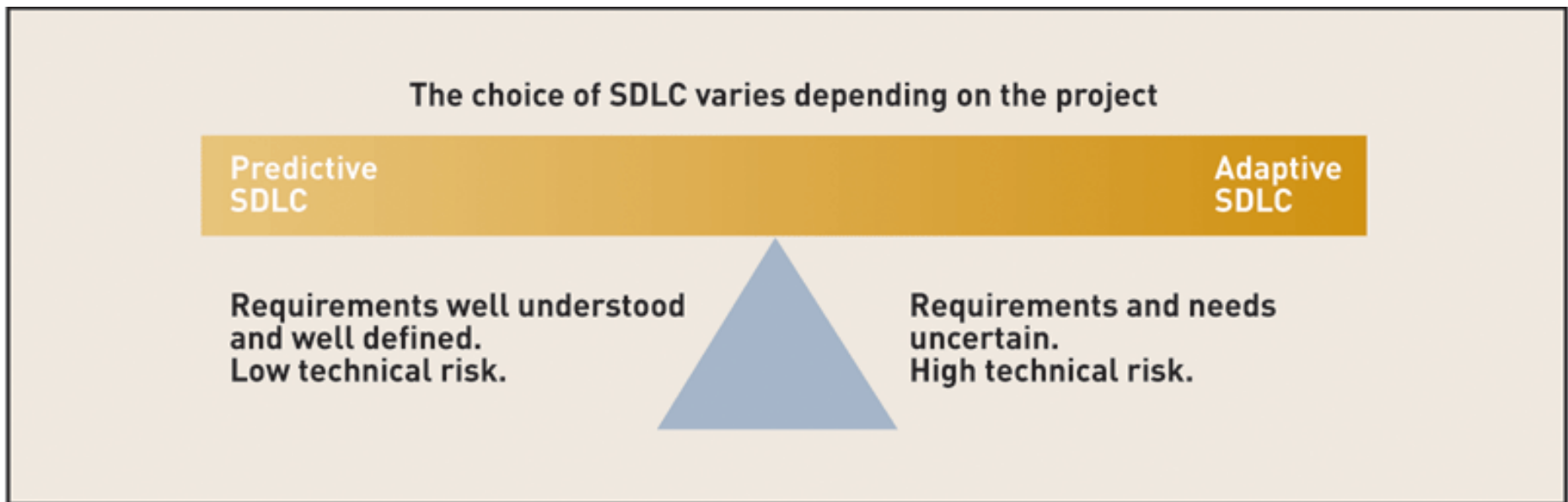
# Overview

- ◆ System development project
  - Planned undertaking with fixed beginning and end
  - Produces desired result or product
  - Can be a large job with thousands of hours of effort or a small one-month project
- ◆ Successful development project
  - Provides a detailed plan to follow
  - Organized, methodical sequence of tasks and activities
  - Produces reliable, robust, and efficient system

# The Systems Development Lifecycle (SDLC)

- ◆ Systems development life cycle (SDLC)
  - Provides overall framework for managing systems development process
- ◆ Two main approaches to SDLC
  - Predictive approach – assumes project can be planned out in advance
  - Adaptive approach – more flexible, assumes project cannot be planned out in advance
- ◆ All projects use some variation of SDLC

# Choosing the Predictive vs. Adaptive Approach to the SDLC (Figure 2-1)



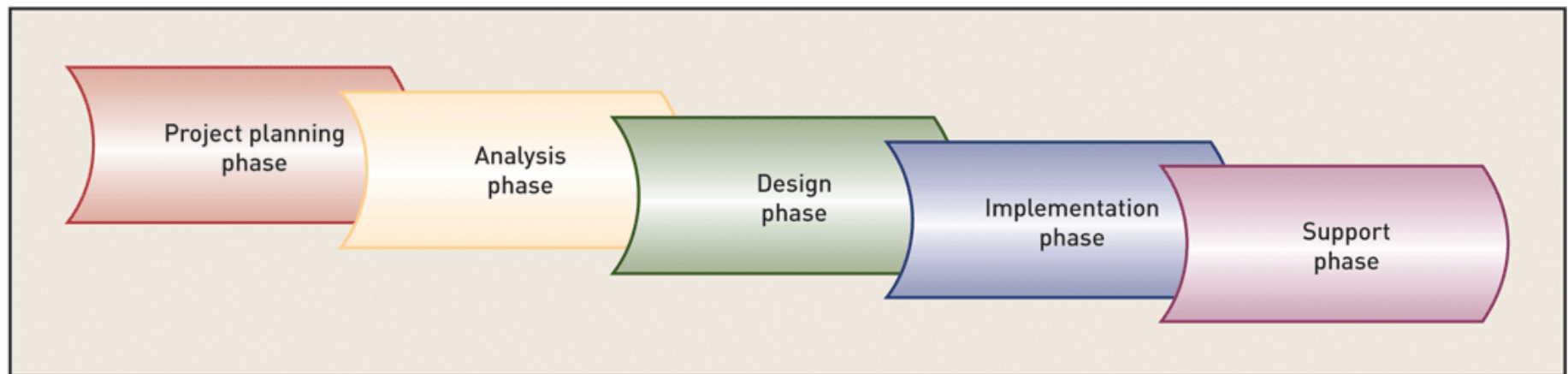
# Traditional Predictive Approach to the SDLC

- ◆ **Project planning** – initiate, ensure feasibility, plan schedule, obtain approval for project
- ◆ **Analysis** – understand business needs and processing requirements
- ◆ **Design** – define solution system based on requirements and analysis decisions
- ◆ **Implementation** – construct, test, train users, and install new system
- ◆ **Support** – keep system running and improve

# Information System Development Phases

**Figure 2-2**

Information system  
development phases





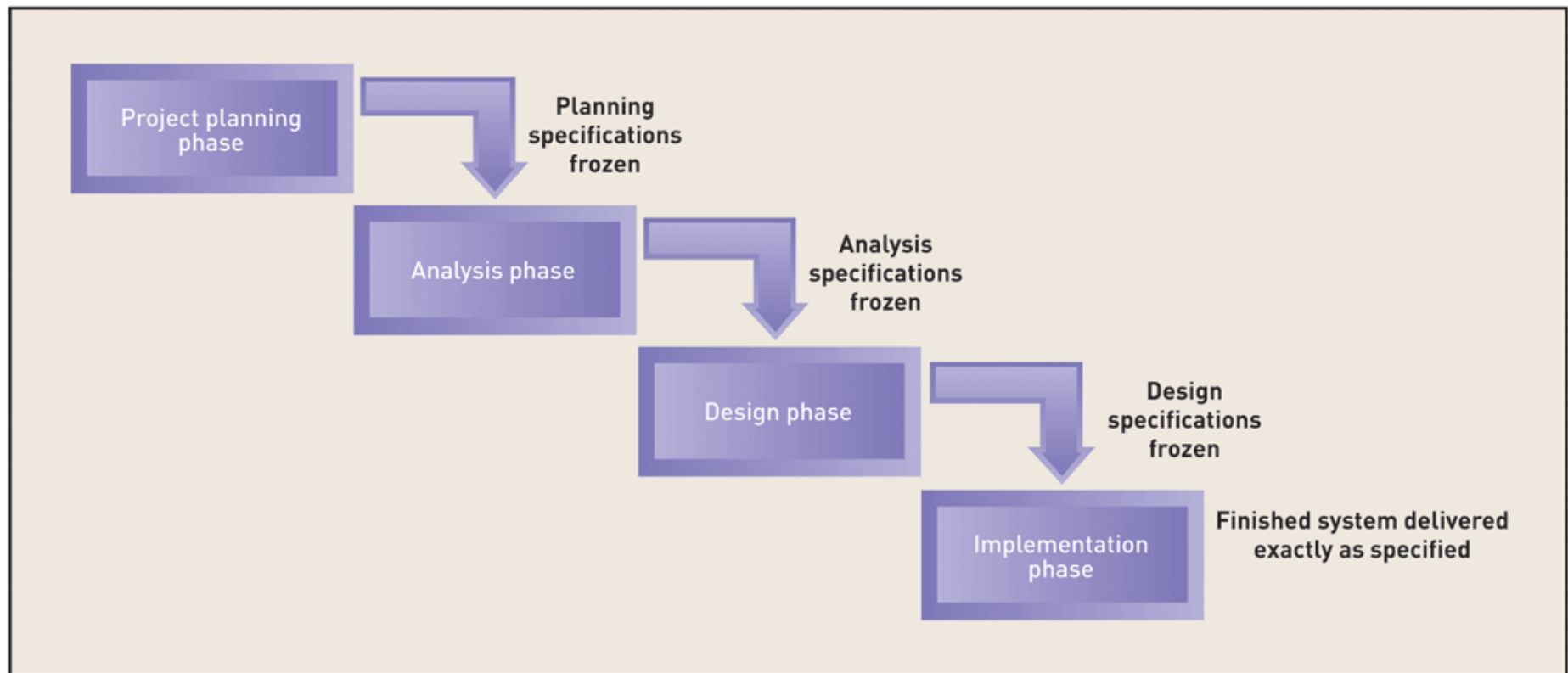
# SDLC and Problem Solving

- ◆ Similar to problem-solving approach in Chapter 1
  - Organization recognizes problem (project planning)
  - Project team investigates, understands problem and solution requirements (analysis)
  - Solution is specified in detail (design)
  - System that solves problem is built and installed (implementation)
  - System used, maintained, and enhanced to continue to provide intended benefits (support)

