
Software Engineering
Dr. Rasha Montaser

Lec 4
Requirement Analysis

Ref: Chapter 7

Requirement engineering

- The mechanism for:
  - Understanding customers needs
  - Analyze needs
  - Assessing feasibility
  - Negotiation
  - Validation of the specifications
  - Managing the requirements as they are transformed into operational

Functions used for accomplishing requirements engineering

1. Inception
2. Elicitation
3. Elaboration
4. Negotiation
5. Specification
6. Validation
7. Requirements management

1- Inception

a. Identify stakeholders
   - "who else do you think I should talk to?"

b. Recognize multiple points of view

c. Work toward collaboration

d. The first questions
   - Who is behind the request for this work?
   - Who will use the solution?
   - What will be the economic benefit of a successful solution
   - Is there another source for the solution that you need?

a. Stake holder

- Definition:
  - A stakeholder can be defined as “anyone who benefits in a direct or indirect way from the system which is being developed”
- How to select stake holders?
  - have appropriate business experience
  - good technical knowledge
  - the authority to negotiate needs in terms of delivery time and resources.

Example of stake holders:

- business managers
- product managers
- marketing people
- internal and external customers
- end users
- Consultants
- product engineers
- Web engineers
- support and maintenance staff

b. Recognizing multiple viewpoints

1. **business managers:** are interested in the feature set that will result in sales growth and improved revenue for the company.
2. **The marketing group:** is interested in features that will excite the potential market, leading to new customers and increased sales.
3. **The product manager:** wants a WebApp that can be built within budget and that will be ready to meet defined market windows.
4. **End users:** may want features that are already familiar to them and that are easy to learn and use.
5. **software engineers:** may be concerned with functions that are invisible to nontechnical stakeholders but that enable the infrastructure that supports more marketable features.
6. **Support engineers:** may focus on the maintainability and extensibility of the WebApp.

c. Working toward collaboration

- Identify areas of commonality and areas of conflict between the customers requiremnts.

d. Asking the first questions

1. Who is behind the request for this work?
2. Who will use the solution?
3. What will be the economics benefit of a successful solution?
4. Is there another source for the solution that you need?

2- Elicitation

- Tasks of elicitation:
  a. Collaborative requirements gathering
  b. Quality function requirements
  c. User scenarios
  d. Elicitation work products

a. Collaborative requirements gathering

- **Goal:**
  - to identify the problem
  - propose elements of the solution
  - negotiate different approaches, and
  - specify a preliminary set of solution requirements
- **Guide lines:**
  - A meeting is conducted and attended by all stakeholders.
  - Rules for preparation and participation are established.
  - An agenda is suggested that is formal enough to cover all important points but informal enough to encourage the free flow of ideas.
  - A facilitator (can be a customer, a Web engineer, or an outsider) controls the meeting.
  - A definition mechanism (can be work sheets, flip charts, wall stickers, an electronic bulletin board, chat room, or virtual forum) is used.

b. Quality Function Deployment

- Normal requirements: these requirements which reflects objectives and goals(Ex: level of performance, system functions, graphical display)
- Expected requirements: those requirements that are implicit to the product, fundamental that the customers doesn’t state (human/machine interaction, reliability, ease of installation)
- Exciting requirements: requirements that reflects the features that goes beyond the customer expectations

c. User scenarios

- stakeholders are asked to play the role of a user and develop usage scenarios for one or more entries on each of the lists. Each usage scenario is typically one or two narrative paragraphs that describe how an end user would apply or create a content object or interact with a WebApp function (use case).

d. 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.
- 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**.**

Problems of Elicitation

1. Problem of scope:
   - Customers define unnecessary technical details that may confuse, rather than the overall scope
2. Problem of understanding:
   - Customers not completely sure of what he needs (poor understanding of the capabilities and limitation of the environment, omit information that is obvious, specify requirements that conflict the need of other customers)
3. Problem of volatility:
   - The requirements change over time

Types of Requirements

- Functional requirements
  - Describe *what* the system should do
- Non-Functional Requirements

- Software Quality requirements
  - *Constraints* on the design to meet specified levels of quality
- Platform requirements
  - *Constraints* on the environment and technology of the system
- Process requirements
  - *Constraints* on the project plan and development methods

Functional Requirements

- What *inputs* the system should accept
- What *outputs* the system should produce
- What data the system should *store* that other systems might use
- What *computations* the system should perform
- The *timing and synchronization* of the above

Functional Requirements

- An individual requirement often covers more than one of the above categories.
  - For example, the requirements for a word processor might say, '*when the user selects "word count", the system displays a dialog box listing the number of characters, words, sentences, lines, paragraphs, pages, and words per sentence in the current document*.'
  - This requirement clearly describes input, output and computation.

