Značaj testiranja za osiguranje kvalitete softvera

Paladin, Mikele

Undergraduate thesis / Završni rad

2022

Degree Grantor / Ustanova koja je dodijelila akademski / stručni stupanj: **University of Pula / Sveučilište Jurja Dobrile u Puli**

Permanent link / Trajna poveznica: https://urn.nsk.hr/urn:nbn:hr:137:027821

Rights / Prava: In copyright / Zaštićeno autorskim pravom.

Download date / Datum preuzimanja: **2024-07-16**

Repository / Repozitorij:

Digital Repository Juraj Dobrila University of Pula

Sveučilište Jurja Dobrile u Puli
Tehnički fakultet u Puli

Mikele Paladin

Značaj testiranja za osiguranje kvalitete softvera

Završni rad

Pula, rujan, 2022. godine.

Sveučilište Jurja Dobrile u Puli

Tehnički fakultet u Puli

Mikele Paladin

Značaj testiranja za osiguranje kvalitete softvera

Završni rad

JMBAG:0303090460, redoviti student

Studijski smjer: Računarstvo

Predmet: Kvaliteta u informacijsko – komunikacijskoj tehnologiji

Znanstveno područje: Područje tehničkih znanosti

Znanstveno polje: Računarstvo

Znanstvena grana: Programsko inženjerstvo

Mentor: Izv. prof. dr. sc. Giorgio Sinković

Pula, rujan, 2022. godine

Tehnički fakultet u Puli

Ime i prezime studenta/ice Mikele Paladin

JMBAG 0303090460

Status: ☒ redoviti ☐ izvanredni

PRIJAVA TEME ZAVRŠNOG RADA

Giorgio Sinković

Ime i prezime mentora

Računarstvo

Studij

Kvaliteta u informacijsko-komunikacijskoj tehnologiji

Kolegij

Potvrđujem da sam prihvatio/la temu završnog/diplomskog rada pod naslovom:

Značaj testiranja za osiguranje kvalitete softvera

(na hrvatskom jeziku)

Importance of testing for software quality assurance

(na engleskom jeziku)

Datum: 14.4.2022.

IZJAVA O AKADEMSKOJ ČESTITOSTI

Ja, dolje potpisani **Mikele Paladin** , kandidat za prvostupnika računarstva ovime izjavljujem da je ovaj Završni rad rezultat isključivo mojega vlastitog rada, da se temelji na mojim istraživanjima te da se oslanja na objavljenu literaturu kao što to pokazuju korištene bilješke i bibliografija. Izjavljujem da niti jedan dio Završnog rada nije napisan na nedozvoljen način, odnosno da je prepisan iz kojega necitiranog rada, te da ikoji dio rada krši bilo čija autorska prava. Izjavljujem, također, da nijedan dio rada nije iskorišten za koji drugi rad pri bilo kojoj drugoj visokoškolskoj, znanstvenoj ili radnoj ustanovi.

Student

U Puli, 28.09., 2022. godine

IZJAVA

o korištenju autorskog djela

Ja, **Mikele Paladin** dajem odobrenje Sveučilištu Jurja Dobrile u Puli, kao nositelju prava iskorištavanja, da moj Završni rad pod nazivom „Značaj testiranja za osiguranje kvalitete softvera“ koristi na način da gore navedeno autorsko djelo, kao cjeloviti tekst trajno objavi u javnoj internetskoj bazi Sveučilišne knjižnice Sveučilišta Jurja Dobrile u Puli te kopira u javnu internetsku bazu završnih radova Nacionalne i sveučilišne knjižnice (stavljanje na raspolaganje javnosti), sve u skladu s Zakonom o autorskom pravu i drugim srodnim pravima i dobrom akademskom praksom, a radi promicanja otvorenoga, slobodnoga pristupa znanstvenim informacijama.

Za korištenje autorskog djela na gore navedeni način ne potražujem naknadu.

U Puli, 28.09.2022.

Potpis

SADRŽAJ

1. UVOD

1.1. Što je softver?

„Softver je skup uputa, podataka ili programa koji se koriste za upravljanje računalima i izvršavanje određenih zadataka“ (Tech Target). Softver je također termin koji može upućivati na aplikaciju ili skriptu. Ono je skup programa koji se mogu izvoditi na računalu. Može se reći, kako se računala dijele na dva dijela, hardver i softver. Softver se obično dijeli na aplikacijsku i sistemsku varijantu. Softver koji je namijenjen za obavljanje specifičnih zadataka naziva se aplikacijski softver. Dok sistemski softver služi za upravljanje računalnim hardverom te također služi kao podloga za pokretanje aplikacija. Jedna od varijanti softvera je također i softver za programiranje, a dolazi s alatima koji se koriste u programiranju (Tech Target).

1.2. Životni ciklus razvoja softvera

„Životni ciklus razvoja softvera“, ili skraćeno SDLC (Software Developmnet Life Cycle) (slika 1.) je proces kojeg kompanije primjenjuju tijekom razvoja softvera. Cijeli proces SDLC-a se sastoji od ukupno šest faza unutar kojih je potrebno proći kroz određene zadatke.

Prva faza SDLC-a označava „Planiranje i analizu zahtjeva“ za koju se može reći da je ujedno i najznačajnija (BIG WATER CONSULTING). U analizi zahtjeva radi se planiranje studije izvedivosti projekta u svim aspektima realizacije. Osiguranje kvalitete također se izvodi u toj fazi razvoja, a sve s namjerom da projekt bude što kvalitetniji i realiziran uz najmanje rizike.

U drugoj fazi, definiranje zahtjeva, definira se i razrađuje dokumentacija koja će uključivati cijeli ciklus rada na projektu. Tu su uključeni kupci i/ili tržišni analitičari koji će dati zeleno svjetlo projektu.

Potom slijedi faza u kojoj se provodi dizajniranje arhitekture softverskog proizvoda. U toj fazi bira se programski jezik, operativne sisteme koji će se koristiti (npr. Android,

iOS, Linux…), određuju se metode rješavanja problema i izvršavanje zadataka, kao i metode koje su poduzete za osiguranje izvedivosti aplikacije. U ovoj fazi dobiva se i prototip, odnosno rana verzija aplikacije. Također, u toj fazi razvoja mogu se vršiti izmjene u pojedinim segmentima.

Četvrta faza je faza razvoja aplikacije, tj. njezine implementacije. Tijekom ove faze do izražaja dolaze programeri kojima je zadaća isprogramirati aplikaciju. U izradi manjih aplikacija može sudjelovati jedan programer, dok na većim aplikacijama rade timovi programera. U izradi složenijih aplikacija vrlo je važna koordinacija svih timova kao i njihove vještine.

Peta faza je faza testiranja softverskog proizvoda. Ova faza je ključna faza u izradi proizvoda, jer tu se otkrivaju i popravljaju nedostaci i greške te se softver ponovno testira dok se ne dobije tražena kvaliteta. Tražena kvaliteta radi se po specifikaciji softverskih zahtjeva (SRS). Zadovoljstvo korisnika, a time i stopa korištenja aplikacije, mjerilo je kvalitetnog, odnosno uspješnog proizvoda.

Šesta faza je faza održavanja aplikacije. Tijekom ove faze, korisnici otkrivaju greške koje se nisu otkrile u fazi testiranja. U ovoj se fazi rješavaju, odnosno popravljaju nedostaci, što ustvari rezultira novim razvojnim ciklusom softvera.

Slika 1. - Prikaz SDLC-a

Izvor: https://bigwater.consulting/2019/04/08/software-development-life-cycle-sdlc/

2. ULOGA TESTIRANJA U OSIGURANJU KVALITETE SOFTVERA

Proces razvoja softvera obuhvaća osiguranje kvalitete, kontrolu kvalitete, kao i testiranje softvera (slika 2) (AltexSoft).

