

Programi për Shkenca Kompjuterike dhe Inxhinierise

**RENDËSIA E TESTIMIT NË ZHVILLIMIN E APLIKACIONEVE DHE BENEFITET E TESTIMIT AUTOMATIK**

Shkalla Bachelor

Suhejb Morina

Shtator / 2023

Prishtinë



Programi për Shkenca Kompjuterike dhe Inxhinierise

Punim Diplome

Viti akademik 2017 – 2018

Suhejb Morina

**RENDËSIA E TESTIMIT NË ZHVILLIMIN E APLIKACIONEVE DHE BENEFITET E TESTIMIT AUTOMATIK**

Mentori: Lamir Shkurti

Shtator / 2023

Ky punim është përpiluar dhe dorëzuar në përmbushjen e kërkesave të pjesshme për Shkallën Bachelor

**ABSTRAKTI**

Ky punim hulumtues shqyrton rendësinë e testimit në zhvillimin e aplikacioneve dhe paraqet përfitimet e testimit automatik në krahasim me testimin manual. Qëllimi kryesor është të ndërtohet një argument i fortë për rëndësinë e testimit si një komponent e rendesishme në procesin e zhvillimit të aplikacioneve dhe të paraqiten përfitimet konkrete që testimi automatik sjell në aspektet e shpejtësisë, cilësisë dhe efikasitetit të procesit të zhvillimit.  
 Në këtë studim, u përdorën një kombinim i burimeve të shkencës kompjuterike, literaturës shkencore, rasteve të suksesit dhe bisedimeve me profesionistë të fushës së zhvillimit të aplikacioneve për të marrë një perspektivë të gjerë mbi temën. Analiza e literaturës dhe burimeve të tjera u bë për të identifikuar dhe vlerësuar konceptet, metodologjitë dhe përparësitë e testimit në zhvillimin e aplikacioneve.  
Rezultatet e këtij studimi tregojnë se testimi është një element kyç për garantimin e sigurisë, stabilitetit dhe cilësisë së aplikacioneve. Testimi automatik ka përfitime të dukshme në krahasim me testimin manual, duke përmirësuar shpejtësinë, përsëritshmërinë, efikasitetin dhe kostot e procesit të zhvillimit. Gjithashtu, testimi automatik rrit aftësinë e aplikacioneve për të përballuar nevojat e një numri të madh të përdoruesve dhe sjell përfitime të dukshme në aspektet e zbulimit dhe korrigjimit të gabimeve.  
Përfundimet e këtij studimi përmblidhen në faktin se testimi është një pjesë e zhvillimit të aplikacioneve dhe duhet të konsiderohet si një proces thelbësor në çdo fazë të zhvillimit. Testimi automatik ofron një ndihmë të çmuar për zhvilluesit në garantimin e cilësisë së aplikacioneve, rritjen e shpejtësisë së zhvillimit dhe zvogëlimin e kostove. Me këto përfitime, testimi automatik përbën një investim të vlefshëm dhe sjell vlerë të shtuar në procesin e zhvillimit të aplikacioneve.  
Fjalë kyçe: testimi automatik, zhvillimi i aplikacioneve, cilësia e aplikacioneve, rëndësia e testimit, paraqitja e rasteve të suksesit.

**FALËNDERIME**

Ne këtë faqe të falënderimeve, dëshiroj të shpreh mirënjohjen time më të thellë dhe respektin për departamentin e Shkencave Kompjuterike dhe Inxhinierisë në Universitetin e Biznesit dhe Teknologjisë (UBT). Kjo është një rast i veçantë për mua për të shprehur përshtypjet dhe ndjenjat e mia në lidhje me rrugën time të studimeve dhe përvojën e shkëlqyer që kam patur gjatë kohës si student dhe përfundimit të shkalles bachelor në shkenca kompjuterike.

Fillimisht, dëshiroj të falenderoj të gjithë stafin akademik të Fakultetit të Shkencave Kompjuterike dhe Inxhinierisë për ndihmën, përkrahjen dhe udhëzimet e tyre gjatë këtij udhëtimi të ngritjes akademike. Profesorët e kualifikuar kanë sjellë një ndikim të madh në rritjen time profesionale dhe kanë rritur pasionin tim për teknologjinë dhe inxhinierinë kompjuterike.

Nuk mund të mos falenderoj kolegët dhe miqtë e mi, që më kanë inkurajuar, mbështetur dhe ndihmuar gjatë gjithë kohës së studimeve.

Gjithashtu, dëshiroj të shpreh mirënjohjen time ndaj familjes sime, që më ka përkrahur me dashuri, inkurajim dhe krenari gjatë gjithë këtijë rrugëtimi të madhe. Pa mbështetjen e tyre të palodhur, nuk do të kisha mundur të arrita deri këtu.

Në përgjithësi, do të dëshiroja ta përshëndes edhe atmosferën fantastike të Universitetit të Biznesit dhe Teknologjisë (UBT), e cila ka bërë që kjo kohë e ndërlikuar të kthehet në një periudhë të paharrueshme të jetës sime. Ambienti i këtij institucioni ka përbërë një burim frymëzimi dhe motivimi për të nxjerrë më të miren prej vetes time.

Nga thellësia e zemërs falënderoj sërish departamentin e Shkencave Kompjuterike dhe Inxhinierisë në UBT për këtë mundësi të çmuar për shperndarjen e njohurive nga profesionistët e fushave të ndryshme për mua si student, kujtime të paharrueshme që do të mbeten në mendjen time përjetë.

**PËRMBAJTJA**

[**LISTA E FIGURAVE** V](#_Toc26128788)

[**LISTA E TERMAVE** VI](#_Toc26128789)

[**1. HYRJE** 1](#_Toc26128790)

1.1 Përshkrimi i temës dhe qëllimi i studimit.............................................................................1

[**2. KONCEPTET THEMELORE TË TESTIMIT**](#_Toc26128790) 2

2.1 Çka është testimi...............................................................................................................15  
 2.1.1 Objetivat tipike të testimit..................................................................................15

2.1.2 Testimi dhe Korrigjimi.......................................................................................15

2.2 Roli i testimit dhe pse testimi është i domosdoshëm.........................................................15

2.2.1 Gabime, Defekte dhe Dështime..........................................................................10

2.3 Shtatë Parimet e Testimit...................................................................................................16

2.4 Aktivitetet dhe Etapat e Testimit.......................................................................................17

2.5 Metodat e testimit dhe krahasimi i testimin manual me testimin automatik.....................17

[**3. RËNDËSIA E TESTIMIT NË ZHVILLIMIN E APLIKACIONEVE** 1](#_Toc26128790)

3.1 Identifikimi i sfidave të mungesës së testimit në zhvillimin e aplikacioneve...................15

3.2 Efektet pozitive të testimit në cilësinë dhe performancën e produktit..............................16

[**4. PËRFITIMET E TESTIMIT AUTOMATIK** 1](#_Toc26128790)

4.1 Përshkrimi i testimit automatik dhe dallimet me testimin manual....................................15

4.2 Shpjegimi i përfitimeve të testimit automatik...................................................................16

4.2.1 Shpejtësia...........................................................................................................17

4.2.2 Përmirësimi i cilësisë.........................................................................................17

4.2.3 Rishfrytëzimi i burimeve...................................................................................17

[**5. FRAMEWORKS DHE MJETE TË TESTIMIT AUTOMATIK** 1](#_Toc26128790)

5.1 Përshkrimi i disa frameworks dhe mjete të përdorura për testimin automatik..................15

5.2 Analiza e karakteristikave dhe avantazheve të secilit framework/mjet.............................16

5.3 Zgjedhja e një frameworku/mjeti për testimin automatik në platformën e UBT-s............17

[**6. TESTIMI NË PLATFORMËN MOODLE E UBT'S** 1](#_Toc26128790)

6.1 Përshkrimi i procesitë të testimit në platformën Moodle...................................................15

6.2 Identifikimi dhe përcaktimi i skenarëve të testimit në platformë......................................16

6.3 Realizimi i testimeve dhe analiza e rezultateve.................................................................17

[**7. ISO STANDARDET** 1](#_Toc26128790)

7.1 Përshkrimi i standardit ISO 27001 për sigurinë e informacionit.......................................15

7.2 Përfshirja e ISO 27001 në procesin e testimit të aplikacioneve........................................15

[**8. DISKUTIM DHE PËRFUNDIM** 1](#_Toc26128790)

8.1 Ndikimit i testimi automatik në cilësinë e produktit dhe performancën e tij....................15

8.2 Përfundimi i projektit dhe përmendja e rekomandimeve për përdorimin e testimit automatik në zhvillimin e aplikacioneve..................................................................................16

8.3 Hapat dhe Sfidat e mundshme të testimit në të ardhmen...................................................17

[**REFERENCAT** 31](#_Toc26128812)

# **LISTA E FIGURAVE**

[Figura 1.](#_Toc26116411)

[Figura 2.](#_Toc26116412)

[Figura 3.](#_Toc26116413)

.

