Major Exam II Reschedule

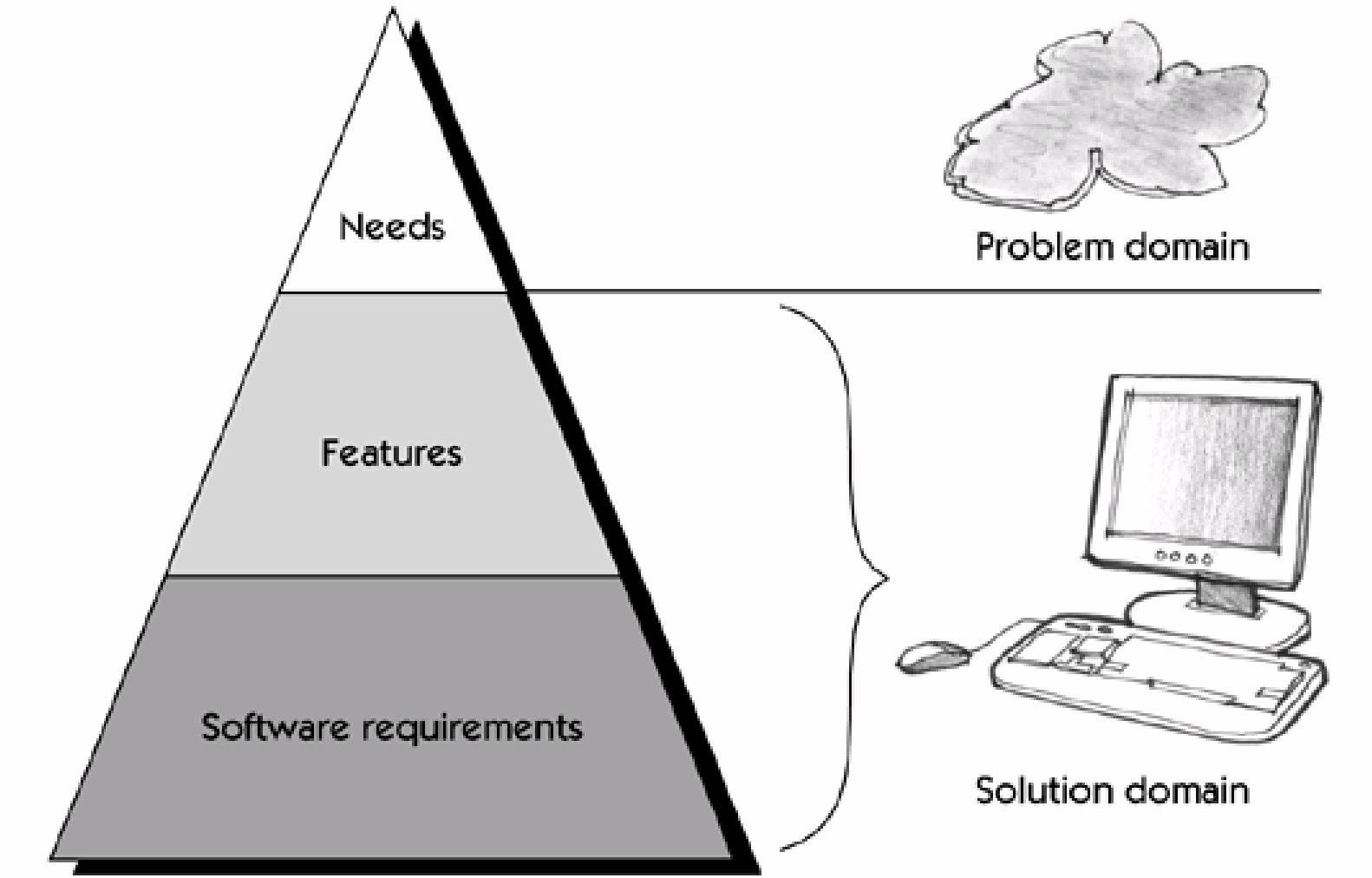
5:30 – 7:30 pm

in Tue Dec 5th

Recall The Team Skills

1. Analyzing the Problem (with 5 steps)
2. Understanding User and Stakeholder Needs
3. Defining the System
4. Managing Scope
    1. Establishing project scope
    2. Managing your customer
5. **Refining the System Definition**
6. Building the Right System

Requirements Pyramid



Team Skill 5
Refining the System Definition

- Ch 20. Software Requirements: a more rigorous look
- Ch 21: Refining the Use Cases
- Ch 22: Developing the Supplementary Specification
- Ch 23: On Ambiguity and Specificity
- Ch 24: Technical Methods for Specifying Requirements

Chapter 20
Software Requirements
A More Rigorous Look

- Relationships between requirements, features and use cases.
- Types of requirements.
- Requirement Vs. Design

Introduction

- In previous team skills, the features and the use-case models were at a high level of abstraction for the following reasons.
  - We can better understand the main characteristics of the system by focusing on its features and key use cases and how they fulfill user needs.
  - We can assess the system for its completeness, its consistency, and its fit within its environment.
  - We can use this information to determine feasibility and to manage the scope of the system before making significant resource investments.