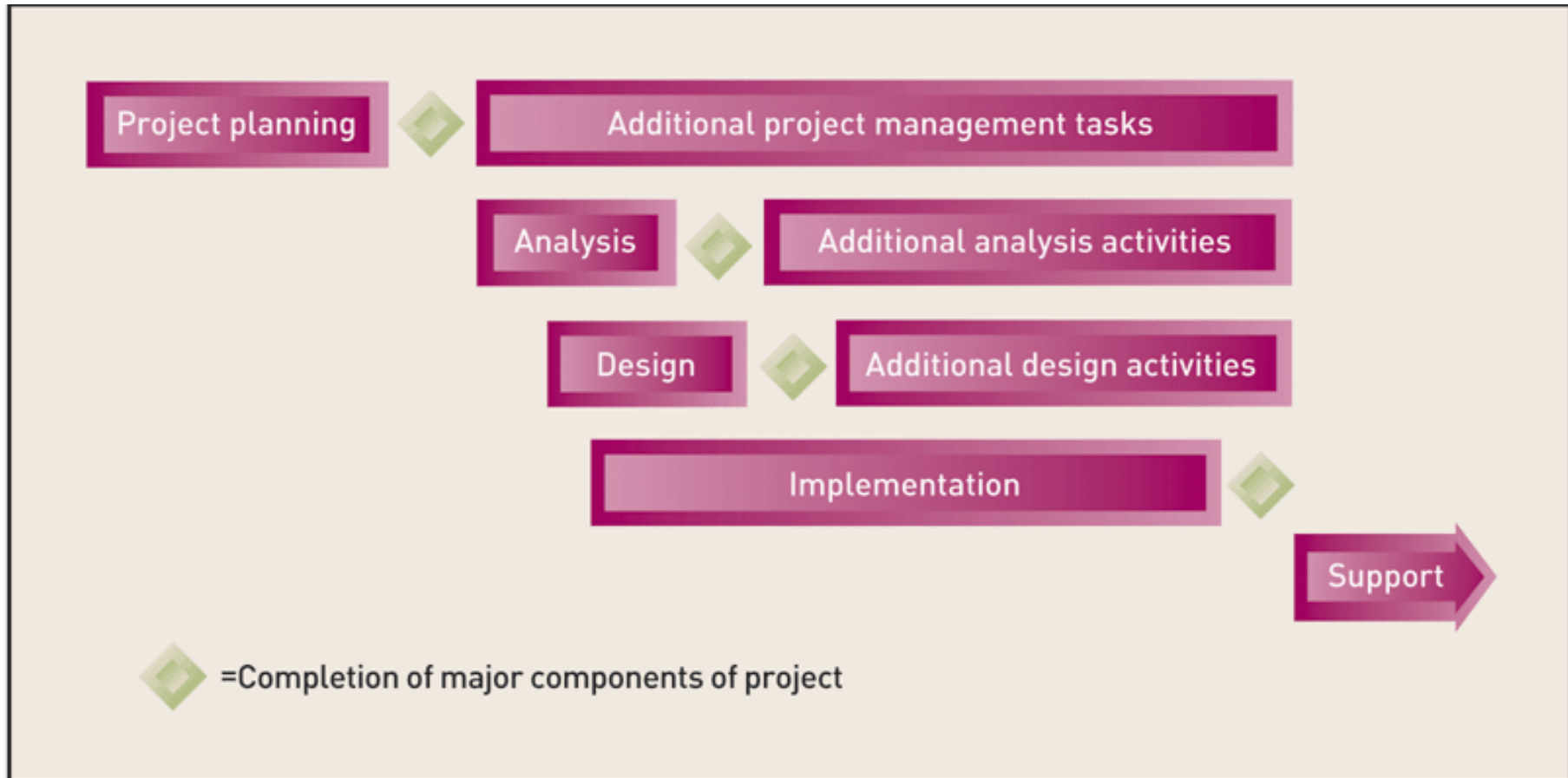
# “Waterfall” Approach to the SDLC

**Figure 2-4**

The waterfall approach to the SDLC



# Modified Waterfall Approach with Overlapping Phases (Figure 2-5)



# Newer Adaptive Approaches to the SDLC

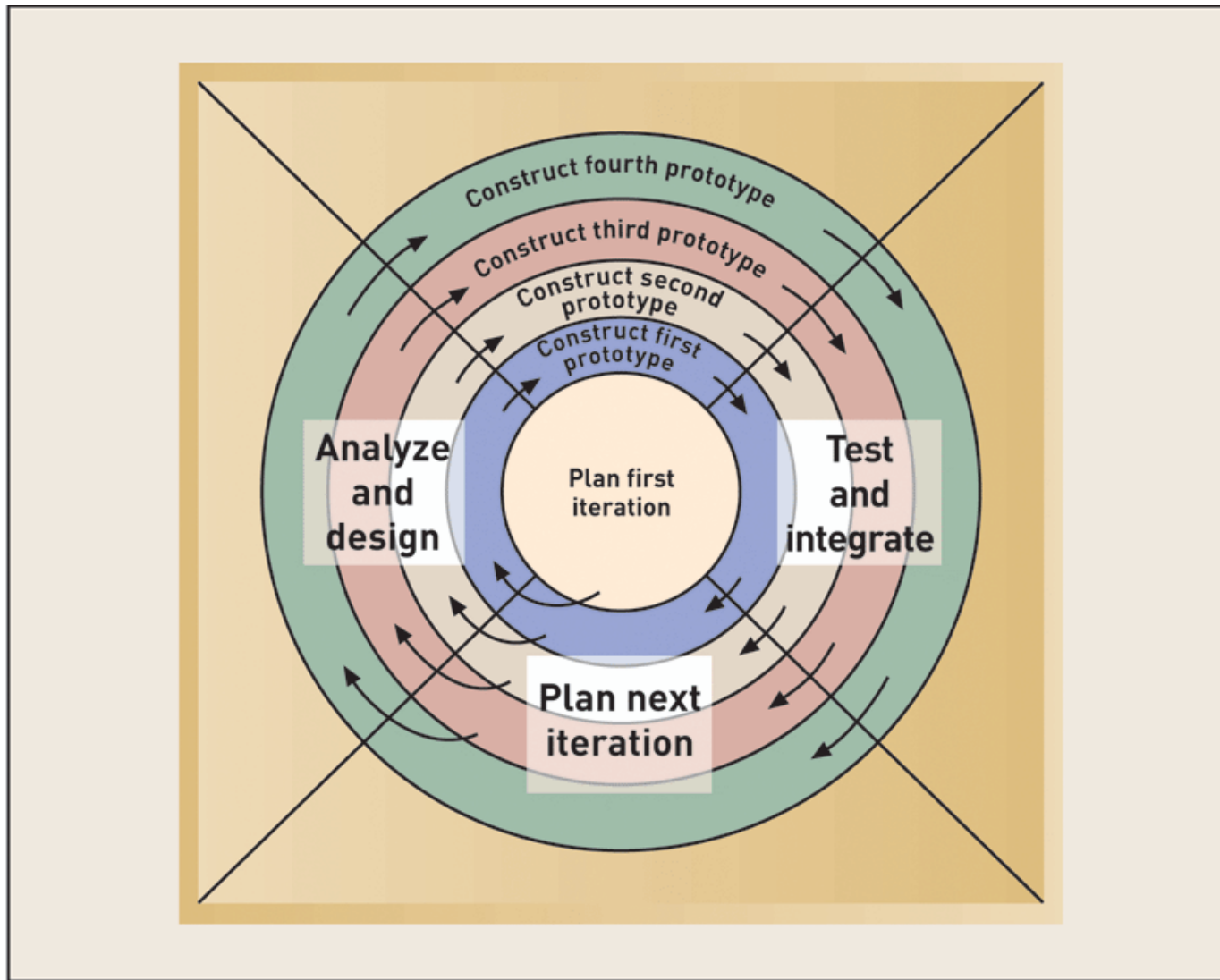
## ◆ Based on spiral model

- Project cycles through development activities over and over until project is complete
- Prototype created by end of each cycle
- Focuses on mitigating risk

