SOFTWARE ENGINEERING.

INTRODUCTION: 1) The evolution of electronic computers began in 1940s. Early efforts in the field of computing were focused on designing the hardware.

2) In early computing systems, there was essentially no operating system; the Pgs. were fed with Paper tapes or by scratches.

3) With evolution of second-generation machines, in early 1950s early concept of OS evolved and (i.e single-user OS came into existence).

4) High-level languages, particularly FORTRAN and COBOL along with their compilers were developed.

5) Then in 1960s multiprogramming OS came into picture. The usability and efficiency of computing machines took a big leap. Prices of hardware also decreased and awareness of computers increased substantially.

6) With the availability of cheaper and more powerful machines, higher-level languages, and more user-friendly OS, the application of Computers grew rapidly.

7) In addition, nature of Software engineering evolved from simple programming exercise to developing software systems, which were much larger in scope and required great effort by many people.

8) The techniques of writing simple programs could not be scaled up for developing software systems, and computing world found itself in the midst of a "SOFTWARE CRISES".

Period the software industry unsuccessfully attempted to build larger and larger s/w systems by simply scaling up existing development techniques. As a result:

* Schedule and cost estimates were often grossly inaccurate
* Poor quality s/w was produced.
* Productivity of programmers could not keep up with demand.

In late 80s and early 90s, s/w crises is characterized by an "inability to develop s/w on time, on budget and within requirements".

As a result, delivered S/W systems are:

* Completely unsatisfactory (not as per specification).
* Extremely late
* Fear over budget
* Poorly suited for intended uses of the system.

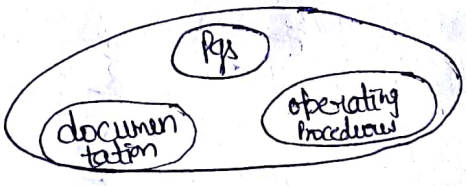
REASONS OF SOFTWARE CRISES:- The main reasons are:

1) Lack of communication b/w S/W developers and users.
2) Increase in cost of S/W compared to hardware.
3) Increase in size of S/W.
4) Increased complexity of problem area.
5) Project management problem.
6) Lack of understanding of problem and its environment.
7) High optimistic estimates regarding S/W development time and cost.

To address these problems, the discipline of S/W engineering came into existence.

PROGRAM VERSUS SOFTWARE:- S/W is more than Pgs. It consists of Pgs, documentation of any part of Pg and procedures used to setup and operate the S/W system.

Components of S/W:-



Cont. on Pg. 4.

S/W = Pg + documentation + operating Procedures.

Any Pg. is a subset of S/W and it become S/W only if documentation and operating procedure manuals are prepared. Pg is a combination of source code & object code.

History of Software Engineering — History of Computing

Year	Before 1970	1971-80	1981-90	1990-date
Style	Programming-any which-way	Programming-in the small	Pg. in the large	Mega Pg.
Characteristic Problems	Small programs	alg. and Pg.	Interface, mgt. system structures	distributed Inf. system
Data Issues	Representing structure and symbolic inf.	Data structure and type	Long-lived data bases, symbolic as well as numeric	Multimedia database
Control Issues	Elementary understanding of Control flow	Pg. execution once and terminate	Pg. assemblies execute continually	Concurrency and distribution
Specification Issues	Mnemonics + Precise use of Rose	Simple I/O specification	System with complex specification	Safety critical system.
State Space	State not well understood apart from control	Small, simple state space	Large, structured state space	Unknown
Mgt. Focus	None	Individual effort	Team efforts, System maintenance	SQA quality Process Improvement
Tools, Methods	assemblers, Core dumps	Programming Pg., Compilers, linker	environments & integrated tools, documents	OO Frameworks

* Dennis (1975) :- S/w eng. is the application of Principles, skills and art to design and Construction of Pgs. and system of Pgs.

* IEEE (1991) :- S/w eng. is the application of a systematic, disciplined and quantifiable approach to development, operation and maintenance of S/w.

* S/w eng. is the practical app. of scientific knowledge for the economical production and use of high quality S/w.

* S/w eng. is a discipline whose aim is production of quality S/w, S/w that is delivered on time, within budget and that satisfies its requirements.

* CHARACTERISTICS OF S/W ENGINEERING :- * It concern the Construction of large Pgs.

* The central theme of S/w eng. is mastering key,

* S/w evolves.

* Regular cooperation b/w people and an integral part of programming.

