Zbornik radova Fakulteta tehničkih nauka, Novi Sad

UDK: 621.39
DOI: https://doi.org/10.24867/05BE18Milisic

METODE ISPITIVANJA I GARANCIJE KVALITETA U OBLASTI SOFTVERSKIH ALATA

TESTING METHODS AND QUALITY ASSURANCE IN THE FIELD OF SOFTWARE TOOLS

Maja Milišić, Željen Trpovski, *Fakultet tehničkih nauka, Novi Sad*

Oblast – ELEKTROTEHNIKA I RAČUNARSTVO

Kratak sadržaj – *U radu je prikazan jedan način automatskog ispitivanja SCS (eng. SoundClear Studio) razvojnog okruženja zasnovanog na regresivnom izvršavanju ispitnih slučajeva na sistemskom novou. Akcenat je stavljen na ispitivanje dve glavne komponente aplikacije: serverski deo, koji radi kao alat iz komandne linije zasnovan na gRPC okviru, napisan u programskom jeziku C# i GUI klijentu, implementiranom u programskom jeziku Java.*

Ključne reči: *ispitivanje, ispitni slučajevi, SCS, IDE, DUT, API*

Abstract – *The paper presents one method of automated testing of the SCS (eng. SoundClear Studio) development environment based on regressive execution of test cases on the system level. The emphasis is placed on testing the two main components of the application: the server part, which works as a command line tool based on the gRPC framework, written in the programming language C # and the GUI client, implemented in the Java programming language.*

Keywords: *testing, test cases, SCS, IDE, DUT, API*

1. UVOD

Integrisana razvojna okruženja (eng. Integrated Development Environment - IDE) najčešće predstavljaju jednu aplikaciju u koju su ugrađeni: razvojni alati, grafička korisnička sprega, komunikacioni protokoli i sve što je potrebno za rad sa savremenim realnim sistemima. Primer ovakve aplikacije je SCS (eng. SoundClear Studio). Ova aplikacija pruža širok opseg mogućnosti: za pisanje programa, njegovo spuštanje na sisteme, za kontrolisano izvršavanje programa, otklanjanje uočenih grešaka i uopšte, za razvoj onoga što će namenski sistemi raditi.

Karakteristika ovog okruženja je da je za njegovo razvijanje izabran agilan pristup – strategija koja označava paralelan rad tima za razvoj i tima za ispitivanje. Cilj pomenutog pristupa je da se razvojno okruženje kao proizvod implementira i isporučuje kroz više nezavisnih međuverzija, gde će svaka od njih posedovati određen skup unapred definisanih funkcionalnosti, ali i zahteva koje su klijenti u međuvremenu definisali. Shodno tome, ispitivanje i povratne informacije moraju se takođe obavljati kroz više nezavisnih iteracija, inkrementalno, onako kako razvoj okruženja napreduje.

Rad je nastao kao rezultat ispitivanja SCS razvojnog okruženja, softverskog alata koji se razvijao u Naučno-istraživačkom institutu "RT-RK" [1] i kompaniji Cirrus Logic [2]. To je novo razvojno okruženje koje predstavlja veći skup pojedinačnih alata namenjenih svim Cirrus-ovim klijentima. Ispitivanje koje treba obaviti, na v1.6.1.0 verziji SCS aplikacije, biće zasnovano na primeni različitih tipova ispitnih slučajeva, jer na taj način garancija da je sistem pouzdan je sve veća. Ispitivanje se u najvećem broju slučajeva sprovodi na udaljenim platformama [3].

Pored toga, potrebno je rešiti i problem ispitivanja razvojnog okruženja na različitim operativnim sistemima. Cilj je da se proces ispitivanja u potpunosti automatizuje, da bude što pouzdaniji, prilagodljiv aktuelnim operativnim sistemima, ali i nezavisan u odnosu na prekid komunikacije između platforme i ispitivača. Automatizacijom procesa ispitivanja se postiže značajna ušteda vremena i resursa. Automatski ispitni slučajevi se izvršavaju brže, zahtevaju manje resursa i manje obučenih ljudi. Ispitno okruženje i ispitni slučajevi su realizovani u Python programskom jeziku, skriptnom jeziku visokog nivoa u kom je čitanje i pisanje koda veoma jednostavno i lako za učenje.

