Characteristics of S/W

→ S/w is developed or engineered, it is not manufactured in the classical sense.

→ S/w does not wear out, However it deteriorates due to change

→ S/w is custom built rather than assembling existing components.

● Although the industry is moving towards component based construction, most S/w continues to be custom b

FIGURE 1.1
Failure curve for hardware

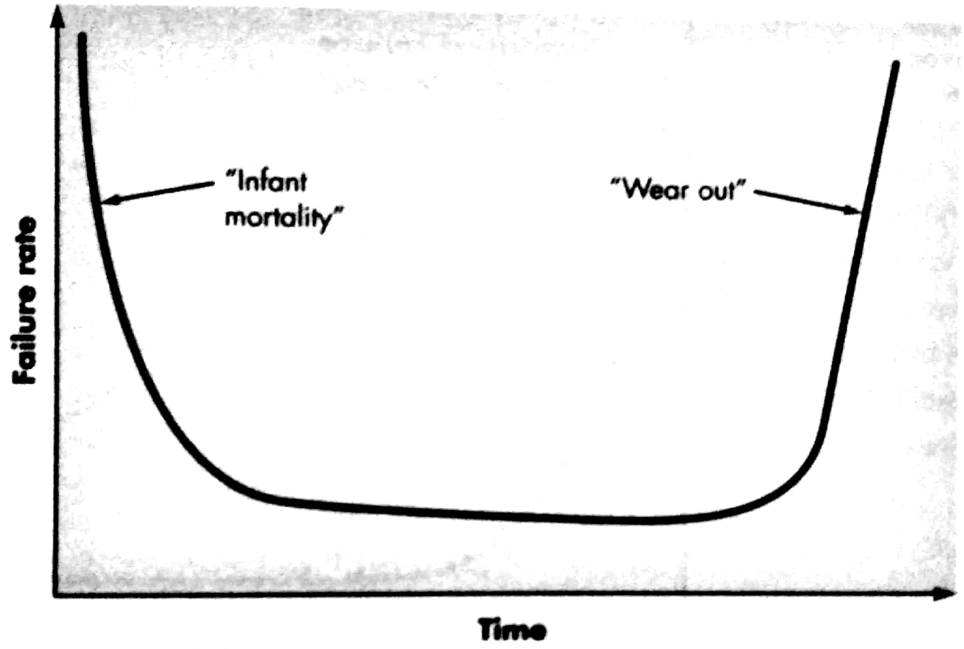
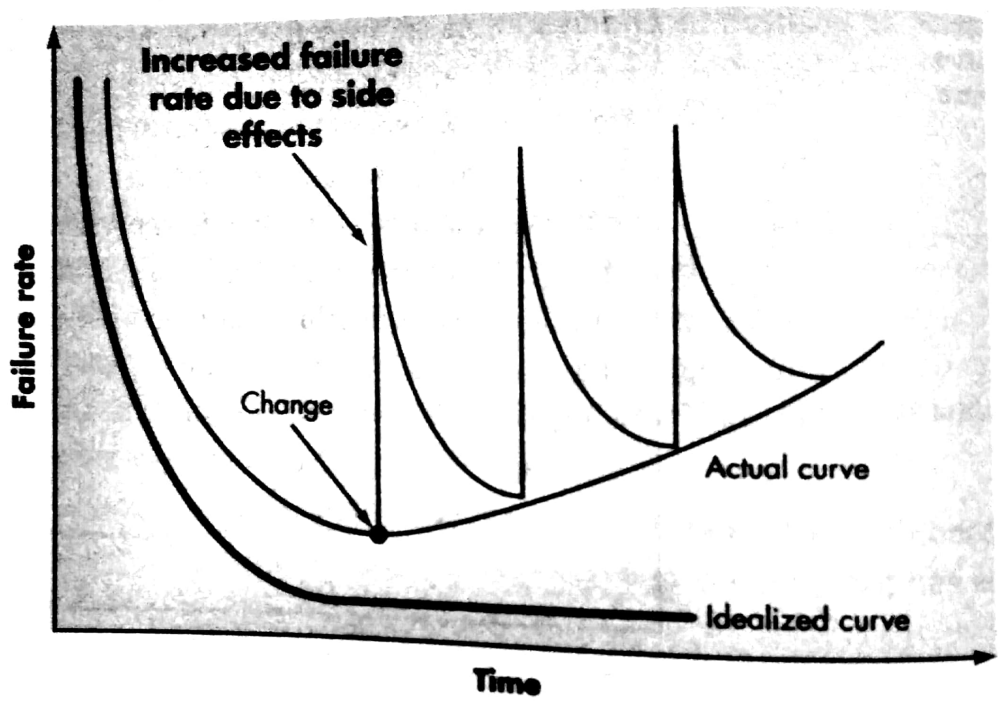


