UNIT-1 UNIFIED PROCESS AND USE CASE DIAGRAMS

Introduction to OOAD with OOBasics - unified process - UML diagrams - Use case - Case study - the Next Gen pos system, Inception - Use Case Modelling - Relating Use cases - include, extend & generalization - when to use Use-Cases

Introduction to OOAD

Analysis is a creative activity or an investigation of the problem & requirements

Eg: To develop a Banking system

Design is to provide a conceptual solution that satisfies the requirements of a given problem.

Eg: For a Book Bank system

Object oriented Analysis (OOA)

Analysis activity in the software life cycle is to create a model of the system's functional requirements ie, independent of implementation constraints.

OOA is a process of identifying classes that plays an important role in achieving system goals & requirements

Object Oriented Design (OOD)

OOD is to design the classes identified during analysis phase & to provide the relationship that exists bhn them that satisfies the requirements

Eg: Book Borrow system

Methods to get Book Details, get payment details

Member of ——— Relationship b/w the classes student & Book borrow

[Student] ——— [Book Borrow]

Object Oriented Analysis & Design (OOAD)

It is a s/w Engg approach that models an appln (real time appln) by a set of s/w development Activities

It emphasis on identifying, describing & defining the s/w objects and shows how they collaborate with one another to fulfill the requirements by applying the OO paradigm & Visual modeling throughout the development life cycles.

UNIFIED PROCESS

Definition

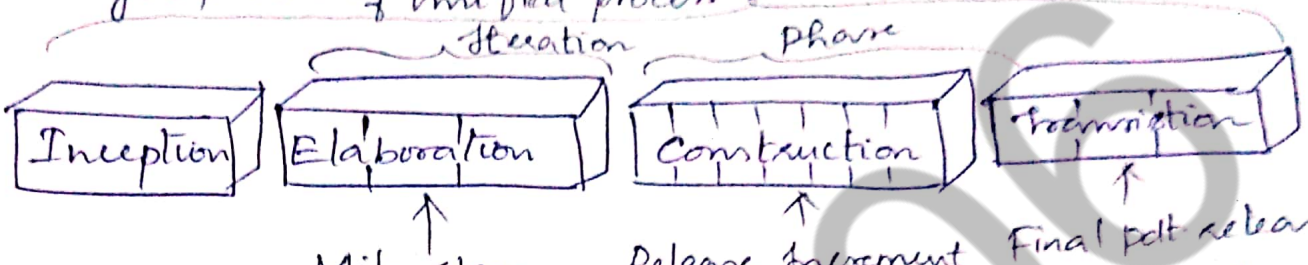
Unified process is a popular iterative process for projects using OOAD.

Best practices & key concepts in UP

* Tackle high risk & high-value issues in early iteration
* Engage users continuously for evaluation, feedback & rqmts.
* Build a cohesive & core Architecture
* Apply use cases provides Visual modeling using UML
* practice change request & configuration mgmt.

UP PHASES

4 major phases of unified process : Development cycle



1. Inception
2. Elaboration
3. Construction
4. Transition

↳ Iteration end pt
Significant decision or evaluation occurs

↳ an executable subset of final pdt.

Sys is released to usage

difference btn releases of 2 subsequent iterations

Inception which emphasizes approximate vision, businesscase, scope, vague estimates

Elaboration is a phase where core architecture is iteratively impl. & includes resolution of high risks & identification of most reqmts. & scope.

Construction phase encompasses iterative impl. of the pdt. & deploying it. Each iteration is a minor release.

Transition is phase of releasing the final pdt. to the customers for usability.

Unified Process Disciplines

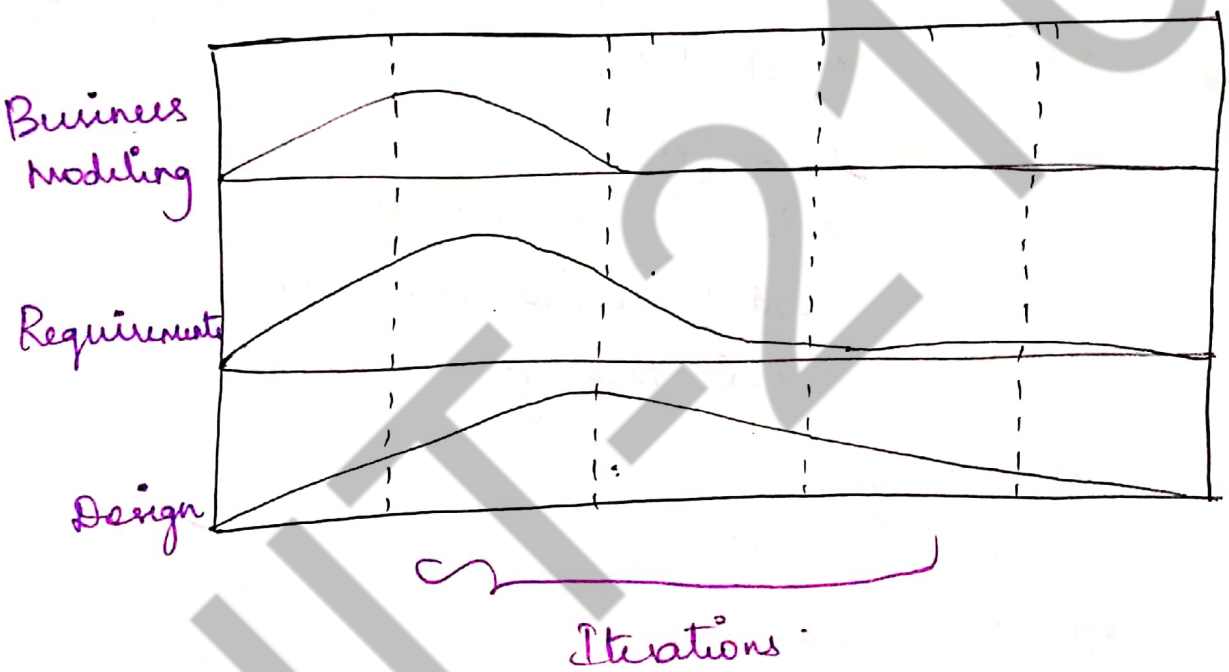
Definition:

Unified process describes work activities such as writing a use case

Use case → set of activities and related artifacts in one subject area within requirement analysis

Artifacts → code, web graphics, database schema, text document, diagrams, models etc.

