Introduction of Estimation

- Problem size estimation એ effort, time duration અને software project તથા Cost નો અંદાજ છે.
- Problem size સ્પષ્ટપણે source કોડ ના બાઇટ્સની સંખ્યા નથી. તે એક્ઝેક્યુટેબલ કોડનો બાઇટ size પણ નથી.
- હાલ માં આ બે metrics નો ઉપયોગ size ના estimate માટે કરવામાં આવે છે.
  - ✓ Lines of code (LOC)
  - ✓ Function point (FP)

Line of Code

- LOC metric ખૂબ popular છે કારણ કે તેનો સરળતાથી ઉપયોગ કરી શકાઈ છે. આ મેટ્રિકનો ઉપયોગ કરીને, project ની size નો અંદાજ develop પ્રોગ્રામમાં source instruction ની સંખ્યાનું counting દ્વારા કરવામાં આવે છે.
- એ Lines કે જેનો ઉપયોગ code ને comment આપવા માટે વપરાય છે તેવી line અને હેડર લાઇન ને ignored કરવામાં છે.
- પ્રોજેક્ટ ના અંતે LOC ની ગણતરી નક્કી કરવી ખૂબ જ સરળ કામ છે.
- પ્રોજેક્ટની શરૂઆતમાં LOC count નો અંદાજ કાઢવા માટે, પ્રોજેક્ટ manager સામાન્ય રીતે problem ને module માં વિભાજિત કરે છે અને દરેક મોડ્યુલ Sub Modules, જ્યાં સુધી વિવિધ leaf-level મોડ્યુલોના કદ અંદાજે predicted ન થાય ત્યાં સુધી.

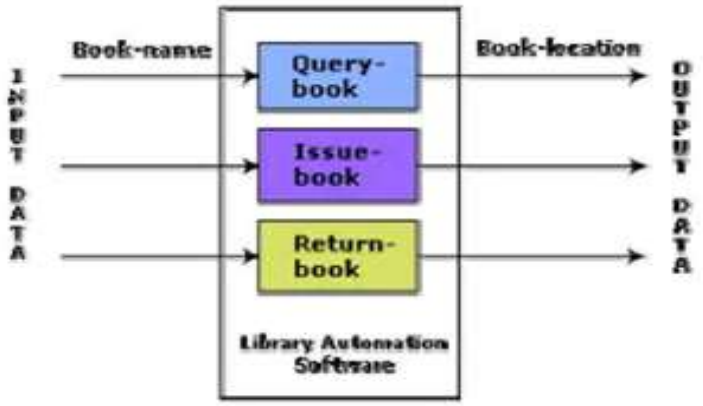
Lines of Code (LOC) metric ના ગેરફાયદા

- LOC એ problem size ની numerical value આપે છે જે individual coding style સાથે બદલાય છે અલગ અલગ programmer's તેમના code ને અલગ અલગ રીતે મૂકે છે. ઉદાહરણ તરીકે, એક પ્રોગ્રામર એક લાઇન પર ઘણી source instruction લખી શકે છે જ્યારે અન્ય programmer એક લાઇનમાં એક instruction ને વિભાજિત(split) કરી શકે છે. Code ની lines કરતાં પ્રોગ્રામમાં language tokens ની ગણતરી કરીને આ સમસ્યા ને સરળતાથી દૂર કરી શકાય છે.
- જો કે, તેના થી વધુ મુશ્કેલ સમસ્યા ઊભી થાય છે કારણ કે program ની લંબાઈ એ program લખવા માટે ઉપયોગમાં લેવાયેલી instruction ની choice પર આધારિત છે. તેથી, same problem માટે પણ, અલગ અલગ programmer ને અલગ અલગ LOC નો counts program માં આવી શકે છે.
- Problem ની size માપવા માંટે problem ની overall complexity અને તેને solve કરવા માટેના effort ને ધ્યાનમાં લેવું જોઈએ. એટલે કે, તે માત્ર coding effort નહીં, પરંતુ design, code, test વગેરે માટે જરૂરી પ્રયત્નોને ધ્યાનમાં લેવા જોઈએ.
- LOC એ ખાલી coding activity પર focus કરે છે; તે ફક્ત final program માં source line ની સંખ્યાને count કરે છે.
- Program માં લાંબા code હોય એવા size ના program માં જરૂરી નથી કે તેમાં સારી quality જ હોય. કેટલાક programmer લાંબા અને complicated code બનાવે છે કારણ કે તેઓ available instruction set નો સારી રીતે ઉપયોગ કરતા નથી.

- જો programmer ઘણી અલગ અલગ library routine નો ઉપયોગ કરે, તો LOC count ઓછો થાય છે. અને તે smaller program size તરીકે દેખાશે.

