Topic4: Requirements Engineering

- Meaning: requirements
- Levels of requirements
- Types of requirements
- Requirements modeling
- Use case modeling
- Drawing use case diagrams
- Example
- Exercise

4.1 What is a requirement?

- A requirement is:

1) a function that a system must perform
2) a desired characteristic of a system
3) a statement about the proposed system that all stakeholders agree that must be true in order for the customer's problem to be adequately solved.

4.2 Requirements Engineering

- ***DEFN.*** **Requirements engineering** is the process of establishing
  - **Services** that customer requires from a system.
  - **Constraints** under which a system will be developed and operate.
- The purpose is to understand and document the exact requirement of the customer.

4.3 Risks of inadequate requirements engineering process

1. Unacceptable products i.e. due to insufficient user involvement.
2. Creeping user requirements i.e. contribute to project overruns and degraded product quality.
3. Ambiguous requirements i.e. lead to ill-spent time and rework.
4. Gold-plating by developers i.e. adding unnecessary features.
5. Missing key requirements i.e. results in an unacceptable product.

6. Poor project planning and tracking i.e. due to incompletely defined requirements.

4.4 Levels of Requirements

- **User requirements**
  - –Statements in natural language of the services the system should provide and its operational constraints. Written for **customers**
- **System requirements**
  - –A structured detailed descriptions of the system services. Written as a contract between **client** and **contractor**
- **Software requirement**

– A detailed software description which can serve as a basis for a design or implementation. Written for **developers**

4.5 Types of Requirements

- **Functional requirements**
  - Statements of services the system should provide and how the system should react to particular inputs and how the system should behave in particular situations

- **Non-functional requirements**
  - Define system properties and constraints or characteristics of the system which cannot be expressed as functions e.g. reliability, portability, maintainability, usability

4.6 Requirements Modeling

–Process of developing graphical representations to express requirements of the system for better understanding of its information flows, structure, content and behavior

4.6 Use Case Modeling

–It is a technique in UMLfor modeling the user requirements using a use case diagram

Components & Notation of Use Case Diagram

Use case diagram has three components:

- Actors
- Use cases
- System boundary
- Relationships
  a) Dependency - between use cases
  b) Generalization - between use cases or actors
  c) Association - between use cases and actors

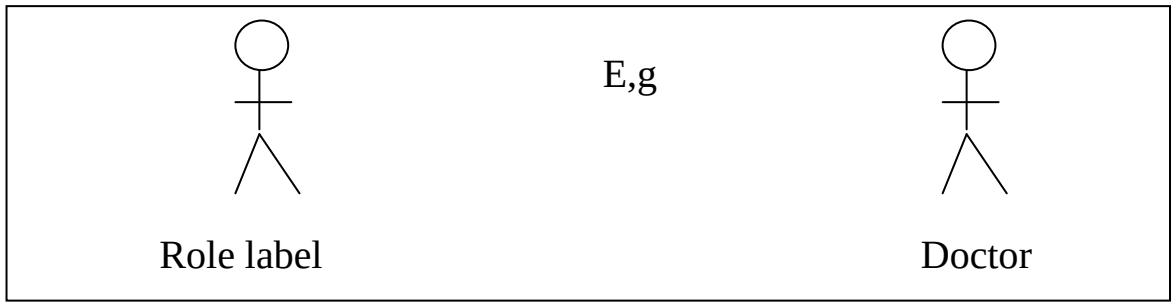
a. ACTOR

–A user of the system (i.e. something external to the system: human or nonhuman) acting in a particular role as they interact with the system causing it to respond to events.

