Propisi i standardi verifikacije softvera u automobilskoj industriji

Seminarski rad u okviru kursa
Metodologija stručnog i naučnog rada
Matematički fakultet

Andrea Pilipović, Vojkan Cvijović, Strahinja Stanojević, Saša Cvetković

5. maj 2018.

Sažetak

U današnje vreme sve je veća upotreba softvera u oblastima u kojima ranije nije bila. S obzirom da eventualne greške u softverima automobila mogu dovesti i do fatalnih posledica, izuzetno je važno verifikovati softvere u ovoj industriji. U ovom radu je opisano testiranje kao osnovni i najzastupljeniji vid verifikacije softvera u automobilskoj industriji. Opisane su osnovne ideje testiranja i verifikacije ,standardi koji postoje i dati su primeri grešaka u softveru kako bi se dodatno objasnila važnost verifikacije.

Sadržaj

1 Uvod **2**

2 Softver u automobilskoj industriji **2**

3 Značaj verifikacije u automobilskoj industriji **3**

4 Izazovi u verifikaciji softvera u automobilskoj industriji **4**

5 Efikasno testiranje u automobilskoj industriji **4**

6 Analiza i dizajn **6**
6.1 Specifikacija test slučajeva koji se testiraju metodom crne kutije 6

7 Implementacija i izvršavanje test slučajeva **7**
7.1 Efektivna selekcija test slučajeva i prioritizacija u testovima crne kutije 7
7.2 Kombinacija test slučajeva 8

8 Rezultat izvršavanja i izveštaj dobijenih rezultata **9**
8.1 Mašinsko učenje u regresionom testiranju 9
8.2 Automatska selekcija kriterijuma u regresionom testiranju . 10

9 Standardi u automobilskoj industriji **10**
9.1 ISO 26262 standard 11

10 Zaključak **13**

1 Uvod

U velikoj većini različitih industrija se koriste embeded sistemi kako bi se podigla pouzdanost različitih sistema. Obično su to sistemi koji zahtevaju visoku pouzdanost i bezbednost. Sistemi od kojih mogu zavisiti i ljudski životi. Jedna od takvih industrija je i automobilska industrija. Potreba za korišćenjem softvera u ovoj industriji kao i sama njihova upotreba je značajno porasla u poslednjoj deceniji [1] [2]. Većina funkcionalnosti u modernim automobilima, posebno one funkcije koje su vezane za bezbednost, kao što su automatske kočnice ili automatski asistenti pri vožnji [3], kontrolišu se pomoću softvera [4]. Predviđa se da će oko 90% automobila u budućnosti biti voženo softverski, bez ljudske pomoći [2].

2 Softver u automobilskoj industriji

Automobilska industrija je tradicionalno pretežno bazirana na konceptima mašinskog inženjerstva [5][6]. Baš kao i neke druge industrije koje su tradicionalno ne-softverske, kao što su avionska, odbrambena ili raketna, i automobilska industrija biva pogođena softverskom revolucijom. Sve veći broj, kako kritičnih tako i nekritičnih, funkcionalnosti automobila počinje da bude kontrolisano pomoću softvera. S toga je razumevanje softvera koji se koriste u automobilskoj industriji sve važnije, i mnogo se više pažnje pridaje nego pre par decenija. Broj (eng. Broy) u [6] opisuje softver u ovoj industriji na sledeći način:

veličina: Za 40 (od 1976. [7]) godina broj linija koda u automobilskoj industriji se povećao sa nula na nekoliko miliona.

uloga: Funkcije u vozilima, kako kritične kao što su kočioni sistem, tako i nekritične (radio, klima i slično) danas se kontrolišu pomoću softvera.

interakcija i rasprostranjenost: U automobilima postoje ECU sistemi [38], koji su u suštini mikrokontroleri koji odgovaraju softverskim komponentama. ECU sistemi rade zajedno kako bi izvršili potrebnu funkciju automobila. Te funkcije mogu biti veoma raznorodne, kao na primer kontrolisanje motora, vazdušnih jastuka ili prikaz nivoa goriva [8]. U prošlosti, svaki ECU sistem je izvršavao posebnu funkcionalnost. Zbog toga su softverski mogle da se izvršavaju samo funkcije koje je mogao da izvršava jedan ECU sistem. Ranih devedesetih, ECU sistemi su bili povezani lokalnom mrežom CAN (eng. Control Area Network) pomoću koje su mogli da komuniciraju i dele informacije. Ovo je dovelo do funkcija koje su bile kompleksnije i bilo je potrebno da ih zajedno izvršava više ECU sistema. U kasnim devedesetim, funkcije su bile toliko kompleksne da je bilo potrebno da postoji komunikacija između više CAN mreža. Danas, sem komunikacije među CAN mrežama unutar samog automobila, vrši se i njihova komunikacija sa spoljašnjim okruženjem pomoću radio veza [2].

Do sada navedene karakteristike su samo dokaz da je upotreba softvera jako velika u automobilskoj industriji danas. Takođe, pokazuje da je softver neizostavni deo današnjih automobila. Broj (Manfred Broj 1949 -) dalje u [7] opisuje karakteristike softvera u automobilskoj industriji. Naredne stavke govore o kompleksnosti samog softvera:

heterogenost podsistema: Pri izradi softvera za automobil koristi se modularnost, kao i uopšte pri proizvodnji automobila. Tako se rade softveri za kontrolisanje različitih sistema, kočnica, menjača, motora, a zatim se oni pojedinačno testiraju. Nakon uspešnog testiranja, sistemi se objedinjavaju u jedan jedinstven sistem vozila.

proizvođači originalne opreme (eng. Original Equipment Manufacturer (OEM)): Proces integracije heterogenih sistema je sam po sebi zahtevan, međutim ovaj zadatak je daleko kompleksniji. To je tako zbog činjenice da veliki broj sistema za vozila dolazi od različitih proizvođača. Kasnije je zadatak automobilskih kompanija da ove sisteme i softvere dobijene od različitih proizvođača objedine i integrišu.

visoko konfigurabilni softver: Softveri u automobilskoj industriji su visoko konfigurabilni, s obzirom na to da postoji veliki broj različitih funkcija koje automobil obavlja. S toga, veliki broj različitih varijanti vozila može biti proizveden od modularnih delova. Na primer, autori [7] navode primer vozila sa 80 različitih softverskih jedinica, gde jednostavan izbor da li se funkcionalnost svakog od njih uključuje ili ne vodi do 2^{80} mogućnosti. Iz ovoga se zaključuje da je moguće napraviti više varijanti vozila koje se razlikuju po funkcionalnostima koje su uključene i dobijene su kombinovanjem različitih softverskih jedinica.