Function Points

- Function Point metric એ LOC metric ની ઘણી ખામીઓને દૂર કરે છે. Function point metric નો ઉપયોગ કરવાનો એક important advantages એ છે કે તેનો ઉપયોગ problem specification થી સીધા જ software product ના size નું estimate કરવા માટે થઈ શકે છે.
- LOC metric માં, product સંપૂર્ણ રીતે develop થઈ જાય તે પછી જ size ને સારી રીતે determine કરી શકાય છે. જ્યાં function point metric પાછળના conceptual idea એ છે કે software product ની size એ અલગ અલગ function અથવા features ના આધાર પર directly dependent હોય છે.
- For example, Library Automation Software ની issue book કરવાના feature (figure માં બતાવ્યા પ્રમાણે) input તરીકે book નું name લે છે અને તેના location ને display કરે છે અને તેની copies ની સંખ્યા ને બતાવે છે.



1) Number of inputs

- User દ્વારા દરેક data item ને input તરીકે ગણવામાં આવે છે. તે ધ્યાનમાં લેવું જોઈએ કે user દ્વારા input કરવામાં આવેલી individual data item એ input ની સંખ્યાના calculation માં consider કરવામાં આવતી નથી, પરંતુ related input ના group ને single input તરીકે consider કરવામાં આવે છે.
- For example, pay roll software માં employee ને લગતા data enter કરતી વખતે; data items name, age, address, phone number, etc. એકસાથે single input તરીકે ગણવામાં આવે છે. આ બધા data item એ related હોઈ શકે છે, કારણ કે તેને single employee ના data તરીકે count કરવામાં આવે છે.

  UFP = (Number of inputs)*4 + (Number of outputs)*5 + (Number of inquiries)*4 + (Number of files)*10 + (Number of interfaces)*10

2) Number of outputs

- Output એ print કરેલા reports, screen outputs, produce થયેલ error messages, વગેરેને refer કરે છે. આઉટપુટની સંખ્યાને આઉટપુટ count કરતી વખતે report ની અંદર individual data items ધ્યાનમાં લેવામાં આવતી નથી, પરંતુ related data item નો set ને one input તરીકે ગણવામાં આવે છે.

3) Number of inquiries

- Inquiries એ user command છે જેમ કે print-account-balance. Inquiries ને અલગ અલગ ગણવામાં આવે છે.

4) Number of files

- દરેક logical file ને count કરવામાં આવે છે. Logical file નો meaning logically related data નું group એવો થાય છે.

5) Number of interfaces

- અહીં interface ને એવા interface તરીકે consider કરવામાં આવે છે જે અન્ય system સાથે information ને exchange કરવા માટે વપરાય છે. અન્ય system સાથે ની communication link એ આવા interface નું Example છે
- એકવાર unadjusted function point (UFP) ની ગણતરી થાય તે પછી, technical complexity factor(TCF) ની ગણતરી કરવામાં આવે છે. TCF એ high transaction rate, throughput અને response time requirements વગેરે જેવા 14 અન્ય factor ને ધ્યાનમાં લઈને UFP measure માં (refine) સુધારો કરે છે.
- આમાંના 14 factor ને 0 (not present) થી 5 (strong influence) થી assign કરવામાં આવે છે. Resulting numbers એ sum છે, તે total degree of influence (DI) આપે છે.
- હવે, TCF = (0.65+0.01*DI). DI એ 0 થી 70 ની વચ્ચે બદલાય છે, TCF એ 0.65 થી 1.35 સુધી બદલાય છે.
- છેલ્લે, FP=UFP*TCF.

Function points ની ગણતરી

Step 1: દરેક function point complexity અનુસાર ક્રમાંકિત છે. Category માં દરેક function point માટે pre-defined weights exists છે.

Step 2: દરેક F.P નો ગુણાકાર કરીને unadjusted function point ગણતરી કરો. તેના corresponding weight factor દ્વારા.

Functional Units	Weighing factors		
	Low	Average	High
EI	3	4	6
EO	4	5	7
EQ	3	4	6
ILF	7	10	15
EIF	5	7	10

Step 3: Final function points ગણતરી કરો.

Final F.P. = UFP * CAF (CAF = Complexity Adjustment Factor)

- CAF ની ગણતરી પ્રક્રિયા complexity 14 aspects નો ઉપયોગ કરીને કરવામાં આવે છે.
- 0 થી 5 ના સ્કેલ પર 14 પ્રશ્નોના જવાબ.
  0 - No Influence
  1 - Incidental
  2 - Moderate

3 - Average
4 - Signification
5 - Essential

CAF = 0.65 + (0.01 * ∑Fi) (Fi = varies from 1 to 14)

Example: Given the following values, compute F.P. when all complexity adjustment factors and weighting factors are average.
User Input = 50
User Output = 40
User Inquiries = 35
User Files = 6
External Interface = 4
Effort = 36.9 p-m
Cost = $7744/ month

