Unit - 03

Software Design

- किसी software को बनाने से पहले उसके concept, planing और structure बनाने के process को software design कहते है।
- यह user requirement (SRS) को design plan में transform करता है जो module को specify, module के data structure और interface बनाने के लिए guideline provide करता है।
- यह phase SRS को as a input लेता है जिसका output SDD (Software Design Document Description)

SRS ⇨ [Design] ⇨ SDD

- यह final product के लिए blueprint का काम करता है।
- Designing process एक creative process है जिसमें हम actually यह decide करते है कि problem कैसे solve होगा।
- इस phase में requirement के आधार पर design document बनाया जाता है।
- इसका use करके programming code को write / implement कर सकते है।

level of Software Design (Step) -

(1) Architectural Design
(2) High level Design
(3) Detail Design

(1) Architectural Design - (1) Software के complete structure को बताता है।
(2) Software के component एक दूसरे module से कैसे interact करेंगे।
(3) Different different component एक दूसरे से कैसे communicate करेंगे।
(4) Data share या information share कैसे होगा।

(2) High level Design - (1) इसको design करने के लिए structure chart और ~~उस~~ technique का use किया जाता है।
(2) Software को different different module में divide किया जाता है।
(3) module के बीच relationship को identify करना।
(4) Decide.

③ Detail Design - ① यह design का last step है
② इसमें data structure और algorithm design किया जाता है।
③ इसमें module data share या exchange कैसे करेंगे इसके बारे में बताता है।

Design Principle / Concept

SDLC में software design एक important phase है। यह software process की complexity को effective way में handle करता है जिसमें की design effort और cost कम से जाते और design के दौरान आने वाले errors कम हो जाते हैं।
Software design के निम्नलिखित principle हैं -

① Problem Partitioning - इस complex और बड़े problem को छोटे-छोटे part में बांटकर किया जाता है।

② Abstraction - यह दो type के होते हैं
① Functional
① यह सब program (function) का group होता है
group के अंदर जितने visible या hidden होते हैं। जो visible components होते हैं इसको दूसरे subprogram में use कर सकते हैं जबकि hidden को नहीं use कर सकते हैं।

② Data - इसमें data element की details को user से hide रखा जाता है

③ Architecture - यह software के structure के बारे में बताता है

④ Modularity - इसमें problem को छोटे-छोटे module में बांटकर किया जाता है।
इसके अंतर्गत दो concept आते हैं।
① Cohesion ② Coupling

⑤ Information Hiding - ऐसे इसमें module को design किया जाता है कि module के अंदर का data structure और algorithm दूसरे module को accessible न हो।
Unnecessary information या details को hide करना information hiding है।

⑥ Functional Independance – यह principle यह बताता है कि function और module कैसे interrelated होगें। इसके लिए cohesion और coupling का use किया जाता है।

Coupling – system के अंदर 'module एक दूसरे से कैसे dependent होगें यह बताता है। एक good software में coupling low होनी चाहिए।

Cohesion – module के अंदर function कैसे related है इसके बारे में बताता है। एक अच्छे software में cohesion high होनी चाहिए।

⑦ Refinement – यह एक Top down design approach है।
यह एक elaboration process है इसमें एक hierarchy बनाया जाता है जिसमें basic और description होता है।

⑧ Refactoring – यह एक ऐसा process है जिसमें external design में (behaviour को) बीना change किए internal errors, redundancy, unused design आदि को सुधारना।

Design process | Design Quality

- Design Process – ① यह एक iterative process है
② इसमें software design करने के लिए requirements को blueprint में translate किया जाता है।
③ Design process के लिए निम्नलिखित characteristics follow होनी चाहिए –
(i) Design process सभी requirements को contain करके रखे।
(ii) यह readable understandable होना चाहिए।
(iii) यह software का एक complete picture होना चाहिए।

- Design Quality – ① यह किसी problem design architecture को follow करे
② Design modular होना चाहिए।
③ Design, data, architecture, interface और components को अलग-अलग representation करना चाहिए।
④ Appropriate data structure algorithm का use करना चाहिए।
⑤ Functional Independance characteristics को follow करना चाहिए।

⑥ complexity को abstract करने के लिए procedural/modular approach का use करना चाहिए।

⑦ effective communication, notation का use करना चाहिए।

Cohesion - Cohesion measurement है जो कि एक module के अंदर elements/ function कि interdependency कि degree को define करता है।

- Cohesion में यह देखा जाता है कि एक module के elements एक दूसरे से कितने करीबी से related है। अर्थात् module के elements के relationship एक दूसरे से कितना गहरा है।
- Software design में cohesion जितना ज्यादा होगा software design उतना ही अच्छा होगा।

Types of Cohesion

① Coincidental Cohesion - यह cohesion सबसे low level का cohesion है, यह तब occure होता है जब module के elements के बीच इसका कोई relationship नहीं होता।

② Logical cohesion - यह module में तब occure होता है जब module के elements के मध्य logical relationship होती है। तथा element एक ही logical level में रखे गये function को perform करते है।

③ Temporal cohesion - यह logical cohesion की तरह होता है इसमें element एक ही time पर execute होने चाहिए। परन्तु, operation तभी perform होता है जब कोई एक specific behaviour या state detect होता है।

④ Procedural Cohesion - जब module के element एक group में होते है और किसी task को complete करने के लिए step by step execute होते है। Procedural cohesion में element किसी task को execute करने के लिए group बनाते है।

⑤ Communication Cohesion - जब module के element एक ही input या output पर work करते है तो उस module को communication cohesion कहा जाता है।
concept

⑥ Sequential Cohesion - इसमे भी module के elements group मे होते है। इसमे एक element का output दूसरे element के input का काम करता है। मतलब operation sequence मे execute होता है।

⑦ Functional Cohesion - यह cohesion का high level होता है इसमे module के सभी element एक function को perform करने से related होते है।

Coupling - Coupling एक measurment है जो program या software के modules के बीच inter dependability के level को define करता है

- Coupling यह define करती है कि software के components एक दूसरे से किस प्रकार depend तथा interact करते है।
- Coupling जितनी कम होगी software design उतनी ही बेहतर होगी।

Types of Coupling

① Content Coupling - content coupling सबसे high level का coupling होता है। जब कोई module दूसरे module के internal machinery पर depend करता है इसका मतलब यह है कि अगर module के internal machinery मे change किया जाए तो उससे related module मे भी change का effect पड़ता है।

② Common Coupling - इसे global coupling भी कहा जाता है। यह तब घटित (occure) होता है जब समान global data, को module द्वारा share किया जाता है तो dependent module पर भी प्रभाव पड़ेगा।

③ External Coupling - यह तब घटित होता है जब operating system function, external data format, communication protocol, data capability आदि को module द्वारा share किया जाता है।

④ Control Coupling - इस प्रकार के coupling मे एक module का flow दूसरे module को control करता है तथा data को दूसरे module pass करता है।

⑤ Stamp Coupling - इस type के coupling में बहुत सारे module समान प्रकार के data structure को share करते हैं तथा इसके different - different part पर काम करते हैं।

⑥ Data Coupling - दो module के बीच data coupling तब होगी जब उनके बीच केवल data pass हो।

⑦ Routine Call Coupling - यह तब occure होता है जब एक operation दूसरे को invoke करते हैं।

DFD (Data Flow diagram) - Data flow diagram, Information system से गुजरने वाले data के flow का एक graphical representation है इसमें यह पता चलता है कि -

① Data का flow
② Data कहा से आया है।
③ Data कहा जायेगा।
④ Data कैसे store होगा के बारे में पता चलता है।

- DFD का use software system के objectives को design करने के लिए किया जाता है।
- DFD input, data flow, output तथा stored data को diagram मतलब graphical form में प्रस्तुत करता है लेकिन इसके process के बारे में कुछ नहीं बताता।

Types of DFD

① Logical - logical DFD यह system के data flow तथा system के process पर focus करता है। यह business activities पर focus रहता है।

② Physical - यह data flow system किस प्रकार implement हुआ है इस पर focused रहता है।

Component of DFD → 2nd unit में है।

Difference between Coupling & cohesion

Mapping data flow into Software architecture

Architecture structure design data flow diagram का software architecture में implement करने का उचित तरीका है। इसमे DFD से s/w architecture में implement किया जाता है।

- Design (mapping) process में information का flow निम्न तरीके से होता है।

① Transaction flow - इसमे information single data item से many paths में जाता है। single data item के द्वारा information flow को characterise किया जाता है।

② Transform flow - इसमे information flow sequential order में होता है। information flow को निम्न तरह से divide किया जा सकता है -

① Incoming flow - यह path external data को internal form में transform करता है।

② Transform center - Incoming data transform center से pass होते हुए output को जाता है

③ Outgoing flow - यह path data को software से बाहर भेजता है।

Steps -

① Review the fundamental System Model - इस step में software 0 level DFD में बनाया जाता है 0 level DFD में simple input process output को दिखाया जाता है। इसे context level diagram भी कहा जाता है।

② Review & Refine Data flow for the Software - इस step में context level diagram (0 level DFD) से 1 level DFD transform किया जाता है। इसका मतलब information flow को details में दिखाया जाता है।

③ Determine whether DFD has Transform or Transaction flow characteristics.

इस step में यह check करते है कि data item से information flow कैसा हो रहा है।

- अगर single data item से multiple path में information flow हो रहा है तो वह transaction information flow characteristic को specify करता है।
- अगर information flow dispersal अवस्था में है तो वह transform information flow characteristic को identify करता है।

(4) Isolate The Boundries - इस step में input, output, process सबका boundaries isolate किया जाता है।

(5) Perform first level factoring - इस step में कौन-कौन से input, output है उसको identify किया जाता है।

(6) Perform second level factoring - इस step में input, output के type को define किया जाता है।

(7) Refine The first Iteration Program Structure Using Design Heuristic.

यह step में structure को बिना किसी difficulty, बिना confusion के testing के बाद implement किया जाता है।

Interface Design

Interface design का उद्देश्य एक easy comfortable interface prepare करना है। जिसे user को smooth और enjoyable experience मिले।

- interface design में हम various element जैसे कि - layout, colour, typography, icon, button और navigation flow को user friendly interface और application बनाने के लिए हम user research, usability testing और ~~bug~~ debugging का use करते है
- एक अच्छे interface design के लिए निम्न factor पर focus करना चाहिए -

(1) Simplicity
(2) Consistency

③ User Centric
④ Accessibility
⑤ Visualization, visual aesthetic

Golden rules for interface design

① Strive for Consistency

② Enable frequent users to use shortcut

③ Offer Informative feedback

④ Design dialog to yield closure

⑤ Offer simple error handling

⑥ Permit easy reversal of action

⑦ Support internal locus of control

⑧ Reduce short term memory load

rule golden ओं main 3 point

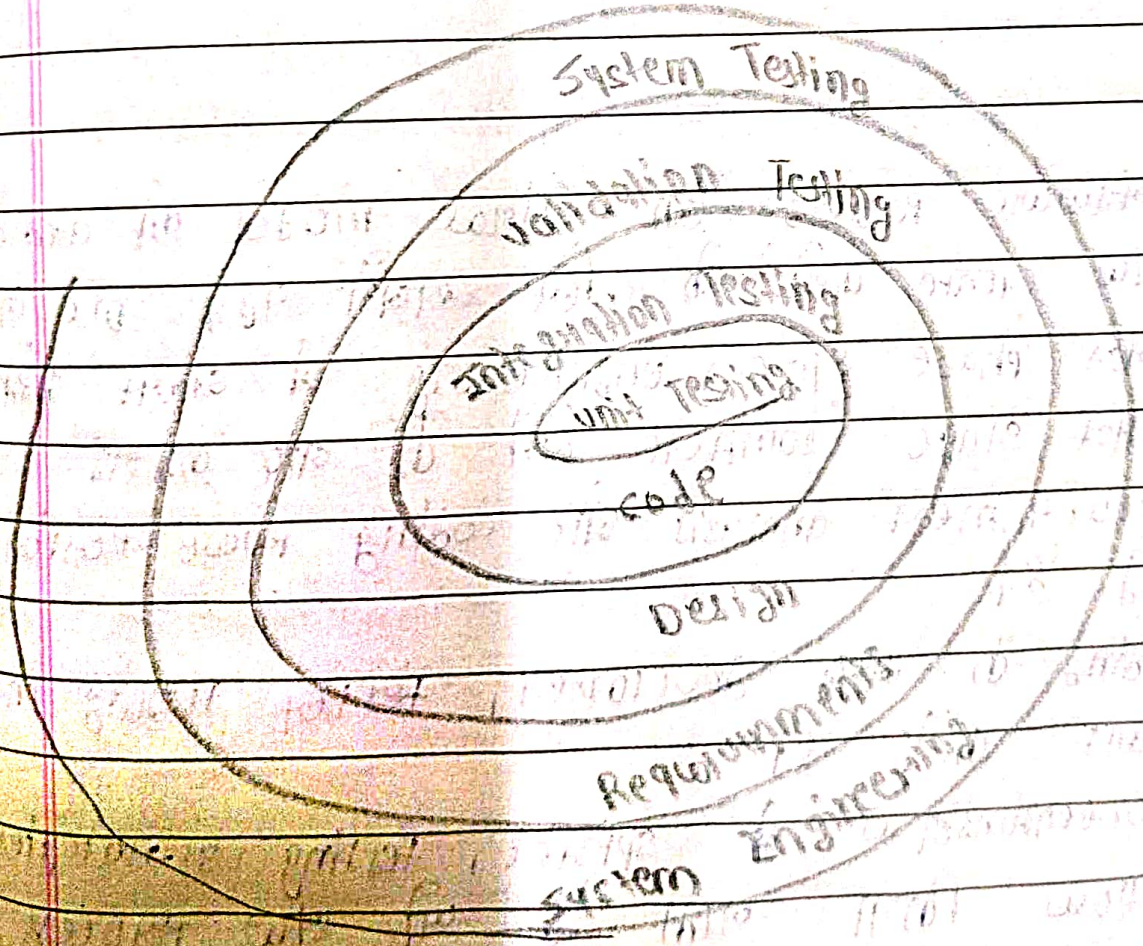
① Place the user in control
② Reduse user's memory load
③ Make the Interface consistent

