UDC: 004.51 INFO M: str. 40-45

KOMPARATIVNA ANALIZA ALATA ZA AUTOMATSKO TESTIRANJE GUI-A
COMPARATIVE ANALYSIS OF TOOLS FOR GUI AUTOMATED TESTING

Aleksandra Džudović, Saša D. Lazarević

REZIME: GUI (Graphical User Interface) testiranje predstavlja testiranje grafičkog korisničkog interfejsa. Sprovodi se kako bi se ispitalo da li korisnički interfejs (okruženje) funkcioniše u skladu sa predviđenim zahtevima. U ovom radu prikazana je komparativna analiza dva alata za automatsko testiranje korisničkog interfejsa – QF-Test i Jacareto. QF-Test je softverski alat za automatsko testiranje grafičkog korisničkog interfejsa za desktop aplikacije pisane u Java programskom jeziku i web aplikacije. Jacareto je softver otvorenog kôda (open-source) alat za automatsko GUI testiranje aplikacija pisanih u Java programskom jeziku i sastoji se od dve aplikacije: CleverPHL i Picorder. Za potrebe testiranja korišćena je aplikacija "Biblioteka", čije su ključne funkcionalnosti osnova za testiranje u gore spomenutim alatima. Nakon izvršenog testiranja, uočene su određene razlike u karakteristikama i ponašanju alata, na osnovu čega je izvršena analiza alata po određenim parametrima
KLJUČNE REČI: testiranje, korisnički interfejs, GUI, QF-Test, Jacareto
ABSTRACT: GUI (Graphical User Interface) teseting is the process of testing a product's graphical user interface. The main goal is to ensure that user interface (environment) meets its written specifications. In this paper is presented comparative analysis of tools for automated testing of graphical user interface – QF-Test and Jacareto. QF-Test is a software tool for automated testing of GUI for applications written in Java and Web applications. Jacareto is open-source tool for automated testing of GUI for applications written in Java and it consists of two applications: CleverPHL and Picorder. Application "Biblioteka" is used for testing purposes as its main functionalities provide basic input for testing tools. After the testing, spotted differences in characteristics and behavior between these two tools are presented and analysed by certain set of parameters.
KEY WORDS: testing, user interface, GUI, QF-Test, Jacareto

1. UVOD

Većina programerskih firmi se bavi testiranjem svojih aplikacija, međutim i pored toga, krajnji proizvod uvek ima neke nepredviđene probleme. Ljudi koji pišu testove ulažu sve svoje napore da pronađu sve greške u proizvodu pre puštanja nove verzije softvera, ali neki problemi se ponavljaju sa svakom novom izmenom proizvoda. Čak i uz najbolje procese ručnog testiranje softvera postoji poteškoća da se isporuči efektivan, efektan, tačan poboljšan proizvod. Kao jedno od rešenja za uspešno testiranje, nameće se kreiranje automatizovanih testova koji bi po tačno određenom scenariju ispitivali da li se ponašanje softvera poklapa sa njegovim očekivanim ponašanjem.

2. GUI TESTIRANJE

U softverskom inženjerstvu, GUI (*Graphical User Interface – grafički korisnički interfejs*) testiranje je proces testiranja grafičkog okruženja interfejsa kako bi se osiguralo da softver ispunjava zahteve svoje specifikacije. GUI testovi omogućavaju pozivanje i izvršavanje logike sistema i poređenje dobijenih rezultata sa očekivanim, uz mogućnost ponavljanja tih izvršavanja više puta i sa različitim ulaznim podacima. Proveravaju kako aplikacija obrađuje događaje izazvane putem miša ili tastature i kako različite GUI komponente kao što su meniji, dijalozi, dugmići, tekst polja, liste ili kontrole reaguju na korisnikovo ponašanje.[1]

Za razliku od jediničnog (*unit*) testiranja gde se porede izlazne vrednosti metoda koje se testiraju, GUI testovi se sastoje od test slučajeva (*test-case*) koji treba da: identifikuju komponente GUI-a, reaguju na događaje (kao što je npr. klik mišem), obezbede ulazne podatke za komponente (npr. popunjavanje tekstualnog polja), testiraju funkcionalnosti koje su implementirane unutar komponenti, provere izlazne vrednosti kako bi utvrdili da li se podudaraju sa očekivanim vrednostima.[2]

2.1. VRSTE TESTIRANJA

Kada govorimo o testiranju korisničkog interfejsa, sam proces bi se u širem smislu mogao podeliti u četiri kategorije:

1. Testiranje standardizovanosti: Ovaj deo testiranja posvećuje pažnju standardizovanosti aplikacije. Podrazumeva da aplikacija koja se razvija treba da ima neka osnovna svojstva ili funkcije kao što su npr. spuštanje programa u liniju poslova (*minimize*), ponovno otvaranje programa (*restore*) i zatvaranje programa (*close*) dugme ili vidljivost aplikacije u liniju poslova (*task bar*) kada je aktivna.

