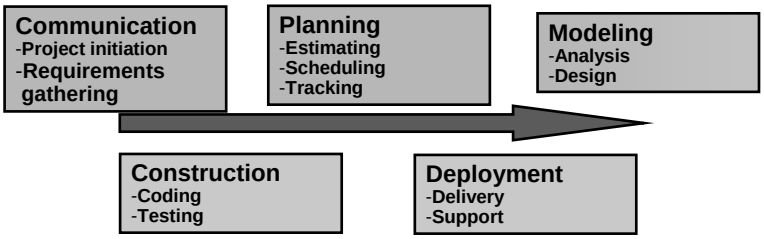
Software Engineering I CS-382

- Lecture 7
- What we will cover: (Continue Requirements Engineering)
  - Chapter 7 Sections 7.6-7.9 in Pressman
  - Goal is to understand the critical tasks associated with deriving the software requirements of a system

Location of Requirements Engineering in the Process



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

Recall the First of the 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

Recall the Next Two of the Seven Steps of Requirements Engineering

2. **Elicitation**—elicit requirements from all stakeholders
3. **Elaboration**—create an analysis model that identifies data, function and behavioral requirements
   - This process continues thru the Modeling stage
   - We keep adding detail (recall the old Stepwise refinement in 175)

Context Diagram & Use Cases

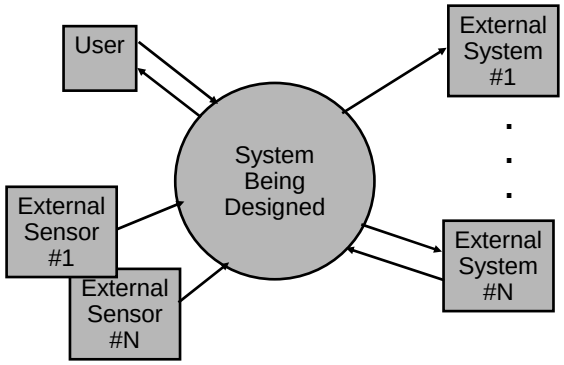
Recall the Context Diagrams for Eliciting Requirements

- 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

Recall the 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

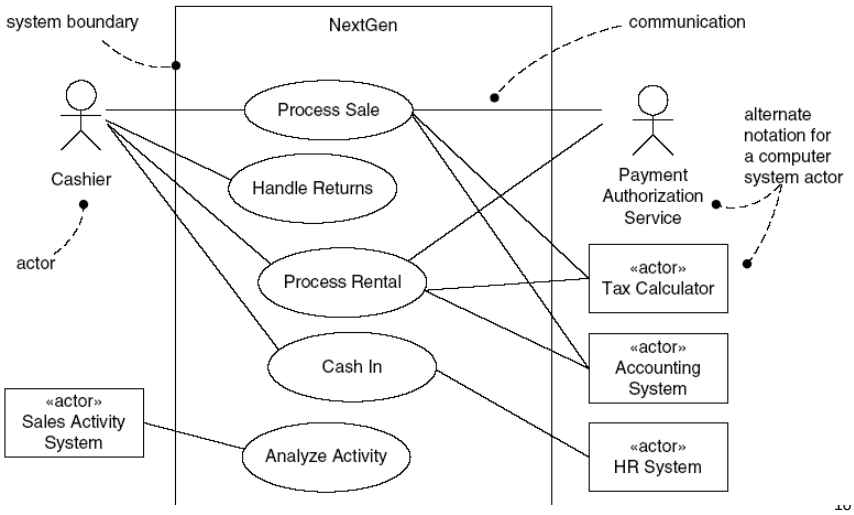
Recall the Format of the Context Diagram



Recall Use-Case Based Elicitation

- **Actors** represent roles people or devices play as the system functions
- Three types of actors:
  1. Users
  2. Administrators
  3. External Programs and Devices
- Actors have these two common features:
  1. External to the application
  2. They take initiative, stimulate and interact with our system

Example System Use Case Diagram



Recall the Simple Template for Use-Cases

Use Case Name:

Primary Actor:

Description of Usage:

Elicitation Work Products

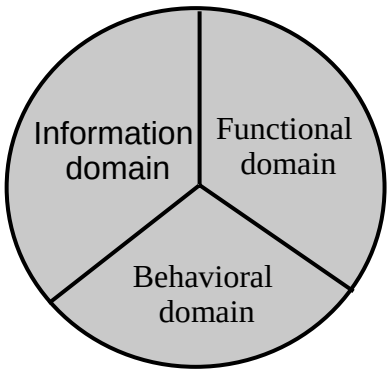
- A statement of need and feasibility.
- A bounded statement of scope for the system or product.
- A list of customers, users, and other stakeholders who participated in requirements elicitation
- A description of the system's technical environment.

Elicitation Work Products II

- A list of requirements (preferably organized by function) and the domain constraints that apply to each.
- A set of usage scenarios that provide insight into the use of the system or product under different operating conditions.
- Any prototypes developed to better define requirements**.**

Requirements Elaboration (Step 3)

- The goal of elaboration is to develop the analysis models defined in Lecture 2:



- These three areas of the analysis model can be modeled by:
  - Scenario modeling (or functional modeling instead)
  - Data/Class modeling
  - Behavior modeling
  - Flow-oriented modeling (data and/or control)

Some Potential Tools for Requirements Elaboration

- **Data Modeling Methods**
  - Define the Entity-Relationship Diagrams of the data in the system
  - Used mostly in database design (can be considered a special case of Class modeling where there are no operations).
- **Structured Analysis-based Techniques**
  - Use Data Flow Diagrams and State Transition Diagrams
  - Finish with process narratives (PSPECS) to define low level reqs.