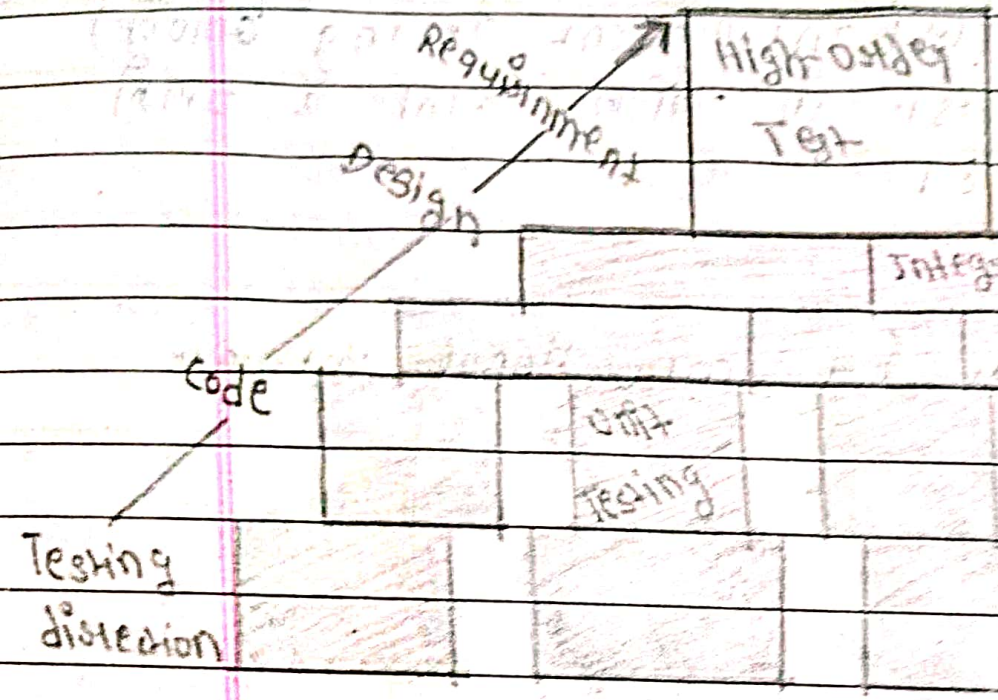
Strategic approach to Software Testing

- testing, set of activity होता है जो systematically और advance planning के साथ conduct कराया जाता है।
- s/w testing के लिए एक template follow किया जाता है। जो set of step होता है।
- s/w testing approach के लिए बहुत सारे template define किये गये है जिसमे से developer/tester उसको follow करके testing करता है।
- testing gradually low level से high level मतलब unit testing से लेकर system testing तक perform किया जाता है।
- testing अलग-अलग developer जो एक guidance provide करता है जिसको follow करके testing किया जाता है।
- testing के साथ-साथ verification और validation भी perform किया जाता है।
- इसके बाद s/w testing organize किया जाता है।
- developer individual unit (component) testing के लिए जिम्मेदार होता है। और यह ensure करता है कि सभी function सही से काम कर रहे है या नही।

इसके बाद ITG (Independent Testing Group) problem को test के दौरान आता है उसको remove करता है।

Testing strategies for conventional software





- Software process एक optimal model की तरह होता है जो software create करने के लिए सबसे पहले SDLC का first phase system engineering से start करते हैं
- First stage complete होने के बाद क्रमशः requirement analysis और coding phase perform करते हैं।
- coding के बाद developer, tester testing phase start करते हैं।
- conventional sw में optimal, testing strategies follow किया जाता है जो s/w process के opposite से start लेकर system testing तक optimal way में testing किया जाता है।
- इसमें testing optimal के vortex से होता है से start लेता है जिसमे सबसे पहला unit

testing (अंदर से बाहर), integration testing, validation testing और system testing तक perform किया जाता है।

- unit testing spiral के vortex से start होता है जो प्रत्येक unit के code को test पर focus करता है। unit testing में coding पर focus किया जाता है, testing complete होने के बाद प्रत्येक unit integrate करके integration testing perform किया जाता है। इसमे design पर focus किया जाता है। integration testing के लिए test case design techniques का use किया जाता है।
- validation testing → validation testing evaluate requirement fulfill हो रहा है या नही इस पर focus करता है। इसमे यह check किया जाता है कि हमारा software सही से काम कर रहा है या नही।
- system testing → यह सभी element properly काम कर रहे है या नही, system performance/function, achieve किया है या नही इस पर focus करता है।
- overall देखा जाए तो यह strategies मुख्यतः चार बिन्दुओं पर sequential order मे implement किया जा सकता है।

Testing strategy for object oriented software

- Testing का objective maximum possible number of errors को find out करके दूर करना।
- Object oriented s/w में भी testing के वही concept को follow किया जाता है।
- Object oriented s/w के concept में निम्न testing किया जाता है।

① Unit testing in object oriented context -

- इसमें unit testing conventional unit testing से अलग होता है। इसमें classes के ऊपर object, instance, attribute आदि पर focus किया जाता है। unit testing के लिए generally encapsulated classes के ऊपर focus किया जाता है।
- unit testing में अगर dependent classes का प्रयोग होता है तो प्रत्येक dependent classes के ऊपर के object i.e attribute आदि का separate test किया जाता है।

② Integration -

- एक class का एक mime में integration testing करना difficult होता है इसलिए integration testing के लिए दो अलग-अलग strategy का use किया जाता है

① Thread based testing ② Use based testing

Integration testing में सबसे पहले independent classes का test किया जाता है उसके बाद next higher class जो दूसरे class पर depend होते है उनको test किया जाता है। testing का यह sequence तब तक चलता रहता है जब तक complete system ना बन जाये।

③ Validation testing
④ System testing
} पहले वाला का लिखना है।

Validation Testing → definition लिखना है और पहले है

- Validation testing integration testing के बाद किया जाता है। इसके अंतर्गत यह check किया जाता है कि s/w requirement specification (SRS) के अनुसार system work कर रहा है या नहीं।
- इसके अंतर्गत ने.लि. approach follow किया जाता है।

① Validation test criteria
② Configuration review
③ Alpha & Beta testing

① Validation test criteria –

- इसमें test plan और test procedure define किया जाता है। test plan के अंतर्गत conducted test का class तैयार किया जाता है। और test procedure specific test case को बताता है।
- test case conduct करने के बाद दो condition प्राप्त होता है –

(i) performance characteristics जो specification में confirm किया गया था।
(ii) specification में जो भी deficiency प्राप्त हुआ था उसका list

2) Configuration Review - इसमे s/w के सभी element properly develop हुआ है या नही check करते है configuration के अनुसार

3) Alpha and Beta Testing - (i) Alpha - alpha test developer द्वारा conduct कराया जाता है। alpha test control environment मे conduct कराया जाता है। इसमे internal user का site (developer और company के अंदर) या selected group को bug और usability issue को check किया जाता है।

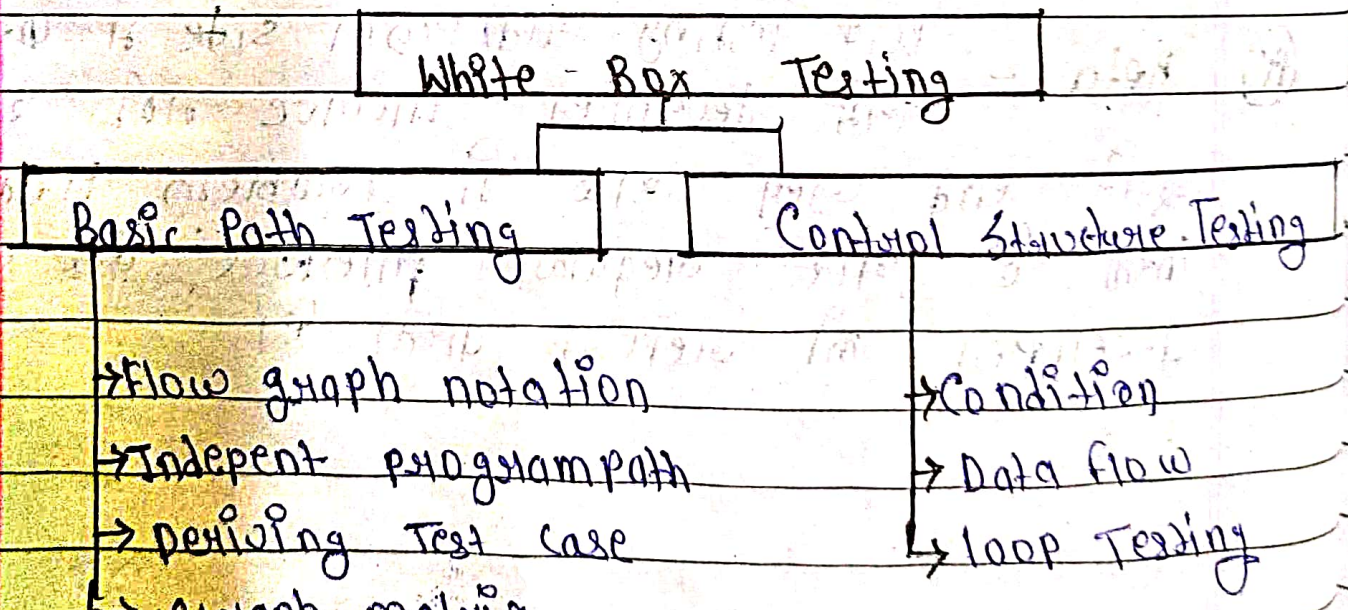
(ii) Beta - Beta testing end user side मे कराया जाता है इसमे developer involve नही रहता, इसमे end user s/w मे problem find-out करता है और feedback interval way मे developer को response करता है।

System Testing → definition लिखना है यहां पर

Types System Testing

1. Recovering Testing –
2. Stress
3. Security
4. performance

White Box Testing



- इसको glass box testing, structure box testing के name से जाना जाता है।
- इसका use test case design जो control structure का use करता है। को test करने के लिए किया जाता है। यह developer द्वारा किया जाता है।
- इसमे Internal structure Knowledge होना है
White box testing के द्वारा s/w engineer test case derive करता है जो निम्न चीजो पर focus करता है-

(i) सभी independent path एक बार cover हुआ हो
(ii) सभी logical decision के true & false case
(iii) loop और इसके boundary condition
(iv) Internal data structure की validity

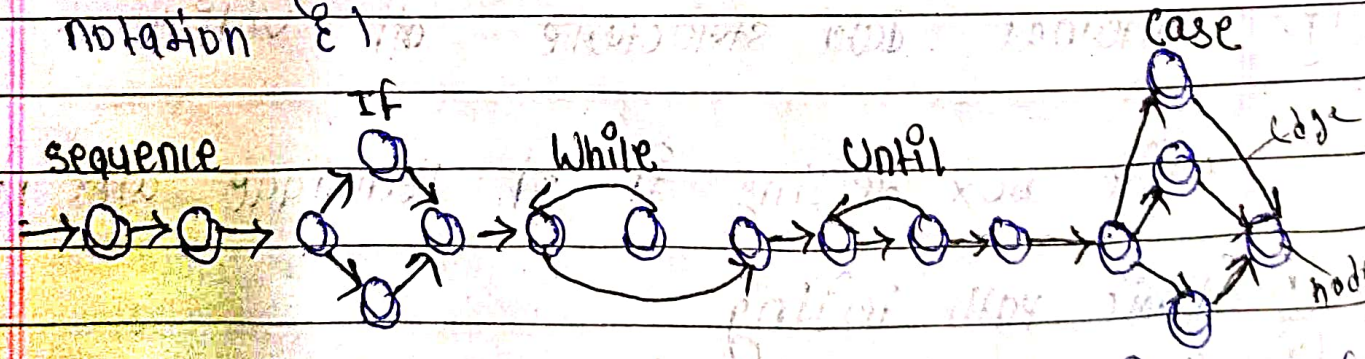
White box testing मे दो technique use किया जाता है-

1. Basic path testing
2. Control structure testing

① **Basic Path Testing** - यह white box testing का एक technique है जो test case designer को procedure design का logical complexity measure करने मे help करता है और इस measure का use set of execution path को define करने मे किया जाता है।

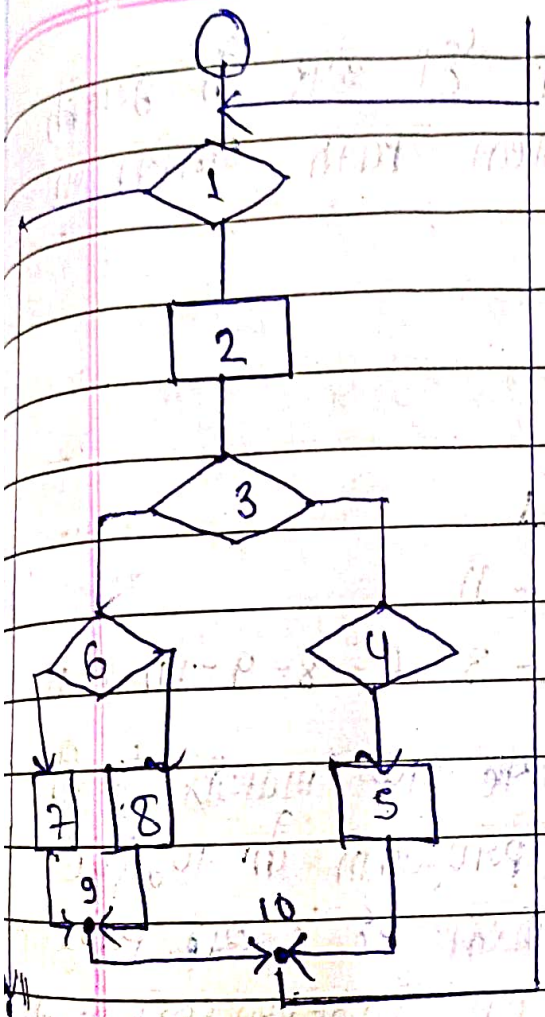
- Basic path testing को cyclomatic complexity, Independent program path, test case, graph matrix आदि Flow graph notation का use करके यानि किया जाता है। इसके अंतर्गत
  (i) Flow graph notation
  (ii) Independent program path
  (iii) Deriving test case
  (iv) Graph matrix के use से test किया जाता है।

(1) Flow graph notation - Basic path testing से पहले program का flow chart व flow graph बनाया जाता है जिसके निम्न notation है।