–An actor is a role an external entity plays as it stimulates a system to react or respond to its request

Notation

–represented by a stickman figure with a role label

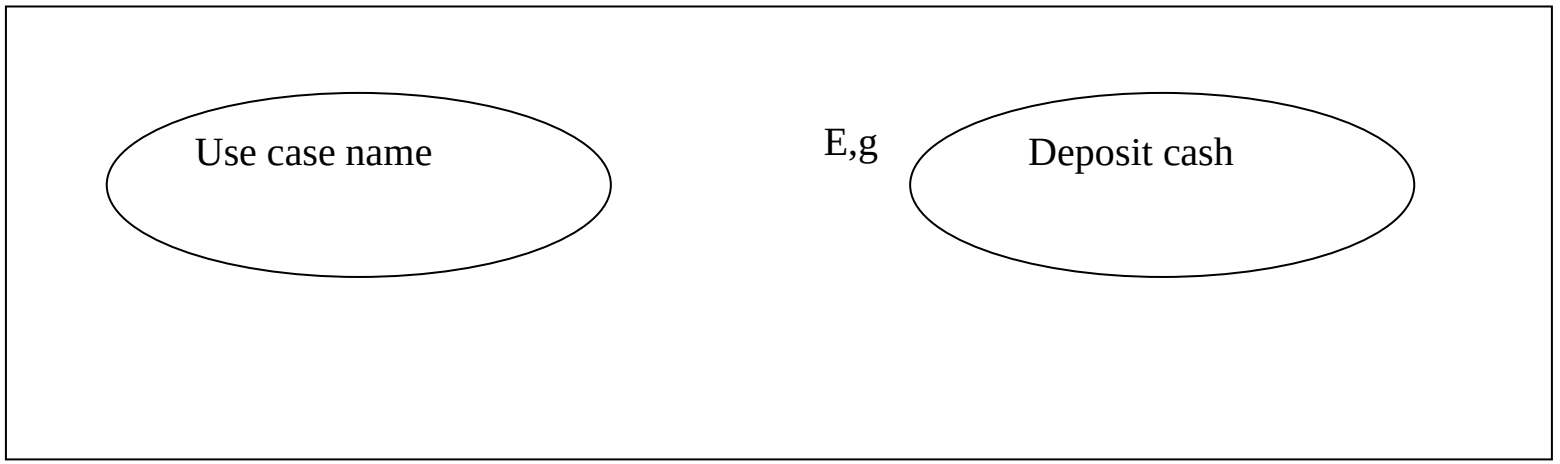


b. USE-CASE

–A task an **actor** needs to perform with the help of the system to get observable results, e.g find details of a book or print a copy of a receipt in a bookshop.

