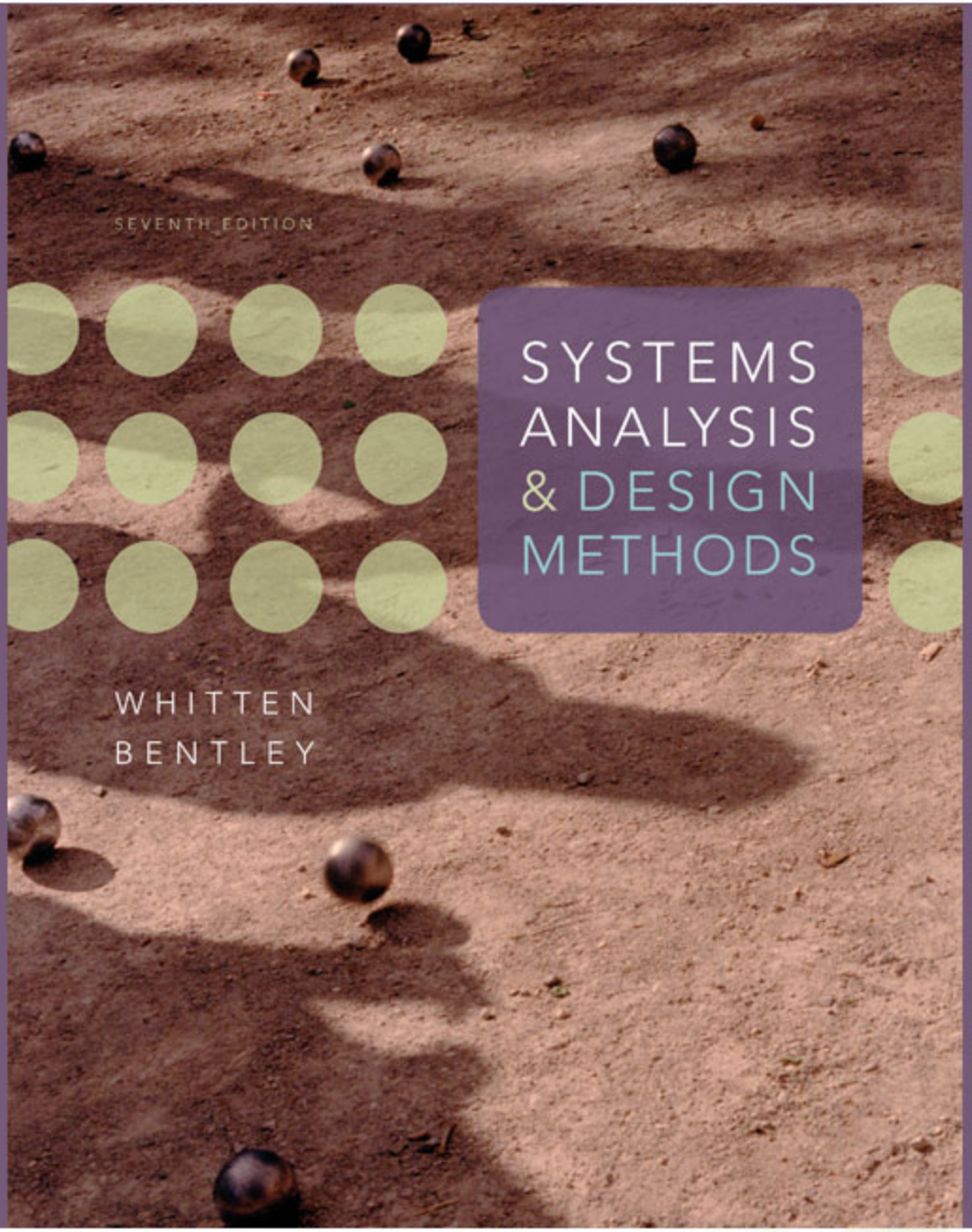
