LATVIJAS UNIVERSITĀTE
DATORIKAS FAKULTĀTE

Windows darbvirsmas lietotņu automatizācijas rīki

BAKALAURA DARBS

Autors: **Gļebs Veprevs**
Studenta apliecības Nr.: gv17020
Darba vadītājs: prof., Dr.sc.comp. Guntis Arnicāns

RĪGA 2021

ANOTĀCIJA

Automatizētā programmatūras testēšana pēdējo 10 gadu laikā ir piedzīvojusi ļoti strauju attīstību, izplatību un izpelnijusies plašu atzinību. Tagad gandrīz katra kompānija, kas izstrādā lietotnes, automatizē vismaz daļu no saviem pastāvīgi atkārtojamajiem testiem. Līdz ar to ir strauji pieaugusi arī pieejamo automatizētās testēšanas rīku izvēle. Kā rezultātā, rodas jautājums: kurš no visiem šobrīd pieejamajiem rīkiem tirgū ir visatbilstošākais konkrētā lietotāja vai kompānijas vajadzībām.

Autors savā bakalaura darbā apraksta un praktiski pielieto vairāk kā 10 gan maksas, gan bezmaksas automatizētās testēšanas rīkus un izvērtē tos pēc vairāk kā 20 dažādiem kritērijiem, kas palīdz daudz objektivāk un visaptverošāk novērtēt katru no uzskaitītajiem rīkiem.

Galvenais šī bakalaura darba pienesums un mērķis ir palīdzēt cilvēkiem atrast un izvēlēties, savām vai savas kompānijas vajadzībām, atbilstošāko Windows darbvirsmu lietotņu automatizētās testēšanas rīku.

Darba autors atrada un izanalizēja pieejamo dokumentāciju, informāciju par rīku internetā, kā arī praktiski uzinstalēja un pielietoja katru no tiem.

Bakalaura darba rezultātā veikta rīku aprakstīšana, izvērtēšana un salīdzināšana.

ATSLĒGVĀRDI: Automatizētā testēšana, Windows darbvirsmas lietotne, automatizētās testēšanas rīks, rīku salīdzināšana.

ABSTRACT

WINDOWS DESKTOP APPLICATION AUTOMATION TOOLS

Automated software testing has developed and gained widespread recognition over the past 10 years. Now, almost every company that develops applications automates at least part of their constanty repeatable tests. As a result, the choice of available automated testing tools has grown rapidly. At this moment, the question occurs: which one from all the tools currently available on the market is the most appropriate for the needs of a particular user or company.

In his bachelor's thesis, the author describes and practically uses more than 10 both paid and free automated testing tools and evaluates them according to more than 20 different criteria, which helps to evaluate each of the listed tools more objectively and comprehensively.

The main contribution and goal of this bachelor's thesis is to help people find and choose the most appropriate Windows desktop application automated testing tool for their own or their company's needs.

The author found and analyzed the available documentation, information about the tool on the Internet, as well as practically installed and used each of them.

As a result of the bachelor's thesis, the tools were described, evaluated and compared.

KEYWORDS: Automated testing, Windows desktop application, automation testing tool, tool comparison.

SATURS

APZĪMĒJUMU SARAKSTS

Agile - iteratīva pieeja projektu vadībai un programmatūras izstrādei.

AI (Artificial Intelligence) - mākslīgais intelekts, ko demonstrē mašīnas, atšķirībā no dabiskā intelekta, ko demonstrē cilvēki un dzīvnieki, kas ietver apziņu un emocionalitāti.

Akcepttestēšana - testēšana, kuru veic, lai noteiktu, vai nodevums apmierina vai neapmierina tā akceptēšanas kritērijus un dotu iespēju pasūtītājam izlemt, vai programmatūra/sistēma ir akceptējama.

ALM (Application Lifecycle Management) – lietotne izstrādājamā produkta dzīves cikla vadībai (pārvaldība, izstrāde un uzturēšana).

API (Application Programming Interface) - saskarne, kas nosaka mijiedarbību starp vairākām lietotnēm.

Atvērtā pirmkoda programmatūra - programmatūra, kura pirmkods ir brīvi pieejams apskatei, lietošanai un rediģēšanai ik vienam, kas to vēlas.

Back-End - lietotnes aizmugurējā daļa jeb viss, kas attiecās uz lietojumprogrammas servera pusi un tā mijiedarbību ar datubāzi.

Codeless tehnoloģija – koda/skripta rakstīšana bez fiziskas koda rakstīšanas. Būtībā ļauj praktiski ikvienam izveidot savas lietojumprogrammas, izmantojot veidnes un moduļus grafiskajās lietotāja saskarnēs.

CRM (Customer Relation Management) - process, kurā bizness vai cita organizācija pārvalda savu mijiedarbību ar klientiem, parasti izmantojot datu analīzi, lai izpētītu lielu informācijas daudzumu.

Dalītā testēšana - lietotāja pieredzes izpētes metodoloģija [1]. Dalītās testēšanas testi sastāv no randomizēta eksperimenta ar diviem variantiem - A un B. Dalītā testēšana ir veids, kā salīdzināt viena mainīgā divas versijas, parasti pārbaudot subjekta reakciju uz A variantu pret B variantu un nosakot, kurš no abiem variantiem ir efektīvāks.

Datubāžu vaicājumi - pieprasījums piekļūt datiem no datu bāzes, lai ar tiem manipulētu vai tos izgūtu.

DDT (Data-Driven Testing) - pieeja automatizētajā testēšanā (vienību, integrāciju, bet visbiežāk tā tiek izmantota back-end testēšanai), kuras laikā konkrētais tests spēj saņemt datu kopumu no ārējiem vai iekšējiem datu avotiem, un paredzamo rezultātu vai paredzamo stāvokli, ar kuru jāsalīdzina faktiskais rezultāts.

DevOps - darbību kopums, kas automatizē un integrē procesus starp programmatūras izstrādi un IT komandām, lai viņi varētu ātrāk, uzticamāk veidot, testēt un piegādāt programmatūru.

ERP (Enterprise Resource Planning) - galveno biznesa procesu integrēta pārvaldība, bieži vien reāllaikā un ar programmatūras un tehnoloģiju starpniecību.

Framework - platforma programmatūras un lietotņu izstrādei. Tas nodrošina pamatu, uz kura programmatūras izstrādātāji var veidot lietotnes noteiktai platformai.

Front-End – lietotnes priekšdaļa jeb viss ar ko mijiedarbojas lietotājs. No lietotāja viedokļa priekšējā daļa ir lietotāja saskarnes sinonīms.

Funkcionālā testēšana jeb melnās kastes testēšana - tiek testētas funkcijas ar ieejas datiem un pārbaudīts izejas rezultāts. Programmas iekšējā struktūra reti kad tiek apskatīta. Tiek salīdzināta programmas uzvedība pret specifikāciju [2].

GUI (Graphival User Interface) - interaktīvu vizuālo komponentu sistēma datoru programmatūrai. GUI parāda objektus, kas nodod informāciju un atspoguļo darbības, kuras var veikt lietotājs.

Haosa inženierija - pārbaudījumi eksperimentiem ar programmatūras sistēmu produkcijā, lai vairotu pārliecību par sistēmas spēju izturēt nemierīgus un negaidītus apstākļus/situācijas.

IDE (Integrated Development Environment) - ļauj programmētājiem konsolidēt dažādus programmatūras rakstīšanas aspektus.

iFrames - HTML dokuments, kas ir iekšā citā vietnes HTML dokumentā.

Laidiens/versija - Lietotājiem izplatāmais kārtējais programmatūras variants.

Log fails – fails ar log paplašinājumu, kurā ir notikumu ieraksts no noteiktas programmatūras un operētājsistēmas. Kaut arī tajos var būt vairākas lietas, žurnāla faili tiek bieži izmantoti, lai parādītu visus notikumus, kas saistīti ar sistēmu vai programmu, kas tās izveidojusi.

ML (Machine Learning) - mākslīgā intelekta pielietojums, kas nodrošina sistēmām iespēju automātiski mācīties un pilnveidoties balstoties uz iepriekšējo pieredzi.

Nepārtrauktā integrācija - prakse automatizēt koda izmaiņas no vairākiem izstrādātājiem vienā programmatūras projektā.

OCR (Optical Character Recognition) - pārveido attēlus vai pat rokrakstu tekstā. OCR programmatūra analizē dokumentu un salīdzina to ar datu bāzē saglabātajiem fontiem.

OS (Operating System) - programmatūras kopums, kas ir atbildīgs par datora aparatūras tiešu kontroli un vadību, kā arī par tādām pamatdarbībām kā lietojumprogrammatūras darbināšanu.

QA (Quality Assurance) - veids, kā novērst izstrādājamās lietotnes kļūdas un defektus un izvairīties no problēmām, piegādājot lietotni vai pakalpojumus klientiem.

Record and Playback - automatizētas testēšanas funkcija, kura reģistrē lietotāja darbību un pēc tam to atkārto un/vai pieraksta.

Regrestestēšana - iepriekš testētas programmas pārtestēšana ar mērķi pārliecināties, ka veikto izmaiņu rezultātā neparādījās jauni vai nezināmi defekti tajā funkcionālā apgabalā, kas netika mainīts.

REST (Representation State Transfer) - programmatūras arhitektūras stils, kurā tiek izmantota HTTP apakškopa. To parasti izmanto, lai izveidotu interaktīvas lietotnes, kas izmanto tīmekļa pakalpojumus.

RPA (Robotic Process Automation) - programmatūras tehnoloģija, kas atvieglo izveidi, izvietošanu un pārvaldību, pateicoties programmatūras robotam, kas līdzinās cilvēku darbībai un mijiedarbojoties ar digitālajām sistēmām/programmatūru.

SaaS (Software as a Service) - programmatūras licencēšanas un piegādes modelis, kurā programmatūra tiek licencēta uz abonēšanas pamata.

Slodzes testēšana - nefunkcionāls programmatūras testēšanas process, kurā lietojumprogrammas veiktspēja tiek pārbaudīta ar noteiktu paredzamo slodzi.

UI (User Inteface) - cilvēka un datora mijiedarbības un komunikācijas punkts ierīcē jeb grafiskā lietotāja saskarne. Jēdziens ietver sevī displeja ekrānus, tastatūras, peli un darbvirsmas izskatu.

Vienībtestēšana - process, kas ietver testēšanas plānošanu, testu kopas iegūšanu un testējamās programmatūras vienības mērīšanu, salīdzinot testējamo vienību ar tai noteiktajām prasībām. Vienības mērīšana ir datu paraugu izmantošana, lai noteiktu vai vienība izpilda tās darbības, kas specificētas PPS.

VM (Virtual Machine) - datorsistēmas emulācija. Virtuālās mašīnas ir balstītas datoru arhitektūras un nodrošina fiziska datora funkcionalitāti.

VPN (Virtual Private Network) - šifrēts pieslēgums pie interneta no ierīces uz tīklu. Šifrētais savienojums palīdz nodrošināt sensitīvu datu drošu pārsūtīšanu.

x86 arhitektūra – arhitektūras instrukciju komplekta datoru procesoriem. Intel Corporation izstrādātā x86 arhitektūra nosaka, kā procesors apstrādā un izpilda dažādas operētājsistēmas un programmatūras pārraidītās instrukcijas.

Xpath - galvenais elements XSLT standartā. XPath var izmantot, lai pārvietotos pa elementiem un atribūtiem XML dokumentā.

IEVADS

Pēdējā laikā tirgū pieejamo Windows darbvirsmas lietotņu automatizētās testēšanas rīku klāsts ir eksponenciāli pieaudzis, taču atvērts paliek jautājums: kurš no viņiem ir piemērotākais konkrētā lietotāja vai kompānijas vajadzībām? Uz doto brīdi pieejamie avoti un zinātniskie raksti, kurus autors izskatija pētnieciskās daļas ietvaros, sniedz ļoti skopu un nedetalizētu atbildi uz šo jautājumu, jo šajos rakstos ir izvērtēti tikai pāris rīki pēc dažiem kritērijiem, kas dod vien ļoti vispārēju novērtējumu un priekšstatu par rīku. Savukārt, autors savā darbā veic vairāk kā 10 rīku detalizētu analīzi un izvērtēšanu gan no teorētiskās, gan no praktiskās puses pēc vairāk kā 20 iepriekš predefinētiem kritērijiem.

Šobrīd autors testu automatizācijas un analīzēs procesā izmanto kompānijas ietvaros izstrādātu rīku, nevis pieejamo maksas/bezmaksas alternatīvu tirgū, tāpēc viena no šī darba pētāmajām problēmām ir noskaidrot, cik konkurētspējīgs un daudzfunkcionāls ir šis izstrādātājs rīks salīdzinājumā ar šobrīd populārajiem, tirgū pieejamajiem rīkiem.

Viens no šī bakalaura darba teorētiskās daļas mērķiem ir pastāstīt par automatizēto testēšanu, analizējot un uzskaitot tās galvenās priekšrocības un trūkumus, pastāstot par šāda veida testēšanas labās prakses iezīmēm, kā arī uzskaitot un apskaidrojot jaunākās tendences šajā jomā.

Otrs šī bakalaura darba teorētiskās daļas mērķis ir pastāstīt vairāk par automatizētās testēšanas rīkiem, vispārēji tos definējot, nodalot un grupējot tos pa dažādām kategorijām, atkarībā no testēšanas fāzes vai testēšanas veida, kā arī veikt vispārēju ieskatu Windows darbvirsmas lietotnēs, to testēšanas unikālajās iezīmēs un automatizētās testēšanas atšķirībās salīdzinot ar tīmekļu lietotņu testēšanu.

Pirmais šī bakalaura darba praktiskās daļas mērķis ir atrast, izanalizēt un praktiski uzinstalēt un pielietot, izpildot kādu konkrētu testpiemēru, katru no vairāk kā 10 automatizētās testēšanas rīkiem, kas būs detalizēti aprakstīti šī darba pamatdaļā, kā arī katru no tiem, pēc iespējas objektīvāk, izvērtēt pēc vairāk nekā 20 dažādiem kritērijiem 5-baļļu sistēmā, tādā veidā nosakot, pēc visu kritēriju vidējā svērtā vērtējuma, visaugstāk novērtēto un labāko rīku no šī darba uzskaitītajiem. Kā arī šī mērķa sasniegšanas ietvaros autors izvirza hipotēzi par to, ka bezmaksas automatizētās testēšanas rīki būs daudz sliktāki, mazfunkcionālāki un līdz ar to arī zemāk novērtēti nekā maksas rīki.

Otrais šī bakalaura darba praktiskās daļas mērķis ir salīdzināt visaugstāk, šajā darbā, novērtēto automatizētās testēšanas rīku ar rīku, kurš ir izstrādāts un tiek pielietots vienas kompānijas ietvaros, kurā strādā un ikdienā lieto šo rīku darba autors. Tas tiks darīts, lai saprastu un izanalizētu izstrādātā rīka vājās puses un svarīgākās/kritiskākās lietas un

funkcijas, kuras nepieciešams implementēt vai uzlabot, lai nākotnē šo rīku varētu virzīt un pārdot Eiropas tirgū kā lētāku un labāku alternatīvu esošajiem rīkiem.

Darba sākumā jeb nodaļās 1-4. ir izklāstīta darba teorētiskā daļa: pastāstīts par automatizēto testēšanu, tās priešrocībam un trūkumiem, labo praksi un jaunākajām tendencēm tajā, kā arī par Windows darbvirsmas lietotnēm un to automatizētās testēšanas rīkiem. Darba praktiskajā daļā jeb nodaļās 5-7. tiek plašāk pastāstīts par katra rīka izvirzītajiem novērtēšanas kritērijiem, kā arī tiek detalizēti aprakstīts katrs no rīkiem, praktiski uzinstalēts, pielietots un novērtēts pēc vairāk kā 20 kritērijiem. Pēc visu rīku novērtēšanas seko Robots rīka salīdzinājums ar labāko no šajā darbā uzskaitītajiem rīkiem, kā arī autora secinājumu kopums par to, kam ir jābūt standarta automatizētās testēšanas maksas/bezmaksas rīkā.

1. TESTĒŠANAS AUTOMATIZĀCIJAS ĪKI

Automatizācija sasaista vienā sistēmā automātiskās vadības sistēmas, automātikas elementus, programmēšanu un tehnoloģisko procesu ar minimālu cilvēka iejaukšanos. Automatizācijas procesi ir dinamiski, komplicēti un savstarpēji cieši saistīti ar inženiertehniskām problēmām. Automatizācijas galvenie mērķi ir palielināt ražīgumu un produktivitāti; samazināt produkcijas pašizmaksu; iegūt atkārtojamu kvalitāti; nodrošināt vairāk informācijas par procesu vai objektu; uzlabot darba apstākļus [26].

Automatizētā testēšana ir programmatūras testēšanas veids/metode, kas ietver iepriekš uzrakstīta konkrēta testa gadījumu skripta izpildi, izmantojot automatizācijas rīku. Automatizētā testēšana tiek izmantota, lai samazinātu kopējo testēšanas laiku, kā arī, lai notestētu gadījumus, kuru veikšana manuāli varētu būt pārāk sarežģīta vai pat neiespējama.

Testēšanas automatizācijas rīks ir rīks/programma, ar kuras palīdzību testētājs nodrošina testa gadījumu skriptu izveidošanu un palaišanu, atkļūdošanu, izpildi un analīzi ar pēc iespējas mazāku cilvēku mijiedarbību, kā arī palīdz gan nelielām komandām, gan lielām organizācijām automatizēt programmatūras testēšanu, tādējādi panākot lielāku ātrumu, uzticamību un efektivitāti [17].

Lai sāktu automatizēto testēšanu, ir jāizvēlas atbilstošs rīks. Tā kā ir ļoti daudz dažādu pieejamo testēšanas rīku, vajadzībām atbilstošākā rīka izvēle nav atkarīga tikai no viena vai pāris lietām. Izvēloties testēšanas rīku, ir jāpārbauda daudzas lietas, īpašības un faktorus, lai izvēlētākos vispiemērotāko automatizētās testēšanas rīku komandas vajadzībām un iecerēm.

Tālāk ir aprakstīti divi veidi pēc kuriem varētu kategorizēt dažādus automatizētās testēšanas rīkus.

Zemāk ir aprakstīts pirmais veids jeb rīku kategorizācija pēc testēšanas fāzes:

- **Vienības fāze.** Šajā fāzē tiek pārbaudīti atsevišķi programmatūras komponenti vai vienības, un šīs fāzes automatizācijas rīki ir vērsti un specializējas tieši uz konkrētu komponentu pārbaudi, tomēr šī ir ļoti šaura fāze, kurā reti kad kaut ko automatizē, tāpēc rīku, kas specializējas vienīgi uz šīs fāzes ir tiešām maz.
- **API fāze.** API sasaista visas pārējās sistēmās, kas nepieciešamas programmatūras darbībai. Šīs fāzes pārbaudi parasti veic pēc programmatūras izstrādes, lai pārliecinātos, ka viss kopā darbojas tā kā paredzēts, un automatizācijas rīku, kas specializējas konkrēti uz šīs fāzes ir daudz.
- **UI fāze.** Šīs fāzes testi ir paredzēti, lai precīzi atkārtotu tipisku lietotāju darbības un pieredzi ar sistēmu. Šāda veida testi nodrošina, ka gala produkts, ar kuru mijiedarbojas lietotāji, darbojas tā, kā vajadzētu un gala produkts ir viegli lietojams un

intuitīvi saprotams. Mūsdienās, ir kļuvuši ļoti izplatīti un populāri automatizācijas rīki, kas atbild tieši par šīs fāzes izpildi/testēšanu.

Vēlētos piebilst, ka mūsdienās bieži vien automatizācijas rīki ir spējīgi automatizēt testa gadījumus un notestēt visu nepieciešamo jebkurā no šīm fāzēm, tāpēc šī kategorizācija, iespējams, uz doto brīdi nav pati aktuālākā.

Zemāk ir aprakstīts otrs veids kā iespējams kategorizēt automatizācijas rīkus – pēc testēšanas veida:

- **Dūmu testēšana.** Dūmu tests ir funkcionāls tests, kas nosaka, vai konkrētais būvējums ir stabils. Tas pārbauda būtiskāko sistēmas funkciju darbību, lai pārliecinātos, ka programma var izturēt turpmāku testēšanu. Šādu testēšanas veidu atbalsta daudz automatizētās testēšanas rīki, piemēram PhantomJS un Selenium.
- **Integrācijas testēšana.** Integrācijas testi ir funkcionāli testi, kuros tiek pārbaudīts un nodrošināts, ka visi atsevišķi lietotnes testa elementi darbojas pareizi mijiedarbojoties savā starpā. Šāda veida testēšana ir diezgan specifiska un to atbalsta lielākoties rīki, kas sākotnēji bija vērsti uz šāda veida testēšanu, kā piemēram DBUnit, Tessy un citi.
- **Regrestestēšana**. Regresijas testi ir gan funkcionāli, gan nefunkcionāli, Tie nodrošina, ka pēc izmaiņu veikšanas neviena programmatūras daļa nav regresējusi. Šāda veida testēšanu atbalsta ļoti daudz rīku, piemēram TestComplete, UFT un citi, kā arī šis ir viens no fundamentālākajiem testēšanas veidiem, kuru būtu jāatbalsta automatizācijas rīkam.
- **Drošības testēšana.** Drošības testi aptver funkcionālos un nefunkcionālos testus, kas pārbauda programmatūru uz iespējamajām ievainojamībām. Tie atklāj programmatūras vājās vietas un palīdz tās laicīgi novērst. Šāda veida rīki arī ir diezgan specifiski, un parasti, šāda veida testēšanu augstā līmenī atbalsta rīki, kas sākotnēji bija vērsti uz šāda veida testēšanu, kā piemēram ZAP, Wfuzz un citi.
- **Veiktspējas/slodzes testēšana**. Nefunkcionāli veiktspējas testi novērtē sistēmas stabilitāti un responsivitāti. Tie nodrošina, ka programmatūra spēj tikt galā ar stresa situācijām, kad sistēmā ir ļoti augsta aktivitāte, un nodrošina labāku lietotāja pieredzi. Šis ir vēl viens diezgan specifisks testēšanas veids, kuru atbalsta ne visi automatizācijas rīki, bet vairāk tie, kas specializējas šajā testēšanas veidā, kā piemēram Apache Jmeter, LoadUI un citi.
- **Akcepttestēšana**. Akcepttesti ir funkcionālie testi, kuru mērķis ir noteikt, kā lietotāji reaģēs uz galaproduktu. Šis testa veids ir veiksmīgi jānokārto, pirms lietotni var izlaist produkcijas vidē un piegādāt galalietotājiem. Šāda testēšanas veida automatizāciju piedāvā ļoti daudz dažādi rīki, kā piemēram Katalon, Ranorex un citi.

- **Vienībtestēšana.** Vienībtestēšanas rīki pārbauda/validē, ka atsevišķas koda vienība/funkcija/bloks strādā korekti. Daži šādu rīku piemēri ir JUnit un TestNG, kas ir balstīti uz Java programmēšanas valodas.
- **Funkcionālā testēšana.** Šie ir rīki, kas ļauj pārbaudīt kopējo funkcionalitāti (pārliecināts, ka lietojumprogrammas uzvedība ir pareiza un tā darbojas kā noteikts dokumentācijā). Izmantojot šāda veida rīku, automatizētie skripti var tikt izveidoti, lai apstrādātu lietojumprogrammas funkcionālās izmaiņas. Daži no funkcionālās testēšanas rīkiem spēj veikt arī regrestestēšanu. Daži šādu rīku piemēri ir Selenium, Sahi un citi.

Ir arī daudzi citi svarīgi faktori, kas jāņem vērā rīka izvēles brīdī, piemēram rīka papildus funkcijas, kurām ir jāpievērš pastiprināta uzmanība:

- Testēšanas veidu daudzums, ko atbalsta rīks.
- Dažādas rīkā pieejamās papildfunkcijas.
- Vairāku pārlūkprogrammu, OS un programmēšanas valodu atbalsts.
- Integrācija ar citiem testēšanas pārvaldības rīkiem.

Kopumā kritēriju un lietu, kurām pievērst uzmanību izvēloties piemērotāko rīku automatizētai testēšanai ir tiešām daudz.

Darba 5-7. nodaļās darba autors daudz detalizētāk izskata vairākus automatizētās testēšanas rīkus gan no teorētiskās, gan no praktiskās puses, kā arī nodefinē striktus kritērijus katra rīka novērtēšanai.

2. AUTOMATIZĒTĀ TESTĒŠANA

2.1. Automatizētās testēšanas definīcija

Automatizētā programmatūras testēšana jeb manuālo testu automatizācija ir programmatūras testēšanas veids, kas tiek realizēts, izmantojot īpašus papildus programmatūras testēšanas rīkus, kas ir nošķirti no programmatūras, ko testē, lai spētu uzreiz izpildīt veselu testu gadījumu kopu un nodrošinātu izstrādātās programmatūras kvalitāti.

Ir pieejami ļoti daudz dažādi rīki, kuri palīdz testētājiem automatizēt viņu testus/skriptus, samazinot laiku un izmaksas. Automatizācijas testēšanas rīkā var ievadīt testa datus pārbaudāmajā sistēmā, ziņot par atgriezto rezultātu, veikt dažāda veida validācijas jeb salīdzināt sagaidāmos ar faktiskajiem rezultātiem un veidot sistēmas ekrānuzņēmumus uz validācijas brīdi, ģenerēt detalizētus testēšanas pārskatus jeb log failus, kā arī kļūdu ziņojumu failus [1].

Automatizētie testi nevar pilnībā aizvietot manuālo testēšanu, tomēr tie var uzskatāmi uzlabot testējamā produkta, un pašas testēšanas kvalitāti, kā arī samazināt izmantojamo resursu daudzumu testēšanas nolūkiem.

Testu automatizācijas mērķis ir atvieglot testētāju darbu un samazināt sistēmas moduļu, atsevišķu komponentu vai pat visas sistēmas testēšanas laiku. To izmanto, lai automatizētu pastāvīgi atkārtojamus uzdevumus (akcepttestēšanā un regrestestēšanā), uzdevumus, kurus var viegli nosimulēt programmatūra, taču ir gandrīz neiespējami nosimulēt cilvēkam, kā piemēram, liela lietotāju daudzuma simulēšana slodzes testiem, un citus testēšanas uzdevumus strādājot ar sarežģītiem projektiem ar vairākām komponentēm un attīstītu funkcionalitāti.

Lai ieviestu automatizēto testēšanu projektā, nepieciešams daudz laika un resursu patēriņa, taču ja runājam par ilggadējiem projektiem, kur nepieciešams lietot regresa testēšanu, tad testu automatizācijas ieviešana ir neatņemama un neapstrīdama lieta.

2.2. Automatizētās testēšanas priekšrocības, trūkumi un labā prakse

2.2.1. Automatizētās testēšanas priekšrocības

Testēšana, it sevišķi regresa testēšana, parasti ir ļoti ilgs un laikietilpīgs process, it īpaši gadījumos, kad programmatūras/lietotnes arhitektūra un funkcionalitāte ir ļoti plaša un bieži tiek izlaistas jaunas versijas, vai atsevišķi versiju laidieni, tieši tāpēc automatizētās testēšanas ieviešana palīdz uzlabot produkta kvalitatīvi, ātri un efektīvi notestēt visu nepieciešamo programmatūru, bez pārlieku lielām izmaksām un darba kvalitātes zaudēšanas, kā tas bieži notiek manuālās testēšanas gadījumos, kad darbs ir nogurdinošs un viendabīgs.

Tālāk šajā sadaļā tiks detalizēti apskatītas automatizētās testēšanas galvenās priekšrocības.