.

.

.

.

.

**LISTA E TERMAVE**

|  |  |
| --- | --- |
| API | Application Programming Interface |

CI/CD Continuous Integration/Continuous Deployment

PIP Python Package Manager

IDE Integrated Development Environment

npm Node Package Manager

# 

# **HYRJE**

1.1 Parashikimi i temës dhe qëllimi i studimit

Në një epokë kur inovacioni teknologjik ka përfshirë çdo aspekt të jetës moderne, zhvillimi i aplikacioneve ka shërbyer si një katalizator i ndryshimeve thelbësore në mënyrën se si ndihemi, veproijm dhe bashkëveproijm me botën rreth nesh. Ky zhvillim madhor sjell me vete një sërë sfidash dhe kërkesash të reja, e njëra prej tyre është siguria dhe cilësia e këtyre aplikacioneve.

Në këtë kontekst, testimi i aplikacioneve bëhet një komponent kritik për garantimin e cilësisë dhe performancës së aplikacioneve. Ky proces përfshin verifikimin e funksionalitetit, stabilitetin, sigurisë dhe performancen e aplikacioneve për tu siguruar që ato plotësojnë pritshmëritë e përdoruesve. Për shkak të përdorimit të madhë të aplikacioneve në jetën tonë të përditshme, domosdoshmëria e një testimi të kujdesshëm dhe efikas është bërë më e rëndësishme se kurrë më parë.

Ky studim synon të hulumtojë thelbin e testimit në zhvillimin e aplikacioneve dhe të paraqesë përfitimet që sjellin testimi automatik dhe testimi manual në këtë kontekst. Do të përqendrohemi në përshkrimin e rëndësisë së testimit si pjesë thelbësore e procesit të zhvillimit të aplikacioneve dhe hulumtime, përfitimet e testimit automatik në krahasim me metodat tradicionale të testimin manual.

Në kuadër të këtij hulumtimi, do të eksplorohen metodat, teknikat dhe mjete të testimit manual dhe testimit automatik të cilat kontribuojnë në përmirësimin e efikasitetit, cilësisë dhe sigurisë së aplikacioneve të zhvilluara. Po ashtu, do të analizohet përdorimi i testimit automatik në platformën e një aplikacioni praktik, duke shqyrtuar përfitimet dhe sfidat që paraqesin këto metoda në një kontekst të vërtetë.

Me anë të këtij studimi, synohet të që të kemi një pamje më të detajuar për testimin dhe rolin e testimit në zhvillimin e aplikacioneve dhe të nxisë përdorimin e metodave të automatizuara për të përmbushur kërkesat e një bote që përdor metoda të avancuara teknologjike. Gjithashtu, do të japë një pamje të standardeve dhe praktikave më të mira të sigurisë që janë të lidhura me testimin e aplikacioneve, duke i bërë ato të qëndrueshme dhe të sigurta në një mjedis ku siguria është prioritet.

# **KONCEPTET THEMELORE TË TESTIMIT**

2.1 Çka është testimi

Sistemet softuerike janë një pjesë integrale e jetës, bizneset si (p.sh., bankat) deri te produktet per konsumatorin si (p.sh., marketet). Shumica e njerëzve kanë pasur përvojë me softuer që nuk ka funksionuar siç pritej. Softueri që nuk funksionon në mënyrë të saktë mund të çojë në shumë probleme, duke përfshirë humbjen e parave, kohës, reputacionin e biznesit dhe madje edhe dëmtimet ose vdekjen. Testimi i softuerit është një mënyrë për të vlerësuar cilësinë e softuerit dhe për të ulur riskun e dështimit të softuerit gjatë perdorimit.

Procesi i testimit përfshin veprimtari si planifikimi i testeve, analiza, projektimi dhe zbatimi i testeve, raporti i progresit dhe rezultateve të testeve, dhe vlerësimi i cilësisë së një objekti. Një ide e gabuar për testimin është se ai përbëhet vetëm nga ekzekutimi i testeve, pra ekzekutimi i softuerit dhe verifikimi i rezultateve.

Disa teste përfshijnë ekzekutimin e një komponenti ose sistemin që po testohet, testet e tilla quhen teste dinamike. Teste të tjera që nuk përfshijnë ekzekutimin e një komponenti ose sistemin që po testohet, testet e tilla quhen teste statike. Kështu testimi përfshin gjithashtu shqyrtimin e produktit final si kërkesat funksionale, veçorit e nje softueri dhe kodi i aplikacionit testohet.

Një ide tjeter e gabuar për testimin është se ai përqendrohet plotësisht në verifikimin e kërkesave, veçorive të softuerit ose specifikimeve të tjera. Përderisa testimi përfshin verifikimin e nëse sistemi plotëson kërkesat e specifikuara, ai gjithashtu përfshin edhe validimin, që është verifikimi nëse sistemi ose softueri do të plotësojë nevojat e përdoruesve dhe palëve të tjera në një mjedisë pune ose në tregun e punës.

2.1.1 Objetivat tipike të testimit

Për çdo projekt të dhënë, objektivat e testimit mund të përfshijnë:

* Parandalimin e defekteve duke vlerësuar produktin në kërkesat funksionale, veçorit e nje softueri si dhe në kodi e shkruar.
* Verifikimin nëse janë plotësuar të gjitha kërkesat e specifikuara.
* Kontrollin nëse objekti i testimit është i plotë dhe nese validimi punon ashtu siç e presin përdoruesit dhe palët e tjera të interesuara.
* Ngritja e besimit në nivelin e cilësisë së objektit të testimit.
* Zbulimi i defekteve dhe dështimeve duke reduktuar nivelin e rrezikut të cilësisë të dobët të softuerit.
* Furnizim i mjaftueshëm me informacion për palët e interesuara për t'u lejuar atyre të marrin vendime të informuara, veçanërisht lidhur me nivelin e cilësisë së objektit të testuar.
* Përputhshmeria në mes kërkesave ose standardet kontraktuale, ligjore dhe krahasimi me përputhshmërinë e objektit të testimit me këto kërkesa ose standarde

Qëllimet e testimit mund të ndryshojnë, në varësi të kontekstit të komponentit ose sistemit që po testohet, nivelit të testimit dhe ciklit të zhvillimit të softuerit. Këto dallime mund të përfshijnë, për shembull:

* Gjatë testimint të komponentit, një qëllim mund të jetë zbulimi i sa më shumë gabimeve të mundshme në mënyrë që defektet bazë të identifikohen dhe riparohen në mënyrë të hershme. Një qëllim tjetër mund të jetë rritja e mbulimit të kodit nga testet e komponentit.
* Gjatë testimit të miratimit, një qëllim mund të jetë konfirmimi që sistemi funksionon ashtu siç pritej dhe plotëson kërkesat. Një qëllim tjetër i këtij testimi mund të jetë të ofrojë informacion palëve të interesuara lidhur me rrezikun e lëshimit të sistemit për një kohë të caktuar.

2.1.2 Testimi dhe Korrigjimi

Testimi dhe korrigjimi i gabimeve janë të ndryshme. Ekzekutimi i testeve mund të tregojë dështime që shkaktohen nga defektet në softuer. Korrigjimi i gabimeve është aktivitet i zhvillimit që gjen, analizon dhe rregullon këto defekte. Testimi pas rregullimit të mëvonshëm verifikon nëse përpunimet rregulluan defektet. Në disa raste, testerët janë përgjegjës për testimin fillestar dhe testimin e rregullimit final, ndërsa zhvilluesit rregullojnë gjetjen e gabimeve, testimin e komponenteve dhe testimi i lidhjeve të komponentave. Megjithatë, në metodologjinë Agile dhe në disa cikle të tjerë të zhvillimit të softuerit, testerët mund të jenë të përfshirë në gjetjen e gabimeve dhe testimin e komponenteve.

2.2 Roli i testimit dhe pse testimi është i domosdoshëm

Testimi kontribuon në sukësë e sidomos kur aty janë të përfshirë edhe testerët, me përfshirjen e testerëve dhe gjetjen e gabimeve në faza të hershme kursen shumë kohë dhe para por më e rëndesishmja sistemi sigurohet që është i aftë të punoj në treg. Duke i pasur testerët e përfshirë në shqyrtimin e kërkesave ose përmirësimin e veçorive të nje softueri mund të zbulohen më leht defektet në këto produkte që do të përdoren më vonë për punë.

* Identifikimi dhe heqja e kërkesave që mendohet që kane defekte zvogëlon rrezikun për të pasur funksionalitetit të gabuar ose defekt të pakapshëm gjatë zhvillimit.

Duke pasur testerët që punojnë ngushtë me dizajnerët e sistemit përderisa sistemi është duke u projektuar mund të rrisë njohurit e dizajnerëve dhe të testerëve si është mënyra e sakt për ta dizajnuar dhe testuar atë produkt.