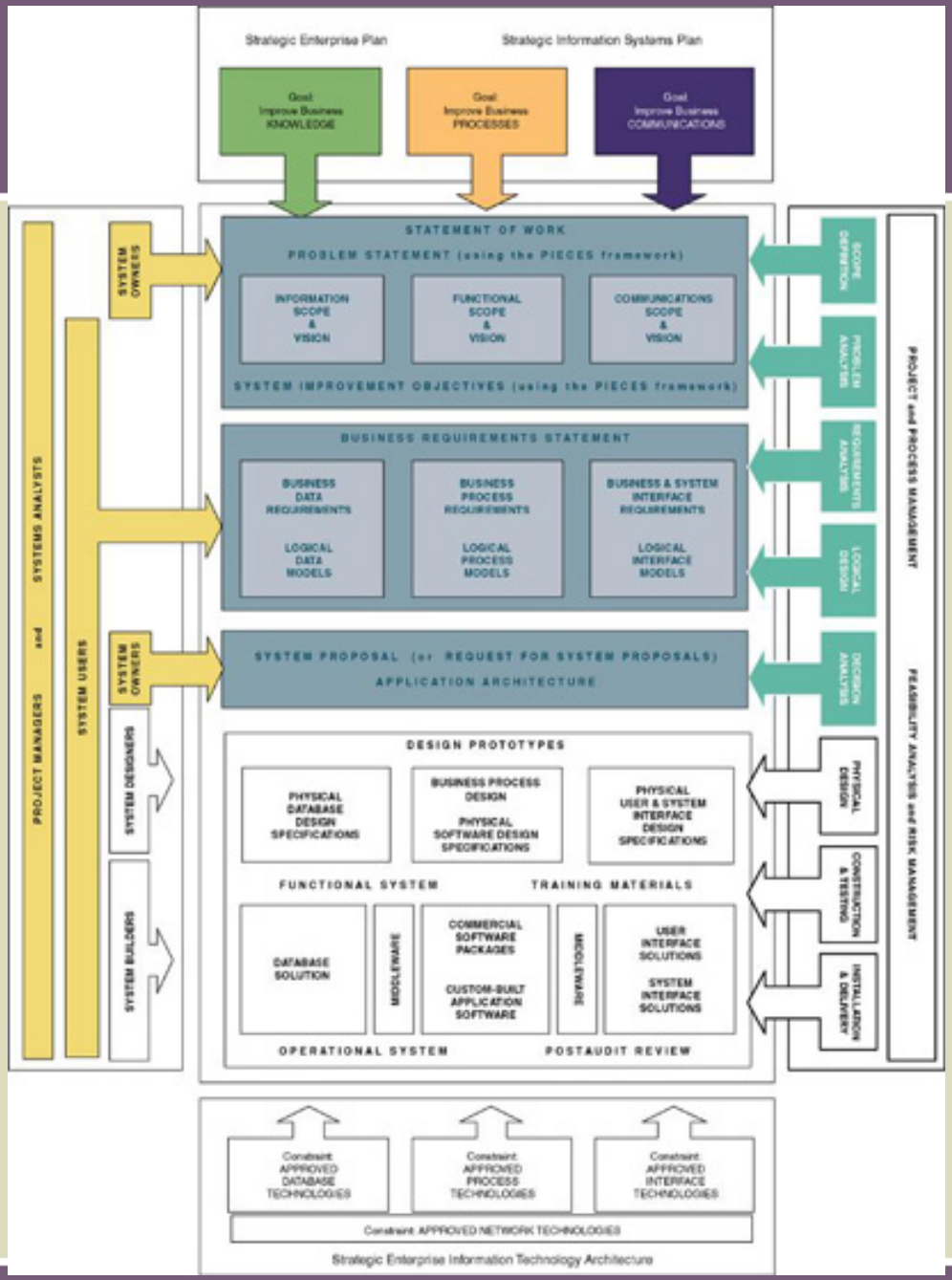

