Assignment-2

① Explain Requirement analysis in detail

Requirement 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. The requirements modeling action results in one or more of the following types of models:

- Scenario-based models of requirements from the point of view of various system "actors"
- Data models that depict the information domain for the problem.
- Class-oriented models that represent object oriented classes
- Flow-oriented models that represent the functional elements of the system and how they transform data as it moves through the system.
- Behavioral models that depict how the software behaves as a consequence of external events

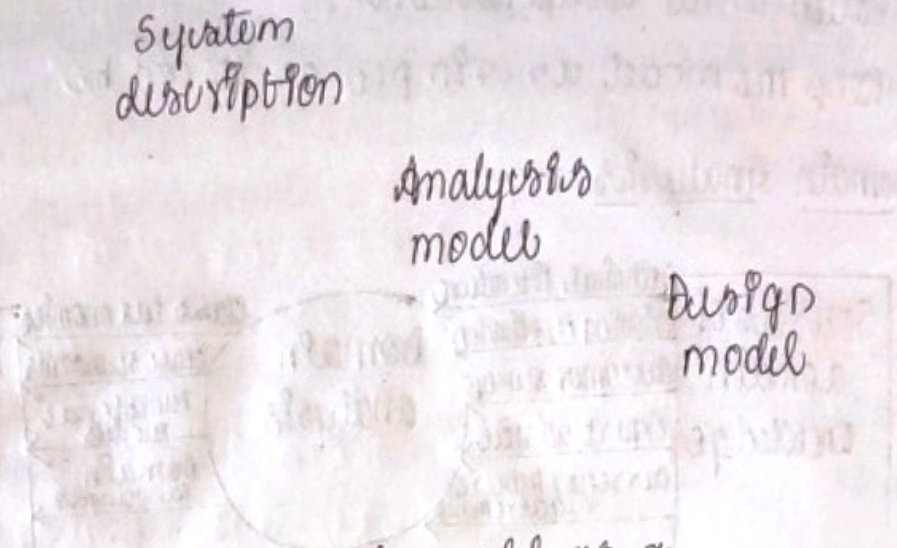


Fig: The requirements model as a bridge between the system description and design model

Analysis Rules of Thumb.

- The model should focus on requirements that are visible within the problem or business domain.
- Model should add to an overall understanding of software requirements and provide insight into the information domain, function and behaviour of the system.
- Delay consideration of infrastructure and other nonfunctional models until design
- Minimize coupling

- Be certain that the requirements model provides value to all stakeholders.
- Keep the model as simple as it can be

Domain Analysis

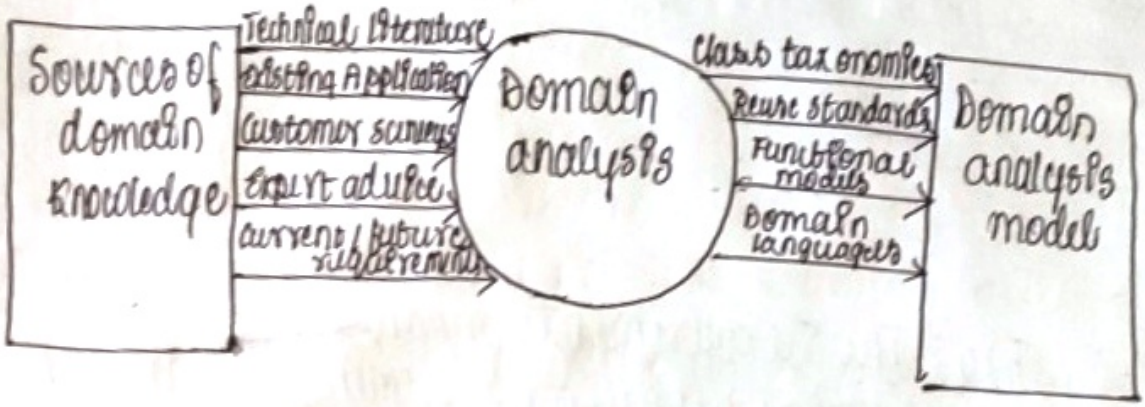


Fig: Input and Output for domain analysis

- Domain analysis doesn't look at a specific application but rather at the domain in which the application resides.
- The intent is to identify common problem solving elements that are applicable to all applications within the domain.
- Sources of knowledge are surveyed in an attempt to identify the objects that can be reused across the domain.

② Explain CRC model with diagram.

Class Responsibility collaborator modelling provides a simple means for identifying and organizing the classes that are relevant to system or product requirements.

Classes:

- Entity classes also called model or business classes are extracted directly from the statement of problem. These classes represent things that are to be stored in a database.
- Boundary classes are used to create the interface that the user sees and interacts with as the software is used. Entity objects contain information.
- Controller classes manage "unit of work" from start to finish.
  (1) Creation or update of entity objects
  (2) Instantiation of boundary objects as they obtain information from entity objects
  (3) Complex communication between sets of objects
  (4) Validation of data communicated between objects or between the user and the applicati

Responsibilities

1. System intelligence should be distributed across classes to best address the needs of the problem
2. Each responsibility should be stated as generally as possible
3. Information and the behaviour related to it should reside within the same class.
4. Information about one thing should be localized with a single class not distributed across multiple classes
5. Responsibilities should be shared among related classes when appropriate.