UP disciplines



i) Business Modeling: It is a domain model artifact to visualize concepts in the application domain.

ii) Requirements - use case model and specification artifacts to analyze functional and non-functional requirements

iii) Design - Artifacts to design software objects

UML Diagrams

Unified Modeling Language

* It is a std. notation for the modeling of real-world Objects as a 1st step in developing an OOD methodology
* It is a Visual language for specifying, constructing documenting the artifacts of a system.

Various UML Diagrams

→ Use Case Diagram
→ Class Diagram
→ Interaction Diagram
→ State Diagram
→ Activity Diagram
→ package Diagram

Use Case Diagram are used to describe a set of actions (that some system or sys. should or can perform in Collabor with one or more external users of the sys. (actors).

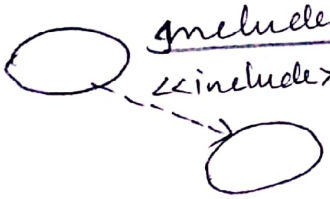
Purpose

- Used to gather reqmts. of a sys.
- Used to get an outside view of a sys
- Identify external & internal factors influencing the sys.
- Show the interaction among the reqmts. & actors

Basic Notations Used in Use case diagram

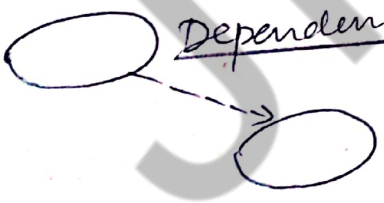
Actor - Needs to exchange information with the sys. May be people, computer h/w, other sys., etc.

System (shape) or system boundary.
- Actors who interact with the system are put outside the system.

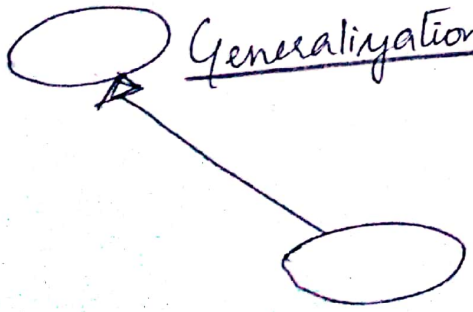


include <<include>> Relationship specifies how the behavior for the inclusion use case is inserted into the behavior defined for the base UC.

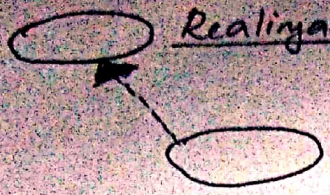
Extend <<extend>> Relationship specifies how the behavior of the extension uc can be inserted into the behavior defined for the base uc.



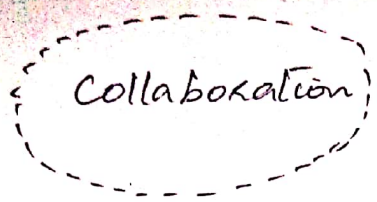
Dependency Relationship represents that a model element relies on another model element for specification &/or impl.



Generalization Relationship is used to represent inheritance relationship btw model elements of same type



Realization Relationship btw a specification & its impl.



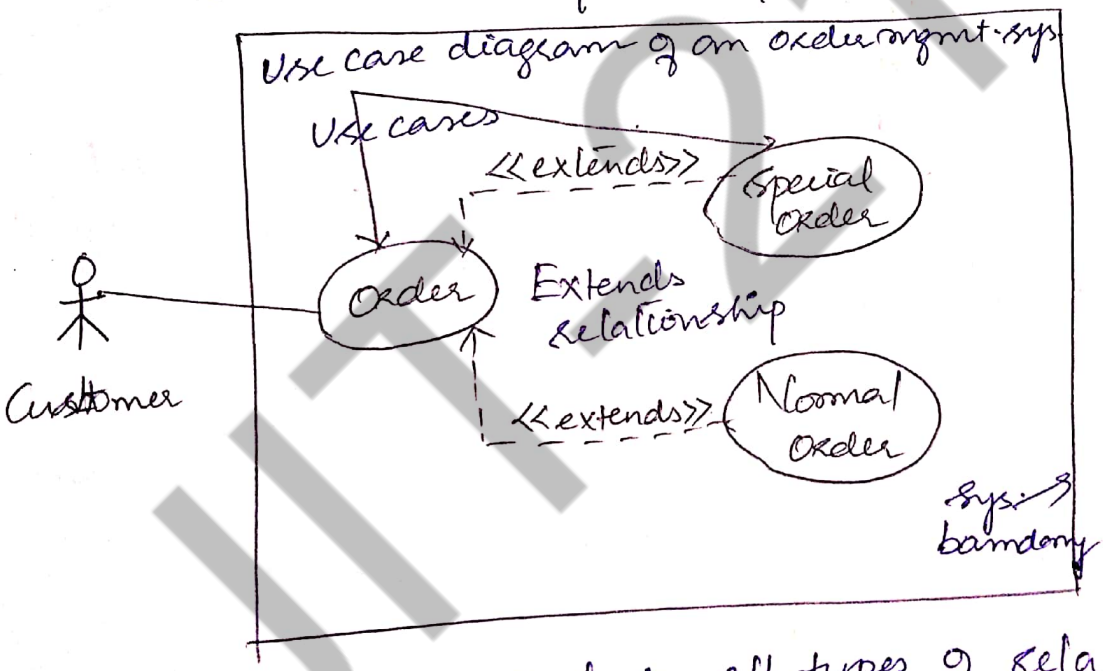
describes a structure of collaborating elements (roles), each performing a specialized fn.

How to draw Use Case diagram?

