Software Engineering

UNIT-1 Introduction to software Engineering: The evolving role of software, changing nature of s/w, legacy software, software myths. A Generic view of process: software engineering - A layered technology, a process framework The Capability maturity model Integration (CMMI), Process patterns, process assessment, personal and team process models. Process models: The waterfall model, Incremental process models, Evolutionary process models, specialized process models, The Unified Process.

1) S/w Engineering: A Practitioners Approach, :- Roger S Pressman, 6th edition
2) s/w E :- Ian Sommerville 7th edition, Pearson edu
3) A.A. Puntambekar

* Introduction to software Engineering

Introduction:-

→ Software engineering is the most important technology with rapid development

→ National economy is also dependent on this technology

→ Day by day more and more systems are getting controlled by s/w.

→ Hence it becomes necessary to understand the s/w development as an engineering displine

→ In this chapter, we will also discuss various categories of software and also various challenges in s/w engineering

Definition:- s/w engineering is an establishment and use of sound engineering principles, methods, tools, that can be used to produce produce high quality software that is reliable and works efficiently on real machines.

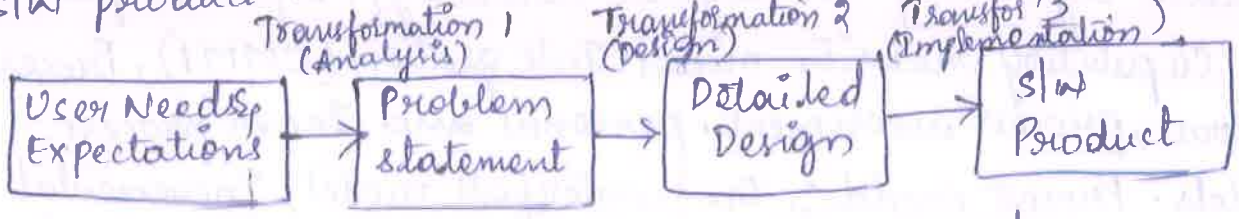
[or]

→ The application of a systematic, disciplined, quantifiable approach to the development, operation and maintenance of s/w, that is the appln of engineering to s/w

Software :- It is more than just a program code. A program is an executable code, which serves some computational purpose.

→ s/w is considered to be collection of executable programming code, associated libraries and documentations

→ s/w, when made for a specific requirement is called s/w product



s/w Development Process Transformation

* Evolving Role of s/w :-

The evolving role of s/w means changing role of s/w. Basically any s/w appears in 2 roles:

1) s/w as a Product 2) s/w as a Process / As a Vehicle

1) s/w as a Product :- Being a product the role of s/w can be recognised by its computing potentials, hardware capabilities (use) (having the capacity to develop into something in future) and accessibility of n/w computers by the local H/W. Being a product it also acts as an information transformer i.e., producing, managing, modifying and conveying the source of information

2) s/w as a Process :- Being a process the s/w acts as a vehicle for driving the product. In this role the duty of s/w is to control the computer or to establish communications b/w the computers

→ The role of s/w is significantly changing over past decade

→ There are many factors affecting the role of s/w & those are :-

1) Changes in computer architectures (Right from Pentium to supercomputers)

2) Improvement in hardware performance.

3) Vast Increase in amount of memory.

→ Wide variety of i/p & o/p's (Ranging from simple text to multimedia videos)

→ Following table presents the different roles of s/w in different era of computing
↳ a long & distinct period of history

Era of Computing	s/w
Early years	Batch Processing, custom s/w
Second Era	Multi user Real time systems, Database systems, Product s/w
Third Era	Distributed systems
Forth Era	Object oriented systems, Expert systems Parallel computing, N/w computers
Fifth Era	Web technologies, mobile Computing

→ But this evolutionary role of s/w brings some crucial problems. Here are some sample problems encountered due to evolution on s/w.

1) Advances in h/w demand for more capable s/w.
2) Ability to build new programs cannot meet the demand for new programs & such programs are not sufficient for business & market needs.
3) Vast use of computer based systems brings less use of manpower and ultimately society becomes more dependent on machine & not on man.
4) Constant struggle for high reliability & quality s/w.

* What is software Engineering?

s/w engineering is a discipline in which theories, methods & tools are applied to develop professional s/w

→ In s/w engineering a systematic & organized approach is adopted
→ Based on the nature of dereto-problem and development constraints various tools & techniques are applied in order to develop quality s/w.

* s/w :- s/w is nothing but a collection of computer programs & related documents that are intended to provide desired features, functionalities & better performance.
s/w products may be
1) Generic → that means developed to be sold to a range of different customers.
2) Custom → that means developed for a single customer according to their specification.

* s/w Characteristics :-
s/w development is a logical activity & therefore it is important to understand basic characteristics of s/w.
→ some important characteristics of s/w are
1) ~~some~~ s/w is engineered, not manufactured
2) s/w does not ware out
3) Most s/w is custom built rather than being assembled from components.

1) s/w is engineered, not manufactured :-
s/w development & h/w development are 2 different activities.
→ A good design is a backbone for both the activities.
→ Quality problems that occur in h/w manufacturing phase cannot be removed easily.
→ On the other hand, during s/w development process such problems can be rectified.
→ In both the activities, developers are responsible for producing quantitative product.

2) S/W does not wore out :-

→ In early stage of H/W development proces the failure rate is very high because of manufacturing defects
→ But after correcting such defects the failure rate gets reduced.
→ The failure rate remains constant for some period of time and again it starts increasing because of environmental maladies (extreme temperature, dusts, & vibrations).
→ On the other hand S/W does get affected from such environmental maladies.
→ Hence ideally it should have an "idealized curve"
→ But due to some undiscovered errors the failure rate is high & drops down as soon as the errors get corrected.
→ Hence in failure rating of S/W the "actual curve" is as shown below.

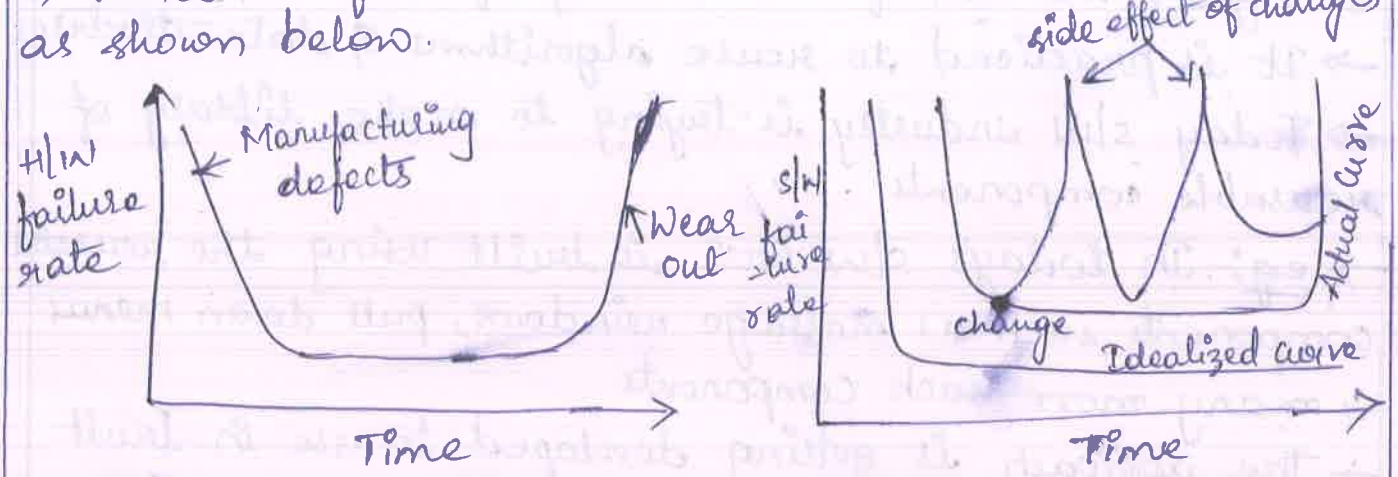


Fig :- Failure Curves for H/W & S/W

→ During the life of S/W if any change is made, some defects may get introduced.
→ This causes failure rate to be high.
→ Before the curve can return to original steady state another change is requested & again the failure rate becomes high.
→ Thus the failure curve looks like a spike. Thus frequent changes in S/W cause it to deteriorate (get worse)

→ Another issue with s/w is that there are no spare parts for s/w.
→ If h/w component wears out it can be replaced by another component but it is not possible in case of s/w
→ ∴ s/w maintenance is more difficult than the h/w maintenance

3) Most s/w is custom built rather than being assembled from components :-

→ While developing any h/w product at first circuit design with desired functioning properties is created. Then required h/w components such as ICs, capacitors & registers are assembled according to the design, but this is not done while developing s/w product
→ Most of the s/w is custom built
→ However, now the s/w development approach is getting changed & we look for reusability of s/w components.
→ It is practiced to reuse algorithms & data structures
→ Today s/w industry is trying to make library of reusable components.
→ eg:- In today's s/w GUI is built using the reusable components such as message windows, pull down menus & many more such components.
→ The approach is getting developed to use-in-built components in the s/w.
→ This stream of s/w is popularly known as component engineering.

Changing Nature of S/W :-

S/W can be applied in a situation for which a predefined set of procedural steps (algorithm) exists.

→ Based on a complex growth of s/w it can be classified into following categories.

1) System s/w :- It is collection of programs written to service other programs.

→ Typical programs in this category are compiler, editors & assemblers.

→ The purpose of the system s/w is to establish a comm with the H/W.

2) Appl" s/w :- It consists of standalone programs that are developed for specific business need.

→ This s/w may be supported by DB systems.

3) Engineering / scientific s/w :- This s/w category has a wide range of programs from astronomy to volcanology from automative stress analysis to space shuttle orbital dynamics & from molecular biology to automated manufacturing.

→ This s/w is based on complex numeric computations.

4) Embedded s/w :- This category consists of program that can reside within a product or system.

→ Such s/w can be used to implement & control features & functions for the end-user & for the system itself.

5) Web Applications :- Web appl" s/w consists of various web pages that can be retrieved by a browser.

→ The web pages can be developed using programming languages like JAVA, PERL, CGI, HTML, DHTML

- JAVA → James Gosling Arthur Vanhalf Andy Bechtolsim
- PERL → Practical Extraction & Reporting language
- CGI → Canadian global Inform" Technology
- DHTML → Dynamic Hyper Text Markup language.

6) Artificial Intelligence s/w :- This kind of s/w is based on knowledge based expert systems.
→ Typically this s/w is useful in robotics, expert systems, image & voice recognition, artificial neural n/w's, theorem proving and game playing.

* Legacy s/w :-

Legacy : Normal terms :
something which you got in inheritance
In terms of s/w industry :-
something which is old but still in use & difficult to replace.
Legacy system :- The combination of legacy s/w & h/w overall is known as legacy system.

* Legacy s/w :-
Legacy s/w is outdate s/w, but still in use by the compan
-ies ~~still in use by the companies~~ because it is fulfill the need of company & it is difficult for company to move again on the new version of the s/w.
→ Most of the time customize s/w become the legacy s/w because the client only once define the requirements & get the custom built s/w & if the requirements do not change for a decade they try to keep using it.

* s/w Myths :- → Values / beliefs
→ There are some misbelieves in the s/w industry about the software & process of building s/w.
→ for any s/w developer it is a must to know such beliefs & reality about them
→ Here are some typical myths :-

1) Myth :- Using a collection of standards & procedures one can build s/w.
Reality :- Eventhough we have all standards and procedures with us for helping the developer to build s/w, it is not possible for s/w professionals to build desired product. This is because - the collection which we have should be complete, it should reflect modern techniques & more importantly it should be adaptable.
→ It should also help the s/w professional to bring quality in the product.

2) Myth :- Add more people to meet deadline of the project
Reality :- Adding more people in order to catch the schedule will cause the reverse effect on the s/w project i.e., s/w project will get delayed.
→ Because, we have to spend more time on educating people or informing them about the project.

3) Myth :- If a project is outsourced to a 3rd party then all the worries of s/w building are over.
Reality :- When a company needs to outsource the project then it simply indicates that the company does not known how to manage the projects.
→ Sometimes the outsourced projects require proper support for development.

4) Myth :- Even if the s/w requirements are changing continuously it is possible to accommodate these changes in the s/w.
Reality :- It is true that s/w is a flexible entity but if continuous changes in the requirements have to be incorporated then there are chances of introducing more & more errors in the s/w.
→ Hy the additional resources & more design modifications may be demanded by the s/w.

5) Myth:- We can start writing the program by general problem statements only. later on using problem description we can add up the required functionalities in the program.
Reality:- It is not possible each time to have comprehensive problem stmt. we have to start with general problem stmts, however by proper communication with customer & the s/w professionals can gather useful infos.
→ The most important thing is that the problem stmt should be unambiguous to begin with.

6) Myth:- Once the program is running then its over).
Reality:- Even though we obtain that the program is running major part of work is after delivering it to customer.

7) Myth:- Working program is the only work product for the s/w project.
Reality:- The working program/s/w is the major component of any s/w project but along with it there are many other elements that should be present in the s/w project such as documentation of s/w, guideline for s/w support.

8) Myth:- There is no need of documenting the s/w project, it unnecessarily slows down the development process.
Reality:- Documenting the s/w project helps in establishing ease in use of s/w.
→ It helps in creating better quality.
→ Hence documentation is not wastage of time but it is a must for any s/w project.

* Goals/Objectives of s/w Engineering:-
While developing s/w following are common objectivities
1) Satisfy user requirements
2) High reliability
3) Low maintenance costs
4) Delivery on time
5) Low production costs
6) High performance
7) Ease of reuse.

1) Satisfy user requirements :- Many programmers simply dont do what the end user wants because they do not understand user requirements.
→ Hence it becomes necessary to understand the demand of end user & accordingly s/w should be developed.
2) High reliability :- Mistakes or bugs in a program can be expensive in terms of human lives, money & customer relation.
→ For instance Microsoft has faced many problems because earlier release of windows has many problems
→ Thus s/w should be delievered only if high reliability is achieved.
3) Low maintenance costs :- Maintenance of s/w is an activity that can be done only after delivering the s/w to the customer.
→ Any small change in s/w should not cause restructuring of whole s/w.
→ This indicates that the design of s/w has poor quality.
4) Delivery on time :- It is very difficult to predict the exact time on which the s/w can be completed. But a systematic development of s/w can lead to meet the given deadline.
5) Low production costs :- The s/w product should be cost effective
6) High performance :- The high performance s/w are expected to achieve optimization in speed & memory usage.
7) Ease of reuse :- Use same s/w in different systems & s/w.
→ Environments reduce development costs & also improve the reliability.
→ Hence reusability of developed s/w is an important property.

* Challenges in s/w Engineering :-

1) Coping with deagacy systems :- old, valuable systems must be maintained & updated.
→ H/w is evalved faster than s/w.
→ If original developer have moved on managing, maintaining or integrating of s/w becomes a critical issue.

2) Heterogeneity Challenge :- Sometimes systems are distributed and include a mix of H/w & s/w.

3) Delivery time challenge :- This is increasing pressure for faster delivery of s/w.
As the complexity of systems that we develop increase, this challenge becomes harder.

→ The main objective of s/w engineering is to adopt systematic, disciplined approach while building high quality s/w.

* A Generic (→ characteristic of special) view of Software Process :-

→ Software engineering is the establishment & sound engineering principles applied to obtain reliable & efficient s/w in an economical manner.
→ s/w engineering includes process, management techniques, technical methods, and the use of tool.
→ While building any s/w, the s/w process provides the interaction b/w user and developer.

* A Layered Technology :-

→ s/w engineering is a layered technology.
→ Any s/w can be developed using these approaches.
→ various layers ~~can~~ on which the technology is based are quality focus layer, process layer, methods layer, tools layer.

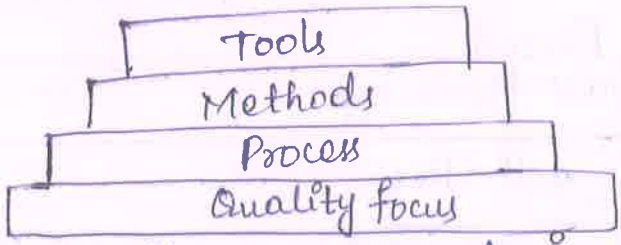


Fig:- s/w engineering layers

→ A disciplined quality management is a backbone of s/w engineering technology.
→ Process layer is a foundation of s/w engineering.
→ Basically, process defines the framework for timely delivery of s/w.
→ In method layer the actual method of implementation is carried out with the help of requirement analysis, designing, coding using desired programming constructs and testing
→ s/w tools are used to bring automation in s/w development process.
→ Thus s/w engineering is a combination of process, methods & tools for development of quality s/w.

* Software Process :- It can be defined as the structural set of activities that are required to develop the s/w system.
The fundamental activities are
• Specification • Design & Implementation • Validation • Evolution
→ A s/w process model is an abstract representation of a process.
→ It presents a description of a process from some particular perspective.

* Common Process Framework :- → a basic structure underlying a system, concept or text
→ The process framework is required for representing the common process activities

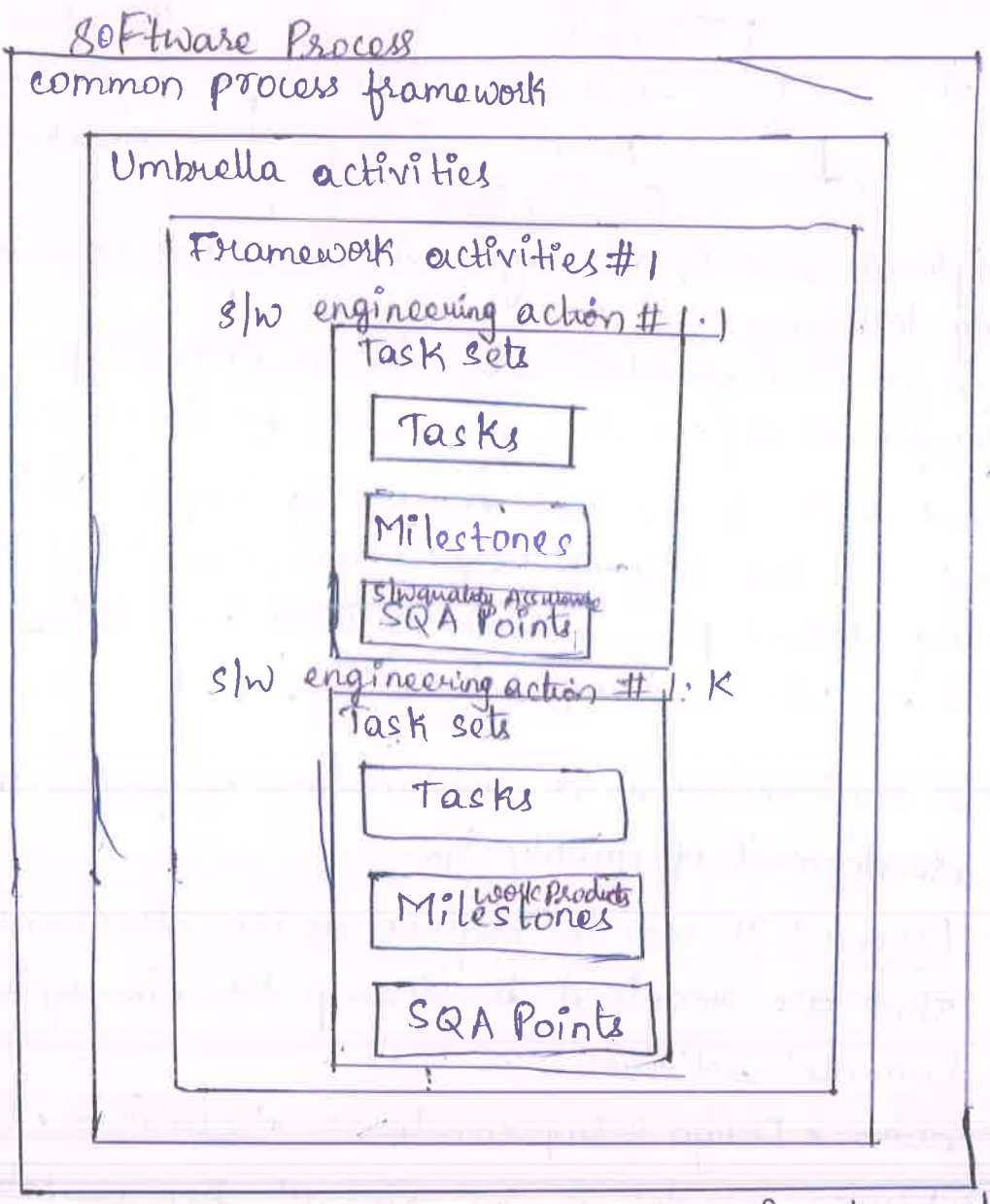


Fig: s/w Process Framework.

→ As shown in figure the s/w process is characterized by process framework activities, task sets and umbrella activities.

* Process framework activities:- • communication:-
• By communicating customer requirement gathering is done.
• Planning - Establishes engineering work plan, describes technical risks, lists resource requirements, work products produced, & defines work schedule.
• Modelling - The s/w model is prepared by
• Analysis of requirements
• Design
• Construction - The s/w design is mapped into a code by:
• Code generation
• Testing
• Deployment - The s/w delivered from customer evaluation & feedback is obtained

* Task sets :- The task sets defines the actual work done in order to achieve the s/w objective. The task set is used to adopt the framework activities & project team requirements using
  - collection of s/w engineering work tasks
  - Project milestones.
  - s/w quality assurance points.

Umbrella activities :- The umbrella activities occur throughout the process. They focus on project management, tracking & control. The umbrella activities are.

1) s/w project tracking & control :- This is an activity in which s/w team can assess progress & take corrective action to maintain schedule.
2) Risk Management :- The risks that may affect project outcomes or quality can be analyzed.
3) s/w quality assurance :- These are activities required to maintain s/w quality.
4) Formal technical reviews :- It is required to assess engineering work products to uncover and remove errors before they propagate to next activity
5) s/w configuration management :- Managing of configuration process when any change in the s/w occurs.
6) Work product preparation & production :- The activities to create models, documents, logs, forms & lists are carried out
7) Reusability management :- It defines criteria for work product reuse.
8) Measurement :- In this activity, the process can be defined & collected. Also project & product measures are used to assist the s/w team in delivering the required s/w.

* The Capability maturity Model Integration (CMMI) :-

→ It is a proven industry framework to improve product quality & development efficiency for both H/w & S/w

→ Sponsored by US Department of Defence in cooperation with carnegie Mellon University & the S/w Engineering Institute (SEI) in 1987

→ Many companies have been involved in CMMI definition such as Motorola and Ericsson

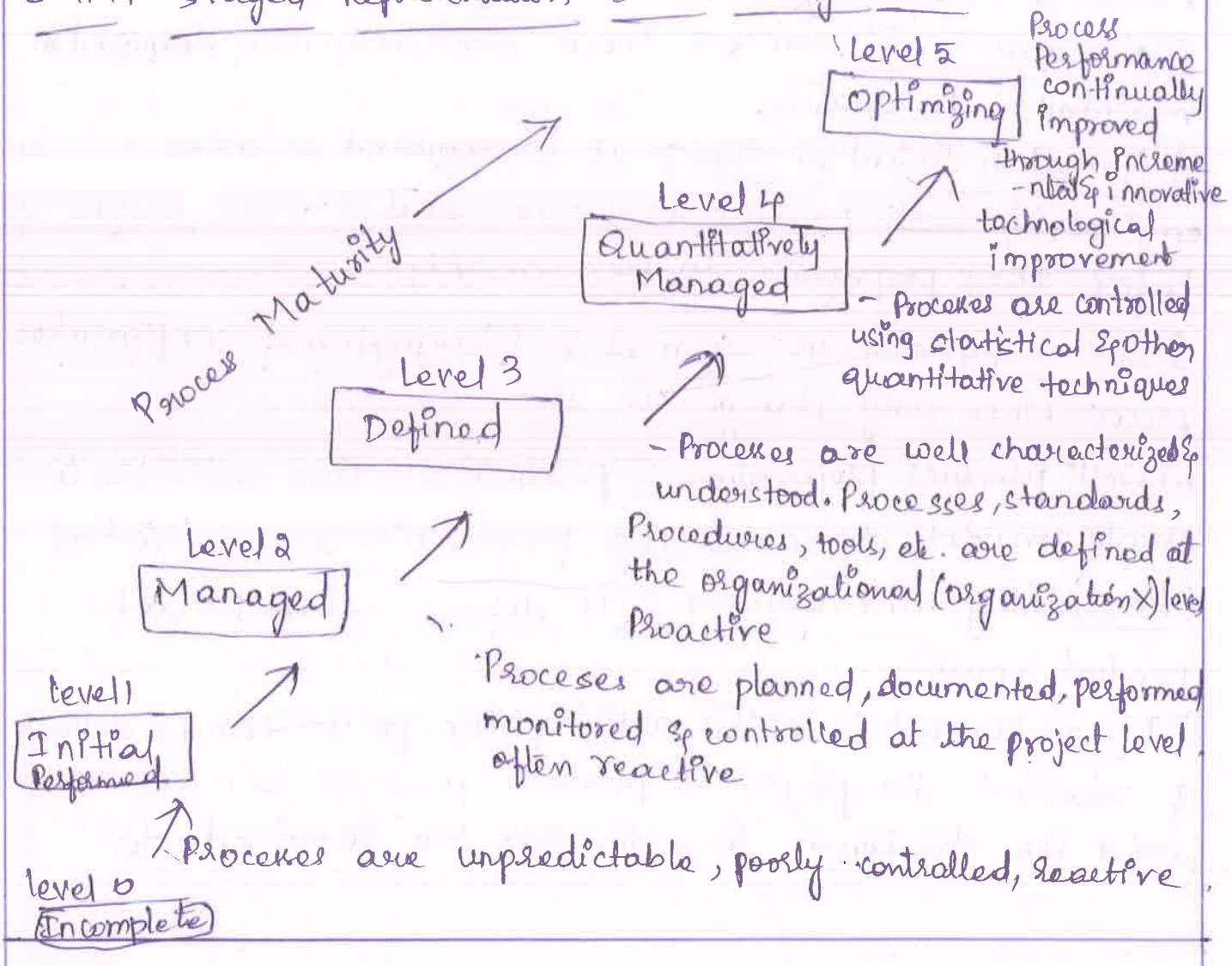
→ CMMI has been established as a model to improve business results

→ CMMI, staged, uses 5 levels to describe the maturity of the organization, same as predecessor CMM.

→ vastly improved version of the CMM.

→ Emphasis on business needs, integration & institutionalization

CMMI Staged Representation - 5 Maturity levels



Maturity level 0 Initial :- Incomplete :-)

→ the CMMI represents a process meta-model in 2 different ways
1) as a continuous model 2) as a staged model.

→ The continuous CMMI meta-model describes a process in 2 dimensions.

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

level 0 : Incomplete :- The process area (e.g., requirements management) is either not performed or does not achieve all goals & objectives defined by the CMMI for level 1 capability

level 1 : Performed :- All the specific goals of the process area have been satisfied. Work tasks required to produce defined work products are being conducted.

level 2 : Managed :- All level 1 criteria have been satisfied. In addition all work associated with the process area conforms to an organizationally defined policy

level 3 : Defined :- All level 2 criteria have been satisfied. In addition, the process is "tailored from the organization's set of standard processes according to the organ" tailoring guidelines & contributes work products..

level 4 :- Quantitavely managed :- All level 3 criteria have been achieved. In addition, the process area is controlled & improved using measurement & quantitative assessment.

level 5 :- Optimized :- All level 4 criteria have been achieved. In addition, the process area is adapted & optimized using statistical means to meet changing customer needs & to continually improve the efficacy of the process area under consideration

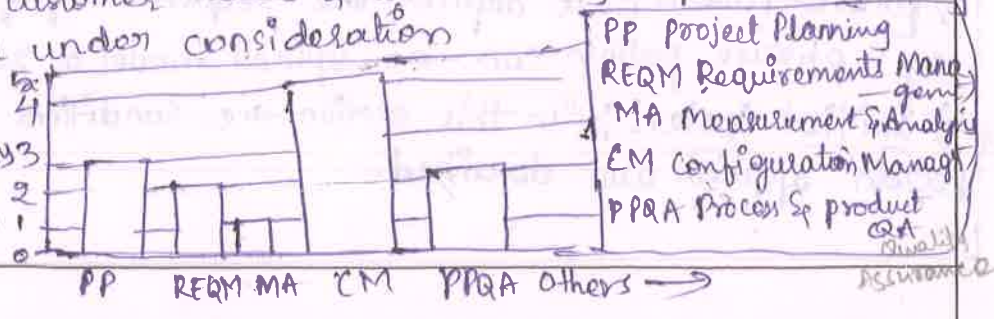


Fig:- CMMI Process area capability profile.

* Process Patterns :- S/W process can be defined as a collection of patterns that define a set of activities, actions, work tasks, work products and required to develop computer s/w.

→ A Process pattern provides us with a template - A consistent method for describing an important characteristic of the s/w process.

→ Patterns can be defined at any level of abstraction

→ In some cases, a pattern might be used to describe a complete process.

→ In other situations patterns can be used to describe an important framework activity (eg:- planning) or a task within a framework activity (eg:- project-estimating).

→ Ambler has proposed the following template for describing a process pattern :-

*1) Pattern Name :- The pattern is given a meaningfull name that describes its function within the s/w process (eg:- customer-comm)

*2) Intent :- The objective of the pattern is described briefly.
eg:- The intent of customer-comm" is "to establish a collaborative relationship with the customer in an effort to define project scope, busines requirements and the project constraints.

*3) Type :- The Pattern type is specified. Ambler suggests 3 types:
→ 1) Task Pattern 2) Stage Pattern 3) Phase Patterns.

1) Task Pattern :- It represents the s/w engineering action or work task which is a part of process
Eg:- Formal Technical review is a task pattern.

2) Stage Pattern :- It defines the process framework activity. A framework activity has multiple work tasks; Hence stage pattern consists of multiple task patterns
Eg:- Coding phase is a stage pattern.

3) Phase Patterns :- It defines the sequence of framework activities
eg:- phrase pattern can be spiral model or rapid prototype model

*4) Initial Context :- In this section the conditions under which the pattern applied are described.

→ Sometimes the entry conditions must be true before the process begins
In this section following issues need to described.
1. The set of organisational or team related activities that have already occured.
2. The list of entry state processed.
3. Already existing s/w engineering or project related information.

5) Problem:- Under this section the problem is mentioned for which the pattern is to be described.
Eg:- Insufficient requirements is a problem.
→ That means customers are not sure about what they want exactly.
→ They could not specify the requirements in proper manner.

6) Solution:- Every problem for which pattern has to be described should be accompanied with same solution.
ex:- The problem of insufficient requirements has solution. i.e.,
→ Establish effective communication with the customer
→ Ask questions in order to obtain meaningful requirements.
→ conduct timely reviews to modify/redefine the requirements.
→ This soln will help the s/w developer to get useful information before the actual work starts.

7) Resulting context:- It describes the results after successful imple'n of Pattern
→ The resulting context should have following type of infor'n on succeful completion of pattern —
1) The team-related or organizational activities that must have occured
2) Exit state for the process.
3) The s/w engineering infor'n or project infor'n that has been developed.

8) Known uses/Examples:-
→ The specific instances or appl'n's in which the described pattern is useful should be mentioned under this section.
→ In other words we describe applicability of the pattern.
→ eg:- spiral model is useful for the large scale projects in which work products must be examined in several iterations.

* Process Assessment :- Normally process is suffered by following problems
1. The s/w has to be delivered on time
2. The s/w should satisfy customer needs & requirements.
3. The s/w " posses the long term quality characteristics

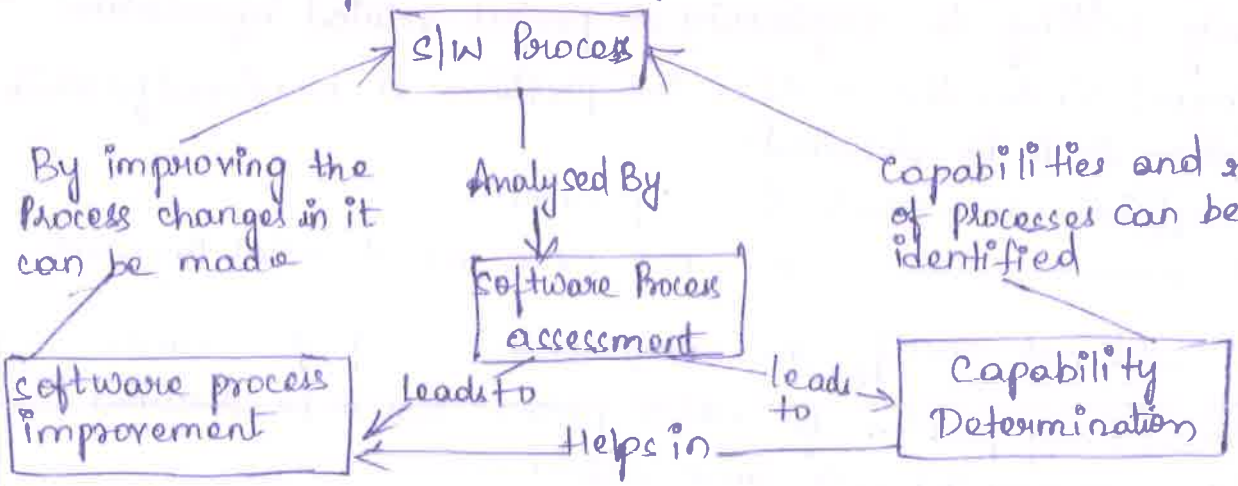


Fig :- s/w process Assessment.

→ Process assessment is an activity in which it is ensured whether the s/w meets the basic criteria for successful s/w engineering Project

→ Following Approaches are used for s/w assessment :-

Standard CMMI assessment method for process improvement :- (to something)

It is a 5 step process assessment model

→ These 5 steps are initiating, diagnosing, establishing, acting & learning.

→ This model makes use of SEI CMM as the base model

CMM-Based appraisal for internal process Improvement :-

→ This method provides the diagnostic technique for internal process assessment

SPICE :- Using this standard all the requirements are analysed for s/w process assessment. This standard helps in doing an objective evaluation on efficiency of any process.

ISO 9001:2000 :- This is a popularly used standard for improving the overall quality of the organization.

→ The ISO has developed this standard.

* Personal and Team Process models:-

→ Watts Humphrey developed Personal s/w Process (PSP) and Team s/w Process (TSP) at the SEI (s/w Engineering Institute) at Carnegie Mellon in the mid-1990's.

→ The PSP is a SEI technology that brings discipline to the s/w development habits of individual s/w engineer.

→ It helps in dramatically improving product quality, increasing cost and schedule predictability and reducing development cycle time for s/w

→ The TSP is a complementary SEI technology that enables teams to develop s/w products more effectively.

→ TSP helps a team of engineers how to produce quality products for planned costs & on aggressive schedules.

→ PSP is like applying Six Sigma to s/w development

* PSP (Personal s/w Process):- (Process Improvement)

→ The computer s/w is built using various processes developed by every individual.

→ There are different kinds of s/w processes; random, intended for specific purpose or may be changing each time.

→ These processes are developed by individuals in an organization.

→ Watts Humphrey suggested that to make each process effective every individual who is developing them has to change themselves first

→ Under PSP every practitioner is made responsible for controlling the quality of corresponding processes.

→ Under PSP model 5 framework activities are suggested as follows.

Fig:- Framework activities in PSP

1) Planning:- In this activity, requirements are identified based on these requirements size, resource estimates & defect estimates are made.
→ All metrics (system) are arranged in template form.
→ Development task are identified & project schedule is created.

2) High level Design:- The specification is created 1st & then component level design is created. To resolve the uncertainty prototypes are created.

3) Review:- Formal technical reviews are made for uncovering the errors. Metrics are formed for important tasks & recorded.

4) Development :- Under this activity code is generated. The generated code is then reviewed, modified and tested. Metrics are formed for important tasks & recorded.

5) Postmortem :- Using the collected measures & metrics the effectiveness of the process is analysed. These measures & metrics should direct certain changes in the process in order to improve its efficiency.

→ The emphasis of PSP is to identify errors in the early stage of s/w development. Using the systematic approach of PSP every work product is assessed with great care.

2) Term Process Models (TSP) :- It is designed to produce strategy & set of operational procedures for using disciplined s/w methods at team levels. The goal of TSP is to have self directed project team for producing high quality s/w. Following are the objectives of TPM.

1. Build the self directed team for planning the s/w project in systematic manner. The normal size of team should be 3 to 20 s/w engineers.
2. Indicate the project manager for the coaching needed by the team members for the performance improvement.
3. Using CMM level 5 (optimizing level), improve the s/w process.
4) Provide the improvement guidance for the s/w team
5) Provide the university teaching.

There are 5 framework activities in TSP.

- Launch
- High level Design
- Implementation
- Integration & test
- Postmortem.

→ In TSP there is great use of scripts, standards & forms.
→ These documents help the team to seek guidance in their work & improve the overall quality of the work product.

Introduction

* **Process Models :-** Process models are proposed in order to adopt systematic approach in s/w development. These models define the distinct set of activities, tasks and work products that are required to create high quality s/w.

→ There are different process models with different terminologies but their generic framework activities are almost same.

