Zbornik radova Fakulteta tehničkih nauka, Novi Sad

UDK: 621.38
DOI: https://doi.org/10.24867/24BE29Vuckovic

UTICAJ TESTIRANJA NA KVALITET I ROBUSNOST SOFTVERA

TESTING INFLUENCE ON QUALITY AND ROBUSTNES OF SOFTWARE

Lazar Vučković, Vladimir Rajs, *Fakultet tehničkih nauka, Novi Sad*

Oblast – ELEKTROTEHNIKA I RAČUNARSTVO

Kratak sadržaj – *U ovom radu, predstavljen je proces razvoja softvera i test okruženja čiji je cilj pronalaženje defekata i poboljšanje kako kvaliteta tako i robusnosti razvijenog softvera. Priliko testiranja korišćeno je više test nivoa kako bi se otkrili potencijalni defekti u svim fazama razvoja. Prilikom razvoja softvera korišćen je V model.*

Ključne reči: *Softverska arhitektura, Unit test, Integracioni test, Sistem test*

Abstract – *In this paper, proces of software development and testing enviroment is presented, with the aim of finding defects, improvement of software quality and robustnest. In testing phase, several test levels were implemented, in order to find potential defects in all development phases. In proces of software development V-model is used.*

Keywords: *Software architecture, Unit test, Integration test, System test*

1. UVOD

U eri ekspanzije programabilnih digitalnih komponenti, mikrokontrolera, mikropocesora, pametnih uređaja, značaj softvera dramatično raste. Tehnološki napredak koji prati današnje društvo zasniva se na svakodnevnim inovacijama koje imaju za cilj poboljšanje životnog standarda.

Tipičan primer ove ekspanzije jeste razvoj automobilske industrije gde se broj inovacija povećava sa svakim novim modelom automobila. Da bi se neka od inovacija sprovela u delo, neophodno je pored hardverskog osmisliti i softversko rešenje. Tada softver postaje proizvod za sebe koji treba dizajnirati, implementirati i testirati.

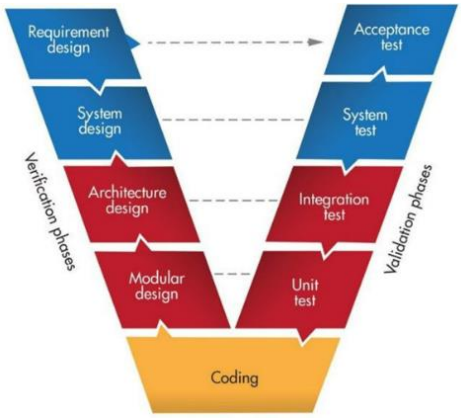
Ako se testiranju pristupi temeljno i u što ranijoj fazi razvoja, nepravilnosti na koje se nailazi biće lakše i jeftinije ukloniti. Zbog efikasnoti pri uklanjanju defekata u ranoj fazi razvoja, u praksi se paralelno pristupa razvoju test okruženja sa razvojem softvera.

U želji za nastajanjem optimalnog rešenja, stvoreni su modeli koji su bili ili su još uvek opšte prihvaćeni, a neki od njih su *waterfall* (vodopad)*,V* (V)*, SDCL, Agile* (agilni)*, Incremental* (inkrementalni) model...

NAPOMENA:
Ovaj rad proistekao je iz master rada čiji mentor je bio dr Vladimir Rajs, vanr. prof.

2. V model

Predstavljen je 1980-ih godina i predstavlja grafičku reprezentaciju razvoja softvera, a trenutno je najzastupljeniji model u okviru automotiv industrije. Ime ovog modela potiče od reči validacija i verifikacija, a formacija faze razvoja predstavljena je latiničnim slovom v. Ovaj model predstavlja sekvencijalni proces kod kog se neka od faza može izvršiti samo u slučaju da je prethodna završena.



Slika 1. *Grafički prikaz V modela*

3.Test tehnike

Tokom kreiranja test plana, a pre početka pisanja testova, neophodno je odlučiti, na osnovu funkcionalnosti koju testiramo, koju test tehniku želimo da primenimo. Nisu sve tehnike primenljive na svim test nivoima ili ne pružaju najefikasnije rešenje.

Da bi se pronašlo što je moguće više defekata, u praski se veoma često pristupa kombinovanju tehnika. Tehnike koje se koriste u praksi su *Black box*, *White box* i iskustveni pristup testiranju.

3.1 Black box

Black box tehnike podrzumevaju testiranje softvera ne znajući ništa o strukturi koda. Testovi koji se oslanjaju na korišćenje ove tehnike, koriste se za ispitivanje ponašanja sistema.

