TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond
Tarkvarateaduse instituut

Aleksandra-Salome Vellemaa
153016IAPM

ANDROID MOBIILIRAKENDUSTE AUTOMAAT-TESTIMISE TÖÖRIISTADE UURING JA VALIKUMETOODIKA VÄLJATÖÖTAMINE

Magistritöö

Juhendaja: Maili Markvardt
MSc

Tallinn 2017

Autorideklaratsioon

Kinnitan, et olen koostanud antud lõputöö iseseisvalt ning seda ei ole kellegi teise poolt varem kaitsmisele esitatud. Kõik töö koostamisel kasutatud teiste autorite tööd, olulised seisukohad, kirjandusallikatest ja mujalt pärinevad andmed on töös viidatud.

Autor: Aleksandra-Salome Vellemaa

08.05.2017

Annotatsioon

Töö eesmärgiks on anda ülevaade hetkel saadaolevatest mobiilirakenduste automatiseerimise tööriistadest Android operatsioonisüsteemile. Võrrelda neid vabalt kättesaadava taustainfo põhjal kui ka teostatud praktilise katse käigus ning luua lähtematerjal vahendi valikuks.

Töö tulemusena selgitati välja seitse laiemalt levinud tööriista, süstematiseeriti nende kohta käiv kättesaadav teave ning viidi selle alusel läbi võrdlus. Koostati näidistestid, mis võimaldaksid kontrollida tööriistade reaalset võimekust erinevates situatsioonides ning võrrelda neid sarnastel alustel. Näidistestid realiseeriti viie tööriista abil, mille tulemuste põhjal tehti järeldused nende tugevate ja nõrkade külgede kohta.

Lõputöö on kirjutatud eesti keeles ning sisaldab teksti 52 leheküljel, 6 peatükki, 12 joonist, 10 tabelit.

Abstract

Research of test automation tools for Android applications and creating selection methodology

The main goal of this work is to give an overview about some available test automation frameworks and tools for mobile applications in Android operational system. The frameworks will be compared by found background information and by empirical assessment results to generate material for selecting optimal test automation tool.

As a result, seven widely known tools were introduced and compared systematically. Sample tests were created that would help to check how capable are different tools in various situations for comparison. Sample tests were created using five different tools to deduct their strong and weak sides.

The thesis is in Estonian and contains 52 pages of text, 6 chapters, 12 figures, 10 tables.

Lühendite ja mõistete sõnastik

Massi põhine	*Crowd-based*, kaasates suurt rahvahulka
Pilve-labor	*Cloud-lab*
Katkematu integratsioon	*Continues Integration*
Ajapaus	*Timeout*
Lehe objekti muster	*Page Object Pattern*, muster automaat-testide kirjutamisel
Salvesta ja taasesita	*Capture & Replay*, meetod testi loomiseks
Integreeritud arenduskeskkond	*IDE – Integrated Development Environment*
Optiline sümbolituvastus	*OCR – optical character recognotion*
(Asukoha) teesklemine	*(Location) mocking*
Tarkvaraarenduse tööriistakast	*SDK – software development kit*
Ehe Android	*Native Android*

Sisukord

Jooniste loetelu

Tabelite loetelu

1 Sissejuhatus

1.1 Taust

Nutitelefonid hõivavad järjest suuremat osa inimeste elust ning nende kasutamise mahud kasvavad pidevalt [1], aina rohkem teenuseid ja funktsioone kolivad nutiseadmetesse. Tarbija muutub järjest nõudlikumaks ning ärid ja teenused peavad hakkama sellega kohanduma, et pakkuda kõrgel tasemel teenuseid. Tarkvaraarendajatele tähendab see uusi funktsioone ja tehnoloogiaid ning ka platvorme millele arendada ja millel töötamist toetada. Veebiarendajad peavad muretsema selle eest, et nende funktsionaalsus töötaks mitte ainult arvutite suurtel ekraanidel, vaid ka väiksematel nutiseadmetel. Lisaks arendatakse ka eraldi rakendusi ainult nutiseadmetele. Iga tarkvaraarendaja peab tagama oma toote kvaliteeti, mille tõttu otsitakse testimise hõlbustamiseks ja efektiivsemaks muutmiseks abi sageli automatiseerimise tööriistadest. Veebilehtede automatiseerimiseks on levinud teatud meetodid, mille kohta on palju infot ning valida võib selle, mis arendajale rohkem meeldib. Nutiseadmete jaoks loodud rakenduste testimine on mõnevõrra erinev, valik automatiseerimise tööriistadest on üsna suur ning millele pöörata tähelepanu esialgu ei teagi.

1.2 Probleem

Nutiseadmete arenguga on turule tulnud hulk mobiili-rakenduste automatiseerimise tööriistu. Valida enda vajaduse järgi õiget tööriista ei ole arendajal tihti triviaalne. Tööriistu on palju ning nende reklaam, või kirjeldus ametlikul veebilehel ei ole piisav otsuse langetamiseks ning võrdlemiseks. Kui otsitakse vahendit pikaajaliseks kasutamiseks, siis ei saa teha otsust kiirustades ning läbimõtlemata, kuna vahetada tööriista on äärmiselt ebamugav. Olemasolevatest toodetest korraliku turu ülevaate saamine võtab palju aega. Arvestada tuleb palju rohkemate omadustega võrreldes veebi jaoks automatiseerimisvahendit valides, kuna rakendused on eri tüüpi, nad on loodud erinevate operatsioonisüsteemide jaoks, nad kasutavad palju rohkem infot seadmest kui veebileht. Kahjuks ei leidu hetkel teadusartiklite seast ega veebikommuunidest ajakohast

tööd, mis teostaks tõesti hea ja põhjaliku ülevaate, mille põhjal saaks midagi järeldada. Leidub lühikesi järjestusi kellegi arvates parimatest tööriistades, kuid puudu on elementaarne info.

1.3 Eesmärk

Töö eesmärk on anda ülevaade hetkel saadaolevatest automatiseerimise tööriistadest nutiseadmete, eeskätt telefoni ja tahvelarvuti jaoks. Analüüsida võimalusi, mida nad pakuvad ning koostada algmaterjal vahendite tugevate ja nõrkade külgede tundmaõppimiseks, mis tooks välja ka vahendite võimalikud kitsaskohad, millele tuleks tähelepanu pöörata. Antud töö aitab arendajal mitmekordselt vähendada aega esmaseks uuringuks ning aitab leida enda tootele sobivamat tööriista. Selle jaoks autor kogub kokku võimalikult palju teoreetilist infot, mille järel tehakse esmane võrdlus ja seatakse oodatavad tulemused ka praktilisele osale. Seejärel teeb valiku erinevatest mobiilirakendustest ning praktilise katsetuse käigus loob testilood ja kirjutab testid erinevates tööriistades. Katse tulemusena antakse ülevaade sellest kui mugavad on need tööriistad, millised piirangud avalduvad neid kasutades ning kui suur on erinevate tööriistade kasutamise ajakulu erinevus.

1.4 Ülevaade tööst

Käesolev töö koosneb kuuest peatükist. Töö algab sissejuhatusega millele järgneb taust ning kirjanduse ülevaade, kus tutvustatakse lähemalt, mis ja kuidas on teadusmaailmas hetkel on kajastatud, tuuakse välja seos antud tööga. Töö jätkub teema tutvustusega, kolmandas peatükis räägitakse testimisest, selle tüüpidest ning peatutakse lähemalt testimise automatiseerimisel. Neljas ja viies peatükk kajastavad vahetult autori enda tööd milles esmalt antakse ülevaade automatiseerimistööriistadest, teostatakse praktiline katse ning kajastatakse tulemusi. Töö lõpetab kokkuvõte.