Solution :

- **Step-1:** complexity adjustment factor average હોવાથી (પ્રશ્નમાં આપેલ છે),
  આથી average=3 then **∑(f_i)** = 14*3 =42
- **Step-2:** CAF = 0.65 + (0.01 * 42) = 1.07
- **Step-3:** weighting factors પણ average છે (પ્રશ્નમાં આપેલ છે) તેથી આપણે દરેક individual function point ને ટેબલમાં અનુરૂપ વેલ્યુ સાથે ગુણાકાર કરીશું.
  UFP(count total) = (50*4) + (40*5) + (35*4) + (6*10) + (4*7) = 628
- **Step-4: Function Point = Count-total * CAF** = 628 * 1.07 = 671.96
- **Step-5: Productivity= FP/Effort** =671.96/36.9= 18.21
- **Cost per function** = cost/Productivity = 7744/18.21= $425

Function point approach ના ફાઈદાઓ

- Delivered સૉફ્ટવેર નું Size language અને technology અને tools થી independent રીતે માપવામાં આવે છે.
- F.P. ડિઝાઇન અને કોડિંગ પહેલાં requirements પરથી સીધો estimated લગાવવામાં આવે છે.
  - ✓ મુખ્ય ડિઝાઇન અને કોડિંગ થાય તે પહેલાં જ આપણને સોફ્ટવેરના size નો અંદાજ મળે છે.
  - ✓ Requirements માં કોઈપણ ફેરફાર F.P count માં સરળતાથી reflected થઈ શકે છે.
- F.P. technical expertise વિનાના વપરાશકર્તાઓ માટે પણ ઉપયોગી છે.

COCOMO Model

- Boehm (1981) દ્વારા COCOMO (Constructive Cost Estimation Model) ને બનાવવામાં આવ્યું હતું. Boehm ના જણાવ્યા મુજબ, કોઈપણ software નું development તે project ની development ની complexity પર આધારિત નીચેની three categories માં classified કરી શકાય છે: organic, semidetached, અને embedded.
- **Organic:** development project ને organic type તરીકે consider કરી શકાય છે, જો project સારી સમજાયેલી application program ના developing સાથે deal કરે છે, તો development team ની size

small
હોય છે અને team member ને similar type નાં project ના developing નો experience હોય છે.

- **Semidetached:** જો development માં અનુભવી અને બિનઅનુભવી staff ના mixture નો સમાવેશ થાય છે, તો development project ને semidetached type માં consider કરી શકાય છે.
- **Embedded:**જો develop કરવામાં આવતો software એ complex hardware સાથે જોડાયેલો હોય તો development project ને embedded type consider કરવામાં આવે છે
- Boehm ના જણાવ્યા અનુસાર, software cost estimation ત્રણ stage માં થવો જોઈએ: Basic COCOMO, Intermediate COCOMO, and Complete COCOMO.

Basic COCOMO Model

- Basic COCOMO model એ project parameter નો approximate estimate આપે છે. basic COCOMO estimation model નીચેના expression દ્વારા આપવામાં આવે છે
  Effort = a1 × (KLOC)^{a2} PM
  Tdev = b1 × (Effort)^{b2} Months

જ્યાં

- ✓ KLOC એ Kilo Lines of Code માં express કરેલા software product ની estimated size છે.
- ✓ a1, a2, b1, b2 એ software product ની દરેક category માટે constant હોય છે
- ✓ Tdev એ months માં express કરેલા software ને develop કરવાનો estimated time છે
- ✓ Effort એ software product ને develop કરવા માટે કેટલો total effort જોઈ છે તે દર્શાવે છે, જે person months (PMs) માં express કરે છે.

- Boehm ને According, source text ની દરેક line ની ગણતરી LOC થી કરીશું.
- Boehm [1981] દ્વારા આપવામાં આવેલ product ની વિવિધ category (i.e. organic, semidetached, અને embedded) માટે a1, a2, b1, b2 ની value નીચે જણાવેલ છે. તેમણે મોટી સંખ્યામાં actual project માંથી collect કરેલ historical data ની તપાસ કરીને ઉપર ના expression ને derive કર્યા છે.
- **Estimation of development effort:** software product ના three class માટે, code size ના આધારે effort ના estimate માટે ના formulas નીચે બતાવેલ છે
  - ✓ Organic : **Effort = 2.4(*KLOC*)$^{1.05}$ PM**
  - ✓ Semi-detached : **Effort = 3.0(*KLOC*)$^{1.12}$ PM**
  - ✓ Embedded : **Effort = 3.6(*KLOC*)$^{1.2}$ PM**
- **Estimation of development time:** software product ના three class માટે, effort ના આધારે development time નો estimate કાઢવા માટેના formulas નીચે આપેલ છે
  - ✓ Organic : **Tdev = 2.5(*Effort*)$^{0.38}$ Months**
  - ✓ Semi-detached : **Tdev = 2.5(*Effort*)$^{0.35}$ Months**
  - ✓ Embedded : **Tdev = 2.5(*Effort*)$^{0.32}$ Months**
- Effort estimation થી, project cost દર મહિને manpower cost દ્વારા આવશ્યક effort ને વધારીને મેળવી શકાય છે.

