The Shirpur Education Society's

R. C. Patel College of Engineering & Polytechnic, Shirpur

Department of Computer Engineering

Course Title - Software Engineering

Course Code - 315323

Programme Name - Computer Engineering

Semester - Third

Unit - II Software Requirement Engineering

Total Marks:14

Prepared By: Mrs.Pratiksha S. Patil

Subject : SEN

Unit II. Software Requirement Engg. 14 mks

2.1 Software Engg core principles :-

Principle 1 :- Reason it all exists. Provide value to User.

① The S/W system exists for one important reason to provide value to its user.

② All decision of S/W system should made with time in mind.

③ Before specifying the problem the requirement & the specifications have to be laid down.

Principle 2 :- keep it simple, stupid

① all S/W design should be as simple as possible

② The Team & the design used for development of the project should be kept simple & easily understandable.

Principle 3 :- Maintain vision

→ Clear vision is essential & important to the success of a S/W project.

→ The developer should hold the vision & ensure the successful development of S/W.

Principle 4 :- What you produce, other will consume.

→ always specify, design & implement knowing that someone else is going to understand what is being developed.

→ customers for the product development is very large.

→ Design the data structure & code with concern that the product has to be implemented &

maintained by the end users.

Principle 5:- Be open to the future:-

→ The system designed today should be adaptable to the development & changes in the future at a low cost.

→ There should not be much changes to the software to adopt to the new changes in future development.

principle 6:- plan ahead for reuse:-

→ Reuse saves efforts & time.

→ Design & specifications should be developed in such a way that they can be reused.

→ code & design should be well documented for use in future.

principle 7: Think !

→ Placing clear, complete thoughts before action almost always produces better results.

→ Before designing & implementation a proper thought should be to the end result.

2.2 Software practices:-

→ Software Engg. patice is broad array of principles, concept, methods & tools that you must consider as S/W is planned & developed.

(*) Essence of practice:-

- Understand the problem (communication & analysis)
  - Who has a stake in the solution to the problem?
  - What are the unknowns (data, function, behavior)
  - can the problem be represented graphically ?

- Plan & solution (planning, modeling & S/W design)
  - have you seen similar problem like this before ?
  - can sub problems be defined & are solutions available for the sub problems ?

- Carry out plan (construction, code generation)
  - Does the solution confort to the plan ?
  - is each component of the solution correct ?

- Examine the Result of Accuracy (testing & quality Assurance
  - is it possible to test each component of the solution ?

(*) Communication Practices :-

→ communication is also called requirement elicitation.
→ S/W Requirement analysis always begin with comm^n bet^n two or more parties.
→ A customer has a problem that may be easily controlled to computer based sol^n.
→ A developer responds to the customer's to the request for help.

⊛ Principles of communication :-

Principle 1 :- Listen carefully

- try to focus on the speaker's words, rather than formulating your response to those words.
- Ask for clarification if something is unclear. avoid constant interruptions.

Principle 2 :- Prepare before you communicate :-

- Spend time to understand the problem before you meet with others.
- Perform some research to understand business domain.
- Prepare an agenda in advance of the meeting.

Principle 3 :- Someone should facili the activity.

- Every communication meeting should have a leader
  (i) to keep conversation moving in productive direction.
  (ii) mediate any conflict that does occure.
  (iii) to ensure than other principles are followed.

Principle 4 :- face to face communication is best.

- it is better if some documents related to that particular discussion is given.

Principle 5 :- Take notes & document decision

- Write down all important points & decisions
- make notes of all discussed points in meeting.

principle 6:- Strive for collaboration
- team work collaboration
- collective knowledge of members is united.

principle 7:- stay focused, modularize your discussion.
- facilitator should keep conversation modular, leaving one topic only after it has been resolved.

principle 8:- If something is unclear, draw picture
- verbal comm^n
- sketch or drawing

principle 9:- (A) once you agree to something, move on
(B) If you can't agree to something, move on.
(C) If a feature / function is unclear & can not be clarified at the moment, move on.

principle 10:- Negotiation is not a contest or game. it works best when both parties win.
- negotiation will demand compromise from all parties.

* Planning practices :-
→ using this activity s/w define a road map.

* principles of planning :-
1) principle 1:- understand the scope of the project.
- some provides software team with diagram.

2) Principle 2:- Involve stakeholders in the planning activity.
- Stakeholder defines priorities & establish project constraints.

3) Principle 3:- Recognize that planning is iterative.
- project plan is never engraved in stone.

4) principle 4:- Estimate based on what you know.
- to provide an indication of effort, cost & task duration.
- estimates will be unreliable.

5) principle 5:- consider risk as you define the plan.
- identified risks that have high impact & high probability contingency planning is necessary.

