CS2062 Object Oriented Software Development

Lecture 1

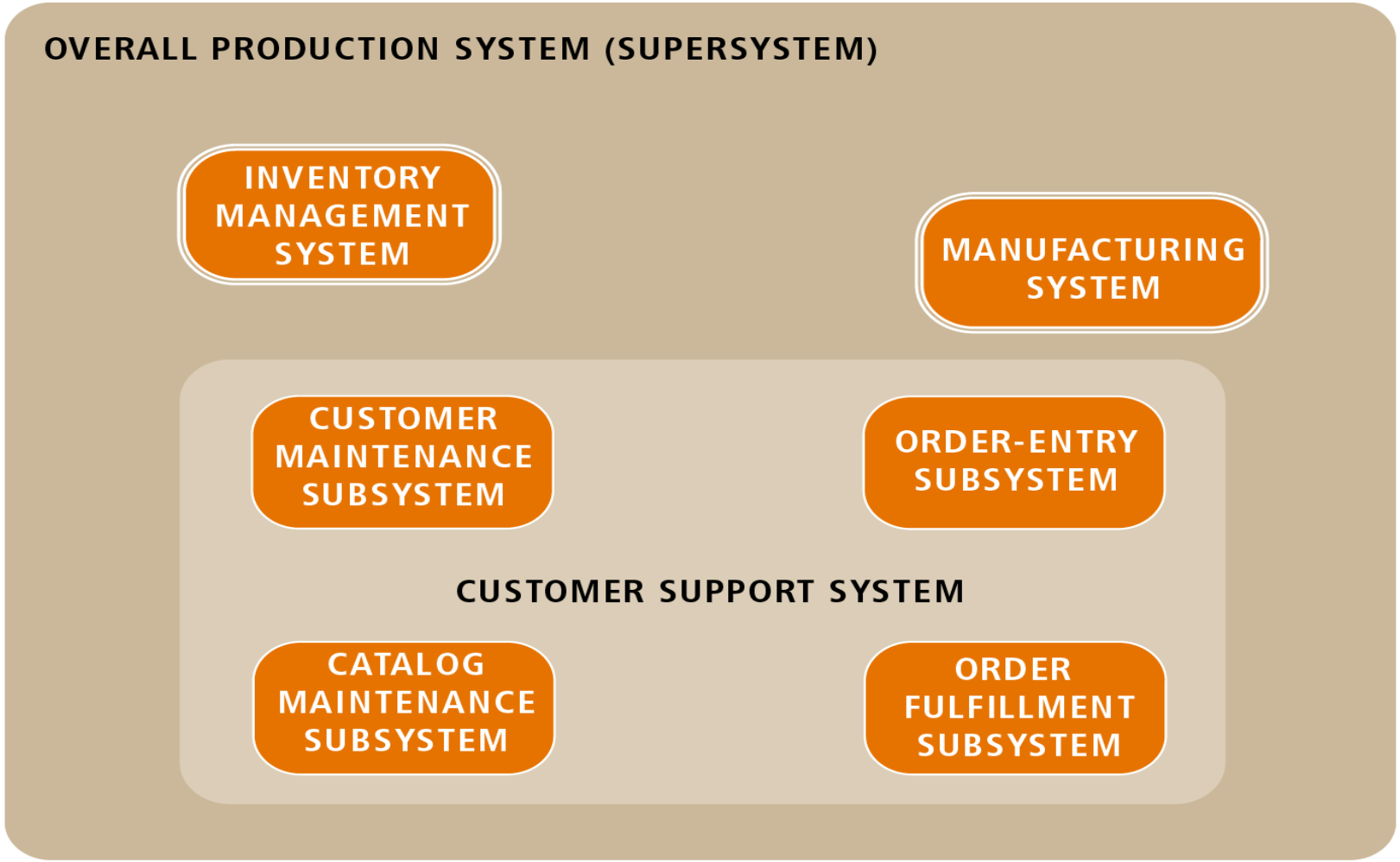
Introduction

Overview

- Systems analysis: comprehend information system functions
- Systems design: specify physical implementation
- Systems implementation: program your system