FIGURE 1.2
Failure curves for software



Legacy s/w :-

* It is older programs that are developed decades ago.
* The quality of legacy s/w is poor becz it has inextensible design, convoluted code, poor and result that are not achieved.
*

→ The s/w must be adapted to meet the needs of new computing environment or technology

→ The s/w must be enhanced to implement new business requirements

→ The s/w must be extended to make it interoperable with modern systems

→ the s/w must be rearchitected to make it viable with in a n/w environment.

S/w Evolution:-

* s/w evolves due to changes
* changes occur due to correction, adaption and enhancement.
* 8 laws of unified theory
  * The law of Continuing change
  → " " " Increasing complexity
  → " " " self-Regulation
  → " " " Conservation of organizational stability
  → " " " Growth of Familiarity
  → " " " Continuing Growth
  → " " " Declining quality
  → The Feedback system Law

form describing an

S/W MYTHS :-

→ widely held but false view
→ propagate misinformation and confusion
→ three types of myth
- management myth
- costomer myth
- practitioner's myth.

1 → myth(1)
- the available standards an procedures for s/w are enough.

myth(2)
- each organization feel that they have state-of-art s/w development tools since they have latest computer.

myth(3)
- Adding more programmers when the work is behind schedule can catch up.

myth(4)
- outsourcing the s/w Project to third Party, we can relax and let the Party build it.

Customer myth

*Myth(1)
→General start of objective is enough to begin writing programs, the details can be filled in later

*myth(2)
→Software is easy to change becz s/w is flexible.

Practitioner's myth:-

* myth(1)
→ once the program is written, the job has been done

* myth(2)
→ Until the program is running, there is no way of assessing the quality.

* myth(3)
→ The only deliverable work product is the working program

*myth(4)
→ SE creates voluminous and unnecessary documentation and invariably slows down s/w development

Process pattern template:-

Scott Ambler:- an object oriented Consultant has proposed a template for process pattern.

- → pattern name
- → intent
- → type.
- → initial context.
- → problem.
- → Solution
- → Resulting Context
- → Known uses.

→ The pattern name should be a meaningful name given to the pattern from pattern name one can guess its functionality.

→ intent the porpose of the pattern should be described here.

→ Type: Ambler has suggested 3 types of patterns.

Task pattern:- defines software engineering action or work task that is part of the process and relevant to successful software engineering practice.

ex:- requirements gathering.

Stage pattern:- defines a framework activities for the process.

Phase pattern:- define the sequence of framework activities that occur with the process, even when the overall flow of activities is iterative in nature.

Initial context:- In this section the conditions under which the pattern applies are described prior to the initiation of the pattern.

→ In this section following issues need to be described.

i) The set of organizational or team related activities that have already occured.

ii) The list of entry state processed.

iii) Already existing software engineering (or) project related information

Problem:- under this section, the problem is mentioned for which the pattern is to described. For example insufficient requirement is a problem. That means customers not sure about what they want exactly. They could not specify the requirements proper manner.

Solution:- Every problem for which pattern has to be described should be accompanied solution.

for ex:- The problem of insufficient requirements has solution. That is effective communication with the customer.

Resulting context:- describes the result after the successful implimentation of pattern.

Known uses:- It describes the uses of the pattern. which are defined from the resulting content.

Continuous model:-

→ lets organisation select specific improvement that best meet its business objectives and minimize risk.

→ levels are called capability levels

→ Describes a process in 2-D

→ Each process area is assessed against specific goals and practices and is rated according to the following capability levels.