- **Example:** Assume કરો કે organic type ના software product ની size એ source code ની 32,000 lines હોવાનો estimate છે. ધારો કે software engineer ની average salary Rs. 15,000/- per month હોય છે.Software product અને nominal development time Develop માટેના effort ને Determine કરો.
- organic software માટેના basic COCOMO estimation માંથી:
  Effort = **2.4 x (32)$^{1.05}$= 91 PM**
  Development time = **2.5 x (91)$^{0.38}$= 14 months**
  Cost required to develop the product = **91 x 15000**

  = **Rs. 1365000/-**

Intermediate COCOMO model

- Basic COCOMO model એ assume કરે છે કે effort અને development time એ product size ના function છે.
- તેથી, effort અને project duration નો ચોક્કસ estimation મેળવવા માટે, બધા relevant parameter ની effect ધ્યાનમાં લેવી જોઈએ.
- intermediate COCOMO model આ fact ને recognize કરે છે અને software development ના various attributes ના આધારે 15 cost drivers (multipliers) set નો ઉપયોગ કરીને તથા basic COCOMO expression નો ઉપયોગ કરીને initial estimate ને refine કરે છે
- સામાન્ય રીતે, cost driver ને નીચેની items ના attribute તરીકે classified કરી શકાય છે
- **Product:** product ની characteristics જેવી કે product ની reliability requirement
- **Computer:** computer ની Characteristics જેમ કે execution speed require છે, storage space જરૂરી છે.
- **Personnel:** કર્મચારીઓનું experience level, programming capability, analysis capability etc.
- **Development Environment:** automation) CASE) tool નો ઉપયોગ software development માટે થાય છે.

Complete COCOMO model

- Basic અને intermediate COCOMO model ની major ખામી એ છે કે તેઓ એક software product ને single entity તરીકે માને છે.
- જો કે, મોટાભાગની મોટી systems એ ઘણી નાની sub-systems થી બનેલી હોય છે. આ subsystem માં મોટે ભાગે અલગ અલગ characteristics હોઈ શકે છે.
- For example, કેટલાક subsystem એ organic type, કેટલાક semidetached અને કેટલીક embedded તરીકે માનવામાં આવે છે. માત્ર એટલા માટે નહીં કે subsystem ની development ની complexity અલગ હોઈ શકે છે, પરંતુ કેટલાક subsystem માટે પણ reliability ની requirement high હોઈ શકે છે, કેટલાક માટે development team પાસે similar development નો કોઈ previous experience હોતો નથી.
- Complete COCOMO model subsystems ની characteristics માં આ તફાવતોને ધ્યાનમાં લે છે અને વ્યક્તિગત subsystem માટેના અંદાજની રકમ તરીકે effort અને development time નો અંદાજ આપે છે.
- country માં અનેક place ઉપર office ધરાવતી organization માટે distributed Management Information System (MIS) product માટે નીચે આપેલા sub-components હોઈ શકે છે:
  - ✓ Database part

- ✓ Graphical User Interface (GUI) part
- ✓ Communication part

- આમાંથી, communication part ને embedded software તરીકે consider કરવામાં આવે છે. Database part ને semi-detached software અને GUI part એ organic software હોઈ શકે છે. આ three components માટેનો cost અલગથી estimate કરવામાં આવે છે, અને system ની overall cost આપવા માટે સમજાવી શકે છે.