→ To be more specific these process models can also be called as prescriptive process model because they prescribe the process elements such as ~~prescriptive process model because they prescribe~~ the framework activities, SE actions, tasks, work products & quality

→ All s/w process models can accommodate the general framework activities hence they are also recognised as generic s/w process models.

→ In this we discuss various generic process models along with their merits and demerits & applicability

* **Process Models :-** This is used to develop a efficient process. The process model can be defined as an abstract represen. of process.

→ The process model is chosen based on nature of s/w project.

Generic s/w process Models are :-

1) The Waterfall Model :- ~~Separate~~ & distinct phases of specifn & development
2) Prototyping Model :- A quick design approach is adopted
3) Incremental Models :- It emphasizes on short development cycle
   ↳ RAD (Rapid Applⁿ & Development) model
4) Evolutionary Process Models :- Specification, development & validation are interleaved
   - Incremental Model
   - Spiral Model
   - WINWIN Spiral Model
   - Concurrent Development

1) Waterfall Model

Advantages
→ Simple & Easy
→ Easy to manage
→ works well for smaller Projects
→ Results are well documented

Disadvantages
→ Risk & uncertainity
→ Not for big & complex Projects
→ Not for Projects where requirements are changing
→ i.e., can't accomodate changing requirements

* Life Cycle Models :-

→ A life cycle is the sequence in which a project specifies, prototypes, designs, implements, tests and maintains a piece of s/w.
→ In SE, the life cycle model depicts various stages of s/w development process.
→ Using life cycle model various development issues can be solved at the appropriate time.

1) Waterfall Model :-

→ The waterfall model is also called as linear-sequential model or classic life cycle model.
→ It is the oldest s/w paradigm. It can be done in small & large Projects
→ This model suggests a systematic, sequential approach to s/w development.
→ The s/w development starts with requirements gathering Phase
→ Then progresses through analysis, design, coding, testing and Maintenance

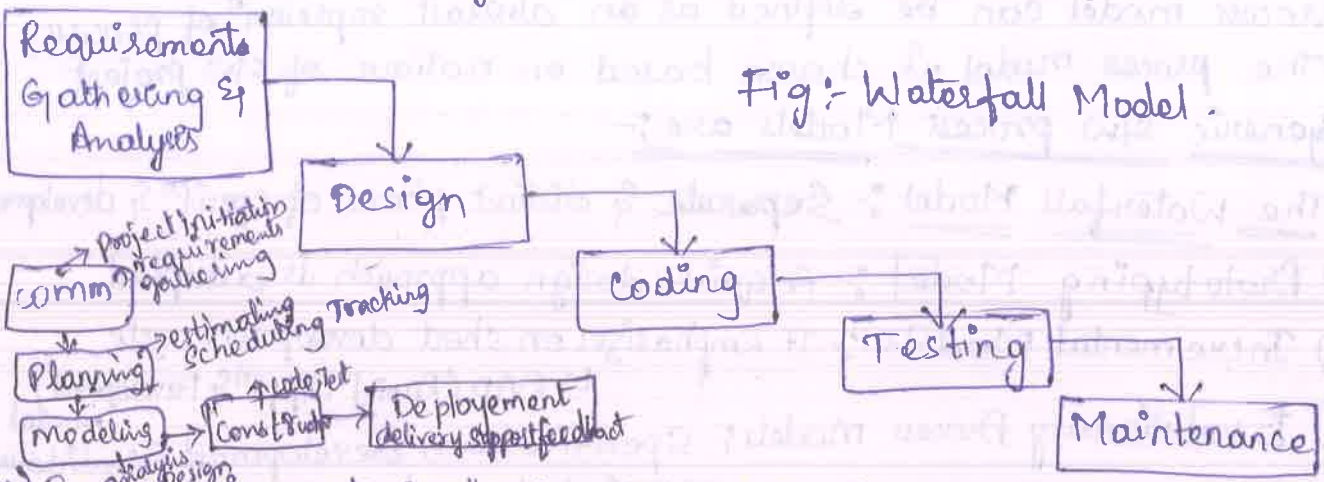


Fig :- Waterfall Model.

→ 1) In requirement Gathering & analysis phase, the basic requirements of the system must be understood by s/w engineer, who is also called Analyst. The infor^n domain, fun^n, behavioural requirements of the system are understood. All these requirements are then well documented & discussed further with the Customer, for reviewing.
→ 2) The design is an intermediate step b/w requirements analysis & Coding.
→ Design focuses on program attributes such as -

- Data structure
- s/w architecture
- Interface Representation
- Algorithmic details.

→ The requirements are translated in some easy to represent form using which coding can be done effectively and efficiently.
→ The design needs to be documented for further use.
→ Coding is a step in which design is translated into machine readable form. If design is done with sufficient detail then coding can be done effectively. Programs are created in this phase
→ Testing begins when coding is done. While performing testing the major focus is on logical internals of the s/w.
→ The testing ensures execution of all the paths, functional behaviours.
→ The purpose of testing is to uncover errors, fix the bugs & meet the customer requirements.
→ Maintenance is the longest life cycle phase.
→ When the system is installed & put in practical use then errors may get introduced, correcting, such errors & putting it in use is the major purpose of maintenance activity.
→ Similarly enhancing system's services as new requirements are discovered is again maintenance of the system.
→ This model is widely used model, although it has many drawbacks.

* Drawbacks of Water fall Model :- There are some problems that are encountered if we apply the water fall model & those are

1) It is difficult to follow the sequential flow in s/w development Process. If some changes are made at some phases then it may cause some confusion.
2) The requirement analysis is done initially, & sometimes it is not possible to state all the requirements explicitly in the beginning. This causes difficulty in the project.
3) The customer can see the working model of the project only at the end. After reviewing of the working model, if the customer gets dissatisfied then it causes serious problems.
4) Linear nature of waterfall model includes blocking states, becoz certain tasks may be dependent on some previous tasks. It may cause long waiting time.

2) Prototyping Model :-

→ In Prototyping model initially the requirement gathering is done
→ Developer & Customer define overall objectives; identify areas needing more requirement gathering.
→ Then a quick design is prepared. This design represents what will be visible to user — in i/p & o/p format.
→ From the quick design a prototype is prepared.
→ Customer or user evaluates the prototype in order to refine the requirements.
→ Iteratively prototype is tuned for satisfying customer requirements
→ Thus prototype is important to identify the s/w requirements.
→ when working prototype is built, developer use existing program fragments (break or a small part broken off) or program generators to throw away the prototype & rebuild the system to high quality

Fig:- Prototyping

→ Certain classes of mathematical algorithms, subset of command driven systems & other appln's where results can be easily examined without real time interaction can be developed using prototyping Paradigm

Drawbacks of Prototyping :-

1. In the 1st version itself, customer often wants "few fixes" rather than rebuilding of the system. Whereas rebuilding of new system maintains high level of quality. (No time)
2. The first version may have some compromises.
3. Sometimes developer may make 'imple' compromises to get prototype working quickly. Later on developer may become comfortable with compromises & forget why they are inappropriate.

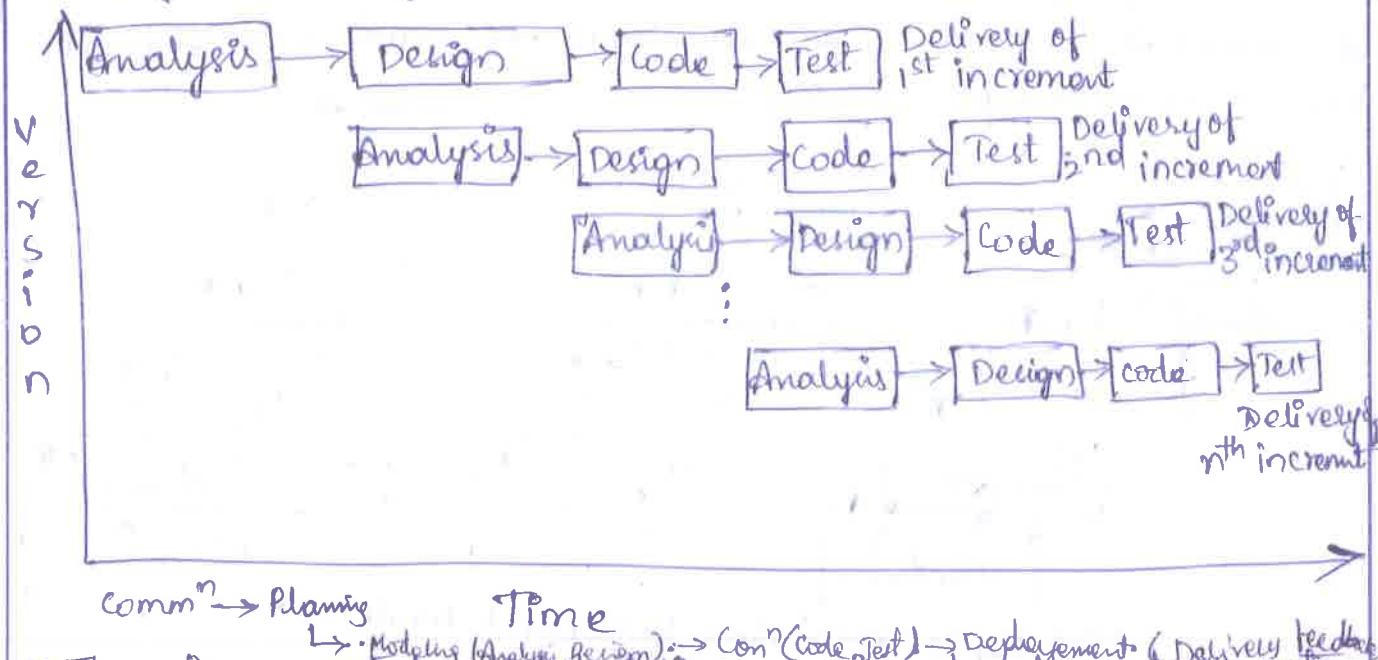
Advantages

→ Customers can see steady progress
→ This is useful when requirements & are changing rapidly

2) Incremental Model:-

→ The Incremental model has same phases that are in waterfall model. But it is iterative (relating) in nature. The incremental model has following phases.

1. Analysis 2. Design 3. Code 4. Test.



→ The Incremental model delivers series of releases to the customer
→ These releases are called increments.
→ More and more functionality is associated with each increment
→ The 1st increment is called core product.
→ In this release the basic requirements are implemented & then in subsequent increments new requirements are added.
→ The word processing s/w package can be considered as an example of incremental model.
→ In the 1st increment only the document processing facilities are available.
→ In the 2nd increment, more sophisticated document producing & processing facilities, file management functionalities are given
→ In the next increment spelling & grammer checking facilities can be given. Thus in incremental model progressive functionalities are obtained with each release.

Merits of Incremental Model:-

1. The Incremental model can be adopted when there are less number of people in the project
2. Technical risks can be managed with each increment
3. For a very small time span, at least core product can be delivered to the customer

Rapid Appln development (RAD) Model :-

→ The rapid appln development model is type of incremental s/w process model in which there is extremely short development cycle.
→ This model is similar to waterfall model which achieves the high speed development using component based construction
→ To develop the fully functional system within short time period using this model it is necessary to understand the requirements fully & to have a restricted project scope.

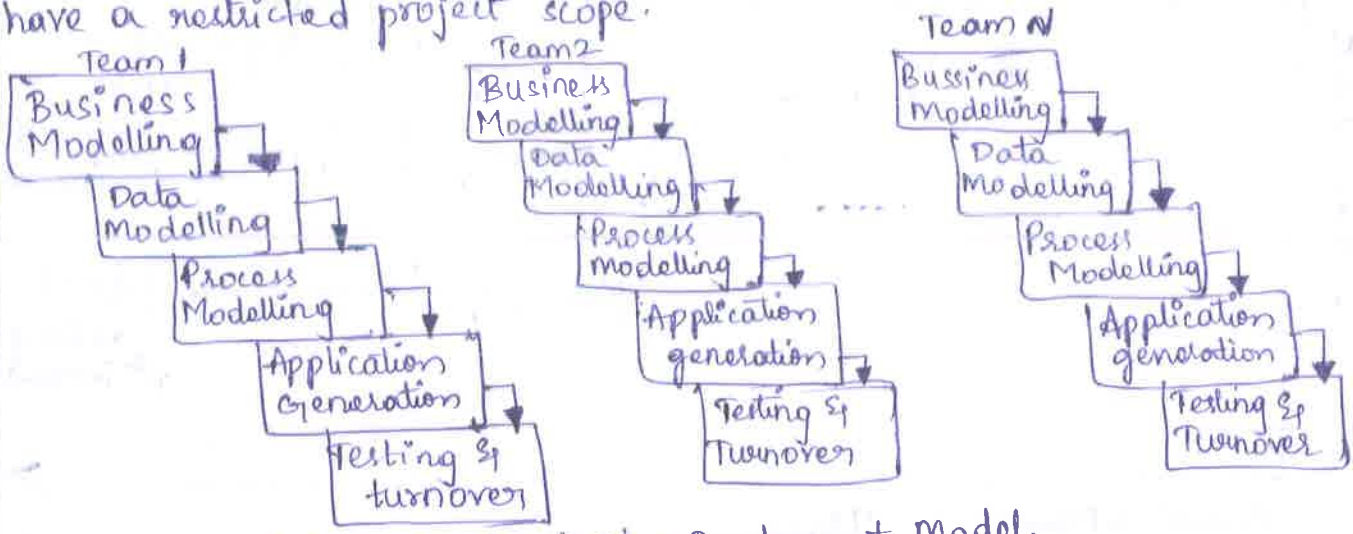
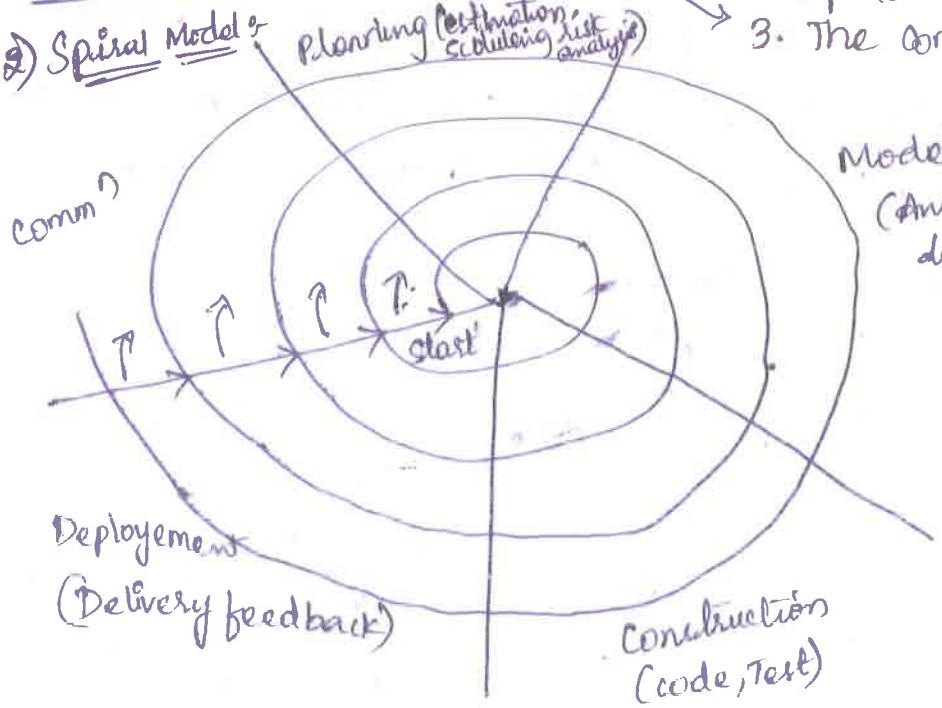


Fig:- Rapid Application Development Model.

→ Various phases of RAD model are
1. Business modeling :- In Business modeling, the infor^n flow is modeled into various business fun's. These business fun's collect following infor^n
→ Infor^n that drives the business process.
→ The type of infor^n being generated
→ The generator of information
→ The information flow.
→ The processor of information.
2) Data Modeling :- In this phase the infor^n obtained in business model is classified into data objects. The characteristics of data objects (Attributes) are identified. The relationship among various data objects is defined.
3) Process modeling :- In this phase the data objects are transformed into processes. These processes are to extract the infor^n from data objects & are responsible for implementing business fun's.
4) Appln generation :- For creating s/w various automation tools can be used. RAD also makes use of reusable components or creates reusable components to have rapid development of s/w.
5) Testing & Turnover :- As RAD uses reusable components the testing efforts are reduced. But if new components are added in s/w development process then such components needs to be tested. It is equally important to test all the interfaces.

3) Evolutionary process Model :- → 1. Prototyping
→ 2. Spiral Model
→ 3. The Concurrent Development model

2) Spiral Model :-



Advantages
1) Requirement changes can be made at every stage.
2) Risks can be identified & rectified before they get problematic

→ The spiral model, originally proposed by Barry Boehm, is an evolutionary s/w process model that couples the iterative nature of prototyping with the controlled & systematic aspects of waterfall model.
→ It provides the potential for rapid development of increasingly more complete versions of the s/w.
→ Boehm describes the model in the following manner.
→ The spiral development is a risk-driven process model generator that is used to guide multi-stake holder concurrent engineering of s/w intensive systems.
→ It has two main distinguishing features.
→ One is a cyclic approach for incrementally growing a system's degree of definition & implementation while decreasing its degree of risk.
→ The other is a set of anchor point milestones for ensuring stakeholder commitment to feasible & mutually satisfory system sol'n.
→ Using the spiral model, s/w is developed in a series of evolutionary releases.
→ During early iterations, the release might be a paper model or prototype.
→ A spiral model is divided into a set of framework activities defined by the s/w engineering team.

3) The Concurrent Development Model :-

The concurrent development model, sometimes called concurrent engineering, can be represented schematically as a series of framework activities, SE actions & tasks & their associated states.

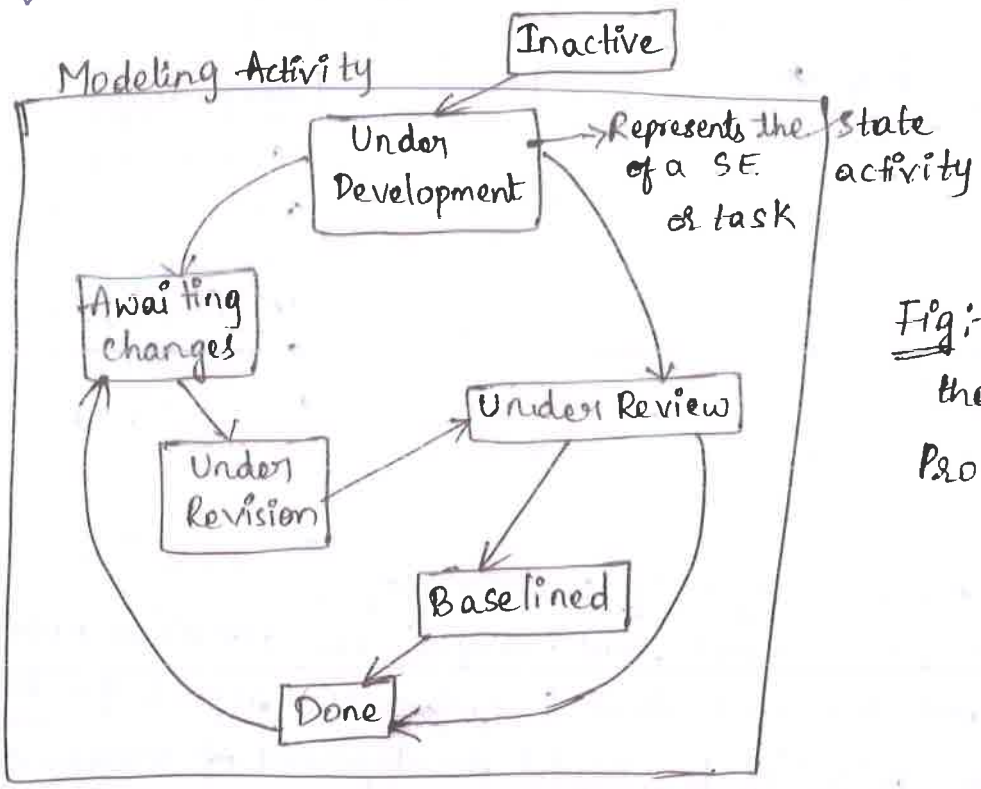


Fig:- One element of the concurrent Process model.

→ The figure a schematic representation of one SE activity within the modeling activity using a concurrent model Approach.
→ The activity- modeling -may be in any one of the states noted at any given time.
→ Similarly other activities, actions or tasks (eg:- comm^n, construction) can be represented in an analogous manner.
→ All S/WE activities exist concurrently but reside in different states.
→ For example, early in a project the comm^n activity has completed its 1st iteration & exists in the awaiting changes state.
→ The modeling activity (which existed in the inactive state while initial comm^n was completed, now make a transition into the under development state.
→ If however, the customer indicates that changes in requirement must be made, the modeling activity moves from the under development state into the awaiting changes state

→ Concurrent modeling defines a series of events that will trigger transitions from state to state for each of the SE activities, actions or tasks.
→ Concurrent modeling is applicable for all types of S/W development & provides an accurate picture of the current state of a Project.

(4) Specialized Process models:-

→ Specialized Process models take on many of the characteristics of one or more of the traditional models. Projects or Process should contain
→ Approaches:-
1) Component-Based Development
2) The formal Methods Model
3) Aspect-oriented S/W Development

1) Component-Based Development:- (Process to apply when reuse is a development objective)
The technology that is used to create the components, the component-based development model incorporates the following steps
→ Available component-based products are researched & evaluated for the appln domain in question
→ Component integration issues are considered.
→ A S/W architecture is designed to accommodate the components.
→ Components are integrated into the architecture
→ Comprehensive testing is conducted to ensure proper functionality.
→ The component-based development model leads to S/W reuse, & reusability provides S/W engineers with a num. of measurable benefits.

2) The formal Methods model:- ~~(emphasizes the~~
→ The formal methods model encompasses a set of activities that leads to formal mathematical specification of computer S/W.
→ Formal methods enable you to specify, develop & verify a computer based system by applying a rigorous, mathematical notation

3) Aspect-oriented s/w Development :-

→ Regardless of the s/w process that is chosen, the builders of complex s/w invariably implement a set of localized features, functions and information content.

→ These localized s/w characteristics are modeled as components & then constructed within the context of a system Architecture.

→ Defining, specifying, designing & constructing aspect.

(2) The Unified Process :- (UML)

→ The unified process is a framework for object oriented model.

→ This model is a called as Rational unified Process model (RUP).

→ It is proposed by Ivar Jacobson, Grady Booch & James Rumbaugh

→ This model is iterative & Incremental by nature.

→ Let us discuss various phases of unified Process.

→ There are 5 phases of unified process model & those are

1) Inception 2) Elaboration 3) Construction 4) Transition 5) Production

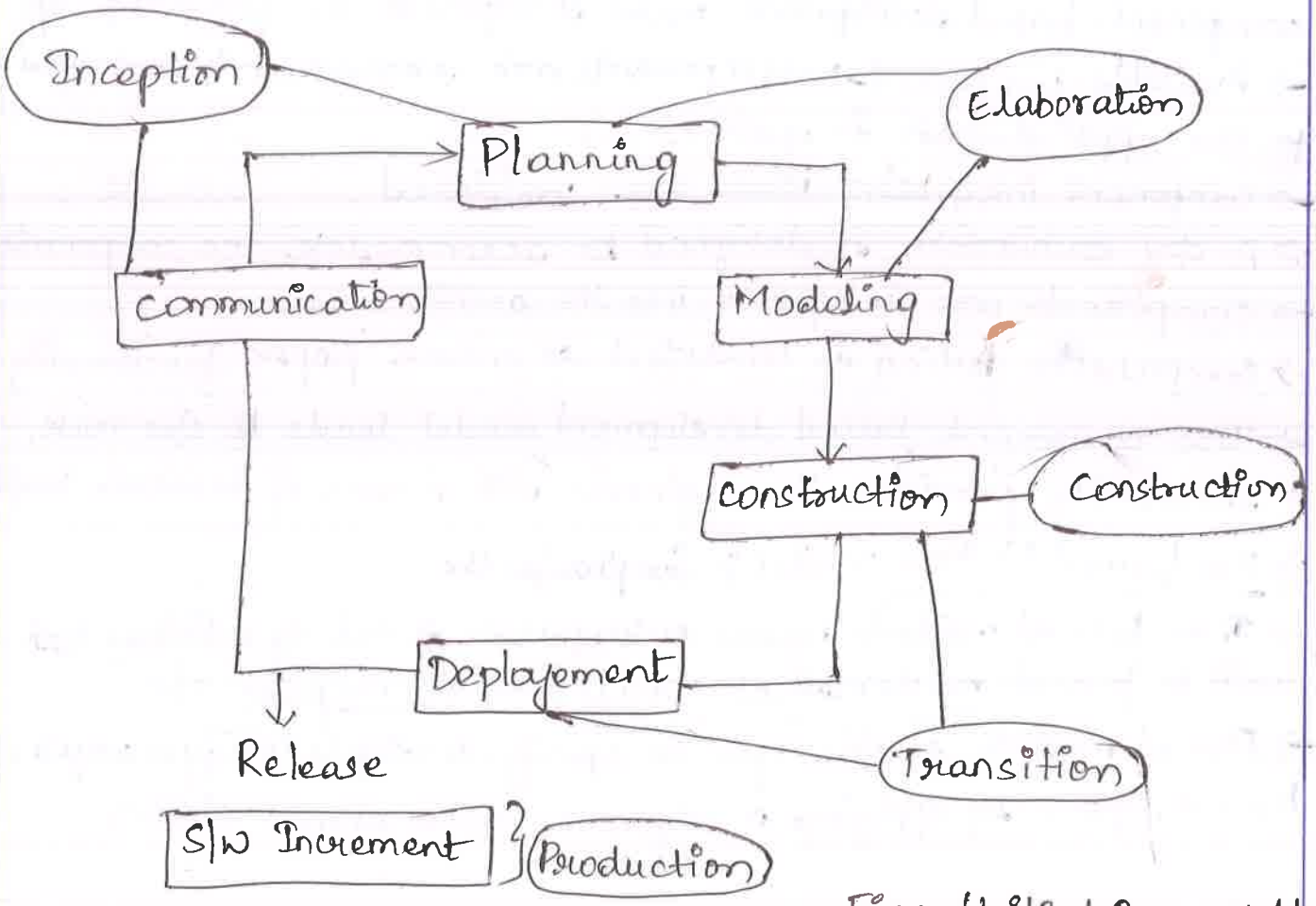


Fig :- Unified Process Model

1. Inception :- In this phase there are 2 major activities that are conducted ; communication and planning.
→ By having customer comm° business requirements can be identified
→ Then a rough architecture of the system is Proposed
→ Using this rough architecture it then becomes easy to make a plan for the project. Use cases are created which elaborates the user scenario.

2. Elaboration :- Elaborsation can be done using 2 activites: planning & modelling.
→ In this phase the use cases are redefined.
→ And an architectural representation is created using 5 models such as use-case model, analysis model, design model, Implementation model and deployement model.
→ Thus executable baseline gets created.

3. Construction :- The main activity in this phase is to make the use cases operational.
→ The analysis and design activities that are started in elaboration phase are completed in this phase & a source code is developed which implements all desired functionalities.

4. Transition :- In the transition phase all the activities that are required at the time of deployment of the s/w product are carried out.
→ Beta testing is conducted when s/w is delivered to the end user.
→ User feedback report is used to remove defects from the created system
→ Finally s/w team prepares user manuals, installation guide & trouble shooting procedures.
→ This make the s/w more usable at the time of release

* Production :- This is the final phase of this model.
→ In this phase mainly the maintenance activities are conducted in order to support the user in operational environment.

Various work Products that may get generated in every phase are as given Below

Inception phase	Elaboration Phase	Construction Phase	Transition Phase
• Initial Use Case model • Initial risk Assessment • Project Plan.	• Use Case model • Requirements Analysis model • Architecture model • Preliminary design model • Risk list • Iterative Project Plan • Preliminary user Manual	• Design model • S/W components • Test Plan • Test Cases • User manual • Installation manual	• Delivered s/w Increments. • Beta test Report • User feedback report

* Differences b/w S/w life cycle & a Process model

S/w life cycle model	Process model
→ This model is based on common activities; Analysis, design, code & testing	→ This model is based on problem definition, technical development, S/w integration & existing status.
→ The S/w Development process can be clearly & systematically defined in phases.	→ The S/w development process cannot be cleaned defined in phases
→ Customer interaction is possible in every stage of S/w development proces in this model	→ Customer interaction is only at initial stage of S/w development
→ Large scale projects can be handled using this approach	→ Large scale projects may be caught in chaotic situation using this approach

UNIT :-2

Software Requirements : Functional and non-functional Requirements, User Requirements, System requirements, Interface specification, the software requirements document.

Requirements engineering Process ; Feasibility studies, Requirements elicitation and analysis, Requirements validation, Requirements management.

System Models :- Context models, Behavioral models, Data models, Object models, Structured methods.

* Software Requirement :-

The software engineer and customer take an active role in requirement engineering

* Requirement engineering :- It is the process of establishing the services that the customer requires from a system and the constraints under which it operates & is developed

* Requirement ; A requirement can range from a high-level abstract statement of a service or of a system constraint to a detailed mathematical functional specification.

→ The requirement must be open to interpretation & it must be defined in detail.

* Types of Requirements ; the requirements can be classified as

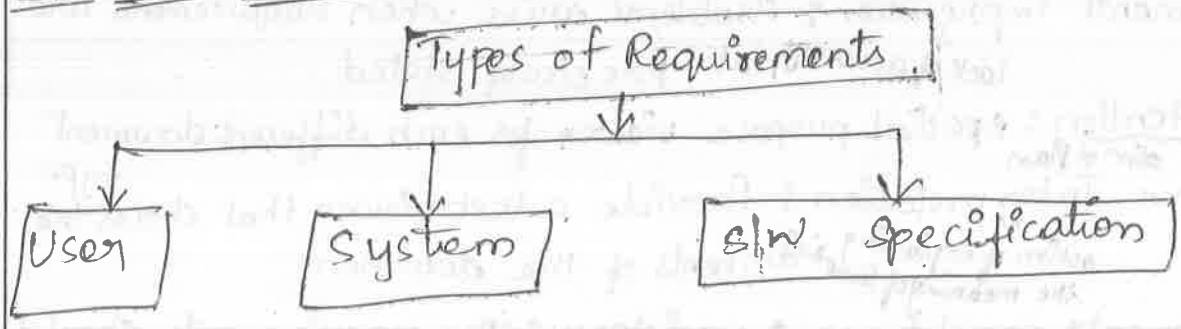


Fig :- Types of requirements

1) User requirements :

It is a collection of statements in natural language plus description of the service the system provides & its operational constraints. It is written for customers.

2) System requirements :- It is a structural document that gives the detailed description of the system services. It is written as a contract b/w client & contractor.

3) S/W specification :- It is a detailed s/w description that can serve as a basis for design implemen^n. Typically it is written for s/w developers.

* Functional and Non-Functional Requirements :

→ S/W system requirements can be classified as functional & non-functional requirements.

1) Functional Requirements :

→ Functional requirements should describe all the required functionality system services.

→ The customer should provide statement of service.

→ Functional requirements are heavily dependent upon the type of s/w expected users & the type of system where the s/w is used.

→ Functional Example :- Consider a library system in which there is a single interface provided to multiple databases.

→ These databases are collection of articles from different libraries.

→ A user can search for, download & print these articles for a personal study.

* Problems associated with Requirements :-

1) Requirements Imprecision :- Problems arise when requirements are not precisely stated. (lack of accuracy)

2) User Intention :- Special purpose viewer for each different document type. (aim or plan)

3) Developer Interpretation :- Provide a text viewer that shows the contents of the document. (action of explaining the meaning of something)

4) Requirements completeness & consistency :- The requirements should be both complete & consistent. Complete means they should include descriptions of all facilities required. Consistent means there should be no conflicts or contradictions in the descriptions of the system facilities. (combining of statements ideas)

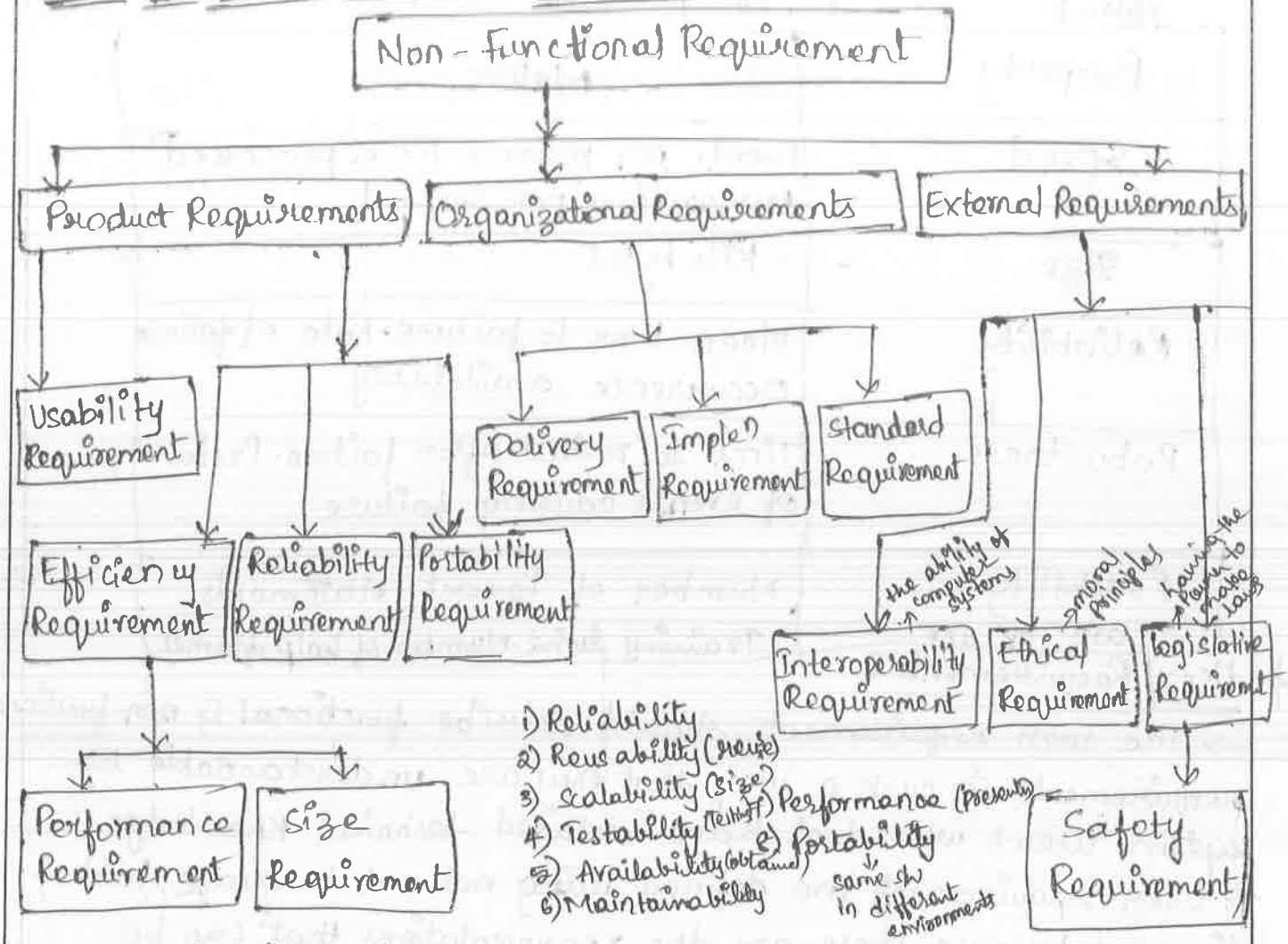
2) Non-Functional Requirements :-

→ The non-functional requirements define system Properties & constraints.

→ Various properties of a system can be: Reliability, response time, storage requirements. And constraints of the system can be: i/p & o/p device capability, system representations etc.,

→ Non-functional requirements are more critical than functional requirements. If the non-functional requirements do not meet then the complete system is of no use.

Types of Non-functional Requirements :-



* Product Requirements: These requirements specify how a delivered should behave in a particular way. For Instance: execution speed, reliability. (performing consistency web)

* Organizational Requirements: The requirements which are consequences of organizational policies & procedures come under this category. For Instance: Process standards used implementation Requirements.

* External Requirements :- These requirements arise due to the factors that are external to the system & its development process. For instance interoperability requirements, legislative requirements.
→ In short, non-functional requirements arise through (Problem)
(i) User needs.
(ii) because of budget constraints.
(iii) Organizational policies.
(iv) the need for interoperability with other s/w or h/w systems
(v) because of external factors such as safety regulations.
→ Metrics (System) used for specifying the non-functional requirements

Property	Metric
Speed	Events per response time. processed transactions per second
Size	Kilo bytes
Reliability	Mean time to failure. Rate of failure occurrence availability
Robustness	Time to restart after failure. Probability of events causing failure
Portability	Number of target statements
Ease of use	Training time, Number of help frames

* User Requirements :-
→ The user requirements should describe functional & non-functional requirements in such a way that they are understandable by system users who dont have detailed technical knowledge.
→ User requirements are defined using natural language, tables & diagrams because these are the representations that can be understood by all users.
→ various problems that can arise in the requirement specification when requirements are given in natural language
Eg: Consider a spell checking & correcting system of a word Processor. The system should poses a traditional word dictionary & user supplied dictionary.

