Software Engineering I CSC-382

W08 Midterm Review

What is a System

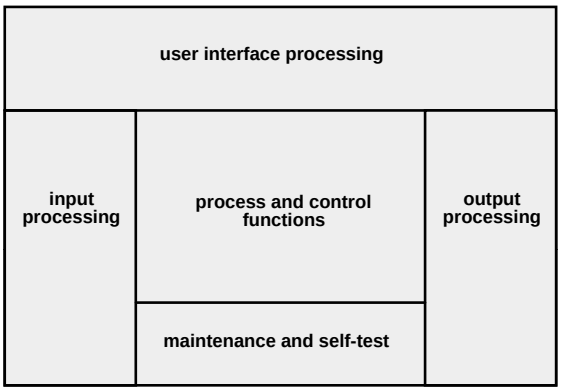
- Elements of a computer-based system
  - **Software**
  - **Hardware**
  - **People**
  - **Database**
  - **Documentation**
  - **Procedures**

What is a System II

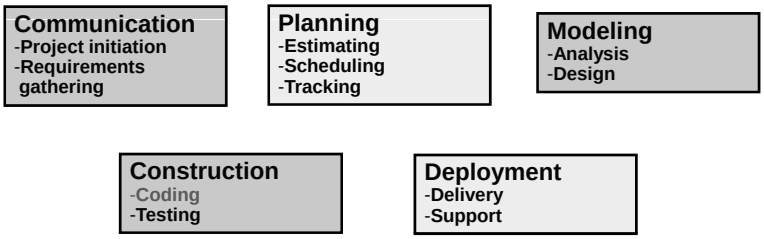
- First goal is to define what the system is and is not.
  - **Define the context in which the system resides**
- Second goal is to define the hardware software partition
  - Then we know what we need to design as software engineers

System Modeling Template

Hatley-Pirbai Model



The Five Critical Stages in Software Engineering



Elements in yellow are covered in CS 383. CS 383 also covers discusses the variety of ways these core stages can be sequenced, managed, and documented.
Note: Coding is not covered here (already well covered in CS 175 & 275)
Coding is typically only 10% of a software engineering effort !!

Communication – Generic Task Set

1. Identify primary customer
2. Meet with and discuss business/end-user issues
3. Develop project scope statement
4. Review and modify project scope statement
   - Sometimes called Operational Concept Description (OCD)
5. Get more detailed:
   - Define customer usage scenarios, system I/O, major functions/features
6. Document data from #5
7. Iterate upon data from #5
8. Prioritize data from #5
9. Review the results with all stakeholders

Planning – Generic Task Set

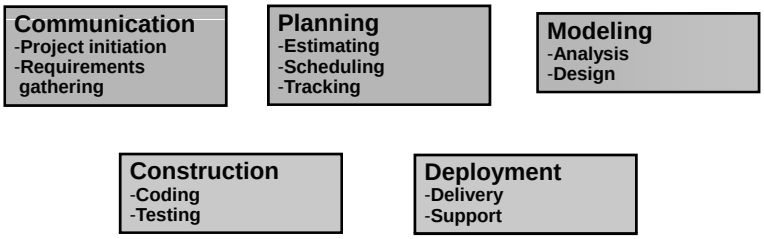
1. Re-assess project scope
2. *Assess risks*
3. Develop/Modify Usage Scenarios
4. Derive functions/features
5. Consider infrastructure functions/features
6. Prioritize features
7. *Create a coarse granularity plan*
8. *Create fine granularity plan for current increment*
9. *Track progress*

***** *CS-383 topics*

Analysis Modeling – Generic Task Set

1. Review requirements from planning stage
2. Expand/refine usage scenarios
3. Model the information domain
4. Model the functional domain
5. Model the behavioral domain
6. Analyze and model the user interface
7. Review all models for completeness, consistency, and **correctness** (missed in book – most subtle and critical errors. E.g “That’s nice but I wanted a Bud Light”)

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

Seven Steps of Requirements Engineering

1. Inception (Scope Statement)
2. Elicitation (Context Diagram &Use Cases)
3. Elaboration (DFD or Class Definitions)
4. Negotiation
5. Specification
6. Validation
7. Requirements management