SEVENTH EDITION
SYSTEMS ANALYSIS & DESIGN METHODS
WHITTEN
BENTLEY

Objectives

- Define systems analysis and relate it to the scope definition, problem analysis, requirements analysis, logical design, and decision analysis phases.
- Describe a number of systems analysis approaches for solving business system problems.
- Describe scope definition, problem analysis, requirements analysis, logical design, and decision analysis phases in terms of information system building blocks.
- Describe scope definition, problem analysis, requirements analysis, logical design, and decision analysis phases in terms of purpose, participants, inputs, outputs, techniques, and steps.
- Identify those chapters in this textbook that can help you learn specific systems analysis tools and techniques.


Strategic Enterprise Plan
Strategic Information Systems Plan
Goal: Improve Business KNOWLEDGE
Goal: Improve Business PROCESSES
Goal: Improve Business COMMUNICATIONS
STATEMENT OF WORK
PROBLEM STATEMENT (using the PIECES framework)
INFORMATION SCOPE & VISION
FUNCTIONAL SCOPE & VISION
COMMUNICATIONS SCOPE & VISION
SYSTEM IMPROVEMENT OBJECTIVES (using the PIECES framework)
BUSINESS REQUIREMENTS STATEMENT
BUSINESS DATA REQUIREMENTS
LOGICAL DATA MODELS
BUSINESS PROCESS REQUIREMENTS
LOGICAL PROCESS MODELS
BUSINESS & SYSTEM INTERFACE REQUIREMENTS
LOGICAL INTERFACE MODELS
SYSTEM PROPOSAL (or REQUEST FOR SYSTEM PROPOSALS)
APPLICATION ARCHITECTURE
DESIGN PROTOTYPES
PHYSICAL DATABASE DESIGN SPECIFICATIONS
BUSINESS PROCESS DESIGN
PHYSICAL SOFTWARE DESIGN SPECIFICATIONS
PHYSICAL USER & SYSTEM INTERFACE DESIGN SPECIFICATIONS
FUNCTIONAL SYSTEM
TRAINING MATERIALS
DATABASE SOLUTION
MIDDLEWARE
COMMERCIAL SOFTWARE PACKAGES
CUSTOM-BUILT APPLICATION SOFTWARE
MIDDLEWARE
USER INTERFACE SOLUTIONS
SYSTEM INTERFACE SOLUTIONS
OPERATIONAL SYSTEM
POSTAUDIT REVIEW
SYSTEM OWNERS
SYSTEM USERS
SYSTEM OWNERS
SYSTEM DESIGNERS
SYSTEM BUILDERS
PROJECT MANAGERS and SYSTEMS ANALYSTS
SCOPE DEFINITION
PROBLEM ANALYSIS
REQUIREMENTS ANALYSIS
LOGICAL DESIGN
DECISION ANALYSIS
PHYSICAL DESIGN
CONSTRUCTION & TESTING
INSTALLATION & DELIVERY
PROJECT and PROCESS MANAGEMENT
FEASIBILITY ANALYSIS and RISK MANAGEMENT
Constraint: APPROVED DATABASE TECHNOLOGIES
Constraint: APPROVED PROCESS TECHNOLOGIES
Constraint: APPROVED INTERFACE TECHNOLOGIES
Constraint: APPROVED NETWORK TECHNOLOGIES
Strategic Enterprise Information Technology Architecture