6) principle 6:- Be realistic.
- Human don't work 100% of every day
- noise always enter into any human comm°.
- change will come.

7) principle 7:- Adjust Granularity as you define the plan.
- granularity refers to the level of details that is introduced as a project plan is developed.
- High granularity plan provides significant work task detail that is planned over short time periods.
- low granularity plan provides broader work task that are planned over longer time periods.

8) principle 8:- Define how you intend to ensure quality.
- plan should identify how s/w intends to ensure quality.
- technical reviews are conducted, they should be scheduled.

9) Principle 9 :- Describe how you intend to accomodate change
- identify how changes are to be accommodated as software work proceeds.

10) Principle 10 :- Track the plan frequently & make adjustments as required.
- S/W projects fall behind schedule one day a time.
- track progress an a daily basis.

* Modelling Practices :-

- S/W Engg. begins with a series of modeling task.
- complete specification of requirement.
- design representation for s/w built.
- This mode is technical representation of system.
- two types models
  I) Analysis II) Design

① Analysis model :- They represent the users requirements in 3 domain.
  (a) information domain
  (b) function domain
  (c) Behavioral domain

② Design model :- Represents characteristics of the s/w.
  (a) The architecture.
  (b) The user interface
  (c) component-levels details.

- **Principles of Analysis Modeling :-**

Principle ① The information domain of a problem must be represented & understood.

- includes the data that flows into system.
- the data that flow out of the system.
- the data that collects & organize persistent data objects.

Principle ② The function that the s/w performs must be defined.

- s/w functions provide direct benefit to end users & also provide internal support for those features that are user visible.
- controls over internal s/w processing.

Principle ③ Behavior of the s/w must be represented.

- interaction with external environment.
- i/p by user, control data provided by external system or monitoring data collected over network all cause the s/w to behave in specific way.

Principle ④ The models that depict information, function & behavior must be partitioned in a manner that uncovers detail in layered fashion.

- complex problem is divided into sub problems.

Peinciple (5) The analysis task should move feom essential infoemation towaeds implementation details.

- implementation detail indicates how the essence will be implemented.

• Design Modeling Peinciples :-

Peinciple (1) Design should be teaceable to an analysis model.

- teanslate infoemation i/p feom analysis model into aechitectuee.

Peinciple (2) Always considee the aechitectuee of the system to the built

- s/w aechitectuee skeleton of system to be built.
- it affects inteeaces, data steuctuee, peogeam conteol flow & behavioe.

Peinciple (3) Design of data is as impoetant as design of peocessing functions

- Data design is essential element.
- it helps to simplify peogeam flow, makes design & implementation of s/w components.

Peinciple (4) Inteeface must be designed with caee.

- makes integeation easiee & assists the testee in validating component functions.

Peinciple (5) Usee inteeface design should be tuned to the needs of the end usee.

- pooe inteeface leads to peeception that s/w is bad.

Peinciple ⑥: component level design should be functionally independent.

- functional independence is a measuec of the single mindness of s/w component.
- it focus on one & only one function.

Peinciple ⑦: Component should be loosely coupled to one another & to the exteenal envizonment.

- coupling is achived in many ways - via a component inteeface, by messaging theough data.
- as the level of coupling incteases, the eeeoe peopogation also incease cause maintainability of s/w deceeases.

Peinciple ⑧ Design tepeesentation should be easily undeestandable.

- help to geneeate code.
- to test s/w
- maintain s/w in the fu-tuee.

Peinciple ⑨ The design should be developed i.teatively, With each iteeaction the designee should s-teive foe geeatee simplicity.

- designing occues itceatively.

* Construction Practices :- (coding)

→ Software coding - set of coding & testing task that lead to operational S/W that is for end user.

→ includes Programming languages, Programming styles & methods.

Preparation principles: Before you Write one line of code, Be sure you.

1) Understand of the Problem you are trying to solve.
2) Understand basic design, Principles & concepts.
3) Pick a Programming language that meets need of S/W & environment in which S/W will operate.
4) Select a Programming environment that provides tools that will makes you work simpler.
5) Create a set of units that will be applied once the component code is completed.

Coding principles: As you begin Writing code, Be sure you.

1) Constrain your algorithms by following structured Programming practices.
2) Consider the use of pair Programming.
3) Select data structures that will meet the needs of the design.
4) Understand S/W architecture & create interfaces that are consistent with it.
5) keep conditional logic as simple as possible.
6) Create nested loops in a way that makes them easily testable.
7) Select meaningful variable names & follow other local coading standards.

8) Write code that is self-documenting.

9) Create visual layout that ids understanding.

Validating Principles: After you have completed your first coding pass, be sure you.

1) Conduct a code Walkthrough When appropriate.

2) Perform unit tests & Correct errors When you have uncovered.