Software Quality Requirementts

- All must be verifiable
- Examples: Constraints on
  - Availability
    - Telecommunications systems have very rigorous availability criteria:
      - for example, you might specify that such a system must not be down more than 10 minutes in its 20-year life-span.
      - This is also called '6-nines' availability, since it is equivalent to saying that the system must be up 99.9999% of the time.
  - Recovery from failure (think a failing banking transaction)
  - Allowances for maintainability and enhancement
  - Allowances for reusability

Software Quality Requirementts

- All must be verifiable
- Examples: Constraints on
  - Response time
  - Quantity
  - Resource usage
  - Reliability
    - The average amount of time between failures or
    - The probability of a failure in a given period.

Example of platform requirements

- Software Configuration: This software package is developed using java as front end which is supported by sun micro system. Microsoft SQL Server as the back end to store the database.
  - Operating System: Windows NT, windows 98, Windows XP
  - Language: Java Runtime Environment, Net beans 7.0.1 (front end)
  - Database: MS SQL Server (back end)
- Hardware Configuration:
  - Processor: Pentium(R)Dual-core CPU
  - Hard Disk: 40GB RAM: 256 MB or more

Example of functional requirements

- The software automatically validates customers against the ABC Contact Management System
- The Sales system should allow users to record customers sales
- The background color for all windows in the application will be blue and have a hexadecimal RGB color value of 0x0000FF.
- Only Managerial level employees have the right to view revenue data.
- The software system should be integrated with banking API
- The software system should pass Section 508 accessibility requirement.

Example of non-functional requirements

- Users must change the initially assigned login password immediately after the first successful login. Moreover, the initial should never be reused.
- Employees never allowed to update their salary information. Such attempt should be reported to the security administrator.
- Every unsuccessful attempt by a user to access an item of data shall be recorded on an audit trail.
- A website should be capable enough to handle 20 million users with affecting its performance
- The software should be portable. So moving from one OS to other OS does not create any problem.
- Privacy of information, the export of restricted technologies, intellectual property rights, etc. should be audited

Exercise

- Which of these are functional requirements?
  a. Users of the library will be either normal or staff
  b. A user will be able to borrow a book
  c. A staff person will be able to borrow a book
  d. The library contains one million books
  e. If a user asks a book that has been borrowed, her request shall be inserted in a waiting list
  f. Staff shall have no priority in borrowing books

Exercise

- Which of these are non-functional requirements?
  a. Pressing the switch, the room shall get lightened
  b. Pressing the switch, the room shall get lightened in less than one second
  c. If the room is dark, pressing the switch it shall get lightened
  d. The light in the room must be sufficient to read
  e. If someone is reading then the light must stay on

Exercise

- Which of the following is a non-functional requirement?
  a. The system enables users to place lunch orders.
  b. The system always responds to user clicks in less than one tenth of a second.
  c. The system displays a list of hotel vacancies.
  d. The system notifies the user when a new order arrives.
  e. None of the above

Exercise

- Which question does non-functional requirements answer?
  a. What does the system do?
  b. When does the system do it?
  c. Where does the system do it?
  d. Why does the system do it?
  e. How well does the system do it?

Extract the functional and non functional requirements from the following description

- You are the manager of Jiovani Naseem Soft; a mobile and web development company. You won a contract to develop a mobile app for Nourhan Habashi Restaurant. The application will support the 2 most popular mobile platforms: Android and iOS (for iPhone). It will allow the customer to pre-order his meal before going to the restaurant. The customer can view the available dishes and their description and prices. He can mark the dishes he wants and specify his expected arrival time and the number of people coming. He must pay in advance by credit card. Then he receives a confirmation number. The kitchen in the restaurant receives the orders on a computer screen and it remains in the list of inactive orders. Half an hour before the expected arrival time of the customer, the order becomes active and they start preparing it to serve it on time. The system should be secure and encrypt financial data and credit card information so no hacker can steal it.