Slika 2. – Osiguranje kvalitete, kontrola kvalitete i testiranje

Izvor: https://www.altexsoft.com/whitepapers/quality-assurance-quality-control-and-testing-the-basics-of-software-quality-management/

Osiguranje kvalitete softvera je širok pojam koji obuhvaća niz aktivnosti koje su usmjerene u pronalasku grešaka tijekom cijelog procesa razvoja softvera. Dok je kontrola kvalitete usmjerena samo na proizvod. U osiguranju kvalitete sudjeluju timovi koji su specijalizirani za analizu svih faza procesa i koji im je cilj da krajnji proizvod bude kvalitetan i pristupačan svakom korisniku. Osiguranje kvalitete softvera omogućuje kompaniji da dobiva na vremenu, smanji troškove razvoja (odnosno povećava zaradu), osigurava kvalitetu softvera, te time ostvaruje konkurentnost na tržištu. Testiranja u okviru razvojnog procesa fokusiraju se na otkrivanju grešaka tijekom svih faza. U procesu testiranja sudjeluju programeri zajedno s testerima (Future Processing).

Svako zanemarivanje osiguranja kvalitete može biti od velike štete za kompaniju i može bit povezano sa nizom rizika poslovanja. Na 3. slici su prikazani rizici u razvoju softvera kao i poslovni rizici. Iz navedene slike se može vidjeti da neotkrivene greške u programskom kodu, loša realizacija programa i sigurnosti, kao i općenito loše performanse sustava utječu na poslovanje kompanije. Ovdje dolazi do „domino efekta“ gdje se, zbog propuštenih rokova, javljaju nezadovoljni klijenti, kompanija dolazi do financijskih gubitaka, a onda se narušava i poslovni ugled (Future Processing).

Slika 3. - Rizici u nepoštivanju osiguranja kvalitete

Izvor: https://www.future-processing.com/blog/why-is-quality-assurance-important-in-software-development/

3. TESTIRANJE SOFTVERA

3.1. Što je testiranje softvera?

Testiranje softvera je skup sustavno planiranih i provedenih aktivnosti koje su usmjerene k ispitivanju valjanosti softvera obazirajući se pritom na njegove atribute, kao što su njegova skalabilnost, upotrebljivost, pouzdanost, prenosivost te mogućnost njegove ponovne upotrebe. Osim toga, kako bi se utvrdile pogreške i nedostatci u programu, ispituju se funkcionalnosti softvera.

Testiranjem se dobivaju informacije o tome da li značajke softvera ispunjavaju sve zahtjeve klijenata te, isto tako, garantira da je softver valjan.

Također, testiranje je izrazito bitno iz razloga što smanjuje mogućnosti zatajenja samog softvera koji bi korisnike mogao dovesti u neugodnu situaciju. Iz tih razloga, prije isporuke softvera kupcima moraju se provesti testiranja nad softverom da bi im se potom isporučio pouzdani proizvod (JavaTpoint).

3.2. Načela testiranja softvera

Prvo načelo naglašava važnost testiranja softvera jer se na taj način mogu detektirati greške koje su prisutne unutar testiranog softvera. Međutim, koliko god je testiranje softvera bilo kvalitetno odrađeno, to ne može dati garanciju kako unutar softvera više ne postoje greške. Stoga, testiranje softvera služi kako bi se broj grešaka smanjio što bliže nuli.

Zatim, drugo načelo testiranja govori kako u stvarnosti, provođenje iscrpnog testiranja softvera, nije moguće. Drugim riječima nije moguće testirati i predvidjeti sve scenarije korištenja aplikacije. Iz tog razloga je bitno usredotočiti se na scenarije koji se smatraju najbitnijima.

Treće načelo daje naglasak na rano testiranje softvera. Razlog k tome je što greške otkrivene u kasnijim fazama SDLC-a rezultiraju većim ukupnim troškovima razvoja

softvera. Iz tog razloga, može se zaključiti kako raniji početak testiranja softvera rezultira manjim šansama pojave greške u kasnijim fazama razvoja softvera.

Četvrto načelo govori o grupiranju grešaka. Ovo načelo se često povezuje s Paretovim načelom koji glasi kako se 80 posto grešaka krije u svega 20 posto uzoraka. Drugim riječima, prilikom detektiranja kvara u jednom uzorku, potrebno je provesti kvalitetno testiranje nad tim uzorkom zbog mogućeg postojanja drugih grešaka.

Peto načelo ima simboličan naziv „paradoks pesticida“. Unutar ovog načela se poručuje se kako je besmisleno se oslanjati na iste testove tokom razvoja softvera. Razlog k tomu je što stari testovi s vremenom postaju beskorisni u otkrivanju novih grešaka unutar softvera. Stoga je bitno da prilagođavanje testova prati razvoj softvera.

Šesto načelo ističe kako testiranje ovisi o primjeni aplikacije. Drugim riječima, različite aplikacije iziskuju jedinstven pristup u testiranju. Razlog je u tome što svaka aplikacije različito pridaje važnost značajkama aplikacije. Dok je kod jednih aplikacija bitna sigurnost, drugima je brzina.

Posljednja, tj. sedma značajka ističe da je krivo misliti kako nepostojanost grešaka u softveru je garancija za ostvarivanje uspjeha aplikacije. Što znači da se kvaliteta aplikacije ne mjeri samo odsutnošću grešaka, već je bitno staviti naglasak i na ostale karakteristike kao što je naprimjer funkcionalnost aplikacije. Drugim riječima, kvalitetnu aplikaciju gleda se kao na jednu cjelinu (AltexSoft).

3.3. Zašto je testiranje softvera neophodno?

Već se navelo što je testiranje te može se reći kako je izrazito važan proces kojeg se ne smije izostaviti jer u suprotnom moglo bi ostaviti loše posljedice kod korisničkog iskustva kao i kod poslovanja tvrtke. Kako bi se još bolje shvatilo kolika je važnost testiranja, moralo bi se proći kroz važne točke koje mogu još dodatno ukazati koliko je zapravo bitno testiranje softvera.

Kao prvu točku može se navesti sigurnost. Za sigurnost treba se naglasiti kako je to jedan od najbitnijih razloga zašto je bitno testiranje softvera. Testiranjem mogu se otkriti ranjivosti i sigurnosni rizici te uz to je bitno napomenuti kako će korisnici uvijek prije odabrati proizvode koje smatraju pouzdanima (Indium Software).

Druga točka je dugoročna isplativost. Ako se softver počinje kontinuirano testirati u ranoj fazi razvoja, tada se dobiva jeftinije i lakše ispravljanje otkrivenih pogrešaka nego što je to u kasnijim fazama (eDureka).

Također, jedan od razloga zašto je dobro testirati proizvod je kvaliteta proizvoda koju se može ostvariti pomoću testiranja. Testiranje softvera i njegova kvaliteta, može se reći, idu jedan s drugim. Testiranjem dobiva se koliki je broj pogrešaka ili nedostataka u našem proizvodu te se po tome može mjeriti kolike je kvalitete proizvod (The ICE way).

Dalje, također jedan od razloga zašto je poželjno testirati softver jest zadovoljstvo kupaca, što je zapravo i ključni cilj. Zbog toga, kada je proizvod još uvijek u fazi razvoja, tvrtkama je važno testiranje kako bi nakon isporuke proizvoda kupci imali što bolje iskustvo u njegovom korištenju (eDureka).

3.4. Životni ciklus testiranja softvera (STLC)

„Životni ciklus testiranja softvera odnosi se na proces testiranja koji ima određene korake koje treba izvršiti u određenom slijedu kako bi se osiguralo da su ciljevi kvalitete ispunjeni“ (Software Testing Help). U nastavku će se redom navesti svih 6 faza STLC-a (slika 4.) te će ih se također i objasniti.

Prva faza je analiza zahtjeva. U ovoj fazi ispitni tim ima zadaću proučiti zahtjeve klijenta gledajući pritom da li postoji mogućnost provedbe testiranja nad tim zahtjevima. Ukoliko testni tim dođe do zaključka kako nije moguće provesti testiranje nad nekim zahtjevom, tada se testni tim može konzultirati s ostalim dionicima kao što su tehnički voditelj, klijent, itd. (Software Testing Material).

U drugoj fazi, koja je ujedno i najvažnija faza u STLC-u, dolazi se do planiranja testiranja. Tijekom ove faze radi se na kreiranju strategija vezanih za testiranje.