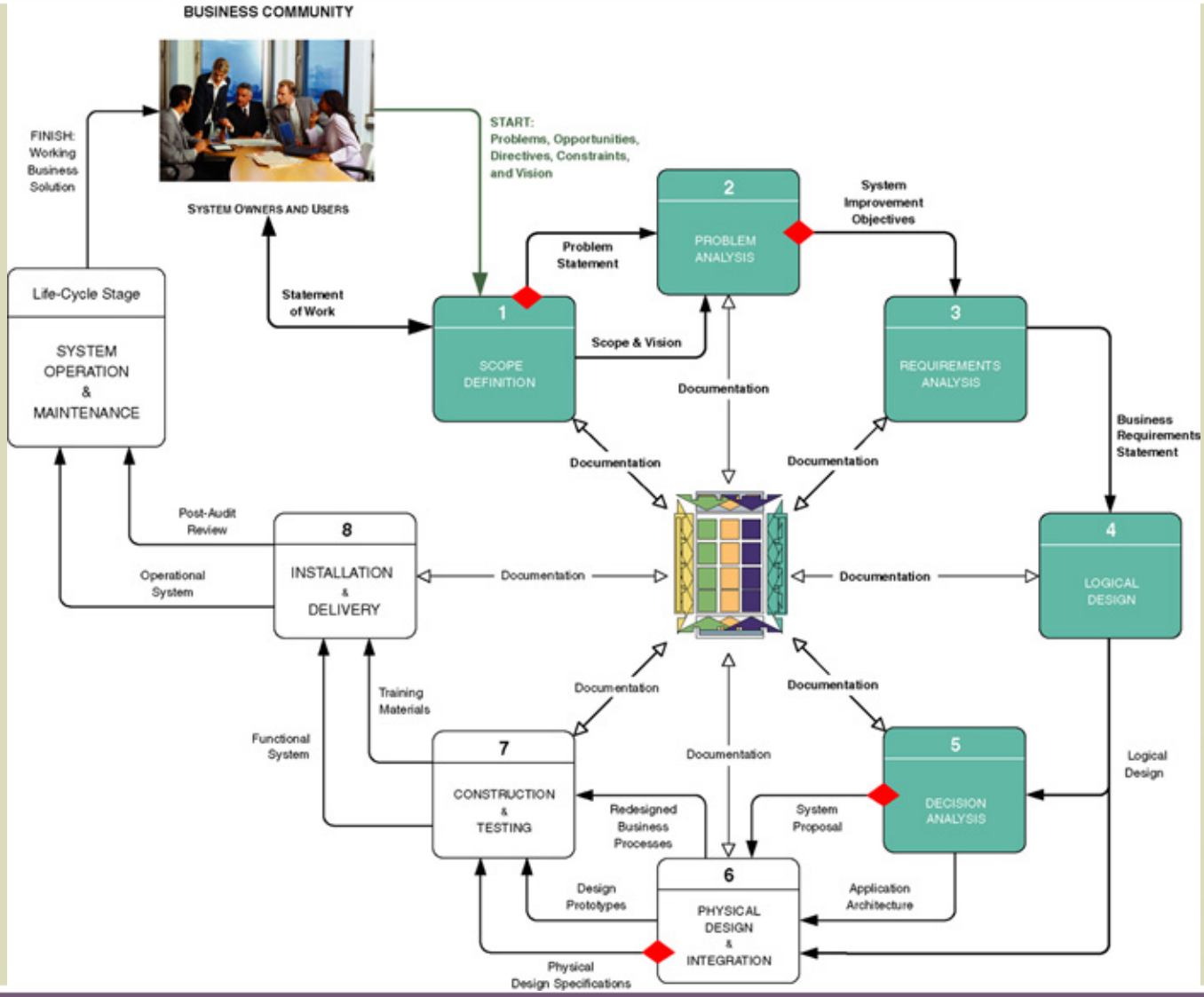
What is Systems Analysis ?

Systems analysis – a problem-solving technique that decomposes a system into its component pieces for the purpose of studying how well those component parts work and interact to accomplish their purpose.

Systems design – a complementary problem-solving technique (to systems analysis) that reassembles a system's component pieces back into a complete system—hopefully, an improved system. This may involves adding, deleting, and changing pieces relative to the original system.

Information systems analysis – those development phases in an information systems development project the primarily focus on the business problem and requirements, independent of any technology that can or will be used to implement a solution to that problem.

Context of Systems Analysis



Repository

Repository – a location (or set of locations) where systems analysts, systems designers, and system builders keep all of the documentation associated with one or more systems or projects.

