10/1/19

Software is a collection of programs that helps to achieve a particular task.

→ Engineering is all about developing products, using well defined scientific principles and method.
→ Software Engineering is a systematic approach to the design, development, operation and maintenance of a software system.

Student software

→ Developer is also the user.
→ Team size (2-3 members)
→ Bugs are tolerable.
→ Requires less investment
→ User interface is not imp.
→ Documentation is not imp
→ Reliability and Robustness does not matter much
→ Quality (Portability, efficiency, maintainability usability, reliability, functionality) is ~~not th~~ doesn't matter much.
→ Cost is less / Time is less required

Industrial software

→ Developer and users are not the same
→ Involve a large group.
→ Bugs are not tolerable
→ Requires more investment
→ User Interface is imp.
→ Documentation is imp.
→ Here it matters alot.
→ Quality is really important.
→ Cost is more required
→ Time is more

Reliability → The software does the task given/assigned to it or not.
Robustness →

Runway :- For the softwares which we can't schedule or define the cost comes under runways.
→ If we can't deliver the software on time, then it gets cancelled (For eg.)

Steps in Software Engineering

① Study of feasibility
→ Determine I/P, O/P, process involved, etc.

② Requirement analysis and specification
→ Here we create a document known as Software Requirement Specification (SRS).

③ Design
→ Data flow diagram (DFDS).

④ Coding and unit testing
① Unit testing ② Integration testing system testing

Till here it is development phases.
After this we have maintenance phases.
↳ System enhancement
→ Handle bugs

Till the lifetime of the software, maintenance phase goes on.
It is a cycle.

Problem domain
→ Software is expensive
→ Late and unreliable.
Till here what we have studied is problem domain. (acc. to syllabus)

→ If some problems / failures arises in the software, then there will be loss to company.

4/1/19.

IEEE definition of Software Engineering:
Software engineering is the application of a systematic disciplined quantifiable approach to the design, development, operation & maintenance of s/w and the study of these approaches. That is the application of engineering to software.

→ The steps which are taking or following for one s/w, then it will be same for other s/w (repeatable).
→ Measurable → we can predict the results of s/w

IEEE definition of Software:
Software is a collection of programs, procedures, rules and associated documentation & data.

Software Engineering challenges

① Scale :-

If we have a complex project, then how to develop is a challenge
But if we have small s/w then we can easily create/develop
scaled/measured

Software (On the basis of KLOC (Thousand lines of code))

Small (<10KLOC) | Medium (<100KLOC) | Large (<1 million KLOC) | Very large (≥1 million KLOC)

Gcc compiler → 980 KLOC
Perl → 320 KLOC
Python → 800 KLOC
Red Hat Linux → 30000 KLOC
Windows XP → 40000 KLOC

② Quality and Productivity (Main goals of S/W Eng)

→ Cost → Schedule → Quality

These are the three imp. terms which are imp in s/w engineering.

→ Cost is measured by persons/month.
→ Efficient software → less schedule time
Schedule → life cycle time

→ Productivity is the ratio of output to resources

$$Prod = \frac{Output\ (KLOC)}{Resources\ (person\text{-}month)}$$

for Productivity ↑ ; Resources ↓ (cost ↓)
" time ↓

Productivity depends both on cost & schedule

② Quality → Quality is good/bad when s/w follow these 6 standard attributes

① Functionality
② Reliability (to maintain a specified level of performance)
③ Usability (capability to be understood, learn or use)
④ Efficiency (" to provide appropiate performance related to the resources used)
⑤ Maintanence (Maintainability)
⑥ Portability
↳ (to be adapted for diff specified environment so that we should not have to do further changes)
↳ (the capability to be modified for the purpose of improvement correction or enhancement)

→ Examples of 2 s/w and which is better.

③ Consistency and Repeatability.

→ Key challenge to ensure succesful results.
→ How can be there some consistency in the degree of quality.

ISO 9001
CMM (Capability maturity model) } Frameworks used to maintain the consistency.

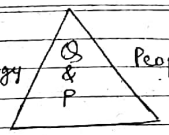
These provides some techniques & methods to ensure consistency which every org. follow.

④ Change.
→ Software should be such that if further enhancement required it should do easily.

⑤ Life → Life of the S/W should be more, such that it works for a large range (apparent period) of time.

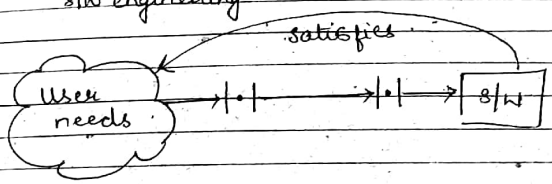
7/1/19

Quality and Productivity depends on 3 factors:-
① Technology
② People
③ Process.



Iron Triangle

→ Tools & Process helps the people to execute any task efficiently.
→ Processes are the systematic approach and core of the S/W engineering.



Two key aspects of S/W eng. approach.
① Development process.
② Managing the development process.

Development process:-
→ It a phased development process.
→ We divide our process in phases so that it is easy to develop & schedule the whole process.

Phase Development process:-
① Requirent analysis.
↳ Problem definition.
↳ Understanding the problem & analysis

→ The O/P of one phase as an I/P to another phase

→ Then we will go for documentation i.e, SRS.

② Software Design:

① Designing of Plan
② Use DFDs.

3 design O/Ps which we can get

- Architectural design → Specify of sub-systems
- High level Design → modules of sub-systems (High level views)
- Detailed Design → Detailed design of every module

③ Coding:

→ Choose prog. lang which is good and easy structure which will help us in easy maintenance.

④ Testing:

① Unit Testing
② Integration testing → Integrating 1st with 2nd and then test → Then combine to 3rd and then test
③ System testing → when all the modules are combined then this testing is done.
④ Acceptance testing → This is done when we show the demo to users.

⑤ Managing the development process:

→ Project management handles all the issues regarding managing the development process.
→ To monitor & control the development process is the function of planning.

Product & Software matrix :- To measure the diff characteristics of product & process.
→ Can measure the characteristics of products based on Quality, productivity, efficiency, tools & technology used, etc.

Software process → set of activities.

H.W

⇒ list some problems that will come up if the methods you currently use for developing ~~small~~ large S/W systems :-

→ Suppose a program for solving a problem cost C. and an industrial-strength S/W for solving that problem cost 10C. Where do you think this extra 9C cost is spent?

{ Suggest a possible breakdown for this extra cost }

→ Industrial - strength software
  ↳ reasonable cost
  → reasonable time
  → should be of good quality

Industrial - strength s/w is very expensive primarily due to the fact that s/w development is extremely labor-intensive.

Main cost involved → manpower employed

→ The cycle time from concept to delivery should be small.
  ↳ Reducing the cost and the cycle time for s/w development are central goals of s/w engineering.

1/19

Software Process.

↳ A particular method of doing something which involves some sequence of steps or operation to achieve some goals is known as process.

Software process :-
The sequence of steps performed to produce s/w with high quality, within budgets and schedule.

In simple way, A s/w process refers to the method of developing s/w.

→ There are some ordering constraint on the set of activities of software process.

↳ s/w process can contain many other processes known as component process.

Process model.

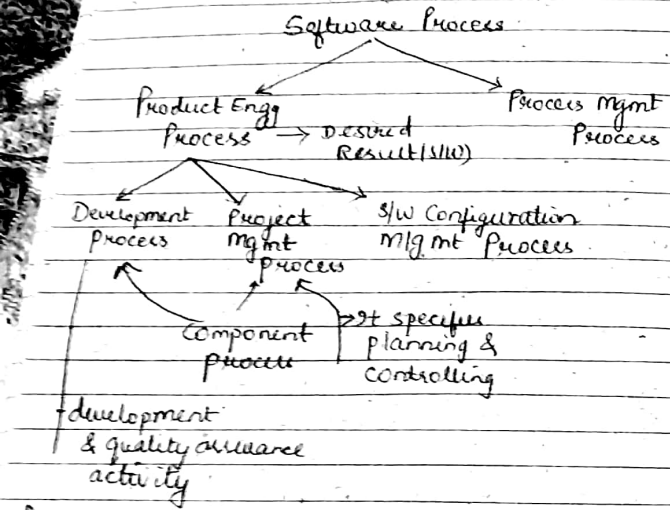
→ It provides guidlines for developing any process

Def → specifies a general process usually as a ~~steps~~ set of stages in which a project should be divided, the order in which the stages should be executed & any other constraints & conditions on the execution of stages.

↳ high quality, low cost and low cycle time are needed to achieve a goal, which is fulfilled by adopting any process model

→ When we execute process we get certain output known as software products.

Component software processes:



Project Mgmt process:

Specifies how to plan & control development activities

S/W Configuration Mgmt Process

→ To manage evolution (& control & change) of your software

Process Management Process:

→ To manage the tools & techniques involved in a process.

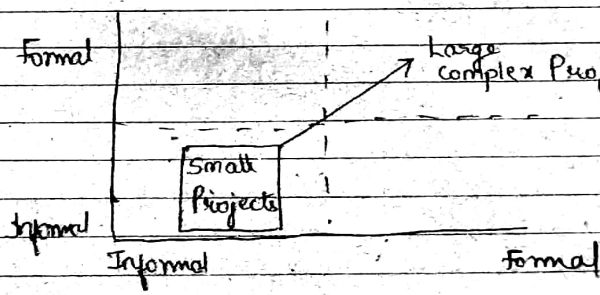
→ Process is a dynamic entity which changes with time so we need to improve our process.

→ Programmers, Designers, testers, developers, etc deals with the Development process

→ Manager, etc deals with Project management process.

→ A group called configuration controllers manages the s/w configuration mgmt process.

→ Software Eng. Process group deals or manages Process mgmt process.



Development Methods

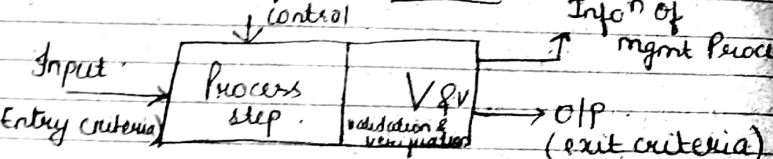
→ For small projects we need informal methods.
→ For large complex projects we need formal methods

ETVX approach for process specification

ETVX → Entry criteria, Task, Verification, Exit criteria

→ specifies the conditions which the entry criteria should meet → entry criteria
→ specifies the conditions that the work product should satisfies to terminate the process → exit criteria

Characteristic of S/W process.