* Do the following things to draw a Use Case diagram

→ Suitable Name for actors

→ Show relationships & dependencies clearly in the diagram

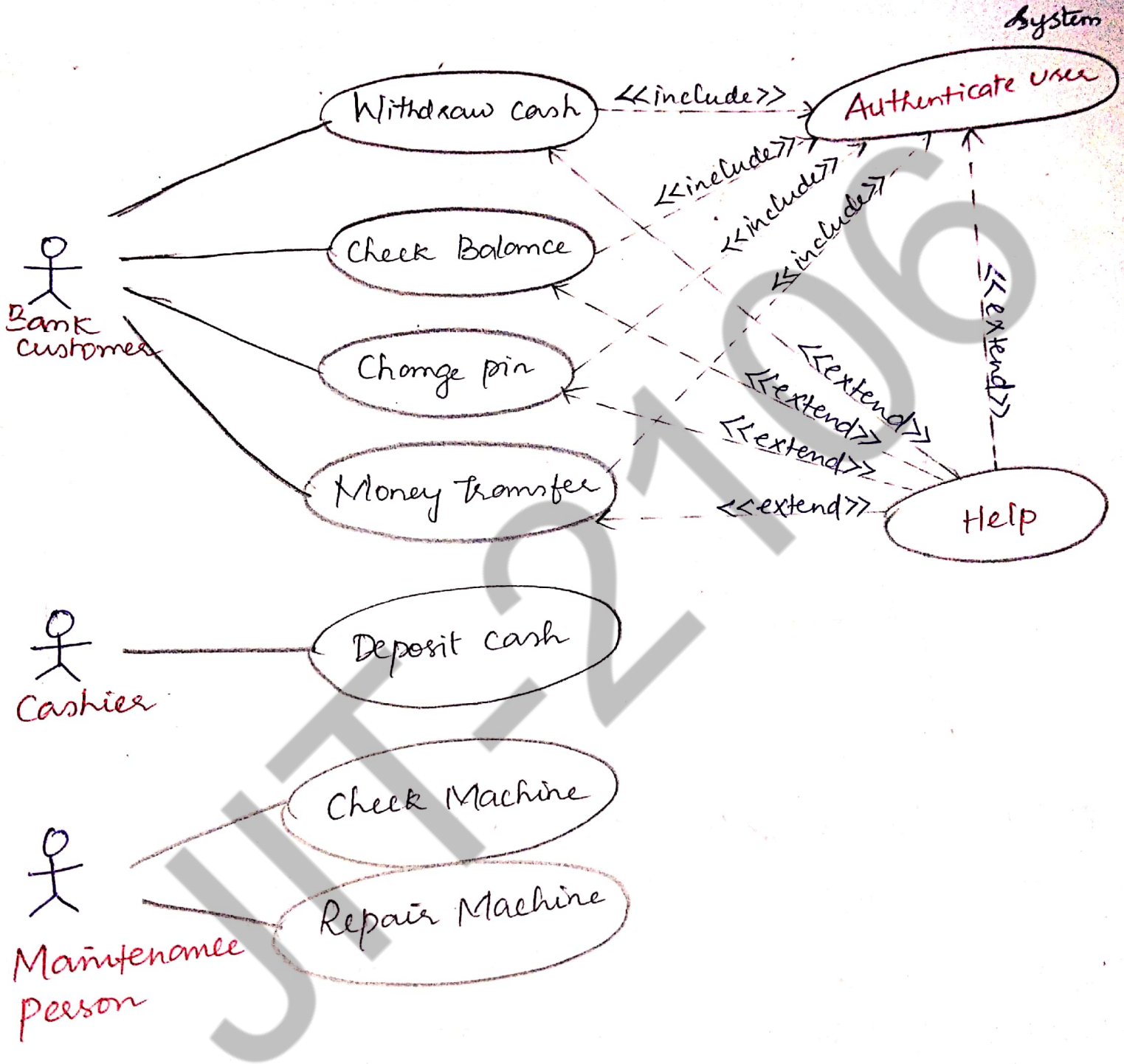


→ Don't try to include all types of relationships, because main purpose of the diagram is to identify rqmts.

Uses * Rqmt. Analysis * high level design

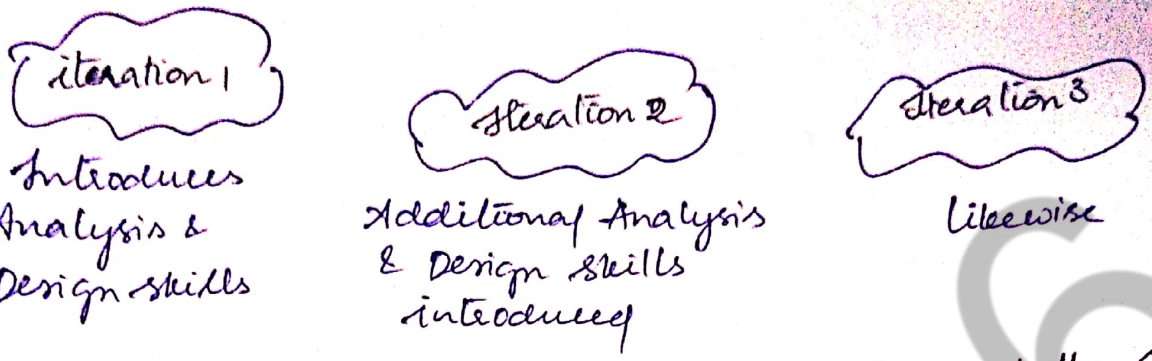
* Model the context of a sys.
* Reverse Engg.
* Forward Engg.

Eg: Use Case diagram for ATM System



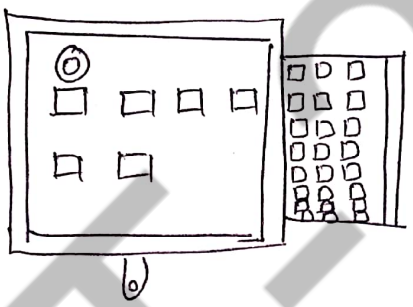
CASE STUDY

Case study Strategy - Iterative Development + Iterative learning



Iterative Development system grows incrementally over time iteration by iteration.

CASE ONE: The Next GEN POS System



* Next Generation Point of scale (POS) sys. is a Computerized appln. to record sales & handle payments
* It is used in retail store, includes hw components such as a computer & bar code scanner & sw to run the sys.
* It can be i/f with various service appln such as calculator & inventory control
* pos is a fault tolerant sys.
* pos sys. supports multiple & varied c/s terminals i/f. It includes a) thin client WB terminal
  b) a regular personal computer with Java swing GUI
  c) touch screen i/p
  d) Wireless PDA, etc.

Inception

Definition

It is the phase that provides approximate Vision, business case, scope & believable estimate of the project.

It is a feasibility phase of making investigation to support a decision of continuing the new sys. or to stop.