3- Elaboration

- create an analysis model that identifies data, function and behavioral requirements
- Elements of the Analysis Model
  a. Scenario-based elements
     - Functional—processing narratives for software functions
     - Use-case—descriptions of the interaction between an “actor” and the system
  b. Class-based elements
     - Implied by scenarios
  c. Behavioral elements
     - State diagram
  d. Flow-oriented elements
     - Data flow diagram

User cases

- Use cases describe how a specific user category (called an actor) will interact with the system to accomplish a specific action.
- **Benefits:**
  - Use cases provide the detail necessary for effective planning and modeling activities.
  - Use cases help the developer to understand how users perceive their interaction with the system.
  - Use cases help to compartmentalize software engineering work.
  - Use cases provide important guidance for those who must test the software.

a. Use-Cases: describing how the user will use the system

- A *use case* is a typical sequence of actions that a user performs in order to complete a given task
  - The objective of *use case analysis* is to model the system from the point of view of...
    - how users interact with this system
    - when trying to achieve their objectives

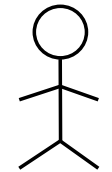
    It is one of the key activities in requirements analysis
  - A *use case model* consists of
    - a set of use cases
    - an optional description or diagram indicating how they are related

Use cases

- A use case should
  - Cover the *full sequence of steps* from the beginning of a task until the end.
  - Describe the *user's interaction* with the system ...
    - Not the computations the system performs.
  - Be written so as to be as *independent* as possible from any particular user interface design.
  - Only include actions in which the actor interacts with the computer.
    - Not actions a user does manually nor internal actions of the system

Start by identifying actors

- An *actor* *is an entity that interacts with the system and/or needs to exchange information*
- Find actors by asking:
  - Who uses the system?
  - Who installs the system?
  - Who starts up the system?
  - Who maintains the system?
  - Who shuts down the system?
  - What other systems use this system?
  - Who gets information from this system?
  - Who provides information to the system?



Next, identify use cases

- Once you identified actors, ask these questions:
  - What functions will the actor want from the system?
  - Does the system store information?
  - What actors will create, read, update or stored information?
  - Does the system need to inform an actor about changes in the internal state?
  - Are there any external events the system must know about? What actor informs the system of those events?

Exercise

- Consider a security online system called safehome which is used for controlling alarms embedded in the home security system
- We consider 3 actors: home owner, system administrator, sensors
- The home owner interact with the system either by the alarm control panel or a pc