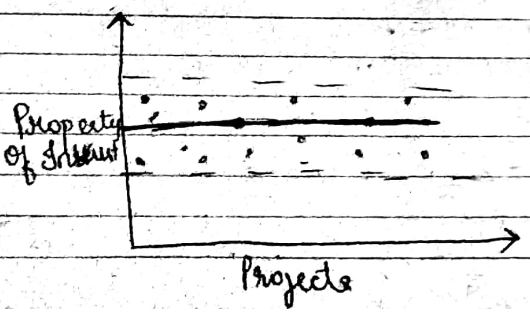


Steps in develop process

Desired characteristics of software Process

① Predictability
It determines how accurately the outcome of following that process in a project can be predicted before the project is completed
→ The fundamental basis for quality prediction is that quality of the product is determined largely by the process used to develop it
→ If we want to use the past experience to control costs and ensure quality, we must use a process that is predictable.
→ A predictable process is also said to be under statistical control

If following the same process produces similar results → results will have some variation which is mostly due to random causes and not due to process issues.



2) Support Testability and Maintainability

→ Maintenance costs generally exceed the development costs
→ Imp obj → to produce s/w that is easy to maintain
→ Overall we can say that the goal of the process should not to be reduce the efforts of design and coding, but to reduce the cost of testing and maintenance.

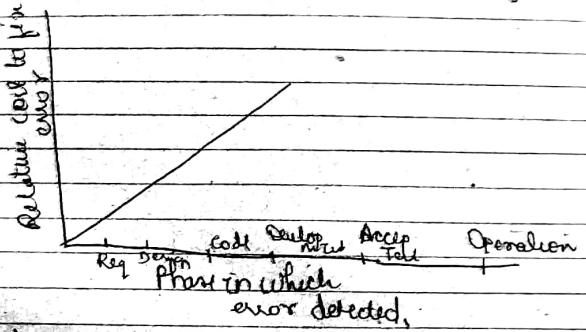
3) Support change

→ Any model that builds s/w and makes change very hard will not be suitable in many situations
→ Change can also take place during development phase
→ Here we are talking about change of requirements
→ Change is prevalent and a process that can handle change easily is desirable.

4) Early Defect Removal

Requirements - 20%.
Design - 30%.
Coding - 50%.

→ Error occurs throughout the development process
→ The greater the delay in detecting error after it occurs the more expensive it is to correct
→ Error detection & correction should be a continuous process that is done throughout s/w development
→ A process should have quality control activities spread through the process and in each phase
→ A quality control (QC) activity is one whose main purpose is to identify and remove defects



5) Process Improvement & Feedback

→ Cost ↓, Quality ↑ (Objective)
→ Having process improvement as a fundamental objective requires that process be a close-loop process

→ Improved based on previous experience
→ Largely done to the process management component of the S/W process
→ Large project → feedback from the early parts of the project can be used to improve the execution of the rest of the project.
→ Iterative development process model → feedback from one iteration is used to improve the execution of later iterations

20/1/19

S/W Development Process Models. (Rajeev Mall book)

1) Classical Waterfall Model. ✓
2) Iterative Waterfall Model. ✓
3) V-model. ✓
4) Prototyping model.
5) Incremental Development model. ✓
6) Evolutionary model. ✓
7) RAD (Rapid application development) model
8) Spiral model.
9) Agile Development Model.

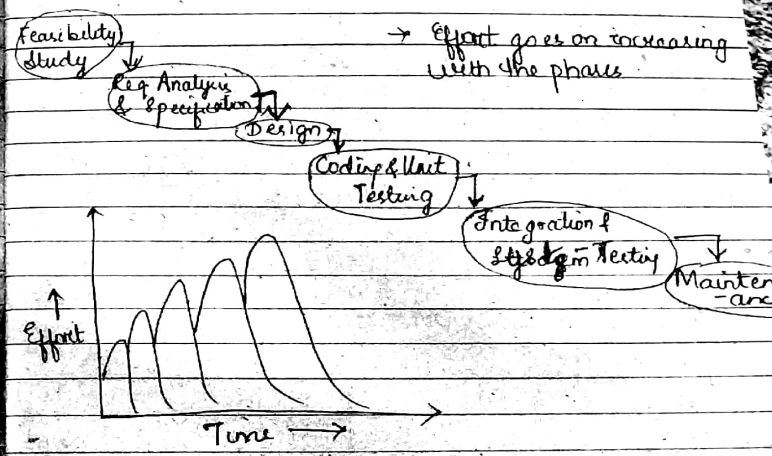
Classical Waterfall model.

Phases:-

Feasibility study

(2) Req. Analysis and specification.
(3) Design.
(4) Coding and Unit Testing
(5) Integration and System Testing
(6) Maintenance.

→ Due to its linear structure we call this as waterfall model.



→ Effort goes on increasing with the phases

Adv. → Simple and easy
→ Can be used in small S/W

Shortcomings → No feedback path.

→ Improved based on previous experience
→ Largely done to the process management component of the S/W process
→ Large project → feedback from the early parts of the project can be used to improve the execution of the rest of the project.
→ Iterative development process model → feedback from one iteration is used to improve the execution of later iterations

28/11/19

S/W Development Process Models. (Rajeev Mall book)

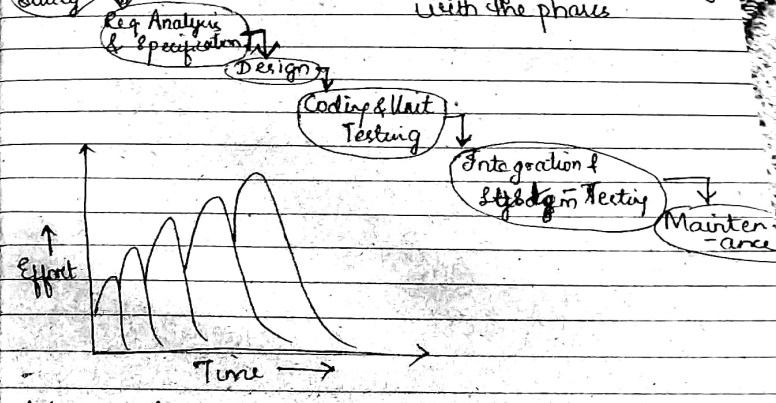
1) Classical Waterfall Model. ✓
2) Iterative Waterfall Model. ✓
3) V-model. ✓
4) Prototyping model.
5) Incremental Development model. ✓
6) Evolutionary Model. ✓
7) RAD (Rapid application development) model
8) Spiral model.
9) Agile Development Model.

1) Classical Waterfall model.

Phases:-

① Feasibility study
② Req. Analysis and Specification
③ Design
④ Coding and Unit Testing
⑤ Integration and System Testing
⑥ Maintenance.

→ Due to its linear structure we call this as waterfall model.



→ Effort goes on increasing with the phases

Adv. → Simple and easy
→ Can be used in small S/W

Shortcomings → No feedback path.

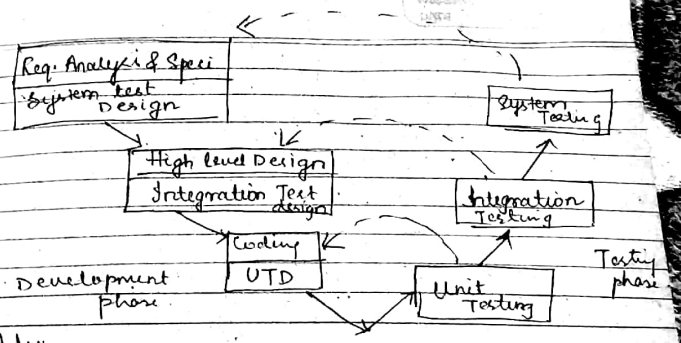
② All the requirements will have to be gathered at the start.
③ Difficult to accomodate change request
④ Inefficient error corrections
⑤ There is no overlapping of phases and phases go sequence wise
⑥ Final SW will be obtained at the end.

2) Iterative Waterfall model.

→ Enhancement of classical waterfall model.
→ We can go back to the previous stage from one stage but there will be no feedback from Req. Analysis & Specification to Feasibility study phase.

Shortcomings:
↳ Incremental delivery not supported.
↳ Error correction is expensive
↳ No support for risk handling and code reuse.

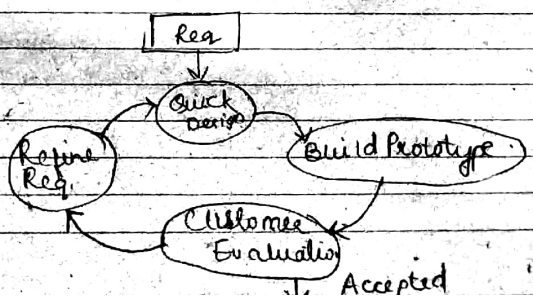
3). V-model.



Adv.
→ Much of testing activities are carried out in parallel with development activities

Disadv.
→ Final software will get at end.
→ Time consuming.

4) Prototyping model.



↳ After accepting same phases as waterfall model.
↳ We build a prototype for user to give its feedback.
↳ We can also revise it.

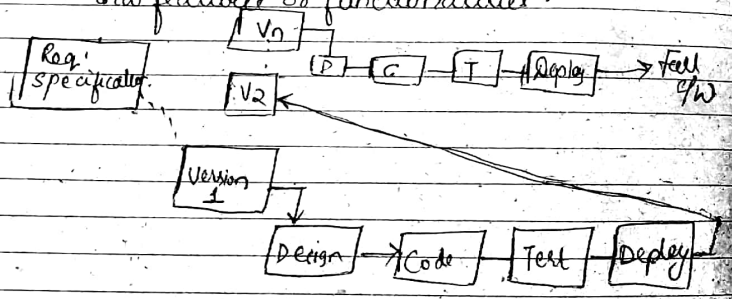
Adv:
↳ Here requirements can be changed with feedback from users/clients.

Disadv:
↳ Time consuming
↳ Cost (expensive)

24/1/19

Incremental development model:-

→ Gather requirement and then divide them into features or functionalities.



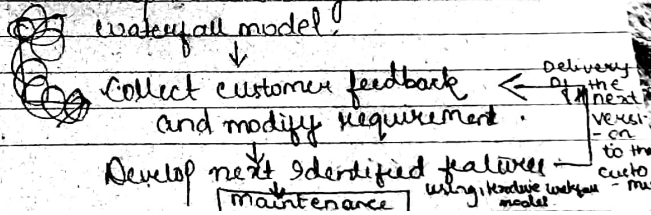
→ Here we do not make any long term plans

Adv ① Error correction.
② Incremental Resource Deployment.

Disad
① Time Consuming & cost
② We can't accomodate any change here.

Evolutionary Model.

→ Note down rough requirements in SRS document.
→ Develop core functionalities.
→ After developing this, we will deliver & take feedback from the customers or users.



Sometimes it is referred as
→ Design a little, build a little, test a little and deploy a little model.

Adv
→ Effective elicitation of actual customer requirements.
→ Easy handling change request.
→ Useful for large systems.

Disadv
→ Feature division into incremental parts can be non-trivial.

RAD (Rapid application development) model

→ Prototyping
→ evolutionary models.

→ Create a prototype and deliver it to user get feedback and then move further.