3 Značaj verifikacije u automobilskoj industriji

Verifikacija softvera, i testiranje kao jedna od oblasti verifikacije je doživela ekspanziju poslednjih godina, posebno u osetljivim oblastima, kao što su avionska, raketna, automobilska i slične industrije. Ipak, što se same automobilske industrije tiče, ne postoji mnogo studija koje se fokusiraju na testiranje i verifikaciju na višim nivoima, pri integraciji pojedinačnih komponenti. Dakle, postoji potreba za fokusom i akademskih i industrijskih istraživača u ovoj oblasti.

S obzirom na sve veći broj funkcionalnosti koje pružaju današnji automobili, softveri bivaju sve kompleksniji. Jedan od glavnih zadataka u automobilskoj industriji je dizajnirati efektivne i efikasne procese izrade i testiranja softvera [9]. S obzirom na sve kompleksnije i veće softvere sa sve većim brojem mogućnosti i funkcionalnosti, njihova verifikacija i testiranje postaje dosta kompleksan zadatak [10]. Neotkrivene greške u softveru u ovoj industriji mogu imati ozbiljne posledice. Može doći do saobraćajnih nesreća, povreda, pa čak i do gubitka ljudskih života. Zbog ovoga je izuzetno važno izvršiti detaljno testiranje svih delova softvera. Samo jedna greška može dovesti do katastrofalnih posledica [11]. Upravo iz ovog razloga, oko 50% od ukupnog vremena koje se utroši na tehničke aktivnosti (vezane za izradu softverskog dela) u razvoju vozila se potroši na testiranje samog softvera [12].

4 Izazovi u verifikaciji softvera u automobilskoj industriji

U ovom delu dajemo kratak pregled izazova koji se mogu sresti u verifikaciji softvera u automobilskoj industriji [13][14][15]:

merenje napora: Poteškoće u određivanju važnosti testiranja na različitim nivoima [14].

znanje osoblja: Različita mišljenja ljudi različite stručnosti [14].

distribuirane funkcionalnosti: Visoka kompleksnost testova usled distribuiranosti softvera [14].

metrika pokrivenosti: Nedostatak podrške za merenje testova [14].

različite varijante: Kombinatorna eksplozija testiranja zbog visokog stepena prilagođavanja [14].

zahtevi i mogućnost praćenja: Problemi vezani za zahteve kao što su nedostatak jasnih zahteva za testove visokog nivoa i nedostatak praćenja napretka kao prepreka za verifikaciju [13].

proces testiranja: Odsustvo jedinstvenog procesa testiranja [13].

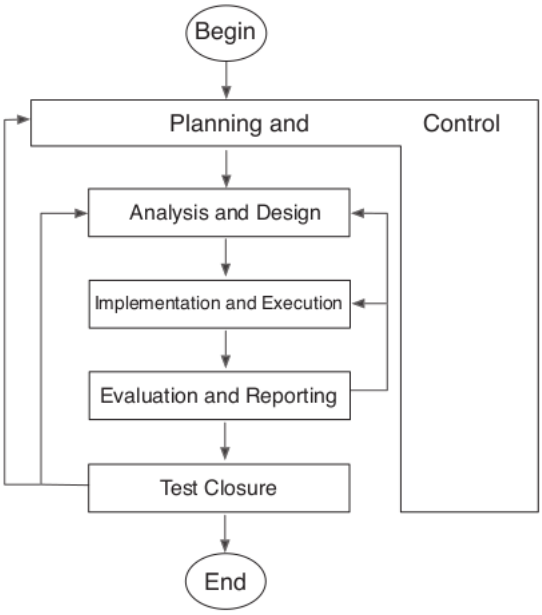
alati za verifikaciju: Nedostatak adekvatnih alata i tehnika za verifikaciju [13].

testiranje i praćenje nedostataka: Visoka cena popravljanja nedostataka kao i neotkrivenih nedostataka [13].

5 Efikasno testiranje u automobilskoj industriji

Softverski sistemi postaju kompleksniji i sve više se izvršavaju u okruženjima gde je bezbednost od izuzetne važnosti, na primer sistemi za pomoć pri upravljanju vozilom. Testiranje softvera je jedna od najbitnijih stavki prilikom razvoja softvera za automobile. Bilo kakve greške na softveru koje se ne otkriju prilikom testiranja, mogu da izazovu ogromne finansijske probleme i čak da rizikuju ljudske živote. Kao na primer kada je Dodge povukao 1.25 miliona vozila zbog greške u softveru koja je blokirala aktiviranje bočnih vazdušnih jastika u slučaju prevrtanja [42].

Proces testiranja softvera je podeljen u pet faza, kao što je prikazano na slici 1. Počinje se sa fazom planiranja u kojoj su definisani koncept testa i plan testa. Resursi moraju biti raspoređeni i ciljevi svakog testa moraju biti definisani. Posle početnog planiranja, sledi faza analize i dizajna. U toj fazi svi slučajevi se opisuju na apstraktan način i definišu se prema specifikaciji softvera. Nakon toga svi test slučajevi se moraju implementirati, dalje se vrši odabir testova, pokreću se na sistemu za testiranje, ručno ili automatski. Kada se izvršavanje testova završi, rezultati se sakupljaju i dokumentuju za dalju analizu [41]. Osoba odgovorna za projekat mora da odluči koji su dalji testovi neophodni. Na primer, posmatranjem učestalosti dešavanja kvarova i popravki na sistemu. Novi testovi se izvršavaju ako je to potrebno.