Systems that Solve Business Problems

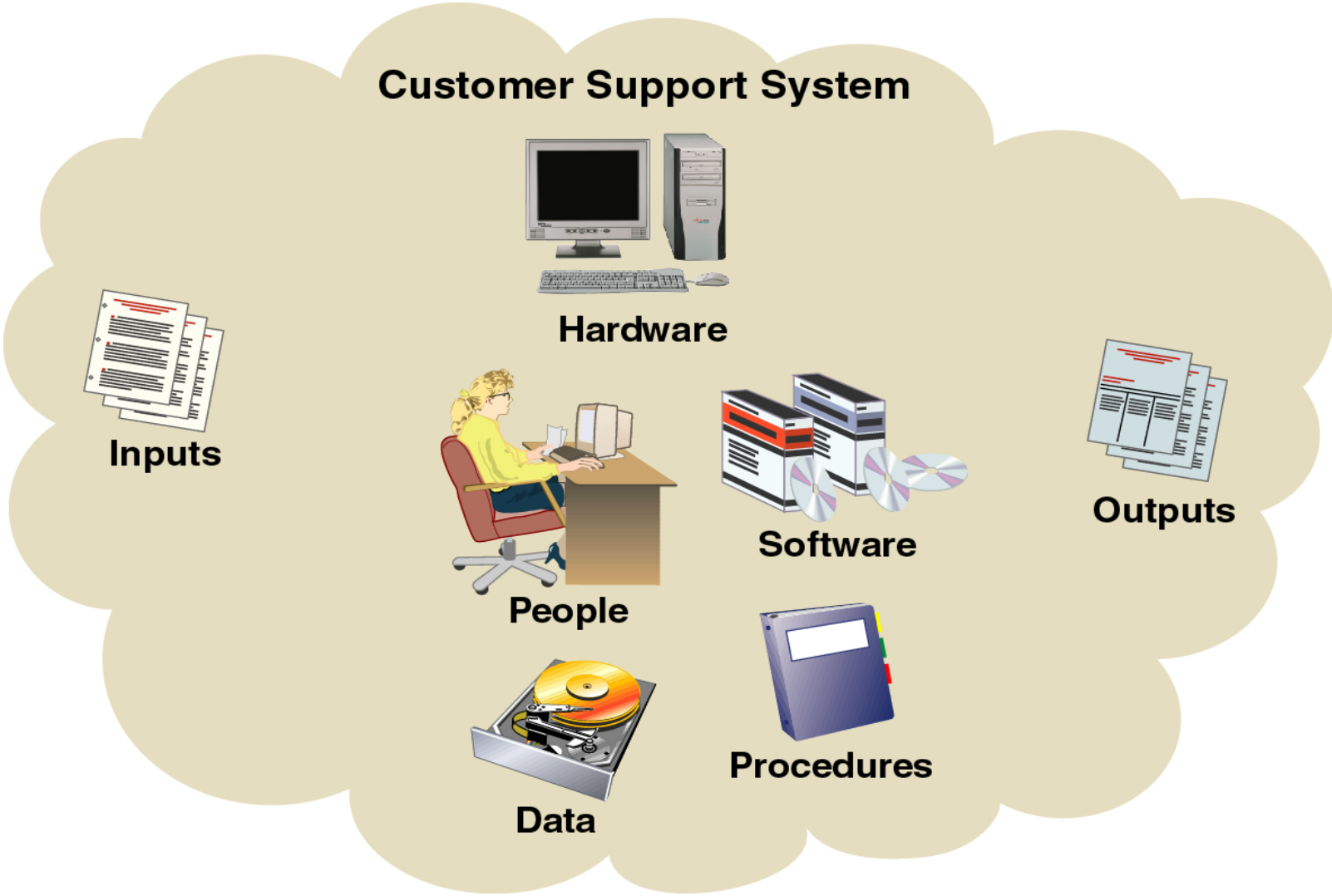
- System make-up: set of interrelated components
- System purpose: solve business problems
- System tools: functions or modules
- Functional decomposition: divide system into components to simplify analysis