The system should propose the following options to user:
1) Ignore the corresponding instance of the word & go to next sentence
2) Ignore all instances of the word
3) Replace the word with a suggested word from the dictionary
4) Edit the word with user-supplied text
5) Ignore this instance & add the word to a specified dictionary

→ Problems in natural language
1) lack of clarity
2) Requirements Confusion → multiple / single
3) Requirements Amalgamation

* System Requirements:-
→ System requirements are more detailed specifications of system functions, services & constraints than user requirements.
→ They are intended to be a basis for designing of the system.
→ They may be incorporated into the system contract
→ The system requirements can be expressed using system models
→ The requirements specify what the system does & design specifies how it does.
→ System requirements should simply describe the external behavior of the system & its operational constraints. They should not be concerned with how the system should be designed & implemented.
→ For a complex s/w system design it is necessary to give all the requirements in detail
→ Usually, natural language is used to write system requirements specification & user requirements.

* Interface Specification:- (Interface → 2 or more separate components of a computer system exchange info)
→ Sometimes there is already existing system which can be used with the newly created s/w system.
→ This conjunction of old system with new system is called system interface.
→ In such situation the interfaces of already existing system must be specified clearly.
→ There are 3 types of interfaces that can be defined
1) Procedural Interfaces.
2) Data Structures 3) Representation of data

1) Procedural Interfaces :- These are popularly known as Appl'n Programming Interfaces (API). Such procedures are intended to offer services that may be used by calling procedures.

2) Data structures :- Data structures are the descriptors of data they play an important role in organization of data for given algorithm. The data structures can be passed from one sub-system to another.

3) Representation of data :- This level of specification is used in certain programming languages like ADA. For real time appl'ns these kind of interfaces are often useful. sometimes to describe this interface diagrams can be used.

Eg :-
```
Interface Hello {
    void initialize(String s);
    string SayHello();
}
```
This Interface will display the msg hello on encountering the Interface

* The S/W Requirements Document :-

The SRD is the specification of the system. It should include both a definition & a specification of requirements. It is not a design document. As far as possible, it should set of what the system should do rather than how it should do it.

* SRS (S/W Requirements Specification) :-

The S/W requirements provide a basis for creating the SRS

→ The SRS is useful in estimating cost, planning team activities, performing tasks and tracking the team's progress throughout the development activity.

→ Typically S/W designers use IEEE STD 830-1998 as the basis for the entire S/W specifications. The standard template for writing SRS is as given below.

Document Title
Author(s)
Affiliation
Address
Date
Document Version.

1. Introduction :-

1.1 Purpose of this document :- Describes the purpose of the document

1.2 Scope of this document : Describes the scope of this requirements This section also details any constraints that were placed upon the requirements elicitation process such as schedules, cost.

1.3 Overview

Provides a brief overview of the product defined as a result of the requirements elicitation (Infor^n) process.

2. General Description :

→ Describes the general functionality of the product
→ " " feature of the user community.

3. Functional Requirements :- This section lists the functional requirements in ranked order. Each functional requirement should be specified in following manner

• short, imperative sentence stating highest ranked functional Requirement.

1. Description :- A full description of the requirement
2. criticality : Describes how essential this requirement is to all overall system
3. Technical issues : Describes any design or implementation issues involved in satisfying this requirement.
4. Cost & Schedule : Describes the relative or absolute costs of the system
5. Risks : Describes the circumstances under which this requirement might not able to be ~~sta~~ satisfied.
6. Dependencies with other requirements : Describes interactions with other requirements
7. ... any other appropriate

4. Interface Requirements: This section describes how the s/w interfaces with other s/w products or users for i/p or o/p.

4.1 User Interfaces
Describes how this product interfaces with the user

4.1.1 GUI → Graphical user Interface

4.1.2 CLI → command-line Interface.

4.1.3 API → Appl^n programming Interface.

4.2 H/W Interfaces: Describes interfaces to h/w devices

4.3 Comm^n Interfaces: Describes n/w Interfaces

4.4 S/W Interfaces: Describes any remaining s/w Interfaces

5. Performance Requirements: Specifies speed & memory requirements

6. Design constraints: Specifies any constraints for the design team such as s/w or H/W limitations

7. Other non-functional attributes
Specifies any other particular non functional attributes required by the system. such as:

7.1 Security
7.2 Binary Compatibility
7.3 Reliability
7.4 Maintainability
7.5 Portability
7.6 Extensibility
7.7 Reusability
7.8 Appl^n Compatibility
7.9 Resource Utilization
7.10 Serviceability ... others as appropriate

8. Operational Scenarios: Describe a set of scenarios.

9. Preliminary schedule: Provides an initial version of the project plan

10. Preliminary Budget: This section provides an initial budget for the project

11. Appendices: 11.1 Definitions, Acronyms, Abbreviations
11.2 References.

* Characteristics of SRS :-
- correct :- SRS should be made up to date
- Unambiguous - requirements are correctly understood.
- Complete - To make SRS complete.
- Consistent - functionalities identified
- Specific - requirements should be mentioned specifically.
- Traceable - what is the need for mentioned requirement.

Example of SRS :- Attendance Maintenance system

Prepared by XYZ
December 1, 2007
Release 1.0
Version 1.0

Table of Contents

7.3 Availability
7.4 Maintenability
7.5 Reusability
8. Operational Scenarios
9. Preliminary Schedule.

1. Introduction

1.1 Purpose

This document gives detailed functional & non-functional requirements for attendance maintenance system

1.2 scope

This system allows the Teacher to maintain attendance record of the classes to which it is teaching. with the help of this system teacher should be in a position to send e-mail to the students who remain absent for the class. The system provides a cumulative report at every month end for the corresponding class

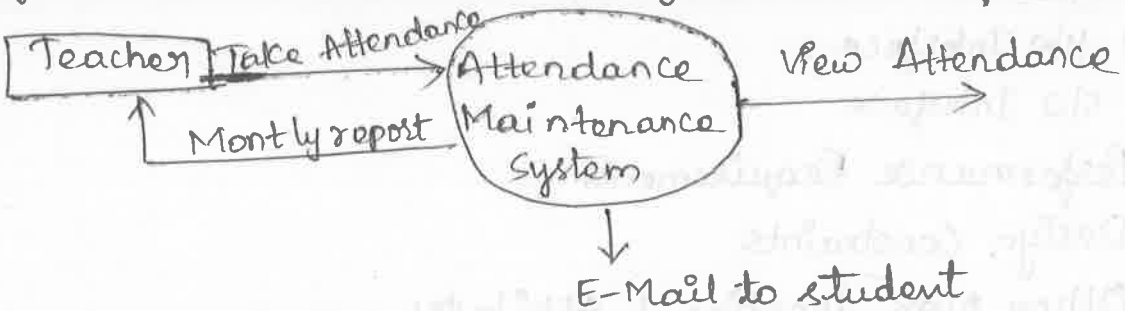
1.3 Overview: This system provides an easy sol^n to the teacher to keep track of student attendance & statistics

2. General Description :-

This AMS replaces the traditional, manual attendance system by which a lot of paper work will be reduced.

→ The teacher should able to view photograph of a student along with his attendance in his laptop.

→ The system should produce monthly attendance report



→ Every Teacher should have Laptop with Wireless Internet connection.

2.1 User Manual :- The system should provide help option in which how to operate the system should be explained & also hardcopy

3) Functional Requirements

Description:

The identity of student is verified & then marked present at particular date & Time. The system should display student photograph along with their names for that corresponding class. A statistical report should display individuals report or cumulative report whenever required.

3.2 Technical Issues: The system should be implemented in VC++

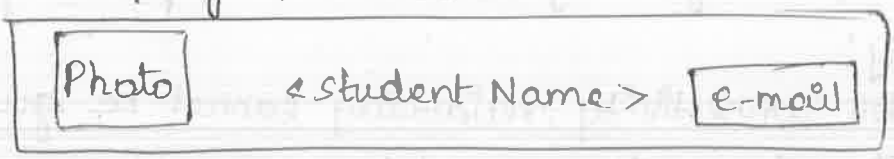
4. Interface Requirements :-

4.1 GUI

GUI 1: Main menu should provide options such as File, Edit, Report, help

GUI 2: In File menu one can create a new record file or can open an existing record file.

GUI 3: The display of record should be



The Photo can be clicked to mark student present for particular class. The e-mail button can be clicked to send e-mail to the student being absent.

GUI 4: Report option should display statical report. It may be for particular student or for the whole class.

GUI 5: Help option should describe functionality of the system It should be written in simple html.

4.2 H/w Interface

H/w Interface 1: The system should be embedded in laptops

H/w Interface 2: The laptop should use wireless Ethernet card to send e-mails.

4.3 S/W Interface:

S/W Interface 1: AMS

S/W " 2: The attendance DB should be transmitted to departmental DB server

S/W " 3: E-Mail msg generator generates absence msg

S/W " 4: Report generators

5. Performance Requirements:

This system should work concurrently on multiple processors b/w the college hours. The system should support 50 users.

→ The email should be sent within 1 hr after the class gets over

→ The system should support taking attendance of max 100 students per class.

6. Design Constraints:

The system should be designed within 6 Months

7. Other Non-Functional Attributes

7.1 Security :- The teacher should provide pwd to log on to system He/she should be able to see the record of his/her class The pwd can be changed by the teacher by providing existing pwd.

7.2 Reliability:

Due to wireless connectivity, reliability cannot be guaranteed.

7.3 Availability: The system should be available during college hrs.

7.4 Maintenability: There should be a facility to add or delete or update Teachers & Students for each semester.

7.5 Reusability: The same system will be used in each new sem

8. Operational Scenarios: There will be student DB, Teacher DB. The student DB will contain student name, class, attendance, e-mail address, address, phone number.

→ The teacher DB will contain Teachers name, class taught, e-mail address, Phone num

9. Preliminary Schedule

The system has to be implemented within 6 Months.

Q. Who are using SRS?

A: Customers or end-users, Project Managers, System Developers, Test engineers, System maintenance engineers.

* Requirements (RE) Engineering Process :-

→ The requirement engineering processes are the processes used to discover, analyse and validate the system requirements

→ Processes used for Requirement Engineering vary widely.

→ The processes are dependent upon the appl^n domain, people involved and the organization developing it.

→ Goal of RE process is to create & maintain system requirement specification document.

→ Before beginning these processes it is necessary to check whether the system that will get generated is useful for the business or not

→ This kind of study is called feasibility study.

→ Once the system is feasible then only the RE process begins.

→ Various Activities that are conducted under RE process are discovering requirements, converting these requirements into some other form and then checking whether the requirements are as per the customer need or not.

* Requirement Engineering Process :-

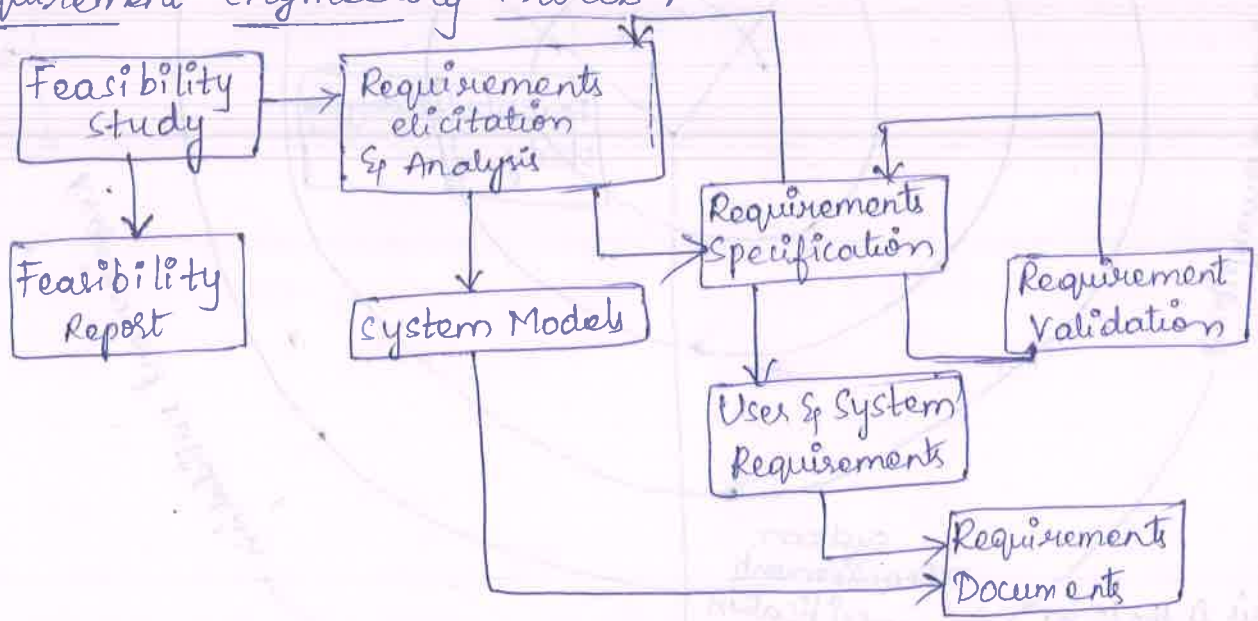
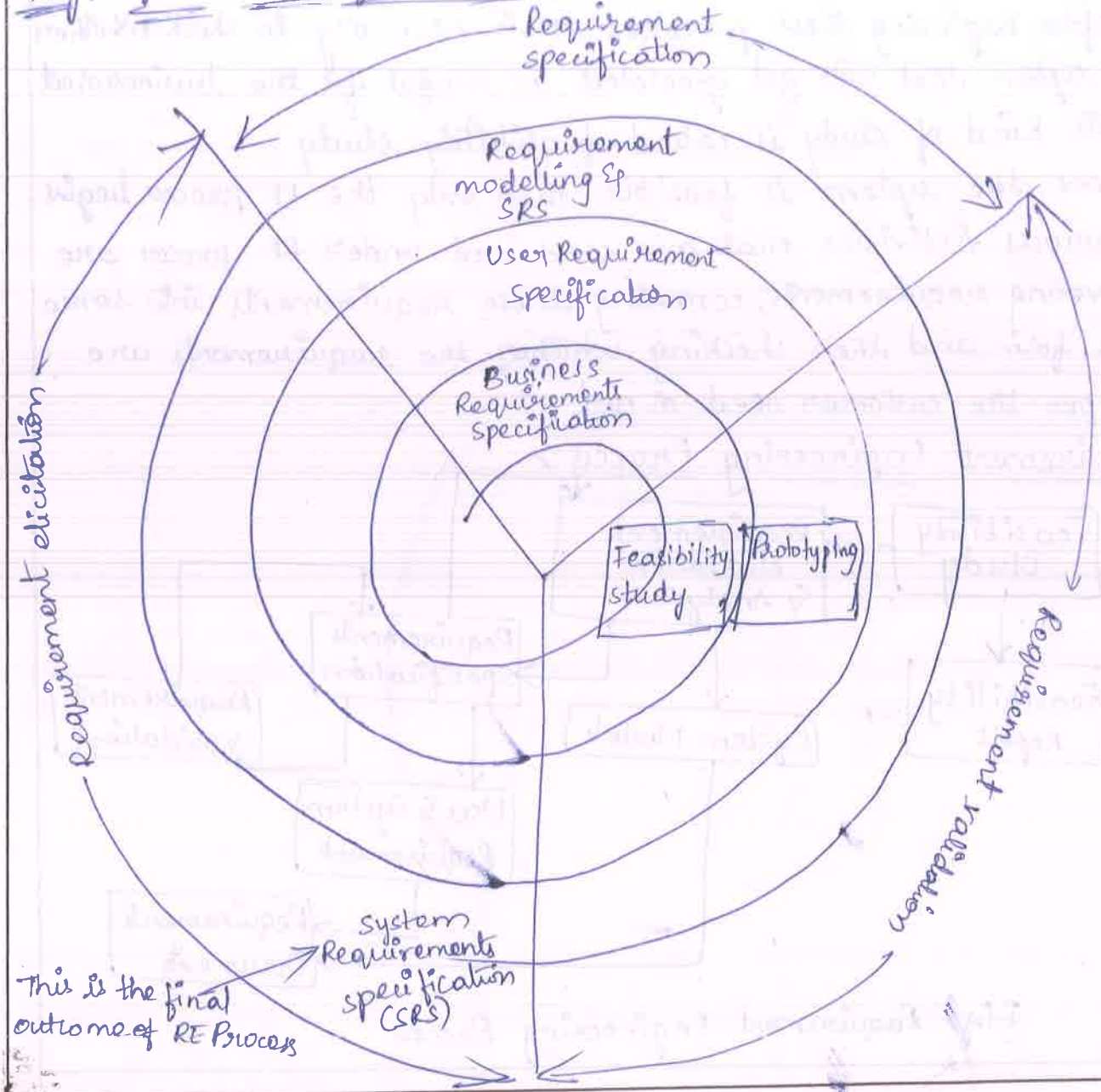


Fig:- Requirement Engineering Process

→ A Requirement Engineering is a Process in which various activities such as discovery, analysis & validation of system requirements are done.
→ It begins with feasibility study of the system and ends up with requirement validation. Finally the requirement document has to be prepared.
→ This process is a three stage activity where the activities are arranged in the iterative manner.
→ In the early stage of this process most of the time is spent on understanding the system by understanding the high-level non functional requirements & user requirements
→ The spiral model of requirement Engineering process is as shown below.

Fig:- Spiral Model of RE Process



→ The generic activities that are common to all processes are
1. Requirements elicitation
2. " Analysis
3. " Validation
4. " Management

→ RE process can also be viewed as structured Analysis method in which the system is analysed fully some system models are prepared.
→ Particularly use cases are developed which help in exposing the functionalities of the systems.
→ Along with creation of system models some additional Infor" is also provided in the RE process

* Feasibility Studies:-

→ It is an initial step for RE where we do process.
→ A feasibility study is a study made to decide whether or not the proposed system is worthwhile.
→ The focus of feasibility study is to check
→ If the system contributes to organizational objectivities.
→ If the system can be engineered using current technology.
→ If the system is within the given budget
→ If the system can be integrated with other useful systems.
→ The implem" of Feasibility study is based on the infor" assessment (what is required), Infor" collection & report writing
→ while performing the Feasibility study, following questionnaires to the people in the organization should be asked.
1. what if the system wasn't implemented?
2. What are current process problems?
3. How will the proposed system help?
4. what will be the integration problems?
5. Is new technology needed? with what skills?
6. what facilities must be supported by the proposed system?

→ Feasibility study should be done with the help of project managers who is going to handle that particular project, s/w engineers who are about to develop that system, technical experts & customers who will be using the system.
→ Typically Feasibility study should be completed within 2 to 3 Weeks
→ Finally the Feasibility study report has to be prepared.

* Requirements ~~elicaton~~ elicitation & Analysis:-

→ After performing feasibility study the requirements elicitation & Analysis can be done.
→ Requirement elicitation means discovery of all possible requirements
→ After identifying all possible requirements the analysis on these requirements can be done.
→ S/w Engineers communicate the end-users or customers in order to find out certain infor^n such as: appl^n domain, expected services from the system, the expected performance level of the system.
→ From this infor^n even constraints of the system can be decided.

1. Stakeholders.
2. Requirement Elicitation and Analysis Process.
3. " Discovery.

1. Stakeholders:-

→ This is a commonly used term in S/w Engineering.
→ The stakeholder means the person(s) who will be affected by the system.
Ex: end-user, system maintenance engineers or s/w engineers can be stakeholders.
→ Following is the list of problems encountered in understanding the requirements of the system

Problems	Description
Unrealistic Expectations	Stakeholders sometimes unable to specify what they want exactly. Sometimes they specify unrealistic demands.
Differences in the Requirements	Different Stakeholders specify different requirements. The requirement engineer has to resolve the conflicts in the requirements with proper comm'n with the stakeholders
Economic & Business Environment	The economic & business environment is dynamic due to which there may be change in the requirements or change in the stakeholders. Both these things may affect the requirement Analysis Process
Political changes	Sometimes political factors affect heavily on the need for the system. Hence there may be change in the requirements or stakeholders which ultimately affect the requirement elicitation & Analysis

2) Requirement Elicitation and Analysis Process :-

→ The spiral model as given below depicts the requirement elicitation & Analysis process.

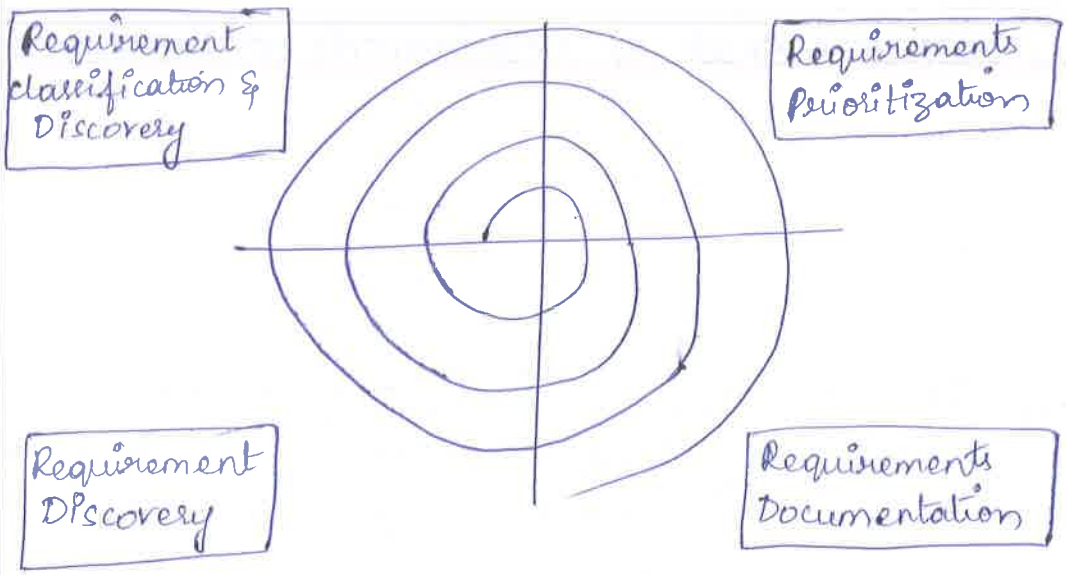


Fig :- Requirements elicitation & Analysis Process.

→ The Process Activities are

1. Requirement discovery :-

By having effective comm^n with the customers the requirements can be identified.

2. Requirements classification & discovery :-

All the unstructured requirements can be categorised systematically depending upon their nature. And they are arranged in groups.

3. Requirement prioritisation :-

There are some conflicting requirements

→ Hence the requirements are prioritised first

→ If there are some unrealistic requirements then negotiations are made & only realistic prioritised requirements are collected

→ If any conflict occurs then it its resolved by requirement engineers

4. Requirement Documentation :-

→ This is the specification of all the requirements. And an important requirement document is created.

3) Requirement Discovery :-

Requirement Discovery means finding all relevant Infor^n about the System.

→ The sources of infor^n for requirement gathering are: docume^n, system stakeholders & specification some other system which is of similar kind.

→ Various methods of requirement discovery are conducting interviews, observations, creating use case scenarios.

→ Let us discuss various methods of requirements discovery.

1. Interviewing.
2. View Point
3. Use Cases
4. Ethnography.

1. Interviewing :-

→ this is an effective method of requirement gathering

→ The Requirement engineering team communicates the stakeholders by asking them various questions about the system & its use

→ There are 2 types of interviews.
1) closed Interview :- Stakeholders answer predefined set of questions
2) Open Interview :- No predefined agenda. Various issues can be raised & discussed with stakeholders.
→ Interviews are useful for understanding stakeholders but they are not much useful for understanding the appln domain.
* Characteristics of effective Interviews :
1) The Interviews should be conducted in a free environment & they should be conducted with open minded approach.
→ The requirement engineers should listen stakeholders with patience.
→ Why if stakeholders are expecting some unrealistic things about the system they should be ready to change their mind & ideas about the system
2) Interviewee should start the discussion by asking questions & the requirements should be gathered together

2) View Point :- View Point Provides the perspective to the system & using these perspectives the requirements can be discovered.
→ The view Point is also useful in classifying the stakeholders.
→ Types of view Points :
1) Interactor ViewPoint :- This viewpoint is useful for finding the interaction one system with other system
eg :- ATM system the user of ATM is the interactor for the bank
2) Indirect View Points :- The user who is not using the system directly but its existence reflects on the requirements of the system
3) Domain ViewPoints :- The domain characteristics & constraints that affect on requirements of the system sets the domain view Points
Ex :- In Library system the rules that are to be followed for reserving the book (the fine or dues should be paid already & book must be returned within 1 week etc)
→ From these viewpoints the requirements can be discovered.
From Interactor viewpoint → system requirements obtained
" Indirect " → organizational requirements & constraints
" Domain " → Domain constructs.

3) Use Cases:-

→ Use Cases are the fundamental units of modelling language in which functionalities are distinctly presented.
→ Use case is a scenario based technique.
→ Use case help to identify individual interactions with the system.
→ Use-cases are extensively used for requirements elicitation.
→ By designing the proper use case for different scenarios major requirements of the system can be identified.
→ The typical notations used in the use cases are

Fig: Use-case notation

The use case for ATM system is as shown below.

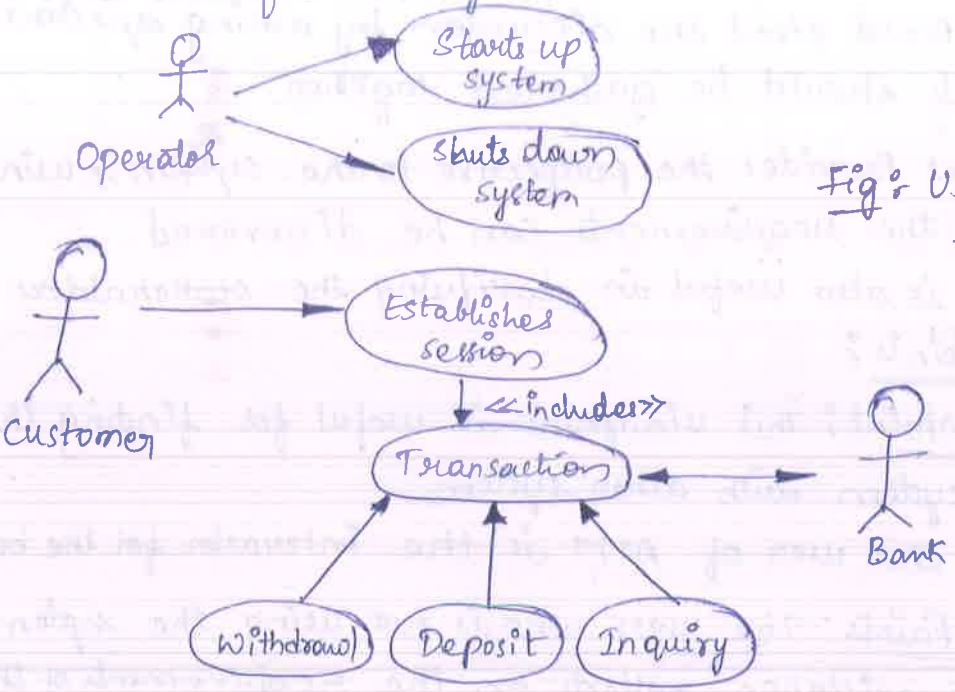


Fig: Use Cases for ATM system.

System start up:- The system is started when the operator turns on the switch. Initially the operator has to enter some amount of money in the cash dispenser. The connection with the bank gets established. Then only the servicing customer can begin.
System shutdown:- The operator can shutdown the system only after confirming that there is no customer currently operating the ATM system. Then the connection to the bank gets disconnected.
Session:- This use case starts when the user inserts the card. ATM card inside & reads it. Then further inquiry info is displayed on display panel.

Transaction:- The transaction use case is comprised of amount withdrawl, deposit & inquiry.

→ The interaction diagram for system startup is as given below

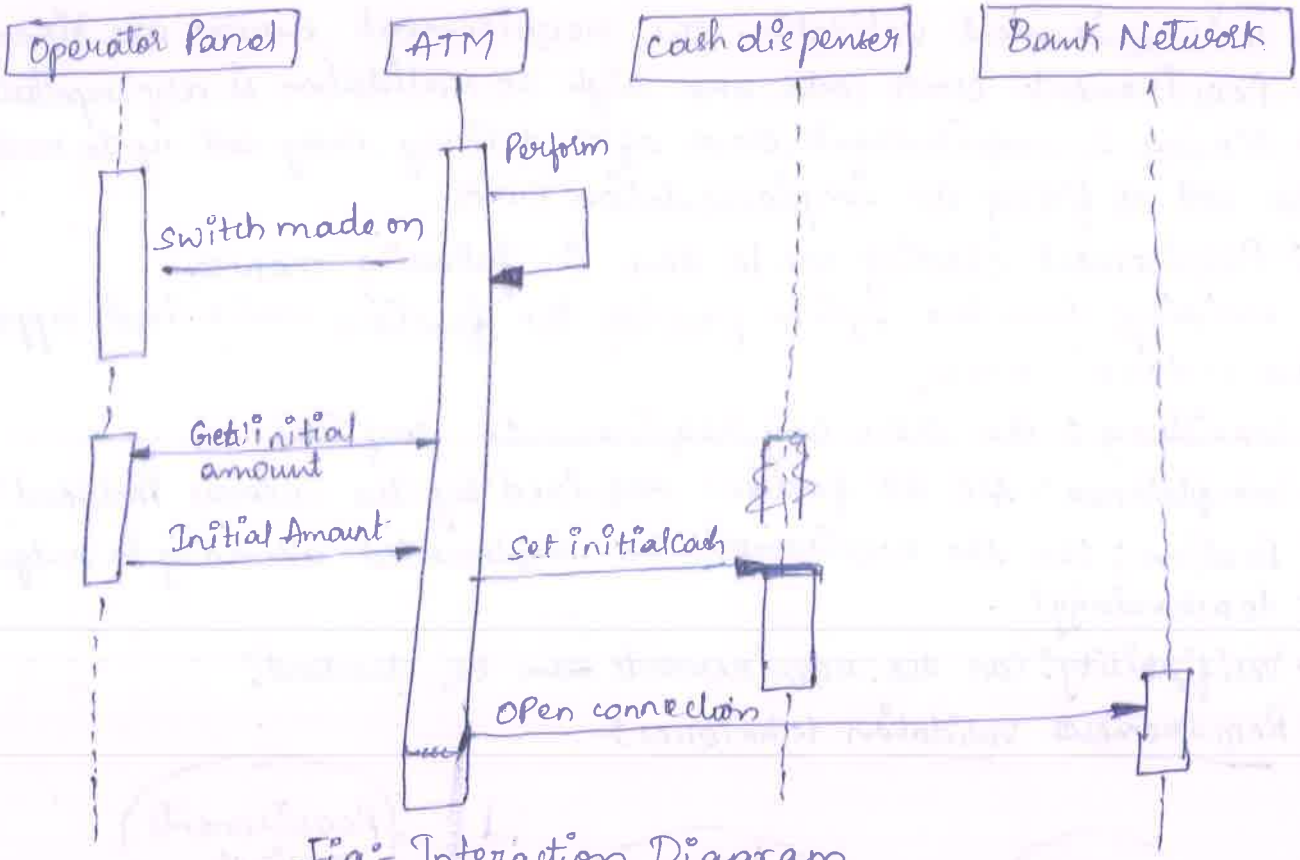


Fig:- Interaction Diagram

→ This diagram depicts the sequence of operations that must be performed in order to accomplish the task.

4. Ethnography:- Ethnography means is a comparative study of people

→ The s/w systems exist for social or organizational purpose.

→ That means the ethnography is a technique of obtaining the social & organizational requirements

→ Ethnography is useful for finding following type of requirements

1. Requirements obtained from working style of People
2. " " " inter-activities performed by the people

* Requirements Validation :-

→ It is a process in which it is checked that whether the gathered requirements represent the same system that customer really want

→ In requirement validation the requirement errors are fixed

→ Requirements error costs are high so validation is very important

→ Fixing a requirements error after delivery may cost up to 100 time the cost of fixing an implementation error.

→ Requirement checking can be done in following manner,

1. Validity :- Does the system provide the functions which best support the customer's needs?
2. Consistency :- Are there any requirements conflicts?
3. completeness :- Are all functions required by the customer included?
4. Realism :- Can the requirements be implemented according to budget & technology?
5. Verifiability :- Can the requirements ~~can~~ be checked?

* Requirements validation techniques :-

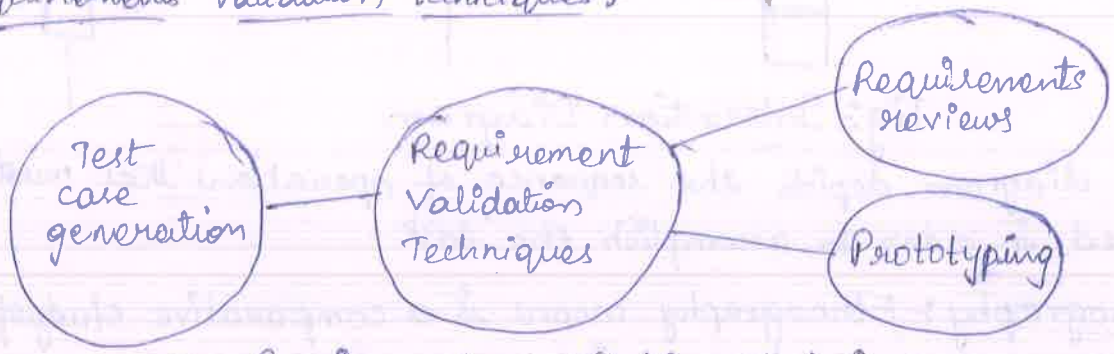


Fig :- Requirement Validation Technique.

1. Requirements reviews :- Requirement review is a systematic manual analysis of the requirements.

→ It should be taken only after formulation of requirement definition and both the customer & contractor staff should be involved in reviews

→ Reviews may be formal or informal (without completed documents)

→ Good comm^n should take place b/w developers, customers & users. such a healthy comm^n helps to resolve problems at an early stage

2. Prototyping :- The requirements can be checked using executable model of a system

3. Test-case generation :- In this technique, the various tests are developed for requirements. The requirement check can be carried out with

1. Verifiability:- Is the requirement realistically testable?
2. Comprehensibility:- Is the requirement properly understood?
3. Traceability:- Is the origin of the requirement clearly stated?
4. Adaptability:- Can the requirement be changed without a large impact on other requirement?

Requirement Management:-

Requirement Management is the process of managing changing requirements during the requirements engineering process & system development

1. Requirements Management Planning
2. " Change Management

1. Requirements Management Planning:-

Following things should be planned during requirement Process

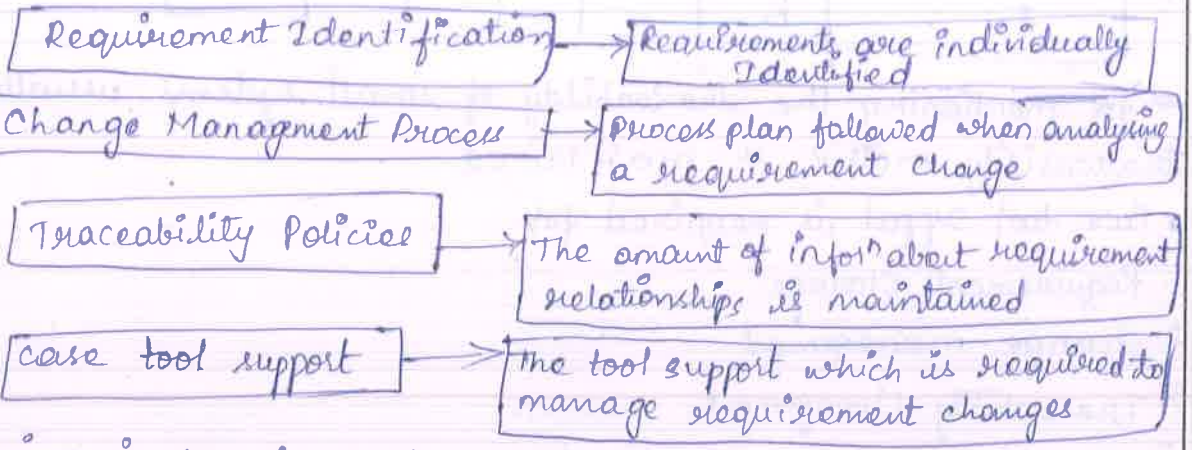


Fig:- Planning in requirement management Process.

→ Traceability is concerned with relationship b/w requirements their sources & the system design. Using traceability the requirement finding become easy.

→ Various types of traceability are.

1. Source traceability:- These are basically the link from requirement to stakeholders who propose these requirements
2. Requirements Traceability:- These are the links b/w dependent requirements
3. Design traceability:- These are the links from requirements to design

→ Traceability info is typically represented by a data structure Traceability matric.

→ If one requirement is dependent upon the other requirement then in that row-column call 'D' is mentioned & if there is weak relationship b/w the requirements then corresponding entry can be denoted by 'R'.