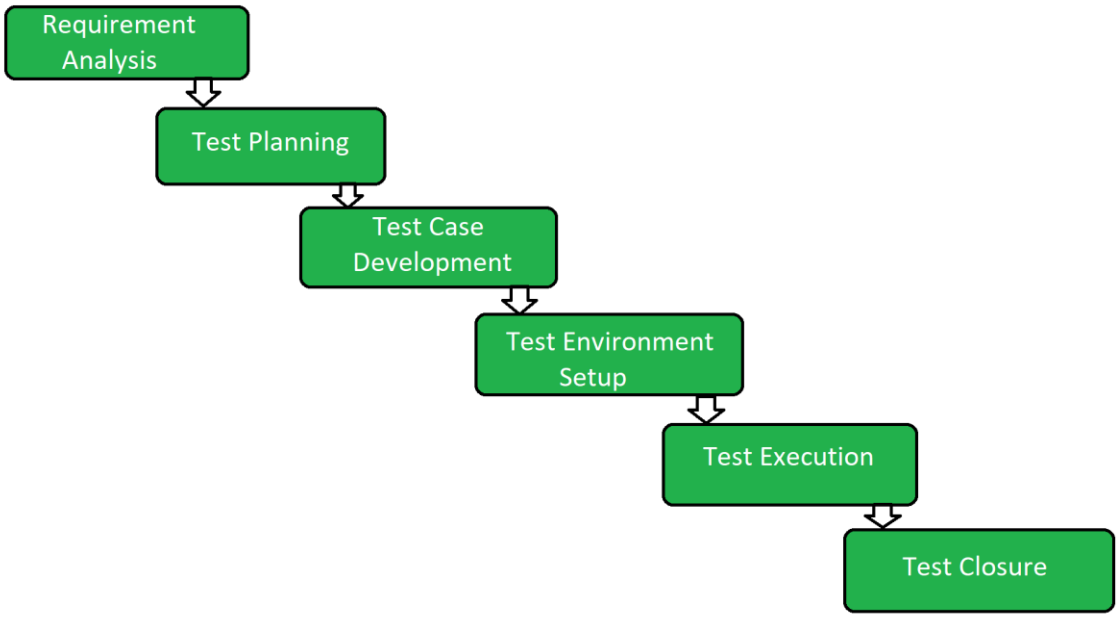
Također, u ovoj je fazi voditelj testiranja zadužen za kalkulaciju troškova kao i za definiranje napora kojeg će biti potrebno uložiti u projekt (eDureka). Isto tako, od aktivnosti koje se u ovoj fazi mogu provodit su dodjeljivanje odgovornosti i zadaća, biranje potrebnog alata (u slučaju automatizacije), itd.

U sljedećoj fazi dolazi se do razvijanja testnih slučajeva. Ova faza dolazi na red tek nakon završetka prethodne faze, tj. nakon završetka planiranja testa (eDureka). Tijekom ove faze testeri su zaduženi za pisanje testnih slučajeva odnosno testnih skripti (u slučaju automatizacije). Tim testera također je zadužen za stvaranje testnih podataka (Software Testing Material). Nakon završetka priprema testnih slučajeva, tim testera dužan ih je predati timu za osiguranje kvalitete (QA) na pregled (Geek for Geeks).

U četvrtoj po redu fazi radi se na postavljanju testnog okruženja što je od velikog značenja u STLC-u (Geeks for Geeks). „Postavljanje testnog okruženja vrši se na temelju popisa hardverskih i softverskih zahtjeva“ (Software Testing Material) . U ovoj se fazi definiraju uvjeti pod kojima će se omogućiti pokretanje testnih slučajeva. U slučaju da tim za razvoj radi na postavljanju testnog okruženja, tada testni tim nije dužan sudjelovati u ovoj fazi. Umjesto toga, testni tim ima zadaću ispitati postavljeno okruženje kako bi bilo moguće utvrditi je li dovoljno spremno za izvođenje testova. Takvo ispitivanje vrši se pomoću testiranja dimom (Guru99).

Nakon završetka postavljanja testnog okruženja, na red dolazi izvođenje testa. Tijekom ove faze testeri imaju zadaću izvođenja testova pridržavajući se pritom prijašnje definiranih planova testiranja, primjenjujući već razvijene testne slučajeve iz prijašnje faze. Prilikom testiranja, u slučaju nailaska na greške u softveru, tim testera dužan je prijaviti otkrivene greške timu za razvoj. Potom, tim za razvoj otklanja greške te vraća ispravljeni softver testerima kako bi ponovo pokrenuli testiranje (Guru99).

„Faza zatvaranja testnog ciklusa je završetak izvođenja koji uključuje nekoliko aktivnosti kao što su izvješćivanja o završetku testa, prikupljanje matrica završetka testa i rezultata testa. Članovi tima za testiranje sastaju se, raspravljaju i analiziraju artefakte testiranja kako bi identificirali strategije koje se moraju implementirati u budućnosti, uzimajući lekcije iz trenutnog ciklusa testiranja.“ (Guru99). Cilj ovoga je spriječiti u procesu nastajanje poteškoća za nadolazeće cikluse testiranja (Guru99).



Slika 4. - Prikaz STLC-a

Link: https://www.geeksforgeeks.org/software-testing-life-cycle-stlc/

4. NAČINI TESTIRANJA SOFTVERA

4.1. Ručno testiranje

U odnosu na primijenjenu tehniku testiranja softvera moglo bi se reći da se izvodi na dva načina, a to su ručno i automatsko testiranje softvera. Ručno je testiranje (slika 5.) najstarija tehnika testiranja koja omogućuje identificiranje opasnih grešaka koje se nalaze u softveru. „Ručno testiranje je vrsta softverskog testiranja u kojem testne slučajeve izvodi tester bez upotrebe automatiziranih alata.“(Guru99)

Bitno je za napomenuti da kod novih aplikacija potrebno je prvo izvesti ručno testiranje kako bi se kasnije moglo preći na automatizirano testiranje. Kod ručnog testiranja potrebno je uložiti više truda, ali je također ručno testiranje i jako važno jer to je nezaobilazan način za ispitivanje mogućnosti realiziranja automatizacije (Guru99). Također, u slučajevima kada je potrebno ponavljati ručno testiranje nad velikim aplikacijama provođenje testiranja može postati naporno (Sharma, 2014, str. 252.).

U procesu ručnog testiranja softvera prvo je potrebno proučiti testne osnove. „Testna osnova su informacije na kojima se temelje testni slučajevima, kao što su zahtjevi, specifikacije dizajna, analiza rizika proizvoda, arhitektura i sučelja.“ (Medium). Nakon toga testeri prelaze na planiranje testiranja da bi kasnije krenuli na kreiranje testnih slučajeva prema zahtjevima kupaca (JavaTpoint). Potom dolazi na red izvršenje testnih slučajeva u kojem je obuhvaćeno izvođenje testiranja koje se može izvesti ručno ili pomoću automatiziranih alata. Zatim slijedi proces tijekom kojeg se pronalaze greške testiranjem ili evidentiranjem pritužba korisnika aplikacije (Medium). Tim zadužen za testiranje, u slučaju pronalaska greške, razvojnom timu šalju informacije o greškama. Nakon što su ispravili sve evidentirane greške, razvojni tim softver šalje natrag testerima kako bi se ponovilo testiranje te utvrdilo postoji li grešaka u softveru (JavaTpoint).