Eg. Textile Business

How long is inception

- Inception phase starts with a common Vision for the Objectives of the project
- Determine the overall purpose & feasibility of the potential new system & decide if it is worth while to invest
- If it is worth, the project will be done. O/w stop
- Inception phase ends & elaboration starts

How much UML During Inception

Inception phase is to collect enough information to establish a common Vision

decide whether the project is worth.

What Artifacts may start in inception

Artifacts	Description
1. Vision & Business case	high level goals & constraints
2. Use-case model	Describe functional Requirements
3. Supplementary Specification	Non-functional Requirements
4. Glossary	key domain terminology & data Dictionary
5. Risk list & Risk mgmt. plan	Business, technical, resource & Schedule risks & their mitigation or response

6. Prototypes & proof of concepts
7. Iteration plan
8. Phase plan & slw development plan
9. Development case

provides clear idea of the Vision & Validates technical ideas
Describes what to do in the 1st Iteration of elaboration
Low precision guess for elaboration phase duration
Customized UP steps & artifacts

Use Case Modeling

Use Case Model provides an external view of the system as application directed towards the users of the actors of the system

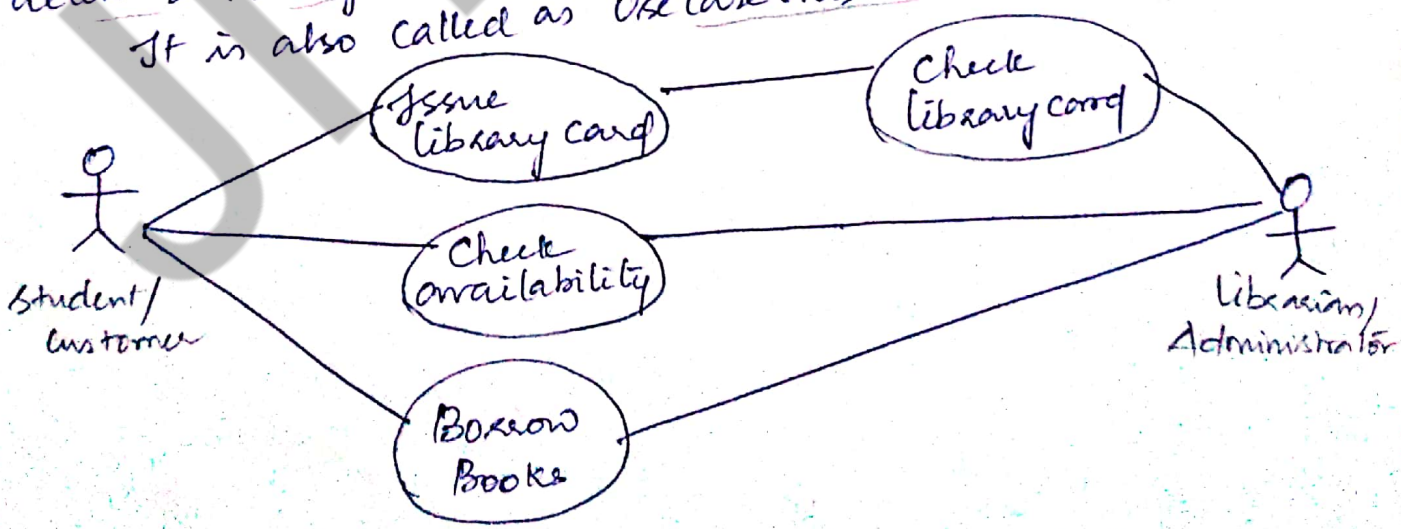
Use-case Diagram is a graph of actors, a set of use-cases enclosed by a system boundary, comm. associations bhn the actors & use cases & generalization among the use cases

Actors

An actor is anything that interacts with a use case

Scenario is a specific sequence of actions & interactions bhn actors & the system.

It is also called as use case instance



Use-cases

A Use-case is a static description of some way in which a system or a business is used by its users or actors.

It is a collection of related success & failure scenarios that describes an actor using a system to achieve the goal

Symbol to represent Use-case

Use Cases & the Use case Model

Use case are defined as text doc. not as diagrams in UP.

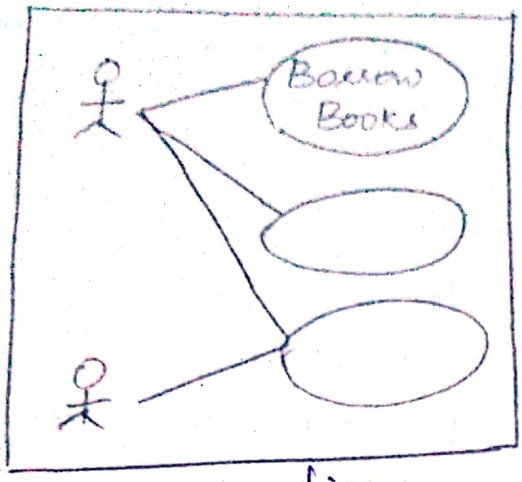
Use case model in UP optionally includes UML Use case diagram

- Vision
- Glossary - knowledge detn
- Business rules
- Supplementary (extra) specification

Why Use-cases?

1. UC are the best way of understanding sys. reqmts.
2. It used to depict the interaction btw users & the sys.
3. Uc depicts the following
   - i) who uses the sys.?
   - ii) what actions they perform
   - iii) what are their goals
4. Experts or Requirement donors prefer uc for its features
   - i) Simplicity ii) Sophistication

Use Case Model



Use-case diagram

Business Modeling

Objects, attributes, associations

Borrow Books
1. Enter ID
2. check availability
3.