2. Testiranje validacije: Zavisi od vrste komponenti od kojih se sastoji korisnički interfejs i od funkcija koje treba izvršiti nad njima. Međutim, postoji neki zajednički set pravila. Na primer, tekstualna polja u kojima se kao ulazni parametar očekuje numerička vrednost treba da primaju samo numeričke vrednosti, a ako je u pitanju ComboBox komponenta, treba proveriti da li u njoj postoji neki element (da nije prazna) itd.

3. Testiranje funkcionalnosti: Proverava da li su svi zahtevi softvera ispunjeni. Iako ovi zahtevi variraju od softvera do softvera, postoje neka zajednička pravila kao npr. ako se pored liste sa stavkama nalazi dugme za dodavanje nove stavke, kada se selektuje stavka liste, dugme za dodavnje će biti deaktivirano, dok će dugme za brisanje i izmenu biti

vidljivo. Posle brisanja, obrisana stavka ne sme biti vidljiva na listi, itd.

4. Testiranje upotrebne vrednosti: Deo testiranja koji ispituje koliko se ponašanje i izgled same aplikacije uklapa u želje i potrebe korisnika. Sprovodi se najčešće puštanjem tzv. Beta verzije aplikacije kako bi povratna informacija od korisnika bila iskorišćena u svrhe poboljšanja proizvoda.[3]

2.2. CAPTURE & REPLAY FUNKCIJA

Kada govorimo o GUI testiranju, za kreiranje 'dobrog' test slučaja, potrebno je pokriti sve funkcionalnosti sistema i uveriti se da navedeni slučajevi zaista testiraju korisnički interfejs. Problem u kreiranju ovakvih testova se ogleda u količini operacija koje je potrebno pokriti testovima i u uspostavljanju redosleda kojim se operacije izvršavaju. U te svrhe, razvijeno je automatsko testiranje uz pomoć alata koji se zasnivaju na metodi snimanja određenih sekvenci i njihovog ponavljanja – „Capture & Replay" (u daljem tekstu CR). CR alati funkcionišu kroz neka četiri osnovna koraka:

1. Režim snimanja (*Capture mode*): U toku testiranja CR alat snima sve ručne interakcije između korisnika i sistema koji se testira. Svi CR alati su objektno-orijentisani tj. prepoznaju svaku izabranu GUI komponentu (radio-button, toolbar, textfield) i snimaju svaku karakteristiku objekta (ime, boju, vrednost itd).

2. Programiranje (*Programming*): Snimljeni koraci testa se čuvaju u CR alatu u test skripti napisanoj u nekom programskom jeziku.

3. Provera validnosti (*Checkpoints*): U svrhu proveravanja da li se softver ponaša u skladu sa željenim zahtevima ili postoje neke greške, u toku snimanja akcija, ili nakon toga, mogu se na željena mesta ubaciti provere (*checkpoints*) da li se u određenom trenutku, pri izvršavanju neke operacije, sistem ponaša kako treba.

4. Režim ponovnog puštanja testova (*Replay mode*): Jednom snimljeni testovi mogu biti reprodukovani u svakom trenutku na istovetan način kao pri snimanju, ali ovoga puta bez ljudske intervencije. Pri svakoj reprodukciji, dobijeni parametri se porede sa unapred zadatim, očekivanim parametrima.

CR alati omogućavaju sprovođenje *regresivnog* testiranje aplikacije. Ono što predstavlja naročit izazov za one koji pišu testove, jeste to što izmena korisničkog interfejsa nosi sa sobom i promenu testova kako bi regresivno testiranje imalo smisla. Neki CR alati omogućavaju kreiranje generičkih metoda za testiranje koje jedan logički postupak za testiranje mogu da primene na više konkretnih komponenti. QF-Test i Jacareto su alati za automatsko testiranje korisničkog interfejsa koji implementiraju CR funkciju i o njima će biti reči u daljem delu rada.[4]

2.3. QF-TEST

QF-Test, proizvod *Quality First Software* kompanije, je softverski alat za automatsko testiranje grafičkog korisničkog interfejsa za desktop aplikacije pisane u Java programskom jeziku i web aplikacije (HTML, AJAX, GWT...). Omogućava regresivno testiranje i dostupan je na svim većim UNIX platformama kao i na Windows platformi. „Capture & Replay" funkcija pruža mogućnosti snimanja i reprodukcije testova, dok je uz kreiranje generičkih metoda moguće testirati kompleksnije i veće sisteme. Za napredne korisnike, ova alatka nudi pristup internim programskim strukturama kroz standardni skript jezik Jython, kao Java implementaciju popularnog Python-a.