2. ANALIZA RAZVOJNOG OKRUŽENJA

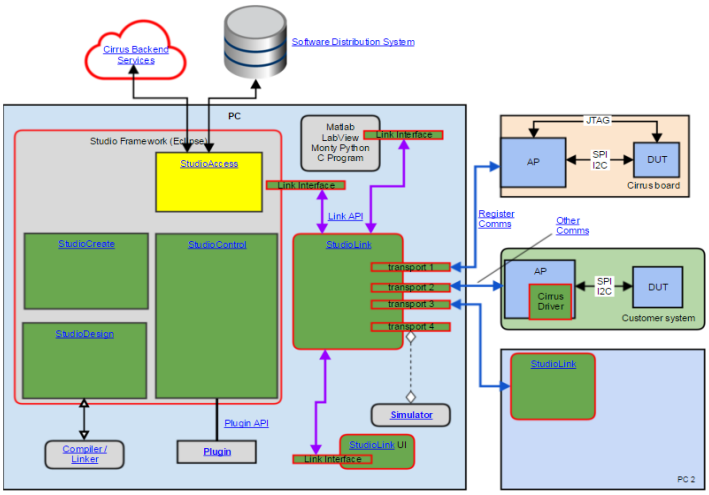
SCS čini centralno mesto razvojnog okruženja predstavljajući uređaj nad kojim se sprovodi automatizovan proces testiranja, (eng. Device Under Test - DUT). Kontrola između SCS aplikacije i sistema ostvaruje se pomoću Studio link-a, podsistema zaduženog za komunikaciju. Arhitektura okruženja prikazana je na slici 1. Kao alat za evaluaciju, programiranje i korisničko konfigurisanje razvojnih ploča, pruža sledeće mogućnosti:

- Instalaciju na različite operativne sisteme
- Komunikaciju sa svim starim i novim DSP procesorima kompanije Cirrus Logic
- Povezivanje računara i ploča sa procesorima preko različitih fizičkih medija
- Različite korisničke sprege: Python, Matlab, LabView
- Pisanje programa, njihovo povezivanje i spuštanje na DSP
- Kontrolisano izvršavanje programa

Jedan od mnogobrojnih podržanih čipova koji je moguće konfigurisati pomoću SCS-a jeste CS46L41, poznatiji pod imenom Gray.

Gray jeste audio kodek male snage sa USB interfejsom. Razvila ga je američka kompanija Cirrus Logic koja je prepoznatljiva po tome što, pored samog čipa, klijentima nudi celokupno hardversko i softversko rešenje.

NAPOMENA:
Ovaj rad proistekao je iz master rada čiji mentor je bio dr Željen Trpovski.



Slika 1. *Arhitektura razvojnog okruženja*

To znači da je kodek integrisan u okruženje i čini jednu od karika u lancu blokova obrade. S toga se javlja potreba za postojanjem alata koji će konfiguraciju čitave platforme učiniti jednostavnom i pristupačnom. Ima najveću primenu u audio adapterima (BAA) i slušalicama (BHS).

Ispitivanjem se moraju prevazići problemi geografske udaljenosti (ispitne mašine se nalaze na različitim kontinentima) i različitih operartivnih sistema (Windows 7, Windows 10, Mac i Linux). Samim ti, SCS aplikacija instalirana na ispitnoj mašini mora biti lako dostupna, a sa druge strane ispitno okrzuženje mora biti kompatibilno i ne sme zahtevati dodatnu programsku podršku za rad.

Ispitivanje SCS razvojnog okruženja se sastoji iz nekoliko zahteva i to:

- Ispitivanje instalacije
- Ispitivanje klijentskog dela okruženja (Studio IDE)
- Ispitivanje serverskog dela okruženja (Studio link)

Ispitivanje instalacije se odnosi na proveru da li je SCS razvojno okruženje ispravno instalirano na mašini gde se obavlja ispitivanje. Nakon instalacije SCS-a, pri inicijalnom pokretanju aplikacije potrebno je instalirati BSP pakete, koji omogućavaju aplikaciji da posredstvom linka komunicira sa željenim Cirrus Logic uređajima, bilo da je u pitanju realan ili virtuelni sistem. Odgovarajući BSP se isporučuju zajedno sa aplikacijom.

2.1. Studio IDE

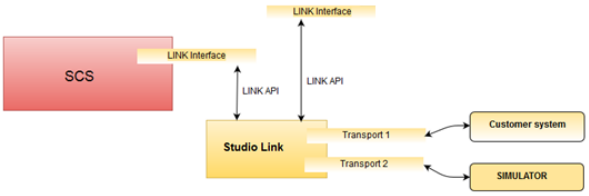
Ispitivanje klijentskog dela počinje od komponente navigator. Ova komponenta otkriva koji su sistemi prisutni na mreži, u našem slučaju ploče sa podržanim čipovima. Nakon toga se proverava da li je moguće povezivanje i njihova kontrola upisom i čitanjem vrednosti iz registara.

