VISOKA TEHNIČKA ŠKOLA

STRUKOVNIH STUDIJA SUBOTICA

KVALITET SOFTVERA

SEMINARSKI RAD IZ PREDMETA SOFTVERSKO INŽENJERSTVO

MENTOR

Janoš Šimon

STUDENT

Nemanja Mamužić

16118221

Subotica, Decembar 2020

SADRŽAJ

SADRŽAJ 1

UVOD 2

Pristup atributima kvaliteta softvera 3

Analiza zasnovana na kodu 4

Pouzdanost 5

Efikasnost 5

Sigurnost 6

Zaključak 7

UVOD

U kontekstu softverskog inženjerstva, kvalitet softvera odnosi se na dva povezana, ali različita pojma:

-Funkcionalni kvalitet softvera odražava se koliko je u skladu sa datim dizajnom il kolikoi se prilagođava njemu, na osnovu funkcionalnih zahteva ili specifikacija. Taj atribut se takođe može opisati kao podobnost softverskog dela ili kao upoređivanje sa konkurentima na tržištu kao vredan proizvod.

-Strukturni kvalitet softvera je stepen do kog je proizveden ispravan softver.
Strukturni kvalitet softvera odnosi se na to kako ispunjava vanfunkcionalne zahteve koji podržavaju isporuku funkcionalnih zahteva, kao što su robusnost ili održivost. Ima mnogo više veze sa stepenom rada softvera po potrebi.

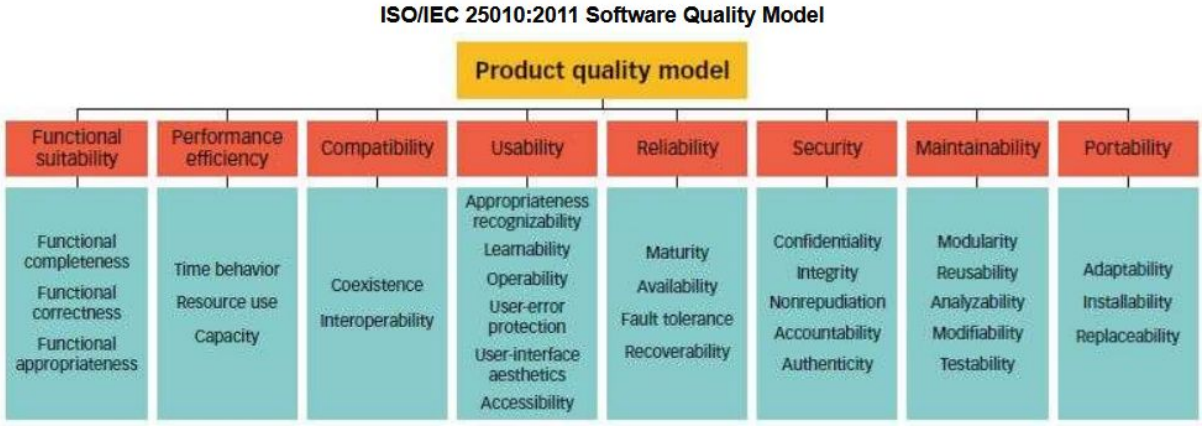
Istorijski gledano, struktura, klasifikacija i terminologija atributa i metrika primenljivi na upravljanje kvalitetom softvera izvedeni su ili izdvojeni iz ISO 9126-3 standarda i sledećeg modela kvaliteta ISO 25000: 2005, takođe poznatog kao SQuaRE.

Na osnovu ovih modela, Konzorcijum za kvalitet informacionog softvera (CISQ-Consortium for IT Software Quality) definisao je pet glavnih poželjnih strukturnih karakteristika potrebnih da bi neki softver pružio poslovnu vrednost: pouzdanost, efikasnost, sigurnost, održivost i (odgovarajuća) veličina.