↳ Prototypes are constructed & incremental features are developed and delivered to customer
↳ Prototype are not thrown away but are enhanced & used

Adv.
To decrease the time & cost incurred to develop s/w

② To limit the cost of accomodating change request.
③ To reduce the comm. gap b/w the customers & developers.

As we are reusing the prototype - so the cost can be less.
→ 4-5 members in a team but make one representative.

H/W
Applicable to which s/w.

Suitable -

① Customised software
↳ Developed for one or two customers only by adapting an existing software.
↳ Substantial reuse is usually made of code from pre-existing s/w.

② Non-critical s/w.
③ Highly constrained project schedule.
→ RAD aims to reduce development time at the expense of good documentation, performance & reliability.

Applicability of RAD model.
→ Where it can be used and where not.
→ How it is different from prototyping, evolutionary waterfall, and incremental

④ Large Use
→ Software supporting many features can incremental development and delivery be successfully carried out.

Not suitable

① Generic products (wide distribution)
② Requirement of optimal performance and/or reliability
③ Lack of similar products
④ Monolithic entity
For small sized software, it may be hard to divide the required features into parts that can be incrementally developed and delivered.

Comparison

① The code developed during prototype construction is usually thrown away.
In contrast, in RAD it is the developed prototype that evolves into the deliverable software.
(RAD versus prototyping model)

RAD versus Iterative waterfall (IWM)

→ In IWM, all the functionalities of a s/w are developed together.
whereas in RAD model the product functionalities are developed incrementally through heavy code and design reuse.

→ It is easy to accomodate any request for requirement changes in RAD.
But in IWM, does not support any such mechanism.

→ Use of IWM leads to production of good quality documentation which can help during software maintenance.

→ The developed s/w by IWM has better quality and reliability than that developed using RAD.

RAD Vs evolutionary model.

→ In RAD each increment results in essentially a quick and dirty prototype., whereas in

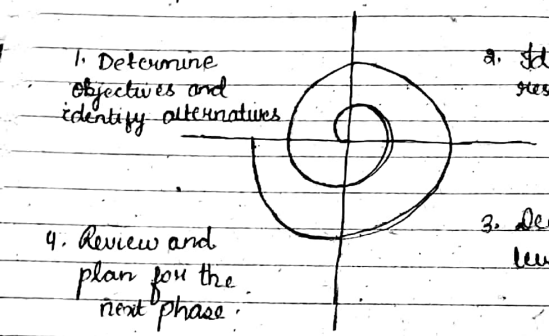
evolutionary model each increment is systematically developed using the IWM.

→ In RAD model, s/w is developed in much shorter increments compared to the evolutionary model.
The incremental functionalities that are developed are of fairly larger granularity in the evolutionary model.

— x —

9th Aug

Spiral Model.



→ It is mainly used for risk handling
→ After every phase you make a prototype.
→ Here is a project planner which checks whether s/w is ready or not.

Disadv:
① It is used for few projects which is complex.

Agile Development model.

→ We deliver a version to customer & then get feedback
→ Iteration → Time box.
Within this time box, a version is created and is delivered to the customer.
→ Face to face communication (main focus).
5 to 9 development teams.
→ Pair programming.
Two programmers work simultaneously at the same time. (If one will write then other will review)

Drawback:
→ If we don't do documentation properly then other will not understand etc.
→ Problems in maintenance.
maintenance.

Types of Agile model:

Extreme programming model

→ If something is beneficial then why not put it to constant use.

→ Code review, testing, incremental review, integration testing, etc.

Scrum

Scrum model.

→ Series of chunks, we will develop a model.

→ Here time-box is known as sprints.

Q Make a chart → application, adv & disadv for every model. (In applications, mention the name of softwares)

Model	Adv	Disadv	Application
① Classical waterfall model.	① Very simple and easy to understand. ② Phases in this model are processed one at a time. ③ Each stage is clearly defined. ④ Process, actions and results are very well documented. ⑤ Works well for smaller projects.	① NO feedback path. ② Difficult to accomodate change requests. ③ No overlapping of phases. ④ Poor model for long & ongoing projects. ⑤ High amounts of risk & uncertainty.	① When requirements are very well documented, clear and fixed. ② Product def. is stable. ③ Technology is understood & is not dynamic. ④ Project is short. ⑤ No ambiguous requirements.

Model	Advantages	Disadv.	Applications.
Iterative waterfall model.	① feedback paths. ② Simple and easy to use. ③ Each stage is clearly defined. ④ Process, actions and results are very well documented.	① Difficult to accomodate change requests. ② Incremental delivery not supported. ③ Phase overlap not supported. ④ Error are correction is unduly expensive. ⑤ Phase overlap not supported. ⑥ Limited customer interactions. ⑦ Heavy weight. ⑧ No support for risk handling & Code reuse.	① There is a time to time market constraints. ② Requirements of the system are clearly defined and understood. ③ Resources with needed skill set are not available and are planned to be use in contract basis.
V-model	① Much of testing activities is carried out in II with the development activities. ② The quality of test cases are better. ③ The test team is associated with the project from beginning.	① High risk & uncertainty. ② It is not a good model for complex and object-oriented projects. ③ It is not suitable for projects where requirements are not clear.	① When req. are clearly defined and fixed. ② When ample technical resources are available with technical expertise.

Model	Advantages	Disadvantages	Application
Prototyping model	① Increased user involvement in the product even before its implementation. ② The users get a better understanding of the system being developed. ③ Missing functionalities can be identified easily. ④ Quicker user feedback is available leading to better solns.	① Can increase the cost of development projects that are routine development work. ② It is effective only for those projects for which the risks can be identified upfront before the development starts. ③ It is ineffective for risks identified later during the development cycle.	① For the development of the graphical user interface (GUI) part of an application. ② It is useful when the exact technical solns are unclear to the development team. ③ It can be used when development of highly optimized and efficient s/w is required.
Incremental Development model	① Error reduction, leading to greater reliability of the software. ② Incremental resource deployment. ③ Lower initial delivery cost. ④ Uses divide & conquer for breakdown of tasks.	① Requires good planning and design. ② Total cost is not lower. ③ Well defined module interfaces are required. or need for early realization of	① Projects with new Technology ② When projects having lengthy development schedules ③ When requirements are known up-front. ④ Funding Schedule, Risk, Program Complexity.

Model	Advantages	Disadvantages	Application
Evolutionary model	① Easy handling change requests. ② Effective elicitation of actual customer requirements.	① Feature division into incremental parts can be non-trivial. ② Ad hoc design.	① It is normally useful for very large products. ② Projects using object-oriented development.
RAD model	① Use of reusable components helps to reduce the cycle time of the project. ② Feedback from the customers is available at initial stages. ③ It is easier to accomodate changing requirements due to the short iteration time spans. ④ Reduced costs as fewer developers are required.	① The use of powerful and efficient tools requires highly skilled professionals. ② Customer involvement is required throughout the life cycle. ③ It is not meant for small scale projects. ④ The absence of reusable component can lead to failure of the project.	① Customized Software ② Non-critical software ③ Large s/w ④ Highly constrained project schedule ③ Can be used if teams consult of domain experts.

Model	Advantages	Disadvantages	Application
Spiral model	① Risk Handling and analysis at every phase ② Good for large & complex projects ③ Flexibility in requirements ④ Customer satisfication	① Complex ② Expensive ③ Difficulty in time management ④ Too much dependable on Risk Analysis. ⑤ Customer is not sure of their requirements which is usually the case.	① New product line which should be released in phases to get enough customer feedback. ② Long-term project commitment ③ When there is budget constraint and risk evaluation is imp. ④ Requirements are complex and need evaluation to get clarity.
Agile development model	① It reduces total development time of the whole project. ② Working through pair programming produce well written compact programs which has fewer errors as compared to programmers working alone	① Due to lack of formal documents, it creates confusion. ② Due to absence of proper documentation, when the project completed and the developer are assigned to another project, maintenance becomes a problem.	① Projects in which there is alot of customer interaction. ② Projects where documentation is not necessary. ③ Suitable for frequent changing requirements

③ Customer representative get the idea of updated form of prod. after each iteration.

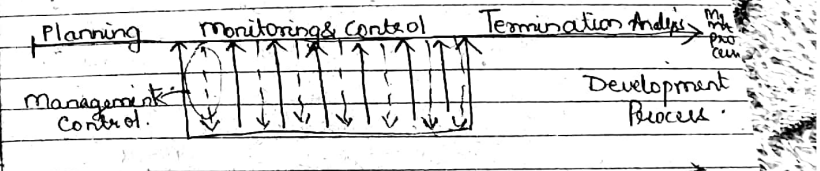
30/1/19 ①

Project Management Process (PMP)

→ All the models studied till now comes under development process.

→ PMP specifies all the activities that have to be planned, control or monitor.

→ Here we will do planning, controlling or monitoring and termination analysis.



↑ → Metric value. ↕ → Management control.

Planning → how much schedule will be req.
→ how many resources " " "
→ All the requirements are specified during the planning process.

Termination analysis → will say whether the results given by development process is good / bad.

→ It is also known as participation process.
→ It says whether development process is good or bad.

Monitoring & control
→ We will monitor the development process here
→ And control the quality, resources, etc

Inspection process

→ Main goal is to detect defects in work products
→ After each phase, the product which we get is work product.
→ Inspection is done earlier then a good software product will be come as o/p.

Characteristics of Inspection process
① conducted by technical people for technical people
② It is a structured process with defined goals for the participants.
Scribe :- write down all defects.
Moderator :- whether communication is done properly or not.
③ Focus is on identifying problems, not resolving them.
④ We can maintain a log, when members can see and resolve the defects.

Stages :-
Planning → Preparation & overview → group review meeting → Rework & follow up

Planning - ① Plan the objective of the inspection process.
② Author will check when the product will be ready for inspection (which is decided by entry & exit criteria)
③ How many members will be there in inspection team.

Preparation & overview ① Reviewer will again prepare a self review report.
② This report is known as self review log

Group review meeting → ① All the reviewer will do a group meeting
② They will discuss about their self review log.
③ Finally a summary report will be created.

Size of work product = 14 pages.
Total product defects = 19
Minor defects = 16
Major defects = 3.

Minor defects per page = $\frac{16}{14} = \frac{8}{7} = 1.14$
(defect density)

Defect density = $\frac{\text{Defects}}{\text{size of work product}}$

④ Rework and follow up :-
→ Author will correct the defects which have been detected earlier.

Roles and responsibilities

We can assign multiple roles to moderator, reader, etc.
→ A moderator can be a reviewer also.
→ But a author can never be moderator.
Moderator work.
→ Schedule the group review meeting.
→ Ensure that meeting stays on focus of defect identification.