* Ky kuptim nga të dy palët mund të zvogëlojë rrezikun e defekteve themelore të dizajnit dhe mundëson që gabimet të identifikohen në një fazë të hershme nga ana e testeve. Ky bashkëpunim dhe kuptim vlen edhe për zhivilluesit e sistemi që duhet të punojn ngushtë ndërmjet vete.

Testimi i ashpër i komponenteve dhe sistemeve, mund të ndihmoj që së bashku me dokumentacioni i tyre, të punohet në zvogëlimin e rrezikut të dështimeve gjatë operacionit në punë. Kur defekte zbulohen dhe më pas riparohen nga zhvilluesit, kjo kontribuon në cilësinë e komponenteve ose sistemeve. Përveç kësaj, testimi i softuerit gjithashtu mund të kërkohet për të përmbushur kërkesat kontraktuale ose ligjore, ose standardet e veçanta të industrisë.

Gjatë historisë së zhvillimit të teknologjisë, është shumë e zakonshme që softueri dhe sistemet t'i dorëzohen testimit, për shkak të pranisë së defekteve, më pas ato shkaktojnë dështime ose ndryshe nuk i plotësojnë nevojat e palëve të interesuara. Megjithatë, përdorimi i teknikave të duhura të testimit mund të zvogëlojë numrin e dorëzimeve të tilla problematike, kur ato teknika zbatohen me nivelin e duhur të ekspertizës së testimit, në nivele të duhura të testimit dhe në pikat e duhura gjatë ciklit së zhvillimit të softuerit.

Përveç këtyre shembujve, arritja e objektivave të përcaktuara të testit kontribuon për suksesin e përgjithshëm të zhvillimit dhe mirëmbajtjes së softuerit.

2.2.1 Gabime, Defekte dhe Dështime

Një person mund të bëjë një gabim, i cili mund të çojë në gjetjen e një defekti (gabim ose mungesë) në kodin e softuerit ose në ndonjë produkt/vegël të punës. Një gabim që çon në gjetjen e një defekti në një produkt të punës mund të shkaktojë një gabim tjeter që çon në gjetjen e një defekti tjeter në një produkt që është i lidhur me produktin e punës. Për shembull, një gabim në identifikimin e kërkesave mund të çojë në një defekt të kërkesave, i cili më pas rezulton në një gabim programimi ose zhvillimi e që çon në një defekt në kod.

Nëse një defekt në kod ekzekutohet, kjo mund të shkaktojë një dështim, por jo domosdoshmërisht në të gjitha rastet. Për shembull, disa defekte kërkojnë të dhëna shumë specifike ose parakushte për të shkaktuar një dështim, i cili mund të ndodhë rrallë ose kurrë.

Gabimet mund të ndodhin për shumë arsye, si:

* Presioni i kohës.
* Padituria e njeriut.
* Pjesëmarrësit e projektit të pa përvojë ose pa eksperienc.
* Komunikimi i gabuar në mesës të pjesëmarrësve në projekt, duke përfshirë komunikimin e gabuar për kërkesat dhe projektimin e projektit.
* Kompleksiteti i kodit, projektimit, arkitekturës, problemi bazë që duhet të zgjidhet dhe/ose teknologjitë e përdorura.
* Teknologjitë e reja, të panjohura për ekipen e projektit.

Përveç dështimeve të shkaktuara nga defektet në kod, dështimet mund të shkaktohen gjithashtu nga kushtet mjedisore. Për shembull, valët elektromagnetike, fushat elektromagnetike dhe ndotja mund të shkaktojnë defekte në softuer ose të ndikojnë në ekzekutimin e softuerit duke ndryshuar kushtet e harduerit.

Jo të gjitha rezultatet e papritura të testeve janë dështime. Rezultatet pozitive të rreme mund të ndodhin për shkak të gabimeve në mënyrën se si janë ekzekutuar testet, ose për shkak të defekteve në të dhënat e testit, mjedisin e testimit, ose për shkaqe të tjera. Situata e kundërt mund të ndodhë gjithashtu, ku gabimet ose defektet e ngjashme çojnë në rezultate negative të rreme. Rezultatet negative të rreme janë testet që nuk zbulojnë defekte që duhet të kishin zbuluar, rezultatet pozitive të rreme raportohen si defekte, por në të vërtetë nuk janë defekte.

2.3 Shtatë Parimet e Testimit

Në të kaluarën për 50 vjet janë sugjeruar një numër parimesh të testimit që ofrojnë udhëzime të përgjithshme të përbashkëta për të gjitha llojet e testimit.

1. **Testimi tregon praninë e defekteve, jo mungesën e tyre.**

Testimi mund të tregojë se defekte janë të pranishme, por nuk mund të provojë që nuk ka defekte. Testimi zvogëlon mundësinë që defekte të paidentifikuar të mbeten në softuer por, edhe nëse nuk gjenden defekte, testimi nuk është një dëshmi e saktësisë.

1. **Testimi i plotë/gjithëpërfshirës është i pamundur.**

Testimi i gjithçkaje (të gjitha kombinimet, inputet dhe parakushtet) nuk është i realizueshëm përveç rasteve trivial (të thjeshta). Më mirë se sa të përpjekemi për një testim të plotë, prioritete duhet te jetë analiza e rrezikut, teknikat e testimi për të përmisuar përpjekjet e testimit.

1. **Testimi i hershëm e kursen kohë dhe para.**

Për të zbuluar defektet në mënyrë të hershme, të dy llojet e testimi ajo statike dhe dinamike duhet të fillojnë sa më herët të jetë e mundur në ciklin/metodologjinë e punës së zhvillimit të softuerit. Testimi i hershëm zakonisht quhet "shift left" (zhvendosje majtas). Testimi i hershëm në ciklin/metodologjinë e punës së zhvillimit të softuerit ndihmon në zvogëlimin ose eliminimin e ndryshimeve të kushtueshme.

1. **Defektet grumbullohen së bashku.**

Një numër i vogël modulesh zakonisht përmban shumicën e defekteve të gjetura para se të bëhet testimit final, ose është përgjegjës për shumicën e dështimeve funksional. Grumbullimet e defekteve që parashikohen dhe grumbullimet e defekteve të vëzhguara në testim ose në operacion janë një hyrje e rëndësishme në një analizë të rrezikut të përdorur për të përqendruar përpjekjet e testimi.

1. **Kujdes me teste të përseritura(paradoksi pesticid).**

Nëse të njëjtat teste përsëriten herë pas here, në fund këto teste nuk gjejnë më defekte të reja. Për të zbuluar defekte të reja, mund të duhet të përditesohen testet dhe të dhënat e testimit ekzistuese, ose mund të duhet të shkruhen teste të reja. (Testet nuk janë më efektive në zbulimin e defekteve, pikërisht ashtu si pesticidet nuk janë më efektive në vrasjen e insekteve pas një kohe të caktuar.) Në disa raste, siç është testimi automatik përsëritës, paradoksi pesticid ka një rezultat të favorshëm, që është numri relativisht i ulët i defekteve përsëritëse.

1. **Testimi është i varur nga konteksti.**

Testimi kryhet në mënyra të ndryshme në kontekste të ndryshme. Për shembull, softueri i qendrës së kontrollit në ndonje industri kritike për siguri testohet ndryshe nga një aplikacion i thjeshtë mobil. Si një shembull tjetër, testimi në një projekt Agile bëhet ndryshe se sa testimi në një projekt pa metodologjinë e punës së një produkti.

1. **Mungesa e gabimeve është një iluzion.**

Disa organizata presin që testerët të mund të kryejnë të gjitha testet e mundshme dhe të gjejnë të gjitha defektet e mundshme, por parimet 1 dhe 2, respektivisht na tregojnë që kjo është e pamundur. Për më tepër, është një iluzion (pra, një mendim dhe besim i gabuar) të presësh që thjesht gjetja dhe rregullimi i një numri të madh të defekteve do të siguronte suksesin e një sistemi. Për shembull, testimi i plotë i të gjitha kërkesave të specifikuara dhe rregullimi i të gjitha defekteve të gjetura ende mund të prodhojë një sistem që është i vështirë për t'u përdorur, që nuk plotëson nevojat dhe pritshmëritë e përdoruesve, ose që është në një nivel më të ulët krahasuar me sistemet të tjera konkurruese në treg.

2.4 Aktivitetet dhe Etapat e Testimit

Procesi i testimi përbëhet nga këto grupe kryesore si më mëposht:

* Planifikimi i testimit
* Monitorimi dhe kontrollimi i testimit
* Analiza e testimit
* Dizajnimi i testimit
* Implementimi i testimit
* Ekzekutimi i testimit
* Finalizimi i testimit

Secila veprimtarive nga këto grupe kryesor përbëhet nga veprimtari përbërëse, të cilat do të përshkruhen më poshtë. Çdo veprimtari përbërëse përbëhet nga disa detyra individuale, të cilat ndryshojnë nga një projekt ose verzion tjetër.

