

GANPAT UNIVERSITY
U. V. PATEL COLLEGE OF ENGINEERING

2CEIT502

SOFTWARE ENGINEERING

UNIT 3

BUILDING THE ANALYSIS MODEL

Prepared by: Prof. Ravi Raval (Asst. Prof in C.E Dept. , UVPCE)

Unit 3: Building the analysis Models

Contents:

- ❑ Requirement Analysis
- ❑ Analysis Modeling Approaches
- ❑ Data Modeling Concepts
- ❑ Object oriented Analysis
- ❑ Scenario based modeling
- ❑ Class based Modeling
- ❑ Creating a behavioral model
- ❑ Flow Oriented Modeling
- ❑ Entity Relationship diagram (E-R)

Unit 3: Building the Analysis Model

Prelude:

- Understanding the requirement is **most difficult task**

Why is that so?

- When you first think about it, developing a clear understanding of requirements doesn't seem that hard.
- After all, doesn't the customer know what is required?
- Shouldn't the end users have a good

understanding of the features and functions that will provide benefit?

- Surprisingly, in many instances the answer to these questions is “**no.**” And even if customers and end-users are explicit in their needs, those **needs will change throughout the project.**

Unit 3: Building the Analysis Model

Requirement Engineering:

□ What is it?

Before you begin any technical work, it's a good idea to apply a set of requirements engineering tasks. These tasks lead to an understanding of what the business impact of the software will be, what the customer wants, and how end users will interact with the software.

□ Who does it?

Software engineers (sometimes referred to as system engineers or “analysts” in the IT world) and other project stakeholders (managers, customers, end users) all participate in requirements engineering.

Unit 3: Building the Analysis Model

Requirement Engineering:

□ Why it is important?

Designing and building an elegant computer program that solves the wrong problem serves no one's needs. That's why it's important to understand what the customer wants before you begin to design and build a computer-based system.

□ What are the steps?

Requirement engineering begins with:

- Inception
- Elicitation
- Elaboration
- Specification

Unit 3: Building the Analysis Model

Requirement Engineering:

□ What is the work product/outcome?

A written understanding of a problem.

- usage scenarios
- functions and features lists
- requirements models, or a specification.

□ How do I ensure that I've done it right?

Requirements engineering work products are reviewed with stakeholders to ensure that what you have learned is what they really meant. A word of warning: even after all parties agree, things will change, and they will continue to change throughout the project

Unit 3: Building the Analysis Model

Requirement Engineering:

- Why to do requirement engineering?
They argue that things will become clear as they build, that project stakeholders will be able to understand need only after examining early iterations of the software, that things change so rapidly that any attempt to understand requirements in detail is a waste of time, that the bottom line is producing a working program and

all else is secondary

What makes these arguments seductive is that they contain elements of truth (true in case of small projects – less than a month). But each is flawed and can lead to a failed software project.

The broad spectrum of tasks and techniques that lead to an understanding of requirements is called **requirements engineering**.

Unit 3: Building the Analysis Model

Requirement Engineering:

- Requirements engineering builds a bridge to design and construction.

But where does the bridge originate?

- it begins at the feet of the project stakeholders (e.g., managers, customers, end users), where business need is defined, user scenarios are described, functions and features are

delineated, and project constraints are identified.

Unit 3: Building the Analysis Model

Requirement Analysis:

- Requirements analysis results in the specification of software's operational characteristics, indicates software's interface with other system elements, and establishes constraints that software must meet.
- Requirements analysis allows you (regardless of whether you're called a *software engineer*, an *analyst*, or a

modeler) to elaborate on basic requirements established during the inception, elicitation, and negotiation tasks that are part of requirements engineering

Unit 3: Building the Analysis Model

Requirement Analysis:

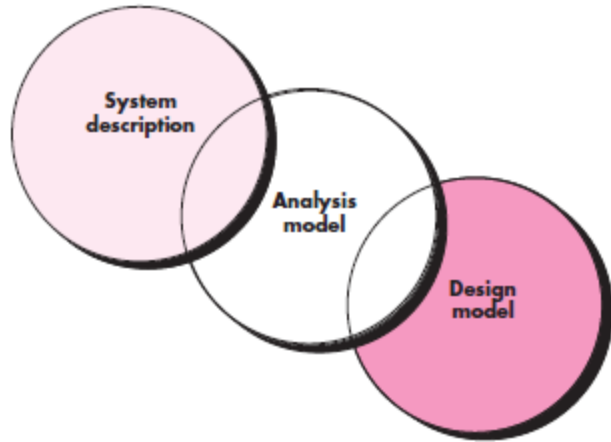
What outcomes will we achieve by following requirement analysis:

- ❑ **Scenario based Models:** from the point of view of various system actors
- ❑ **Data Models:** Information domain of a problem
- ❑ **Class oriented Models:** Object oriented classes (attributes and operation) and their collaboration to achieve system requirements
- ❑ **Flow Oriented Models:** how data flows through functional elements of system
- ❑ **Behavioral Models:** how the software behaves as a consequences of external “events”.

Unit 3: Building the Analysis Model

Requirement Analysis:

Overall Objectives and philosophy



The requirement model must achieve three primary objectives:

- (1) To describe what the customer requires
- (2) To establish a basis for the creation of a software design
- (3) To define a set of requirements that can be validated once the software is built

Unit 3: Building the Analysis Model

Analysis rules of Thumb:

- ❑ *The model should focus on requirements that are visible within the problem or business domain. The level of abstraction should be relatively high. “Don’t get bogged down in details” that try to explain how the system will work.*
- ❑ *Each element of the requirements 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 nonfunctional models until design.*
That is, a database may be required, but the classes necessary to implement it, the functions required to access it, and the behavior that will be exhibited as it is used should be considered only after problem domain analysis has been completed.