Scheduling

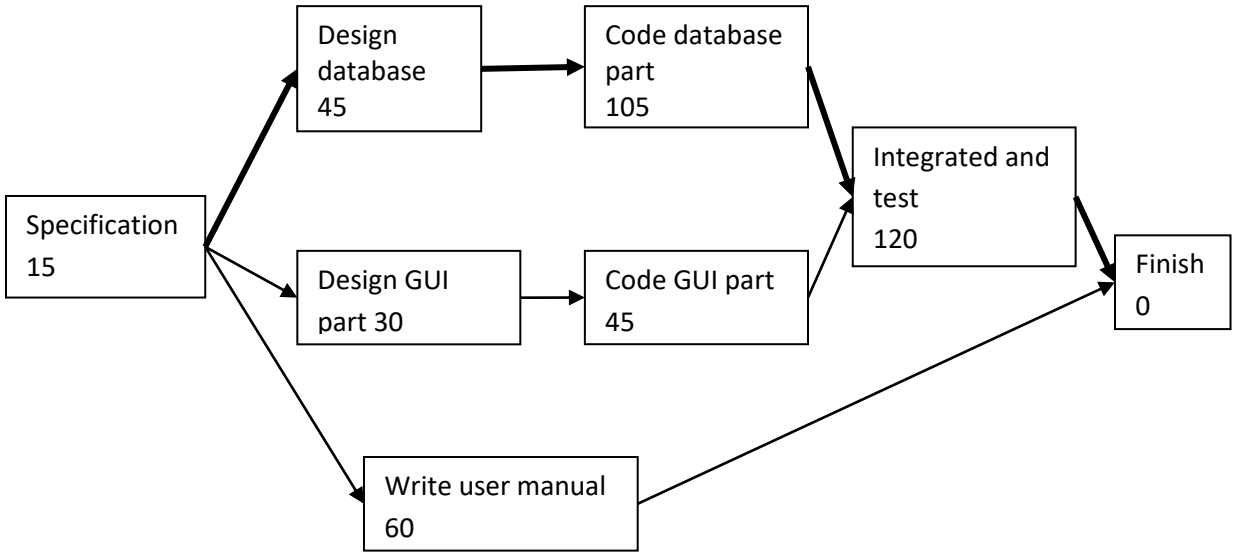
- Project scheduling એ એક important project planning activity છે. તે decide કરે છે કે કયા task ક્યારે હાથ ધરવામાં આવશે. Activity નુ schedule નક્કી કરવા માટે, software project manager ને નીચે મુજબ ની activity કરવાની જરૂર છે
  - ✓ Project ને Complete કરવા માટે જરૂરી બધા task ને Identify કરો.
  - ✓ Large task ને small activities માં Break કરો
  - ✓ Different activities માં dependency નક્કી કરો
  - ✓ Activities ને complete કરવા માટે જરૂરી time duration માટે નુ Estimate કાઢે છે
  - ✓ Activities માટે resource ને Allocate કરો.
  - ✓ જુદી જુદી activity માટે starting અને ending ની date નુ Plan બનાવો.
- દરેક activity ના end ને milestone કહેવામાં આવે છે. Project manager એ milestone ના સમયસર completion ને monitoring કરીને project ની progress ને track કરે છે. જો તે observe કરે છે કે milestone માં delay થવાનું શરૂ થયું છે, તો તેણે activity ને carefully control કરવું પડશે, જેથી overall deadline પુરી થઈ શકે.

વર્ક બ્રેંકડાઉન સ્ટ્રકચર

- Work Breakdown Structure (WBS) નો ઉપયોગ small activity માં વારંવાર આપવામાં આવેલા task set ને decompose કરવા માટે થાય છે. WBS એ problem ને solve કરવા માટેના મુખ્ય task ને represent કરવા માટે notation આપે છે.
- Tree નું root એ problem name દ્વારા label કરેલ છે. Tree ના દરેક node ને smaller activity માં break કરવામાં આવે છે જે node ના children બને છે.
- Leaf level, સુધી દરેક activity ને ફરીથી નાના sub-activities માં decompose કરવામાં આવે છે.
- Time એ project complete કરવા માટે, manager ને large task ને નાના task માં break કરવાની જરૂર છે.

Activity networks અને critical path method

- Project ના WBS representation માં તેમની interdependency સાથેની activity ને રજૂ કરીને activity network માં transfer થાય છે.
- એક activity network project બનાવતી different activity, તેમના estimated duration અને interdependency(figure માં બતાવ્યા પ્રમાણે) દર્શાવે છે. દરેક activity ને rectangular node દ્વારા represent કરવામાં આવે છે અને activity ને દરેક task ની સાથે બતાવવામાં આવે છે.
- Manager વિવિધ task માટેના time duration's ના વિવિધ way નો estimate આપી શકે છે. એક possibility એ છે કે તેઓ વિવિધ task ને duration assign કરી શકે છે.
- Possible alternative એ છે કે engineer પોતે જે activity આપી શકે તે માટેનો time ને estimate કરે. જો કે, કેટલાક manager વિવિધ activity માટે time નો estimate કાઢવાનું પસંદ કરે છે.
- ઘણા manager માને છે કે સારુ schedule engineer ને વધુ સારી અને ઝડપી job કરવા motivate કરે છે. જો કે, careful schedule ન હોય તો તે schedule delay નું કારણ બને છે.
- વધારે પડતાં schedule ને pressure કર્યા વગર task duration ના estimation ને ઓક્ક્સપણે achieve કરવા માટેનો એક સારો way એ છે કે people પોતાના schedule set કરે.