2.2.1.1. Kvalitātes zaudēšanas novēršana

Gadījumā ar manuālo testēšanu, ļoti bieži kaut kādas kļūdas un nepilnības tiek pieļautas cilvēciskā faktora dēļ, un tas ir pašsaprotami, jo pēc daudzu testpiemēru izpildes un analizēšanas, testētājs nav spējīgs visu laiku strādāt ar pilnu atdevi un koncentrēšanos, līdz ar ko palielinās varbūtība pieļaut kļūdu vai nepamanīt kādu nepilnību. Tieši šis ir galvenais iemesls dēļ kā krītas testēšanas kvalitāte. No otras puses, ja mēs ņemam "mašīnu" jeb automatizācijas rīku, tad šādas kļūdas pieļaušanas iespēja sarūk līdz minimumam, jo rīks, ar kuru tiek testēts nekad nenogurst un vienmēr vienādi izpilda vienus un tos pašus testus, un pārbauda rezultātus, respektīvi veicot automatizēto programmatūras testēšanu, "cilvēciskais faktors" vairs neietekmē testēšanas kvalitāti [3].

2.2.1.2. Ātrums, efektivitāte un produktivitāte

Manuālā testēšana ir ilgs un daudz resursu prasošs process, savukārt automatizēto testu gadījumā skripta scenārijs tiek izveidots vienu reizi, palaists vairākas reizes un mazliet modificēts izstrādes gaitā, un skripts vienmēr izpildīsies ātrāk nekā testētājs to pašu izdarīs manuāli.

Manuālo testēšanu veic testētāji savā darba laikā, savukārt automatizētos testus var palaists atrodoties jebkurā pasaules nostūrī, kā arī Jūs tos varat palaist jebkurā laikā, piemēram ieplānojot automatizēto testu izpildi uz vakaru, nakti, lai darba laikā pievērstos citiem uzdevumiem, bet atnākot nākamajā rītā uz darbu, Jūs uzreiz saņemtu rezultātus par testu norisi, kļūdām un citām lietām [3].

Efektivitātes ziņā, kad testētājs atkārto kādu no scenārijiem manuāli, tad viņš tajā brīdī var nodarboties tikai ar šī scenārija izpildi, savukārt testējot automatizēti, testētājam ir iespēja

palaist testēšanas skriptu, piemēram uz virtuālās mašīnas caur attālināto piekļuvi, un skripta izpildes laikā paralēli veikt citus svarīgus darbus [3].

2.2.1.3. Atkārtojamība

Automatizētos testus var palaists vairākas reizes, kas ir īpaši noderīgi un svarīgi ieviešot sistēmā jaunu funkcionalitāti. Testu automatizēšana neprasa testētājiem ik reizi sākt visu testēšanu no paša sākuma pēc katras jaunas funkcionalitātes ieviešanas sistēmā [3].

2.2.1.4. Izmantojamo resursu ietaupīšana

Ir situācijas, kad testētājs nevar izpildīt vienu vai vairākus testus, jo testa izpildei ir nepieciešams noteikts lietotāju vai aparatūras vienību skaits (parasti tas attiecās uz slodzes testiem), bet izstrādātājam nav pieejas šādiem resursiem, tieši tādos gadījumos testēšanas automatizācijas rīks var tikt pielietots, lai nosimulētu nepieciešamo situāciju (lietotāju vai aparatūras skaitu), līdz ar to samazinot/ietaupot reālos fiziskos resursus [17].

2.2.1.5. Ieguldījumu atdeves pieaugums un izmaksu samazinājums

Ir vajadzīgi lieli sākotnējie kapitālieguldījumi automatizētajā testēšanā. Tieši šī iemesla dēļ daudzi uzņēmumi joprojām izmanto tikai manuālo testēšanas pieeju, taču prakse un pieredze liecina, ka ilgtermiņa perspektīvā šāda investīcija vairākkārt attaisno ieguldījumus tajā [3].

2.2.1.6. Pārklājuma dziļuma pieaugums

Testēšanas automatizācijas izmantošana palīdz aptvert lielāku testu pārklājumu un izgūt no tā vairāk lietderīgas informācijas, kā piemēram atmiņas saturu, failu saturu un datu tabulas, kā arī iekšējo programmatūras stāvokli, kurā viņa atrodas testa laikā. Tādā veidā šie testi paaugstina produkta kvalitāti un gala rezultāta mēs iegūstam kvalitatīvāku programmatūru [17].

Varam secināt, ka testu automatizācija manāmi paātrina un atvieglo testētājiem testēšanas procesu, taču ir svarīgi atcerēties, ka to visu var panākt tikai tad, ja rūpīgi un laicīgi plāno, un maksimāli efektīvi vada testēšanas procesu.

2.2.2. Automatizētās testēšanas trūkumi

Automatizētas testēšanas ieviešana nevar pilnībā atrisināt laika, kvalitātes un resursu trūkumu problēmas. Reālajā dzīvē neefektīva automatizētā testēšana var, tieši otrādi, novest pie vēl lielāka laika un resursu trūkuma, jo automatizētās testēšana ir pareizi jāievieš un pareizi jāizplāno tās process, tāpēc zemām ir aprakstīti automatizētās testēšanas trūkumi un nepilnības, kas var rasties, ja testēšanas process nav pareizi izplānots.

2.2.2.1. Programmatūras pastāvīga mainība

Programmatūrā/lietotnē pastāvīgi notiek dažāda veida izmaiņas, tāpēc gadījumā ar automatizētajiem skriptiem tajos tieši tā pat pastāvīgi ir jāveic izmaiņa un tie jāpapildina. Vēl jo vairāk, ja pēc jaunajām izmaiņām kāds skripts nestrādā, tad no sākuma ir viennozīmīgi jānosaka, ka skripts nestrādā dēļ jaunajām izmaiņām, nevis citu iemeslu dēļ un tikai tad jālabo skripts. Tāpēc, lai ieekonomētu laiku un resursus, kas ir galvenais automatizācijas mērķis, automatizēto testpiemēru testēšanu jāorganizē tādā veidā, lai būtu iespējams ātri un efektīvi saprast kāpēc skripts nestrādā, un veikt izmaiņas tajā.

2.2.2.2. Nepatiesa pārliecība par kvalitāti

Automatizēti testi pārbauda tikai to, uz ko tie tika ieprogrammēti. Tests var noritēt veiksmīgi, taču kāds defekts (visbiežāk vizuāls) var palikt nepamanīts, un tas notiek tāpēc, ka skripts netika ieprogrammēts uz to, lai noķertu šāda veida defektu. Rezultātā defekts var palikt nepamanīts līdz pat nodošanai klienta vidē, kad to ieraudzīs klients un tas, visticamāk, novedīs pie izstrādātāju reputācijas pasliktināšanās un papildus neapmaksāta darba [4].

2.2.2.3. Skriptu izveidei nepieciešamais laiks.

Ir svarīgi saprast, ka skriptu izveide, it īpaši lielās, daudzfunkcionālās sistēmās ir ļoti sarežģīta. Ir jābūt noteiktām zināšanām par sistēmu, tās komponentēm un skriptu rakstīšanu kā tādu, tieši tāpēc laba un droša skripta izstrāde prasa daudz laiku, kā arī skriptu izstrādei veltītais laiks nav kļūdu atklāšanai veltītais laiks [18].

2.2.2.4. Automatizācija nenozīmē testēšana

Automatizēti testi ir tikai uzprogrammēti soļi kāda scenārija veikšanai, tā nav kaut kāda mašīna, kurai var iedot programmu un cerēt, ka tiks paveikta pilnīga sistēmas testēšana un izveidota testēšanas dokumentācija. Daudzi, kuri pirmo reizi saskaras ar automatizāciju, grib automatizēt visus testus, lai atbrīvotos no manuāliem testētājiem. Patiesībā tas nav iespējams,

jo pastāv īpatnības un lietas, kuras automatizēti testi nevar notestēt. Bez tam, automatizētā testēšana neveic testu plānošanu, paša automatizētā skripta uzrakstīšanu, skriptu izpildi un rezultātu analīzi. To visu manuāli veic testētājs, tāpēc automatizāciju nevar uzskatīt par testēšanu, un ar to nevar aizstāt testētājus [4].

2.2.2.5. Neuzticamība rīkam un automatizācijas procesam

Automatizēti testi var jebkurā brīdī pārstāt strādāt, piemēram dēļ programmatūras īpatnībām, kļūda var notikt pašā testēšanas rīkā, un šādā gadījumā to būs diezgan pagrūti identificēt, kā arī testi var pēkšņi izbeigt izpildi daudzu iemeslu dēļ. Varam minēt pāris no tiem: jebkādas izmaiņas interfeisā, servera nokrišana/pārslogotība, problēmas ar interneta savienojumu, VM noslogotība konkrētos momentos. Visu šo un daudzu citu iemeslu dēļ automatizētajai testēšanai nevar uzticēties pilnībā un paļauties tikai uz to [18].

2.2.3. Automatizētās testēšanas labā prakse

Strādājot ar automatizēto testēšanu ir ļoti iespējams saskarties ar visdažādākajām problēmām, kuras ir iespējams apiet vai pilnībā izvairīties no tām, ja automatizētā testēšana tiek balstīta uz sekojošiem pamatprincipiem jeb labo praksi:

2.2.3.1. Veidot un uzturēt skriptus pēc iespējas vienkāršākus un īsākus

Bieži vien, veidojot skriptus, testētāji pievieno vairākus vienas un tās pašas funkcijas izsaukumus, katru reizi testējot savādākas ieejas datu kombinācijas jeb padodot savādākus ieejas datus, tādā veidā vienā skriptā mēģinot notestēt vairākus dažādus testpiemērus. Tā nav laba prakse, jo tādā veidā, ja pats funkcijas izsaukums ir uzrakstīts kļūdaini, tad skripts nokritīs jau pie pirmā no viņiem, un pārējās funkcijas un izsaukumi netiks izpildīti. Tieši šī iemesla dēļ skriptus ir jāveido pēc iespējas īsākus, taču ne pārlieku īsus, jo tādā gadījumā sistēma būs pārslogota ar skriptu skaitu [18].

2.2.3.2. Veidot un uzturēt skriptus neatkarīgus

Pilnīgi nevienam skriptam nevajadzētu būt atkarīgam no cita skripta, tikai gadījumos ja vienā skriptā notiek kaut kāda darbība, kas ietekmē veselu skriptu kopu, piemēram pamatdatu ievade, kuri būs nepieciešami pilnīgi visiem nākošajiem skriptiem viena biznesa scenārija ietvaros. Būtībā katru skriptu ir jāspēj izpildīt individuāli, bez citu skriptu iepriekšējas darbināšanas. Tas ir vajadzīgs, jo gadījumā, ja viens tests izpildīsies neveiksmīgi, tad tas pilnīgi nekā neietekmēs citu testu darbību un pareizību un neapstādinās procesu [18].

2.2.3.3. Veidot testus ar nemainīgu rezultātu

Izveidotajam skriptam, izpildoties atkārtoti vairākas reizes ar vieniem un tiem pašiem parametriem un ieejas datiem ir jāsniedz viens un tas pats rezultāts. Ja testēšanas laikā neizpildās šis nosacījums, tad būs pastāvīgi nepieciešams veikt daudz izmaiņu un mērījumu atkarībā no rezultāta, kuru vēlēsimies iegūt [17].

2.2.3.4. Testu pārklāšanās minimizēšana

Ja eksistē vairāki skripti, kuri testē vai veic pārbaudes vienā un tajā pašā sistēmas daļā vai komponentē, tad, augot skriptu izmēriem un to skaitam, nevajadzīgi un lieki palielināsies laika izmaksas jeb citiem vārdiem sakot: viens un tas pats funkcionalitātes apgabals tiks vairākkārt testēts bez vajadzības [17].

3. JAUNĀKĀS TENDENCES AUTOMATIZĒTAJĀ TESTĒŠANĀ

Arvien vairāk un vairāk dažādu nozaru organizāciju sāk atzīt QA automatizācijas nozīmi savu produktu panākumos, līdz ar to QA automatizētās testēšanas popularitāte turpina aug un šī nozare manāmi attīstīties. Tajā pašā laikā pieaug arī kopējā testēšanas dzīves cikla produktivitāte un efektivitāte pateicoties jaunu, gudrāku risinājumu ieviešanai, kā arī automatizētās testēšanas rīku evolūcijai un attīstībai.

Šajā nodaļā tiks aprakstītas jaunākās tehnoloģijas un paņēmieni automatizētajā testēšanā jeb lietas, kuras var saukt par nākamo gadu tendencēm automatizētajā testēšanā.

3.1. Mākslīgais intelekts un mašīnmācīšanās

Mākslīgais intelekts un mašīnmācīšanās (AI/ML) automatizētajā programmatūras testēšanā nav kas pilnīgi jauns, un tas ir ticis izmantots jau iepriekš, taču joprojām ir ļoti liela vieta attīstībai šajā nozarē.

AI padara testēšanu gudrāku. Domāju, ka AI tiks atrasts pielietojums pilnīgi visas automatizētās testēšanas jomās, bet it īpaši svarīga loma AI būs analīzei un dažādu pārskatu veidošanā, piemēram:

- Žurnāla analīze: AI identificēs unikālus testa gadījumus, kuriem nepieciešama automatizēta testēšana.
- Automatizēto testu komplektu optimizācija: AI palīdzēs atklāt automatizēto testu pārklāšanos un novērst liekus, nevajadzīgus testa gadījumus [19].
- Paredzamā analīze: AI spēs prognozēt galalietotāju uzvedības specifiku un identificēt lietas/jomas programmā, kurām jāpievērš pastiprināta uzmanība.

Otra lieta, uz kura balstās viedā automatizācija, ir mašīnmācīšanās. Tiek paredzēts, ka 2021. gadā ML sasniegs daudz augstāku "brieduma pakāpi". Saskaņā ar Capgemini pasaules kvalitātes ziņojumu, 38% organizāciju bija plānojušas īstenot ML projektus jau 2019. gadā, taču eksperti prognozē, ka līdz 2021. gada beigām šis skaitlis manāmi pieaugs [5].

Kas attiecas uz testēšanas automatizācijas rīku izstrādātājiem, viņiem jākoncentrējas uz praktisku rīku izveidi. Labam AI atbalstāmam rīkam ir jābūt gan ne pārāk dārgam, gan jāietver sevī vairāki tehniskie aspekti, piemēram, programmu logu lasīšana, testa scenāriju ģenerēšana vai reaģēšanu uz programmatūras darbībām.

3.2. Bezkoda automatizācijas testēšana

Bezkoda testēšana galvenokārt ir saistīta ar skriptu izveidi automatizācijai, neuzrakstot nevienu koda rindiņu. Pieeja ir nepārtraukti attīstījusies līdz ar tehnoloģiju jauninājumiem,

galveno uzmanību pievēršot kodēšanas procesa samazināšanai un tā padarīšanu par lietotājam draudzīgāku. Bezkoda automatizētās testēšanas galvenais mērķis ir padarīt procesu pietiekami vieglu, lai ietaupītu skriptu rakstīšanai patērēto laiku, vienlaikus gandrīz neprasot no testētāja ar kodēšanu saistītas zināšanas un iemaņas. Tā kā šie testa gadījumi ir izstrādāti bez koda, tad tie ir skaidri un viegli lasāmi/saprotami arī cilvēkiem, kuriem nav iepriekšēju zināšanu par kodēšanu. Bezkoda testa gadījumus bez iepriekšējām padziļinātām zināšanām var pārbaudīt pat netehniskie uzņēmuma/projekta darbinieki [19].

Šim nolūkam šobrīd tirgū var atrast dažādus rīkus, no kuriem lielākā daļa testētājiem vienkārši nodrošina abstrakcijas slāni, kas ir lietotājam draudzīgāks un ar labu grafisko saskarni. Tas padara testēšanas procesu vieglāku un vadāmāku.

Kopēji var secināt, ka bezkoda automatizācijas process ir labs ieguldījums. Tas neprasa, lai testētāji mācētu kodēt, un uz šo pozīciju nav jāpieņem darbā kodēšanas profesionāļi, tādējādi ietaupot izmaksas un resursus [19].

3.3. Robotizēta automatizētā testēšana (RPA)

Kas ir RPA? Tā vietā, lai automatizētu funkcionālo testēšanu, jūs darbināsiet un automatizēsiet dažādus datu apstrādes skriptus. Pieņemsim, ka jūs strādājat pie apdrošināšanas maksājumu sistēmas, tas ir, dokumentu pārvaldības sistēmas. Tradicionāli, automatizējot procesus dokumentu pārvaldības sistēmās programmatūras izstrādātājs izveido darbību sarakstu, lai automatizētu uzdevumu un mijiedarbotos ar iekšējo sistēmu, izmantojot API vai īpašu skriptu valodu. Turpretī RPA sistēmas automatizēti izstrādā darbību sarakstu, novērojot, kā lietotājs izpilda uzdevumu sistēmas grafiskajā lietotāja saskarnē (GUI), un pēc tam automatizē šo procesu, atkārtojot šos soļus un uzdevumus tieši iekš GUI. Tātad jūs varat izmantot RPA, lai kontrolētu kādu šī procesa daļu, vai visu sistēmu vienlaikus. Automatizējot populārākos biznesa procesus un scenārijus, jūs mazināt cilvēcisko kļūdu rašanos. RPA ar funkcionālo automatizāciju ir daudz līdzību, izņemot to, ka tā vairāk attiecas uz biznesa funkciju automatizēšanu, nevis jebkuras funkcionalitātes pārbaudi [19].

4. WINDOWS DARBVIRSMU LIETOTNES

4.1. Windows darbvirsmu lietotņu definīcija

Windows darbvirsmas lietotnes ir pilnīgas lietotnes, kas darbojas neatkarīgi no citām lietojumprogrammām un neprasa, lai tās darbotos paralēli. Lai strādātu ar tādu lietotni, ir nepieciešami noteikti datora aparatūras resursi, kā arī pati lietojumprogramma ar strādājošu funkcionalitāti [7].

Šāda veida lietotnes atrodas uz lietotāja datorā. Tām nav vajadzīgs nepārtraukts savienojums ar internetu, tās komunicē ar lietotāju izmantojot grafisko interfeisu un Windows API, tām ir augstāka veiktspēja, jo tās ir neatkarīgas no ārējiem faktoriem, piemēram interneta savienojuma kā tāda un tā ātruma, savukārt tās ir visai atkarīgas no pielietotās operētājsistēmas, mūsu gadījumā Windows, un pieprasa šīs konkrētās lietotnes instalēšanu uz katra atsevišķa lietotāja datora, kurš vēlas ar to strādāt. Par šāda veida lietotnēm tiek uzskatīti sākotnēji iebūvētie teksta redaktori, tādi kā MS Word, Notepad, video un audio atskaņotāji, piemēram Windows Media Player, dažādas aprēķina lietotnes, kā piemēram kalkulators. Kopumā var teikt, ka visas lietotnes, kas ir instalētas un uzstādītas uz datora, kurš strādā uz Windows OS, ir Windows darbvirsmas lietotnes.

Šāda tipa lietotnēs, lietotājiem vienmēr ir piekļuve lietotnes sistēmas failiem, tieši tāpēc šādas programmas ir lielā mērā neaizsargātas no lietotāja kļūdām un uzvedības, un ir pilnībā atkarīgas no lietotāja darbībām [7].

4.2. Windows darbvirsmu lietotņu unikālās iezīmes

IT nozares profesionāļi bieži vien, savās zinātņu publikācijas, terminus lietojumprogramma, programma un darbvirsmas lietotne uzskata par savstarpēji aizstājamiem terminiem jeb citiem vārdiem sakot: uzskata, ka tas ir viens un tas pats. Tas tā ir tāpēc, ka visi šie trīs termini ir sinonīmi pēc to teorētiska un praktiska pielietojuma: tradicionālā Windows lietojumprogramma, kuru instalējat un izmantojat pielietojot peli un tastatūru, tieši tāpat kā tas tiek darīts kopš Windows pirmajām versijām.

Zemāk ir aprakstītas vissvarīgākās Windows darbvirsmas lietotņu īpašības un unikālās iezīmes:

- Šādas lietotnes parasti labi darbojas tikai ar peles un tastatūras palīdzību un sliktāk darbojas ar pieskāriena ievadi.
- Sākot ar Windows 10 un Windows 8 OS darbvirsmas lietotnes darbojas ar ierobežotām atļaujām, taču lietotājs tām var piešķirt administratīvās atļaujas, kas uzreiz paaugstina lietotnes izmantošanas iespējas. Dažas lietotnes, piemēram,

standarta pretvīrusu programma, nevar darboties korekti, ja tai nav administratīvās atļaujas.

- Vairākas vienādas vai atšķirīgas Windows lietotnes var darboties paralēli. Piemēram, vienu un to pašu Windows darbvirsmas lietotni ir iespējams atvērt divas vai vairāk reizes, cik nepieciešams, un paralēli strādāt ar katru atvērto atsevišķo instanci.
- Pārsvarā Windows darbvirsmas lietotnes var izmantot jebkurā Windows versijā: Windows 10, Windows 8 vai Windows 7. Tomēr gadās arī tā, ka dažas lietotnes, nav saderīgas ar vecākām Windows versijām, jo tās sākotnēji netika uz to ieprogrammētas.
- Dažas lietotnes, pēc lietotāja atļaujas, var instalēt papildus Windows pakalpojumus, kas šīm lietotnēm piešķir piekļuvi dažādiem sistēmas resursiem un veic sarežģītākus uzdevumus lietotāja labā. VPN un pretvīrusu programmas ir visizplatītākie Windows darbvirsmas lietotņu piemēri, kas instalē papildus Windows pakalpojumus.
- Windows darbvirsmas lietotnes parasti var instalēt ļoti dažādos veidos, caur dažādiem avotiem, piemēram: vietnēm, instalācijas diskiem, USB zibatmiņas diskiem un citiem atmiņas nesējiem, kuros ir vajadzīgie instalācijas faili.
- Visbiežāk lietotājs var manuāli vai automātiski atjaunināt programmatūru, izmantojot īpašus atjaunināšanas pakalpojumus, ko izveidojis lietotnes izstrādātājs, vai arī izmantojot trešo pušu lietotnes pastāvīgai atjaunināšanai.
- Windows darbvirsmas lietotņu atjauninājumi ne vienmēr var būt bezmaksas. Šīs lietotnes izstrādātājs var iekasēt papildus maksu no lietotājiem par lietojumprogrammas atjaunināšanu uz jaunāku versiju.
- Windows darbvirsmas lietotnēm var būt jebkurš licencēšanas modelis/veids: sākot no patentētiem modeļiem beidzot ar bezmaksas un atvērtā koda licencēm.
- Šāda veida lietotnes darbojas sistēmās ar Intel un AMD procesoriem, izmantojot sistēmas x86 arhitektūru. Šādas lietotnes nevar strādāt ar ARM procesoriem, piemēram tādiem, kas ir viedtālruņos, ja vien netiek izmantota kaut kāda veida virtualizācija.

4.3. Windows darbvirsmu lietotņu testēšana iezīmes

Darbvirsmas lietotņu automatizētā testēšana ir automatizētās testēšanas prakse, kas pārbauda lietotnes funkcionalitāti, drošību, lietojamību un stabilitāti pēc tās instalācijas uz konkrētā datora [7].

Automatizēti testējot Windows darbvirsmas programmas pastiprināta uzmanība jāpievērš instalācijas un atinstalēšanas testiem, lai pēc iespējas plašāk noklātu svarīgākos testēšanas aspektus, kā arī pilnībā ievērotu testēšanas prasības.

Windows darbvirsmu lietotņu automatizētā testēšana ir vienkāršāka, ja tā notiek iekš konkrēta tīkla, kur ir viennozīmīgi un nemainīgi noteikts tā serveru un klientu skaits.

Windows darbvirsmas lietotni normālos apstākļos izmanto viens lietotājs, kas ir ielogojies Windows operētājsistēmā. Tas nozīmē, ka testētājam ir jāuzkonfigurē un jāsagatavo attiecīgā vide, lai pārbaudītu, testētu un pārraudzītu rezultātus.

Tiek uzskatīts, ka Windows darbvirsmas lietotnēs ir daudz vieglāk izveidot plašu testu pārklājumu, kā arī automatizēt tos, jo automatizētās testēšanas rīks var piekļūt lietotnei pa tiešo.

4.4. Automatizētās testēšanas atšķirības tīmekļa un Windows darbvirsmas lietotnēs

Lietotnēs pēdējo divdesmit gadu laikā ir novērota milzīga attīstība ne tikai attiecībā uz to, ko var paveikt ar programmatūru, bet arī attiecībā uz to, cik lielā mērā lietotnes var piegādāt līdz galalietotājiem [8]. Windows lietotnes, piemēram, standarta Microsoft Office, tagad ir pieejams arī kā tīmekļa lietotne Office 365, savukārt tīmekļa lietotne Amazon, tagad ir pieejama galalietotājam arī kā mobilā lietotne.

Automatizējot testēšanu katrā no šīm lietotnēm, neskatoties uz to, vai tā ir viena un tā pati lietotne, kuras vienlaicīgi darbojas dažādās platformās vai pavisam cita, neatkarīga lietotne dažādās platformās, ir konkrēti jāsaprot katras uzskaitītās platformas unikālā nozīme un iezīmes, tās funkcionālās un nefunkcionālās prasības attiecībā pret lietotāju vai nepieciešamo aprīkojumu, kā arī ierobežojumi.

Zemāk ir aprakstītas un uzskaitītas fundamentālās atšķirības veicot tīmekļa un Windows darbvirsmas lietotnes automatizēto testēšanu.

4.4.1. Mijiedarbība un savienojamība

Tīmekļa lietotnes ir pieejamas ik vienam un tās pārsvarā tiek uzturētas uz kāda noteikta tīmekļa servera. Savukārt Windows darbvirsmas lietotnes vienmēr darbojas tikai uz datoriem vai darbstacijām, kurām var piekļūt "pa taisno" no viena konkrēta datora, ja vien netiek pielietotas papildus virtuālās mašīnas, CRM vai ERP, kurām vienmēr nepieciešama aktīva datu bāze vai interneta pieslēgums to korektam darbam [24].

Tīmeklī bāzētas lietotnes vienmēr ir vairāk atkarīgas un pakļautas kādai konkrētā lietotāja izvēlētajai pārlūkprogrammai, tāpēc tīmekļa lietotnēs ir fundamentāli svarīgi veikt automatizēto testēšanu pārbaudot lietotnes darbību uz vairākām atšķirīgām pārlūkprogrammām un fiksēt to darbību uz katras no viņām. Neko līdzīgu nav nepieciešams veikt Windows darbvirsmas lietotnēs. Kā arī, tīmekļa lietotnēs ir aktuāla UI pielāgošana dažādām ierīcēs, precīzāk to ekrānu izmēriem un dažkārt pat izšķirtspējai, sākot ar lieliem datoru monitoriem un beidzot ar dažādu modeļu telefoniem, tāpēc arī šis aspekts tīmekļa lietotnēs ir jātestē, savukārt Windows darbvirsmas lietotnēm parasti nekas tāds nav jāiestrādā un jātestē standarta gadījumos [24].

4.4.2. Lietotņu ieviešana un modernizācija

Liela starpība starp darbvirsmas un tīmekļa lietotņu automatizēto testēšanu ir tūlītējs izmaiņu atspoguļojums. Ja izmaiņas tiek veiktas tīmeklī, tās tiek tūlītēji parādītas visiem lietotājiem. Darbvirsmas lietotņu izmaiņas netiek parādītas līdz brīdim, kad lietotājs uzinstalē jaunāko versiju [8].

4.4.3. Testēšana ar un bez interneta savienojuma

Tīmekļa lietotnes ir pilnīgi atkarīgas no tā, vai ir pieejams interneta savienojums, lai tās vispār varētu atvērt un ar tām strādāt, savukārt Windows darbvirsmas lietotnes lielāko daļu savas funkcionalitātes var nodrošināt bezsaitē jeb nepieslēdzoties internetam [8].