Edhe pse shumë prej këtyre grupeve kryesore veprimtaritë e tyre mund të duken të logjikisht, ato shpesh implementohen me iteracione. Për shembull, zhvillimi në metodologjinë Agil përfshin iteracione të vogla të projektimit të softuerit, zhvillimit dhe testimin që ndodhin në bazë të vazhdueshme, të mbështetur nga planifikimi i vazhdueshëm. Kështu që veprimtaritë e testimit ndodhin gjithashtu në një mënyrë të vazhdueshme brenda procesit të zhvillimit të softuerit. Edhe në zhvillimin e softuerit në menyrë të planifikuar, sekuenca logjike e grupeve kryesore të do të përfshijë përplasje, kombinim, konkurencë, kështu që zakonisht këto veprimtarive të grupe kryesore duhet të përshtaten brenda kontekstit të sistemit dhe projektit.

**Planifikimi i testimit**

Planifikimi i testimit përfshin aktivitete që përcaktojnë objektivat e testimit dhe qasjen për të përmbushur ato objektiva të testimit brenda kufizimeve të imponuara nga konteksti (p.sh., përcaktimi i teknikave dhe detyrave të përshtatshme të testimit dhe formulimi i një kohe testimi në një afat kohor). Planifikimet e testimit mund të rishikohen bazuar në vlersimin e aktiviteteve të monitorimit dhe kontrollimit.

**Monitorimi dhe kontrolli i testimin**

Monitorimi i testimit përfshin krahasimin e vazhdueshëm të suksesit aktual me suksesin e planifikuar duke përdorur metrika(grafe) për monitorimin e testimit të përcaktuara në planin e testimit.

Kontrolli i testimit përfshin veprimet e nevojshme për të përmbushur objektivat e planit të testimit (që mund të përditësohen me kalimin e kohës). Monitorimi dhe kontrolli i testimin mbështetet nga vlerësimi i kritereve finale, të cilat referohen si “definim i përfunduara” në disa modele ose metedologji të zhvillimit të softueri. Për shembull, vlerësimi i kritereve të finale për ekzekutimin e testimit si pjesë e një nivel të caktuar, testimi mund të përfshijë:

* Kontrollimi i rezultateve të testimin dhe qasjes ndaj një kriteri specifike.
* Vlerësimin e niveleve të cilësisë së komponentit ose softuerit bazuar në rezultatet e testimit dhe regjistrave
* Përcaktimin nëse kemi nevojë më shumë teste (p.sh. nëse testet që ishin parashikuar për të arritur një nivel të caktuar të mbulimit të rrezikut të produktit nuk kanë mundur ta bëjnë këtë, duke kërkuar shkrimin dhe ekzekutimin e testeve të shtuara)

Suksesi i testimin në përputhje me planin ju komunikohet(tregohet) palëve të interesuara në raportin e përparimit të testimit, duke përfshirë devijimet nga plani dhe informacionin për të mbështetur çdo vendim për të ndaluar testimin(si proces i përfunduar ose kompletuar).

**Analiza e testimit**

Gjatë analizës së testimit, faza fillestare e testimit është që të analizohen dhe të identifikohen karakteristikat e testimit dhe për të përcaktuar kushtet e testimit. Me fjalë të tjera, analiza e testimit përcakton "çfarë të testohet" si kriterë për mbulimit të caktuar.

Aplikimi i teknikave të testimit black-box (kuti e zeze), white-box (kuti e bardhë), dhe të bazuar në përvojë mund të jetë i dobishëm në procesin e analizës së testimit, për të reduktuar mundësinë e harrimit të kushteve të rëndësishme të testimit dhe për të përcaktuar kushte më precize dhe më të sakta në testim.

Në disa raste, analiza e testimit prodhon kushte të testimit që duhet të përdoren si objektiva të testimit p.sh grafet e testimit. Grafet e testimit janë produkte të zakonshme të punës në disa lloje të testimit të bazuar në përvojë. Kur këto objektiva të testimit janë të lidhura me fazën fillestare të testimit, mbulimi i arritur gjatë këtij testimi të bazuar në përvojë mund të matet(caktohet).

Identifikimi i defekteve gjatë analizës së testimit është një potencial i rëndësishëm, veçanërisht kur nuk përdoret ndonjë proces tjetër i rishikimit dhe/ose procesi tjetër i testimit. Aktivitetet e analizës së testimit jo vetëm verifikojnë nëse kërkesat janë konsistente, të shprehura me mënyrë të duhur dhe të plota, por edhe validojnë nëse kërkesat plotësojnë në mënyrë të duhur nevojat e klientit, përdoruesit dhe palë tjera të interesuara.

**Dizajnimi i testimit**

Gjatë dizajnimit të testimit, kushtet e testimin elaborohen në raste të komplikuara të testimit, grupohen në rasteve të komplikuara të testimit dhe materiale të tjera për testimit. Prandaj, analiza e testimin përgjigjet pyetjes "çfarë të testohet?" ndërsa dizajnimi i testimit përgjigjet pyetjes "si të testohet?"

Dizajnimi i testimit përfshin veprimtaritë kryesore të mëposhtme:

* Dizajnimi dhe prioritetizim i skenarëve të testimit dhe grupimeve të skenarëve të testimin.
* Identifikimi i të dhënave të nevojshme të testimit për të mbështetur kushtet e testimin dhe skenarëve e testimit.
* Dizajnimi i mjedisit të testimit dhe identifikimi i infrastrukturës dhe mjetëve të nevojshme.
* Kapja e dydrejtimore midis bazës së testimit, kushteve të testimit dhe skenarëve të testimit.

Elaborimi i kushteve të testimin në skenarëve të testimit dhe grupime të tyre gjatë dizajnit të testimin shpesh përfshin përdorimin e teknikave të testimin.

Siç ndodh edhe me analizën e testimit, dizajnimin i testimit mund të rezultojë në identifikimin e llojeve të ngjashme të defekteve në bazën e testimit. Gjithashtu, ashtu si me analizën e testimin, identifikimi i defekteve gjatë dizajnit të testimit është një përfitim potencial i rëndësishëm.

**Implementimi i testimit**

Gjatë implementimit të testimit, krijohet dhe/ose përfundohet pajisja e nevojshme për ekzekutimin e testimi, përfshirë vendosjen e skenarëve të testimit në procedura të testimit. Prandaj, dizajni i testimin përgjigjet pyetjes "si të testohet?" ndërsa implementimi i testimin përgjigjet pyetjes "a kemi tani të gjitha gjërat në vend për të ekzekutuar testet?" Implementimi i testimit përfshin veprimtaritë kryesore të mëposhtme:

* Zhvillimi dhe prioritetizimi i procedurave të testimit dhe, potencialisht, krijimi i skripteve të automatizuara të testimit.
* Krijimi i koleksione të testimit nga procedurat e testimin dhe (në qoftë se ka) skriptet e automatizuara të testimit.
* Vendosja e koleksione të testimit në një orar ekzekutimi për testimi në një mënyrë që rezulton në një ekzekutim efikas.
* Ndërtimi i mjedisit të testimit (përfshirë potencialisht pajisjet e testimit, virtualizimin e shërbimit, simulatorët dhe të tjera elementë të infrastrukturës) dhe verifikimi që çdo gjë e nevojshme është vendosur në mënyrë të duhur.
* Përgatitja e të dhënave të testimit dhe sigurimi që ato janë ngarkuar në mënyrë të duhur në mjedisin e testimit.
* Verifikimi dhe përditësimi i të dhënave dydrejtimore midis bazës së testimit, kushteve të testimit, skenarëve të testimit, procedurave të testimit dhe koleksione të testimit.

Detyrat e dizajnit të testimit dhe implementimit të testimit shpesh kombinohen. Në testimin e thjeshtë dhe në lloje të tjera të testimit të bazuar në përvojë, dizajni dhe implementimi i testimit mund të ndodhin dhe mund të dokumentohen, si pjesë e ekzekutimit të testimin.

Testimi i thjeshtë mund të bazohet në grafe (prodhuar si pjesë e analizës së testimin), dhe testimet e thjeshta ekzekutohen menjëherë pasi ato janë projektuar dhe implementuar.

**Ekzekutimi i testimit**

Gjatë ekzekutimit të testimit, koleksionet për testimi ekzekutohen në përputhje me orarin e caktuar për ekzekutim të testimit.