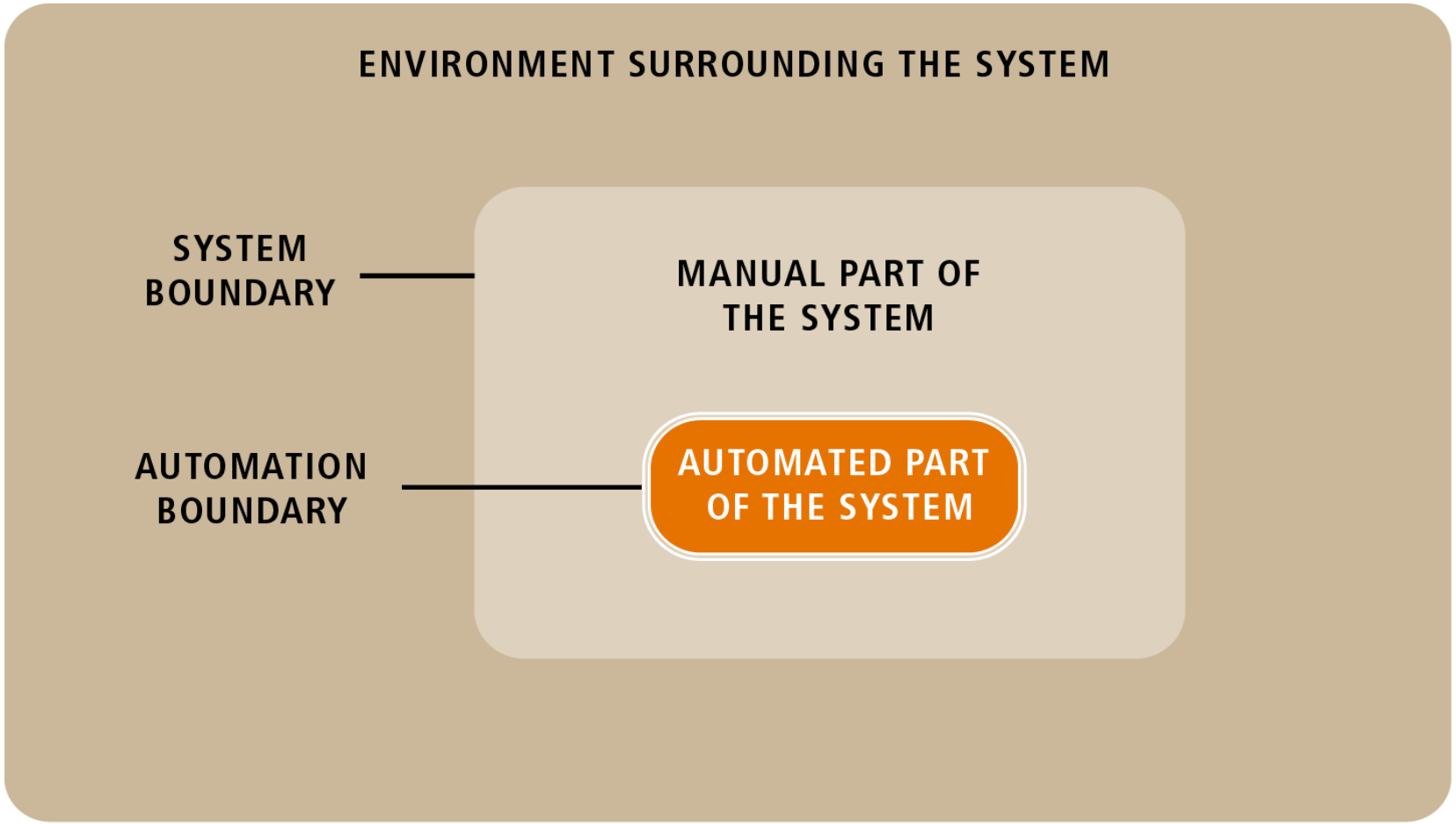
Information Systems and Subsystems

Information Systems

- Information system: collects, processes, stores, and outputs information
- Subsystem: components of a system
- Components: hardware, software, inputs, outputs, data, people, and procedures
- Supersystem: collection of systems
- Automation boundary: separates automated part of system from manual (human)



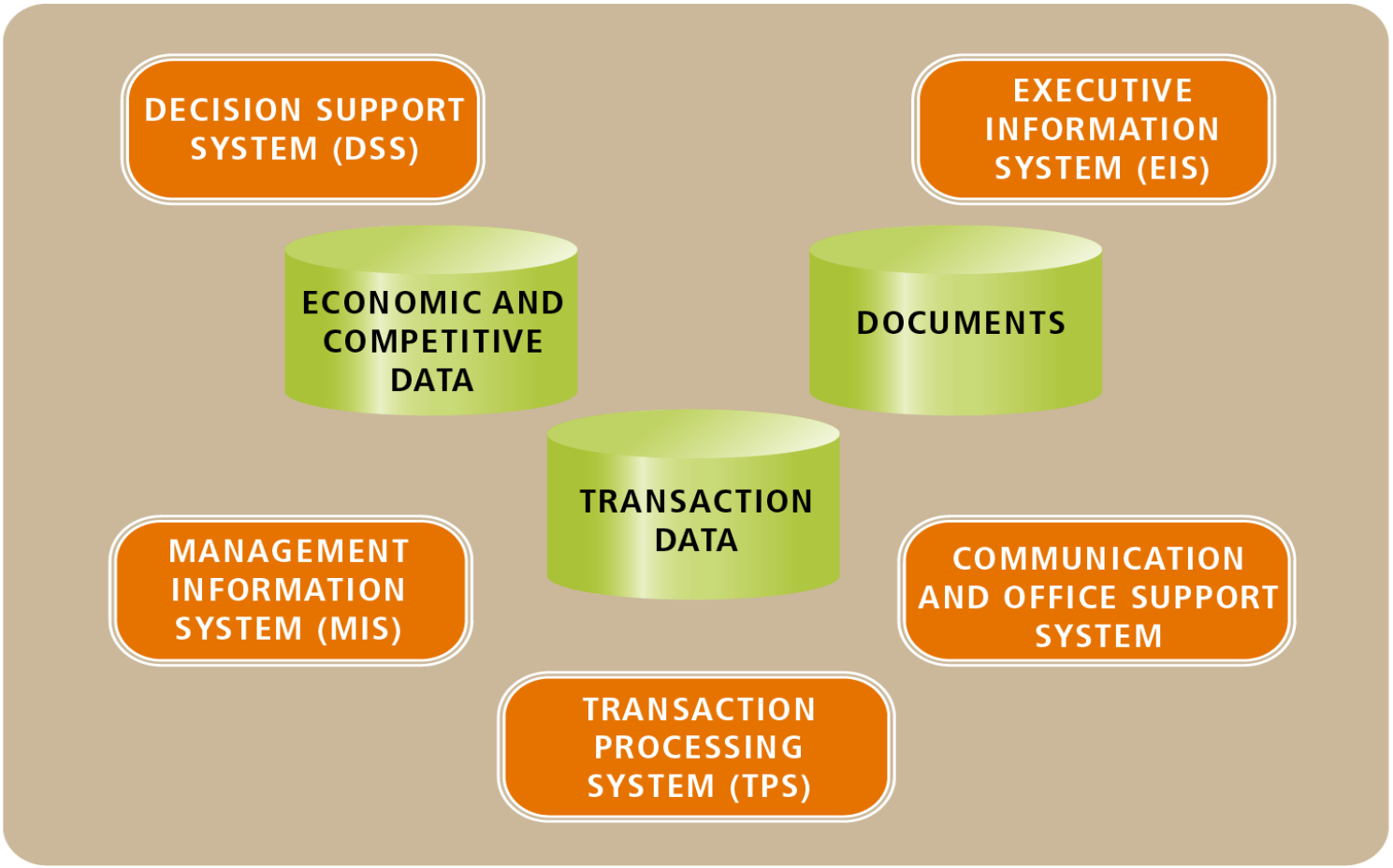
Information Systems and Component Parts



The System Boundary versus the Automation Boundary

Types of Information Systems

- There are many types of information systems
- Some common systems are found in most businesses
- Business systems center around transactions
- Systems must adapt to changing technology



