**Tehničko veleučilište u Zagrebu**

**Programsko inžinjerstvo u otvorenim sustavima**

**Funkcionalno testiranje**

“SEMINARSKI RAD”

Nikola Platnjak, Dorian Hajnić, Kristina Aničić, Nina Šalković

G3T3

Zagreb, travanj 2024.

# Sadržaj

[1. Sadržaj 2](#_Toc163247531)

[2. Uvod 3](#_Toc163247532)

[3. Funkcionalno i nefunkcionalno testiranje 4](#_Toc163247533)

[4. Proces funkcionalnog testiranja 5](#_Toc163247534)

[5. Vrste funkcionalnog testiranja 6](#_Toc163247535)

[6. Automatizacija 11](#_Toc163247536)

[7. Zaključak 12](#_Toc163247537)

[8. Literatura 13](#_Toc163247538)

# Uvod

Funkcionalno testiranje je način testiranja koji ispituje radi li svaki aspekt softverskog proizvoda na način koji je definiran u specifikacijama. Svaka funkcija definirana u programu se uspoređuje sa njoj odgovarajućim zahtjevom kako bi se utvrdilo je li njen izlaz u skladu sa očekivanjima krajnjeg korisnika.

Funkcionalno testiranje može omogućiti identifikaciju nedostataka ili grešaka u softveru koje su propuštene u razvojnom procesu, a bile bi nepoželjne isporučiti krajnjem korisniku. Nadalje, funkcionalno testiranje pomaže osigurati da softver bude jednostavan i intuitivan za korištenje, što omogućava veće zadovoljstvo i korisnost za korisnika. Takav način smanjivanja rizika od grešaka i fokusa na funkcionalnost i korisnost osigurava visoku kvalitetu softvera.

Bez funkcionalnog testiranja postoji rizik da softver neće raditi na način koji se očekuje, što može rezultirati nezadovoljstvom korisnika te oštećenjem reputacije tvrtke, tima ili osobe koja je zadužena za razvoj softvera.

Testiranje se vrši pružanjem testnih ulaznih podataka, kontrolom njima izazvanih izlaza te provjerom jesu li dobiveni izlazi jednaki očekivanim izlazima.

# Funkcionalno i nefunkcionalno testiranje

Funkcionalno testiranje svoj fokus stavlja na rezultate obrade testiranja, to jest izlaznih faktora, te minimalna očekivanja korisnika; ono ispituje 'Što proizvod radi.' Za razliku od njega, postoji i ne-funkcionalno testiranje, koje se usmjerava na istraživanje kvalitete, sigurnosti ili performansi izvornog koda aplikacije. Funkcionalno testiranje ne mjeri brzinu, skalabilnost i pouzdanost - mehaniku obrade, već samo ispituje fikcionalne aspekte - rezultate obrade.

Razlike se mogu prikazati sljedećom tablicom:

|  |  |  |
| --- | --- | --- |
| Parametar | Funkcionalno testiranje | Ne-funkcionalno testiranje |
| Definicija | Provjerava operacije i akcije aplikacije | Provjerava ponašanje aplikacije |
| Testiranje se bazira na | Zahtjevima korisnika | Očekivanjima korisnika |
| Cilj | Validirati akcije (izlazne vrijednosti) softvera | Performanse softverskog sustava |
| Zahtjevi | Funkcionalna specifikacija | Specifikacija performansi |
| Funkcionalnost | Što proizvod radi | Kako proizvod radi |

U tom smislu, definicija funkcionalnog testiranja može se asocirati sa definicijom 'testiranja crne kutije', dok je definicija ne-funkcionalnih testova bliža definiciji 'testiranja bijele kutije'.

To jest - Testiranje crne kutije provjerava funkcionalnost sustava bez previše brige o unutarnjim implementacijama. Identificira greške na razini sučelja i rješava ih. Dok testiranje bijele kutije temeljito ispituje i analizira sustav provjerom njegovog unutarnjeg programiranja. Na taj način pomaže u pronalaženju grešaka na razini dizajna, rekonstrukciji sustava, pokretanju i izvođenju.

No, to ne mora značiti da je svako funkcionalno testiranje crne kutije, niti da je svako ne-funkcionalno testiranje bijele kutije; što će biti prikazano u samim vrstama funkcionalnih testova.