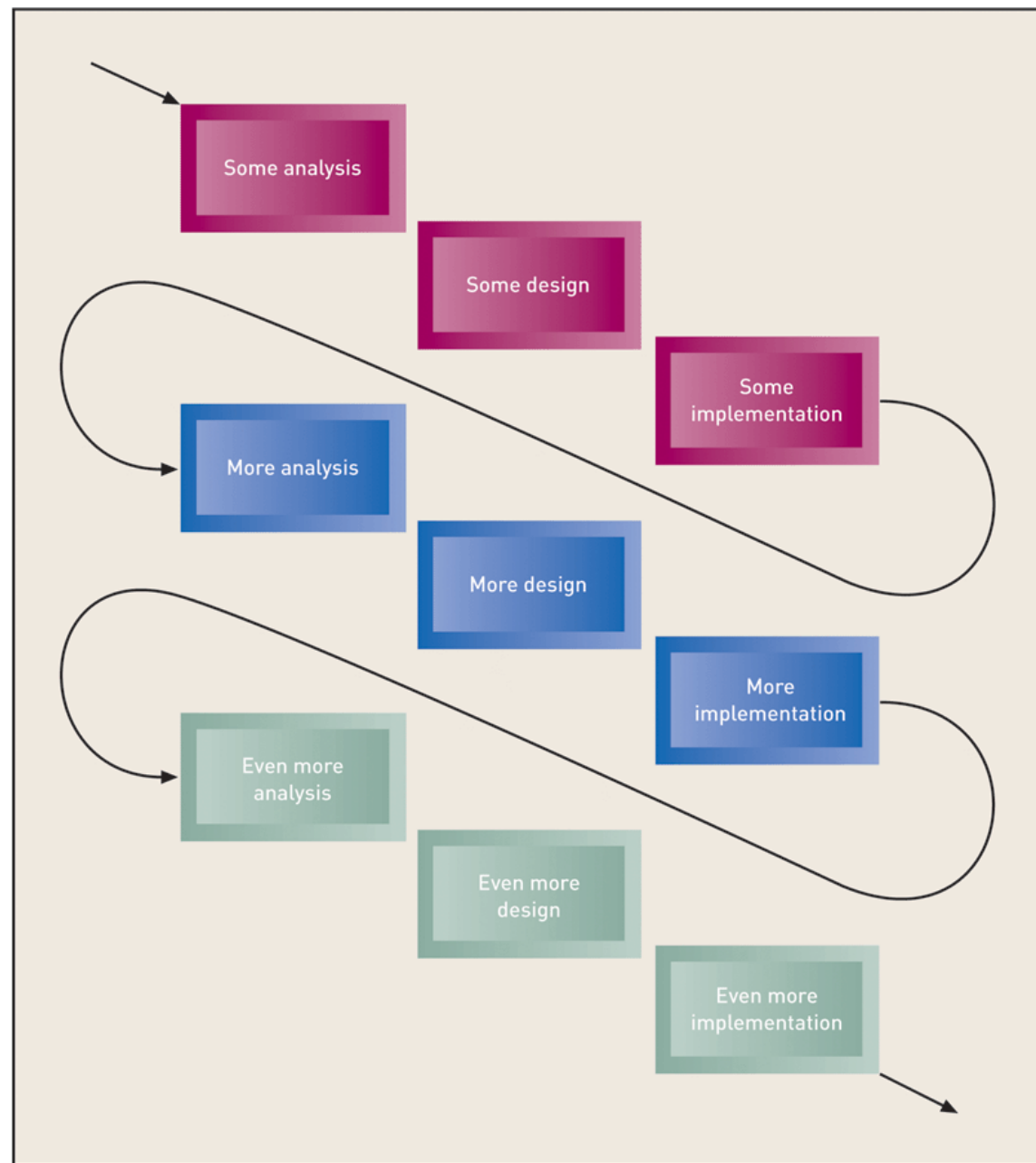
## ◆ Iteration – Work activities are repeated

- Each iteration refines previous result
- Approach assumes no one gets it right the first time
- There are a series of mini projects for each iteration

# The Spiral Life Cycle Model (Figure 2-6)



# Iteration of System Development Activities (Figure 2-7)



## Activities of Each SDLC Phase

- ◆ Predictive or adaptive approach use SDLC
- ◆ Activities of each “phase” are similar
- ◆ Phases are not always sequential
- ◆ Phases can overlap
- ◆ Activities across phases can be done within an iteration

## Activities of Planning Phase of SDLC

- ◆ Define business problem and scope
- ◆ Produce detailed project schedule
- ◆ Confirm project feasibility
  - Economic, organizational, technical, resource, and schedule
- ◆ Staff the project (resource management)
- ◆ Launch project → official announcement



## Activities of Analysis Phase of SDLC

- ◆ Gather information to learn problem domain
- ◆ Define system requirements
- ◆ Build prototypes for discovery of requirements
- ◆ Prioritize requirements
- ◆ Generate and evaluate alternatives
- ◆ Review recommendations with management

## Activities of Design Phase of SDLC

- ◆ Design and integrate the network
- ◆ Design the application architecture
- ◆ Design the user interfaces
- ◆ Design the system interfaces
- ◆ Design and integrate the database
- ◆ Prototype for design details
- ◆ Design and integrate system controls

# Activities of Implementation Phase of SDLC

- ◆ Construct software components
- ◆ Verify and test
- ◆ Convert data
- ◆ Train users and document the system
- ◆ Install the system

# Activities of Support Phase of SDLC

- ◆ Maintain system
  - Small patches, repairs, and updates
- ◆ Enhance system
  - Small upgrades or enhancements to expand system capabilities
  - Larger enhancements may require separate development project
- ◆ Support users
  - Help desk and/or support team

# Methodologies and Models

## ◆ Methodologies

- Comprehensive guidelines to follow for completing every SDLC activity
- Collection of models, tools, and techniques

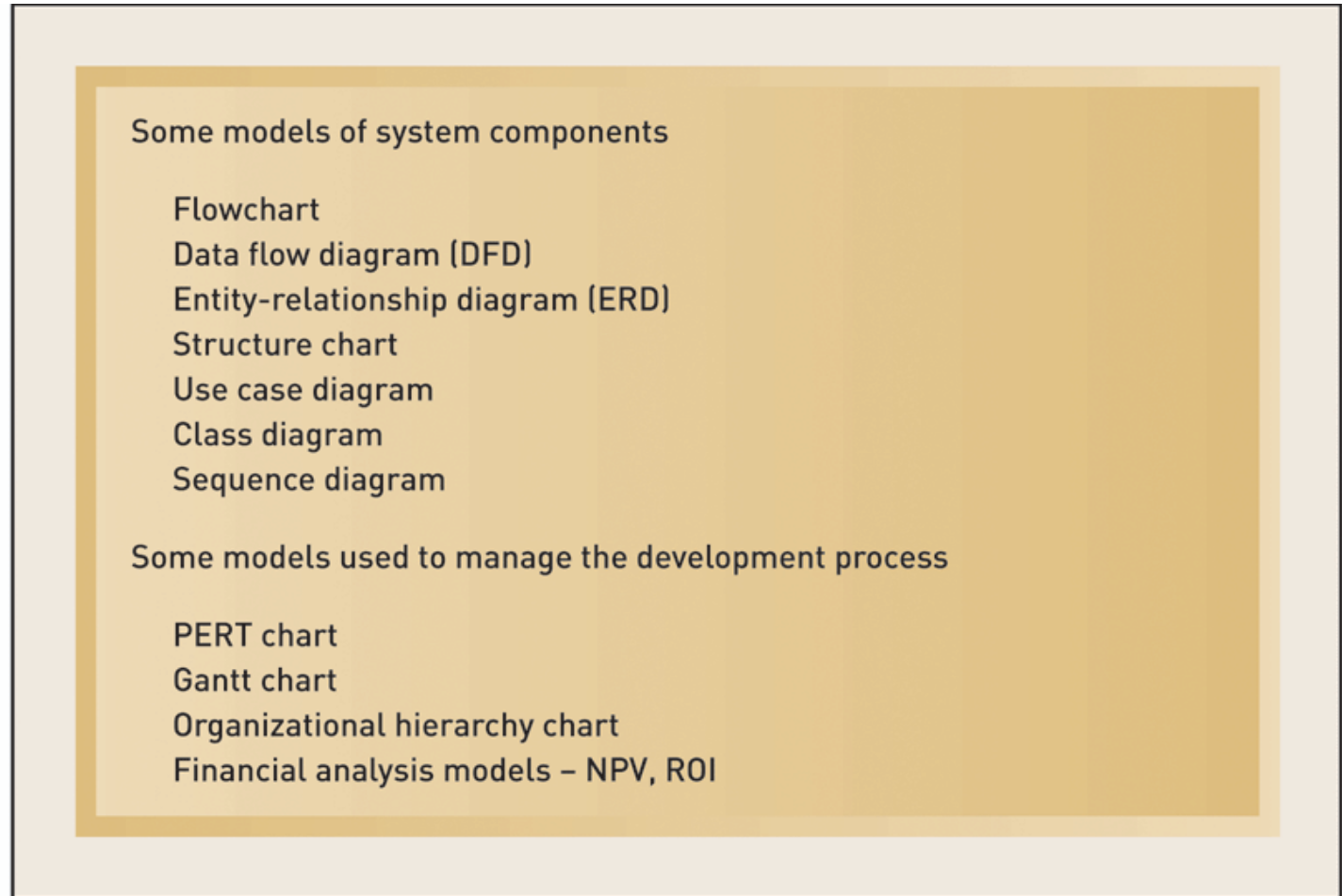
## ◆ Models

- Representation of an important aspect of real world, but not same as real thing
- Abstraction used to separate out aspect
- Diagrams and charts
- Project planning and budgeting aids

# Some Models Used in System Development

**Figure 2-8**

Some models used in system development



# Tools and Techniques

## ◆ Tools

- Software support that helps create models or other required project components
- Range from simple drawing programs to complex CASE tools to project management software