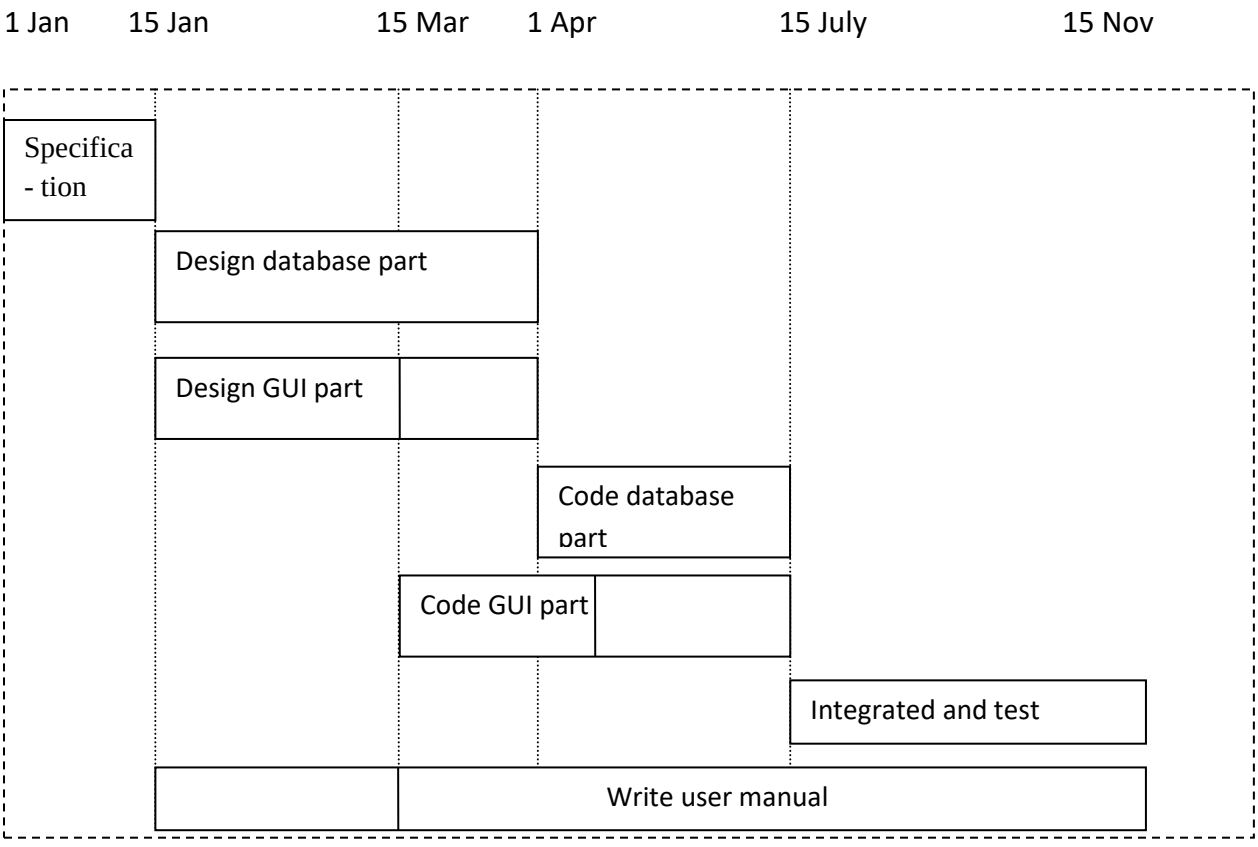


Critical Path Method (CPM)

- CPM એ mathematical calculations પર આધારિત છે અને તેનો ઉપયોગ project activity ને scheduling કરવા માટે થાય છે.
- project manager એ project ની શરૂઆતથી project ની critical activity ને ઓળખે છે
- Critical activity ની series project ના critical path તરીકે ઓળખાય છે. તે network દ્વારા સૌથી longest path છે.

જેન્ટ ચાર્ટ

- જેન્ટ ચાર્ટ નો ઉપયોગ activity ને resource allocate માટે થાય છે. Activity માટે allocate કરવામાં આવેલા resource માં staff, hardware અને software include છે.Gantt chart resource planning માટે ઉપયોગી છે.
- જેન્ટ ચાર્ટ એ special type ના bar chart છે જ્યાં દરેક bar activity ને represent કરે છે. Bar ને time line સાથે draw કરવામાં આવે છે. દરેક bar ની length એ corresponding activity માટેના time plan માટે અનુરૂપ હોય છે.
- જ્યારે દરેક activity શરૂ થાય છે અને end થાય છે ત્યારે વિવિધ activity તમને જોવા માટે allow કરે છે, દરેક activity કેટલો સમય schedule થાય છે, તે સમગ્ર project ની શરૂઆત અને end date હોય છે.
- MIS problem માટે નું જેન્ટ ચાર્ટ representation figure માં બતાવવામાં આવી છે.