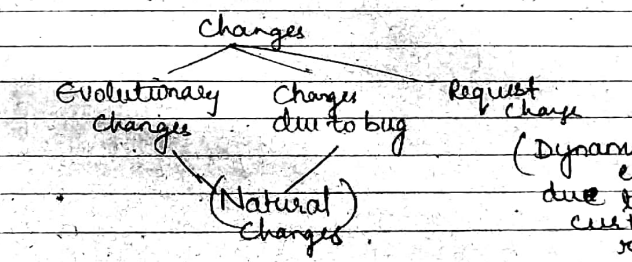
→ Check whether reviewers have prepared log record or not.

work product different → Participants different → Inspection focus will be different

31/1/19
S/W (11)
Configuration Management Process

→ To control the evolutionary change request
→ If Bugs arise then we have to change our S/W
→ Requirement change request.

It is a discipline for systematically controlling the changes that take place during development



See IEEE def on your own.

It give some functionalities i.e.,

① Gives latest version of a program.

② Undo a change or revert back to a specified version.

③ Prevent unauthorized changes or deletion.

④ Gather all sources & information & documents for current version of system.

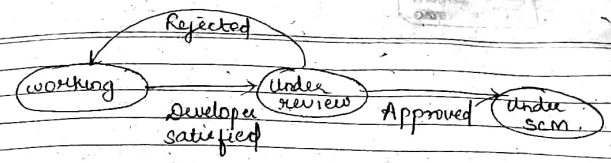
This process is simply known as configuration management process or (CM) process.

Configuration management mechanisms

① Configuration Identification & base lining
② Version control & Mgmt.
③ Access control.

SCI → software configuration Item.

SCM life cycle of an item.



SCM library → To store diff. versions of the software

steps in this process :-

Planning, execution, to store properly, to control & monitor

CC → Configuration controller
CCB → Configuration control Board

(IV) Requirements change Management process

→ Manage the requirement change in systematic way.

Steps :-

① log the changes :- Maintain the history.
② Perform impact analysis on the work product :- identify software config. item.
③ Estimate impact on efforts & schedules
④ Review impact with concerned stake holders
⑤ Rework work products.

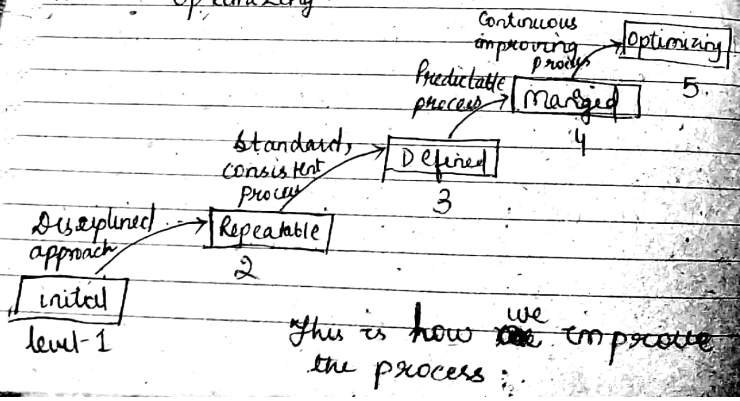
V Process Management Process

→ It is a dynamic entity.
→ Main focus :- how to improve process.

Process capability :- Expect results, quality, productivity of a project.

CMM :- Capability maturity model.
↳ Provides framework in the form of levels to improve process.

CMM levels :-
Level 1 :- initial
level 2 :- Repeatable
level 3 :- Defined
level 4 :- Managed
level 5 :- Optimizing.



This is how we improve the process.

→ Project mgmt, quality assurance etc we have to improve in level-1.
So we move to level-2.
To go to level-2 we need a disciplined approach as it is lacking in level-1.

1/2/19

Software Requirement Analysis And Specification

→ In small software, you don't have to write the SRS document as the problem & steps are in the mind of the developer.
→ But for large scale software, you have to meet with the end users and understand their requirements and document it in SRS document.

IEEE def of requirement.
A condition or a capability that must be possessed by a system to satisfy a ~~compliant~~ contract, standard, specification or other ~~informally~~ imposed document.

→ SRS document describes what a proposed s/w should do.

Need for SRS:

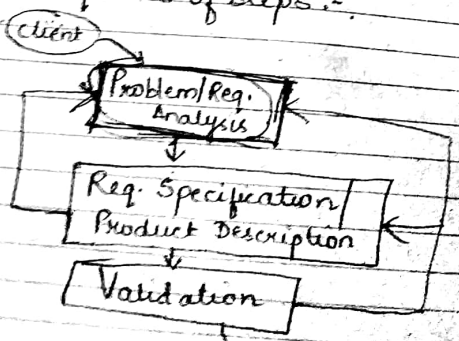
↳ SRS document is a medium through which client and user needs are accurately specified to the developer.

Requirement:

→ It forms or establish the basis of agreement b/w the user and the supplier.
→ Helps users to understand his/her needs.
→ SRS provides a reference for validation of the final product.
→ High quality SRS essential for high quality software.
→ Good SRS reduces the development cost.
→ It can minimize changes and errors.

Requirements Process

→ We follow a sequence of steps:-



→ For problem analysis → we apply divide and conquer strategy.
→ Organization is the key for the large volume of information.
→ We can also take help of DFDs for problem analysis.

Approaches for Problem Analysis:

① Informal approach. → Here, we just do analysis informally.
② Data flow modelling
↳ Here, we make use of DFDs.
③ Object oriented modelling/approach.

1/2/19

Tutorial

Tasks involved in problem analysis:-
- interviewing clients & users.
- Reading manuals.
- Studying current system.
- Helping clients & users understand new possibility

Informal approach,

→ Requirements will in the mind of analyser or developer and he/she will write it down in SRS document.

Data flow modelling

→ Structured analysis technique.
→ We use function based decomposition.
→ Top down approach
→ Use data flow diagrams & data dictionaries.

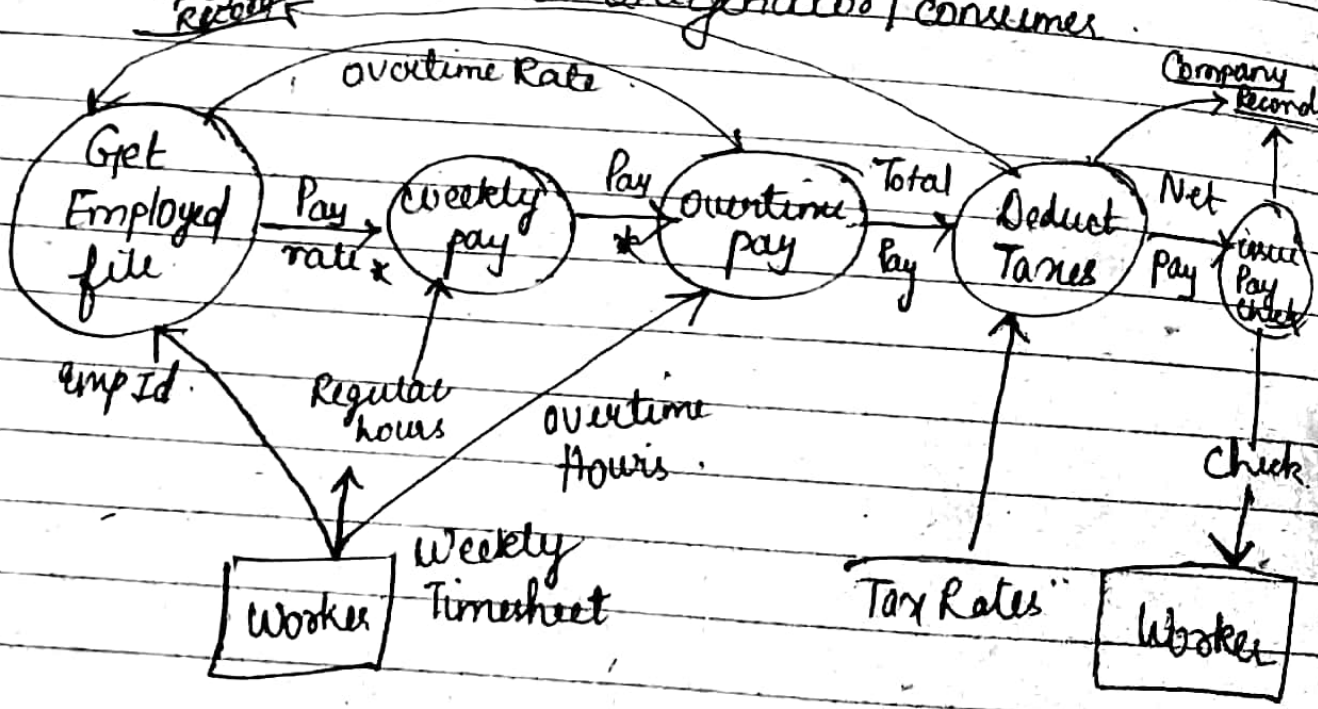
In data-flow diagram,

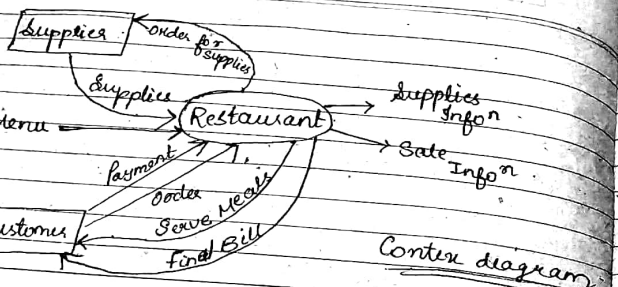
○ → represent process

→ → arrow is for data flow.

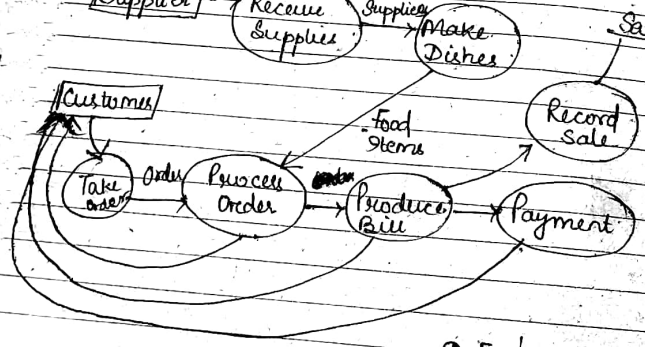
□ → source / sink.

Source can be originator/consumer.





Context diagram

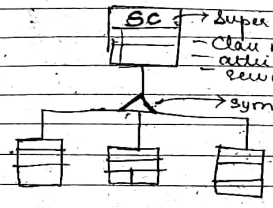


DFD's

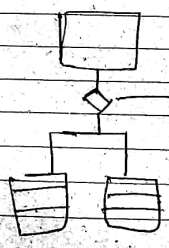
→ Further according to the user requirements we will change our DFD's with details.

Object oriented approach.

→ Notation → Objects, classes.
→ Here we have class diagrams.



→ In case of Inheritance → generalization Specialization structure



④ Prototyping approach.

→ Read yourself.

— x —

7/2/19

Characteristics of an SRS.