→ Six levels of CMMI

- level 0 : incomplete
  level 1 : Performed
  level 2 : managed
  level 3 : Defined
  level 4 : Quantitatively managed
  level 5 : optimized.

Incomplete:-

→ Process is adhoc. objective and goals of process areas are not known.

Performed:-

→ Goal, objective, work tasks, work products and other activities of slw process are carried out

Managed:-
→ Activities are monitored, reviewed, evaluated and controlled.

Defined:-
→ Activities are standardized, integrated and documented.

Quantitatively managed:-
→ Metrics and indicators are available to measure the process and quality.

• optimized:-
→ Continous process improvement based on quantitative feedback from the user.
→ use of innovative ideas and techniques, statistical quality control and other methods for process improvement.

Staged model:-
→ This model is used if you have no clue of how to improve the process for quality s/w
→ It gives a suggestion of what things other organizations have found helpful to work first
→ Levels are called maturity levels.

Level	Focus	Process Area
optimizing	continuous process improvement	→ organizational innovation and deployment. → causal Analysis and Resolution.
Quantitative/ managed	Quatitative management	→ organization process performance → Quantitative project managment.
Defined	process standardized	→ Requirements Development → Technical solution → Product integration → Verification → Validation → organizational process focus → organizational process defination → organizational Training → integrated project management → Risk managment → integrated Teaming - integrated supplier management
	→ Decision Analysis and research → organizational environment for integration	

Managed

Basic project management

→ Requirements management
→ Project planning
→ Project Monitoring and Control
→ Supplier Agreement
→ measurement and Analysis
→ process and product
→ Quality Assurance
→ Configuration Management

Performed → focused on Goals

Personal & Team process models:- watts humphrey developed PSP and TSP at the SEI in the mid-1990's.

The personal software process (PSP) is an SEI technology that brings disciplines to the practices of individual software engineers, dramatically improving product quality, increasing cost and schedule predictability and reducing development cycle time for software.

The team software process is a complementary SEI technology that enables teams to develop software-intensive products more effectively. TSP shows a team of engineers how to produce quality products for planned costs and on aggressive schedu

framework activities in PSP

→ planning → high level design → review → development

→ postmorten (deployment)

The objectives of team process models:-

i) Build the self directed team, for planning the software project in systematic manner. The normal size of the team should be 3-20 Software engineers.

ii) indicate the project manager for the coaching needed by the team members for the performance ~~enhancement~~ improvent.

iii) using CMM level 5, improve the software process.

iv) provide the improvement guidance for the software team.

v) provide the university teaching.