Moguće je kreirati i *batch* fajl koji će izvršavati testove i generisati XML ili HTML izveštaje, tako da može biti integrisan u već postojeće okruženje za testiranje ili framework kao što je Ant ili Maven.[5]

2.4. JACARETO

Jacareto je *open-source* alat razvijen je od strane nekolicine programera. Omogućava automatsko GUI testiranje aplikacija pisanih u Java programskom jeziku. Sastoji se iz dve aplikacije:

1. CleverPHL – GUI aplikacija koja omogućava snimanje testova uz pomoć „Capture & Replay" funkcije, logovanje izvršavanja testova, prikaz i izmenu testova, procenu izvršavanja i reprodukciju testova.

2. Picorder – konzolna aplikacija koja koristi Jacareto okruženje (*framework*) i omogućava snimanje testova u fajl koji po potrebi može biti reprodukovan pojedinačno ili sa više drugih fajlova.

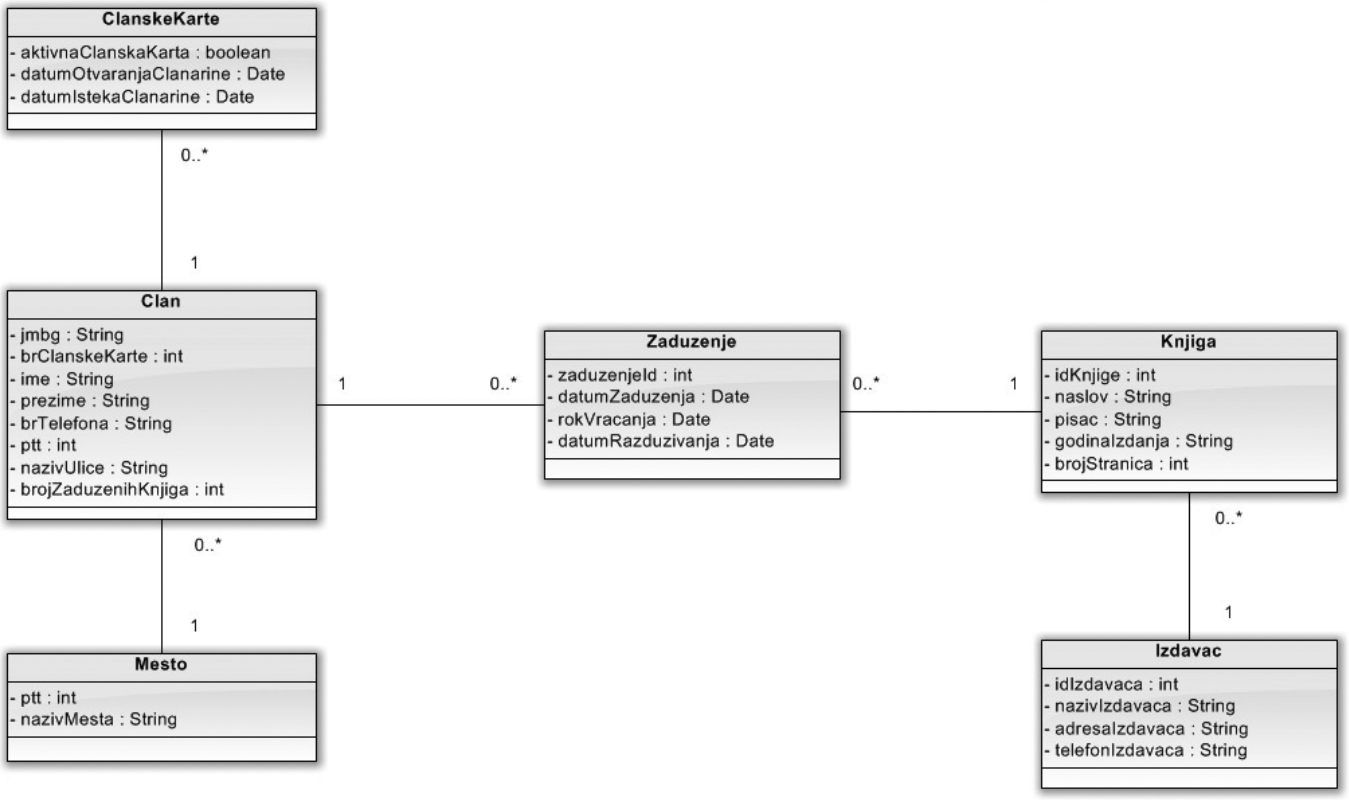
3. TESTIRANJE APLIKACIJE

Testiranje korisničkog interfejsa vršićemo nad aplikacijom pod nazivom „Biblioteka", koja omogućava vođenje evidencije o članovima biblioteke i njihovim zaduženim knjigama u cilju što boljeg funkcionisanja sistema i potraživanja zaostalih dugovanja. Aplikacija obuhvata sve aspekte funkcionisanja biblioteke, od unošenja podataka o novom članu, do evidentiranja svih pozajmljenih knjiga, rokova vraćanja i prekoračenjima.

Polazna tačka za svako testiranje jeste definisanje test slučajeva, odnosno određivanje koji deo logike sistema će se testirati. Za tu svrhu, definisana su tri test slučaja i u okviru svakog od njih, par test metoda.