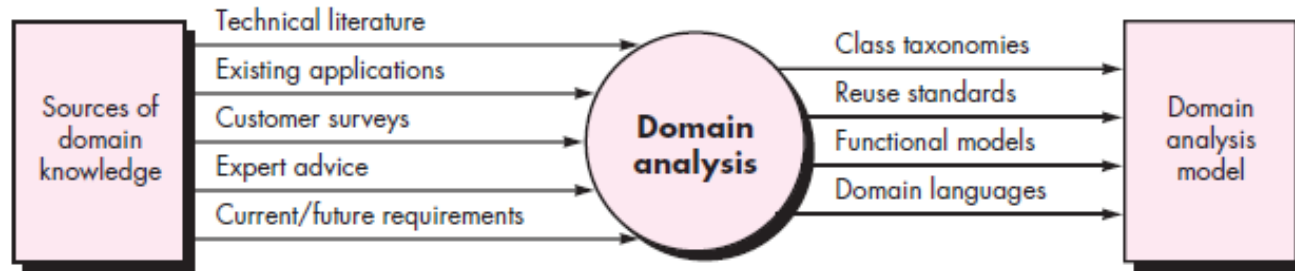
Unit 3: Building the Analysis Model

Analysis rules of Thumb (cont.):

- ❑ *Minimize coupling throughout the system.* It is important to represent relationships between classes and functions. However, if the level of “interconnectedness” is extremely high, effort should be made to reduce it.
- ❑ *Be certain that the requirements model provides value to all stakeholders.* Each constituency has its own use for the model. For example, business stakeholders should use the model to validate requirements; designers should use the model as a basis for design; QA people should use the model to help plan acceptance tests.
- ❑ *Keep the model as simple as it can be.* Don't create additional diagrams when they add no new information. Don't use complex notational forms, when a simple list will do.

Unit 3: Building the Analysis Model

Domain Analysis:



The analysis patterns often reoccur across many applications within a specific business domain. If these patterns are defined and categorized in a manner that allows you to recognize and apply them to solve common problems, the creation of the analysis model is expedited. More important, the likelihood of applying design patterns and executable software components grows dramatically. This improves time-to-market and reduces development costs.

Unit 3: Building the Analysis Model

Domain Analysis:

- How are analysis patterns and classes recognized in the first place?
- Who defines them, categorizes them, and readies them for use on subsequent projects?

The answers to these questions lie in *domain analysis*.

- Software domain analysis is the identification, analysis, and specification of common requirements from a **specific application domain**, typically for reuse on multiple projects within that application domain.... [Object-oriented domain analysis is] the identification, analysis, and specification of common, reusable capabilities within a specific application domain, in terms of common objects, classes, subassemblies, and frameworks.

Unit 3: Building the Analysis Model

Domain Analysis:

- ❑ The **Specific application domain** can range from avionics to banking, from multimedia video games to software embedded in medical devices.
- ❑ Domain analysis is an ongoing software engineering activity that is not connected to any one software project.
- ❑ In a way, the role of a domain analyst is similar to the role of a master toolsmith in a heavy manufacturing environment. The job of the toolsmith is to design and build tools that may be used by many people doing similar but not necessarily the same jobs.
- ❑ The role of the domain analyst is to discover and define analysis patterns, analysis classes, and related information that may be used by many people working on similar but not necessarily the same applications.

Unit 3: Building the Analysis Model

Requirement Analysis and modeling approaches:

- One view of requirements modeling, called *structured analysis*, considers data and the processes that transform the data as separate entities.

Data Objects	Processes
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

- The 2nd approach to analysis modeling, called **object-oriented analysis**, focuses on the definition of classes and the manner in which they collaborate with one another to effect customer requirements

Unit 3: Building the Analysis Model

Requirement Analysis and modeling approaches:

Scenario-based

elements depict how the user interacts with the system and the specific sequence of activities that occur as the software is used.

Scenario-based models
e.g.,
use cases
user stories

Behavioral elements

depict how external events change the state of the system or the classes that reside within it

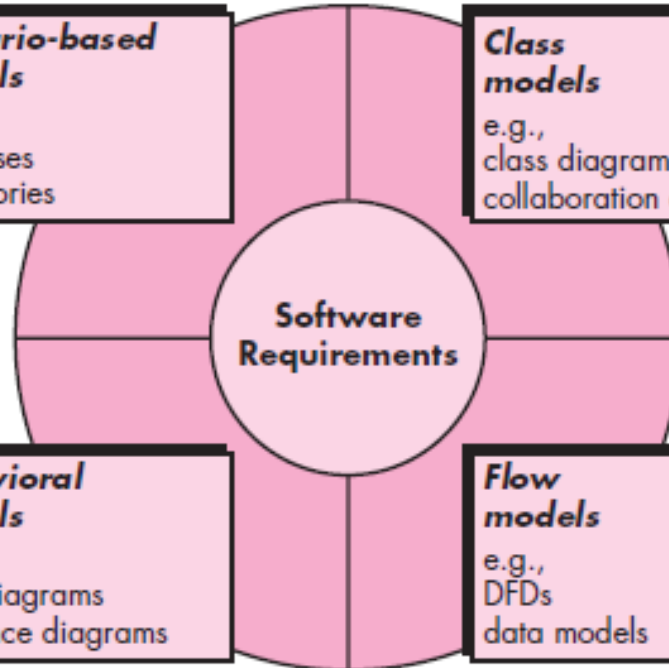
Behavioral models
e.g.,
state diagrams
sequence diagrams

Class models
e.g.,
class diagrams
collaboration diagrams

Class-based elements model the objects that the system will manipulate, the operations that will be applied to the objects to effect the manipulation, relationships (some hierarchical) between the objects, and the collaborations that occur between the classes that are defined