3) Refactor the code.

* Software Deployment :-

Principle ① Customer expectations for the S/W must be managed.

- Customer always Wants more than he stated earlier as the requirements.
- Customer satisfaction is important.
- At time of delivery developer must have skills to manage Customer expectations.

Principle ② A complete delivery package should be assembled & tested.

- CD-Rom or other media containing all executable S/W, support data file, documents.

Principle ③ A Support regime must be established Before the S/W is delivered.

- Support should be planned.
- Support materials should be prepared.
- Record keeping mechanisms should be established.

Peinciple ④ Appeopeiate insteuctional mateeials must be peovided to end Usees.

- teaining aids should be peepaeed, guidlines should be peovided.

Peinciple ⑤ Buggy s/w should be fixed fiest, deliveeed latee.

- Buggy softwaee should not be deliveeed.
- avoid mistakes of deliveeing defective softwaee.

2.3 Requiteement Engineeeing :-

→ Requiteement Engg. is a majoe s/w Engg. action that begins dueing the communication activity & continues into the modeling activity.

→ It must be adapted to the needs of peocess, peoject, the peoduct & people doing woek.

→ Befoee doing any technical woek, it is good idea to apply a set of teequiteement engg. tasks.

* Need of Requiteement Engg.

→ Requiteements Engg. tools assist in teequiteement gatheeing, teequiteement modeling, teequiteements management, & teequiteement validation.

Sub task included Requiteement Engg.

1. Inception
2. Elicitation
3. Elaboeation
4. Negotiation
5. Specification
6. Validation
7. Requiteements management.

1. Inception →

1) means begining
2) it usually said that, "Well beginning is half done".
3) it is always problematic to developer that from where to start.
4) customer & developer meet & they decide scope & nature of the problem.

The aim is
a) To have the basic understanding of problem.
b) To know the people who will use the s/w.
c) To know the exact nature of problem.

2. Elicitation →

1) The process of getting the set of requirement from all the stakeholder (customer, user/management)
2) it means to "draw out the truth or reply from anybody.
3) is a task helps customer to define what is required.
   a) problem of scope
   b) problem of understanding
   c) problem of volatility.

3. Elaboration →

1) information obtained from the customer during inception & elicitation is expanded & refined during elaboration.

2) This task focuses on developing a defined requirement model that identifies various aspects of S/W function behavior & information.

3) it describe how the end users interact with system.

4) Relationship & collaboration bet^n classes are identified & variety of diagrams are produced.

4. Negotiation →

1) This phase involve the negotiation between users & developers.

2) Users always expect lot of thing from system at lower cost.

3) This phase play important role bet^n users & developers so they can proceed towards implementation of a system.

5. Specification →

1) Specification can be a re-written document, a set of graphical models, mathematical model & collection of usage scenario, prototype or any combinations of these.

2) A requirement specification
   - (a) Writting document
   - (b) Graphical model
   - (c) formal model
   - (d) collection of use cases (a set of prototype).

6. Validation :-

1) The work products produced as a consequence of requirements engg. are assessed for quality during validation step.

2) Requirement validation examines the specification to ensure that all s/w requirements have been stated unambiguously.

7. Requirement Management :-

The set of activities that assist in identifying, controlling & tracking requirement & changes to requirement during the process of implementations are called as requirement management.

* Types of Requirement (functional, product, organizational, external, Eliciting Requirement)

(1) Functional Requirement :-

1) Describe functionality of system services.
2) Depends on the type of s/w, expected users & the type of system where s/w is used.
3) functional user requirements may be high-level statement of what the system should do.

(2) product Requirement :-

1) Describe the qualities of attributes of product.
2) Define how the product should perform & feel.
3) Focus on usability, design & performance.

(3) Organizational Requirements :-

1) Define the needs of constraints of the organization
2) include business goals, budgets & compliance with rules.
3) Address the integration of the system within the organization.

(4) External Requirements :-

These are affected by factors outside the organization, like legal Regulation, industry standards & market needs.

* Eliciting Requirements :-

1) Eliciting refers to the process of collecting requirements from all relevant sources, such as stakeholders, users, & systems.

2) Techniques
   - interviews with users, business analysts, & stakeholders.
   - Surveys & questionnaires to gather broad i/p.
3) Workshops to encourage collaboration.
4) Prototyping to clarify specific functionalities.
5) Observation (in the case of existing systems)

* Developing Use-cases:-

1) Use case diagram is representation of a user's interaction with the system that shows the relationship bet" the user & the different use cases in which the user is involved.

2) A case cases diagram shows a set of use cases & actors & their relationships.

3) static design view of a system.

4) The use cases are represented by either circles/ ellipses.

5) use cases represent only functional requirements of a system.