Ekzekutimi i testimit përfshin veprimtaritë kryesore të mëposhtme:

* Krahasimi i rezultateve aktuale me rezultatet e pritura.
* Regjistrimi i ID-ve dhe versioneve të artikullit të testimit ose objektit të testimit, mjeteve të testimit dhe testeve.
* Ekzekutimi i testeve, ose me dorë ose duke përdorur mjete të ekzekutimit të testeve.
* Regjistrimi i rezultateve të ekzekutimit të testimit (për shembull, kalon, dështoj, bllokoj)

**Finalizimi i testimit**

Veprimtaritë e finalizimit të testimit mblidhin të dhëna nga veprimtaritë e përfunduara të testimin për të konsoliduar eksperiencën, testimet dhe çdo informacion të rëndësishëm të tjera. Veprimtaritë e përfundimit të testimin ndodhin në pikat e ndërhyrjeve të projektit, të tilla si kur një sistem softueri lansohet, një projekt testimi përfundohet (ose anullohet), një cikel projektimi Agile përfundohet, një nivel testimi përfundohet, ose një version mirëmbajtje është përfunduar.

Përfundimi i testimit përfshin veprimtaritë kryesore të mëposhtme:

* Kontrollimi nëse të gjitha raportet e defekteve janë mbyllur, duke regjistruar kërkesa për ndryshime në "listen e detyrave" për defektet që mbeten të pazgjidhura dhe në fund të ekzekutimit të testimin.
* Krijimi i një raporti të përgjithshëm të testimit që do të komunikohet me palët e interesit.
* Përfundimi dhe arkivimi i mjedisit të testimit, të të dhënave të testimit, infrastrukturës së testimit dhe testimet e tjera.
* Dorëzimi i testimeve tek ekipet e mirëmbajtjes, të tjerët ekipet e projektit, dhe/ose palët interesit dhe të tjerë që mund të përfitojnë nga përdorimi i tyre.
* Analiza e mësimeve të nxjerra nga veprimtaritë e përfunduara të testimit për të përcaktuar ndryshimet e nevojshme për ciklet e ardhshme, lasnsime dhe projekte.
* Përdorimi i informacionit të mbledhur për të përmirësuar kohen e procesit të testimin.

2.5 Testimi automatik dhe roli i testimit automatik

Testimi automatik është një praktikë e rëndësishme në fushën e zhvillimit të softuerit, e cila përdoret për automatizimin e procesit të testimit për aplikacioneve kompjuterike, celulare etj. Kjo praktikë përdor skripta dhe mjete tjera te automatizuara për të ekzekutuar teste, krahason rezultatet e tyre me pritshmëritë dhe identifikon çdo defekt apo problem në softuer. Testimi automatik është i rëndësishëm për shkak të shumë arsyeve të ndryshme.

Një nga arsyet kryesore përse testimi automatik është i rëndësishëm është që ai mund të përmirësojë efikasitetin dhe saktësinë e procesit të testimit. Përdorimi i skriptave automatike lejon ekzekutimin e shpejtë dhe të përsëritshëm të testeve, duke kursyer kohën dhe përpjekjet e ndryshme nga persona të cilët do të kryenin testet manualisht. Kjo gjithashtu zvogëlon rrezikun e gabimeve nga faktori njeri në procesin e testimit, pasi skriptat janë të programuara për të ekzekutuar të njëjtin test në mënyrë konsistente çdo herë.

Testimi automatik gjithashtu ndihmon në zbulimin e defekteve në fillimet e zhvillimit e softuerit. Duke ekzekutuar teste automatike në çdo fazë të zhvillimit, problemet mund të identifikohen dhe zgjidhen më herët, duke zvogëluar kostot dhe kohën e riparimit të tyre. Kjo gjithashtu ndihmon në sigurimin e cilësisë së përgjithshme të produktit final.

Një avantazh tjetër i testimit automatik është aftësia për të kryer teste të shumta dhe komplekse që nuk do të ishin të mundshme për t'u kryer manualisht. Kjo përfshin teste me të dhena te shumta, teste të performancës, dhe teste të ndërveprimit të aplikacioneve në mjedis të ndryshëm. Testimi automatik siguron që të gjitha këto aspekte të aplikacionit të testohen në mënyrë efikase dhe të përsëritshme.

Në përgjithësi, testimi automatik është i rëndësishëm për të siguruar cilësinë dhe performancën e lartë të aplikacioneve te zhvilluara. Ai ndihmon në zbulimin e defekteve, përmirëson efikasitetin e procesit të testimin, dhe mundëson zhvilluesve dhe ekipet e testimin të jenë më produktive në projektet e tyre.

# **RENDESIA E TESTIMIT NE ZHVILLIMIN E APLIKACIONEVE**

3.1 Identifikimi i sfidave të mungesës së testimit në zhvillimin e aplikacioneve

Edhe testimi automatik ka sfida ku do ti shqyrtojmë me hollësi sfidat që shfaqen kur procesi i testimit mungon ose nuk është i përsosur në zhvillimin e aplikacioneve. Është e rëndësishme të kuptojmë që testimi është një komponent kritik i sigurisë dhe cilësisë së produkteve digjitale, dhe mungesa ose përdorimi i pakët i testimit mund të sjellë pasojë të rënda për cilësinë, performancën dhe besueshmërinë e aplikacioneve, disa nga ato jane:  
  
**Rreziku i zbulimit të defekteve në fazat e vona të zhvillimit**

Një nga sfidat kryesore që vjen me mungesën e testimit të duhur është rreziku i zbulimit të defekteve në fazat e vona të zhvillimit. Kur testimi fillohet vonë në proces, defektet mund të jenë të fshehta dhe të vështira për t'u zbuluar. Kjo shpesh çon në kosto të lartë riparimesh dhe vonesa të mëdha në projekt. Në raste ekstreme, aplikacionet me defekte të paidentifikuara mund të dalin në treg, duke sjellë pasojat më të rënda për reputacionin e biznesit dhe besimin e klientëve.

**Mundesia e humbjes e reputacionit të biznesit dhe besimit të klientëve**

Në një mjedis të tregut të eger ose të veshtirë, reputacioni është thelbësor për suksesin e një biznesi. Kur aplikacionet dështojnë ose shfaqin defekte të dukshme, klientët humbin besimin dhe ndikimi negativ ndaj reputacionit të biznesit është i paevitueshëm. Nëse një konkurrent ofron një alternativë më të besueshme, klientët mund të largohen dhe të mos kthehen më.

**Mungesa e sigurisë dhe privatësisë së të dhënave**

Testimi është thelbësor për garantimin e sigurisë dhe privatësisë së të dhënave. Aplikacionet që nuk janë të testuar mjaftueshëm për sigurinë mund të jenë të prekura nga shkelësit dhe rreziqet e tjera të sigurisë. Kjo mund të çojë në shfrytëzimin e të dhënave personale dhe në rrezikun e ndërhyrjeve të paligjshme ne te dhënat personale.

**Kostot e ndërhyrjeve pas publikimit të aplikacioneve në treg**

Nëse defektet zbulohen pasi aplikacionet janë publikohen në treg, kostoja e riparimeve është shumë më e lartë se sa kur ato zbulohen në fazat e hershme të zhvillimit. Këtu kostoja përfshijnë kohën dhe burimet e nevojshme për të adresuar defektet, përditësimin e aplikacioneve dhe ndonjë dëm të mundshëm të reputacionit.

Nëpërmjet analizës së këtyre sfidave, bëhet e qartë se testimi është thelbësor për të arritur një cilësi të lartë të aplikacioneve dhe për të minimizuar rreziqet dhe koston e riparimeve të vonshme.

3.2 Efektet pozitive të testimit në cilësinë dhe performancën e produktit

Testimi ka një ndikim të rëndësishëm në performancën, cilësinë dhe optimizimin e aplikacioneve. Ky ndikim vjen nëpërmjet identifikimit dhe korrigjimit të problemeve potenciale në fazën e zhvillimit të aplikacioneve. Këtu janë disa nga mënyrat se si testimi ndikon në këto aspekte:

1. **Performance e pëmirësuar:** Testimi i performancës identifikon ngadalësitë dhe problemet e performancës në një aplikacion. Me identifikimin e këtyre problemeve, zhvilluesit kanë mundësi të kryejnë optimizime dhe përmirësime për të siguruar që aplikacioni të punojë më shpejt dhe të jete me efikas.
2. **Cilësi të përmirësuar:** Testimi në cilësi zbulon defekte, publikime të gabuara dhe probleme të tjera që ndikojnë në cilësinë e aplikacionit. Duke korrigjuar këto probleme, aplikacioni bëhet më i qëndrueshëm dhe më i besueshëm për përdoruesit.
3. **Identifikimi i rreziqeve në siguri:** Testimi i sigurisë identifikon mënyrat potenciale të shkeljes së sigurisë në aplikacion. Ky proces ndihmon në përmirësimin e sigurisë së aplikacionit dhe në parandalimin e incidenteve të mundshme të sigurisë ne pjesen e te dhenave ose gabime te shkaktuara nga njeriu.
4. **Optimizimi i resurseve:** Testimi mund të ndihmojë/ndihmonë në identifikimin e përdorimit te panevojshëm të burimeve të sistemit si procesori, memorja, dhe lëndët e tjera. Kjo ndihmon në optimizimin e resurseve dhe në uljen e koston e infrastrukturës, ose ne raste te tjera nese kemi te bjem me aplikacione te medha dhe komplekse at’here duhet ti shtojm resurset per nje optimizim sa me te mire te aplikacionit qe ndodhet nen proces te testimit.
5. **Identifikimi i problemeve para publikimit:** Testimi zbulon probleme dhe defekte para se aplikacioni të lansohet në treg. Kjo na lejon korrigjimin e problemeve në fazat e hershme të zhvillimit dhe redukton kostot dhe vonesat e mundshme.

