Software Engineering

Professor M. Brian Blake

Requirements Engineering
(Problem Statements and Use Cases)

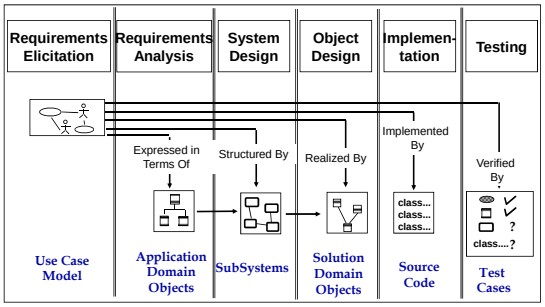
Where are we right now?

- Three ways to deal with complexity:
  - Abstraction
  - Decomposition (Technique: Divide and conquer)
  - Hierarchy (Technique: Layering)
- Two ways to deal with decomposition:
  - Object-orientation and functional decomposition
  - Functional decomposition leads to unmaintainable code
  - Depending on the purpose of the system, different objects can be found
- What is the right way?
  - Start with a description of the functionality (Use case model). Then proceed by finding objects (object model).
- What activities and models are needed?
  - This leads us to the software lifecycle we use in this class

Software Lifecycle Definition

- Software lifecycle:
  - Set of **activities** and their relationships to each other to support the development of a software system
- Typical Lifecycle questions:
  - Which activities should I select for the software project?
  - What are the dependencies between activities?
  - How should I schedule the activities?
  - What is the result of an activity

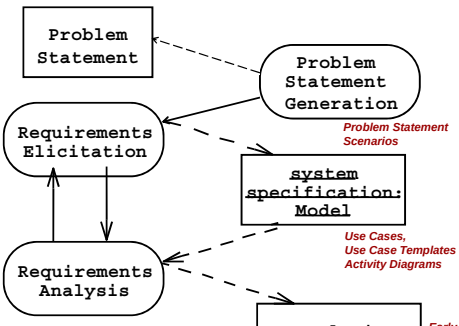
Software Lifecycle Activities



First Step in Establishing the Requirements: System Identification

- 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?
- These two questions are answered in the requirements process
- 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")

Products of Requirements Process



(Activity Diagram)

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

System Specification vs Analysis Model

- Both models focus on the requirements from the user's view of the system.
- ***System specification*** uses natural language (derived from the *problem statement*)
- The ***analysis model*** uses formal or semi-formal notation (for example, a graphical language like UML)
- The starting point is the problem statement

Problem Statements

Problem Statement

- A Problem Statement is probably the simplest concept in software engineering.
- Just a written narrative that tells how the domain/system is supposed to initiate and execute.
- Generally the nouns of this statement become the objects of the system
- Write as though you are explaining the system to a designer

Problem Statement

- The problem statement is developed by the client as a description of the problem addressed by the system
- Other words for problem statement:
  - Statement of Work
- A good problem statement describes
  - The current situation
  - The functionality the new system should support
  - The environment in which the system will be deployed
  - Deliverables expected by the client
  - Delivery dates
  - A set of acceptance criteria

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

Current Situation: The Problem To Be Solved

- There is a problem in the current situation
  - Examples:
    - The response time when playing letter-chess is far too slow.
    - I want to play, but cannot find players on my level.
- What has changed? Why can you address the problem now?
  - There has been a change, either in the application domain or in the solution domain
  - ***Change in the application domain***
    - A new function (business process) is introduced into the business
    - Example: We can play highly interactive games with remote people
  - ***Change in the solution domain***
    - A new solution (technology enabler) has appeared
    - Example: The internet allows the creation of virtual communities.

What's wrong with the T2V Problem Statement?

- Mobile customers are irritated when they have to ignore phone calls when attending important meetings that they cannot leave. Although they cannot leave the meeting, they are still able respond just not verbally. You are developing software additions to the existing mobile infrastructure that allows a person to answer a call in text messaging mode. The user will receive voice from the caller in the form of text and will be able to respond by typing on the phone and the responses will be spoken back to the caller. The mobile user may configure the phone with common phrases, in advance. The mobile user will be able specify a type of voice from several options (e.g. female, male, robotic). The mobile user should have a specific module on his/her phone that handles this mode. Instead of just answer and ignore buttons, this module will present a third button that will allow the user to answer in T2V mode.

Exercise: Simple Problem Statement

- Write a problem statement for :
  - A Futuristic Frisbie
  - Futuristic Shoes
  - Futuristic Refrigerator

Task List

- Go over Assignment 1
- Finalize teams for Final Projects
  - Go over Final Project Deliverables again
- Lecture on Requirements
  - Demonstrate Requisite Pro
- Lecture on Scenarios
  - Example for the T2V Problems statement
  - Student Group Work in Class on Frisbee, Shoes, Refrigerator

Defining Requirements

Types of Requirements

- Functional requirements:
  - Describe the interactions between the system and its environment independent from implementation
  - Examples:
    - A Facebook user should be able to add photos
- Nonfunctional requirements:
  - User visible aspects of the system not directly related to functional behavior.
  - Examples:
    - The response time must be less than 1 second
    - 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
    - The implementation language must be Java
    - Facebook should allow AJAX-based applications that extend existing functions.