Slika 5. - Prikaz procesa ručnog testiranja

Procedura ručnog testiranja softvera Link: https://www.esds.co.in/blog/manual-testing-process-lifecycle/

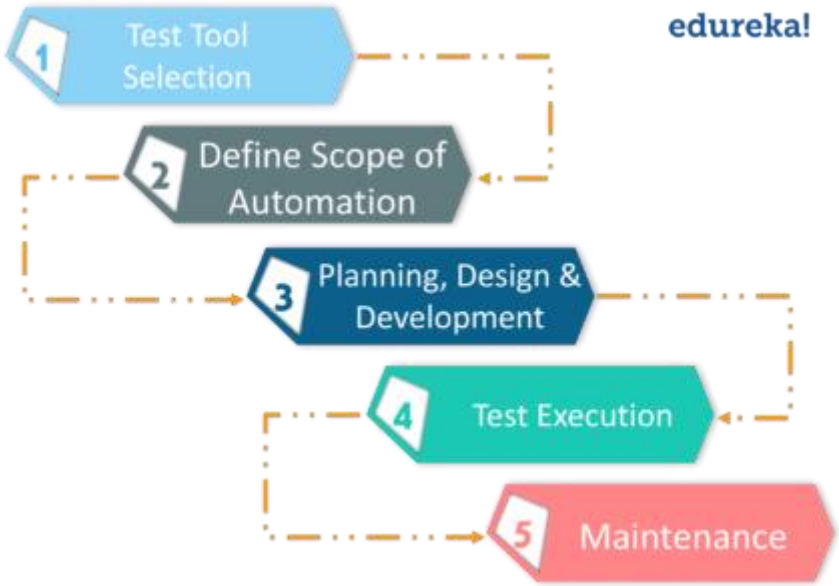
Kod ručnog testiranja moglo bi se reći kako su njegove prednosti to što prilikom testiranja tester može upravljati aplikacijom na način kao što bi njome upravljali njeni pravi korisnici. To znači da bi tester bio u mogućnosti otkriti probleme koji se kriju u korisničkom sučelju kao i kod korištenja same aplikacije (JavaTpoint). Za razliku od automatiziranog testiranja, u ručnom testiranju testerima nije potrebno imati znanje iz programiranja, već je potrebno imati iskustvo u obavljanju ovakvog posla (LITSLINK). Također, ručno testiranje je pogodno za testiranje malih promjena u softveru iz razloga što je moguće testiranje softvera u hodu, dok bi kod automatizacije prilikom testiranja neke promjene bilo potrebno kodiranje (KRONE). Uz to , ručno testiranje je kratkoročno jeftinija opcija za razliku od automatiziranog testiranja iz razloga što nije potrebna kupnja skupocjenih alata za automatizaciju (Base36).

Kod nedostataka u ručnom testiranju, jedan od prvih može biti manja pouzdanost. Razlog k tome je što ručno testiranje obavljaju ljudi, pa samim time ponekad može doći do ljudske pogreške, neuočavanje grešaka te manje preciznosti u testiranju softvera (Sharma, 2014, str 252.). Isto tako, kada su u pitanju veliki softverski sustavi, tada bi bilo potrebno zapošljavanje velikog broja testera. Zbog toga, u

takvim situacijama, financijski prihvatljivija bila bi automatizacija minimalno jednog dijela testova koji bi se izvodili (UTOR). Ručna testiranja također iziskuju više vremena za razliku od automatiziranih testiranja kod kojih je, između ostalog, njihovo izvršavanje moguće i bez nadziranja (Uilicious).

4.2. Automatizirano testiranje

Automatizirano testiranje je metoda prilikom koje se vrši testiranje softvera kako bi se mogla potvrditi njegova ispravnost. „To se može postići pisanjem testnih skripti ili korištenjem bilo kojeg alata za automatizirano testiranje“ (Software Testing Help). Prije samog početka automatiziranog testiranja, bitno je odabrati alat. Pri odabiru alata kojim će se obavljati testiranje potrebno je voditi se po tome na kojoj se tehnologiji, aplikacija koju će se testirati, bazira (Guru99). Daljnji korak koji slijedi je da se definira opseg automatizacije (slika 6.). Neke od točaka koje pomažu u određivanju opsega mogu biti kompleksnost testnih slučajeva te isto tako i značajke koje su ključne za poslovanje kao i tehnička izvedivost. Nadalje, slijedi faza u kojoj se pripremaju plan te također i strategija automatizacije. Plan automatizacije može sadržati detalje kao što su alat za automatizaciju koji se koristiti, vremenski period za provođenje testiranja i drugo. Četvrti korak je izvršavanje testa. U ovom se koraku, nakon izvršenja testa, dobiva izvješće koje sadrži informacije o testu koji se izvršio (Katalon). U petom, tj. zadnjem koraku slijedi faza održavanja. U toj fazi provode se testiranja kojima je cilj provjeriti funkcioniraju li nove značajke koje su ugrađene u softveru (Guru99). „Održavanje u automatiziranom testiranju provodi se kada se dodaju nove skripte za automatizaciju i potrebno ih je pregledati i održavati kako bi se poboljšala učinkovitost skripti za automatizaciju sa svakim sljedećim ciklusom izdanja.“ (Guru99).



Slika 6. - Prikaz procesa automatiziranog testiranja

Proces automatiziranog testiranja. Link: https://medium.com/edureka/automation-testing-tutorial-157d269e60db

Kao prvu prednost kod automatsko testiranja može se navesti to da takvo testiranje daje mogućnost pronalaska više grešaka nego što je to moguće kod ručnog testiranja. Uz to, za razliku od ručnog testiranja kojeg provode testeri, kod automatskog ispitivanja koriste se softverski alati. Drugim riječima softverski alati ne mogu se umoriti kao što je to slučaj kod ručnog testiranja s testerima, što znači da se automatizirano testiranje može izvoditi danonoćno. Kod automatiziranog testiranja, softverski alati također omogućuju brže izvršavanje testova u usporedbi s ručnim testiranjem, što u konačnici može pridonijeti bržem razvoju proizvoda I njegovog dolaska na tržište (Guru99).

Dok pod nedostatke može se navesti to što bez prisustva ljudi u testiranju aplikacije, teško je dobiti povratne informacije vezane za kvalitetu korisničkog sučelja. Takve informacije se odnose na vizualne elemente kao što su dimenzije gumba, font, itd. Isto tako, kao jednu od mana automatskog testiranja, može se navesti cijena samog alata za automatizirano testiranje koja može biti visoka. Još jedan od nedostataka ovakvog testiranja je i to što svaki alat dolazi sa svojim određenim limitacijama te se time dobiva umanjeni opseg automatizacije (Guru99).

5. METODOLOGIJE TESTIRANJA SOFTVERA

5.1. Tipovi testiranja softvera

5.1.1. Funkcionalno testiranje

Funkcionalno testiranje je testiranje koje uglavnom predstavlja testiranje crne kutije, što znači da ga ne zanima izvorni kod softvera kojeg se testira. Prilikom ovog testiranja zadaća je usporediti aplikaciju s funkcionalnim zahtjevima. Ovakvo testiranje se provodi na način da se značajke ispituju unoseći traženi ulaz da bi potom aplikacija vratila izlaz kojeg bi se usporedilo s funkcionalnim zahtjevima. „Ovo testiranje provjerava korisničko sučelje, API-je, bazu podataka, sigurnost, komunikaciju između klijenta i poslužitelja i druge funkcije aplikacije koja se testira.“(Guru99).

5.1.2. Nefunkcionalno testiranje

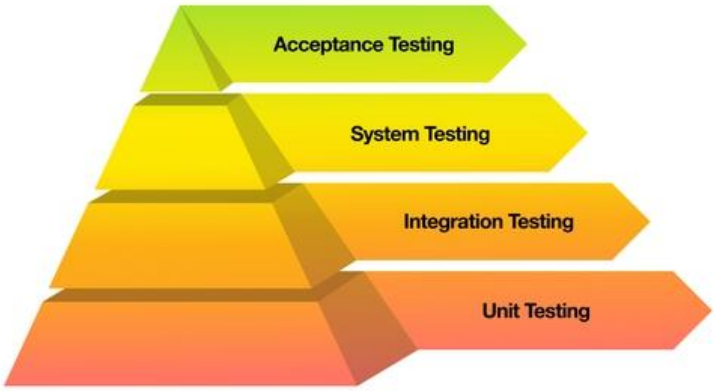
Nefunkcionalno testiranje je proces testiranja atributa, odnosno karakteristika sustava, za razliku od prethodne vrste testiranja koji se temelji na testiranju funkcionalnosti aplikacije. Nefunkcionalno testiranje jednako je važno kao i funkcionalno testiranje, jer se preko takvog testiranja sustav bolje prilagođava krajnjem korisniku (ArtOfTesting).