Ispitivanje se nastavlja pregledom istorijie kontrolnih akcija, gde je u listi akcija snimljena svaka prethodna akcija korisnika, vreme kada se desila kontrola nad registrima, itd. Pored toga, ispituju se i razni meniji, tulbarovi i svi ostali delovi dostupni klijentu. Pomenute komponente i funkcionalnosti ispitane su u dve dimenzije i to pri radu sa simuliranim i realnim sistemima tj. pločama.

2.2. Studio link

Studio link je komponenta razvojnog okruženja koja se isporučuje zajedno sa evaluacionim pločama i pruža jedinstveni interaktivni interfejs koji omogućava kontrolu i pregled podešavanja uređaja. Ispitivanje serverskog dela, Studio linka, treba da proveri funkcionalnosti koje se odnose na: ispravno otkrivanje ploča na mreži, komunikaciju i kontrolu ploča, čuvanje istorije poslatih komandi i njihovih povratnih vrednosti, vreme opsluživanja klijentskih zahteva i slično.

Višestruke klijentske aplikacije mogu se povezati i koristiti dostupan hardver koji je istovremeno povezan na mrežu. Studio link API omogućava pisanje programa koji komuniciraju sa hardverom preko Studio link-a. Svi delovi SCS-a komuniciraju sa sistemima na ovaj način. API je realizovan preko Studio link interfejsa, tako što koristi Studio link protokol za slanje komandi aplikaciji. Postoji podrška za SCS Client API u više programskih jezika, što daje mogučnost komunikacije SCS-a sa komponentama koje su realizovane u različitim programskim jezicima. Trenutno SCS podržava Java i Python API. Sa druge strane link komunicira sa sistemom (hardverom) preko Studio link transporta, kojih takođe ima više u zavisnosti od vrste sistema sa kojim se komunicira. Obostrana komunikacija je prikazana na slici 2.



Slika 2. *Prikaz komunikacije aplikacije i sistema*

3. REALIZACIJA ISPITNOG OKRUŽENJA

Ispitivanje SCS razvojnog okruženja realizovano je na sistemskom nivou, što znači da se razvojno okruženje posmatra kao integrisan sistem. Potrebno je sprovesti ispitivanje na različitim platformama, pa ispitno okruženje mora biti realizovano tako da se identično ponaša na svakoj od njih.

To je razlog za implementaciju okruženja na bazi klijent-server arhitekture. Upotreba skriptnog programskog jezika Python pokazala se kao odličan izbor pri realizaciji iz više razloga. Njegove osnovne prednost su da se lako i brzo uči, dostupan je bez posebnih komercijalnih dozvola, pruža podršku za objektnu metodologiju i što je najvažnije nezavisan je od platforme na kojoj se prevedeni program izvršava.

Ispitno okruženje sadrži više različitih modula napisanih u Python-u. Postoje moduli koji služe za zapis i čuvanje poziva svih funkcija, svih akcija i događaja u tekstualne datoteke (tako da se posle izvršavanja ispitnog slučaja one mogu pregledati), moduli koji razrešavaju o kojoj ispitnoj mašini se radi i svim putanjama na njoj, modul koji pruža podršku za pokretanje Squish alata, modul u kome je realizovana podrška za automatizaciju koraka u ispitnim slučajevima, kao i modul koji obezbeđuje i održava komunikaciju između ispitne i klijent mašine.

Glavne softverske komponente okruženja su:

- Client-server
- Squish
- Test case i test suite
- Machine specific

Client-server paket se sastoji od server i klijent API-ja. Server se pokreće na ispitnoj mašini na kojoj se izvršavaju automatizovani ispitni slučaj. Može biti mašina sa bilo kojom podržanom platformom (Windows 7, Windows 10 ili Mac). Okruženje je implementirano tako da je prvo potrebno pokrenuti server API, tada je server spreman i čeka zahteve od klijent host-a. Klijent strana je strana na kojoj se pokreće RT-Executor i to mora biti isključivo Windows mašina. Klijent API pokreće Intent+ skripta koju poziva RT-Executor prilikom inicilalizacije ispitnih slučajeva.