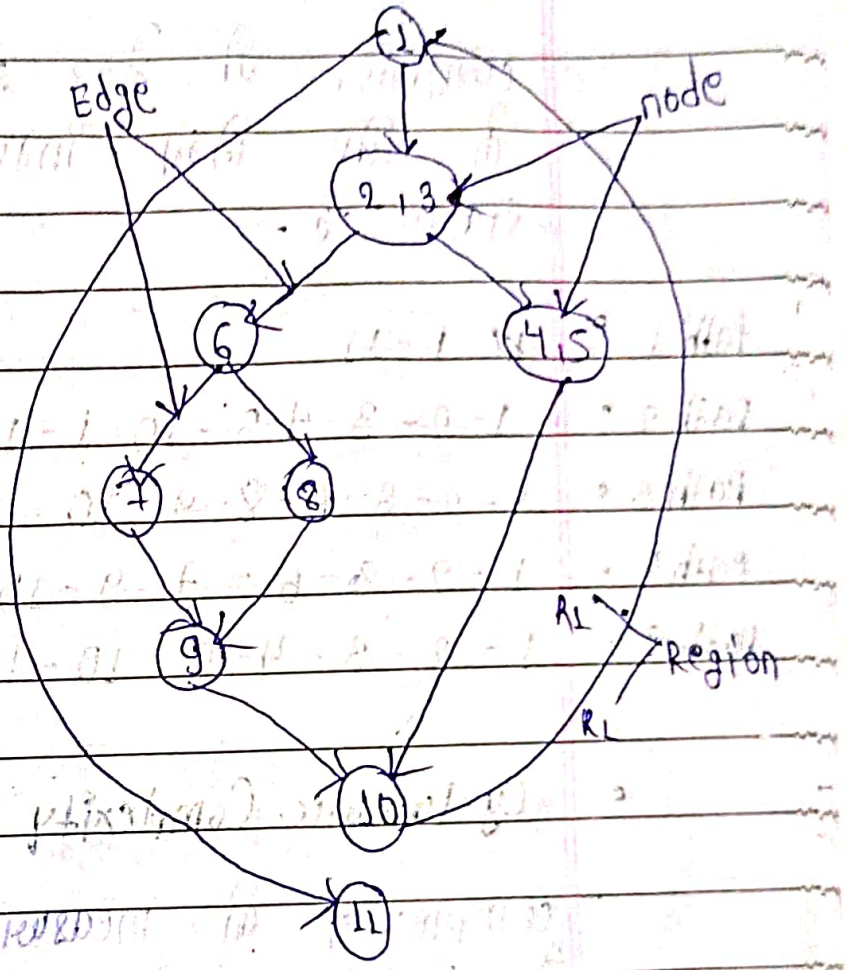


- Flow graph logical control के flow को दिखाता है निम्न flow chart से flow graph निम्न तरीके बनाया जाता है।

(a)



(b)



Flow graph में (→) link या edge को बताता है जबकि ○ state ○ node को represent करता है।

- **Region** – edge और node से closed area को region कहते हैं।

- **Predicate node** – ऐसा node जो condition को show करता है

(10) **Independent Program Path** – यह program path के द्वारा introduce किया जाता है जो कम से कम एक बार processing statement या new

condition से होकर गुजरता है। ऊपर के graph के लिए निम्न Independent Path बनाया जा सकता है -

Path 1 : 1 - 11
Path 2 : 1 - 2 - 3 - 4 - 5 - 10 - 1 - 11
Path 3 : 1 - 2 - 3 - 6 - 8 - 9 - 10 - 1 - 11
Path 4 : 1 - 2 - 3 - 6 - 7 - 9 - 10 - 1 - 11
Path 5 : 1 - 2 - 3 - 4 - 5 - 10 - 1 - 2 - 3 - 6 - 8 - 9 - 10 - 1 - 11

- **Cyclomatic Complexity** - यह S/w metrics है जो program की logical complexity को measure करता है। यह program के basis पर no. of independent path define करता है और कितने सारे test होंगे उसका upper limit बताता है।
Cyclomatic complexity निम्न तरीकों से ज्ञात किया जाता है -

(i) Cyclomatic complexity = no. of regions
(ii) Cyclomatic complexity of V(G) = E - N + 2
(iii) Cyclomatic complexity of V(G) = P + 1

ऊपर बनाए गए graph के लिए निम्न cyclomatic complexity होगा :-

(i) graph में 4 region है तो cyclomatic complexity भी 4 होगा

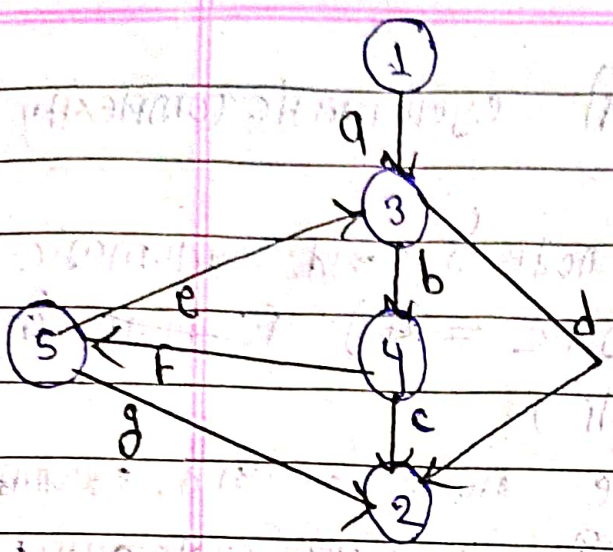
(ii) graph में 11 edge व 9 node है अतः cyclomatic formula E - N + 2 (11 - 9 + 2 = 4) ~~formula~~ के अनुसार cyclomatic complexity 4 होगा।

(iii) graph में 3 predicate node है अतः cyclomatic formula P + 1 (3 + 1 = 4) के अनुसार cyclomatic complexity 4 होगा।

(iii) **Deriving Test Case** - यह series of steps होता है इसके निम्न step होते है -

(i) design या code के अनुसार flow graph बनाया जाता है।
(ii) flow graph का cyclomatic complexity निकाला जाता है।
(iii) independent flow path निकाला जाता है।
(iv) Test case तैयार किया जाता है।

(iv) **Graph Matrix** - यह basic path testing के लिए एक data structure होता है यह square matrix होता है जिसमे row और column की संख्या no. of nodes के बराबर होती है। इसमे link weight (input) control flow को बताता है।



Flow graph

node \ Connected to node	1	2	3	4	5
1			a		
2					
3		d		b	
4		c			F
5		g	e		

Graph matrix

(2) Control Structure Testing -

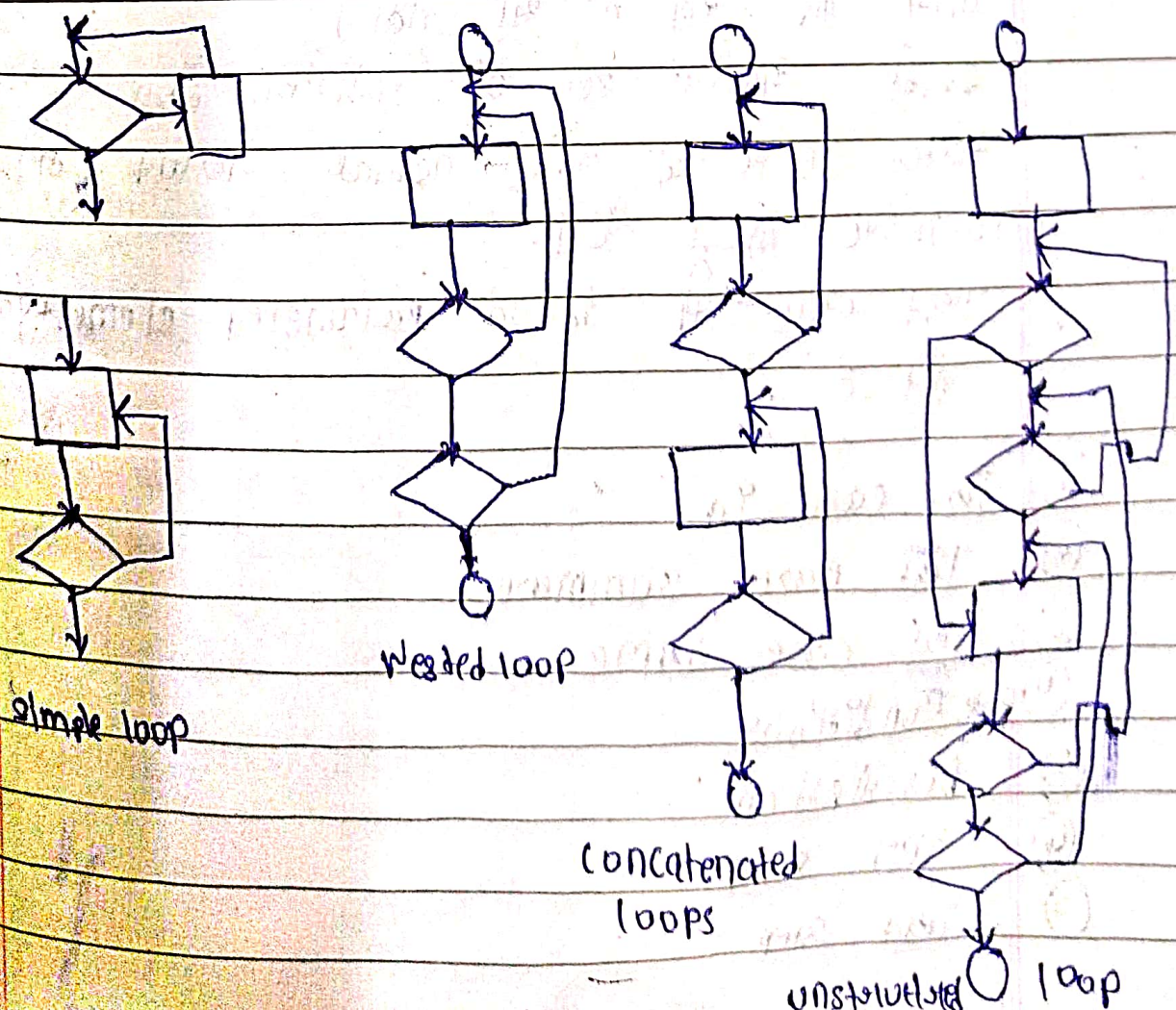
- यह भी white box testing के अंतर्गत basic path testing में आये कमियों को दूर करके testing करने का एक technique है।
- Common structure testing में test case design करने के लिए condition testing, data flow testing, loop testing आदि method से किया जाता है।

(i) Condition Testing -

- यह method test case design करने के लिए program module का logical condition को test करता है। जहाँ E1 < relational operator > E2 जहाँ E1 और E2 arithmetic expression है और relational operator में $<, \leq, =, \neq, \geq, >$ आदि का use किया जाता है।

(ii) Data Flow Testing - data flow testing test path को select करता है और उसके definition और variable के अनुसार data flow को test करता है।

(iii) Loop testing - यह white box testing का technique है जो loop construction के validity पर focus करता है, इसके लिए simple loop, nested loop, concated loop, unstructured loop आदि loop based test किया जाता है।
इन loop के लिए निम्न notation use किया जाता है।



Test Case

- Test case set of condition, input, step और expected result का set होता है।
- test case से क्या test करना है, कैसे test करना है और expected result के बारे में test किया जाता है।
- Test case operation का set होता है जिसे sw application की कार्य क्षमता को verify करने के लिए execute किया जाता है।
- इसके द्वारा tester यह निर्धारित करता है कि sw customer के requirement के अनुसार काम कर रहा है या नहीं।
- इसमे special set of condition होता है जिसे tester expected और actual result को compare करता है।
- Test case में निम्नलिखित parameter element/component होते है।

1. Test case Id
2. Test case summary
3. Test case name
4. Pre Conditions
5. Description
6. Test data
7. Test Step

8. Test Expected result
9. Test actual result
10. Test criteria
11. Test scenario
12. Test Environment

Types Of Test Case

1. Functional Test Case
2. Integration Test case
3. User Interface Test case
4. Non-functional Test case

Techniques Of Test Case

1. White Box Testing
2. Black Box Testing

Black Box Testing

- इसको Behavioral testing भी कहा जाता है। यह generally s/w के functional requirement पर focus करता है। यह tester द्वारा किया जाता है।
- Tester internal structure के बजाय input या output पर focus करता है।
- Black box testing का tester false है जबकि white box testing का tester true है।
- Black box testing यह निम्न categories में errors को find out करने की कोशिश करता है:-
  (i) Incorrect या missing function
  (ii) Interface Errors
  (iii) Data structure, database Access Errors
  (iv) Behaviour / performance Errors
  (v) Initialization / Termination Errors
- Black box testing निम्न requirement के errors को find out करने के लिए design किया गया है:-
  (i) Functional validity test
  (ii) System behaviour or performance testing
  (iii) classes of input will make good test case
  (iv) Sensitive ~~input~~ value test
  (v) Boundaries of data class test

Black Box testing के निम्नलिखित method है -

1. Graph based testing method (object)
2. Equivalence Partitioning (class)
3. Boundrie value Analysis (link/edge)
4. Orthogonal Array testing (Region)

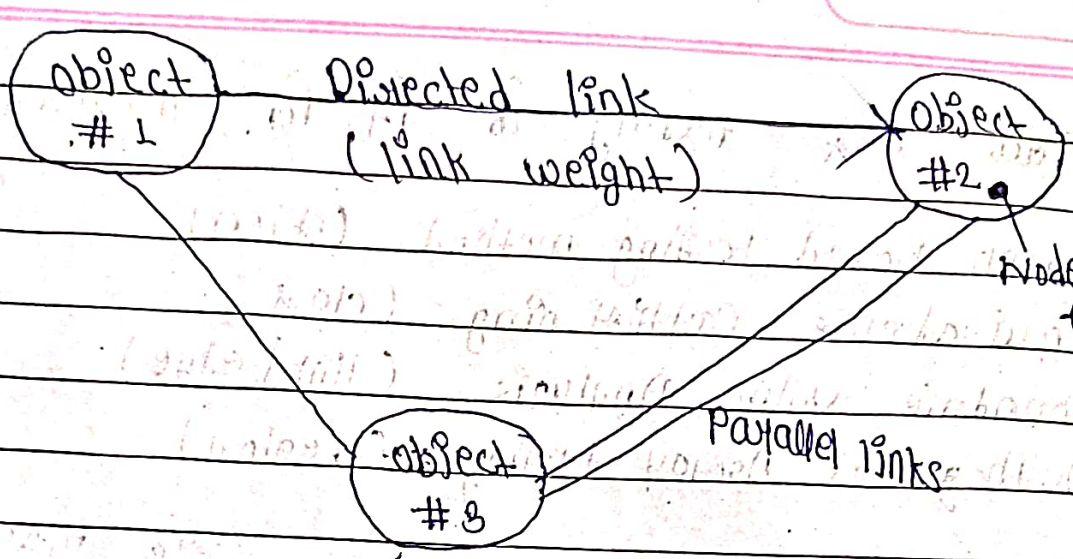
1. Graph Based Testing Method -

यह black box testing का पहला step है जिसमे s/w model के object और relationship को समझा जाता है।

इस step को करने के लिए एक graph बनाया जाता है जिसमे node object को represent करता है जबकि link object के relationship को।

इसके अलावा link weight : link के characteristics को represent करता है।

Directed link : एक direction मे move होने वाले relationship को represent करता है जबकि bidirection link दोनो दिशाओ मे relationship को represent करता है। इसके अलावा parallel link node के different different relationship को define करता है।



(2) Equivalence Partitioning :- यह भी black box method है जिसमें program के input domain को classes में divide किया जाता है और फिर से test case derive किया जाता है।
Equivalence class input condition पर depend करता है जो valid/invalid हो सकता है। Input condition specific numeric value, value का range, Boolean condition या set of related value हो सकता है।

(3) Boundary value analysis - यह भी black box technique है जिसमें हम input range के (माना कि -10 से +10) boundary value जैसे - -12, -11, -9 ..... +9, +11, +12 ... boundary value पर test case

बनाते है क्योंकि अधिक boundaries पर ही जाया होते है। Boundaries value generally class के extreme पर edge से case test derive करता है। input variable के लिए हम निम्न value लेते है.