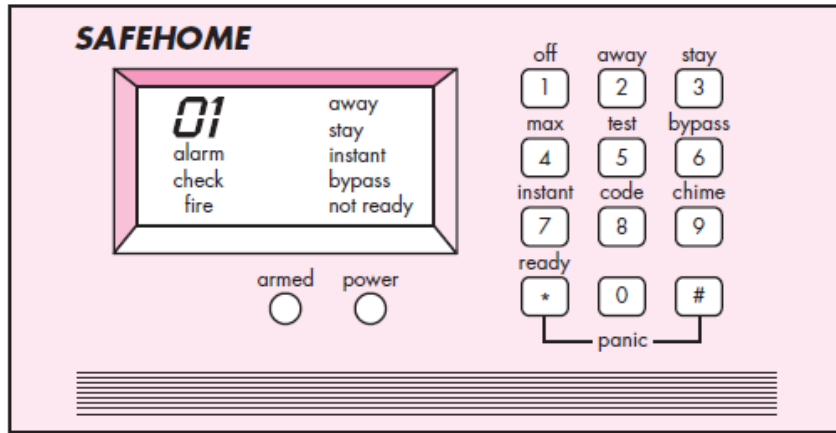
Flow models
e.g.,
DFDs
data models

Flow-oriented elements represent the system as an information transform, depicting how data objects are transformed as they flow through various system functions.

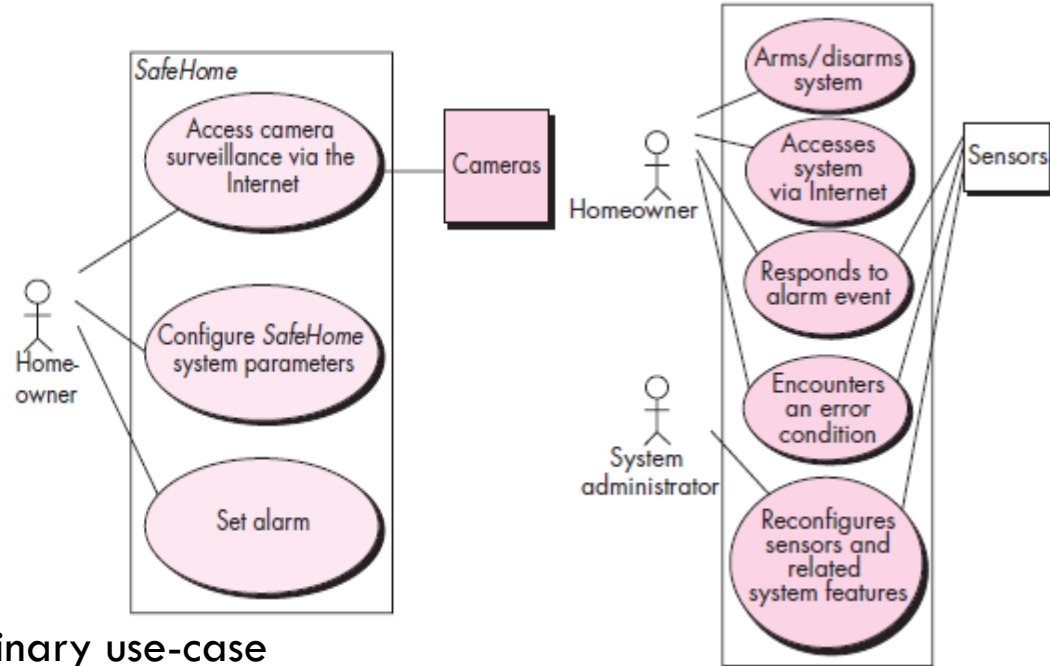


Unit 3: Building the Analysis Model

Scenario-based modeling:



SafeHome System Control Panel



Preliminary use-case
diagram for the
SafeHome system

UML use case diagram for *SafeHome*
home security function

Unit 3: Building the Analysis Model

Scenario-based modeling:

Use case template for Surveillance

- 1) **Use case:** Access camera surveillance via the Internet—display camera views (ACS-DCV)
- 2) **Iteration:** 2, last modification: January 14 by V. Raman.
- 3) **Primary actor:** Homeowner.
- 4) **Goal in context:** To view output of camera placed throughout the house from any remote location via the Internet.
- 5) **Preconditions:** System must be fully configured; appropriate user ID and passwords must be obtained.
- 6) **Trigger:** The homeowner decides to take a look inside the house while away.

Unit 3: Building the Analysis Model

Scenario-based modeling:

Use case template for Surveillance

7) Scenario

1. The homeowner logs onto the *SafeHome Products* website.
2. The homeowner enters his or her user ID.
3. The homeowner enters two passwords (each at least eight characters in length).
4. The system displays all major function buttons.
5. The homeowner selects the “surveillance” from the major function buttons.
6. The homeowner selects “pick a camera.”
7. The system displays the floor plan of the house.
8. The homeowner selects a camera icon from the floor plan.
9. The homeowner selects the “view” button.
10. The system displays a viewing window that is identified by the camera ID.
11. The system displays video output within the viewing window at one frame per second.

Unit 3: Building the Analysis Model

Scenario-based modeling:

Use case template for Surveillance

8) Exceptions:

1. ID or passwords are incorrect or not recognized - see use case **Validate ID and passwords**.
2. Surveillance function not configured for this system—system displays appropriate error message; see use case **Configure surveillance function**.
3. Homeowner selects “View thumbnail snapshots for all camera”—see use case **View thumbnail snapshots for all cameras**.
4. A floor plan is not available or has not been configured—display appropriate error message and see use case **Configure floor plan**.
5. An alarm condition is encountered—see use case **Alarm condition encountered**.