2 Taust ja kirjanduse ülevaade

Nutiseadmed on muutunud harjumuspäraseks osaks meie igapäevaelust, siiski ei ole nad turul olnud ülemäära kaua. Android platvorm tarniti esimest korda avalikkusele vähem kui kümme aastat tagasi [2]. Pärast seda saavutas see kiiresti populaarsust ning on hetkel maailma kõige kasutatavam nutiseadmete operatsioonisüsteem, hõivates aastal 2016 ca 87% turust [3]. Sellest tulenevalt lähtub antud töö eelkõige Androidi seadmetele orienteeritud toodetest. Lisaks oma populaarsusele tõmbab antud platvorm ligi nii teaduslikke uuringuid kui ka hobi-inimesi oma avatusega. Sellega on lihtne tööd teha, kuna tegu on vabavaralise tootega. Androidi turul on aja jooksul rakenduste loomine nutiseadmetele saanud uueks normiks ning uut laadi toote arendamisega tuleb kaasa ka vajadus luua lahendused nende rakenduste testimiste automatiseerimiseks.

2.1 Erinevad vahendid ja meetodid

J.Gao *et al.* oma töös [4] kirjutavad põhjalikult lahti mobiilse seadme testimise eripärad. Kuidas neid vaadelda, mille poolest see erineb teiste toodete testimisest. Antakse ülevaade testimise meetoditest ning nende tugevatest ja nõrkadest külgedest. Samuti kuulub töö juurde võrdlustabel erinevate automatiseerimisvahendite kohta, kuid kuna töö on ilmunud kolm aastat tagasi siis hetkeseisuga ei ole see tabel enam täies ulatuses relevantne, see tähendab, et on selle ajaga muutunud, täiustunud, turule on tulnud uusi võimalusi. Mobiili graafilist kasutajaliidest on võimalik testida erinevates tehnikates [5], töös kajastatakse pildivõrdlust, optilist sümbolite tuvastust ja pildivõrdlust varustades igaühe ka kirjeldusega ning plusside ja miinuste väljatoomisega.

C.Hu ja I.Neamtiu kirjutavad oma töös [6] Android operatsioonisüsteemi testimisest, kui selle avalikuks tulemisest oli möödas alles 2.5 aastat. Seal uuritakse niinimetatud ahvi meetodil testimist (*monkey-testing)* ning muuhulgaks katsetatakse ka *Monkey* tööriista, mis tuleb kaasa Androidi tarkvaraarenduse tööriistakastiga.

Erinevatest nutiseadmete operatsioonisüsteemidest on just Android platvorm väga palju leidnud kajastust teadustöödes, kuna sellel on mitmeid eeliseid: see on avatud lähtekoodiga, sellel on väga suur turg, mis kasvab iga aasta, versioonide paljususe pärast kannatab see tihti versiooni ja platvormi ühilduvusprobleemide all, mis teeb selle manuaalse testimise väga kalliks. [7] Just erinevatele Android platvormil baseeruvate rakenduste testimistööriistadele keskendub S.R. Choudhary *et al.* artikkel [7]. Selles vaadeldakse erinevaid testi sisendite genereerimise [8] tööriistasid. Oma töös nad vaatlevad põhjalikumalt kuut tööriista, mis käivitatakse ligi kuuekümnel erineval rakendusel. Peamisteks võrdlusmomentideks on leitud vigade arv, tööriista käsitlemise keerukus, erinevate versioonide ühilduvus, koodi katvus. Samuti on töös välja toodud suuremad tööriistade puudused. Artikkel on autori töö kontekstis kindlasti relevantne, kuid katab ainult osaliselt käesolevas magistritöös käsitletavat teemat. Tegu on automaatsete sisendite genereerimise tööriistaga, mis tähendab, et testija ise ei saa suures plaanis ette määrata, milliseid teste ja kuidas käivitada. Testilugude järgi testimine on välistatud. Sama põhimõtet kasutavat tööriista kajastatakse D. Amal *et al.* töös [9] täpsemalt MobiGUITAR [10] (*Mobile GUI Testing Framework*) raamistikku. Kuna tegu on nende endi poolt arendatud raamistikuga antakse ülevaade selle toimimise põhimõtetest, mis on kasulikud arendajatele, kes võtavad selle raamistiku enda testimisevahendiks. Teadmine, mis peitub abstraktsioonide taga on väga vajalik. Samuti tehakse kiire võrdlus teiste sarnaste toodetega Monkey [8] ja Dynodroid [11].

S. Gunasekaran ja V. Bargavi töös [12] on samuti keskendatud erinevate automatiseerimise tööriistade tutvustamisele. Erinevalt S.R. Choudhary *et al.* artiklist [1] käsitleb see ka tööriistasid, kus on võimalik koostada testistsenaariume ning neid ka korrata, töös käsitletakse viite erinevat tööriista, kuid seda väga pealiskaudselt. Antakse paarilauseline iseloomustus ning ekraanipilt. Töö on sobilik erinevate tööriistadega tutvumiseks, et neid hiljem edasi uurida ja katsetada.

C-H.Liu *et al.* keskenduvad oma töös [13] Android operatsioonisüsteemil käivitatavatele tööriistadele meetodiga “Salvesta ja taasesita” (*capture-replay)* [14]. Töös selgitatakse lahti kuidas see meetod töötab ning millised on levinud tööriistad, mis rakendavad seda võimalust. Kuna salvesta ja taasesita meetod võimaldav luua automaatseid teste testilugude järgi, siis on selles artiklis kirjeldatud informatsioon kasulik minu tulevases töös.

N. Saad ja N. Baker [15] püüavad samuti võrrelda erinevaid mobiili automatiseerimise tööriistu eesmärgiga valida parim. Töös koostatakse tabel, mille põhilisteks võrdlusmomentideks on OS toetus, saadavus (kas tasuta või tasuline) ning kolm kuni neli iseloomustavat tööriistavõimalust. Selle võrdlustabelile toetudes julgevad töö autorid valida parima tööriista. Töös ei selgitata, kelle jaoks on see parem või millistele vajadustele paremini sobitub. Siiski on tööd hea kasutada osaliselt erinevate tööriistade kirjelduse allikana.

2.2 Sarnased tööd

L. Kasvandi töö [16] teema on väga sarnane käesoleva töö omale. Nimelt antud töös võrdleb autor erinevaid automatiseerimisevahendeid ning valib parima. Kuigi töös on palju informatsiooni, mida ka selles töös kajastatakse on olemas põhimõttelised erinevused, mis põhjendavad käesoleva töö uudsust ning vajalikkust. L. Kasvandi töös valitakse eelkõige tööriista olemasoleva süsteemi jaoks. See tähendab, et olid teada juba kitsendused ja testitava rakenduse eripärad. Sellel põhjusel oli ka võrreldavate tööriistade hulk väiksem kui käesolevas töös. Käesoleva töö eesmärgiks aga ei ole „parima“ välja selgitamine. Eesmärgiks on erinevate tööriistadele leida parima rakendus. Mõni vahend võib sobida ideaalselt väikestele firmadele ning suurtele vastupidiselt kasu mitte tuua. L.Kasvandi praktiline katse oli pigem sümboolne ning selle eesmärgiks oli proovida, kui keerukas on üles saada keskkonda ja kirjutada esimene test, nii nimetatud „Hello world“ [17].