1. Test član
   1) Test unosa novog člana
   2) Test izmene člana
2. Test knjiga
   1) Test unosa nove knjige
   2) Test izmene knjige
3. Test zaduženje
   1) Test kreiranja zaduženja
   2) Test unetog zaduženja
   3) Test razduživanja knjige

Dijagram klasa može se predstaviti na sledeći način (slika 1):

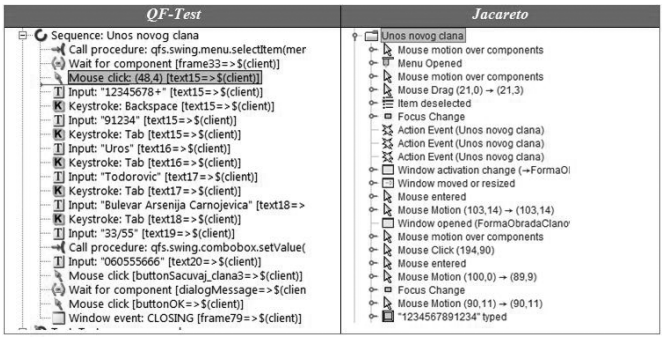


Slika 1. – *Dijagram klasa testne aplikacije*

3.1. TEST ČLAN

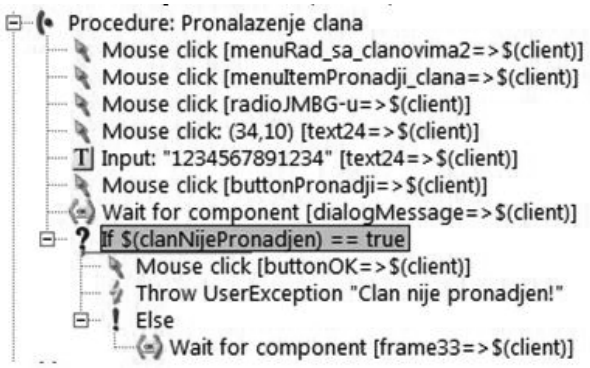
U okviru ovogtest slučaja, testira se grafičko okruženje tj. ekranska forma za čuvanje i izmenu podataka o članovima. Testiranje se odvija u sledećim koracima:

1. Unose se podaci o novom članu
   Interakcija između korisnika i sistema, koja se u alatima predstavlja kao niz akcija nad identifikovanim komponentama kojima se kao ulaz prosleđuju određene vrednosti (slika 2).



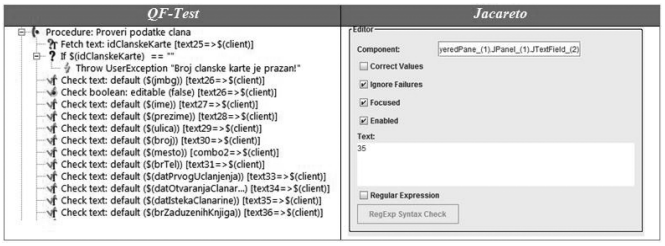
Slika 2. – *Unos podataka o članu*

2. Uneti podaci o članu se čuvaju u bazi
   Klikom na odgovarajuće dugme, konvertuju se podaci iz grafičkih elemenata koji se prosleđuju odgovarajućoj metodi poslovne logike.
3. Vrši se pretraga članova na osnovu JMBG-a unetog člana
   QF-Test dozvoljava korišćenje procedure. Procedura predstavlja izdvojen deo koda, definisan da se može pozivati po potrebi, tako da se procedura „Pronalaženje člana“ poziva i u koraku (3) i koraku (6). U okviru nje se nalazi „IF naredba“ koja u zavisnosti od rezultata odziva komponente baca definisan izuzetak ili nastavlja sa radom (slika 3).



