chapter 5

1&Final

Types of software development complexity

1. structural
2. conceptual
3. computional

structural complexity

1. size :- measure LOC or FP
2. cyclomatic complexity (CC) :- control flow
3. Halstead's complexity :- measure the number of operands and operator

Information Flow :-

1&Final

4. measure Flow of data into and out of modules
5. system complexity

18 FINAL

$\boxed{\text{what is sweetspot ?}}$

→ In the sweet spot, detect density /LOC (defect per) is Lowest. 200-750 LOC per module

$$\boxed{D_m(S) = \frac{a}{S}}$$

S = module size

Dm = Defect Density (in detEcts/LOC)

c, a = empirically derived constant

Total ; $\boxed{D(S) = \frac{a}{S} + b + c*S}$

$$S_{min} = \sqrt{a/c}$$

S_{min} range (200 - 400)

a good software should be less than 2 detects per LOC

— True

Cyclomatic Complexity (CC)

– is a measure of control flows within module

$$\boxed{CC = V(g) = e - n + 2}$$

e = number of edge

n = node of node

g = control flow graph

$$\boxed{CC = V(g) = bd + 1}$$

bd = binary decision → No of innerloops in graph

— o —

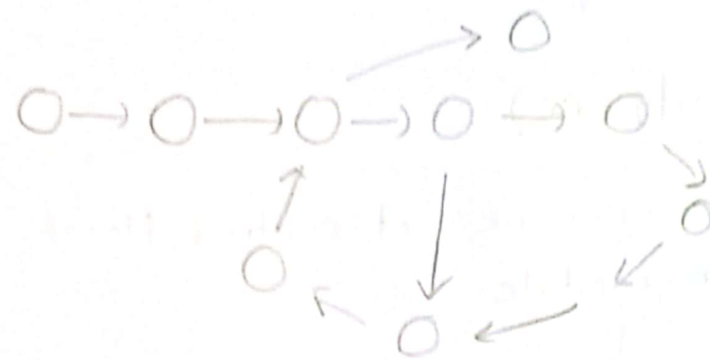
```
char strncat (char des, ....)
{  char * temp = dest;
 if (count) {
   while (dest)
         dest++;
   while (dest++ = src++) {
    if (--count ==0) {
        dest =0
        break
    }
 }
```

1 if, 2 while, 3 while, 4 if

∴ CC = binary decision + 1

= 4 + 1

= 5



$cc = e - n + 2$

$= 10 - 9 + 2 = 3$

$cc = bd + 1$

$= 2 + 1 = 3$

bd = it
case
while
Repeat
Loop

—o—

cyclomatic complexity represents the minimum number of test required to execute every path in code

18 cc > 20 :- definitely cause for concern

cc > 50 :- cause for alarm

measure of unstructuredness of code

ECC = esential cyclomatic complexity

= a piece of code after removing
structured constaints
(if case, while, repeat, sequence)

Halstead metrics

Length $N = N_1 + N_2$

Vocabulary $\Rightarrow n = n_1 + n_2$

Volume, $V = N(\log_2(n))$

Difficulty, $D = (n_1/2) * (N_2/n_2)$

Effort, $E = D * V$

— o —

n_1 = number of distinct operators

n_2 = number of distinct operands

N_1 = total Number of operators

N_2 = total Number of operands

variables
constants
strings

```
char * strncat (char *dest , const char *src , size)
{   char *temp = dest ;
    if (count) {
        while (*dest)
            dest++ ;
        while((*dest++ == *src++)) {
            if(--count == 0 ) {
                *dest = '\0' ;
                break;
            }
        }
    }  return temp;
}
```

$n_1 =$ {}, ++, if, while, break, == , * , return

;, =, --

= 11

$n_2 =$ temp, dest, count, 0, '\0', src

= 6

Total No of n_1 = $N_1 \Rightarrow 26$

Total No of $N_2 \Rightarrow 10$

$\therefore$ Length $N = N_1 + N_2$

$= 26 + 10$

$= 36$

$\therefore$ Vocabulary, $\overset{n}{\cancel{V}} = n_1 + n_2$

$= 11 + 6 = 17$

$\therefore$ Volume, $V = N(\log_2(n))$

$= 36(\log_2(17))$

$= 147$

$\therefore$ Difficulty $D = (11/2) * (10/6)$

$= 9.2$

Effort $D * V = 1348$

Infonmation Flow matnic

$$IFC = (fanin \times fanout)^2$$

$$\text{weighted } IFC = length \times (fanin \times fanout)^2$$

Fanin = local flow into a pnocedune

\+

numben of data stnuctunes ~~that~~

~~the pnocedune updates~~

which the pnocedune netnives

Fanout = local flow fnom a pnocedune

\+

Numben of data stnuctune

the pnocedune updates

length = numben of sounce statement

in pnocedune (without comment)

Previous code

```
chan *stnncat (chan *dest,
               const chan *snc, size-t
               count) {
                          flow in = 3

chan *temp = dest          fan/flow out = 1
if (count) {
    while (*dest)                        flow need
        dest++;                            = 3

    while ((*dest++ = *snc++)) {
        if(-count == 0) {
flow        *dest = '\0';
written     break;
= 2     }
    }
}

netunn temp
```

No of flows in = 3

No of data structure
read = 3

fanin = 3+3 = 6

No of flow out = 1

No of data structure written = 2

fanout = 1+2
= 3

∴ weight IFC = 14 * (6*3)²

Maintainability Index :-

$$MI = 171 - 5.2\ln(aV) - 0.23aV(g') - 16.2\ln(aLOC) + 50\sin[(2.4 * per\,cm)^{\frac{1}{2}})]$$

aV = average Halstead volume V per module

$V(g')$ = average extended cyclomatic complexity per module

$aLOC$ = average lines of code per module

perCM = average percent of lines of comments per module

Highly maintable → > 85

moderate → > 65 and ≤ 85

difficult to maintain → < 65

Agnesti cand Glass complexity metric

based on structonal design

modulanity pninciple coupling and conhession

$$C_t = S_t + D_t$$

C_t = system complexity

$$\therefore S_t = \Sigma (f(i))^2$$

S_t = structural / intenmodule complexity

$$\therefore D_t = \Sigma \frac{D(i)}{n}$$

D_t = data model intnomodule complexity

$$\therefore D(i) = \frac{V(i)}{f(i)+1}$$

$f(i)$ = fanout of module i

$$\therefore RSC = \frac{S_t}{N} + \frac{D_t}{n}$$

Ck suite

Chidamben kemenen metnics

consists 6 metnics that measone class size, complexity, use of inhenitance, coupling, cohession and collabonation between classes.

1. WMC (weighted methods pen class)
2. DIIT (Depth of Inhentance tnee)
3. NOC (Numben of children)
4. CBO (coupling between object classes)
5. RFC (Response for class)
6. LCOM (Lack of cohession method)