(i) Minimum value (Min)
(ii) Minimum +1 (Min +1)
(iii) Minimum -1 (Min -1)
(iv) Max -1
(v) Max

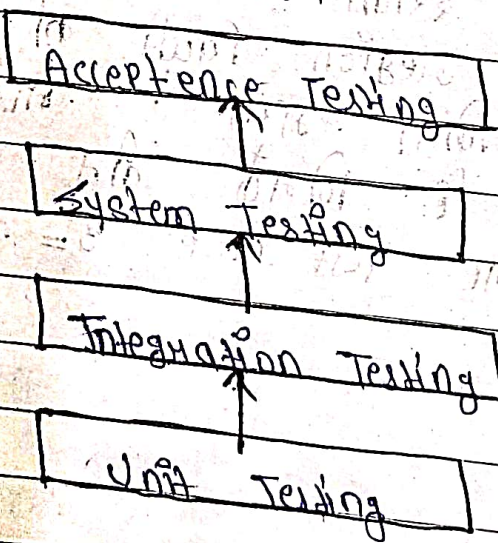
(4) Orthogonal Array Testing - यह तब apply किया जाता है जब input domain छोटा-सा होता है, पर पूरे test को मिलाकर इसका size large हो जाता है। यह generally region fault को find out करने के लिए design किया गया है। माना कि 3 input item x, y, z है जिनके तीन अलग-अलग value है तो possible test case $3^3 = 27$ test case हो सकते है।

...load के attributes के बारे में information
...लेने हैं जिन्हें

Page: / / Date:

System testing

① Complete system का check किया जाता है।
② SRS के base पर s/w product के behaviour को check किया जाता है।
③ यह black box testing का एक type है जिसमे s/w के external structure की test किया जाता है।
④ System testing, system मूल्यांकनता और functional मूल्यांकनता के आधार पर किया जाता है।
⑤ इसके मुख्य उद्देश्य end-to-end system specification को evaluate करना है।
⑥ System testing unit testing और integration testing के बाद होता है।



Importance of System Testing

1. Improved Product Quality
2. Error Reduction
3. Software Performance
4. Customer Satisfaction
5. Security
6. Cost saving

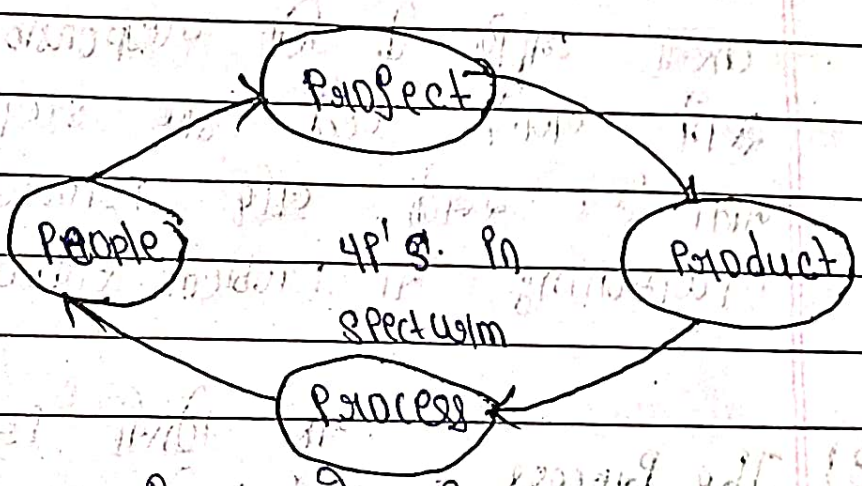
Process of Software System Testing

1. Test environment prepare करना
2. Test case ready करना
3. Test case के लिए test data तैयार करना
4. Test case को execute करना और result को analysis करना।
5. Failure condition में defect को report करना।
6. Regression test करना।
7. regression test के बाद fault को report करना
8. सभी defect को reset करना या ठीक करना

The Management Spectrum

किसी product को successful बनाने के लिए यह एक important concept है
effective s/w project management के लिए 4P पर focus किया जाता है -

1. People
2. Process
3. Product
4. Project



ये components project में important role play करते है यह team को goal और objective को achieve करने में help करते है।

① **The People** - यह management में सबसे important component है और इसका successful implementation human resource है। किसी product को बनाने के लिए एक well managed team होता है जिनका role clearly defined रहता है। इसके अंतर्गत

(i) Project manager
(ii) team leader
(iii) Software team
(iv) IT professional
(v) stackholder आदि आते है।

② The Product - Project planing के पहले Product का objective और scope define कर लेना चाहिए। यह project का अध्ययन होता है project manager successful result, team member को coordinate करना, Technical problem को दूर करना आदि के लिए responsible होता है। इसमें सबसे पहले software scope find out किया जाता है, इसके बाद problem decomposition (Partitioning या problem elaboration)

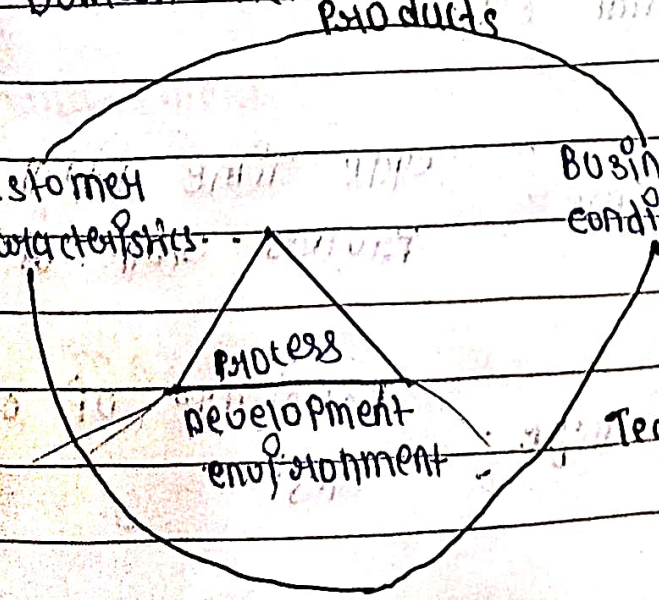
③ The Process - यह किसी product का ~~success~~ success के लिए key होता है। project manager कौन से process model सही रहेगा उसका decision लेता है। जिसमें developer, working team, product का characteristics, project का environment etc के आधार पर decision लेता है।
Process कई step का होता है जिसमे communication phase, implementation phase, Deployment phase, Integration phase आदि आता है।

(4) The Project - यह s/w management spectrum का सबसे important और final "P" है। इसे process का development भी कहा जाता है। इसमें project manager का critical role रहता है। वह team को guide करने का काम, cost और budget को check करने का काम, target और objective को achieve करने का काम, deadline आदि के लिए responsible होता है।

Metrics In The Process & Project Domain

Metrics - ~~s/w~~ development process और s/w project का quantitative measure है जिसका use Planning, Monitoring, control और Improvement के लिए किया जाता है।
Metrics को दो part में divide किया जा सकता है-

(i) Process Domain Metrics (ii) Project Domain Metrics



Process Metrics - यह metrics s/w development process कितना अच्छा और effective है उसके बारे मे analysis करता है। यह process improvement, quality enhancement और productivity analysis मे help करता है।

- इसका मुख्य उद्देश्य
  (i) Development और testing process को improve करना।
  (ii) Defect और analysis को reduce करना।
  (iii) effectiveness और quality को बढ़ाना।

- Process metrics के important - या प्रकार -

(1) Defect Density = $\frac{\text{Total Defect}}{\text{Size (FP/LOC)}}$

इसमे size के respect मे no. of defect measure किया जाता है।

(2) Cycle Time - एक time जितना लग होगा process उतना ही efficient होगा

(3) Test Coverage - Percentage of code covered by test

Date: / / Page

④ Code review efficiency — Percentage of work free from deffect

⑤ Rework Percentage = $\frac{\text{Rework effort}}{\text{total effort}} \times 100$

⑥ Defect Removal efficiency DRE = $\frac{\text{Defect Removed before Delivery}}{\text{Total Deffects}}$

Project Domain ~~Metho~~ Metrics -

यह metrics cost, schedule, effort और size के base पर project performance को measure करता है।

इसका मुख्य उद्देश्य -

i. Project planning & estimation
ii. Tracking progress
iii. Risk management
iv. Cost & Time control

Project matrix के important ~~म~~ use -

(1) Size Metrics - इसका use effort और cost के estimate करने के लिए किया जाता है। size metrics में LOC (line of code) और FP (Functional Point) का use measure करने के लिए किया जाता है।

(2) Effort Metrics - effort = person * time
जहां time या तो hour में हो सकता है या तो month में।

(3) Schedule Metrics - इसके अंतर्गत planned और actual time के बीच schedule variation measure किया जाता है।

(4) Cost Metrics - budget cost, actual cost के बीच cost variation measure किया जाता है।

(5) Productivity Metrics - Productivity = size / effort
Productivity बढ़ेगा तो performance भी बढ़ेगा।

Software Measurment :-

- यह किसी process या product के attribute के size, quantity और यह dimensional dimension के indicator है।
- यह project manager को successful completion के लिए decision लेने में help करता है।
- sw measurment के दो category है -

1. Direct
   - i. cost & Effort
   - ii. LOC
   - iii. Execution
   - iv. Total No.
2. Indirect
   - i. Functionality
   - ii. Quality
   - iii. Complexity
   - iv. Reliability
   - v. Maintainability

Sw Measurment के लिए निम्न methods use होता है-

1. Size Oriented (LOC) - इसमें quality या productivity line of code के base पर normalize करके किया जाता है।
2. Function Oriented (FP) - इसमें quality या productivity functional point के base पर normalize करके किया जाता है।

(3) Mixture of LOC & FP metrics :-

(4) Object Oriented Metrics - इसमें measurment के लिए no. of key classes, no. of support classes, no. of subsystem आदि का use किया जाता है।

(5) Use case Oriented Metrics - यह process के लिए use case define किया जाता है और उसी आधार पर यह measure किया जाता है।

यह Measurment Principal

(1) Formation
(2) collection
(3) Analysis
(4) Interpretation
(5) Feedback

Classes and Objects in Java - Classes and Objects Object-Oriented Programm...
जो real-life entities के around revolve करते हैं।

Class (कक्षा):

- Class objects का एक set होता है ...
  share करते हैं।
- Class एक re...
  जिस

Sw quality metrics का इस category में sw के quality attributes को directly measure किया जाता है।

Metrics for Software quality

- sw quality metrics process, product health आदि को measure करता है। इसमें performance, security, usability test coverage आदि parameters को use किया जाता है।
- sw quality में निम्न category हो सकते है-

(1) Measuring Quality	(2) DRE (Defect Removal Efficiency)
(i) Correctness	DRE एक important process metrics होती
(ii) Maintainability	है measure करता है कि development
(iii) Integrity	process के दौरान कितना defect detect और
(iv) Usability	remove किये गये है before sw delivery.

Software Scope feasibility & Resources

(1) Software Scope - sw scope एक process को define करने वाले function और feature का boundaries के बारे में बताता है।
- boundaries के अंतर्गत यह input/output performance, interface और maintainability आता है।
- scope को define करने के दो तरीके है -

(1) Narrative Description -
(2) UML (use case)

Component → Functional, data, use, [illegible]

- एक बार scope define होने के बाद यह पता करना होता है कि इस मे project feasible है।

(2) Feasibility - यह check करता है कि proposed s/w बनाना feasible है या नही।
s/w feasibility के 4 component है -
(i) Technology → क्या यह technical feasible है या नही
(ii) Finance → यह finance feasible है या नही
(iii) Time →
(iv) Resource →

Type of feasibility
(i) Technical
(ii) Economical
(iii) Operational
(iv) Schedule
(v) legal

(3) s/w Resources - s/w development को complete करने के लिए क्या-कितना resource की जरूरत है - इसके बारे मे पता चलता है।
resource के अंतर्गत - ~~People~~ (1)

(1) People (manager, worker, Developer, Stackholder etc)

(ii) Reusable s/w Resource
(iii) Environmental Resource (hardware + Software etc.)
(iv) Helping
(v) Finance

S/w Project Estimation

- इसमें project को complete करने के लिए time, cost, effort और resource का estimation किया जाता है।
- गलत estimation project को delay, over budget और failure का कारण हो सकता है।
- यह एक systematic process है जिसमें :-

(i) effort
(ii) Time (schedule)
(iii) cost और resource का estimation किया जाता है.

(1) Effort - कितने person-month / person-hour लगेंगे।

(2) Developer और tester का workload ⇒ Time

(3) Cost ⇒ (1) Salary (2) H/w & s/w (3) Training & maintance cost

④ Resource - (Human Resource, H/W & S/W Resource)

Techniques -

① Size based estimation < LOC / FP

② Algorithmic estimation < COCOMO / ~~Delphi~~ Putnam

③ Expert Judgement (Delphi)

④ Analogous estimation

⑤ Top : Down estimation

⑥ Bottom Up estimation

Decomposition Technique

- इसमे project को छोटे छोटे manageble part में विभाजित करके estimation किया जाता है।
- Decomposition निम्न प्रकार से किया जा सकता है -

(i) S/w के size के अनुसार (LOC/FP)
(ii) Product based या s/w decomposition
(iii) Problem या process based decomposition decomposition
(iv) Top ~~button~~ down या bottom up ~~estimation~~

Empirical Estimation Model -

- इस model में effort (LOC/FP), cost, productivity आदि को estimate करने के लिए empirically derived formula का use करते है यह formula project के sample से derive किया जाता है।
- यह model data collection के द्वारा test किया जाना चाहिए Empirical estimation में mathematical formula आदि के base पर estimation किया जाता है।
- cost estimation के लिए COCOMO model का use किया जाता है।

COCOMO Model (Constructive cost model) -

- यह model 1981 में Barry B Boehm द्वारा दिया गया।
- यह model मुख्यतः effort, cost, schedule

यह complexity, software के size (KLOC) के base पर estimation करता है।
- यह तीन type के होते है-

COCOMO
- Organic
- Semidetached
- Embedded

Project size	Small (2-50KLOC)	Medium (50-300KLOC)	Large (300KLOC)
Nature	Simple	complex	More complex

Level of COCOMO

(1) Basic COCOMO - Effort (E) = $a*(KLOC)^b$ Hm.
Schdule time (D) = $c*(E)^d$ month

(2) Intermediate (3) Detailed / Complete
Effort (E) = $a*(KLOC)^b * EAF$ mm
Schdule time (D) = $c*(E)^d$ months (m).

E = Total effort required for the project in men month

Classes and Objects in Java - Classes and Objects Object-Oriented Programm:
जो real-life entities के around revolve करते हैं।

Class (कक्षा):

- Class objects का एक set होता है → share करते हैं।
- Class एक rea
जिम्

nase → waterfall model or step लिख ना है।

Page
Date: / /

stal - time required for project development
- in month (m)
= Kilo Line of Code
+d = Constant parameter for a s/w Project
= Effort Adjustment Factor
vantage / Disadvantage

oftware Reliability
सी s/w system के specific environment में specified
ime period में failure free operation perform
करने की probability को reliability कहते हैं।

Unit-01

Software Engineering

Software engineering दो शब्दों से मिलकर बना है Software और engineering.