Unit 3: Building the Analysis Model

Scenario-based modeling:

Use case template for Surveillance

- 9) **Priority:** Moderate priority, to be implemented after basic functions.
- 10) **When available:** Third increment.
- 11) **Frequency of use:** Moderate frequency.
- 12) **Channel to actor:** Via PC-based browser and Internet connection.
- 13) **Secondary actors:** System administrator, cameras.
- 14) **Channels to secondary actors:**
 - 1. System administrator: PC-based system.
 - 2. Cameras: wireless connectivity.

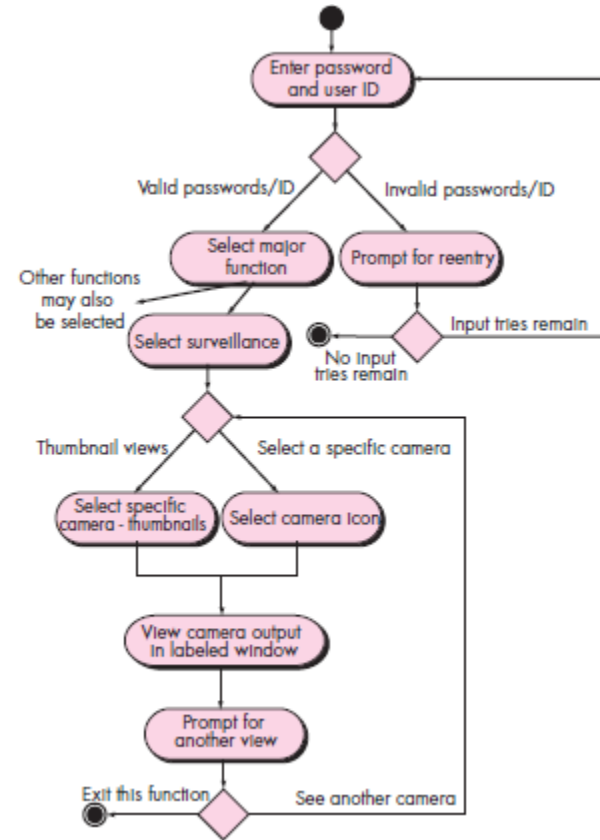
15) Open issues:

- 1. What mechanisms protect unauthorized use of this capability by employees of *SafeHome Products*?
- 2. Is security sufficient? Hacking into this feature would represent a major invasion of privacy.
- 3. Will system response via the Internet be acceptable given the bandwidth required for camera views?
- 4. Will we develop a capability to provide video at a higher frames-per-second rate when high bandwidth connections are available?

Unit 3: Building the Analysis Model

Scenario-based modeling:

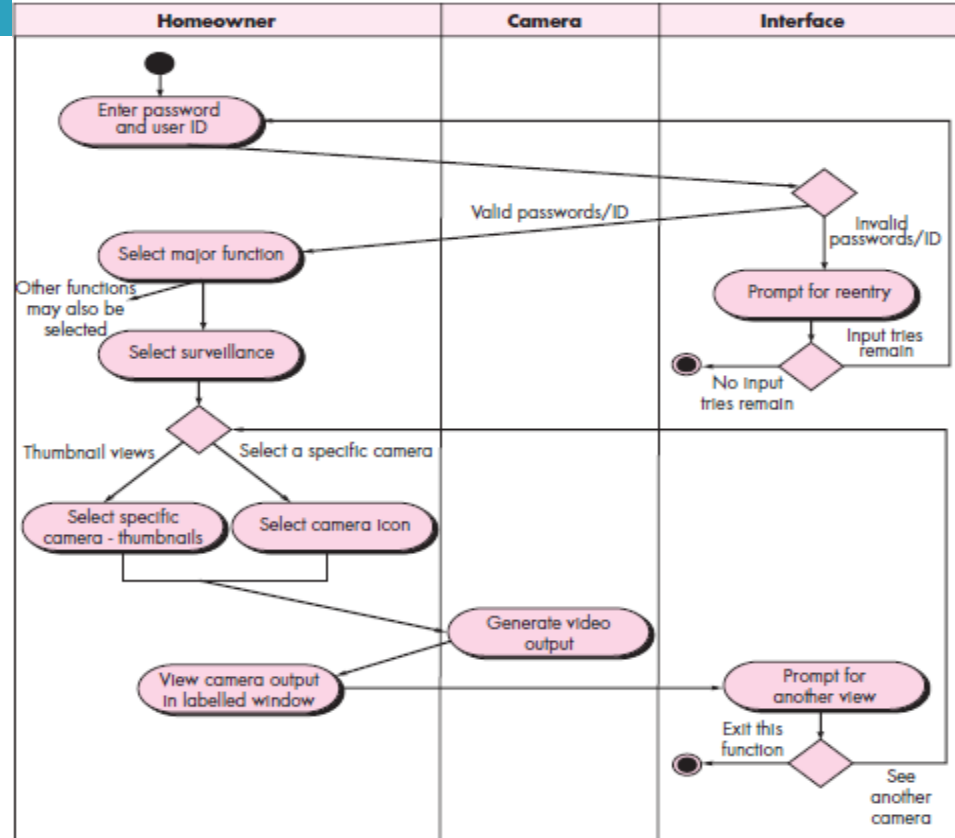
Activity diagram for access camera surveillance via the internet – display camera views function



Unit 3: Building the Analysis Model

Scenario-based modeling:

**Swimlane diagram for
access camera surveillance
via the internet – display
camera views function**



Unit 3: Building the Analysis Model

Class based modeling:

□ **Class:**

- Perform “grammatical parse” on use case developed, Identify noun or noun-phrase and prepare a table of it.⁴⁾
- Now classify the classes based on following list:⁵⁾
 - 1) **External entities:** Who produce/consume info. (e.g. device, people)⁶⁾
 - 2) **Things:** part of information domain (e.g. display, letters, signals)⁷⁾

3)

Occurrences or events: (e.g. robot movement)

Roles: people who'll interact with the system. (e.g. manager, engineer, sales person)

Organizational Units: (e.g. division, dept., group)

Places: (e.g. manufacturing floor, loading dock)

Structures: defines class of objects or related class of objects (e.g. sensors, computers, vehicles)

Unit 3: Building the Analysis Model

Class based modeling:

□ Example : SafeHome

Potential class	General Classification
Homeowner	Role or external entity
Sensor	External entity
Control Panel	External entity
Installation	Occurrence
System (alias security system)	Thing
Number, type	Not objects, attributes of sensor
Master password	Thing
Telephone number	Thing
Sensor event	Occurrence
Audible alarm	External entity
Monitoring service	Organization unit or external entity

Unit 3: Building the Analysis Model

Class based modeling:

- Once your table of classes is prepared, classification get done apply this six selection characteristics to either retain or reject the classes for your analysis model.

1. **Retained information.** The potential class will be useful during analysis only if information about it must be remembered so that the system can function.
2. **Needed services.** The potential class must have a set of identifiable operations that can change the value of its attributes in some way.
3. **Multiple attributes.** During requirement analysis, the focus should be on “major” information; a

class with a single attribute may, in fact, be useful during design, but is probably better represented as an attribute of another class during the analysis activity.

4. **Common attributes.** A set of attributes can be defined for the potential class and these attributes apply to all instances of the class.

5. **Common operations.** A set of operations can be defined for the potential class and these operations apply to all instances of the class.

6. **Essential requirements.** External entities that appear in the problem space and produce or consume information essential to the operation of any solution for the system will almost always be defined as classes in the requirements model.

Unit 3: Building the Analysis Model

Class based modeling:

□ Example : SafeHome

Potential class	Characteristics number that applies
Homeowner	Rejected: 1, 2 fail even though 6 applies
Sensor	Accepted: all apply
Control Panel	Accepted: all apply
Installation	Rejected
System (alias security system)	Accepted: all apply
Number, type	Rejected: 3 fails, attributes of sensor
Master password	Rejected: 3 fails
Telephone number	Rejected: 3 fails
Sensor event	Accepted: All apply
Audible alarm	Accepted: 2,3,4,5,6 apply
Monitoring service	Rejected: 1,2 fails even though 6 applies

Unit 3: Building the Analysis Model

Class based modeling:

□ **Specifying attributes:**

- Attributes of a baseball player in the context of statistics of professional baseball players: name, position, batting average fielding percentage, years played, games played etc.
- But for Professional baseball pension system, the attributes might be like: average, salary, credit toward full vesting, pension plan option chosen, mailing address etc.

□ How to select meaningful set of attributes?
What data items (composite and/or

elementary) fully define this class in the context of the problem at hand.

- identification information = system ID + verification phone number + system status
- alarm response information = delay time + telephone number
- Activation / deactivation information = master password + number of allowable tries + temporary password

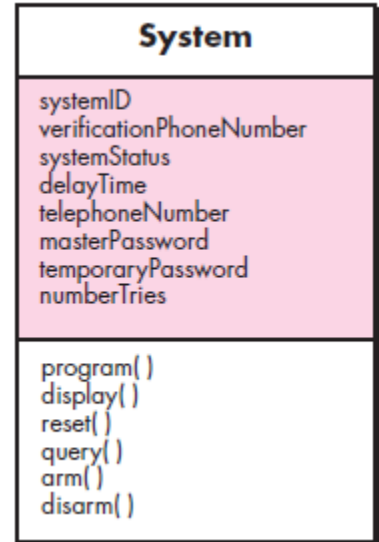
Unit 3: Building the Analysis Model

Class based modeling:

□ **Specifying Operations:** Operation (4) defines the behavior of an object. they can generally be divided into four broad categories:

- (1) operations that manipulate data in some way (e.g., adding, deleting, reformatting, selecting),
- (2) operations that perform a computation
- (3) operations that inquire about the

state of an object,
operations that monitor an object for the occurrence of a controlling event.






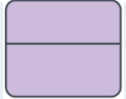




Unit 3: Building the Analysis Model

Flow oriented modeling:

- ❑ Some software considers data flow oriented techniques as outdated.
- ❑ Data-flow diagrams (DFD) are not formal part of UML. They complement UML diagrams and provide insight into requirements and flow.
- ❑ Input-Process-Output
- ❑ Context level (0-level) DFD represents the system as a whole
- ❑ Subsequent data flow diagrams refine the context diagram, providing increasing

detail with each subsequent level.