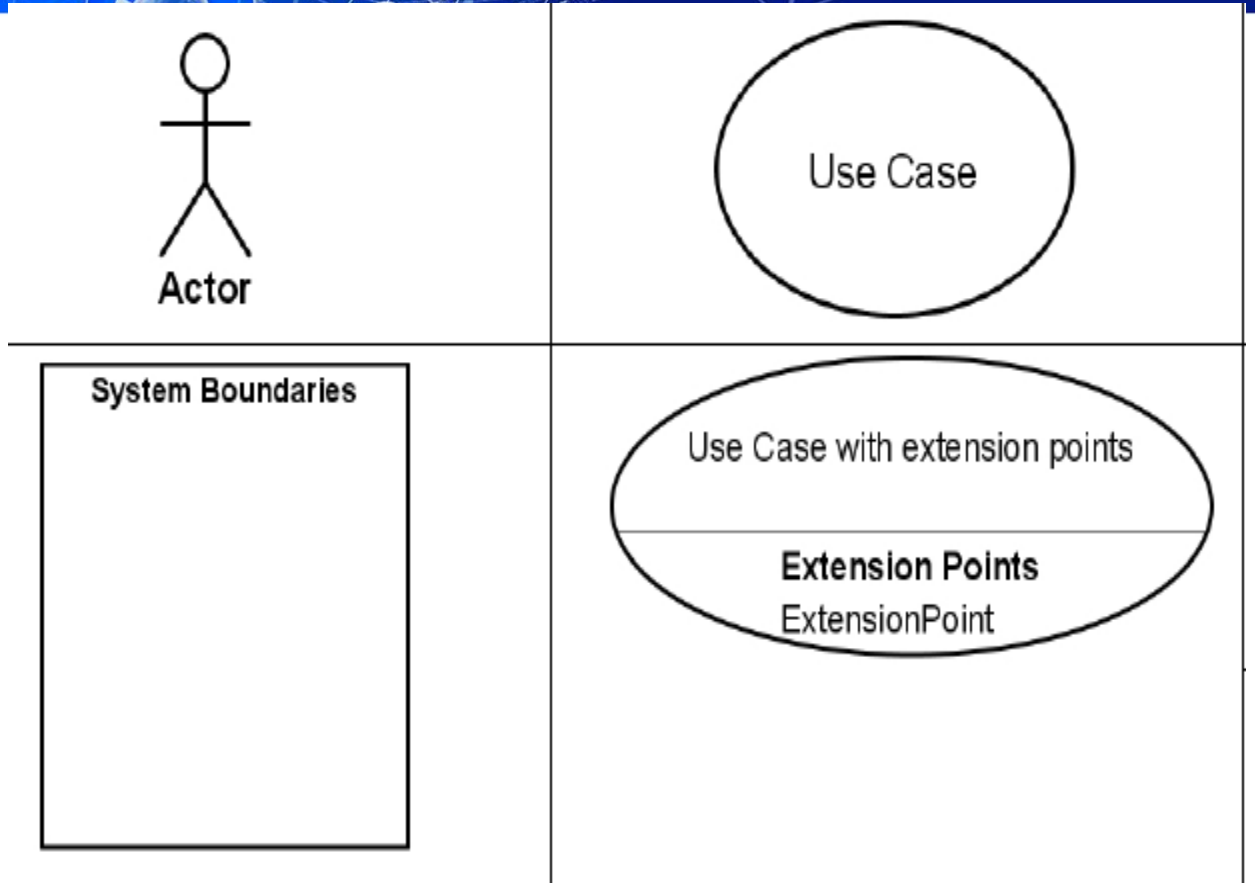
Exercise

- Functions of home owner
  - Arms/disarms the system
  - Access the system via internet
  - Responds to alarm event
  - Encounter an error condition
- Functions of system administrator
  - Organize sensors and related system features
- Functions of sensors
  - Responds to alarm event
  - Encounter an error condition
  - Organize sensors and related system features

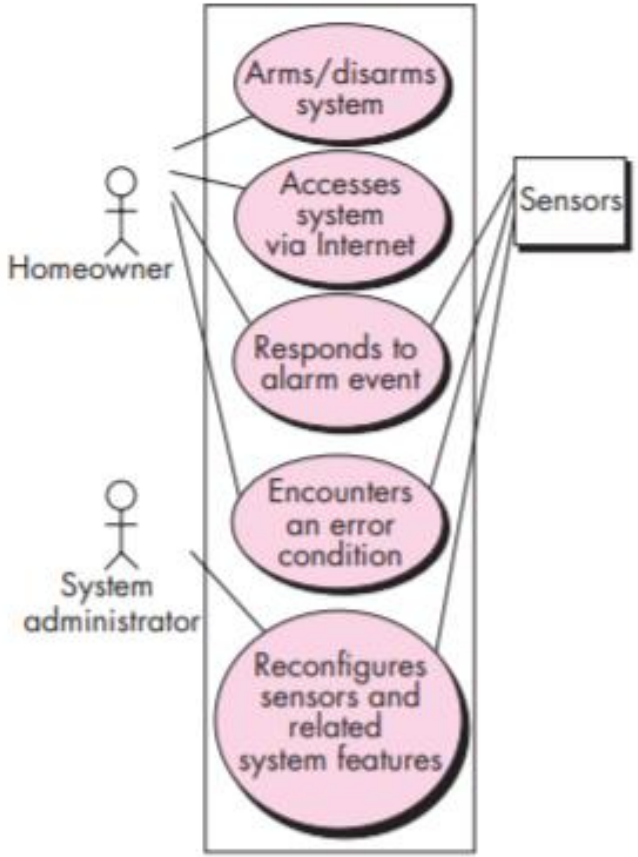
Exercise

- Draw use case diagram for the safehome security function
- Write the usecase for the home owner function (arms/ disarms the system)

Use case diagram



Use case diagram



How to describe a single use case

A. Name: Give a short, descriptive name to the use case.
B. Actors: List the actors who can perform this use case.
C. Goals: Explain what the actor or actors are trying to achieve.
D. Preconditions: State of the system before the use case.
E. Trigger: Give a short informal description.
F. Scenario: Describe each step (Actor actions and System responses).
G. Exception: Alternative flow of events.
H. Priority – when available – frequency of use –channel to actor – open issues