Class: FloorPlan	
Description	
Responsibility	Collaborator:
Defines floor plan name/type	
Manages floor plan positioning	
Scales floor plan for display	
Scales floor plan for display	
Incorporates walls, doors and windows	wall
Shows position of video cameras	camera

Fig: CRC Model Index card.

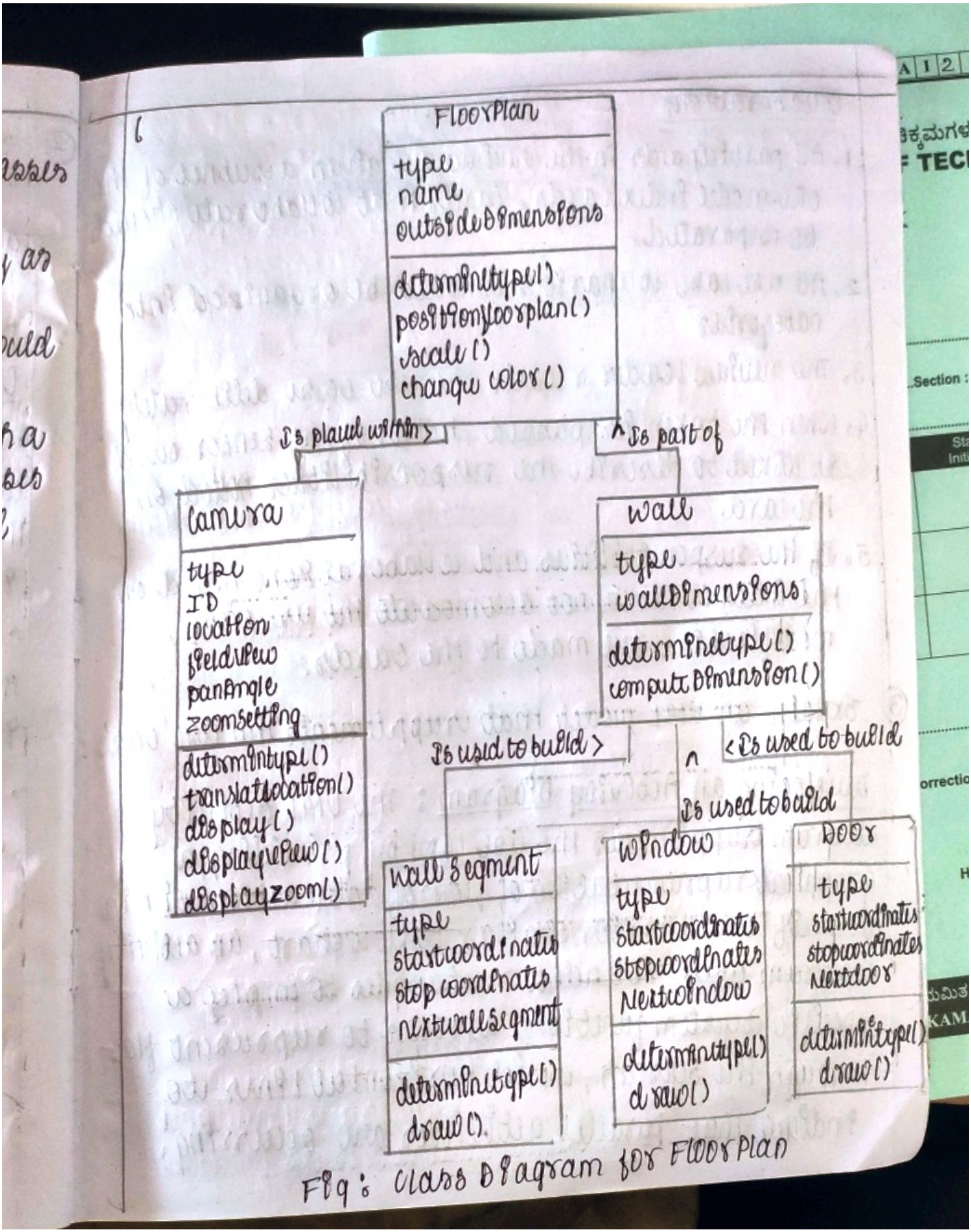


Fig: Class Diagram for FloorPlan

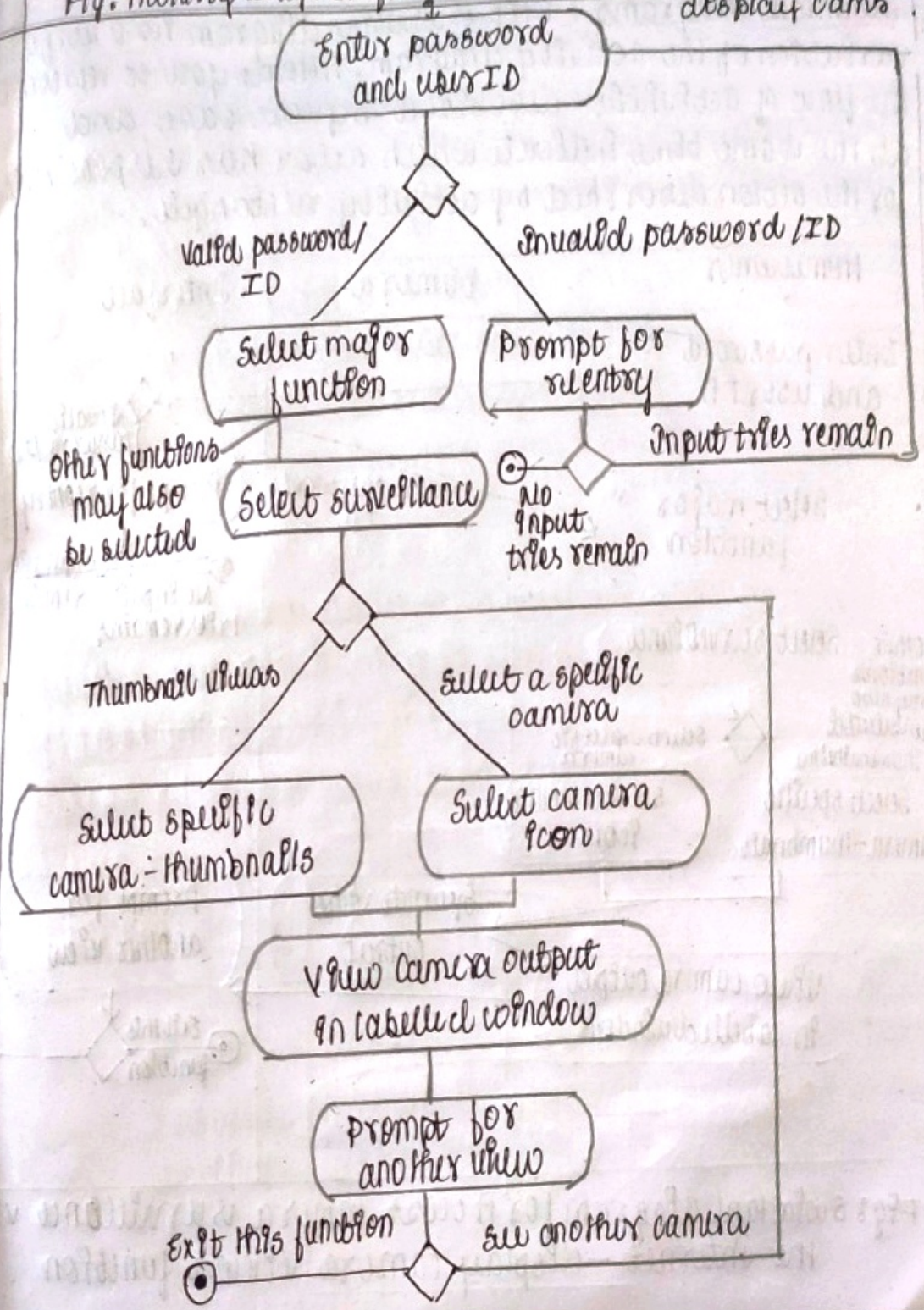
Collaborations

1. All participants in the review are given a subset of the CRC model index cards. Cards that collaborate should be separated.
2. All use case scenarios should be organized into categories
3. The review leader reads the use case deliberately.
4. When the token is passed, holder of the sensor card is asked to describe the responsibilities noted on the card
5. If the responsibilities and collaborations noted on the index cards cannot accomodate the use case, modifications are made to the cards.

(3) Explain an UML model that supplements the use case.

Developing an Activity Diagram: The UML activity diagram supplements the use case by providing a graphical representation of flow of interaction within a specific scenario. Similar to flowchart, an activity diagram uses rounded rectangles to imply a specific system function, arrows to represent flow through the system, solid horizontal lines to indicate that parallel activities are occurring.

Fig: Activity diagram for Access Camera surveillance via Internet



Swimlane diagrams : UML swimlane diagram is a useful variation of the activity diagram. Allows you to represent the flow of activities described by use case and at the same time indicate which actor has responsibility for the action described by activity rectangle.