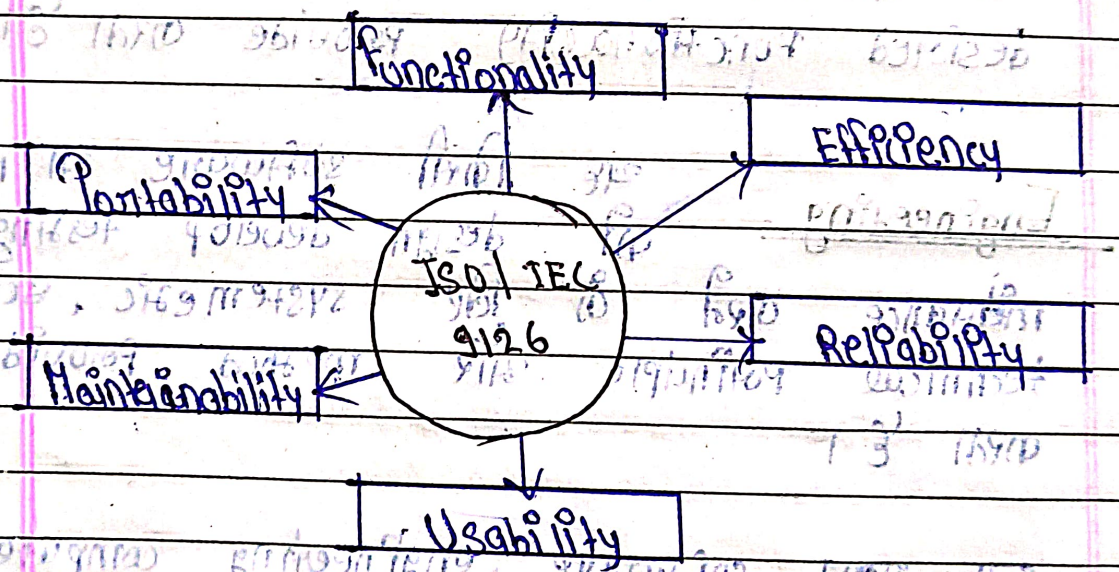
Software - Software program associated data documentation और process का set होता है। जो specific task को perform करने के लिए desired functionality provide करता है।

Engineering यह किसी software या project को design develop testing और maintance करने के लिए systematic, scientific technical principle और method provide करता है।

- इस प्रकार software engineering computer science का एक branch है जो software या project को design, develop, testing और maintance करने के लिए systematic और disciplineapproach है।
- यह high quality reliable और maintainable software develop करने में help करता है।

- Software engineering एक process है जिसमे user के requirement को analyse किया जाता है और उसके requirement के अनुसार software develop किया जाता है।

Characteristics of software



Software characteristics के 6 component होते है -

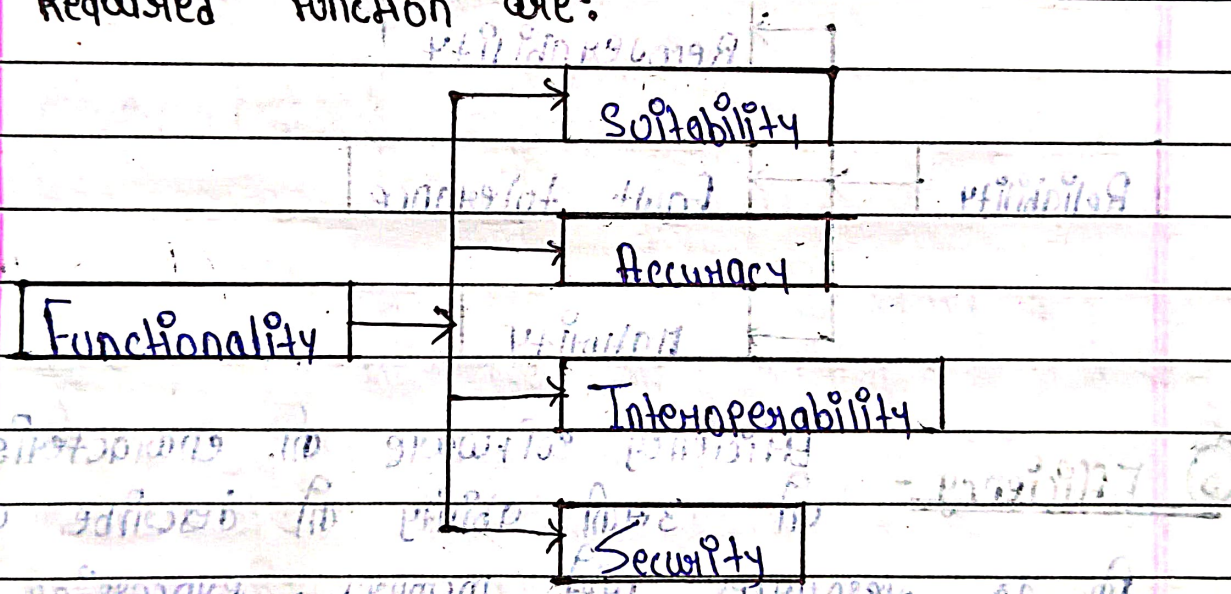
① Functionality - Functionality का मतलब है वो set of features और capabilities जो कोई software program या system अपने user को provide करता है।
यह software की सबसे important characteristics मे से

Date: / / Page

एक है. क्योंकि यही software की usefulness को निर्धारित करता है उसके intended purpose के लिए।
Software के functionality के example -

- data storage और data retrieval
- data processing और data manipulation
- security और access control
- user interface और navigation
- communication और networking

Required function are:

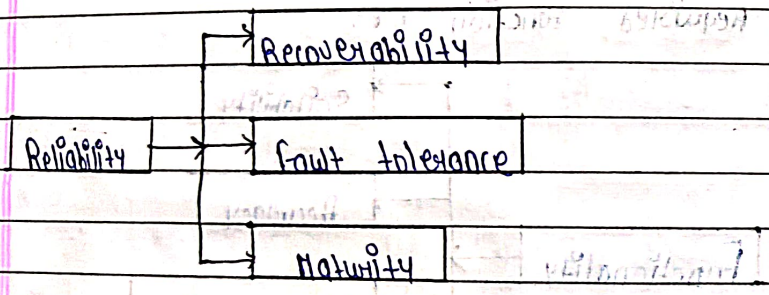


(2) **Reliability** - यह उन attribute का set होता है जो software की capability को दर्शाता है कि वह दिए गए condition में एक stated period of time तक अपना performance level maintain कर सके। Reliability software की एक characteristic है जो इसके intended functionalities को सही तरीके से और consistent रूप से time के साथ perform करने की ability को सुनिश्चित करती है।

Reliability software की capability का बहुत important aspect है, क्योंकि इससे यह सुनिश्चित होता है कि software सही तरीके से काम करेगा और अचानक fail नहीं होगा।

Example –
- code में bugs और errors
- proper testing और validation का अभाव
- other software और hardware के साथ incompatibilities

Required functions are:



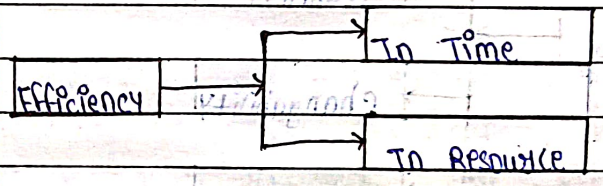
③ Efficiency – Efficiency software की characteristic है, जो उसकी capability को describe करती है कि वह resources जैसे memory, processing power, और network bandwidth का optimal तरीके से use कर सके।

High efficiency का मतलब है कि software अपने intended functions को जल्दी और कम resource का use करके पूरा कर देता है, जबकि low efficiency का मतलब होता है कि software slow चलता है या ज्यादा resources consume करता है।

Functionality –

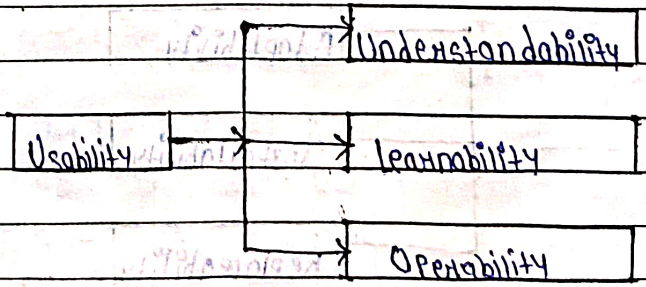
Software की efficiency को effect करने वाले factors के example –
- poorly designed algorithm और data structures
- memory और processing power का inefficient use
- unoptimized code
- High network latency या excessive bandwidth usage

Required functions are:

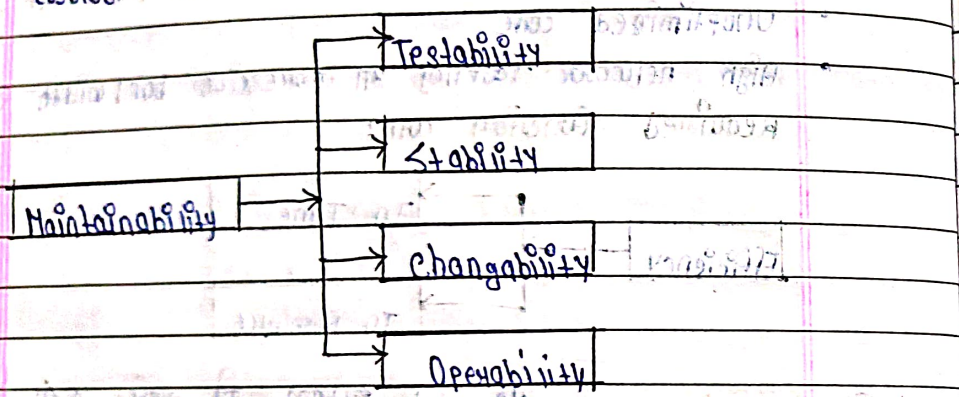


④ Usability – यह उस extent को define करता है कि software को आसानी से कितनी असानी से use किया जा सकता है। मानी software को use करना सीखने में कितना effort या समय लगता है।

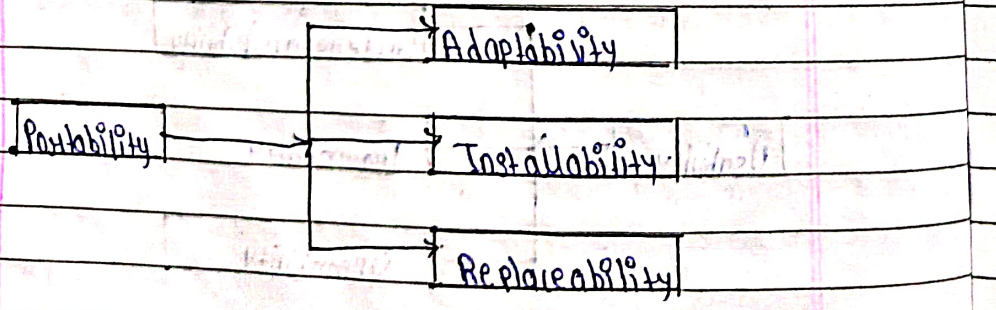
Required functions are:



⑤ Maintainability - यह उस ease को मापित करता है जिससे किसी software system में modifications किए जा सकते हैं ताकि उसकी functionality easily की जा सके, performance improve हो सके या system correct किए जा सके।



⑥ Portability - यह attributes का set होता है जो software की उस ability को दर्शाता है कि उसे एक environment से दूसरे environment में minimum changes के साथ transfer किया जा सके।



Changing nature of software -

इसके अंतर्गत software के ongoing process का evolution driven by new technology उसके environment और आवश्यकता के अनुसार changing environment को adopt करे।
इस base पर software को different-different category में divide किया गया है, जिसमें changing challenge बना हुआ है।

① System Software - System software programs का collection होता है जो program (application software) को service देने के लिए बनाया गया है।
System software generally hardware interaction scheduling, resource sharing etc को manage करता है।

② Application Software - Application software specific need को manage करने के लिए बनाया गया है।
इसके help से data processing, business calculation, आदि किया जाता है।

- Application software जिसे अक्सर 'app' कहा जाता है, यह type का computer program है

जो user को विभिन्न कार्य करने में help करता है।

③ Engineering Scientific Software - यह software engineering और scientific task को solve करने के लिए function provide करता है। जैसे कि numerical algorithm, computer aided design, real time system etc.

④ Embedded Software - यह किसी system या product के भीतर embedded होता है।

- Embedded software एक type का software होता है जो किसी विशेष hardware devices में "embedded" यानी पुड़ा होता है। यह software devices को control करता है और उसे विभिन्न कार्य करने में help करता है।

Example - microwave oven, smartphone etc.