- **A , B and F are the most important**

Use case

- Use case: Arms the system
- Primary actor: Homeowner.
- Goal in context: To set the system to monitor sensors when the homeowner leaves the house or remains inside.
- Preconditions: System has been programmed for a password and to recognize various sensors.
- Trigger: The homeowner decides to “set” the system, i.e., to turn on the alarm functions.
- Scenario:
    1. Homeowner: observes control panel
    2. Homeowner: enters password
    3. Control panel checks the validity of the password and give the user access to the system functionality
    4. Homeowner: selects “stay” or “away”
    5. The device read alarm light to indicate that SafeHome has been armed

Use case

- Exceptions:
  1. Control panel is not ready: homeowner checks all sensors to determine which are open; closes them.
  2. Password is incorrect (control panel beeps once): homeowner reenters correct password.
  3. Password not recognized: monitoring and response subsystem must be contacted to reprogram password.
  4. Stay is selected: control panel beeps twice and a stay light is lit; perimeter sensors are activated.
  5. Away is selected: control panel beeps three times and an away light is lit; all sensors are activated.

Use case

- Priority: Essential, must be implemented
- When available: First increment
- Frequency of use: Many times per day
- Channel to actor: Via control panel interface
- Open issues:
  1. Should there be a way to activate the system without the use of a password or with an abbreviated password?
  2. Should the control panel display additional text messages?
  3. How much time does the homeowner have to enter the password from the time the first key is pressed?
  4. Is there a way to deactivate the system before it actually activates?

Exercise

- Consider a university registration system, which is used for registering in the course and uploading the students grade to it.

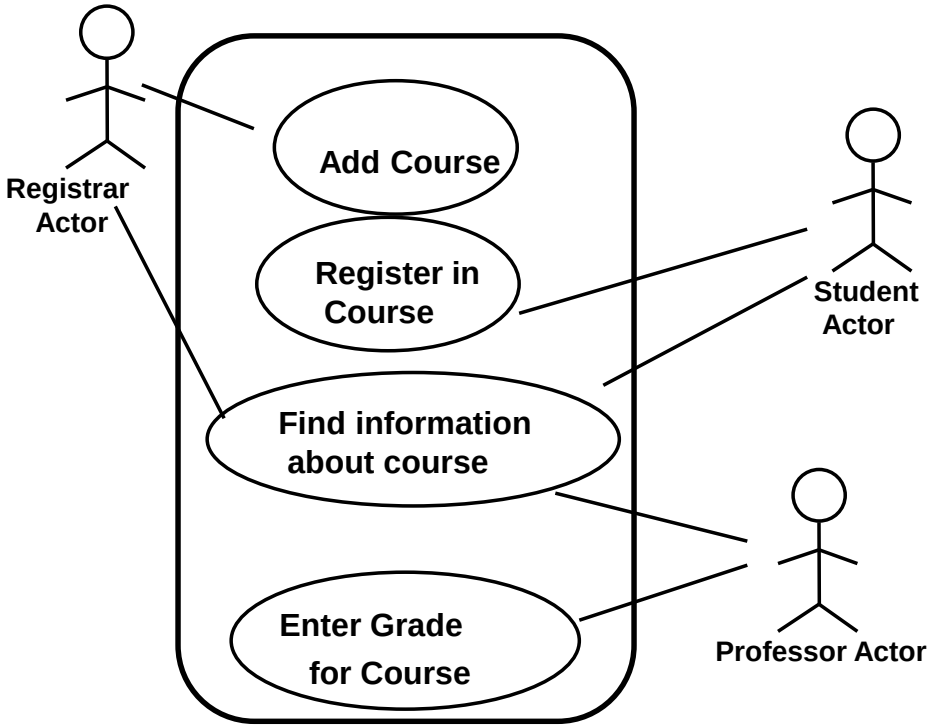
a. Who are the actors?

b. What are the functions which each actor use to interact with the system?

c. Draw use case diagram

d. Write the use case (scenario) for 3 functions

Use case diagrams for registration system



Library System

- Consider a small library management which supports the following functionalities:

1. Check out a copy of a book. Return a copy of a book.
2. Put a book on hold if already borrowed.
3. Add a copy of a book to the library. Remove a copy of a book from the library.
4. Browse the list of books using different filtering sorting criteria.
5. Find out the list of books currently checked out by a particular borrower.
6. Check users' personal information.
7. Suspend a user.
8. Set borrowing rules.