Use-case text

Scope, goals, actors, features

Vision

Terms, attributes Validation

Glossary

operation
Enter ID()
Check Status()

System

Enter ID
Validate()
Check availability
Show status

System sequence Diagram

Non-functional requirements; quality attributes

Supplementary Specification

Requirements

Design

Three Kinds of Actors

External Actors related to Sys. Under Discussion (SUD)

1. Primary Actor (To find user goals)
2. Supporting Actor (To provide clear picture of external i/f & protocol)
3. Offstage Actor (To ensure all the goals are identified & satisfied)

Common Use-case formats

1. Brief - UC is a one paragraph summary consists of main scenario
2. Casual - UC is an informal paragraph
3. Fully dressed - UC written in detail with supporting sections such as preconds & success guarante

Template and an example of fully dressed Use-case for Next Generation

Use-case selection	Comment
Use-case name	Starts with a Verb
Scope	System under design
Level	User-goal or subfunction
Primary actor	Calls on the system to provide services
Stake holders & Interests	Who cares about the use case & what do they want
Preconditions	What must be true on start & worth & worth telling the reader
Success Guarantee	What must be true on successful completion

Use-case selection	Comment
Main success scenario	An unconditional path scenario of success
Extensions	Alternate scenarios of success or failure
Special Requirements	Related Non-functional Requirements
Technology & Data Variations list	Varying I/o methods & Data formats
Frequency of occurrences	Influences investigation, testing & timing of implementation
Miscellaneous	Such as open issues

For basic ATM system

Use case name : Process transaction

Level : User-goal

Scope : Easy money transaction

Stake holders & Interests : who cares about the use case ? what do they want

Bank client : A person who deposits or withdraw an amount from this or her acct. using touch screen

ATM system : A m/c for making money transactions

Bank Database : Each bank client must be authenticated

Preconditions : what must be true on start & worth

Success Guarantee (Post cdn's) : Saving the transactions (deposit & withdrawal)
Updating the acct. dB of the client
Generating the report or transaction receipt

* Main Success Scenario (Basic flow)

1. Client arrives at ATM with ATM card 1
2. Inserts ATM card 2
3. Enters pin Number 3
4. ATM System checks for the correctness 5
5. performs approval process Validates the User 5
6. Asks the type of transaction 5
7. Enter the " " " 4
8. perform Transaction 5
9. Ejects card 1
10. ATM system represents receipt of transaction 5

* Extensions (Alternative flows)

a) Client Enters wrong pin no

1. client enters ATM card
2. Enter pin number
3. System detects the pin no. is wrong
4. Transaction is denied

* Special Requirements
  1. Touch screen GUI
  2. Client authorization within a short response time
  3. Software available in all regional languages

* Technology & Data Variation list
  1. Use of Authentication Algorithm
  2. Making GUI - ease of use
  3. ~~Store~~ on Setting constraints for pin no.

* Frequency of occurrence
  Nearly continuous

Other Format of Representing Use case

Two column Variation where the interaction bw the actors & the system are described in each column

Eg. Use Case - ATM System

Primary actor
Bank client or customer

Main success Scenario

Actor Action	System Responsibility
1. Client arrives at ATM with ATM card	* ATM System checks for the correctness
	* perform approval process Validates the user
2. Insert ATM card	
3. Enter pin Number	* Asks the type of transaction
4. Enter the type of transaction	* perform transaction
	* Ejects card
	* ATM system represents receipt of transaction

Write Use Cases in an Essential Free style

Essential Style Writing

Write Use cases in an essential style, Keep the user ilf & focus on actor intent

Essential Style

Assume that the Manage users, Use case requires identification & authentication.

Eg. Actor Intention | System Responsibility

1. Administrator identifies self. | 2. Authenticates identity
3. . .

2-column format

1-column format

1. Administrator identifies self
2. System Authentication identity
3. . . .

Concrete Style

User Interface decisions are embedded in the use case text

Eg. 1. Administrator enters ID and password in dialog box
2. System authenticates Administrator
3. System displays the "edit users" window
4. . . .

Guideline: Write Black Box Use Cases

One can specify what the system must do (the behavior or functional requirements) without deciding how it will do it (the design)

Guideline: Take an Actor & Actor-Goal perspective

A set of use-case instances, where each instance is a sequence of action a system performs that yields an observable result of value to a particular actor

An observable result of value to a particular actor

Guideline: how to find Use Cases

Use cases are defined to satisfy the goals of the primary actors.

The Basic procedure is

1. Choose the system Boundary
2. Identify the primary actors
3. Identify the goals for each primary actor

Actor	Goal
Cashier	process sales process rentals handle returns Cash in cash out ...
System Administrator	Add users Modify users delete Users manage security manage system tables ...
Manager	start up Shut down ...
Sales Activity System	Analyze sales & performance data ~~performance~~

Guideline: what tests can help find Useful Use cases?

Which of these is a Valid Use case?

* Negotiate a Supplier contract

Much broader & longer than an EBP.

Could be modeled as a business use case, rather than a system use case

* Handle Returns

Ok with the boss. Seems like a EBP. Size is good

* Log In

Boss not happy if this is all you do all day!

* Move piece on Game Board

Single step - fails the size test

Tests Used to find Useful Use cases

1. **Boss Test**

"what the Use case does" in the same way boss asks "what have you been doing all day"

If the reply is some useful action then the tests pass

2. **EBP Test**

Elementary Business protocol

UC that depicts the action performed by a person in response to a business events are tested using EBP test

If it yields a measurable business value, then the uc pass/succeeds EBP test.

3. Size Test

Whether it has single step or multiple steps under it.

- Full dressed use-cases pass size test
- Eg. Enter amount - Fails in size test
  Process transaction - Pass size test

Relating Use Cases

Use Cases can be related to each other using include relationship, extend relationship or generalize relationship

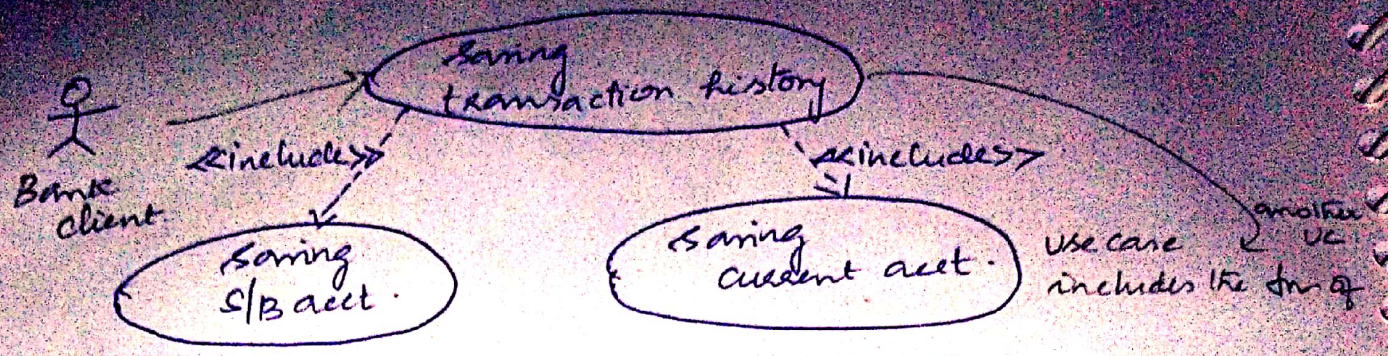
Use case relationship results in improvement of comm & comprehension of usecases & Reduce duplication of text also to manage Use case doc

Guideline: Avoid Agonizing over use case relationships
↓ To struggle
Figuring out writing use case text

Include relationship

Include is a directed relationship btn two use cases, implying that the behaviour of the included use case is inserted into the behaviour of including use case.