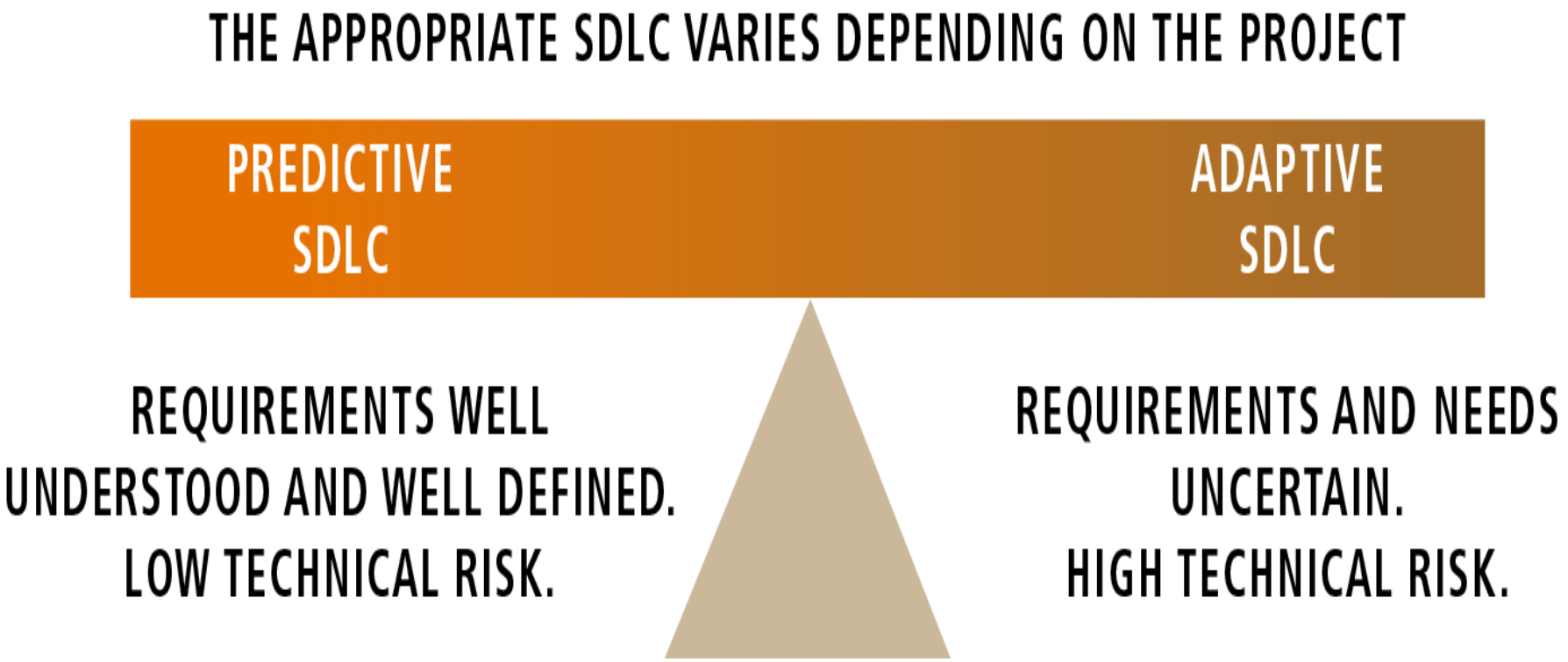
Types of Information Systems

Types of Technology

- Wide range: from desktops to large scale information systems
- Variety of computers connected by complex networks
- Technology change is continuous
- Innovation often drives information system change
- Regular upgrades of knowledge and skills essential

The Systems Development Life Cycle

- SDLC: process of building, deploying, using, and updating an information system
- Text focus: initial development project
- Chief variations of SDLC
  - Predictive: project planned entirely in advance
  - Adaptive: planning leaves room for contingencies
- Pure approaches to SDLC are rare
- Most projects have predictive and adaptive elements