Example:

Requirement ID	A	B	C	D	E	F
A		D			R	
B			D			
C				R		
D			D			R
E						
F	R			D		

→ For mentioning the traceability of small systems usually the traceability matrix is maintained.

- Case tool support is required for

1. Requirement storage
2. change management
3. Traceability Management

2) Requirement Change Management :

→ The requirement change management is a technique that can be applied to the processes in which requirements may get changed

→ The need for requirement change management is that even through the changes are made consistently in the requirements it is possible to incorporate these changes in a controlled manner.

→ The requirement change management process can be applied in 3 stages

1. Problem analysis & change Specification
2. Change Analysis & costing.
3. Change Implementation.

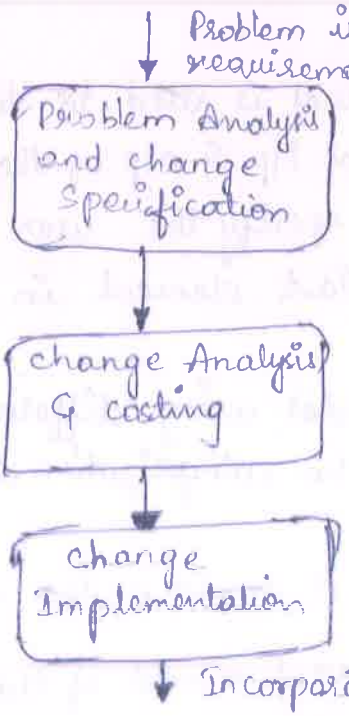


Fig: Requirement change management Process

1. Problem Analysis & change Specification :-
When requirement change request is made for some particular problem then the problem with older requirement is mentioned or cometimes simply change specification is given.
→ Then 1st of all, problem analysis or change specification is analysed in order to validate the required change.
→ If necessary the feedback of this analysis is given to the person who is demanding such change.

2. change analysis & costing :- Following actions are carried out in this stage
i) The effect of change is assessed using traceability infor?
ii) The cost of such change is estimated
iii) After getting the cost of changes the decision is made on whether to go for implementation of these changes or not.

3. change Implementation :- Once it is decided to implement the proposed changes in the requirement, the requirement document has to be modified.
→ The requirement document has to either re-written or re-organised
→ This can be acheeved by making the modularity in the requirement specifications, so that it becomes easy to change individual section without affecting other part of requirement document

* System Models :-

→ It is a graphical representation that is used to describe various processes of the system, the type of i/p & o/p of the system

→ These system models not only specify the user requirements but they also serve as an important element in analysis & design phase

→ We can develop the system model using different perspectives & use of these perspectives gives the categorization of system models into different models

1) Using External Perspective context model of the system can be developed

2) Using Behavioural " behavioural model of the system can be developed.

3) Using structural " data model of the system can be developed.

1) Context Model :-

→ It is a graphical representation of the system in which the system boundaries are specified.

→ This is the model which represents the system environment in which system is working

→ While deciding the environmental factors of the system it is very much necessary to take decisions on certain aspects such as overall cost of the system & time required to analyse such system.

Example :- Consider the Inventory control system for which the context model of the system can be created

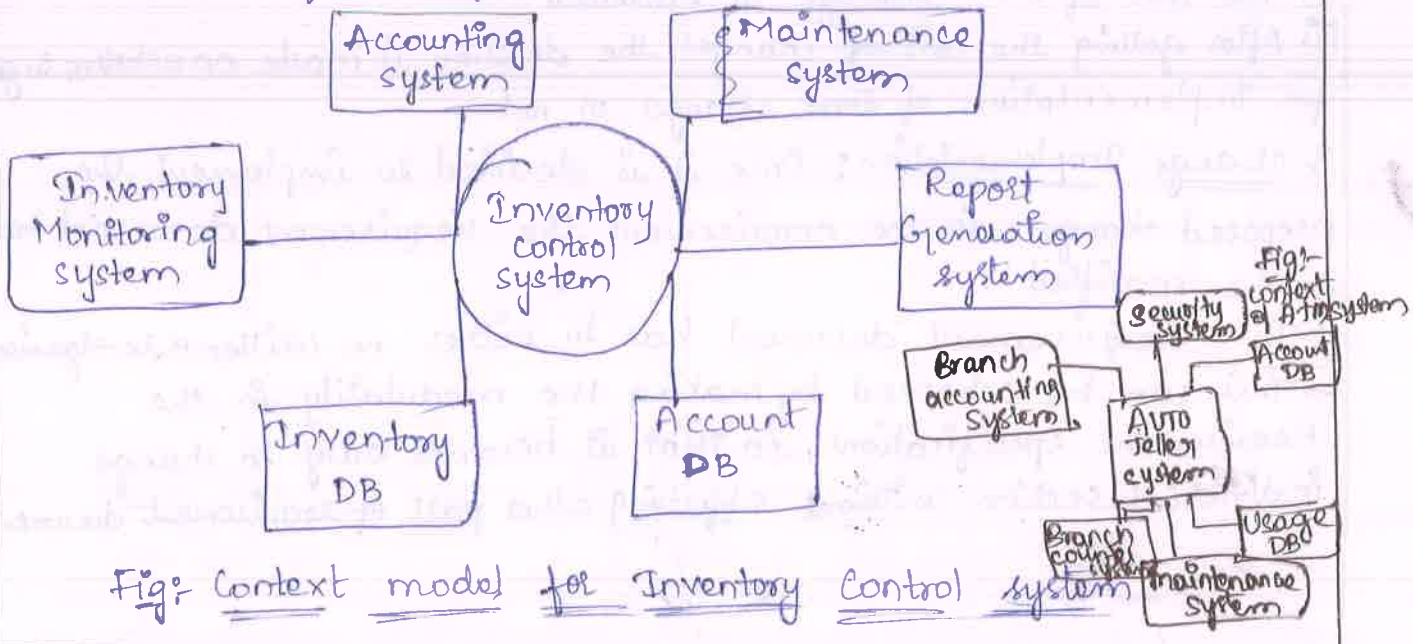


Fig:- Context model for ATM system

Fig:- Context model for Inventory Control system

→ First of all using the requirements of the system boundaries are decided & the dependencies of the system are specified in order to define the system environment

→ In figure the system boundaries are clearly shown & the environmental factors are defined.

→ The Inventory control system is connected to various systems such as Inventory monitoring system, Accounting system, Report generation system & Maintenance system.

→ This is an architectural model in which an abstract representation of Inventory control system is given.

→ Thus context models are the architectural models in which environment of the system is shown.

→ The relationship that exists with other systems is shown but nature of these relationships is not mentioned in the context model.

→ All these defined relationships help in finding the requirements of the system.

→ The process model is again a graphical representation of system

→ The order of execution of various events can be understood with the help of such Process models. Processes

Example :- For the same Inventory control system we can draw a process model.

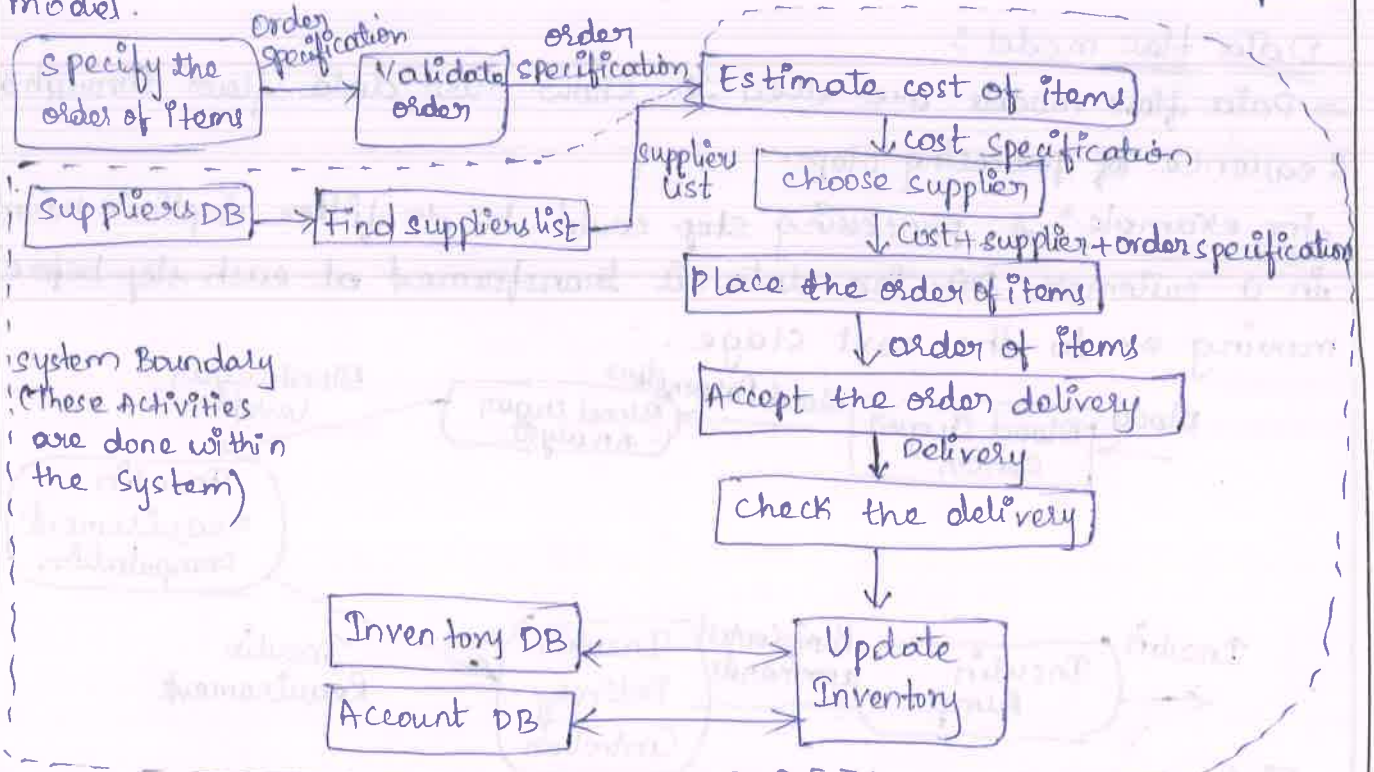


Fig :- Process Model

Behavioural Models :-

→ Behavioural models are used to describe the overall behaviour of a system. There are 2 types of models that depict the behaviour of the system.

1) → Data flow Model
2) → State chart Diagram / Chart machine models.

→ The data flow model represents the flow of data & state chart diagram represent the states that are occuring in the system.

1) Data flow Model/Diagram :-

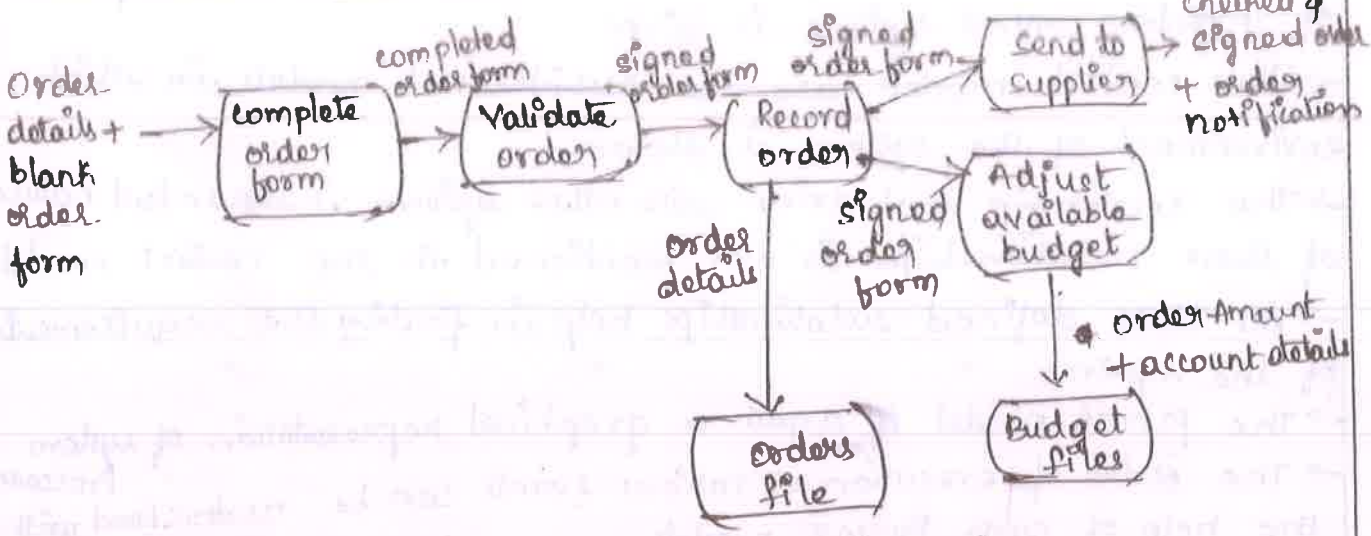


Fig :- Data flow Diagram of order Processing

Data flow model :-

→ Data flow models are used to show how data flows through a sequence of processing steps.

For example :- A processing step could be to filter duplicate record in a customer DB. the data is transformed at each step before moving on to the next stage.

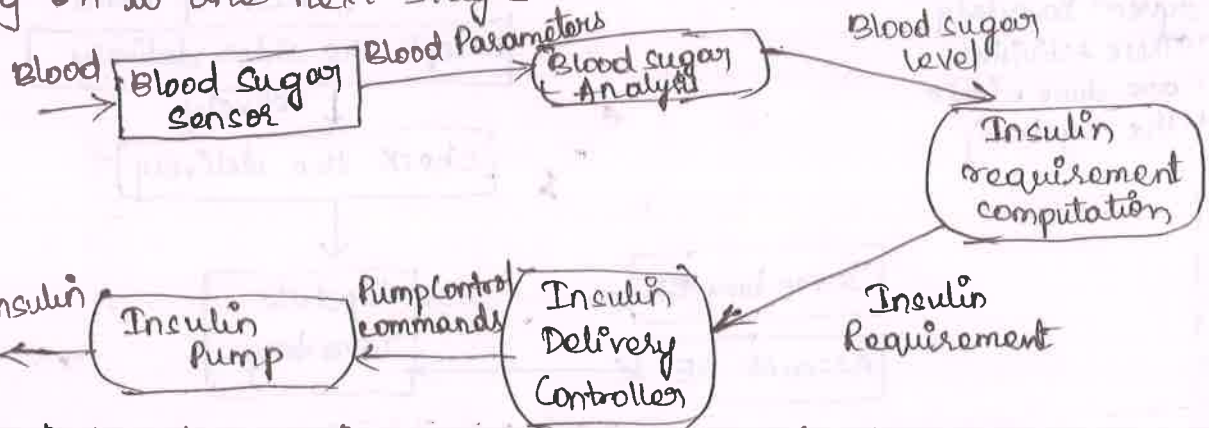


Fig :- Data-flow Diagram of an Insulin Pump

→ Data flow models are valuable because tracking & documenting how the data associated with a particular process moves through the system helps analysts understand what is going on.
→ Data flow diagrams have the advantage that some other modelling notations, they are simple & intuitive. (using or based on what one feels to be true, even without conscious reasoning).

2) State Machine models :-

→ A state machine model describes how a system responds to internal or external events.
→ The state machine model shows system states & events that cause transitions from one state to another.
→ It does not show the flow of data within the system.
→ This type of model is often used for modelling real-time systems because these systems are often driven by stimuli from the system's environment.

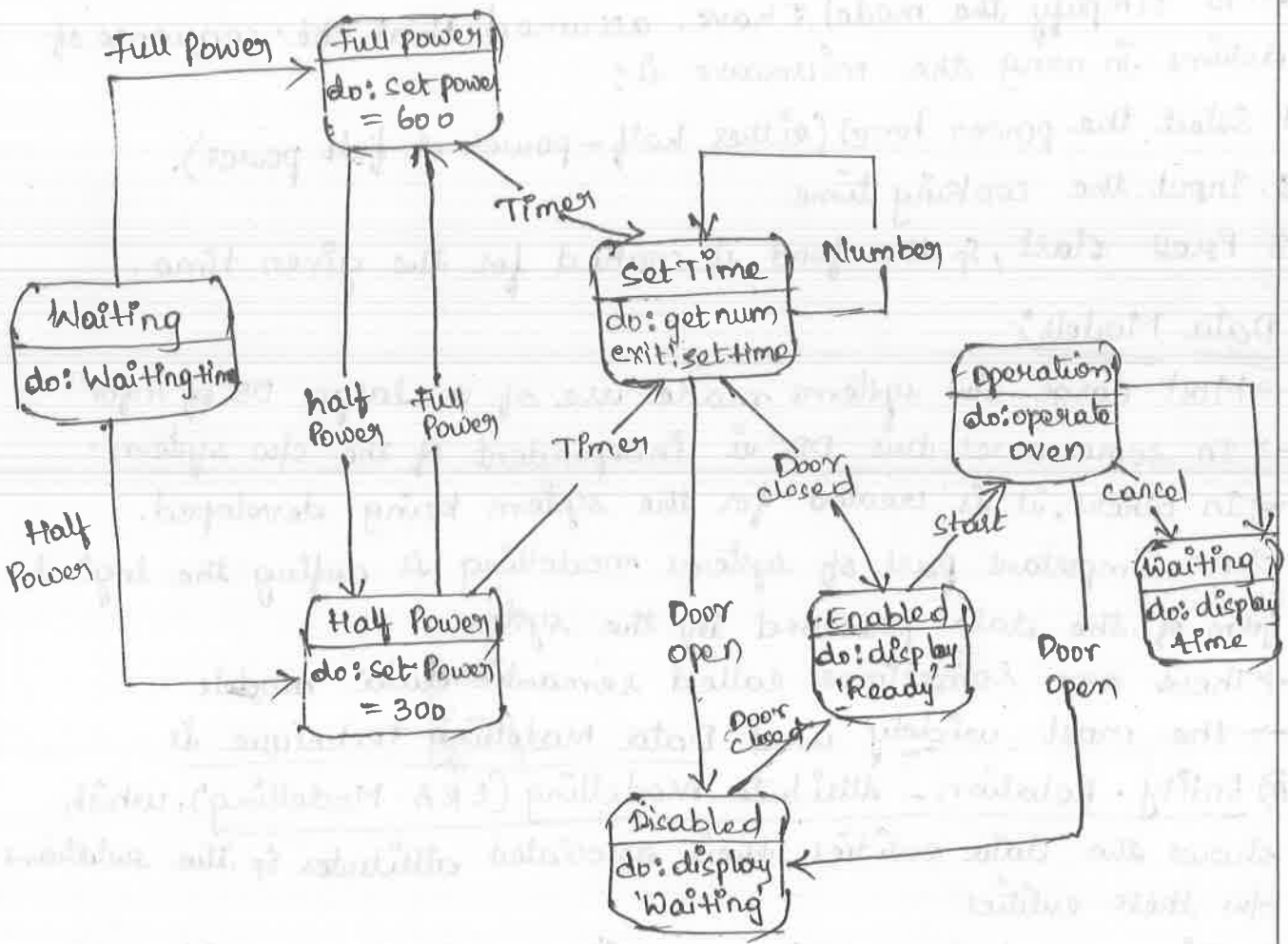


Fig:- State Machine model of a simple microwave oven

→ State machine models are an integral part of real-time design methods such as that proposed by Ward and Mellor (Ward & Mellor, 1985) and Harel (Harel, 1987 & Harel 1988).
→ Harel's method uses a notation called statecharts and these were the basis for the state-machine-modelling notation in the UML.
→ A state machine model of a system assumes that, at any time the system is in one of a number of possible states.
→ This diagram shows a state machine model of a simple microwave oven equipped with buttons to set the power & the timer & to start the system.
→ This above diagram shows a state machine model of a simple microwave oven equipped with buttons to set the power & the timer & to start the system.
→ Real microwave ovens are actually much more complex than the system described here.
→ To simplify the model, I have assumed that the sequence of actions in using the microwave is:

1. Select the power level (either half-power or full power).
2. Input the cooking time
3. Press start, & the food is cooked for the given time.

3) Data Models:

→ Most large s/w systems make use of a larger DB of infor.
→ In some cases, this DB is independent of the s/w system.
→ In others, it is created for the system being developed.
→ An important part of systems modelling is defining the logical form of the data processed by the system.
→ These are sometimes called semantic data models.
→ The most widely used Data Modelling Technique is
* Entity-Relation-Attribute Modelling (ERA Modelling), which shows the data entities, their associated attributes & the relations b/w these entities.
→ This approach to modelling was 1st proposed in the mid-1970s by Chen, several variants have been developed since then (Codd, 1979,

Fig:- State and stimulus description for the microwave oven.

State	Description
Waiting	The oven is waiting for i/p. The display shows the current time
Half Power	The oven power is set to 300 watts. The display shows 'half power'
Full Power	The oven power is set to 600 watts. The display shows 'Full power'
Set time	The cooking time is set to the user's i/p value. The display shows the cooking time selected & is updated as the time is set
Disabled	Oven operation is disabled for safety. Interior oven light is on. Display shows 'Not ready'
Enabled	Oven operation is enabled. Interior oven light is off. Display shows 'Ready to cook'
Operation	Oven in operation. Interior oven light is on. Display shows the times countdown. On completion of cooking, the buzzer is sounded for 5 sec. oven light is on. Display shows 'Cooking complete' while buzzer is sounding

Stimulus	Description
→ Half Power	The user has pressed the half power button
→ Full Power	The user has pressed the full power button
→ Timer	The user has pressed one of the times buttons
→ Number	The user has pressed a numeric key
→ Door open	The oven door switch is not closed
→ Door closed	The oven door switch is closed
→ Start	The user has pressed the start button
→ Cancel	The user has pressed the Cancel button

Fig:- State & Stimulus description for the microwave oven

Hammer & McLeod, 1981 ; Hull & King, 1987), all with the same basic form

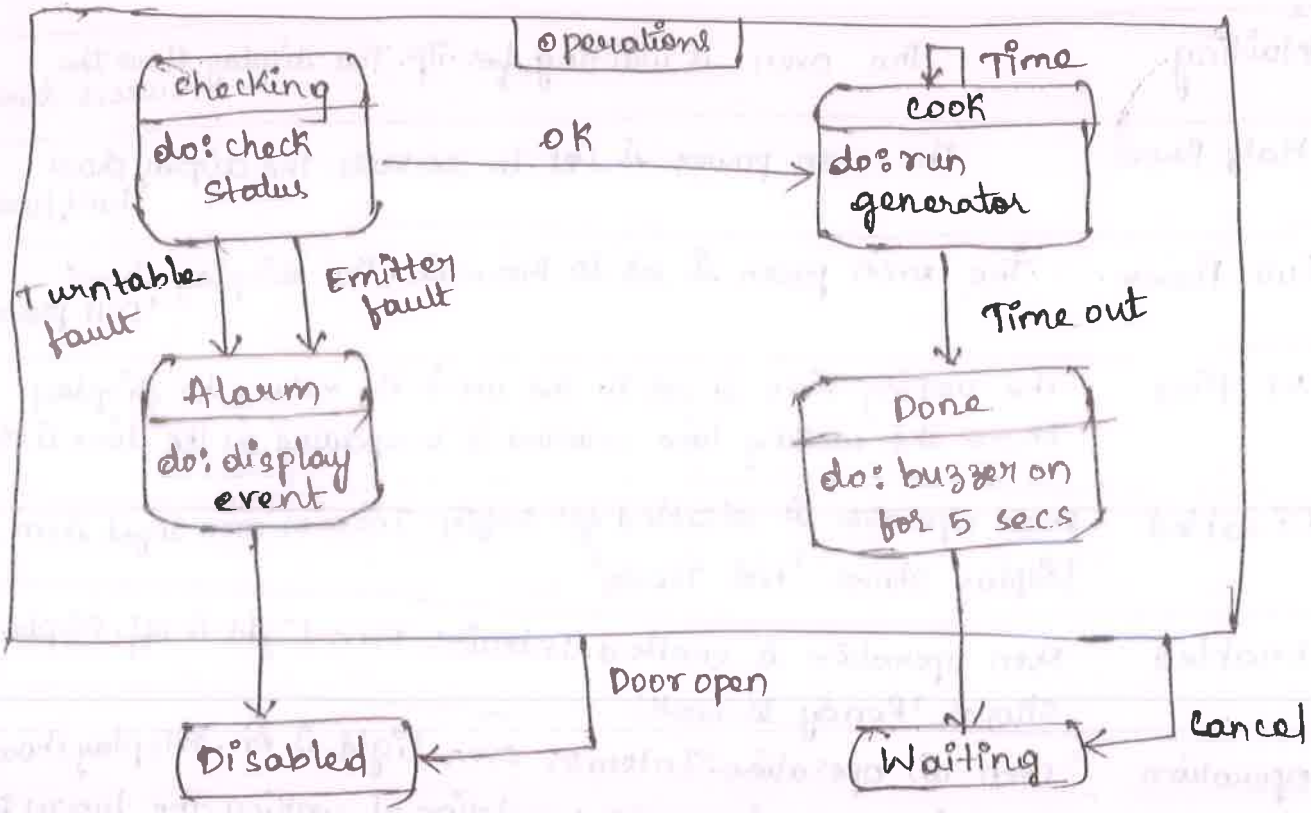


Fig:- Microwave over operation

→ ER-model have been widely used in DB Design.
→ The relational DB schemas derived from these models are naturally in 3rd normal form, which is desirable characteristic.
→ The figure is an example of a data model that is part of the library system LIBSYS introduced.
→ The below figure shows that an Article has attributes representing the articles has attributes representing the title, the authors, the name of the PDF file of the article & the fee payable.
→ This is linked to the source, where the article was published & to the copyright agency for the country of publication.
→ Both Copyright agency & source are linked to the country.
→ The country of publication is important because copyright laws vary by country.
→ The Diagram also shows that Buyers place orders for Articles

→ Like all graphical models, Data models lack detail, & you should maintain more detailed descriptions of the entities, relationships & attributes that are included in the model

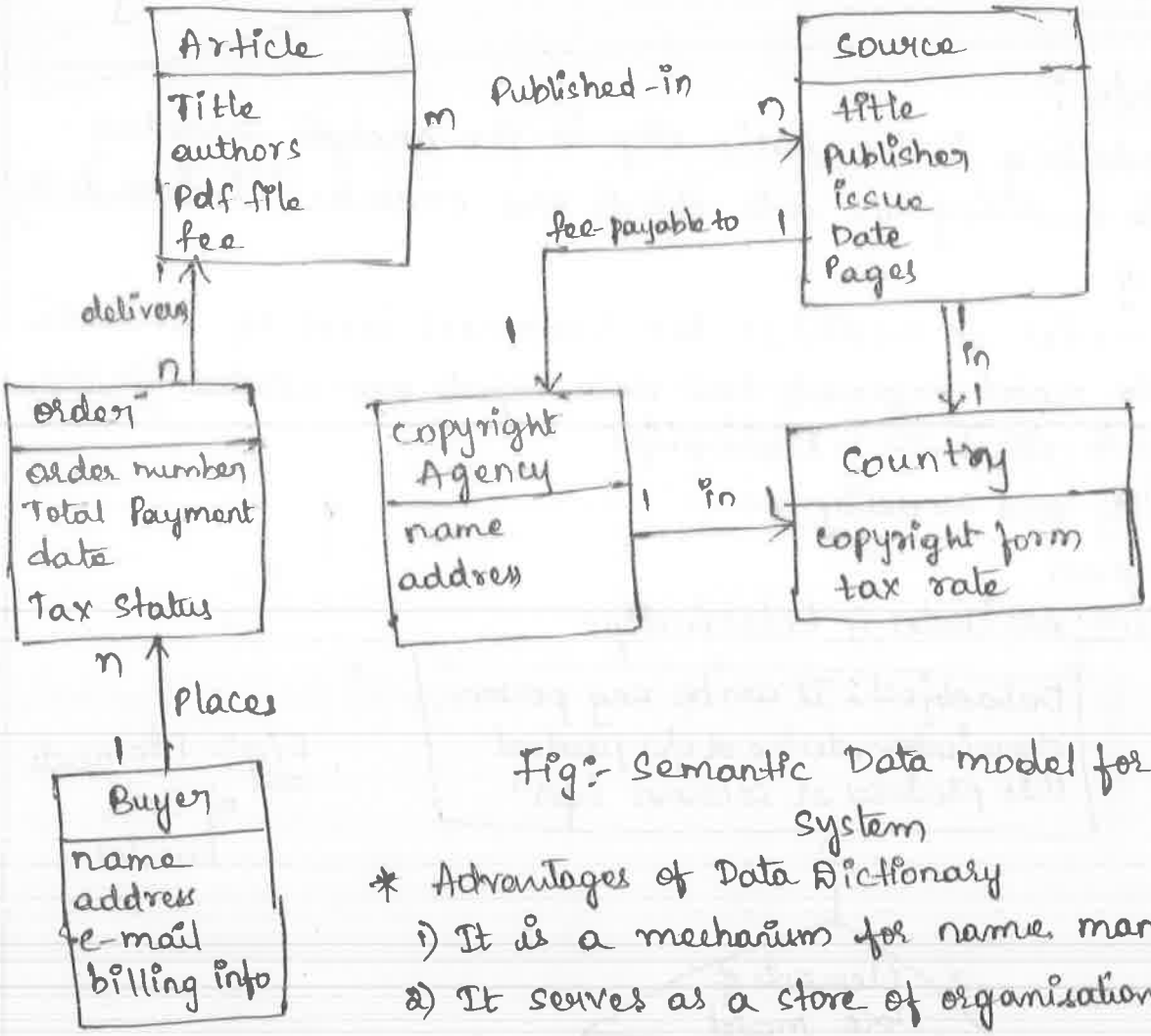


Fig:- Semantic Data model for the LIBSYS System

* Advantages of Data Dictionary
  1) It is a mechanism for name management
  2) It serves as a store of organisational Information

Fig: Examples of Data Dictionary entries

Name	Description	Type	Date
Article	Details of the published article that may be ordered by people using LIBSYS	Entity	30.12.2002
Authors	The names of the authors of the article who may be due a share of the fee	Attribute	30.12.2002
Buyer	The person or organisation that orders a copy of the article	Entity	30.12.2002
fee-payable-to	A 1:1 relationship b/w Article & the copyright agency who should be paid the copyright fee	Relation	29.12.2002
Address (Buyer)	The address of the buyer. This is used to carry paper billing info" that is required	Attribute	31.12.2002

* ~~Data~~

Object Models :- classified as → Inheritance models
→ Object Aggregation
→ Object Behaviour modelling

3) Data Model :-

→ Data modelling is the basic step in the analysis modeling.
→ In Data modelling the data objects are examined independently of processing.
→ And a model is created at the customer's level of abstraction
→ The data model represents how data objects are related with one another.

1) Data objects, Attributes & Relationships
2) Cardinality and Modality
3) ER - Diagram

1) Data objects, Attributes & Relationships

Dataobject : It can be any person, organisation, device or s/w product that produces or consumes infor^n

Elements of Data model

Attributes : These are used to name Data object instance to describe characteristics or to make reference to another Data subject object

Relationship :- These represent how data objects are connected to one another

Fig :- Elements of Data model.

1) Data object :- It is a set of attributes (Data Items) that will be manipulated with the S/W (system)
→ Each instance of data object can be identified with the help of unique identifier. For example :- A Student can be identified by roll num.
→ Data object is a collection of attributes that act as an aspect, characteristic, quality, or descriptor of the object

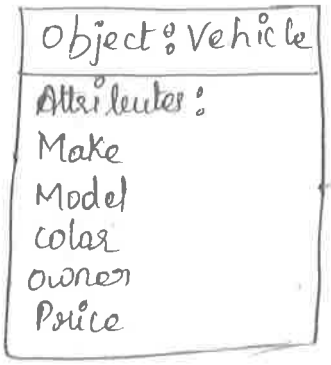


Fig:- Object

Typical Data objects are
- External entities such as printer, user, speakers
- Things such as reports, displays, signals
- Occurrences or events such as interrupts, alarm, telephone call.
- Roles such as manager, engineer, customer.
- organizational units such as division, departments.
- Places such as manufacturing floor, workshops
- Structures such as student records, accounts, file.

* Attributes:- Attributes define properties of data object.

→ Typically there are 3 types

1) Naming :- Used to name an instance of Data object Eg:- vehicle → make, model, Naming

2) Descriptive : Used to describe characteristics or features of Data object

Ex:- Vehicle → color → Descriptive

3) Referential :- Used for making the reference to another instance in another Table. Ex:- Vehicle → owner (Data object) → Referential Attribute

* Relationship:- It represents the connection b/w Data objects.

Eg:- Shopkeeper and a toy

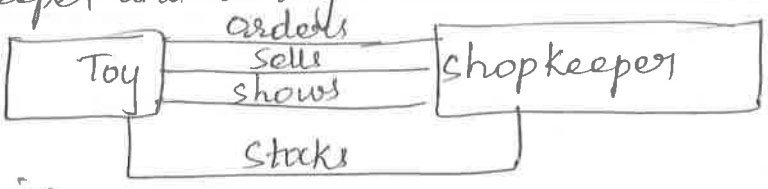


Fig:- Relationship

- Relationships:-
- Shopkeeper orders toys
- " sells "
- " shows "
- " stocks "

2) Cardinality and Modality :

Cardinality in data modelling, cardinality specifies how the num. of occurrences of one object is related to the num. of occurrences of another object.

- one-to one (1:1) - one object can relate to only one other object.
- one to many (1;N) - " " " " " many objects
- many to many (M;N) - some num. of occurences of an object can relate to some other number of occurences of another object

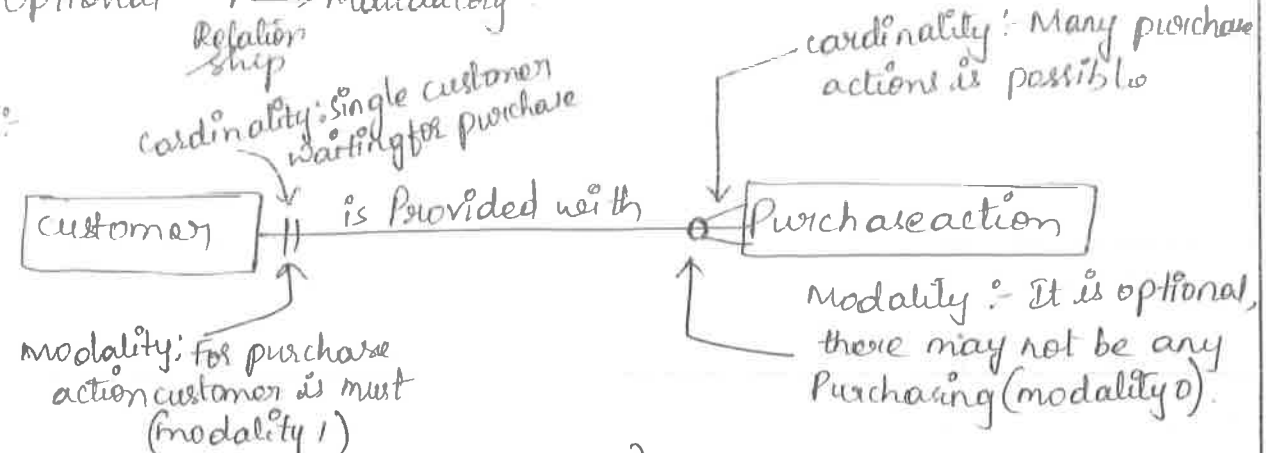
Modality indicates whether or not a particular data object must participate in the relationship

0 —> Optional 1 —> Mandatory

Relation-ship

Relation ship

Example :



3) ER-Diagram :- (Entity Relationship D)

→ Object relationship pair can be graphically represented by a ERD

→ ERD is mainly used in DB appl^n but now it is more commonly used in Data Design.

→ It was proposed by Peter Chen for design of Relational DB systems

→ Primary purpose of ERD is to represent the relationship b/w Data Objects

- Components of ERD are

1) Entity :- (▭) It is an object that exists & is distinguishable.

2) Relationship :- (◇) - Relationships may have attributes & cardinality (eg :- 1 to many)

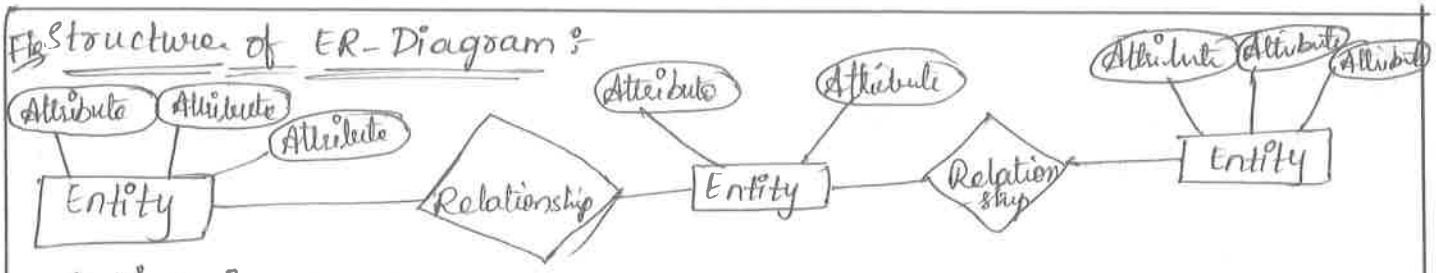
3) Attribute :- (⬭)

→ Similar to record fields in a programming lang.