2.3 Probleemid mobiilirakenduste testimises

Uuring [18] püüab leida vastused, kui hästi on üldse mobiilirakendused testitud. Selleks lähenetakse probleemile kahest erinevast otsast: arendajate küsitlus ja reaalsete rakenduste analüüs. Selleks autorid valisid välja ~600 erinevat avatud lähtekoodiga rakendust ning uurisid, kas neil on koodi tasemel sisse kirjutatud teste. Valikust vaid ~14% rakendustest oli vähemalt üks testijuht kirjeldatud. Samuti tuli välja, et omakorda sellest hulgast on suur osa rakendusi, kus on kirjutatud vaid 1 või alla 5 testijuhte. Suures plaanis see tähendab, et suur osa rakendusi ei läbi regulaarselt automatiseeritud testijuhtusid.

Järgnevalt toon välja mõned küsitluse tulemused. Püstitatud küsimuseks oli, milliseid tööriistasid/tehnoloogiad kasutatakse enim. Tuli välja, et esimesel kohal on JUnit [19] raamistik, seejärel sellised tooted nagu MonkeyRunner [20] ja Robotium [21]. Põhjuseks, miks automatiseerimise tööriistasid välditakse on välja toodud väikeste rakenduste arendamine – automatiseerimine ei tasu ära, automatiseermisvahendite kasutamise piiratus, see tähendab, et paljusid funktsioone ei ole võimalik automatiseerida.

Täpsema analüüsi mobiilirakenduste testimise kirjeldatud probleemidest ning lahendustest teeb T. Samuel oma töös [22]. Nimelt analüüsitakse seal teadusartiklites enim viidatud probleemidele mobiilirakenduste testimises ning püüab selgeks teha, kas reaalsetes ärides mobiilirakenduste arendamise ja testimise vallas nad samuti esinevad. Uuringus tuleb välja, et teaduskirjanduses kirjeldatud kitsaskohad ei pea niiöelda tööstuslikes oludes reaalselt 100% paika. Ükski probleem ei olnud küll täiesti tagasi lükatud, kuid üksmeelt nende relevantsuses samuti uuringus osalejate vahel ei esinenud. Kitsaskohtade kõrval toob autor välja lahendused mida nendele probleemidele pakutakse teadusartiklites. Ka siin ei ole küsitlusele vastajad päris üksmeelel teadusmaailmaga. Osad nõndanimetatud lahendused probleemidele ei ole reaalselt kasutatavad, kuna on näiteks liiga teoreetilised, osa lahendusi oli olemas enne artikli ilmumist ja palju muud. Muuhulgas on artiklis lahenduste osas pakutud ka erinevaid automatiseerimise vahendeid ning kajastatud küsimustiku vastajate hinnang. Suurem osa vastanutest kiitis VALERA lahenduse kasulikuks. TestDroid lahenduse kohta olid nii mõnedki vastajad arvamusel, et see sobib pigem väikestele ettevõtetele, kellel ei ole mõtet omada oma seadmeteparki.

Käesolevast töös lähtub autor automaattestide kirjutamisel headest tavadest ning toetub automatiseerimise manifestis [23] väljatoodud põhimõtetele. Selleks on vaja jälgida koodis esinevaid omapärasid, mis võivad viidata, et see on kehvasti kirjutatud. Samuti tuleb jälgida, mis juhtub testide refaktooreerimisel. Võtta arvesse testide kirjutamisel erinevaid mustreid ning parimaid soovitusi.

3 Testimine

Tarkvara testimine on üks osa tarkvaraarendusese protsessist. Mõiste on lai ning tähendab paljusid tegevusi erinevates variatsioonides. Igal juhul on testimine pigem protsess kui üksik tegevus. [24] Testimine algab planeerimisest, seejärel on disainimine, testi implementeerimine, käivitamine ja valideerimine. [25]

Järgnevalt kajastan täpsemalt mõisteid, mis on vajalikud edasise töö mõistmiseks:

Erinevad lähenemised

- Musta kasti printsiip (Black box testing) – testitakse rakendust teadmata, kuidas see on üles ehitatud, näiteks ei ole ligipääsu koodile. Testija saab ainult sisendite-väljundite kaudu kontrollida programmi käitumist. [26]

- Valge kasti printsiip (White box testing) – loogilise järelduse testimine (logic-driven testing), võimaldab uurida programmi sisemist struktuuri, lihtsustatult - testija kontrollib programmi koodi. [26]

Testimise tüübid

- Regressiooni testid (Regression testing) – testid, mida käivitatakse peale programmis paranduste või uuenduste sisseviimist veendumaks, et eelnevalt arendatud ja testitud funktsionaalsus endiselt töötab. Tavaliselt käivitatakse mingit osahulka rakenduse testilugudest. [26]

- Uuriv testimine (Exploratory testing) – testimise meetod, kus ei ole valmis kirjutatud testilugusid, testi disainimine ja teostamine toimub samal ajal. Muuhulgas toetub testija kogemusele tootega ja teadmisele, kus võib tekkida veakohti. [27]

Nutiseadmel rakenduse testimisel on oluline läbi mõelda võimalikult palju. Üheks võimalikuks mudeliks, mille järgi rakenduse kasutajaliidese testimine lahti mõtestada on LONG FUN CUP [28].

Joonis 1 - LONG FUN CUP mudel (kasutatud "Mobile App Test Coverage Model : LONG FUN CUP")

Nimetus LONG FUN CUP on abreviatuur kümnest omadusest või funktsioonist, millele peaks tähelepanu pöörama mobiiliseadmel rakenduse kasutajaliidese testimisel. Käesolevas töös püüab autor automatiseerimistööriistade analüüsimisel võtta arvesse LONG FUN CUP-i loetletud omadused ning anda vastus, millseid nendest on võimalik katta automaat-testidega.

3.1 Testimise automatiseerimine

Automatiseerimistööriistad suudavad käivitada teste, koostada raporteid, võrrelda testitulemusi, teste saab käivitada mitu korda järjest igal kellaajal. [29] Kõige elementaarsem osa automaattestidest on koodi kontrollimiseks kirjutatud testid, kasutajaliidesega rakendusele on võimalik kirjutada automaatsed kasutajaliidese testid. Automaattestidel on mitmed eelised ja/või eesmärgid, miks neid kasutada. Näiteks suurendada testikaetavust, säästa aega, parendada kvaliteeti. Peamine põhjus aga on nende korratavus. Automaattesti kirjutamine tasub suuresti ära nende testilugude puhul, mida tuleb tihti ja regulaarselt korrata, näiteks regressioontestid [30]. Regressioontestide

hulk kasvab ajas rakenduse kasvamisega, seega iga kord kulub aina rohkem ja rohkem aega testide käivitamisele.

Otsust, milliseid teste automatiseerida tuleb langetada kaalutletult. Igat testi ei ole mõistlik automatiseerida. Valik sõltub suuresti eesmärgist ja ka eelarvest. Ühe võimaliku lahenduse pakuvad välja R.Ramler ja K.Wolfmaier oma töös [31], kus automatiseeritud testide hulk arvestatakse välja sõltuvalt eelarvest.