⑤ Product line Software - यह software बहुत सारे customer के लिए बनाया जाता है यह generally market place में use होता है।

⑥ Web Application - यह client server based computer program होता है ~~जो~~ ~~generally~~ ~~market~~ जो web browser पर run होता है।

- यह किसी भी devices (जैसे computer, smartphone, tablet) पर work करता है। इस आपको एक functional connection और एक browser (जैसे - chrome, firefox, safari) की need होती है।

⑦ AI Software - यह complex problem non numerical algorithm को solve करने के लिए बनाया गया है।

- यह software data को analys करता है, pattern को समझता है और उसके आधार पर काम करता है। AI software का use आजकल कई area में किया जा रहा है जैसे कि स्वास्थ्य, Education, business और technology etc.

layer Approach

software का development layer approach को follow करता है। इसमे एक layer complete होने के बाद next layer को process किया जाता है।
प्रत्येक layer का different - different task होता है software development को निम्नलिखित layer पर divide कर सकते है।

① Quality Focus - software engineering मे main commitment quality focus पर होना चाहिए। यह security, integrity of function आदि को focus करता है।
यह maintainability और usability को focus करता है।

② Process - यह software engineering का foundation या base होता है। इसका main task यह होता है सारे layer को एक दूसरे से bind करके रखना।
यह framework भी define करता है जो software engineering मे effective delivery के लिए responsible होता है। Process के अंतर्गत निम्नलिखित activity, communication, planning, modeling, construction, deployment आते है।

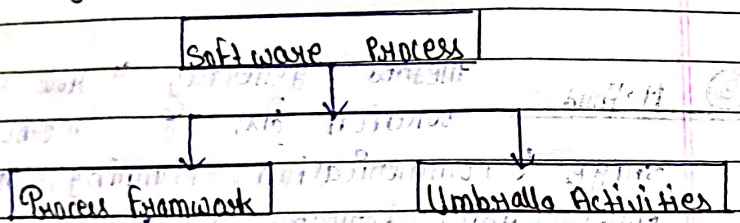
③ Method - method generally "How-to-do" का solution होता है। method के पास सारे stage (communication, planning, modeling, construction, deployment) आदि का information होता है।

④ Tools - process और method को automated और semiautomated support provide करता है इस layer मे सभी tools को integrate किया जाता है ताकि एक device द्वारा बनाई गई information को दूसरे मे use किया जा सके।
जब भी हम किसी type का software बनाते है तो हमे self-operating system की need होती है। तो इस तरह के जो self-operating को प्रदान करता है

उसे ही software engineering tools कहा जाता है।

Software Process

Software Process (जिसे हम software methodology भी कहते है) एक structured set of activities का group होता है, जिसका उद्देश्य एक high-quality software product को बनाना या existing system को modify करना होता है।



Software process किसी software को develop manage करने का एक structure provide करता है इसके मुख्यतः दो भाग होते है -

① Umbrella Activity - यह process framework के सभी stages के parallel work करता है। इसके अंतर्गत जैसे risk management, software quality assurance, technical review, measurability आदि आता है।

② Process Framework -

Process - यह activity, action और task का combination होता है जो quality product develop करता है।

Process framework - यह software development का main key है। यह किसी process के step और chronological order का information देता है। यह software development के लिए base का काम करता है। software process निम्नलिखित step को include करता है -

Process framework से determine करता है की कौन से processes essential होते है एक complex software project को complete करने के लिए। ये framework कुछ activities को identify करता है, जिसे की framework activities भी कहते है।

① Communication - इसमे customer के साथ नीक तरीके से communication किया जाता है जिससे उन्हे requirement को सही रूप मे समझा जा सके।

② Planning - इसमे एक plan को स्वाभाविक किया जाता है इस project को accomplish करने के लिए जिसमे project की schedule को describe किया जाता है।

उसमे प्रयुक्त technical tasks, expected गोल और प्रारम्भ के resources आदि।

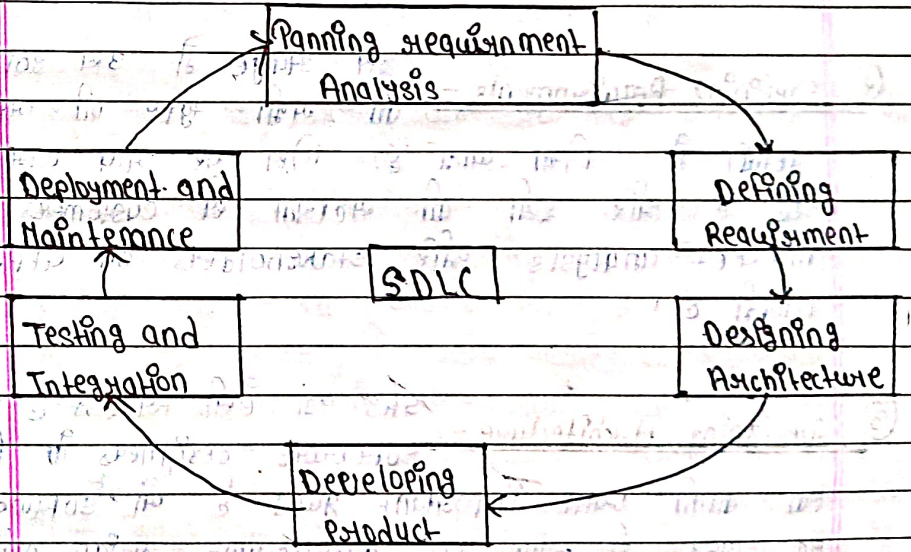
③ **Modeling** – इसमे models के creation पर भी ध्यान दिया जाता है। जिससे developer और user को allow किया जाता है ये समझने के लिए की software requirements क्या है और वो सभी testing जिससे की उन requirements को पूर्ण किया जा सके।

④ **Construction** – इसमे generation of code को testing के साथ combine किया जाता है जिससे वो code मे स्थित सभी errors को uncover कर सके।

⑤ **Deployment** – ये imply करता है की final product (software) को ठीक तरीके से user को deliver किया जाये। फिर user उस delivered product को evaluate करता है और इस evaluation के आधार मे feedback प्रदान करता है।

SDLC (Software Development life Cycle) –

- SDLC का पूरा name software development life cycle होता है और इस type का architecture process होती है, जिसका use software developing company किसी भी high-quality software को design करने, develop करने या फिर उन्हे test करने के लिए करती है।
- SDLC एक तरह की methodology है जिसमे किसी भी software को develop करने की सारी process को step-by-step define किया जाता है।



① Planning & Requirement Analysis - Planning जो है वह हर process का एक बहुत ही important part होता है और यह process software engineering और software development के लिए उतना ही ज्यादा जरूरी होता है। इस stage पर developers सभी type की need को analysis करते है, जो customer और user द्वारा दिए गए input और sales department / market के द्वारा दी गई suggestions के माध्यम से प्राप्त किया जाता है।

② Defining Requirements - इस stage मे उस software की सभी type की जरूरतो को define मे लिखा जाता है जिस पर आप काम कर रहे है और इसी की सहायता से customers, market analysts और stakeholders की approval मिलती है।

③ Designing Architecture - SRS एक ऐसी process है जो software designers के लिए एक काफी ज्यादा important guide है जो software का एक बहुत ही ज्यादा best architecture design करने मे help करता है।
SRS मे जो भी requirements define की जाती है इनके आधार पर ही कई संभावित product architecture designs को Design Document Specification (DDS) मे show किया जाता है।

④ Developing Product - यह SDLC की वह stage है जिस पर product का primary development process शुरू होता है। developers हमेशा ही DDS के द्वारा define किए गए design के आधार पर specific programming code लिखते है, जो पहले से ही software की functionality और architecture के साथ पूरी तरह से align होता है।

⑤ Testing - जब हमारा code पूरी तरह develop हो जाता है तो उसके बाद, ~~से~~ उसके बाद उस requirement के against test किया जाता है ताकि यह सुनिश्चित किया जा सके की यह जो product है वह उन सभी जरूरतो को solve कर रहा है की नही। इस stage मे unit testing, integration testing, system testing और acceptance testing की जाती है।

⑥ Deployment and maintance - Detailed testing के बाद, जो भी product होता है उसकी organization की strategy के हिसाब से कई अलग-अलग phases मे release किया जाता है फिर उसके बाद, product को real production environment मे पूरी तरह

deploy करके उसकी performance को test किया जाता है। अगर product expected अच्छा देता है तो organization उसको पूरी तरह से release कर देती है।

To develop software there are many model of SDLC...

① **Waterfall Model** – • waterfall model सबसे पहला SDLC model है जिसका use software development process के रूप में किया जाता है। इस model को waterfall नाम इसलिए दिया गया है क्योंकि एक झरना ऊपर से नीचे की ओर बहता है तो कभी नीचे आने के बाद ऊपर की ओर नही बहता।

• यही condition इस SDLC approach के साथ भी है कि अगर आप पहले phase से दूसरे phase में चले गए तो आप वापस पहले phase में नही जा सकते। यानी phases नीचे की ओर बढ़ते है।

• इसे linear sequential life cycle model भी कहते हैं। इसका अर्थ हुआ कि इसमें एक phase के खत्म होने के बाद ही दूसरा phase शुरू होता है और पहले phase से जो output generate होता है उसे दूसरे phase के लिये input के रूप में serve किया जाता है।

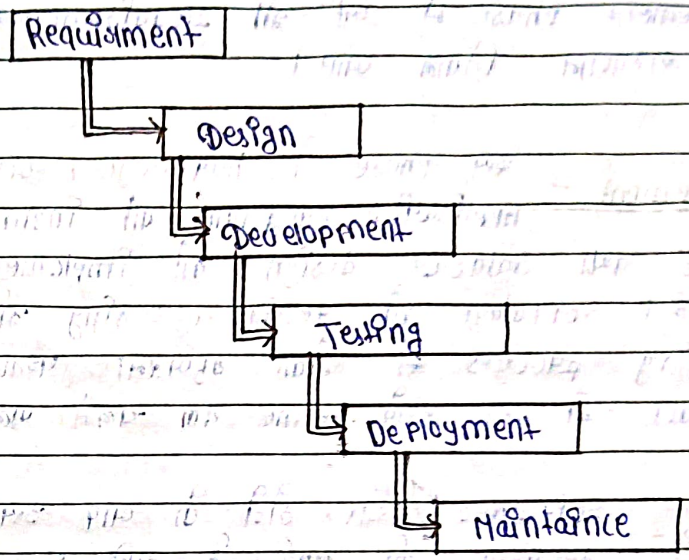


Fig - Waterfall Model

① **Requirment** – Requirment phase waterfall model का सबसे पहला phase है। इस phase system की requirment को एकत्रित और documented किया जाता है। यह phase बहुत important होता है क्योंकि इसी phase पर अगले phases आधारित होते है।

② **Design** – Design phase इस तथ्य पर आधारित होता है कि software का निर्माण किस type होगा। इसका मुख्य उद्देश्य software system का blueprint तैयार करना है जिससे कि आने वाले phases पर किसी type की कोई दिक्कत का सामना न करना पड़े और

requirement phase में जो भी requirements है उनका समाधान निकाल जाए।

③ Development - इस phase में hardware, software और application program को install किया जाता है तथा database design को implement किया जाता है। software को testing, coding और debugging process से होकर गुजरना पड़ता है। waterfall में यह लंबे time तक चलने वाला phase है।

④ Testing - software तैयार होने के बाद उसकी testing की जाती है ताकि यह check किया जा सके कि सब कुछ सही तरीके से काम कर रहा है या नहीं। testing से bugs और errors का पता चलता है, जिन्हे fix किया जाता है। इससे software की quality और reliability दोनो बढ़ती है।

⑤ Deployment - जब software पूरी तरह से test हो जाता है, तब उसे client के system पर deployment किया जाता है। यह वो stage है जब actual users उसे use करना शुरू करते है। Deployment से पहले कभी-कभी beta version भी release किया जाता है ताकि feedback लिया जा सके।

⑥ Maintenance - software launch के बाद भी खत्म नही होता। client की future needs और issues को ध्यान में रखते हुए time-time पर update और improvement किए जाते है।

Advantages of waterfall models -

① Simple and easy to understand होता है।
② हर stage clearly defined होती है।
③ इस phase में proper documentation होता है।
④ good for small projects
⑤ Easy to manage

Disadvantage of waterfall model -

① Inflexible होता है।
② Testing देर से होता है।
③ Not good for large project
④ High risk

② # Incremental Process Model

- Incremental process model एक software development life cycle (SDLC) model है। जिसमें software को small parts (increments) में develop किया जाता है।
- हर एक increment में software का एक working part बनाया जाता है, जो already developed parts के साथ integrate हो जाता है।
- इस model में software को एक ही बार बनाने के बजाए, उसे step-by-step (छोटे-छोटे parts) में develop किया जाता है।
- हर increment के साथ एक new feature add होता है, जो पिछले increment के साथ मिलकर काम करता है।

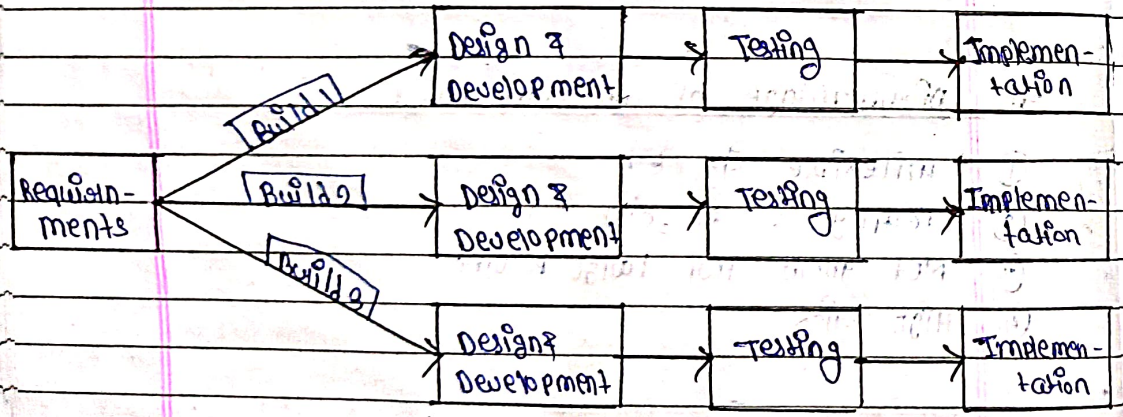


Fig- Incremental Model

① **Requirement Analysis** – इस phase में client की requirement को detail में समझा जाता है और उन्हें छोटे-छोटे increments (parts) में divide किया जाता है। सबसे पहले उन features को define किया जाता है जो most important होते हैं और जिन्हें जल्दी deliver करना जरूरी होता है।

② **Design phase** – इस phase में system architecture और हर increment की design तैयार की जाती है। हर increment के लिए specific design बनाई जाती है ताकि नया increment पहले से बने system के साथ smoothly work कर सके।
इसमें UI/UX, database structure और software components को define किया जाता है।

③ **Development and coding phase** – इस phase में actual software development और coding की जाती है। सबसे पहले पहला increment develop किया जाता है और फिर धीरे-धीरे आगे increment को add किया जाता है।
हर increment को पहले से बने हुए version में integrate किया जाता है।

④ Testing and Integration phase - हर increment को properly test किया जाता है ताकि कोई bug या error न रहे। अगर testing के दौरान कोई error आता है, तो उसे fix करके software को update किया जाता है। हर नया increment system में add होने के बाद दोबारा integration testing की जाती है ताकि पूरा system सही से काम करे।

Advantages

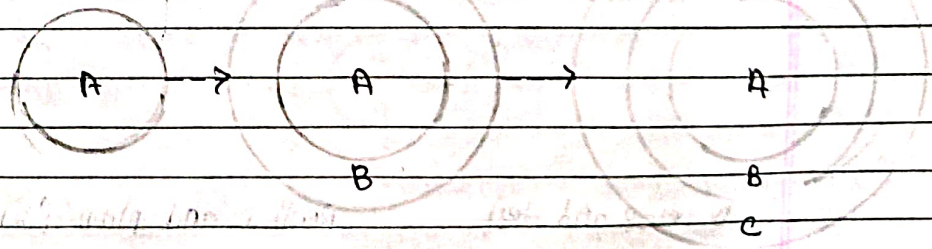
① Early delivery
② Flexibility
③ Risk management
④ Customer feedback
⑤ Easy testing & debugging

Disadvantages

① Planning complexity
② Costly in long run
③ Not suitable for small project
④ Integration issues

Evolutionary Model.

SDLC (Software development Life Cycle) के iterative और increment model को मिलाकर evolutionary model बनता है इसे कभी-कभी successive versions model और कभी-कभी increment भी कहते है।
इस model में software की requirements को कुछ सारे modules में तोड़ा जाता है और इन modules को increment करके बनाया जाता है और develop किया जाता है।
इसमें सबसे पहले software के core modules को develop किया जाता है, core modules वे modules होते है जिन्हे दूसरे modules से services की need नहीं होती है, प्रत्येक evolutionary model को iterative waterfall model का use करके develop किया जा सकता है।



① Spiral Model.