Library System

- There are two types of users: staff users and ordinary borrowers. Staff are divided into two categories: Librarians and Clerks.
- A library user can do tasks 2, 4, 5 and 6. Buy he can do 5 and 6 only on his own records.
- A clerk can 1, 2, 4, 5, 6 and 7
- A librarian can do all the tasks.

Library System

- There could be multiple copies of the same book.
- No copy of the book may be both available and checked out at the same time.
- A borrower may not have more than a predefined limit on the number of books checked out at one time.
- Putting a hold on a borrowed book, will give priority to borrow it when it comes.

Library System

- Late borrowers pay a fine.
- Fine rules are ……..
- A book is defined by …….
- Borrowing rules are ……..
- Borrower (user) is defined by ……
- Process for adding a book is ……..
- Rules for suspending a borrower are ……

Identify Library System Actors

- Read the description and find who interacts with the system, who inputs information, who queries the system for information, etc.
- User: a library user who can browse catalog, read books and borrow books.
- Clerk: is an employee that serves library users
- Librarian: maintains library catalog and adds and removes books from the library.

Identify Library System Use Cases$_1$

- Borrower:
- ❑ Search for items by title.
  - ❑ … by author
  - ❑ … by subject
  - ❑ … by title
  - ❑ … by ISBN
  - ❑ … by a combination of the above
- ❑ Place a book on hold if it is on loan to somebody else.
- ❑ Check the borrower’s personal information and list of books currently borrowed.