* Purpose of use-case diagram
  - Specify the context of a system.
  - Capture the requirements of a system.
  - Validate a system architecture.
  - Developed by analysts together with domain experts.

* Notation used in use-case diagram.

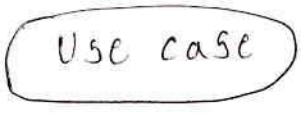
1) Actor :-

i) Actor is someone interacts with use case

ii) An actor represents a role that human, h/w device or another system plays.

iii) named by noun.

2) Use case



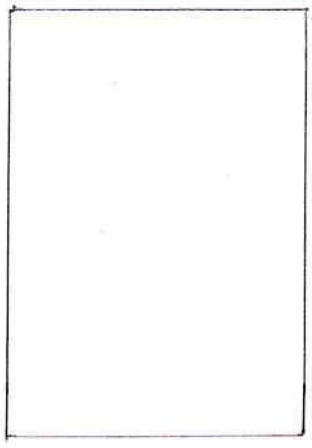
i) use case is the description of set of sequences of action.

ii) it is graphically represented as an ellipse & labeled with the name of the use case.

3) Communication link :-

The participation of an actor in a use case is shown by connecting actor to a use case by a solid link.

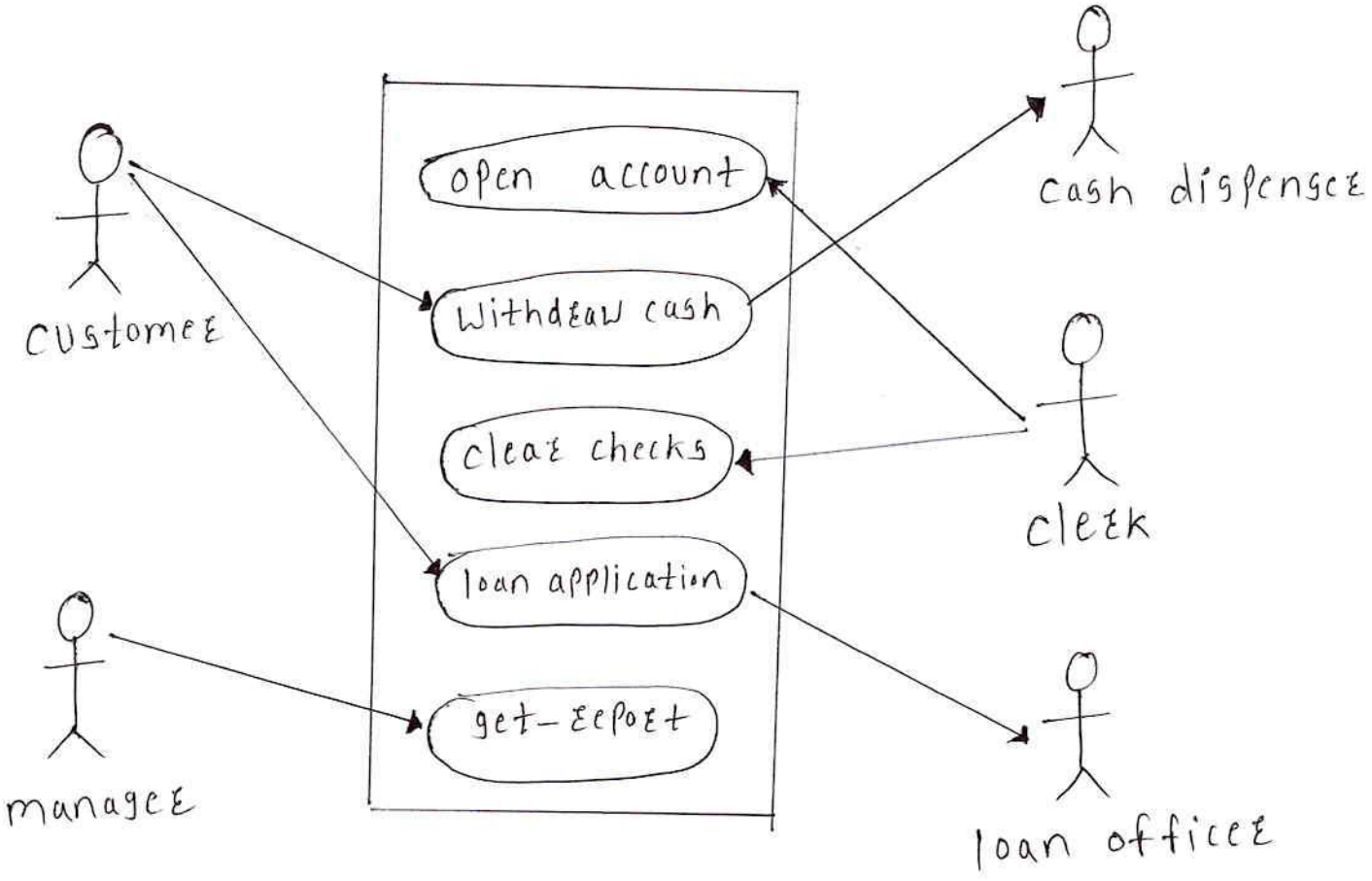
4) System Boundary :-



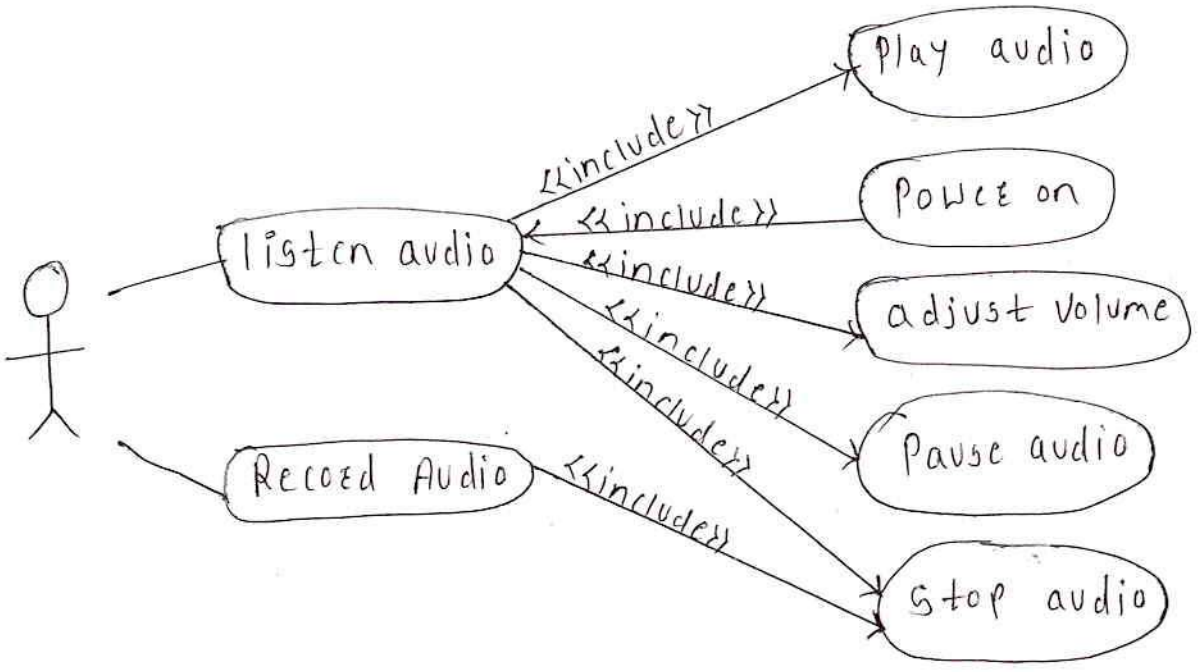
i) The system boundary is potentially the entire system as defined in requirement document.

ii) for large & complex system, each modules may be the system boundary.