- Spiral model एक software development process model है। इस model में iterative और waterfall model दोनों की ही feature होती है। ऐसे project जो large और complex हो वहा पर इस model का use किया जाता है।
- इस model का name spiral इसलिए पड़ा क्योंकि अगर हम इसका diagram देखे तो वह एक spiral जैसा दिखता है, जिसमे एक लम्बी घुमावदार रेखा center point से शुरू होकर उसके चारो ओर कई घेरे (loops) बनाती है।
- Spiral मे कितने loops होंगे यह पहले से तय नही अपितु यह project के आकार और प्रकार की बढ़ती requirement पर depend करता है। Spiral के प्रत्येक loop को हम software development process का एक phase भी कहते है।

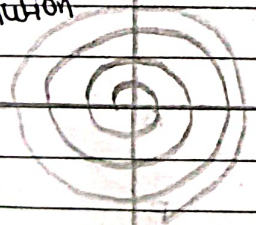


Fig - Spiral Model

Phase of spiral model -

① **Planning Phase** - • project के objective (लक्ष्य) को तय किया जाता है।
• Alternative solution और constraints को दिए किया जाता है।
• Resource और time schedule भी इसी phase मे किया जाता है।

② **Risk Analysis Phase** - • हर solution के साथ जुड़े risk को check किया जाता है।
• अगर कोई risk critical है तो उसका prototype (model / working demo) बनाया जाता है।
• Risk को minimize करने के लिए strategies बनाई जाती है।

③ **Engineering Phase** - • software का design बनाया जाता है।
• coding और unit testing की जाती है।
• फिर integrate करके पूरे system की testing की जाती है।

④ **Evaluation Phase** - • customer से feedback लिया जाता है।
• यह check किया जाता है कि developed product requirement की fulfill कर रहा

है या नहीं।
- Feedback के आधार पर अगली cycle में सुधार किया जाता है।

Advantage of spiral model

1. यह risk analysis पर ज्यादा focus करता है।
2. customer से feedback लिया जाता है।
3. complex software project के लिए ideal है।
4. Prototype और testing हर cycle में की जाती है।

Disadvantage of spiral models -

1. यह बहुत expensive होता है।
2. simple project के लिए suitable नहीं है।
3. इसमें development time बढ़ जाता है।
4. यह model practical नहीं है।

(2) Prototyping Model

Prototyping model software development का एक तरीका है जिसमें software का एक छोटा और easy prototype पहले बनाया जाता है।
- इस prototype को customer को दिखाया जाता है और उनकी feedback ली जाती है। इस feedback के आधार पर prototype को बार-बार improve जाता है जब तक कि customer को यह satisfaction न हो जाए कि यह उनकी requirement को पूरा करता है। इसके बाद ही final project बनाया जाता है।

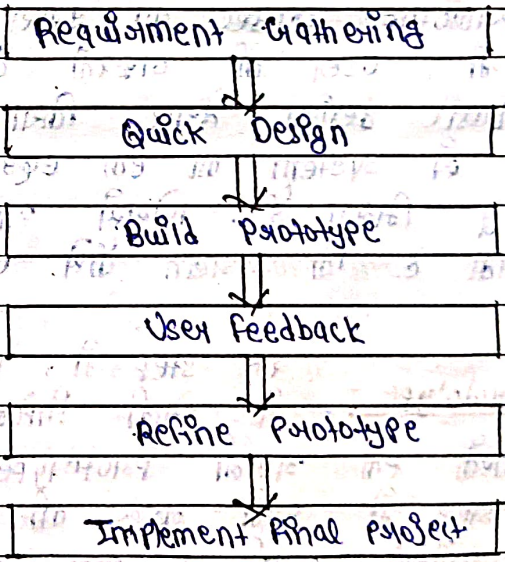


Fig - Prototyping model

① Requirement Gathering - सबसे पहले customer से उनकी आवश्यकताओं के बारे में पूछा जाता है। (आवश्यकताओं का collection और analysing) Prototype model design करने का पहला step होता है। इस step में, system के user से यह पूछा जाता है कि वह system से क्या उम्मीद करते हैं या उन्हें system से क्या चाहिए।

② Quick Design - आवश्यकताओं के आधार पर एक easy design तैयार किया जाता है। यह Prototype model का दूसरा step है। इस step में, user की आवश्यकताओं के आधार पर एक basic design तैयार किया जाता है। इस design से system का एक easy और overview मिलता है, जिससे यह समझना आसान हो जाता है कि system कैसे work करेगा।

③ Build Prototype - इस step में, Prototype design से मिली information का use करके एक असली Prototype बनाया जाता है। जिस step में हमने user की आवश्यकताओं और अपेक्षाएं समझी। अब उस information का use करके एक प्रारंभिक model (Prototype) तैयार किया जाता है।
यह Prototype यह दिखाता है कि system कैसे काम करेगा और क्या features होंगे।

④ User Feedback - Prototype को customer को दिखाया जाता है और उनकी feedback ली जाती है।
इस step में Preliminary testing से संबंधित है। इसमें Prototype के performance की check की जाती है। customer या उपयोगकर्ता इस Prototype को देखकर अपनी प्रतिक्रिया देते हैं और बताते हैं कि design में क्या अच्छाई है और क्या कमियां है।

⑤ Refine Prototype -
customer की feedback के आधार पर Prototype में improvement किया जाता है।
इस step में पहले अपने द्वारा Prototype नमूना sample को customer से evaluate कराने के बाद यदि customer से कोई को इससे कोई कमी है तो उसे सुधार किया जाता है।

⑥ Implement Final Product - जब customer prototype से satisfied हो जाता है, तो final product बनाया जाता है। जब prototype में सुधार हो जाये और customer को सही लगे तब उस prototype का actual implementation का work किया जाता है जहाँ software की coding की जाती है।

Advantage of Prototyping

① Better requirment understanding
② Early feedback
③ Reduced risk of failure
④ Flexibility

Disadvantages -

① Time consuming
② Costly
③ Complex ~~and~~ for large systems

Software Engineering Practice

- Practice एक broad concept होता है जो software development के लिए principle, method और tools provide करता है।
- Practice यह बताता है कि "how-to-do" कैसे करना है और उसके लिए क्या-क्या चाहिए। इसका practice से software development के लिए road map तैयार किया जाता है और software का details find out कर सकते है।
- इस type software के planning करते time method, tools, principle तैयार करने को software engineering practice कहा जाता है।
- इसका उद्देश्य high quality software develop करना होता है। जो software engineering और उसके manager द्वारा use किया जाता है।

Phase of Software Engineering

① Understanding the problem - इस phase में problem पर विचार विमर्श करके analysis किया जाता है और निम्न problem पर focus किया जाता है।
- problem का solution कैसे मिलेगा और कौन problem को solve कर सकता है।

Unit-02

Date: / /

Software Engineering Practice

- Practice एक broad concept होता है जो software development के लिए principle, method और tools provide करता है।
- Practice यह बताता है कि "how-to-do" कैसे करना है और उसके लिए क्या-क्या चाहिए।
- इसका practice से software development के लिए broad map तैयार किया जाता है और software का details find out कर सकते हैं।
- इस type software के planning करते time method, tools, principle तैयार करने को software engineering practice कहा जाता है।
- इसका उद्देश्य high quality software develop करना होता है। जो software engineering और उसके manager द्वारा पूरा किया जाता है।

Phase of Software Engineering

① **Understanding the Problem.** – इस phase में problem पर विचार विमर्श करके analysis किया जाता है और निम्न problem पर focus किया जाता है।

- Problem का solution कैसे मिलेगा और कौन problem को solve कर सकता है।

② Problem का solution find out करने के लिए किस type के data और function की need होगी।

③ Problem को easily solve करने के लिए इसे छोटे-छोटे part में divide किया जा सकता है या नहीं।

④ Problem को easily समझने के लिए उसे graphics form में interpret कर सकते है या नहीं।

② **Plan a solution** - इस phase में यह देखा जाता है कि Problem को के solution के लिए क्या-क्या किया जा सकता है इस phase में निम्न point पर focus किया जाता है -

① क्या ऐसी problem पहले आ चुकी है और उसका solution किस type निकाला गया था।
② कौन-सा model तैयार किया गया था।
③ कौन सा technique use किया गया था।

③ **Carry out the plan** - इस phase पर यह focus किया जाता है कि solution के लिए कौन-सा plan fix किया है।

- जो-जो component use कर रहे है क्या वह सही काम करेगा।

④ **Examin the result** - इस phase पर यह focus किया जाता है कि software कितना accurate result देगा।
- क्या इसके सभी component को test किया जा सकता है।
- क्या software data, function, feature के according result produce कर रहा है।

Core Principle -

Software engineering में बहुत सारे basic Principle use किए जाते है जो software developer और engineer द्वारा use किए जाते है software engineering एक difficult process है जिसके लिए software engineer द्वारा निम्न principle follow करता है।

① KISS
② DRY
③ Future based

④ Open Source
⑤ Solid
⑥ Divide & Conquer

① KISS (Keep It Simple Stupid) - Software engineering में engineer द्वारा complicated system develop किया जाता है इस कारण यह जरूरी हो जाता है कि code और debugging process को simple रखा जाए क्योंकि simple चीजें दिमाग को अच्छी लगती है।

② DRY (Don't Repeat Yourself) - Effective code को बार-बार use किया जा सकता है परन्तु एक ही code को कई जगहों पर से use नहीं करना चाहिए। क्योंकि उस code में in future कोई बदलाव आ जाता है तो सभी जगह पर correct करना पड़ेगा इसलिए ऐसा system रखना चाहिए कि एक ही जगह से code में correct किया जा सके।

③ Future Based - एक system की value पूरे lifetime होनी चाहिए technology और environment हर दूसरे दिन change होते रहते हैं तो ऐसा system create करना चाहिए जो changes हो या future में आने वाले नए technology को easy adopt कर ले इसके अलावा बहुत आगे (future) की सोच के software create नहीं करना चाहिए क्योंकि इससे time और resource का wastage होता है।

④ Open Source - हमें open source से और दूसरे के द्वारा या internet में available source code से बचना चाहिए।
Open source principle का मतलब है कि software का source code publicly available होता है जिसे कोई भी देख सकता है, use कर सकता है, modify कर सकता है और distribute कर सकता है।

⑤ Divide & Conquer - हमें बड़ी problem को छोटे-छोटे problem में divide कर लेना चाहिए फिर उसे solve करना चाहिए।
इस principle का मतलब है बड़े complex problem को छोटे manageable sub-problems में divide करना और फिर उन्हें individually solve करना।

⑥ Solid - SOLID पाँच design principle का set है, जो object-oriented programming (OOP) को maintainable, reusable और scalable बनाते हैं।

① S - single responsibility system
② O - open/closed principle
③ L - Liskov substitution principle
④ I - Interface segregation principle
⑤ D - dependency inversion principle

① S - प्रत्येक class की केवल एक responsibility होती है।

② O - open/closed principle software unit extension के लिए open होना चाहिए और modification के लिए close होना चाहिए।

③ L - program के correctness को change किए बिना object को instance के साथ easily replace किया जा सके।

④ I - software में use किए गए सारे interface independent होना चाहिए व बहुत सारे general purpose interface से एक client-specific interface design करना ज्यादा बेहतर है।

⑤ D - software abstraction पर dependent होना चाहिए concretion पर।

Requirment Engineering Task -

Requirment engineering, software engineer को वो problem solve करना सिखाता है। उसे अच्छे तरीके से समझने में help करता है।
Requirment Engineering task में निम्नलिखित task किए जाते हैं।

① Inception
② Elicitation
③ Elaboration
④ Negotiation
⑤ Specification
⑥ Requirment validation
⑦ Requirment Management

① Inception → How does a software project get started.
Starting में requirment engineer इसी प्रकार बेसिक्स तैयार करते हैं जिसमें problem के बारे में information fond out किया जा सके इसके अलावा किस problem को solve करना है उसका solution कैसा होगा customer और developer के बीच आपसी संबंध कैसा होगा।

② Elicitation - इस phase में problem का scope, problem की समझ और problem की awareness पर focus किया जाता है। इस phase में customer किस type का software चहता है software हमारे goal को कैसे achieve करेगा, customer का software बनवाने के पीछे goal क्या है और उसका use किस type किया जाना है।

③ Elaboration - इस phase में software कैसा project करेगा और उसका काम क्या होगा इस पर focus किया जाता है। इस phase में inception और elicitation phase से received information को expand किया जाता है।

④ Negotiation - इस phase पर customer के need कि priority, उसके classification को sorted बनाके रखा जाता है। इस phase में software के working process की accuracy पर focus किया जाता है।

⑤ Specification - इस phase पर software के working और वह कैसा दिखेगा के साथ-साथ document use of graphical model prototype आदि पर focus किया जाता है।

⑥ Requirement Validation - इस phase में इस बात पर focus किया जाता है कि software proper specification के अनुसार बना है या नही। उसके quality के maintain किया गया है या नही As per need बना है या नही।

⑦ Requirement Management - project team को इस बात पर help पहुचाने का कार्य करना कि वे current को trace और control कर सके। इसमे customer need manage करने के लिए project team को help software engineering team द्वारा किया जाना चाहिए।

Initiating requirements engineering process.