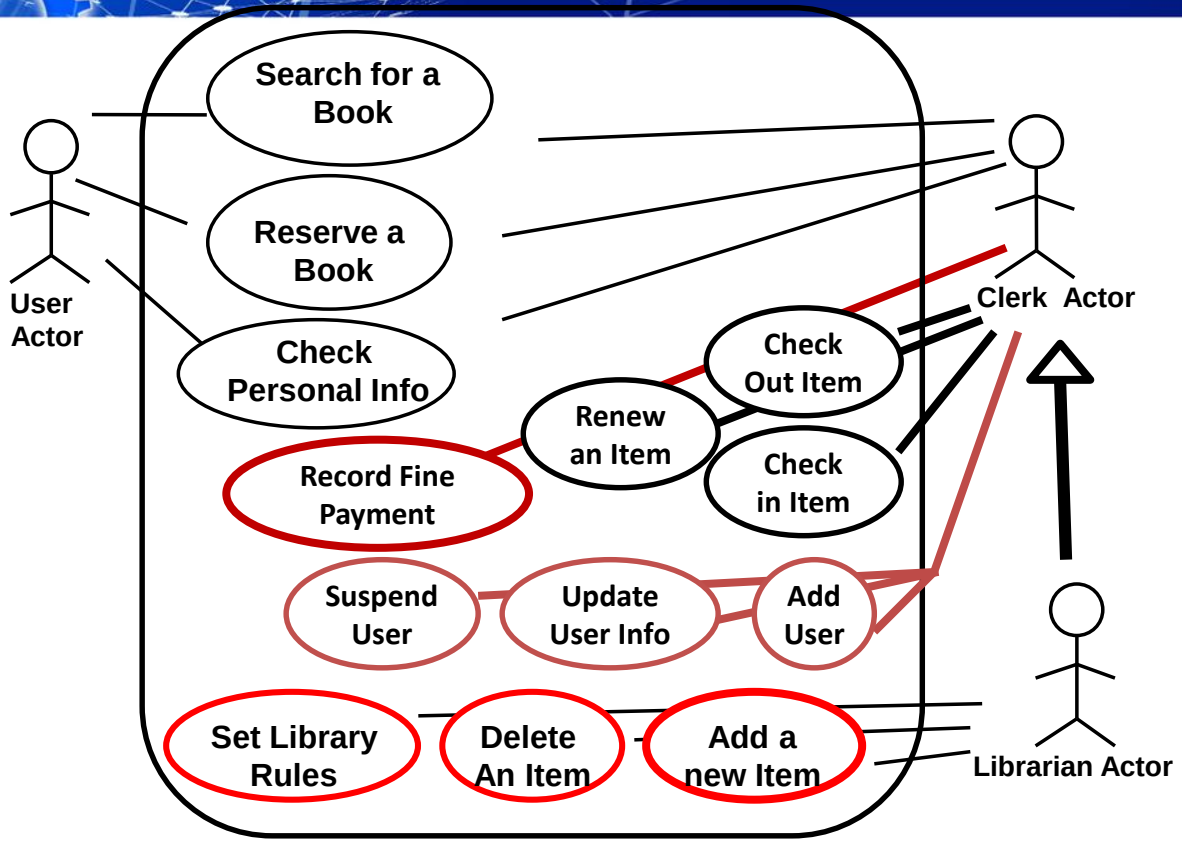
Identify Library System Use Cases$_2$

- Checkout Clerk:
- ❑ All the Borrower use cases, plus
- ❑ Check out an item for a borrower.
- ❑ Check in an item that has been returned.
- ❑ Renew an item.
- ❑ Record that a fine has been paid.
- ❑ Add a new borrower.
- ❑ Update a borrower’s personal information
- ❑ Suspend a borrower.

Identify Library System Use Cases$_3$

- Librarian:
- ❑ All of the Borrower and Checkout Clerk use cases, plus
- ❑ Add a new item to the collection.
- ❑ Delete an item from the collection.
- ❑ Change the information the system has recorded about an item.

Library System Use Case Diagram

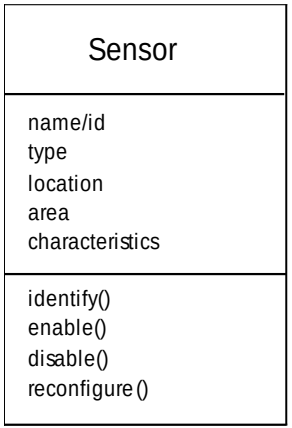


b. Class-based elements

- Set of objects that are manipulated as an actor and interact with the system.
- These objects are categorized into classes – a collection of things that have similar attributes and common behaviors.

Class Diagram

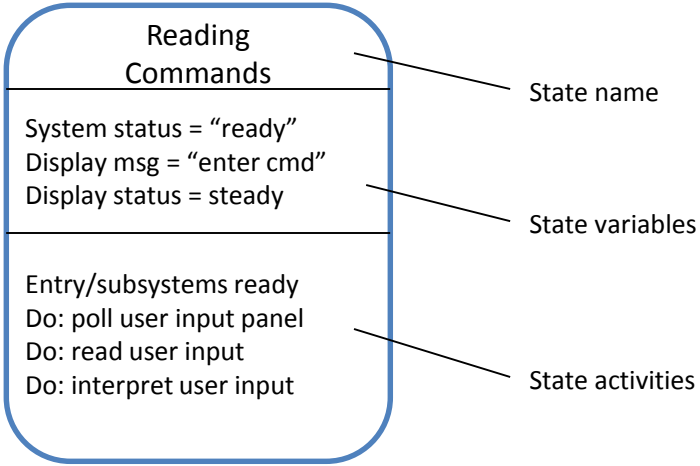
From the *SafeHome* system …



c. Behavioral elements

- Modeling elements that depict the behavior of the system
- State diagram: a method for representing the behavior of a system by depicting it’s states and the events that cause the system to change it’s state.

State Diagram



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- Validating Requirements - I

- 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 an add-on feature that may not be essential to the objective of the system?
- Is each requirement bounded and unambiguous?
- Does each requirement have attribution? That is, is a source (generally, a specific individual) noted for each requirement?
- Do any requirements conflict with other requirements?

5- Validating Requirements - II

- Is each requirement achievable in the technical environment that will house the system or product?
- 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.
- Have requirements patterns been used to simplify the requirements model. Have all patterns been properly validated? Are all patterns consistent with customer requirements?

6- Requirement management

- Set of activities that help the project team identify, control and trach requirements and changes to requirements at any time as the project proceeds.

Managing Changing Requirements

- Requirements change because:
  - Business process changes
  - Technology changes
  - The problem becomes better understood

- Requirements analysis never stops
  - Continue to interact with the clients and users
  - The benefits of changes must outweigh the costs.
    - Certain small changes (e.g. look and feel of the UI) are usually quick and easy to make at relatively little cost.
    - Larger-scale changes have to be carefully assessed
      - Forcing unexpected changes into a partially built system will probably result in a poor design and late delivery
  - Some changes are enhancements in disguise
    - Avoid making the system *bigger*, only make it *better*

Exercise

- Solve the following lecture practice in groups
- Exercise