Metrika programske opreme

- Omogoča kvantitativno ocenjevanje programske opreme
- Merjenje: določanje številskih vrednosti lastnostim programske opreme
- 1. korak: določanje postopka in meritev programske opreme
- 2. korak: zajem podatkov, potrebnih za izvršitev meritev
- 3. korak: analiza meritev za določanje kvalitete PO in potrebnih popravkov v različnih modelih, izvorni kodi ali v testnih primerih

Kvaliteta programske opreme

- Kako definirati kvaliteto? Ujemanje z eksplicitno izraženimi funkcionalnimi in performančnimi zahtevami in eksplicitno dokumentiranimi standardi, ter implicitnimi karakteristikami, ki se pričakujejo od vsake profesionalne programske opreme
- Faktorji, ki vplivajo na kvaliteto:
  - Direktno merljivi (npr. število odkritih napak med testiranjem)
  - Indirektno merljivi (npr. enostavnost uporabe)
- McCallova definicija faktorjev, ki vplivajo na kvaliteto: pravilnost, zanesljivost, učinkovitost, integriteta, enostavnost uporabe, enostavnost vzdrževanja, fleksibilnost, enostavnost testiranja, prenosljivost, enostavnost ponovne uporabe, interoperabilnost
- ISO 9126 definicija: funkcionalnost (ustreznost, natančnost, interoperabilnost, varnost), zanesljivost, enostavnost uporabe, učinkovitost, enostavnost vzdrževanja, prenosljivost

Metrike za ocenjevanje kvalitete PO

- Vse metrike so indirektne!
- Ne merimo direktno kvalitete ampak neko manifestacijo kvalitete
- Problem je v določanju povezanosti med tem, kar merimo in kvaliteto
- Postopek: formulacija, zbiranje podatkov, izračun metrike, interpretacija, reakcija
- Lastnosti metrike:
  - enostavna, izračunljiva, konsistentna in objektivna
  - neodvisna od programskega jezika
  - ima zaželjene matematične lastnosti
  - spremembe v PO se adekvatno odražajo v spremembi metrike
  - empirično preverjena

Metrike za ocenjevanje kvalitete PO

- Metrike analitičnega modela: funkcionalnost, obseg, kvaliteta specifikacije
- Metrike načrta: arhitekturne metrike, komponentne metrike (kompleksnost), metrike uporabniškega vmesnika (uporabnost), OO metrike (karakteristike razredov)
- Metrike izvorne kode: Halsteadove metrike (metrike kompleksnosti, metrike dolžine)
- Metrike testiranja: metrike pokrivanja poti skozi kodo, metrike na osnovi najdenih napak, efektivnost testiranja, procesne metrike

Metrike enega tipa se lahko uporabljajo za kasnejše aktivnosti

Metrike analitičnega modela

- FP (function point) metrika
  - Število zunanjih vhodov (EI)
  - Število zunanjih izhodov (EO)
  - Število zunanjih poizvedb (EQ)
  - Število internih logičnih virov podatkov (ILF)
  - Število zunanjih virov podatkov (EIF)

Podatek	Št.		Enostavno	Povpr.	Kompleksno		
EI		x	3	4	6	=	
EO		x	4	5	7	=	
EQ		x	3	4	6	=	
ILF		x	7	10	15	=	
EIF		x	5	7	10	=	
Skupaj							

Metrike analitičnega modela

- FP = Skupaj x (0.65 + 0.01 x sum)
- Sum = vsota ovrednotenja 14-tih vprašanj (vsako od 0 do 5):

1) Ali sistem zahteva zanesljivo izdelovanje varnostnih kopij podatkov in restavriranje le teh?
2) Ali je potrebna posebna komunikacija za prenos podatkov v ali iz aplikacije?
3) Ali se uporabljajo distrubuirane funkcije?
4) Je čas izvajanja kritičnega pomena?
5) Ali bo aplikacija tekla v obstoječem, precej obremenjenem okolju?
6) Ali sistem zahteva online vnos podatkov?
7) Ali online vnos podatkov zahteva transakcijo preko več dialogov ali operacij?
8) Se ILFji popravljajo z online transakcijami?
9) So vhodi, izhodi ali poizvedbe kompleksni?
10) Je interno procesiranje kompleksno?
11) Naj bi bila koda primerna za ponovno uporabo?
12) Je inštalacija vključena v načrt?
13) Je aplikacija načrtovana za inštalacijo v različnih okoljih?
14) Je aplikacija načrtovana za enostavno nadgradnjo in za enostavnost uporabe?

Na osnovi FP vrednosti lahko ocenimo kompleksnost in čas izdelave kode!