Requirement Engineering Process - stack holder के need और expectation के अनुसार requirement को identify elicitation analysis specify validation और management करने का process है
इस प्रकार software requirement को collect करना analysis करना और उसका documentation

Initiating requirement engineering process

इसका अर्थ होता है कि requirement engineering के process को कैसे start किया जाए
इसके planning में क्या करना होता है इसके लिए निम्नलिखित step follow किए जाते है।

1. Identify the stack holder
2. Recognizing multiple view point
3. Working toward collaboration
4. Asking the questions.

(1) **Identify the stack holder** - stack holder के अंतर्गत वे सभी आते है जो developing software से directly या indirectly लाभान्वित होते है

(i) End user, manager, maintainer engineer, domain specialist और trade union को identify करना।

(ii) stack holder क्या सोच रहे है।
(iii) stack holder के बीच आपसी संबंध कैसे है। इन सब की पहचान करना।

(2) **Recognizing multiple view point** - बहुत सारे stack holder होते है और system का requirement different-different point of view से explore किए जाए है उनका पहचान करना।

- हर stackholder का viewpoint अलग होता है। requirement engineer का काम है उन सबकी बातो को ध्यान से सुनना और analyse करना।

③ Working toward collaboration – इस phase में सारे stack holder meeting (collaborate) करते है और assignment के अनुसार final decision लेते है।

- assignment gathering सिर्फ एक तरफा process नही है। इसमे सभी stackholder का active participation चाहती होता है।
- teammembers और stackholders को मिलकर common understanding पर पहुंच जाना चाहती है।

④ Asking the question – इस phase मे system के advantage, customer के benifit और stack holder से related question पूछे जाते है। जो निम्न है –

① यह software को बनाने के लिए किसकी जरूरत है।
② इसका use कौन करेगा।
③ फायदा किसको मिलेगा।
④ क्या इसके अलावा और कोई resource है।

इसी प्रकार developer से related कुछ question होते है जो निम्न है।

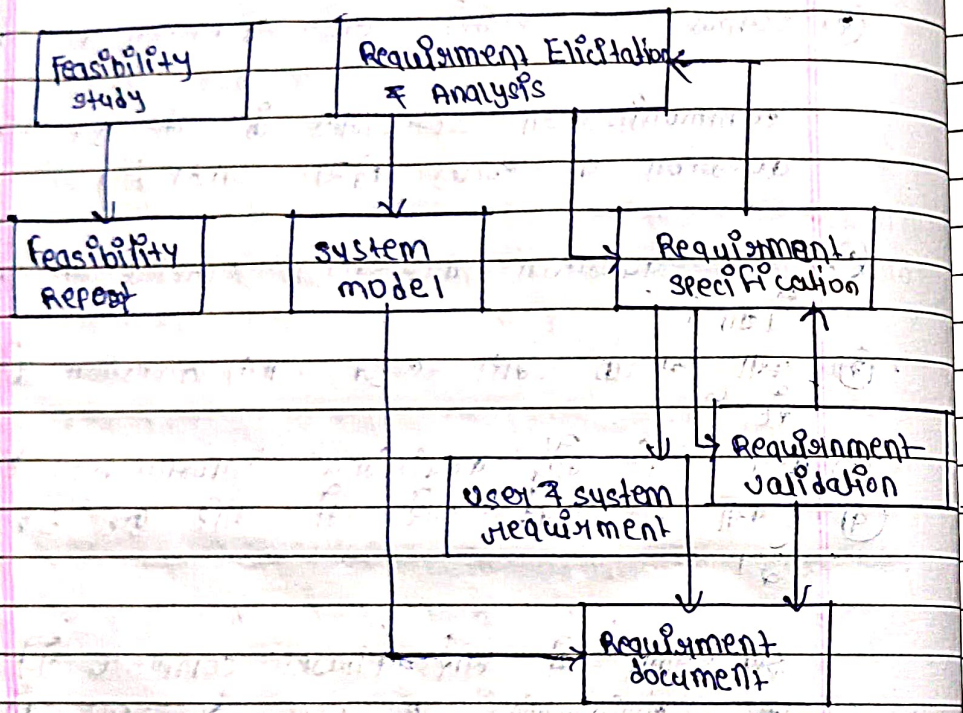
① problem को solve करने के लिए सबसे easy way कौन सा है।
② output का सबसे easy solution क्या है।

communication activities के लिए कुछ simple question पर focus किया जाता है।

① इन question का सबसे effective answer कौन देगा।
② क्या आपके द्वारा बहुत ज्यादा question पूछे जा रहे है।
③ क्या और कोई documental information है।
④ क्या मैं इसके बारे मे और कुछ पूछ सकता हूँ।

इस प्रकार से सारे phase complete होने के बाद engineering process के need को complete कर सकते है।

Requirement Engineering Process



Requirment engineering process के step निम्न है

1. Feasibility study
2. Requirment Elicitation and Analysis
3. Requirment specification
4. Requirment validation

① Feasibility study - software को develop करने का कारण क्या है जो user द्वारा acceptable हो। software को develop करने के पीछे क्या aim और goal है क्या-क्या चीजो की जरूरत पड़ेगी, क्या-क्या problem available होंगे जो user के requirment के अनुसार हो।

Feasibility study के निम्न type है -

① Technical feasibility study
② Operational operational study
③ Economical feasibility study

इसके अलावा legal feasibility study, cultural & political F.S & market F.S

① Technical Feasibility study - Technical feasibility में current resource (hardware और software) और required technology का analysis/ assessment किया जाता है ताकि project develop किया जा सके।

② Operational feasibility - Operational feasibility में यह analyze किया जाता है कि

product requirement को fulfill करने में कितना capable होगा और develop होने के बाद उसे operate और maintain करना कितना easy होगा।

③ Economic Feasibility - Economic feasibility study में project के cost और benefit का analyse किया जाता है। इसका मतलब है कि इस feasibility study के तहत project development की सही लागत का analysis किया जाता है।

② Requirement Elicitation and Analysis - इसका मतलब होता है requirement इकट्ठा करना जो customer के need और expectation के अनुसार हो। Analysis मतलब होता है बहुत सारे requirements में से जरूरी और important point को analyse करना और उसे select करना। Requirement Elicitation के निम्नलिखित technique है -

① Interview
② Survey
③ Observation
④ Prototyping

③ Software Requirement Specification (SRS) - Client से collect किये requirement natural language में होते है software analysed उस requirement को technical language में document बनाता है। SRS में data flow diagram (D.F.D), Entity-Relationship diagram (ER diagram), data dictionary आदि का use किया जाता है। SRS निम्न चीजो को define करता है -

① How software, hardware से कैसे interact करेगा
② external interface / GUI
③ structure of software
④ speed of operation
⑤ speed of recovering
⑥ Maintance, security, quality etc.

④ Software Requirement Validation - इस phase में document को validate और test किया जाता है। यह process के लिए निम्न condition check किया जाता है -

① क्या इसको practical implement किया जा सकता है।
② क्या इसे limited resource में कर रहे है

③ कोई ambiguity तो नही है।
④ कोई changes या additional requirement तो नही है।

⑤ Requirement management - इस phase मे engineering process के दौरान changing requirement को manage किया जाता है। requirement management मे निम्नलिखित activity involve होते है -

① Tracting & controlling changes
② Traceability
③ version control
④ communication

Software Requirement Specification -

SRS एक complete document होता है जो system के performance requirement functional और non-functional requirement के बारे मे बताता है।
यह एक blueprint होता है (formal report) जो customer के requirement के अनुसार activities facility provide करता है, यह एक प्रकार से developing system का complete description/document होता है।

User of SRS :-

① client
② Development team
③ Maintance team
④ Technical writter

Need of SRS :-

① clear understanding between client and developer - client और developer के बीच clear understanding देता है।

② Help and testing
③ Improves communication
④ Legal protection
⑤ Designing Made easy
⑥ Avoids rework and wastage of efforts

② testing के लिए base provide करता है।
③ communication improve करता है।
④ Legal protection देता है।
⑤ designing और development easy हो जाता है।
⑥ Rework और wastage कम करता है।

Structure of SRS -

① Introduction -
1.1 Purpose
1.2 Scope
1.3 Definition
1.4 Reference

② Overall Description
2.1 User Interface
2.2 System Interface
2.3 SW & HW requirement
2.4 Constraints
2.5 User characteristics

③ System feature & Requirment
3.1 function requirment
3.2 Sequence diagram
3.3 External Interface requirment
3.4 Database requirment
3.5 Non functional requirment

④ Deliver for Approval

Characteristics of (SRS) -

① Complete - SRS जो है वह complete होना चाहिए अर्थात जो भी software की need है वह सभी SRS से उल्लेखित होनी चाहिए।

② Correct - यह accurate होना चाहिए अर्थात जो customer की चाहते है उसके रिसाब होना चाहिए।

③ clear - यह स्पष्ट होना चाहिए, software की need को स्पष्ट रूप से declared होना चाहिए।

④ Verifiable - यह verifiable होना चाहिए जो expects तथा testable होते है।

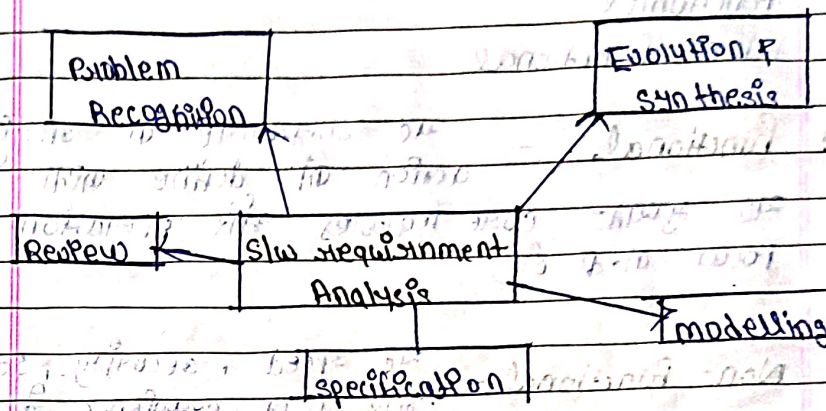
⑤ Modifiable - SRS का वो document होता है जिसमे specify की गयी सारी requirements modify करने योग्य होनी चाहिए।

⑥ Testable - यह testable होना चाहिए।

⑦ Accurate - इसमे accuracy होनी चाहिए अगर यह ही accurate नही होगा तो software का निर्माण नही हो सकता।

⑧ Traceable - यह traceable होना चाहिए अर्थात् इसमे प्रत्येक requirement अलग-अलग type से identify होनी चाहिए, प्रत्येक requirement की अपनी अलग एक पहचान होनी चाहिए।

Requirement Analysis



① requirement analysis software development का foundation step है जो stack holder के need के अनुसार identify और documentation पर focus करता है।

② यह software development process का development होता है।

③ इस phase मे functional और non-functional requirement client पक्ष और domain stack holder से collect किया जाता है।

④ यह phase designing के लिए usage prepare करता है।

⑤ यह cost को resolve करता है और misunderstanding को avoid करता है। यह process को categories मे classify किया जाता है।

① Functional
② Non- functional

① Functional - यह software के specific action को define करता है। यह मुख्यतः core features और operation पर focus करता है।

② Non- functional - यह speed, security, scalability और user experience के साथ system various conditions में कैसे perform करेगा से बताता।

Importance of requirement analysis :-

① clear understanding of stack holder need
② Avoiding scope creep
③ Improved communication and collaboration
④ Minimize risk and errors
⑤ Reduce development cost
⑥ Improved quality and user satisfaction

① clear understanding of stakeholder needs - Requirement analysis के माध्यम से developers को stakeholders (client, end-users, business leaders) की needs, expectations और goals की clear understanding मिलती है।

② Avoiding Scope Creep - अगर requirement पहले से ही clearly define और document हो, तो project में scope creep (यानी uncontrolled changes या unnecessary additions) को avoid किया जा सकता है।

③ Improved Communication and collaboration - Requirement analysis से developers, project managers और stakeholders के बीच regular communication और collaboration होता है। इस process से misunderstanding कम होते हैं और problem को पहले ही solve किया जा सकता है।

④ Reduced Development Costs - जब requirement clear और documented हो, तो development team unnecessary rework के बिना सही solution design कर पाती है। इससे time और cost दोनों save होते हैं।

⑤ Minimized risks and Errors - Requirements को early stage पर identify और document करने से technical जोखिम और errors कम हो जाते हैं।

⑥ Better Project Planning - Clear requirement के base पर project managers timelines, resource allocation और budget estimates आसानी से बना सकते हैं। इससे project schedules पर रहता है और budget के अंदर पूरा होता है।

⑦ Improved Quality and User Satisfaction - Functional और non functional requirements को carefully gather करने से final product high quality का बनता है और user expectations को fulfill करता है। इसका result होता है better user satisfaction.

Activities involved in S/w requirement Analysis

S/w requirement analysis का मतलब complete study analysis describe S/w requirement जो problem को solve करने में help करता है।

requirement analysis में निम्नलिखित activities involve है -

diagram

① Problem Recognition - requirement analysis का main मुख्य उद्देश्य requirement के objective को समझना है जैसे कि ① इसकी क्यों जरूरत है, ② क्या यह beneficial है ③ यह project के quality को बढ़ा सकता है। ④ S/w का behaviour कैसा रहेगा

② Evolution & Synthesis - Evolution का मतलब क्या यह सही काम कर रहा है या नहीं का judgment करना है। Synthesis का मतलब कुछ बनाना होता है इस phase में निम्न बिन्दु पर focus किया जाता है -

① S/w में कौन-कौन से function जरूरी है।

② Data flow सही है या नही।
③ System सही ज्वार कर रहा है या नही।
④ Analysis के हिसाब से design किया है या नही।

③ Modelling - Information gathering और उपर के दोनो stage complete होने के बाद functional और behaviour model prepare किया जाता है

④ Specification - इसका मतलब functional और non-functional requirement को specify करना होता है।

⑤ Review - SRS develop होने के बाद इसे
① क्या इसे improve किया जा सकता है
② क्या कैसे refine करके quality को increase किया जा सकता आदि को consider करना चाहिए।

Requirement Analysis Technique

① Data Flow Diagram (DFD)
② Entity relation (ER)
③ Unified modeling language (UML)
④ Prototyping Tools
⑤ Business Process Model and Notation (BPMN) ×

① Data Flow Diagram (DFD) - DFD एक graphical representation दिखाता है जो किसी system में data flow होता और

Analysis Modeling Approach

① यह system का technical representation होता है
② यह customer का requirement बताता है।
③ यह software design के लिए base ready करता है।
④ यह text और diagram का combination होता है जो requirement को बताता है।
⑤ Analysis model तैयार करके customer के need के कमियों को दूर किया जाता है।

Analysis modeling approach दो type के होते है-

① Structured
② Object oriented Approach

① Structured Analysis Approach - structured analysis में data और process होते है जिसमे data एक separate entity के form मे रहता है।
Analysis data और process पर किया जाता है process data को separate entity के रूप मे transform करता है data attribute और relationship के form मे model किया जाता है।
process को निम्नलिखित को दिखाने के लिए model किया जाता है।

① Input data
② Data transformation
③ The resulting output data

structured approach 3 type के होते है।

① Data modelling → ER Diagram
② Functional modelling → DFD, Data Dictionary
③ Behavioural modelling → state Diagram, sequence

① Data modelling - इसमे data को process से independent examine किया जाता है यह निम्नलिखित चीजो को identify करता है।

① Data object (entity)
② Data Attribute
② Relationship

④ Cardinality

E-R - Diagram बनाना है

Data modelling concept

यह formal technique use करके information system के लिए data model बनाने का process होता है।

जिस type data के flow को दिखाने के लिए text और symbol का use किया जाता है उसी प्रकार difficult software system के design को समझाने के लिए diagram के form में represent करने के process को data modelling कहते है।

Data model का use data को किस प्रकार exchange करना है उसको समझाने के लिए किया जाता है।

(2) Object Oriented Approach

object oriented analysis customer need को represent करने एवं classes को define करने पर focused रहता है।
इसके लिए निम्न model बनाया जाता है-

(1) Conceptual model

(1) Identify the important thing
(2) modify objic correct it
(3) define its behaviour and relation
(4) make a simple diagram

(2) Elicit Requirement - यह define करे की software का purpose क्या है। और उसे बनाने मे किस प्रकार की problem आ रही है।

(3) Specify Requirement - Analyst handles के अनुसार बताये गए story के base पर need को describe करना।

(2) Logical

(3) Physical