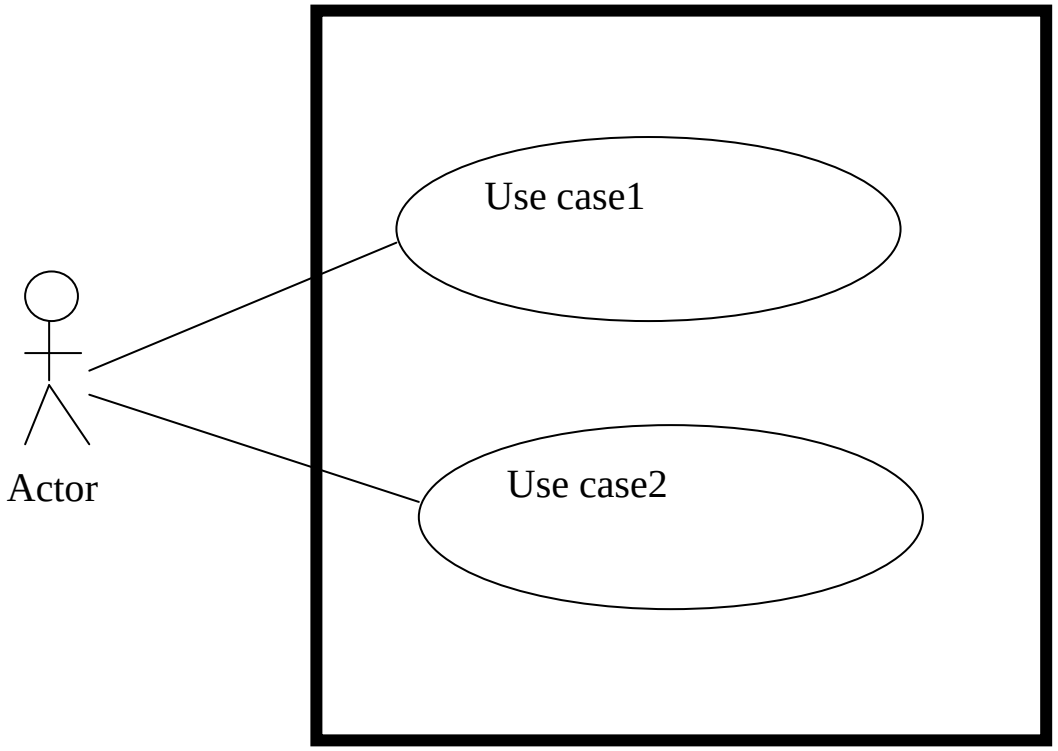
Notation

–represented by an oval shaped symbol labeled with name of use case.



c. SYSTEM BOUNDARY

–A box line showing extend of the system within which there are use-cases that actors are interacting with



d. RELATIONSHIPS

i) Dependency - between use cases

–Types of use case dependency relationships

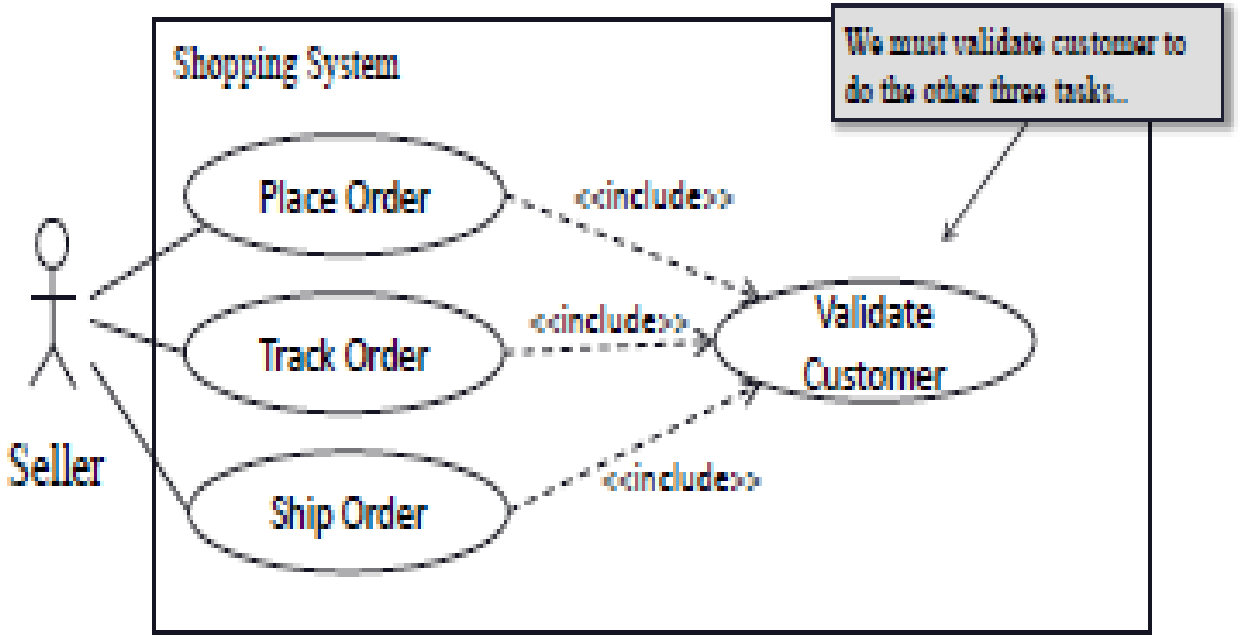
1) Include

–One use case includes, as part of their functionality, another use case.

Notation

–represented by an arrowed dotted line pointing from dependent (main use case) to independent (include use case) with a label of **<<include>>** relation.