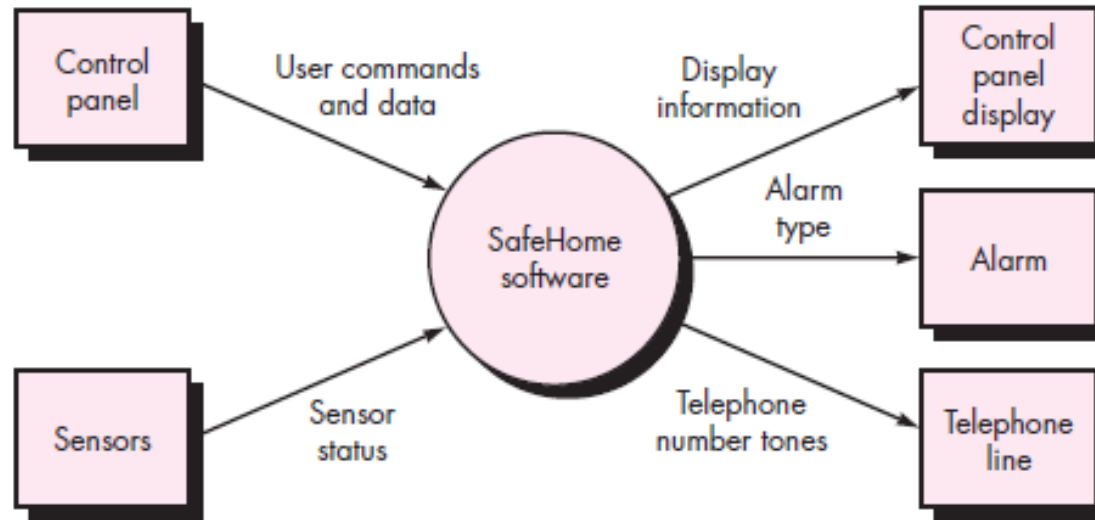
- ❑ Notations: Yourdon-coad, Gane-Sarson

	Yourdon and Coad	Gane and Sarson
External Entity		
Process		
Data-Store		
Dataflow		

Unit 3: Building the Analysis Model

Flow oriented modeling:

Context level (0-level) DFD for SafeHome System



Unit 3: Building the Analysis Model

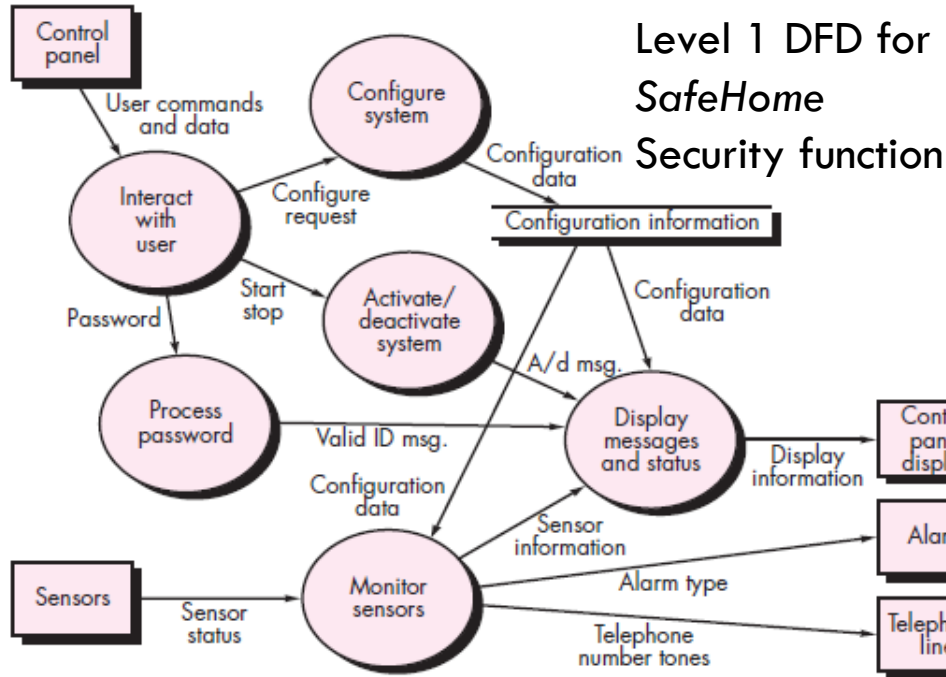
Flow oriented modeling:

Guidelines of derivation of a data flow diagrams

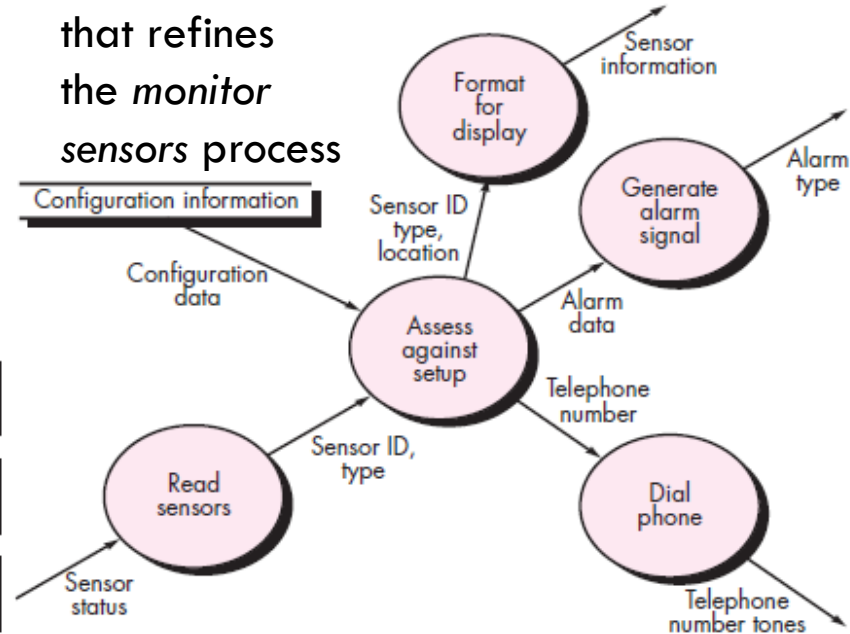
- (1) the level 0 data flow diagram should depict the software/system as a single bubble;
- (2) primary input and output should be carefully noted;
- (3) Refinement should begin by isolating candidate processes, data objects, and data stores to be represented at the next level;
- (4) all arrows and bubbles should be labeled with meaningful names;
- (5) *information flow continuity* must be maintained from level to level, 2 and
- (6) one bubble at a time should be refined

Unit 3: Building the Analysis Model

Flow oriented modeling:



Level 2 DFD
that refines
the *monitor*
sensors process



Unit 3: Building the Analysis Model

Behavioral modeling:

- Purely dynamic model
- The model indicates how to software will respond to external events
- Steps to create behavioral model:
 - 1) Evaluate all use-case to understand fully the sequence of interaction within the system
 - 2) identify events that drive the interaction sequence and understand how this events relate to specific objects
 - 3) Create a sequence diagram for each use case
 - 4) Build a state diagram for the system.
 - 5) Review the behavioral model to verify

accuracy and consistency.