Predictive versus adaptive approaches to the SDLC

The Traditional Predictive SDLC Approaches

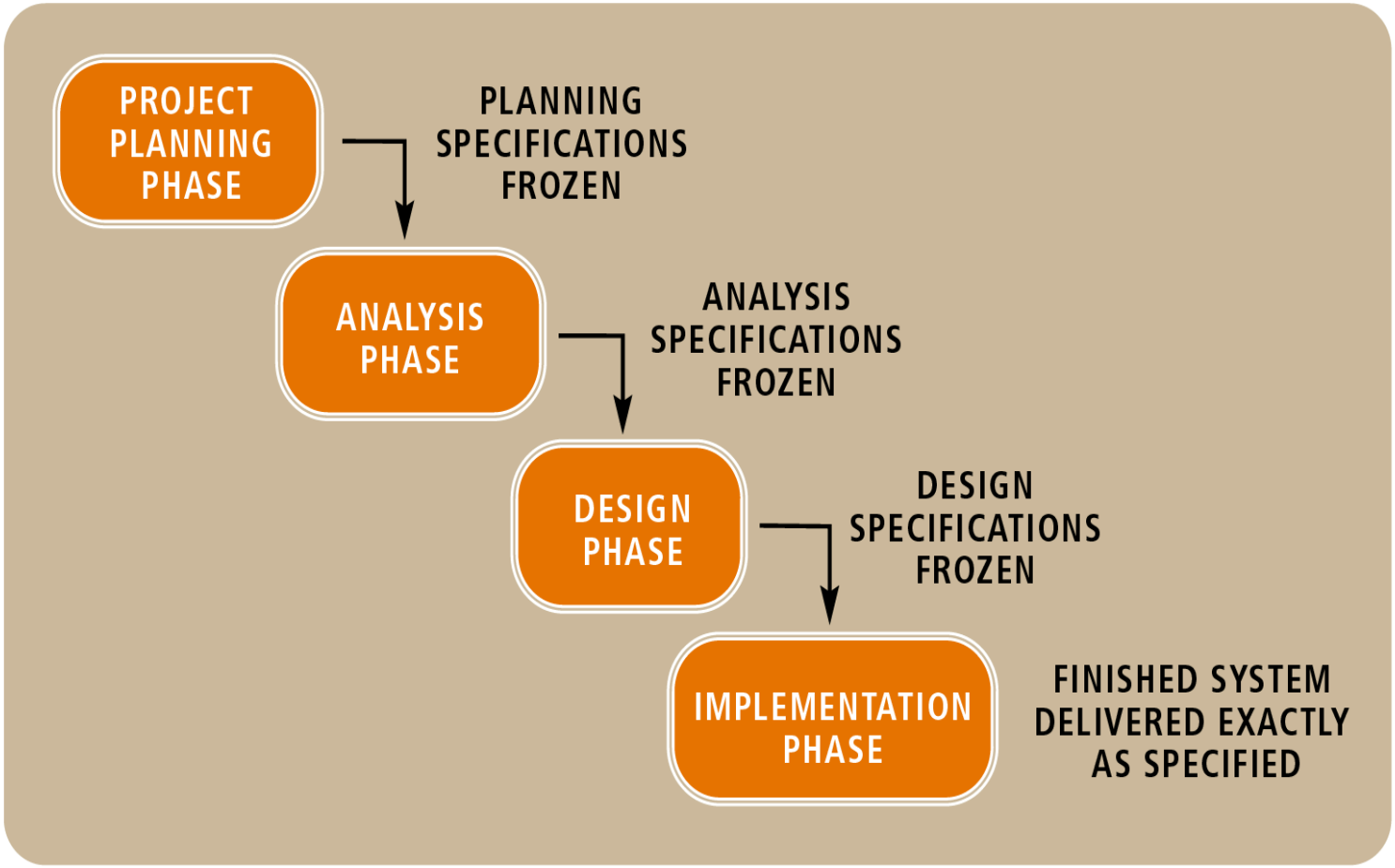
- Five activities or phases in a project
  - Planning, analysis, design, implementation, support

SDLC PHASE	OBJECTIVE
Project Planning	To identify the scope of the new system, ensure that the project is feasible, and develop a schedule, resource plan, and budget for the remainder of the project
Analysis	To understand and document in detail the business needs and the processing requirements of the new system
Design	To design the solution system based on the requirements defined and decisions made during analysis
Implementation	To build, test, and install a reliable information system with trained users ready to benefit as expected from use of the system
Support	To keep the system running productively initially and during the many years of the system's lifetime

SDLC Phases and Objectives

The Traditional Predictive SDLC Approaches

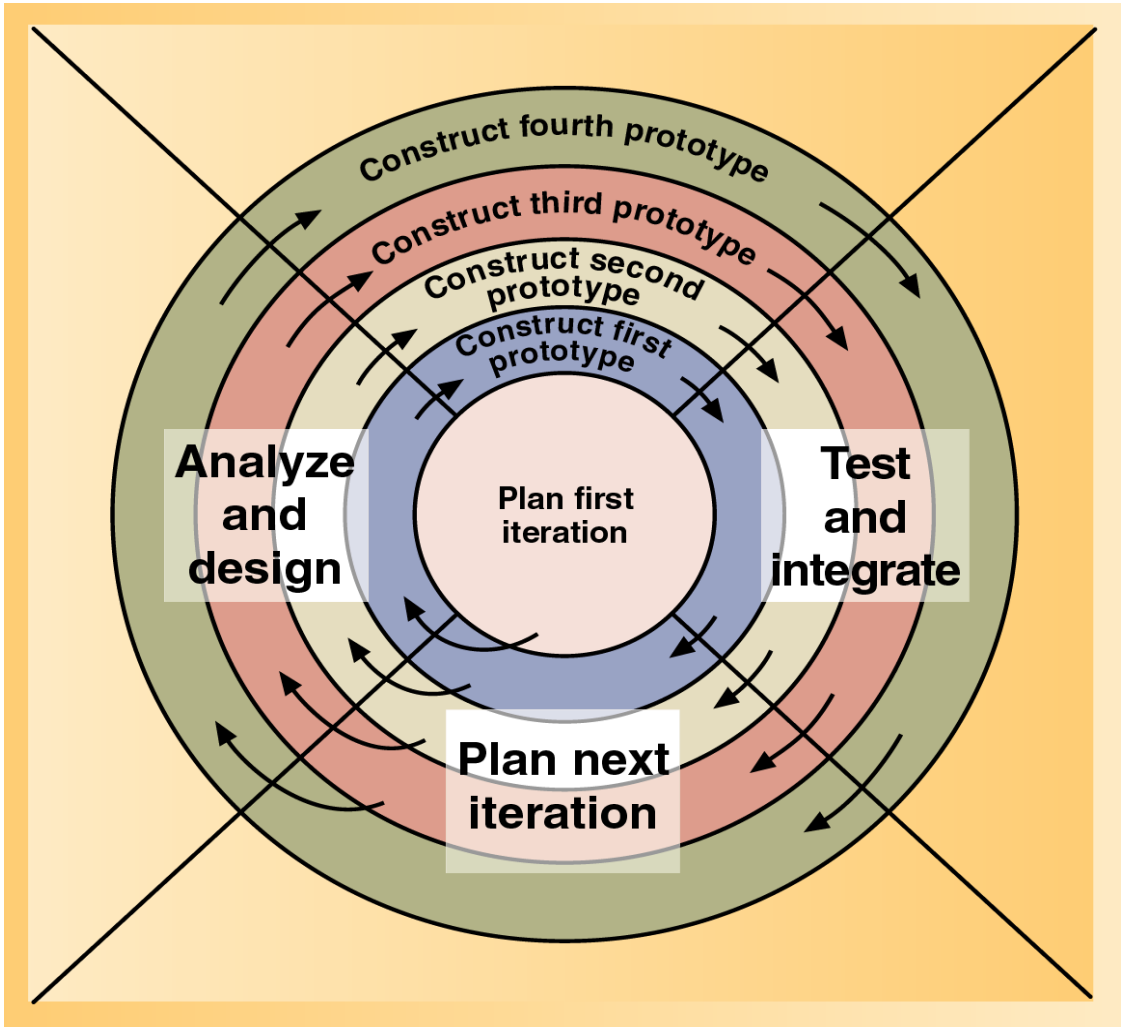
- Pure waterfall approach (predictive SDLC)
  - Assumes project phases can be sequentially executed
  - Project drops over the “waterfall” into the next phase
- Modified waterfall approach
  - Tempers pure waterfall by recognizing phase overlap
  - Informs many current projects and company systems