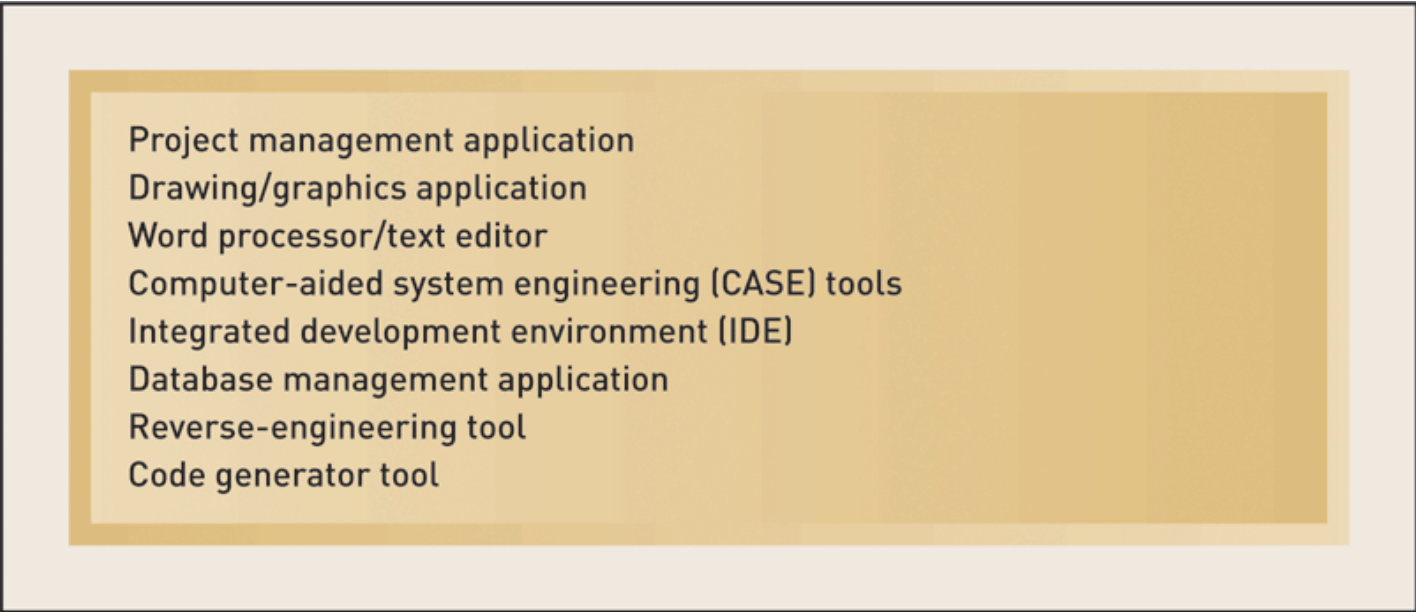
## ◆ Techniques

- Collection of guidelines that help analysts complete a system development activity or task
- Can be step-by-step instructions or just general advice

# Some Tools Used in System Development

**Figure 2-9**

Some tools used in  
system development




- Project management application
- Drawing/graphics application
- Word processor/text editor
- Computer-aided system engineering (CASE) tools
- Integrated development environment (IDE)
- Database management application
- Reverse-engineering tool
- Code generator tool



# Some Techniques Used in System Development

**Figure 2-10**

Some techniques used in system development

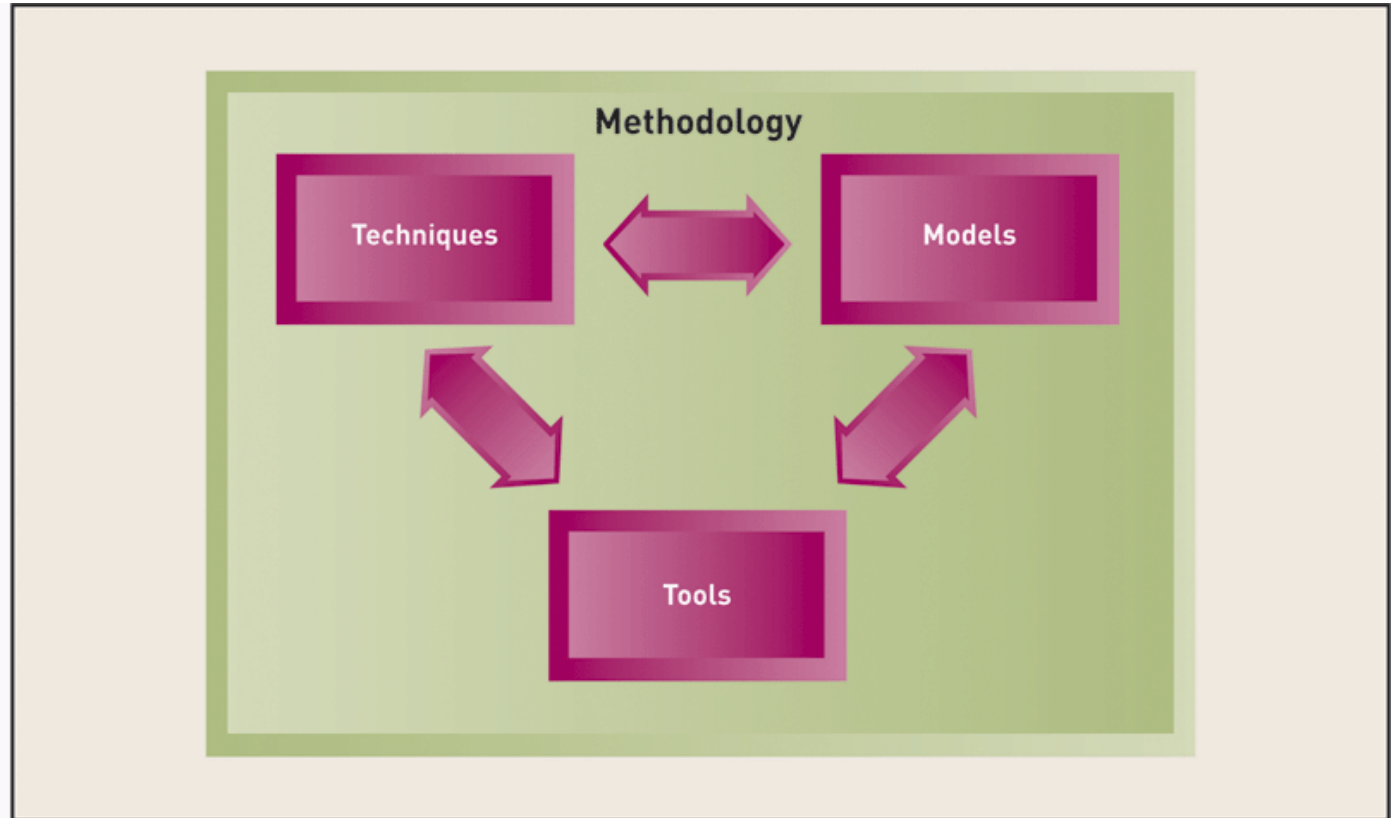


- Strategic planning techniques
- Project management techniques
- User interviewing techniques
- Data-modeling techniques
- Relational database design techniques
- Structured analysis technique
- Structured design technique
- Structured programming technique
- Software-testing techniques
- Object-oriented analysis and design techniques

# Relationships Among Components of a Methodology

**Figure 2-11**

Relationships among components of a methodology



# Two Approaches to System Development

## ◆ Traditional approach

- Also called structured system development
- Structured analysis and design technique (SADT)
- Includes information engineering (IE)

## ◆ Object-oriented approach

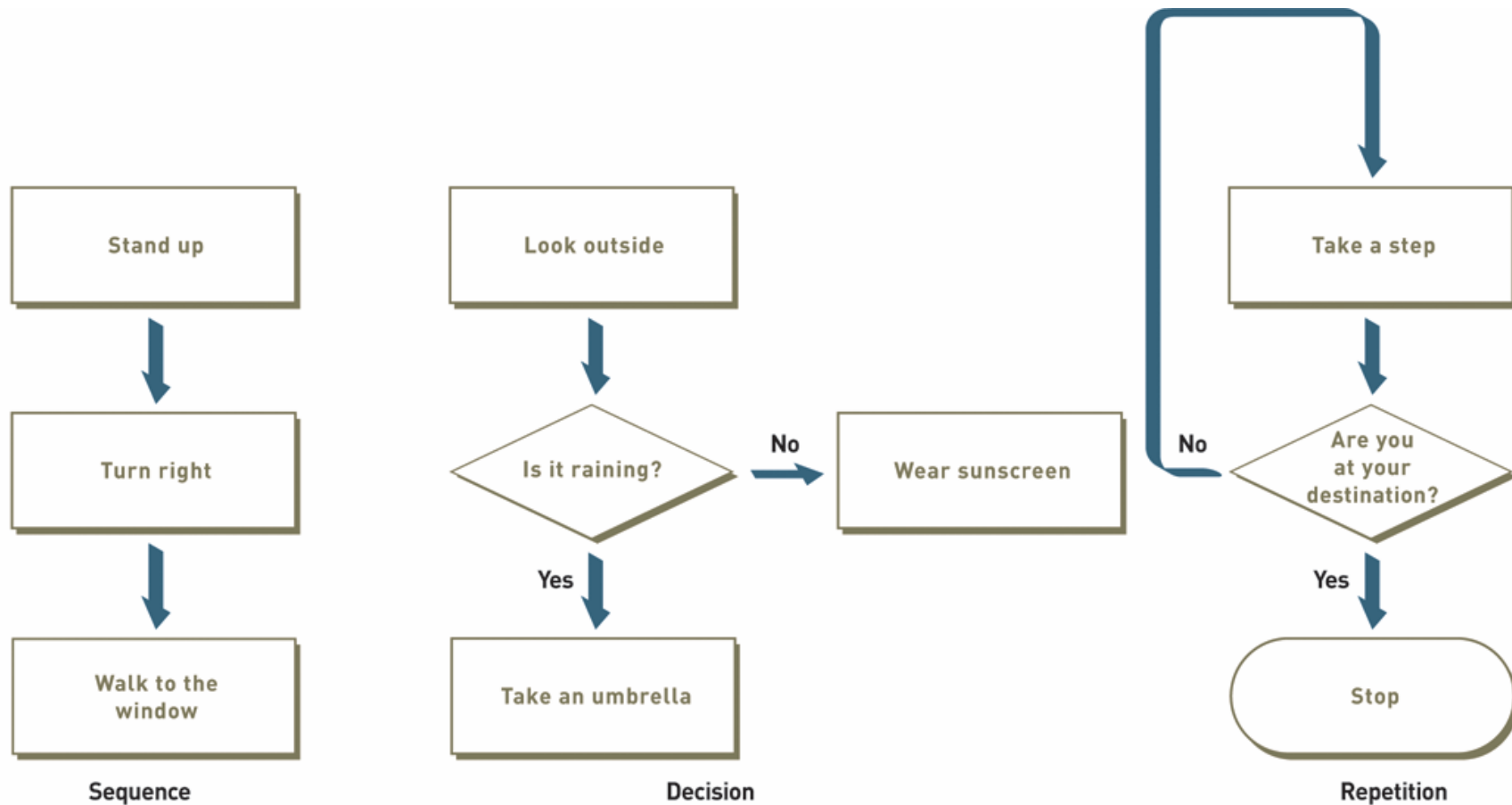
- Also called OOA, OOD, and OOP
- Views information system as collection of interacting objects that work together to accomplish tasks