Slika 3. – *Procedura „Pronalaženje člana“*

4. Proverava se da li su podaci o članu ispravno sačuvani
   Podaci sa forme se porede sa unapred zadatim podacima. Uz pomoć dodatnih opcija, moguće je definisati i pod kojim uslovima se vrši poređenja podataka (da li će se greške ignorisati, ispraviti, baciti izuzetak itd.) (slika 4).



Slika 4. – *Testiranje podataka*

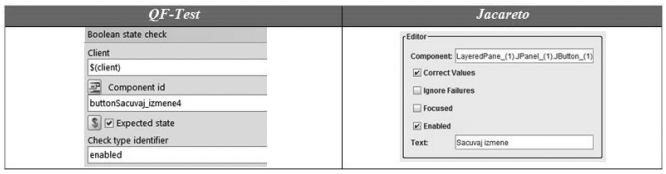
QF-Test nudi mogućnost skladištenja varijabli koje mogu biti korišćene u okviru test metode. Tako smo vrednost tekstualnog polja idClanskeKarte sačuvali u promenljivoj, kako bismo je u narednom koraku uporedili sa nekom vrednošću.

5. Podaci o članu se menjaju
6. Vrši se pretraga članova na osnovu JMBG-a izmenjenog člana
7. Proverava se da li su podaci člana ispravno izmenjeni

3.2. TEST KNJIGA

U okviru ovog test slučaja, testira se grafičko okruženje tj. ekranska forma za čuvanje i izmenu podataka o knjigama. Testiranje se odvija u sledećim koracima:

1. Unose se podaci o novoj knjizi
2. Uneti podaci o knjizi se klikom na određeno dugme čuvaju u bazi
3. Vrši se pretraga knjiga na osnovu naziva unete knjige
4. Proverava se da li su podaci o knjizi ispravno sačuvani
5. Podaci o knjizi se menjaju
6. Vrši se pretraga knjiga na osnovu naziva izmenjene knjige
7. Proverava se da li su podaci o knjizi ispravno izmenjeni
   Prilikom provere izmenjenih podataka, vrši se i provera funkcionalnosti grafičkih elemenata, kao npr. da li je dugme uključeno (slika 5).



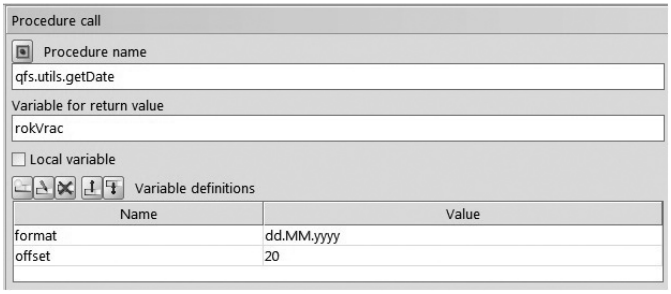
Slika 5. – *Testiranje funkcionalnosti grafičkih elemenata*

Jacareto nudi mogućnost automatske korekcije vrednosti koja se prilikom testiranja pokaže kao netačna. Tako ćemo u slučaju prikazanom na slici 5. ukoliko se desi da dugme ima drugi naziv, odmah izmeniti naziv na samoj aplikaciji na željenu vrednost.

3.3. TEST ZADUŽENJE

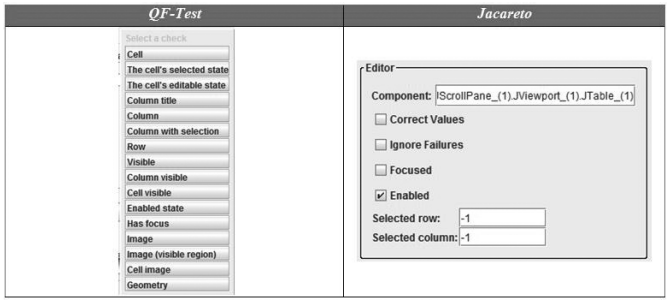
U okviru ovog test slučaja, testira se grafičko okruženje za kreiranje novog zaduženja knjige i razduživanje knjige i neke nove komponente kao što je tabela. Testiranje se odvija u sledećim koracima:

1. Proverava se da li su ispravni podaci na formi za kreiranje novog zaduženja knjige.
2. Kreira se novo zaduženje knjige za određenog člana
   QF-Test korisniku stavlja na raspolaganje biblioteku „qfs“ sa gotovim procedurama koje znanto olakšavaju testiranje. U okviru te biblioteke, nalazi se i procedura vezana za čitanje sistemskog vremena, koju smo iskoristili kako bismo popunili tekstualno polje „Rok vraćanja knjige“ kao datum koji predstavlja dvadeseti dan od trenutnog datuma (slika 6).