Slika na kojoj se prikazuje kutija, tekst, dizajn, origami

Opis je automatski generiran

Slika 1: Testiranje crne i bijele kutije

# Proces funkcionalnog testiranja

Plan funkcionalnog testiranja može se definirati slijedom:

**Identificiranje ciljeva testiranja -** Ciljevi testiranja uključuju potvrdu da aplikacija radi onako kako je bilo namijenjeno te da obrađuje pogreške i neočekivane scenarije na prikladan način.

**Stvaranje testnih scenarija -** Razvija se popis svih mogućih (ili barem svih najvažnijih) testnih scenarija za određenu značajku. Testni scenariji opisuju različite načine na koje će se značajka koristiti. Na primjer, za modul plaćanja, testni scenariji mogu uključivati više valuta, obradu nevažećih ili isteklih brojeva kartica te generiranje obavijesti o uspješnom završetku transakcije.

**Stvaranje testnih podataka -** Stvaraju se testni podaci koji simuliraju normalne uvjete korištenja na temelju identificiranih testnih scenarija. Testni podaci mogu se unijeti ručno (npr. iz Excel tablice ili ispisa) ili automatski putem skripte ili alata za testiranje koji čita i unosi podatke iz baze podataka, datoteke, XML-a ili tablice. Svaki skup ulaznih podataka također bi trebao imati povezane podatke koji opisuju očekivani rezultat koji bi ulazni podaci trebali generirati.

**Dizajniranje testnih slučajeva -** Testni slučajevi se kreiraju na temelju različitih željenih ishoda za testne ulaze. Na primjer, ako se unese nevažeći broj kreditne kartice, aplikacija bi trebala prikazati smislenu poruku o grešci.

**Izvršavanje testnih slučajeva -** Testni slučajevi se pokreću kroz aplikaciju te se stvarni ishodi uspoređuju s očekivanim rezultatima. Ako su stvarni i očekivani izlazi različiti, značajka je propustila test i greška bi trebala biti zabilježena.

**Praćenje i rješavanje grešaka -** Nakon što se identificira greška, treba je zabilježiti u formalnom sustavu praćenja koji je dostupan cijelom projektu. Potrebne promjene trebaju biti napravljene u aplikaciji, a testni slučaj izvršen ponovno kako bi se potvrdilo rješenje prije nego što se greška označi kao zatvorena.

# Vrste funkcionalnog testiranja

Postoji velik broj vrsta testova koje bismo mogli definirati kao funkcionalne, a neki od njih svojom primjenom mogu obuhvatiti i aspekte ne-funkcionalnog testiranja. Vrste testova koje će se provesti odabiru se ovisno o potrebama i mogućnostima softverskog proizvoda koji obrađujemo.

Neki projekti, poput tražilica za turističke agencije, mogu zahtijevati vrhunsku implementaciju i nevjerojatnu responzivnost kako bi zadovoljili očekivanja korisnika, te time zahtijevaju veći broj testova te fokusiranje na mnoge branše. Drugi, poput interne aplikacije za obradu poslovnih dokumenata, možda manje brinu o performansama softvera, pa se fokus prebacuje na sigurnost sustava.

Slijedeće su neke od relevantnijih i najčešće korištenih vrsta funkcionalnog testiranja:

|  |  |
| --- | --- |
| 1. Jedinično testiranje 2. Integracijsko testiranje 3. Testiranje sučelja 4. Testiranje sustava | 1. Regresijsko testiranje 2. 'Smoke' testiranje, ili testiranje dima 3. Testiranje ispravnosti 4. Testiranje prihvaćanja korisnika |

1. Jedinično testiranje

Jedinično testiranje (eng. Unit testing) je vitalan tip funkcionalnog testiranja koji uključuje testiranje najmanje funkcionalne jedinice koda.