# Traditional Approach

## ◆ Structured programming

- Improves computer program quality
- Allows other programmers to easily read and modify code
- Each program module has one beginning and one ending
- Three programming constructs (sequence, decision, repetition)

# Three Structured Programming Constructs



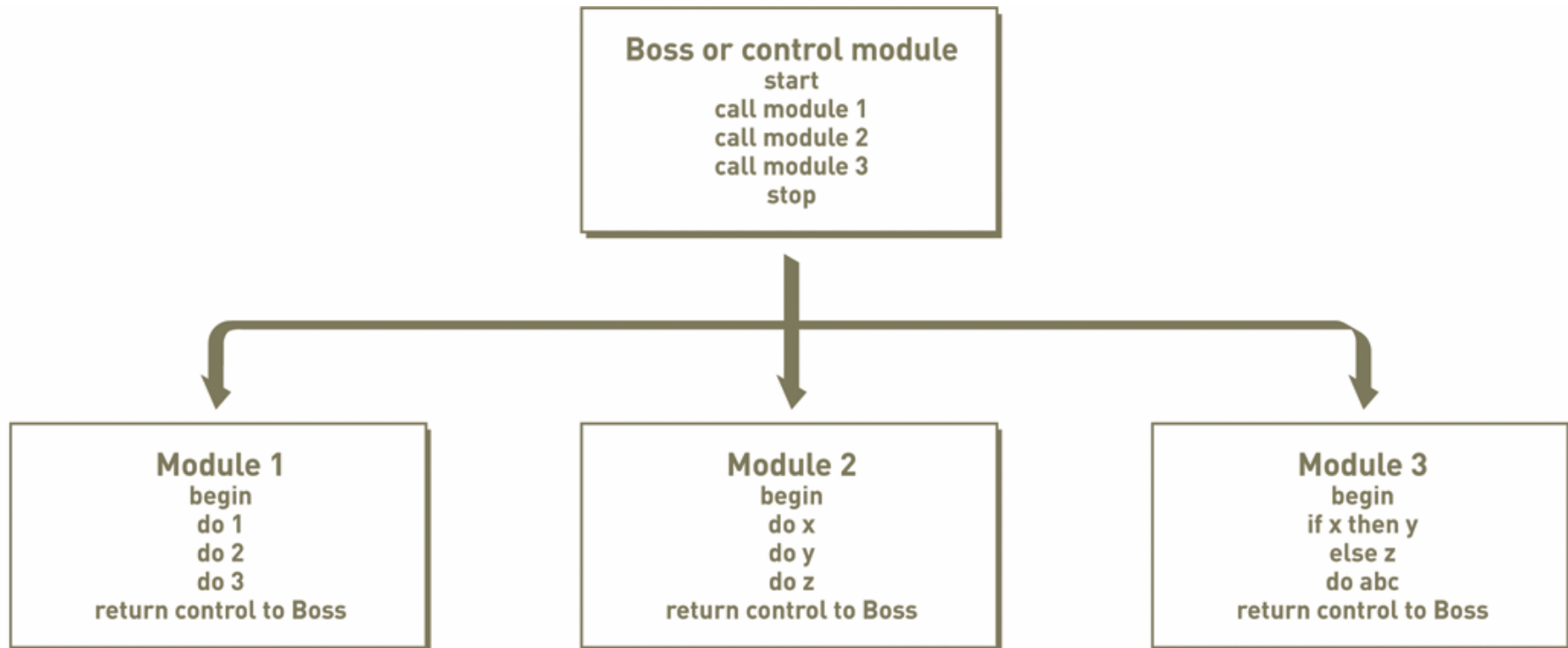
**Figure 2-12**

Three structured  
programming constructs

# Top-Down Programming

- ◆ Divides complex programs into hierarchy of modules
- ◆ The module at top controls execution by “calling” lower level modules
- ◆ Modular programming
  - Similar to top-down programming
- ◆ One program calls other programs to work together as single system

# Top-Down or Modular Programming



**Figure 2-13**

Top-down, or modular,  
programming

# Structured Design

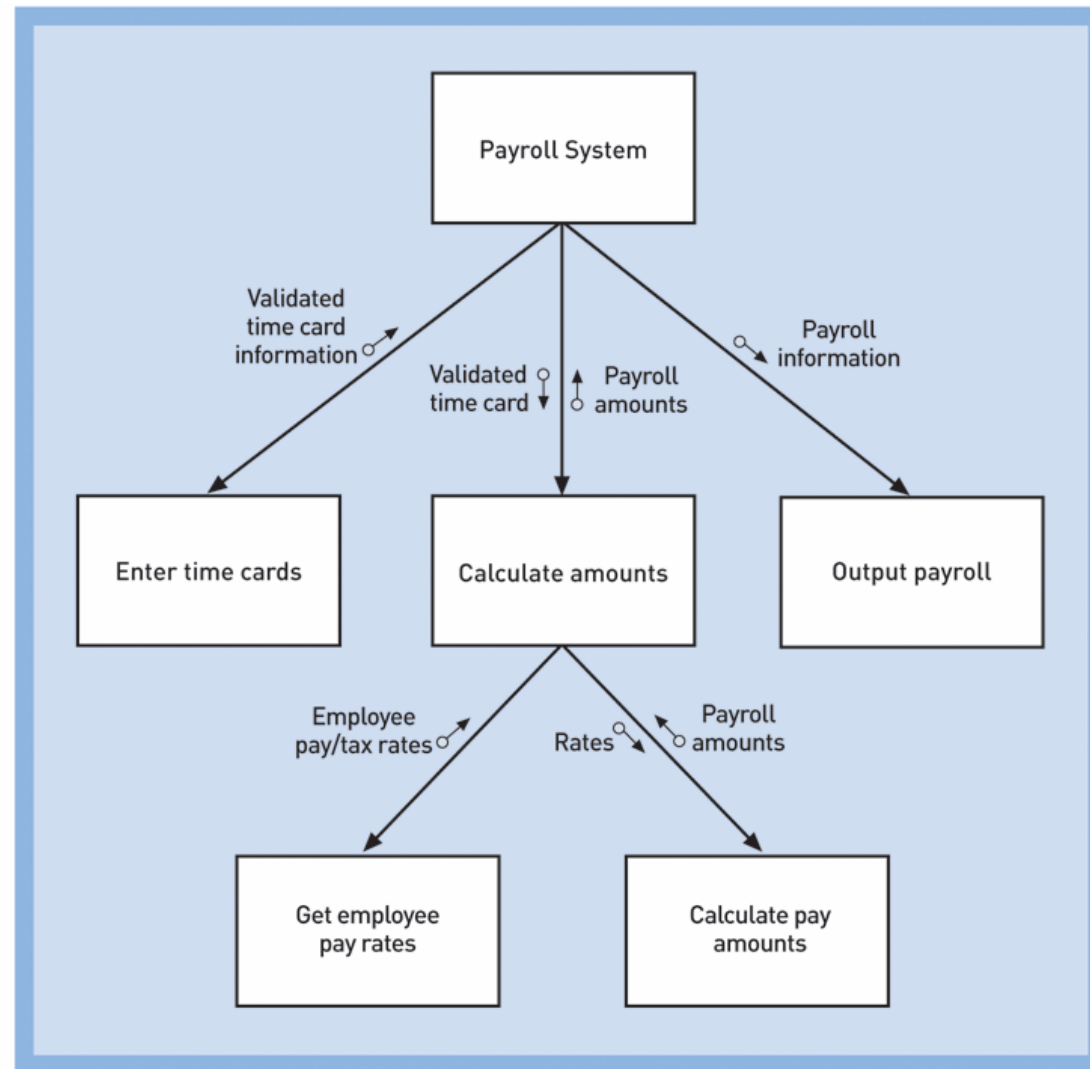
- ◆ Technique developed to provide design guidelines
  - What set of programs should be
  - What program should accomplish
  - How programs should be organized into a hierarchy
- ◆ Modules are shown with structure chart
- ◆ Main principle of program modules
  - Loosely coupled – module is independent of other modules
  - Highly cohesive – module has one clear task