Slika 6. – *Procedura u QF-Testu*

3. Pronalaze se sva zaduženja za tog člana
4. Proveravaju se podaci zadužene knjige u tabeli zaduženja
   QF-Test omogućava testiranje komponente kao što je tabela tako što prepoznaje svaki pojedinačno selektovani red u tabeli i svaku pojedinačnu kolonu u redu, pritom nudeći mogućnost testiranja raznih atributa kolone tj. reda. Jacareto prepoznaje tabelu kao objekat, ali može testirati samo da li je neki red, ili kolona u njemu, selektovan.

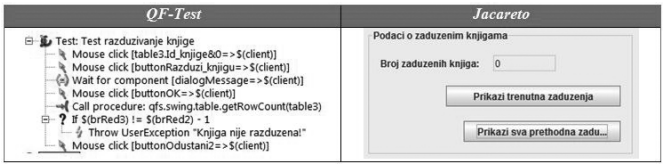


Slika 7. – *Testiranje tabele*

U našem slučaju, u alatu QF-Test, iskoristili smo opciju „Cell“ koja nam je vratila trenutnu vrednost selektovane ćelije koju smo poredili sa unapred zadatom vrednošću, dok smo u alatu Jacareto testirali samo to da li je neki red u tabeli selektovan. (slika 7)

5. Razdužuje se knjiga
6. Proverava se da li je knjiga razdužena
   Pozivanjem procedure getRowCount() iz paketa „qfs“ u alatu QF-Test, došli smo do trenutnog broja redova u tabeli, koji bi posle razduživanja knjige trebalo da bude jed-

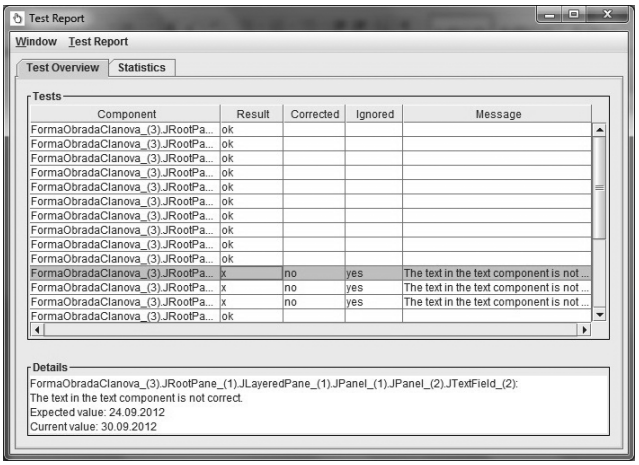
nak broju redova pre razduživanja umanjenim za jedan. Sa druge strane, Jacareto je izvršio proveru razdužene knjige tako što je na formi za unos članova proverio da li je vrednost tekstualnog polja „Broj zaduženih knjiga“ jednaka nuli. (slika8) Jacareto dozvoljava ostavljanje komentara u toku izvršavanja testova, tako da smo u ovom slučaju dodali komenatar da je proveravanj vrednosti tekstualnog polja „Broj zaduženih knjiga“ izvršeno zbog nemogućnosti provere sadržaja ćelija tabele. Taj komentar se pojavljuje u log fajlu programa.



Slika 8. – *Testiranje razduživanja knjige*

3.4. REZULTATI TESTIRANJA

Rezultati testiranja se mogu videti kroz izveštaj. Oba alata nude pregled rezultata izvršenih testiranja, konkretno, svake test metode kroz koju je vršeno poređenje.