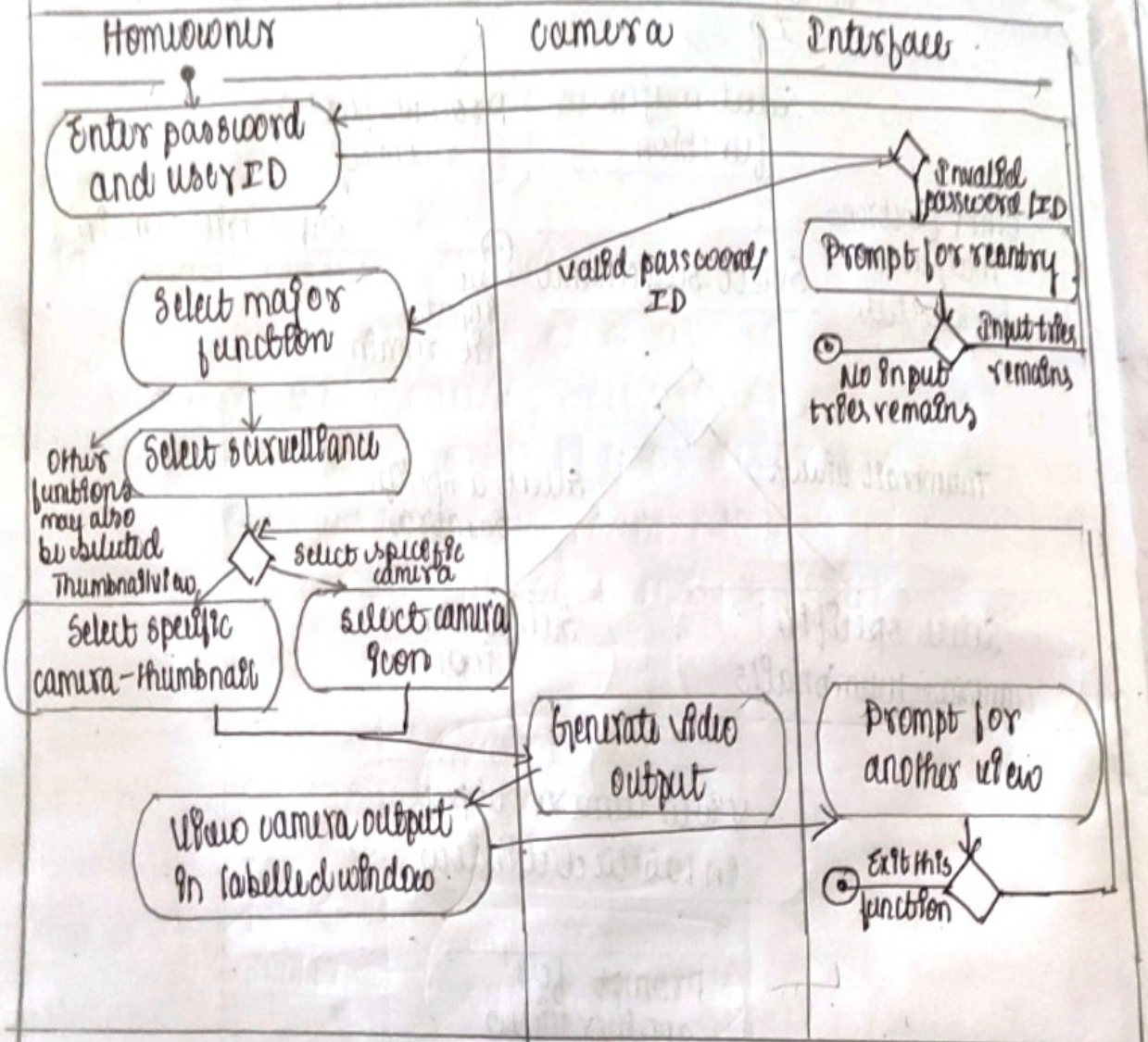
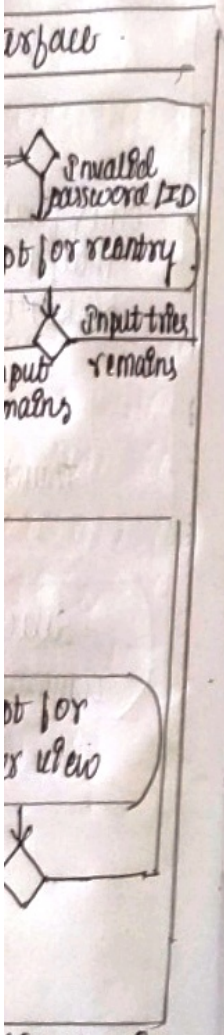


Fig: Swimlane diagram for Access camera surveillance via the Internet - display camera views function.

m is a useful
ow to represent
se and
responsibility
cle.



(14) Explain Use cases in detail.

Creating a Preliminary Use case:

Use case describes a specific usage scenario in straightforward language from the point of view of defined actor.

(1) What to write about
(2) How much to write about it
(3) How detailed to make your description
(4) How to organize the description?

What to write about?

The first two requirements engineering tasks – inception and elicitation provide you with the information you will need to begin writing use cases.

Refining a Preliminary Use case

A description of alternative interactions is essential for complete understanding of function that is being described by use case.

- Can the actor take some action at this point?
- Is it possible that the actor will encounter some error condition at this point? If so what it be?

- Is it possible that the actor will encounter some other behaviour at this point. If so, what it be?
- Are there cases in which "validation function" occurs during this use case?
- Are there cases in which a supporting function occurs will fail to respond appropriately?
- Can poor system performance result in unexpected or improper user actions?

Writing a Formal Use Case:

Informal use cases presented are sometimes sufficient for requirements modeling. However, when a usecase involves a critical activity or describes a complex set of steps with significant number of exceptions a more formal approach may be desirable.