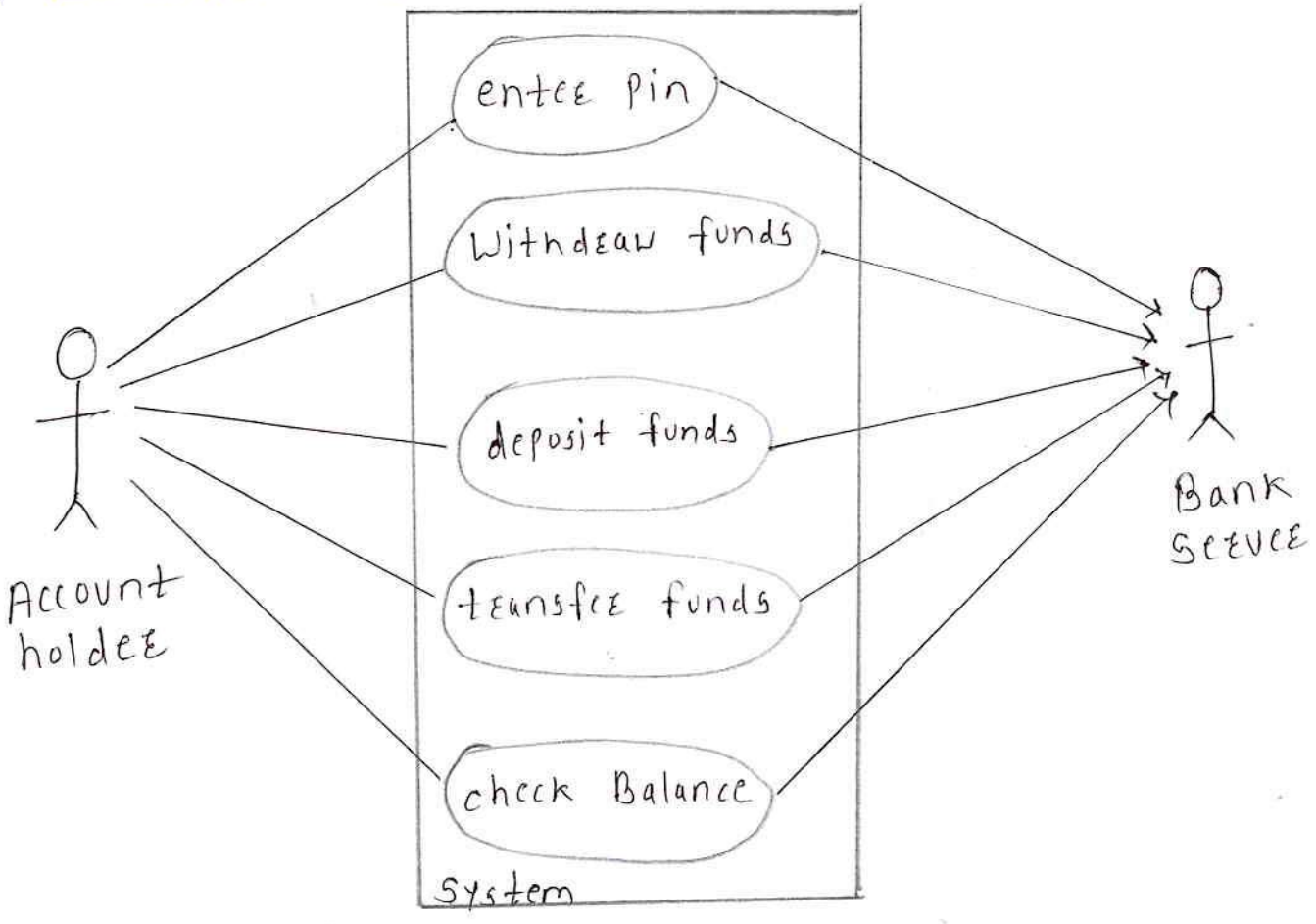
eg. Use case diagram for Bank Management system.



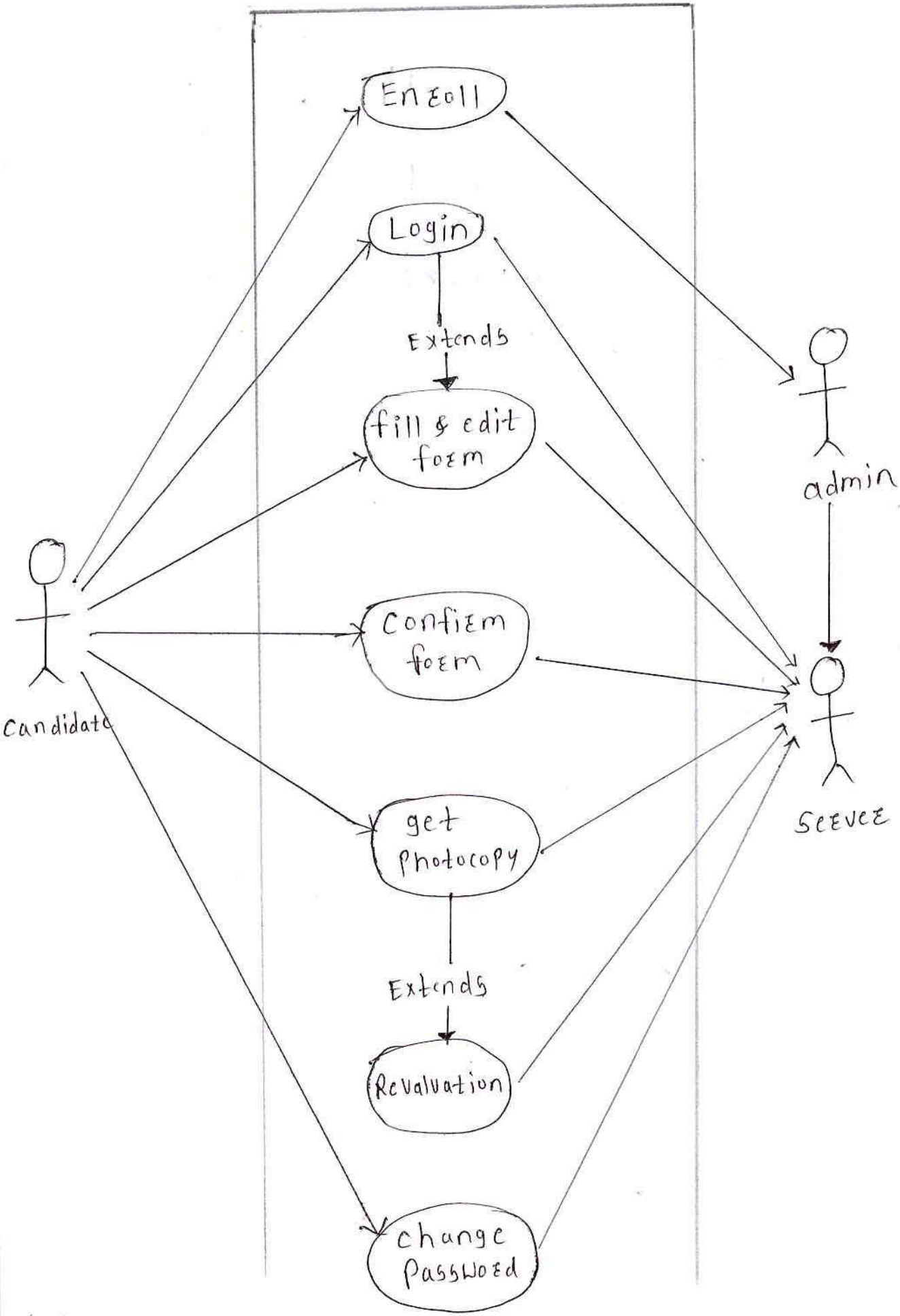
* Draw a use case diagram for music system.