Nutiseadmel töötava rakenduse automatiseerimine, selle eesmärk ja meetodid erinevad mõnevõrra veebilehe automatiseerimisest. Kui arendatakse arvutist veebibrauseriga avatav rakendus, siis tavaliselt võivad tekkida erinevused sisu vaatamisega erinevates veebibrauserites, mõnikord ka erinevate veebilehitsejate versioonis. Nutiseadmele rakenduse loomisega on see alla laetav kümnetesse eri seadmetesse, mis kõik võivad käituda erinevalt. Samuti on nutiseadmetel rohkem omadusi, mida rakendus jälgib.

Mobiilseadmete testimise lähenemisi on mitmeid, igal omad plussid ja miinused. Enamasti ei ole ühte parimat lahendust, pigem on võimalik parim lahendus leida lähtudes ettevõtte ja toote omadustest [4].

- Emulaatori põhine – rakendusi testitakse arvutis mobiiliseadme emulaatoril. Suhteliselt soodne lahendus, kuna ei pea investeerima reaalsetesse seadmetesse. Omab mitmeid piiranguid, kuna paljusid asju ei ole võimalik emulaatoril järgi teha: spetsiifilised liigutused, seadme-spetsiifilised funktsioonid.

- Seadme põhine – tähendab endale seadme-pargi loomist. On kulukam, kui emulaatori kasutamine, kuid katab ära paljud emulaatori augud. Probleemiks on pidev seadmete ja operatsioonisüsteemide areng, mis nõuab seadmete uuendamist.

- Pilves testimine – kasutatakse pilve seadmete parki teenusena. See tähendab, et ettevõte ei pea ise seadmeid soetama.

- Massi põhine – kaasatakse vabakutselisi testijaid, kogukonda või lõppkasutajaid toote testimises.

4 Tööriistade võrdlus

Järgnevas peatükis koostatakse valim turul olevatest mobiilirakenduste automatiseerimise tööriistadest, kirjeldatakse nende põhilisi omadusi, mille põhjal tehakse teoreetiline võrdlus. Teoreetilise võrdluse toeks teostab autor ka praktilise katse.

4.1 Valiku alused

Tööriistade valikul toetus autor nii teadusartiklitele kui ka veebikommuunide soovitustele ja võrdlustele. Viimased olid tööriistade leidmiseks optimaalsemad, kuna uus info levib erialakommuunides kiiremini kui teadusmaailmas.

Esimeseks filtriks oli Android operatsioonisüsteemi toetus, kuna antud töö keskendub Android platvormile saadaolevatest tööriistadest. Suur boonus on siiski mitme platvormi toetus tööriistal. Oluline on, et tööriist oleks veebis „olemas“, selle all mõtleb autor, et tööriista kohta oleks olemas info, juhendid, arvustused, õpetused, kommuunid. Selle kriteeriumi põhiliseks argumendiks on antud töö orienteeritus tööstusele, see tähendab et arendajale, kes hakkab tulevikus looma testi-skripte on oluline, et probleemi korral oleks võimalik otsida ka abi.

4.2 Tööriistade võrdlus allikate põhjal

Järgnevalt esitatakse valitud tööriistade tutvustuse ning ülevaate. Ülevaatlikuma esitamise eesmärgil kirjeldatakse tööriistad sarnase struktuuri järgi, milles esmalt tutvustatakse lühidalt toodet ning seejärel esitatakse võrdlev tabel.

Calabash

Tööriist vastuvõtutestide automatiseerimiseks, toetab rist-platvorme, Androidi ja iOS-i. Toode on vabavaraline ja tasuta, seda arendab ettevõte Xamarin, kes muuhulgas arendab ka tasulist pilve-labori teenust [32]. Calabash on pilve-labori toetatud tehnoloogia. Struktureeritud tööriista kirjeldus esitatakse järgnevas Tabel 1-s.

Tabel 1 Calabash tööriista kirjeldus

OP-süsteemide toetus	Android, iOS

Rakenduse tüübi toetus	Ehedad ja hübriidrakendused
Tasuta/tasuline	Vabavaraline
Toetatud tehnoloogiad	Ruby, Cucumberi toetus
Salvesta ja taasesita	Ei
Objektide määratlemine	Elemendipõhine
Lähtekoodile ligipääs	Ei
Millel käivitada?	Seadmed, emulaatorid
Mitme seadme toetus	Ei*
Katkematu integratsioon	Jah

*Mitme seadme toetus – korraga saab testi käivitada ainult ühel seadmel, lahendusena võib kasutada pilve-laboreid.

Ranorex

Ranorex on tasuline toode mis hõlmab endas tervet keskkonda testide käivitamiseks ja loomiseks [33]. Muuhulgas on toode pärjatud mitmete auhindadega tarkvara kvaliteedi maailmas. Muuhulgas saab testiseadme ühendada arvutiga nii kaabli kui ka wifi kaudu, elemente pigem aeglaselt [32], see tuleb välja ka teostatus katses, mille kirjelduse leiab töö järgnevates osades. Struktureeritud tööriista kirjeldus esitatakse järgnevas Tabel 2-s.

Tabel 2 Ranorex tööriista kirjeldus

OP-süsteemide toetus	Android alates 2.2, iOS, Windows
Rakenduse tüübi toetus	Ehedad, hübriid-, veebirakendused ning töölaua rakendused.
Tasuta/tasuline	Tasuline
Toetatud tehnoloogiad	Windows, SAP ERP, Qt, WPF UI, .NET ja Java
Salvesta ja taasesita	Jah
Objektide määratlemine	Elemendipõhine
Lähtekoodile ligipääs	Ei
Millel käivitada?	Seadmed, emulaatorid

Mitme seadme toetus	Jah
Katkematu integratsioon	Jah

Robotium

Robotium on vabavaraline toode, mis oli algselt mõeldud valge karbi printsiibil testimise jaoks, mis tähendab, et lähtekood oli vajalik [35]. Hiljuti tööriist täienes, lisandus võimalus teha ekraanipilte, testida hübriid-rakendusi ning testimist reaalsetel seadmetel. Lisaks tuldi välja tööriista tasulise täiendusega, mis võimaldab teste luua salvesta ja taasesita meetodil ning ilma lähtekoodita. Sellegipoolest on vabavaraline versioon orienteeritud pigem lähtekoodi kasutamisega testidele. Struktureeritud tööriista kirjeldus esitatakse järgnevas Tabel 3-s.

Tabel 3 Robotium tööriista kirjeldus

OP-süsteemide toetus	Android alates 1.6
Rakenduse tüübi toetus	Ehedad ja hübriidrakendused
Tasuta/tasuline	Vabavaraline
Toetatud tehnoloogiad	Java
Salvesta ja taasesita	Jah, tasulises versioonis*
Objektide määratlemine	Elemendipõhine
Lähtekoodile ligipääs	Jah / (Ei*)
Millel käivitada?	Seadmed, emulaatorid
Mitme seadme toetus	Ei
Katkematu integratsioon	jah

*Salvesta ja taasesita – kuigi Robotium on vabavaraline toode, lõid nad oma salvestaja, mis on kommertstoode. Sellega tuli ka kaasa järgmine uuendus, et lähtekoodile ligipääs ei olnud enam vajalik. Siiski jääb Robotiumi vabavaraline versioon orienteerituks arendajatele ning lähtekoodiga integreerimisele.

Espresso

Espresso tööriist on sarnaselt Robotiumile orienteeritud arendajatele osana arendustsüklist ning on mõeldud valge karbi printsiibil testimiseks [36]. Tegemist on Androidi tootja poolt arendatud tööriistaga spetsiifiliselt Android platvormi tarvis. Struktureeritud tööriista kirjeldus esitatakse järgnevas Tabel 4-s.

