
# Req Elicitation - Scenario

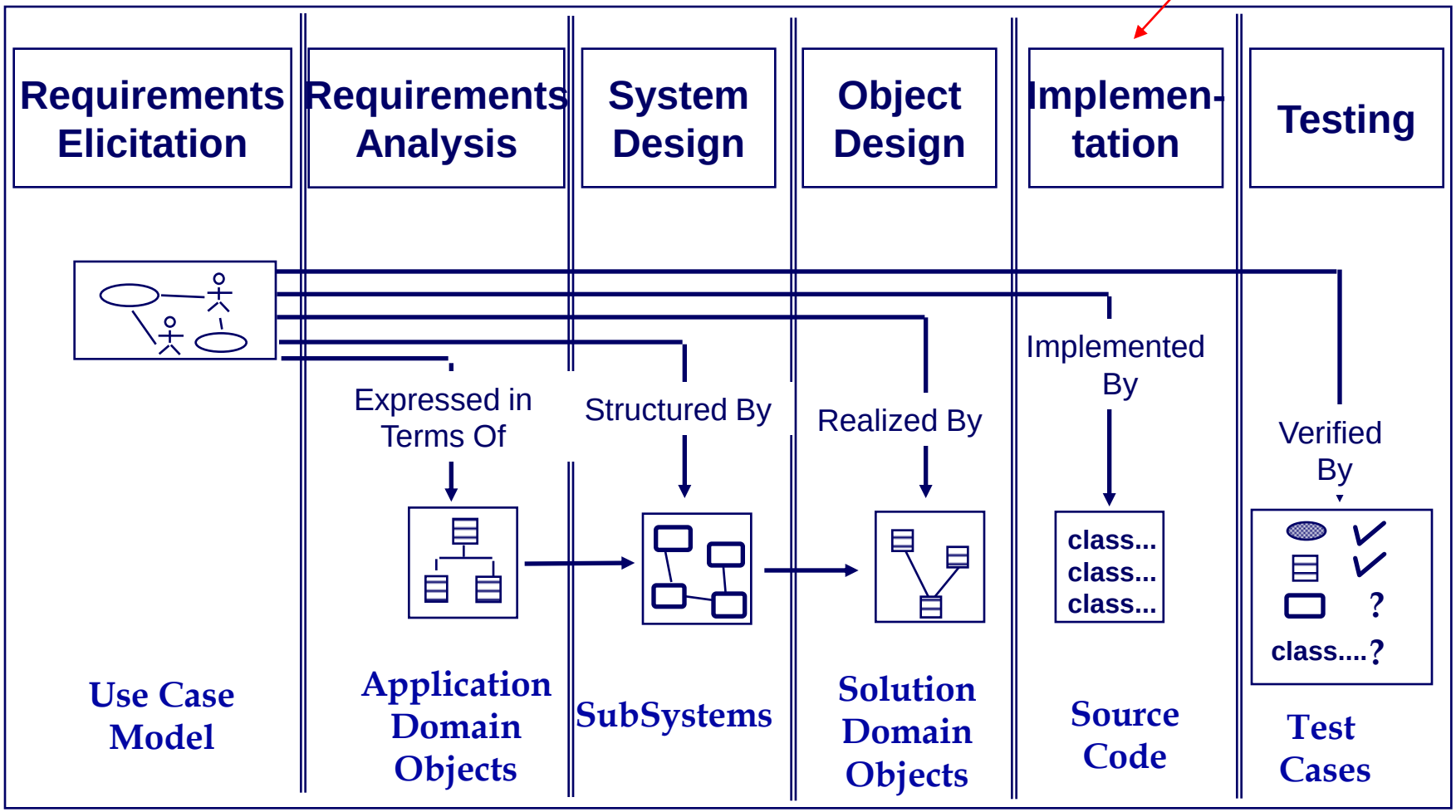
**Munawar, PhD**

# Software Lifecycle Activities

Deliverable 0

| Deliverable 1 | Deliverable 2 | Deliverable 3 | Deliverable 4 | Deliverable 5 | Deliverable 6 |
|---|---|---|---|---|---|