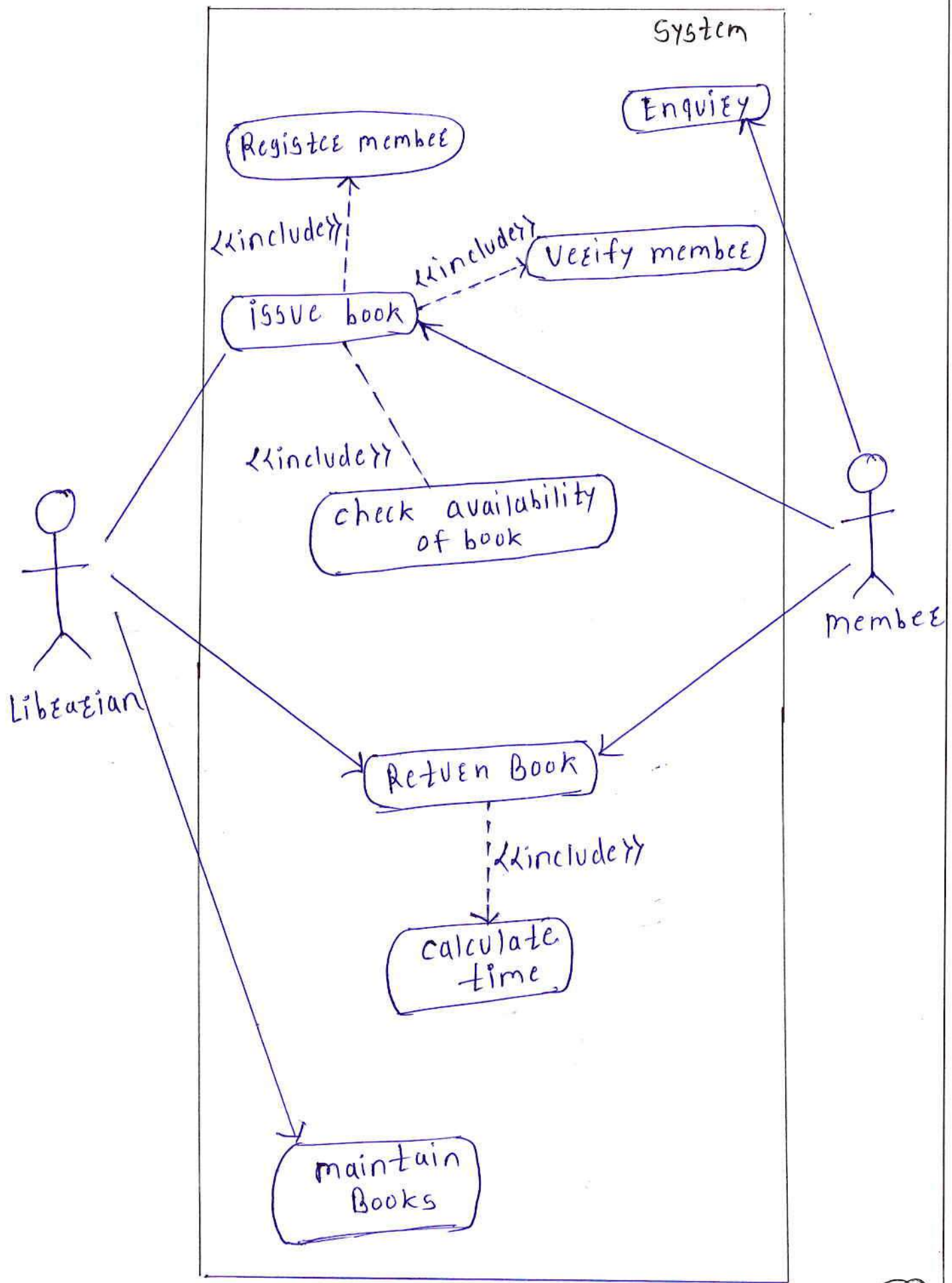
* Use case diagram for ATM machine.