- Network directory of computer-generated files that contain project correspondence, reports, and data
- CASE tool dictionary or encyclopedia (Chapter 3)
- Printed documentation (binders and system libraries)
- Intranet website interface to the above components

Model-Driven Analysis Methods

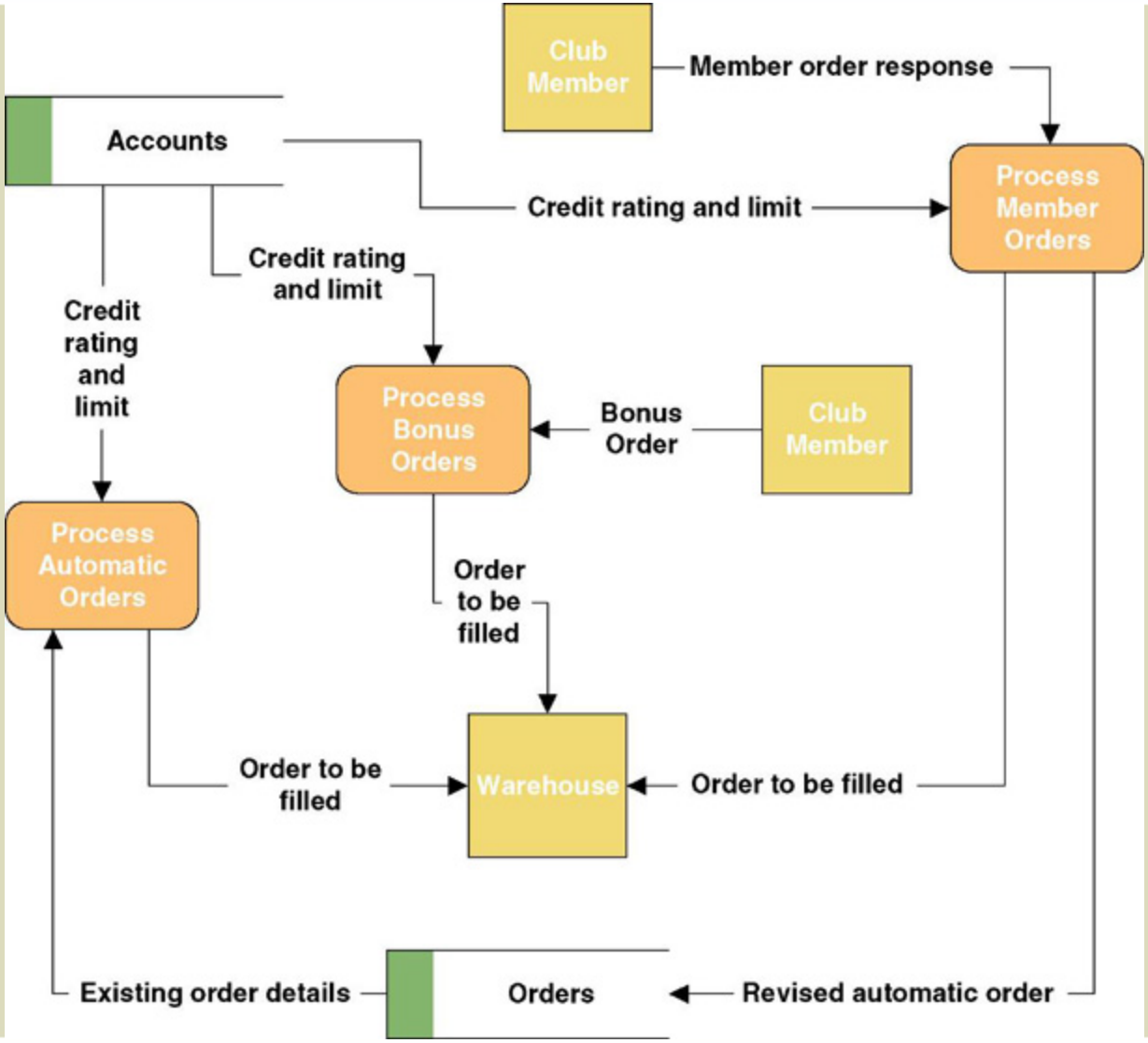
Model-driven analysis – a problem-solving approach that emphasizes the drawing of pictorial system models to document and validate both existing and/or proposed systems. Ultimately, the system model becomes the blueprint for designing and constructing an improved system.

