Software Engineering
Dr. Raj Singh
Requirements Analysis

• specifies software’s operational characteristics
• indicates software’s interface with other system elements
• establishes constraints that software must meet

Requirements analysis allows the software engineer to

• elaborate on basic requirements
• build models that depict user scenarios, functional activities, problem classes and their relationships, system and class behavior, and the flow of data as it is transformed
Elements of Requirements Analysis
**REQUIREMENTS MODELING**

**Scenario-based**  
System from the user’s point of view

**Data**  
Shows how data are transformed inside the system

**Class-oriented**  
Defines objects, attributes, and relationships

**Flow-oriented**  
Shows how data are transformed inside the system

**Behavioral**  
Show the impact of events on the system states
The model should focus on requirements that are visible within the problem or business domain. The level of abstraction should be relatively high.

Each element of the analysis model should add to an overall understanding of software requirements and provide insight into the information domain, function and behavior of the system.

Delay consideration of infrastructure and other non-functional models until design.

Minimize coupling throughout the system.

Be certain that the analysis model provides value to all stakeholders.

Keep the model as simple as it can be.
Define the domain to be investigated.

Collect a representative sample of applications in the domain.

Analyze each application in the sample.

Develop an analysis model for the objects.
SCENARIO-BASED METHODS
**Use case / scenario** defines how a user uses a system to accomplish a particular goal.

A modeling technique that defines the features to be implemented and the resolution of any errors that may be encountered.

A methodology used in system analysis to identify, clarify, and organize system requirements.

A set of possible sequences of interactions between systems and users in a particular environment and related to a particular goal.
What are the main tasks or functions that are performed by the actor?

What system information will the actor acquire, produce or change?

Will the actor have to inform the system about changes in the external environment?

What information does the actor desire from the system?

Does the actor wish to be informed about unexpected changes?
Use-cases are written first in narrative form and mapped to a template if formality is needed. Each primary scenario should be reviewed and refined to see if alternative interactions are possible.

- Can the actor take some other action at this point?
- Is it possible that the actor will encounter an error condition at some point? If so, what?
- Is it possible that the actor will encounter some other behavior at some point? If so, what?
SafeHome

- Access camera surveillance via the Internet
- Configure SafeHome system parameters
- Set alarm

Homeowner

Cameras
Describe situations that may cause the system to exhibit unusual behavior

Brainstorm to derive a reasonably complete set of exceptions for each use case

Are there cases where a validation function occurs for the use case?

Handling exceptions may require the creation of additional use cases
Supplements the use case by providing a graphical representation

The flow of interaction between actor and the system within a specific scenario
Allows the modeler to represent the flow of activities described by the use-case

Indicates which actor or analysis class has responsibility for the action described by an activity rectangle
CLASS-BASED METHODS
One view of requirements modeling, called structured analysis, considers data and the processes that transform the data as separate entities. Data objects are modeled in a way that defines their attributes and relationships. Processes that manipulate data objects are modeled in a manner that shows how they transform data as data objects flow through the system.

A second approach to analysis modeled, called object-oriented analysis, focuses on the definition of classes and the manner in which they collaborate with one another to effect customer requirements.
Class-based modeling represents objects that the system will manipulate operations (also called methods or services) relationships between the objects collaborations that occur between the classes

The elements of a class-based model include classes and objects attributes and operations collaboration diagrams and packages
### IDENTIFYING CLASSES

- **Classes are determined by underlining each noun or noun phrase and entering it into a simple table**

- If the class is required to implement a solution, then it is part of the solution space;

- If a class is necessary only to describe a solution, it is part of the problem space.
### External entities
- (e.g. other systems, devices, people) that produce or consume information

### Things
- (e.g. reports, displays, letters, signals) that are part of the information domain for the problem

### Occurrences or events
- (e.g. completion of actions) that occur within the context of system operation

### Roles
- (e.g. manager, engineer, salesperson) played by people who interact with the system

### Organizational units
- (e.g. division, group, team) that are relevant to an application

### Places
- (e.g. manufacturing floor) that establish the context of the problem and the overall function

### Structures
- (e.g. sensors, computers) that define a class of objects or related classes of objects
Attributes describe a class that has been selected for inclusion in the analysis model.

Attributes describe the structure and value of an instance of a class.
An operation is a method or function that can be performed by a class.

Operations define the behavior of a class, what a class can do.

Operations can perform computation, take an action, call another method, etc.
**CLASS TYPES**

**Entity classes**
also called model or business classes, are extracted directly from the statement of the problem

**Boundary classes**
are used to create the interface that the user sees and interacts with the software

**Controller classes**
manage a “unit of work” from start to finish
System intelligence should be distributed across classes to best address the needs of the problem.

Each responsibility should be stated as generally as possible.

Information and the behavior related to it should reside within the same class.

Information about one thing should be localized with a single class, not distributed across multiple classes.

Responsibilities should be shared among related classes, when appropriate.
COLLABORATIONS

Classes fulfill their responsibilities in one of two ways:
- a class can use its own operations to fulfill a particular responsibility
- 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
ASSOCIATION, AGGREGATION AND COMPOSITION

Association is a (*a*) relationship between two classes, where one class use another

Aggregation, a special type of an association, is the (*the*) relationship between two classes.

Composition can be recognized as a special type of an aggregation.

Aggregation is a special kind of an association and composition is a special kind of an aggregation.

Association → Aggregation → Composition

Association is non-directional, aggregation insists a direction.
COMPOSITE AGGREGATE CLASS
WallSegment is used to build Window

Wall

1

Door

0..*

is used to build

1..*

MULTIPLICITY
The behavioral model indicates how software will respond to external events.

Evaluate all use-cases to fully understand the sequence of interaction within the system.

Identify events that drive the interaction sequence and understand how these events relate to specific objects.

Create a sequence for each use-case.

Build a state diagram for the system.

Review the behavioral model to verify accuracy and consistency.
In the context of behavioral modeling, two different characterizations of states must be considered:

- the state of each class as the system performs its function and
- the state of the system as observed from the outside as the system performs its function

The state of a class takes on both passive and active characteristics:

- A passive state is simply the current status of all of an object's attributes.
- The active state of an object indicates the current status of the object as it undergoes a continuing transformation or processing.
THE STATES OF A SYSTEM

State
a set of observable circumstances that characterizes the behavior of a system at a given time

State transition
the movement from one state to another

Event
an occurrence that causes the system to exhibit some predictable form of behavior

Action
process that occurs as a consequence of making a transition
reading

key hit

password entered

comparing

do: validatePassword

password = incorrect & numberOfTries ≤ maxTries

numberOfTries > maxTries

locked

timer < lockedTime

timer > lockedTime

password = correct

selecting

activation successful
Figure 8.27 Sequence diagram (partial) for SafeHome security function
FLOW MODELS
A flowchart to represent the flow from one activity to another activity.

The activity can be described as an operation of the system.
A data-flow diagram is a way of representing a flow of data of a process or a system.

The DFD also provides information about the output and input of each entity and the process itself.

A data-flow diagram has no control flow, there are no decision rules and no loops.
REFERENCE