Përmbushja e testeve të cilësisë dhe performancës është një hap kyç në zhvillimin e çdo aplikacioni. Kjo ndihmon në sigurimin e një produkti të aftë dhe përmbushës për nevojat e përdoruesve dhe të konkurrojë në një treg gjithnjë me produkte sa më cilësore.

# **PËRFITIMET E TESTIMIT AUTOMATIK**

4.1 Përshkrimi i testimit automatik

Testimi automatik ka një histori të gjatë, dhe ka filluar të përdoret që në vitet e hershme të kompjuterit. Kjo histori ka ndjekur zhvillimin e teknologjisë dhe nevojat e fushës së shfrytëzimit të mjeteve digjitale. Këtu janë disa momente të rëndësishme në historinë e testimit automatik:

1. **Vitet e 1940 dhe 1950:** Fillimet e testimit automatik lidhen me zhvillimin e kompjuterëve të hershëm. Gjuhët programuese si Fortran dhe Assembly u përdorën për të zhvilluar mjete për testimin automatik të softuerit. Këto ishin periudha kur kompjuterët filluan të përdoren në mënyrë më të gjerë në shkencë dhe industri për aplikacione të ndryshme.
2. **Viti 1979:** Përparimi i testimit automatik filloi të ndikojë më shumë në fushën e inxhinierisë së softuerit. "Mercury Interactive" mundesoj krijimin e nje mjedisin te testimit automatik të njohur si "WinRunner."
3. **Vitet e 1990:** Gjuhët programuese si C++ dhe Java u përdorën për të zhvilluar mjete të avancuara të testimit automatik. Këto mjete lejuan organizatat të zhvillojnë dhe të ekzekutojnë teste komplekse automatike.
4. **Vitet e 2000 dhe më pas:** Teknologjitë e reja, si Selenium dhe Appium, janë zhvilluar për të lejuar testimin automatik të aplikacioneve web dhe mobile. Këto teknologji janë zhvilluar duke përdorur gjuhë programuese të ndryshme si Python dhe JavaScript.
5. **Automatizimi i proceseve të DevOps:** Në dekadat e fundit, testimi automatik ka fituar një rol të rëndësishëm në kuadër të praktikave të zhvillimit të shpejtë si DevOps. Kjo është bërë e mundur përmes skriptimit të testeve automatike dhe integrimit të tyre në “CI/CD”.

Testimi automatik dhe testimi manual janë dy metoda të ndryshme për të verifikuar cilësinë dhe performancën e një aplikacioni ose softueri. Këtu do t'ju jap një përshkrim të metodës se testimit automatik dhe do të përmend karakteristikat kryesore te tije.

Testimi automatik është një metodë e cila përdor skripte të programuara ne gjuhe te ndryshme për ekzekutimin e testeve automatike për një aplikacion ose softuer. Këto skripte janë të shkruara paraprakisht dhe janë të cilësuara ose përgatitura për të verifikuar funksionalitetin e aplikacionit nëpërmjet një game të hollësishëm të veprimeve.

Përparësitë e testimit automatik përfshijnë:

1. **Shpejtësia dhe Efikasiteti:** Testet automatike ekzekutohen shpejt dhe pa ndërhyrje njerëzore, duke e bërë procesin më efikas.
2. **Riprodhimi i testeve:** Testet automatike mund të riprodhohen me konsistencë dhe mund të ekzekutohen në çdo moment.
3. **Gama e gjerë e testimit:** Testet automatike lejojnë testimin në game të gjerë dhe mund të përfshijnë shumë teste dhe skenarë te ndryshem.

Gjuhët programuese e kanë bërë të mundur testimin automatik dhe kanë ndikuar në zhvillimin e mjeteve të ndryshme për testimin automatik të softuerit. Përdorimi i gjuhëve të programimit për të zhvilluar skripte të testimit ka lejuar organizatat të automatizojnë shumicën e testeve të tyre dhe të sigurojnë cilësi më të lartë dhe efikasitet në zhvillimin e softuerit.

4.2 Shpjegimi i përfitimeve të testimit automatik

Testimi automatik është një praktikë e rëndësishme në zhvillimin e aplikacioneve, dhe ka shumë përfitime që i ofron kompanive te cilat e shfrytëzojnë atë. Këtu do të shpjegojmë disa nga këto përfitime kryesore:

4.2.1 Shpejtësia

Shpejtësia është një nga përfitimet kyçe të testimit automatik në kontekstin e zhvillimit të aplikacioneve. Ky përfitim ka ndikim të madh në mënyrën se si kompanitë zhvillojnë dhe lansojnë produkte të tyre. Disa aspekte të shpejtësisë në testimin automatik janë të rëndësishme:

1. **Ekzekutimi i shpejtë i testeve:** Skriptet e testimi automatik mund të ekzekutohen shpejt dhe pa ndërprerje. Në krahasim me testimin manual, ku çdo skenar testimi kërkon kohë dhe përpjekje të manuale, testimi automatik është jashtëzakonisht efikas dhe i shpejtë. Kjo do të thotë që ndryshimet në kodin e aplikacionit mund të testohen dhe verifikohen shpejt para lansimit.
2. **Testimet e përsëritur:** Testimi automatik lejon testet të ekzekutohen ne menyre te përsëritur, ndërsa ndryshimet bëhen në kodin e aplikacionit. Kjo është e rëndësishme në mjediset e zhvillimit të shpejtë, të tilla si zhvillimi i softuerit me metoden e zhvillimit Agile. Përmes testimit automatik, ndryshimet mund të testohen në çdo cikel të zhvillimit, duke përmirësuar cilësinë dhe stabilitetin e produktit.
3. **Identifikimi i shpejtë i problemeve:** Testimi automatik lejon identifikimin e shpejtë të problemeve dhe defekteve. Pas ekzekutimit të një testi, raportimi automatik i rezultateve identifikon problemet në mënyrë automatike. Kjo lejon ekipet e testimit të adresojnë problemet menjëherë, duke reduktuar koston dhe vonesën që lidhet me zbulimin e problemeve në fazat më të vona të zhvillimit.

Në përfundim, shpejtësia është një avantazh thelbësor i testimit automatik. Ajo ndihmon organizatat të ofrojnë aplikacione më cilësore në një kohë më të shpejtë, duke i lejuar ato të reagojnë më efikasitet në ndryshimet e kërkesave dhe zbulimin e defekteve.

4.2.2 Përmirësimi i cilësisë

Përmirësimi i cilësisë është një ndër përfitimet kryesore të testimit automatik në procesin e zhvillimit të aplikacioneve. Testimi automatik kontribuon në përmirësimin e cilësisë në mënyra të ndryshme:

1. **Konsistencë dhe përsëritshmëri:** Testimi automatik siguron që testet të ekzekutohen në mënyrë konsistente dhe përsëriten në mënyrë të vazhdueshme. Përdoruesit e nje softueri mund të kenë eksperienca të ndryshme rreth performancës dhe funksionalitetit të tij. Përmes testimit automatik, mund të sigurohet që aplikacioni është konsistent dhe funksionon pa ndonjë ndryshim të papritur në cilësi.
2. **Identifikimi i defekteve:** Testimi automatik ndihmon në identifikimin e shpejtë të defekteve dhe problemeve. Përmes skripteve të testimi automatik, aplikacioni testohet me dhena të ndryshme dhe në këtë mënyrë mund te shpërblehet me defektet që mund të shfaqen në kushte të ndryshme. Ky proces përmirëson cilësinë e aplikacionit duke identifikuar dhe adresuar problemet më herët në zhvillim.
3. **Kohëzgjatja e testeve:** Testimi automatik lejon ekzekutimin e testeve në kohëzgjatje të gjatë dhe të përsëritura pa humbur cilësi. Në krahasim me testimin manual, ku testet e gjata mund të jenë të lodhshme dhe të supozuara për gabime njerëzore, skriptet automatike mund të kryejnë këto teste pa humbur kujdesin dhe cilësinë.

Në përfundim, përmirësimi i cilësisë është një pasuri e rëndësishme që sjell testimi automatik në procesin e zhvillimit të aplikacioneve. Duke përdorur skripte të testimi automatik, kompanitë mund të sigurojnë që aplikacionet e tyre janë të konsistente, të përsëritura, dhe cilësore, duke adresuar defektet herët dhe duke përmirësuar përvojën e përdoruesit.