Šī faktora dēļ, darbojoties un testējot tīmekļa lietotnes, jāņem vērā, ka būs nepieciešams veidot papildus akcepttestēšanas scenārijus, lai nosegtu un notestētu scenārijus, kuros tiek paredzēta pēkšņa stabila interneta savienojuma pazušana , un analizēt rezultātus jeb saprast, kā tas kopēji ietekmē lietotnes funkcionalitāti un darbību, un saprast vai konkrētā lietotne pareizi strādā arī pēc tam, kad interneta savienojums atkārtoti kļūst stabils. Protams, ka arī ar darbvirsmas lietotnēm šāda veida testi ir jāveido un jāveic, taču tie nav tik kritiski svarīgi [8].

4.4.4. Unikāli katras lietotnes aspekti

Darbvirsmas lietotnes:

- Lietotne darbojas vienā līmenī jeb citiem vārdiem sakot: gan front-end, gan back-end daļa atrodas vienā un tajā pašā vietā. Tas nozīmē, ka manāmi tiek atvieglota automatizētā testēšana un skriptu rakstīšana.

- Vienlaicīgi ar sistēmu/lietotni var darboties tikai viens konkrēts lietotājs jeb viens OS konts, kurā šobrīd konkrētais lietotājs ir autorizējies. Tas nozīmē, ka ja ir nepieciešams veikt automatizēto slodzes testu, lai pārbaudītu cik stabili strādā un kā uzvedās lietotne pārmērīgas slodzes laikā, tad šāda veida darbība ir mākslīgi jāsimulē.

- Ir ātra un droša piekļuve lietotnes failu sistēmai.

Tīmekļa lietotnes:

- Ir skaidri nodalīta infrastruktūra.

- Tā kā tīmekļa veida lietotnēm ir publisks API, tad tas dod iespēju testētājiem izveikt automatizēto slodzes testēšanu, pielietojot HTTP pieprasījumus, un šajā gadījumā nav vajadzības nepieciešamās darbības, kas vajadzīgas slodzes testam, mākslīgi simulēt.

- Daudzas grūti izsekojamas un automatizēti notestējamas problēmas, piemēram, aparatūras saderība, pārlūka saderība, versiju savietojamība, drošības problēmas, veiktspējas problēmas [9].

5. WINDOWS DARBVIRSMU LIETOTŅU AUTOMATIZĀCIJAS RĪKU NOVĒRTĒJUMA KRITĒRIJI

Viens no galvenajiem šī bakalaura darba mērķiem ir ļoti rūpīgi un detalizēti gan no teorētiskās, gan no praktiskās puses izpētīt Windows darbvirsmas lietotnēm paredzētos automatizētas testēšanas rīkus un pēc iespējas objektīvāk izvērtēt katru no tiem balstoties uz vairākiem kritērijiem, atbilstību jaunākajām tendencēm, kā arī paša darba autora subjektīvo viedokli pēc katra rīka praktiskas pielietošanas. Šo analīzi un izvērtēšanu var uzskatīt par vienu no lielākajiem šī bakalaura darba pienesumiem, jo kaut arī internetā var atrast dažus avotus, kur tiek izskatīti un izvērtēti tieši Windows darbvirsmas lietotnes automatizētās testēšanas rīki, tomēr izskatot tos detalizētāk, var viegli pamanīt, ka šajos avotos parasti tiek norādīti tikai daži testēšanas rīki, kas tiek analizēti tikai no teorētiskās puses un tiek objektīvi izvērtēti vien pēc pāris kritērijiem. Savukārt šī bakalaura darba ietvaros tiek analizēti daudz vairāk dažādi rīki pēc ļoti plašu kritēriju klāsta, kā arī tie tiek izvērtēti gan no teorētiskās, gan no praktiskās puses, lai varētu daudz objektīvāk noteikt labāko automatizētās testēšanas rīku. Autors uzskata, ka šāda veida plaša un padziļināta vairāku rīku analīze un izvērtēšana palīdzēs ikvienam lietotājam, ja runa iet tieši par Windows darbvirsmas lietotņu automatizācijas rīku, nekļūdīgi izvēlēties vislabāko un efektīvāko testēšanas rīku savām vai kompānijas vajadzībām.

Vēl viens šī darba pienesums, kas skar tikai kompāniju, kurā es šobrīd strādāju, ir manā kompānijā izstrādātā Windows darbvirsmu lietotņu automatizētās testēšanas rīka padziļināta analīze, izvērtēšana un salīdzināšana ar citiem populāriem un daudzfunkcionāliem rīkiem, kas šobrīd pieejami tirgū, lai saprastu vai izstrādātājs rīks atbilst automatizēto rīku kopējiem standartiem, tajā ir visa nepieciešamā funkcionalitāte un to var uzskatīt par pilnīgu rīku, kā arī, lai objektīvi izvērtētu šo rīku kā konkurētspējīgu alternatīvu populāriem un izplatītiem automatizētās testēšanas rīkiem. Visa šī analīze beigās palīdzēs saprast vai šo alternatīvo rīku nākotnē ir vērts attīstīt, virzīt un pārdot Latvijas un Eiropas tirgū.

Tieši šo divu galveno iemeslu dēļ ir ļoti svarīgi izveidot paplašinātu un daudzveidīgu sarakstu ar kritērijiem, pēc kuriem tiks vērts katrs no šajā darbā uzskaitītajiem rīkiem. Šie kritēriji un katra rīka novērtēšana atbilstoši tiem palīdzēs salīdzināt automatizētās testēšanas rīkus un dot tiem objektīvu novērtējumu.

Darba autors izskatija un izanalizēja vairāk kā 10 dažādus avotus, zinātniskos rakstus un publikācijas, no kurām aizguva lielāko daļu kritēriju, kas atspoguļoti šajā darbā, kā arī papildus ieviesa un aprakstija, pēc autora domām, trūkstošos kritērijus vispusējai rīka izvērtēšanai un apkopoja tos vienā tabulā šī darba ievaros. Līdz ar to, var uzskatīt šo tabulu

par vienu no šī darba pienesumiem, jo šāda veida kritēriju tabula ir unikāla un apskatāma tikai šī darba ietvaros. Darba autors centās piemeklēt tādus kritērijus, kurus ir iespējams izvērtēt maksimāli objektīvi, kā arī tie tika izvēlēti tādā veidā, lai noklātu un novērtētu pēc iespējas vairāk dažādus rīka aspektus: dažāda veida atbalsts (programmēšanas valodu, OS, lietotņu, atkļūdošanas, testēšanas tipu, datubāžu), iebūvētie rīki, kopējā funkcionalitāti un atsevišķas funkcijas, integrācijas, dokumentācija, tiešsaites atbalsts un tā tālāk.

Katrs uzskaitītajs kritērijs tiks vērtēts pēc 5 baļļu sistēmas, kur 5 – Izcili (rīks pilnībā atbilst norādītajam kritērijam), 4 – Labi, 3 – Vidēji, 2 - Slikti un 1 – Ļoti slikti (rīks pilnībā neatbilst norādītajam kritērijam), kā arī katra rīka novērtējuma beigās tiks izrēķināts aritmētiskais vidējais (saņemto vērtējumu kopsumma dalīta ar kritēriju skaitu). Šis aritmētiskais vidējais arī tiks uzskatīts par konkrētā rīka gala vērtējumu. Visi kritēriji, pēc kuriem tiks vērtēti un salīdzināti rīki, un to detalizēti skaidrojumi ir apskatāmi *tabulā 5.1.:*

5.1. tabula. **Testēšanas rīku vērtēšanas kritēriji**

Kritērijs	**Paskaidrojums**
Programmēšanas valodu atbalsts	Vai rīks ļauj rakstīt automatizētos testus dažādās programmēšanas valodās, kā piemēram C#, Java, Python, Ruby, JavaScripts un citās?
Iebūvēts UI inspekcijas rīks	Vai rīkā ir iebūvēts UI inspekcijas rīks, kas ļauj inspektēt UI iekšējās īpašības (*internals*), vai arī tas balstās uz kādu trešās puses rīku?
Nepieciešamās kodēšanas prasmes	Cik lielas kodēšanas prasmes nepieciešamas, lai izmantotu un strādātu ar šo rīku?
Iebūvētas ALM integrācijas	Vai rīks integrējās ar ALM rīkiem, kā piemēram Jenkins, TFS, Azure un citiem?
Pieejama dokumentācija	Vai pieejama dokumentācija par rīku ir gana plaša, saprotama, viegli atrodama un visaptveroša?
Rīka instalēšanas un lietošanas vienkāršums	Vai rīku ir gana viegli uzinstalēt un lietot ikdienā?
Testu skriptu izpilde	Cik korekti un ātri rīks izpilda uzrakstīto testa skriptu?
Cena	Vai tas ir bezmaksas, atvērtā pirmkoda rīks? Ja tas ir maksas, vai tas piedāvā bezmaksas izmēģinājuma iespēju?
Record and playback funkcija	Vai rīkā ir iespējams ierakstīt noteiktas darbības

	un to secību, un pēc izpildes šīs darbības atkārtot? Cik šāda veida funkcionalitāte ir parocīga un ērta lietošanā?
Lietotāja saskarne	Autora subjektīvs novērtējums par to, cik rīka saskarne ir ērta, parocīga un pašsaprotama priekš gala lietotāja. Cik laba ir grafiskā ziņā gala lietotāja saskarne? Cik ērti un parocīgi ir atrast vajadzīgās izvēlnes un pogas?
Operētājsistēmu atbalsts	Vai konkrētais rīks atbalsta un var darboties uz vairākām operētājsistēmām, vai tomēr tā funkcionalitāte ir vērsta vienai konkrētai OS?
Lietotņu atbalsts	Vai konkrētais rīks atbalsta un var darboties ar vairākiem lietotņu veidiem, kā piemēram darbvirsmas, tīmekļa, mobilā, vai tomēr ir vērsts uz kādu konkrētu lietotņu tipu?
Rīka apmācību process	Vai ir viegli/grūti iemācīties darboties ar šo rīku? Vai apmācības process ir ilgs? Vai ir vajadzīgas kāds iepriekšējas priekšzināšanas testēšanā, lai iemācītos strādāt ar šo rīku?
Rīka tiešsaistes atbalsts	Autora subjektīvais viedoklis, pēc praktiskas komunikācijas ar atbalsta personālu, par to, kādā laikā tas ir pieejams, cik labs, ātrs, profesionāls un izpalīdzīgs ir rīka tiešsaistes atbalsts.
Atkļūdošanas atbalsts	Vai rīkā ir iestrādāts mehānisks, kas ļauj noķert kļūdu, to atkārtoti izpildīt, noteikt rindu kodā un konkrēto vai iespējamo iemeslu, kāpēc kļūda radusies?
Atskaišu un logu izveide un glabāšana	Vai tiek veidotas atskaites un glabāti logi? Cik labi izprotama un lasāma ir rīka uzģenerētā testēšanas atskaite?
Dažādu testēšanas tipu atbalsts	Cik labi konkrētais rīks ir piemērots un spēj korekti veikt darbības un strādāt vairākos testēšanas tipos?

Datubāžu lietotņu atbalsts	Cik labi konkrētais rīks sadarbojas/strādā kopā ar datubāzes aplikācijām, piemēram, pielietojot MySQL izsaukumus konkrētajā automatizācijas skriptā?
Data-driven testēšana	Vai testēšanas rīks vispār spēj akceptēt un ievadīt datus no ārējiem datu failiem un saglabāt tos kā mainīgos testa skriptos? Vai šos ievadītos var saglabāt arī pašās lietotnēs? Un cik dažādos formātos to ir iespējams izdarīt? Vai rīkā ir paredzēta tabulu testēšana, kas nolasa datus no ievadītās tabulas, piemēram Excel, un validē tos datus atbilstībā pret sagaidāmajiem? Cik šis process ir vienkāršs un pašsaprotams?
Rīka evolūcija ar gadiem	Vai rīkam pastāvīgi ir jauni laidieni ar jaunu un uzlabotu funkcionalitāti? Vai rīkā pastāvīgi tiek uzlabotas un pievienotas papildu lietas vieglākai testu automatizācijai? Vai rīka dokumentācija arī tiek papildināta? Vai rīkā pastāvīgi tiek novērstas kļūdas un nepilnības?
Koda/skriptu atkārtota izmantojamība	Vai ir iespējams uzglabāt atsevišķu funkciju vai pat veselu funkciju koda bloku kā bibliotēku un līdzīgos veidos, lai pēc tam būtu vieglāk to atkārtoti izmantot un uz tiem atsaukties?
Atbilstība jaunākajām tendencēm	Vai rīks tiecās/atbilst jaunākajām tendencēm automatizētajā testēšanā, kas tika uzskaitītas šajā rakstā?

6. RĪKU PĀRSKATS

Automatizācijas rīki dalās atšķirīgās kategorijās: gan pēc to iebūvētā funkcionāla, gan pēc konkrēto lietotņu vai operētājsistēmu atbalsta un vēl pēc ļoti daudzveidīgiem faktoriem.

Gūstot padziļinātu priekšstatu, no vairākiem izskatītajiem avotiem, par Windows darbvirsmas lietotņu automatizētās testēšanas rīkiem, šī darba autors ir izvēlējies savā bakalaura darbā detalizētāk apskatīt gan no teorētiskās, gan no praktiskās puses tālāk minētos maksas rīkus: TestComplete, Unified Functional Testing, Katalon Studio, Squish, Tricentis Tosca, Ranorex, TestArchitect, kā arī atvērtā pirmkoda bezmaksas rīkus: WinAppDriver, SikuliX, AutoIt, Winium, Robots (manā kompānijā izveidotais rīks). Darba autors izvēlējās šos 12 rīkus, balstoties uz to meklējumu skaita lielākajos meklēšanas vietnēs (Google, Bing, Yahoo, Yandex), atrodamo materiālu skaita šajās pašās vietnēm, kā arī pieminējumu skaita nopietnos avotos un pieminējumu skaita citos avotos, kur notika vairāku rīku salīdzināšana.

Autors šī bakalaura darba ietvaros izvēlējās gan atvērtā pirmkoda rīkus, kurus var uzinstalēt un lietot pilnīgi bezmaksas, gan maksas rīkus, lai labāk saprastu cik liela nozīme ir rīka cenai attiecībā uz rīka funkcionalitāti, dokumentāciju un infrastruktūru.

Darba autors aprakstīs katru no uzskaitītajiem augstāk rīkiem, izdalot rīka stiprākās un vājākās puses, tā atbilstību jaunākajām tendencēm, kā arī, pēc iespējas objektivāk, izvērtēs rīka atbilstību vai neatbilstību izvirzītajiem kritērijiem. Katra rīka pārskata beigās autors izrēķinās rīka arifmētisko vidējo vērtējumu, 5 baļļu sistēmā, iepriekš detalizēti novērtējot katru atsevišķu kritēriju, kas aprakstīti *tabulā 5.1.*

Pēc šīs analīzes, autors atbildēs uz jautājumu, ko ietver sevī standarta Windows darbvirsmas lietotnes automatizētās testēšanas rīks, kā arī izvēlēsies labāko rīku no izanalizētajiem (to varēs noteikt pēc vidējā vērtējuma), un salīdzinās šo rīku ar alternatīvu rīku, kas ir izstrādāts un, pagaidām, tiek lietots tikai vienas firmas ietvaros, kurā strādā šī darba autors.

6.1. TestComplete rīks

6.1.1. Rīka apraksts

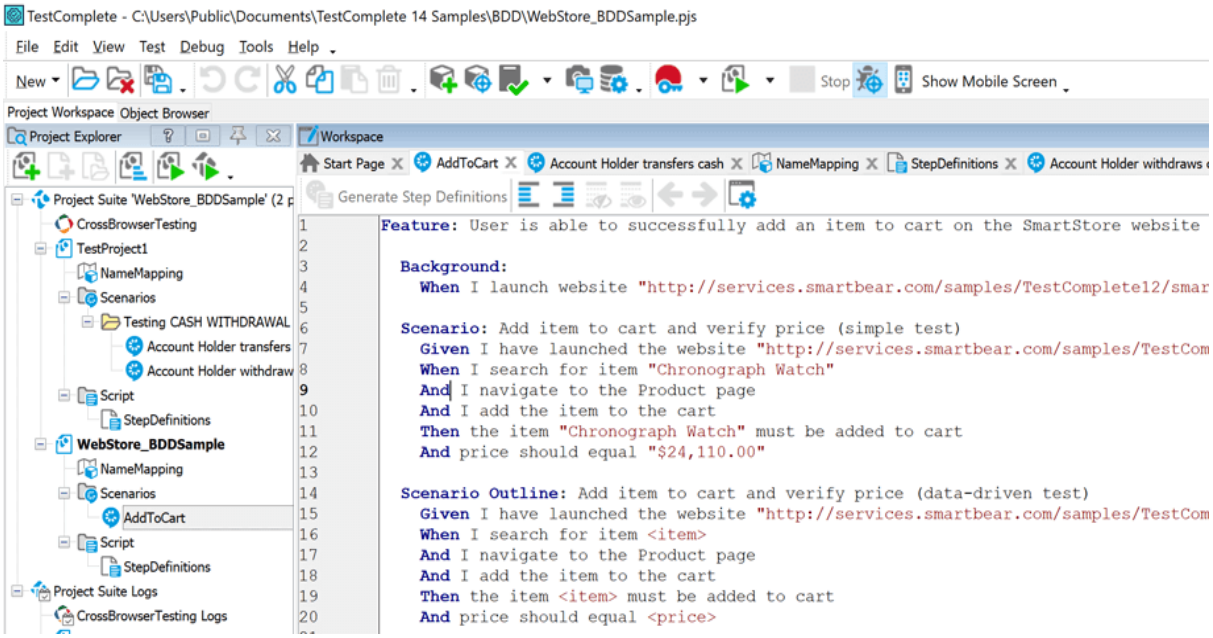
TestComplete izstrādātāju moto skan šādi: “Testējiet vairāk. Testējiet ātrāk. Testējiet gudrāk” [20].

TestComplete ir testēšanas rīks programmu un tehnoloģiju automatizētai testēšanai, kurš piedāvā testēt gan darbvirsmas, gan tīmekļa, gan mobīlās lietotnes. Rīks atbalsta un palīdzēs notestēt dažāda veida lietotnes, gan 32-bitu, gan 64-bitu, kuras ir tikušas izstrādātas

sekojošās vidēs: C/C++; .NET; WPF (XAML); Visual Basic; Java; JavaFX; Delphi; C++Builder; Qt [20].

TestComplete atbalsta skriptu rakstīšanu vairākas programmēšanas valodās: JavaScript, Python, VBScript, Jscript, C++, Delphi un C#, kā arī pats skripta piemērs ir apskatāms *attēlā 6.1.1.*

TestComplete ir iebūvēta ALM integrācija ar tādām lietotnēm kā Jenkins, HP Quality Center un citām [20].



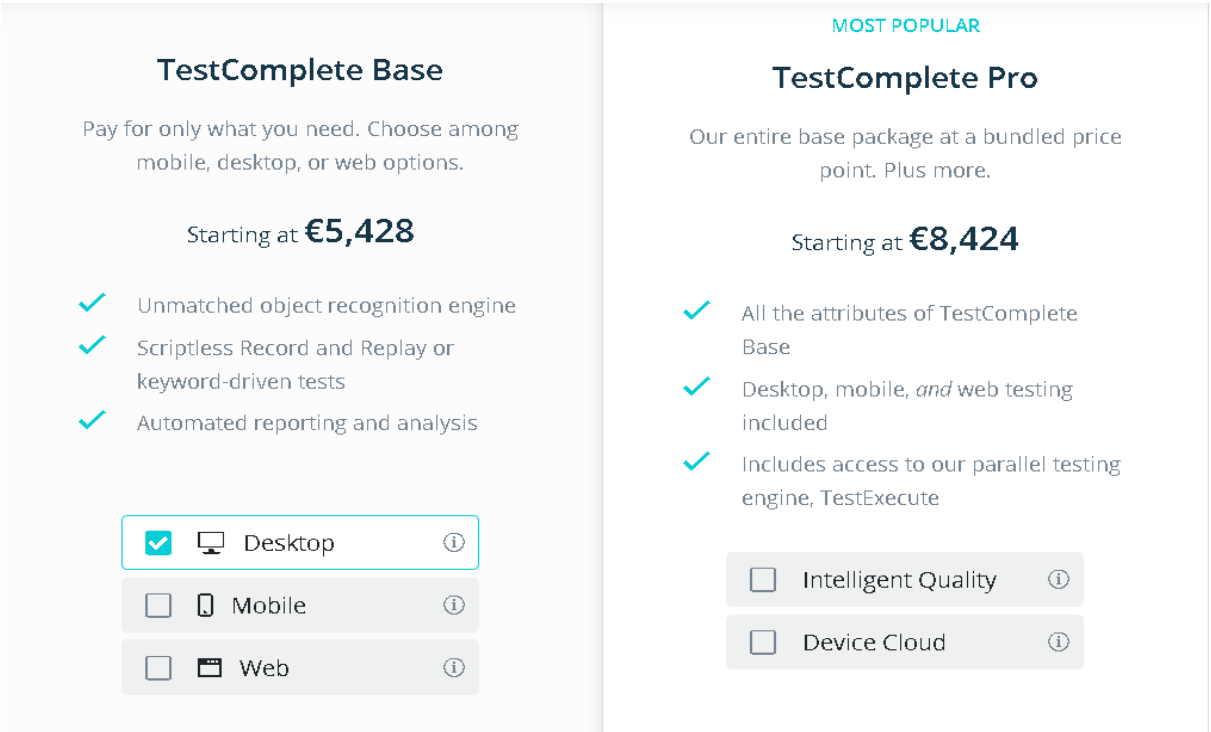
6.1.1. att. **TestComplete BDD-stila testa gadījuma apraksts**

TestComplete ir izmantojams gan funkcionālajā, gan vienībtestēšanā. Rīks nodrošina izcilu regrestestēšanas atbalstu un to var izmantot daudzos testēšanas veidos, kā piemēram: datu bāzētu testēšanā, objektorientētā testēšanā un citos [20].

TestComplete rīka dokumentācija ir ļoti plaša, detalizēta, strukturēta un viegli saprotama. Apraksts mājas lapā sniedz priekšstatu, ka ar šo rīku gana viegli var strādāt testētāji, kuriem ir pieredze skriptu rakstīšanā, gan testētāji, kuriem nav programmēšanas prasmes. Tas iespējams, pateicoties darbību ierakstīšanas un izpildīšanas (record and playback) funkcionalitātei.

Taču testēšanas rīkam TestComplete neskatoties uz visām labajām lietām ir arī pāris būtiski trūkumi, kā piemēram:

- Pārāk daudz "klikšķināšanas" un pāreju rakstot testu skriptus.
- Nevar integrēt ar GIT versiju kontroli.
- Šis ir maksas rīks, kas pašā vienkāršākajā komplektācijā maksā sākot no 5428 euro gadā, ko var redzēt *attēlā 6.1.2.*



6.1.2. att. **Rīka pieejamās komplektācijas un cenas**

6.1.2. Rīka novērtējums

Balstoties uz autora izlasīto informāciju par konkrēto rīku, kā arī uz neliela praktiskā rīka pielietojuma pamata, ko autors speciāli veica, lai spētu objektīvāk izvērtēt rīku, tika veikti kopējie secinājumi par katru no kritērijiem, kā arī katram no tiem sniegts atbilstošs vērtējums, kas parādīts *tabulā 6.1.2.1.* zemāk:

6.1.2.1. tabula. **TestComplete rīka novērtējuma tabula**

Kritērijs	**Vērtējums**	**Komentārs**
Programmēšanas valodu atbalsts	5	Rīks atbalsta vairākas programmēšanas valodas, tādas kā JavaScript, Python, VBScript, Jscript, C++, Delphi un C#,
Iebūvēts UI elementu/objektu atpazīšanas rīks	5	Ir iebūvēts diezgan saprotams un viegli lietojams UI elementu/objektu atpazīšanas rīks ar mākslīgo intelektu, kas ļauj identificēt dinamiskos UI elementus balstītos uz īpašībām, gan ar mākslīgā intelekta darbinātu vizuālo atpazīšanu.
Nepieciešamās kodēšanas prasmes	4	Testēšanas rīks ir gana labs un parocīgs gan pieredzējušiem skriptu rakstītājiem, gan arī testētājiem, kas līdz šim nav saskārušies ar programmēšanu un skriptu rakstīšanu.
Iebūvētas ALM	4	Ir iebūvēta ALM integrācija ar tādām lietotnēm kā

integrācijas		Jenkins, HP Quality Center.
Pieejama dokumentācija	5	TestComplete rīka dokumentācija ir ļoti plaša, detalizēta, strukturēta un viegli saprotama, jo tajā ir arī daudz paskaidrojošo attēlu, kā arī liels skaits pamācību video, kas ir ļoti noderīgi.
Rīka instalēšanas un lietošanas vienkāršums	5	Darba gaitā darba autors instalēja TestComplete 14.72 (Free Trial) versiju ar darbvirmas lietotņu moduli. Rīks ātri instalējas, instalēšana aizņem mazāk par 10 minūtēm.
Testu skriptu izpilde	3	Testi izpildās diezgan ātri, bet ne vienmēr korekti, jo bieži nākas secināt, ka kādu elementu skripts vienreiz ir spējis atrast, bet otrreiz vairs nē, kā arī pietiekami ilgi nākas gaidīt, lai sāktos darbība izvēlētajā aplikācijā.
Cena	2	Tas ir diezgan dārgs maksas rīks, kas piedāvā 30 dienu bezmaksas izmēģinājuma laiku, taču pēc šīm 30 dienām nāksies maksāt vismaz 5428 euro gadā, kas ir vislētākais un primitīvākais variants.
Record and playback funkcija	5	Record and playback funkcija ir pieejama, tā strādā nekļūdīgi un tā ir viegla lietošanā un saprotama gala lietotājam.
Lietotāja saskarne	4	Lietotāja saskarne kopēji ir diezgan intuitīva un saprotama gala lietotājam, taču ir arī lietas/pogas, kuru izvietojums nav pavisam skaidrs un tās varētu atrasties tām piemērotākā vietā.
Operētājsistēmu atbalsts	2	Ar TestComplete var strādāt tikai lietotājs ar Windows OS jeb citiem vārdiem sakot: TestComplete nepiedāvā vairāku OS atbalstu.
Lietotņu atbalsts	5	TestComplete atbalsta visus trīs populārākos lietotņu tipus: tīmekļa, darbvirsmas un mobīlās lietotnes.
Rīka apmācību process	5	Apmācības process darbam ar TestComplete ir diezgan vienkāršs un saprotams, jo rīks ir intuitīvi saprotams, kā arī dokumentācija ir augstā līmenī.
Rīka tiešsaistes atbalsts	5	Labs, ātrs un atsaucīgs atbalsts, kas iespējams gan caur telefonu, gan e-pastu, gan arī uzdodot savu jautājumu izstrādātāja oficiālajā forumā.

Atkļūdošanas atbalsts	5	Ļoti stiprs atkļūdošanas atbalsts, kas palīdz noteikt kļūdas faktisko/iespējamo iemeslu, kā arī palīdz to atrisināt.
Atskaišu, logu izveide un glabāšana	3.5	Atskaites tiek veidotas HTML un XML formāta. TC parāda izpildes rezultātus, bet tas parāda visus soļus vienā panelī, kā arī nesniedz informāciju par katru testa soli. Logi ir skaidri un saprotami.
Dažādu testēšanas tipu atbalsts	5	TestComplete atbalsta dažādus testēšanas veidus un metodikas: vienības testēšanu, funkcionālo un GUI testēšanu, regresijas testēšanu, sadalīto testēšanu un citus.
Datubāžu lietotņu atbalsts	5	Rīks ļoti labi sadarbojas/strādā (ielasa vaicājumus uz DB utt) ar populārākajām datubāžu lietotnēm.
Data-driven testēšana	5	Rīks nodrošina piekļuvi un ielasīšanu no dažādiem ārējiem avotiem kā Excel vai MySQL. Rīks var glabāt skriptus atsevišķi no ievadāmajiem datiem.
Rīka evolūcija ar gadiem	5	Rīks pastāvīgi tiek uzturēts, atjaunināts, novērstas kļūdas tajā. Bieži tiek izlaisti jauni laidieni ar jaunu vai uzlabotu funkcionalitāti.
Koda/skriptu atkārtota izmantojamība	5	Rīkā ir viedās atpazīšanas funkcijas, kas ļauj izmantot skriptu atkārtoti jaunā būvējumā/projektā, kā arī pieglabāt funkcijas bibliotēkās.
Atbilstība jaunākajām tendencēm	3.5	Ir ieviesta un turpina attīstīties AI/ML. Ir iestrādes arī saistība ar bezkoda automatizēto testēšanu, kā arī ar RPA.