* 1st use case depends on the outcome of include use case
* Notation <<include>>
* An include relationship is a relationship in which one use case includes the functionality of another use case



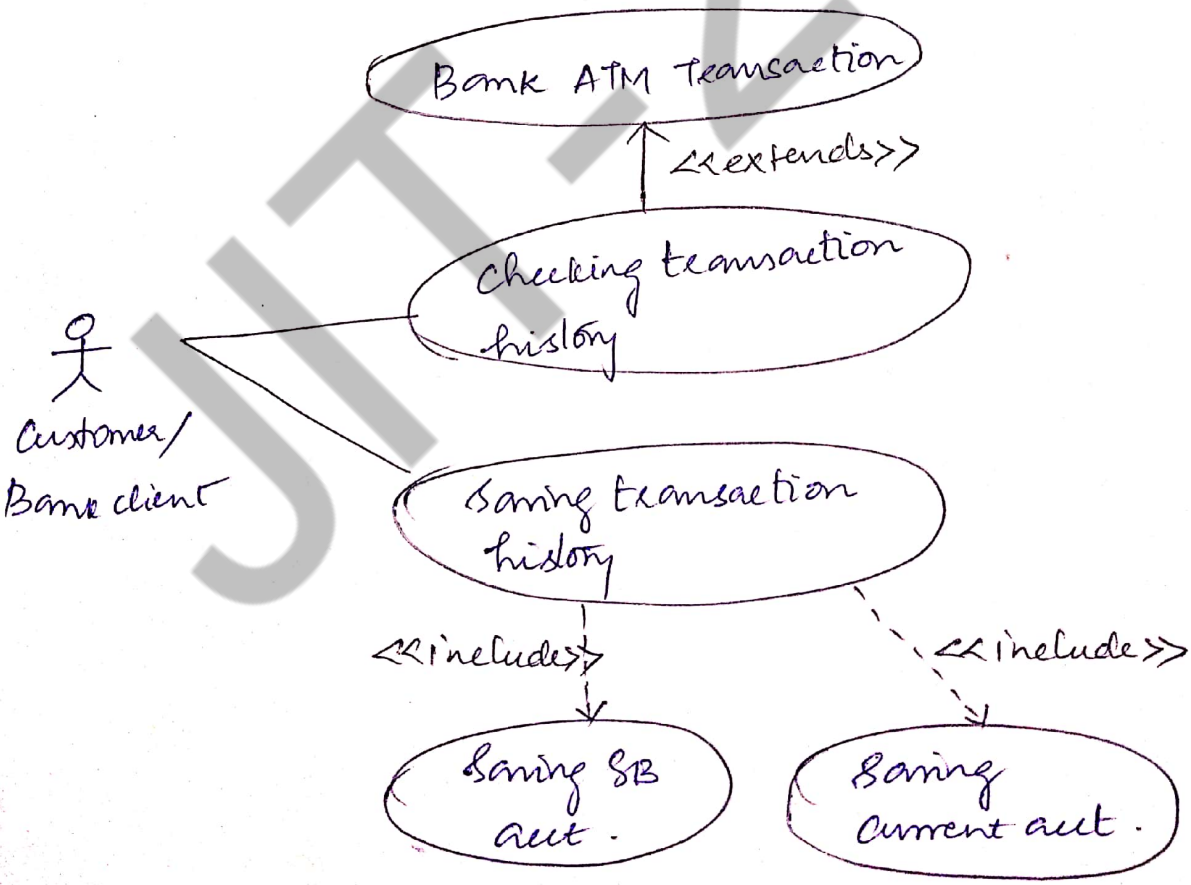
Use case diagram with include relationship

Concrete Use case

An use case initiated by an actor & performs the entire behavior desired by the actor

Abstract Use case

Abstract never instantiated by itself.
It is a subfn. Use case that is part of another use case

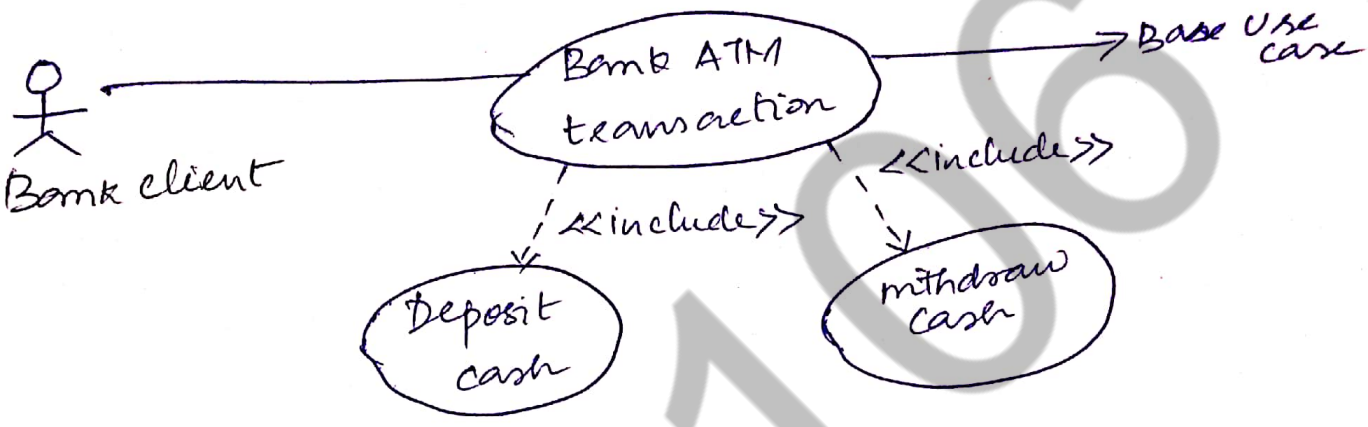


Abstract Use case depicted by include relationship

Abstract Usecase can be depicted as <<include>> in include relationship & <<extends>> in extend relationship