Model – a representation of either reality or vision. Since “a picture is worth a thousand words,” most models use pictures to represent the reality or vision.

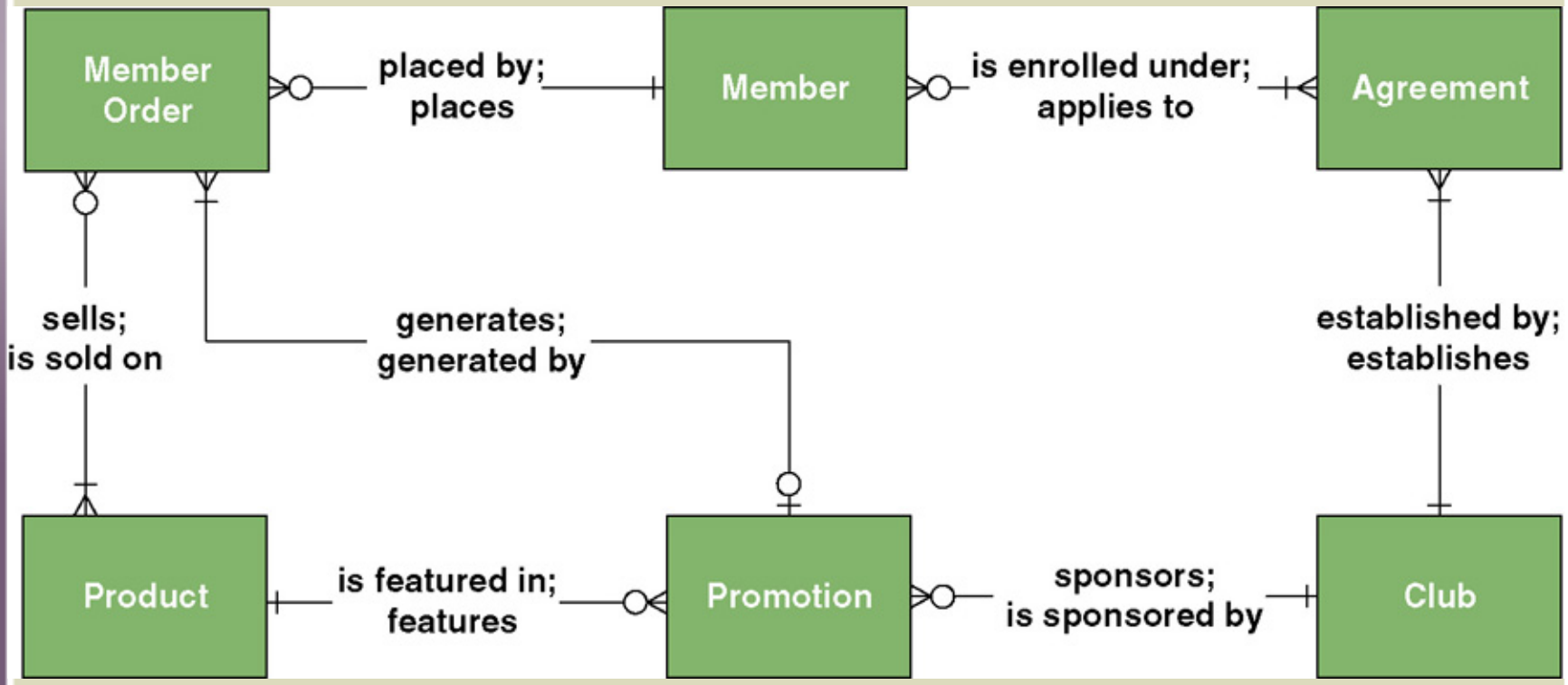
Model-Driven Approaches

- **Traditional Approaches**
  - Structured Analysis
    - Focuses on the flow of data through processes
    - Key model: data flow diagram
  - **Information Engineering**
    - Focuses on structure of stored data
    - Key model: entity relationship diagram
- **Object-Oriented Approach**
  - integrates data and process concerns into objects
    - Object – the encapsulation of the data (called properties) that describes a discrete person, object, place, event, or thing, with all the processes (called methods) that are allowed to use or update the data and properties. The only way to access or update the object's data is to use the object's predefined processes.
  - **Unified Modeling Language (UML)**