Squish paket čini Squish API koji omogućava pokretanje i izvršavanje automatizovanih ispitnih sktipti implementiranih u Squish-u. Glavne komponente API-ja su Squish server, runner i metoda za razrešenje statusa ispitnog slučaja. Izvršavanje ispitnog slučaja započinje pozivom metode *runSquishTestCase(squishTest)* koja prvo pokreće server, izvršava ispitni slučaj, postavlja status i zaustavlja server. Poziva se u main funkciji ispitne skripte na server ispitnoj mašini.

Test case i test suite paketi služe za sumiranje rezultata izvršenih ispitnih slučajeva. Sadrže metode za ispisivanje i generisanje log poruka i izveštaja na predefinisanim putanjama. Pored logova ispitnih slučajeva generišu se i log poruke čitavog okruženja.

Metode implementirane u ovom paketu koriste osobine takođe implementirane *Logger* klase. Ovaj paket je značajan jer pored ispitnih slučajeva implementiranih u Squish-u moguće je izvršavati i čiste Python ispitne skripte.

Machine specific paket sadrži xml u kom su pobrojane sve ispitne mašine na kojima se izvršavaju ispitni slučajevi. Ova skripta se koristi za uspostavljanje komunikacije i putanja između klijenta i servera na automatizovan način.

Na klijent strani, u sklopu RT-Executor-a u *user_spec_config.py* skripti se navodi putanja do klijenta i ime ili IP adresa ispitne mašine na kojoj će se vršiti ispitivanje.

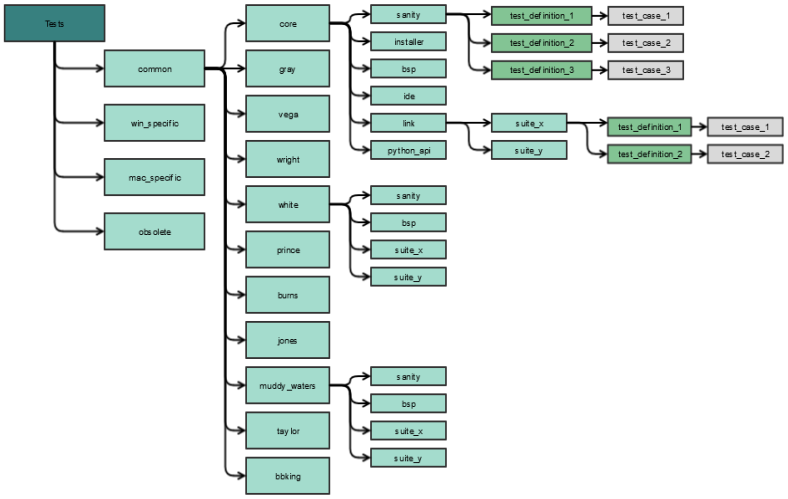
Na server strani, machine specific metode vrše parsiranje xml-a i na osnovu poslatog imena ispitne mašine znaju se svi ostali potrebni podaci o računaru. Xml polje svakog računara sadrži predefinisan skup putanja koje moraju postojati na svakom računaru na kom se izvršavaju ispitni slučajevi.

3.1. Tipovi ispitnih slučajeva

Ispitivi slučajevi se mogu podeliti u tri veće celine: ispitivanje instalacije, ispitivanje serverskog dela (Studio link) i ispitivanje klijentskog dela (Studio IDE).

Najveću grupu ispitnih slučajeva čini ispitivanje IDE-a tj. korisničkog interfejsa. Proverava se ispravnost na korisničkom nivou, kucanjem po tastaturi ili klikom miša se ispituje da li se dobijaju očekivani rezultati na akcije koje su preduzete.

Ispitivanje SCS alata se najvećim delom zasniva na GUI ispitivanju. Podela i struktura realizovanih ispitnih slučajeva prikazana je na slici 3.



Slika 3. *Struktura ispitnih slučajeva*

Gray, Vega, White testovi – su grupe ispitnih slučajeva za određene SCS projekte. SCS podržava više razvojnih ploča, i samim tim poseduje različite funkcionalnosti u zavisnosti od želja i potreba klijenta. *Reed* ploča se koristi kao podrazumevana za ispitivanje core funkcionalnosti. Na ovoj ploči se ispituju svojstva alata koja su zajednička za sve razvojne ploče.

U situacijama kada klijent zahteva ispitivanje aplikacije za potrebe Gray projekta tada se izvršavaju core + gray testovi (koji su karakteristični za funkcionalnosti tog projekta).