Slika 1: Faze testiranja softvera

Generalno testiranje je u velikoj meri ograničeno resursima, stoga je cilj da testiranje bude efikasnije i da u isto vreme bude i efektivnije. U automobilskoj industriji postoji posebna veza između proizvođača, dobavljača i testera. Pošto su softverske komponente obično razvijene od strane dobavljača i testirane lokalno, integraciju obavlja proizvođač. Usled ugovorenih ograničenja, izvorni kod nije dostupan za integraciono i sistemsko testiranje. Pa se iz tog razloga softver testira metodom crne kutije gde je testiranje zasnovano na specifikaciji sistema. Drugo, proizvodi testiranja su često naznačeni u prirodnom jeziku [20]. Ovo otežava pravljenje pretpostavki u vezi progresa testiranja i pokrivenosti sistema na kome se testira. Treće, u većini slučajeva, ne mogu svi naznačeni test slučajevi biti izvršeni zbog ograničenja resursa. Mora biti izabran podskup test slučajeva. Četvrto, u automobilskom softverskom razvoju, dosta potprocesa od celokupnog procesa testiranja se izvršavaju ručno. Automatizacija testiranja procesa je, uključujući izbor i prioritet test slučaja, željeni cilj, pošto može da smanji napor testiranja u poređenju sa ručnim izvršenjem [16].

Kako bi automobilsko softversko testiranje napravili više efikasnim, u ovom radu, predstavićemo šest različitih izazova testiranja i prema rešenjima u automobilskom razvoju softvera uzimajući u obzir određena ograničenja i izazove u ovom području. Ovaj pristup pokriva test slučajeve u prirodnom jeziku, automatizacija izbora i prioritizaciju test slučaja. Prikazaćemo zadati koncept pomoću primera Body Comfort System ili BCS [17].

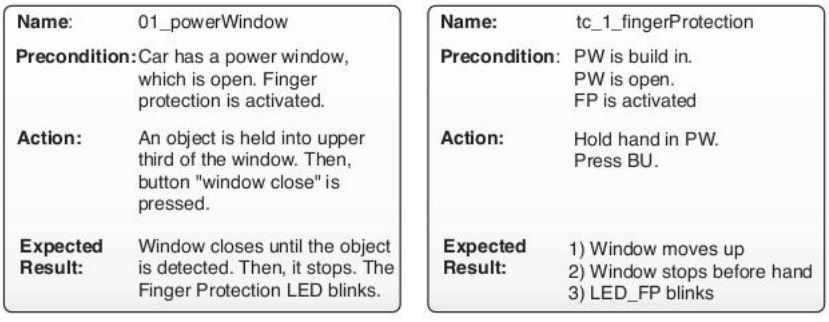
6 Analiza i dizajn

U ovoj fazi glavni zadatak je kreiranje slučajeva za testiranje. Greške iz ove faze mogu biti jako skupe u narednim fazama, na primer, nedovoljan opis testa, redudantan test slučaj[43]. U daljem teksu se opisuju dva načina na koja mogu da se unaprede testovi u ovoj fazi testiranja. Prvo se opisuje specifikacija za opis test slučajeva, dalje se predlaže koncept za uklanjanje redundantnih test slučajeva.

6.1 Specifikacija test slučajeva koji se testiraju metodom crne kutije

U fazi analize, test slučajevi su formulisani u nekom govornom jeziku. Svaki test slučaj se sastoji od preduslova, izvršavanja i očekivanog reziltata. Izvršavanje je podeljeno na veći broj koraka koji tester mora da izvrši. Test slučajevi su obično povezani određenim uslovnostima, što je obično zadato u specifikaciji tog test slučaja [43]. I preduslovi i koraci test slučaja se čuvaju u određenim alatima poput, HP Quality Center, IBM DOORS. Takvi alati omogućavaju praćenje test slučajeva, njihovih razlika. Na primer, test slučaj za zaštitu prstiju prilikom zatvaranja prozora, može biti opisan na dva različita načina kao što je pokazano na slici 2. Najčešći problem prilikom definisanja test slučaja je sam opis test slučaja.

Na primer, u prikazanim testovima na slici 2, u opisu jednog testa se nalaze pune rečenice dok kod drugog se nalaze samo stavke. Skraćenice mogu da izazovu probleme, moraju se definisati. U suprotnom test slučaj postaje nedovoljno jasan drugim saradnicima. Izazov je rukovati velikim skupom testova prilikom razvoja softvera. Kako novi projekti u kompaniji se često baziraju na prethodnim verzijama softvera i koriste proizvode koji su kreirani u drugim projektima, broj test slučajeva se uvećava vremenom[18]. Na primer, razvija se nova verzija sistema, za novi model. Prethodna verzija je imala veliki broj testova, od kojih većina može biti ponovo korišćena. Testovi se kopiraju u novi projekat i dizajniraju se novi testovi. Kako tester ne može da proveri opis i cilj velikog broja starih testova, moguće je da su pojedini novi testovi već pokriveni starim testovima.



Slika 2: Test slučaj otiska prsta

Redundantnost može da dovede do velike neefikasnosti testiranja kako se jedan isti test slučaj testira nekoliko puta bez promene rezultata [45]. Jedno rešenje jeste da se ne koriste gotovi proizvodi. To bi značilo da testeri moraju da napišu svaki test slučaj od početka, pa zbog velikog broja testova sledi da ovo nije efikasan način. Potreban je pametniji pristup, pristup koji može da detektuje i otkloni redundantnosti dok u isto vreme omogućava veliku iskorišćenost gotovih proizvoda. Testovi se porede kako bi se našli isti delovi. Ovo zahteva da su test slučajevi napisani na taj način da mogu da se porede. Nakon procesiranja specifikacije test slučaja, redudantni test slučajevi se obeležavaju i grupišu se u jedan novi test slučaja [46].