Tools for Eliciting Requirements

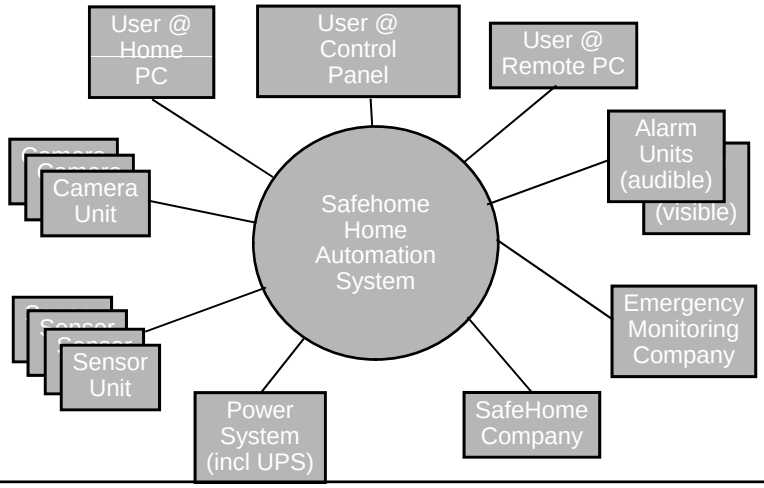
- Structured Analysis-based Techniques
  - Use the **context diagram** for system and the software
    - Defines the boundaries of the system
    - Context helps to define the data and control flow within system
- Object Oriented Analysis-based Techniques
  - Scenario-based with **use cases** to define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

In-class Context Diagram

Recall Safehome effort from last week:



Use-Case Based Elicitation

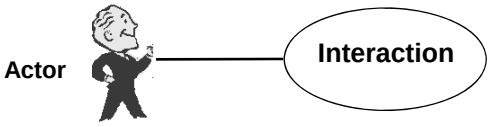
"[Use-cases] are simply an aid to defining what exists outside the system (actors) and what should be performed by the system (use-cases)." Ivar Jacobson

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

Format of the Use-Case

•Basic format of the Use Case Diagram:

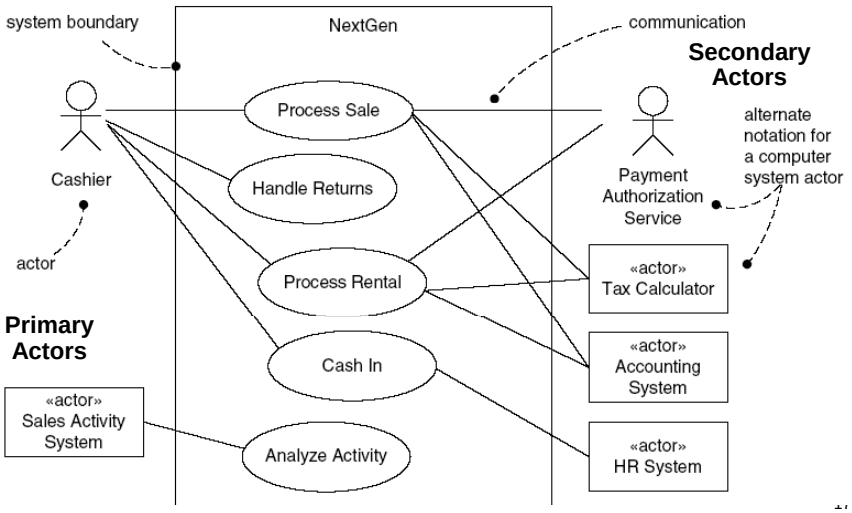


•The Use Case diagram defines the Cases to define in more detail:

•Define using the Use Case descriptions:

Use Case Name:	**Primary Actor:**
Data:	
Stimulus:	
Response:	
Comments:	

Example System Use Case Diagram



Even Simpler Template for Use-Cases

Use Case Name:

Primary Actor:

Description of Usage:

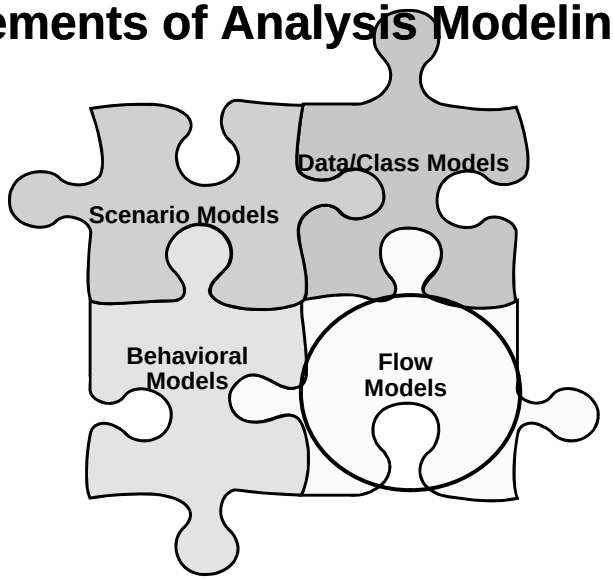
Tools We Will be Using for Analysis Modeling