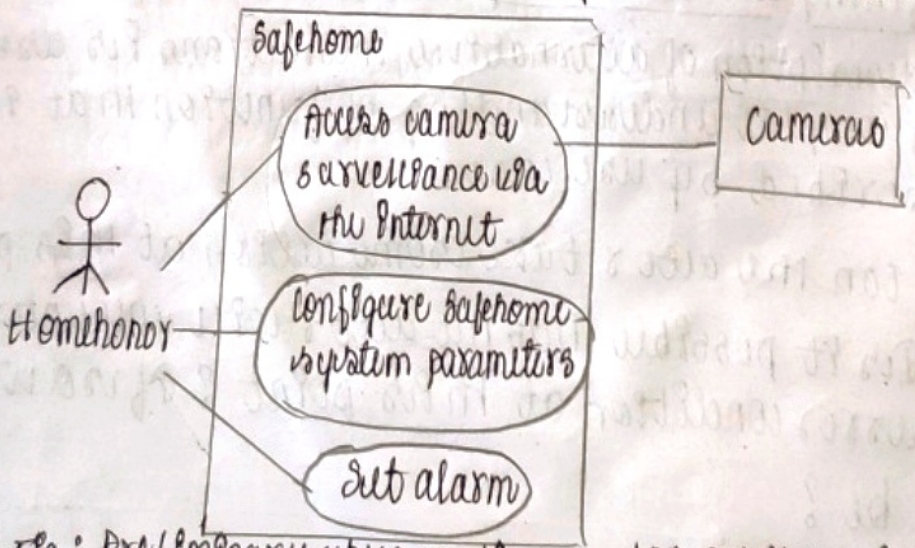


Fig: Preliminary usecase diagram for SafeHome System

⑤ What is a

Agile Proces
development
development
Principles

1. Our hig
through
software
2. Welco
develop
3. Deliver
couple
to sho
4. Bus
daily
5. Bui
them
trus
6. Th
info

⑤ What is agile process? Explain agility principles.

Agile Process model refers to as software development approach based on Iterative development.

Principles

1. Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.
2. Welcome changing requirements, even late in development
3. Deliver working software frequently, from a couple of weeks to couple of months with a preference to shorter timescale.
4. Business people and developers must work together daily throughout the project.
5. Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done
6. The most efficient and effective method of convey information to and within development team is

face to face conversation.

7. Working software is the primary measure of progress.

8. Agile processes promote sustainable development. The sponsors, developers and users should be able to maintain a constant pace indefinitely.

9. Continuous attention to technical excellence and good design enhance agility.

10. Simplicity - the art of maximizing the amount of work not done - is essential

11. The best architectures, requirements and designs merge from self-organizing teams.

12. At regular intervals, the team reflects on how to become more effective, then tunes and adjust its behavior accordingly.

⑥ Explain XP

user stories
value
acceptance
test criteria
iteration
plan

Releas
softw
proj

XP Pla
- Bug
- Ag
a
- St
g
- A
- Af
Ius
cl

asure of

velopment.
ould be
tely.

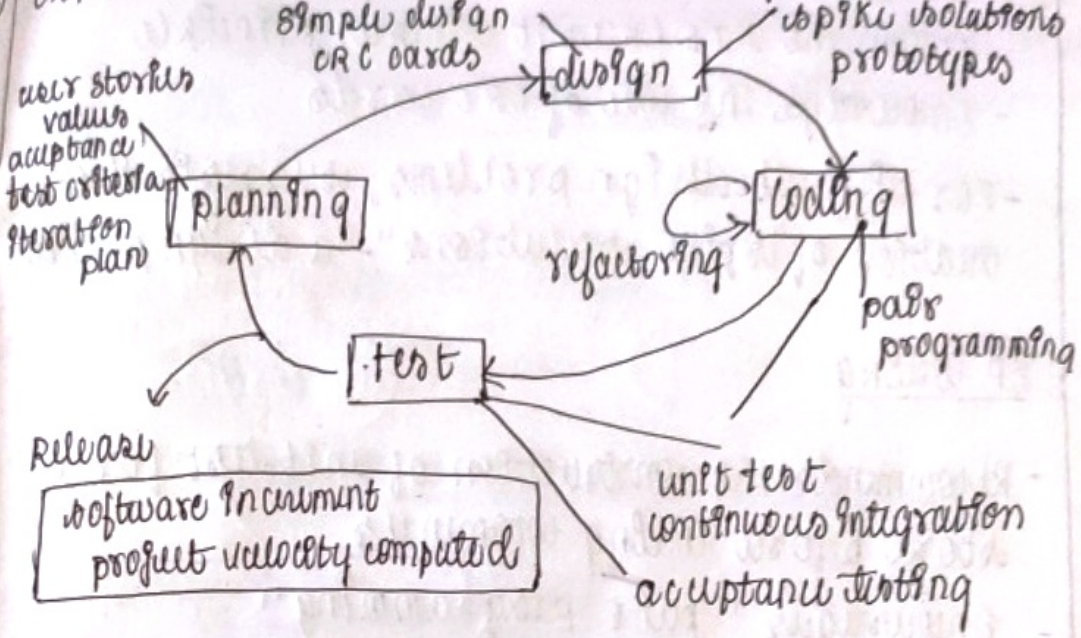
ing and

nt of work

d designs

n how
dfast

⑥ Explain XP process with neat diagram.