Base Use Case

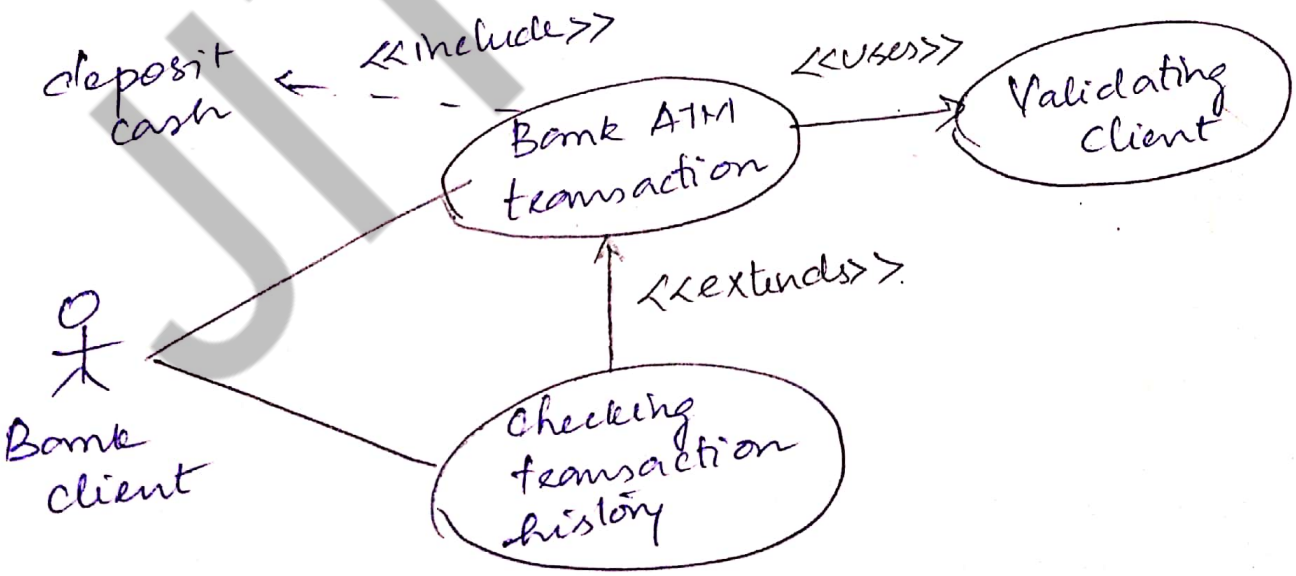
It includes another usecase or that is extended or specialized by another Use case is called a Base Use case



Addition Use Case

It is an inclusion, extension or specialization is called an addition Use case.

Eg: ATM system with extends, include & uses relationship

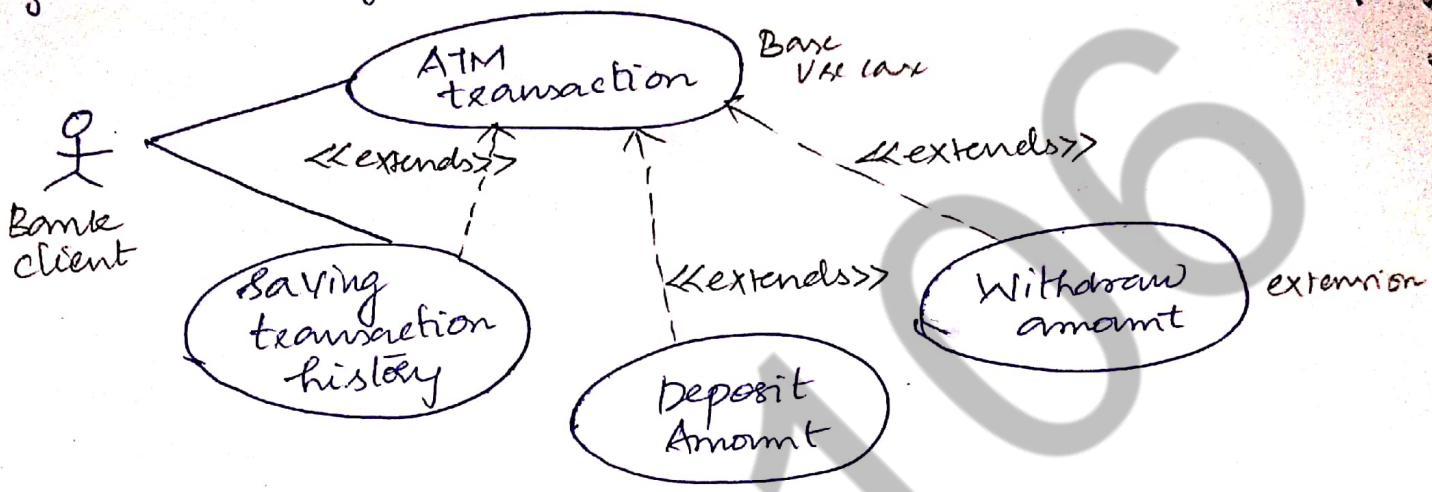


Extend relationship

It is to specify that one use case (extension) extends the behaviour of another use case (base use case)