- **Structured Analysis-based Techniques**
  - Use Data Flow Diagrams for the Flow Models and State Transition Diagrams for the Behavior Models
  - Finish with process narratives (PSPECS) to define low level requirements.
- **Object Oriented Analysis-based Techniques**
  - Derive the details of Class Models (using CRC Cards) and hierarchies (Class Diagrams)

Analysis Modeling via Structured Analysis

- **Structured Analysis** views a system as a sequence of ***transformations*** operating on the input data and leading to the final outputs
  - Represents how data objects are transformed at they move through the system

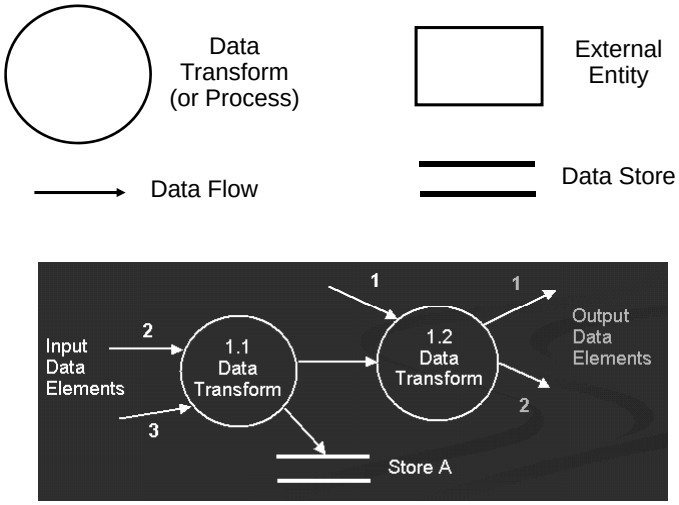
Elements of Analysis Modeling



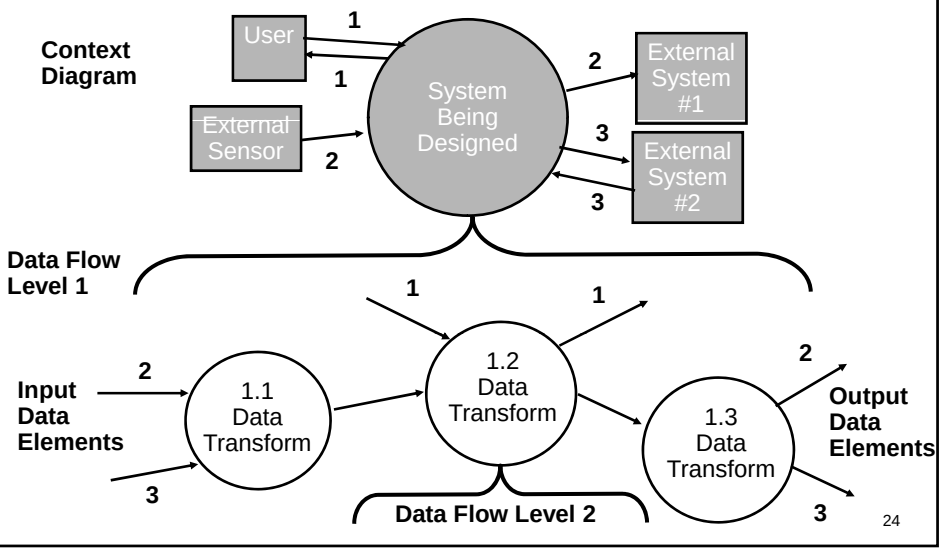
Flow Modeling with the Data Flow Diagram

- Elements of Data Flow Model
  - **Input Data** – the inputs to the current transformation
  - **Data Transformation** – the processing to be executed on input data
  - **Output Data** – the outputs from the current transformation
- 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
  - Then use Process Specifications (P-SPECS) to define the process