Jedinično testiranje uglavnom obavljaju programeri, jer ono koristi tehniku testiranja bijele kutije. Izvodi se tijekom najranijih faza razvoja, što pomaže u otkrivanju grešaka tijekom početnih faza, prije nego što mogu postati veći problemi. U jediničnim testiranju koriste se različite tehnike, uključujući pokrivenost grananja, pokrivenost izjava, analizu graničnih vrijednosti i pokrivenost odluka:

* *Pokrivenost grananja* osigurava da su sve logičke putanje i uvjeti pokriveni tijekom testiranja.
* *Pokrivenost izjava* osigurava da su tijekom testiranja sve izjave u funkciji ili modulu barem jednom prisutne.
* *Analiza graničnih vrijednosti* stvara testne podatke za granične vrijednosti i za vrijednosti koje leže odmah prije i poslije granice, a zatim pokreće testni slučaj koristeći sve stvorene skupove podataka.
* *Pokrivenost odluke* osigurava da su tijekom izvršavanja kontrolnih struktura poput "Do-While" ili "Case statement" sve putanje odluka testirane.

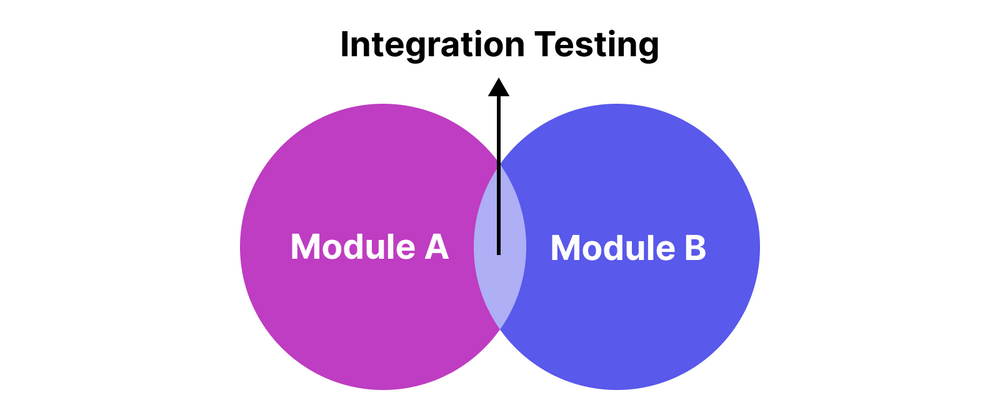
1. Integracijsko testiranje

Tijekom integracijskog testiranja cilj je potvrditi interakciju između dviju ili više jedinica testiranih komponenti softverskog sustava. Za postizanje toga, testira se slijedeće:

* Komunikacija naredbi, podataka, poziva baza podataka, API poziva te obrade mikro-servisa između jedinica
* Nema neočekivanog ponašanja tijekom određene integracije

Postoje dva tipa integracijskog testiranja: Inkrementalno i Big-Bang.

*Inkrementalno* integracijsko testiranje uključuje kombiniranje jedne ili više komponenti u fazama i testiranje sve dok cijeli sustav nije uspješno testiran. Postoje tri pristupa inkrementalnom integracijskom testiranju: *Od vrha prema dnu* (Top-Down), *Od dna prema vrhu* ( Bottom-Up) i *Hibridni pristup*.

*Big-Bang* testirnje, s druge strane, uključuje integriranje i testiranje svih komponenti kao cjelovitog sustava.

Slika 2: Jednostavan grafički prikaz principa integracijskog testiranja

1. Testiranje sučelja

Testiranje sučelja i integracijsko testiranje su dva različita tipa testiranja softvera koja se provode kako bi se osiguralo dobro funkcioniranje softverskih aplikacija. Testiranje sučelja je uglavnom usmjereno na testiranje komunikacije između različitih sučelja, dok se integracijsko testiranje usredotočuje na testiranje integrirane grupe modula kao jedinstvene cjeline.

Tijekom testiranja sučelja, testira se ispravnost razmjene podataka, prijenosa podataka, poruka, poziva i naredbi između dviju integriranih komponenata. Testira se i komunikacija između baza podataka, web-servisa, API-ja ili bilo koje vanjske komponente i aplikacije. Tijekom ove komunikacije podataka ili naredbi ne bi trebalo biti nikakvih pogrešaka ili neusklađenosti formata. Ako se naiđe na takav problem, potrebno ga je ispraviti.