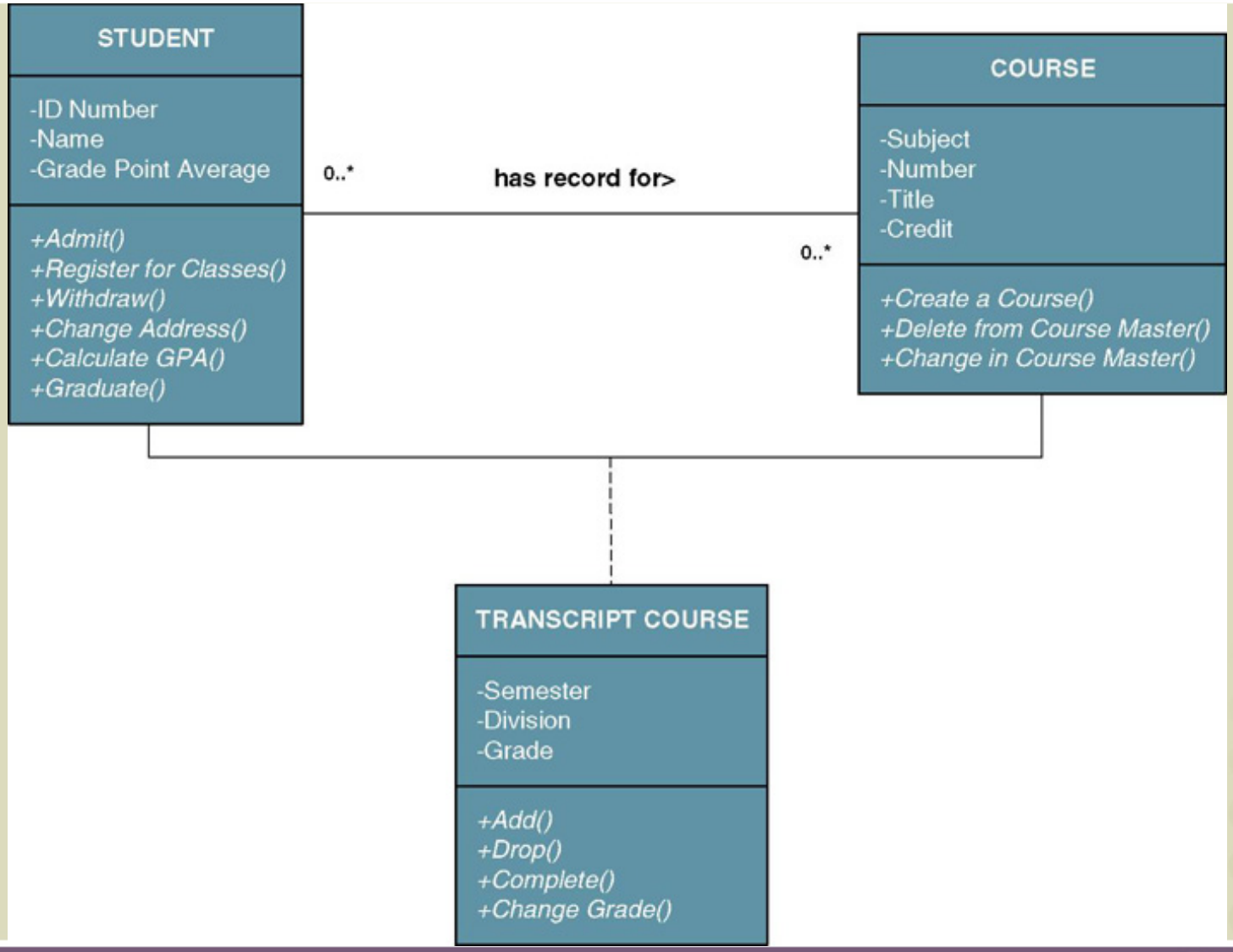
A Simple Process Model



A Simple Data Model



A Simple Object Model



Key Terms of Requirements Analysis Phase

Functional requirement – a description of activities and services a system must provide.

