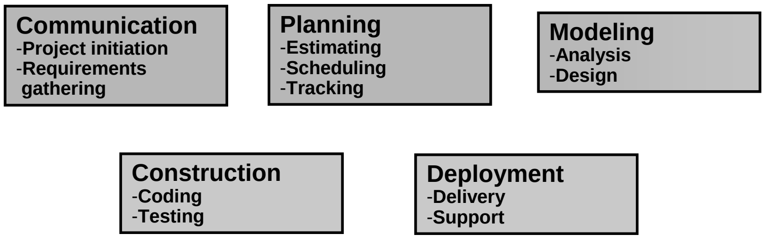
Software Engineering I CSC-382

Lecture 4

Software Engineering I CS-382

- Lecture 4
- What we will cover: (Requirements Engineering)
  - Chapter 7, Sections 7.1-7.5 and In Chapter 8, Section 8.5 (Use Cases) in Pressman
  - Goal of the next 2 lectures is to understand the critical tasks associated with deriving the software requirements of a system

Location of Requirements Engineering in the Process



Communication
-Project initiation
-Requirements gathering

Planning
-Estimating
-Scheduling
-Tracking

Modeling
-Analysis
-Design

Construction
-Coding
-Testing

Deployment
-Delivery
-Support

- **Requirements engineering continues to address:**
  - **The context of the software within the overall system**
  - **The customer/user needs**
  - **The prioritization of those needs**

What Types of Requirements Must be Defined in Requirements Engineering

1. Functional Requirements
   - These define the things the product must <u>do</u>**.**
     - These are what people always think of as requirements

2. Non-Functional Requirements
   - Define <u>properties</u> or <u>qualities</u> the software must have

3. Constraints
   - These define global requirements within which the product must operate

Functional Requirements

- Define what the product must **do**
  - Defines the scope of the product (its boundaries with the external world)
  - Include:
    - Actions the product must take to satisfy its basic need for existence
    - Transforms (functions) required to generate the data the product must produce from the inputs it is given
    - Derived from the fundamental purpose of the product
    - Not a quality of the system

Non-functional Requirements

- Define the **qualities** of the product
  - Look and feel requirements
  - Usability requirements (intended types of users)
  - Performance requirements (how fast, etc.)
  - Operational requirements (operating environment)
  - Maintainability requirements
  - Portability requirements
  - Security requirements
  - Cultural and political requirements
  - Legal requirements

Constraint Requirements

- Define restrictions or limitations of the product
  - Purpose of the product
  - The client, customers, stakeholders
  - Users
  - Requirements constraints (project and design constraints)
  - Naming conventions and standards
  - Relevant facts
  - Assumptions

Seven Steps of Requirements Engineering

1. **Inception**—ask a set of questions that establish …
   - Basic understanding of the problem
   - The people who want a solution
   - The nature of the solution that is desired, and
   - The effectiveness of preliminary communication and collaboration between the customer and the developer

Seven Steps of Requirements Engineering II

2. **Elicitation**—elicit requirements from all stakeholders
3. **Elaboration**—create an analysis model that identifies data, function and behavioral requirements
4. **Negotiation**—agree on a deliverable system that is realistic for developers and customers

Seven Steps of Requirements Engineering III

5. **Specification**—can be any one (or more) of the following:
   - A written document
   - A set of models
   - A formal mathematical description
   - A collection of user scenarios (use-cases)
   - A prototype

Seven Steps of Requirements Engineering IV

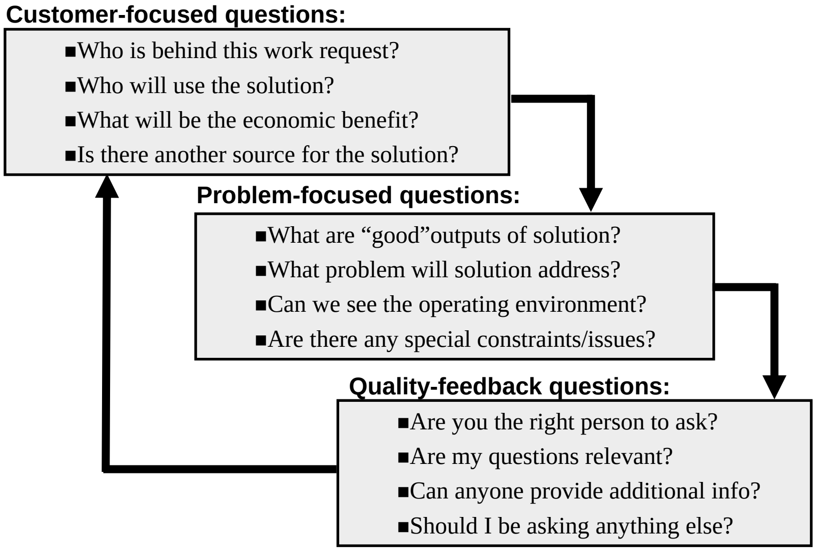
6. **Validation**—a review mechanism that looks for:
   - errors in content or interpretation
   - areas where clarification may be required
   - missing information
   - inconsistencies (a major problem when large products or systems are engineered)
   - conflicting or unrealistic (unachievable) requirements.