1. Testiranje sustava

Testiranje sustava ključna je faza u životnom ciklusu testiranja softvera koja osigurava da su svi dijelovi sustava besprijekorno integrirani i rade zajedno prema specifikacijama zahtjeva. Ova tehnika testiranja koristi metodu crne kutije koja provjerava integrirani sustav prema unaprijed definiranim zahtjevima.

Testiranje sustava provodi se u gotovo stvarnom okruženju i prema stvarnoj uporabi. Bitno je provesti testiranje sustava prije testiranja prihvaćanja korisnika kako bi se osiguralo da je sustav potpuno funkcionalan i ispunjava poslovne zahtjeve.

1. Regresijsko testiranje

Nakon što programeri naprave poboljšanja ili isprave kod, u bilo kojoj fazi, važno je pokrenuti skup testova regresije. Njima se osigurava da promjene u kodu nisu utjecale na postojeće funkcionalnosti i da nisu uvele nove greške.

Testovi regresije su podskup funkcionalnih testova koji pokrivaju glavne funkcionalnosti sustava. Slučajevi moraju biti ažurni (dodani i izbrisani) prema promjenama aplikacije. Testovi regresije najbolji su kandidati za testiranje automatizacijom, budući da se često pokreću i zahtijevaju vrijeme za izvođenje. Testovi regresije koji će se pokrenuti mogu se odabrati na tri načina: pokrenuti cijeli skup testova regresije, odabrati testne slučajeve visokog prioriteta iz skupa regresije ili odabrati slučajeve iz skupa regresije koji testiraju funkcionalnosti povezane s promjenama u kodu.

1. Smoke testiranje

Smoke testiranja, također nazvano testiranje povjerenja, metoda je testiranja softvera koja se koristi kako bi se utvrdilo je li nova 'izgradnja' (eng. build) softvera spremna za sljedeću fazu testiranja. Ova metoda testiranja određuje jesu li najvažnije funkcije programa ispravne, ali ne ulazi u sitnije detalje.

Kao preliminarni pregled softvera, smoke testiranje pronalazi osnovne i ključne probleme u aplikaciji prije nego što se obavi detaljnije testiranje. Ako softver prođe smoke test, timovi za osiguranje kvalitete nastavljaju s daljnjim testiranjem. Neuspjeh znači da softver mora biti vraćen razvojnom timu.

1. Testiranje ispravnosti

Testiranje ispravnosti je vrsta testiranja softvera kojoj je cilj brzo ocijeniti radi li osnovna funkcionalnost novog izdanja softvera ispravno ili ne. Obično se provodi na izdanjima koja su u početnim fazama razvoja prije nego što se izvrši potpuno regresijsko testiranje.

Testiranje ispravnosti je ograničeno u opsegu i obično se usredotočuje na ključne funkcionalnosti, a ne cilja otkriti svaku moguću pogrešku ili grešku u sustavu. To je brz i lagani način osiguravanja da softver radi kako se očekuje prije nego što se provode daljnja testiranja.

1. Testiranje prihvaćanja korisnika

Testiranje prihvaćanja korisnika, ponekad nazvano beta testiranje ili testiranje krajnjeg korisnika, faza je razvoja softvera u kojoj se softver testira u "stvarnom svijetu" od strane ciljane publike ili poslovnog predstavnika. Ova vrsta testiranja nije namijenjena vođenju preko izbornika, već ju obavljaju poslovni korisnici kako bi provjerili hoće li aplikacija zadovoljiti potrebe krajnjeg korisnika, s scenarijima i podacima koji su reprezentativni za stvarnu upotrebu u praksi.

Svrha testiranja prihvaćanja korisnika je potvrditi dva ključna aspekta:

* Korisnički zahtjevi. - Odgovara li aplikacija korisničkim očekivanjima? Mogu li korisnici intuitivno navigirati i uživati u funkcionalnostima aplikacije?
* Poslovni zahtjevi. - Može li aplikacija učinkovito i efektivno obraditi stvarne slučajeve uporabe?

Drugim riječima, softver bi trebao pomoći korisnicima da obavljaju zadatke u stvarnom svijetu bez prepreka.

Potvrda ovih zahtjeva obično dolazi u obliku odobrenja dionika, npr. kada vaš klijent bude zadovoljan konačnom verzijom aplikacije ili web stranice.