Nefunkcionalno testiranje može se podijeliti na više vrsta, jedna od njih je i testiranje opterećenja. U ovom testiranju cilj je testirati spremnost sustava na velika opterećenja od više korisnika istodobno. Rezultati će pokazati da li je sustav spreman na takva opterećenja ili nije. Druga vrsta nefunkcionalnog testiranja je testiranje sigurnosti, koje je ujedno i neizostavna. U ovom procesu provodi se ispitivanje neprobojnosti sustava od mogućih vanjskih napada od strane hakera. Treća vrsta nefunkcionalnog testiranja je testiranje upotrebljivosti. Tijekom ovog procesa vrši se ispitivanje sustava do koje mjere je prilagođen korisniku. Četvrta vrsta nefunkcionalnog testiranja je testiranje kompatibilnosti čiji je cilj ispitivanje rada

sustava na različitim platformama. Peta vrsta nefunkcionalnog testiranja je testiranje oporavka. Svrha provođenja ovog testiranja je ispitivanje spremnosti sustava u njegovom oporavku nakon pojave kvara. Šesta vrsta nefunkcionalnog testiranja je testiranje performansi. Ovim se testiranju utvrđuje kolika je spremnost, odnosno reakcija sustava prema zadanom (ArtOfTesting).

5.2. Razine testiranja softvera

Moglo bi se reći kako testiranje softvera se može podijeliti na četiri različite vrste, a to su ispitivanje modula, integracijsko testiranje, sistemsko testiranje te također i testiranje prihvatljivosti (slika 7.).



Slika 7. - Prikaz razina testiranja softvera

Izvor: https://www.exposit.com/blog/4-levels-software-testing-how-develop-reliable-product/

5.2.1. Ispitivanje modula

Ispitivanje modula ili na engleskom jeziku „unit testing“, je testiranja softvera za vrijeme njegova razvoja (GeeksforGeeks). To je testiranje softvera, od strane programera, prilikom kojeg se pojedinačno vrši ispitivanje nad svakom softverskom

komponentom (modulom). Izoliranjem modula od ostatka programskog koda moguće je ispitati njegovu funkcionalnu ispravnost (tutorialspoint).

Prednosti ove vrste testiranja je što se primjenjuje modularni pristup, što znači da je moguće istovremeno testiranje više pojedinih dijelova programskog koda (JavaTPoint). Uz to, još jednu od prednosti koju pruža ova vrsta testiranja je to što „čine sigurnijim i lakšim refaktoriranje koda postavljanjem testova koji osiguravaju da se refaktoriranje odvija bez problema i smetnji.“ (The QA Lead). Isto tako, bitno je napomenuti kako ispitivanje modula daje mogućnost otkrivanja grešaka već u ranim fazama razvoja softvera, čime se dobiva na vremenu, te isto tako i na smanjenju troškova (hackr.io).

Jedna od mana ispitivanja modula je što nije korisno u otkrivanju nedostataka korisničkog sučelja. Uz to, potrebno je i spomenuti kako ovom vrstom ispitivanja nije moguće pronaći sve nedostatke softvera (The QA Lead).

5.2.2. Integracijsko testiranje

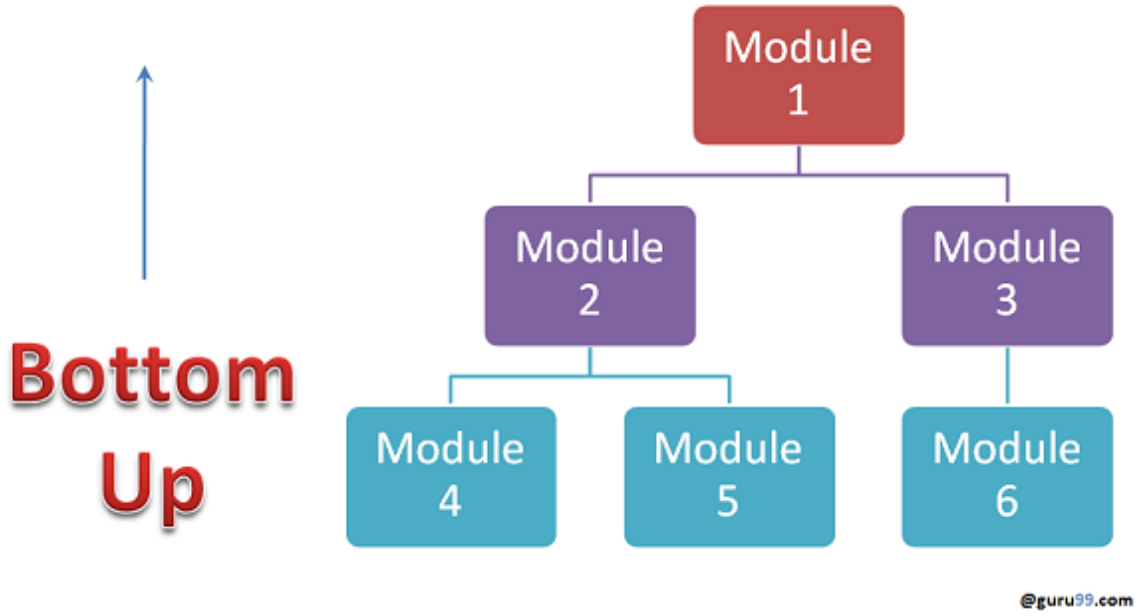
„Integracijsko testiranje je faza u testiranju softvera u kojoj se pojedinačni softverski moduli kombiniraju i testiraju kao grupa.“ (Mahfuz, 2016, str. 66). Cilj integracijskog testiranja je ispitivanje veze između grupiranih modula, tj. zadatak je pronaći greške u komunikaciji unutar grupe modula (JavaTPoint). Prije prelaza na integracijsko testiranje, potrebno je provesti ispitivanje nad svakim modulom. Moduli su isprogramirani od strane programera, dok je izvođenje integracijskih testiranja zadaća testera (Guru99).

Kod integracijskog testiranja može se reći da postoje četiri vrste pristupa koji će se navesti te pritom i objasniti u nastavku.

Jedan od tih četiri pristupa je „Big Bang“ integracija. Koristeći se ovakvim pristupom testiranja, kombiniraju se sve softverske jedinice u jedan skup na kojeg se prilikom testiranja gleda kao na jednu cjelinu. Uz to, također je bitno naglasiti kako u situacijama kada u toj cjelini postoje nedovršene jedinice, nije moguće provesti postupak integracije. Prednost Big Bang testiranja je to što se smatra prikladnim za sustave manjih veličina. Dok mana Big Bang testiranja može biti to što je kod

ovakvog pristupa integracijskog testiranja teško utvrditi gdje se nalazi greška. Isto tako, kompleksniji softveri mogu pogodovati tome da se pojedina sučelja nenamjerno izostave prilikom testiranja. Još jedna od mana ove vrste pristupa može biti i to što preostalo vrijeme za testiranje može biti znatno skraćeno iz razloga što je prioritet dovršenje svih modula (Guru99).

Sljedeći pristup o kojem će se pisati je „integracijsko testiranje odozdo prema gore“ (Guru99). Ovo testiranje spada pod inkrementalna testiranja što ustvari znači da „testiranje se provodi integracijom dva ili više modula koji su međusobno logički povezani, a zatim se testira ispravnost rada aplikacije.“ (Guru99) Riječ je o postepenom dodavanju, tj. integriranju modula prilikom čega je svaki put potrebno izvršiti testiranje nad kombiniranim modulima kao nad cjelinom (Software Testing Help). Ovaj proces ne smije se prestati ponavljati sve „dok se svi logički povezani moduli ne integriraju i uspješno testiraju.“ (Guru99). U procesu testiranja odozdo prema gore pravilo je početi s testiranjem od modela koji pripadaju najnižoj razini (Slika 8.) kako bi se potom postepeno dodavali i testirali moduli većih razina (Software Testing Help). „Za modul se kaže da je na najnižoj razini ako ne poziva drugi modul.“ (Naik i Tripathy, 2008.)