4.2.3 Rishfrytëzimi i burimeve

Rishfrytëzimi i burimeve është një avantazh tjetër i testimit automatik që ndikon pozitivisht në zhvillimin e aplikacioneve. Ky koncept lidhet me përdorimin e burimeve të disponueshme, përfshirë kohën, shkathtësitë e ekipit, dhe infrastrukturën teknike, në një mënyrë të zgjuar dhe efikase. Këtu janë disa mënyra se si testimi automatik rishfrytëzon burimet:

1. **Rishfrytëzimi i kohës:** Testimi automatik ndihmon në rritjen e efikasitetit të kohës së përdorur për testim. Pasi skriptet e testimi automatik mund të ekzekutohen në mënyrë automatike dhe të përsëriten në kohë shumë më të shpejtë, zhvilluesit dhe ekipet e testimi kanë më shumë kohë për të fokusuar në përmirësimin e cilësisë së produktit. Kjo rrit produktivitetin dhe lejon që burimet e kohës të përdoren më efikasitet.
2. **Rishfrytëzimi i shkathtësive:** Me testimin automatik, shkathtësitë e ekipit të testimi janë të rishfrytëzueshme në mënyrë të përsëritur. Pas zhvillimit të skripteve të testimi automatik, këto skripte mund të përdoren për shumë teste të ndryshme dhe për shumë versione të produktit. Kjo lejon që eksperienca dhe shkathtësitë e fituara nga ekipi të rishfrytëzohen në projekte të tjera dhe në fazat e tjera të zhvillimit.
3. **Rishfrytëzimi i infrastrukturës:** Për testimin automatik, është e nevojshme të krijohet një infrastrukturë e testimit që përfshin pajisje dhe mjete të automatizuara. Kjo infrastrukturë mund të përdoret në mënyrë të përsëritur për teste të ndryshme dhe aplikacione të ndryshme. Nese një herë infrastruktura është e krijuar mire, mund të rishfrytëzohet për shumë projekte të tjera pa shpenzime të mëdha shtesë.

Si perfundim, rishfrytëzimi i burimeve është një avantazh kyç i testimit automatik. Duke përdorur me efikasitet burimet e disponueshme, kompanit mund të zhvillojnë produkte cilësore, të përmirësojnë produktivitetin dhe të jenë të gatshme për sfidat e të ardhmes.

# **FRAMEWORKS DHE MJETET E TESTIMIT AUTOMATIK**

5.1 Përshkrimi i disa frameworks dhe mjete të përdorura për testimin automatik

Me poshte jane te listuara disa gjuhe programuese te cilat ofroj si struktura te llojllojshme per testimin automatik disa prej tyre do ti permendim me posht duke filluar nga me te njohurat dhe te perdorurat aktualisht ne treg.

Python dhe Robot Framework janë një kombinim i fuqishëm për testimin automatik. Python është një gjuhë programimi e njohur për sintaksën e thjeshtë dhe fuqinë e saj të gjerë, ndërsa Robot Framework është një framework i leht dhe fleksibil për menaxhimin e testeve automatike. Kombinimi i këtyre dy mjeteve ofron disa përfitime të rëndësishme në testimin automatik.

**Përdorimi i Python në testimin automatik**: Python është një gjuhë programimi e përshtatshme për zhvillimin e skripteve të testeve automatike për shkaq të avantazheve si në vijm:

1. **Sintaksa e thjeshtë:** Python ka një sintaksë të lexueshme që e bën të lehtë që të shkruhet dhe të kuptohen skriptet e testeve.
2. **Shumëllojshmëria:** Python përdoret gjerësisht në zhvillimin e aplikacioneve, kështu që është e lehtë të integrohet me projektin tuaj ekzistues.
3. **Komuniteti dhe Bibliotekat:** Python ka një komunitet të madh dhe librari të pasura të përdorura për testimin automatik. Për shembull, ‘*unittest’* dhe ‘*pytest’* janë dy librari të njohura për testimin në Python.
4. **Përdorimi i Python në testimin automatik**: Python është një gjuhë programimi e përshtatshme për zhvillimin e skripteve të testeve automatike për shkaq të avantazheve si në vijm:

**Robot Framework në testimin automatik**: Robot Framework është një framework i fuqishëm për menaxhimin e testeve automatike. Disa nga përfitimet kryesore të përdorimit të Robot Framework janë:

1. **Lehtësi e përdorimit:** Robot Framework është i njohur për sintaksën e thjeshtë që e bëjnë të lehtë shkrimin dhe leximin e testeve.
2. **Ekzekutimi i testeve në gamë të gjërë:** Mundëson ekzekutimin e testeve në mënyrë paralele në gamëtë madhe, duke përmirësuar efikasitetin.
3. I**ntegrimi i përmirësuar:** Robot Framework mund të integrohet me shumë mjete të tjerë dhe platforma të zhvillimit, përfshirë Jenkins, Docker dhe shërbime tjerat të testimit.

Ky kombinim i Python dhe Robot Framework është i fuqishëm dhe fleksibël dhe ofron një mjedis të përshtatshëm për zhvilluesit dhe inxhinierët e testimit për të krijuar dhe ekzekutuar teste automatike në mënyrë efikase.

Selenium dhe Java janë një tandem i fuqishëm për testimin automatik të aplikacioneve web. Selenium siguron mjetet dhe bibliotekat e nevojshme për të automatizuar veprimet në faqet e internetit, ndërsa Java është një gjuhë programimi e fuqishme dhe e njohur që mund të përdoret për të zhvilluar skripte të testeve automatike në Selenium. Kombinimi i këtyre dy mjeteve sjell disa përfitime të rëndësishme në testimin automatik.

Përdorimi i Selenium dhe Java në testimin automatik:

1. **Kontrollimi i shfletuesit ne web:** Selenium ofron një API të pasur për kontrollin e shfletuesve web të ndryshëm si Chrome, Firefox, dhe Safari. Java përdoret për të shkruar skriptet e testeve që përdorin këto API për të ekzekutuar veprimet në shfletues.
2. **Stabiliteti dhe Efikasiteti:** Përdorimi i Java për të zhvilluar teste automatike në Selenium sjell një nivel të lartë të stabilitetit dhe efikasitetit. Java është një gjuhë e kompiluar/përkthyes dhe ofron performancë të shkëlqyer në ekzekutimin e testeve.
3. **Bibliotekat dhe Komuniteti:** Java ka një gamë të pasur të librave dhe bibliotekave që mund të përdoren në testimin automatik. Përveç Selenium, mund të përdorni biblioteka të tjera të njohura si TestNG dhe JUnit për të menaxhuar dhe ekzekutuar testet në Java.
4. **Përputhshmëria:** Java është një gjuhë e gjerë që përdoret gjithandej në zhvillimin e aplikacioneve, duke lehtësuar integrimin e testeve automatike në projektin tuaj të zhvillimit.

Selenium dhe Java janë një kombinim i fuqishëm për testimin automatik të aplikacioneve web. Ky tandem ofron aftësinë për të automatizuar veprimet në faqet e internetit në mënyrë efikase dhe të sigurojë cilësinë dhe performancën e aplikacioneve tuaja.

Cypress dhe Javascript janë një kombinim i fuqishëm për testimin automatik të aplikacioneve web. Cypress është një framework modern i testimin automatik për aplikacionet e internetit, ndërsa Javascript është gjuhe programuese e përdorur për të zhvilluar skriptet e testeve automatike në Cypress. Ky kombinim sjell disa përfitime të rëndësishme në testimin automatik të faqeve të internetit.

Përdorimi i Cypress dhe JavaScript në Testimin Automatik:

1. **Ekzekutimi i testeve mbrenda ne shfletues:** Cypress integron një shfletues të brendshëm për ekzekutimin e testeve, duke lejuar teste të shpejta dhe të efikase. Javascript përdoret për të zhvilluar skriptet e testeve në këtë mjedis pune.
2. **Lehtësi për përdorimi:** Cypress është i njohur për sintaksën e thjeshtë dhe të kuptueshme që e bën të lehtë shkrimin dhe leximin e testeve. Javascript është gjithashtu një gjuhë me sintaksë të qartë dhe të lexueshme.
3. **Integrimi i leht:** Cypress lejon integrimin me shumë mjete dhe platforma të tjera të zhvillimit, përfshirë Jenkins, Git dhe shërbimet e tjera të testimit.
4. **Gjurmimi i defekteve:** Cypress vjen me mjetet e fuqishme të gjurmimit te defekteve, dhe Javascript është një gjuhë e njohur për mjetet e saj të zgjidhjes së problemeve.

Ky tandem i Cypress dhe Javascript ofron një mënyrë efikase dhe të përshtatshme për të automatizuar testimin e aplikacioneve web. Kjo e bën atë një zgjedhje të preferuar për zhvilluesit dhe inxhinierët e testimit që punojnë në faqet e internetit dhe dëshirojnë të sigurojnë cilësi dhe performancë të lartë për produktet e tyre.