The Elements of the Data Flow Diagram

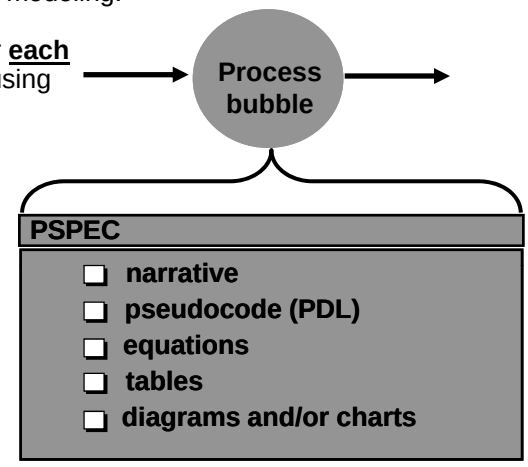


Data Flow Diagram Process

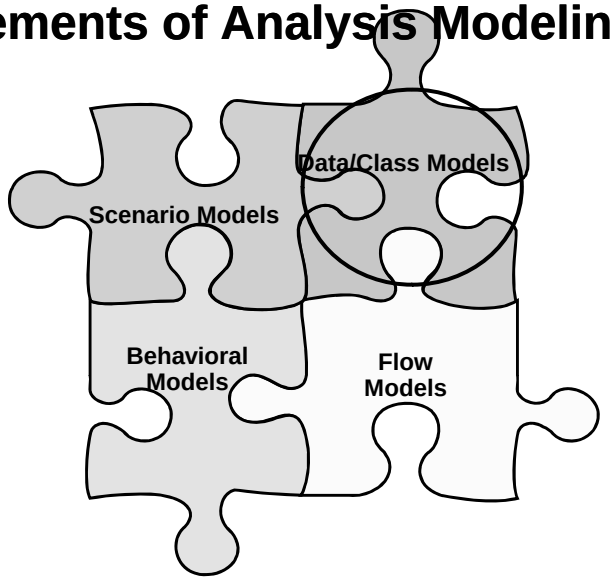


Process Specification (PSPEC)

- The **last** stage in the Data Flow modeling.
  - The required processing for **each** process is defined in detail using the PSPEC.



Elements of Analysis Modeling



Class-Based Modeling Thru OOA

- Object Oriented Methods view a system as a collection of these objects that communicate to each other thru messages
  - These messages request the various other objects to perform some function or task

More on Objects

- "An object represents an individual, identifiable item, unit, or entity, either real or abstract with a well defined role in the problem." – Wilkinson
  - Objects (and their classes) should represent tangible or visible things, roles, events, or concepts within the system.
  - **An object is more than just data and functions bound together**
  - Each object has a set of essential and unique static attributes
  - The ***state*** of an object is the values of these at any time.

Search Strategies for Objects

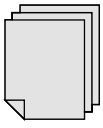
- Good Object candidates often represent:
  - The work the system performs
  - Things directly affected by or connected to the application
  - Information that flows thru the software
  - Decision making, control, and coordination activities
  - Structures and groups of other lower level objects
  - Representations of real-world things the system needs to know something about

A Simple Set of Stereotypes

Actor Classes:
•People
•Organizations

Interface Classes:
•Screens
•Menus

Business Classes:
•Places
•Things
•Concepts
•Events



Report Classes:
•Printed
•Electronic

Introducing the CRC Card

- CRC stands for: Class, Responsibility and Collaborators
  - **Class**: An object is a person, place, thing, event, or concept
  - **Responsibility**: Anything that a class knows or does
  - **Collaborator**: A class that another class needs to accomplish its purpose
    - In general, a collaboration implies either a request for information or a request for some action.

Elements of CRC Modeling

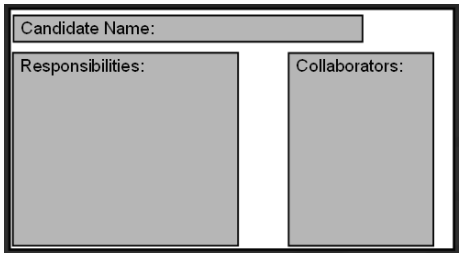
- Note the focus of CRC modeling is on the ***behavior*** of the objects and not the specific structure of the objects
  - The attributes will be clearly defined and reflected in the "***know responsibilities***" of the objects to ***know*** certain things as defined in the CRC cards.
  - The methods will be reflected in the "***does responsibilities"*** that we identify in the CRC card

Collaborations

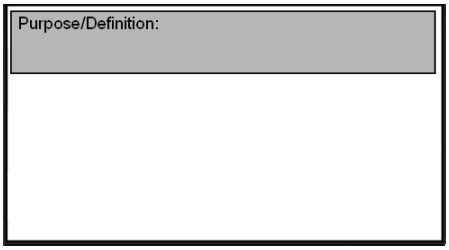
- Classes fulfill their responsibilities in one of two ways:
  1. A class can use its own operations to manipulate its own attributes, thereby fulfilling a particular responsibility, or
  2. A class can collaborate with other classes.
- Collaborations identify relationships between classes
- Collaborations are identified by determining whether a class can fulfill each responsibility itself

More Common Structure of CRC Card

Front of Card:



Back of Card:



This is often (and originally) used as the CRC
I prefer it since it is simpler and focuses on the main info