The Waterfall Approach to the SDLC

The Newer Adaptive Approaches to the SDLC

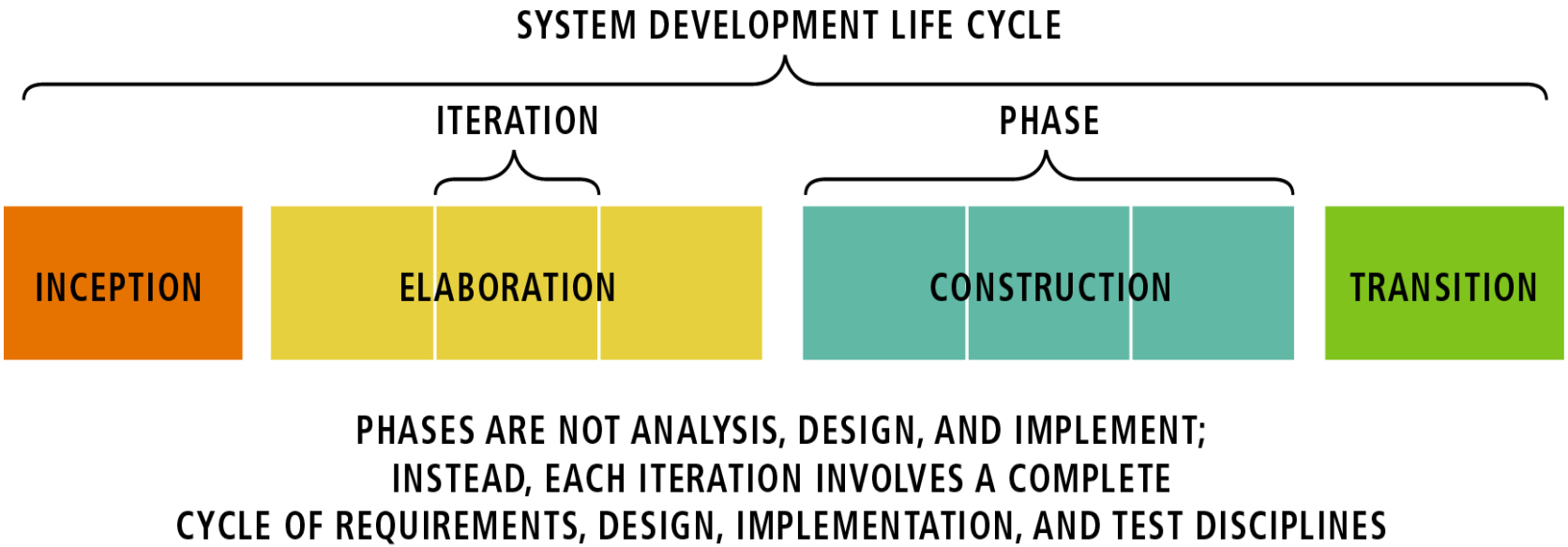
- The spiral model: early form of adaptive SDLC
  - Activities radiate from center starting point
  - Prototypes are artifacts of each phase
- Iterative problem solving: repeats activities
- Several approaches to structuring iterations
  - Define and implement the key system functions
  - Focus on one subsystem at a time
  - Define by complexity or risk of certain components
  - Complete parts incrementally



The Spiral Life Cycle Model

The Unified Process Life Cycle

- UP life cycle
  - Includes (4) phases which consist of iterations
  - Iterations are “mini-projects”
- Inception: develop and refine system vision
- Elaboration: define requirements and core architecture
- Construction: continue design and implementation
- Transition: move the system into operational mode