Risk Management

- Risk એ loss નું expectation છે, future માં potential problem હોઈ શકે છે અથવા ન પણ હોય શકે. તે information, control અથવા time ની ખામી ના લીધે સામાન્ય રીતે થાય છે. Software development process માં થતાં loss ની possibility ને software risk કહેવામાં આવે છે. Loss કંઈપણ હોઈ શકે છે, production cost માં વધારો, poor quality વાળા software નું development,અને આ બધુ project ને

time એ complete કરવા માટે capable નથી. Software માં risk exist હોય છે કારણ કે તેમાં future ખબર હોતી નથી. અને ઘણી જાણીતી અને unknown વસ્તુઓ છે જે project plan માં ઉમેરી શકાતી નથી.

- Risk management નું aim એ એક project ને affect કરી શકે તેવા તમામ પ્રકારના risk ની સંખ્યા ને ઘટાડવાનું છે. Risk management માં three જરૂરી activity હોય છે:
  (1) Risk identification
  (2) Risk assessment
  (3) Risk containment

Risk Identification

- વિવિધ પ્રકારના risk દ્વારા software project ને અસર થઈ શકે છે. Software project ને અસર કરી શકે તેવા important risk ને વ્યવસ્થિત રીતે ઓળખવા માટે, risk ને જુદા જુદા class માં categorize કરવુ જરૂરી છે.
- Project manager પછી ચકાસી શકે છે કે દરેક class માંથી કયા risk project માટે સંબંધિત છે. Risk ની ત્રણ main કેટેગરી છે જે software project ને અસર કરી શકે છે.

Project risks

- Project ના risk ની ચિંતા એ budgetary, schedule, personnel, resource અને customer-related problem ના form માં બદલાય છે. એક important project માં risk એ schedule છે. Software project નું monitoring અને control કરવું ખૂબ મુશ્કેલ છે.
- જે જોઈ શકાતું નથી એવું કંઈક control કરવું ખૂબ મુશ્કેલ છે. કોઈપણ manufacturing project માટે, જેમ કે car નું manufacturing, project manager એ product ના shape ને જોઈ શકે છે. દાખલા તરીકે, તે જોઈ શકે છે કે engine fit થઈ ગયું છે, તેના પછી door fit થાય છે, અને car paint થઈ રહી છે, વગેરે. તેથી તે સરળતાથી કામની progress નું મૂલ્યાંકન કરી શકે છે અને તેને control કરી શકે છે.
- Product દેખાતું નથી એ એક important reason છે કેમ કે ઘણા software project schedule ના risk ને સહન કરે છે.

Technical risks

- Technical risk માં design, implementation, interfacing, testing અને maintenance ની સમસ્યાઓ નો સમાવેશ થાય છે.
- Technical risk માં સ્પષ્ટ ના હોય તેવું specification, પૂર્ણ ના હોય તેવું specification, બદલાતું રહે તેવું specification, technical અનિશ્ચિતતા include છે. Project વિશે development team ના insufficient knowledge ને કારણે મોટાભાગના technical risk થાય છે.

Business risks

- આ પ્રકારનાં risk માં excellent product કે જે કોઈ ઈચ્છે છે, તેમાં budgetary અથવા personnel commitment ગુમાવવાનું risk વગેરેનો સમાવેશ કરે છે.

Example

- ચાલો આપણે satellite આધારિત mobile communication product ને ધ્યાનમાં લઈએ. Project manager આ project માં ઘણા risk ને ઓળખી શકે છે. અમે તેમને યોગ્ય રીતે classify કરી શકે છે.
  ✓ જો project ના cost નો અંદાજ કરતાં વધારે પ્રમાણમાં વધારો થયો હોય તો શું થશે? project risk

- ✓ જો mobile phone ખૂબ large થઈ જાય તો લોકો તેને સહેલાઈથી એક જગ્યા થી બીજી જગ્યાએ લઈ જાઈ શકશે નહિ તો શું થશે? Business risk

Risk Assessment

- Risk assessment માં risk ને ઓળખવા, તેનું analyzing કરવાનો સમાવેશ થાય છે અને પછી analysis ના આધારે તેમને priority સોંપી દેવામાં આવે છે.
- Risk assessment નો goal તેમના damage ના term માં risk ને ક્રમ આપવો છે. Risk assessment માટે, પ્રથમ દરેક risk ને બે રીતે rate કરવામાં આવે છે.
  - ✓ Risk આવવાની સંભાવના સાચું પડે ત્યારે(r તરીકે દર્શાવવું).
  - ✓ તે risk સાથે સંકળાયેલ problem નું result (s તરીકે દર્શાવવું).
- આ બે factor ને આધારે, દરેક risk ની priority ની ગણતરી કરી શકાય છે.