Tabel 4 Espresso tööriista kirjeldus

OP-süsteemide toetus	Android alates 2.2
Rakenduse tüübi toetus	Ehedad rakendused
Tasuta/tasuline	Vabavaraline
Toetatud tehnoloogiad	Java
Salvesta ja taasesita	Jah
Objektide määratlemine	Elemendipõhine
Lähtekoodile ligipääs	Jah / (Ei*)
Millel käivitada?	Seadmed, emulaatorid
Mitme seadme toetus	Jah**
Katkematu integratsioon	Jah

*Lähtekoodile ligipääs – kuigi on toetatud musta karbi printsiibil testimine, tööriist on loodud töötama lähtekoodiga.

**Mitme seadme toetus – paralleelselt mitmel seadmel on võimalik jooksutada mõnes pilveserveris. Arvutil selle tööle saamine pigem keerukas.

Ui Automator

Sarnaselt Espressole on antud tööriist loodud Androidi tarkvara arendajate poolt [37]. Erinevus seisneb tööriista suunitlusest, mis antud juhul on mõeldud rakenduse kasutajaliidese efektiivseks testimiseks, eriti kui kasutuslugudes vaheldub testitava rakenduse tegevus teiste rakenduste ja süsteemi kasutajaliidesega. Võimaldab ligipääsu mobiiliseadme nuppudele nagu „Kodu“, „Tagasi“, helinupud. Struktureeritud tööriista kirjeldus esitatakse järgnevas Tabel 5-s.

Tabel 5 UI Automator tööriista kirjeldus

OP-süsteemide toetus	Android
Rakenduse tüübi toetus	Ehedad rakendused
Tasuta/tasuline	Vabavaraline
Toetatud tehnoloogiad	Java
Salvesta ja taasesita	Ei
Objektide määratlemine	Elemendipõhine
Lähtekoodile ligipääs	Ei
Millel käivitada?	Seadmed ja emulaatorid
Mitme seadme toetus	Jah
Katkematu integratsioon	Jah

Appium

Appium on vabavaraline automatiseerimisraamistik, mis toetab levinumaid mobiili operatsioonisüsteeme ning tüüpe. Arendamisel saab valida mitmete tehnoloogiate vahel [38]. Struktureeritud tööriista kirjeldus esitatakse järgnevas Tabel 6-s.

Tabel 6 Appium tööriista kirjeldus

OP-süsteemide toetus	Android alates 2.3, iOS , Windows
Rakenduse tüübi toetus	Ehedad, hübriid- ja veebirakendused
Tasuta/tasuline	Vabavaraline
Toetatud tehnoloogiad	Java, C#, Ruby, Python, PHP, JS
Salvesta ja taasesita	Ei
Objektide määratlemine	Elemendipõhine
Lähtekoodile ligipääs	Ei
Millel käivitada?	Seadmed, emulaatorid
Mitme seadme toetus	Jah

Katkematu integratsioon	Jah

SeeTest

Tegu on tasulise tarkvaraga mis pakub mitmeid mobiiliseadmete testimisega seotuid tooteid, mille hulgas on pilve-testimine, mobiilitestimise automatiseerimine ning ka jõudluse testimise tööriistad [39]. Antud töös vaadeldakse automaat-testimise tööriista SeeTestAutomation. Antud tööriistaga loodud teste on võimalik käivitada kõikidel operatsioonisüsteemidel, ei ole vaja kirjutada erinevaid teste. Võimaldab ligipääsu GPS-le, kaamerale, audiole, samuti toetab kõiki liigutusi (mitmik-puude, viibe, lohista ja vabasta, sisse suurendamine, kerimine). Struktureeritud tööriista kirjeldus esitatakse järgnevas Tabel 7-s.

Tabel 7 SeeTest tööriista kirjeldus

OP-süsteemide toetus	Android, iOS, Windows, Blackberry
Rakenduse tüübi toetus	Ehedad, hübriid- ja veebirakendused
Tasuta/tasuline	Tasuline
Toetatud tehnoloogiad	WebDriver (Selenium), HP UFT (QTP), Microsoft VisualStudio, IBM RFT, Appium, C#, Java, Perl, Python, Ruby
Salvesta ja taasesita	Jah
Objektide määratlemine	Elemendipõhine, pildipõhine, visuaalne tekstituvastus
Lähtekoodile ligipääs	Ei
Millel käivitada?	Seadmed, emulaatorid
Mitme seadme toetus	Jah
Katkematu integratsioon	Jah

4.3 Kokkuvõte

Alloleval tabelil on kokku toodud kõik tööriistad nende parema võrdluse tagamiseks. Tärniga tekstid on lahti seletatud iga tööriista kirjelduse juures.

Tabel 8 on näha allikate põhjal saadud hinnangute kokkuvõtvat infot. Vahendi valikul tuleks eelkõige läbi mõelda ja tähelepanu pöörata järgmistele asjaoludele:

- Operatisoonisüsteemi toetus – kas on vaja mitme platvormi toetust, või piisab ainult ühest, samuti on oluline jälgida, mis versioonist alates platvormi toetatakse. Tabel 8-st on näha, et tööriistad, mis toetavad mitmeid platvorme kõige sagedamini toetavad lisaks Androidile iOS-i, Windowsi toetavad ka mõned tööriistad, ainult SeeTest toetab ka BlackBerryt. Siiski tuleb arvestada, et Android ja iOS kokku hõlmavad 99,3% nutitelefonide turust [3].

- Rakenduse tüübi toetus – kas on arendatud ehedat, hübriid või veebielementidega rakendust. Seadme valikul tuleb kindlaks teha, kas vastavat tehnoloogiat toetatakse. Ainult ehedaid rakendusi toetavad Androidi arendajate enda tooted. Ülejäänud tööriistad on paindlikumad ning toetavad ka teisi tehnoloogiaid.

- Toetatud keeled – kriteerium jaguneb suures osas kaheks: millises keeles on võimalik teste kirjutada ning kas on olemas „Salvesta ja taasesita“ testide loomise formaat, viimase puhul on oluline jälgida, mis meetodiga süsteem salvestab, ning mis lisafunktsioone see pakub (nt valideerimise lisamine). Robotiumis, UI Automatoris ja Espressos programmeerimiskeele valikut ei ole. Kirjutada saab ainult Javas. Calabash-is saab kirjutada ainult Ruby-s. Teistel tööriistadel on võimalik mingil määral valida kasutatavate tehnoloogiate vahel

- Objektide määratlemisel on juba enamus tööriistasid jõudnud elemendipõhise määratlemiseni, mis on väga hea. SeeTest pakub lisaks pildituvastust ja optilist sümboli tuvastust.

- Pakutavad erifunktsionaalsused – mõned lisafunktsioonid on sellised, mida ei saa ühisesse tabelisse panna, kuid neile on oluline tähelepanu pöörata lähtudes testitava rakenduse spetsiifikast. SeeTest pakub võimalust käivitada samu teste nii Android kui iOS peal. UI Automator omab ligipääsu seadme enda nuppudele ning mõnes teises tööriistas on paremini defineeritud liigutused.

Alloleval tabelil on kokku toodud kõik tööriistad nende parema võrdluse tagamiseks. Tärniga tekstid on lahti seletatud iga tööriista kirjelduse juures.