7. **Requirements management**

1. Inception

- Identify stakeholders
  - “who should we talk to?”
- Recognize multiple points of view
- Work toward collaboration
- Begin the requirements gathering with questions to “break the ice” and get the whole team working together

1. Inception



Customer-focused questions:

- Who is behind this work request?
- Who will use the solution?
- What will be the economic benefit?
- Is there another source for the solution?

Problem-focused questions:

- What are “good”outputs of solution?
- What problem will solution address?
- Can we see the operating environment?
- Are there any special constraints/issues?

Quality-feedback questions:

- Are you the right person to ask?
- Are my questions relevant?
- Can anyone provide additional info?
- Should I be asking anything else?

2. Elicitation

- The goal is
  - to identify the problem
  - propose elements of the solution
  - define different possible approaches, and
  - specify a preliminary set of solution requirements

2. Elicitation II

- Elicitation may be accomplished by:
  - Meetings are conducted and attended by both software engineers and customers
    - A "definition mechanism" is used (can be work sheets, flip charts, or wall stickers or an electronic bulletin board, etc)
    - These are more formal than the inception meetings
  - Use of specific tools/techniques to speed the process

Some Potential Tools for Eliciting Requirements

- General Engineering Methods
  - Quality Function Deployment (QFD), TRIZ, etc.

- Structured Analysis-based Techniques
  - Use the **context diagram** for system and the software
    - Defines the boundaries of the system
    - Context helps to define the data and control flow within system

Some Potential Tools for Eliciting Requirements II

- Object Oriented Analysis-based Techniques
  - Scenario-based with **use cases** to define key actors of system
    - Actors then help to define the classes
    - Things they do specify the functional requirements

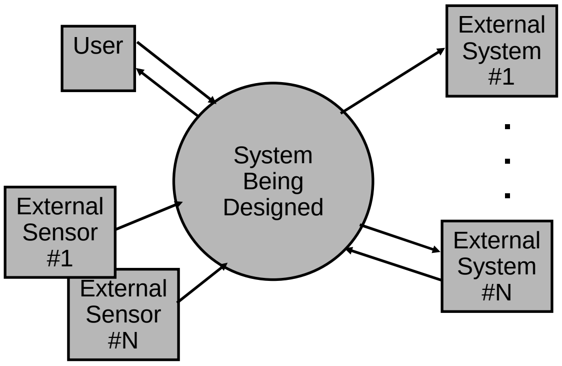
Context Diagrams for Eliciting Requirements

- Defining the boundaries of the system is critical to beginning the requirements analysis process.
  - In the Context Diagram the system is represented as a circle in the center of the diagram
  - All the external users, systems, sources/sinks of data, etc. are then represented by boxes surrounding the system
  - The Inputs/Outputs are defined at a very abstract level

Context Diagrams for Eliciting Requirements II

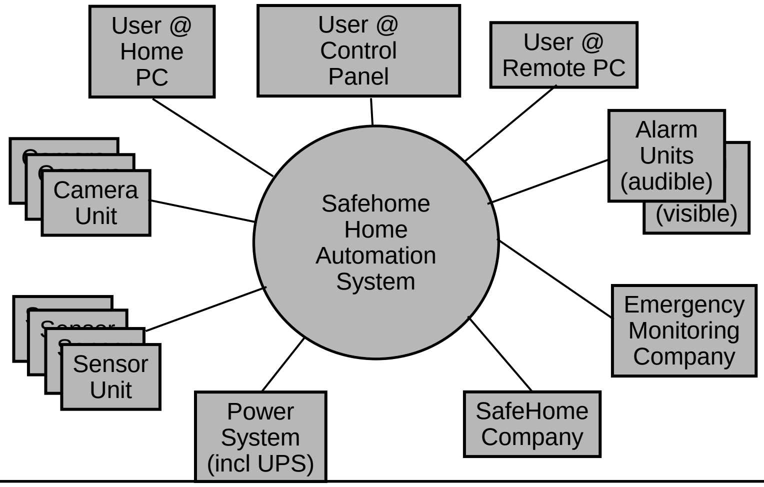
- Useful for subsequent requirements analysis and modeling for both OO and Structured methods
  - Actors for OO use cases are more easily defined
  - I/Os for Structured method also more visible

Format of the Context Diagram



In-class Context Diagram

Recall Safehome effort from last week:



Another In-class Context Diagram

For Next Class

- Continue to Study Chapter 7 Sections 7.6-7.9 in Pressman (Requirements Engineering)