Pristup atributima kvaliteta softvera

Ovaj pristup kvalitetu softvera najbolje pokazuju modeli fiksnog kvaliteta, kao što je ISO / IEC 25010: 2011. Ovaj standard opisuje hijerarhiju od osam karakteristika kvaliteta, od kojih se svaka sastoji od podkarakteristika:

1. Funkcionalna pogodnost
2. Pouzdanost
3. Operativnost
4. Efikasnost
5. Bezbednost
6. Kompatibilnost
7. Održivost
8. Prenosivost



Pored ovog postoje još pet kategorija koje definišu kvalitet softvera u upotrebi:

1. Efektivnost
2. Efikasnost
3. Zadovoljstvo
4. Sigurnost
5. Upotrebljivost

Merenje kvaliteta softvera motivisano je iz najmanje dva razloga:

-Upravljanje rizikom
Neuspeh softvera izazvao je više nego neprijatnosti. Softverske greške prouzrokovale su smrtne slučajeve. Uzroci su se kretali od loše dizajniranog korisničkog interfejsa do direktnih programskih grešaka.

-Upravljanje troškovima

Kao i u bilo kojoj drugoj oblasti inženjerstva, aplikacija sa dobrim strukturnim kvalitetom softvera košta manje za održavanje i lakša je za razumevanje i promenu kao odgovor na hitne poslovne potrebe. Loš strukturni kvalitet aplikacija u osnovnim poslovnim aplikacijama (kao što je planiranje resursa preduzeća (ERP), upravljanje odnosima sa kupcima (CRM) ili veliki sistemi za obradu transakcija u finansijskim uslugama) može da rezultira prekoračenjem troškova i rasporeda i stvara potrebu za korekcijom.
Štaviše, loš strukturni kvalitet u snažnoj je korelaciji sa velikim prekidima poslovanja usled oštećenih podataka, prekida rada aplikacija, kršenja bezbednosti i problema sa performansama.

Međutim, razlika između merenja i poboljšanja kvaliteta softvera u ugrađenom sistemu (sa naglaskom na upravljanje rizikom) i kvaliteta softvera u poslovnom softveru (sa naglaskom na upravljanju troškovima i održavanjem) postaje pomalo irelevantna.

Analiza zasnovana na kodu

Mnoge postojeće softverske mere računaju strukturne elemente aplikacije koji su rezultat raščlanjivanja izvornog koda za takve pojedinačne instrukcije
Merenje kvaliteta softvera odnosi se na kvantifikaciju u kojoj meri sistem ili softver funkcioniše u odnosu na zahteve. Analiza se može izvršiti korišćenjem kvalitativnog ili kvantitativnog pristupa ili kombinacijom oba da bi se dobio zbirni prikaz.

Ovaj pogled na kvalitet softvera na linearnom kontinuumu mora biti dopunjen identifikacijom diskretnih kritičnih grešaka u programiranju. Ove ranjivosti možda neće proći test slučaj, ali su rezultat loših praksi koje u određenim okolnostima mogu dovesti do katastrofalnih prekida rada, pogoršanja performansi, proboja bezbednosti, oštećenih podataka i bezbroj drugih problema koji čine dati sistem faktički neprikladanim za upotrebu bez obzira na njegovu ocenu zasnovanu na agregatnim merenjima.
Poznati primer ranjivosti je Common Veakness Enumeration, [31] spremište ranjivosti u izvornom kodu koje čine aplikacije izložene kršenju bezbednosti. Merenje kritičnih karakteristika aplikacije uključuje merenje strukturnih atributa arhitekture aplikacije, kodiranja i redovne dokumentacije.
Čak i dinamičke karakteristike aplikacija kao što su pouzdanost i efikasnost performansi imaju svoje uzročne korene u statičkoj strukturi aplikacije. Analiza i merenje strukturnog kvaliteta vrši se kroz analizu izvornog koda, arhitekture, softverskog okvira, šeme baze podataka u odnosu na principe i standarde koji zajedno definišu konceptualnu i logičku arhitekturu sistema.
Ovo se razlikuje od osnovne, lokalne analize koda na nivou komponente, koju obično izvode razvojni alati koji se uglavnom bave razmatranjima implementacije i presudni su tokom otklanjanja grešaka i aktivnosti testiranja.