Looking Deeper into Software Requirements

- Definition of a software requirement:

1. A software capability needed by the user to solve a problem or to achieve an objective
2. A software capability that must be met or possessed by a system or a system component to satisfy a contract, standard, specification, or other formally imposed documentation

Looking Deeper into Software Requirements

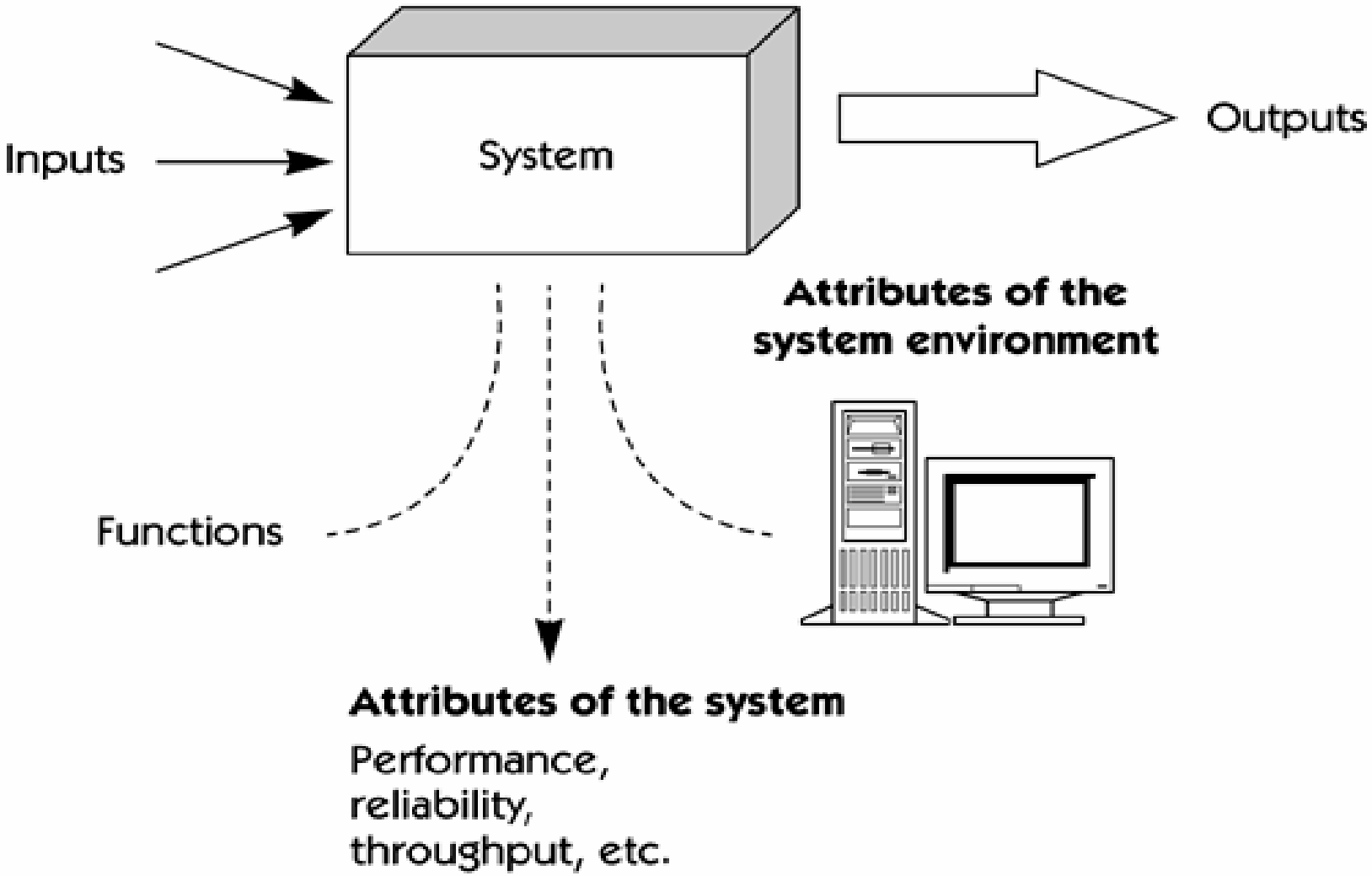
- To fully describe the behavior of a software system we need 5 major classes:

1. Inputs to the system: Not only the content of the input but also, as necessary, the details of input devices and the form, look, and feel—protocol—of the input.
2. Outputs from the system: A description of the output devices, such as voice-output or visual display, that must be supported, as well as the protocol and formats of the information generated by the system.