Tabel 8 Erinevate testi-tööriistade koond-võrdlustabel

	Calabash	Ranorex	Robotium	UIAutomator	Appium	SeeTest	Espresso
OP-süsteemide toetus	Android, iOS	Android alates 2.2, iOS, Windows	Android alates 1.6	Android	Android alates 2.3, iOS , Windows	Android, iOS, Windows, Blackberry	Android alates 2.2
Rakenduse tüübi toetus	Ehedad ja hübriidrakendused	Ehedad, hübriid-, veebirakendused ning töölaua rakendused.	Ehedad ja hübriidrakendused	Ehedad rakendused	Ehedad, hübriid- ja veebirakendused	Ehedad, hübriid- ja veebirakendused	Ehedad rakendused
Tasuta/tasuline	Vaba-varaline	Tasuline	Vaba-varaline	Vaba-varaline	Vaba-varaline	Tasuline	Vaba-varaline
Toetatud tehnoloogiad	Ruby, Cucumberi toetus	Windows, SAP ERP, Qt, WPF UI, .NET ja Java	Java	Java	Java, C#, Ruby, Python, PHP, JS	WebDriver (Selenium), HP UFT (QTP), Microsoft VisualStudio, IBM RFT, Appium, C#, Java, Perl, Python, Ruby	Java
Salvesta ja taasesita	Ei	Jah	Jah, tasulises versioonis*	Ei	Ei	Jah	Jah

Objektide määratlemine	Elemendi-põhine	Elemendi-põhine	Elemendi-põhine	Elemendi-põhine	Elemendi-põhine	Elemendi-põhine, pildipõhine, visuaalne tekstituvastus	Elemendi-põhine
Lähtekoodile ligipääs	Ei	Ei	Jah / (Ei*)	Ei	Ei	Ei	Jah / (Ei*)
Millel käivitada?	Seadmed, emulaatorid	Seadmed, emulaatorid	Seadmed, emulaatorid	Seadmed ja emulaatorid	Seadmed, emulaatorid	Seadmed, emulaatorid	Seadmed, emulaatorid
Mitme seadme	Ei*	Jah	Ei	Jah	Jah	Jah	Jah**
Katkematu integratsioo	Jah	Jah	jah	Jah	Jah	Jah	Jah

4.4 Automatiseerimisvahendite sobivus LONG FUN CUP mudeliga

Peatükis number 3 on toodud mudel testimise planeerimise hõlbustamiseks. Järgnevalt esitatakse iga mudeli alamosa ning vaadeldakse seda automaat-testimise kontekstis. Elemente, mida on raske testida, on mitmeid. Antud olukorras tuleb läbi mõelda, mida soovitakse automaat-testidega katta ning kas need raskesti automatiseeritavad funktsioonid on vaja tingimata realiseerida.

Asukoht (*Location*)

Android platvormi olemus annab võimaluse asukoha teesklemiseks igas tööriistas. Siiski ei ole see funktsionaalsus kõikidel tööriistadel pakutud raamistiku osana. Näiteks SeeTest dokumentatsioonis on kirjas, et neil on loodud funktsioonid asukoha teesklemiseks, kuid teistes tööriistades peab selle ise valmis programmeerima.

Orientatsioon (*Orientation*)

Seadme ekraani orientatsiooni on võimalik sätestada käsuga kõigis võrreldavates automatiseerimise tööriistades.

Võrk (*Network*)

Calabashis ja RanoRexis ei saa vahetada wifi ja mobiilse andmeside kasutamise vahel. Robotiumis, appiumis, UI Automatoris saab lülitada wifit sisse ja välja, SeeTesti kohta infot ei leidnud.

Viiped (*Gestures*)

Kõik vahendid lubavad toetada mingil määral erinevad liigutusi. SeeTest lubab toetada kõiki liigutusi. Reaalselt tuleb eraldi katsetada, kas vahendite poolt toetatavad liigutused ka reaalselt toetavad, sest autori loodud katsetes olid liigutused tööriistade nõrk koht. Vahendi valikul tuleks üle vaadata rakenduses kasutatavad spetsiifilised viiped, liigutused ja nende toetust tööriistal kontrollida. Mõnes rakenduses ei kasutata kunagi mõndasid liigutusi, seega nende mittetoetamine tööriista poolt ei ole dramaatiline.

Funktsioonid (*Function*)

Rakendusesiseselt kõikvõimalike sisendite ja puudutuste katsetamine. Iga automatiseerimistööriist peaks sellega hakkama saama, kuna need funktsioonid on vundament.

Uuendused (*Updates*)

Automatiseerimine on ideaalne võimalus jooksutada teste erinevatel operatsioonisüsteemi versioonidel. Siiski tuleb olla tähelepanelik, mis versioonist alates toetab automatiseerimisvahend operatsioonisüsteemi ning see tuleb kindlasti kõrvutada ettevõtte vajadustega.

Kasutuslood (*User scenarios*)

Kasutuslood on kirjutatavad üsnagi autonoomselt automatiseerimisvahendi valikust.

Märguanded (*Notification*)

Mõndade teavituste testimistega võib tekkida probleeme. Näiteks need, mis ilmuvad rakenduse vaate väliselt. Kui mitmed kasutuslood hõlmavad teiste rakenduste ja süsteemi

kasutajaliidesesse sekkumist, siis tuleks kaaluda tööriistu, mis seda võimaldavad. Vaatluse all olevatest tööriistades on see funktsionaalus olemas UI Automatoril. Heli ja vibratsiooniga edastatavaid märguandeid on raske testida, SeeTest lubab ka heli kasutamist.

Suhtlus (Communication)

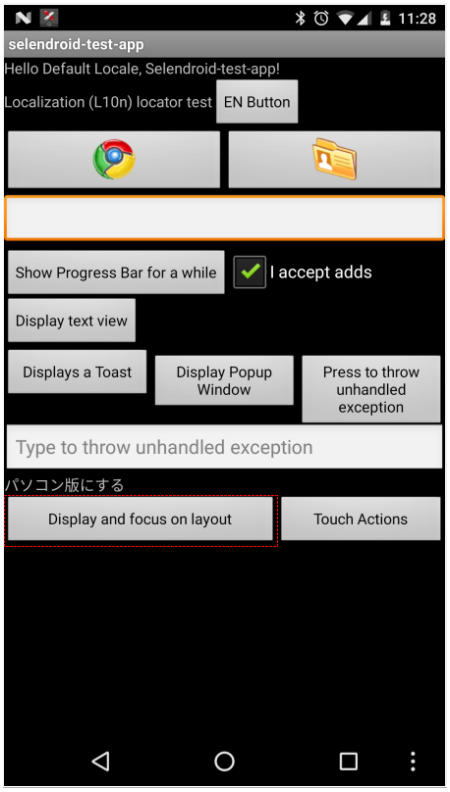
Selle jaotise eesmärgiks on tuvastada ja testida, kuidas reageerib rakendus kõne sissetulekule ning kuidas ta suudab oma tööd jätkata. Sellise funktsionaalsuse automatiseerimine on küllalt tülikas ning tuleks valida tööriist, millel oleks ligipääs ka seadme kasutajaliidesele ning muudele rakendustele. Juhul kui antud tüüpi funktsionaalsus ei ole rakenduse põhifunktsionaalsus tuleb kaaluda selle automatiseerimise mõistlikkust.

Platvorm (*Platform*)

Platvormist, millel rakendus jookseb, oleneb palju. Juhul, kui rakendus on saadaval nii iOS kasutajatele, kui ka Android kasutajatele on mõistlik kaaluda sellist tööriista, milles kirjutatud testid saaks käivitada mõlemal platvormil. Ülaltoodud vahenditest on selleks SeeTest. Muu võimalusena jääb kas automatiseerida ainult ühele platvormile või kirjutada kaks komplekti teste.