① Correct
② Complete
③ Unambiguous.
④ Verifiable.
⑤ Consistent.
⑥ Ranked for Importance/Stability
⑦ Modifiable.
⑧ Traceable.

→ Your SRS is correct when every requirement specifies perform some function.

→ Every requirement must be specified and all the responses to all the input variables in the software. Then we can say our SRS is complete.

⇒ If and only if every requirement stated have only one interpretation. Then we can say SRS is unambiguous.

⇒ There must exist a cost effective to verify the requirements (to check s/w satisfy the requirements). Then we can say SRS is Verifiable.

⇒ Some req. can be unimp. So we prioritize the requirements. There can be core req. (so we will develop it first) as it is stable and important. Then we can say SRS is stable or ranked for importance.

→ If any change request arrives, and we can change SRS then it is modifiable.

→ We can trace the origin of each of its requirements. Then SRS is traceable.

→ Word is different but referring to single object. Then we can say SRS is inconsistent. This should not happen to feel that SRS is consistent.

Components of SRS.

① Functionality (heart of the SRS)
② Performance
③ Design Constraints.
④ External interfaces.

① Specifies all the functionalities that a system should support.

② Specifies all the performance of the s/w system. Performance requirements can be static or can be dynamic.

Performance Requirement

1) Static
- that does not impose any constraint on the SW execution characteristics

2) Dynamic
- specify the constraint on the external behaviour of the sys.

State const includes
no. of users, files, etc.
→ Dynamic const / requirements includes
Response time and throughput constraint.
- No. of operations that can be performed in a unit time.

Here we use measurable terms

3) Design constraints
→ deals with the factors in the client's environment.
→ Standards, reliability, fault tolerance, security, backup, hardware, etc are the design const.

1) External interface
Specify all interaction with the people, s/w or hardware.

Specification language
→ The language should be such that it support the desired characteristics / qualities of the SRS
→ For easy understanding → we use natural languages
But at natural lang. it is ambiguous so we use formal lang as well
→ Natural lang → to specify the structure
Formal lang → to " " features / characteristics
→ Thus we use both natural and formal languages.

Structure of an SRS document
→ We mainly use IEEE structure

1. Introduction
1.1 Purpose
1.2 Scope
1.3 Def, acronyms and abbreviation.

Performance Requirement

1) Static
↓
that does not impose any constraint on the s/w execution characteristics.

2) Dynamic
↓
specify the constraint on the execution behaviour of the sys.

→ Static const. includes no. of users, files, etc.
→ Dynamic const / requirements includes Response time and throughput constraint.
↓
No. of operations that can be performed in a unit time.

Here we use measurable terms.

③ Design constraint
→ deals with the factors in the client's environment.
→ Standards, reliability, fault tolerance, security, backup, hardware ref, etc are the design const.

④ External interface
Specifies all interaction with the people, s/w or hardware.

Specification language

→ The language should be such that it support the desired characteristics / qualities of the SRS.
→ For easy understanding → we use natural language.
But as natural lang. is ambiguous so we use formal lang as well.
→ Natural lang → to specify the structure
Formal lang → to " " features / characteristics.
→ Thus we use both natural and formal languages.

Structure for an SRS document.

→ We mainly use IEEE structure.

1. Introduction
   1.1 Purpose
   1.2 Scope
   1.3 Def, acronyms and abbreviation.

1.4 References
1.5 Overview.

2. Overall Description
↳ Factors that affect the product
2.1 Product perspective
2.2 " functions
2.3 User characteristics
2.4 General constraints.
2.5 (See from book)

3. Specific Requirements.
↳ We write the detail of development
(See from book)

Traditional approach → to specify all the functions.

Then we use Use cases :- to specify all the functions by specifying the behaviour of the system.
→ We also use it in design analysis & problem analysis.
Here we have → Actors.
→ Primary Actor.
→ Scenario.
→ Main Success Scenario
→ Extension Scenario.

Primary
→ Actor can be a person or system, for which we are ~~developing a product to achieve some goals~~ starting a Use Case or for which the product has been developed.
→ ~~Primary~~ Actor :- which uses a system.

→ Scenario :- set of actions performed to achieve a goal.

→ Main Success scenario :- if nothing fails, and all the cases of Scenario succeed. Then that set of actions is called main Success scenario.

→ Some scenarios in the main fails then these Scenarios come up → Extension Success scenario.

eg
Auction :- There can be Use cases for other Use cases.
↳ Seller :- put an item up for auction.
Use Case 1:
Primary Actor.
→ Precondition → Seller have logged in.

→ Main Success Scenarios :-

↳ All the steps and actions come under this.

Here; → Seller set prices.
→ System shows past prices to seller.
→ etc.

Use case → Make a bid for the seller.

HW

→ Study Use cases and come.
→ Make Use cases for auction system.

Actors :- A person or a system which uses the system being built for achieving some goal.

Primary Actor :- The main actor for whom a use case is initiated and whole goal satisfaction is the main objective of the use case.

Scenario - A set of actions that are performed to achieve a goal under some specified conditions.

Main Success Scenario :-
Describes the interaction if nothing fails and all steps in the scenario succeed.

Extension scenarios :-
Describe the system behaviours if some of the steps in the main scenario do not complete successfully.

Auction

- Use case 1 : Put an item up for auction.
  Primary Actor : Seller.
  Precondition : Seller has logged in.
  Main Success Scenario :
  1. Seller posts an item for auction.
  2. System shows past prices of similar items to seller.
  3. Seller specifies the starting bid price & a date when auction will close.
  4. System accepts the item & post it.
  Exception scenarios :-
  — 2 a) There are no past items of this category.
  • System tells the seller this situation.

• Use case 2: Make a bid
Primary Actor: Buyer
Precondition: The Buyer has logged in.
Main success scenario:-
1) Buyer Searches and browses and selects some item.
2) System shows the rating of the seller, the starting bid, the current bids, and the highest bid; asks buyer to make a bid.
3) Buyer specifies a bid price, maximum bid price, and an increment.
4) System accepts the bid; Blocks funds in bidders account.
5) System updates the bid prices of other bidders when needed, and updates the records for the item.

Exception Scenarios:-

-3 a) The bid price is lower than the current highest.
• System informs the bidder and asks to rebid.

-4 a) The bidder does not have enough funds in his account.
• System cancels the bid, asks the user to get more funds

• Use case 3: Complete auction of an item
Primary Actor: Auction System
Precondition: The last date for bidding has been reached.
Main success scenario:-
1. Select highest bidder; send email to selected bidder and seller informing final bid price; send email to other bidders also.
2. Debit bidder's account and credit seller's.
3. Transfer from seller's account commission amount to organization's account.
4. Remove item from the site; update records.

Exception scenarios:- None.

Developing use cases -

1. Identify actors and goals.
   - → The purpose is actors goal list -
   - → Involve every actor [illegible] of the use case
   - — Define scope of the system

3. [illegible] Specify main success scenario
   - → For each use case expand main scenario
   - → Describe normal behaviour of the system.

4. Identify failure conditions
   - → You list all possible failure conditions for use case
   - — Specify how it may fail (use case)
   - → [illegible]

5. Specify failure handling
   - → If any failure occurs, what will be the system behaviour.

Buy a book (Draw use case)

II

Validation

- → SRS document contain some errors
- → Errors can be like -
  Omission, Incorrect facts, Inconsistency and Ambiguity
- → Any requirement may be omitted

Two Requirements should not be conflicting

→ Omission - 35 %
Incorrect fact - 10 %
Inconsistency - 38 %
Ambiguity - 28 %.

→ We remove these errors in SRS document to reduce the error in design process

→ To remove these errors we do requirement review / Inspection.
↳ It is a review by a group of ppl to find out errors.

Scanned by CamScanner

Team:-

① Author of the requirement document.
② Someone who understand the needs of the client.
③ A person of the design team.
④ Person responsible for maintaining the requirement document.

→ We then prepare a check list for the review ← contains
→ All hardware resources defined etc (see from book)

* We didn't get answers of any of these checklist questions, we go for again SRS requirement

→ If our SRS is not validated then we can again go for SRS specification & requirement.

Metrics (It is used to measure the performance).

① Size.
② Quality

Considering buying a book online.

Use case 1: ~~Seller~~ Check for stock and update it online.
Primary Actor: Seller
Precondition: Seller has logged in to the system or have updated it to admin.
Main success scenario:-
1) Seller check its stock and update the availability of copies
2) Set the price and discount rate
3) ~~Seller~~ Update the previous copy

Extension scenario
→ If he found that stock is 0
• Orders the stock from its buying source

Use Case 2: Check in for the book
Primary Actor: Buyer or customer
Precondition: Customer has logged in to the online shopping portal.

Main success scenario:-
1) Search the book with its keyword or full name.
2) From the many available seller of the book choose one.

3) Find the best seller according to some criteria (like cost).
4) Add to cart.
5) Place the order by filling all details like address, quantity, etc.
6) Money blocked from its account or deducted from its account.

Extension scenario:
-1 Could not find the book in its first search
- Change keyword.
- Search again

-2. No seller have stock any
- Wait for the seller till the seller update the stock.

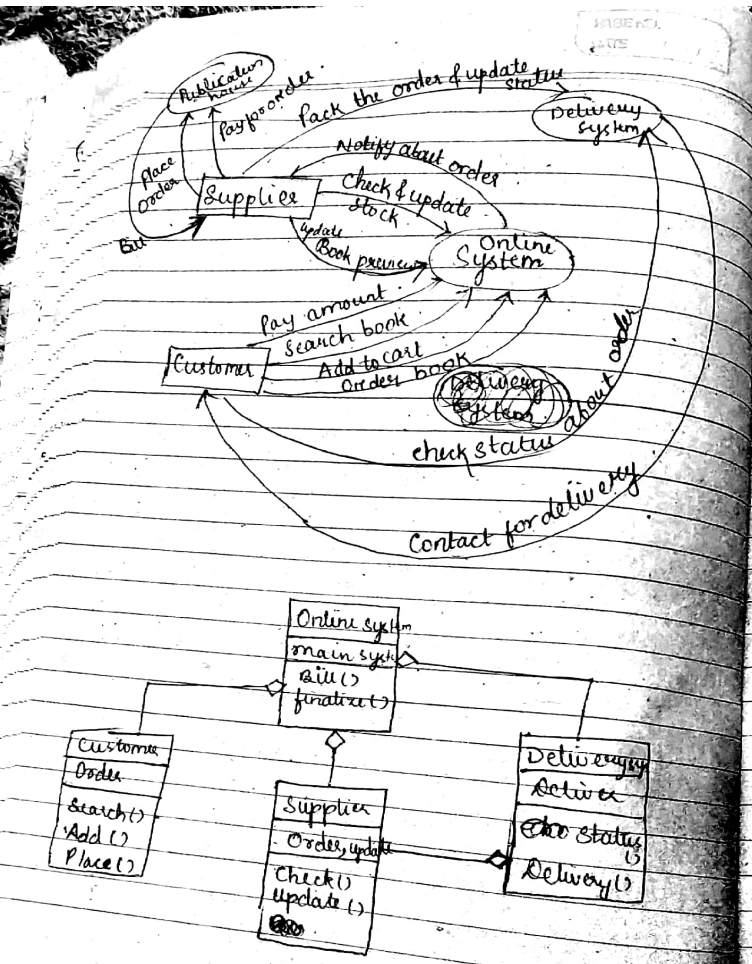
-6 Not sufficient money in the account
- Being notified about it to credit required balance in its account.
- Transaction failed for that moment.