Metrike analitičnega modela

- Kvaliteta specifikacije:

Število zahtev v specifikaciji: $n_r = n_f + n_{nf}$

Nedvoumnost zahtev: $Q_1 = n_{ui} / n_r$

Kompletnost zahtev: $Q_2 = n_u / (n_i \times n_s)$

Stopnja validacije zahtev: $Q_3 = n_c / (n_c + n_{nv})$

n_f – število funkcionalnih zahtev

n_{nf} – število nefunkcionalnih zahtev

n_{ui} – število zahtev, o katerih imajo vsi naročniki enako interpretacijo

n_u – število unikatnih funkcionalnih zahtev

n_i – število vhodov

n_s – število stanj

n_c – število validiranih zahtev

n_{nv} – število nevalidiranih zahtev

Arhitekturne metrike

Hierarhične arhitekture:

Strukturna kompleksnost modula i: $S(i) = f_{out}(i)^2$

Podatkovna kompleksnost modula i: $D(i) = v(i) / (f_{out}(i) + 1)$

Sistemska kompleksnost modula i: $C(i) = S(i) + D(i)$

$f_{out}(i)$ – število modulov, ki jih uporablja modul i

v(i) – število vhodnih in izhodnih spremenljivk modula i

Arhitekturne metrike

S1 = število vseh modulov

S2 = število modulov, katerih pravilno delovanje zavisi od vnosa podatkov ali pa generirajo podatke, ki se uporabljajo v drugih modulih

S3 = število modulov, katerih pravilno delovanje zavisi od predprocesiranja

S4 = število vseh podatkov v bazi (podatkovni objekti + vsi atributi)

S5 = število unikatnih podatkov v bazi

S6 = število segmentov v bazi (različni zapisi)

S7 = število modulov z eno vstopno in izstopno točko

Struktura programa: D1 = 1 ali 0

Neodvisnost modulov D2 = 1 – S2/S1

Moduli, neodvisni od predprocesiranja D3 = 1 – S3/S1

Velikost baze D4 = 1 – S5/S4

Razdeljenost baze D5 = 1 – S6/S4

Število enostavnih modulov D6 = 1 - S7/S1

Indeks kvalitete strukture načrta:
$DSQI = \Sigma\ w_i \times D_i$

w_i = relativna utež

Če je DSQI bistveno nižji od povprečja, je treba popraviti načrt

OO metrike

- CK metrika:
  - Utežena kompleksnost metod v razredu: WMC = Σc_i
  - Globina dedovanja: DIT
  - Število direktnih podedovanih razredov: NOC
  - Medrazredna prepletenost: CBO (število razredov s katerimi razred sodeluje)
  - Število reakcijskih metod: RFC
  - Pomanjkanje kohezije v metodah: LCOM (število metod, ki uporabljajo enega ali več istih atributov)

OO metrike

- MOOD metrika:
  - Faktor dedovanja metod: $MIF = \Sigma M_h(C_i) / \Sigma M_a(C_i)$
  - $M_a(C_i) = M_d(C_i) + M_h(C_i)$
  - Faktor prepletenosti: $CF = \Sigma_i \Sigma_j S(C_i, C_j) / (T_c^2 - T_c)$
  - $S(C_i, C_j) = 1$, če razred C_i sodeluje s C_j
- Lorenz-Kidd metrika:
  - Velikost razreda: CS (število metod, število atributov)
  - Število dodanih metod in atributov glede na bazni razred: NOA

Komponentne metrike

- Metrika prepletanja: $m_c = k / M$
- $M = d_i + (a \times c_i) + d_o + (b \times c_o) + g_d + (c \times g_c) + w + r$
  - d_i – število vhodnih parametrov
  - c_i – število vhodnih kontrolnih parametrov
  - d_o – število izhodnih parametrov
  - c_o – število izhodnih kontrolnih parametrov
  - g_d – število globalnih spremenljivk, ki so uporabljene kot podatki
  - g_c – število globalnih kontrolnih spremenljivk
  - w – število uporabljenih komponent
  - r – število komponent, ki to komponento uporabljajo
- Konstante k, a, b in c se določijo empirično.
- Metrika kompleksnosti: najbolj pogosto se uporablja ciklomatična kompleksnost

Metrike uporabniškega vmesnika

- Metrike kohezije: če določeno okno/dialog prikazuje podatke, ki so del enega samega podatkovnega objekta → visoka stopnja kohezije
- Metrike interakcije, ki merijo čas, ki je potreben za dokončanje določene naloge
- Najboljša metrika UV: odziv uporabnikov

Metrike izvorne kode in testiranja

- Halsteadova metrika
  - Metrika dolžine: $N = N_1 + N_2$
  - Težavnost programa/metode: $D = n_1 \times N_2 / (2 \times n_2)$
  - Napor, potreben za testiranje: $E = D \times N \times \log_2(n_1 + n_2)$
    - n_1 – število različnih operatorjev v programu/metodi
    - n_2 – število različnih operandov v programu/metodi
    - N_1 – število vseh operatorjev v programu/metodi
    - N_2 – število vseh operandov v programu/metodi
- Metrika za vzdrževanje:
  - Zrelost PO: $SMI = (M_T – (F_a + F_c + F_d))/M_T$
    - M_T – število modulov v trenutni verziji
    - F_c – število spremenjenih modulov v tej verziji
    - F_a – število dodanih modulov v tej verziji
    - F_d – število izbrisanih modulov v tej verziji