Aritmētiskais vidējais vērtējums pa visiem kritērijiem TestComplete rīkam ir: **4.363**, ko var uzskatīt par diezgan augstu rezultātu.

6.2. Unified Functional Testing rīks

6.2.1. Rīka apraksts

Unified Functional Testing (UFT) rīks, iepriekš pazīstams kā QuickTest Professional (QTP), ir Hewlett-Packard (HP) kompānijas radīts licenzēts/komerciāls automatizētās testēšanas rīks, kuram ir pieejama 30 dienu izmēģinājuma versija.

UFT ir funkcionāls testēšanas rīks, kas vislabāk piemērots lietotnes funkcionālajai un regresijas testēšanai.

UFT ir grafiskais lietotājs interfeisa (GUI) record-playback automatizācijas rīks. Rīks tiek saukts arī par vienotās funkcionalitātes testēšanas rīku, jo izmanto vienu grafisko saskarni gan API testēšanai, gan GUI testēšanai.

Tas ir vienkāršs un ļoti lietotājam draudzīgs rīks, kas mijiedarbojas ar Windows darbvirsmas un tīmekļa lietotnēm. Tas ir funkcionāls testēšanas rīks, kuram ir iebūvēta funkcija saglabāt ekrānuzņēmumu no katras lapas, kurā tas ir pārvietojies skripta izpildes laikā.

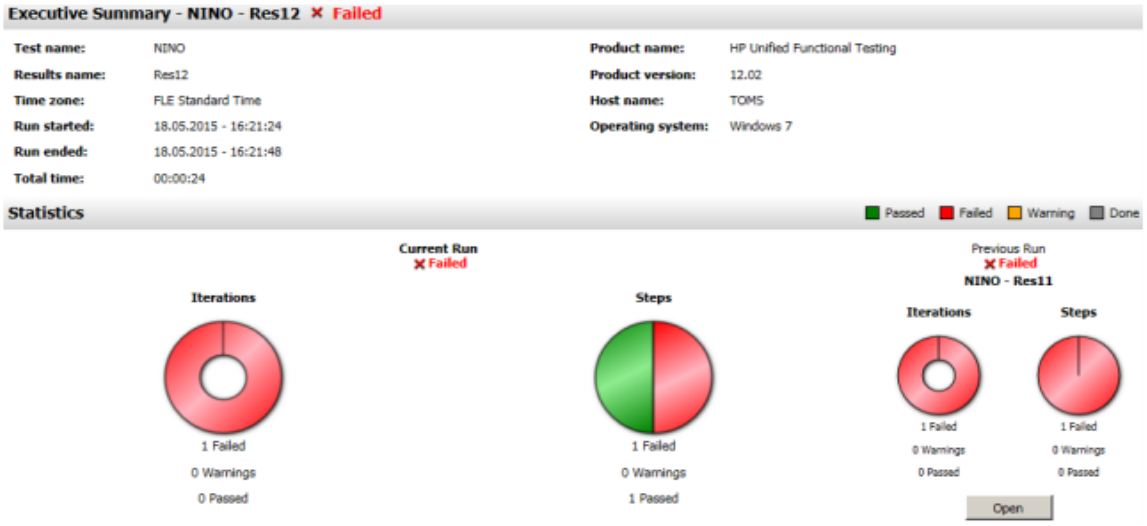
Testu rakstīšanai UFT izmanto Visual Basic skriptu valodu (VBScript).

Šis rīks ļauj automatizēt un notestēt arī dažādus biznesa procesus, ja nepieciešams.

Rīks atbalsta populāras automatizācijas sistēmas - uz atslēgvārdiem balstītu testēšanas pieeju, moduļu testēšanas pieeju, uz datiem balstītu testēšanas pieeju utt.

UFT rīka dokumentācija nav tik viegli atrodama, kā arī ir nepieciešama iepriekšēja reģistrācija, lai varētu iegūt un izmantot HP pasi, lai piekļūtu rīka dokumentācijai.

UFT arī ģenerē automatizētās testu atskaites gan tekstuāla, gan grafiskā veidā, ko var redzēt *attēlā 6.2.1.*



6.2.1. att. **Unified Functional Testing testēšanas atskaite**

6.2.2. Rīka novērtējums

Balstoties uz autora izlasīto informāciju par konkrēto rīku, kā arī uz neliela praktiskā rīka pielietojuma pamata, ko autors speciāli veica, lai spētu objektīvāk izvērtēt rīku, tika veikti kopējie secinājumi par katru no kritērijiem, kā arī katram no tiem sniegts atbilstošs vērtējums, kas parādīts *tabulā 6.2.2.1.* zemāk:

6.2.2.1. tabula. **UFT rīka novērtējuma tabula**

Kritērijs	Vērtējums	Komentārs

Programmēšanas valodu atbalsts	2	Rīks atbalsta tikai vienu programmēšanas valodu: VBScript
Iebūvēts UI elementu/objektu atpazīšanas rīks	5	Ir iebūvēts viegli lietojams UI rīks, kā arī ir iespējams izmantot un uzstādīt papildu UI rīku ar plašāku funkcionalitāti, kas var ne tikai "izspiegot" objektus, bet arī "iemācīties" tos.
Nepieciešamās kodēšanas prasmes	4	Testēšanas rīks ir gana labs un parocīgs iesācējiem, jo tam ir record-playback funkcija, katra koda rindiņa ir skaidra un saprotama lietotājam. Skriptus ir diezgan viegli rediģēt un parametrizēt.
Iebūvētas ALM integrācijas	4	Ir iebūvēta ALM integrācija ar citiem HP produktiem, tādiem kā HP Quality Center, HP Load Runner un citiem.
Pieejamā dokumentācija	4	UTF rīka dokumentācija ir ļoti plaša, detalizēta, strukturēta, bet dažviet nav tik viegli saprotama. Tomēr, lai piekļūtu šai dokumentācijai ir jāreģistrējas un jāveido HP pase.
Rīka instalēšanas un lietošanas vienkāršums	4	Ir iepriekš jāreģistrējas un jāaizpilda anketa, lai izmantotu izmēģinājuma versiju, bet pats instalēšanas process ir triviāls un intuitīvi saprotams.
Testu skriptu izpilde	4	Testi izpildās samērā ātri, un bieži vien korekti, jo pilnīgi visi skripti ir rakstīti vienā noteiktā programmēšanas valodā.
Cena	2.5	Tas ir dārgs maksas rīks, kas piedāvā 30 dienu bezmaksas izmēģinājuma laiku. Pēc šo 30 dienu iztecēšanas par visparastāko licenci nāksies maksāt 3200$ gadā.
Record and playback funkcija	5	Record and playback funkcija ir pieejama, tā strādā nekļūdīgi un tā ir viegla lietošanā un saprotama gala lietotājam.
Lietotāja saskarne	3.5	Lietotāja saskarne kopēji ir viegli saprotama gala lietotājam, taču ir arī lietas, kuru izvietojums nav pavisam skaidrs, kā arī izstrādātājiem vēl vajadzētu piedomāt pie labāka/kvalitatīvāka grafiskā interfeisa.
Operētājsistēmu atbalsts	2	Ar UFT var strādāt tikai lietotājs ar Windows OS jeb citiem vārdiem sakot: UFT nepiedāvā vairāku OS atbalstu.

Lietotņu atbalsts	4	UFT atbalsta divus no trim populārākos lietotņu tipus: tīmekļa un darbvirsmas lietotnes.
Rīka apmācību process	4	Apmācības process darbam ar UFT ir diezgan vienkāršs un saprotams, jo rīks ir intuitīvi saprotams, kā arī dokumentācija ir augstā līmenī, tomēr ir dažas nepilnības dēļ kurām process var ieilgties.
Rīka tiešsaistes atbalsts	4	Diezgan ātrs un atsaucīgs atbalsts, taču atbalsta personāls ne visu saprot no pirmās reizes un uzreiz spēj palīdzēt. Atbalsts iespējams caur tiešsaistes čatu, kā arī atbalsta forumu.
Atkļūdošanas atbalsts	5	Ļoti stiprs un intuitīvi saprotams atkļūdošanas atbalsts, kas palīdz noteikt kļūdas faktisko/iespējamo iemeslu, kā arī palīdz to atrisināt.
Atskaišu, logu izveide un glabāšana	5	UFT sniedz testa atskaites. Tajās tiek parādītas testa darbības hierarhijas kokā, kā arī tā sniedz katra testa kopsavilkumu par katru soli. UFT sniedz statistiku par iepriekšējā un šī brīža testa rādītājiem salīdzinot tos savā starpā un atspoguļo to diagrammu veidā. Atskaišu un logu lasīšana ir lietotājam draudzīga un viegli saprotama.
Dažādu testēšanas tipu atbalsts	3.5	UFT atbalsta dažādus testēšanas veidus un metodikas, piemēram funkcionālo, GUI testēšanu un regresijas testēšanu. Taču ir arī veidi/tipi, kurus šis rīks neatbalsta.
Datubāžu lietotņu atbalsts	5	Rīks ļoti labi sadarbojas/strādā (ielasa vaicājumus uz DB utt) ar populārākajām datubāžu lietotnēm.
Data-driven testēšana	5	Rīks nodrošina piekļuvi un ielasīšanu no dažādiem ārējiem avotiem kā Excel vai MySQL. Rīks var glabāt skriptus atsevišķi no ievadāmajiem datiem, kā arī rīkam ir iebūvētas datu tabulas, kuras nodrošina daudz vieglāk ceļu uz DDT.
Rīka evolūcija ar gadiem	5	Rīks pastāvīgi tiek uzturēts, atjaunināts, novērstas kļūdas tajā. Bieži tiek izlaisti jauni laidieni ar jaunu vai uzlabotu funkcionalitāti.
Koda/skriptu	5	Rīkā ir viedās atpazīšanas funkcijas, kas ļauj izmantot

atkārtota izmantojamība		skriptu atkārtoti jaunā būvējumā/projektā, kā arī pieglabāt funkcijas bibliotēkās.
Atbilstība jaunākajām tendencēm	4	Ir ieviesta un turpina attīstīties AI/ML. Aktīvi tiek attīstīta arī bezkoda automatizētā testēšana, un izstrādātāji ir sākuši arī RPA ieviešanu.

Aritmētiskais vidējais vērtējums pa visiem kritērijiem Unified Functional Testing rīkam ir: **4.068**, ko var uzskatīt par samērā augstu rezultātu.

6.3. Katalon Studio rīks

6.3.1. Rīka apraksts

Katalon Studio ir viens no jaunākajiem un straujāk attīstošākajiem automatizētās testēšanas rīkiem, tieši tāpēc arvien vairāk testēšanas inženieru izvēlas Katalon. Platforma atvieglo automatizētus testus tīmekļa saskarnēm, API un mobilajām lietotnēm (gan iOS, gan Android), nodrošina testu ierakstīšanu un analīžu atskaišu veidošanu [11].

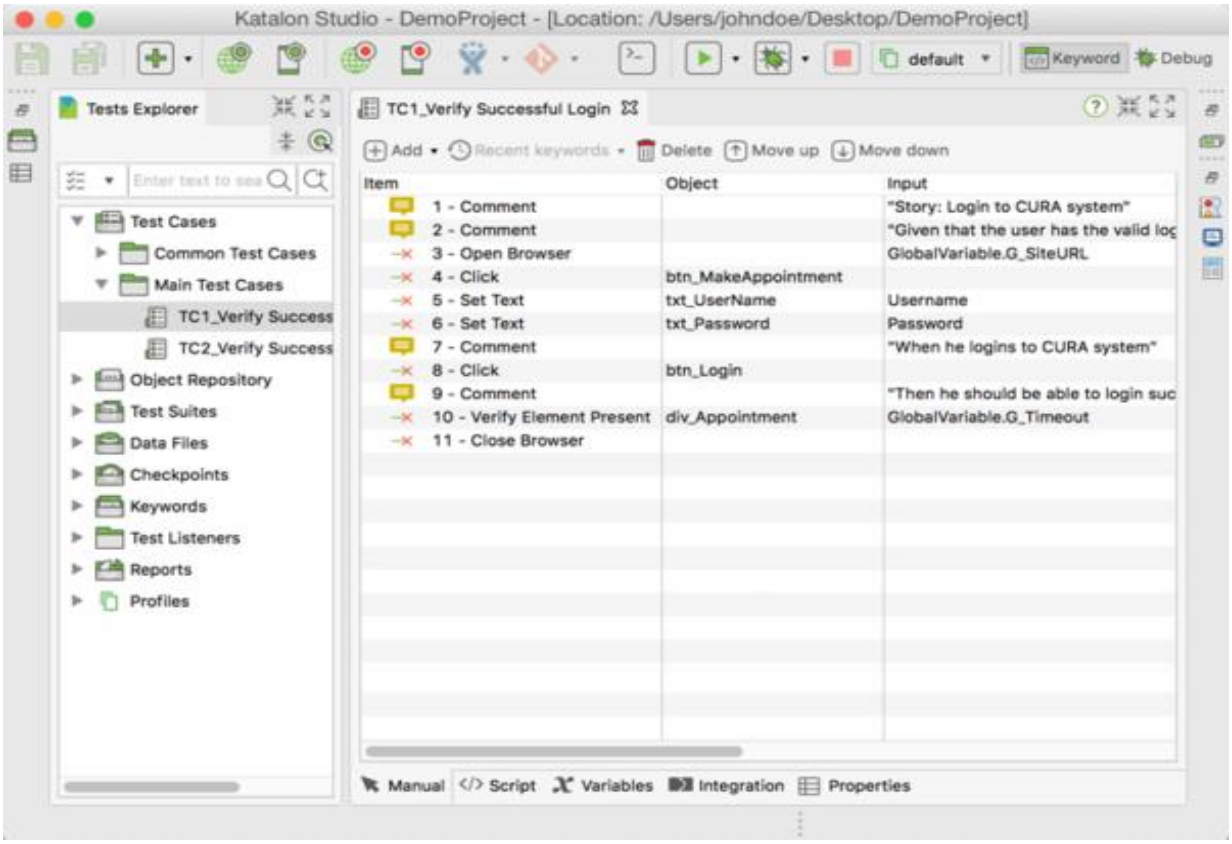
Katalon lielākoties ir paredzēts, lai izveidotu un atkārtoti izmantotu automatizētus testa skriptus, kas izveidoti bez kodēšanas. Katalon Studio spēj palaist automatizētus lietotāja interfeisa elementu testus, tostarp uznirstošos logus, iFrames un paredzēt gaidīšanas laiku. Rīku var palaist uz Microsoft Windows, MacOS un Linux operētājsistēmām, kā arī skripta koda piemērs uz MacOS ir atrodams *attēlā 6.3.1.* zemāk.

Sākot ar Katalon Studio 7 ir iespējams automatizēt arī darbvirsmas lietotnes, ja tās darbojas uz Windows 10 OS. Katalon Studio 7 balstās uz Windows lietojumprogrammu draivera (WinAppDriver) servera, kā arī ir iespējams automatizēti testēt lietotnes, kas pamatā ir izveidotas uz vienas no šīm platformām: UWP, WinForms, WPF un klasiskās Windows (Win32) [11].

Šis rīks ir aprīkots ar daudzām funkcijām, kas atbalsta vienkāršu testu automatizāciju. Neatkarīgi no tā, vai Jums ir vai nav priekšzināšanas un iemaņas skriptu rakstīšanā, Jūs joprojām varat bez problēmām automatizēt testu darbvirsmas lietotnēm izmantojot Katalon. Šīs funkcijas ietver sevī:

- Viegla iestatīšana un konfigurēšana
- Māk noteikt un izspiegot Windows objektus
- Spēj ierakstīt Windows darbības un veselus manuālos testus.
- Vieds elementu atrašanas mehānisms
- Windows iebūvētie un pielāgotie atslēgvārdi

- Minimāla apkope



6.3.1. att. **Skripta piemērs uz MacOS**

6.3.2. Rīka novērtējums

Balstoties uz autora izlasīto informāciju par konkrēto rīku, kā arī uz neliela praktiskā rīka pielietojuma pamata, ko autors speciāli veica, lai spētu objektīvāk izvērtēt rīku, tika veikti kopējie secinājumi par katru no kritērijiem, kā arī katram no tiem sniegts atbilstošs vērtējums, kas parādīts *tabulā 6.3.2.1.* zemāk:

6.3.2.1. tabula. **Katalon Studio rīka novērtējuma tabula**

Kritērijs	**Vērtējums**	**Komentārs**
Programmēšanas valodu atbalsts	4	Rīks atbalsta vairākas populāras programmēšanas valodas: C#, Java, Ruby, Python.
Iebūvēts UI elementu/objektu atpazīšanas rīks	5	Ir iebūvēts viegli lietojams UI rīks, kas atpazīst un izspiego Windows objektus, kā arī vieds elementu atrašanas mehānisks.
Nepieciešamās kodēšanas prasmes	4.5	Katalon Studio ir iespējams lietot gandrīz bez jebkādām sākotnējām kodēšanas prasmēm.
Iebūvētas ALM	5	Ir iebūvēta ALM integrācija ar HP produktiem, kā arī ir

integrācijas		iespējams savienot ar Jenkins, TeamCity.
Pieejamā dokumentācija	5	Katalon rīka dokumentācija ir ļoti labi strukturēta, saprotams un detalizēta. Ir daudz tabulāru lietu vieglākai uztveramībai. Dokumentācijai ir iespējams brīvi piekļūt un to ir viegli atrast.
Rīka instalēšanas un lietošanas vienkāršums	5	Instalācijas grafiskais interfeiss ir ļoti augstā līmenī, viss ir ļoti labi saprotams un viegli konfigurējams.
Testu skriptu izpilde	5	Testi izpildās ātri un korekti. Ir iespējota diezgan unikāla, starp pārējiem uzskaitītajiem šajā darbā testēšanas rīkiem, funkcija: paralēlā skriptu izpilde.
Cena	3	Tas ir maksas rīks, kas piedāvā 30 dienu bezmaksas izmēģinājuma laiku. Pēc šo 30 dienu iztecēšanas, lai turpinātu lietot standarta licences iespējas ir jākonsultējas ar pārdošanas speciālistu par nosacījumiem.
Record and playback funkcija	5	Record and playback funkcija ir pieejama, tā strādā nekļūdīgi un tā ir viegla lietošanā un saprotama gala lietotājam.
Lietotāja saskarne	5	Lietotāja saskarnes grafiskais interfeiss ir ļoti augstā līmenī. Kopēji gala lietotājam saskarne ir viegli saprotama, tajā nav nekā liela vai nevietā.
Operētājsistēmu atbalsts	5	Rīks atbalsta visas trīs populārāks operētājsistēmas: Windows, MacOS, Linux.
Lietotņu atbalsts	5	Katalon atbalsta visus trīs populārākos lietotņu tipus: tīmekļa, darbvirsmas lietotnes un mobīlās lietotnes.
Rīka apmācību process	5	Apmācības process darbam ar Katalon ir vienkāršs, jo rīks ir intuitīvi saprotams, tas izmanto lielākoties bezkoda skriptu rakstīšanu, kā arī dokumentācija ir augstā līmenī.
Rīka tiešsaistes atbalsts	2	Ir iespējams uzdot jautājumu gan oficiālajā forumā, gan paskatīties video, taču tikt pie kaut kāda ātra tiešsaistes atbalsta ir ļoti grūti.
Atkļūdošanas atbalsts	5	Ļoti stiprs un intuitīvi saprotams atkļūdošanas atbalsts, kas palīdz noteikt kļūdas faktisko/iespējamo iemeslu, kā arī palīdz to atrisināt.

Atskaišu, logu izveide un glabāšana	4	Katalon sniedz testa atskaites: gan kopējās, gan paralēlās, gan visu pēdējo atskaišu kopumu. Ir pieejama arī atskaišu vēsture. Ir iespējams automātiski ģenerēt atskaites HTML un CSV formātos, kā arī ir iespējams iespējot ekrānuzņēmumu un video uzņemšanu, taču vajadzētu izstrādātājiem piestrādāt pie kopējā grafiskā interfeisa.
Dažādu testēšanas tipu atbalsts	4	Katalon atbalsta tādus testēšanas veidus un metodes kā funkcionālo, GUI testēšanu, slodzes un regresijas testēšanu. Taču ir daži veidi/tipi, kurus šis rīks neatbalsta.
Datubāžu lietotņu atbalsts	5	Rīks ļoti labi sadarbojas/strādā (ielasa vaicājumus no DB un tā tālāk) ar populārākajām datubāžu lietotnēm, kā piemēram MySQL.
Data-driven testēšana	4.5	Rīks nodrošina piekļuvi un ielasīšanu no dažādiem ārējiem avotiem kā CSV, Excel vai MySQL. Rīks var glabāt skriptus atsevišķi no ievadāmajiem datiem.
Rīka evolūcija ar gadiem	5	Šis ir viens no rīkiem, kurš tiek vislabāk un viskvalitatīvāk uzturēts. Rīks pastāvīgi tiek atjaunināts, tajā tiek novērstas kļūdas. Bieži tiek izlaisti jauni laidieni ar jaunu vai uzlabotu funkcionalitāti.
Koda/skriptu atkārtota izmantojamība	4	Rīkā ir iespējams atkārtojošās funkcijas pieglabāt atsevišķās bibliotēkās.
Atbilstība jaunākajām tendencēm	4	Ir ieviesta un turpina attīstīties AI/ML. Sen ir ieviesta un aktīvi tiek attīstīta bezkoda automatizētā testēšana.

Aritmētiskais vidējais vērtējums par visiem kritērijiem Katalon Studio rīkam ir: **4.5**, ko var uzskatīt par ļoti augstu rezultātu.

6.4. Squish rīks

6.4.1. Rīka apraksts

Squish ir komerciāls starpplatformu GUI un regresijas testēšanas rīks, kas var pārbaudīt lietotnes, kuru pamatā ir dažādas GUI tehnoloģijas, automatizēt skriptu izpildi un pārbaudīt reālajā laikā, vai konkrētā pārbaudāmā lietotne atbilst tās specifikācijai [25].

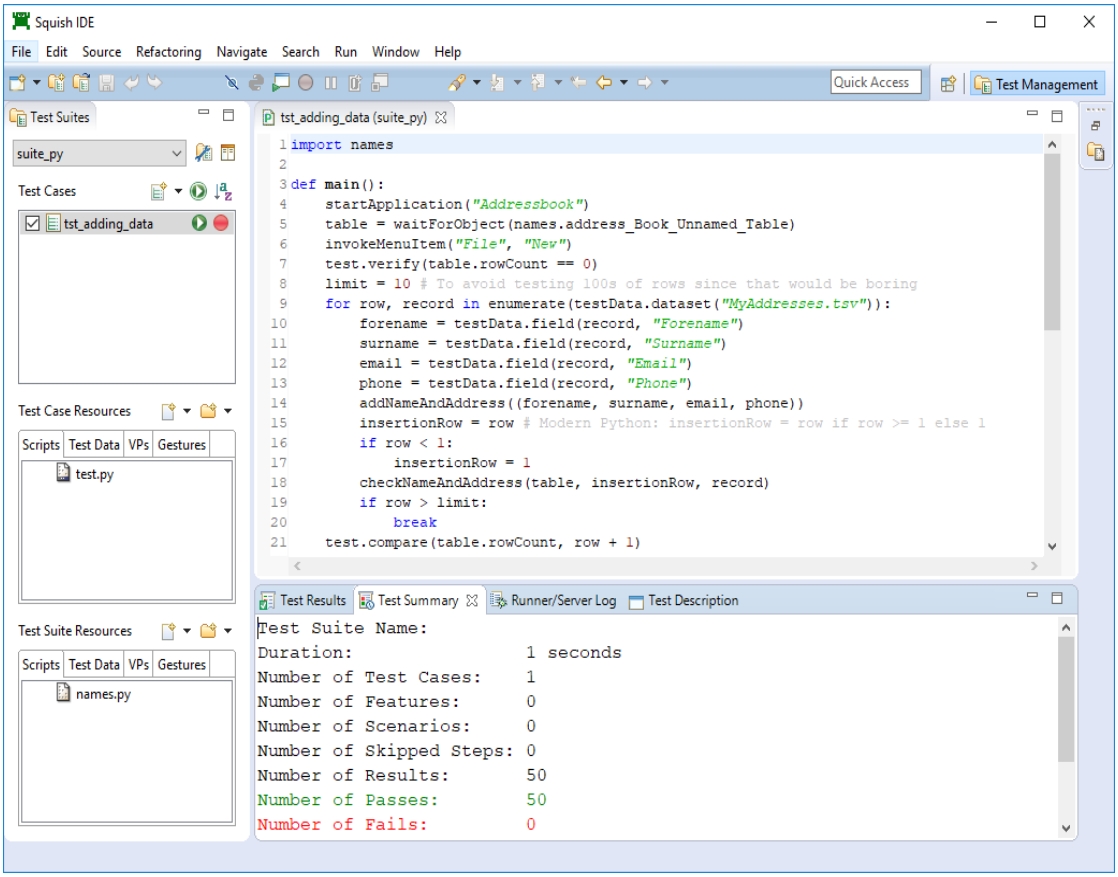
Squish ir saderīgs ar darbvirsmas, tīmekļa un mobilajām grafiskajām saskarnēm, kā arī tas atbalsta darbību uz visām populārākajām operētājsistēmām: Windows, Linux, MacOS, Android, iOS.

Squish izmanto uz īpašībām balstītu objektu identifikāciju jeb objektu identifikāciju neatkarīgi no ekrāna pozīcijas un spēj ierakstīt un atkārtot testa skriptus, kas rakstīti JavaScript, Perl, Python, Ruby vai Tcl. Tā ir divkomponentu sistēma, kas sastāv no skrējēja ("runner"), kas interpretē un izpilda skriptus, un no servera, kas ieslēdz un kontrolē testējamo lietotni. Squish grafiskā interfeisa paraugs ir atrodams zemāk, *attēlā 6.11.1.1.* [25].

Squish darbība ir pavisam vienkārša: programmatūra pārvērš darbības, kuras veicam uz GUI, skriptos, lai pēc tam varētu automātiski atkārtot šīs darbības secīgi. Squish neatkārto lietotāja darbības. Tas ņem vērā lietotāju darbības un pēc iespējas ātrāk tās atkārto interfeisā.

Lai pārliecinātos, ka saskarne darbojas pareizi, var pievienot atbilstības testus, kurus Squish veiks, lai pārbaudītu, vai rezultāts atbilst sagaidāmajam.

Kad testa komplekts ir palaists, Squish izveido XML failu Junit formātā, kurā glabāsies testa rezultāti. Pēc tam šo failu var izmantot nepārtrauktas integrācijas rīkā, piemēram, Jenkins vai Bitbucket Pipelines [25].