Use case 3: Delivery of the book Notification of book status
Primary Actor: System
Precondition: Order placed by the buyer.

Main success scenario:
1) The seller has been notified about the booking or purchase.
2) Packing is done by seller and then out for shipping system updates status
3) System updates regularly about the status
4) On delivery time, contact the buyer and deliver the book.

Extension scenario:-
-4 Buyer contacting failed.
- Buy again, if still not then order being returned.
- All the system changes in buyer and seller a/c is updated.

— X —



11/02/2018

Metrics

→ These are used to provide quantitative information to the management process so that information can be used to effectively control the development process.

→ To estimate your efforts, cost and schedule we can use some metrics.

① Size :- According to the size of SRS.

② Quality :- - no:- of paragraphs
- no:- of pages
- no:- of functional requirements

These all factors are highly dependent on author. So no:- of pages and paragraphs is not a good indication to measure metrics.

Function point :- → to measure the size wrt to what system performs.

↳ System functionality is calculated in terms of no:- of functions implemented, the no:- of inputs & outputs, etc.

→ Five different parameters used :-

① External I/P type.

④ External o/p type
⑤ Logical internal file type
⑥ External interface " "
⑦ " inquiry type

Each parameter is divided into 3 categories
① Simple ② Average ③ Complex.

Logical internal file type -
→ Every application have to perform some functions. These form logical internal file type.

External
→ Every type can be input or output.

To calculate the function point, we first calculate Unadjusted function point.

$$UFP = \sum_{i=1}^{i=5} \sum_{j=1}^{j=3} w_{ij} c_{ij}$$

w_{ij} - entry in the ith row & jth column

Adjusted function point (DFP)
Complexity adjustment factor (CAF)

$$DFP = CAF * UFP.$$

w_{ij} represents the contribution of an element of the type i and complexity j

$i \rightarrow$ parameters. (Among 5)
$j \rightarrow$ complexity (Simple, Avg, Complex)
c_{ij} = count of no. of elements of type i

→ 14 different characteristics of the system are there to calculate CAF.

→ 0 → not present
1 → Insignificant influence.
2 → moderate "
3 → average "
4 → significant "
5 → strong "

} 5 levels of value for characteristics (Value of N)

$$CAF = 0.65 + 0.01 N$$

→ {1, 2, 3, 4, 5}

→ The 14 characteristic value can have any 1 of these 5 levels or values.

→ CAF can have value in range 0.65 – 1.35.

→ CAF can have total values = 14 X 5 = 70.

Size is measured in KLOC (Kilo lines of code).

Advantages of function points over KLOC

→ function points can be measured using SRS but KLOC is known after.

II. Quality

1) No. of errors found.
2) Change Request Frequency.

→ less errors, high quality product.

Case Study

① Course scheduling
② Personal Investment Management System (Will be discussed in tuts)

—X—

Software Architecture

→ Definition
→ Role of software Architecture

4/2/19

→ It is a top level view of software
→ S/w architecture helps in
  (i) communication
  (ii) understanding
  (iii) Reuse.
  (iv) Construction of evolution.
  (v) architecture analysis.
    ↳ we can analyse reliability and security

→ 3 main views of S/w:
  (i) Module view
  (ii) Component and connector view
  (iii) Allocation view.

(i) Module view:- In this we define system and units of codes.
  It includes - classes & packages, procedure.

(ii) It includes run time.
  We also define relationships in this.
  Relationships:-
  - is a part of (Module A is a part of module B)
  - depends on / uses (Module A uses services of module B)
  - uses.

→ The system is viewed as units each [illegible] software.

(b) In this, system is viewed as collection of runtime entities.

eg:- objects, process
↳ instance of classes.

- all these are runtime entity
- Interaction b/w entities is provided with connectors

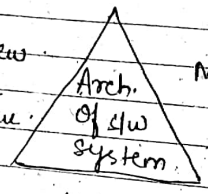
eg:- pipes and sockets.

- Shared data / shared data memory

(C) Allocation:-

→ how different units are allocated.

Block diagram of s/w architecture:-

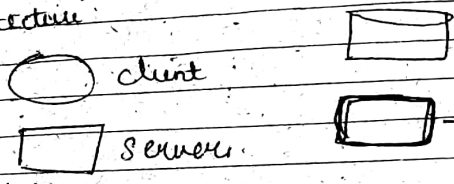


Main view is component and connection view

It has 2 main components

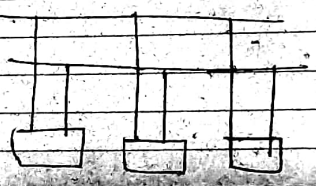
- Components → Any computational model tools like servers.
- connectors → It provide interaction between components.
- Components are data stores.

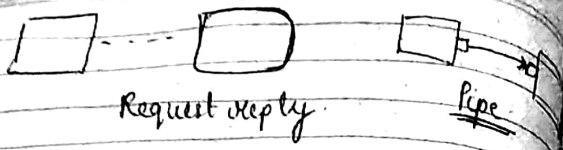
→ These symbols vary from architecture to architecture



Connectors

(i) Bus type connectors

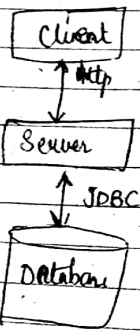




Eg:- Suppose we have to design and build a simple system for taking and online survey of students on campus.

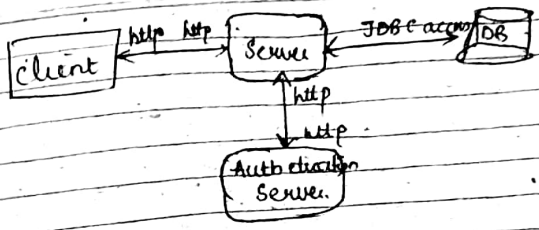
There is a set of MCQs and, and the proposed system will provide the survey form to the student, who can fill and submit it online.

Main components:

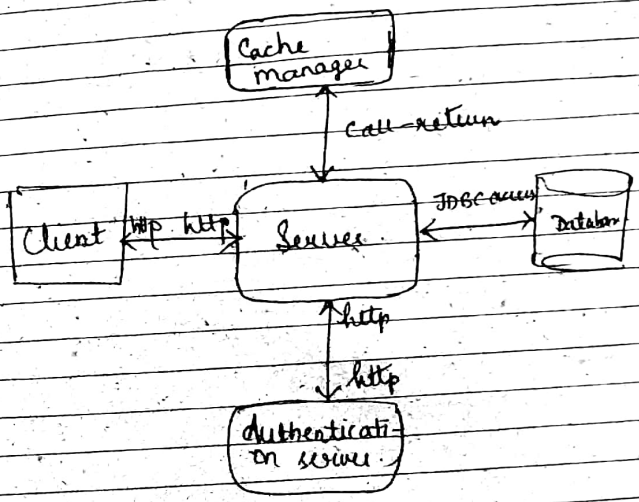


Architecture of the survey system

→ To incorporate authentication we use cookies.



→ To make the system more reliable cache component is used.



15/2/19

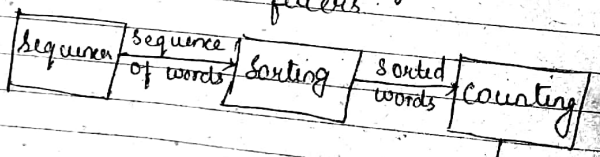
Architecture styles for C&C View

① Pipe and filter
② shared datastyle
③ client/server
④ Publish & subscribe style
⑤ Peer to peer
⑥ Communicating process style.

① Pipe and filter:

→ Pipe and filter is used when we do transformation.
- Pipe acts as connector, connect one filter to another.
- I/P of the filter can have more than one pipes.
- Filter is used for transformation.

Eg:- count the frequency of different words in a file.
→ Arrange the words in sequence
→ Then do sorting.
→ Then we will do counting.
So we have three filters.

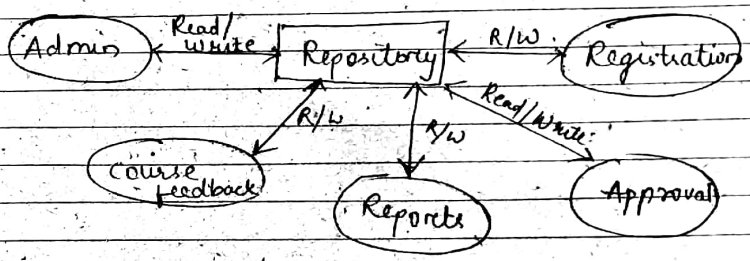


② Shared datastyle:

→ Here we have two components:-
ⓐ Repository ⓑ Data accessers.
↓ ↓
We will store data in repository. For data access from the repository.

Connector used here is read/write

Eg:- Student registration system.
① Repository → all the student info will be stored here.
② Admin → is an accessor which can read or write the data.



③ client/server ~~model~~ style

→ There will be a client and a server.
→ client can send a request to a ~~client~~ server and then server respond to a client.

Eg:- 3-tire architecture.

Middle tire process the data of the client.

Read on your own (n-tire architecture).

④ Publish - Subscribe

Two components ① Subscribe.
↘ it will subscribe to set of events.
② Publish.
↳ it will generate events.

Eg:- click of an mouse.
↘ event.
Then associated actions will be then executed (Subscribe).

⑤ Peer to peer style.

→ System can act as a client or server.
→ Object oriented ~~style~~ style.
↳ (comm. b/w objects through methods)

⑥ Communicating process style

Components → ① processes ② threads.

→ In OS data communicate by shared memory, etc.

Discussions:

① Architecture and Design.

→ Architecture is a high-level design.
→ Architecture and design are related

② Preserving the integrity of an architecture:
→ Specify all the Rules in the architecture that can be used in further steps.

③ Deployment view and performance analysis.
↓
→ How to allocate resources.
Then we used it to analyze the performance

④ Documenting architecture design.

→ What are the things should be in document?

→ Element catalog (detailed info about elements).

→ architecture rationale (Specifying why these components are used in these architecture)

→ Behaviour (what will happen in the execution steps)

→ other info. (related to future decisions)

Evaluating Architecture

↳ Archi. can impact security, performance, reliability, portability, etc. (non-functional)

→ Identify the attributes of interest for which we are evaluating arch.

→ Shown in tabular format.

→ Analysis is done. ← Based on experience subjective analysis.