The frame work activities in TSP are:-

→ Launch, → high level design, → implimentation

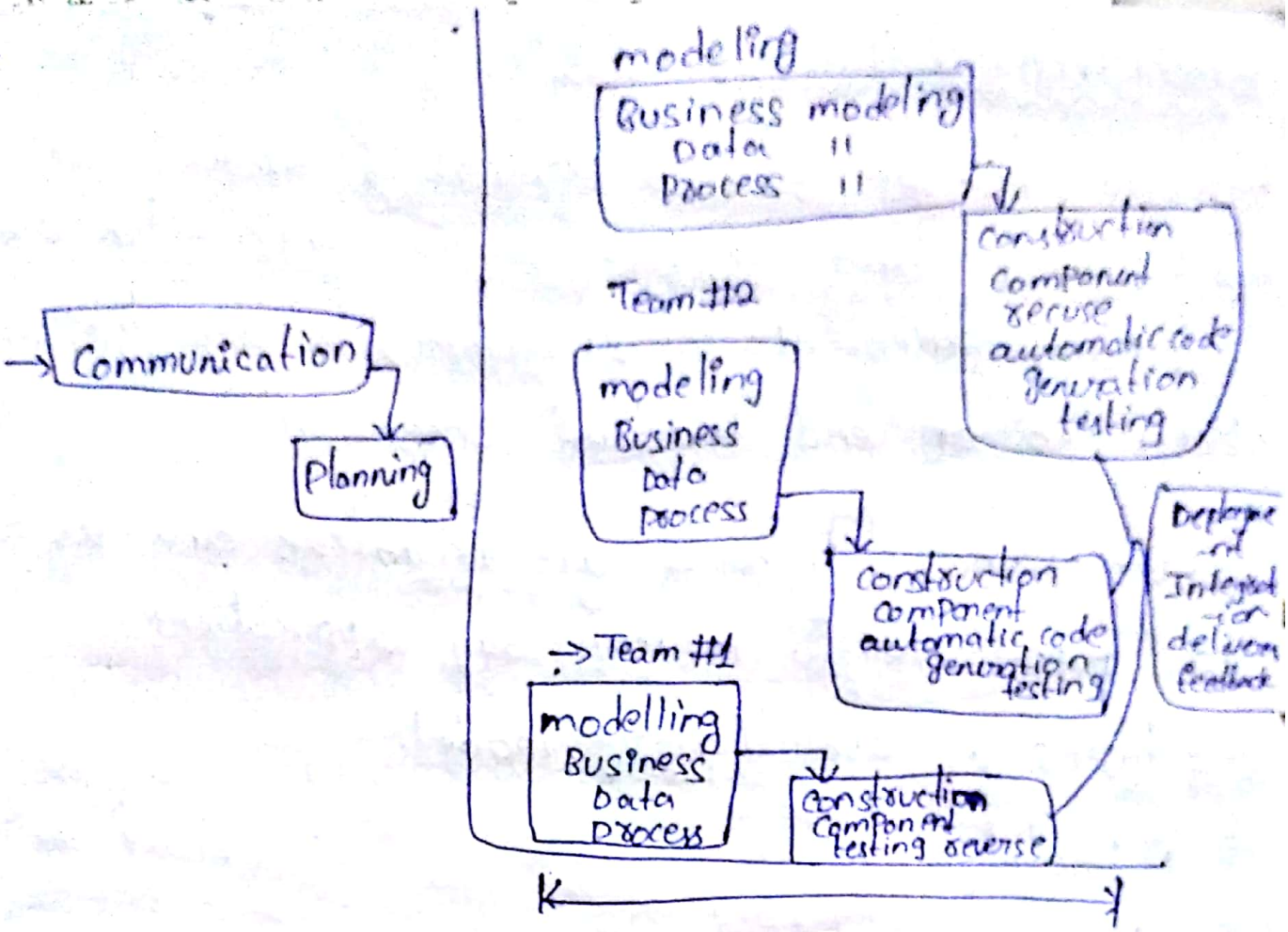
→ integration and test, → postmorten.

Software process model:- Is a strategy that a software development team incorporates in order to build software.

Is choosen based on the nature of the project and application, methods and tools to be used, and the controls and deliverables that are required.

The RAD model: (Rapid application Development)

→ An incremental slw process model
→ Having a short development cycle.
→ High speed adoption of the waterfall model using a componant based Construction approch
→ Creates a fully functional system within a very short span time of 60 to 90 days



→ multiple slw teams work in parallel on different function.

→ modelling encompasses three major phases: Business, modeling, Data modeling and process modeling

→ Construction uses reusable components, automatic code generation and testing.

Problems in RAD :-

→ Requires a number of RAD teams

→ Requires Commitment for both developer and customer for rapid fire completion of activity