* S/w has to effectively support its users.

* The efficiency with which S/w is developed is of crucial importance.

GOALS OF S/W ENG:- The general goal of s/w eng. is to produce s/w that is economic and useful & safe for people.

* Goal focus on development of s/w that supports the needs of user and helps the maintainer to adapt the s/w so that it will continue to support the evolving needs of user.

MAIN GOALS:- * Improve the productivity of Programming / development process.

* Producing what the customer wants.
* Producing s/w that satisfies specifications.
* Controlling and predicting the cost of s/w development.

* Producing the following products in addition to pgms.:
  - documentations
  - Specifications
  - development Plans.

* Improve the quality of s/w product at all levels.

* Producing s/w systems with foll. attributes:

- Reliability:- is a measure of s/w's ability to respond logically to unexpected conditions and to respond predictably under normal conditions.
- Efficiency:- is a measure of s/w's ability to perform its operations in an expedient manner.
- Clarity:- to be understood by those who maintain it.
- Extensibility:- of how easily s/w can have its scope modified or expanded with minimal or (no) effect on s/w's existing internal interface.

PROCESS AND ITS CHARACTERISTICS:-

S/w Process is set of activities and associated results that produce a S/w product. So it is way in which we produce S/w. This differs from org. to org.

Following are the FUNDAMENTAL PROCESS ACTIVITIES that are common to all SOFTWARE Processes:

1) SOFTWARE SPECIFICATION:- The functionality of S/w and constraints on its operation must be defined.

2) SOFTWARE DEVELOPMENT:- S/w must be developed that meets the specifications.

3) SOFTWARE VALIDATION:- S/w must be validated to ensure that it performs what customer needs.

4) SOFTWARE EVOLUTION:- S/w must evolve to meet changing customer requirements.

CHAR:-
* UNDERSTANDABILITY:- To what extent is Process explicitly defined and how easy is it to understand the Process definition.
* VISIBILITY:- Process activities provide clear results so that the progress of Process is extremely visible?
* ROBUSTNESS:- Can Process continue in spite of unexpected problems.
* RELIABILITY:- Is Process designed in such a way that Process errors are avoided or trapped b/f they result in Product errors?

* ACCEPTABILITY :- Is defined process acceptable to and usable by engineers responsible for producing SW product?

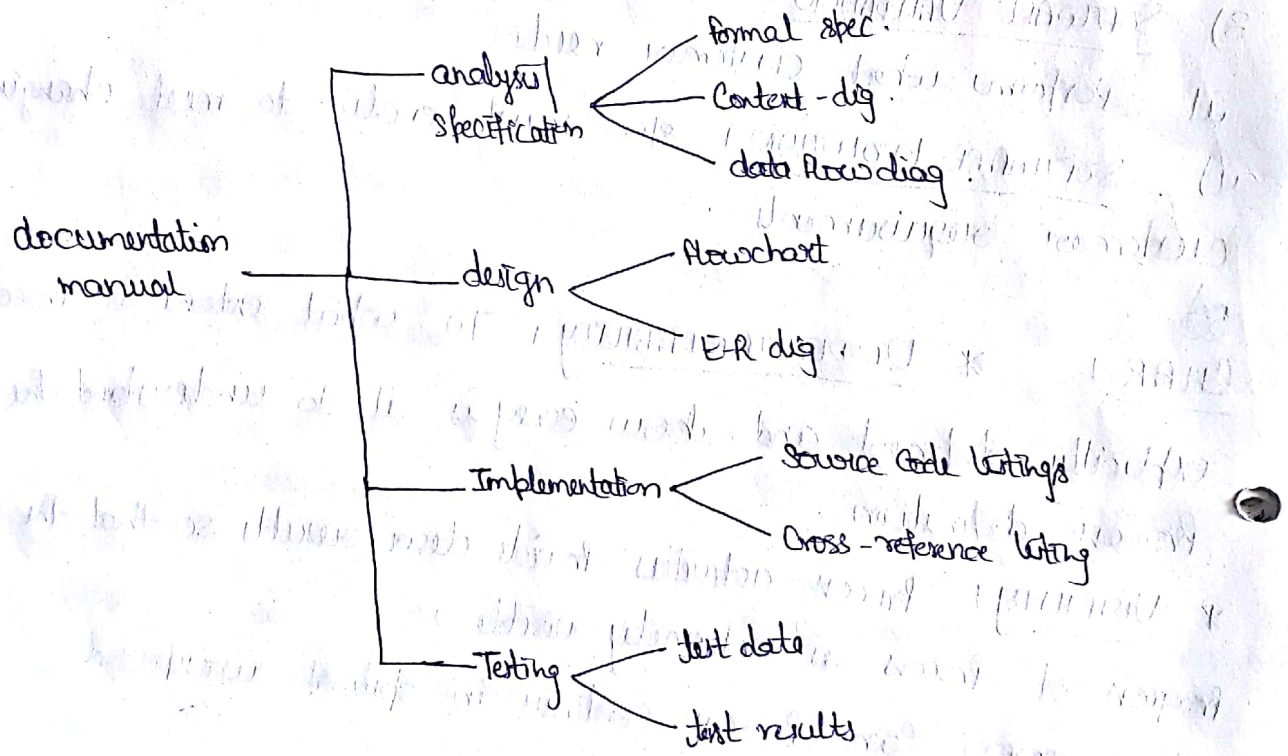
* MAINTAINABILITY :- Can process evolve to reflect changing organisational requirements or identified process improvements?