# Automatizacija

Ručno funkcionalno testiranje zahtijeva od programera ili testnog inženjera da dizajnira, stvori i izvrši svaki test ručno. Takav način testiranja je fleksibilan i može biti moćno uz pravu uporabu. Međutim, kako softverski produkt postaje složeniji, a vremenski periodi za izdavanje novih verzija postaju kraći, ručna strategija testiranja će se suočiti s izazovima u održavanju visokog stupnja pokrivenosti testovima.

Automatizirano funkcionalno testiranje automatizira mnoge dijelove procesa testiranja, omogućavajući testovima da se izvršavaju kontinuirano bez ljudske interakcije - i s manjom šansom za ljudsku pogrešku. Testovi još uvijek moraju biti dizajnirani i njihovi rezultati evaluirani od strane ljudi, ali nedavna poboljšanja u umjetnoj inteligenciji znače da uz pravi alat sve veći dio tereta može biti rukovano autonomno.

Neke od prednosti automatiziranog funkcionalnog testiranja uključuju sljedeće:

**Pruža trenutne povratne informacije** - Kada se uvedu nove značajke i funkcionalnosti u kod, svi se jedinični testovi mogu automatski ponovno pokrenuti kako bi se osiguralo da nema problema i da postojeće značajke funkcioniraju kako se očekuje. Budući da jedinični testovi rade na malim dijelovima koda, ovi testovi se gotovo trenutno izvršavaju i daju odmah povratnu informaciju.

**Štedi vrijeme i smanjuje količinu rada -** Automatizirano funkcionalno testiranje omogućuje brzo i ponovljeno izvođenje testova, bez potrebe za ručnim intervencijama, čak i preko više verzija aplikacije.

**Povećava pokrivenost testova**: Implementacija, na primjer, jediničnog testiranja, integracijskog testiranja i regresijskog testiranja zajedno pokriva različite slojeve i komponente aplikacije. To osigurava da je kod testiran na iscrpan način.

**Smanjuje rizike**: Zahvaljujući brzim povratnim informacijama, automatizirano funkcionalno testiranje smanjuje rizike povezane s aplikacijom već u ranim fazama razvojnog procesa. Na primjer, kada jedinični test ne uspije, lako se može utvrditi osnovni uzrok i riješiti problem prije nego što se proširi na više razine testiranja.

# Zaključak

Funkcionalno testiranje je temelj testiranja softvera koje osigurava da softverske aplikacije zadovoljavaju zahtjeve i pružaju izvrsno korisničko iskustvo. Ono potvrđuje da svaka komponenta softvera radi kako se očekuje. Da bi se funkcionalno testiranje učinkovito provodilo, treba slijediti strukturirani pristup. To uključuje temeljito planiranje, dizajniranje testnih slučajeva, izvršenje i izvještavanje.

Kod provedbe funkcionalnog testiranja bitno je odbrati vrste testiranja koje su prikladne za veličinu i potrebe sustava te mogućnosti tima ili tvrtke koja proizvodi softverski proizvod. Ako je sustav veći, zahtjevniji i kompleksniji, provodi se veći broj testova u češćem intervalu. Ako je potrebno, dobro je i uvesti neki od automatskih sustava testiranja.

Razvojem AI tehnologija uvelike se razvila i efikasnost i brzina, to jest mogućnost alata za automatsko testiranje. Oni su široko dostupni i generalno pružaju visok povrat na ulaganje, pa je tako sve bitnije dobro razmisliti koje resurse koristiti u testiranju te za veće sustave koristiti svu pomoć koja je njima prikladna.

# Literatura

<https://www.opentext.com/what-is/functional-testing>

<https://testerwork.com/the-importance-of-functional-testing/>

<https://www.geeksforgeeks.org/software-testing-functional-testing/>

<https://www.careerist.com/insights/a-guide-to-white-box-black-box-and-gray-box-testing>

<https://www.browserstack.com/guide/black-box-testing-and-white-box-testing>

<https://testsigma.com/guides/functional-testing/#Functional_Testing_Types>

<https://saucelabs.com/resources/blog/automated-functional-testing-an-introduction-to-efficient-software-testing>