Testovi se pišu prema specifikacijama koje opisuju kako će sistem reagovati na osnovu ulaznih signala i koji su očekivani rezultati.

U *black box* tehike spadaju *Equivalence partition, Boundary value analysis, Decision table, State transition* i *Use case* tehnika.

3.2 White box

White box testiranje se fokusira na strukturu softvera i zahteva poznavanje programskog jezika u kome je pisan kod. Ove metode testiranja se koriste u slučaju provere arhitekture, dizajna i implementacije. *White box* tehnike mogu biti *Control flow, Data flow, Statement coverage, Decision coverage, Branch testing i Path testing* tehnika.

3.3 Iskustven pristup testiranja

Ovaj način testiranja oslanja se na iskustvo, znanje i intuiciju testera. Ovakvo testiranje podrazumeva poznavanje softvera kao, a i njegove slabe tačke. Testovi se kreiraju na osnovu pređašnjeg iskustva i intuicije. Ova tehnika je komplementarna tehnika.

Test tehnike koje se baziraju na iskustvu su Error guessing, Checklist based, Exploratory testing i Fault Attack testiranje.

4. TEST NIVOI

Prilikom testiranja sistema, a naročito kompleksnih sistema, testiranje se deli na nivoe. U većini slučajeva koriste si tri do četiri test nivoa pri validaciji i to *Unit*, integracioni, sistemski i *Acceptance* testovi.

4.1 Unit test

Unit, odnosno modul, testiranje je white box tehnika testiranja koja se fokusira na testiranje manjih gradivnih blokova programa. Testiranje modula olakšava proces nalaženja defekata (debugging), jer se problem lokalizuje na nivo funkcije.

Kada se modul testiranju pristupi na adekvatan način, osigurava se potpuna pokrivenost koda, gde se sve putanje i sva grananja pokrivaju testovima. Svrha modul testiranja nije provera funkcionalnosti modula, već provera da je modul razvijem tačno kako je navedeno specifikacijom.

4.2 Integracioni test

Nakon modul testiranja, svi zasebni moduli se integrišu u grupu, odnosno sistem. Faza spajanja modula, naziva se integraciona faza, a da bi moduli mogli međusobno da komuniciraju neophodno je dodati interfejse, funkcije, koji će ih povezivati. Svrha integracionih testova je u testiranju kako se modul ponaša u celini, ali i defekte koji nastaju u interfejsima koji povezuju module.

4.3 Sistemski test

Nakon što su svi moduli integrisani u sistem i integracioni testovi izvršeni može se pristupiti sistemskom testiranju. Sistemski testovi su testovi koji se baziraju na *black box* tehnikama i bave se proverom ponašanja sistema. Ovi testovi vrše pored funkcionalne i proveru pouzdanosti, performansi i bezbednosti sistema.

4.4 *Acceptance* test

Poslednji tip testiranja u nizu, podrazumeva proveru jesu li svi zahtevi u okviru specifikacije implementirani i jesu li implementirani na adekvatan način.

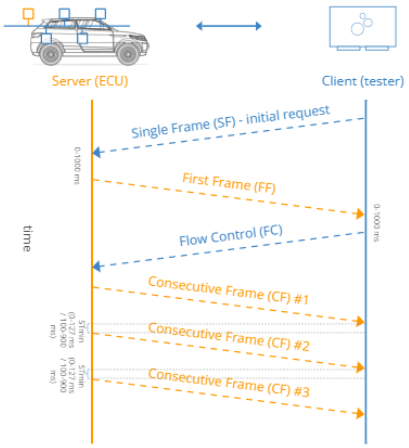
Cilj ovog testiranja jeste potvrda da implemetirano rešenje zadovoljava i funkcionalne i ne funkcionalne zahteve. Ovaj tip testiranja "zatvara" test fazu i daje potvrdu da je faza razvoja produkta završena.

5. CAN komunikacija

CAN, *Controller area network*, protkol razvijen od strane *Bosh* kompanije početkom 1980 ih, postao je međunarodni standard (*ISO 11898*) 1994-e godine. Razvijan za potrebe brze serijske komunikacije između kontrolnih modula, široku upotrebu je našao u automobilskoj industriji.

Komunikacionim linkom, koga čine dva signala CANH i CANL, prostire diferencijalni signal čija je brzina prostriranja definisana je standardima ISO 11898-2 i ISO 11898-3.

Standardna *CAN* poruka može biti dužine 8B (bajtova) dok dužina *CANFD* poruke može biti do 64B(bajta). Jedna *CAN* poruka, dužine osam bajtova, naziva se frejm (*frame*), u slučaju standardne *CAN* komunikacije vrši se razmena pojedinačnih (*single frame*) poruka, dok u slučaju *CANFD* komunikacije postoji i poruka kontrole prenosa (*flow control frame*) gde se šalje prva poruka, frame, a kao odgovor se dobija potvrda od kontrolera kome je poruka poslata i nastavlja se sa slanjem ostatka poruka (*Consecutive Frame*).