There is a method for evaluation :-

Architectural trade of analysis method (ATAM)

① Collect scenarios
↳ interaction of system
② collect requirements or constraints.
③ Describe architectural views.
④ Attribute specific analysis.
⑤ Identify sensitivities and trade-offs.
(Here we determine the sensitivity point and then we see the impact of each attribute on architecture).

Eg:- Evaluation & Analysis of railway system (Read on your own).

—x—

Tutorial

Case Study 1:-

Course Scheduling

Problem Description

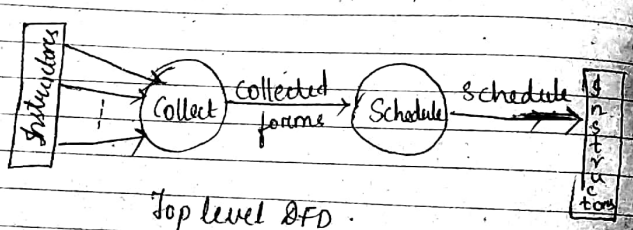
Currently the scheduling of courses is done manually but the department would like to automate it.

These courses are scheduled based on some policy of directions of the dept.

Problem Analysis

Parties involved :-
Client :- Chairman of the CSE dept.
End Users :- Dept. secretary and instructors

The system operates as follows :-
① Each instructor specifies the course he is teaching, expected enrollment and his preferences for lecture times.
② The sheets are then given to the dept. secretary who keep them in the order they are received.
③ After the deadline expires, the secretary does the scheduling. Copies of the final schedule are sent to the instructors.



Top level DFD.

④ The DFD was discussed with the chairman and the department secretary and approved by them.

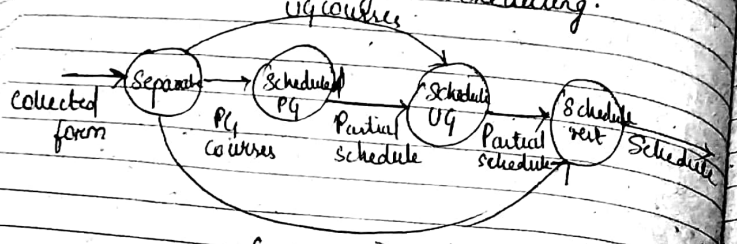
Scheduling process

→ From the chairman, the two major policies regarding scheduling are found out :-
(1) the PG courses are given more preference over UG courses.
(2) no two PG courses can be scheduled at the same time.

→ The department secretary was interviewed at length to find out the details of the scheduling activity.
→ The schedules of the last few semesters, together with their respective inputs were also studied.

Basic process is as follows :-
① The sheets are separated into 3 piles :- one for PG courses, one for UG courses and one for courses with no pref.
② The order of the sheets in the piles are maintained
③ First the course of PG file were scheduled then the UG file.
④ The courses were scheduled in order they appeared in the file.
⑤ During scheduling no backtracking was done.
⑥ Courses without preference are scheduled in the available slots.

It was found that info about classrooms and the dept-approved lecture time was tactly used during the scheduling.



The DFD for the scheduled process.

However, while scheduling, the following was being checked.

1. Classroom capacity is sufficient for the course.
2. A slot for a room is never alloted to more than one course.

Two basic data flows:-

① Sheets containing preferences.
② The final schedule.

The data dictionary entry for these is:-

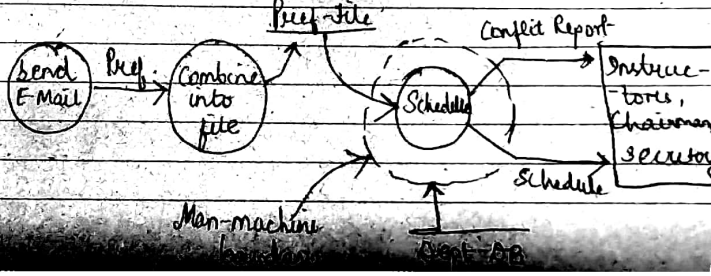
collected forms = [inst_name + course_no + [preferences]*]

schedule = [course_no class_room lecture_time]*

New system:-

① The preferences will be mailed to the secretary electronically by the instructors. The secretary will put the files separately in diff files and in order they are received. Secretary will also make entries for courses for which no response have been given.

② Format should be same as currently being used.
③ System should be able to check for errors.
④ The current approach of scheduling should be followed.
⑤ A reason for unschedulability should be given for the pref. that are not satisfied or for courses that are not scheduled.
⑥ Info. about dept courses, classrooms & valid lecture times will be kept in a file.



Data dictionary entry :-

prefprofile = [pref]*

pref = course_no + enrollment + [preference]*

dept-DB = [class_rooms]* + dept-course-list + [valid-lecture time]*

class-rooms = room-no + capacity

Case Study 2 :- Personal Investment Management System.

Problem Description :-

It is proposed to build a system - personal investment management system (PIMS) to help investors keep track of their investments, as well as determine the rate of returns he/she is getting on the individual investments as well as on the overall portfolio.

The system should also allow an investor to determine the net worth of the portfolios.

System description

① An investor can have multiple portfolios of investments. A portfolio can have many investments.

→ In each investment, the investor invests some money from time to time, and withdraws some fund from time to time.
The amount invested/withdrawn and the dates are provided by the investor.

→ There is a current value of each investment. Provision should be made to get the current value from some recognized site on web.

→ An investor may also invest in instruments having maturity date and/or fixed rate of return. The system should handle such investments and the system should provide a provision of alerts.

→ The investor should be allow to save info about its portfolio.

→ The investor should be allowed to edit entered data.

→ The investor should be allowed to view any of his portfolio.

→ Data being stored is very personal, so system should provide some security.

→ For each investment, the investor can determine the rate of return on each investment. The investor should be given the overall rate of return for each portfolio as well as total investments.

→ Rate of returns can be computed on monthly basis.

(yearly return = (1 + monthly return)^12 - 1)

Scope

→ Managing investment of single user, which would include maintaining bookkeeping info about the entities like Portfolio, Security and Transc.

→ Computation of Net Worth and Rate of Investment (ROI) of the investor.

→ Giving alerts to the user.

→ Downloading the current prices of share from the Web.

→ User authentication.

Out of scope

→ Features of actual purchasing and selling of securities.

→ Tax computation for gains or loss

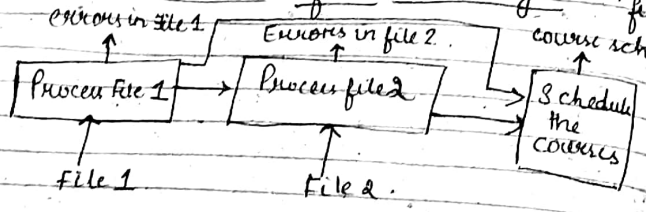
→ Any market related prediction.

Key Use Cases.

Main actors → user and the system.

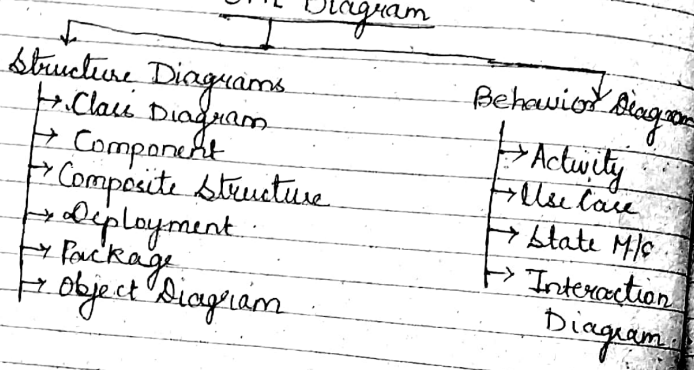
Use Case category	Use Cases.
① Installation	① Installation
② System authorization	② Login, Change password
③ Portfolio related	③ Create portfolio, Rename portfolio, delete portfolio
④ Securities related	④ Create security, Rename Security and delete security
⑤ Transaction related	⑤ Add trans, edit trans and delete trans.
⑥ Info. display	⑥ Display investment, display portfolio and display security
⑦ Computation	⑦ compute net-worth, Compute ROI
⑧ Share prices	⑧ Get current share price, edit share price
⑨ Alerts	⑨ set alerts, show alerts and delete alerts.

S/w architecture for Case Study 1 (Pipe & filter)



18/02/19 — X —

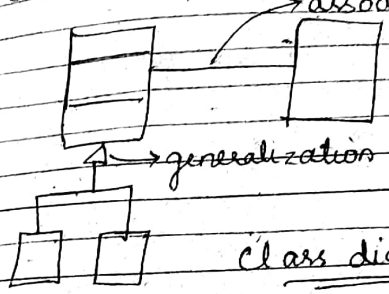
UML



UML → Unified modelling language
→ It is a blue print for your s/w.
→ Structure Diagram → used to show static structure of your diagram

→ Behaviour Diagram → dynamic behaviour

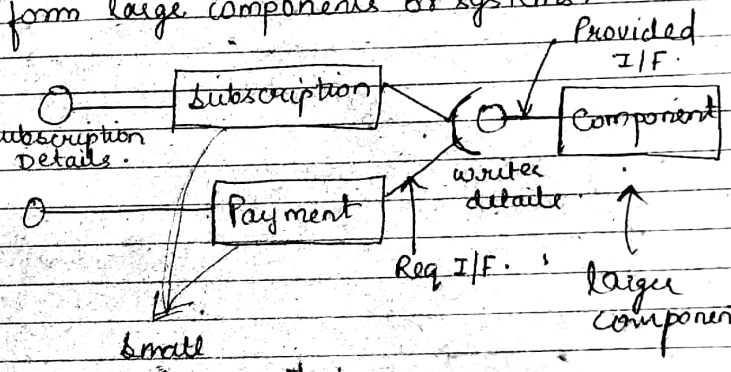
We use relationship in class diagram ie, association, inheritance



Class diagram

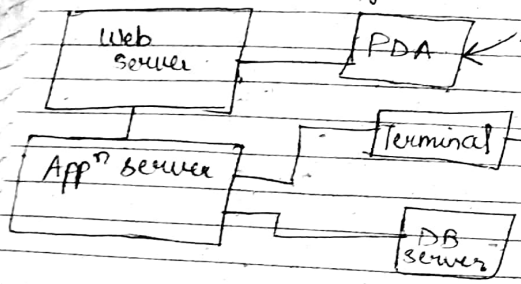
Component

→ Depicts how components are wired together to form large components or systems.



Deployment

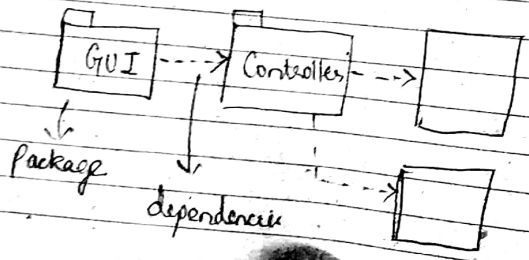
→ Helps to model the physical aspect of object oriented system.



