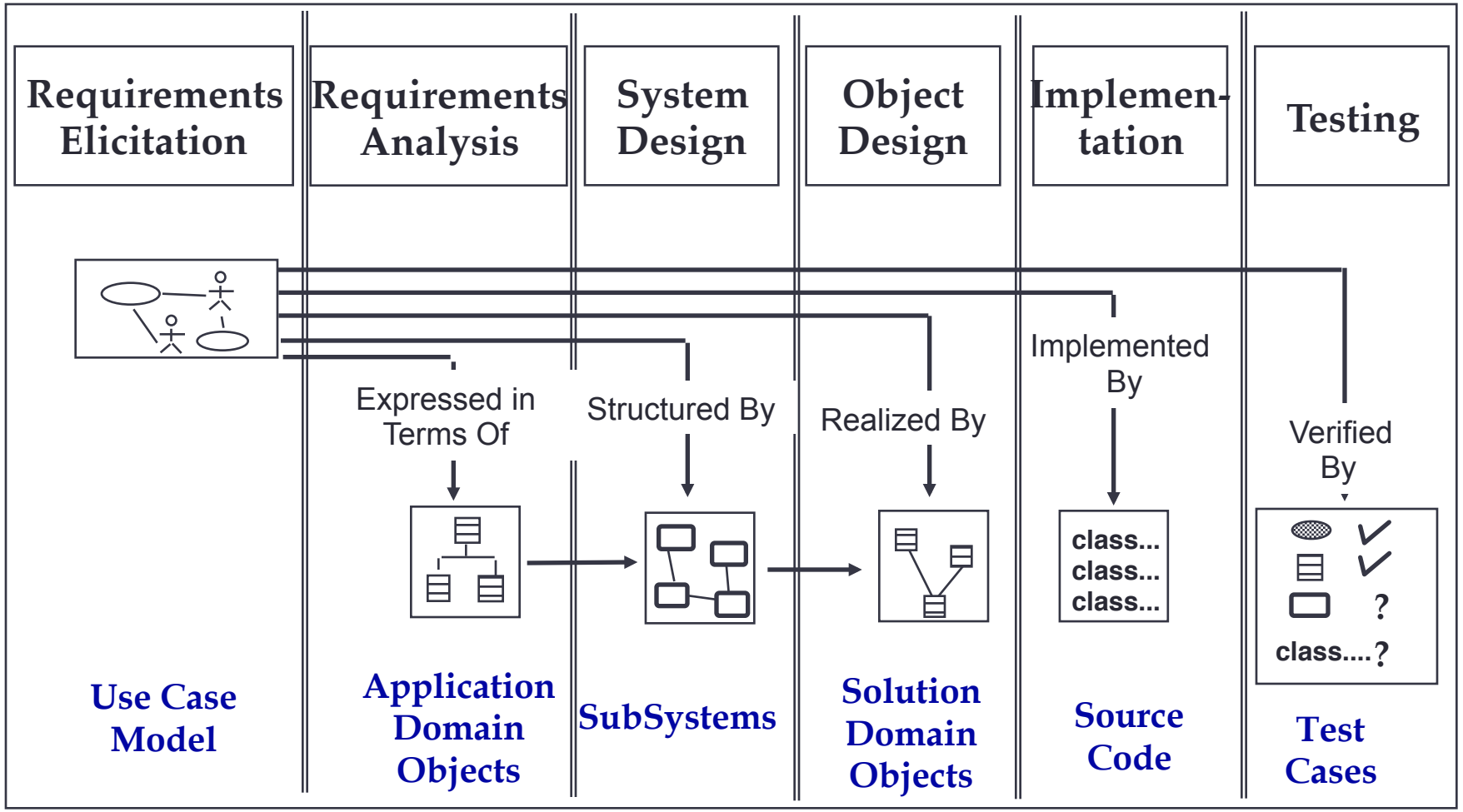
SOFTWARE ENGINEERING: REQUIREMENTS ENGINEERING

Professors M. Brian Blake and Iman Saleh

Software Lifecycle Activities



Requirements Elicitation

- Very challenging activity
- Requires collaboration of people with different backgrounds
  - **Users** with application domain knowledge
  - **Developers** 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

Types of Requirements

Functional requirements
Describe the interactions between the system and its environment independent from implementation

Example:
A Facebook user should be able to add photos

Non-Functional requirements
User visible aspects of the system not directly related to functional behavior

Example:
Facebook's main page should completely load in less than 15 seconds.

Constraints ("Pseudo requirements")
Imposed by the client or the environment in which the system operates

Example:
Facebook should allow AJAX-based applications that extend existing functions.

What is usually **not** in the requirements?

- System structure, implementation technology
- Development methodology
- Development environment
- Implementation language
- Reusability

- It is desirable that none of these above are constrained by the client. Fight for it!

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**

Requirements Validation

Real-Life Requirement Documents

http://www.nd.edu/~soseg/requirement_software

Requirements Validation

- Problem with requirements validation: Requirements change very fast during requirements elicitation.
- Tool support for managing requirements:
  - Store requirements in a shared repository
  - Provide multi-user access
  - Automatically create a system specification document from the repository
  - Allow change management
  - Provide traceability throughout the project lifecycle

- GatherSpace

DEMONSTRATION

Types of Requirements Elicitation

- Greenfield Engineering
  - Development starts from scratch, no prior system exists, the requirements are extracted from the end users and the client
  - Triggered by user needs
  - **Example:** Develop a game from scratch: Asteroids

- Re-engineering
  - Re-design and/or re-implementation of an existing system using newer technology
  - Triggered by technology enabler

- Interface Engineering
  - Provide the services of an existing system in a new environment
  - Triggered by technology enabler or new market needs

SCENARIOS

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.

- This is one iteration through a process so there should be no need for branching or looping.

Scenarios: An Example

- System: ATM Machine
- Requirement: User should be able to use an ATM to print an account balance
- Scenario

1. Alice uses an ATM
2. She enters her card and PIN into the ATM
3. She requests an account balance
4. Alice receives a printed account balance, takes out her bank card and leaves

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

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

Next goal, after the scenarios are formulated:

- Find all the use cases in the scenario that specifies all possible instances of how to report a fire

- Describe each of these use cases in more detail
  - Participating actors
  - Describe the Entry Condition
  - Describe the Flow of Events
  - Describe the Exit Condition
  - Describe Exceptions
  - Describe Special Requirements (Constraints, Nonfunctional Requirements)

Exercise:Scenario Development

- What are scenarios for the Frisbee?
  - Create scenarios for the first project

Any Questions?

iman@miami.edu