Slika 9. – *Test izveštaj u alatu Jacareto*

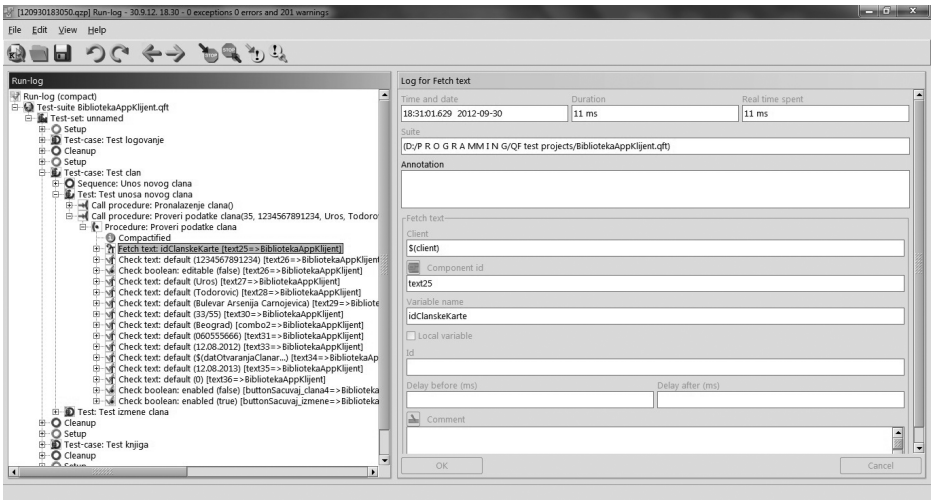
4. ANALIZA PERFORMANSI

Nakon testiranja aplikacije, možemo napraviti analizu performansi korišćenih alata na osnovu mogućnosti koje smo uočili pri testiranju. (tabela 1).

1.	**Logovanje grešaka pri testiranju (iznenadni otkazi koji nisu u test-case-u)**	Da	Ne
2.	**Rad sa izuzecima**	Da	Ne
3.	**Ostavljanje komentara u toku izvršavanja**	Ne	Da
4.	**Skladištenje varijabli u okviru testa**	Da	Ne
5.	**Korišćenje sistemskog vremena**	Da	Ne
6.	**Generalizacija test metoda**	Da	Ne
7.	**Snimanje glasa tokom snimanja test scenarija**	Ne	Da
8.	**Debugg-ovanje**	Da	Ne
9.	**Pravljenje izveštaja**	Da	Da
10.	**Mogućnost ručnog okidanja testova**	Da	Da

Tabela 1. – *Analiza performansi alata*

1. **Logovanje grešaka pri testiranju (iznenadni otkazi koji nisu u test slučaju)**: U toku testiranja moguće je doći do otkaza koji je posledica greške koja nije u test slučaju, kao što je npr. da se ne pojavi forma koja se očekuje u test scenariju. U takvim slučajevima, QF-Test baca izuzetak i loguje grešku koju čini niz pozivanih funkcija (**stacktrace**) i slika izgleda test okruženja u trenutku nastanka greške. Jacareto, iako pruža mogućnost logovanja izvršavanja testova, iznenadne otkaze ne registruje, tako da nije moguće ući u trag razlogu zbog kojeg je testiranje stalo.
2. **Rad sa izuzecima**: Ukoliko na nekom mestu u aplikaciji očekujemo grešku i želimo da definišemo specifičnu akciju ukoliko do nje dođe,QF-Test nudi mogućnost realizacije te akcije definisanjem **try-catch-finally** bloka u test slučaju. Moguće je odabrati i tačan tip izuzetka koji se očekuje.
3. **Ostavljanje komentara u toku izvršavanja**: U toku snimanja test scenarija, alat Jacareto omogućava ostavljanje komentara koji se mogu videti u log fajlu koji



Slika 10. – *Test izveštaj u alatu QF-Test*

se kreira nakon svake reprodukcija testa. QF-Test alat nema ovu mogućnost.