* Rapidity :- How fast can the process of delievering a system from a given specification be Completed?

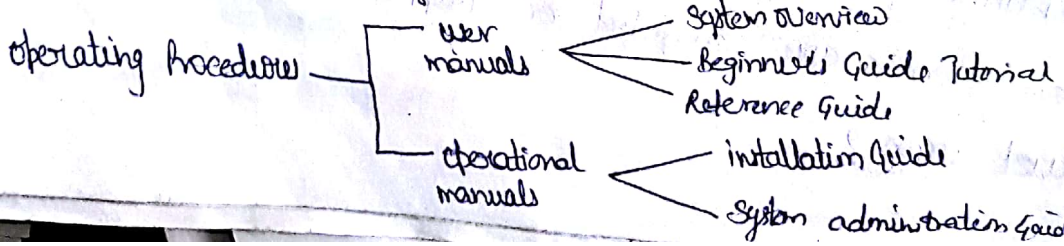
* Supportability :- To what extents can CASE tools support process activities.

---

documentation Consists of different types of manuals :-



Operating Procedures Consist of instructions to setup and use of SW system and instructions on how to react to system failure.



REASONS WHY IT IS DIFFICULT TO IMPROVE S/W PROCESS:-

(1) **Not enough time**:- (1) The development team often has not sufficient time to develop the S/W.

(2) No S/W group has spare time to devote to exploring what is wrong with their current development Process. and what they should be doing differently.

(3) Customers and senior managers are demanding more S/W, of higher quality in minimum possible time. So there is always a shortage of time.

(2) **LACK OF KNOWLEDGE**:- (1) Many S/W developers don't seem to be familiar with industry best Practices.

(2) Developers may buy books on Java, VB or ORACLE, but don't look for anything about Process, testing or quality on their bookshelves.

(3) **INSUFFICIENT COMMITMENT**:- (1) Many times, S/W Process improvement fails (despite best of intensions) due to lack of true commitment.

(2) Management sets no expectations from the development community around Process improvement, they devote insufficient resources, write no improvement

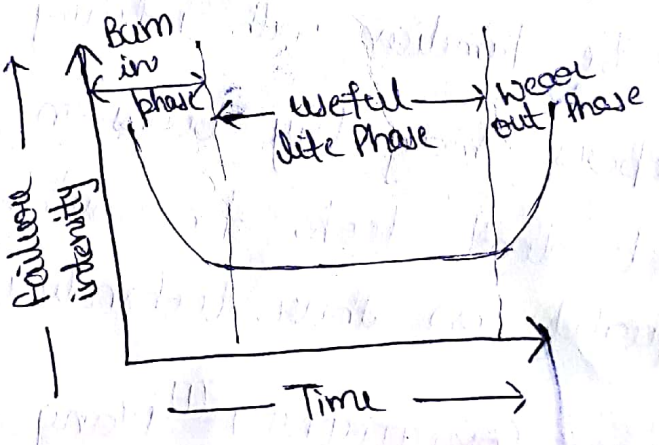
S/W characteristics:- Important characteristics are:

(1.) S/w does not wear out:- There is well known "bath tub curve" in reliability study for hardware Products.

The shape of curve is like "bath tub" and is known as bath tub curve.

There are 3 phases for life of a hardware Product.