7 Implementacija i izvršavanje test slučajeva

Iako su test slučajevi sistematično opisani i bez redundantnosti, neophodno je pronaći balans između mogućeg velikog broja dizajniranih testova i veoma ograničenih resursa za testiranje. Dva glavna izazova jesu selekcija test slučajeva koji će biti izvršeni, i način na koji će odabrani test slučajevi biti sortirani. Prioritizacija test slučajeva [21][19] je neophodna, i retko se dešava da se svi odabrani test slučajevi izvrše, jer se neki testovi moraju ručno izvršiti. Zato odabir testova i prioritizacija predstavlja izazov jer je izvorni kod sistema koji se testira nepoznat. Takvo testiranje “metodom crne kutije", omogućava vrlo ograničene informacije [47].

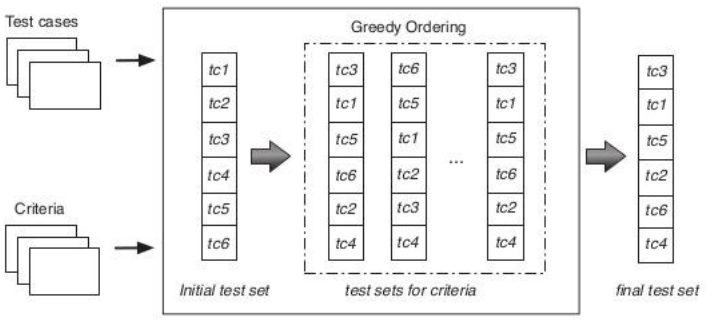
7.1 Efektivna selekcija test slučajeva i prioritizacija u testovima crne kutije

Kaner (Kaner) [1] [22] je istraživao i opisao način dobijanja kvalitetnih test slučajeva koristeći nekoliko faktora. U opštem testiranju, dobar test slučaj se može definisati pomoću nekoliko različitih kriterijuma. Na primer, kada test slučaj ima veliku verovatnoću da pronađe grešku, test slučaj testira funkcije koje se verovatno koriste u stvarnosti itd. Zbog toga za odabir test slučajeva je važno odabrati one test slučajeve koji odgovaraju trenutnom test cilju[22].

U slučaju testiranja crne kutije koriste se test slučajevi višeg nivoa, i njihovo kreiranje obično predstavlja ručni zadatak, koji zahteva veliku količinu stručnog znanja i iskustva u vezi sa sistemom koji se testira [34]. Prvi problem predstavlja činjenica da se poznavanje kreiranih test slučajeva i njihovog dizajna često gubi kada osoba napusti projekat ili kompaniju. Zadatak kreiranja novih test slučajeva je proces koji obično dugo traje, jer za posledicu ima potrebu za razvijanjem potrebne veštine za kreiranje “dobrih” test slučajeva. Drugi problem predstavlja činjenica da je ručna selekcija neefikasna u pogledu raspodele test slučajeva, vremena za njihov odabir i neuspeha. Procedura izbora test slučajeva nije formalno definisana, pa rezultati nisu predvidljivi, pa zbog toga zavise od lične procene osobe koja kreira test slučajeve. Podrška u vidu alata pomaže u kreiranju testova na mnogo stroži i sistematizovaniji način. Neiskusnim testerima je vodilja pri izboru i kreiranju probnih test slučajeva sa ciljem da prvo njih testiraju sa većom verovatnoćom pronalaženja neuspeha [34].

[1] Cem Kaner J.D., Ph.D., profesor softverskog inženjerstva na institutu u Floridi

Testiranje crne kutije zahteva definisan skup test slučajeva i skup kriterijuma za odabir/prioritizaciju [34]. Šematski opis ovog pristupa je prikazan na slici 3. Pristup koristi unete test slučajeve koji su sortirani na neki proizvoljan način, abecedno ili nakon datuma nastanka. Odabran je skup unapred definisanih kriterijuma, kao što je vreme merenja, prioritet, testirana funkcionalnost ili istorija test slučajeva. Jedan od mogućih kriterijuma je istorija testiranja, koji obuhvata broj neuspeha pronađenih testom i njihovim prioritetom. Takvi kriterijumi su korisni u scenariju regresionog testiranja, gde se test slučajevi moraju izvršiti nakon nekog ažuriranja softvera. Neuspešnu funkcionalnost bi trebalo ponovo testirati. Pohlepni algoritam izračunava različite uređene testne grupe za svaki od izabranih kriterijuma i u odnosu na izabrani kriterijum najbolji test slučaj se postavlja u prvi plan. Kako bi se ovaj korak učinio boljim za različite projektne domene i faze projekta, testovi se često kombinuju pomoću funkcije težine definisane ili od strane testera ili automatski postavljene između kriterijuma. Na primer, ako se izda novi softver, gde je važna istorija testa, ali su resursi za testiranje ograničeni, vreme provere testa i istorija testova se koriste kao kriterijumi. Istorija testa može biti 75% prioriteta, a vreme izvršenja sa 25% prioriteta da utiče na rezultat[34] Nije garantovan pronalazak optimalnog test slučaja, ali ovaj pristup se koristi kao brži način da se pronađe prilično dobar rezultat. Ključni faktor za uspeh ovog pristupa su definisani kriterijumi i njihovi prioriteti.