Contd…


Shopping System
We must validate customer to do the other three tasks..
Place Order
Track Order
Ship Order
«include»
«include»
«include»
Validate Customer
Seller

2) Extend

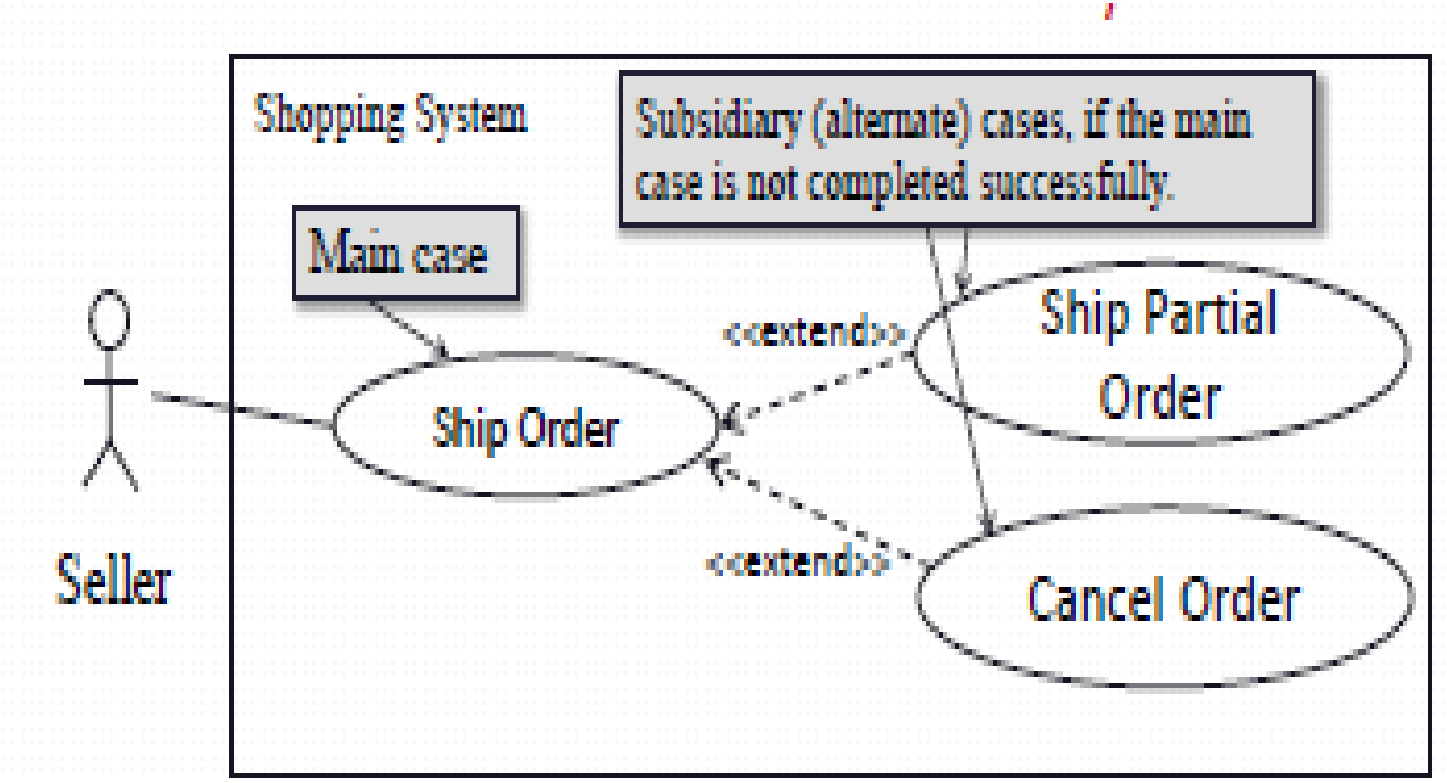
–one use-case has one or more alternative conditional use cases with significantly different outcomes.