1.) Initial phase is BURN-IN PHASE:- where failure intensity is high. It is expected to test Product in industry before delivery.

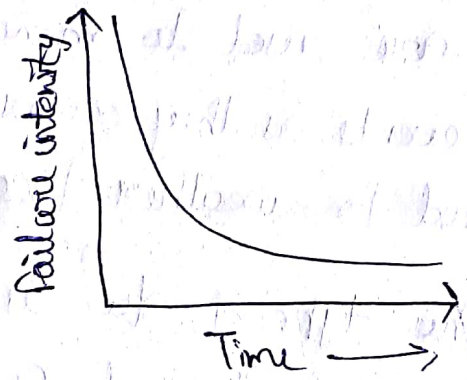


Due to testing and fixing faults failure intensity will come down initially and may stabilise after certain time.

The second phase is "USEFUL LIFE PHASE" where failure intensity is approximately constant and is called "USEFUL LIFE OF A PRODUCT".

We don't have this phase for S/W as it doesn't wear out.

The curve for S/W is given as:



(2) S/W is flexible :- We thought that S/W is flexible. A Pg. can be developed to do almost anything. Sometime this characteristics may be best and may help us to accomodate any kind of change. But many times this char. has made S/W development difficult to plan, monitor and control.

(3) REUSABILITY OF COMPONENTS :- In S/W, every project is a

Software applications :- S/W has become integral part of most of fields of human life.
S/W appl. are grouped into eight areas :

(1) System S/W :- Infrastructure S/W come under this category like Compilers, OS, drivers etc. System S/W is a coll. of pgs. to provide service to other programs.

(2) Real Time S/W :- are used to monitor, control and analyze real world events as they occur.
For eg :- S/W required for weather forcasting.

(3) Embedded S/W :- This type of S/W is placed in "Read Only Memory (ROM)" of Product and control the various functions of Product. The Product can be aircraft, automobile etc. The embedded S/W handles hardware components and is also termed as intelligent S/W.

(4) Business S/W :- This is the largest application area. The S/W designed to process business application is called business S/W. It can be Payroll, account management, file monitoring system etc.

(5) Personal Computer S/W : The S/W's used in Personal Computers are covered in this category. eg's. are word Processors, database mgt, computer graphics, multimedia etc.

6) Artifical Intelligence S/W :- makes use of non-numerical algorithms to solve complex problems that are not amenable to computation or straight forward analysis. Eqs are expert system, signal processing S/W.

7) Web Based S/W :- The S/W related to web application come under this category. Eqs are. HTML, Perl, DHTML.

8) Engineering and Scientific S/W :- Scientific and engineering app. S/W are grouped in this category.

Eqs :- CAD/CAM etc.

SOFTWARE LIFE CYCLE Model

Before studying Software life Cycle Model, we have to study the "SOFTWARE DEVELOPMENT PROCESS".

Acc. to IEEE, S/w life cycle is:

" The Period of time that starts when a S/w Product is conceived and ends when the Product is no longer available for use".

The SDLC typically includes the:

(1.) Feasibility Study
(2.) Requirement analysis and specification Phase
(3.) Design Phase
(4.) Coding Phase
(5.) Test Phase
(6.) Installation and check out phase
(7.) operation and maintenance phase.

four types of models are:- Variety of models have been proposed and are based on tasks involved in developing and maintaining S/w.

(1.) Waterfall
(2.) Prototyping
(3.) Evolutionary
(4.) Spiral

Waterfall Model

The simplest Process model is "Waterfall Model", which states that the Phases are organized in a linear order.

This model has five Phases:

(i) REQUIREMENT ANALYSIS AND SPECIFICATION PHASE :-

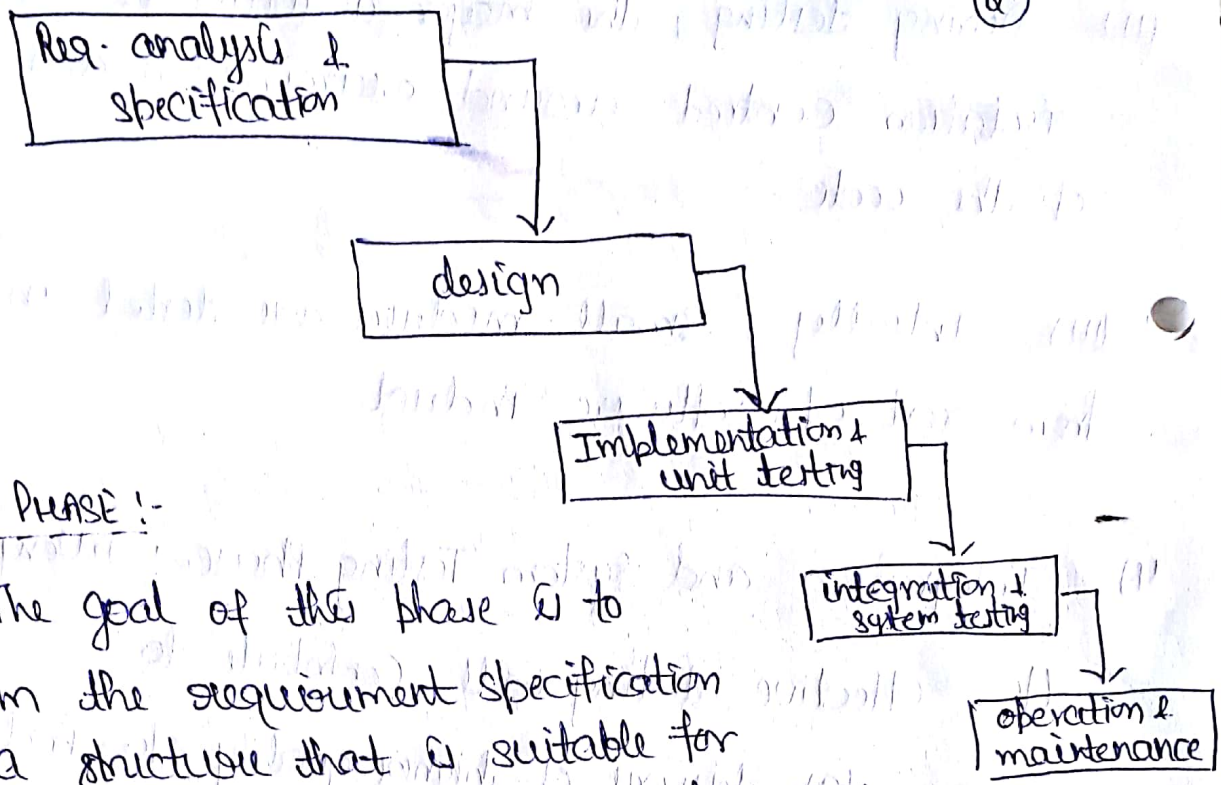
(i) The goal of this phase is to understand the exact requirements of the customer and to document properly.

(ii) The requirement describes the "what" of a system, not the "how".

(iii) This Phase Produce a large document, written in a natural language, contains a description of what the system will do without describing how it will be done.

(iv) The resultant document is known as SRS "Software requirement specification" document.

The SRS may act as contract b/w developer and customer.



(2.) DESIGN PHASE :-

(i) The goal of this phase is to transform the requirement specification into a structure that is suitable for implementation in some programming language.

(ii.) Overall architecture is defined, the high level and detail design work is performed.

(iii) This work is documented and known as "Software design description".

(iv.) The info contained in SDD should be sufficient to begin coding phase.

(3.) IMPLEMENTATION AND UNIT TESTING PHASE :-

(i) During this phase, design is implemented. If SDD is complete, the implementation or coding phase proceeds smoothly, b/c all the info needed by s/w developer is contained in SDD.

(ii.) During testing, the major activities are ~~tested in~~ ~~isolation~~ centred around examination and modification of the code.

(iii.) Initially, small modules are tested in isolation from rest of software Product

4.) Integration and System Testing Phase : (i) Very imp. Phase.

b/c effective testing will Contribute to

(a) delivery of higher quality s/w Product
(b) more satisfied users
(c) lower maintenance cost
(d) more accurate & reliable results.

(ii) It is a very expensive activity and consume one third to one half of the cost of a typical development Project.

(iii) The Purpose of unit testing is to determine that each independent module is correctly implemented and to Combine or integrate all modules integration testing is evolved.

(iv.) System testing involves testing of entire system.

Operation and maintenance Phase: Maintenance is a task that every development group has to face, when s/w is delivered to customer's site, installed and is operational.

s/w maint. is a very broad activity that includes error correction, enhancement of capabilities and optimization.

The purpose of this phase is to preserve the value of s/w over time.

PROBLEMS OF WATERFALL MODEL:-

(1.) It is difficult to define all requirements at beginning of a project.