Some Potential Tools for Requirements Elaboration II

- **Object Oriented Analysis-based Techniques**
  - Use Class definitions and hierarchies, State Transition Diagrams, and narrative specs for the methods
- **Prototyping**
  - Excellent method for clearly defining requirements of new to the world systems where the customer/user may not fully understand what they want.

Starting to Building the Analysis Model

- Elements of the analysis model
  - Scenario-based elements
  - Data/Class-based elements
  - Behavioral elements
  - Flow-oriented elements

Starting to Building the Analysis Model II

- Scenario-based elements
  - Functional—processing narratives for software functions
    - Called P-Specs in Structured Method
    - Called generic scenarios in OO methods (i.e. not user directed)
  - Use-case—descriptions of the interaction between an “actor” and the system
  - Discussed in great detail during elicitation phase so no need to discuss further

Starting to Building the Analysis Model III

- Data/Class-based elements
  - Implied by scenarios and context diagrams, ERDs, and DFDs
  - Book only mentions Class, but expanding definition to include data allows us to abstract this to other SW engineering methods
  - In Structured methods use a data dictionary to capture the data flows as they appear during flow analysis
  - In OO approach, the Class definitions capture this
    - We will show an effective approach for performing this function

Starting to Building the Analysis Model IV

- Behavioral elements
  - The basic tool is the State diagram (also called State Transition Diagram, State Model, etc.)
  - Used in both Structured and OO methods of analysis

Starting to Building the Analysis Model V (more on behavior)

- Basic idea:
  - A system at any point in time can be modeled as being in a particular **State**.
  - The system can **Transition** to another state at any time based on the **Events** (also called **Stimuli**) to the system
  - The system can perform some **Action** when it enters a particular state
  - The current state of the system is defined by the last state and the last event

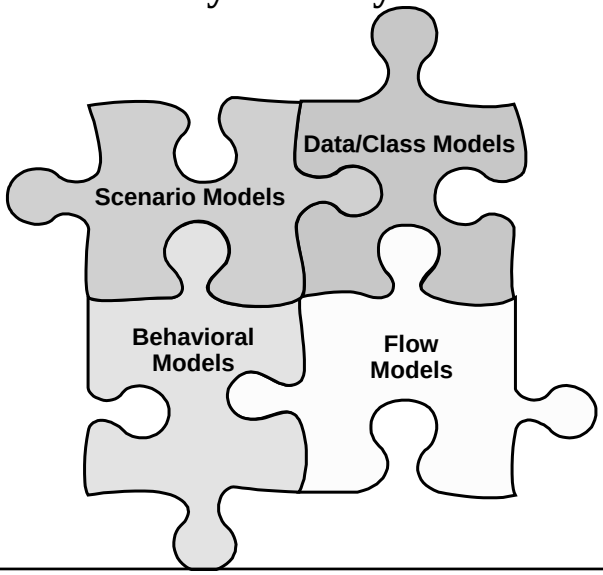
Starting to Building the Analysis Model VI

- Flow-oriented elements
  - Based on the core tool of Structured Analysis: the Data Flow Diagram
  - Elements of Data Flow Diagram
    - **Data Flow** – the inputs and outputs to each transform
    - **Data Transform** – the processing to be executed on input data
    - **Data Stores** – persistent data in the system
    - **External Entities** – Same as in Context Diagram

Starting to Building the Analysis Model VII

- Context diagram is the level ‘0’ Data Flow
- The data flows are generated in a top-down process
  - Stop when the process defined in a single process bubble is easily explained

Summary of Analysis Models



4. Negotiating Requirements

- Identify the key stakeholders
  - These are the people who will be involved in the negotiation
- Determine each of the stakeholders "win conditions"
  - Win conditions are not always obvious
- Negotiate
  - Work toward a set of requirements that lead to "win-win"

5. Specification

- This is the actual task of collecting the requirements generated through Elaboration and Negotiation into a document
  - Important for capturing the whole system for future team members
  - Writing this is the goal of your project!!

6. Validating Requirements

- Is each requirement consistent with the overall objective for the system/product?
- Have all requirements been specified at the proper level of abstraction? That is, do some requirements provide a level of technical detail that is inappropriate at this stage?
- Is the requirement really necessary or does it represent a non-essential feature? (“gold plating”)

6. Validating Requirements-II

- Is each requirement bounded and unambiguous?
- Is each requirement traceable? Is there a clear source for each requirement?
- Do any requirements conflict with other requirements?
- Is each requirement achievable in the technical environment that will house the system or product?

6. Validating Requirements-III

- Is each requirement testable, once implemented?
- Does the requirements model properly reflect the information, function and behavior of the system to be built.
- Has the requirements model been “partitioned” in a way that exposes progressively more detailed information about the system.

7. Managing Requirements

- Recall we noted that requirements development is an iterative process
  - This implies there will be changes to requirements over time
  - These changes must be properly managed to ensure a quality final product
- Label each requirement after initial requirements generation
  - Now as additional requirements are added, or the initial requirements are further refined and expanded, the identifiers follow the requirement
  - Enter these into a traceability table

7. Managing Requirements II

- Traceability Tables can be based on:
  - Features traceability
  - Source traceability
  - Subsystem traceability
  - Interface traceability
  - Etc.

For Next Class

- Begin to Study Chapter 8 (Analysis Modeling)