XP Planning

- Begins with the creation of "user stories"
- Agile team assesses each story and assigns a cost.
- stories are grouped to for a deliverable increment.
- A commitment is made on delivery date
- After the first increment "project velocity" is used to help define subsequent delivery dates for other increments.

XP Design

- Follow the KIS (Keep it simple) principle
- Encourage the use of CRC cards
- For difficult design problems, suggests the creation of "spike solutions" – a design prototype

XP Coding

- Recommends the construction of unit test for a store before coding commences.
- Encourages "pair programming"
- Pair programming : Two people work together at one computer workstation to create code for a story. This provides a mechanism for realtime problem solving and real time quality assurance
- Encourages "refactoring" – an iterative refinement of the internal program design.

XP Testing

- All unit test are executed daily
- "Acceptance tests" are defined by the customer and executed to assess customer visible functionality.

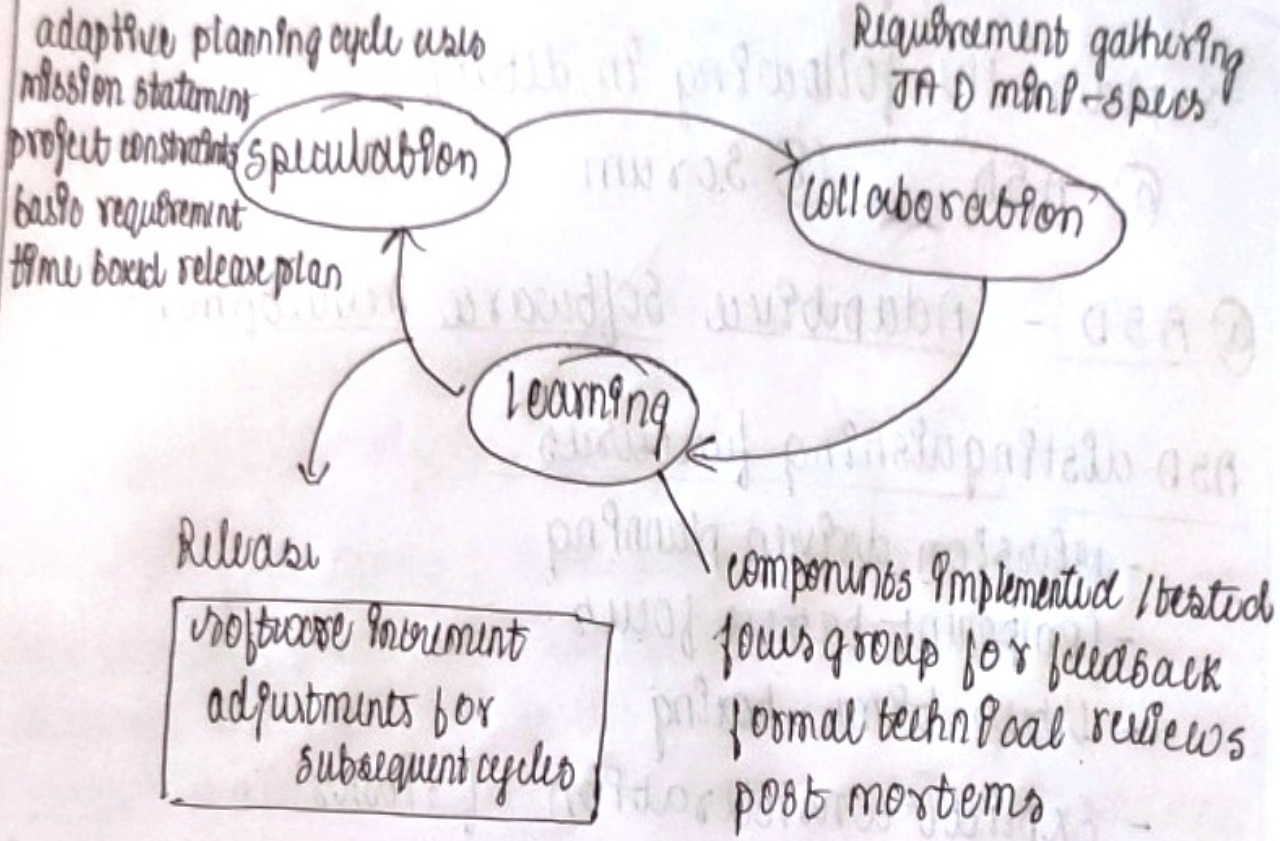
(9) Explain the following in detail
ⓐ ASD ⓑ Scrum

ⓐ ASD - Adaptive Software Development

ASD distinguishing features
- Mission driven planning
- Component based focus
- Use time-boxing
- Explicit consideration of risks
- Emphasizes collaboration for requirements gathering
- Emphasizes "learning" throughout the process

ASD "lifecycle" Incorporates three phase:

- Speculation
- Collaboration
- Learning



Speculation:

Here the project initiated and adaptive cycle planning is conducted.

Collaboration :

It encompasses communication and team work but it also emphasizes individualism.