5 Praktiline võrdlus

Praktilise katsetuse jaoks valis autor selendroid-test-app rakenduse. Tegu on Selendroid tööriista arendusmeeskonna rakendusega, millega nad ise valideerivad oma tööriista. Valiku kriteeriumiteks oli võimalikult palju erinevaid objekte (nupp, märkeruut, tekstiala jne) ja liigutusi võimaldavat rakendust. Ekraanipilti rakenduse avaekraanilt on kujutatud alloleval pildil Joonis 2.



Joonis 2 selendroid-test-app rakenduse ekraanipilt

Erinevate vahendite omavaheliseks võrdluseks otsustati luua seitse erinevat testilugu. Testide kirjutamisel lähtuti eesmärgist katsetada võimalikult erinevaid sisendeid samas jäädes lihtsaks ja arusaadavaks. Testides katsetatakse kõige elementaarsemaid liigutusi, millele nutiseade peaks reageerima ja püütakse kasutada enimkasutatavaid ekraaniobjekte. Seadme asukohta, güroskoopi jm. ei testitud. Järgnevalt esitatakse seitse testilugu ning nende valiku ja vajalikkuse põhjendus. Kaldkirjas olev nimetus on erinevates tööriistades testi nimetus. Enne iga testi käivitamist, käivitatakse skript, mis alustab telefoni avaekraanist, avab rakenduse, veendub, et avanes rakenduse avaleht.

Testilood:

TS1: *CheckElementsOnHomeScreenTest*

Antud testi eesmärgiks on sooritada esimene lihtne katse uue tööriistaga. Testi eesmärgiks on tuvastada, kas kõik vajalikud objektid on rakenduse avaekraanil olemas.

- Valideerida nupu „EN Button“ olemaolu
- Valideerida nupu Google Chrome veebibrauseri märgiga olemasolu
- Valideerida nupu kausta märgiga olemasolu
- Valideerida sisestusvälja nr. 1 olemasolu
- Valideerida nupu „Show Progress Bar for a while“ olemasolu
- Valideerida märkeruudu „I accept adds“ olemasolu
- Valideerida „Display text view“ nupu olemasolu
- Valideerida nupu „Displays a Toast“ olemasolu
- Valideerida nupu „Display Popup Window“ olemasolu
- Valideerida sisestusvälja nr. 2 olemasolu
- Valideerida nupu „Display and focus on layout“ olemasolu
- Valideerida nupu „Touch actions“ olemasolu

TS2: *EnButtonTest*

Esmalt kontrollitakse, kas tööriist suudab tuvastada nuppu ja seejärel sellele vajutada. Selle järel avaneb dialoogi aken. Kontrollitakse, kas tööriist suudab tuvastada selle avanemist ja sellel omakorda elementide tuvastust ja vajutamist.

- Kontrollida teksti „This will end the activity“ olemasolu
- Kontrollida nuppude „I Agree“ ja „No, no“ olemasolu

- Vajutada nupule „No, no“

TS3: *PopupTest*

Esimene samm kontrollib nupu vajutamist, seejärel avaneb hüpikaken (*popup*). Kontrollitakse, kas tööriist suudab tuvastada hüpikakna ja sellel asuvad elemendid.

- Kontrollida nuppu „Display Popup Window“ olemasolu
- Vajutada nuppu „Display Popup Window“
- Kontrollida teksti „It’s a popup window“ olemasolu
- Kontrollida nupu „Dismiss“ olemasolu
- Vajutada nuppu „Dismiss“

TS4: *UnhandledExceptionTest*

Nupuvajutus tekitab seadmes kontrollimata vea (*unhandled exception)*. Kontrollitakse, kas tööriist suudab näha selle tekitatud veateadet ja vajutada nupule „Taaskäivita rakendus“, tegu ei ole rakenduse enda vaid teise protsessi veateatega.

- Veenduda nupu „Press to throw unhandled exception“ olemasolus ja klikitavuses
- Klikkida nupul „Press to throw unhandled exception“
- Tuvastada veateate hüpikaken
- Vajutada „Taaskäivita rakendus“

TS5: *GesturesTest*

Nupu „Touch actions“ vajutamisel avaneb teatud moodi „tööpind“, mis suudab tuvastada erinevaid vajutusi ja liigutusi kuvades need ekraanile.

- Sooritada ekraanil üks vajutus (*tap)*
- Veenduda, et rakendus kuvab ekraanil „SINGLE TAP CONFIRMED“
- Sooritada ekraanil kaks vajutust järjest *(Doube tap)*

- Veenduda, et rakendus kuvab ekraanil „ON DOUBLE TAP EVENT“
- Sooritada ekraanil pikk vajutus *(long press)*
- Veenduda, et rakendus kuvab ekraanil „LONG PRESS“

TS6: *RegisterUserTest*

See test keskendub ehe androidi (native-android) ekraanivormi tuvastuse, täitmise ja sellel navigeerimisele. Kasutatakse progressiriba (kontrollitakse selle järel ootamist), tekstivälju, märkeruutu, rippmenüüd, ainuvalikut.

  - Veenduda nupu „Show Progress Bar for a while“ olemasolus
  - Vajutada nuppu „Show Progress Bar for a while“
  - Kontrollida „Waiting dialog“ teksti olemasolu
  - Oodata, kuni progressiriba laeb ära
  - Avanenud vormil kontrollida kas kõik objektid on olemas
    - Kasutajanime väli
    - Emaili väli
    - Salasõna väli
    - Nimi
    - Programmeerimiskeel
    - Kinnituse märkeruut
    - Registreerimisnupp
  - Täita kõik väljad
  - Avatud vaates kontrollida, kas väljad on samad, mis täitmisel
  - Vajutada „Registreeri kasutaja“

TS7: *WebViewTest*

Veebilehitseja nupu taga on veebielementidega funktsionaalsus. Testi eesmärk on kontrollida, kas tööriist suudab töötada ka veebielementide peal.

- Kontrollida Chrome veebilehtiseja nupu olemasolu
- Vajutada Chrome veebilehitseja nupule
- Veenduda, et on olemas õiged objektid lehel
  - Nupp „Go to home screen“
  - Rippmenüü „Say hello demo“
- Valida rippmenüüst „form-page“
- Valida rippmenüüst „Say hello demo“
- Veenduda, et vormil on kõik elemendid olemas
- Täita ära kõik vormi elemendid
- Vajutada nuppu „Send me your name“
- Veenduda, et vorm väljastab õiged andmed
- Vajutada lingile „here“
- Vajutada nupule „Do to home screen“

5.1 Tulemused

Praktilisse võrdlusesse kaasasin viis allpool loetletud tööriista. Tööriistade valiku kriteeriumiks osutus orienteeritus pigem testijatele kui arendajatele, see tähendab, et teste sai kirjutada ilma rakenduse lähtekoodita. See järgib ka põhimõtet, et testitakse seda versiooni, mis hiljem laetakse üles veebipoodi.

- UI Automator – testid kirjutati programmeerimiskeeles Java.