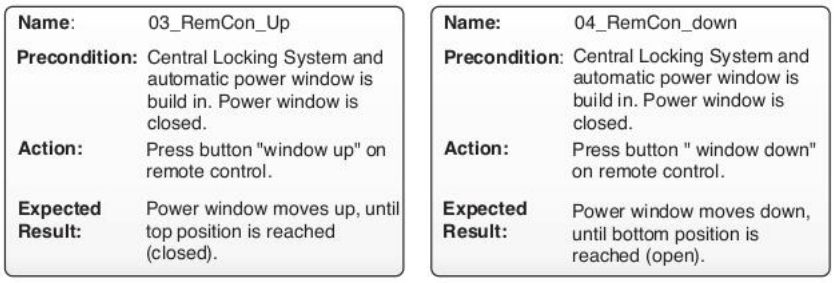
Slika 3: Pohlepna regresiona strategija selekcije

7.2 Kombinacija test slučajeva

Jedan od primera jeste ideja da se povećava broj izvršenih test slučajeva pod vremenskim ograničenjima. Slično poput pristupa opisanom u delu 8.2, prioritizacija test slučajeva može dovesti do povećane efikasnosti testiranja, dok smanjenje vremena izvršavanja testa može dovesti do poboljšanih rezultata u poređenju sa ručnim izborom testa [23]. Ovo se postiže njihovim uređenjem prema njihovim pre–uslovima i post-uslovima.

Na primer za studiju slučaja elektronskih podizača stakala postoje dva test slučaja za funkciju kontrole, koja omogućavaju korisniku da pomera

prozor pomoću dugmadi na daljinskom upravljačkom tasteru 4. Ova dva slučaja se mogu povezati logičkim redosledom, pri čemu se prvi prozor pomera prema gore, a potom se pomeri prema dole. Post-stanje prvog testnog slučaja je preduslov drugog. Iako je dat primer prilično trivijalan, pronalazak takvih struktura i koherencija u nekoliko hiljada test slučaja, koji su napisani na prirodnom jeziku, nije trivijalan. Jedna mogućnost je istraživanje pojmova i rečenica prirodnog jezika kako bi se pronašle moguće kombinacije test slučajeva u testne sekvence. Ovo zahteva jedinstvenu specifikaciju test slučajeva kako je opisano u delu [23].



Slika 4: Test slučajevi za kontrolu električnih prozora

8 Rezultat izvršavanja i izveštaj dobijenih rezultata

Rezultati testova se nakon izvršavanja ocenjuju i dobijaju se nove informacije pomoći kojih se vrši prilagođavanje procesa testiranja i test slučajeva. Ovo uključuje promenu odabira test slučaja, prioritizaciju testiranja, raspodelu resursa itd. U regresionom testiranju, Kaner (Kaner) [22] definiše ideju pisanja test slučaja koji bi mogao biti ponovo iskorišćen.

8.1 Mašinsko učenje u regresionom testiranju

Test slučajevi visokog nivoa se često označavaju metapodacima, npr. prioritet test slučajeva, testirana funkcionalnost, istorija izvršavanja testa. Metapodaci se mogu koristiti za odabir test slučaja i određivanje prioriteta, kako je opisano u delu . Na primer, u okviru studije slučaja može se tvrditi da test slučaj vezan za funkcionalnost alarmnog sistema ima viši prioritet od testnog slučaja koji testira odgovarajuću LED indikaciju za ovu funkciju[26]. Metapodatke definiše dizajner testova, i predstavlja skup proces u njegovom stvaranju, jer nije trivijalno odlučiti, na primer, koji je prioritet test slučaja. Ovaj problem se povećava sa svakom testnom iteracijom, jer pridružene metapodatke treba menjati tokom vremena, jer se sistem koji se testira menja (npr., popravke grešaka, nove funkcije itd.). Ovo dovodi do pitanja kako se metapodaci mogu menjati nakon ciklusa testiranja bez oslanjanja na ručnu intervenciju i stručno znanje[27].

Jedno od mogućih rešenje za izbegavanje ručne intervencije prilikom prilagođavanja metapodataka nakon ciklusa testiranja jeste primena tehnike mašinskog učenja[26][27]. Mašinsko učenje pristupa identifikovanju uzoraka u podacima koji se mogu koristiti za izradu novih znanja i mogu se primeniti za donošenje odluka. Mašinske tehnike mogu analizirati specifikaciju testa, zahteve, rezultate testa i dostupnost testiranja njegove istorije[27]. Na osnovu nalaza prilagođava metapodatke povezane sa test slučajevima za sledeću test iteraciju i proces selekcije. Na primer, ako imamo klasifikator koji odlučuje da li će neki probni slučaj u velikoj verovatno otkriti neuspeh ili ne, dodelićemo mu viši prioritet za sledeće izvršenje testiranja.

8.2 Automatska selekcija kriterijuma u regresionom testiranju

Selekcija test slučaja za regresiono testiranje je rekurzivni zadatak. Kao što je opisano u delu , može se uvesti koncept, pri čemu se specifični kriterijumi selekcije koriste za odabir uređene liste test slučajeva. Opisani metod i dalje zahteva ljudski ulaz, pošto metapodaci dodeljuju testove za odabir, kriterijume selekcije i testeri moraju obezbediti odgovarajuće težine.

Kao što je prikazano u prethodnom odeljku, tehnike mašinskog učenja se mogu koristiti za prilagođavanje metapodataka u kontekstu testiranja crne kutije.[26] Međutim, nakon svakog testiranja, tester bi i dalje morao da odluči kako treba da izmeni prethodno korišćene kriterijume i njihove težine. Zbog toga se predlaže proširenje pristupa dela kako bi ga učinili automatizovanijim. Umesto ručnog definisanja kriterijuma selekcije, dobijeni podaci iz prethodnih testova mogu se koristiti za prilagođavanje kriterijuma i težina za sledeći ciklus testiranja.