→ Each attribute has a set of permitted values called domain

→ Primary key attributes may be underlined.

Structure of ER-Diagram :-



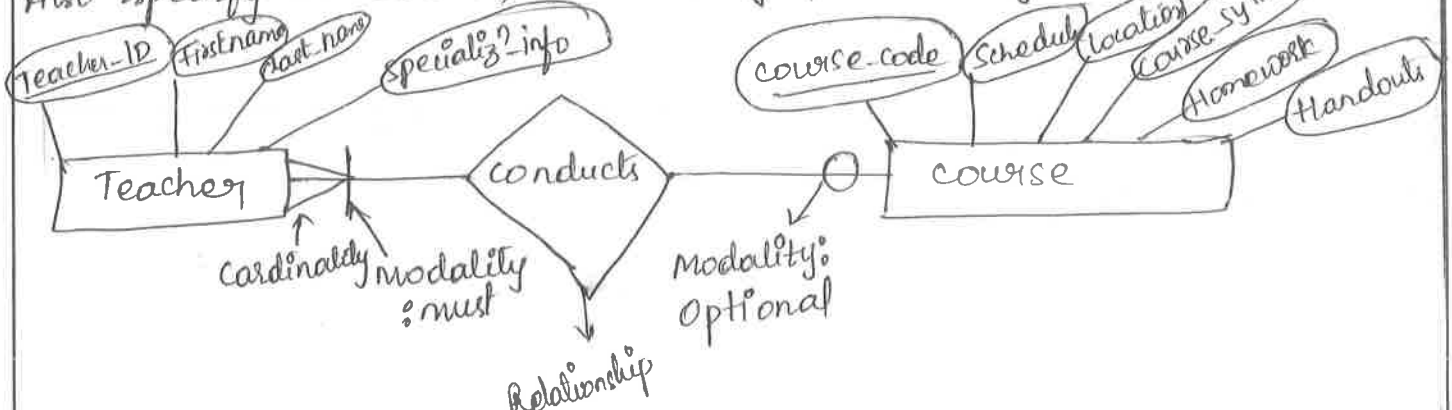
Notations in ER-D :-

1) Entity
2) Entity — Weak Entity :- This entity is dependent upon another entity
3) Attribute — Properties of entity
4) Attribute — Derived Attribute :- Attribute based on another attribute
5) Attribute — Key attribute :- Unique attribute representing entity. Primary key of record is a Key attribute
6) Attribute — Multivalued Attribute :- It has more than 1 value
7) Relationship — Relationship :- 2 entities share some info^n then it is denoted

Notations to show Cardinality :-

1) —— one to one
2) |—< one to many (must)
3) >—| many to one
4) >|— one or more (must)
5) ||— one & only one (must)
6) O|— 0 or one (optional)
7) >O— 0 or many (optional)

Example :- Draw an ER-Diagram for relationship of Teacher & courses
Also specify association, cardinality & modality.

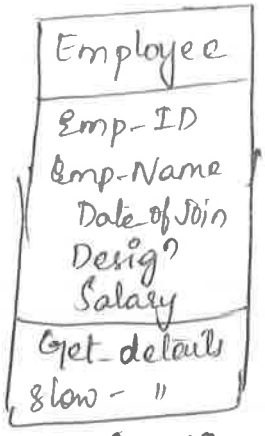


4) ✱ Object Model :-

→ It is created for the systems possessing object oriented programming approach.
→ Generally object models are created for interactive systems
→ such systems can then be implemented using OOPL (object oriented pgmg lang) such as C++ or JAVA
→ For the creation of object model during requirement analysis the info about data & its processing is collected
→ It is a natural representation of real world entities
→ Such entities can be student, book, Customer, Account, Ticket Car & so on.
→ In some object model how the object gets aggregated from other objects is shown while in other object model the Interaction b/w objects can be represented.

UML :- UML is for object modelling
→ various modelling Diagrams such as can be created using UML such as class Diagram, Use-Case Diagram, Sequence Diagram, State Diagram & component Diagram

object class in UML :-



→ In this section we will discuss 3 types of model :-

1) Inheritance model
2) Object aggregation
3) " Behaviour model.

1) ~~Object~~ Inheritance model :-

→ It is a property which the child class can inherit some properties from parent class.

Ex:-

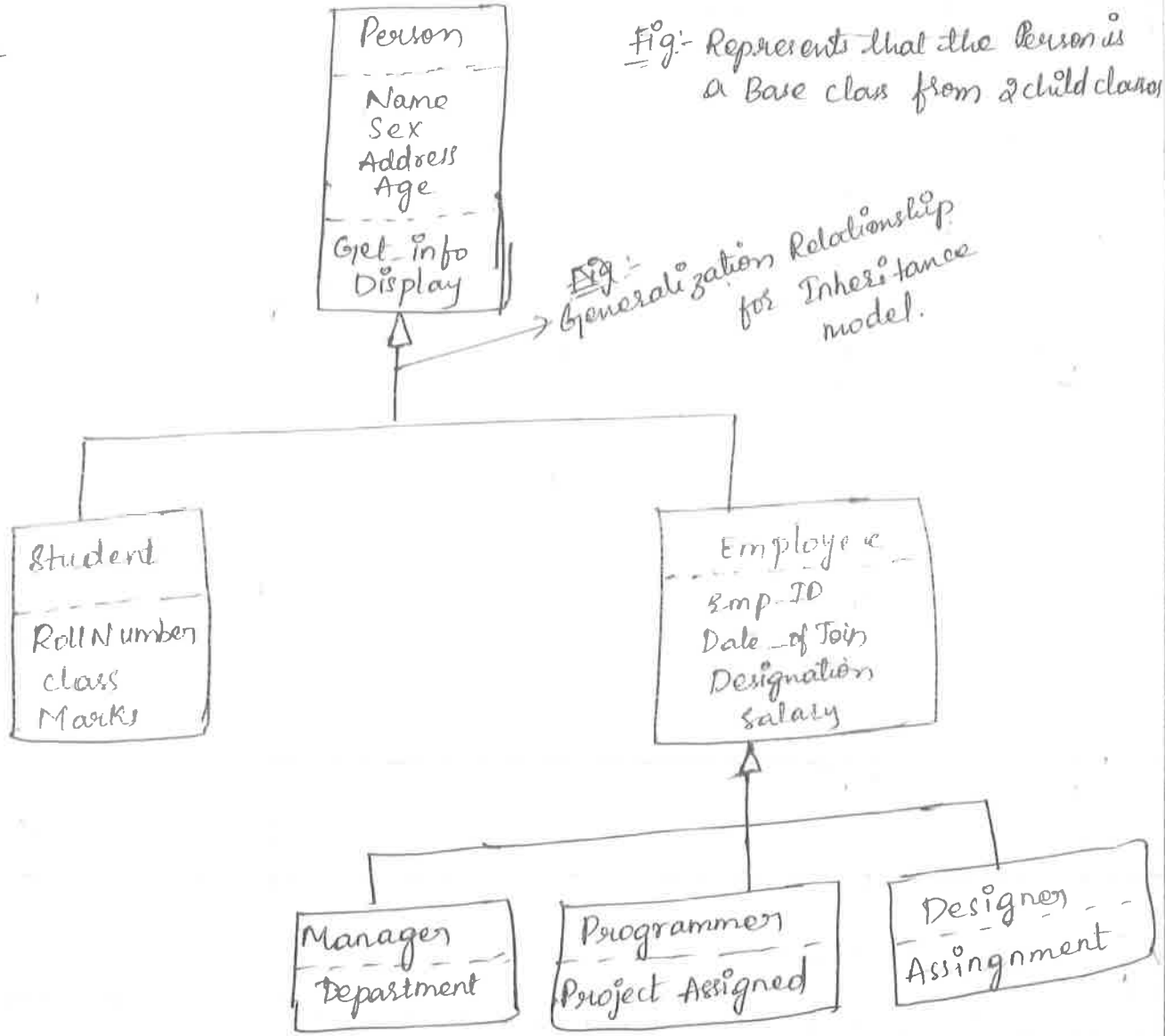


Fig:- Represents that the Person is a Base class from 2 child classes

Fig:- Generalization Relationship for Inheritance model.

2) Object Aggregation:-

→ The Aggregation relationship is a has a relationship

→ ◇ → aggregate Relationship

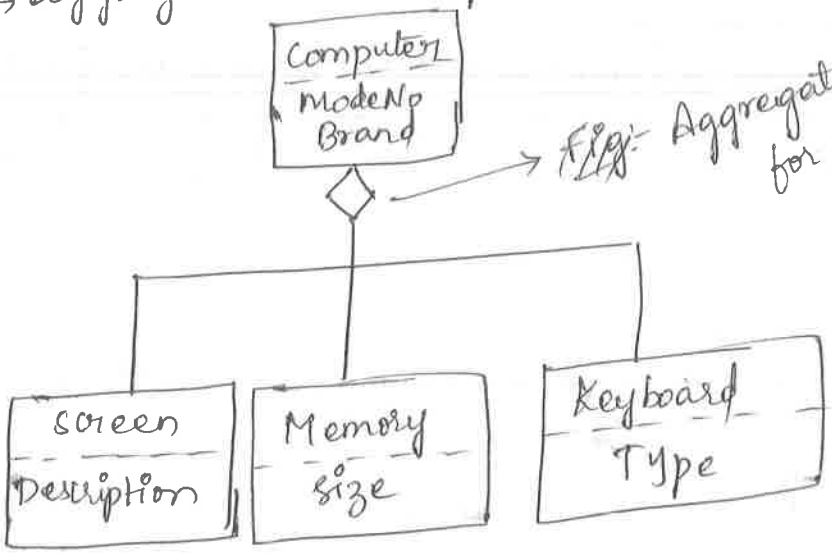
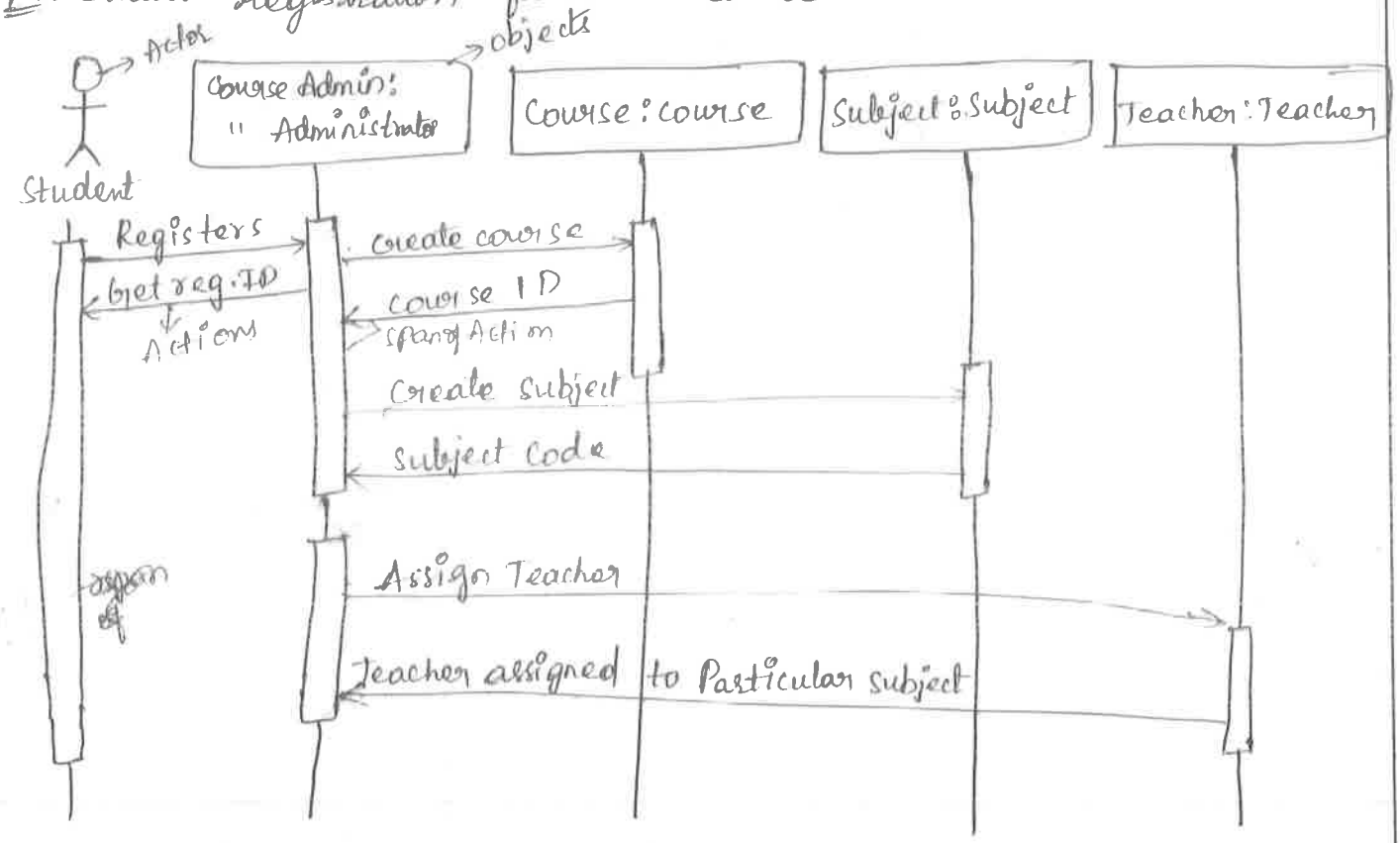


Fig:- Aggregation Relationship for object Aggregation

Fig:- Computer is a class which has screen, memory, & Keyboard

3) Object Behaviour Model :- It is for representing the sequences of events occuring in the system. It is represented by a sequence Diagram in UML.

Ex:- student registration for some course



5) Structured Methods :-

→ It is used for designing different models to represent Particular system.

→ various structured models can be context models, Process models, Data flow diagrams, ER Diagrams, object models.

→ It provides the systematic framework for system modelling as a part of requirement elicitation & Analysis

→ Along with these models documen is essential to provide guideline to system development

→ Use of case tool support also help to generate reports or code generation from system model

Drawbacks :-

1) Non-functional requirements can't be represented effectively using structured methods

2) It is hard to predict whether the structured methods are applied to particular problem are suitable for solving that problem or not

3) Even it is not possible to decide whether the structured method is suitable or not

4) sometimes too much documentation hides the basic requirements of system

5) The system models are represented in more detail so casual user can't understand the system because he gets lost in the unnessary details

6) Unnessary detailing is alone in Structured methods

* Design Concepts :-

→ The s/w design concept provides a framework for implementing the right s/w.
→ Following issues are considered while designing the s/w.

1. Abstraction
2. Modularity
3. Architecture
4. Refinement
5. Pattern
6. Information hiding
7. Functional independence
8. Refactoring.

1.) Abstraction :- Many levels of abstractions are there
→ At the highest level of abstraction, a soln is stated in broad terms using the language of the problem environment.
→ At lower levels of abstraction, more detailed description of soln is provided
→ A Procedural abstraction refers to a sequence of instn's that have a specific and limited fun^n.
→ A data abstraction is a named collection of data that describe data object
eg :- Procedure search the data abstraction will be Record. The record consists of various attributes such as Record ID, name, address & designation

2) Modularity :-
→ The s/w is divided into separately named & addressable components that called as modules.
→ Meyer defines 5 criteria that enable us to evaluate design method with respect to its ability to define an effective modular system
1) Modular decomposability :- A Design method provides a systematic mechanism for decomposing the problem into sub-problems. This reduces the complexity of the problem & the modularity can be achieved.
2) Modular composability :- A design method enables existing design components to be assembled into a new system
3) Modular understandability :- A module can be understood as a stand alone unit. It will be easier to build & easier to change.
4) Modular continuity :- small changes to the system requirements result in changes to individual modules, rather than system-wide changes

5) Modular Protection :- An aberrant condition occurs within a module & its effects are constrained within the module.

3) Architecture :- It means representation of overall structure of an integrated system. In architecture various components interact & the data of the structure is used by various components.
→ These components are called system elements.
→ Architecture provides the basic framework for the s/w system so that important framework activities can be conducted in systematic manner.
→ In Architectural Design various system models can be used there are

Model	Functioning
1) Structural Model →	Overall architecture of the system can be represented using this model.
2) Framework model →	This model shows the architectural framework & corresponding applicability.
3) Dynamic model →	This model shows the reflection of changes on the system due to external events.
4) Process model →	The sequence of processes & their functionality is represented in this model.
5) Functional model →	The functional hierarchy occuring in the system is represented by this model.

4) Refinement :- It is a process of elaboration (Planning & modelling)
→ stepwise refinement is a top-down design strategy proposed by Niklaus WIRTH.
→ The architecture of a program is developed by successively refining levels of procedural detail.
→ The process of program refinement is analogous to the process of refinement & Partitioning that is used during requirements Analysis.

5) Pattern :- Broad Appleton the design pattern can be defined as - It is a named nugget (something valuable) of insight which conveys the essence of proven sol^n to a recurring problem within a certain context
→ In other words, design pattern acts as a design sol^n for a particular problem occuring in specific domain. Pattern can be reusable. Pattern can be used for current work & used to solve similar kind of problem with different functionality

UNIT-3

Design Engineering: Design Process & Design quality, Design concepts, The design model, Pattern Based s/w design

Creating an Architectural Design: s/w Architecture, Data Design, Architectural Styles and patterns, Architectural Design, Assessing alternative architectural Designs, mapping Data flow into a s/w Architecture.

Modeling Component-level Design: Designing class-Based components, conducting component-level design, object constraint language, Designing conventional components

Performing User Interface Design: Golden rules, User Interface Analysis & Design, Interface Analysis, Interface Design steps, Design evaluation.

→ Design Engineering:-

* Design:- Design is the only way by which we can accurately translate the customer's requirements into a finished s/w product or system

* Design Engineering:- Design engineering are set of principles, concepts and practices are used in order to develop high quality s/w or Product.

→ Design allows a s/w engineer to model the system or product.

→ Goal of design engineering is to produce a model or representation

→ McGlaughlin suggests 3 characteristics for a good design

1) The design must implement all the explicit requirements in analysis model, & it must accomodate all the implicit requirements desired by a customer.

2) The design must be a readable, understable guide for those who generate code & for those who test & subsequently support s/w.

3) The design should provide a complete picture of s/w, addressing the data, functional & behavioral domains from an implementation Perspective

* Design Process and Design Quality:-

→ s/w Design is an iterative process through which requirements are translated into a "blue print" for constructing the s/w

* Design Process :-

→ It is a sequence of steps carried through which the user requirements are translated into a system or s/w model.

→ The design is represented at high level of abstraction

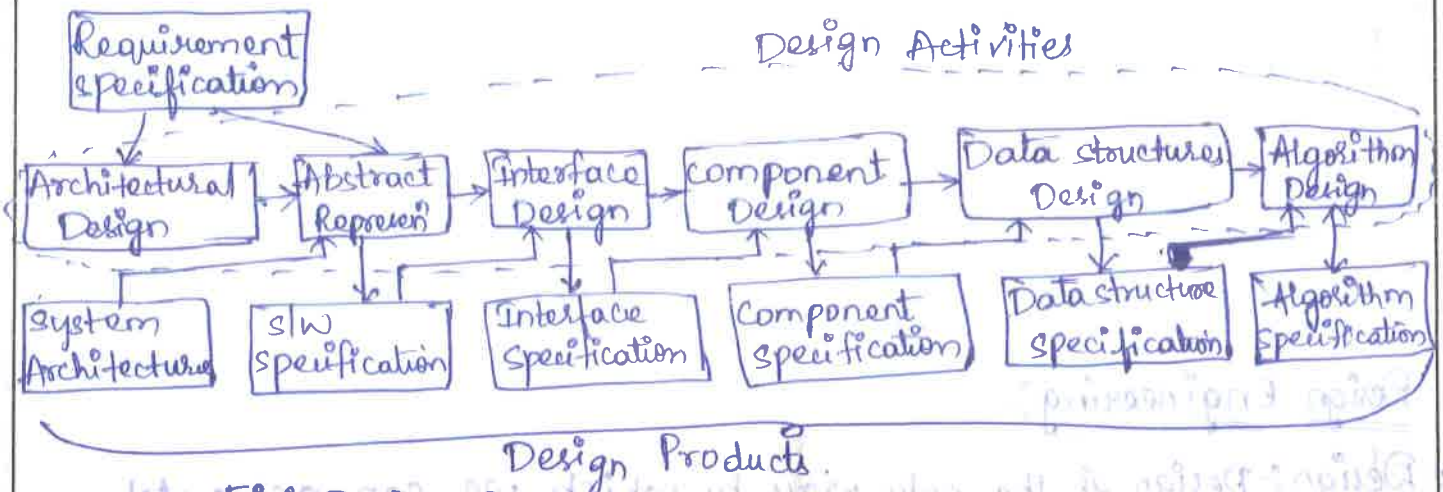


Fig:- Design Process

→ From Requirements specification the architectural Design and abstract specification is created.

→ In architectural Design the sub system components can be identified

→ And the abstract specification is used to specify the subsystems.

→ Then the Interface b/w the subsystems are design which is called Interface Design.

→ In component Design of subsystems components is done.

→ Once the components of the system are identified & designed the decision on use of particular data structures is taken.

→ And the data structure is designed to hold the data.

→ For performing the required functionality, the appropriate algorithm is designed. These algorithms should suit the data structures selected for s/w product.

→ The design process occurs in iterations so that subsequent refinement is possible.

→ At each Design step the corresponding specifications are created.

* Design Principles:-

Davis suggested a set of principles for s/w Design is

→ The design process should not suffer from "tunnel vision"
→ " " should be traceable to the analysis model
→ " " " not reinvent the wheel.
→ " " " "minimize the intellectual distance" b/w the s/w & the problem in the real world.
→ The design should exhibit uniformity & integration
→ " " " be structured to accomodate change
→ " " " " " " degrade gently.
→ Design is not coding & coding is not design
→ The design should be assessed for quality.
→ " " " " reviewed to minimize conceptual errors

* Design quality:

→ The goal of any s/w design is to produce high quality s/w.
→ In order to evaluate quality of s/w there should be some predefined rules or criteria that need to be used to assess the s/w product.
→ Guidelines for quality is as follows.

1. The design architecture should be created using following issues
→ The design should be created using architectual styles & Patterns.
→ Each component of design should posses good design characteristics
→ The imple^{n} of design should be evalutionary, so that testing can be performed at each phase of imple^{n}.
2. In the design the data, architecture, interfaces & components should be clearly represented.
3. The design should be modular. That means the subsystems in the design should be logically partitioned.
4. The Data structure should be appropriately chosen for the design of specific problem.
5. The components should be used in the design so that functional Independency can be achieved in the design
6. Using the infor^{n} obtained in s/w requirement analysis the design should be created

7) The Interfaces in the design should be such that the complexity b/w the connected components of the system gets reduced.
→ i.e the interface system with external interface should be simplified one
8) Every Design of the s/w system should convey its meaning appropriately and effectively.

* Design quality Attributes :-
→ It is popularly known as FURPS (Functionality, Usability, Reliability, Performance & Supportability) is a set of criteria developed by Hewlett & Packard.

~~1) Fun~~

→ Following represents meaning of each quality Attribute.

	Quality Attribute	Meaning
1.	Functionality	→ It can be checked by assessing the set of features & capabilities of the fun's. The fun's should be general & should not work only for particular set of i/p's. i.e the security aspect should be considered while designing fun.
2)	Usability	→ The usability can be assessed by knowing the usefulness of the system.
3)	Reliability	→ It is the measure of frequency & severity of failure. Repeatability refers to the consistency & repeatability of the measures. The mean time to failure (MTTF) is a metric that is widely used to measure the product's performance & reliability
4)	Performance	→ It is a measure that represents the response of system. Measuring the performance means measuring the processing speed, memory usage, response time & efficiency.
5)	Supportability	→ Also called Maintainability. It is the ability to adopt the enhancement or changes made in s/w. It also means the ability to withstand in a given environment.

6) Information hiding :-
→ It is the characteristics of module in which major design decisions can be hidden from all others.
→ Hidding is necessary because infor^n of one module cannot be accessed by another module.
→ Advantage of Infor^n hiding is that modifications during testing & Maintenance can be made independently without affecting other modules.

7) Functional Independence :- It can be achieved by modularity, abstraction & infor^n hiding. It is obtained by creating each module that is Performing only one specific task. This is a key to good design & good design leads a quality in s/w product
→ This is assessed using 2 factors 1) cohension 2) coupling.
1) Cohension :- With the help of cohension the infor^n hiding can be done
→ Cohension module performs only "one task" in s/w procedure with little interaction with other modules.
→ Different types of cohesion are:
1) Coincidentally Cohesive → The modules in which the set of tasks are related with each other loosely then such modules are called coincidentally cohesive
2) logically cohesive :- A module that performs the tasks that are logically related with each other is called logically cohesive.
3) Temporal Cohesion :- The module in which the tasks need to be executed in some specific time span is called temporal cohesion
4) Procedural cohesion :- When processing elements of a module are related with one another & must be executed in some specific order then such module is called procedural cohesion.
5) communicational cohesion :- When the processing elements of a module share the data then such module is communicational cohesive.
→ The goal is to achieve high cohesion for modules in the system.

2) Coupling :- coupling effectively represents how the modules can be "connected" with other module or with the outside world.
→ coupling is a measure of interconnection among modules in a program structure.
→ Coupling depends on the interface complexity b/w modules.
→ The goal is to strive for lowest possible coupling among modules in s/w design

→ The property of good coupling is that it should reduce or avoid change impact & ripple effects.
→ It should also reduce the cost in program changes, testing & maintenance

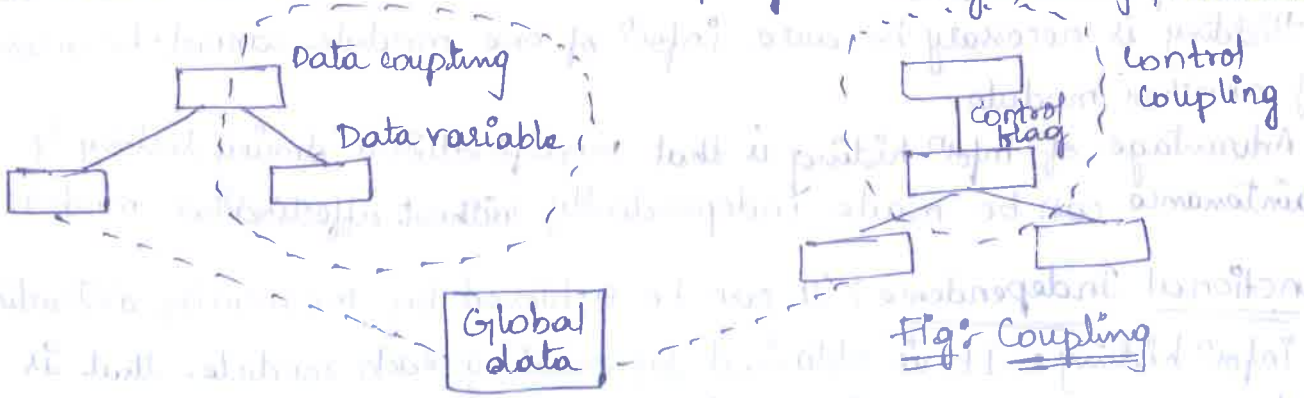


Fig: Coupling

→ Various types of Coupling :-

1) Data Coupling :- The data coupling is possible by parameter passing or data interaction
2) Control Coupling :- The modules share related control data in control coupling
3) Common coupling :- In common coupling common data or a global data is shared among the modules.
4) Content coupling :- It occurs when one module makes use of data or control info maintained in another module.

8) Refactoring :- Refactoring is necessary for simplifying the design without changing the fun or behaviour. Fowler has defined refactoring as the process of changing a s/w system in such a way that the external behaviour of the design do not get changed, however the internal structure gets improved.
Benefits of refactoring are
1) The redundancy can be achieved.
2) Inefficient algorithms can be eliminated or can be replaced by efficient one
3) Poorly constructed or inaccurate data structures can be removed or replaced
4) Other design failures can be rectified.
→ The decision of refactoring particular component is taken by the designer of the s/w systems.

* Design Models :-

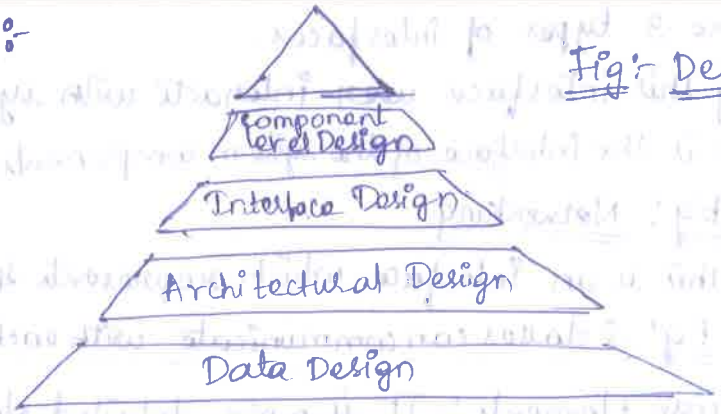


Fig :- Design Model

→ Design model is represented as pyramid. The pyramid is stable object Representing design model in this way means that the s/w design should be stable
→ The design model has broad foundation of data design, stable mid-region with architectural & interface design & the sharp point to for component level Design
→ The design model represents that the s/w which we create should be stable such that any changes should not make it collapsed. And from such a stable design a high quality s/w should be generated.

*1) Data Design Element :- It is high level of abstraction.
→ This data represented at data design level is refined gradually for Implementing the computer Based system.
→ The data has great Impact on the architecture of s/w systems
→ structure of data is very important factor in s/w design.
→ Data appears in the form of data structures & algorithms at the program component level.
→ At the appln level it appears as the DB & at the business level it appears as Data warehouse & data mining.

*2) Architectural Design Element :- It gives the layout for overall view of s/w architectural model can be built using following sources -
- Data flow models or class Diagrams
- Infon obtained from appln domain
- Architectural patterns & Styles

*3) Interface Design elements :- It represents the detailed design of the s/w system.
→ In Interface design how infon flows from one component to other component of the system is depicted.

→ Typically there are 3 types of interfaces.

1) User interface :- By this interface user interacts with system Eg:- GUI

2) External Interface :- This is the interface of the system components with the external entities Eg :- Networking

3) Internal Interface :- This is an interface which represents the inter component comm^n of the system Eg :- 2 classes can communicate with each other by operation

* 4) Component level Design Elements :- It is more detailed design of the s/w system along with the specifications.

→ These design elements describe the internal details of the component.

→ In this all the local data objects, required data structures & algorithmic details & procedural details are exposed.

* Pattern Based s/w Design :-

→ In SE, a s/w design pattern is general, reusable sol^n to a commonly occuring problem within a given context in s/w design.

→ It is not a finished design that can be transformed directly into source or machine code.

→ It is a description or template for how to solve a problem that can be used in many different situations.

→ Design patterns are formalized best practices that the programmer can use to solve common problems when designing an appl^n or system.

* Pattern :- It is a reusable sol^n to a commonly occuring problem within a given context in s/w design.

* Creating an Architectural Design:

S/W Architecture:

→ The architectural Design is the design process for identifying the subsystems making up the system & framework for subsystem control & comm°

→ The goal of architectural design is to establish the overall structure of S/W system. It represents the link b/w design specification & actual design process

Definition:

S/W Architecture is a structure of systems which consists of various components, externally visible properties of these components & the inter-relationship among these components.

* Importance of S/W Architecture: (3 reasons)

1.) S/W Architecture gives the representation of the computer-based system that is to be built. Using this system model even the stakeholders can take active part in the S/W development process.

2) Some early design decisions can be taken using S/W architecture hence system performance & operations remain under control.

3) The S/W architecture gives a clear cut idea about the computer based system which is to built.

→ Two types of Partitioning:- 1) Structural Partioning
2) ~~vertical~~ *

1) Structural Partitioning:-

→ The program structure can be partioned horizontally or vertically.

* Horizontally Partitioning:- It separate branches of the modular hierarchy each major program fun°. It can be done by partitioning system into: i/p, data transformation (processing) & o/p. In horizontal partitioning the design making modules are at the top of the architecture.

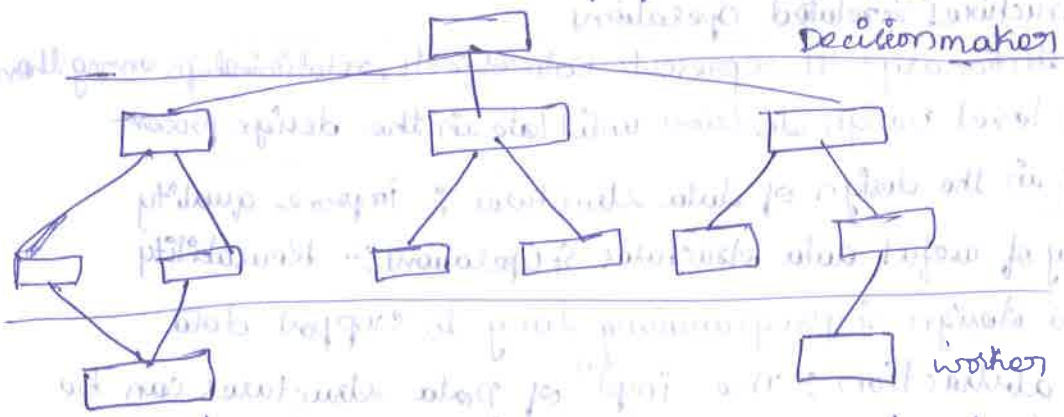


Fig:- Horizontal Partitioning

Advantages:

1) These are easy to test, maintain & extend
2) They have fewer side effects in change propagation & error propagation.

Disadvantage

i) more data has to be passed across module interfaces which complicate the overall control of program flow

* Vertical Partitioning :- It suggests the control & work should be distributed top-down in program structure.

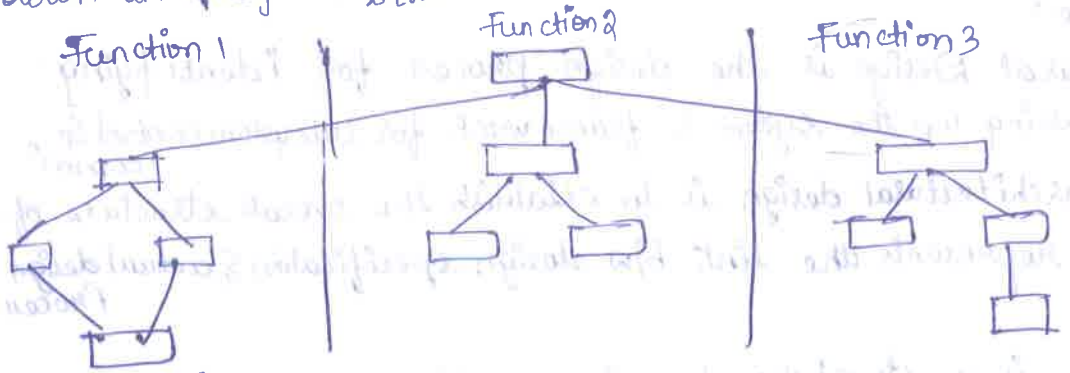


fig :- Vertical Partitioning

→ In vertical Partitioning
- Define separate branches of the module hierarchy for each major fun?
- Use control modules to co-ordinate comm^n b/w fun^n's.

Advantages :- 1) These are easy to maintain the changes
2) They reduces the change impact & error propagation.

* Data Design :-
→ Data Design is basically the model of data, that is represented at the high level of abstraction.
→ The data Design is then progressively refined to create impl^n specific requirements
→ Various elements of Data design are :-

1) Data object :- The data objects are identified & relationship among various data objects can be represented using ER-Diagram
2) DataBases :- Using s/w Design model, the data models are translated into data structures & DB at the appl^n level.
3) Data Warehouses :- At the Business level useful infor^n is identified from various DB & the data warehouses are created.

* Guideline for data Design :-
1) Apply systematic analysis of data :- Represent Data objects, relationships among them & data flow along with the contents
2) Identify Data structures & related Operations
3) Establish Data dictionary :- It represent Data objects, relationships among them
4) Defer the low-level Design decisions until late in the design process
5) Use infor^n hiding in the design of data structures :- improve quality
6) Apply a library of useful data structures & operations :- Reusability
7) Use a s/w design & programming lang to support data specification & abstraction :- The impl^n of Data structures can be done by effective s/w design & by choosing suitable Programming language.

* Architectural Styles and Pattern :-

* Architectural styles :-

→ The Architectural model or style is a pattern for creating the system architecture for given problem.

→ However, most of the large systems are heterogeneous & don't follow single architectural style.

→ System categories define the architectural style.

1. components : They perform a fun^n
Ex: DB, simple computational modules, clients, servers. & filters.
2. Connectors : Enable comm^{n}s. They define how the components communicate, co-ordinate & co-operate Ex: Call, event broadcasting: Pipes.
3. Constraints : Define how the system can be integrated.
4. Sematic models : Specify how to determine a system's overall properties from the properties of its parts.