PDA → Personal digital assistant.

Package

→ To show the packages and dependencies between them.



Composite structure

→ It is used to represent some individual diagram.

Behaviour Diagram

(i) Use case

→ used to show the functional requirements

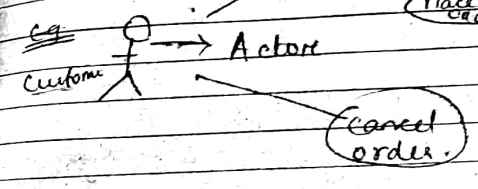
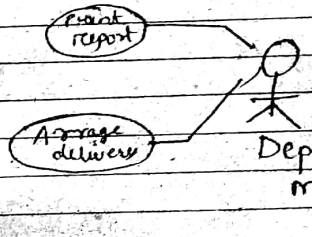


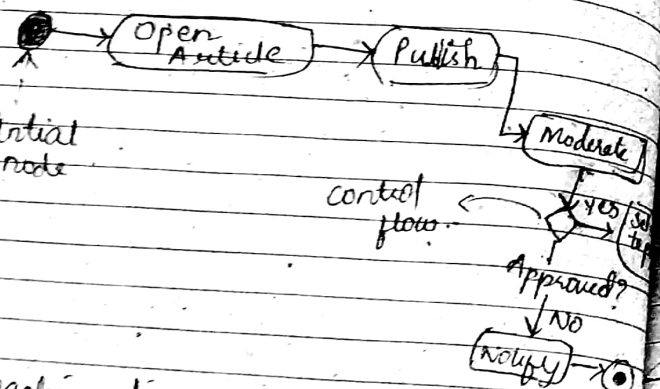
fig-2.



Activity Diagrams

→ graphical representation of work flows of stepwise activities and actions which support for choice iteration and concurrency

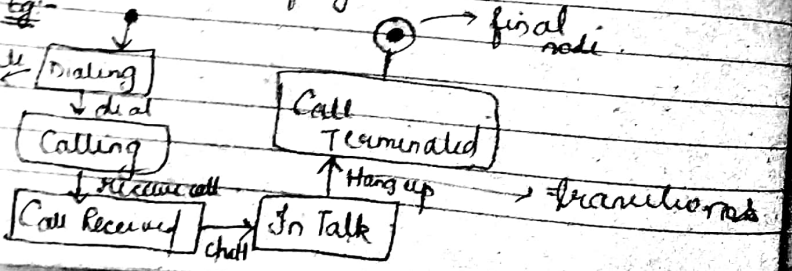
→ It describes the flow of the control of the target system.



State machine diagrams

→ To show the life cycles.

Eg:-

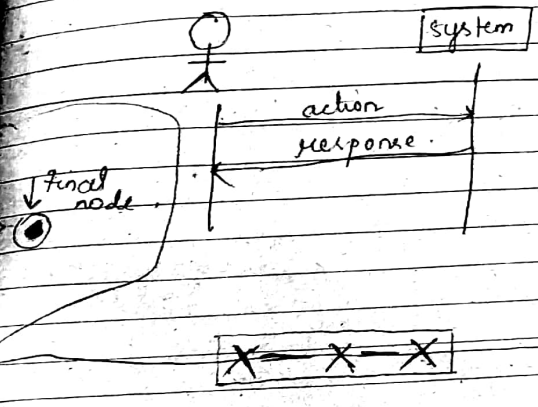


→ ∞ represents transitions

Interaction Diagram

→ Sequence diagram

→ It is used with time.



21/02/19 Planning A Software Project.

Basic goal of software project :-
is to build a s/w system to meet commit-ments on cost, schedule and quality.

Failures due to :-
→ Unclear Objective.
→ Bad planning.
→ No project management methodology
→ New technologies. (If people using it is unable to use new technologies)
→ Insufficient staff.

→ Effective project mgmt is a key to successfully execute s/w project.

→ Our main focus is on planning a project.
→ Basic objective of planning -
to create a plan, if we follow that plan to meet the commitment of the project which will lead to a successful project.

→ Cost and schedule depend on effort.

→ The output of planning is a project mgmt plan and schedule.

→ Key planning tasks :-

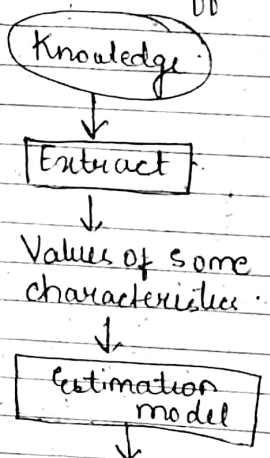
① Process planning
② Effort estimation.
③ Schedule and resource estimation.
④ Quality plan.
⑤ Configuration management plans.
⑥ Risk management.
⑦ Project monitoring plans.

① Process planning
→ Here we plan that which process to use during planning and to project making
→ Tailoring :- according to size of project we change some activities according to us.
→ Here we mention entry & exit criteria

② Effort estimation.
→ For a project total cost & duration have to be committed in the start.
→ We measure effort in terms of person-month.
→ Effort estimate is a key to planning.

→ Effort estimation can increase project quality.

Before effort measurement or estimation,
→ You should have knowledge about S/W project.
→ Extract values of some characteristics.
→ Use estimation models
→ As output we have effort estimate.



Use
→ Constructive cost model (COCOMO) for estimation model.

Effort estimation models

→ It requires some input value about the project.

Two approaches:- Top Down and Bottom Up approach.

Top Down:
Total size of the project → to estimate total effort.

Bottom up:
Determine size of sub-parts to estimate effort of each sub-part and we will use it to estimate total effort.

$Effort = a * size^{b}$
→ project size.
a, b → some constants.

Imp
COCOMO model (Constructive cost model)

→ Boehm → proposed this model.

It depend on various I/P variables.
Uses size but adjust efforts using some factors (15 factors)

Procedure :-

Step-1

Obtain initial estimate using size.

$$\text{Effort } (E_i) = a * (\text{size})^b$$

We divide the project into 3 categories.
→ Organic. (simple, st. forward, KLOC less)
→ Semi detached.
→ Embedded (complex, more KLOCs)

The value of a and b is already defined.

	a	b
Organic	3.2	1.05
Semidetached	3.0	1.12
Embedded	2.8	1.20

Step 2:- Determine a set of 15 multiplying factors from different project attributes.

(1) Requird reliability.
(2) Database size.
(3) Execution time constraint
etc. (total 15 factors are there)

Calculate EAF.

We will give rating to factors (low, very low, high, very high, etc).
These rating have some values.

EAF → effort adjustment factors.

Eg:- Factor → Reliability → Normal → 1.00
DB size → very low → 0.75

We will multiply all these values to estimate EAF.

EAF = 1.00 * 0.75 * — * — * —

Step 3:

Final effort (E) = Adjustment factor * Initial effort

$$E = EAF * E_i$$

Cost drivers	Rating				
	Very low	low	Normal	High	Very high
Product Attributes					
RELY, required reliability	0.75	0.88	1.00	1.15	1.40
DATA, database size		0.94	1.00	1.08	1.16
CPLX, product complexity	0.70	0.85	1.00	1.15	1.30
Computer Attributes					
TIME, exec. time constraint			1.00	1.11	1.30
STOR, main storage "			1.00	1.06	1.21
VITR, virtual M/c volatility		0.87	1.00	1.15	1.30
TURN, computer turnaround time		0.87	1.00	1.07	1.15
Personal Attributes					
ACAP, analyst capability	1.46	1.19	1.00	0.86	0.71
AEXP, application exp.	1.29	1.13	1.00	0.91	0.82
PCAP, programmer capability	1.42	1.17	1.00	0.86	0.70
VEXP, virtual M/C exp.	1.21	1.10	1.00	0.90	
LEXP, prog. language exp.	1.14	1.07	1.00	0.95	
Project Attributes					
MODP, modern prog. practices	1.24	1.10	1.00	0.91	0.82
TOOL, use of SW tools	1.24	1.10	1.00	0.91	0.83
SCHED, development schedule	1.23	1.08	1.00	1.04	1.10

22/02/19

COCOMO

① Basic COCOMO — It is used for rough calc. for your effort (E) and time (Tdev)

② Intermediate COCOMO → The model which we have studied is intermediate COCOMO.

③ Complete / Detailed COCOMO.

→ this model is insufficient due to absense of these 15 factors.

Basic COCOMO:

$$E = a \times (size)^b \quad ; \quad T_{dev} = c \times (E)^d$$

↑ KLOC.

For this model, a and b value varies

model can be of 3 types

→ Organic → Semi detatched → Embedded.

	a	b	c	d
Organic	2.4	1.05	2.5	0.38
Semidetatched	3.0	1.12	"	0.35
Embedded	3.6	1.20	"	0.32

a → multiplying factor

b → exponential

Assume that a size of an organic type s/w product has been estimated to be 32000 lines of source code.
Assume that the avg salary of a s/w developer is rate 15000 per month.
Determine the effort required to develop the s/w product.
The nominal develop time & the cost of development. (Using basic COCOMO)

$E = 2.4 \times (32)^{1.05} = 91.33 \approx 91$ PM

$T_{dev} = 2.5 \times (91)^{0.38}$
$= 13.88 \approx 14$ months

Cost = E x salary x T_{dev}
$= 91 \times 14 \times 15000$
$= ₹ 1911,0000$

For small softwares, (8 KLOC)

	Intermediate s/w
Product design → 16%	16%
Detailed design → 26%	25%
C & UT → 42%	40%
IT → 16%	19%

Suppose a system for office automation has to be designed.
Here we are using different modules :-
1) Data Entry — 0.6 KLOC
2) Data Update — 0.6 KLOC
3) Query → 0.8 KLOC
4) Report → 1.0 KLOC
3.0 KLOC

Factors :-
Complexity — High — 1.15
Storage High — 1.06 (Organic)
Experience low — 1.13
Prog. Capability low — 1.17

$E_i = (3.2) * (3)^{1.05}$
$= 10.14$ PM

EAF = 1.15 * 1.06 * 1.13 * 1.17 = 1.611

$E = E_i * EAF$
$= 10.14 * 1.61$
$= 16.32 \approx 16$ PM

As it is a small s/w, we can calculate person required for each step.

PD - 16% X 16 = 2.56 PM ≈ 3
DD - 26% X 16 = 4.16 PM ≈ 4
C&UT - 42% X 16 = 6.72 PM ≈ 7
IT - 16% X 16 = 2.56 ≈ 3

16 PM

The disadv. of basic and intermediate cocomo is that it assume the who project is homogeneous and take whole project into consideration.

So we move on to complete cocomo.

Here we can have many model having different structure like 1 model may be organic, one may be semi-detatched and one may be embedded

→ We will calculate efforts of each of these model separately & will combine them to get final results (effort)