Slika 2. *Princpi razmene CAN poruke*

6.UDS

UDS, Unified Diagnostic Services, definisan standardom ISO 14229, je komunikacioni protokol koji koriste dijagnostički sistemi pri komunikaciji sa procesorskim jedinicama. U sklopu *UDS* protokola definisani su servisi. Lista servisa sa servisnim identifikatorima, koji su podržani ovim protokolom prikazana je na slici.

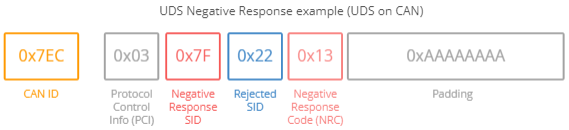
SID	Service	SID	Service
0x10	Diagnostic Session Control	0x31	Routine Control
0x11	ECU Reset	0x34	Request Download
0x14	Clear Diagnostic Information	0x35	Request Upload
0x19	Read DTS Information	0x36	Transfer Data
0x22	Read Data By Identifier	0x37	Request Transfer Exit
0x23	Read Memory By Address	0x38	Request File Transfer
0x24	Read Scaling Data By Identifier	0x3E	Tester Present
0x27	Security Access	0x3D	Write Memory By Address
0x28	Communication Control	0x83	Access Timing Parameters
0x29	Authentication	0x84	Secured Data Transmission
0x2A	Read Data By Periodic Identifier	0x85	Control DTC Setting
0x2C	Dynamically Define Data Identifier	0x86	Response on Event
0x2E	Write Data By Identifier	0x87	Link Control
0x2F	Input Output Control By Identifier		
		0x7F	Negative Response SID

Slika 3. *UDS podržani servisi*

Dijagnostička poruka se procesorskoj jedinici prosleđuje kao *CAN* poruka.

Slika 4. *UDS poruka sa pozitivnim odgovorom*

Negativan odgovor na zahtevani servis ima predefinisani servisni identifikator 0x7F, sledeći bajt označava za koji servis se šalje negativan odgovor i na kraju šta prouzrokuje takav odgovor.



Slika 5. *Negativan odgovor UDS poruke*

7. CANoe simulacija

Za potrebe testiranja kreirana je simulacija u CANoe softveru. Ovaj program, razvijen od strane Vector Informatik GmbH kompanije, primenjuje se u automobilskoj industriji u toku testiranja i analize rada razvijanog sistema. Simulacija se koristi pri komunikaciji sa kontrolnim modulima razmenjujući CAN poruke.

8. ZAKLJUČAK

U ovom radu detaljno je prikazan proces razvoja i testiranja softera, gde je akcenat stavljen na nivoe i metode testiranja svakog segmenta razvoja. Predstavljene metode se koriste u automotiv industriji i neizostavan su deo svakog test plana. Ukoliko se testiranju pristupi temeljno i u što ranijoj fazi, povećava se verovatnoća pronalaženja defekata, čime se smanjuju mogućnosti otkaza ili neadekvatnog ponašanja programa. Svaki program, koliko god bio jednostavan, mora proći kroz adekvatnu test fazu jer se samo na taj način može potvrditi da je korišćenje tog softvera bezbedno i da se nepredviđeno ponašanje neće dogoditi.

9. LITERATURA

[1] https://www.csselectronics.com/pages/uds-protocol-tutorial-unified-diagnostic-services, jun 2023.
[2] https://www.csselectronics.com/pages/can-bus-simple-intro-tutorial, jun 2023.
[3] https://www.csselectronics.com/pages/can-fd-flexible-data-rate-intro, jun 2023
[4] Bart Broekman, Edwin Notenboom, “Testing Embedded Software”, Addison-Wesley Longman, Amsterdam, 2003
[5] Greg Fournier, “Essential Software Testing, A Use-Case Approach”, New York, 2008

Kratka biografija:

Lazar Vučković rođen je u Kragujevcu 1995. god. Diplomski rad na Fakultetu tehničkih nauka iz oblasti elektrotehnike i računarstva – Autoelektronika odbranio je 2020. godine.

Vladimir Rajs rođen je 1982. godine u Apatinu. Diplomirao je 2007, a doktorirao 2015. godine na Fakultetu tehničkih nauka u Novom Sadu. Od 2016. godine zaposlen je kao docent, od 2021. kao vanredni profesor na Departmanu za elektroniku, energetiku i telekomunikacije FTN-a.