6.4.1.1. att. **Squish rīka lietotāja grafiskais interfeiss**

6.4.2. Rīka novērtējums

Balstoties uz autora izlasīto informāciju par konkrēto rīku, kā arī uz neliela praktiskā rīka pielietojuma pamata, ko autors speciāli veica, lai spētu objektīvāk izvērtēt rīku, tika veikti kopējie secinājumi par katru no kritērijiem, kā arī katram no tiem sniegts atbilstošs vērtējums, kas parādīts *tabulā 6.4.2.1.* zemāk:

6.4.2.1. tabula. **Squish rīka novērtējuma tabula**

Kritērijs	**Vērtējums**	**Komentārs**
Programmēšanas valodu atbalsts	4	Rīks atbalsta vairākas populāras programmēšanas valodas, tādas kā: Perl, Python, Ruby.
Iebūvēts UI elementu/objektu atpazīšanas rīks	4	Ir iebūvēts unikāls, Squish izstrādāts, rīks, kas spēj atpazīt un izspiegot ne tikai elementus/objektus.
Nepieciešamās kodēšanas prasmes	4	Rīks neprasa nekādas sākotnējās testēšanas prasmes, jo skripti tiek veidoti pēc codeless tehnoloģijas, taču pēc tam izveidotos skriptus ir problemātiski labot.
Iebūvētas ALM integrācijas	5	Ir iespējama ALM integrācija ar HP ALM, RQM, Seapine TCM, kā arī ar nepārtrauktās integrācijas rīkiem, tādiem kā Jenkins, Bamboo, TeamCity
Pieejama dokumentācija	4.5	Dokumentācija ir viegli pieejama/atrodama, labi strukturēta, plaša un detalizēta, kā arī tajā ir grafiski elementi, kas palīdz izprast tekstu.
Rīka instalēšanas un lietošanas vienkāršums	5	Rīka instalācija bija ļoti vienkārša, intuitīvi saprotama un aizņēma vien pāris minūtes.
Testu skriptu izpilde	5	Testi izpildās ātri un nekļūdīgi, jo pamatā tiek izmantota codeless tehnoloģija, kas šajā rīkā strādā nekļūdīgi.
Cena	3	Tas ir maksas rīks, kas piedāvā 30 dienu bezmaksas izmēģinājuma laiku, taču pēc tam, lai iegādātos visparastāko licenci pieciem lietotājiem, gadā būs jāsamaksā vismaz 3000 EUR.
Record and playback funkcija	5	Record and playback funkcija ir pieejama, tā strādā nekļūdīgi un tā ir viegla lietošanā un saprotama gala lietotājam.
Lietotāja	3	Bija nelielas grūtības atrast nepieciešamās pogas un

saskarne		izvēlnes. Kopumā darbošanās ar rīku nelikās pārāk ērta, jo saskarne nav tik intuitīvi saprotama kā gribētos.
Operētājsistēmus atbalsts	5	Squish atbalsta visas populārākās gan datoru, gan mobīlo ierīču operētājsistēmas.
Lietotņu atbalsts	5	Squish atbalsta visus trīs populārākos lietotņu tipus: tīmekļa, darbvirsmas un mobīlās lietotnes.
Rīka apmācību process	5	Apmācības procesam darbam ar Squish ir samērā vienkāršs un labi saprotams pateicoties plašai dokumentācijai.
Rīka tiešsaistes atbalsts	2	Atbalstu iespējams saņemt tikai sūtot e-pastu par savu problēmu, nav pieejams nekāds tiešsaistes čats vai forums.
Atkļūdošanas atbalsts	5	Ļoti stiprs un intuitīvi saprotams atkļūdošanas atbalsts, kas palīdz noteikt kļūdas faktisko/iespējamo iemeslu, kā arī palīdz to atrisināt.
Atskaišu, logu izveide un glabāšana	3	Atskaites tiek ģenerētas automātiski, taču pati atskaite ir ne sevišķi laba. Tajā netiek attēloti visi veiktie soļi, kā arī netiek veikti ekrānuzņēmumi.
Dažādu testēšanas tipu atbalsts	3	Squish atbalsta tikai dažus testēšanas tipus, kā GUI un regresijas testēšana.
Datubāžu lietotņu atbalsts	5	Rīks ļoti labi sadarbojas/strādā un ielasa vaicājumus no populārākajām datubāžu lietotnēm.
Data-driven testēšana	5	Squish atbalsta data-driven testēšanu saskarņu un datu bāžu arhitektūrā. Squish nodrošina piekļuvi, ielasīšanu un apstrādi no dažādiem ārējiem datu avotiem, tādiem kā: Excel, TSV, TXT, CSV, datubāzēm un citām.
Rīka evolūcija ar gadiem	4	Rīks pastāvīgi tiek uzturēts un laiku pa laikam atjaunināts. Diezgan bieži tiek izlaisti jauni laidieni vai versijas ar jaunu vai uzlabotu funkcionalitāti.
Koda/skriptu atkārtota izmantojamība	4	Var pieglabāt un atkārtoti izmantot jaunizveidotas funkcijas un veselus skriptus.
Atbilstība	4	Ir ieviesta un turpina aktīvi attīstīties AI/ML. Tiek ļoti

jaunākajām tendencēm		aktīvi virzīta uz priekšu un popularizēta bezkoda automatizētā testēšana.

Aritmētiskais vidējais vērtējums par visiem kritērijiem Squish rīkam ir: **4.204**, ko var uzskatīt par ļoti augstu kopējo vērtējumu.

6.5. Tricentis Tosca rīks

6.5.1. Rīka apraksts

Tricentis Tosca ir nepārtrauktas testēšanas platforma, kas nodrošina ātru atgriezenisko saiti ar Agile un DevOps. Tas ļauj noformēt automatizētus, funkcionālus programmatūras testus visos uzņēmuma arhitektūras slāņos [12].

Tricentis Tosca apvieno vairākus programmatūras testēšanas aspektus (testa gadījumu noformēšanu, testēšanas automatizāciju, testa datu noformēšanu un ģenerēšanu, kā arī analīzi), lai pārbaudītu gan GUI, gan API no biznesa viedokļa puses. Divas vispopulārākās un biežak izmantotās Tricentis Tosca tehnoloģijas ir uz moduļiem un uz riskiem bāzētā testēšana [12].

Tā vietā, lai testa automatizēšanai izmantotu skriptu, Tricentis Tosca izmanto uz modeļiem balstītu testēšanas pieeju un izveido pārbaudāmās lietotnes modeli. Tehniskā informācija par testējamo lietotni, testa gadījuma loģika un testa dati tiek saglabāti atsevišķi un savienoti kopā testa izpildes laikā.

Pamatojoties uz riska novērtējumu par pieteikumu saskaņā ar testa prasībām, Tricentis Tosca izmanto uz risku balstītu testa plānu, lai ieteiktu visefektīvākos testa gadījumus lielākai aptveramībai un noteiktu katra testa gadījuma pārklājumu.

Tosca rīku var izmantot, lai automatizētu gan tīmekļa, gan darbvirsmas lietotnes, kuras bāzētas uz vienas no sekojošām tehnoloģijām: SAP GUI, Microsoft.NET, Java Swing, AWT, Delphi un tā tālāk.

Tosca atbalsta arī vairākus populārākos nepārtrauktās integrācijas rīkus, tādus kā: Bamboo, Jenkins, TeamCity un citiem.

Tosca ir viens no pievilcīgākajiem automatizācijas rīkiem, kas ir aprīkots ar plašu automatizētās programmatūras testēšanas testu komplektu un funkcionalitāti.

6.5.2. Rīka novērtējums

Balstoties uz autora izlasīto informāciju par konkrēto rīku, kā arī uz neliela praktiskā rīka pielietojuma pamata, ko autors speciāli veica, lai spētu objektīvāk izvērtēt rīku, tika veikti

kopējie secinājumi par katru no kritērijiem, kā arī katram no tiem sniegts atbilstošs vērtējums, kas parādīts *tabulā 6.5.2.1.* zemāk:

6.5.2.1. tabula. **Tricentis Tosca rīka novērtējuma tabula**

Kritērijs	**Vērtējums**	**Komentārs**
Programmēšanas valodu atbalsts	4.5	Šis rīks atbalsta virkni programmēšanas valodu, piemēram, Delphi, .NET, WPF, Java Swing, Visual Basic utt.
Iebūvēts UI elementu/objektu atpazīšanas rīks	5	Ir iebūvēts viegli lietojams UI rīks, kas izmanto AI-driven testu automatizācijas tehnoloģiju.
Nepieciešamās kodēšanas prasmes	5	Nav nepieciešamas pilnīgi nekādas sākotnējās kodēšanas prasmes, jo tiek izmantota gan record-playback funkcionalitāte, gan arī bezkoda skriptu rakstīšanas tehnoloģija.
Iebūvētas ALM integrācijas	5	Ir iebūvēta ALM integrācija ar HP produktiem, tādiem kā HP Quality Center, HP Load Runner, kā arī Tosca atbalsta nepārtrauktās integrācijas rīkus: Bamboo, Jenkins un citus.
Pieejamā dokumentācija	3.5	Tosca rīka dokumentācija ir ļoti plaša, detalizēta, bet, pēc autora domām, ne tik labi strukturēta, un dažviet nav tik viegli saprotama. Dokumentācija ir viegli atrodama un tās izmantošanai nav nepieciešami papildus faktori.
Rīka instalēšanas un lietošanas vienkāršums	5	Instalācijas process ir ātrs, viegls un intuitīvi saprotams gala lietotājam.
Testu skriptu izpilde	4.5	Testi izpildās ātri un korekti, jo tiek izmantots skript less princips (nav jāraksta kods/skripts).
Cena	2	Tas ir maksas rīks, kas piedāvā 30 dienu bezmaksas izmēģinājuma laiku. Pēc šo 30 dienu iztecēšanas ir jākonsultējas ar pārdošanas speciālistu par individuālu cenu, bet no pieejamās informācijas internetā var secināt, ka par pilnu licenci nāksies maksāt apmēram 10 tūkstošus dolāru gada.
Record and playback funkcija	5	Record and playback funkcija ir pieejama, tā strādā nekļūdīgi un tā ir viegla lietošanā un saprotama gala lietotājam.

Lietotāja saskarne	4	Lietotāja saskarne kopēji ir labi uztverama un saprotama. Grafiskais interfeiss ir augstā līmenī, un kopējais pogu/funkciju izvietojums ir skaidrs.
Operētājsistēmu atbalsts	2	Ar Tosca var strādāt tikai lietotājs ar Windows OS jeb citiem vārdiem sakot: Tosca nepiedāvā vairāku OS atbalstu.
Lietotņu atbalsts	4	Tosca atbalsta divus no trim populārākos lietotņu tipus: tīmekļa un darbvirsmas lietotnes.
Rīka apmācību process	4.5	Apmācības process darbam ar Tosca ir diezgan vienkāršs, jo rīks ir intuitīvi saprotams, kā arī tiek piedāvātas apmācības, dokumentācija un grafiskais interfeiss ir augstā līmenī, taču dokumentācijā dažreiz trūkst attēlu/tabulu labākai informācijas uztveršanai.
Rīka tiešsaistes atbalsts	3	Atbalstu iespējams saņemt tikai zvanot pa telefonu un rakstot e-pastu. Nav tiešsaistes čatu un līdzīgu lietu, tāpēc jebkādas problēmas atrisināšana aizņem vairāk laika un tas nav sevišķi ērti.
Atkļūdošanas atbalsts	5	Ļoti stiprs un intuitīvi saprotams atkļūdošanas atbalsts, kas palīdz noteikt kļūdas faktisko/iespējamo iemeslu, kā arī palīdz to atrisināt.
Atskaišu, logu izveide un glabāšana	4	Šis rīks sniedz diezgan detalizētus pārskatus/atskaites, kas parāda, kā pārbaudāmās sistēmas vājās vietas ietekmē noteikto prasību izpildi. Tiek veidoti un pieglabāti ar log faili.
Dažādu testēšanas tipu atbalsts	4.5	Tosca atbalsta dažādus testēšanas veidus un metodikas, piemēram funkcionālo, GUI, API testēšanu un regresijas testēšanu, kā arī tādas metodes/tehnoloģijas kā uz moduļiem un riskiem bāzētu testēšanu.
Datubāžu lietotņu atbalsts	5	Rīks ļoti labi sadarbojas/strādā (ielasa vaicājumus no DB) ar populārākajām datubāžu lietotnēm, kā arī Tosca nodrošina savu vaicājumu valodu tāpat kā SQL, ko sauc par TQL.
Data-driven testēšana	5	Rīks nodrošina piekļuvi un ielasīšanu no dažādiem ārējiem avotiem kā XML, Excel, JSON, MySQL un citiem. Rīkam

		ir iebūvētas datu tabulas, kuras nodrošina daudz vieglāk ceļu uz DDT.
Rīka evolūcija ar gadiem	5	Rīks pastāvīgi tiek uzturēts, atjaunināts, novērstas kļūdas tajā. Bieži tiek izlaisti jauni laidieni ar jaunu vai uzlabotu funkcionalitāti.
Koda/skriptu atkārtota izmantojamība	5	Rīkā ir viedās atpazīšanas funkcijas, kas ļauj noteiktas funkcijas/soļus atkārtoti izmantot citos skriptos, ja šīs funkcijas/soļi ir attiecīgi pieglabāti bibliotēkā.
Atbilstība jaunākajām tendencēm	4.5	Ir ieviesta un turpina aktīvi attīstīties AI/ML. Tiek ļoti aktīvi virzīta uz priekšu un popularizēta bezkoda automatizētā testēšana, kas ir viena no labākajām šobrīd pieejamajām tirgū.

Aritmētiskais vidējais vērtējums par visiem kritērijiem Tricentis Tosca rīkam ir: **4.318**, ko var uzskatīt par diezgan augstu rezultātu.

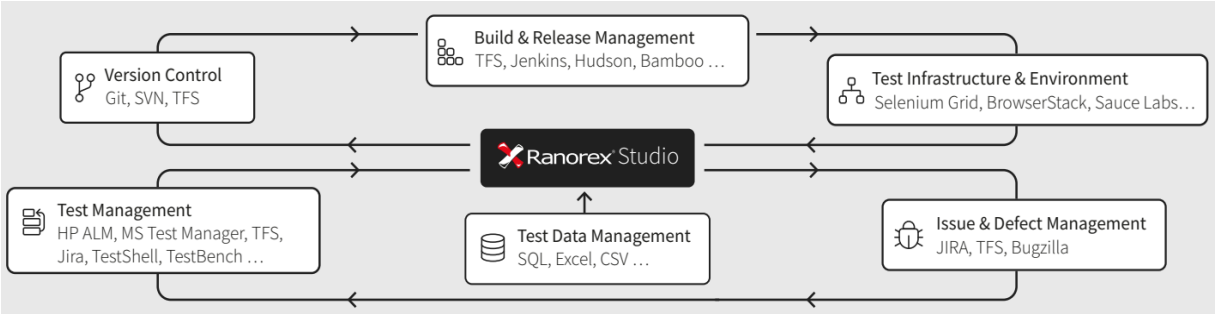
6.6. Ranorex rīks

6.6.1. Rīka apraksts

Ranorex ir testēšanas rīks programmu un tehnoloģiju automatizētai testēšanai, kurš piedāvā testēt gan darbvirsmas, gan tīmekļa, gan mobīlās lietotnes. Rīks atbalsta un palīdzēs notestēt dažāda veida lietotnes, gan 32-bitu, gan 64-bitu, kuras ir tikušas izstrādātas sekojošās vidēs: WinForms; WPF; Silverlight; .NET; Java; Delphi; Windows Apps (Native/Hybrid) un citās [22].

Ranorex nav savas skriptu rakstīšanas valodas lietotņu automatizēšanai. Tas izmanto standarta programmēšanas valodas VB.NET un C# [22].

Ranorex ir labs risinājums automatizētas testēšanas ieviešanai nepārtrauktā piegādē un regrestestēšanā, kur nepieciešama nepārtraukta testēšana un ātra atgriezeniskā saite ar testēšanas rezultātiem. Ranorex piedāvā plašu testēšanas rīku klāstu, lai tos varētu integrēt ar: vispopulārākajiem nepārtrauktās integrācijas (*CI*) rīkiem, testu pārvaldības (*TM*) rīkiem un uzdevumu plānošanas rīkiem [22]. Pilns Ranorex iespēju klāst integrācijai ar citiem rīkiem ir parādīts zemāk, *attēlā 6.5.1.*



6.5.1. att. **Ranorex integrācija ar citiem rīkiem**

Ranorex dokumentācija ir ļoti strukturēta, plaša un viegli saprotama.

Ranorex piedāvā arī lietotnes izmēģināju versiju, caur kuru var veikt automatizēto programmatūras skriptu izveidi, ierakstīšanu/atspēlēšanu un analīzi.

Ranorex piedāvā arī ļoti daudzas soli-pa-solim apmācības, kuras lietotāju ērtībai ir dublētas veselās trīs valodās: angļu, franču un vācu, kā arī regulāri tiek organizēti bezmaksas semināri par testēšanu un testu automatizāciju.

Ar šo rīku var strādāt gan pieredzējuši testētāji-skriptētāji, gan testētāji, kuriem nav iepriekšējās pieredzes skriptu rakstīšanā pateicoties record and playback funkcionalitātei.

Taču testēšanas rīkam Ranorex neskatoties uz visām labajām lietām ir arī pāris būtiski trūkumi, kā piemēram:

- Atbalsta tikai divas skriptu rakstīšanas valodas: C# un VB.NET.

- Nestabilas relīzes. Bieži vien tiek izlaistas jaunas versijas, kas savukārt, prasa laiku, lai atjauninātu esošos komponentus, kā arī bieži vien izrādās, ka jaunā funkcionalitāte ir diezgan nestabila un bieži kļūdaina, un tad nākas gaidīt nākamo, stabilo versiju.

- Šis rīks nav atvērta pirmkoda, bet gan maksas rīks, kas pašā vienkāršākajā komplektācijā maksā sākot no 4790 euro gadā.

6.6.2. Rīka novērtējums

Balstoties uz autora izlasīto informāciju par konkrēto rīku, kā arī uz neliela praktiskā rīka pielietojuma pamata, ko autors speciāli veica, lai spētu objektīvāk izvērtēt rīku, tika veikti kopējie secinājumi par katru no kritērijiem, kā arī katram no tiem sniegts atbilstošs vērtējums, kas parādīts *tabulā 6.6.2.1.* zemāk:

6.6.2.1. tabula. **Ranorex rīka novērtējuma tabula**

Kritērijs	**Vērtējums**	**Komentārs**
Programmēšanas valodu atbalsts	4	Rīks atbalsta četras skriptu rakstīšanas valodas: C++, C#, Python un VBScript.
Iebūvēts UI	4	Ir iebūvēts unikāls, Ranorex izstrādāts, UI

elementu/objektu atpazīšanas rīks		elementu/objektu atpazīšanas rīks, kas ir viegli lietojams un saprotams, taču nav apveltīts ar mākslīgo intelektu.
Nepieciešamās kodēšanas prasmes	5	Testēšanas rīks ir gana labs un parocīgs gan pieredzējušiem skriptu rakstītājiem, gan arī testētājiem, kas līdz šim nav saskārušies ar programmēšanu un skriptu rakstīšanu, jo tajā ir iestrādā Record-Playback funkcija.
Iebūvētas ALM integrācijas	5	Vesels saraksts ar iebūvētām ALM integrācijām, tādām kā Bamboo, HP Quality Center, TFS/MTM, Jenkins un citām.
Pieejama dokumentācija	5	Dokumentācija ir ļoti plaša, detalizēta, strukturēta un viegli saprotama. Bez tā, ir pieejama arī demo versija, kur var iziet pilnvērtīgu apmācību par to kā strādāt ar rīka UI. Ranorex piedāvā arī ļoti daudz soli-pa-solim apmācību trīs valodās, kā arī regulārus automatizācijas testēšanas seminārus.
Rīka instalēšanas un lietošanas vienkāršums	5	Darba gaitā darba autors instalēja Ranorex Free trial 30 dienu versiju (bezmaksas). Rīks ātri instalējas, instalēšana aizņem mazāk par 10 minūtēm.
Testu skriptu izpilde	4.5	Testi izpildās ātri, un ļoti reti kad tests tiek izpildīts kļūdaini rīka vainas dēļ.
Cena	2.5	Tas ir diezgan dārgs maksas rīks, kas piedāvā 30 dienu bezmaksas izmēģinājuma laiku, taču pēc šīm 30 dienām nāksies maksāt vismaz 4790 euro gadā, kas ir vislētākais un primitīvākais variants.
Record and playback funkcija	5	Record and playback funkcija ir pieejama, tā strādā nekļūdīgi un tā ir viegla lietošanā un saprotama gala lietotājam.
Lietotāja saskarne	4	Lietotāja saskarne kopēji ir labi uztverama un saprotama, tai ir labs grafiskais interfeiss, taču autors uzskata, ka tā ir pārlieku noslogota un dažos momentos nav tik intuitīvi saprotama.
Operētājsistēmu atbalsts	2	Ar Ranorex var strādāt tikai lietotājs ar Windows OS jeb citiem vārdiem sakot: Ranorex nepiedāvā vairāku OS atbalstu.
Lietotņu atbalsts	5	Ranorex atbalsta visus trīs populārākos lietotņu tipus:

		tīmekļa, darbvirsmas un mobīlās lietotnes.
Rīka apmācību process	4	Apmācības process darbam ar Ranorex nav tik vienkāršs, bet šeit palīgā nāk ļoti laba dokumentācija, apmācību video, kā arī soli-pa-solim apmācības trīs valodās, kas palīdz ātrāk un vieglāk apgūt šo rīku.
Rīka tiešsaistes atbalsts	2.5	Atbalstu iespējams saņemt tikai rakstot e-pastu. Nav tiešsaistes čatu un līdzīgu lietu, tāpēc jebkādas problēmas atrisināšana aizņem vairāk laika un tas nav sevišķi ērti.
Atkļūdošanas atbalsts	5	Ļoti stiprs un intuitīvi saprotams atkļūdošanas atbalsts, kas palīdz noteikt kļūdas faktisko/iespējamo iemeslu, kā arī palīdz to atrisināt.
Atskaišu, logu izveide un glabāšana	4	Atskaites tiek ģenerētas automātiski, tās ir strukturētas tāpat kā izpildāmais testa komplekts, uz kura balstījās testa palaišana. Katrs elements testa komplektā arī tiek parādīts pārskatā, kā arī katrs konkrēts testa darbības vienums. Tiek ģenerēti un pieglabāti arī log faili.
Dažādu testēšanas tipu atbalsts	4	Ranorex atbalsta dažādus testēšanas veidus un metodikas, piemēram funkcionālo, atslēgvārdu, GUI, API testēšanu un regresijas testēšanu.
Datubāžu lietotņu atbalsts	5	Rīks ļoti labi sadarbojas/strādā (ielasa vaicājumus no DB) ar populārākajām datubāžu lietotnēm.
Data-driven testēšana	4	Ranorex nodrošina piekļuvi, ielasīšanu un apstrādi no dažādiem ārējiem datu avotiem, tādiem kā: vienkārša iekšēja tabula, CSV, Excel vai SQL.
Rīka evolūcija ar gadiem	3	Rīks pastāvīgi tiek uzturēts, bet daudz retāk atjaunināts. Samēra reti tiek izlaisti jauni laidieni ar jaunu vai uzlabotu funkcionalitāti.
Koda/skriptu atkārtota izmantojamība	3	Rīkā ļauj pieglabāt un atkārtoti izmantot noteiktas funkcijas, taču šis process ir diezgan apgrūtinošs.
Atbilstība jaunākajām tendencēm	4	Ir ieviesta un turpina aktīvi attīstīties AI/ML. Tiek ļoti aktīvi virzīta uz priekšu un popularizēta bezkoda automatizētā testēšana.

Aritmētiskais vidējais vērtējums par visiem kritērijiem Ranorex rīkam ir: **4.068**, ko var uzskatīt par ļoti augstu kopējo vērtējumu.

6.7. TestArchitect rīks

6.7.1. Rīka apraksts

TestArchitect ir testa automatizācijas rīks lietotņu funkcionālajiem un regresijas testiem. TestArchitect ļauj testētājiem izveidot automatizētus testus tīmekļa pakalpojumiem un lietotnēm, kā arī Windows, Android un iOS lietotnēm.

TestArchitect atbalsta plašu tehnoloģiju un platformu klāstu darbvirsmas lietotņu testēšanai operētājsistēmās Windows un Linux, tostarp ERP programmatūru, piemēram, SAP un Oracle EBS. TestArchitect nodrošina vairāk nekā 420 dažādu funkciju un darbību, lai pārbaudītu darbvirsmas lietotnes, kas izveidotas, izmantojot .Net WPF, .NET, WinForms, MFC, VB, Java, QT, Oracle Forms un citas [13].

TestArchitect ir bezkoda automatizācijas testēšanas rīks, kas testu izveidē un testu izstrādē izmanto uz darbībām balstītu testēšanu. Atšķirībā no atvērtā koda testēšanas rīkiem, piemēram, Selenium vai Protractor, ar TestArchitect nav nepieciešams iepriekšējas programmēšana zināšanas, lai izmantotu Test Automation ietvaru [13].

Izmantojot TestArchitect kopā ar uz darbībām balstītu testēšanu, vienā solī var izveidot testa skriptus, testa gadījumus, dokumentēt testa scenārijus un automatizēt visu testa skriptu.

TestArchitect darbvirsmu testēšanas rīkā ir pieejamas arī advancētas atpazīšanas un izspiegošanas tehnoloģijas, kā: Microsoft UI automatizācija, optiskā simbolu atpazīšana (OCR) un attēlu atpazīšana.

Izmantojot šo rīku kompānijas ietvaros, nav nepieciešamības pēc testētajiem ar izteiktām kodēšanas prasmēm un iemaņām. Šāda veida automatizāciju var paveikt arī biznesa testētājs bez iepriekšējām iemaņām kodēšanā.

Biznesa testētāji var ātri izstrādāt un automatizēt testus visdažādākajām darbvirsmas lietotnēm, izmantojot TestArchitect pilnfunkcionālo iepriekš ieprogrammēto darbību bibliotēku. Testus pat vissarežģītākajiem scenārijiem var izstrādāt ar minimālu programmētāju atbalstu. Testus var izveidot, ierakstot darbības tieši intuitīvajā IDE vai vienkārši “drag-and-drop” tos no darbību bibliotēkas [14].