# Structure Chart Created Using Structured Design Technique

**Figure 2-14**

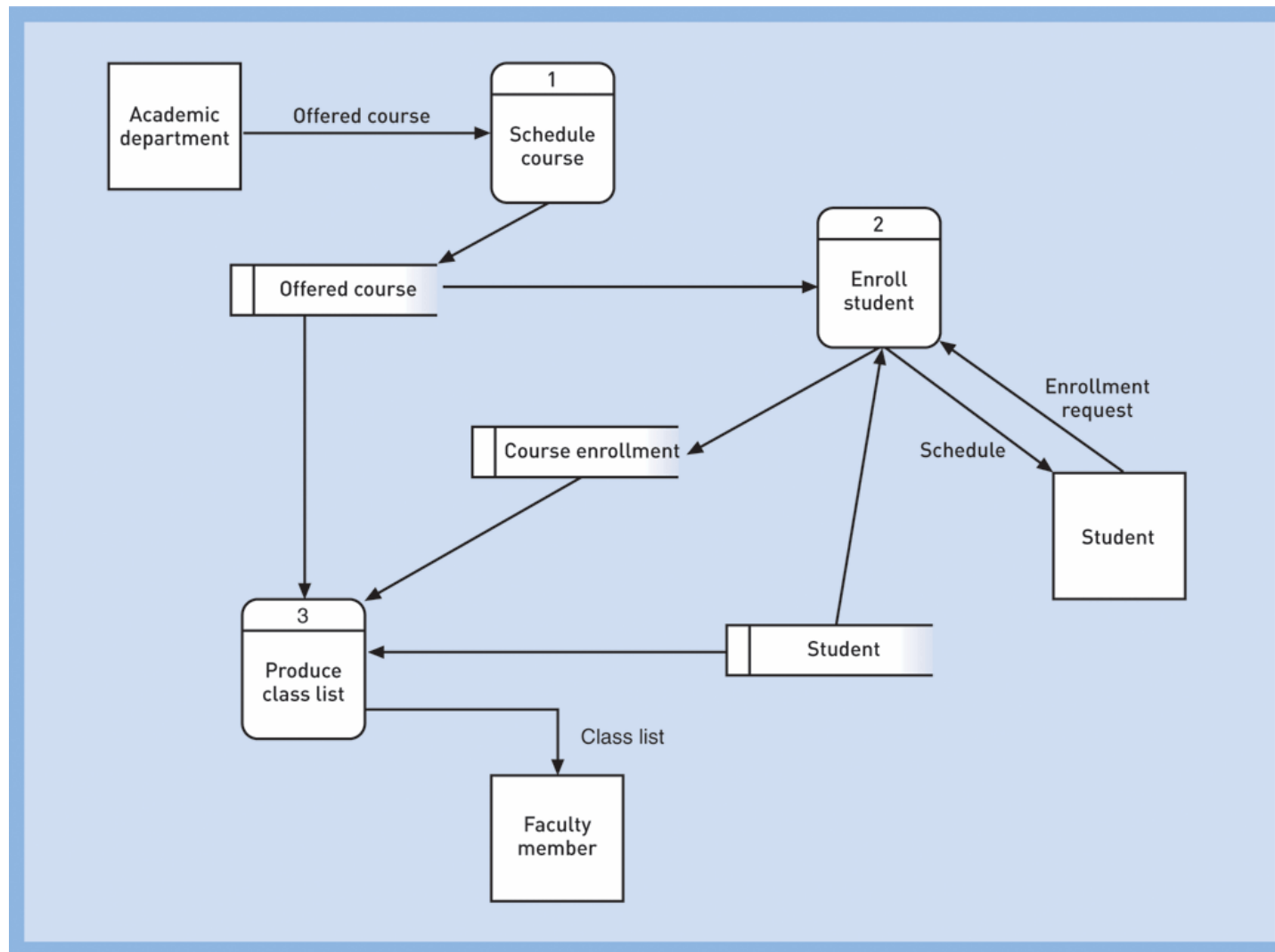
A structure chart created using the structured design technique



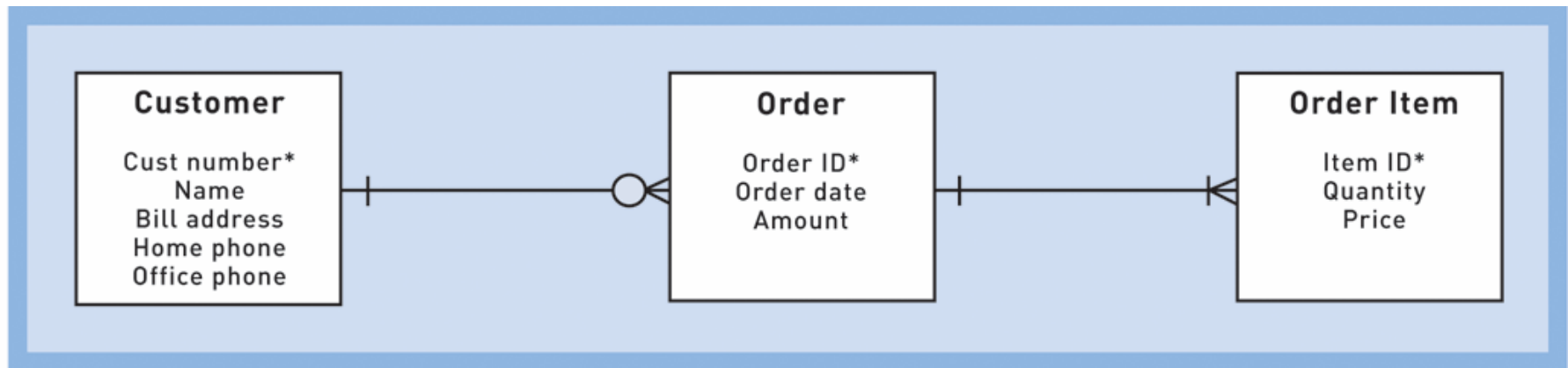
# Structured Analysis

- ◆ Define what system needs to do (processing requirements)
- ◆ Define data system needs to store and use (data requirements)
- ◆ Define inputs and outputs
- ◆ Define how functions work together to accomplish tasks
- ◆ Data flow diagrams (DFD) and entity relationship diagrams (ERD) show results of structured analysis

# Data Flow Diagram (DFD) Created Using Structured Analysis Technique (Figure 2-15)



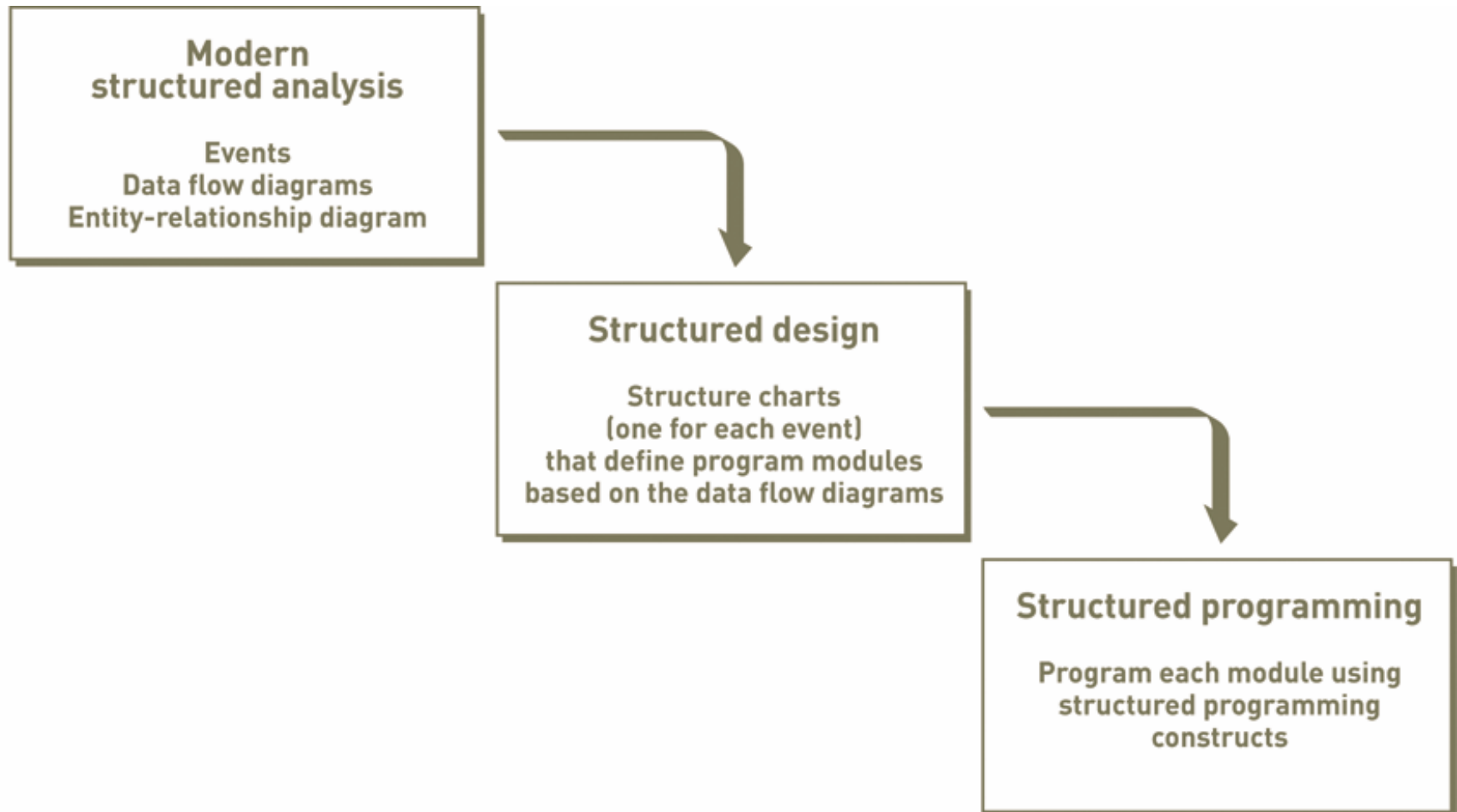
# Entity-Relationship Diagram (ERD) Created Using Structured Analysis Technique