Looking Deeper into Software Requirements

3. Functions of the system: The mapping of inputs to outputs, and their various combinations.
4. Attributes of the system: non-functional requirements like reliability, maintainability, availability, and throughput that the developers must taken into account.
5. Attributes of the system environment: additional non-functional requirements as the ability of the system to operate with other applications, loads, and operating systems.

System Elements



The Relationship between Software Requirements and Use Cases

- Use cases are just one way to express software requirements.
- Use cases can't conveniently express certain types of requirements
- Example: "the application must support up to 100 simultaneous users“
- There are better ways to express other types of requirements as well (Chapter 22).

The Relationship between Features and Software Requirements

Features

- Simple descriptions of system services in a shorthand manner.
- Help us understand and communicate at a high level of abstraction.
- We can't fully describe the system and write code from those descriptions. They are too abstract for this purpose.

Software Req.s

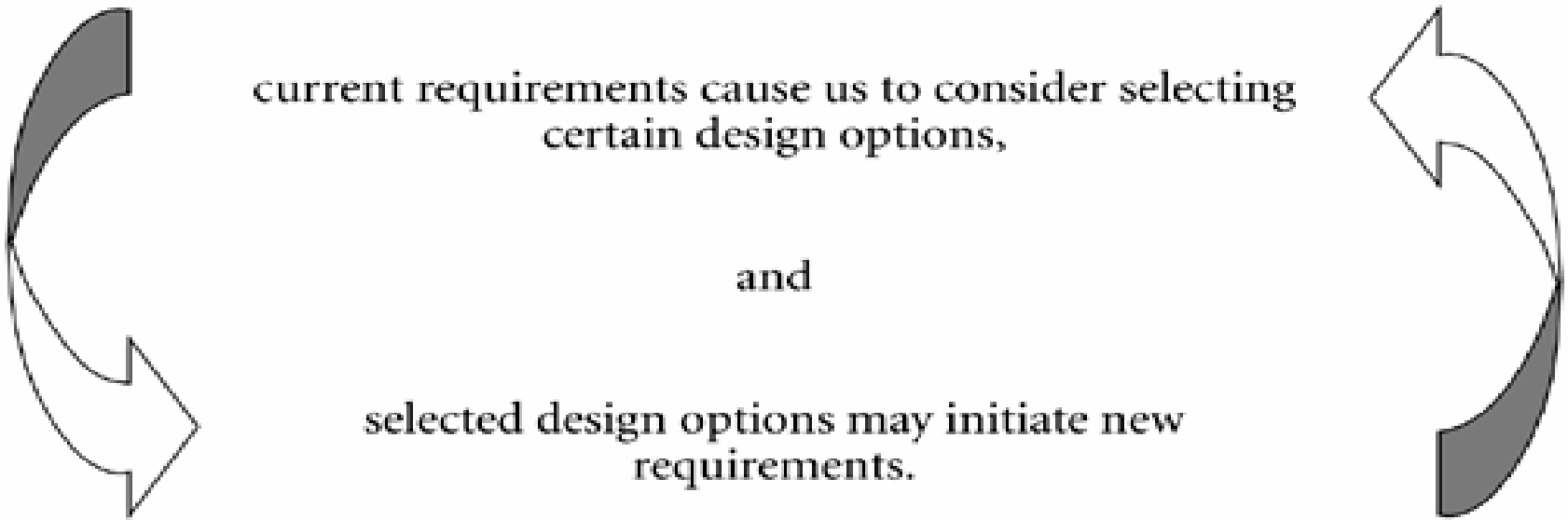
- Detailed descriptions of system services (features).
- We can code from them.
- They should be specific enough to be "testable"

The Requirements Dilemma
What versus How

- Requirements shall tell us what the system is to do, and NOT how the system shall do it.
- Exclude project information:
  - Information associated with project management (schedules, verification and validation plans, budgets, and staffing schedules)
  - Information about how the system will be tested.
- Exclude design information
  - System design or architecture.