The Hacker knows only one activitity




Each activity produces one or more models

# Elicitation TEchnique

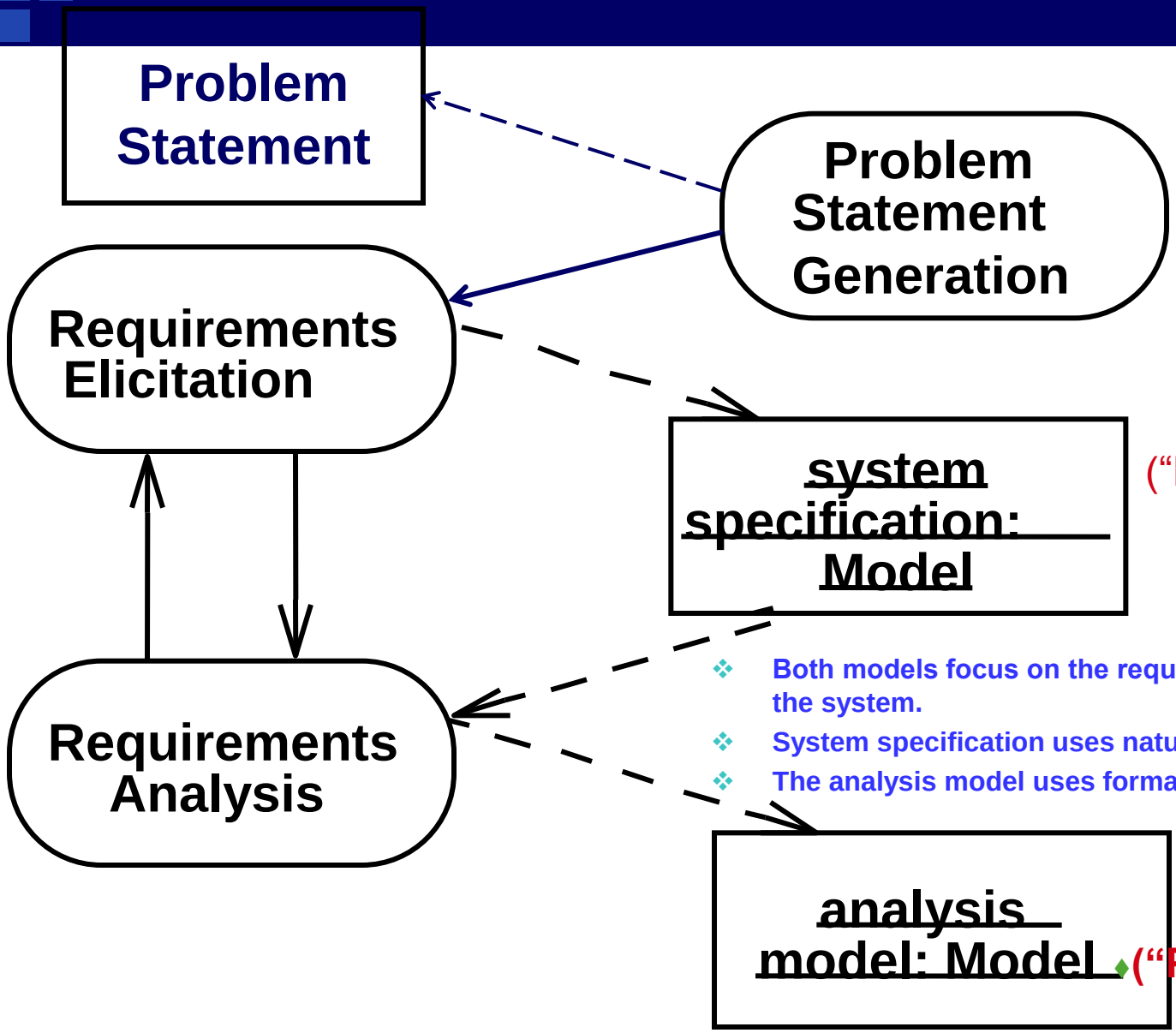
- **Analysis of Existing Systems**
  - Documentation, Observation, and Ethnography
- **Interviews**
- **Brainstorming**
- **Joint Application Design (JAD)**
- **Prototyping**
- **Use Cases**