How to identify events within use case?

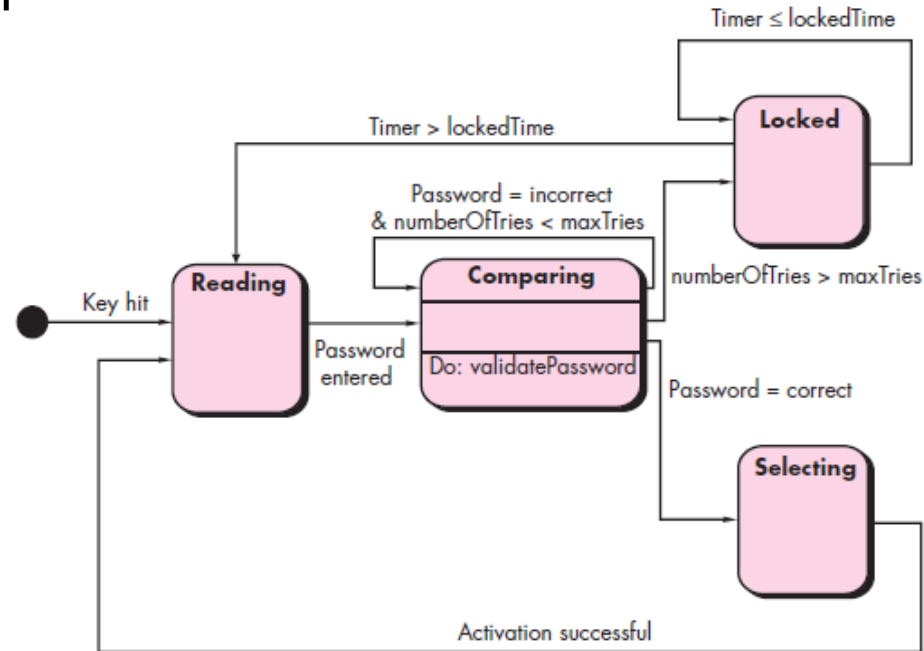
Ex: The home owner uses the keypad to key in a four-digit password. The password is compared with valid password stored in the system. If the password is incorrect, the control panel will beep once and reset itself for additional input. If the password is incorrect, the control panel awaits further action.

Unit 3: Building the Analysis Model

Behavioral modeling:

- ❑ Two different characterization of states:
- ▣ The state of each class as the system performs its functions
- ▣ The state of system as observed from the outside as the system perform its function.

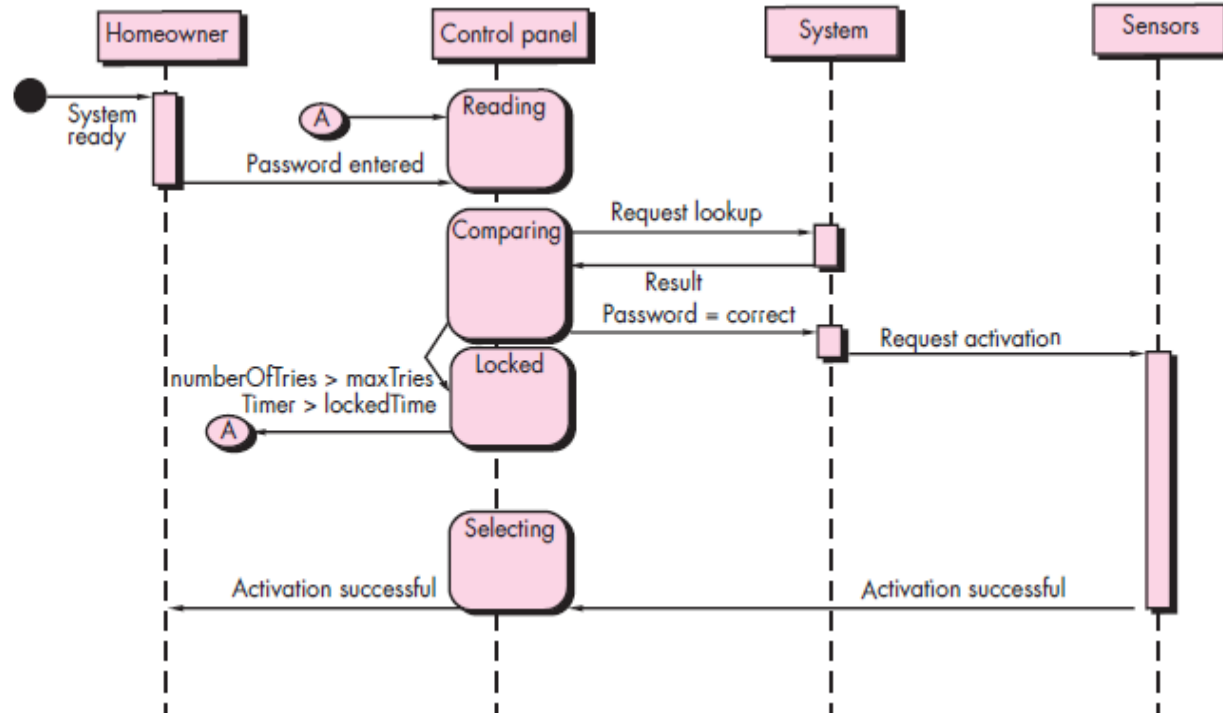
[State Diagram for Control Panel Class]