Requirements versus Design

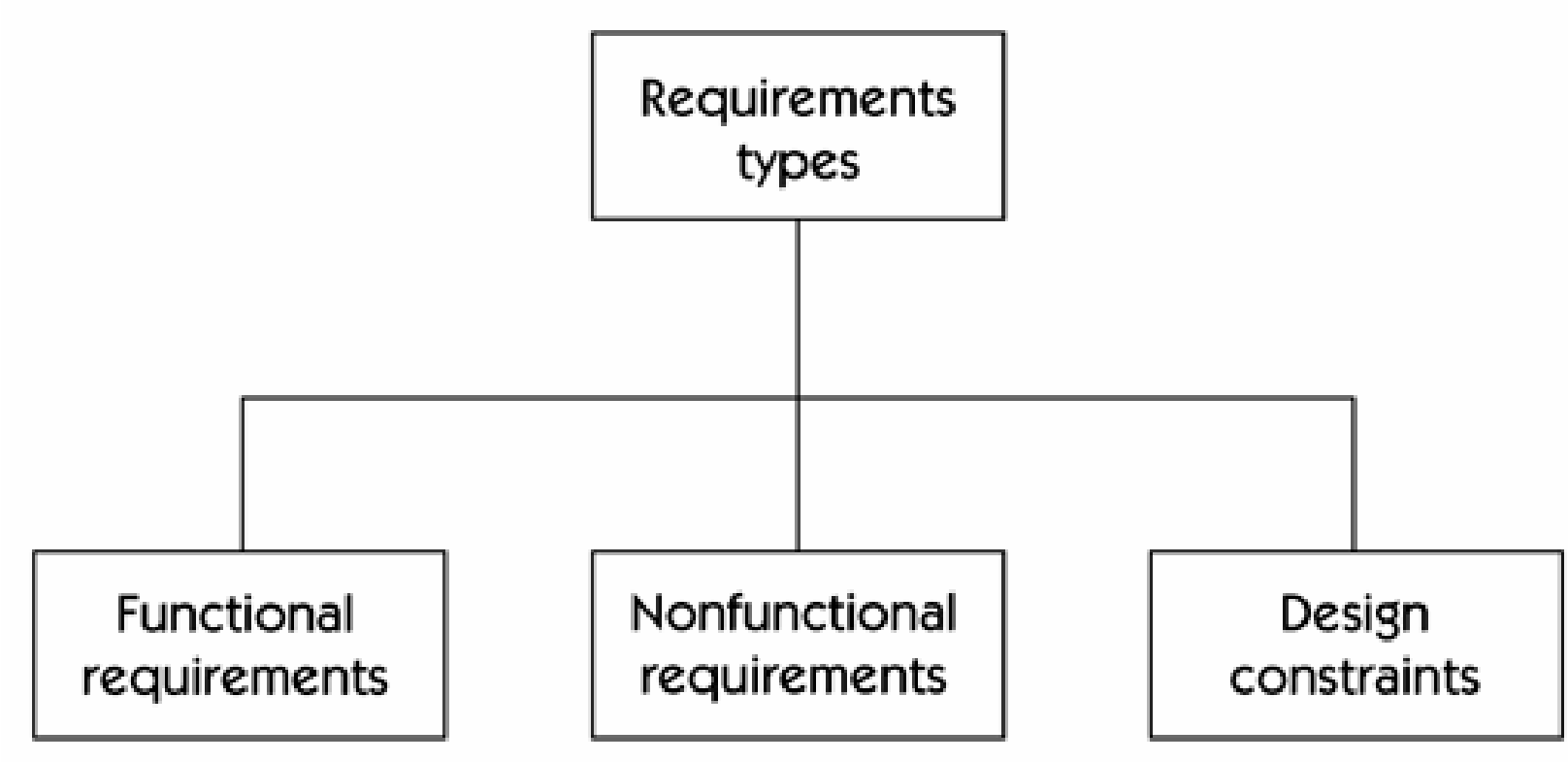
- Software requirements and design are iterative
  - Current requirements cause certain design decisions
  - Design decisions develop new requirements



Types of Requirements

- Functional software requirements: Express how the system behaves—its inputs, its outputs, and the functions it provides to its users.
- Nonfunctional software requirements: To express some of the "attributes of the system" or "attributes of the system environment" such as usability, reliability, performance and supportability
- Design constraints: restrictions on the design of a system, or the process by which a system is developed, that do not affect the external behavior of the system but that must be fulfilled to meet technical, business, or contractual obligations.

Types of Requirements



Key Points

- A complete set of requirements can be determined by defining the inputs, outputs, functions, and attributes of the system plus the attributes of the system environment.
- Requirements should exclude project-related information, such as schedules, project plans, budgets, and tests, as well as design information.
- The requirements/design process is iterative; requirements lead to the selection of certain design options, which in turn may initiate new requirements.
- Design constraints are restrictions on the design of the system or on the process by which a system is developed.