Slika 8. – Simbolični prikaz testiranja „odozdo prema gore“

Izvor: https://www.guru99.com/integration-testing.html

Treći pristup u integracijskog testiranja je pristup u kojem se vrši integracijsko testiranje modula najviše razine prema modulima nižih razina, te iz tog razloga ovakav pristup u integracijskom testiranju zove se „odozgo prema dolje“. Prednost ovakvog integracijskog testiranja je to što je lakše moguće pronaći mjesto nastanka greške, kao i dobivanje uvjeta za ranog prototipa aplikacije. Dok nedostatak ovakvog pristupa može biti to što niži moduli nisu dobro ispitani (Guru99).

Zadnji od četiri pristupa integracijskog testiranja je takozvano „sendvič testiranje“ (engl. Sandwich Testing). „To je kombinacija pristupa odozgo prema dolje I odozdo prema gore, stoga se naziva testiranje hibridne integracije“ (Guru99). To zapravo znači da se u ovom pristupu, moduli pri vrhu testiraju zajedno s nižim modulima. Uz to, u ovom se pristupu također moduli viših razina integriraju s modulima koji pripadaju nižim razinama da bi ih se moglo testirati zajedno u sustavu (Guru99). Ovakav pristup može biti prikladan za testiranje integracije kod velikih programa koji se dijele na više pod programa (Software Testing Help).

5.2.3. Sistemsko testiranje

Gledajući razine testiranja, sistemsko testiranje dolazi kao treća po redu razina testiranja, tj. ova razina testiranja dolazi nakon testiranja integracije. Sistemsko testiranje je postupak tijekom kojeg tim koji zadužen za QA, dužan je testirati međusobnu funkcionalnost svih komponenti unutar jedne aplikacije. Svrha testiranja sustava je provjeriti obavlja li softver zadatke onako kako je bio zamišljen da ih izvršava. Sistemsko se testiranje može smatrati kao testiranjem crne kutije, te je kao takvo usmjereno na testiranje funkcionalnosti softvera. Ovakvim testiranjem QA tim ocjenjuje softver gledajući pritom na sve zahtjeve koji je potrebno zadovoljiti. Kako bi u tome uspio, QA tim koristi različite vrste testiranja pod koje spadaju: testiranje performansi softvera, njegove upotrebljivosti, itd. Uz to, zadaća QA tim-u tijekom ovog procesa je napraviti procjenu o tome da li testni slučaj uključuje sve temeljne zahtjeve softvera kao i korisničku priču (engl. user story). Služeći se definiranim testnim slučajevima, QA timu je moguće (prije njegova izdavanja) uočiti sve velike

postojeće nedostatke u softveru koji ga sprječavaju u obavljanju svojih funkcija. Time QA tim, tijekom ovog procesa ima priliku evidentirati sve otkrivene nedostatke za svaki zahtjev posebno. Tijekom ove razine testiranja, potrebno je izvršiti ispitivanje nad svim dijelovima softvera kako bi se uspjelo utvrditi njihovu funkcionalnost unutar jedne cjeline. U slučaju da verzija softvera koju ispituju uspije ostvariti sve definirane zahtjeve, slijedi testiranje prihvatljivosti, nakon čega slijedi izdavanje aplikacije (TechTarget).

5.2.4. Testiranje prihvatljivosti

Testovi prihvatvljivosti su testiranja koja provode kupci s ciljem utvrđivanja kvalitete sustava. „Testovi prihvaćanja predstavljaju interese kupaca“ (Miller i Collins, 2001.). Također, ovom razinom testiranja kupci mogu biti sigurni kako funkcioniraju sve značajke te da sustav radi na način na koji je zamišljen. Kada sustav uspije zadovoljiti na svim testovima prihvaćanja, tada se testiranje softvera smatra završenim. Testovi prihvaćanja su bitni za developere softvera iz više razloga, a prvi razlog je taj što ovakva testiranja developerima daju povratnu informaciju o „zrelosti“ sustava. Drugi razlog je taj da se testovima prihvaćanja mogu otkriti svi nedostaci koji nisu bili vidljivi u procesu ispitivanja modula. Na kraju, treći je razlog taj test prihvaćanja developerima pruža informacije o tome koji su korisnički zahtjevi kao i u kojoj je mjeri sustav uspio ispuniti zahtjeve korisnika (Miller i Collins, 2001).

5.3. Metode testiranja

5.3.1. Testiranje crne kutije

Jedna od metoda testiranja je testiranje crne kutije koje se temelji na ulazu i izlazu nekog programa. Testiranje crne kutije je dobilo naziv po tome što aplikaciju koju je potrebno testirati gleda se kao na crnu kutiju, drugim riječima, nepotrebno je poznavanje unutarnje strukture programa. Prilikom ovog testiranja, fokus se stavlja na rad samog programa kako bi se moglo uočiti bilo kakve nepravilnosti u njegovom djelovanju (Myers, 2004, str. 9). Program da bi bio ispravan, mora prilikom danog ulaza vratiti željeni rezultat. Uloga QA tima je da u ovom procesu testiraju program dajući mu određen ulaz, kako bi kasnije mogli usporediti vraćeni izlaz s željenim izlazom, tj. rezultatom (Mahfuz, 2016, str. 65).

Prednost testiranja crne kutije je to što je ova metoda vrlo pogodna za velike projekte. Druga prednost je ta da testni tim ne mora poznavati programski jezik koji je korišten u razvoju programa, kao i o detaljima implementacije. Kod ove metode također je moguće definirati testne slučaje odmah po završetku specificiranja funkcionalnih zahtjeva. Drugim riječima, nije potrebno čekanje završetak razvoja aplikacije da bi se testni slučajevi mogli pripremiti. Isto tako, još jedna od prednosti ove metode jest to što se testiranje programa izvodi iz perspektive budućih korisnika (I Answer 4 U).

Jedan od nedostataka ove metode je taj što proces testiranja ne obuhvaća sve značajke kojima program raspolaže. Uz to, kao nedostatak može se navesti teže otkrivanje uzroka nastalih greški prilikom testiranja samog programa. Također jedan od nedostataka jest da testni slučajevi mogu biti manjkavi, što znači da rezultati testiranja (u tom slučaju) ne prikazuju pravu sliku o stanju softvera (I Answer 4 U). Isto tako, nedovoljno dobro definirane specifikacije mogu rezultirati težim definiranjem odgovarajućih testnih slučajeva (Invensis).

5.3.2. Testiranje bijele kutije

Druga metoda testiranja je testiranje bijele kutije. „Testiranje bijele kutije je tehnika softvera koja se temelji na unutarnjoj strukturi koda aplikacije“ (Software Testing Material). Kako bi se testni slučajevi mogli definirati, tester mora poznavati unutarnje dijelove softvera te također mora posjedovati vještine programiranja. Tijekom ovog procesa testeri, za razliku od testera kod testiranja crne kutije, imaju pristup programskom kodu (Software Testing Material). U početku ovog procesa je potrebno je da se tester upozna s programskim kodom softvera kojeg će testirati, za što će mu biti potrebno razumijevanje programskog jezika korištenog u projektu. Uz to, tester mora biti dobro upoznat s filozofijom sigurnog kodiranja iz razloga što je sigurnost kod testiranja softvera na prvome mjestu. Testeru je dužnost pronaći sve sigurnosne rupe kako bi se moglo izbjeći proboj hakera ili zlonamjernog softvera koji bi bio ubačen u sustav od strane korisnika. U drugom koraku ove metode potrebno je testirati programski kod softvera kako bi se osigurao odgovarajući protok podataka kao i kvalitetnu strukturu softvera (Guru99).

Prednost ove metode je što testeri mogu krenuti s testiranjem aplikacije prije nego što je njeno korisničko sučelje kompletirano. Testiranja bijele kutije također pomaže testerima riješiti se linija programskog koda koje su višak te pritom uspijeva locirati neopažene greške unutar samog koda. Uz to, metodom testiranja bijele kutije postiže se veća temeljitost nego što je to u slučaju prijašnje metode (I Answer 4 U). Dok nedostatak je što su veliki troškovi testiranja, te je provođenje ovakvog testiranje samo po sebi kompleksno. Samim time, kada je riječ o velikim projektima, testiranje bijele kutije postaje dugotrajno (Software Testing Material).