Risk Containment

- Project ના બધા ઓળખાયેલા risk ને ઓળખ્યા પછી, સૌથી વધુ નુકસાનકારક અને સંભવિત risk ને રોકવા માટે plan બનાવવા જરૂરી છે.
- જુદા જુદા risk ને વિવિધ control process ની જરૂર છે. હકીકતમાં, મોટાભાગના risk ને risk handling માં project manager ના ભાગ પર expert ની જરૂર છે.
- Risk control ના plan માટે ત્રણ મુખ્ય category છે:
  - ✓ **Avoid the risk:** આમાં work ના scope ને ઘટાડવા માટે તથા requirements ને બદલવા માટે customer સાથે discuss કરવું એવા ઘણા form લઈ શકે છે.
  - ✓ **Transfer the risk:** આ strategy માં third party દ્વારા develop થયેલા risky component ને મેળવે છે.
  - ✓ **Risk reduction:** risk ના કારણે નુકસાનને રોકવા માટેના way ના planning નો સમાવેશ કરવામાં આવે છે.

Coding Standard અને coding Guidelines

- Good software development organization ને સામાન્ય રીતે તેમના programmer's ને coding standards તરીકે ઓળખાતા coding ના કેટલાક સારી રીતે define કરેલા અને standard style ને follow કરવાની જરૂર હોય છે.
- મોટા ભાગના software development organizations તેમના પોતાના coding standards ને follow કરે છે.
- coding ની standard style નો purpose નીચે મુજબ છે:

✓ જુદા જુદા engineers દ્વારા લખેલા codes નું સરખુ look.
✓ Code સમજવા માટે easy છે.
✓ સારી programming practice થાય છે.

Code Review (કોડ રીવ્યુ)

- બધી syntax error ને દૂર કર્યા પછી model માટે Code review કરવામાં આવે છે.

- Coding error માં ઘટાડો અને high quality code બનાવવા માટે આ cost વગરની strategies છે.
- બે પ્રકારની code review technique એ code inspection અને code walk through છે.

કોડ વોક થ્રુ

- કોડ વોક થ્રુ એ Code analysis technique છે.
- Module માં code કર્યા પછી આ technique માં successfully code compile થાય અને બધી syntax errors દૂર થાય પછી આ technique કરવામાં આવે છે.
- Development team ના કેટલાક member ને code read કરવા અને સમજવા માટે વોક થ્રુ meeting દ્વારા થોડા દિવસ પહેલા code આપવામાં આવે છે.
- દરેક member કેટલાક test case select કરે છે અને code દ્વારા execution માં મુકવાની મંજૂરી આપે છે .
- વોક થ્રુ ના મુખ્ય objectives એ code માં algorithmic અને logical error ને શોધવાનું છે
- Module ના coder હાજર હોય ત્યારે વોક થ્રુ meeting માં discuss કરવા માટે member તેમને find કરેલી error ને note down કરે છે.
- આ ટેકનીક ની થોડી ગાઈડલાઈન્સ નીચે આપેલી છે:
- તેમાં ત્રણથી સાત member નો સમાવેશ થવો જોઈએ
- Error ને find કરવા પર ના Discussion પર focus કરવુ જોઈએ અને errors ને કેવી રીતે fix કરવી તેના પર નહીં.
- Manager ને walk through meetings ને attend ન કરવી જોઈએ.

Code Inspection (કોડ ચેકિંગ)

- Code inspection નો ધ્યેય improper programming ને કારણે કેટલીક સામાન્ય પ્રકારની ભૂલો શોધવાનો છે.
- Code inspection દરમિયાન code માં કોઈ ચોક્કસ પ્રકારની error ની હાજરી છે કે નહીં તે માટે check કરવામાં આવે છે.
- આ પ્રકારની mistake ને code માં શોધીને Error મળી આવશે.
- Code inspection દરમિયાન Coding standard પણ check કરવામાં આવે છે.
- સારી software development company બધી સામાન્ય રીતે તેમના engineer દ્વારા કરવામાં આવતી વિવિધ પ્રકારની error ને collect કરે છે અને મોટાભાગે વારંવાર કરવામાં આવેલી error ના પ્રકારને ઓળખે છે.
- Possible error ને check કરવા માટે code inspection દરમ્યાન સામાન્ય રીતે કરવામાં આવેલી error ના List નો ઉપયોગ કરી શકાય છે.
- નીચે કેટલીક error નું list છે જે code inspection દરમિયાન check કરી શકાય છે.
- Initialize ના કર્યા હોઈ એવા વેરીયેબલ નો ઉપયોગ
- Terminate ના થતી લૂપ્સ.
- Incomputable હોઈ એવા અસાઈમેન્ટ

- Array ની સાઈઝ out of bounds હોઈ.
- Storage નુ improper અલોકેશન અને ડીલોકેશન હોવું.
- procedure calls માં actual અને formal parameter વચ્ચેના Mismatches
- Operator વચ્ચે ખોટા logical operator અથવા ખોટી precedence નો ઉપયોગ.