- inputs, outputs, processes, stored data

Nonfunctional requirement – a description of other features, characteristics, and constraints that define a satisfactory system.

- Performance, ease of learning and use, budgets, deadlines, documentation, security, internal auditing controls

Key Terms of Requirements Analysis Phase (cont.)

Use case – a business scenario or event for which the system must provide a defined response. Use cases evolved out of object-oriented analysis; however, their use has become common in many other methodologies for systems analysis and design.

Good Requirements- Understandable by the end users

Must be able to comprehend what is written

Requirements/Specifications docs are a contract

Ex: To ensure predictable operation, the system shall not employ nondeterministic methods. (NG)

REWRITE:

System operation shall be predictable and repeatable.

The system shall not employ nondeterministic methods.

Good Requirements- Nonprescriptive

Describe what the system will do, NOT how it will do it

Ex:

The software shall employ red-black trees for storage of information kept in memory. (NG)

It does not describe the behavior of the system from an external viewpoint. Any discussion of algorithms or/and data structures belong in the design document.

Good Requirements- Correct

A requirement I s defective if it is not correct. The user is the judge of correctness. If the user's intent is misrepresented, the requirements doc. Plain wrong.

The system shall accept operator input from up to and including 29 consoles. (NG)

Do the users intend to connect more consoles in the future? If the number is unknown, this is incorrect. If the users intend to connect 5 consoles, then a lot of waste of time, resources, and extra cost. The users are not expected to know the crucial factors that affect cost

Good Requirements- Complete

1) Completeness of requirements- no necessary requirements missing
2) No information is missing from each requirement

Ex: The system shall provide the operator with information needed to safely shut down the controlled machinery when an exception occurs. (NG)

The system shall provide the operator with time-stamped messages describing system exceptions.

[List of exceptions]

The messages shall not lag more than TBD seconds behind the exceptions they describe.

Good Requirements- Concise

We feel that good system provide the end user with good value. Because of this, we think that the system should provide adequate performance with 1 500GB disk, since this is the least expensive we can buy from the designated vendor….the user may want to configure the system with a larger disk (NG)

The system shall fulfill all specified functions when configured with a 500 GB disk.

Good Requirements- Precise

The requirement must say exactly what it means to say- no vagueness or misrepresentations.

Ex: The system shall accept valid employee ID numbers from 1 to 9999999. (NG)

It seems OK but any questions!

The system shall accept only valid ID numbers as defined elsewhere (Item Ref). No otherwise valid number will be accepted unless it is between 1 and 9999999 inclusive, represented without leading zeros.

Good Requirements- Consistent

A set of requirements is inconsistent when 2 parts contradict each other.

The system shall track detected airborne objects travelling at speeds between 200 to 400 miles per hour inclusive.

The system shall flag all detected airborne objects traveling at speeds from 300 to 500 miles per hour inclusive.

Is there a problem?

Good Requirements- Traceable

Do you see the 4th paragraph from the top? No, I mean the 4th complete paragraph? The requirements seem untestable to me!

Good Requirements- Modifiable

See Requirements for the section on consistent requirements. Suppose that the problem arose from changing 500 to 400– Best way to write

1.1.1 The normal operational range is 200 to 500 mph, inclusive.

1.1.2 Airborne objects are called “exceptional” if they are traveling at speed more than 100 mph above the floor of the normal range.

1.1.3 The system shall only track airborne objects traveling in the normal operational range

1.1.4 The system shall flag all exceptional objects.

Good Requirements- Feasible

Absurd requirements:

The software shall operate on a 100 MGHz 486 system.

The software shall respond to any critical event within 1 pico-second.

The required software is a C main program that links with an arbitrary collection of user supplied functions.

An error message will be issued if the program should try to call a nonexistent user function.