5.3.3. Testiranje sive kutije

Treća metoda testiranja je testiranje sive kutije te je hibrid prijašnjih dviju metoda (slika 9.). Ova metode se od prijašnjih metoda razlikuje po tome što testeri imaju nepotpuno znanje o mehanizmu softvera nad kojim provode testiranje (Software Testing Help). „Svrha testiranja sive kutije je pretraživanje i prepoznavanje nedostataka zbog nepravilne strukture koda i nepravilnog korištenja aplikacije“ (Guru99).

Prednost ove metode je što uključuje dobre strane prethodnih dviju metoda. Za razliku od testiranja crne kutije, kod ove tehnike testeri imaju više uvida u mehanizma softvera nad kojim vrše testiranje. Što im na kraju daje veću vjerojatnost za kreiranje boljih testnih scenarija. Tijekom ove metode testeri mogu identificirati problem kojeg su programeri propustili tokom procesa testiranja modula. Također, testerima je moguće izvesti testiranje bez prijašnjeg velikog iskustva u programiranju (ArtOfTesting).

Nedostatak testiranja sive kutije je što testeri nemaju mogućnost uvida u izvorni kod, a to može rezultirati izostavljanjem važnih, odnosno bitnih grešaka. Nedostatak je i što može doći do toga da tester provodi testiranje nad istim dijelovima koje je već testirao programer, pa to isto testiranje postaje suvišno (ArtOfTesting).

Slika 9. - Simboličan prikaz testiranja sive kutije

Izvor: https://www.geeksforgeeks.org/gray-box-testing-software-testing/

6. ZAKLJUČAK

Kvaliteta softvera predstavlja imperativ za svaku kompaniju. Razlog k tomu je što kvalitetan softver „znači“ zadovoljan korisnik, a zadovoljstvo korisnika pomaže kompaniji na putu do njenog uspjeha. Da bi kompanija mogla ponuditi kvalitetan softver, mora imati dobro razrađeni proces razvoja softvera. U procesu razvoja postoje faze kojih se treba pridržavati da bi se dobio što kvalitetniji i bolji proizvod. Jedna od najznačajnijih faza u razvoju softvera je testiranje u osiguranju kvalitete. Kompanije koje se nisu pridržavale načela važnosti osiguranja kvalitete, doživjele su velike financijske gubitke. Samo osiguranje kvalitete velik je i značajan proces kojim se smanjuju troškovi razvoja, štedi vrijeme, pruža sigurnost, ostvaruje zadovoljstvo kod kupaca te na kraju dobiva na profitu i ugledu kompanije.

7. SLIKE

8. LITERATURA

TechTarget (2021.) software. TechTarget. [Online]. Dostupno na: https://www.techtarget.com/searchapparchitecture/definition/software

[Pristupljeno: 11.8.2022.]

JavaTpoint (2022.) Software Testing Tutorial. JavaTpoint. [Online]. Dostupno na: https://www.javatpoint.com/software-testing-tutorial

[Pristupljeno: 11.8.2022.]

Indium Software (2021.) 7 Reasons Why Software Testing is Important. Indium Software. [Online]. Dostupno na: https://www.indiumsoftware.com/blog/why-software-testing/

[Pristupljeno: 12.8.2022.]

Edureka! (2020.) Types of Software Testing : All You Need to Know About Testing Types. Edureka!. [Online]. Dostupno na: https://www.edureka.co/blog/types-of-software-testing/#importanceofsoftwaretesting

[Pristupljeno: 12.8.2022.]

The ICE way (2021.) Why is software testing so important?. The ICE way. [Online]. Dostupno na: https://www.theiceway.com/blog/why-is-software-testing-so-important

[Pristupljeno: 12.8.2022.]

Edureka!(2020.) Importance of software testing. edureka!. [Online]. Dostupno na: https://www.edureka.co/blog/types-of-software-testing/#importanceofsoftwaretesting

[Pristupljeno: 12.8.2022.]

Guru99!(2022.) Manual Testing Tutorial: What is, Types, Concepts. Guru99. [Online]. Dostupno na: https://www.guru99.com/manual-testing.html

[Pristupljeno: 13.8.2022.]

Sharma, R. M. (2014), Software Quality Assurance; Integrating Testing, Security, and Audit. [Online]. Dostupno na: https://www.imvcportal.com.au/uploads/study_material/1504593504IJEIT14122014007$%23@_46.pdf

[Pristupljeno: 13.8.2022.]

Medium(2018.) Manual testing process. Medium. [Online]. Dostupno na: https://medium.com/oceanize-geeks/manual-testing-process-340173d40141

[Pristupljeno: 13.8.2022.]

JavaTpoint(2022.) Manual Testing. JavaTpoint. [Online]. Dostupno na: https://www.javatpoint.com/manual-testing

[Pristupljeno: 13.8.2022.]

LITSLINK(2020.) Manual vs Automated Testing: Pros and Cons. LITSLINK. [Online]. Dostupno na: https://litslink.com/blog/manual-vs-automated-testing-pros-and-cons

[Pristupljeno: 14.8.2022.]

KRONE(2022.) Tomas's Post: Manual Testing vs. Test Automation. Pros and Cons – quick overview. KRONE. [Online]. Dostupno na: https://kroneit.com/tomass-post-manual-testing-vs-test-automation-pros-and-cons-quick-overview/

[Pristupljeno: 14.8.2022.]

Base36(2022.) Automated vs. Manual Testing: The Pros and Cons of Each. Base36. [Online]. Dostupno na: http://www.base36.com/2013/03/automated-vs-manual-testing-the-pros-and-cons-of-each/

[Pristupljeno: 16.8.2022.]

UTOR (2020.) Manual vs Automation Testing: What to Choose For Your Project. UTOR. [Online]. Dostupno na: https://u-tor.com/topic/manual-vs-automation

[Pristupljeno: 17.8.2022.]

Uilicious (2022.) Pros and Cons of Manual Testing. Uilicious. [Online]. Dostupno na: https://uilicious.com/blog/pros-and-cons-manual-testing/

[Pristupljeno: 17.8.2022.]

Software Testing Help (2022.) What Is Automation Testing (Ultimate Guide To Start Test Automation). Software Testing Help. [Online]. Dostupno na: https://www.softwaretestinghelp.com/automation-testing-tutorial-1/

[Pristupljeno: 18.8.2022.]

Guru99 (2022.) What is Automation Testing? Test Tutorial. Guru99. [Online]. Dostupno na: https://www.guru99.com/automation-testing.html

[Pristupljeno: 18.8.2022.]

Katalon (2022.) Automation Testing 101: What, Why and How. Katalon. [Online]. Dostupno na: https://katalon.com/resources-center/blog/what-is-automation-testing

[Pristupljeno: 18.8.2022.]

Guru99 (2022.) Difference Between Manual and Automation Testing. Guru99. [Online]. Dostupno na: https://www.guru99.com/difference-automated-vs-manual-testing.html

[Pristupljeno: 18.8.2022.]

Software Testing Help (2022.) What Is Software Testing Life Cycle (STLC)?. Software Testing Help. [Online]. Dostupno na: https://www.softwaretestinghelp.com/what-is-software-testing-life-cycle-stlc/

[Pristupljeno: 19.8.2022.]

Software Testing Material (2022.) What is Software Testing Life Cycle (STLC) & STLC Phases. Software Testing Material. [Online]. Dostupno na: https://www.softwaretestingmaterial.com/stlc-software-testing-life-cycle/

[Pristupljeno: 19.8.2022.]

Guru99 (2022.) STLC (Software Testing Life Cycle) Phases, Entry, Exit Criteria. Guru99. [Online]. Dostupno na: https://www.guru99.com/software-testing-life-cycle.html

[Pristupljeno: 19.8.2022.]

Edureka! (2020.) Software Testing Life Cycle – Different Stages of Testing .edureka!. [Online]. Dostupno na: https://www.edureka.co/blog/software-testing-life-cycle/

[Pristupljeno: 19.8.2022.]

GeeksforGeeks (2019.) Software Testing Life Cycle (STLC). GeeksforGeeks. [Online]. Dostupno na: https://www.geeksforgeeks.org/software-testing-life-cycle-stlc/

[Pristupljeno: 20.8.2022.]