Pouzdanost

Osnovni uzroci loše pouzdanosti nalaze se u kombinaciji nepoštovanja dobre arhitektonske prakse i prakse kodiranja. Ova neusklađenost se može otkriti merenjem statičkih atributa kvaliteta aplikacije. Procenom statičkih atributa na kojima se zasniva pouzdanost aplikacije daje se procena nivoa poslovnog rizika i verovatnoće potencijalnih kvarova i kvarova u aplikaciji kada se pokrene.

Procena pouzdanosti zahteva proveru najmanje sledećih najboljih praksi i tehničkih svojstava softverskog inženjerstva:
-Praksa arhitekture aplikacija
-Kodiranje prakse
-Složenost algoritama
-Kompleksnost programskih praksi
-Usklađenost sa najboljim praksama objektno orijentisanog i strukturiranog programiranja (kada je primenljivo)
-Odnos ponovne upotrebe komponenata ili uzoraka
-Prljavo programiranje
-Rukovanje greškama i izuzecima (za sve slojeve - GUI, logika i podaci)
-Višeslojna usklađenost dizajna
-Upravljanje granicama resursa
-Softver izbegava obrasce koji će dovesti do neočekivanog ponašanja
-Softver upravlja integritetom i doslednošću podataka
-Nivo složenosti transakcija

U zavisnosti od arhitekture aplikacije i nezavisnih komponenti koje se koriste (kao što su spoljne biblioteke ili framework), prilagođene provere treba definisati u skladu sa linijama navedenim u gornjoj listi najboljih praksi kako bi se obezbedila bolja procena pouzdanosti isporučenog softvera.

Efikasnost

Kao i kod pouzdanosti, uzroci neefikasnosti performansi često se nalaze u kršenjima dobre arhitektonske prakse i prakse kodiranja koja se mogu otkriti merenjem statičkih atributa kvaliteta aplikacije.
Ovi statički atributi predviđaju potencijalna uska grla u operativnim performansama i buduće probleme skalabilnosti, posebno za aplikacije kojima je potrebna velika brzina izvršavanja za rukovanje složenim algoritmima ili ogromnim količinama podataka.

Procena efikasnosti performansi zahteva proveru najmanje sledećih najboljih praksi i tehničkih atributa softverskog inženjerstva:
-Praksa arhitekture aplikacija
-Odgovarajuće interakcije sa skupim i / ili udaljenim resursima
-Učinak pristupa podacima i upravljanje podacima
-Upravljanje memorijom, mrežom i diskovnim prostorom
-Praksa Kodiranja

-Usklađenost sa najboljim praksama objektno orijentisanog i strukturiranog programiranja (prema potrebi)
-Usklađenost sa najboljim praksama SQL programiranja

Sigurnost

Većina sigurnosnih ranjivosti rezultat je lošeg kodiranja i arhitektonskih praksi, poput SQL Injection ili Cross-Site Scripting-a. Oni su dobro dokumentovani u spiskovima koje vode CVE, [33] i SEI / Computer Emergenci Center (CERT) na Univerzitetu Carnegie Mellon. Procena sigurnosti zahteva bar proveru sledećih najboljih praksi i tehničkih atributa softverskog inženjerstva:
-Praksa arhitekture aplikacija
-Višeslojna usklađenost dizajna
-Najbolje prakse u vezi sa bezbednošću (provjera unosa, SQL Injection, Cross-Site Scripting)
-Praksa programiranja (nivo koda)
-Rukovanje greškama i izuzecima
-Najbolje prakse u bezbednosti (pristup funkcijama sistema, kontrola pristupa programima)

Zaključak

I ako postoje različiti stavovi po pitanju šta je to kvalitet softvera, kako se meri i ocenjuje, jedno je sigurno, a to je da je kvalitet softvera jedan od bitnijih činitelja funkcionalnog softvera