Notation <<extends>>

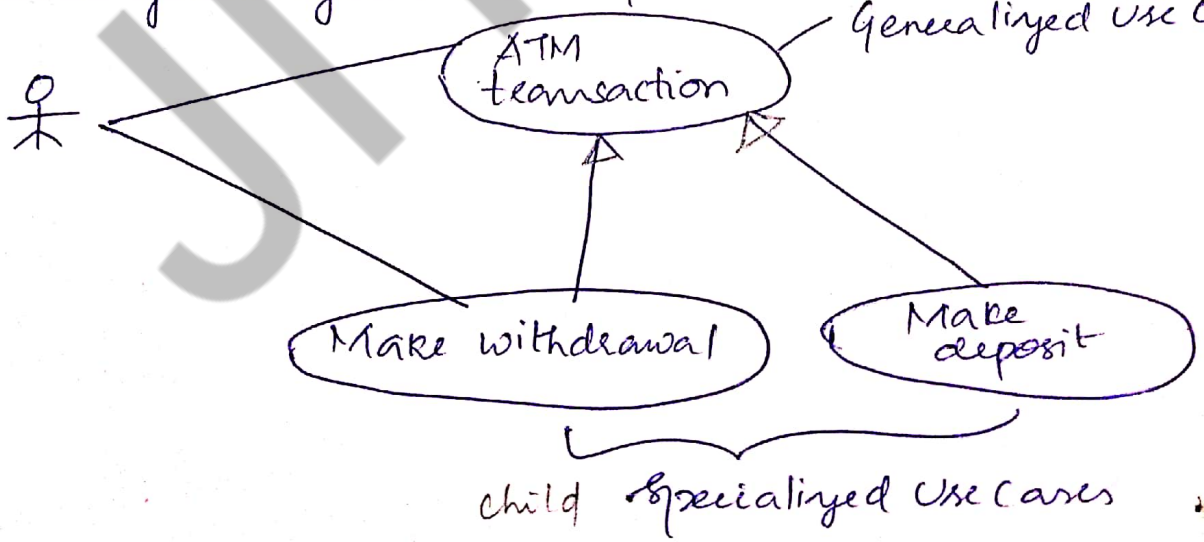
Eg: Use case Diagram for an ATM system with extends relationship



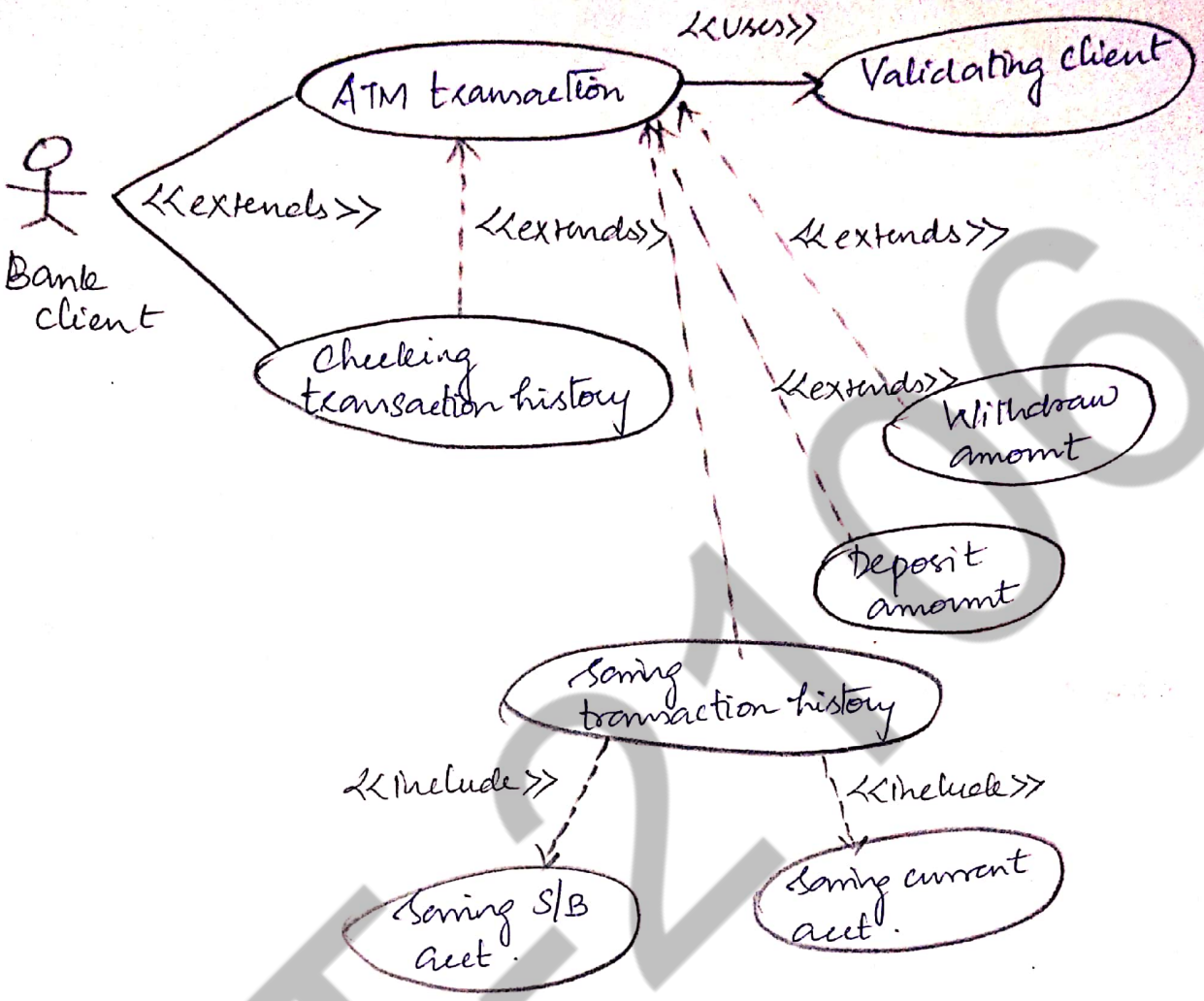
Generalize Relationship

It is a relationship in which one model element (Child) is based on another model element (parent)

Notation is a solid line ending in a hollow triangle drawn from the specialized use case (child use cases) to more generalized use case (parent use case)



Use case Diagram with Generalized & Specialized Use case for an ATM system



Usecase Diagram for an ATM system with extends and include relationship

When to use Use Cases

* They are an essential tool in requirements capture & in planning & controlling an iterative project
* Capturing use cases is one of the primary tasks of the elaboration phase
* Most of your use cases will be generated during that phase of the project. Keep an eye out for them at all times.

* Every usecase is a potential requirement, & until you have captured a requirement, you can't plan to deal with it.

* I have also found that conceptual modeling with users helps uncover use cases. So, I tend to do use cases & conceptual modeling at the same time.

* That usecases represent an external view of the system. Don't expect any correlations btw use cases & the classes inside the system