- **When people talk, listen completely. Most people never listen.**[1]

# First Step in Establishing the Requirements: System Identification *In Brugge's methodology*

- **The development of a system is not just done by taking a snapshot of a scene (domain)**
- **Two questions need to be answered:**
  - How can we identify the *purpose* of a system?
  - Crucial is the definition of the *system boundary*: What is inside, what is outside the system?
- **The requirements process consists of two activities:**
  - Requirements Elicitation:
    - Definition of the system in terms understood by the *customer* ("Problem Description")
  - Requirements Analysis:
    - *Technical* specification of the system in terms understood by the *developer* ("Problem Specification")

# Brugge's Products of Requirements Process




- Both models focus on the requirements from the user's view of the system.
- System specification uses natural language
- The analysis model uses formal or semi-formal notation

# Requirements Elicitation

- **Very challenging activity**
- **Requires collaboration of people with different backgrounds**
  - Users with application domain knowledge
  - Developer with solution domain knowledge (design knowledge, implementation knowledge)
- **Bridging the gap between user and developer:**
  - *Scenarios:* Example of the use of the system in terms of a series of interactions with between the user and the system
  - *Use cases:* Abstraction that describes a class of scenarios

# Ingredients of a Problem Statement

- **Current situation: The Problem to be solved**
- **Description of one or more scenarios**
- **Requirements**
  - Functional and Nonfunctional requirements
  - Constraints (“pseudo requirements”)
- **Project Schedule**
  - Major  milestones that involve interaction with the client including deadline for delivery of the system
- **Target environment**
  - The environment in which the delivered system has to perform a specified set of system tests
- **Client Acceptance Criteria**
  - Criteria for the system tests

# Requirements Validation

- **Requirements validation is a critical step in the development process, usually after requirements engineering or requirements analysis. Also at delivery (client acceptance test).**
- **Requirements validation criteria:**
  - Correctness:
    - The requirements represent the client's view.
  - Completeness:
    - All possible scenarios, in which the system can be used, are described, including exceptional behavior by the user or the system
  - Consistency:
    - There are functional or nonfunctional requirements that contradict each other
  - Realism:
    - Requirements can be implemented and delivered
  - Traceability:
    - Each system function can be traced to a corresponding set of functional requirements

# Scenarios

- **"A narrative description of what people do and experience as they try to make use of computer systems and applications" [M. Carrol, Scenario-based Design, Wiley, 1995]**
- **A concrete, focused, informal description of a single feature of the system used by a single actor.**
- **Scenarios can have many different uses during the software lifecycle**
  - *Requirements Elicitation*: As-is scenario, visionary scenario
  - *Client Acceptance Test:* Evaluation scenario
  - *System Deployment:* Training scenario.

# Types of Scenarios

- **As-is scenario:**
  - Used in describing a current situation. Usually used in re-engineering projects. The user describes the system.
- **Visionary scenario:**
  - Used to describe a future system. Usually used in greenfield engineering and reengineering projects.
  - Can often not be done by the user or developer alone
- **Evaluation scenario:**
  - User tasks against which the system is to be evaluated..
- **Training scenario:**
  - Step by step instructions that guide a novice user through a system

# How do we find scenarios?

- **Don’t expect the client to be verbal if the system does not exist (greenfield engineering)**
- **Don’t wait for information even if the system exists**
- **Engage in a *dialectic* approach (evolutionary, incremental engineering)**
  - You help the client to formulate the requirements
  - The client helps you to understand the requirements
  - The requirements evolve while the scenarios are being developed

# Heuristics for finding Scenarios

- **Ask yourself or the client the following questions:**
  - What are the primary tasks that the system needs to perform?
  - What data will the actor create, store, change, remove or add in the system?
  - What external changes does the system need to know about?
  - What changes or events will the actor of the system need to be informed about?
- **However, don't rely on *questionnaires* alone.**
- **Insist on *task observation* if the system already exists (interface engineering or reengineering)**
  - Ask to speak to the end user, not just to the software contractor
  - Expect resistance and try to overcome it

# Lab Activities

- Do brainstorming in your group
- Define the scenario for your project (descriptive or narative)


# Thank You !

**Munawar, PhD**