→ The commonly used architectural styles are

1. Data centered architectures
2. Data flow "
3. Call & return "
4. Object oriented "
5. Layered "

1. Data centered architectures :-

→ In this architecture the data store lies at the centre of architecture and other components frequently access it by performing, add, delete & modify operation

→ Data centered architecture posses the property of interchangeability.

→ Interchangeability means any component from the architecture can be replaced by a new component without affecting the working other components

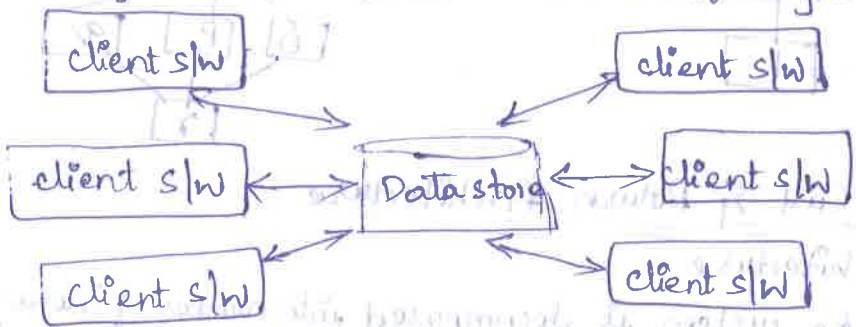


Fig : Data centered Architecture

→ In Data centered architecture the data can be passed among the components.

→ In Data centered architecture
components are : DB elements such as tables, queries.
comm^n are : By relationships
Constraints are : client s/w has to request central data for infor^n.

2) Data flow architecture:

→ In this architecture series of transformations are applied to produce the o/p data.
→ The set of components called filters are connected by pipes to transform the data from one component to another.
→ These filters work independently without a bothering about the working of neighbouring filter

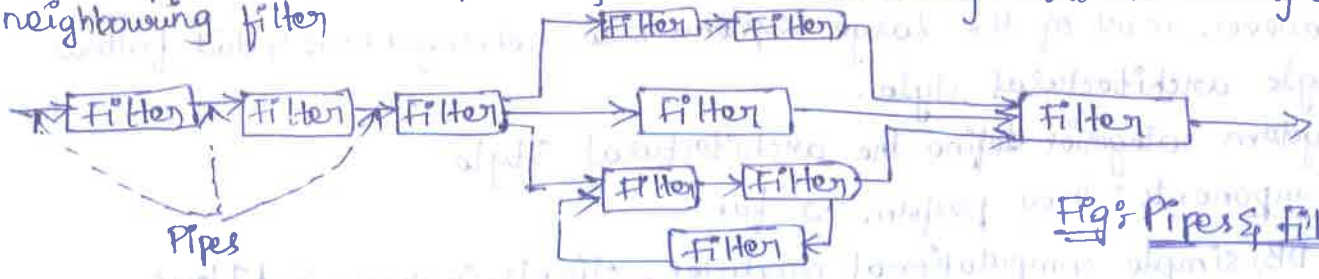


Fig: Pipes & filters

→ If the data flow degenerates into a single line of transforms, it is termed as batch sequential

Fig: Batch sequential.

→ In this pattern the transformation is applied on the batch of data.

3) Call & Return Architecture:

→ The program structure can be easily modified or scaled.
→ The program structure is organized into modules within the program
→ In this architecture how modules call each other.
→ The program structure decomposes the fun into control hierarchy where a main program invokes number of program components.
→ In this architecture the hierarchical control for call & return represented

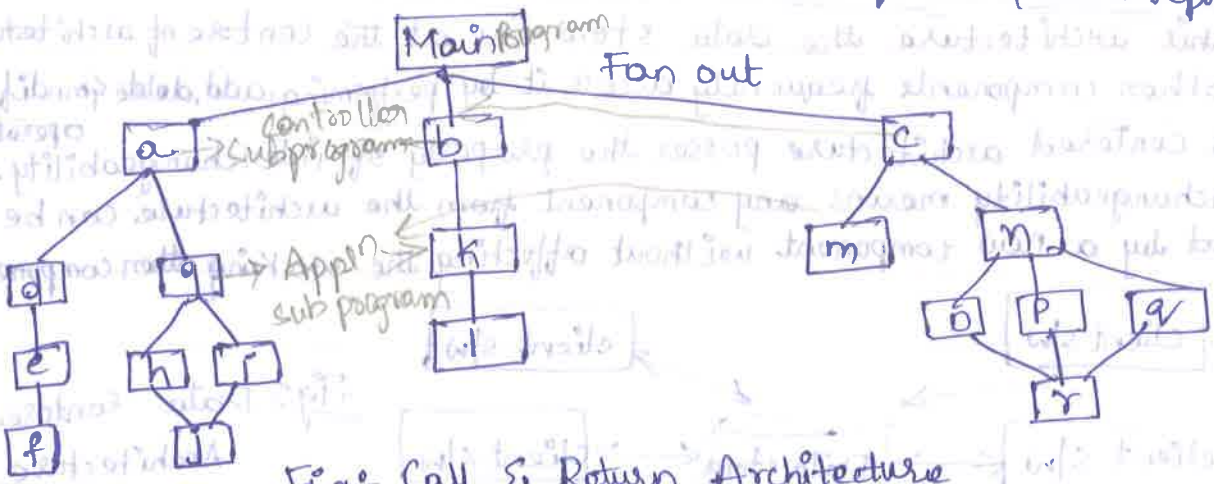


Fig: Call & Return Architecture

4) Object oriented Architecture

→ In this architecture the system is decomposed into number of interacting objects
→ These objects encapsulate data & the corresponding operations that must be applied to manipulate the data
→ The object oriented decomposition is concerned with identifying object classes, their attributes & the corresponding operations. There is some control models used to co-ordinate the object operations

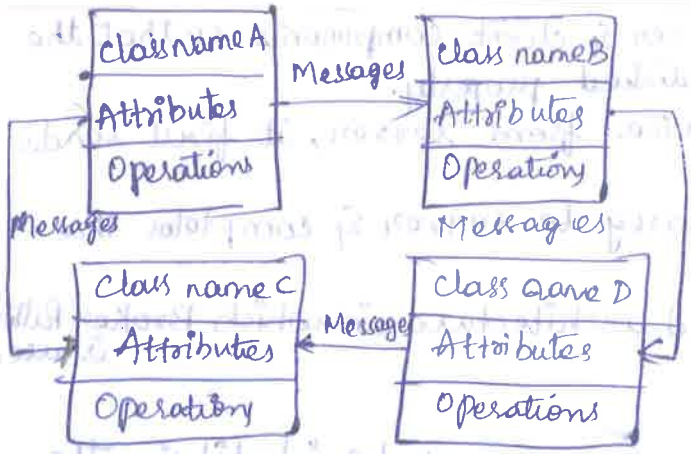


Fig:- Object oriented Architecture

5) Layered Architecture

→ The layered Architecture is composed of different layers.
→ Each layer is intended to perform specific operation so machine instr? set can be generated.
→ various components in each layer perform specific operations.
→ The outer layer is responsible for performing the user interface operations while the components in the Inner layer perform OS interface

Fig:- layered Architecture components

→ The components in intermediate layer perform utility servies & appl? s/w fun's.

2) Architectural Patterns:-

→ The architectural pattern is basically an approach for handling behavioural characteristics of s/w system. Following are pattern domain
1. Concurrency:- concurrency means handling multiple tasks in parallel
Ex: In OS, multiple tasks are executed in parallel. Benefit → efficiency
2. Persistence:- Continuity in the data can be maintained by persistence Pattern
3. Distribution:- Distribution pattern refers to the way in which the system components communicate with each other in distributed systems.
→ There are 2 major problems in distribution pattern
1) Nature of Interconnection of components 2) The nature of comm?
→ These problems can be solved by other pattern called broker Pattern.

→ The broker pattern lies b/w server & client components so that the client server comm^n can be established properly.
→ when client want some service from server, it first sends message to broker.
→ The broker then conveys this msg to server & completes the connection
Ex:- CORBA → This is distributed architecture in which Broker Pattern is used.

* Architectural Design:-

→ The architectural design is the design process for identifying the subsystems making up the system & framework for subsystem control & communication.
→ The goal of architectural design is to establish the overall structure of s/w system.
→ Architectural design represents the link b/w design specification & actual design process.
→ Architectural design at the initial stage a context model is prepared.
→ This model defines the external entities that interact with s/w.
→ Along with this model the nature of s/w interaction with external entities is also described.
→ The context model is prepared by using info^n obtained from analysis model & requirement specification
→ Let us discuss how an architectural design gets generated using some simple representations
1. Representing system in context
2. Defining Archetypes
3. Refining Architecture into components
4. Defining Instantiations of the system → high level,

1. Representing system in context:-

→ Context model is a graphical model in which the environment of the system is defined by showing the external entities that interact with the s/w system
→ In architectural design the architectural context Diagram (ACD) is created
→ The difference b/w context model & architectural context diagrams is that ACD the nature of interaction is clearly described.
→ Following are the basic terminologies associated with Architectural context diagram

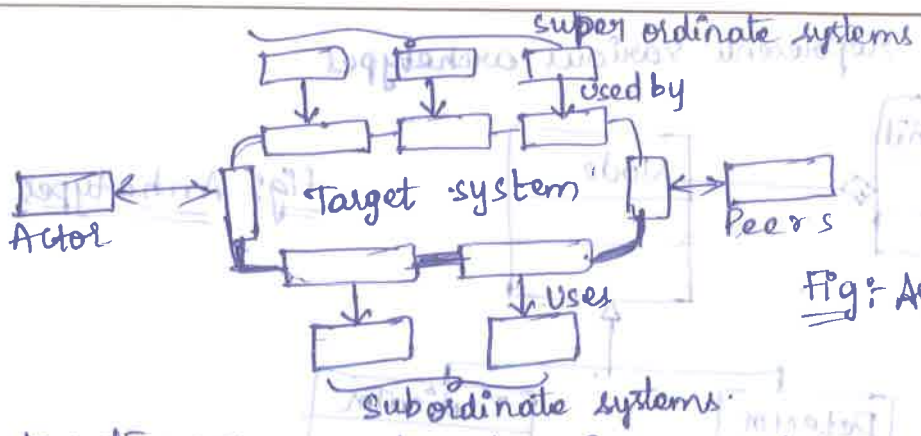


Fig: Architectural context Diagram.

Target system: The target system is a computer based system for which the architectural context diagram has to be prepared.

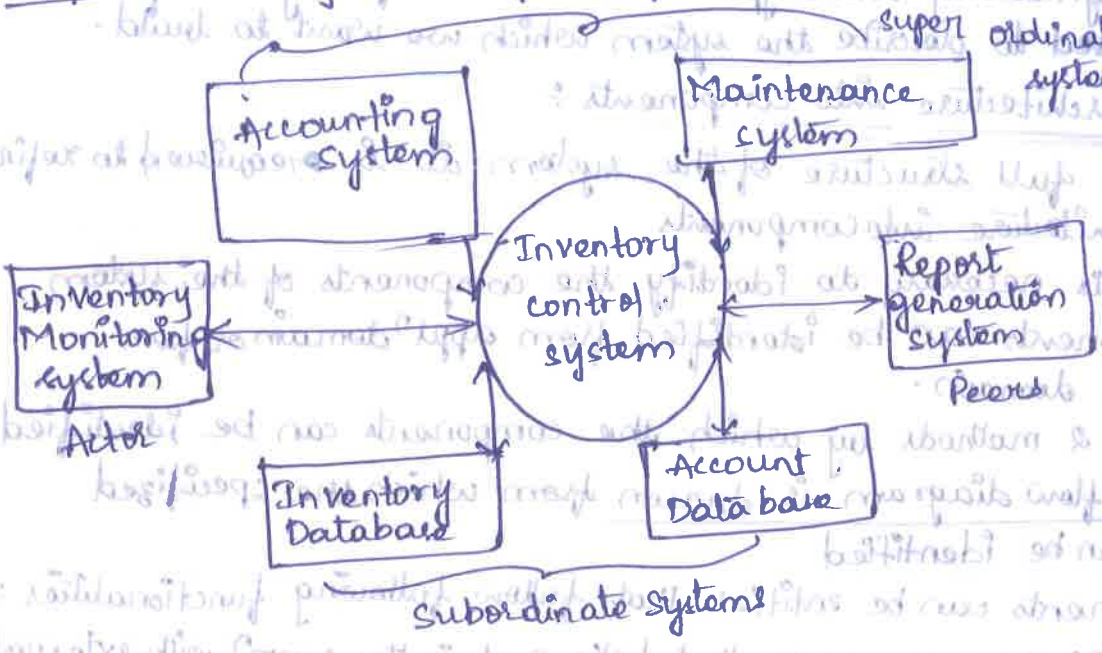
Super ordinate systems: These systems are created at higher level of processing when the target system is being developed.

Sub ordinate systems: These systems are used by the target system for Processing the data that are necessary to complete target system functionality.

Actors: These are the systems or entities that interact with the target system for producing or consuming information of the target system.

Peer-level systems: These are the systems that interact on peer to peer Basis

Example: Inventory control system for which the architectural Context Diagram



2) Defining Archetypes:

→ It is a basic step in architectural design, more precisely in functionality based design.

→ Archetype is a core abstraction using which the system can be structured.

→ Using archetypes, a small set of entities that describe the major part of system behaviour can be described.

→ Archetypes are stable elements & don't change even system changes.

→ Following figure represents various archetypes

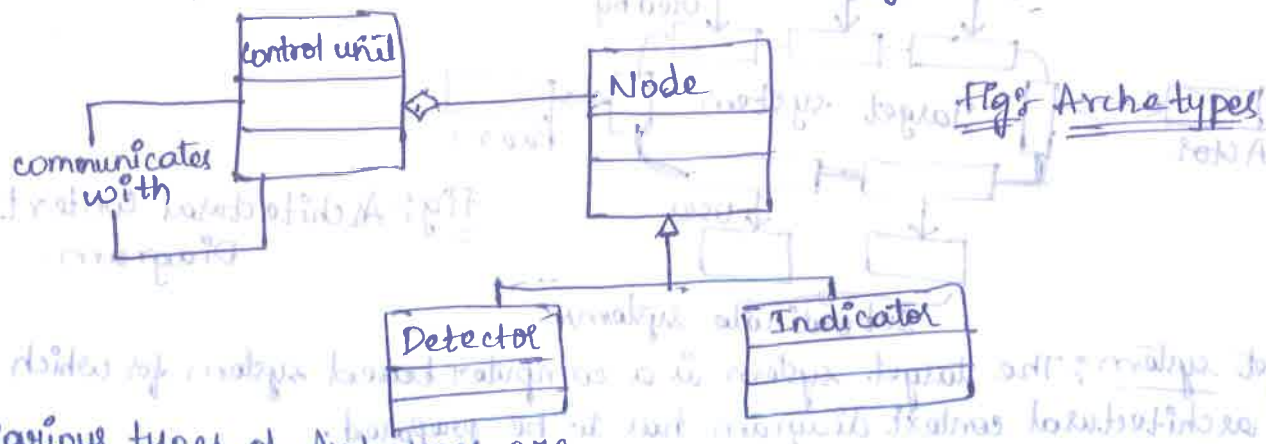


Fig: Archetypes

Various types of Archetypes are

Point or node :- It refers to the highest level of abstraction in which there is a cohesive collection of i/p & o/p functionalities.

Detector:- Detectors capture the core functionalities of the system. For example in temperature control system sensors for temperature is a sensor.

Control unit or controller:- Controllers are the entities that are useful for controlling behaviour of the system.

Indicator or output :- It represents the generic o/p functionalities.

→ In s/w engineering archetype is a number of major components that are used to describe the system which we want to build.

3) Refining architecture into components :-

→ To create full structure of the system it is required to refine the s/w architecture into components.

→ Hence it is necessary to identify the components of the system

→ The components can be identified from appl^n domain or from infrastructure domain.

→ There are 2 methods by which the components can be identified.

1. The Data flow diagram is drawn from which the specified components can be identified

2. The components can be entities that follow following functionalities :-

External comm^n :- The components that take part in the comm^n with external entities are the comm^n components.

Control Panel Processing :- These components perform all control panel management activities.

Detection :- These are the components that perform detection activities

Indicator Management :- These are the components that perform the o/p controlling activities.

4) Describing Instantiation of the system:-

→ The architectural design that has been modeled to this point is still relatively high level.

→ The content of the system has been represented.

→ To accomplish this, an actual instantiation of the architecture is developed.

→ The figure illustrates an instantiation of the safehome architecture for the security system

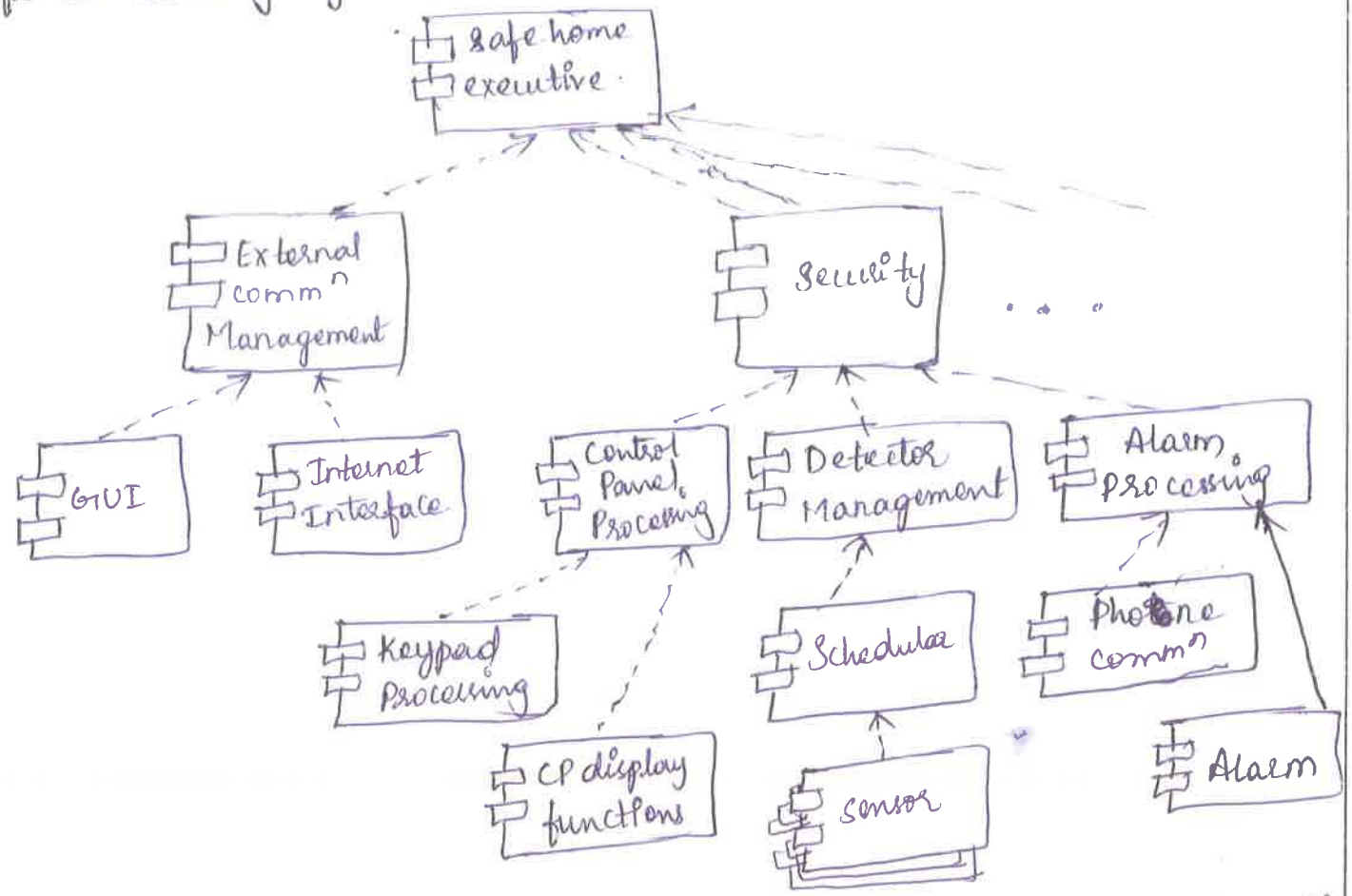


Fig:- An instantiation of the security function with component elaboration

* Assessing Alternative Architectural Design :-

→ In this we have two different approaches for the analysis of alternative Architectural design.

1) → The 1st method uses an iterative method to assess design trade-offs
2) → The 2nd " applies a pseudo-quantitative technique for assessing design quality.

i) An Architecture Trade-off Analysis method :- (ATAM)

→ The s/w engineering Institute has developed an ATAM that establishes an iterative evaluation process for s/w architecture.
→ The design analysis activities that follow are performed iteratively.

1) Collect use cases :- A set of usecases is developed to represent system from the user's point of view
2) Elicit requirements, constraints and environment description :- Used for customer, user & stakeholder.
3) Describe the architectural styles/ patterns that have been chosen to address the scenarios & requirements :-
→ The style should be described using architectural view such as
1) Module view 2) Process View 3) Data flow view.
4) Evaluate quality attributes by considering each attribute in isolation
5) Identify the sensitivity of quality attributes to various architectural attributes for a specific architectural style :-
6) Critique candidate architecture (developed in step 3) using the sensitivity analysis conducted in step 5.

→ The SEI describes this approach in the following manner
→ These 6 steps represent the 1st ATAM iteration.
→ Based on the results of step 5 and 6 some architecture alternatives may be eliminated, one or more of the remaining architecture may be modified & represented in more detail, & then the ATAM steps are applied.

2) Quantitative Guidance for Architectural Design :-

→ One of the many problems faced by s/w engineers during the design process is a general lack of quantitative methods for assessing the quality of proposed designs
→ Work in the area of quantitive analysis of architectural design is still in its formative stages.

→ Asada & his colleagues suggest a number of pseudo-quantitative techniques that can be used to complement the ATAM approach as a method for the analysis of Architecture design quality.
→ The 1st model called spectrum analysis, assesses an architectural Design on a "goodness" spectrum from the best to worst possible designs.
→ Once the s/w architecture has been proposed, it is assessed by assigning a "score" to each of its design dimensions.
→ We then calculate a spectrum Index I_S using the equation

$$I_S = [(S - S_W) / (S_b - S_W)] \times 100$$

S = total score of design
S_W = Worst case scores
S_b = Best case score (optimal design)

→ If modifications are made or new design is proposed, the spectrum Indics for both may be compared & an Improvement Index Imp, may be computed

$$Imp = I_{S1} - I_{S2}$$

Imp = It is +ve, then we can conclude that system 1 has been improved relative system 2

→ Design selection analysis is another model that requires a set of design dimensions to be defined.
→ We calculate a Design selection index (d) as

$$d = (N_S / N_a) \times 100$$

N_S = Number of design dimensions
N_a = total Number of dimensions in design space

3) Architectural Complexity:-
→ A useful technique for assessing the overall complexity of a proposed architecture is to consider dependencies b/w components within the architecture
→ These dependencies are driven by info?/control flow within the system
→ Zhao suggests 3 types of dependencies
1. sharing dependencies :- It represents relationships among customers who use the same resource or producers who produce for the same consumers.
2. Flow dependencies :- It represent relationships b/w producers & consumers of resources
3. Constrained dependencies :- It represent constraints on the relative flow of control among a set of activities.

* Mapping Data flow into a S/W Architecture

→ To illustrate one approach to architecture mapping we consider the call & return architecture - an extremely common structure for many types of systems.

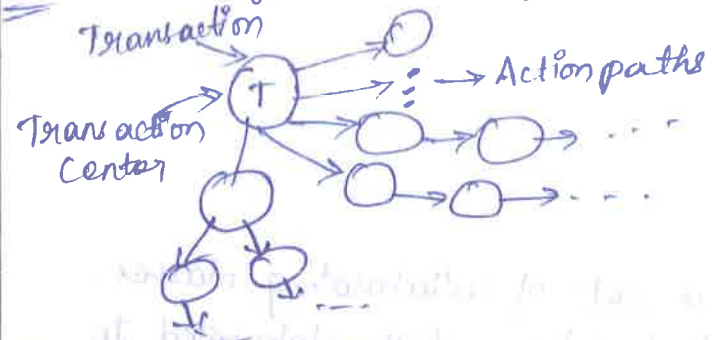
→ the mapping technique to be presented enables a designer to derive reasonably complex call and return architecture from data flow diagrams within the requirement model.

→ In the following sections we examine 2 flow types.

1. Transform flow:

→ Recalling the fundamental system model (level 0 data flow diagram), information must enter & exit S/W in an "external world" form.

Ex:- Data - typed on a keyboard, tones on a telephone line



2) Transaction flow:

→ The fundamental system model implies transform flow.

→ It is possible to characterize all data flow in this category

→ Infor flow is often characterized by a single data item called a transaction, that triggers other data flow along one of many paths

→ Transaction flow is characterized by data moving along an incoming path that converts external world infor into a transaction

→ The ~~transa~~ hub of infor flow from which many action paths emanate is called is called a transaction center.

→ It should be noted that, within a DFD for a large system both transform & transaction flow may be present.

* Modeling Component-level Design

Designing class-Based Components :-

→ Component-level design occurs after the first iteration of architectural design has been completed.

* **Component :** A component is a modular building block for computer s/w.

→ OMG (Object management Group), UML defines a component as a "modular, deployable & replaceable part of a system that encapsulates implementation & expose a set of interfaces".

→ The true meaning of the term "component" will differ depending on the point of view of the s/w engineer who uses it.

→ * 3 important views of what a component is & how it is used as design modeling proceeds.

1. An object-oriented view
2. The conventional view.
3. A Process-Related view.

1. An object-oriented view :-

→ In this a component contains a set of collaborating classes.

→ Each class within a component has been fully elaborated to include all attributes and operations that are relevant to its implementation.

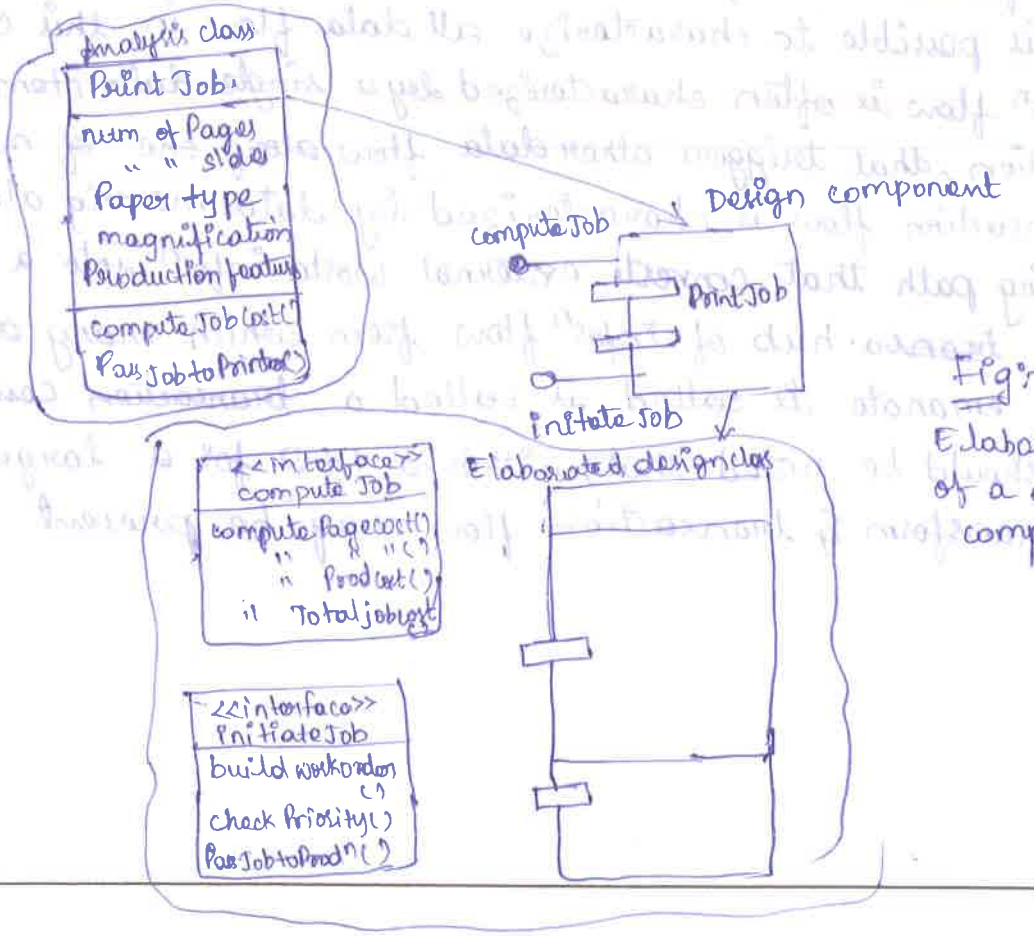


Fig:- Elaboration of a design component.

2) The conventional view :-

→ In the context of conventional s/w engineering, a component is a functional element of a program that incorporates processing logic, the internal data structures that are required to implement the processing logic, & an interface that enables the component to be invoked & data to be passed to it

→ A conventional component, also called module, resides within the s/w architecture & serves one of 3 important roles as :

1) A control component that coordinates the invocation of all other problem domain components.

2) A Problem domain component that implements a complete or partial fun^n that is required by the customer.

3) An Infrastructure component that is responsible for fun^n's that support the processing required in the problem domain.

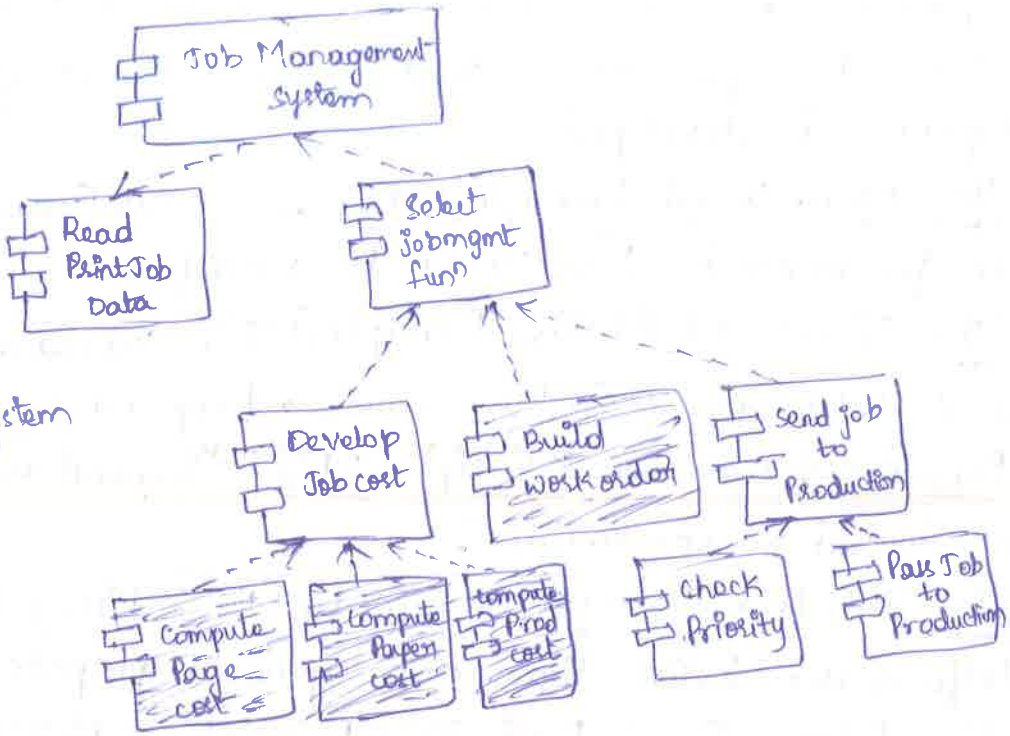


Fig :- Structure chart for a conventional system

3) Process- Related view :-

→ The object-oriented & convential views of component-level design presented in the preceding sections assume that the component is being designed from scratch.

→ That is the designer must create a new component based on specifications designed from the analysis model

→ As the s/w architecture is developed components or design patterns are chosen from the catalog & used to populate the architecture.

* Designing class-Based Components:-

→ We have seen component-level design draws on info developed as part of the analysis model & represented as part of the architectural model.

→ When an object-oriented s/w Engineering approach is choosen, component-level design focuses on the elaboration of analysis classes & the definition & refinement of infrastructure class.

* Basic Design Principles:-

→ Four basic Principles are applicable to component-level design & have been widely adopted when object-oriented SE is applied.

→ The motivation for the appln of these principles is to create designs that are more amenable to change & to reduce the propagation of side effects when changes do occur.

→ These Principles can be used to guide the designer as each s/w component is developed.

1. The open-closed Principle (OCP):- A module [component] should be open for execution but closed for modification.

2. The Liskov Substitution Principle (LSP):- Subclasses should be substitutable for their base classes. Proposed by Barbara Liskov

3. Dependency Inversion Principle (DIP):- "Depend on abstractions. Do not depend on concretions".

4. The Interface Segregation Principle (ISP):- Many client-specific interfaces are better than one general purpose Interface.

5. The Release Reuse Equivalency Principle (REP):- The granule of reuse is the granule of release"

6. The common closure Principle (CCP):- classes that change together belong together.

7. The common Reuse Principle (CRP):- "classes that aren't together should not be grouped together".

* Conducting Component-level Design :-

→ Component-level design is elaborative (↑ additional infor^n) in nature

→ The designer must transform infor^n from the analysis & architectural models into a design representation that provides sufficient detail to guide the construction (coding & testing) activity.

→ The following steps represent a typical task set for component-level design, when it is applied for an object-oriented system

step1 :- Identify all design classes that correspond do the problem domain

step2 :- Identify all design classes that correspond to the infrastructure domain

step3 :- Elaborate all design classes that are not acquired as reusable components.

step3a :- Specify message details when classes or components collaborate

step3b :- Identify appropriate interfaces for each component.

step3c :- Elaborate attributes & define data types & data structures required to implement them.

step 3d :- Describe processing flow within each operation in detail

step4 :- Describe persistent data sources (dB & files) & identify the classes required to manage them.

step5 :- Develop & elaborate behavioral representations for a class or component.

step6 :- Elaborate deployement diagrams to provide additional implementation detail.

step7 :- Factor every component-level design representation & always consider alternatives.

* Object constraint language :- (OCL)

→ The OCL complements UML by allowing a s/w engineer to use a formal grammer and syntax to construct unambiguous stmts. about various design model elements

eg :- classes & objects, events, messages, interfaces.

→ The simplest OCL language stmts are constructed in 4 parts.
(1) a context that defines the limited situation in which stmt is valid
(2) A Property that represents some characteristics of the context (eg:- context → class | property ↳ Attribute)
(3) An operation (eg:- arithmetic)
(4) ~~An~~ ope keywords (eg:- If, then, else, and, or, not, implies) that are used ~~for~~ to specify conditional expressions

* Designing conventional Components :-

→ The foundations of component-level design for conventional s/w components were formed in the early 1960s & were solidified with the work of Edsgar Dijkstra & his colleagues.
→ In the late 1960s, Dijkstra & others proposed the use of a set of constrained logical constructs from which any program could be formed
→ The constructs emphasized "maintenance of functional domain"
→ The following are notations in Designing conventional components
1) Graphical Design Notation
2) Tabular " "
3) Program Design language
4) comparision of Design Notation

1) Graphical Design Notation :-

→ The activity diagram allows a designer to represent sequence, condition & repetition — all elements of structured programming — & is the descendent of an earlier pictorial design repres called flowchart
→ A flow chart like an activity diagram is quite simple pictorially
→ A box is used to indicate processing step, Diamond represents a logical condition & arrows show the flow of control.

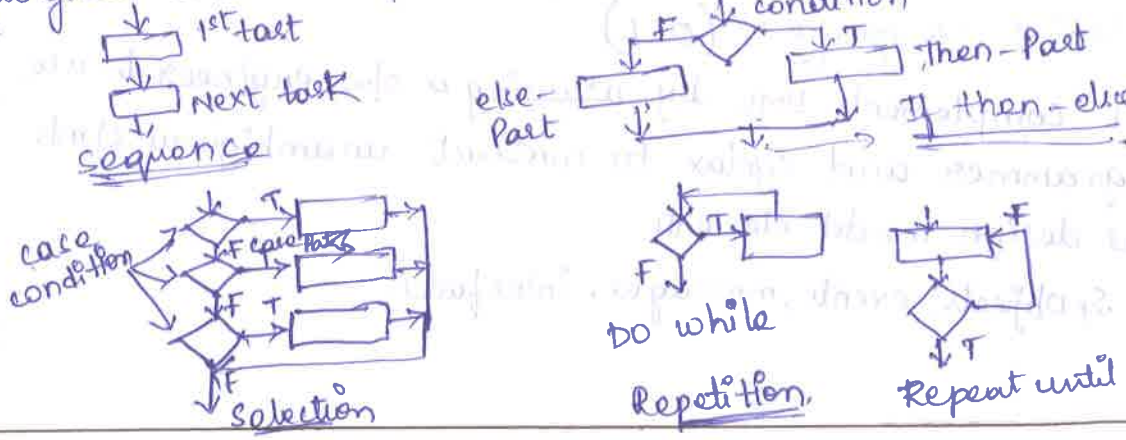


Fig:- Flowchart constructs.