→ Requires modularity

→ Not suited when technical risks are high

***

Unified Process model :-

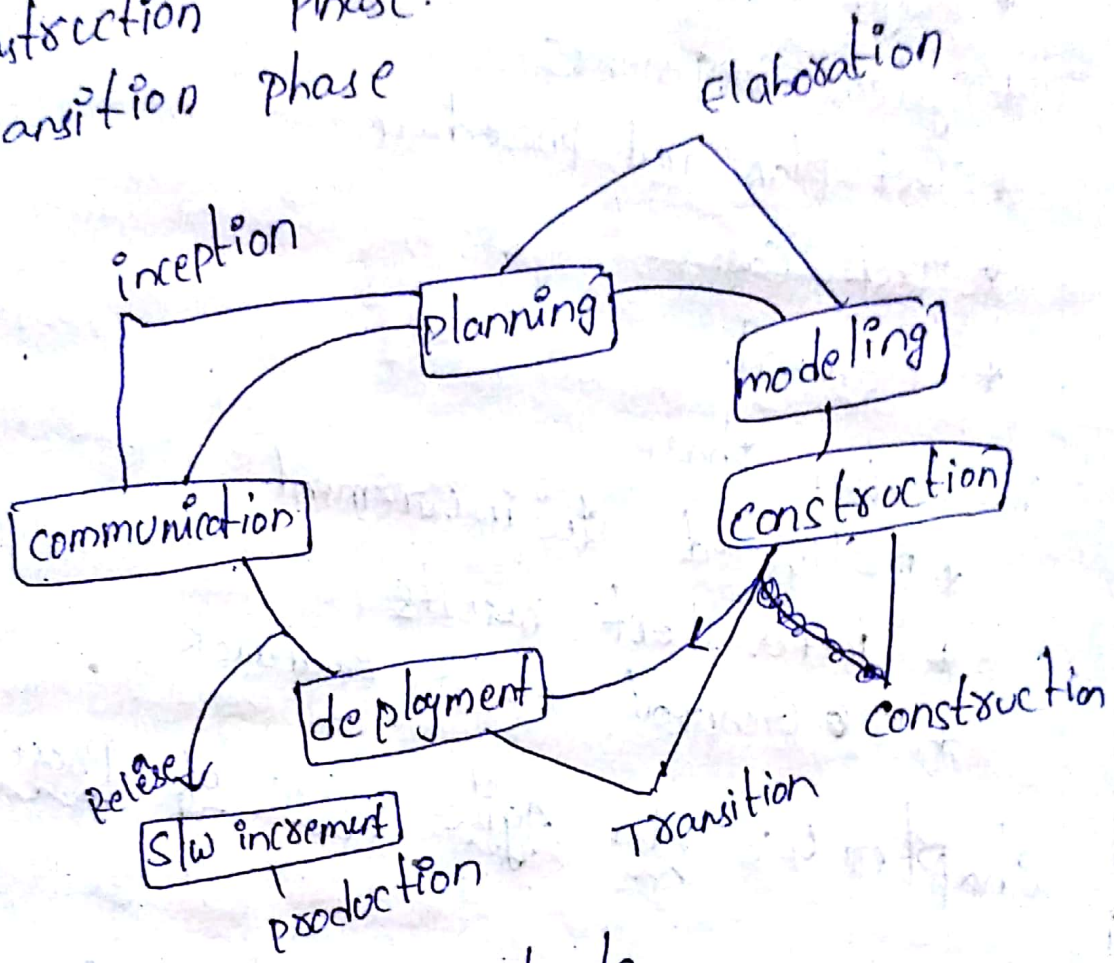
* Evolved by Rumbaugh, Booch, Jacobson.
* Combines the best features their OO models
* Adopts additional features proposed by other experts.
* Resulted in Unified modeling Language (UML)

→ Unified PD

→ A framework of OO SE using UML

phases of unified process

* Inception process phase
* Elaboration phase
* Construction phase.
* Transition phase



unified process work products

→ Tasks which are required to be completed w during different phases.

→ Inception phase
  * vision document
  * use-case model, initial project glossary
  initial assessment

→ Elaboration phase

* use-case model
* Analysis model
* s/w architecture description.

Construction phase

* Design model
* System Components
* Test plan and procedure
* Test Cases
* Manual.

Transition phase

* Delivered s/w increment
* Beta test results
* General user feedback

Evolutionary Process model:-

→ s/w evolves over a period of time

→ Business and Product requirements often change as development proceeds making a straight line Path to an end Product unrealistic

→ Evolutionary models are iterative and as such are applicable to modern day applications

→ types of evolutionary model.

1) Prototyping
2) Spiral model
3) Concurrent development model.

1) Prototyping:

→ mock up (or) model (throw away version) of a s/w Product

→ Used when customer defines a set of objective but does not identify i/p, o/p or processing requirements.

→ Developer is not sure of
  → efficiency of an algorithm
  → adaptability of an os
  → human / Machine interaction

steps:

→ Begins with requirement gathering.