(2.) This model is not suitable for accomodating any changes.

(3.) It doesn't scale up well to large projects.

(4) Real projects are rarely sequential.

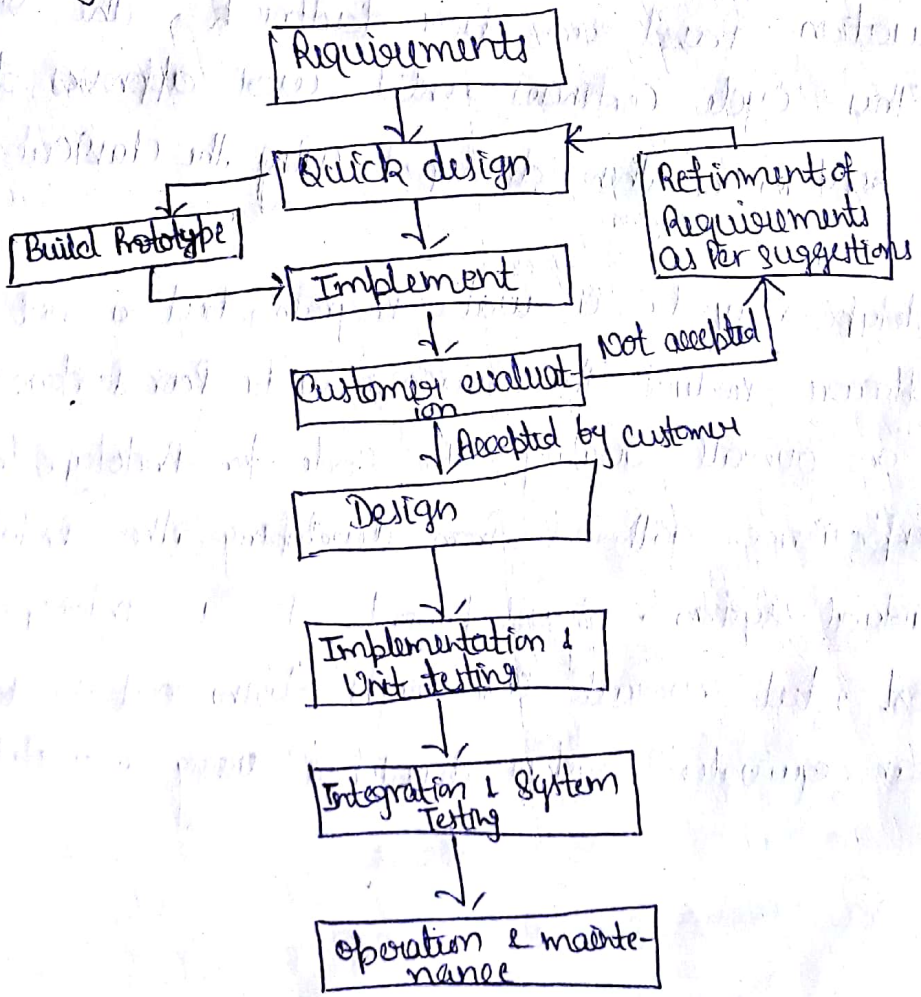
PROTOTYPING MODEL

This model states that to develop a working Prototype of the s/w instead of developing the actual software.

The working Prototype is developed as per current available requirements.

Basically, it has limited function capabilities, low reliability and untested Performance.

The developer use this Prototype to refine requirements and Prepare the final specification document. Because the working Prototype has been evaluated by customer, it is reasonable to expect that the resulting specification document will be correct.



...nt analysis —

The Prototype model suggest that bf developing the actual sfw, a working Prototype of the system should be built. A Prototype is a Partially developed Product.

It has limited functional capabilities, low reliability and inefficient performance.

The model starts with an initial requirements gathering Phase. A Quick design is carried out and Prototype model is built using several shortcuts. The shortcut might involve using inefficient, inaccurate or dummy fns e.g. a fn. may Produce the desired result by using a table look-up rather than performing the actual computation.

The developed Prototype is submitted to the customer for his evaluation. Based on user feedback, the requirements are refined. This cycle continues until user approves the Prototype. The actual system is then developed using the classical waterfall model.

The Prototype may be a usable Program, but is not suitable as final software Product. The reason may be Poor performance, maintainability or overall quality. The Code for Prototype is thrown away, the experience gathered from developing the Prototype helps in developing actual system. Development of a Prototype might involve extra cost, but overall cost might turn out to be lower than that of an equivalent system developed using waterfall model.

equations used for this model.