4. **Skladištenje varijabli u okviru testa**: QF-Test omogućava skladištenje vrednosti u varijable. To znači da se jednom definisane, ili sa korisničkog interfejsa pročitane, vrednosti mogu koristiti na više mesta na nivou test slučaja ili celog testa (**test-suite**).
5. **Korišćenje sistemskog vremena**: U toku testiranja je ponekad potrebno doći do nekih opštih podataka kao što je npr. trenutno vreme u sistemu. Sa QF-Test-om je uz korišćenje biblioteke "qfs" moguće doći do ove vrednosti i koristiti je u daljem toku testiranja na željeni način. Jacareto sa druge strane pruža uvid u sistemsko vreme, ali samo kao informaciju, tako da se ne može iskoristiti u daljem toku testiranja.
6. **Generalizacija test metoda:** Kada želimo da skratimo postupak testiranja tako što ćemo izdvojiti logiku koja se ponavlja i iskoristiti je na određenim mestima, u QF-Testu taj deo logike kreiramo kroz proceduru. Moguće je definisati i ulazne parametre procedure, tako da se može pozivati za različite vrednosti. Tip ulaznog parametra može biti i identifikaciona oznaka komponente, čime se generalizacija podiže na još veći nivo.
7. **Snimanje audio zapisa uz test scenario**: Vreme potrebno za izvršavanje testa pisanog u Jacareto alatu je jednako onom koje je bilo potrebno za kreiranje samog testa. To znači da se test ne ubrzava prilikom reprodukcije, već da je svaki trenutak u toku snimanja na identičan način ponovljen i u toku reprodukcije. Zbog ove funkcije, moguće je snimiti audio zapis tj. komentarisati izvođenje testa koje će biti reprodukovano svaki put kada test bude pušten. Sa druge strane QF-Test ubrzava izvršavanje testova, pa stoga ne nudi ovu mogućnost.
8. **Debugg-ovanje:** U okviru QF-Test-a je integrisan i debugger – alat koji omogućava praćenje izvršavanja testa korak po korak i manipulisanje podacima u toku samog procesa. Moguće je selektovati čvor na kome će se izvršavanje testa zaustaviti radi pregleda stanja (**breakpoint**).
9. **Pravljenje izveštaja:** Nakon izvršenog testa, oba alata pružaju uvid u rezultate testiranja kroz test izveštaj. U njemu se nalazi spisak svih test metoda, grešaka i upozorenja koja su nastala u toku izvršavanja.
10. **Mogućnost ručnog okidanja testova:** Oba alata nude mogućnost automatskog izvršavanja testova uz automatsko kreiranje test izveštaja. QF-Test može poslati test izveštaj na željenu e-mail adresu.

5. ZAKLJUČNO RAZMATRANJE

QF-Test se, nudeći neke naprednije funkcije, pokazao kao dosta kvalitetniji alat za testiranje u odnosu na Jacareto. Mogućnost debugg-ovanja, skladištenja varijabli u testu, biblioteka sa gotovim funkcijama, samo su neke od pogodnosti koje nudi ovaj alat. Jacareto sa druge strane, iako alat za testiranje, poseduje neke elemente koji ga svrstavaju u alate za prezentaciju. Nudi snimanje i izvršavanja programa uz audio komponentu, pa i menjanje izgleda aplikacije u toku izvršavanja uz pomoć crtanja. Pravac u kome bi se dalje trebalo razvijati testiranje korisničkog interfejsa je integracija test okruženja sa skladištem podataka. Time bi u toku testiranja bilo moguće menjanje stanja objekta, postavljanje okruženja direktno kroz bazu podataka i dobijanje informacija o stanju nekih atributa objekta. To bi omogućilo testiranje podataka koji nisu prikazani korisniku, a koji se indirektno menjaju kroz akcije koje sprovodi nad korisničkim interfejsom.

6. REFERENCE

[1] AppPerfect Corp. Gui testing (30.09.2012). http://www.appperfect.com/products/application-testing/app-test-gui-testing.html
[2] Alessandro Marchetto (30.09.2012). GUI Based Testing. http://selab.fbk.eu/swat/slide/2.5_GUI.ppt
[3] ASK-Labs (30.09.2012). GUI Testing – Checking Application Functions. http://www.comparesuite.com/solutions/tests-automation/hb_gui_testing_introduction.htm
[4] Tilo Linz, Matthias Daigl (30.09.2012). Capture-and-Replay Tools. http://www.imbus.de/forschung/pie-gui-test/how-to-automate-testing-of-graphical-user-interfaces/
[5] QF-Test (30.09.2012). http://www.qfs.de/
[6] Saša D. Lazarević, Marko Petrović, "Korelacija jednakosti, HASHING-a i nasleđivanja", časopis INFO M god 2, sv. 8, Beograd, 2003.
[7] Saša D. Lazarević, Dušan Marković, Ivan Stamenić, "Konstrukcija korisničkog interfejsa integracijom FLASH komponenti u .NET WIN APP", časopis INFO M god 7, sv. 28, Beograd, 2008.

Džudović Aleksandra, student master studija na Smeru za softversko inženjerstvo na Fakultetu organizacionih nauka Univerziteta u Beogradu.
Kontakt: dzudovic.a@gmail.com
Oblasti interesovanja: .NET, testiranje, refaktorisanje, baze podataka

Saša D. Lazarević, Katedra za softversko inženjerstvo, Fakultet organizacionih nauka, Univerzitet u Beogradu
Kontakt: sasa.lazarevic@fon.bg.ac.rs
Oblasti interesovanja: konstrukcija softvera, testiranje softvera, kvalitet softvera, razvoj informacionih sistema, structure podataka i algoritmi, baze podataka, sistemi za upravljanje dokumentacijom, .NET platforma