Learning

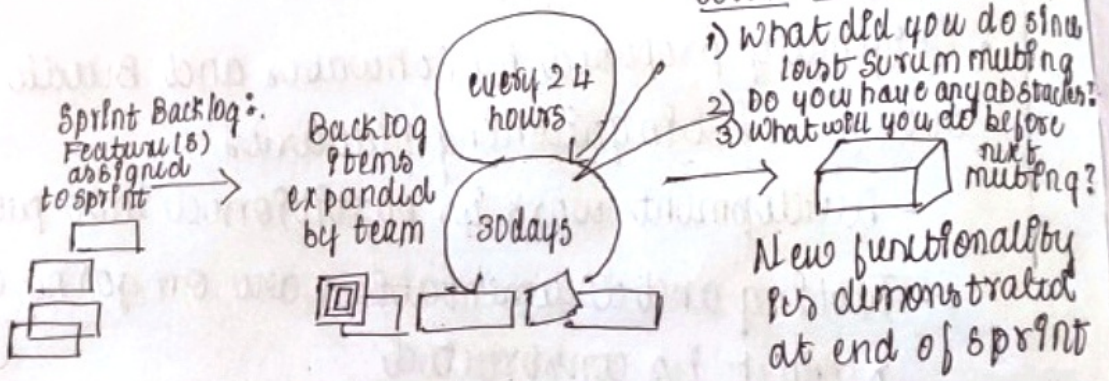
It will help team members to improve their level of real understanding. ASD team learn in 3 ways focus groups, technical reviews and project post mortems.

6. Scrum

- Originally proposed by Schwaber and Beedle.
- Scrum - distinguishing features.
  - Development work is partitioned into packets
  - Testing and documentation are on going as the product is constructed
  - work occurs in "sprints" and is derived from a "backlog" of existing requirements.
  - Meetings are very short and sometimes conducted with out chairs
  - "demos" are delivered to the customer with the time - box allocated.

Scrum is software development that was proposed by Jeff Sutherland and his development team in the early 1990s.

Scrum principles are consistent with the agile manifesto and are used to guide development activities with in a process that incorporates the following framework activities: requirements, analysis, design, evolution and delivery.



Scrum Master

- Making the process runs smoothly
- Removing obstacles that impact productivity
- Organizing and facilitating the critical meetings.

t was proposed
team in the

ru agile
lopment
rporates
uirements,
ry.

5 minute daily
did you do since
Scrum meeting
have any obstacles?
u you do before
next
meeting?

unctionality
monstrated
d of sprint

ty
eetings.

⑧ Explain the following in detail.
ⓐ FDD ⓑ LSD

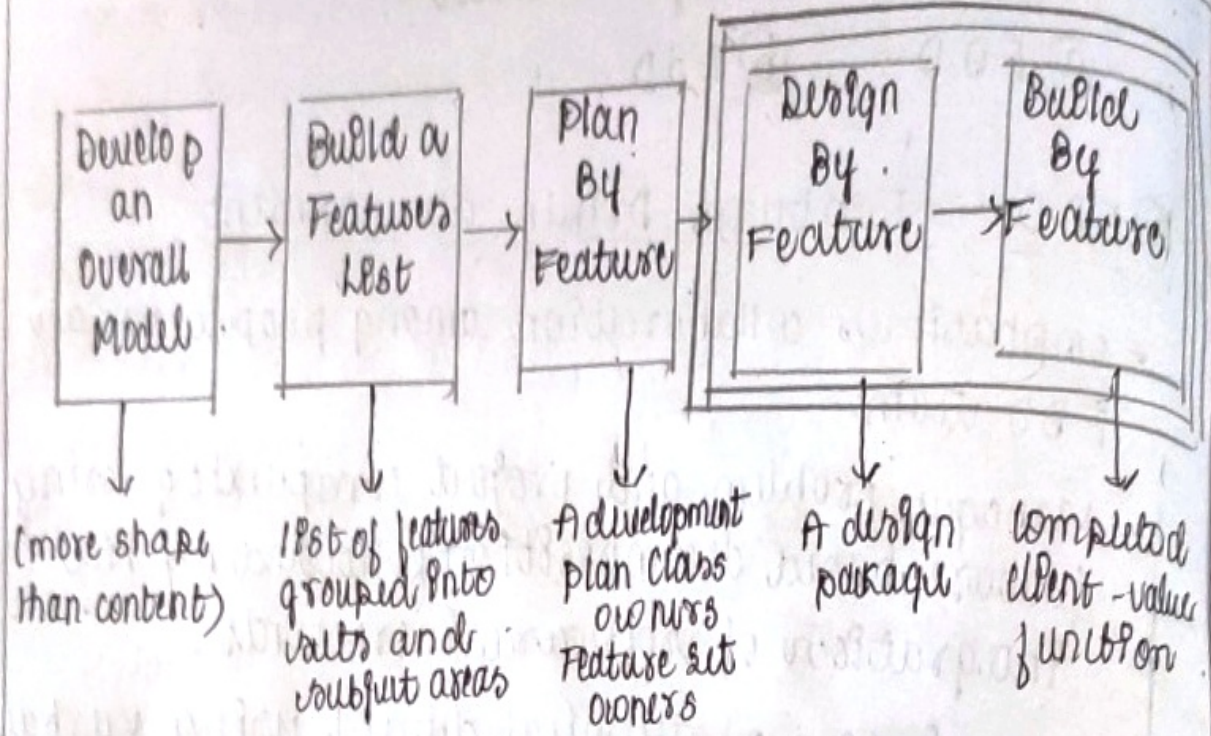
ⓐ FDD - Feature Driven development.