(1)

(2)

Advantages :- (1.) A Partial Product is built in initial stage. Therefore customer get a chance to see the Product early in life cycle and gives necessary feedback.

(2.) Requirements become more clear resulting into an accurate Product.

(3.) Flexibility in design and development also supported by model.

(4) A user is involved from starting of Project he tends to be more secure, comfortable and satisfied.

Disadvantages :-

(1.) If end user is not satisfied with initial Prototype, he may loose interest in Project.

(2.) After seeing an early Prototype the user may demand the actual system to be delievered soon.

(3.) Poor documentation.

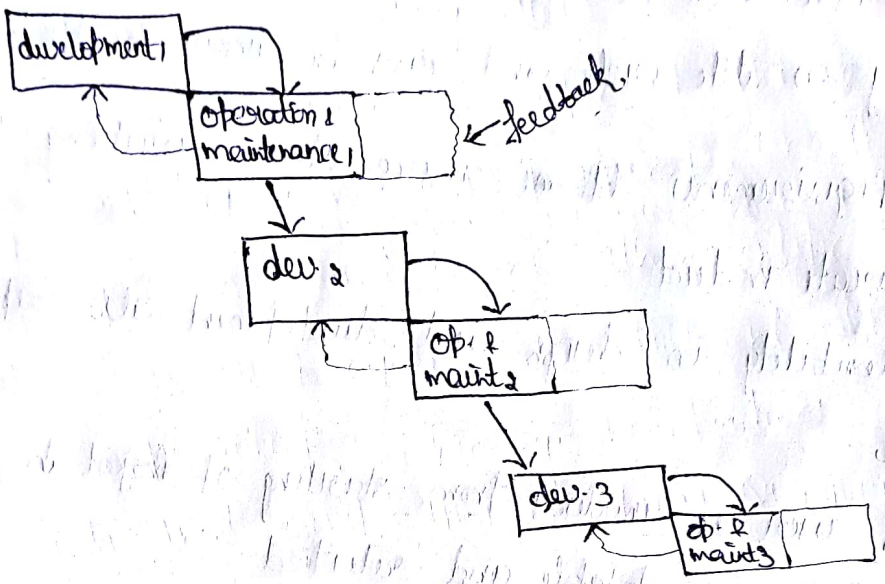
EVOLUTIONARY DEVELOPMENT MODEL

This model accomodates incremental development using experience from earlier increments to help define requirements for subsequent increments.

This model also known as Evolutionary Prototyping, Prototyping / Simulation, Rapid delivery, Evolutionary delivery Cycle or

...ment analysis & spec...

Rapid application delivery (RAD).



Increments are developed in a sequential manner, rather than in parallel, and within each incremental development cycle, there is a normal progression through analysis, design, code, test and implementation, operation and maintenance.

During operations and maintenance stage for an increment, feedback from customer may cause a reiteration of some steps of development for that increment, resulting in a release of that increment.

From customer's point of view, the system will "evolve" as increments that are delivered over time.

From developer's point of view, those requirements that are clear at beginning of project will dictate initial increment and requirements for each development cycle there after will