→ identify whatever requirements are known

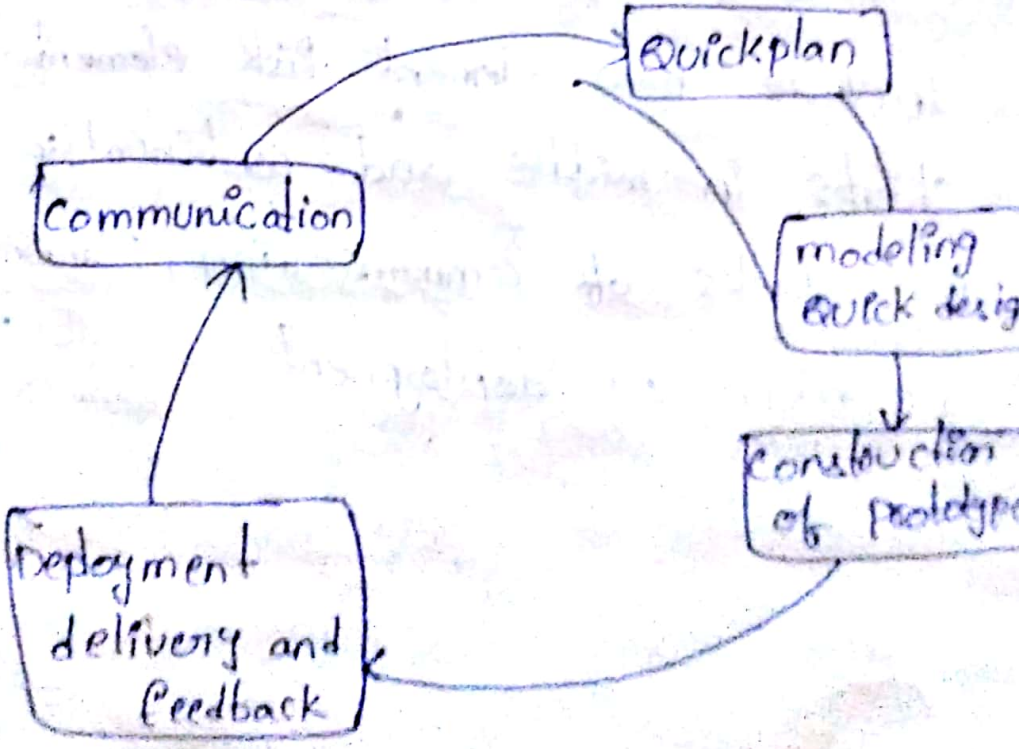
→ outline areas where further definition is mandatory.

→ A quick design, leads to the construction of prototype

→ Prototype is evaluated by the customer

→ requirements are refined

→ Prototype is turned to satisfy the needs of customer.



Limitations

→ In a rush to get it working, overall s/w quality (or) long term maintainability are generally overlooked

→ Use of inappropriate os (or) PL

→ use of inefficient algorithm.