9 Standardi u automobilskoj industriji

Uvođenje novih tehnoloških inovacija u takozvane "pametne automobile", koje su namenjene podršci i savetovanju vozača, ima za cilj da olakšaju i obezbede veću sigurnost i udobnost u vožnji [36]. Istovremeno, ovakve inovacije nose sa sobom i mogućnost rizika od pogrešnog rada, zbog kontrolisanja i interakcije ugrađenih softvera sa važnim sistemima kao što su kočnice i upravljački sistem. Kompanija Fijat Krajsler (eng. Fiat Chrysler) u maju 2017. godine imala je grešku u softveru koja je dovela do tragičnih saobraćajnih nesreća. Greška je prouzrokovala to da bočni vazdušni jastuci kao i pojas u slučaju naglog kočenja ne rade. Sličan problem je imala i kompanija Dženeral Motors (eng. General Motors) koja je grešku otkrila tek nakon saobraćajne nesreće koja je ubila jednog i ranila tri čoveka. [48] [39]

Standardi funkcionalne bezbednosti (eng. Functional Safety Standard) postoje kako bi se izbegle nesreće ovog tipa. Pridržavanjem ovih standarda održava se siguran i pravilan rad vozila, a fokus je prvenstveno na rizicima koji poizilaze iz slučajnih hardverskih kvarova, sistemskih grešaka u dizajnu sistema, hardveru ili razvoju softvera, proizvodnji, pa sve do funkcionalne aplikacije. Dva standarda imaju važnu ulogu u razvoju softvera automobilske industrije, MISRA i ISO 26262, zajedno sa još jednim koji ih

prati: AUTOSAR. Većina programera u automobilskoj industriji prepoznaje ova imena, ali se postavlja pitanje koliko se oni sami pridržavaju ovih standarda i koliko su upoznati sa njima. Kako su ovi standardi obavezni kod većine automobilskih dobavljača, bitno je razumeti namenu koja stoji iza njih i to kako se analiza statičkog koda uklapa u testiranje krajnjeg proizvoda zarad ispunjavanja ovih standarda.

9.1 ISO 26262 standard

ISO (eng. International Organization for Standardization) je najveća svetska organizacija koja se bavi razvojem i izdavanjem internacionalnih standarda. ISO 26262 standard, poznat i pod nazivom Đrumska vozila - Funkcionalna bezbednost "(eng. Road vehicles - Functional safety) je rezultat potrebe za internacionalnim standardom specifičnim za automobile koji bi se fokusirao na bezbednosno-kritične komponente automobila. On je nastao kao adaptacija šire primenjivanog IEC 61508 standarda Međunarodne Elektrotehničke Komisije (eng. International Electrotechnical Commission - IEC), generičko funkcionalno-bezbednosni standard za električne i elektronske sisteme, koji postavlja uslove za osiguravanje da su sistemi dizajnirani, implementirani, funkcionalni i napravljeni tako da obezbeđuju potreban nivo sigurnosti integriteta (SIL, eng. Safety integrity level).

Ovaj standard ima za cilj da smanji rizik na prihvatljiv nivo, pokrivajući sigurnost sistema u kompletnom životnom veku. Počiva na konceptu rizika, verovatnoći opasnih događaja i ozbiljnosti njihovih posledica kao i sigurnosnim funkcijama koje takav rizik smanjuju. Tri stava na kojima se zasniva standard IEC 61508 su: nulti rizik nikada ne može biti postignut, sigurnost se mora uzimati u obzir od samog početka i rizici koji se ne mogu tolerisati moraju biti redukovani [36]. ISO 26262 je standard funkcionalne bezbednosti koji se primenjuje na proizvodnju jednog ili više automobilskih električnih i elektronskih sistema koji se ugrađuju u putnička vozila do 3500 kilograma. Primeri ovakvih sistema uključuju elektronske parkirne kočnice, napredni sistemi za pomoć vozačima (ADAS, eng. advanced driver assistance system), elektronski program stabilnosti, motorne kontrolne jedinice, prilagodljiva kontrola brzine, itd.

Serija standard ISO 26262 sadrži 10 delova, 43 poglavlja, 180 inženjerskih metoda, 600 zahteva koji su sadržani u 450 stranica sa oko 750 rečenica [36]. U tabeli 1 prikazana je struktura serije standarda ISO 26262 sa područjem primene svih delova od 1-10.

Kratak opis delova standarda [37] [40]:

- ISO 26262-1 predstavljen na samom vrhu šematskog prikaza specificira pojmove, definicije i skraćenice za aplikaciju koje se koriste u svim ostalim delovima standarda
- ISO 26262-2 specificira zahteve za upravljanje funkcionalnom sigurnošću za automobilske aplikacije, uključujući zahteve nezavisne od projekta a koji se tiču organizacija uključenih u projekat (sveobuhvatno upravljanje sigurnošću) i zahteve zavisne od projekta a koji se tiču aktivnosti upravljanja u sigurnosnom ciklusu (upravljanje tokom konceptualne faze i razvoja proizvoda, i nakon puštanja u rad za proizvodnju)

Tabela 1: Struktura serije standarda ISO 26262

<table>
<tr><td colspan="4">1. Rečnik</td></tr>
<tr><td colspan="4">2. Upravljanje funkcionalnom sigurnošću</td></tr>
<tr><td colspan="2">2.-5. Upravljanje sigurnošću -uopšteno</td><td>2.-6.Upravljanje sigurnošću u proizvodnji</td><td>2.-6.Upravljanje sigurnošću nakon izlaska iz proizvodnje</td></tr>
<tr><td rowspan="2">3. Konceptualna faza
3.5-3.8</td><td colspan="2">4. Razvoj proizvodnje: sistemski nivo
4.5-4.11</td><td rowspan="2">7. Proizvodnja i operacije
7.5-7.6</td></tr>
<tr><td>5. Razvoj proizvodnje: hardverski nivo
5.5-5.10</td><td>6. Razvoj proizvodnje: softverski nivo
6.5-6.11</td></tr>
<tr><td colspan="4">8. Procesi podrške
8.5-8.14</td></tr>
<tr><td colspan="4">9. ASIL-orijentisane i sigurnosno-orijentisane analize
9.5-9.8</td></tr>
<tr><td colspan="4">10. Uputstvo za ISO 26262</td></tr>
</table>