2) Tabular Design Notation :-

→ Decision tables provide a notation that translates actions & conditions into a tabular form.

→ The following steps are applied to develop a decision table

1. List all actions that can be associated with a specific procedure
2. " " conditions during execution of procedure.
3. Associate specific set of conditions with specific actions, eliminating impossible combinations of conditions; Alternatively, develop every possible permutation of conditions.
4. Define rules by indicating what action occurs for a set of conditions.

Fig:- Resultant Decision table

	Rules					
conditions	1	2	3	4	5	6
Regular customer	T	T				
Silver "			T	T		
Gold "					T	T
Special Discount	F	T	F	T	F	T
Actions						
No discount	✓					
Apply 8% Discount			✓	✓		
Apply 15% Discount					✓	✓
Apply additional X% Discount		✓		✓		✓

✱ 3) Program Design Language :-

comm" b/w 2 or more ↑ groups

→ PDL also called structured English or pseudocode, is a pidgin language in that it uses the vocabulary of one language & the overall syntax of another. (i.e., a structured programming language)

→ In this PDL is used as a generic reference for a Design language

→ A 1st glance PDL may look like a programming language.

→ The difference b/w PDL & a real programming language lies in the use of narrative text (eg:- english)

4) Comparision of Design Notation :-

→ Design notation should lead to a procedural representation that is easy to understand.

→ In addition, the notation should enhance "code to" ability so that code does in fact become a natural by product of design

→ Finally design reprsen must be easily maintainable so that design represents program Correctly.

* Performing User Interface Design :-

→ The blue print for a house (its architectural Design) is not complete without a representation of doors, windows & utility connections for water, electricity and telephone.

→ The "doors, windows, & utility connections" for computer s/w make up the Interface design of a system.

→ Interface Design focuses on three areas of concern

1) The design of interfaces b/w s/w components.
2) " " " " " s/w & other non human producers & consumers of infor" (i.e., other external entities)
3) The design of the ~~infor~~ Interface b/w a human (user) & computer

* Golden Rules :-

→ On Interface Design, Theo Mandel coins 3 "golden rules".

1. Place the user in control
2. Reduce the user's memory load
3. Make the Interface consistent.

→ These golden rules actually form the basis for a set of user Interface design principles that guide this important s/w designaction

1. Place the user in control :-

→ During a requirements-gathering session for a major new infor" system, a key user was asked about the attributes of the windows-oriented graphical Interface.

→ She/He wanted a system that reacted to her needs & helped her get things done.

→ she/he wanted to control the computer, not have the computer control her.

→ Mandel defines a number of design principles that allow the user to maintain control.

1) Define Interaction modes in a way that doesn't force a user into unnecessary or undesired actions.
2) Provide for flexible Interaction
3) Allow user Interaction to be Interruptible & undoable
4) Streamline Interaction as skill levels advance & allow the Interaction to be customized.
5) Hide technical Internals from the casual user.
6) Design for direct Interaction with objects that appear on the screen.

2) Reduce the user's Memory load

→ The more a user has to remember, the more error-prone interaction with the system.

→ Mandel defines design principles that enable an interface to reduce the user's memory load:

1) Reduce demand on short-term memory.
2) Establish meaningful defaults
3) Define shortcuts that are intuitive (Easy to remember)
4) The visual layout of the interface should be based on real world metaphor
5) Disclose infon in a progressive fashion

3) Make the Interface consistent:-

→ The Interface should present & acquire infon in a consistent fashion

1) All visual infon is organized according to a Design standard that is maintained throughout all screen Displays.
2) I/p mechanisms are constrained to a limited set that is used consistently throughout the appln &
3) Mechanisms for navigating from task to task are consistently defined & implemented

→ ~~Mandel~~ Mandel defines a set of design principles that help make the Interface consistent:

1) Allow the user to put the current task into a meaningful context.
2) Maintain consistency across a family of applns.
3) If past interactive models have created user expectations, donot make changes unless there is a compelling reason to do so.

* User Interface Analysis & Design:-

→ The overall process for analyzing & designing a user interface begins with the creation of different models of system fun.
→ The human & computer-oriented tasks that are required to achieve system fun are then delineated, design issues that apply to all interface designs are considered; tools are used to prototype & ultimately implement the design model & the result is evaluated by end-users for quality.

1) Interface Analysis & Design model
2) The Process.

1) Interface Analysis & Design model :-
- → User model (To design user interface it is must understad user)
- → Design model (architectural) etc
- → Mental model (system perception)
- → Impln. " (look & feel of Interface)

The role of interface designer is to reconcile these differences & derive a consistent representation of the interface.

2) The Process :-

The analysis & design process for user interfaces is iterative & can be represented using spiral model.

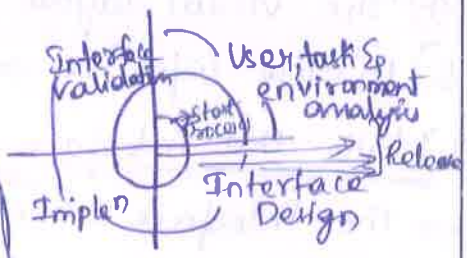


Fig:- The User Interface Design Process.

* Interface Analysis :-

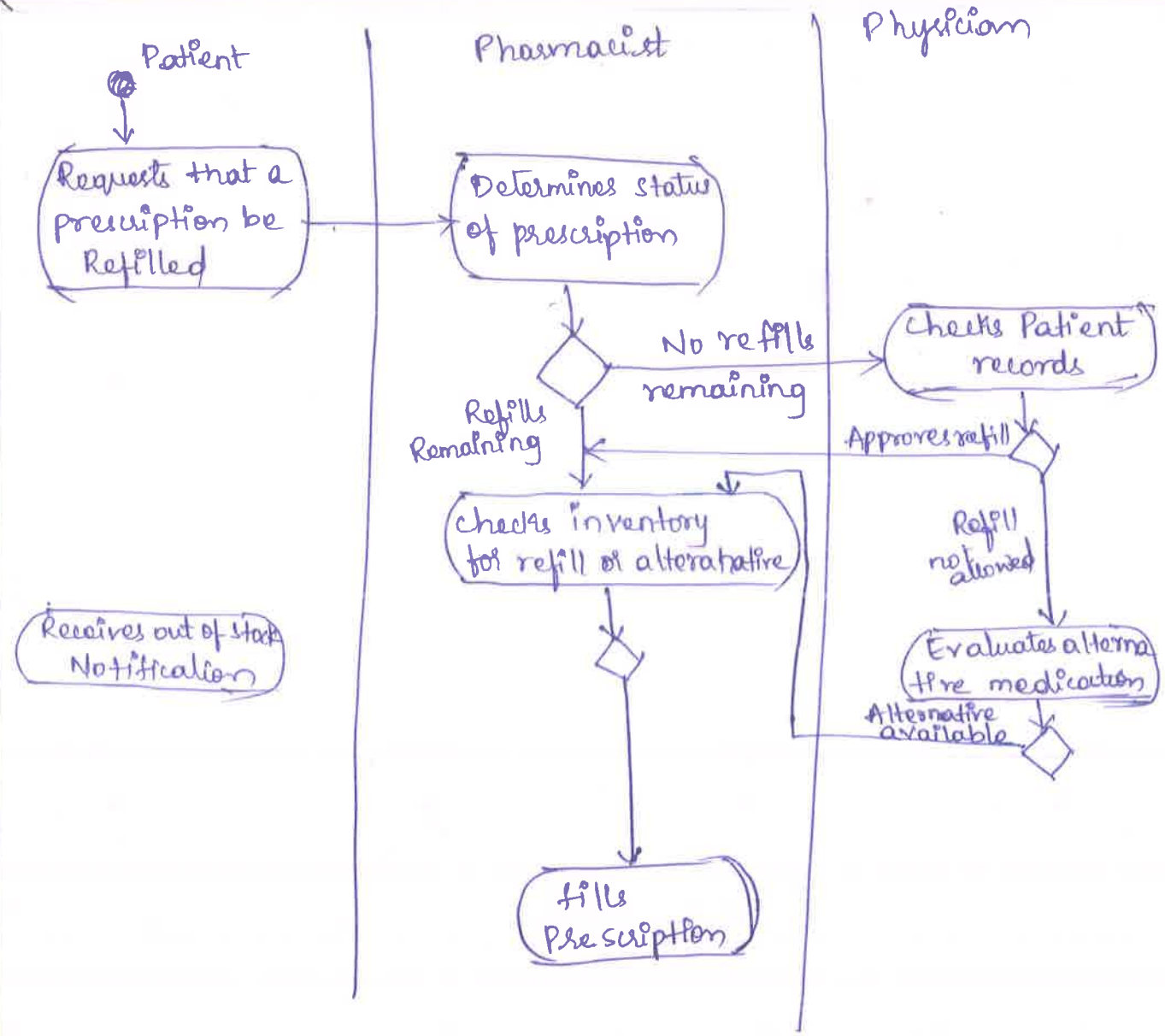
→ In the user Interface analysis design, understanding the problems means understanding
1) The people (end-users) who will interact with the system through the interface.
2) The tasks the end-users must perform to do their work
3) The content that is presented as part of the Interface &
4) The environment in which tasks will be conducted.

→ Design Tasks will follow :-

1) User Analysis
- → User Interviews
- → Sales i/p
- → Marketing i/p
- → support i/p.

2) Task Analysis & Modeling :- (Goal to answer questions)

→ To answer questions s/w engineer must draw upon analysis techniques :-
1) Use cases.
2) Task elaboration (Refining the processing tasks)
3) Object " (User must Perform)
4) Workflow Analysis
5) Hierarchical Representation


Patient
Pharmacist
Physician
Requests that a prescription be Refilled
Determines status of prescription
No refills remaining
Checks Patient records
Refills Remaining
Approves refill
Checks inventory for refill or alterative
Refill not allowed
Receives out of stock Notification
Evaluates alternative medication
Alternative available
fills Prescription

* Interface Design steps :-

→ After Interface analysis all the tasks and corresponding actions are identified.
→ Interface design is an iterative process in which each design process occurs more than once.
→ Each time the design step gets elaborated in more detail.
→ Following are the commonly used Interface design steps -
1) During the interface analysis step define Interface objects & corresponding actions or operations.
2) Define the major events in the interface. These events depict the user actions. Finally model these events.
3) Analyse how the interface will look like from user's point of view.
4) Identify how the user understands the interface with the info^n provided along with it.
→ While designing the Interface s/w engineer communicates the user's according to his thinking about the interface, s/w engineer draws the Sketches
→ while designing the interface the designer has to follow
- Golden rules
- Model the Interface
- Analyse the working environment.

Eg:- Online student registration

Applying design steps :-

1) Appl^n of Interface Design step
→ The 1st step in Interface design is to identify all necessary objects & corresponding actions
3 types of objects
1) Target object (some object can be merged)
2) Source " (object which can be dragged & dropped to other object)
3) Appl^n " (Represents the Appl^n specific data)

2) User Interface Design Pattern :- It serve as sol^n to specific design Problem

Eg:	Pattern	Meaning
1)	Progress Indicator	→ we found this type of pattern in installation program
2)	Wizard	→ It serves as a guideline for completion of task through various windows
3)	Shopping cart	→ It is typically used in on-line purchasing system. It provides the list of items that can be purchased

3) Design Issues (4 Design Issues)

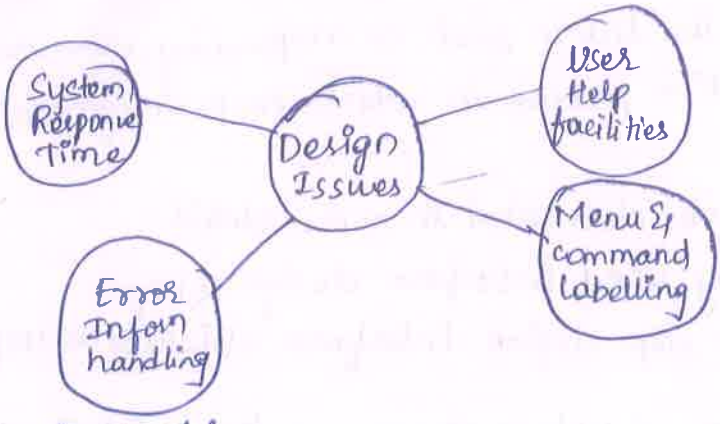


Fig:- Design Issues

* Design Evaluation

→ The Imple° phase of user Interface include creation of prototype that enables user to evaluate the usage scenarios.
→ This evaluation is necessary to determine whether the created interface satisfies user demands or not.
→ After evaluating the usage scenario the user immediately submits his comments to the designer.
→ The designer then makes corresponding modifications in the interface & then again it is submitted to further evaluation.
→ This design evaluated of user Interface is an Iterative Process.

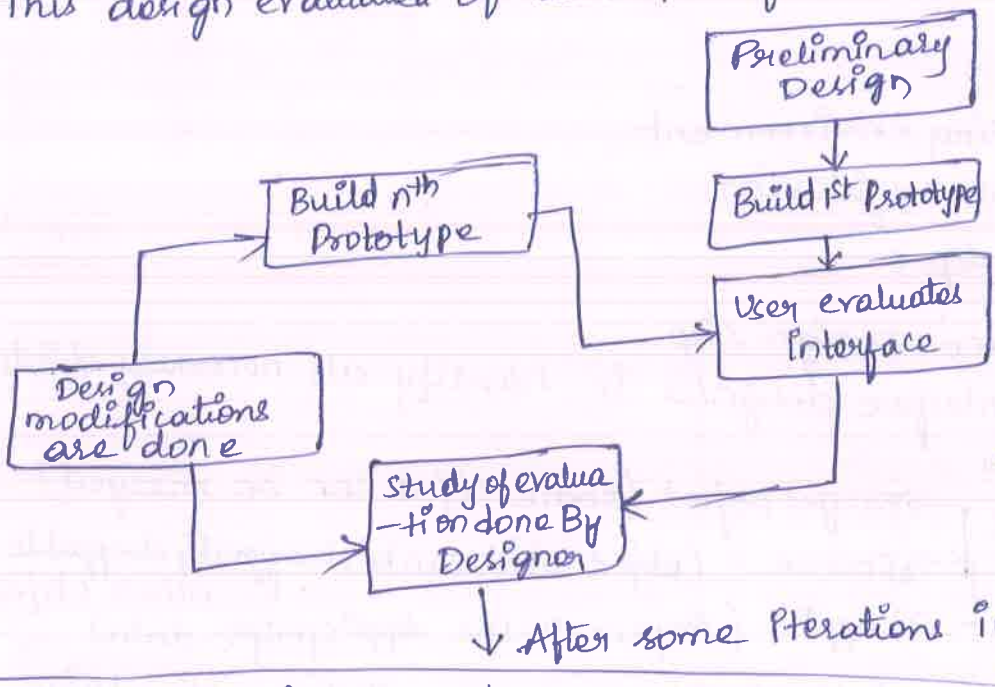


Fig: Design Evaluation

→ After creation of 1st Prototype it is evaluated by the user
→ The evaluation made by the user is then submitted to the designer who then makes necessary changes in the interface & builds the next prototype.
→ This process is repeatedly performed until the user gets satisfied completely & no further modifications are necessary in the Interface.
→ Thus prototype approach proves to be effective for design evaluation.

UNIT-4

Testing Strategies :- A Strategic Approach to S/W Testing, test strategies for conventional S/W, Black-box & white-Box testing, Validation Testing system testing, the art of Debugging.

Product Metrics :- S/W quality, Frame work for Product metrics, Metrics for Analysis model, Metrics for design model, metrics for source code, Metrics for testing, Metrics for maintenance.

Metrics for Process and Products :- S/W Measurement, Metrics for S/W quality

* A Strategic Approach to S/W Testing :-

→ S/W Testing is one of the important phases of S/W development
→ Testing is the process of execution of a program with the intention of finding errors.
→ Involves 40% of total project cost.
→ Testing strategy :- A road map that incorporates test planning, test case design, test execution & resultant data collection & execution.
→ People are not perfect. We make errors in design & code.
→ Hence testing is an essential activity in S/W life cycle.
→ The goal of testing is to uncover as many as errors as possible.
→ The S/W testing is an important activity carried out in order to improve the quality of S/W.
→ For finding out all possible errors the testing must be conducted systematically & test cases must be designed using disciplined techniques.
→ According to Glen Myers testing is a process of executing a program with the intend of finding an error.
→ Testing can be done by by S/W developer & independent testing group.
→ Testing and debugging are different activities. Debugging follows testing
→ Low level tests verifies small code segments.
→ High level tests validate major system function against customer requirements.
→ Perform formal Technical reviews (FTR) to uncover errors during S/W development.

* Testing Strategies for Conventional S/W :-

→ we begin by 'testing-in-the-small' & more toward 'testing in large'

→ various testing strategies for conventional S/W are

1) unit testing
2) Integration testing
3) validation "
4) System testing

1) Unit testing :- In this type of testing techniques are applied to detect the errors from each S/W component individually

Scope = Individual component

Focus is on :- Component correctness

white-box & black box Techniques.

2) Integration testing :- It focuses on issues associated with verification & program construction as components begin interacting with one another.

Integration testing : Scope = set of interacting components.

Focus is on = correctness of component interactions

Mostly black box, some white-box.

3) System testing :- In system testing all system elements forming the system is testing as a whole.

system testing : Scope = entire system

Focus is on = overall system correctness

only black-box techniques.

3) Validation testing :- It provides assurance that the S/W validation criteria (established during requirements analysis) meets all functional, behavioural, & performance requirements

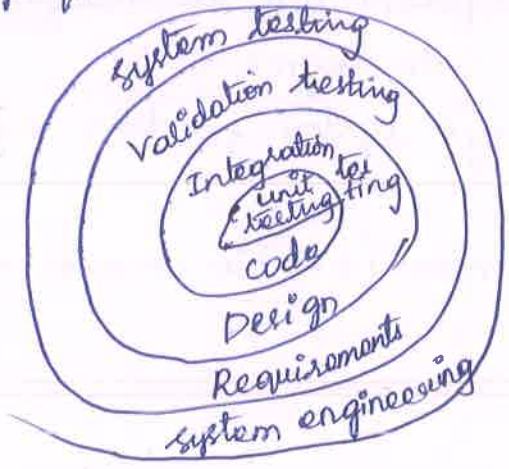
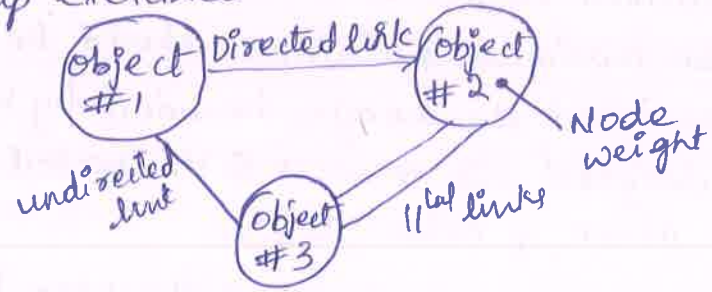


Fig:- Spiral model for conventional S/W in testing strategy

* S/w Testing:- Two major categories of s/w Testing
1) Black box testing
2) white box testing

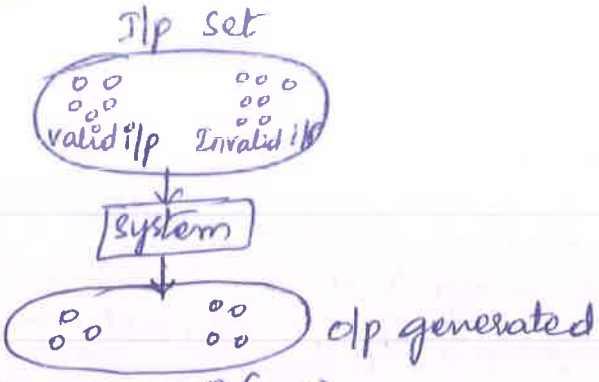
1) Black box testing:- (Also called behavourial testing)
→ Black box testing methods focus on the functional requirements of the s/w
↔ Tests sets are derived that fully exercise all functional requirements.
→ Black box testing is not an alternative to white box testing & it uncovers different class of errors than white box testing
→ Black box testing uncovers following types of errors
1. Incorrect or missing fun^n's.
2. Interface errors.
3. Errors in data structures
4. Performance errors
5. Initialization or termination errors
→ The black box testing is used to demonstrate that the s/w fun^n's are operational. As the name suggests in black box testing it is tested whether the i/p is accepted properly & o/p is correctly produced.
→ The major focus of black box testing is on fun^n's, operation, external interfaces, external data & infor^n.
1) Graph based testing method.
2) Equivalence partitioning

1) Graph based testing:-
→ Draw a graph of objects & relations
→ Devise test cases uncover the graph such that each object & its relationship exercised.



2) Equivalence Partitioning:-
→ Divide all possible i/p's into classes such that there are a finite equivalence classes.

→ Equivalence Partitioning the equivalence classes are evaluated for given i/p condition.
→ Equivalence class represents a set of valid or invalid states for i/p condition
→ Equivalence class guidelines can be as given below.
→ If i/p condition specifies a range, one valid & 2 invalid equivalence classes are defined.
→ If an i/p condition requires a specific value, one valid & 2 invalid equivalence classes are defined.
→ If an i/p condition specifies a member of a set, one valid & one invalid equivalence class is defined.
→ If an i/p condition is Boolean, one valid & one invalid equivalence class is defined.



Boundary value Analysis (BVA)

A boundary value Analysis is a testing technique in which the elements at the edge of the domain are selected & tested

→ Guidelines for BVA are

1) If the i/p condition specified the range bounded by values X & Y then test cases should be designed with values X & Y..

2) If i/p condition specifies the number of values the test cases should be designed with minimum & maximum values as well as with the values that are just above & below the maximum & minimum should be tested

3) If the o/p condition specified the range bounded by values X & Y, then test cases should be designed with values X & Y. Also test cases should be with the values above & below X & Y.

4) If o/p condition specifies the number of values then the test cases should be designed with minimum & maximum values as well as with the values that are just above & below the max & min should be tested.

2) White box testing :- (also called Glass box testing) → Testing of a s/w soln's internal structure, design & coding.

→ The white box testing is a testing method which is based on close examination of procedural details

→ In white box testing the internals of s/w are tested to make sure that they operate according to specifications & designs.

→ Thus major focus of white box testing is on internal structures, logic paths, control flows, data flows, internal data structures, conditions, loops etc.

→ There are 3 main reasons behind performing white box testing

1. Programmers may have some incorrect assumptions while designing & implementing some functions. Due to this there are chances of having logical errors in the program. To detect & correct such logical errors procedural details need to be examined.

→ Certain assumptions on flow of control & data may lead programmer to make design errors. To uncover the errors on logical path, white box testing is must.

→ There may be certain typographical errors that remain undetected even after syntax & type checking mechanisms. Such errors can be uncovered during white box testing

* Cyclomatic Complexity :-

cyclomatic complexity is a s/w metric that gives the quantitative measure of logical complexity of program.

* 3 methods of computing cyclomatic complexities :

Method 1 :- The total number of regions in the flow graph is a cyclomatic complexity

Method 2 :- The cyclomatic complexity, V(G) for a flow graph G can be defined as $V(G) = E - N + 2$ E = total num of edges in flow graph, N = " " " nodes " "

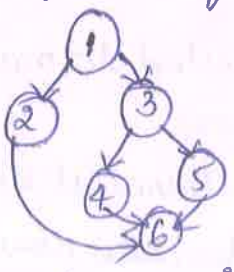
Method 3 :- The cyclomatic complexity V(G) for a flow graph G can be defined as $V(G) = P + 1$ P = Total num. of Predicate nodes, G = Graph.

Example; Code fragment with line numbered.

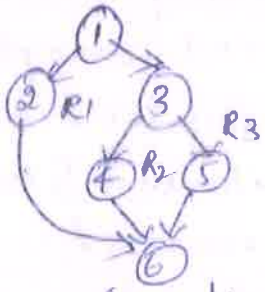
```
{
1. if (a<b)
2.   F1();
   else
{
3. if(a<c)
4.   F2();
   else
5.   F3();
6. }
}
```

→ To compute cyclomatic complexity we follow these steps

step 1: Design flow graph for given code fragment



step 2:- Compute regions, predicate (i.e, decision nodes) edges & total nodes in the graph



→ There are 3 regions denoted by R1, R2 & R3

→ Nodes 1 & 3 are predicate nodes because which branch to be followed is decided at these points.

→ Total edges = 7, Total Nodes = 6 Predicate Nodes = 2 (1 & 3)

step: 3: Apply formula in order to compute cyclomatic complexity

1) cyclomatic complexity = Total num. of regions = 3

2) " " = E − N + 2

∴ " " = 7 − 6 + 2 ⇒ 3.

3) " " = P + 1 ⇒ 2 + 1 ⇒ 3.

cyclomatic complexity is 3 for given code.

* Validation testing :

→ The Integrated s/w is tested based on requirements to ensure that the desired product is obtained

→ In validation testing the main focus is to uncover errors in

→ system i/p / o/p
→ system functions & Info" data
↔ " Interfaces with external Parts
→ User Interface
→ system behavior & performance.

→ S/W validation can be performed through a series of black box test
→ After performing the validation tests there exists 2 conditions
1) The fun^n or performance characteristics are according to the specifications & are accepted.
2) The requirement specifications are derived & the deficiency list is created. The deficiencies then can be resolved by establishing the Proper comm with customer.
→ Finally the validation testing a review is taken to ensure that all the elements of the S/W configuration are developed as per requirement This review is called configuration review or audit.

* System Testing:
→ The system test is a series of tests conducted to fully the computer based system.
→ various tests of system tests are
1) Recovery testing
2) Security "
3) Stress "
4) Performance "
→ The main focus of such testing is to test
- System fun^n's & performance
- " reliability & recoverability (recovery test)
- " installation (Install^n test)
- " behavior in the special conditions (Stress test).
- " user operations (Acceptance test / Alpha test) → S/W works correctly in user work environment
- H/W & S/W integration & collaboration
- Integration of external S/W & the system.

1) Recovery testing: It is intended to check the system's ability to recover from failures.
2) Security Testing: It verifies that system Protection mechanism prevent improper penetration or data alteration. It also verifies protection mechanism built into the system
3) Stress Testing: In stress testing the system is executed in manner that demands resources in abnormal quantity, frequency or volume.
4) Performance Testing: It evaluates the run time performance of S/W especially real time S/W.

* Art of Debugging :-

→ Debugging is a process of removal of a defect. It occurs as a consequence of successful testing

→ Debugging process starts with execution of test cases.

→ The actual test results are compared with the expected results.

→ The debugging process attempts to find the lack of correspondence b/w actual & expected results.

→ The suspected causes are identified & additional tests or regression tests are performed to make the system to work as per requirement.

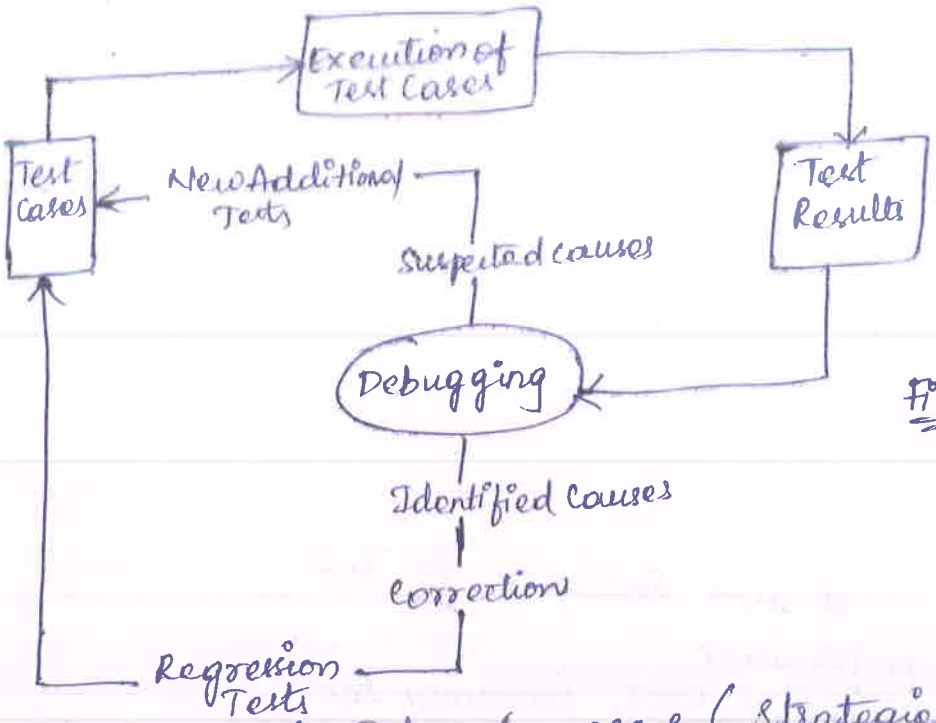


Fig:- Debugging Process.

Common Approaches in Debugging are :- (Strategies)

1) Brute force Method :- The memory dumps & run-time traces are examined & program with write statements is loaded to obtain clues to error causes

→ In this method "Let computes find the error" approach is used.

2) Backtracking Method :- It is applicable to small programs

→ In this method, the source code is examined by looking backward from symptom to potential causes of errors.

3) Cause Elimination Method :- This method uses binary partitioning to reduce the number of locations where errors can exist

4) Automated Debugging :-

* Product Metrics :-

→ s/w development activity is comprised of 5 important phases those are : Analysis, design, coding, testing & Maintenance.

→ The Product metrics defines the measure over all these phases.

* Software quality.

s/w quality can be defined as "the conformance to explicitly stated functional & performance requirements, explicitly documented development standards, & implicit characteristics that are expected of all professionally developed s/w".

→ Factors that affect s/w quality can be categorized into 2 broad groups

1) Factors that can be directly measured (e.g:- defects uncovered during testing)
2) " " " " measured only indirectly (eg:- usability or maintainability)

*1) Mc Calls Quality factors :-

→ Mc Cally Richards & Walters have proposed that s/w quality factors can be classified into 3 categories which can be illustrated by Mc Calls triangular metho model for quality factor

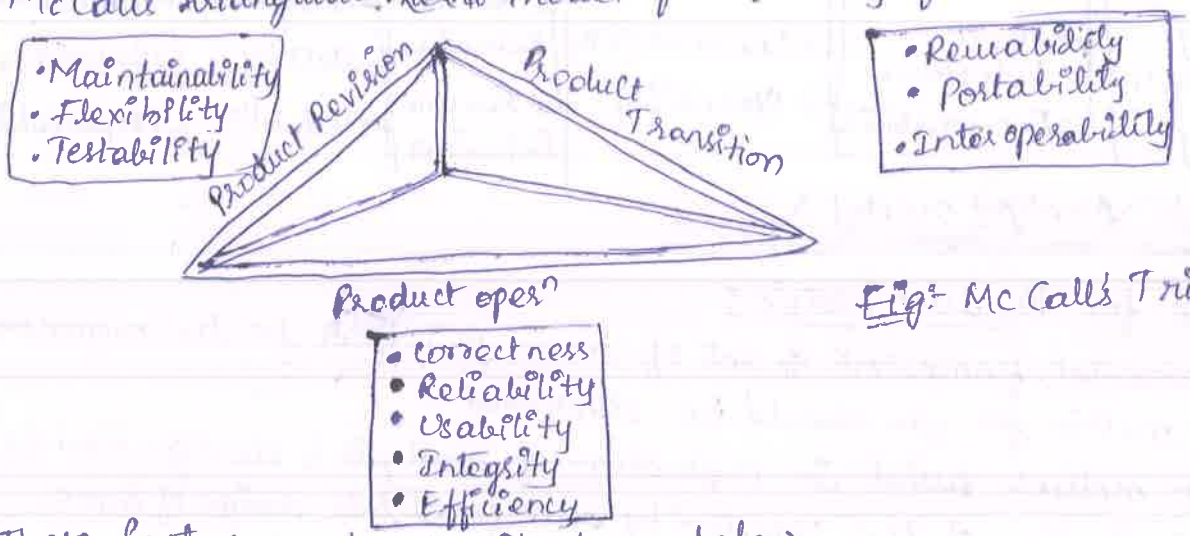


Fig:- Mc Calls Triangular model

These factors can be described as below

Operational Characteristics

1. correctness → The ability to fulfil the specification & customer requirements.
2. Reliability → The degree by which the s/w should work as per the requirement precision.
3. Usability → The ability to prepare the valid i/p & interpret the correct o/p
4. Efficiency → The measure of computing resources & time required by the program to perform.
5. Integrity → This is the controlling ability by which unauthorized access to the system can be Prevented

Adaptability to new environment:
6. Maintainability → The ability required to locate or fix the bugs in the s/w
7. Flexibility → The extent by which it is allowed to make certain modifications in the program
8. Testability → The ability to check that the fun is working as per the requirement

Ability to undergo change:
9. Reusability → The ability by which the particular component in the s/w can be reused by some other program in that s/w
10. Portability → The ability of the s/w to work properly even if the environment gets changed (i.e., change in h/w or s/w)
11. Interoperability → The ability of the system to work with other systems

2) ISO 9126 Quality factors:

The ISO 9126 has developed following attributes for s/w quality

ISO 9126 Quality factor

- (s/w satisfies) → Functionality
  - → Suitability
  - → Accuracy
  - → Compliance
  - → Security
- (s/w available) → Reliability
  - → Maturity
  - → Fault Tolerance
  - → Recoverability
- (useful new s/w) → Usability
  - → Understandability
  - → Learnability
  - → Operability
- (Pgm to perform) → Efficiency
  - → Time Behaviour
  - → Resource Behaviour
- (Locate or fix bugs in s/w) → Maintainability
  - → Analyzability
  - → changability
  - → Stability
  - → Testability
- (work properly if environment changed) → Portability
  - → Adaptability
  - → Installability
  - → Conformance
  - → Replaceability

* Metrics for Analysis model:

* Framework for Product Metric:

→ A fundamental framework & set of basic principles for the measurement of product metrics for s/w should be established

→ Product metrics assist in evaluation of analysis & design models, given an indication of the complexity and facilitate design of more effective testing

→ Steps for an effective measurement Process are

1) Formulation → derivation of s/w measures & metrics
2) Collection → It accumulate data required to derive the metrics
3) Analysis → computation of metrics
4) Interpretation → evaluation " "
5) Feedback → Derived after Interpretation

* Metric: It is the quantitative measure of the degree to which a system, component or proces posses a given attribute.

→ It relates individual measures in some way.

* Metrics for Analysis model :-

→ Metrics for the analysis model are useful in estimating the project.

→ In order to determine the metrics in analysis model "size" of the s/w is used as a measure. Let us discuss the metrics based on fun^n point proposed By Albrecht in 1979 for IBM.

* Function Point model :- It is based on functionality of delivered appl^n

→ Fun^n points are derived using

1. Countable measures of the s/w requirements Domain
2. Assessments of the s/w complexity.

* To calculate fun^n Point :-

1. Number of external (users) i/p's (EIS) → Each user i/p which provides distinct appl^n data to the s/w is counted.
2. Number of external (users) o/p's (EOS) → Each user o/p provides appl^n data to user is counted eg:- screens, reports, error messages
3. Number of External (User) inquiries (EQS) → An on-line i/p that results in the generation of some Immediate s/w response in the form of an o/p.
4. Number of internal logical files (ILFS) → Each logical master file i.e., a logical grouping of data that may be part of a DB or a separate file.
5. Number of external Interface files (EIFS) → All machine-readable interfaces that are used to transmit infor^n to another system are counted.

* Each Parameter is classified as simple, Average, or complex & Weights are assigned as follows

Domain characteristics	count	simple	avg	complex
EIS	3	4		6
EOS	4	5		7
EQS	3	4		6
ILFS	7	10		15
EIFS	5	7		10

→ Function Point (FP) = count total $\times (0.65 + (0.01 \times \text{sum}(Fi)))$

→ Once the fun^n point is calculated then we can compute various measures as follows

- Productivity = FP / Person-month
- Quality = number of faults / FP
- Cost = $ / FP
- Documentation = Pages of Documentation / FP

* Metrics for Design model :

→ These metrics quantify design attributes in a manner that allows a s/w engineer to assess design quality.

Metrics include:

1) Architectural metrics :- provide an indication of the quality of the architectural design

2) Component-level metrics :- Measure the complexity of s/w components & other characteristics that have a bearing on quality.

3) Interface design metrics :- focus primarily on usability.

4) Specialized OO Design metrics :- measure characteristics of classes & their comm^n & collaboration characteristics.

* Metrics for source code :- These metrics measure the source code and can be used to assess its complexity, maintainability & testability among other characteristics :

1) Halstead metrics :- controversial but nonetheless fascinating, these metrics provide unique measures of a computer program

2) Complexity metrics :- measure the logical complexity of source code (can also be considered to be component-level design metrics)

3) Length metrics :- provide an indication of the size of s/w.

* Metrics for Testing :

→ These metrics assist in the design of effective test cases & evaluate the efficacy of testing :

1) Statement & branch coverage metrics :-
lead to the design of test cases that provide program coverage

2) Defect-related Metrics :- focus on bugs found, rather than on the tests themselves.

3) Testing effectiveness :- provide a real-time indication of the effectiveness of tests that have been conducted.

4) In-Process metrics :- process related metrics that can be determined as testing is conducted.