Guru99 (2022.) What is Functional Testing? Types & Examples. Guru99. [Online]. Dostupno na: https://www.guru99.com/functional-testing.html

[Pristupljeno: 20.8.2022.]

tutorialspoint (2022.) Unit Testing. tutorialspoint. [Online]. Dostupno na: https://www.tutorialspoint.com/software_testing_dictionary/unit_testing.htm

[Pristupljeno: 21.8.2022.]

GeeksforGeeks (2022.) Unit Testing | Software Testing. GeeksforGeeks. [Online]. Dostupno na: https://www.geeksforgeeks.org/unit-testing-software-testing/

[Pristupljeno: 21.8.2022.]

JavaTpoint (2022.) Unit Testing. JavaTpoint. [Online]. Dostupno na: https://www.javatpoint.com/unit-testing

[Pristupljeno: 21.8.2022.]

The QA Lead (2022.) Unit Testing: Advantages & Disadvantages. TheQALead. [Online]. Dostupno na: https://theqalead.com/topics/unit-testing/

[Pristupljeno: 22.8.2022.]

Hackr.io (2022.) What is Unit Testing? Types, Pros, Cons and Best Tools

. hackr.io. [Online]. Dostupno na: https://hackr.io/blog/what-is-unit-testing

[Pristupljeno: 22.8.2022.]

Guru99 (2022.) Integration Testing: What is, Types with Example. Guru99. [Online]. Dostupno na: https://www.guru99.com/integration-testing.html

[Pristupljeno: 24.8.2022.]

Mahfuz, A. S. (2016), Software Quality Assurance; Integrating Testing, Security, and Audit

JavaTPoint (2022.) Levels of Testing. JavaTpoint. [Online]. Dostupno na: https://www.javatpoint.com/levels-of-testing

[Pristupljeno: 24.8.2022.]

Guru99 (2022.) Integration Testing: What is, Types with Example. Guru99. [Online]. Dostupno na: https://www.guru99.com/integration-testing.html

[Pristupljeno: 25.8.2022.]

Software Testing Help (2022.) What Is Integration Testing (Tutorial With Integration Testing Example). Software Testing Help. [Online]. Dostupno na: https://www.softwaretestinghelp.com/what-is-integration-testing/

[Pristupljeno: 25.8.2022.]

Naik, K., Tripathy P. (2008), Software Testing and Quality Assurance : Theory and Practice

Software Testing Help (2022.) What Is Incremental Testing: Detailed Explanation With Examples. Software Testing Help. [Online]. Dostupno na: https://www.softwaretestinghelp.com/incremental-testing/

[Pristupljeno: 26.8.2022.]

TechTarget (2018.) system testing. TechTarget. [Online]. Dostupno na: https://www.techtarget.com/searchsoftwarequality/definition/system-testing

[Pristupljeno: 26.8.2022.]

Miller, R., Collins, C. (2001), Acceptance Testing, Link: http://www.dsc.ufcg.edu.br/~jacques/cursos/map/recursos/Testing05.pdf

[Pristupljeno: 28.8.2022.]

Myers, G. (2004), The Art of Software Testing (Second Edition)

I Answer 4 U (2022.) Advantages and Disadvantages of Black-box testing. I Answer 4 U. [Online]. Dostupno na: https://www.ianswer4u.com/2016/11/advantages-and-disadvantages-of-black.html

[Pristupljeno: 28.8.2022.]

Guru99 (2022.) Levels of Testing in Software Testing. Guru99. [Online]. Dostupno na: https://www.guru99.com/levels-of-testing.html

[Pristupljeno: 3.9.2022.]

Invesis (2015.) What is Black Box Testing: Advantages and Disadvantages. Invesis. [Online]. Dostupno na: https://www.invensis.net/blog/black-box-testing-advantages-disadvantages/

[Pristupljeno: 4.9.2022.]

Software Testing Material (2022.) What is White Box Testing and its Types with Examples?. [Online]. Dostupno na: https://www.softwaretestingmaterial.com/white-box-testing/

[Pristupljeno: 7.9.2022.]

I Answer 4 U (2022.) Advantages and Disadvantages of White box testing. I Answer 4 U. [Online]. Dostupno na: https://www.ianswer4u.com/2016/11/advantages-and-disadvantages-of-white.html

[Pristupljeno: 7.9.2022.]

AltexSoft (2022.) Quality Assurance, Quality Control and Testing — the Basics of Software Quality Management. AltexSoft. [Online]. Dostupno na: https://www.altexsoft.com/whitepapers/quality-assurance-quality-control-and-testing-the-basics-of-software-quality-management/

[Pristupljeno: 10.9.2022.]

TechTarget (2019.) quality assurance (QA). TechTarget. [Online]. Dostupno na: https://www.techtarget.com/searchsoftwarequality/definition/quality-assurance

[Pristupljeno: 15.9.2022.]

Investopedia (2022.) What Is Quality Control (QC)?. Investopedia. [Online]. Dostupno na: https://www.investopedia.com/terms/q/quality-control.asp

[Pristupljeno: 15.9.2022.]

Guru99 (2022.) White Box Testing – What is, Techniques, Example & Types. Guru99. [Online]. Dostupno na: https://www.guru99.com/white-box-testing.html

[Pristupljeno: 17.9.2022.]

Software Testing Material (2022.) Grey Box Testing Guide | What You Should Know. Software Testing Material. [Online]. Dostupno na: https://www.softwaretestingmaterial.com/grey-box-testing/

[Pristupljeno: 17.9.2022.]

ArtOfTesting (2021.) Grey Box Testing. ArtOfTesting. [Online]. Dostupno na: https://artoftesting.com/grey-box-testing

[Pristupljeno: 17.9.2022.]

ArtOfTesting (2022.) Non Functional Testing. ArtOfTesting. [Online]. Dostupno na: https://artoftesting.com/non-functional-testing

[Pristupljeno: 17.9.2022.]

BIG WATER CONSULTING (2019.) Software Development Life Cycle (SDLC). BIG WATER CONSULTING. [Online]. Dostupno na: https://bigwater.consulting/2019/04/08/software-development-life-cycle-sdlc/

[Pristupljeno: 18.9.2022.]

PhoenixNAP (2022.) What Is the Software Development Life Cycle?. PhoenixNAP. [Online]. Dostupno na: https://phoenixnap.com/blog/software-development-life-cycle

[Pristupljeno: 18.9.2022.]

Future Processing (2022.) Why is quality assurance important in software development?. Future Processing. [Online]. Dostupno na: https://www.future-processing.com/blog/why-is-quality-assurance-important-in-software-development/

[Pristupljeno: 18.9.2022.]

9. SAŽETAK

U ovom se završnom radu pisalo o važnosti testiranja u osiguranju kvalitete softvera. U početku, tj. unutar uvoda se pisalo o tome što je softver, te se opisalo sve korake SDLC-a. U samom osiguranju kvalitete softvera obuhvaćeni su pojmovi osiguranja kvalitete kao i kontrola kvalitete. Nakon osiguranja kvalitete softvera pojašnjeno je testiranje softvera kao i njegova načela te se pojasnilo zbog čega je provođenje testiranja softvera neophodno te se potom opisao životni ciklus testiranja softvera. Između ostalog, opisani su i načini testiranja softvera koji mogu biti ručno, kao i automatizirano testiranje softvera. U zadnjoj cjelini metodologije testiranja softvera naveli su se tipovi testiranja softvera pod kojim spadaju funkcionalno te također i nefunkcionalno testiranje softvera. Nakon toga su opisane razine testiranja softvera, da bi se na kraju navele i objasnile metode testiranja.

10. ABSTRACT

In this final paper, it was written about the importance of testing in software quality assurance. In the beginning, i.e. within the introduction, it was written about what the software is, and all the steps of the SDLC were described. Software quality assurance itself includes the concepts of quality assurance and quality control. After software quality assurance, software testing and its principles were explained, why software testing is necessary, and then the life cycle of software testing was described. Among other things, methods of software testing are described, which can be manual, as well as automated software testing. In the last part of the software testing methodology, the types of software testing were listed, which includes functional and also non-functional software testing. After that, the levels of software testing are described, and finally the testing methods are listed and explained.