- ISO 26262-3 specificira zahteve za konceptualnu fazu automobilskih aplikacija, uključujući: definicija stavki (eng. item), iniciranje sigurnosnog ciklusa, analize opasnosti i procene rizika i koncept funkcionalne sigurnosti
- ISO 26262-4 specificira zahteve za razvoj proizvoda na sistemskom nivou, uključujući: uslove za pokretanje razvoja proizvoda, specifikaciju tehničkih sigurnosnih zahteva, tehnički koncept sigurnosti, dizajn sistema, integraciju stavki i testiranje, sigurnosnu proveru, procenu funkcionalne sigurnosti i puštanje proizvoda u rad
- ISO 26262-5 specificira zahteve za razvoj proizvoda na hardverskom nivou, uključujući: uslovi za pokretanje razvoja proizvoda na hardverskom nivou, specifikacija hardverskih sigurnosnih zahteva, dizajn hardvera, podaci hardverskog nivoa i procena kršenja bezbednosne granice usled slučajnih grešaka hardvera, njegova integracija i testiranje
- ISO 26262-6 specificira zahteve za razvoj proizvoda na softverskom nivou, uključujući: uslovi za pokretanje razvoja proizvoda na softverskom nivou, specifikacija softverskih sigurnosnih zahteva, dizajn softverske arhitekture, dizajn softverskih jedinica i njihovih implentacija, testiranje softverskih jedinica, softverska integracija i testiranje i provera softverskih sigurnosnih zahteva
- ISO 26262-7 specificira zahteve za proizvodnju, rad, usluge i razgradnju
- ISO 26262-8 specificira zahteve za prateće procese
- ISO 26262-9 specificira zahteve za ASIL (eng. Automotive Safety Integrity Level) orijentisanu i sigurnosno-orijentisanu analizu
- ISO 26262-10 predstavlja sveobuhvatni prikaz ISO 26262 standarda, pruža dodatna obrazloženja u cilju razumevanja drugih delova ISO 26262

10 Zaključak

U ovom radu dali smo pregled nekih izazova u testiranju softvera za automobile. Fokusirali smo se na scenarije testiranja crne kutije, pošto izvorni kod često nije dostupan u automobilskom domenu. Nakon opšteg procesa testiranja, predstavili smo pristupe za poboljšanje efikasnosti testiranja s obzirom na ograničenja razvoja softvera za automobile. Neki od predloženih koncepata su već ostvarivi, na primer, sistematski opis slučaja testa. Drugim konceptima treba dodatni budući napor da bi bili od praktične upotrebe, npr., pristupi u mašinskom učenju u testiranju crne kutije.

Literatura

[1] F. Saglietti, "Testing for dependable embedded software," in 36th EUROMICRO Conference on Software Engineering and Advanced Applications (SEAA). IEEE, 2010, pp. 409–416.

[2] K. Grimm, "Software technology in an automotive company: major challenges," in Proceedings of the 25th International Conference on Software Engineering. IEEE Computer Society, 2003, pp. 498–503.

[3] Umar Zakir Abdul, Hamid; et al. (2017). "Autonomous Emergency Braking System with Potential Field Risk Assessment for Frontal Collision Mitigation". 2017 IEEE Conference on Systems, Process and Control (ICSPC). Retrieved 14 March 2018.

[4] B. Katumba and E. Knauss, "Agile development in automotive software development: Challenges and opportunities," in Product-Focused Software Process Improvement. Springer, 2014, pp. 33–47.

[5] F. Fabbrini, M. Fusani, G. Lami, and E. Sivera, "Software engineering in the European automotive industry: Achievements and challenges," in 32nd Annual IEEE Computer Society International Conference on Computers, Software and Applications(COMPSAC), 2008, pp. 1039–1044.

[6] M. Broy, "Automotive software and systems engineering," in Proceedings of the 3rd ACM and IEEE International Conference on Formal Methods and Models for Co-Design(MEMOCODE), 2005, pp. 143–149.

[7] M. Broy, I. H. Kruger, A. Pretschner, and C. Salzmann, "Engineering automotive software," Proceedings of IEEE, vol. 95, no. 2, pp. 356–373, 2007.

[8] J. S. Her, S. W. Choi, J. S. Bae, S. D. Kim, and D. W. Cheun, "A component-based process for developing automotive ecu software," in Product-Focused Software Process Improvement. Springer, 2007, pp. 358– 373.

[9] F. Franco, M. Mauro, S. Stevan, A. B. Lugli, and W. Torres, "Modelbased functional safety for the embedded software of automobile power window system," in 11th IEEE/IAS International Conference on Industry Applications (INDUSCON), 2014, pp. 1–8.

[10] M. Conrad, "Verification and validation according to ISO 26262: A workflow to facilitate the development of high-integrity software"Proceedings to 6th European Congress on Embedded Real Time Software and Systems (ERTS2), 2012.

[11] S. S. Barhate, "Effective test strategy for testing automotive software"in International Conference on Industrial Instrumentation and Control (ICIC). IEEE, 2015, pp. 645–649.

[12] R. Awedikian and B. Yannou, "A practical model-based statistical approach for generating functional test cases: application in the automotive industry Software Testing, Verification and Reliability, vol. 24, no. 2, pp. 85–123, 2014.

[13] A. Kasoju, K. Petersen, and M. V. Mäntylä, “Analyzing an automotive testing process with evidence-based software engineering,” Information and Software Technology, vol. 55, no. 7, pp. 1237–1259, 2013.

[14] D. Sundmark, K. Petersen, and S. Larsson, “An exploratory case study of testing in an automotive electrical system release process"in 6th IEEE International Symposium on Industrial Embedded Systems (SIES). IEEE, 2011, pp. 166–175.

[15] A. M. Pérez and S. Kaiser, “Top-down reuse for multi-level testing,” in 17th IEEE International Conferenceand Workshopson Engineering of Computer Based Systems (ECBS), 2010, pp. 150–159.