* Metrics for Maintenance :-

→ All the s/w metrics can be used for the development of new s/w & the maintenance of existing s/w.

→ However, metrics designed explicitly for maintenance activities have been proposed.

→ IEEE std. 982.1-1988 [IEEE94] suggests a s/w maturity index (SMI) that provides an indication of the stability of a s/w product

→ The SMI (s/w maturity index) is computed in the following manner.

$$SMI = [M_T - (F_a + F_c + F_d)] / M_T$$

M_T = num. of modules in the current release

F_c = " " " " " " " that have been changed.

F_a = " " " " " " " that have been added.

F_d = " " " from the preceding release that were deleted in the current version.

→ The SMI approaches 1.0, the product begins to stabilize

→ SMI may also be used as a metric for planning s/w maintenance activities.

* Metrics for Process & Products :-

→ Process Metrics :- They are collected across all projects & over long periods of time (Set of Process indicators that are used to improve the s/w Processes)

→ Their intent is to provide a set of process indicators that lead to long-term s/w process improvement

→ Product Metrics :- It enable a s/w project manager to

1) assess the status of an ongoing project
2) track potential risks.
3) uncover problem areas before they go "critical".
4) adjust work flow or tasks &
5) evaluate the project team's ability to control quality of s/w work products.

→ Measures that are collected by a Project team & converted into metrics for use during a project can also be transmitted to those with responsibility for s/w process improvement.

→ For this reason, many of same metrics are used in both the process & project domain.

* Process Metrics and s/w Process Improvement :-

→ The only rational way to improve any process is to measure specific attributes of the process, develop a set of meaningful metrics based on these attributes & then use the metrics to provide indicators that will lead to a strategy for improvement

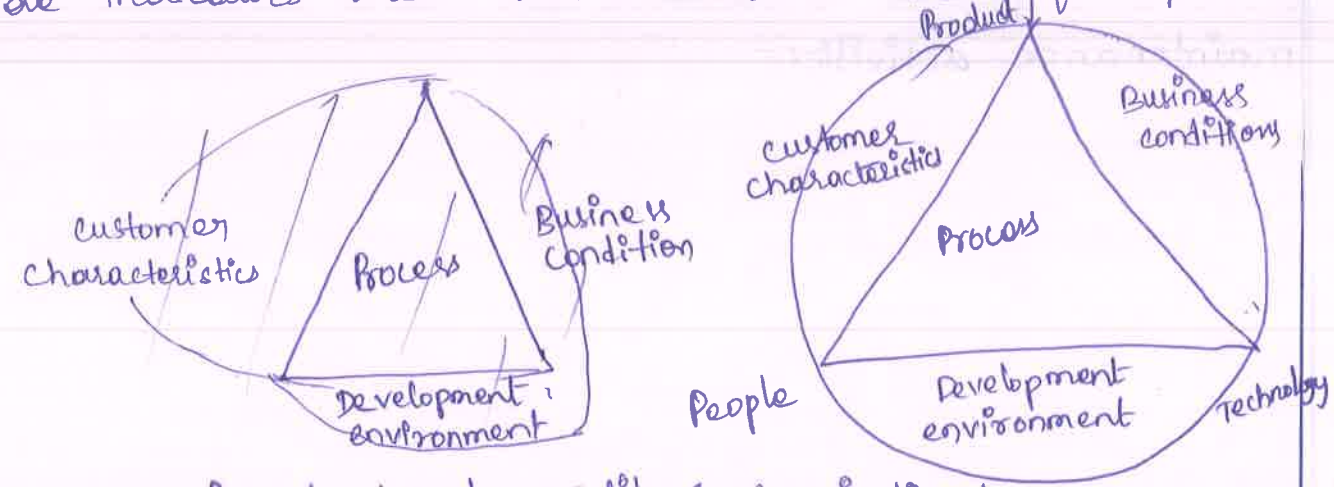


Fig :- Determinants for s/w quality & organizational effectiveness

→ In making improvements to any s/w system, there are 3 basic quality factors to consider : Product, people & technology.
→ These three are the major determinants of s/w cost, schedule, productivity & quality.
→ The factor people includes hiring the best people you can find, motivating them to do the best job, & training them on the skills needed to perform their jobs effectively.
→ The Technology factor includes acquiring & installing tools that help automate. (eg.. Java, Visual C++, Oracle)
→ The complexity of the factor product has great impact on quality & team performance.
Eg:- If an organiz^n makes use of new testing tool (technology) then the staff (people) must be trained to work with this tool.
→ This process Δ recides within the circle. This circle specifies the environmental conditions such as
- Customer characteristics (comm^n & collaboration b/w user & developer)
- Business conditions (Organizational policies, Business rules)
- Development environment (use of new Technologies, use of automated tools)

* s/w Measurement :-

→ The measurement of s/w can be classified into 2 categories

s/w measurement →
- Direct measure (measures cost & effort applied lines of code produced, execution, speed & defects)
- Indirect Measure (measures functionality, quality, reliability, efficiency, maintainability, complexity).

Let us discuss the metrics based on these approaches :-
1) Size oriented metrics :- It is derived by considering the size of s/w that has been produced
→ The organiz^n builds a simple record of size measure for the s/w projects. It is built on past experiences of organ^n's
→ It is a direct measure of s/w.

Project	LOC	Effort	cost($)	Doc(pgs)	Errors	Defects	People
ABC	10,000	20	170	400	100	12	4
PQR	20,000	60	300	1000	129	32	6
XYZ	35,000	65	522	1290	280	87	7
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮

Table :- Size Measure.

→ A simple set of size measure that can be developed is as given below

- Size = KLOC (Kilo lines of code)
- Effort = Person/month
- Productivity = KLOC/person-month
- Quality = Num. of faults/KLOC
- cost = $/KLOC
- Documentation = Pages of Documentation/KLOC.

→ The size measure is based on the lines of code computation
→ The lines of code is defined as one line of text in a source file
→ While counting the lines of code the simplest standard is

- Don't count black lines
- " " comments
- count everything else

Advantages :- 1) Artifacts of S/W development which is easily counted
2) Many existing methods use LOC as a key input.
3) A large body of literature & data based on LOC already exists.

Disadvantages :- 1) This measure is dependent upon programming language
2) This method is well-designed but shorter program may get suffered
3) It doesn't accomodate non procedural languages.
4) In early stage of development it is difficult to estimate LOC

2) Fun^n oriented Metrics :-

→ The fun^n point model is based on functionality of the delivered appl^n
→ These are generally independent of programming language used.
→ This method is developed by Albrecht in 1979 for IBM
→ Fun^n points are derived using
1) countable measure of S/W requirements Domain.
2) Assessments of the S/W complexity.

3) Object Oriented Metrics:

Lorenz & Kidd have proposed set of metrics for Object oriented projects

Metrics	Description
Num. of scenario scripts	→ The number scenarios can be typically obtained by designing the use cases for the system. → It basically represents the user interaction with appl^n → The total num. of scenario is dependent upon size of appl^n & num. of test cases
Num. of Key classes	In any OO Design classes are the logical entities of the system
Num. of support classes	The support classes are logical entities to support the key classes.
Average number of support class per key class	There are more num. of support class per key class for the appl^n with GUI than appl^n without GUI
Number of subsystems	Subsystem means aggregation of classes. To schedule the project systematically it is necessary to identify all the subsystem of the system being developed.

* Metrics for s/w quality:-

→ The goal of s/w engineering is to produce high quality s/w.
→ To achieve this goal s/w engineers use effective methods along with modern tools while developing the s/w.
→ Basically the quality of s/w depends upon –

- Requirements that describe the problem
- The design method used to produce s/w.
- The code that leads to executable program.
- And the tests that are carried out in order to uncover the errors from the s/w.

→ The project manager evaluates the quality of s/w project using following factors -

- Errors & defects in the s/w.
- Quality metrics collected by each s/w engineer who is involved in the s/w development process.

→ Typically following metrics are used for s/w assessing the s/w quality

1) work product errors per fun^n
2) Errors found in the per review hour.
3) " " " " Testing.

→ This error data is useful in computing the defect removal efficiency (DRE)

* Defect Removal Efficiency (DRE) :-

→ While developing the s/w project many work products such as SRS, design Document, source code are being created.

→ Along with these work products many errors many get generated

→ The DRE can be defined as

$$DRE = E/(E+D) \qquad E = Error,\ D = Defect$$

→ During error tracking activity following metrics are computed

1. Errors per requirements specification page : denoted by E_{req}.
2. " " component - Design level : Denoted by E_{design}
3. " " " - code level : " " E_{code}
4. DRE - requirement analysis.
5. DRE - Architectural Design -
6. " - component level "
7. DRE - coding.

→ These error tracking metrics can also be used for better target review & testing resources.

2) Measuring Quality :- Following are the measure of the s/w quality.

1. correctness :- It is a degree to which the s/w produces the desired functionality

correctness = Defects per KLOC

2. Integrity :- It is basically an ability of the system to withstand against attacks

Integrity = Σ (1-threat) × (1-security).

3. Usability :- It means user friendliness of the system (Usefulness of system)
4. Maintainability :- It is an ability of system to accomodate the corrections made after encounting errors, adapting the environment changes & adapt the changes in system in order to satisfy user.

UNIT-5

Risk Management:-Reactive vs Proactive Risk strategies, s/w risks, Risk Identification, Risk Projection, Risk refinement, RMMM, RMMM Plan.
Quality Management: Quality concepts, s/w quality assurance, s/w Reviews, Formal technical views, Statistical s/w quality assurance, s/w reliability, The ISO 9000 quality standards.

* Risk Management:-

→ It is a popular Buzzword in today's business world, but many s/w professionals have only a vague (unclear character) idea of what it means & how it fits into their job description.
→ It is a fairly simple concept, it refers to the process of making decisions based on an evaluation of the factors that threats to the business.

* Reactive vs Proactive Risk strategies:-

→ Reactive & Proactive risk strategies are the approaches used for managing the risks.

* Reactive risk strategy:-

→ It is a strategy in which when project gets into trouble then only corrective action is taken
→ But when such risks can't be managed & new risks come up one after the other, the s/w team files into action in an attempt to correct Problem rapidly. These are called "fire-fighting activities.
→ Resources are utilized to manage such risks. And if still the risks don't get managed then project is in damager.
→ This is an older approach of risk management

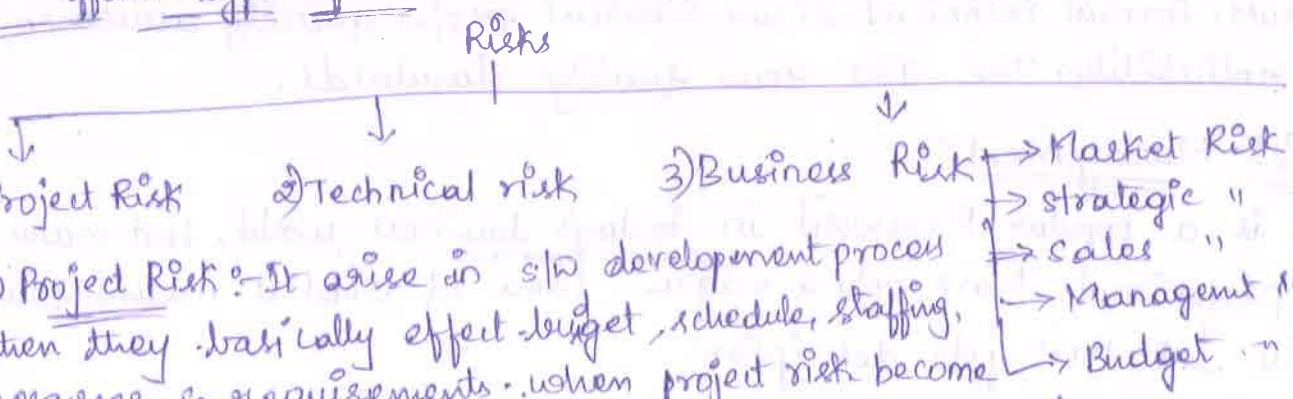
* Proactive risk strategy:- It is a strategy begins before the technical activity by considering the probable risk.
→ In this strategy potential risks are identified first then their probability & impact is analyzed.
→ The objective of this strategy is to avoid risks.
→ Such risks are specified according to their priorities.
→ Finally the s/w team prepare a plan for managing these risks
→ It is an Intelligent strategy for risk management & now a day it is used more in IT industries.

* S/W Risks :-

→ There are 2 characteristics of risks.

1. The risk may or may not happen. It shows uncertainty of risks.
2. When risks occur, unwanted consequences or losses will occur.

* Different types of risk :-



1) Project Risk :- It arise in s/w development process then they basically effect budget, schedule, staffing, resources & requirements. When project risk become severe then the total cost of projects gets increased.

2) Technical Risks :- It affects quality & timelines of project. When it becomes reality then potential design, Impln, Interface, Verification & maintenance problems get created. It occur when problem becomes harder to solve

3) Business Risk :- When feasibility of s/w product is in suspect then business risks occur. They are categorized as

(i) Market Risk :- No market for the product

(ii) Strategic Risk :- When a product is built & not following the company's business policies then such a product brings strategic risk.

(iii) Sales Risk :- When a product is built but how to sell is not clear. then such a situation brings sales risk.

(iv) Management Risk :- When senior management or staff leaves organ, then management risk occurs

(v) Budget Risk :- Losing the overall budget of project is called Budget Risk.

→ Another categorisation of risk proposed by charatte is known Risks are those risk that are identified after evaluating Project Plan

→ Known risks → 1) Predictable Risks (these risks can be identified in advance based on past project experience) Eg:- Experience & skilled staff leaving

2) Unpredictable Risks (There risks that can't be guessed earlier) eg:- Government Policies may effect the business people.

** Risk identification:- It can be defined as the efforts taken to specify threats to the project Plan. It can be done by identifying the known & predictable risks.

→ It is on Based on 2 approaches.

1. Generic Risk identification:- It includes potential threat identification having support to s/w project.

2. Product-specific ~~task~~ risk identification:- It includes product specific threat identification by understanding people, technology & working environment in which the product gets built.

→ Normally the risk identification is done by the project manager who follows following steps.

Step 1:- Preparation of Risk item check list

→ The risk items can be identified using following known & predictable components.

(i) Product size:- risk item based on overall size of s/w product is identified

(ii) Business impact:- Risk items related to market place.

(iii) customer characteristics:- Risks associated with customer-developer comm

(iv) Process definition:- Risks that get raised with definition of s/w process.

(v) Development environment:- Risks associated with technology & tool being used for developing the product.

(vi) Staff size & experience:- Once the technology & tool related risks items are identified it is essential to identify the risk associated with highly experienced & skilled staff who will do the development.

(vii) Technology to be built:- complexity of the system should be understood & related risk items need to be identified.

→ After preparing this risk item checklist a questionaire is prepared. These set of questions should be answered & based on these answers the impact or seriousness of particular risk item can be judged.

Step 2:- Creating ~~risk~~ components & drivers list

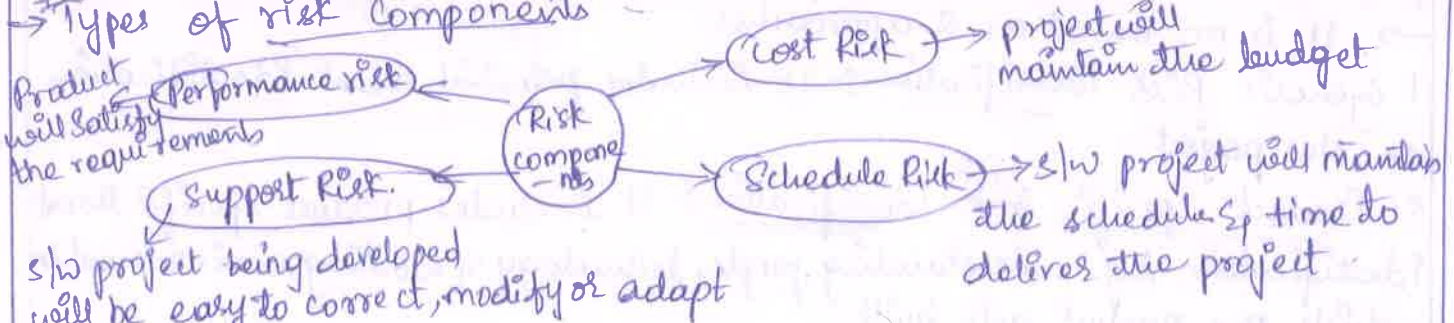
→ The set of risk components & drivers list is prepared along with their probability of occurence.

→ Let us understand which are the risk components & Drivers.

1) Risk components & Drivers :-

→ U.S Air force has written a guideline for risk identification which is based on identification of risk component & risk drivers.

→ Types of risk Components



Risk Drivers (Analyse the impact of risk)

↓ Negligible ↓ Marginal ↓ Critical ↓ Catastrophic (causing sudden great damage or suffering)

2) How to Assess Overall Project Risk ?

→ The best Approach is to prepare set of questions that can be answered by project managers in order to asses the overall project risks

* Risk Projection :- (Also called Risk estimation)

→ There are 2 ways by which risk can be rated

↓ Probability that the risk is real ↓ consequences of Problems associated with the risk

→ The project Planner, technical staff, project manager performs following steps to perform following steps for risk projection

- Establish a scale that indicates the probability of risk being real.
- Enlist the consequences of the risk
- Estimate the impact of the risk on the project & product.
- Maintain the overall accuracy of the risk projection.

→ These steps help to prioritize the risks. Once the risks are prioritized then it becomes easy to allocate the resources for handling them

1) Building Risk table : It is the simplest & most commonly used technique adopted by project managers in order to project the risks.

→ The sample risk ~~data~~ table is as given below

Risk Table

Risk	Category	Probability	Impact	RMMM
Is that skilled staff available	staff	50 %.	Catastrophic	
Is that the team size sufficient	"	62 %.	critical	
Have the staff received sufficient training	"	25 %.	Marginal	
Will technology meet the expectations	Technology	30 %.	Critical	
Is the s/w mgmt tool available	Environment	40 %.	Negligible	
How much amt of reused s/w is required	Project size	60 %.	Marginal	
Will customer change the requirement?	customer	20 %.	Critical	

* While building the risk table

→ The project team first of all enlists all probable risks with the help of risk item check list.

→ Each risk is then categorized. As we know various categories of risk can be a) Project size b) Technology c) customer d) staff e) business f) developing environment.

2) After building table it is then sorted by probability & impact.

3) Then the project manager goes through this 1st order prioritized risk table & draws a horizontal line at some point in the table.

4) The risk table cut off line is again sorted & second order priority is applied on table.

5) The risk table above the cut-off line is having the risks with high probability & high impact. & such risks should occupy the significant amount of management time.

6) All the risks that lie above the cut off line should be managed.

2) Assessing Risk Impact:

→ while Assessing the risk impact 3 factors are considered

- Nature of risk → type of risk
- Scope of risk → severity "
- Timing at which risk occurs.

Risk Exposure = probability of occurence of risk x cost

* Risk Refinement: It is a process of specifying the risk in more detail.

→ The risk refinement can be represented using CTC format suggested by D. P. Gluch

→ CTC stands for condition-transition-consequence.

→ The condition is 1st stated & then based on this conditions sub conditions can be derrived.

→ Then determine the effects of these sub conditions in order to refine the risk.

→ This refinement helps in expaning the underlaying task

→ This approach makes it easier for the project manager to analyse the risk in greater detail.

* RMMM:- It stands for Risk mitigation, monitoring & management:

→ There are 3 issues in strategy for handling the risk is

1. Risk Avoidance 2. Risk monitoring 3. Risk management

* Risk Mitigation:- It means preventing the risks to occur (risk avoidance)

→ Following are the steps to be taken for mitigating the risks.

1) communicate to concerned staff to find probable risk.
2) Find out & eliminate all those causes that can create risk before the project starts.
3) Develop a policy in an org^n which will help to continue the project (staff)
4) Everybody in the project team should be acquainted (together).
5) Maintain the corresponding documents in timely manner.
6) conduct timely reviews in order to speed up the work.
7) For conducting every critical activity during s/w development provide the additional staff if required.

* Risk Monitoring:- In this process following things must be maintained by the project manager.

1) The approach or the behaviour of the team members as pressure of project varies

2) The degree in which the team performs with the spirit of "team work"
3) The type of cooperation among the team members
4) The types of problems that are occuring.
5) Availability of jobs within & outside the organization.

* **Risk Management:**
→ Project manager performs this task when risk becomes reality.
→ If project manager is succesful in applying the project mitigation effectively then it becomes very much easy to manage the risks.

* **RMMM Plan:** It is a document in which all the risk analysis activities are described.
→ sometimes project manager includes this document as a part of overall project plan.
→ sometimes specific RMMM Plan is not created, however each risk can be described individually using risk info sheet.
→ Typical template for RMMM Plan or Risk info sheet can be

Risk Information sheet

Project name <enter name of project for which risks can be identified>

Risk Id <#>	Date <Date at which risk is identified>	Probability <risk probability>	Impact <low/medium/high>
Origin <the person who has identified the risk>		Assigned to <who is responsible for mitigating the risk>	

Description <Description of risk identified>

Refinement / context <associated info for risk refinement>

Mitigation / Monitoring <enter the mitigation/monitoring steps taken>

Trigger / contingency plan <if risk mitigation fails then the plan for handling the risk>

Status <Running status that provides a history of what is being done for the risk & changes in the risk. Include the date the status entry was made>

Approval <name & signature of person approving closure>	closing date <Date>

→ Risk infor" sheet can be maintained by DB systems
→ After documenting the risks using either RMMM Plan or Risk infor" sheet the risk mitigation, monitoring & analysis activities are stopped.

* Quality Management :-

→ It is an umbrella activity that is applied throughout the s/w process
→ Quality management encompasses
1) a s/w quality assurance (SQA) process.
2) Specific quality assurance & quality control tasks (including formal technical reviews & a multitiered testing strategy)
3) effective s/w engineering practice. (methods & tools)
4) Control of all s/w work products & the changes made to them
5) a procedure to ensure compliance with s/w development standards
6) Measurement & reporting mechanisms.

* Quality Concepts :-

→ Variation control is the heart of quality control
→ A manufacturer wants to minimize the variation among the products that are produced, even when doing something relatively simple like duplicating DVDs.
→ Surely, this cannot be a problem - duplicating DVDs is a trivial manufacturing operation, & we can guarantee that exact duplicates of the s/w's are always created.
→ s/w quality measures how well s/w is designed (quality of design) & how well the s/w conforms to that design (quality of conformance).

* Quality :- The American Heritage Dictionary defines quality as "a characteristic or attribute of something".
→ Quality of design refers to the characteristics that designers specify for an item.
→ Quality of conformance is the degree to which the design specifications are followed during manufacturing.

* Quality control :-

→ Quality control involves the series of inspections, reviews & tests used throughout the s/w process to ensure ~~does~~ each work product meets the requirements placed upon it.

→ Quality control includes a feedback loop to the process that created the work product.

* Quality Assurance :-

→ Quality Assurance consists of a set of auditing & reporting fun"s that assess the effectiveness & completeness of quality control Activities

→ The goal of quality assurance is to provide management with the data necessary to be informed about product quality, thereby gaining insight & confidence that product quality is meeting its goals.

* Cost of Quality :-

→ cost of quality studies are conducted to provide a baseline for the current cost of quality, identify opportunities for reducing the cost of quality & provide a normalized basis of comparison

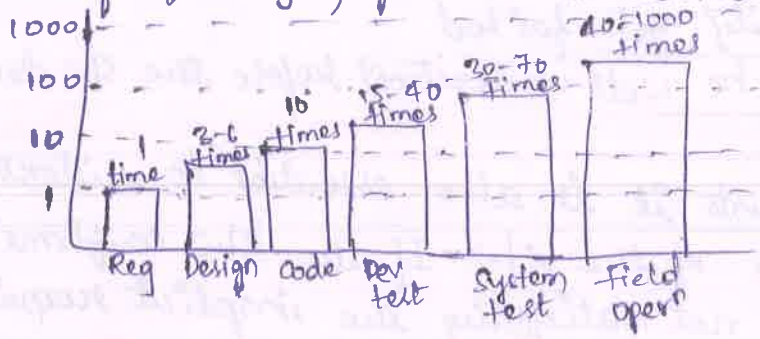


Fig: Relative cost of correcting an error.

* S/W Quality Assurance :- (SQA)

→ It is a set of Activities for ensuring quality in SE processes.

→ It is an ongoing process within the SDLC that routinely checks the developed s/w to ensure it meets the desired quality measures.

→ Cost of quality can be defined as the total cost required to obtain the quality in the product & to conduct the quality related activities

→ The cost of quality has various components such as

1) Prevention cost :- This is the cost of quality required for conducting quality Planning, formal technical reviews, test equipments & training.

2) Appraisal cost :- This is the cost of quality required for gaining the insight into the product. It includes the cost required for in-process & inter process. inspection, maintenance & testing

3) Failure cost :- Failure cost means the cost required to remove the defects in the s/w product before delivering it to customer

↳ 2 types of failure costs → [Internal Failure cost, External "]

a) Internal failure cost :-

It is nothing but the cost of defects occured in the product before delivering it to customer

eg:- repair in networking, repairing of comm^n n/w.

Fig Cost of quality

b) External failure cost :-

It is the cost of defects occured in the product after delivering it to the customer

eg:- product repair/replace, complaint processing, warranty work

* SQA (s/w quality Assurance)

→ quality is the conformance to functional & non-functional requirements of the product

→ 3 reasons why s/w quality gets failed

1) s/w requirements must be well understood before the s/w development process begins.

2) Finally to explicit requirements it is also essential to understand the implicit requirements of the s/w. If the s/w confirms the explicit requirements but not satisfying the implicit requirements then surely quality of s/w being developed is poor.

3) The set of development criteria has to be decided in order to specify the standards of the product

* s/w QA Activities

s/w QA tasks are associated with

↓ s/w engineers (Responsible for developing the Product)

↓ SQA group (Responsible for performing quality Assurance Planning, oversight, record keeping, analysis & reporting)

Let us list out the SQA Activities conducted by SQA group.

1) Prepare the SQA plan for a project
2) Participates in the development of the project's s/w process description
3) Reviews s/w engineering activities to verify compliance with the defined s/w process.
4) Audits designated s/w work products to verify compliance with those defined as part of the s/w process.
5) Ensure the deviations in s/w work. These work products are documented & handled according to documented procedure.
6) Records any non compliance & reports to senior management.

* S/W Reviews

→ s/w Reviews are filter to SE process.
→ such reviews are applied at various points during s/w development life cycle
→ The objective of s/w reviews is to uncover errors & defects that can be removed.
→ There are 3 different ways by which s/w review can be conducted
1) Informal meeting: An informal meeting can be conducted outside the working environment & an informal discussion of technical issues can be held.
2) Formal presentations: It can be conducted for customer, management & technical staff
3) Formal technical reviews (FTR) sometimes called as walkthrough or an inspection is the most effective way of s/w review. This helps a lot for uncovering the s/w errors & to improve the quality of s/w.
→ Benefit of FTR is that errors can be discovered in early stage before they become defects in the next release of s/w

* Defect Amplification model: It can be used to illustrate the generation & detection of errors during the steps in the s/w engineering process.

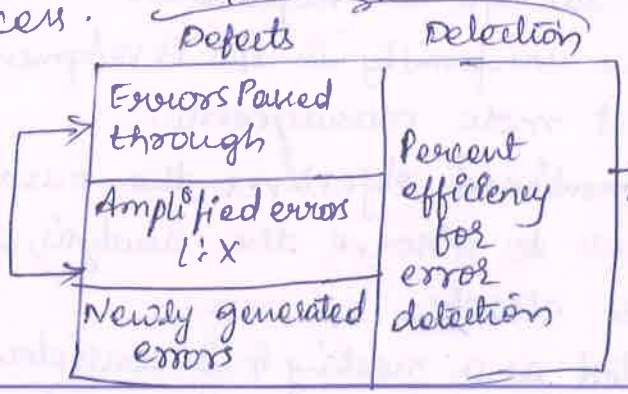
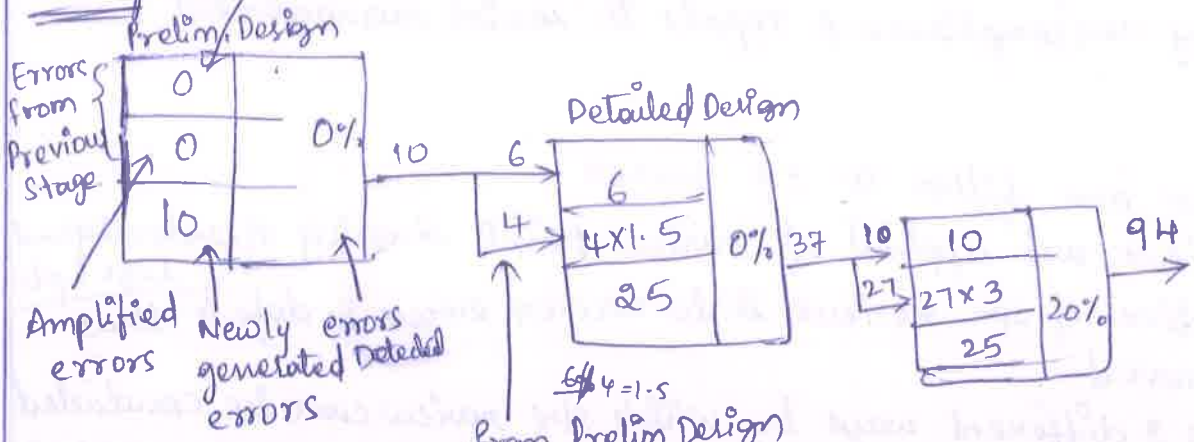


Fig:- Defect Ampln model

→ In the defect Ampl" model the outer box indicates the s/w developed stage.
→ This box is Partitioned into 2 categories, defects & detection of errors.
→ Under the defect column we list out numbers of errors.
→ Errors can be those coming from previous stage or it could be newly generated errors.
→ The previous errors can be those that are passed through errors as well as the amplified errors.

Example:- Errors Passed through



* Formal Technical Reviews :- (FTR)

→ It is a SQA activity Performed by s/w engineer.

Objectivies of FTR:-

1) FTR is useful to uncover errors in logic, fun" & impl" for any represen" of the s/w
2) The purpose of FTR is to ensure that s/w meets specified requirements
3) It also ensures that the s/w is represented according to predefined standards
4) It helps to review the uniformity in s/w Development process.
5) It makes the project more manageable.

→ Besides the above mentioned objectives the purpose of FTR is to enable junior engineers to observe the analysis, design, coding & testing approaches more closely.
→ Each FTR is conducted as a meeting & is considered successful only if it is properly planned, controlled & attended.

* The Review Meeting:- Every review meeting should be conducted by considering the following constraints

1) Involvement of people:- B/w 3 & 5 people should be involved in the review
2) Advance Preparation:- It should occur but it should be very short i.e., at the most 2 hours of work for each person can be spent in this preparation
3) Short duration:- The duration of the review meeting should be < 2 hrs.

* Review Reporting and Record keeping:-

Review Reporting & record Keeping → FTR summary report
→ Review issues list

→ During the FTR, the reviewer actively records all issues that have been raised.
→ At the end of meeting these all raised issues are consolidated & review issues list is prepared.
→ Finally a FTR summary report is produced.

* Review Guidelines:-

→ Guidelines for the conducting of formal technical reviews must be established in advance.
→ This guideline must be distributed to all reviewers, agreed upon & then followed. For example - Guideline for review may include following things.

1) Concentrate on work product only. That means review the product, not the Producer.
2) Set an agenda of review & maintain it
3) When certain issues are raised then debate or arguments should be limited
4) Find out problem areas, but don't attempt to solve every problem noted
5) Take written notes (it is for the recorder)
6) Limit the num. of participants & insist upon advance preparation
7) Develop a checklist for each product that is likely to be reviewed
8) Allocate resources & time schedule for FTRs in order to maintain the schedule.
9) Conduct meaningful trainings for all reviewers in order to make the reviews effective.
10) Review earlier reviews which serves as the base for the current review being conducted.

* Statistical s/w Quality Assurance :-

→ It is a simple concept which represents that changes in the s/w can be made in order to improve those elements of the process that introduce error.

→ Statistical s/w QA can be performed with the help of following steps.

1. Collect the information about s/w defects. Categorize them
2. Make an attempt to trace each defect to its root cause.
3. Isolate the vital few causes of the major source of all errors by using the 80-20 principle (Known as Pareto Principle).

→ This principle is "80% of the defects can be traced to 20% of all possible causes".

↳ Then move to correct the problems that have caused the defects.

Example:- Consider that a well known s/w firm has collected some information about that defects that are occuring in s/w product

Collection of defects occur in 2 stages
- Defects occured during s/w Development Process
- Defects occured after delivering the products to end-user

The causes of defects are tabularized as below.

Causes of defects	Abbreviations
Error in Data representation	EDR
Error in Design logic	EDL
Unclear human computer Interface	HCI
Incomplete &/or erroneous Specifications	IES
Intentional Deviations from specifications	IDS
Inconsistent Component Interface	ICI
Incomplete &/or erroneous testing	IET
Incomplete or inaccurate Documentation	IID
Error in programming language translation of Design	PLT
violation of programming standards	VPS
Misinterpretation of customer comm?	MCC
Miscellaneous	MIS

→ Suppose that some sample representative Data of defect causes is collected then one can built a stastical SQA from it

→ The sample Data about the defect causes is as given below.

Causes of defect	Number of errors	Percentage
IES	200	21
MCC	158	17
IDS	48	5
VPS	20	2
EDR	128	14
ICI	60	6
EDL	45	5
IET	94	10
MD	35	4
DLT	60	6
HCI	30	3
MIS	58	6
Total	936	100%

Table Sample Data Collection for Stastical SQA

→ From above table it is clear that the major defect causes is IES+MCC+EDR which generates 21+17+14=52% total errors

→ Thus if IES+MCC+EDR = 52% of defect cause is removed. This will ultimately help in improving S/w quality.

Six Sigma :- It is originated at Motorola in the early 1980s

→ It is widely used stastical SQA strategy. It is a business driven approach to process improvement reduced costs & ↑ profit.

→ The word "Six sigma" is derived from 6 Standard deviations — 3.4 defects per million

Define → Measure → Analyze → Improve → Control

Fig :- 6 Sigma framework.

→ There are 3 core steps in 6 sigma method

Define :- The customer requirements, project goals & deliverables are defined by communicating the customers

Measure :- The existing process & its o/p is measured in order to determine current quality performance.

Analyze :- In this phase defect metrics are analyzed in order to determine the few causes.
→ If an improvement is needed to an existing S/w then there are additional 2 methods in 6 Sigma -
Improve :- By eliminating the root causes of defects the process can be improved.
Control :- The process can be controlled in such a way that the causes of defects can't be introduced.
→ These steps can sometimes be referred as DMAIC.
→ For a newly Developing S/w, some organizations are suggesting following 2 alternating steps -
Design - In this step avoid root causes of defects & meet the customer requirements.
Verify - To verify the process, avoid defects & meet customer requirements.
→ These steps can sometimes be referred as DMADV

* S/w Reliability :- It is defined as the probability of failure free operation of a computer program in a specified environment for a specified time.
→ The S/w reliability can be measured, directed & estimated.

* Measure of Reliability & Availability :- Normally there are 2 measures of S/w Reliability.
1) MTBF (Mean time B/w failure) :- It is a simple measure of S/w reliability which can be calculated as

$$MTBF = MTTF + MTTR$$

MTTF → means time to failure
MTTR → " " " repair

→ Many S/w researches feel that MTBF is more useful measure of S/w reliability than defects/KLOC or defects/FP.
2) Availability :- It is defined as probability that the program is working according to the requirements at a given point in time. It is measured as.

$$Availability = (MTTF / (MTTF + MTTR)) * 100\%$$

(more sensitive than MTTR)

* S/w Safety :- It is a quality assurance activity in which potential hazards are identified & Assessed.

* The ISO 9000 Quality Standards :-

→ In order to bring quality in the product & service, many organizations are adopting the QA system.
→ The Quality Assurance system in the organizational structures that are used to bring quality in responsibilities, procedures, processes & resources.
→ ISO 9000 is a family of family assurance system.
→ It can be applied to all types of organizations.
→ It doesn't matter what size they are or what they do.
→ It can help both product & service oriented organiz^n to achieve standards of quality.
→ ISO 9000 is maintained by ISO, the International Organ^n for Standardization & is administered by accrediation & certificatin bodies.
→ In ISO 9000, company's quality system & operation are scrutinized by 3rd Party auditors for a complaince to the standard & effective operation.
→ This process is called registratin to ISO 9000.
→ On Successful registration, the company gets a certificatin from accreditatin bodies of ISO.
→ Such a company is then called "Iso certified Company".
→ ISO 9001:2000 is a quality assurance standard which is applied to S/W engineering system.
→ The guideline steps for ISO 9001:2000 are

- Establish quality management system
- Document the quality " "
- Support the quality
- Satisfy the customers
- Establish quality policy
- conduct " planning
- control quality systems
- Perform management reviews
- Provide quality resources
- " " personnel
- " " infrastructure
- " " environment
- control realization Planning
- control customer Processes
- " product Development
- " purchasing fun^n's
- " operational activities
- " monitoring Devices
- " non confirming products
- Analyze quality infor^n
- Make quality improvement

← THE END →

ALL THE BEST