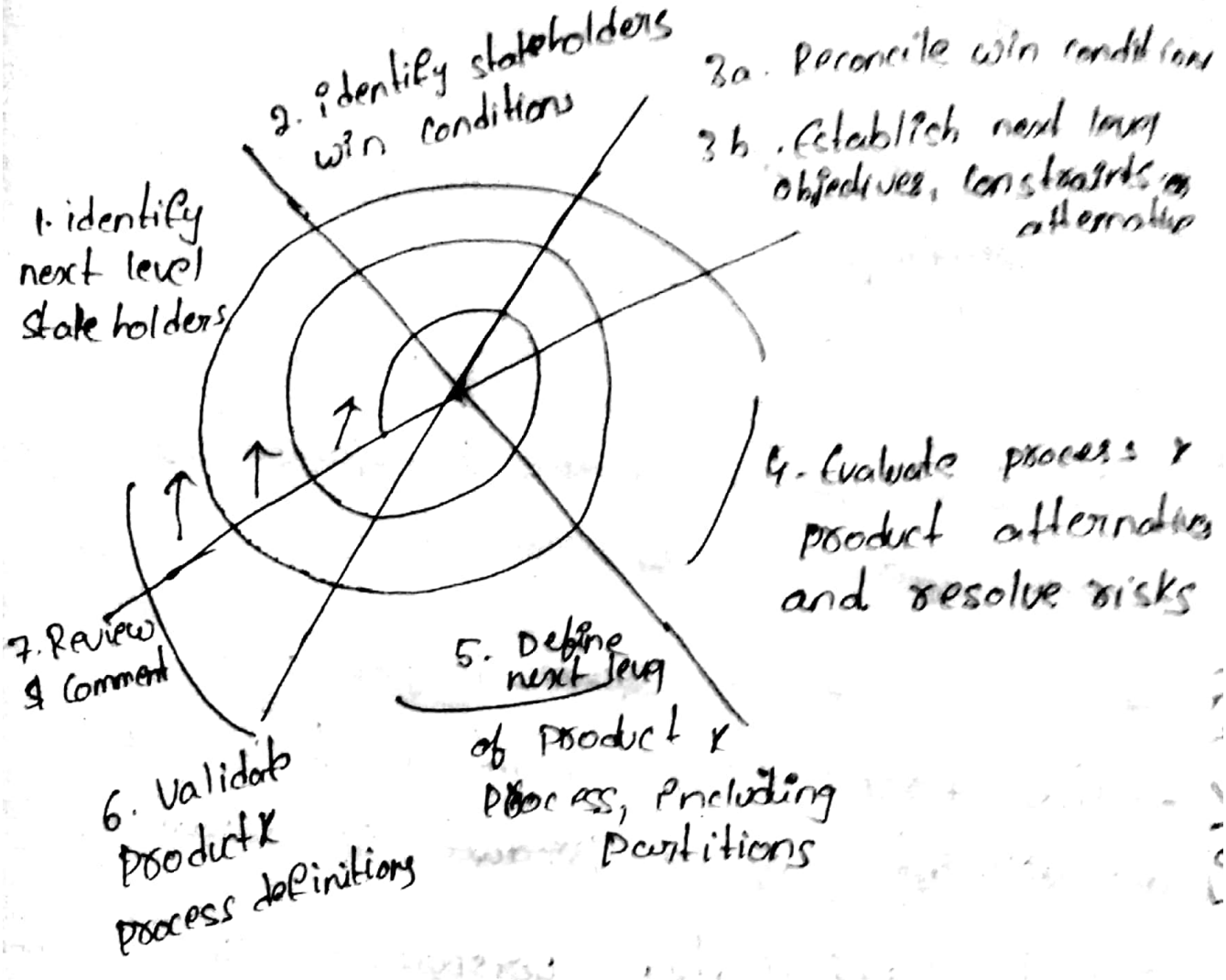
***

The spiral Model

→ An evolutionary model which combines the best feature of the classical life cycle and the iterative nature of prototype model.

→ Include new element Risk element.

→ starts in middle and continually visits the basic tasks of communication, Planning, modeling, construction or deployment.



→ 1. Communication

→ Tasks required are establish effective communication b/w developer

2. Planning

estimation

scheduling.

Risk analysis

3. Modeling

Analysis

design

4. Construction

code
test

5. Deployment

Delivery
Support
feedback

→ Realistic approch to the development of large scale system or s/w

→ s/w evolves a process progresses and customer

→ The first circuit might result in the development of a product specification

→ And sophisticated version of s/w.

The concurrent development model:-

→ Also called concurrent engineering

→ Constitutes a series of framework activities, s/w engineering action, tasks and their associated tasks.

→ All activites exist concurrent but reside in different status

→ All activites exist concurrent but reside different states.

→ Applicable to all types of sw development

→ Event generated.

Human Factors

→ the process molds to the needs of the People and team; not the other way around.