[16] J. Kasurinen, O. Taipale, and K. Smolander. Software Test Automation in Practice: Empirical Observations. Advances in Software Engineering, 2010: 571-579, 2010.

[17] S. Lity, R. Lachmann, M. Lochau, and I. Schaefer. Delta-oriented Software Product Line Test Models - The Body Comfort System Case Study. Technical report, TU Braunschweig, 2013.

[18] M. Lochau, S. Lity, R Lachmann, I. Schaefer, and U. Goltz. Delta-oriented model based integration testing of large-scale systems. The Journal of Systems and Software, 91: 63-84, 2014.

[19] L. Zhang, D.Han, L.Zhang, G.Rothermel, and H. Mei. Bridging the Gap between the Total and Additional Test-Case Prioritization Strategies. In International Conferenceon Software Engineering, ICSE 2013, 2013

[20] R. Lachmann and I. Schaefer. Herausforderungen beim Testen von Fahrerassistenzsystemen. In 11. Workshop Automotive Software Engineering (ASE), 2013.

[21] G. Rothermel, R. H. Untch, C. Chu, and M. J. Harrold. Prioritizing Test Cases For Regression Testing. IEEE Transactions on software engineering, Vol.27 No.10:929-948,2001.

[22] C. Kaner. What is a good test case? In Software Testing Analysis and Review Conference (STAR) East, 2003.

[23] M.Utting and B.Legeard. Practical Model-based Testing. Morgan Kaufmann, 2007.

[24] M. A. Sindbu and 1c. Meinkc. ms: An lncnemntal Lemming Algorithm for Finite Automata. CoRR, abs/1206.2691, 2012.

[25] H. Raffelt, B. Steffen, and T. Margaria. Dynamic Testing via Automata Learning. In Proceedings of the 3rd International Haifa Verification Conference on Hardware and Software: Verification and Testing, HVC’07, pages 136-152. Springer-Verlag, 2008

[26] L.C. Briand. Novel Applications of Machine Learning and Software Testing. In Quality Software, 2008 QSIC ’08. The Eighth International Conference on, pages 3-10, Aug 2008.

[27] A R. Lenz, A. P020, and S. R. Vergilio. Linking software testing results with a machine learning approach. Engineering Applications of Artificial Intelligence, 266-6): 1631-1640, 2013.

[28] F. A. Barros, L. Neves, E. Hori, and D. Torres. The ucsCNL: A Controlled Natural Language for Use Case Specifications. In SEKE, pages 250-253. Knowledge Systems Institute Graduate School, 2011.

[29] J. Ferrer, P. M. Kruse, F. Chicano, and E. Alba. Evolutionary Algorithm for Prioritized Pairwise Test Data Generation. In Proceedings of the 14th Annual Conference on Genetic and Evolutionary Computation, GECCO ’12, pages 1213-1220. ACM, 2012.

[30] C.Strobbe, S.Herramhof, E.Vlachogiannis, and C.A. Velasco. Test Case Description Language (TCDL): Test Case Metadata for Conformance Evaluation. In ICCHP, pages 164-171, 2006.

[31] E. Engstrom and P. Runeson. Software product line testing - A systematic mapping study. Information and Software Technology, 53:2-13, 2011.

[32] A. Cmyrev and R. ReiBig. Optimierte Varianten- und Anforderungsabdeckung im Test. In Automotive Software Engineering Workshop. 43. GI Jahrestagung, 2013.

[33] M.F. Johansen, O.Haugen, and F. Fleurey. An algorithm for generating t-wise covering arrays from large feature models. In SPLC, pages 46-55, 2012.

[34] L.C. Briand, Y. Labiche, and Z. Bawar. Using the Machine Learning to Refine Black-Box Test Specifications and Test Suites. In Quality Software, 2008. QSIC'08. The Eighth International Conference, pages 135-144, Aug 2008.

[35] B. Engstrom, P. Runeson, and M. Skoglund. A systematic review on regression test selection techniques. Information and Software Technology, 52: 14-30, 2010.

[36] Sabira S. Salihović, Suada F. Dacić, Azra A. Ferizović, Funkcionalna sigurnost cestovnih vozila prema seriji standarda ISO 26262, 2015.

[37] International Organization for Standardization (ISO), ISO 26262-Part 1 - Part 9: Road vehicles - Functional safety, 2011.

[38] Distributed Development of Automotive Real-time Systems based on Function-triggered Timing Constraints, Oliver Scheickl, Christoph Ainhauser, Michael Rudorfer BMW Car IT, Munich, Germany

[39] David Shepardson, GM recalls 4.3 million vehicles over air bag-related defect, https://www.reuters.com/article/us-gm-recall-idUSKCN11F2AH

[40] International Organization for Standardization (ISO), ISO 26262-Part 10: Road vehicles - Functional safety, 2012

[41] Andreas Spillner, Tilo Linz, Hans Schaefer, Software Testing Foundations, pp 17-19.

[42] https://www.nbcnews.com/business/autos/1-25-million-dodge-ram-pickups-recalled-over-fatal-software-n759476

[43] Andreas Spillner, Tilo Linz, Hans Schaefer, Software Testing Foundations, pp 23.

[44] Kyungsoo Im, Tacksoo Im, John D. McGregor, Automating Test Case Definition Using a Domain Specific Language, pp 2.

[45] Nagendra Pratap Singh, Rishi Mishra, Rajit Ram Yadav, Analytical Review of Test Redundancy Detection Techniques, pp 1.

[46] Gordon Fraser and Franz Wotawa, Redundancy Based Test-Suite Reduction, Institute for Software Technology, Graz University of Technology pp 6.

[47] Andreas Spillner, Tilo Linz, Hans Schaefer, Software Testing Foundations, pp 110.

[48] David Shepardson, Nick Carey, Fiat Chrysler recalls 1.25 million trucks over software error, https://www.reuters.com/article/us-fiatchrysler-recall/fiat-chrysler-recalls-1-25-million-trucks-over-software-error-idUSKBN1881I6