The Unified Process System Development Life Cycle

Methodologies and System Development Processes

- System development methodology
  - Provides guidelines for every activity in system development
  - Includes specific models, tools, and techniques
- UP is a system development methodology
- Process is a synonym for methodology
- Methodologies supported with documentation

Models

- Model abstract (separate) aspects of the real world
- Models come in many forms
  - Physical analogs, mathematical, graphical
- System development models are highly abstract
  - Depict inputs, outputs, processes, data, objects, interactions, locations, networks, and devices
- Unified Modeling Language (UML): standard notation
- PERT or Gantt charts: model project itself

Models of system components using UML

Use case diagram
Class diagram
Activity diagram
Sequence diagram
Communication diagram
Package diagram

Models used to manage development process

PERT chart
Gantt chart
Organization hierarchy chart
Financial analysis models (net present value, return on investment)

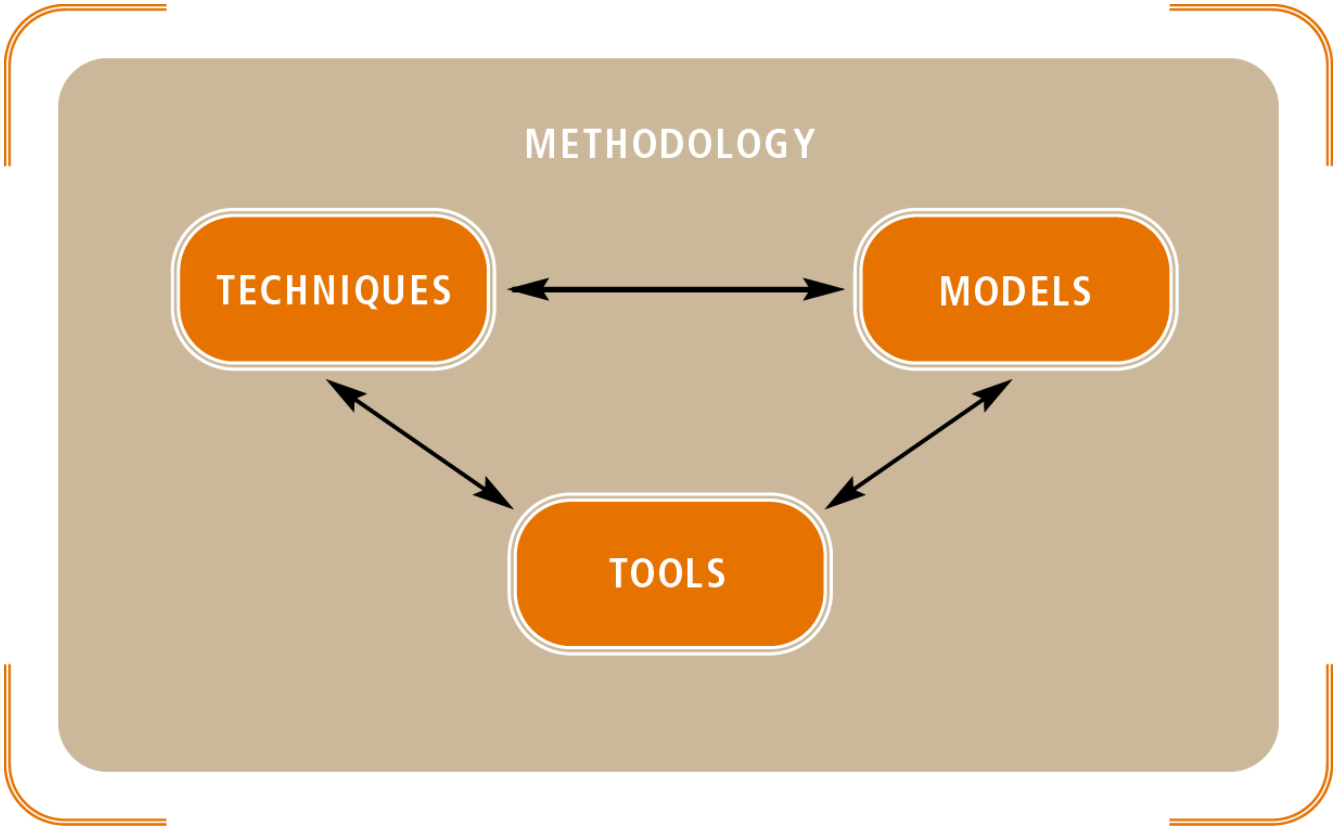
Some Models used in System Development

Tools

- Tool: software used to create models or components
- Example tools
  - Project management software tools (Microsoft Project)
  - Integrated development environments (IDEs)
  - Code generators
  - Computer-aided system engineering (CASE)

Techniques

- Technique
  - Collection of guidelines
  - Enables an analyst to complete an activity or task
- Example techniques
  - Domain-modeling , use case modeling, software-testing, user-interviewing techniques, relational database design techniques
- Proven techniques are embraced as “Best Practices”



Relationships of Models, Tools, and Techniques in a System Development Methodology

Overview of Object-Oriented Concepts

- OOA views system as a collection of objects
- Object: entity capable of responding to messages
- Languages: Simula, C++, Java, C#, Visual Basic .NET
- Object-oriented design (OOD)
  - Defines additional types of communication objects
  - Shows how the objects interact to complete tasks
  - Refines definition of objects for implementation
- Object-oriented programming (OOP): object coding