→ Key traits must exist among the people on an agile team and the team itself:

* Competence. (talent, skills, knowledge)
* Common focus. (deliver a working slew increment)
* Collaboration. (peers and stakeholders)
* Decision-making ability. (freedom to control its own destiny)
* fuzzy problem-solving ability (ambiguity and constant changes, today problem may not be tomorrow's problem)
* mutual trust and respect.
* Self-organization. (themselves for the work done, process for its local environment, the work schedule)

Extreme programming:-

→ The most widely used agile process, originally proposed by Kent Beck

- **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 defines subsequent delivery dates for other increments.
- **Xp Design:-**
  - follows the KIS (Keep It simple) Principle
  - Encourage the use of CRC (class responsibility collaborator) cards.
  - For difficult design problems, suggest the creation of "Spike solutions" a design prototype.
  - Encourages "refactoring"- an iterative refinement of the internal program design.
- **Xp Coding**
  - Recommends the Construction of a unit test for a story before coding commence.
  - Encourages "pair programming"
- **Xp testing**
  - All unit tests are executed daily
  - "Acceptance tests" are defined by the customer and executed to assess Customer visible functionality.

Adaptive software development model:- (ASD)

Orginally proposed by jim highsmith

→ An ASD life cycle has six basic charectristics:

i) Mission focused
ii) feature based
iii) Iterative.
iv) Time boxed
v) Risk driven.
vi) change tolerant.

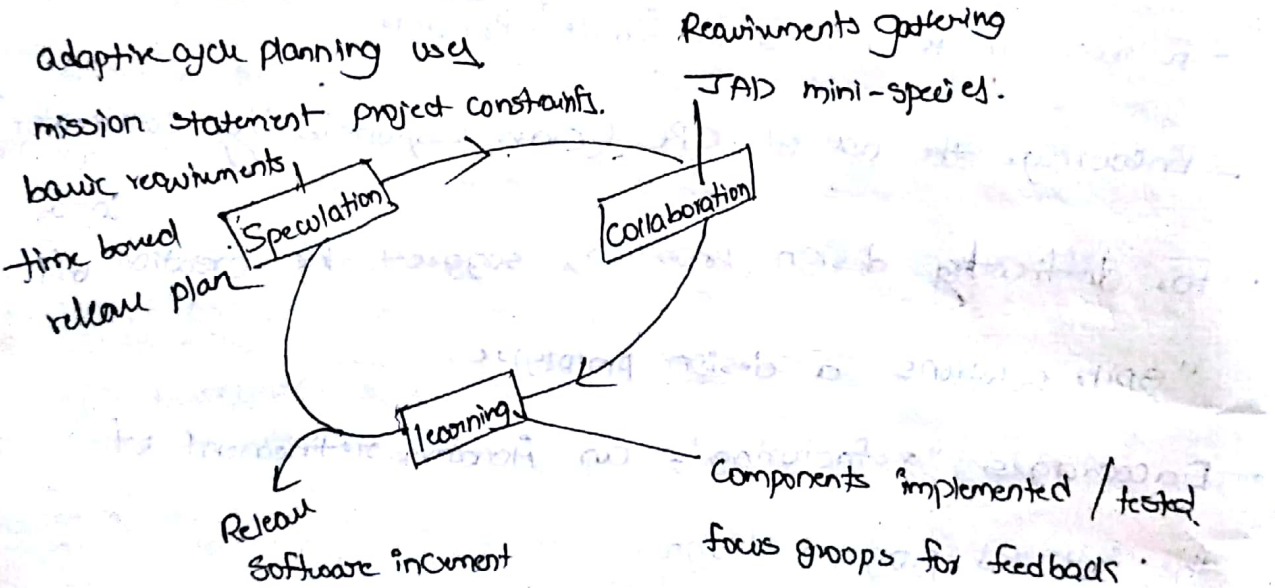
→ ASD is made of three steps (speculate, collaborate and learn)

So, here those steps described briefly

Speculate → initiation & planning

Collaborate → Concurrent feature development

Learn → Quality Review.



→ The first step is speculation = initiation and planning.

→ During this phase, coders attempt to understand the exact nature of the software and the requirements of the users. The phase relies on bug and user reports to guide the project. if no reports are available. the developers use the basic requirements outlined by the purchaser.

→ The collaboration phase = Concurrent feature development.

→ When the individual developers solidify what they are each doing and how to combine their portions. This phase is generally completely in-house. The developers don't need any additional information or outside input to manage this portion of the software.

→ The last step is learning = Quality Review.

→ During the learning phase, the newest version of the software is released to users. This generates the bug and user reports used during the first phase of the project, and the cycle repeats itself.