Notation

–represented by an arrowed dotted line pointing from dependent (extend use case) to independent (main use case) with a label of **<<extend>>** relation.

Contd…

–Inside main use case is extension point showing condition under which alternative is selected

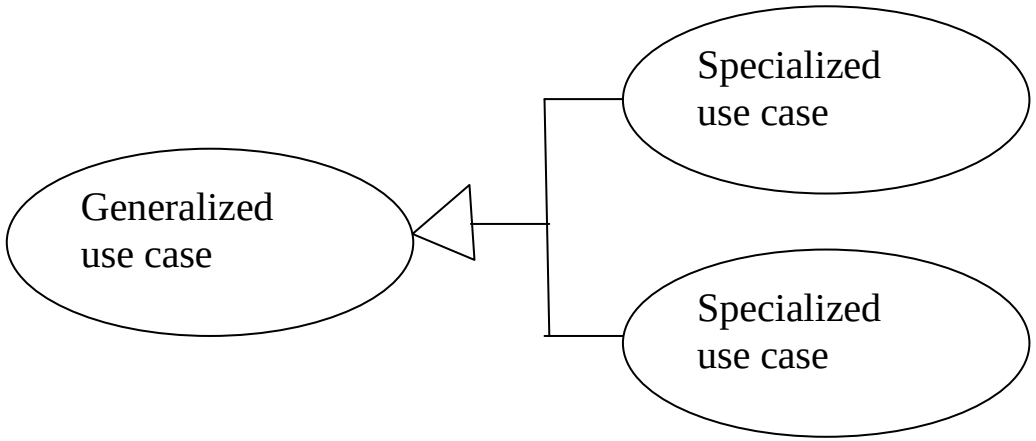


ii) Generalization - between use cases

–one use case has one or more specialized use cases

Notation

–represented by a solid line with a hollowed arrow head pointing from specialized use case to generalized use case

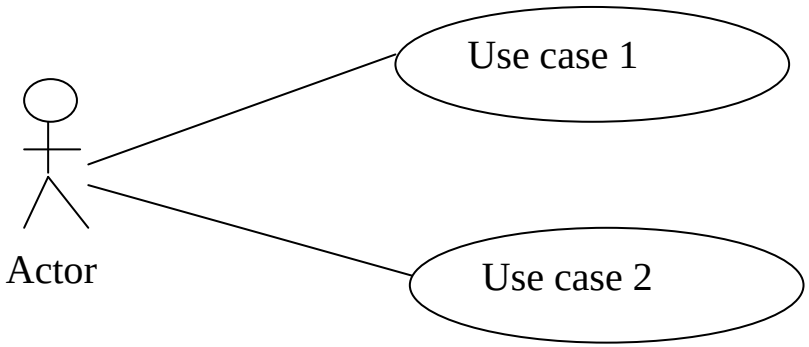


iii) Association - between actor and use case

–one actor interacts with one or more use cases

Notation

–represented by a solid line connecting actor and a use case



4.7 Drawing use case diagram

1. identify use cases

1. functions user wants a system to accomplish
2. operations that create, read, update, delete information
3. notifications to actors of changes in the internal state of the system

2. name use cases

use concrete verb-noun phrases:

1) weak verb indicates uncertainty, strong verb identified action taken:

a) strong verbs: create, calculate, migrate, activate, etc.

b) weak verbs: make, report, use, organize, record, etc.

2) weak noun refer to several objects while strong noun only one object

a) strong nouns: property, payment, transcript, etc.

b) weak nouns: data, paper, report, system, etc.

3. identifying actors

Determine who the actors are, try to answer the following questions:

1) who uses the system?

2) who gets information from the system?

3) who provides information to the system?

Contd…

4) who installs, starts up or maintains the system?

4. name actors

1) identify roles they adopt while using the system

2) name each role and define its distinguishing characteristics

5. identify relationships

1) identify associations

2) identify generalizations

3) identify dependencies