- emphasizes collaboration among people on an FDD team
- Manages problem and project complexity using feature based decomposition followed by the integration of software increments.
- Communication of technical detail using verbal, graphical and text - based means.

FDD - distinguishing features.

- Emphasis is on defining "features"
- Uses a feature template
- A features list is created and "plan by feature" is conducted.

Making a products sale is a feature.
Then the features set would include:
Add the product to shopping cart



(b) LSD - Lean Software Development

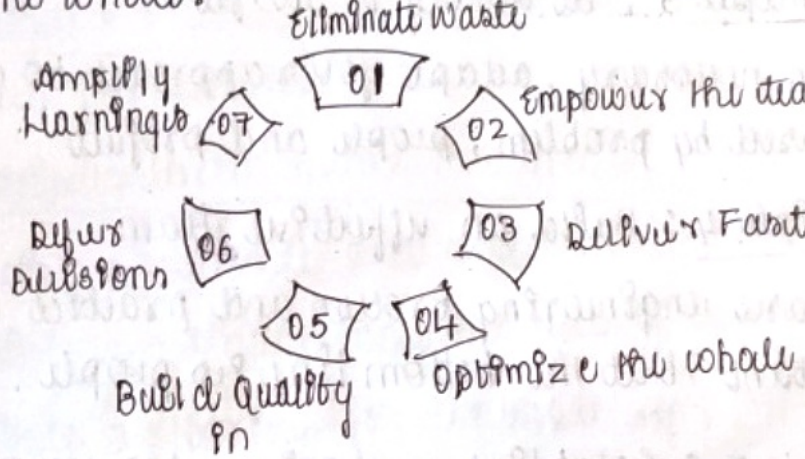
- Lean software development is an agile framework based on optimizing development time and resources, eliminating waste, and ultimately delivering only what the product needs.
- A primary goal of software development team is delivering valuable features and products as quickly and efficiently as possible.

Build By Feature

ompleted
lent-value
unction

nuork
nd
nately
.
rt
l

- The lean principles that inspire the LSD process can be summarized as eliminate waste, build quality in, create knowledge, defer commitment, deliver fast, respect people and optimize the whole.



(9) Explain the principles that Guide Process and Practice.

Principles that Guide Process

Principle 1: Be Agile

Every aspect of the work you do should emphasize economy of action.

Principle 2 : Focus Quality at every step

The next condition for every process activity, action and task should focus on quality of the work product that has been produced.

Principle 3 : Be ready to adapt

When necessary, adapt your approach to constraints imposed by problem, people and project

Principle 4 : Build an effective team.

Software engineering process and practice are important, but the bottom line is people.

Principle 5 : Establish mechanisms for communication and coordination.

Projects fail because important information falls into the cracks and stakeholders fail to coordinate their efforts to create a successful end product

Principle 6 : Manage change

The approach may be either formal or informal but mechanisms must be established to manage the way changes are requested

assessed, approved and implimented.

Principle 7 : Assess Risk.

Lots of things can go wrong as software is being developed.

Principle 8 : Create work products that provide value for others.

Principles that Guide Practice :

Principle 1 : Divide and conquer.

Principle 2 : Understand the use of abstraction

Principle 3 : Strive for consistency.

Principle 4 : Focus on the transfer of information

Principle 5 : Build software that exhibits effective modularity

Principle 6 : Look for patterns

Principle 7 : When possible, represent the problem and its solution from a number of different perspectives.

Principle 8 : Remember that someone will maintain the software.

(10) Explain Communication and Planning Principles.

Communication Principles

Principle 1 : Listen.

Try to focus on the speaker words, rather than formulating your response to those words

Principle 2 : Prepare before you communicate

Spend the time to understand the problem before you met with others.

Principle 3 : Someone should facilitate the activity

Every communication meeting should have a leader to keep the conversation moving in a productive direction;

Principle 4 : Face - to - Face communication is best

Principle 5 : Take notes and document decisions

Principle 6 : Strive for collaboration

Principle 7 : Stay focused, modularize your discussion

Principle 8 : If something is unclear, draw picture.

Principle 9 : Once you agree to something, If you can't agree to something move on, If a feature or function is unclear and cannot be clarified at the moment, move on

Principle 10 : Negotiation is not a contest or a game. It works best when both parties win.

Planning Principles

Principle 1 : Understand the scope of the project.

Principle 2 : Involve the customer in the planning activity.

Principle 3 : Recognize that planning is iterative

Principle 4 : Estimate based on what you know

Principle 5 : Consider risk as you define the plan

Principle 6 : Be realistic

Principle 7 : Adjust granularity as you define the plan.

Principle 8 : Define how you intend to ensure quality

Principle 9 : Describe how you intend to accommodate change

Principle 10 : Track the plan frequently and make adjustment as required.