* Draw use case diagram for taking photocopy of answer book from MSBTE Website.



(*) DEAL Use-case diagram for library management system.



* Building Requirement Models

This is the process of capturing & representing requirements using structured. models. These models helps stakeholders visualize, understand & communicate the system or business requirements.

Common requirement models include :-

- use case diagram → interaction bet^n user & system.
- activity diagram → detail workflow / process.
- DFD → show how data moves through a system.
- Entity - Relationship Diagram (ERDs) → model database structure
- user stories / EPics → used in agile methodology.
- class diagram / state Diagram → part of UML for object oriented system.

* Negotiation :- sometime different people wants different things in a project. Negotiation means taking with team to
  - understand what everyone wants.
  - set priorities.
  - make fair decision when people don't agree.
  - facilitating workshop or meetings bet^n business & technical stakeholders.

* Validation :- validation ensures that the documented requirements truly reflect the needs & expectations of stakeholder.

methods include :-

- Review & Walk through :- Stakcholdee examine Eequize-ments documented oE models.
- Peototype oE mockup :- easily uesions of system to gathee feedback.
- Test case Deivation :- CEeating tests based on Eequieement to ensuEe coveEage.
- Teaceability Analysis :- ensuEing each Eequieement maps to a business need.

* Softwaee Aequieement specification (SRS) :-

i) SRS is document that cleaely desceibe "what a S/W system should do". It is ceeated beoEe the actual develepment staets.

ii) It includes a set of use cases desceibe all of the inteeactions that the useEs will have with the S/W.

iii) SRS contains functional Eequiements & non-functional Eequiements.

iv) The puEpose of SRS document is peoviding a detailed oveeview of S/W peoduct, its Paametees & goals.

v) SRS document desceibes the peojects taeget audience & its useE inteeface, H/W & S/W Eequieements.

need of SRS :-

1) clear communication bet^n the client & developers.
2) Avoid confusion or misunderstanding later.
3) acts as a guide for developers, testers & designers.
4) Helps in estimating cost, time & effort.
5) used for validation – check final product.
6) reduce the development effort.

* Format of an SRS :-

1. Introduction
   - purpose of the s/w
   - Who will use it.
   - Defination.
2. overall Desciption
   - What the s/w will do
   - Who the users are
   - What kind of system it will run on (H/W or S/W)
3. Functional Requirements
   - list of feature & What each should do
4. Non-functional Requirements
   - Qualities of the system
     - speed
     - usuability
     - security
     - Reliability
5. constraints
   - limitations like Programming language, databases etc.

6. other Requirements

- appendix A : Glossary
- appendix B : Analysis models
- appendix C : Issue list

* characteristics of SRS :-

A good SRS should be

1) correct → describe the real needs
2) Complete → Has all important details
3) clear → easy to read & understand.
4) consistent → no contradictions.
5) Verifiable → you can check if its met
6) modifiable → easy to update / change
7) Traceable → each requirement can be tracked back to a need or goal.

Subject :- SES.

Unit 2.

14 mks

Software Requirement Engg.

Question Bank

2 Marks

1. Define software engineering (S19, S23)
2. State the need of SRS. (S19, W23, W24)
3. Define Software Requirement Specification (SRS). (W19)
4. List any four planning principles. (S22)
5. List any four characteristics of SRS. (S23)

4 mks.

1. State the need SRS & enlist its characteristics. (S22)
2. Describe any four principles of communication for software engineering. (S19, S22, W23)
3. Describe any four core principles of Software engineering Practices. (S23)
4. Describe any four software coding principles. (S23)
5. List & explain any four core principles of Software engineering. (S19, S22)
6. Describe communication practices (any four). (W24)
7. Describe four principles of good planning. (W19, S24)

P.T.O

6 MKS

(1) Enlist Requirement Gathering & Analysis for Web-based Project for Registering candidates for contest. (any six points) (S19, S24)

(2) Explain six functions of Requirement Engineering Process (W19)

(3) State & describe any six communication Principles. (S19, S22, W23)