6.7.2. Rīka novērtējums

Balstoties uz autora izlasīto informāciju par konkrēto rīku, kā arī uz neliela praktiskā rīka pielietojuma pamata, ko autors speciāli veica, lai spētu objektīvāk izvērtēt rīku, tika veikti kopējie secinājumi par katru no kritērijiem, kā arī katram no tiem sniegts atbilstošs vērtējums, kas parādīts *tabulā 6.7.2.1.* zemāk:

6.7.2.1. tabula. **TestArchitect rīka novērtējuma tabula**

Kritērijs	**Vērtējums**	**Komentārs**
Programmēšanas valodu atbalsts	4	Rīks atbalsta trīs populāras skriptu rakstīšanas valodas: Java, Python un C#, kas varētu būt pilnīgi pietiekami.
Iebūvēts UI elementu/objektu atpazīšanas rīks	4.5	Ir iebūvēts unikāls, TestArchitect izstrādāts, kas spēj atpazīt un izspiegot ne tikai elementus/objektus, bet arī optiskos simbolus un attēlus.
Nepieciešamās kodēšanas prasmes	5	Rīks neprasa nekādas sākotnējās testēšanas prasmes, jo skripti tiek veidoti pēc codeless tehnoloģijas.
Iebūvētas ALM integrācijas	5	Ir iespējama ALM integrācija ar HP ALM, Zephyr un Jira, kā arī ar Jenkins nepārtrauktās testēšanas ietvaru.
Pieejama dokumentācija	3.5	Dokumentācija ir labi strukturēta, tomēr nevarētu teikt, ka ļoti plaša un detalizēta.
Rīka instalēšanas un lietošanas vienkāršums	5	Darba gaitā darba autors instalēja TestArchitect Enterprise 9.0 izmēģinājuma versiju. Rīks ātrā instalācija tiešām aizņēma vien pāris minūtes un bija intuitīvi saprotama.
Testu skriptu izpilde	5	Testi izpildās ātri un nekļūdīgi, jo tiek izmantota codeless tehnoloģija.
Cena	3	Tas ir maksas rīks, kas piedāvā 30 dienu bezmaksas izmēģinājuma laiku, taču pēc šīm 30 dienām ir jāsazinās ar klientu konsultantu un jāvienojas par cenu. Pēc internetā atrodamās informācijas, tas varētu būt ap 3000 EUR gadā.
Record and playback funkcija	5	Record and playback funkcija ir pieejama, tā strādā nekļūdīgi un tā ir viegla lietošanā un saprotama gala lietotājam.
Lietotāja saskarne	4	Lietotāja saskarne ir visai parasta, bet tajā pašā laikā viegli saprotama un ne pārlieku aizpildīta ar visādām pogām un

		nevajadzīgām lietām. Pie grafiskā interfeisa iespējams vajadzētu piestrādāt, bet kopēji izskatās labi.
Operētājsistēmu atbalsts	3.5	TestArchitect atbalsta divas populāras operētājsistēmas: Windows un Linux.
Lietotņu atbalsts	5	TestArchitect atbalsta visus trīs populārākos lietotņu tipus: tīmekļa, darbvirsmas un mobīlās lietotnes.
Rīka apmācību process	4	Apmācības procesam darbā ar TestArchitect nevajadzētu būt pārlieku grūtam, taču varētu būt mazliet vairāk apgrūtinošs dēļ brīžiem skopās dokumentācijas.
Rīka tiešsaistes atbalsts	2	Atbalstu iespējams saņemt tikai piesakot savu problēmu un gaidot rindā, kā arī ir iespējams pazvanīt, bet autors veicot trīs zvanus, tā arī netika apkalpots.
Atkļūdošanas atbalsts	5	Ļoti stiprs un intuitīvi saprotams atkļūdošanas atbalsts, kas palīdz noteikt kļūdas faktisko/iespējamo iemeslu, kā arī palīdz to atrisināt.
Atskaišu, logu izveide un glabāšana	3.5	Atskaites netiek ģenerētas automātiski, to izveide ir jāaktivizē manuāli, taču pašas atskaites ir labi strukturētas un parāda visas nepieciešamās detaļas. Log faili arī tiek pieglabāti un ir pieejami.
Dažādu testēšanas tipu atbalsts	3.5	TestArchitect atbalsta dažādus testēšanas veidus un metodikas, piemēram funkcionālo, API un regresijas testēšanu.
Datubāžu lietotņu atbalsts	5	Rīks ļoti labi sadarbojas/strādā (ielasa SQL vaicājumus no DB) ar populārākajām datubāžu lietotnēm.
Data-driven testēšana	4	TestArchitect atbalsta data-driven testēšanu saskarņu un datu bāžu arhitektūrā. TestArchitect nodrošina piekļuvi, ielasīšanu un apstrādi no dažādiem ārējiem datu avotiem, tādiem kā: Excel vai SQL.
Rīka evolūcija ar gadiem	4	Rīks pastāvīgi tiek uzturēts un laiku pa laikam atjaunināts. Diezgan bieži tiek izlaisti jauni laidieni vai versijas ar jaunu vai uzlabotu funkcionalitāti.
Koda/skriptu atkārtota izmantojamība	5	Var pieglabāt un atkārtoti izmantot gan jaunizveidotas funkcijas un veselus skriptus, gan izmantot funkcijas no milzīgās bibliotēkas, kas ir iestrādāta rīkā.

Atbilstība jaunākajām tendencēm	4	Ir ieviesta un turpina aktīvi attīstīties AI/ML. Tiek ļoti aktīvi virzīta uz priekšu un popularizēta bezkoda automatizētā testēšana.

Aritmētiskais vidējais vērtējums par visiem kritērijiem TestArchitect rīkam ir: **4.204**, ko var uzskatīt par ļoti augstu kopējo vērtējumu.

6.8. WinAppDriver rīks

6.8.1. Rīka apraksts

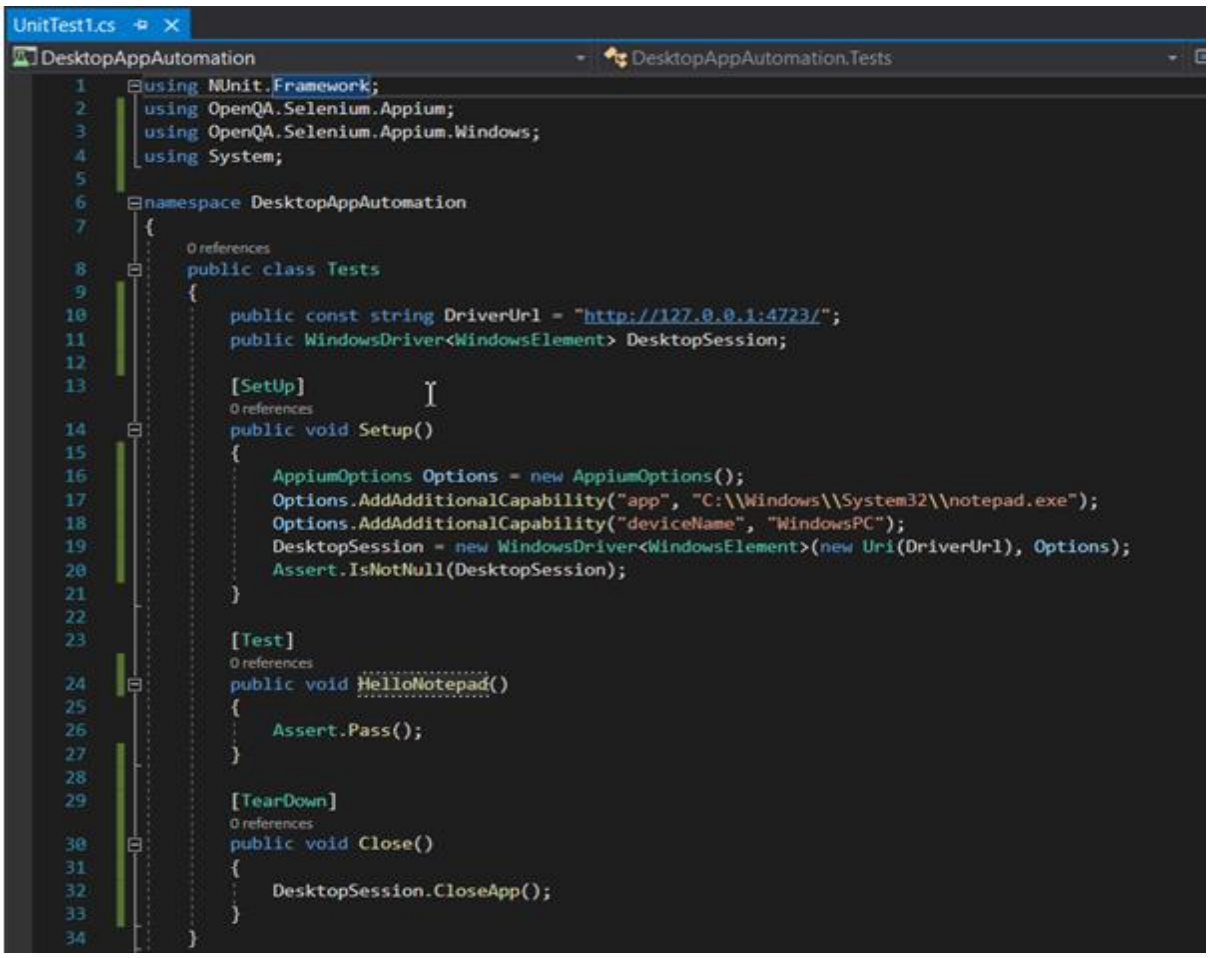
WinAppDriver ir balstīts uz populārā WebDriver protokola – standarta protokola mobīlo un tīmekļa lietotņu testēšanai.

WinAppDriver ir specializēts Windows darbvirsmu aplikāciju testēšanas rīks, kuru Microsoft izstrādājis kā atvērtā pirmkoda projektu, tā ir Appium implementēšana, kas galvenokārt ir mobilo lietotņu sistēma, kuras pamatā ir Selenium. Tāpēc WinAppDriver ir Selenium tipa (pēc līdzības) automatizācijas sistēma. Šis rīks apvieno labāko no diviem rīkiem, no vienas puses, tas ietver lielāko daļu, tagad novecojušās, CodedUI tehnoloģiju un apvieno to ar Selenium elastību, lietošanas vienkāršību un mainību [23].

Tāpat kā Selenium, WinAppDriver ir bibliotēku kopums, ko var integrēt jebkurā *Test Runner*, kas atbalsta Appium. Piemēram, WinAppDriver skriptus var izstrādāt un izpildīt ar Visual Studio MSTest [23].

Rīks atbalsta gan 32-bitu, gan 64-bitu lietotnes, kuras ir izstrādātas vienā no šīm vidēm: WinForms; Windows Presentation Foundation (WPF); Universal Windows Platform (UWP); Classic Windows (Win32).

WinAppDriver rīkam nav pašam savas skriptēšanas valodas, taču tajā var rakstīt skriptus automatizēšanai populārākajās programmēšanas valodās, tādās kā: C#, Java, Python, Ruby. Kā arī skripta piemēru, rakstītu WinAppDriver lietotnē C# programmēšanas valodā var apskatīt *attēlā 6.8.1.1*.



6.8.1.1. att. **WinAppDriverī rakstītā koda paraugs**

Līdzīgi kā Selenium, arī WinAppDriver elementi tiek atrasti pēc identifikācijas īpašībām, piemēram, klase, Id, nosaukums, Xpath utt. WinAppDriver nav iebūvēta elementu atpazīšanas rīka, tomēr var izmantot trešās puses rīkus, kā piemēram: Inspect.exe, kā arī ar Spy++ - objektu identifikācijas rīks, kas tiek piegādāts kopā ar Visual Studio [23].

WinAppDriver dokumentācija ir diezgan viegli atrodama, taču tā ir ļoti nestrukturēta un nedetalizēta, tajā nav nekādu atsevišķu video/audio apmācību, kā arī ir tikai viena soli-pa-soli uzrakstīta apmācība: kā uzinstalēt WinAppDriver.

6.8.2. Rīka novērtējums

Balstoties uz autora izlasīto informāciju par konkrēto rīku, kā arī uz neliela praktiskā rīka pielietojuma pamata, ko autors speciāli veica, lai spētu objektīvāk izvērtēt rīku, tika veikti kopējie secinājumi par katru no kritērijiem, kā arī katram no tiem sniegts atbilstošs vērtējums, kas parādīts *tabulā 6.8.2.1.* zemāk:

6.8.2.1. tabula. **WinAppDriver rīka novērtējuma tabula**

Kritērijs	**Vērtējums**	**Komentārs**
Programmēšanas	5	Rīks atbalsta vairākas populārākās programmēšanas

valodu atbalsts		valodas, tādas kā Java, Python, Ruby, C#.
Iebūvēts UI elementu/objektu atpazīšanas rīks	2	Nav iebūvēts UI elementu/objektu atpazīšanas rīks. Tā vietā kopā ar WinAppDriver iesaka lietot Inspect.exe vai Spy++, kas nav paši modernākiem, ērtākiem un vieglākie izpratnē rīki.
Nepieciešamās kodēšanas prasmes	3	Testēšanas rīks ir gana labs un parocīgs pieredzējušiem skriptu rakstītājiem, it īpaši, ja tiek agrāk ir strādājuši ar Selenium, taču testētājiem, kas līdz šim nav saskārušies ar programmēšanu un skriptu rakstīšanu, tas nav pats vieglākais rīks, jo šeit ir nepieciešamas priekšzināšanas par to kā rakstīt skriptu.
Iebūvētas ALM integrācijas	1	WinAppDriver nesniedz nekādu iebūvētu atbalstu ALM integrācijām
Pieejama dokumentācija	2	WinAppDriver rīka pieejamā dokumentācija ir grūti atrodama, diezgan skopa, neuztverama un nestrukturēta. Tajā nav pamācošu video/audio failu, kā arī nav soli-pa-solim pamācību.
Rīka instalēšanas un lietošanas vienkāršums	4	Rīks ir ļoti viegli un ātri instalējams, taču nepieciešams obligāti izpildīt pāris priekšnosacījumus.
Testu skriptu izpilde	4	Testi izpildās diezgan ātri un bieži vien korekti, taču pietiekami ilgi nākas gaidīt, lai sāktos darbība izvēlētajā aplikācijā.
Cena	5	WinAppDriver ir atvērtā pirmkoda rīks un tas ir pieejams pilnīgi bezmaksas.
Record and playback funkcija	1	Ir pieejama kaut kāda UI recorder funkcija, taču darba autoram nesanāca izdarīt tā, lai viņa strādātu korekti.
Lietotāja saskarne	2	Lietotāja saskarne ir ļoti parasta, ne sevišķi pārskatāma un nav labi pielāgota gala lietotāja vajadzībām.
Operētājsistēmu atbalsts	2	WinAppDriver atbalsta tikai Windows operētājsistēmu.
Lietotņu atbalsts	2	WinAppDriver atbalsta tikai Windows darbvirsmas lietotnes.

Rīka apmācību process	2	Apmācību process darbā ar WinAppDriver ir pagrūts, jo nav pieejamas normālas dokumentācijas.
Rīka tiešsaistes atbalsts	0	Nav pieejams nekāds tiešsaistes atbalsts.
Atkļūdošanas atbalsts	3	Ir pieejama atkļūdošanas funkcija, taču tā ne vienmēr darbojas pareizi.
Atskaišu, logu izveide un glabāšana	1.5	Atskaites netiek ģenerētas automātiski, to izveide ir jāaktivē manuāli. Pašas atskaites nav sevišķi caurskatāms, kā arī nav labi noformētas.
Dažādu testēšanas tipu atbalsts	2	WinAppDriver pārsvarā atbalsta tikai vienību un funkcionālo testēšanu.
Datubāžu lietotņu atbalsts	0	Rīks neapstrādā pieprasījumus no datubāžu lietotnēm.
Data-driven testēšana	0	WinAppDriver neatbalsta data-driven testēšanu saskarņu un datu bāžu arhitektūrā.
Rīka evolūcija ar gadiem	1	Rīks it kā tiek uzturēts, bet reāli jauna funkcionalitāte tajā parādās ļoti reti. Pēdējie uzlabojumi rīkā tika veikti vēl 2020. gadā.
Koda/skriptu atkārtota izmantojamība	0	Nav iespējams pieglabāt un atkārtoti izmantot skriptus.
Atbilstība jaunākajām tendencēm	0	Neatbilst jaunākajām tendencēm, jo jebkāda jauna funkcionalitāte rīkā parādās ļoti reti.

Aritmētiskais vidējais vērtējums par visiem kritērijiem WinAppDriver rīkam ir: **1.931**, kas ir uzskatāms par ļoti zemu vērtējumu.

6.9. SikuliX rīks

6.9.1. Rīka apraksts

SikuliX ir atvērtā pirmkoda bezmaksas rīks, kas spēj automatizēt visu, ko ir iespējams ieraudzīt uz darbvirsmas ekrānā, kurš darbojas Windows, Mac vai Linux/Unix

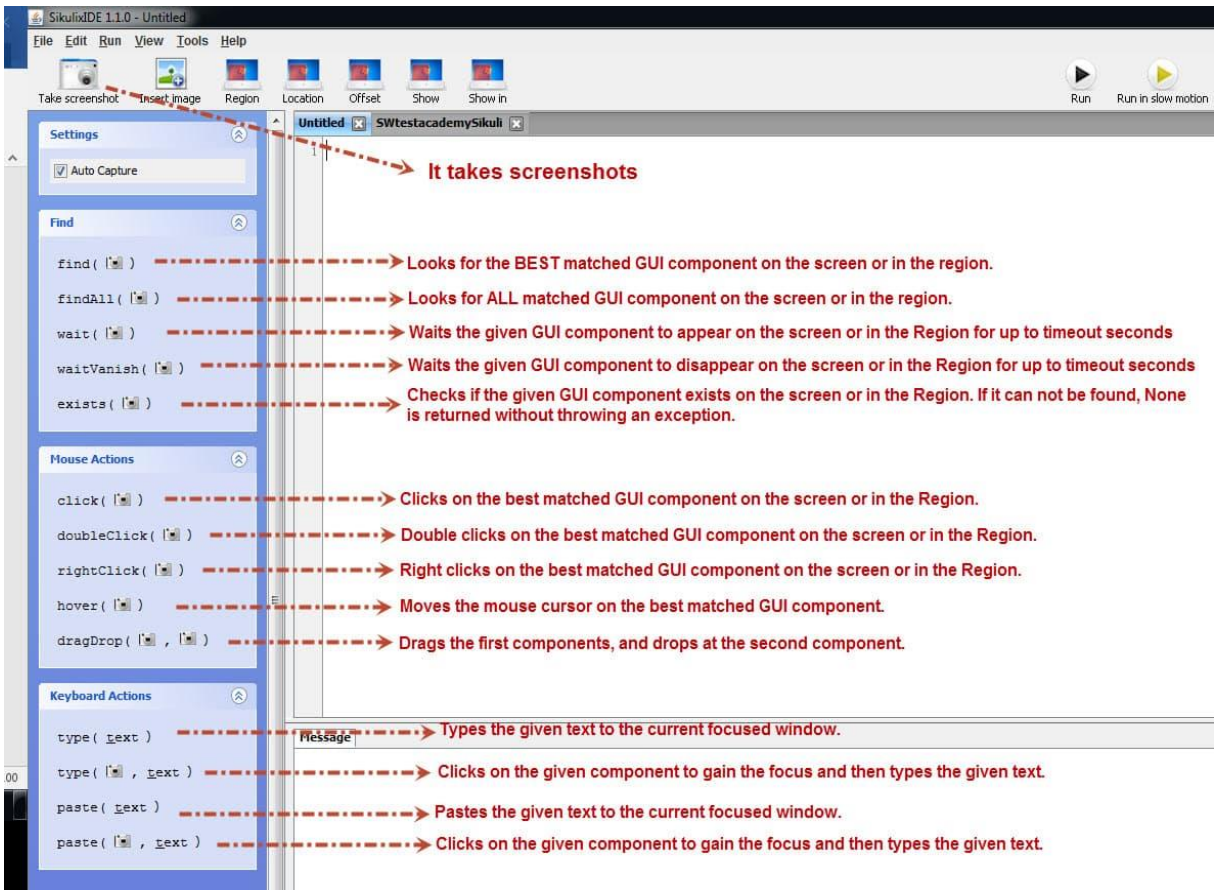
operētājsistēmā. Lai identificētu GUI komponentus, SikuliX izmanto attēlu atpazīšanas tehnoloģiju, ko nodrošina OpenCV [15].

SikuliX ir vizuāla pieeja GUI automatizēšanai, izmantojot ekrānuzņēmumus. SikuliX ļauj veikt GUI elementu ekrānuzņēmumu (piemēram, rīkjoslas pogu, ikonu vai dialoglodziņu) un vaicāt palīdzību sistēmai/mašīnai, un automatizēt darbības ar elementiem izmantojot ekrānuzņēmumus, nevis elementu nosaukumus. SikuliX ir arī integrēta vizuālo skriptu API, kas palīdz automatizēt GUI iterācijas, izmantojot attēlu modeļus peles un tastatūras notikumiem/darbībām, kā arī SikuliX ir aprīkots ar optisko simbolu atpazīšanu (OCR), un to var izmantot, lai meklētu tekstu attēlos. Sīkāk lietotāja grafiskā interfeisa skaidrojums ir atrodams zemāk, *attēlā 6.9.1.*

Visa augstāk aprakstītā tehnoloģija ir ērta darbvirsmas un tīmekļa lietotņu automatizācijai, gadījumos kad nav viegli piekļūt GUI iekšējiem elementiem vai tās lietotnes vai tīmekļa lapas avota kodam.

SikuliX ir viegli apgūt un iemācīties ar to strādāt pat ne sevišķi pieredzējušiem testētājiem, jo pētījumi liecina, ka elementu meklēšana un koda rakstīšana ar ekrānuzņēmumiem ir vienkārša un ātrāka, salīdzinājumā, piemēram ar atslēgvārdiem.

SikuliX kā primāro skriptēšanas valodu atbalsta Python, bet ir iespējams izmantot arī Ruby. Papildus iespējams izmantot SikuliX API Java programmā, pievienojot sikulixapi.jar failu Java projektā, importējot nepieciešamās bibliotēkas un rakstot kodu [16].



6.9.1. att. **SikuliX lietotāja interfeisa apraksts un paskaidrojums no:**
https://www.swtestacademy.com/quick-start-to-sikulix/

6.9.2. Rīka novērtējums

Balstoties uz autora izlasīto informāciju par konkrēto rīku, kā arī uz neliela praktiskā rīka pielietojuma pamata, ko autors speciāli veica, lai spētu objektīvāk izvērtēt rīku, tika veikti kopējie secinājumi par katru no kritērijiem, kā arī katram no tiem sniegts atbilstošs vērtējums, kas parādīts *tabulā 6.9.2.1.* zemāk:

6.9.2.1. tabula. **SikuliX rīka novērtējuma tabula**

Kritērijs	**Vērtējums**	**Komentārs**
Programmēšanas valodu atbalsts	4	Rīks atbalsta trīs populāras skriptu rakstīšanas valodas: Java, Python un Ruby, kas varētu būt pilnīgi pietiekami.
Iebūvēts UI elementu/objektu atpazīšanas rīks	4.5	Ir iebūvēts unikāls elementu atpazīšanas rīks caur ekrānuzņēmumiem, kas spēj atpazīt un izspiegot ne tikai elementus/objektus, bet arī optiskos simbolus un attēlus.
Nepieciešamās kodēšanas prasmes	4	Rīks neprasa nekādas dižas sākotnējās kodēšanas prasmes pavisam vienkāršu skriptu rakstīšanai.
Iebūvētas ALM integrācijas	0	Nav iespējota ALM integrācija.
Pieejama dokumentācija	2.5	Dokumentācija ir viegli atrodama un pieejama, tomēr nevarētu teikt, ka tā ir labi strukturēta, plaša un detalizēta.
Rīka instalēšanas un lietošanas vienkāršums	5	Darba gaitā darba autors instalēja SikuliX 2.0.5. jeb pēdējo stabilo versiju. Rīks instalācija aizņēma vien pāris minūtes un bija intuitīvi saprotama.
Testu skriptu izpilde	3.5	Testi izpildās diezgan ātri un bieži vien nekļūdīgi.
Cena	5	Tas ir atvērtā pirmkoda bezmaksas rīks.
Record and playback funkcija	5	Record and playback funkcija ir pieejama, tā strādā nekļūdīgi un tā ir viegla lietošanā un saprotama gala lietotājam.
Lietotāja saskarne	3.5	Lietotāja saskarne ir visai parasta, bet tajā pašā laikā viegli saprotama un ne pārlieku aizpildīta ar liekām lietām.
Operētājsistēmu atbalsts	5	SikuliX atbalsta visas trīs populārākās operētājsistēmas: Windows, MacOS un Linux.
Lietotņu atbalsts	3.5	SikuliX atbalsta divus no trim populārākajiem lietotņu

		tipiem: tīmekļa un darbvirsmas.
Rīka apmācību process	3.5	Apmācības procesam, kā jau minēts iepriekš, nevajadzētu būt pārlieku grūtam. Vienīgais, tas varētu tikt apgrūtināts dēļ skopas dokumentācijas.
Rīka tiešsaistes atbalsts	1	Atbalstu iespējams saņemt tikai piesakot savu problēmu forumā un gaidot pagaidām kāds atbildēs uz to, un nekā savādāk.
Atkļūdošanas atbalsts	5	Ļoti stiprs un intuitīvi saprotams atkļūdošanas atbalsts, kas palīdz noteikt kļūdas faktisko/iespējamo iemeslu, kā arī palīdz to atrisināt.
Atskaišu, logu izveide un glabāšana	4.5	Atskaites tiek ģenerētas automātiski, pašas atskaites ir labi strukturētas, viegli lasāmas un parāda visas nepieciešamās detaļas. Atskaitēs iespējams redzēt arī iepriekšējo testu rezultātus, kā arī atskaites tiek parādītas gan grafiski, gan caur komandu logu. Log faili arī tiek pieglabāti un ir pieejami.
Dažādu testēšanas tipu atbalsts	3	SikuliX atbalsta tikai pāris testēšanas veidus: vienībtestēšanu, funkcionālo un GUI.
Datubāžu lietotņu atbalsts	2.5	Ir iespējams, bet diezgan problemātiski savienot SikuliX ar populārākajām datubāzēm.
Data-driven testēšana	3	SikuliX atbalsta, tam ir pieeja datiem un tas var tos apstrādāt no dažiem ārējiem avotiem.
Rīka evolūcija ar gadiem	2.5	Rīks, pēc izstrādātāju komentāriem, tiek uzturēts, taču tam ļoti reti iznāk stabilas versijas un laidieni ar jaunu funkcionalitāti vai izlabotu veco.
Koda/skriptu atkārtota izmantojamība	4	Var pieglabāt un atkārtoti izmantot gan jaunizveidotas funkcijas un veselus skriptus speciālās bibliotēkās.
Atbilstība jaunākajām tendencēm	3	Ir ieviesta AI/ML. Ir veiktas darbības un centieni, lai pārietu uz bezkoda skriptēšanu.

Aritmētiskais vidējais vērtējums par visiem kritērijiem SikuliX rīkam ir: **3.522**, ko var uzskatīt par diezgan augstu vērtējumu priekš bezmaksas atvērtā pirmkoda rīka.

6.10. AutoIt rīks

6.10.1. Rīka apraksts

AutoIt ir gan bezmaksas automatizācijas rīks, gan atsevišķa skriptēšanas valoda, kas speciāli radīta Windows GUI automatizēšanai, kā arī skriptu rakstīšanai. Šīs skriptēšanas valodas piemērs ir atrodams zemāk, *attēlā 6.10.1.1.* Rīks ir speciāli radīts, lai tas aizņemtu pēc iespējas mazāk vietas uz lietotāja cietā diska, kā arī, lai tas spētu darboties patstāvīgi, bez trešo pušu programmatūru paralēlas darbības. Rīks atbalsta gan 32-bitu, gan 64-bitu lietotnes.

AutoIt parasti izmanto Microsoft Windows utilītprogrammatūras ražošanai un ikdienas uzdevumu automatizēšanai, piemēram, sistēmu pārvaldībai, uzraudzībai, apkopei vai programmatūras instalēšanai. To lieto arī, lai simulētu lietotāju mijiedarbību ar sistēmu automatizējot attiecīgo skriptu.

AutoIt ir BASICam līdzīga sintakse, kas nozīmē, ka lielākajai daļai cilvēku, kuri jebkad ir uzrakstījuši skriptu vai lietojuši augsta līmeņa valodu, vajadzētu viegli izprast šīs skriptēšanas valodas būtību.

Papildus rīka labums ir fakts, ka ar rīku izveidotos skriptus var pārkonvertēt par neatkarīgām izpildāmām programmām (.exe failiem). Tas nozīmē, ka izveidotos skriptus var viegli izmantot arī uz datoriem, uz kuriem nav instalēta AutoIt programmatūra. Tas izdarāms ar Aut2Exe programmatūru, kas nāk līdzi standarta AutoIt instalācijā [16].