Recognizing the Benefits of OO Development

- Original application of object-oriented technology
  - Computer simulations
  - Graphical user interfaces
- Rationale for use in information systems
  - Benefits of naturalness
  - Reusability

Objects Are More Natural

- OO approach mirrors human perception: objects moving through space
- OOA, OOD, and OOP imitate perceptual processes by modeling classes of objects
- Some system developers resist OO development
- New programmers are more receptive to OO approach
- System users appreciate object-orientation
  - They discuss the objects involved in their work
  - Hierarchies are common tools for organizing knowledge

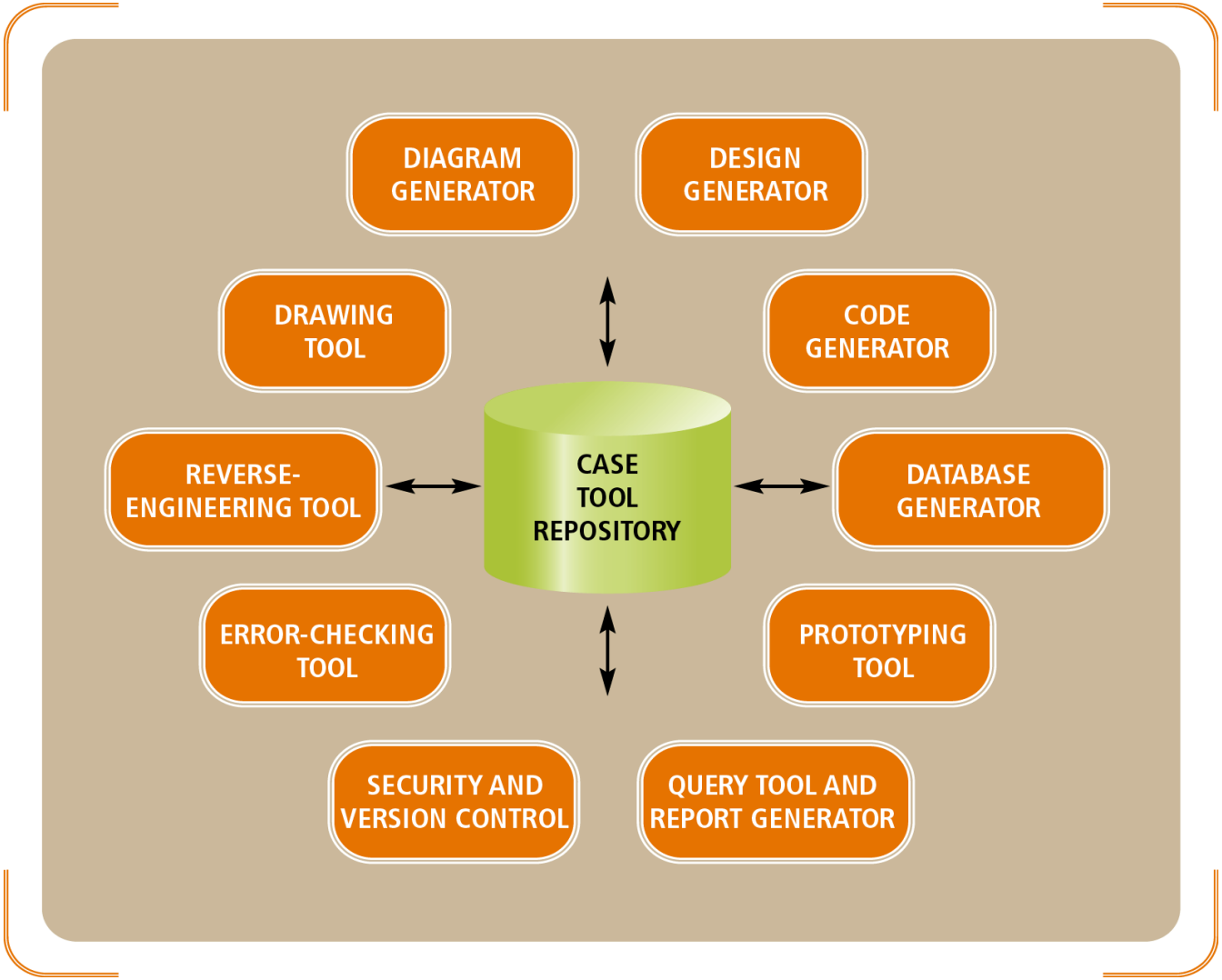
Classes of Objects Can Be Reused

- Classes of objects have a long shelf life
- Example: Customer class adaptability
  - Reused in systems where customer objects needed
  - Extended through inheritance to a new subclass
  - Reused during analysis, design, or programming
- Classes may be stored, with implementation hidden, in class libraries

Refresh Your OOP Knowledge ☺

Tools to Support System Development

- CASE (Computer Aided System Engineering)
  - Database repository for information system
  - Set of tools that help analysts complete activities
  - Sample artifacts: models, automatically generated code
- Variations on CASE
  - Visual modeling tools
  - Integrated application development tools
  - Round-trip engineering tools



A Case Tool Repository Contains All Information About the System

Tools to Support System Development (continued)

- Microsoft Visio: emphasizes technical drawing
- Rational Rose
  - CASE tool supporting object-oriented approach
  - Strongly identified with UP methodology
- Together
  - Pioneers round-trip engineering
    - synchronizes graphical models with generated program code
  - Leverages UML diagrams