5.2 Analiza e karakteristikave dhe avantazheve të secilit framework/mjet

Karakteristikat dhe Avantazhet e Robot Framework:

Karakteristikat:

1. **Sintaksa e thjeshtë:** Robot Framework ka një sintaksë të thjeshtë dhe lexueshme që e bën të lehtë shkrimin e testeve.
2. **Lehtësim gjatë përdorimit:** Është i njohur për lehtësinë e përdorimit dhe mundëson krijimit të testeve të shpejta dhe të thjeshta.
3. **Mundesi për integrim të gjerë:** Ka aftësi të mira për integrim me mjete të tjera të testimi dhe burime të hapura për komunitetin.

Avantazhet:

1. **Ndryshime të shpejta:** Lejon ndryshimin e testeve pa shumë vështirësi.
2. **Ekzekutim i testeve në masa të mëdha:** Mundëson ekzekutimin e testeve në mënyrë paralele, përmirëson efikasitetin dhe shpejtësinë e testimit.
3. **Komuniteti i madh:** Ka një komunitet të gjerë të zhvilluesve dhe mbështetësve që kontribuojnë në zhvillimin dhe përmirësimin e framework.

Karakteristikat dhe Avantazhet e Seleniumit:

Karakteristikat:

1. **Shumëllojshmëria e shfletuesve:** Mund të përdoret në mënyrë efektive për të testuar në shumë shfletues të ndryshëm, përfshirë Chrome, Firefox, dhe Safari.
2. **Përshtatshëmëria e gjuhëve të programimit:** Mund të përdoret me gjuhë programimi të njohura si Java, Python, C#, etj.
3. **Librari të pasura:** Ka një gamë të pasur të librarive dhe metodave që mund të përdoren për të kontrolluar veprimet në faqet web.

Avantazhet:

1. **Kompatibiliteti i shfletuesve:** Mund të sigurojë që aplikacionet janë kompatibile me shumë shfletues të ndryshëm.
2. **Proceset:** Mund të përdoret me integrimin e proceseve të zhvillimit, siç është CI/CD.
3. **Sintaksa e njohur:** Përdor sintaksën dhe metoda të njohura të programimit në gjuhët si Java dhe Python.

Karakteristikat dhe Avantazhet e Cypress:

Karakteristikat:

1. **Shpejtësia:** Cypress është një framework i shpejtë që ekzekuton teste në kohë reale.
2. **Shprehje të drejtpërdrejta:** Lejon përdorimin e shprehjeve të drejtpërdrejta për të identifikuar elementet në faqen web.
3. **Nëndërtimi i testeve në përgjithësi i lehtë:** Cypress është i dizajnuar që të jetë i lehtë për përdorim, me një sintaksë të thjeshtë dhe të kuptueshme.

Avantazhet:

1. **Testim i shpejtë dhe i efektshëm:** Cypress ofron ekzekutim të shpejtë të testeve dhe ndihmon në zbulimin e problemeve në kohë reale.
2. **Integrim me devTools:** Mund të integrohet me DevTools për të analizuar dhe menaxhuar veprimet në faqen web.
3. **Përdorim i këndshëm për zhvilluesit:** Përmban mjetet e nevojshme për zhvilluesit për të shkruar teste cilësore.

Përfundimisht, secila nga këto mjete mund të jetë i përshtatshme për projekte të ndryshme dhe varet nga nevojat specifike dhe përvoja e ekipit të zhvillimit. Gjithsesi, secili prej tyre ka karakteristikat dhe avantazhet e veta që mund të përdoren në përputhje me kontekstin dhe kërkesat e projekteve.

5.3 Zgjedhja e një frameworku/mjeti për testimin automatik në platformën e UBT-s

Një zgjedhje e shkëlqyer për testimin e platformës UBT dhe sistemit Moodle eshte robot framework i cili siguron mjete dhe burime të nevojshme për të zhvilluar dhe ekzekutuar teste të suksesshme automatike dhe për të siguruar cilësi dhe performancë të lartë gjatë testitmit të platformës të UBT-s.

1. **Popullariteti i madh i komunitetit të robot framework:** Robot Framework është një nga framework-ët më të njohur dhe të përdorur për testimin automatik. Ky fakt është shumë i rëndësishëm për një ambient si UBT, pasi ka një komunitet të madh të zhvilluesve dhe inxhinierëve të testimi që përdorin dhe kontribuojnë në këtë framework. Ky komunitet ofron një burim të pasur të dokumentacionit, librave, dhe përgjigjeve të pyetjeve që mund të përdoren për të zhvilluar teste të sofistikuara dhe për të zgjidhur problemet e mundshme gjatë testimin.
2. **Përqindja e madhe e testerëve që përdorin robot framework:** Robot Framework është një zgjedhje popullore në botën e testimin automatik. Shumë kompani dhe organizata e përdorin për testimin e aplikacioneve të tyre. Kjo do të thotë që ka një bazë të madhe të inxhinierëve të testimi dhe profesionistëve të testimit që kan njohuri ne robot framework.
3. **Përditësime të reja dhe të shpeshta nga ekipi i zhvilluesve të robot framework:** Robot Framework është një projekt i hapur me një ekip të përkushtuar të zhvilluesve. Përditësimet dhe përmirësimet janë të rregullta dhe janë në përputhje me zhvillimet më të fundit në botën e testimin automatik. Kjo do të thotë se robot framework është i gatshëm për të përballuar sfidat dhe ndryshimet e reja në industrinë e testimin. Për platformën Moodle të UBT-s, kjo do të thotë që testet automatike janë në gjendje të mbulojnë funksionalitetin dhe ndryshimet e këtij sistemi në mënyrë efikase.

# **TESTIMI NË PLATFORMEN MOODLE TË UBT'S**

6.1 Përshkrimi i procesitë të testimit në platformën Moodle

Përshkrimi i procesit të testimit në platformën Moodle është një hap i rëndësishëm që duhet të përfshijë disa elemente kyç. Këtu janë disa prej tyre:

1. **Ambienti i zhvillimit:** Për të filluar me testimin në platformën Moodle, është e rëndësishme të keni një ambient të zhvillimit të para përgatitur. Kjo përfshin një sistem operativ të përshtatshëm, dhe një IDE (Integrated Development Environment) për shkrimin e skripteve të testeve. IDE të njohura si Visual Studio Code, PyCharm, ose Eclipse janë zgjedhje të mira, une kam përzgjedhur Visual Studio Code.
2. **Gjuhët programuese:** Për të shkruar teste automatike në moodle me mjetin që ne kemi përzgjedhur, ne do të përdorim python për të automatizuar veprimet në faqe.
3. **Mjetet e testimit:** Pasi ne kemi zgjedhur një mjet për testimin automatik që do të përdorim për të shkruar dhe ekzekutuar testet. Duke pasur parasysh që kemi zgjedhur Robot Framework, do të duhet ta instaloni robot framework dhe ndonjë librari të nevojshme për të automatizuar veprimet në moodle. Në këtë rast, mund të jetë e nevojshme të instalojm edhe Selenium Library për Robot Framework.
4. **Node.js dhe npm (Node Package Manager):** Pasi kemi të ngjarë të përdorim ndonjë mjet ose skriptë shtesë për testimin automatik në Robot Framework, siç është "robotframework-seleniumlibrary," këto mund të bazohen në Node.js. Prandaj, është e rekomandueshme të keni Node.js dhe npm të instaluar në kompjuterin tuaj.
5. **Pip (Python Package Manager):** Nëse përdorni Python për të shkruar testet, ju mund të keni nevojë të instaloni librari të ndryshme ne Python. Pip është menaxheri i paketave i Python-it.

Pas instalimit të këtyre komponentëve dhe mjeteve të nevojshme, ju do të keni një ambient të gatshëm për të filluar me procesin e shkrimit, ekzekutimit dhe monitorimit të testeve në platformën Moodle. Kjo do të përfshijë krijimin e skripteve të testeve, konfigurimin e mjetit të testimi (Robot Framework në këtë rast), dhe ekzekutimin e testeve automatike për të vlerësuar performancën, cilësinë dhe funksionalitetin e platformës Moodle.

# **7. REFERENCAT**

|  |  |
| --- | --- |
| [1] | O. G. M. a. K. W. Jan Henrik Ziegeldorf, në *Privacy in the internet of things: Threats and challenges*, DOI:http://dx.doi.org/10.1002/sec.795, 2014, p. 2728–2742. |
| [2] | R. Z. W. X. W. Q. a. A. Z. Minqi Zhou, «Security and privacy in cloud computing.,» në *Proceedings of the 2010 6th International Conference on Semantics Knowledge and Grid (SKG’10).*, 2010. |
| [3] | K. F. a. E. B. F. Anton V. Uzunov, në *A comprehensive pattern-oriented approach to engineering security methodologies*, 2015, p. 217–247. |