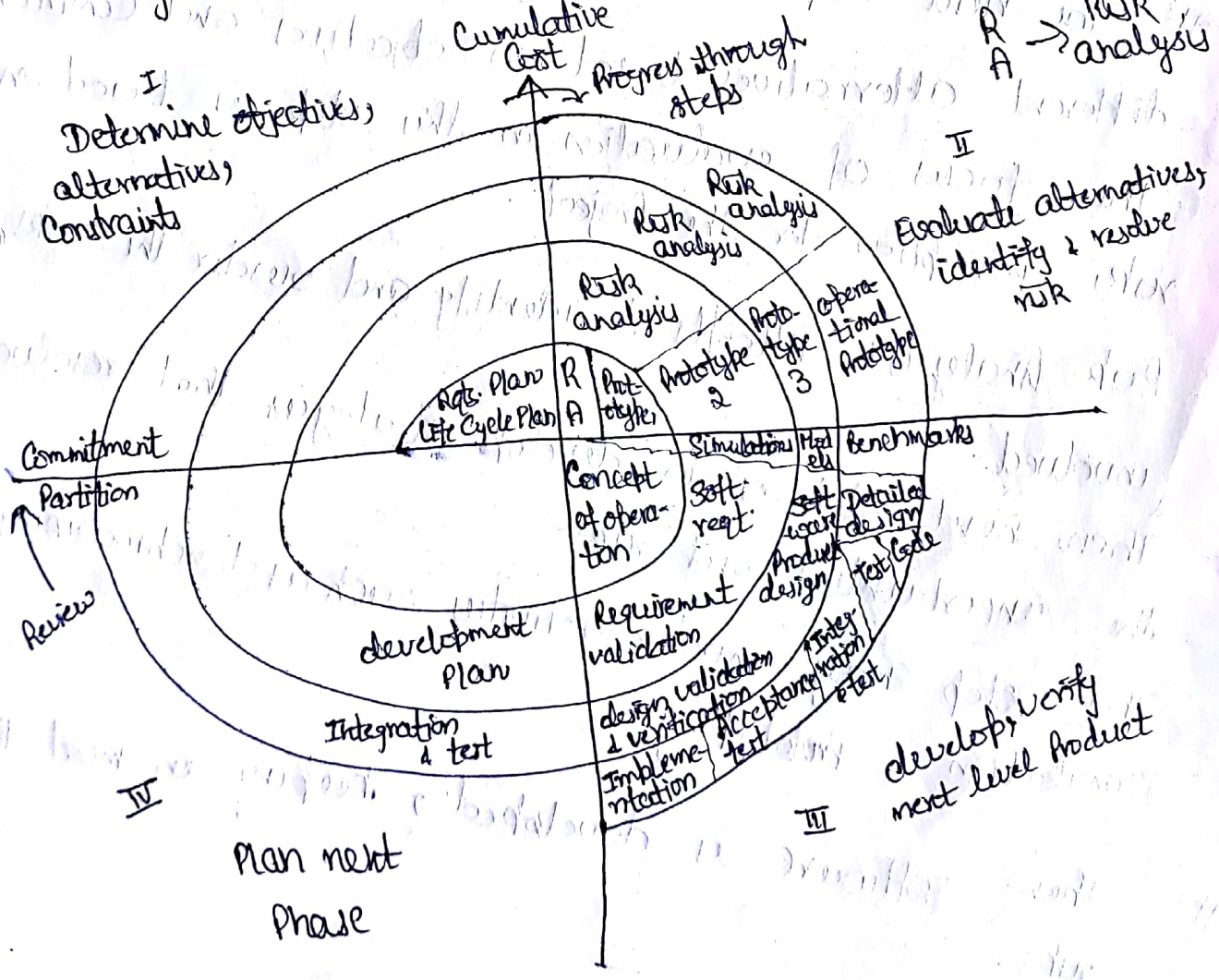
be classified through experience of developing an increments.

Spiral Model

Boehm's Spiral Lifecycle Model: Recent model that has been proposed by Boehm.

As the name suggests, the activities in this model can be organized like a spiral that has many cycles.

The spiral model combines elements of waterfall lifecycle model, along with an emphasis on the use of risk management techniques.



The radial dimension represents the cumulative cost incurred in accomplishing steps.

The angular dimension represent the progress made in completing each cycle of spiral. One phase is split roughly into four sectors of major activities: Planning, Risk analysis, development, Assesment

Each cycle in spiral begins with identification of objectives for that cycle, the different alternatives that are possible for achieving the objectives, and Constraints that exist. This is FIRST QUADRANT OF CYCLE (Upper-left quardant).

The next step in the cycle is to evaluate these different alternatives based on objectives and Constraints. The focus of evaluation in this step is based on risk perception for the project.

Risk Analysis: Attempts to identify and resolve the risks involved.

Then next step is to develop strategies that resolve the uncertainties and risks.

This step may involve activities such as benchmarking, simulation & Prototyping.

Then software is developed, keeping in mind the risks.

Finally next stage is plan i-e Plan next Phases.

An imp. feature of Spiral model is that each phase is Completed with a review by people Concerned with the Project. This review Consists of a review of all Products developed upto that Point and includes Plan for next cycle.

Advantages :- (1) wide range of options to accomodate the good feature of other life cycle models.

(2) It also incorporates S/w quality objectives into S/w development.

(3) The risk analysis and validation steps eliminate errors in early Phases of development.

Disadvantages :- This model has some difficulties that need to be resolved bfr it can be a universally applied life cycle model. These difficulties include lack of explicit Proces guidance in determining objectives, Constraints, alternatives, Providing more flexibility etc.