**Figure 2-16**

An entity-relationship diagram (ERD) created using the structured analysis technique

# Structured Analysis Leads to Structured Design and Structured Programming (Figure 2-17)



# Information Engineering (IE)

- ◆ Refinement to structured development
- ◆ Methodology with strategic planning, data modeling, automated tools focus
- ◆ More rigorous and complete than SADT
- ◆ Industry merged key concepts from structured development and information engineering approaches into traditional approach

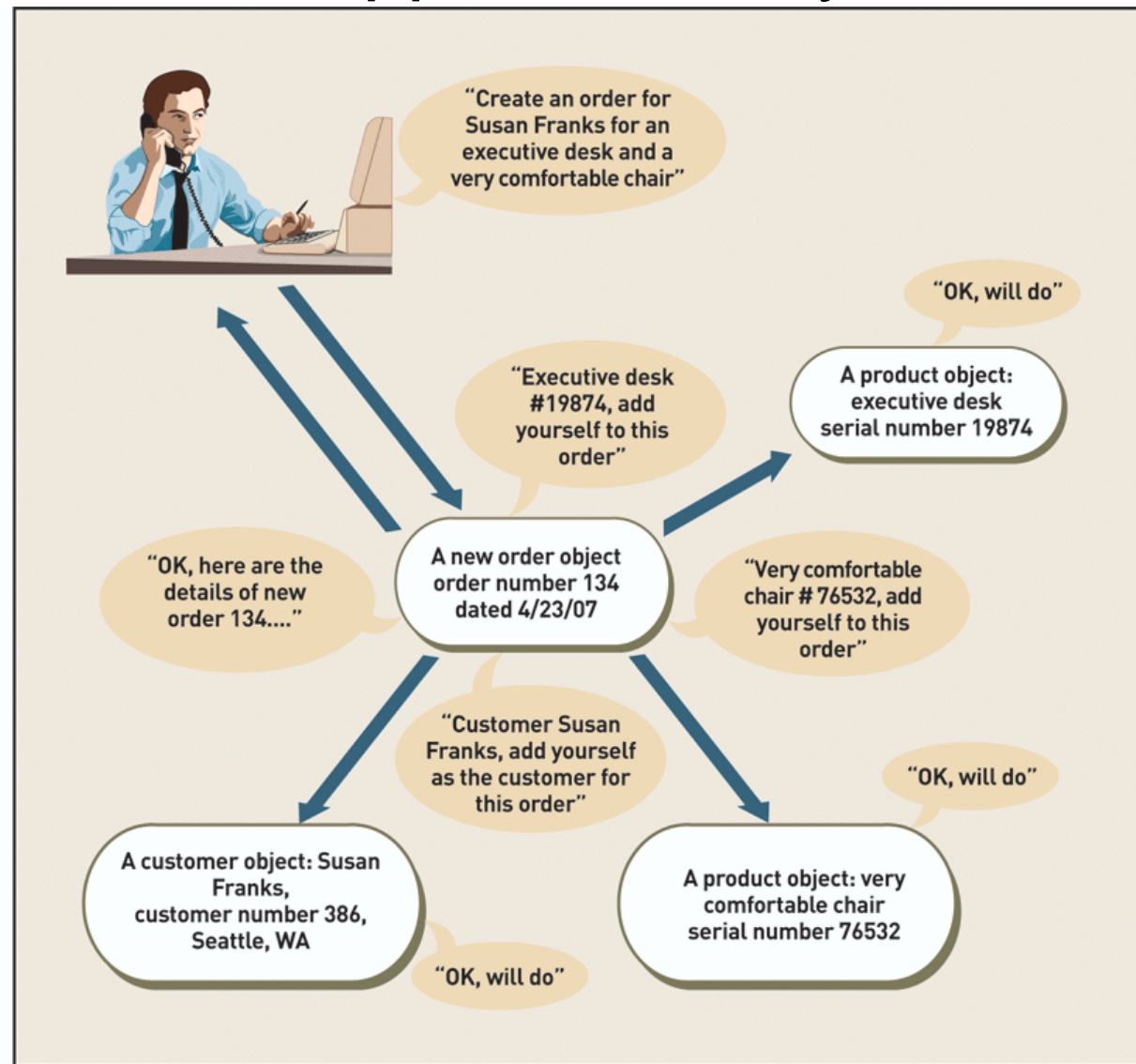
# Object-Oriented Approach

- ◆ Completely different approach to information systems
- ◆ Views information system as collection of interacting objects that work together to accomplish tasks
  - **Objects** – things in computer system that can respond to messages
  - Conceptually, no processes, programs, data entities, or files are defined – just objects
- ◆ OO languages: Java, C++, C# .NET, VB .NET

# Object-Oriented Approach to Systems

**Figure 2-18**

The object-oriented approach to systems (read clockwise starting with user)





# Object-Oriented Approach (continued)

## ◆ Object-oriented analysis (OOA)

- Defines types of objects users deal with
- Shows use cases are required to complete tasks

## ◆ Object-oriented design (OOD)

- Defines object types needed to communicate with people and devices in system
- Shows how objects interact to complete tasks
- Refines each type of object for implementation with specific language of environment

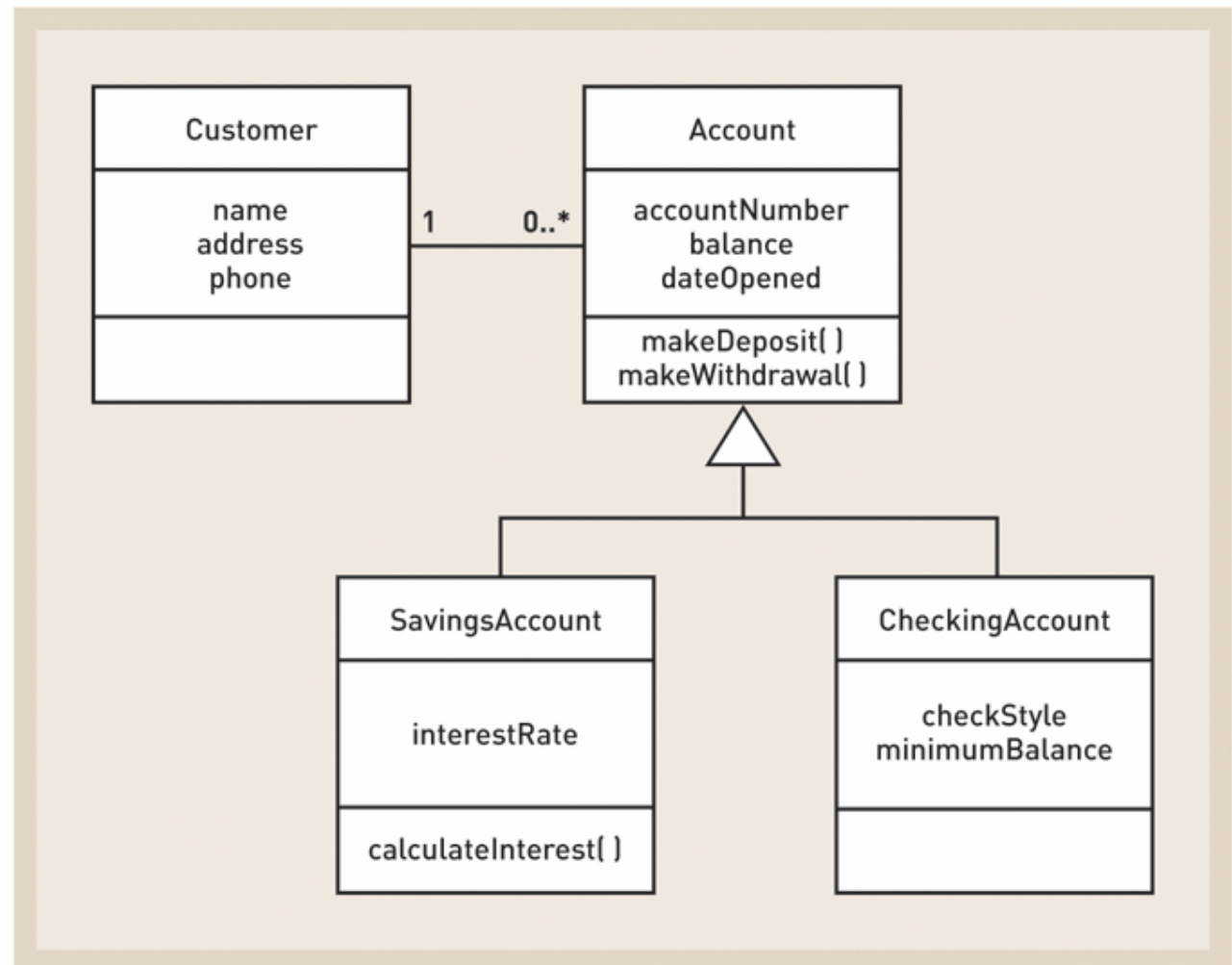
## ◆ Object-oriented programming (OOP)