Unit 3: Building the Analysis Model

Behavioral modeling:

[Sequence Diagram
for SafeHome
Security Function]

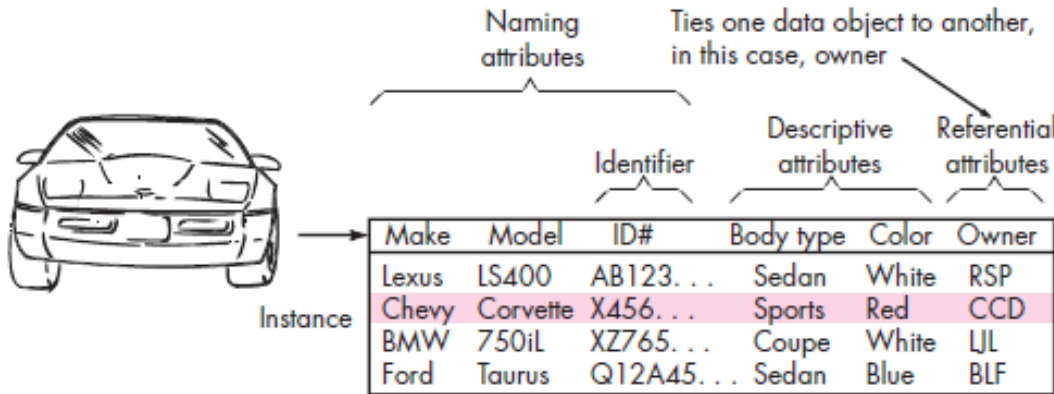


Unit 3: Building the Analysis Model

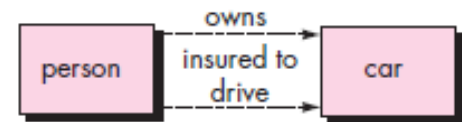
Data Modeling Concepts:

- ❑ **When to Use?** If your software needs to create, extend or interface with the database OR complex data structures (stack/queues etc.) must be constructed / manipulated.
- ❑ Data objects: A data object is a representation of composite information that must be understood by software.
- ❑ Composite info? No. of diff. properties : **Width** / **Dimension** (height x width x depth)

Tabular
representation
of data objects



(a) A basic connection between data objects

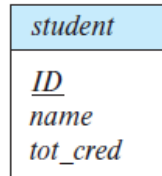
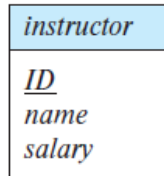


(b) Relationships between data objects

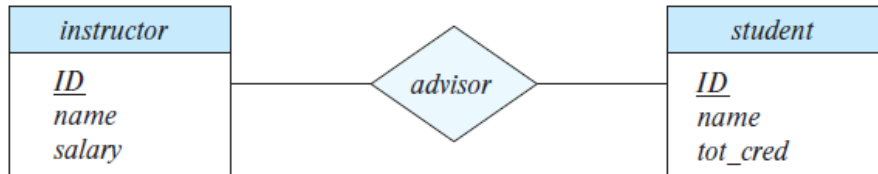
Unit 3: Building the Analysis Model

Data Modeling Concepts (E-R Modeling):

- Entity sets: An entity is a “thing” or “object” in the real world that is distinguishable from all other objects



- Relationship sets: A relationship is an association among several entities



- Attributes:

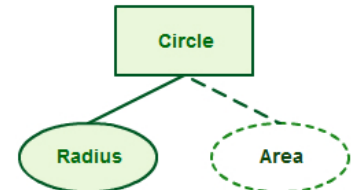
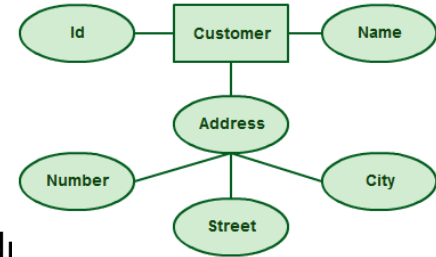
- Simple: age

- Composite: name, addi

- Single valued: weight

- Multivalued: Phone numbers

- Derived attributes: age from DoB

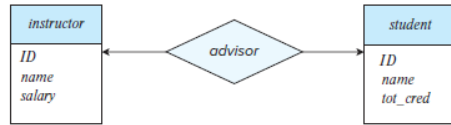


Unit 3: Building the Analysis Model

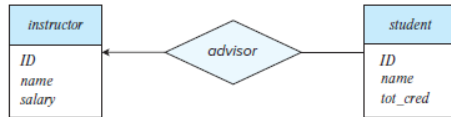
Data Modeling Concepts (E-R Modeling):

□ Constraints:

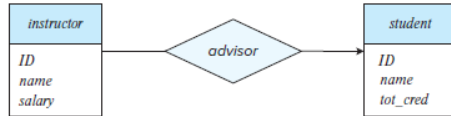
▣ Cardinality:



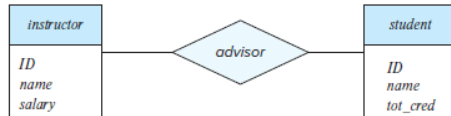
(a) One-to-one



(b) One-to-many



(c) Many-to-one



▣ Participation: Fully/Total, Partial

□ **Keys:** Primary, Foreign Key

□ **Dashed Lines:** link attributes of relationship sets to relationship set

□ **Doubled line:** total participation of an entity in relationship set.

□ **Doubled diamond:** Identifying relationship sets to weak entity sets

THANK YOU!