Rīks piedāvā peles un klaviatūras darbību simulāciju, kā arī dažādu darbību veikšanu ar Windows formām, piemēram, loga samazināšana, paplašināšana, pārvietošana, atvēršana, aizvēršana utt. Īsumā sakot, ar šo rīku vajadzētu varēt ar Windows formām izdarīt visu to, ko varētu izdarīt jebkurš lietotājs [16].

```
#AutoIt3Wrapper_Au3Check_Parameters=-d -w 1 -w 2 -w 3 -w- 4 -w 5 -w 6 -w- 7

; Checks To See If The Internet Is Connected

ConsoleWrite("Internet Is Connected" & " = " & _IsInternetConnected() & @CRLF) ; ( Returns "True" Or "False" )

Func _IsInternetConnected()
    Local $aReturn = DllCall('connect.dll', 'long', 'IsInternetConnected')
    If @error Then
        Return SetError(1, 0, False)
    EndIf
    Return $aReturn[0] = 0
EndFunc ;==>_IsInternetConnected
```

6.10.1.1. att. **AutoIt skriptēšanas valodas koda piemērs**

6.10.2. Rīka novērtējums

Balstoties uz autora izlasīto informāciju par konkrēto rīku, kā arī uz neliela praktiskā rīka pielietojuma pamata, ko autors speciāli veica, lai spētu objektīvāk izvērtēt rīku, tika veikti kopējie secinājumi par katru no kritērijiem, kā arī katram no tiem sniegts atbilstošs vērtējums, kas parādīts *tabulā 6.10.2.1.* zemāk:

6.10.2.1. tabula. **AutoIt rīka novērtējuma tabula**

Kritērijs	Vērtējums	Komentārs
Programmēšanas valodu atbalsts	2	AutoIt izmanto pašizveidotu programmēšanas valodu skriptu rakstīšanai, kuru vēl nāksies apgūt.
Iebūvēts UI elementu/objektu atpazīšanas rīks	2	Ir iebūvēts AU3Recorder, kas ieraksta darbības, taču pats rīks izmanto ekrāna koordinātas, lai noteiktu, kur jāveic peles klikšķi, kas nav sevišķi parocīgi.
Nepieciešamās kodēšanas prasmes	2	Ja izmanto ieraksta funkciju, tad nav nepieciešamas dižas kodēšanas prasmes, bet ja vēlas kaut ko pierakstīt atsevišķi, tad to izdarīt būs pagrūti.
Iebūvētas ALM integrācijas	0	Nav iespējotas ALM integrācijas ar citiem rīkiem.
Pieejama dokumentācija	2	Dokumentācija ir viegli atrodama un pieejama, tomēr nevarētu teikt, ka tā ir labi strukturēta, plaša un detalizēta, nav arī kaut kādu grafisku attēlojumu.
Rīka instalēšanas un lietošanas vienkāršums	5	Darba gaitā darba autors instalēja AutoIt v3.3 14.3 jeb pēdējo stabilo versiju. Rīks instalācija aizņēma vien pāris minūtes un bija intuitīvi saprotama.
Testu skriptu izpilde	2	Testi izpildās diezgan ātri, taču pēc dažām reizēm testi izpildās kļūdaini dēļ koordinātu nobīdēm.
Cena	5	Tas ir atvērtā pirmkoda bezmaksas rīks.
Record and playback funkcija	2.5	Ieraksta funkcija ir pieejama, taču tā ir derīga tikai skripta rakstīšanas brīdī jeb to nevar atspēlēt vēlreiz vai kā pieglabāt.
Lietotāja saskarne	2	Lietotāja saskarne ir ļoti maza un parasta, tomēr tajā iekšā ir sagrūzts ļoti daudz, kas padara to par neparocīgu.
Operētājsistēmu atbalsts	2	AutoIt atbalsta tikai Windows operētājsistēmu.
Lietotņu atbalsts	2	AutoIt atbalsta tikai Windows darbvirsmas lietotnes.
Rīka apmācību process	2.5	Apmācības procesam, kā jau minēts iepriekš, nevajadzētu būt pārlieku grūtam. Vienīgais, tas varētu tikt apgrūtināts dēļ skopas dokumentācijas un jaunas programmēšanas valodas.
Rīka tiešsaistes atbalsts	1	Atbalstu iespējams saņemt tikai piesakot savu problēmu forumā un gaidot pagaidām kāds atbildēs uz to, un nekā

		savādāk.
Atkļūdošanas atbalsts	2	Nav pieejams gandrīz nekāds atkļūdošanas atbalsts.
Atskaišu, logu izveide un glabāšana	0	Rīkā nav iebūvēta veikto soļu atskaišu ģenerēšana. Vienīgais veids, kā iegūt atskaiti par veiktajām darbībām, ir integrēt AutoIt ar kādu citu rīku
Dažādu testēšanas tipu atbalsts	2	Rīks atbalsta tikai pāris testēšanas veidus: vienībtestēšanu un GUI.
Datubāžu lietotņu atbalsts	0	Nav pieejams datubāžu atbalsts.
Data-driven testēšana	0	Nav pieejams datu imports un apstrāde no ārējiem avotiem.
Rīka evolūcija ar gadiem	2.5	Rīks, pēc izstrādātāju komentāriem, tiek uzturēts, taču tam ļoti reti iznāk stabilas versijas un laidieni ar jaunu funkcionalitāti vai izlabotu veco.
Koda/skriptu atkārtota izmantojamība	1	Ierakstītos skriptus praktiski nav iespējams pielietot atkārtoti.
Atbilstība jaunākajām tendencēm	2	Ir veiktas darbības un centieni, lai pārietu uz bezkoda skriptēšanu.

Aritmētiskais vidējais vērtējums par visiem kritērijiem AutoIt rīkam ir: **1.885**, ko var uzskatīt par ļoti zemu vērtējumu, pat priekš bezmaksas rīka.

6.11. Winium rīks

6.11.1. Rīka apraksts

Winium ir atvērtā pirmkoda automatizācijas rīks, ko izmanto darbvirsmas lietotņu automatizēšanai. Uzbūvēts uz Selenium bibliotēkas pamata, lai mijiedarbotos ar Windows darbvirsmas lietotnēm [21].

Winium darbojas ar visām WebDriver saderīgām programmēšanas valodām, kā piemēram: Java, Objective-C, JavaScript ar Node.js PHP, Python, Ruby, C #, Perl ar Selenium WebDriver API un valodai specifiskām klientu bibliotēkām [21].

Rīks atbalsta un palīdzēs notestēt dažāda veida lietotnes, kuras ir tikušas izstrādātas sekojošās vidēs: WPF un WinForms [21].

Darbs ar Winium ir diezgan ātrs un vienkāršs, ja testētājam ir pieredze un priekšzināšanas darbā ar Selenium. Winium izmanto UISpy.exe vai Inspect.exe (pieejama Windows pēc noklusējuma), lai identificētu elementus Windows lietotnē [21].

Winium var izmantot kopā ar jebkuru no Front-end testēšanas bibliotēkām, piemēram Cucumber, Gauge vai Gherkin, lai lasītu testa automatizācijas soļus. Ir iespēja arī implementēt jebkuru no automatizācijas ietvarstruktūrām(*framework*), piemēram Page Object Model Framework, Data Driven Framework, Keyword Key Framework vai Robot Framework [21].

Neskatoties uz vairākām priekšrocībām, atvērtā pirmkoda rīkos galvenais ir aktīvā uzraudzība, apkope un atkļūdošana. Taču kopš 2016. gada Winium izstrādātāji to neuztur, tāpēc, ja ir vēlēšanās izmantot šo rīku, ir jārēķinās ar to, ka kļūdas netiks novērstas un rīks vienmēr paliks tieši tāds, kāds viņš ir šobrīd.

Winium nesniedz iebūvētu atbalstu ar ALM integrācijām, piemēram HP Center, kā arī tajā nav iebūvēts UI inspekcijas rīks, tāpēc Winium gadījumā tiek izmantoti trešās puses rīki, kas nav īpaši komfortabli izmantošanā, kā piemēram Inspect.exe.

6.11.2. Rīka novērtējums

Balstoties uz autora izlasīto informāciju par konkrēto rīku, kā arī uz neliela praktiskā rīka pielietojuma pamata, ko autors speciāli veica, lai spētu objektīvāk izvērtēt rīku, tika veikti kopējie secinājumi par katru no kritērijiem, kā arī katram no tiem sniegts atbilstošs vērtējums, kas parādīts *tabulā 6.11.2.1.* zemāk:

6.11.2.1. tabula. **Winium rīka novērtējuma tabula**

Kritērijs	**Vērtējums**	**Komentārs**
Programmēšanas valodu atbalsts	5	Rīks atbalsta vairākas programmēšanas valodas, tādas kā Java, Objective-C, JavaScript ar Node.js PHP, Python, Ruby, C #, Perl.
Iebūvēts UI elementu/objektu atpazīšanas rīks	2	Nav iebūvēts UI elementu/objektu atpazīšanas rīks. Tā vietā kopā ar Winium iesaka lietot Inspect.exe vai UISpy.exe, kas nav paši modernākiem, ērtākiem un vieglākie izpratnē rīki.
Nepieciešamās kodēšanas prasmes	3	Testēšanas rīks ir gana labs un parocīgs pieredzējušiem skriptu rakstītājiem, it īpaši, ja tiek agrāk ir strādājuši ar Selenium, taču testētājiem, kas līdz šim nav saskārušies ar

		programmēšanu un skriptu rakstīšanu, tas nav pats vieglākais rīks.
Iebūvētas ALM integrācijas	1	Winium nesniedz nekādu iebūvētu atbalstu ALM integrācijām
Pieejama dokumentācija	2	Winium rīka dokumentācija ir diezgan skopa un grūti atrodama. Tajā nav pamācošu video.
Rīka instalēšanas un lietošanas vienkāršums	2	Pirms paša rīka uzstādīšanas, ir priekšnosacījumi, kas jāizpilda un programmas, kas jāuzstāda, tāpēc visa instalācija ir diezgan laikietilpīga un nav tik vienkārša, kā arī rīks netiek pastāvīgi uzturēts kopš 2016. gada.
Testu skriptu izpilde	4	Testi izpildās diezgan ātri un bieži vien korekti, taču pietiekami ilgi nākas gaidīt, lai sāktos darbība izvēlētajā aplikācijā.
Cena	5	Winium ir atvērtā pirmkoda rīks un tas ir pieejams pilnīgi bezmaksas.
Record and playback funkcija	3.5	Record and playback funkcija ir pieejama, bet tā nav diezgan komforta un parocīga.
Lietotāja saskarne	2	Kopēji ļoti vājš grafiskais interfeiss, tas nav intuitīvs, un ir ļoti vienkāršs.
Operētājsistēmu atbalsts	2.5	Rīks atbalsta tikai Windows lietotnes.
Lietotņu atbalsts	2.5	Winium atbalsta mobīlās un darbvirsmas lietotnes iekš vienas operētājsistēmas.
Rīka apmācību process	2.5	Apmācības process, kā arī pieejamā dokumentācija ir zemā līmenī, līdz ar to iemācīties pareizi pielietot šo rīku būs ilgāk un grūtāk.
Rīka tiešsaistes atbalsts	0	Ja autors ir pareizi sapratis, tad nav pieejams nekāds tiešsaistes atbalsts.
Atkļūdošanas atbalsts	2.5	Ir pieejams atkļūdošanas atbalsts, tomēr tas nav perfekts, kā arī nepiedāvā nekādus iespējamos risinājumus.
Atskaišu, logu izveide un glabāšana	2	Netiek veidotas nekādas atskaites. Ir iespējama logu apskate.

Dažādu testēšanas tipu atbalsts	3	Winium atbalsta tādus testēšanas veidus un metodes kā funkcionālo, GUI testēšanu un regresijas testēšanu. Taču ir daudzi veidi/tipi, kurus šis rīks neatbalsta.
Datubāžu lietotņu atbalsts	4	Rīks ielasa un māk apstrādāt datubāžu vaicājumus.
Data-driven testēšana	3.5	Rīks nodrošina piekļuvi un ielasīšanu no dažādiem ārējiem avotiem kā CSV, Excel vai MySQL.
Rīka evolūcija ar gadiem	0	Rīka evolūcija pilnībā apstājās 2016. gadā.
Koda/skriptu atkārtota izmantojamība	0	Rīkā nav iespējams atkārtoti izmantot/pieglabāt skriptus, funkcijas.
Atbilstība jaunākajām tendencēm	0	Rīks neatbilst jaunākajām tendencēm, jo netiek pilnveidots un uzlabots kopš 2016. gada

Aritmētiskais vidējais vērtējums par visiem kritērijiem Winium rīkam ir: **2.363**, ko var uzskatīt par ļoti zemu vērtējumu.

6.12. Robots rīks

6.12.1. Rīka apraksts

Robots ir bezmaksas, bet ne atvērtā pirmkoda rīks, kas ir izstrādāts un tiek lietots vienas kompānijas ietvaros, kurā darba autors šobrīd strādā par automatizācijas inženieri.

Robots ir būvēts balstoties uz Microsoft UI Automation API slāni, taču Robots pats par sevi ir atsevišķs frameworks, kurš strādā uz .NET bāzes un ir rakstīts C# programmēšanas valodā., kā arī skriptu rakstīšanas valoda ir C#, un šāda skripta piemēru var redzēt 1.pielikumā.

Robots frameworks ir paredzēts tam, lai tajā iekšā ievietotu Microsoft API slāni, jo Robots framework slānis ir uzrakstīts saprotamā, cilvēcīgā valodā un uztaisīts maksimāli ērts un parocīgs gala lietotājam, jo tādā veidā nav nepieciešama pastāvīga tieša komunikācija ar Windows API, un tieši tāpēc skripta kods ir viegli saprotams un uzskatāms. Piemēram, ja šāds slānis netiku iestrādāts, tad, lai atrastu konkrētu pogu Windows API slānī, būtu jānorāda kāds konkrēti elements tiek meklēts, tā tips, raksturojošās īpašības un citas atpazīšanas zīmes, taču

pateicoties Robots framework slānim, šis viss jau ir iebūvēts konkrētā metodē. Taču ir paredzēts arī, ka var lietot Windows API slāni "pa taisno", ja nepieciešams.

Robots specializējas uz Windows darbvirsmas lietotņu automatizāciju, taču tiek lietots arī REST automatizācijai, kā arī tam ir iestrādes tīmekļa lietotņu automatizācijā.

Robots ir responsīvs rīks, tas nozīmē, ka vienmēr, pirms jebkuras darbības, tiek pārbaudīts vai konkrētā lietotne ir pabeigusi iepriekšējo soli un vai vispār tā ir responsīva šajā brīdī.

Robots frameworkā ir iestrādāta iespēja komunicēt tiešā veidā ar konkrēto lietotni, kas tiek izstrādāta un testēta kompānijas ietvaros, taču šādu iespēju var iestrādāt arī ar jebkuru citu lietotni.

Kopā ar Robots frameworku un automatizācijas rīku, ir izstrādāta arī atsevišķa lietotne sagatavju ģenerēšanai, UI objektu inspekcijai, gridu kopēšanai un XML failu ģenerēšanai – Script Writing Helper. Tā grafisko interfeisu var apskatīt zemāk 2.pielikumā.

Robots rīkam ir ļoti attīstīta iekšējā infrastruktūra, tāpēc nekādas papildus integrācijas ar ALM nav nepieciešamas. Robots rīka iekšējā infrastruktūrā ir izvietoti desmiti virtuālo mašīnu, kas veic automatizētos testus vai nu konkrētā laikā, vai pēc pieprasījuma. To sarakstu no iekšējās rīka infrastruktūras puses var atrast 3.pielikumā. Katrā iterācijā tiek glabāta atskaite ar ekrānuzņēmumiem, rezultātu salīdzināšanu (validācijām), kā arī log faili. Sarakstu ar ejošajām un pabeigtajām iterācijām no infrastruktūras puses var atrast 4.pielikumā. Atskaitēs ir pieejami arī grafiskie rādītāji. Tiek glabāta arī datubāze no konkrētās iterācijas ar izpildītajām darbībām. Testētajiem nav nepieciešamības veikt manipulācijas ar MySQL datubāzes lietotni, visu nepieciešamo ir iespējams izdarīt caur Robots rīka infrastruktūru.

6.12.2. Rīka novērtējums

Balstoties uz autora izlasīto informāciju par konkrēto rīku, kā arī uz neliela praktiskā rīka pielietojuma pamata, ko autors speciāli veica, lai spētu objektīvāk izvērtēt rīku, tika veikti kopējie secinājumi par katru no kritērijiem, kā arī katram no tiem sniegts atbilstošs vērtējums, kas parādīts *tabulā 6.12.2.1.* zemāk:

6.12.2.1. tabula. **Robots rīka novērtējuma tabula**

Kritērijs	**Vērtējums**	**Komentārs**
Programmēšanas valodu atbalsts	2	Rīks atbalsta tikai vienu skriptu programmēšanas valodu: C#
Iebūvēts UI elementu/objektu atpazīšanas rīks	5	Ir iebūvēts unikāls, UI elementu atpazīšanas rīks Script writing helper, kas ir ļoti daudzfunkcionāls.

Nepieciešamās kodēšanas prasmes	2.5	Rīks prasa sākotnējās kodēšanas prasmes un C# programmēšanas valodas zināšanas, jo arī kaut kādā mērā UI rīks palīdz rakstīt kodu, tomēr ir jābūt iepriekšējai pieredzei skriptu rakstīšanā, lai to uzrakstītu un automatizētu.
Iebūvētas ALM integrācijas	5	Nav nepieciešama papildu ALM integrācijas, jo Robots rīka infrastruktūra ir ļoti attīstīta un visu nepieciešamo var atrast/izsekot tur, taču ir iestrādāts nepārtrauktās integrācijas rīks Jenkins.
Pieejama dokumentācija	1	Dokumentācija ir ļoti skopa un pieejama tikai kompānijas darbiniekiem.
Rīka instalēšanas un lietošanas vienkāršums	3	Rīka instalācija ir diezgan vienkārša, tomēr ir jāveic noteiktas manipulācijas un manuāli soļi, lai tas tiktu veiksmīgi uzstādīts uz lietotāja datora.
Testu skriptu izpilde	5	Testi izpildās ātri un nekļūdīgi, ja vien pats kods ir korekti un nekļūdaini uzrakstīts.
Cena	5	Šobrīd tas ir bezmaksas rīks.
Record and playback funkcija	3	Record and playback funkcija, tā tiešā nozīmē nav pieejama, taču Script writing helperī ir iespējams ierakstīt katru soli/darbību, lai pēc tam būtu vieglāk uzrakstīt visu skriptu.
Lietotāja saskarne	4	Lietotāja saskarne ir ļoti parasta un tajā pašā laikā viegla un parocīga. Viss ir intuitīvi saprotams un viegli atrodams.
Operētājsistēmu atbalsts	2	Robots atbalsta tikai Windows operētājsistēmu.
Lietotņu atbalsts	2.5	Robots atbalsta Windows darbvirsmas lietotnes, REST lietotnes, kā arī ir iestrādes tīmekļa lietotņu atbalstā.
Rīka apmācību process	4	Apmācības process darbam ar Robots rīku ir samērā vienkāršs un labi saprotams, jo ir izveidots soli-pa-solim apmācošs dokuments par to kā uzsākt darbu ar Robots rīku.
Rīka tiešsaistes atbalsts	0	Nav pieejams nekāds tiešsaistes atbalsts, jo rīks tiek lietots tikai vienas kompānijas ietvaros.
Atkļūdošanas	5	Ļoti stiprs un intuitīvi saprotams atkļūdošanas atbalsts, kas

atbalsts		palīdz noteikt kļūdas faktisko/iespējamo iemeslu, kā arī palīdz to atrisināt.
Atskaišu, logu izveide un glabāšana	5	Atskaites tiek ģenerētas un pieglabātas automātiski, tieši tā pat kā log faili. Atskaites tiek ģenerētas XML un JSON formātos. Atskaitēs ir pieejami arī ekrānuzņēmumi, kā arī rezultātu salīdzināšana. Pašas atskaites ir viegli skatāmas un saprotamas gala lietotājam.
Dažādu testēšanas tipu atbalsts	4	Robots atbalsta vairākus testēšanas veidus, kā piemēram vienībtestēšanu, funkcionālo, GUI un regrestestēšanu.
Datubāžu lietotņu atbalsts	5	Rīks ļoti labi sadarbojas/strādā un ielasa vaicājumus no populārākajām datubāžu lietotnēm.
Data-driven testēšana	5	Robots atbalsta data-driven testēšanu saskarņu un datu bāžu arhitektūrā. Robots nodrošina piekļuvi, ielasīšanu un apstrādi no dažādiem ārējiem datu avotiem, tādiem kā: Excel, XML, CSV un SQL.
Rīka evolūcija ar gadiem	4	Rīks pastāvīgi tiek uzturēts un laiku pa laikam atjaunināts. Samērā bieži tiek izlaisti jauni laidieni vai versijas ar jaunu vai uzlabotu funkcionalitāti.
Koda/skriptu atkārtota izmantojamība	4	Var pieglabāt un atkārtoti izmantot jaunizveidotas funkcijas un veselus skriptus. Kā arī ir iespējams veidot koplietošanas bibliotēkas.
Atbilstība jaunākajām tendencēm	4	Ir ieviesta un turpina aktīvi attīstīties ML (infrastruktūra māk nolasīt bildi, saprast kas tā ir par kļūdu un kategorizēt to). Tuvākajā laikā ir doma pievērsties arī AI ieviešanai kļūdu meklēšanai. Ir ieviests un aktīvi virzīts RPA.

Aritmētiskais vidējais vērtējums par visiem kritērijiem Robots rīkam ir: **3.636**, ko var uzskatīt par ļoti augstu kopējo vērtējumu.

7. RĪKU KOPVĒRTĒJUMS

7.1. Rīku kopvērtējumu tabula

Zemāk ir redzama rīku kopvērtējumu tabula (*Tabula 7.1.*), kurā ar "*" nosaukuma sākumā ir atzīmēti bezmaksas rīki.

7.1. tabula. **Visu rīku kopvērtējumu tabula**

Kritērijs/Rīka nosaukums	*TestComplete*	*UFT*	*Katalon Studio*	*Squish*	*Tricentis Tosca*	*Ranorex*	*TestArchitect*	**WinAppDriver*	**SikuliX*	**AutoIt*	**Winium*	***Robots*
Programmēšanas valodu atbalsts	5	2	4	4	4.5	4	4	5	4	2	5	2
Iebūvēts UI atpazīšanas rīks	5	5	5	4	5	4	4.5	2	4.5	2	2	5
Nepieciešamās kodēšanas prasmes	4	4	4.5	4	5	5	5	3	4	2	3	2.5
Iebūvētas ALM integrācijas	4	4	5	5	5	5	5	1	0	0	1	5
Pieejamā dokumentācija	5	4	5	4.5	3.5	5	3.5	2	2.5	2	2	1
Rīka instalēšanas un lietošanas vienkāršums	5	4	5	5	5	5	5	4	5	5	2	3
Testu skriptu izpilde	3	4	5	5	4.5	4.5	5	4	3.5	2	4	5
Cena	2	2.5	3	3	2	2.5	3	5	5	5	5	5
Record and playback funkcija	5	5	5	5	5	5	5	1	5	2.5	3.5	3
Lietotāja saskarne	4	3.5	5	3	4	4	4	2	3.5	2	2	4
Operētājsistēmu atbalsts	2	2	5	5	2	2	3.5	2	5	2	2.5	2
Lietotņu atbalsts	5	4	5	5	4	5	5	2	3.5	2	2.5	2.5
Rīka apmācību	5	4	5	5	4.5	4	4	2	3.5	2.5	2.5	4

process												
Rīka tiešsaistes atbalsts	5	4	2	2	3	2.5	2	0	1	1	0	0
Atkļūdošanas atbalsts	5	5	5	5	5	5	5	3	5	2	2.5	5
Atskaišu, logu izveide un glabāšana	3.5	5	4	3	4	4	3.5	1.5	4.5	0	2	5
Dažādu testēšanas tipu atbalsts	5	3.5	4	3	4.5	4	3.5	2	3	2	3	4
Datubāžu lietotņu atbalsts	5	5	5	5	5	5	5	0	2.5	0	4	5
Data-driven testēšana	5	5	4.5	5	5	4	4	0	3	0	3.5	5
Rīka evolūcija ar gadiem	5	5	5	4	5	3	4	1	2.5	2.5	0	4
Koda/skriptu atkārtota izmantojamība	5	5	4	4	5	3	5	0	4	1	0	4
Atbilstība jaunākajām tendencēm	3.5	4	4	4	4.5	4	4	0	3	2	0	4
Rīka arifmētiskais vidējais vērtējums	4.4	4.1	4.5	4.2	4.3	4.1	4.2	1.9	3.5	1.9	2.4	3.6

7.2. Rīku kopvērtejuma analīze

Tika izvērtēti 14 dažādi automatizācijas rīki pēc 22 dažādiem kritērijiem. Tika izvērtēti gan bezmaksas un atvērtā pirmkoda, gan maksas rīki, gan tādi, kas specializējās tieši uz Windows darbvirsmas lietotnēm, gan tādi, kas atbalsta vairākas operētājsistēmas un lietotņu tipus.

Rīku salīdzināšana un katra rīka novērtēšana tika realizēta gan balstoties uz pieejamo dokumentāciju par rīku un pieejamo informāciju internetā, gan katra rīka praktisko pielietojumu konkrētam autora testēšanas darbam.

Pēc rīku padziļinātas analīzes un izvērtēšanas, darba autors ir novērojis sekojošu likumsakarību: bezmaksas, atvērtā pirmkoda rīki manāmi piekāpjas kvalitātes un funkcionāla ziņā maksas rīkiem.

Par labāko automatizācijas rīku, pēc rīku augstākā vidējā vērtējuma (kopējais iegūtais kritēriju baļļu skaits dalīts ar kopējo kritēriju skaitu) tiek atdzīts Kalaton Studio rīks, kura vidējais vērtējums par visiem kritērijiem ir 4.5, kas ir ļoti augsts vērtējums ņemot vērā kopēju kritēriju skaitu un to plašo specifiku, tāpēc, ja Jums ir iespēja investēt testēšanas automatizācijas rīkos, tad noteikti vajadzētu pievērst uzmanību tieši šim rīkam.

Par labāko bezmaksas automatizācijas rīku, no tiem, kas tika apskatīti šī darba ietvaros, tiek atdzīts SikuliX rīks ar vidējo vērtējumu: 3.5, tāpēc, ja Jūs izstrādājat Windows darbvirsmas lietotni un Jums šobrīd nav iespējas investēt testēšanas automatizācijas rīkos, tad noteikti Jūsu izvēle ir tieši SikuliX, jo pārējie bezmaksas atvērtā pirmkoda rīki manāmi piekāpjas tam.

7.3. Robots rīka salīdzināšana

Tā kā viens no galvenajiem bakalaura darba mērķiem bija arī saprast vai Robots rīks spēj konkurēt ar citiem populāriem un daudzfunkcionāliem rīkiem, kas šobrīd pieejami tirgū, kā arī objektīvi izvērtēt šo rīku kā konkurētspējīgu alternatīvu populāriem un izplatītiem automatizētās testēšanas rīkiem, tad tālāk *tabulā 7.3.1.* būs neliela labākā automatizācijas rīka (Katalon Studio) salīdzināšana attiecībā pret Robots rīku:

7.3.1. tabula. **Katalon Studio salīdzinājums ar Robots rīku**

Kritērijs/Rīka nosaukums	**Katalon Studio**	**Robots**
Programmēšanas valodu atbalsts	4	**2**
Iebūvēts UI atpazīšanas rīks	5	5
Nepieciešamās kodēšanas prasmes	4.5	**2.5**
Iebūvētas ALM integrācijas	5	5
Pieejamā dokumentācija	5	**1**
Rīka instalēšanas un lietošanas vienkāršums	5	**3**
Testu skriptu izpilde	5	5
Cena	3	5
Record and playback funkcija	5	**3**
Lietotāja saskarne	5	4
Operētājsistēmu atbalsts	5	**2**
Lietotņu atbalsts	5	**2.5**
Rīka apmācību process	5	4
Rīka tiešsaistes atbalsts	2	**0**
Atkļūdošanas atbalsts	5	5
Atskaišu, logu izveide un glabāšana	4	5
Dažādu testēšanas tipu atbalsts	4	4
Datubāžu lietotņu atbalsts	5	5
Data-driven testēšana	4.5	5
Rīka evolūcija ar gadiem	5	4
Koda/skriptu atkārtota izmantojamība	4	4

Atbilstība jaunākajām tendencēm	4	4
Rīka arifmētiskais vidējais vērtējums	**4.5**	**3.6**

Ar sarkanu augstāk redzamajā tabulā autors ir izcēlis kritiskākās vietas/jomas, kurās ir nepieciešami nelieli vai manāmi uzlabojumi, lai Robots automatizētas testēšanas rīks spētu konkurēt ar spēcīgākajiem maksas rīkiem tirgū.