3.2. Implementacija ispitnih slučajeva na sistemskom novou

Automatske ispitne procedure u ovom radu implementirane su u Python programskom jeziku. Python ispitne skripte sadrže niz komandi/koraka preko kojih se, uz pomoć API biblioteka, šalju poruke uređajima koji čine sklop ispitnog okruženja.
Svaka ispitna skripta nosi naziv *test.py* i sadrži sledeće elemente:

1. Zaglavlje, u kojem se nalaze informacije o ispitnom slučaju,
2. Uključivanje biblioteka i konstanti koje su korišćene u ispitnoj proceduri,
3. Glavnu funkciju, *main(),* u kojoj je sadržana logika čitavog ispitnog slučaja,
4. *executeSteps()* metoda koja omogućava izvršavanje ispitnog slučaja.

4. PRIKAZ REZULTATA ZA VERZIJU v1.6.1.0

U okvuru SCS projekta najviše je zastupljeno regresivno ispitivanje. Svaki put kada se pojavi nova verzije SCS alata potrebno je izvršiti regresivno ispitivanje u predviđenom vremenskom roku. Na taj način se proveravaju funkcionalnosti novih komponenti i proverava se da nije došlo do promena u načinu rada nekih funkcija koje nisu bile obuhvaćene izmenama.

Upravo se ovde ogleda značaj automatizacije procesa ispitivanja jer je u tom slučaju ušteda vremena i resursa znatno veća. Regresivno ispitivanje je najčešće praćeno istraživačkim ispitivanjem, koje se oslanja na iskustvo i sposobnost pojedinca da pronađe nepravilnost u radu sistema. Defekti koji se tom prilikom otkriju, evidentiraju se i prijavljuju nadležnom timu kako bi što pre bili otklonjeni.

Ispitivanja počine izvršavanjem sanity grupe ispitnih slučajeva na sva tri operativna sistema. Ukoliko je sanity ispitivanje uspešno izvršeno, prelazi se na regresivno ispitivanje svih ostalih komponenti sistema.

U zavisnosti od zahteva klijenta regresivno ispitivanje se sprovodi ili u celosti ili samo nad zavisnim delovima sistema na koje promene mogu uticati. Isto tako ispitivanje se može sprovesti samo na nekoj od podržanih platformi.

Nakon analize zahteva za verziju v1.6.1.0, kreiranja ispitnih slučajeva, njihove automatizacije i izvršavanja na Windows 10 i Mac platformi, dobijeni su sledeći rezultati:

Tabela 6.1. *Sanity ispitni izveštaj sa statistikom*

Test plan	failed	passed	pending	Not tested	Not applicable	delayed	Total
SCS_sanity_win10	0	80	0	0	0	0	80
SCS_sanity_win7	0	80	0	0	0	0	80
SCS_sanity_mac	0	80	0	0	0	0	80

Tabela 6.2. *Regresivni ispitni slučajevi sa statistikom*

Test plan	failed	passed	pending	Not tested	Not applicable	delayed	Total
SCS_v1.6.1.0_win10	6	400	0	0	0	0	406
SCS_v1.6.1.0_mac	5	452	0	0	0	0	457

5. ZAKLJUČAK

U radu je predstavljena uspešna realizacija automatskog ispitivanja SCS razvojnog okruženja. Ostvarena je nezavisnost ispitnog okruženja u odnosu na različite platforme na kojima se vrši ispitivanje. Ispitno okruženje je jednostavno za upotrebu i daje mogućnost odlične podrške za nastavak ispitivanja svih novih verzija SCS alata.

6. LITERATURA

[1] RT-RK, "RT-RT Institute for Computer Based Systems", 2019, [Online]. Available: http://www.rt-rk.com/. [Accessed: 27.05.2019.]

[2] Cirrus Logic, "Cirrus Logic, Inc.", 2019, [Online]. Available: https://www.cirrus.com/. [Accessed: 27.05.2019.]

[3] M. Tankosić, U. Mićić, Đ. Miljković, I. Kaštelan, " Automatizacija ispitivanja integrisanog okruženja za DSP na principu crne kutije ", Novi Sad, 2016.

Kratka biografija:

Maja Milišić rođena je u Šapcu 1992. god. Master rad na Fakultetu tehničkih nauka iz oblasti Elektrotehnike i računarstva – Telekomunikacije i obrada signala odbranila je 2019.god.
kontakt: milisic.maja@yahoo.com

Željen Trpovski rođen je u Rijeci 1957. godine. Doktorirao je na Fakultetu tehničkih nauka 1998. god. Oblast interesovanja su telekomunikacije i obrada signala.