- Writing statements in programming language to define what each type of object does

# Class Diagram Created During OO Analysis

**Figure 2-19**

A class diagram created during object-oriented analysis



# SDLC Variations

- ◆ Many variations of SDLC in practice
  - Based on variation of names for phases
  - No matter which one, activities/tasks are similar
- ◆ Some increase emphasis on people
  - User-centered design, participatory design
  - Sociotechnical systems
- ◆ Some increase speed of development
  - Rapid application development (RAD)
  - Prototyping

# Life Cycles with Different Names for Phases (Figure 2-20)

	Early Example of an SDLC	Information Engineering	Unified Process (UP)	SDLC with Activity Names for Phases
Planning Phase	Feasibility study	Information strategy planning	Inception phase	Organize the project and study feasibility
Analysis Phase	System investigation	Business area analysis		Study and analyze the current system
	Systems analysis		Elaboration phase	Model and prioritize the functional requirements
Design Phase	Systems design	Business system design	Construction phase	Generate alternatives and propose the best solution
		Technical design		Design the system
Implementation Phase	Implementation	Construction	Transition phase	Obtain needed hardware and software
		Transition		Build and test the new system
Support Phase	Review and maintenance	Production		Install and operate the new system

# Current Trends in Development

- ◆ More adaptive approaches
  - The Unified Process (UP)
  - Extreme Programming (XP)
  - Agile Modeling
  - Scrum
- ◆ Details on each in Chapter 16

# The Unified Process (UP)

- ◆ Object-oriented development approach
- ◆ Offered by IBM / Rational
  - Booch, Rumbaugh, Jacobson
- ◆ Unified Modeling Language (UML) used primarily for modeling
- ◆ UML can be used with any OO methodology
- ◆ UP defines four life cycle phases
  - Inception, elaboration, construction, transition

## The Unified Process (UP) (continued)

- ◆ Reinforces six best practices
  - Develop iteratively
  - Define and manage system requirements
  - Use component architectures
  - Create visual models
  - Verify quality
  - Control changes

# Extreme Programming (XP)

- ◆ Recent, lightweight, development approach to keep process simple and efficient
- ◆ Describes system support needed and required system functionality through informal user stories
- ◆ Has users describe acceptance tests to demonstrate defined outcomes
- ◆ Relies on continuous testing and integration, heavy user involvement, programming done by small teams



# Agile Modeling

- ◆ Hybrid of XP and UP (Scott Ambler); has more models than XP, fewer documents than UP
- ◆ Interactive and Incremental Modeling
  - Apply right models
  - Create several models in parallel
  - Model in small increments
- ◆ Teamwork
  - Get active stakeholder participation
  - Encourage collective ownership
  - Model with others and display models publicly

## Agile Modeling (continued)

### ◆ Simplicity

- Use simple content
- Depict models simply
- Use simplest modeling tools

### ◆ Validation

- Consider testability
- Prove model is right with code

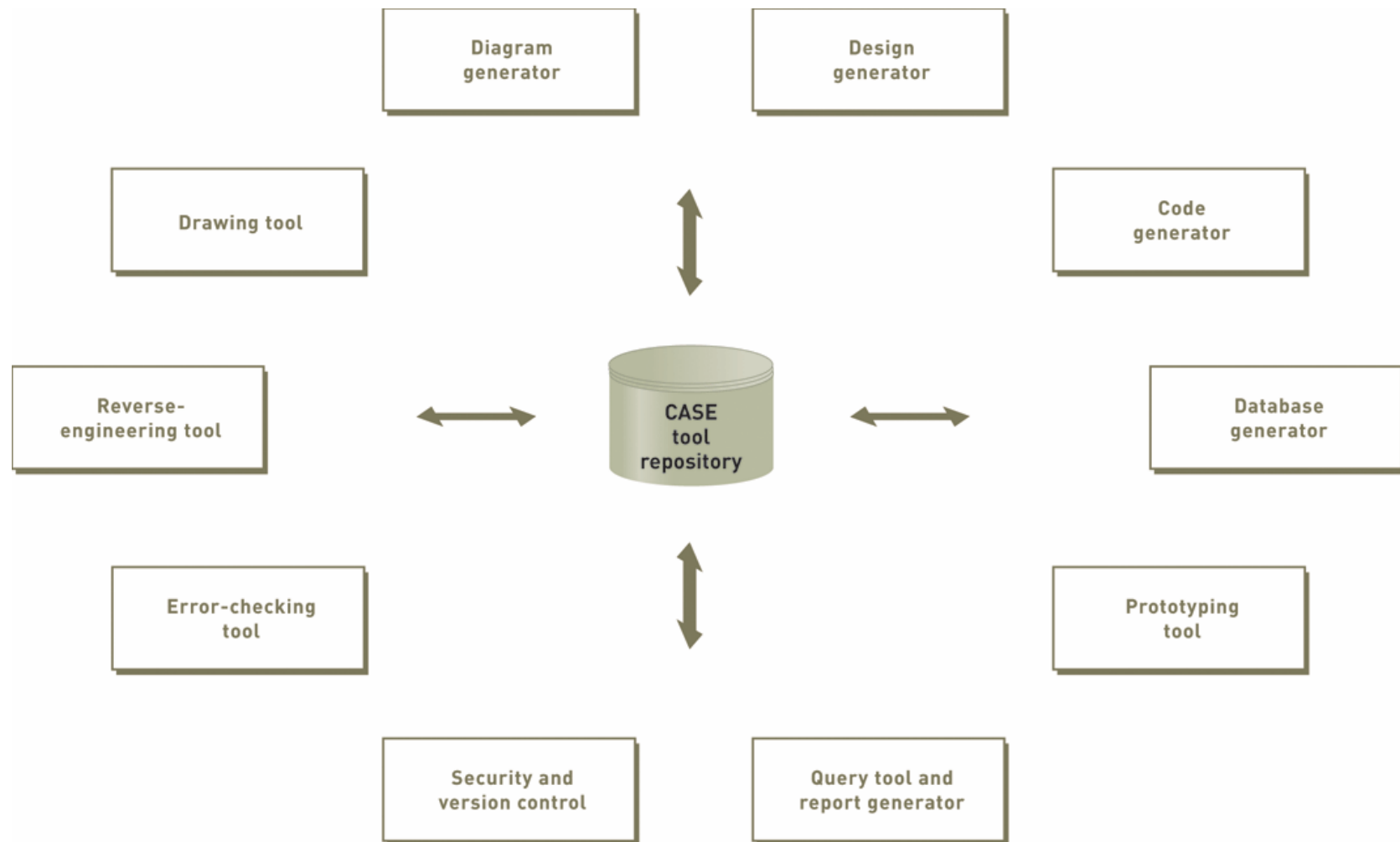
# Scrum

- ◆ For highly adaptive project needs
- ◆ Respond to situation as rapidly as possible
- ◆ Scrum refers to rugby game
  - Both are quick, agile, and self-organizing
- ◆ Team retains control over project
- ◆ Values individuals over processes

# Tools to Support System Development

- ◆ Computer-aided system engineering (CASE)
  - Automated tools to improve the speed and quality of system development work
  - Contains database of information about system called **repository**
- ◆ Upper CASE – support for analysis and design
- ◆ Lower CASE – support for implementation
- ◆ ICASE – integrated CASE tools
- ◆ Now called visual modeling tools, integrated application development tools, and round-trip engineering tools

# CASE Tool Repository Contains All System Information



## Summary

- ◆ System development projects are organized around the systems development life cycle (SDLC)
- ◆ Some projects use a predictive approach to the SDLC, and others use a more adaptive approach to the SDLC
- ◆ SDLC phases include project planning, analysis, design, implementation, and support

## Summary (continued)

- ◆ In practice, phases overlap, and projects contain many iterations of analysis, design, and implementation
- ◆ Models, techniques, and tools make up a system development methodology
- ◆ System development methodology provides guidelines to complete every activity in the SDLC

## Summary (continued)

- ◆ System development methodologies are based on traditional approach or object-oriented approach
- ◆ Current trends include: Extreme Programming (XP), Unified Process (UP), Agile Modeling, and Scrum
- ◆ CASE tools are designed to help analysts complete system development tasks