Zemāk ir pieejams detalizēts katras kritiskās vietas/jomas paskaidrojums, kā arī autora ierosinājums par tālāko attīstību:

- Programmatūras valodu atbalsts – Robots rīks šobrīd atbalsta tikai vienu programmēšanas valodu: C#, bet, lai nākotnē konkurētu ar citiem rīkiem, tam ir jāatbalsta vismaz dažas populārākās skriptēšanas valodas, piemēram Python, Java vai Ruby.
- Nepieciešamās kodēšanas prasmes – Lai rakstītu skriptus ar Robots rīku, šobrīd ir jābūt priekšzināšanām programmēšanā, taču nākotnē ir vērts padomāt par codeless un record & playback tehnoloģiju ieviešanu, lai minimizētu priekšzināšanu nepieciešamību.
- Pieejamā dokumentācija – Šobrīd Robots dokumentācija nav pieejama plašam lietotāju klāstam, kā arī tā ir ļoti skopa un nedetalizēta. Nākotnē, lai šo rīku varētu pārdot, būs nepieciešams izveidot pilnu un labi strukturētu rīka dokumentāciju ar paskaidrojošiem attēliem, kuru pastāvīgi vajadzētu uzturēt.
- Rīka instalēšanas vienkāršums – Šobrīd Robots rīka instalēšanas process nav gluži intuitīvi saprotams gala lietotājam, tam ir jāpārvieto faili un jāveic vairākas manipulācijas, lai pareizi instalētu Robots rīku. Nākotnē izstrādātājiem vajadzētu piedomāt pie intuitīvi saprotama instalēšanas grafiskā interfeisa, kurš izdarītu visas šobrīd manuāli veicamās manipulācijas ar failiem – automātiski.
- Record and Playback funcija – Šobrīd šī funkcionalitāte ir pusautomatizēta, jeb testētājs var secīgi ierakstīt darbības, bet tās neveido pilnvērtīgu, pilnībā automatizētu skriptu, tāpēc ir nepieciešams arī rakstīt/pārveidot kodu manuāli. Nākotnē izstrādātājam vajadzētu piedomāt pie šādas funkcijas pilnīgas ieviešanas.
- Operētājsistēmu atbalsts – Šobrīd Robots atbalsta tikai Windows operētājsistēmu, taču nākotnē vajadzētu piedomāt pie tā, lai šis rīks pilnvērtīgi spētu atbalstīt vismaz divas/trīs galvenās operētājsistēmas.

- Lietotņu atbalsts – Šobrīd Robots atbalsta augstā līmenī tikai Windows darbvirsmas lietotnes, taču nākotnē vajadzētu piedomāt pie tā, lai sniegtu augsta līmeņa atbalstu arī tīmekļa un mobīlajām lietotnēm.
- Rīka tiešsaistes atbalsts – Šobrīd rīkam nav pieejas tiešsaistes vai pilnīgi jebkāds publisks atbalsts, kas pieejams ikvienam, jo rīks šobrīd vēl netiek pārdots tirgū, taču nākotnē, ja rīks tiks virzīts un pārdots tirgū, tad noteikti šāda veida atbalsts būs jāsniedz.

Autors uzskata, ka ja nākotnē visi augstāk minētie kritēriji Robots rīkā tiks ieviesti vai uzlaboti, tad šo rīku varēs uzskatīt par konkurētspējīgu alternatīvu populāriem un izplatītiem automatizētās testēšanas rīkiem, un nākotnē šo rīku būs vērts virzīt un pārdot Latvijas un Eiropas tirgū. Taču pat šobrīd, arī bez augstāk minētajiem uzlabojumiem, rīks ir konkurētspējīgs analogs, un to jau tagad varētu virzīt tirgū kā ļoti kvalitatīvu bezmaksas Windows darbvirsmas lietotņu automatizācijas rīku.

7.4. Standarta automatizētās testēšanas rīka komplektācija

Izpētot šī bakalaura darba ietvaros vairāk kā 10 dažādus rīkus un novērtējot katru no tiem vairāk kā pēc 20 dažādiem kritērijiem, darba autors ir nonācis pie šādiem secinājumiem atbildot uz jautājumu, kam ir jābūt katrā maksas/bezmaksas standarta automatizētās testēšanas rīkā.

Katrā maksas standarta automatizētās testēšanas rīkā ir jābūt, tam jāatbalsta sekojošas lietas vai jāatbilst sekojošām prasībām:

- Tam ir jāatbalsta vismaz pāris pielietojamākās un izplatītākās programmēšanas valodas.
- Tam ir jābūt iebūvētam pašu izstrādātam, bet ne obligāti, UI atpazīšanas rīkam, vēlams balstītam uz jaunākajām tendencēm un ar papildu funkcionālu.
- Rīkā ir jābūt iebūvētai Record and Playback funkcijai, kā arī būtu vēlams, lai rīks pats automātiski rakstītu skriptu šīs funkcijas laikā jeb izmantotu "codeless" tehnoloģiju, lai testētājam nevajadzētu obligāti būt arī labam skriptu rakstītājam.
- Rīkā ir jābūt vai ļoti labam iekšējai infrastruktūrai, kurā ir visa nepieciešamā informācija par iterācijām, to grafiki un citas svarīgas lietas, jeb ir obligāti jābūt iebūvētām integrācijām, vai iespējai integrēt rīku ar ALM.
- Pieejamai dokumentācijai jābūt plašai, labi strukturētai un saprotamai, vēlams, lai tajā būtu arī attēli, grafiki un citas teksta vizualizācijas.
- Ir jābūt iespējai viegli un ātri instalēt rīku.
- Testu skriptiem ir jāizpildās ātri un nekļūdīgi, ja pašā skriptā nav pieļauta kļūda, kā ir jābūt iespējai konkrēto testa skriptu vairākkārt atkārtot.
- Lietotāja saskarnei ir jābūt labam grafiskam interfeisam, tai ir jābūt intuitīvi saprotamai, vieglai lietošanā un izpratnē.
- Rīkam ir jāatbalsta vismaz pāris populārākās operētājsistēmas, piemēram Windows un Linux, kā arī lietotņu veidus.
- Rīkam ir jābūt pastāvīgam un ātram tiešsaistes atbalstam.
- Rīkā jābūt iebūvētam atkļūdošanas atbalstam, kas palīdz gan atrast kļūdu, gan piedāvā iespējamos risinājumus tās novēršanai.
- Rīkam ir jāveic automātiska soli-pa-solim atskaišu ģenerēšanu, vēlams, lai tajās būtu arī grafiki un attēli, kā arī logu veidošanu un pieglabāšanu.
- Rīkam ir jāatbalsta vairāki testēšanas tipi un metodes, kā piemēram funkcionālo, GUI, regrestestēšanu un tā tālāk.

- Rīkam ir jāmāk sadarboties un ielasīt vaicājumus no populārākajām datubāžu lietotnēm.
- Rīkā ir jābūt iestrādātai iespējai veikt data-driven testēšanu jeb rīkam ir jāmāk piekļūt, ielasīt un apstrādāt datus no dažādiem ārējiem datu avotiem, piemēram Excel.
- Rīku ir pastāvīgi jāuztur, tam ir jābūt pastāvīgiem atsevišķu laidienu un veselu stabilu versiju publikācijām ar jaunu funkcionalitāti vai izlabotu veco.
- Ir jābūt iespējai pieglabāt un atkārtoti izmantot konkrētas funkcijas un veselus skriptus.
- Rīka izstrādātājiem un uzturētājiem ir jāseko jaunākajām tendencēm un jāmēģina izstrādāt un implementēt jaunākās tehnoloģijas rīkā.

Minimālās prasības un pieejamā funkcionalitāte, kurām ir jāatbilst vai jābūt standarta bezmaksas automatizētās testēšanas rīkā:

- Rīkam ir jāatbalsta vismaz viena populāra programmēšanas valoda.
- Rīkam ir jābūt vismaz kaut kādai pieejamai dokumentācijai.
- Ir jābūt labai un saprotamai grafiskai lietotāja saskarnei.
- Rīkam ir jāatbalsta vismaz viena populāra operētājsistēma, piemēram Windows, un lietotņu tips.
- Rīkā jābūt iebūvētam atkļūdošanas atbalstam.
- Rīkam ir jāmāk automātiski/manuāli veidot pārskatāmas atskaites par testu, kā arī jāveido žurnāla faili.

REZULTĀTI

Bakalaura darba praktiskajā daļā tika izpētīti, izanalizēti un praktiski pielietoti vairāk nekā 10 dažādi automatizētās testēšanas rīki, kas atbalsta Windows darbvirsmu lietotņu testēšanu, kā arī katrs no tiem tika izvērtēts pēc vairāk nekā 20 dažādiem specifiskiem kritērijiem 5 baļļu sistēmā lai, pēc visu kritēriju kopējā arifmētiskā vidējā vērtējuma, viennozīmīgi noteiktu labāko Windows darbvirsmu automatizētās testēšanas rīku no šajā darbā pieminētajiem, un salīdzinātu to ar alternatīvu, vienas kompānijas ietvaros izstrādātu rīku. Zemāk redzamajā *tabulā 9.* autors ir apkopojis visus, šajā darbā, pieminētos rīku nosaukumus, kā arī katra rīka iegūto kopējo vidējo vērtējumu par visiem 22 kritērijiem. Bezmaksas rīki ir atzīmēti ar "*" to nosaukumu priekšā:

9. tabula. **Visu pieminēto rīku arifmētiskais vidējais vērtējums**

Rīka nosaukums	TestComplete	UFT	Katalon Studio	Squish	Tricentis Tosca	Ranorex	TestArchitect	*WinAppDriver	*SikuliX	*AutoIt	*Winium	**Robots
Rīka arifmētiskais vidējais vērtējums	4.4	4.1	4.5	4.2	4.3	4.1	4.2	1.9	3.5	1.9	2.4	3.6

Pēc šīs tabulas var secināt, ka autora sākotnēji izvirzītā hipotēze par to, ka bezmaksas automatizētās testēšanas rīki būs daudz sliktāki, mazfunkcionālāki un līdz ar to arī zemāk novērtēti nekā maksas rīki pilnībā apstiprinājās.

SECINĀJUMI

Bakalaura darba teorētiskajā daļā tika izklāstīts par automatizēto testēšanu, tās priekšrocībām un trūkumiem, kā arī par labo praksi automatizētajā testēšanā un par jaunākajām tendencēm tajā. Tika vispārēji izklāstīts arī par automatizētās testēšanas rīkiem un to kategorijām, kā arī autors sīkāk pastāstīja par Windows darbvirsmas lietotnēm, to unikālajās iezīmēs, un nodalīja fundamentālās atšķirības starp darbvirsmu un tīmekļa lietotņu automatizēto testēšanu. Līdz ar ko var secināt, ka sākotnēji izvirzītie mērķi, kas attiecināmi uz teorētisko daļu, tika pilnībā izpildīti šī darba ietvaros.

No uzskaitītajiem šajā darbā rīkiem, visaugstāko vērtējumu pēc visu kritēriju kopējā arifmētiskā vidējā vērtējuma *(sk. tabulu 9. rezultātu sadaļā)* saņēma Katalon Studio, kurš arī tiek atdzīts par labāko automatizācijas rīku šī darba ietvaros un to var uzskatīt par labāko un piemērotāko izvēli, kad runa iet par Windows darbvirsmas lietotņu automatizēto testēšanu.

Par labāko bezmaksas rīku šī darba ietvaros tika atdzīts SikuliX rīks *(sk. tabulu 9. rezultātu sadaļā)*, kas pēc vairākiem kritērijiem un aspektiem bija manāmi labāks/pārāks par citiem bezmaksas rīkiem. Mazliet augstāku vērtējumu nekā SikuliX izpelnijās tieši alternatīvais Robots rīks, kas ir ticis izstrādāts vienas kompānijas ietvaros, kurā darba autors strādā par automatizācijas inženieri, taču uz doto brīdi tas nav atvērtā pirmkoda rīks un to nevar pieskaitīt pie bezmaksas rīkiem.

Šis pats rīks tika detalizēti salīdzināts ar labāko rīku jeb Katalon Studio pēc iepriekš predefinētajiem kritērijiem. Pēc šo rīku salīdzināšanas, darba autors secināja, ka Robots rīks uz doto brīdi var sevi pozicionēt kā ļoti labu, daudzfunkcionālu bezmaksas rīku, taču nākotnē, lai pozicionētu sevi kā labu maksas rīku, un spētu konkurēt ar populārākajiem un stiprākajiem rīkiem tirgū, Robots rīkam ir jāveic daudz fundamentāli uzlabojumi:

- jāatbalsta plašāks programmēšanas valodu, OS un lietotņu tipu klāsts,
- jāizveido plaša, detalizēta, labi strukturēta un viegli pieejama dokumentācija,
- jāpiedomā pie codeless un record&playback tehnoloģiju ieviešanas,
- jāpiedomā pie tiešsaistes atbalsta izveides.

Darba autors uzskata, ka ja nākotnē visi augstāk minētie kritēriji Robots rīkā tiks ieviesti vai uzlaboti, tad šo rīku varēs uzskatīt par konkurētspējīgu alternatīvu populāriem un izplatītiem automatizētās testēšanas rīkiem, un nākotnē šo rīku būs vērts virzīt un pārdot Latvijas un Eiropas tirgū.

IZMANTOTĀ LITERATŪRA UN AVOTI

1. Automation Testing Tutorial: What is Automated Testing? [tiešsaistē] Guru99. [atsauce 15.04.2021]. Pieejams internetā: https://www.guru99.com/automation-testing.html
2. Testēšanas metodes [tiešsaistē] LU. [atsauce 15.04.2021] Pieejams internetā: http://home.lu.lv/~garnican/Testesan/test_n40.htm
3. Test Automation Benefits: 12 Reasons to Automate in 2020. [tiešsaistē] Testim. [atsauce 15.04.2021]. Pieejams internetā: https://www.testim.io/blog/test-automation-benefits/
4. Advantages and disadvantages of automates testing. [tiešsaistē] Nasscom. [atsauce 17.04.2021]. Pieejams internetā: https://community.nasscom.in/communities/it-services/advantages-and-disadvantages-of-automated-testing.html
5. Top Software Testing Trends to Watch Out For in 2020. [tiešsaistē] Katalon. [atsauce 17.04.2021]. Pieejams internetā: https://www.katalon.com/resources-center/blog/software-testing-trends-2020/
6. The Future Is Here: Top Software Testing Trends 2020. [tiešsaistē] Qamadness. [atsauce 18.04.2021]. Pieejams internetā: https://www.qamadness.com/the-future-is-here-top-software-testing-trends-2020/
7. What is Desktop Application Testing? [tiešsaistē] Katalon. [atsauce 18.04.2021]. Pieejams internetā: https://www.katalon.com/desktop-testing/
8. Testing desktop, mobile and web applications: what`s the difference? [tiešsaistē] Testlio. [atsauce 19.04.2021]. Pieejams internetā: https://testlio.com/blog/desktop-vs-web-vs-mobile-app-testing/
9. Automation of desktop application. [tiešsaistē] Future Processing. [atsauce 19.04.2021]. Pieejams internetā: https://kariera.future-processing.pl/blog/automation-of-desktop-applications/
10. Kaur, M. and Kumari, R., 2011. Comparative study of automated testing tools: Testcomplete and quicktest pro. *International Journal of Computer Applications*, *24*(1), pp.1-7. [atsauce 21.04.2021].
11. Introduction to Desktop App Testing. [tiešsaistē] Katalon. [atsauce 23.04.2021]. Pieejams internetā: https://docs.katalon.com/katalon-studio/docs/introduction-desktop-app-testing.html#windows-execution-type
12. Manuals. [tiešsaistē] Tricentis. [atsauce 23.04.2021]. Pieejams internetā: https://support.tricentis.com/community/manuals.do

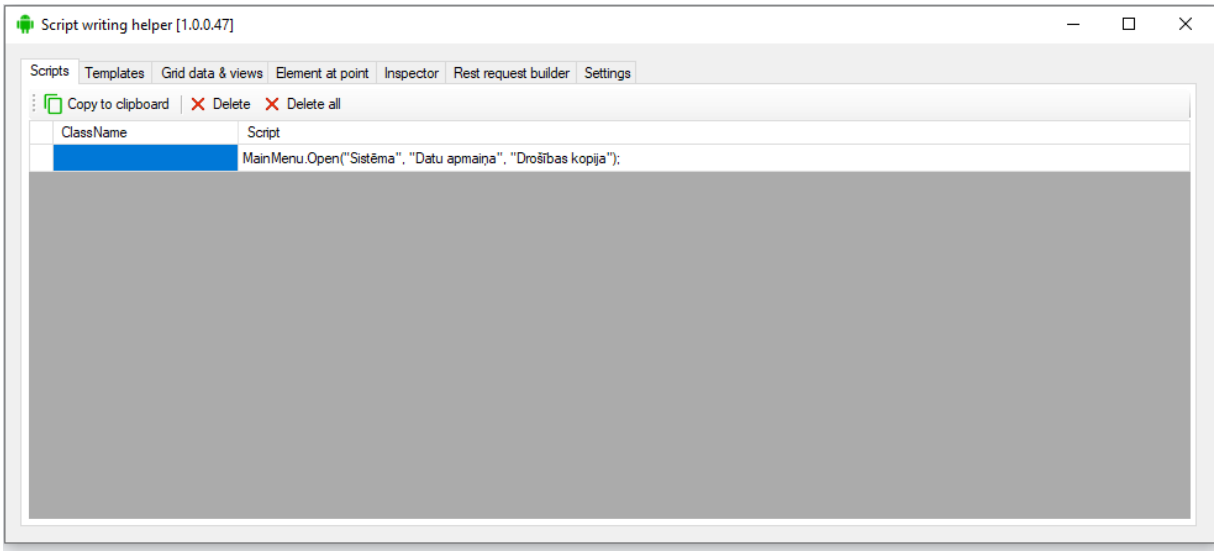
13. Testing desktop applications. [tiešsaistē] TestArchitect [atsauce 26.04.2021] Pieejams internetā: https://docs.testarchitect.com/automation-guide/application-testing/testing-desktop-applications/
14. Desktop Testing. [tiešsaistē] TestArchitect [atsauce 26.04.2021]. Pieejams internetā: https://www.testarchitect.com/product/features-and-supported-platforms/testarchitect-supported-platforms/desktop-testing
15. SikuliX by RaiMan. [tiešsaistē] SikuliX [atsauce 30.04.2021]. Pieejams internetā: http://sikulix.com/
16. Toms Freibergs. *Regresa testēšanas automatizācijas rīka izvēle darbvirsmas lietotnēm.* [tiešsaistē] [atsauce 01.05.2021]. Pieejams internetā: https://dspace.lu.lv/dspace/bitstream/handle/7/29488/302-47919-Freibergs_Toms_tf13003.pdf?sequence=1
17. Nigam, C. and Malik, S., 2018. Comparative Analysis of Automation Testing Tools. *IITM Journal of Information Technology*, p.45. [atsauce 05.05.2021]
18. Islam, N., 2016. A Comparative Study of Automated Software Testing Tools. [atsauce 08.05.2021].
19. Mohanty, H., Mohanty, J.R. and Balakrishnan, A. eds., 2017. *Trends in Software Testing*. Springer Singapore. [atsauce 12.05.2021].
20. TestComplete Documentation. [tiešsaistē] SmartBear [atsauce 15.05.2021]. Pieejams internetā: https://support.smartbear.com/testcomplete/docs/
21. Winium. [tiešsaistē] Github. [atsauce 17.05.2021]. Pieejams internetā: https://github.com/2gis/Winium
22. Ranorex Studio fundamentals. [tiešsaistē] Ranorex. [atsauce 19.05.2021]. Pieejams internetā: https://www.ranorex.com/help/latest/ranorex-studio-fundamentals/
23. WinAppDriver Documentation. [tiešsaistē] Github. [atsauce 21.05.2021]. Pieejams internetā: https://github.com/microsoft/WinAppDriver
24. Differences in Automating Web and Desktop Application Testing. [tiešsaistē] Leapwork. [atsauce 24.05.2021]. Pieejams internetā: https://www.leapwork.com/blog/differences-automating-web-and-desktop-application-testing
25. Squish Documentation. [tiešsaistē] FrogLogic. [atsauce 25.05.2021]. Pieejams internetā: https://doc.froglogic.com/squish/latest/
26. G. Moskvins. *Automatizācija.* [atsauce 25.05.2021]

PIELIKUMI

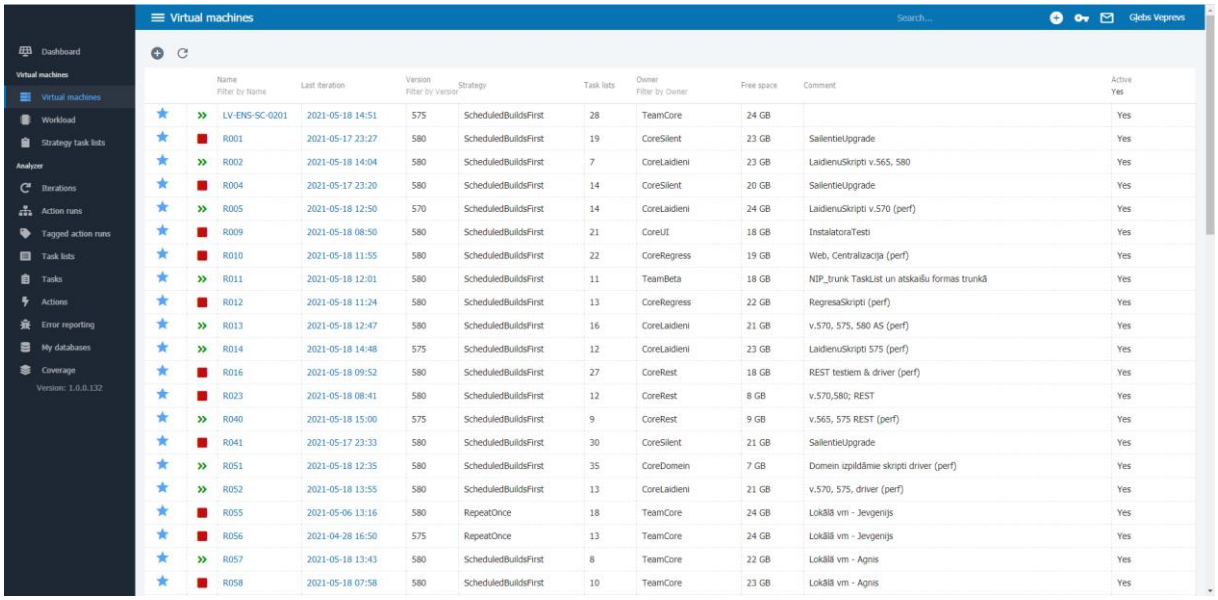
1. pielikums. Robots rīka sagataves skripts (koda piemērs)

```
[RobotCoreWinApplication.UIElementAttribute(Name="cbDelGram", Path="pgcDType->Apraksts")]
public RobotCoreWinApplication.Elements.IUICheckBox AtcelotIzpildiDzestKontejumuSagataves
{
    get
    {
        this.FindTabControl("pgcDType").OpenTab("Apraksts");
        return this.FindCheckBox("cbDelGram");
    }
}

[RobotCoreWinApplication.UIElementAttribute(Name="cmbRule", Path="pgcDType->Apraksts")]
public RobotCoreWinApplication.Elements.IUIComboBox KontrolesNosacijumi
{
    get
    {
        this.FindTabControl("pgcDType").OpenTab("Apraksts");
        return this.FindComboEdit("cmbRule");
    }
}

[RobotCoreWinApplication.UIElementAttribute(Name="cmbKSH", Path="pgcDType->Apraksts")]
public RobotCoreWinApplication.Elements.IUIComboBox KontejumuShema
{
    get
    {
        this.FindTabControl("pgcDType").OpenTab("Apraksts");
        return this.FindComboEdit("cmbKSH");
    }
}

[RobotCoreWinApplication.UIElementAttribute(Name="rbOvrCopy", Path="pgcDType->Lauku atribūti;pcLapas->Vērtība")]
public RobotCoreWinApplication.Elements.IUIRadioButton LaukuAtributi_Vertiba_Kopejot
{
    get
    {
        this.FindTabControl("pgcDType").OpenTab("Lauku atribūti");
        this.FindTabControl("pcLapas").OpenTab("Vērtība");
        return this.FindRadioButton("rbOvrCopy");
    }
}
```

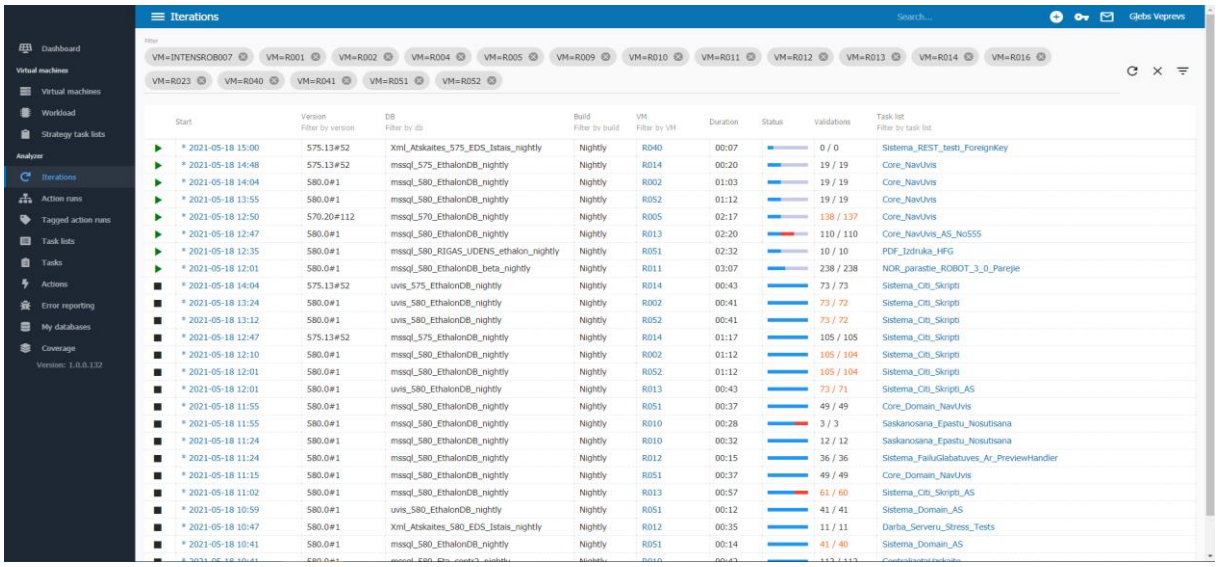
2.pielikums. Script writing helper grafiskais interfeiss Robots rīkā



3.pielikums. Robots infrastruktūra (virtuālo mašīnu saraksts)



4. pielikums. Robots infrastruktūra (pabeigto/aktīvo iterāciju saraksts)



Bakalaura darbs „Windows darbvirsmas lietotņu automatizācijas rīki” izstrādāts LU Datorikas fakultātē.

Ar savu parakstu apliecinu, ka pētījums veikts patstāvīgi, izmantoti tikai tajā norādītie informācijas avoti un iesniegtā darba elektroniskā kopija atbilst izdrukai.

Autors: Gļebs Veprevs 31.05.2021.

Rekomendēju darbu aizstāvēšanai

Vadītājs: prof., Dr.sc.comp. Guntis Arnicāns 31.05.2021.

Recenzents: profesors Dr.sc.comp. Uldis Straujums

Darbs iesniegts Datorikas fakultātē 31.05.2021.

Dekāna pilnvarotā persona: vecākā metodiķe Ārija Sproģe

Darbs aizstāvēts bakalaura gala pārbaudījuma komisijas sēdē

__.06.2021. prot. Nr. ___.

Komisijas sekretārs: asociētais profesors Dr.sc.comp. Sergejs Kozlovičs