- Ranorex Studio – testid loodi kasutades toote laiendatud funktsionaalsust sündmuste loomiseks ning valideerimiseks.
- Appium – testid kirjutati Java programmeerimiskeeles, kasutati Selenium Webdriverit.
- SeeTest – katses kasutab autor „salvesta ja taasesita“ meetodit, et hinnata selle paindlikkust, kuna see on üks tööriista müügiargumentidest.
- Calabash – testid kirjutatud Cucumberis ja Rubys

5.1.1 Testide realiseerimine

Järgnevalt esitatakse võrdlus testide õnnestumise kohta. Vasakpoolses veerus esitatakse testilood, mida püüti realiseerida ning parempoolsetes veergudes on näidatud testi läbimise/kirjutamise tulemused iga katsetatud tööriista kohta. Tulemusi tuleks lugeda järgmiselt: OK – Korras, test õnnestus, NOK (*not OK)* – mitte korras, test ebaõnnestus, OK/NOK – osa testist õnnestus, osa mitte.

Tabel 9 Tööriistade UI Automator ja Ranorex Studio tulemused testilugude TS1-TS7 koostamisel

	UI Automator	**Ranorex Studio**	**Appium**	**SeeTest (recorded)**	**Calabash**
TS1: *CheckElementsOnHomeScreen Test*	OK	OK	OK	OK	OK
TS2: *EnButtonTest*	OK	OK	OK	OK	OK
TS3: *PopupTest*	NOK	OK	NOK	NOK	OK
TS4: *UnhandledExceptionTest*	OK	NOK	OK	OK	NOK
TS5: *GesturesTest*	OK/	OK/	OK/	OK/	OK/

	NOK	NOK	NOK	NOK	NOK
TS6: *RegisterUserTest*	OK	OK	OK	OK	OK
TS7: *WebViewTest*	OK	OK	OK	OK	-

Alloleval tabelil on kokku toodud kõik tööriistad nende parema võrdluse tagamiseks. Tärniga tekstid on lahti seletatud iga tööriista kirjelduse juures.

Tabel 8-st selgub, et probleemsed kohad on üldiselt sarnased. Järgnevale esitatakse teostatud katse analüüs ja selgitatakse lahti probleeme tekitanud testilood.

- TS3: PopupTest – kolm tööriista ei saanud hakkama selle testiga ning kõigil oli põhjus sama. Tööriistad mingil põhjusel ei tuvasta ära hüpikakent. See tähendab, et nad ei oska sellel olevaid objekte tuvastada ega vajutada. Esimesena tuli see välja UI Automatoriga. Selle kohta on registreeritud ka viga internetis [33]. Samuti esines see mure ka Appiumis ning tasulises tööriistas SeeTest. Leevendavaks asjaoluks võib arvata, et rakendustes eelistatakse dialoogi aknaid hüpikakendele. Siiski, tööriista valikul tuleks seda asjaolu arvesse võtta.

- TS4: UnhandledExceptionTest – selle testi läbikukkumine ei ole otseselt tööriistade viga. See testilugu hõlmab endas töötamist väliste protsessidega, antud juhul Androidi süsteemse rakendusega. Need tööriistad, kus test ei õnnestunud, ei toeta sekkumist välistesse rakendustesse.

- TS5: GesturesTest – see oli ainuke testilugu, millega tekkisid probleemid kõikidel tööriistadel. Rakenduses oli üks väli, mis reageeris liigutustele ning kuvas välja, mille ta registreeris, näiteks vajutades ühe korra ekraanile kuvas rakendus, et registreeritud on ühekordne vajutus. UI Automatoril ei olnud valikus topelt klikki, kuid seda saaks korrata teostades kaks ühest vajutust järjest, aga sel puhul peaks muutma vahendi konfiguratsioonis tegevuste vahelist ajapausi väiksemaks, mis võib halvasti mõjuda vahendi tööle. Liiga kiirete liigutuste tagajärjel hakkavad

tekkima vead ning seade ei jõua järgi. Pikk vajutus oli funktsioonina olemas, kuid reaalselt seda tööle ei saadud. See tähendab, et rakendus ei registreerinud seda pika vajutusena. Ranorex Studios oli samuti topelt klikk funktsioonina olemas, kuid see oli liiga aeglane, mistõttu tuvastati see kahe eraldi klikina. Appiumis sai teostada pikka vajutust, kuid topelt klikiga oli tegelemist rohkem. Seda on siiski võimalik teostada, kuid see tuleb endal programmeerida.

5.1.2 Näited testidest

Alljärgnevalt tuuakse igast katsetatud automatiseerimistööriistast ühe testi näide.

UI Automator

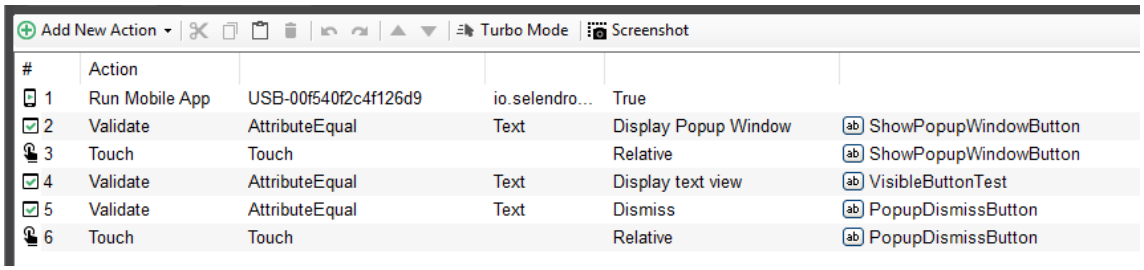
Joonis 3 on kujutatud ekraanipilt testiloost UnhandledExceptionTest. Testi kirjutamisel on kasutatud programmeerimiskeelt Java ja Junit4 raamistikku.

```
18  @RunWith(AndroidJUnit4.class)
19  @SdkSuppress(minSdkVersion = 18)
20  public class UnhandledExceptionTest extends BaseUnitTest {
21
22      @Test
23      public void testUnhandledException() throws UiObjectNotFoundException {
24          UiObject exceptionButton = mDevice.findObject(new UiSelector()
25                  .resourceId("io.selendroid.testapp:id/exceptionTestButton"));
26          exceptionButton.isClickable();
27          exceptionButton.click();
28          assertNotNull(mDevice.findObject(new UiSelector().resourceId("android:id/alertTitle")));
29          UiObject rebootButton = mDevice.findObject(new UiSelector().resourceId("android:id/aerr_restart"));
30          rebootButton.isClickable();
31          rebootButton.click();
32      }
33
34  }
35
```

Joonis 3 UI Automatoris realiseeritud testilugu UnhandledExceptionTest

RanorexStudio

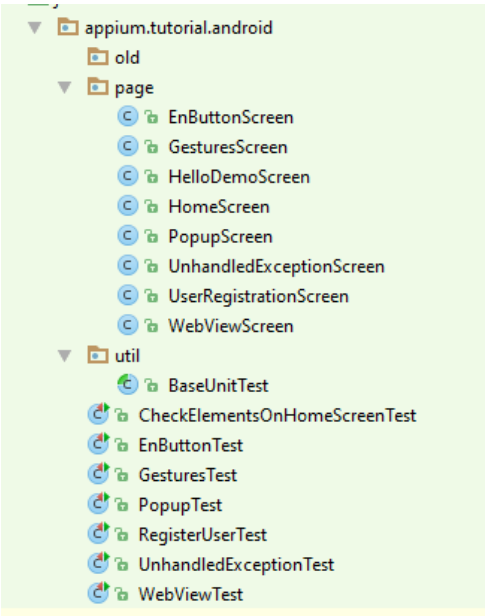
Ekraanipilt Ranorex Studios loodud testiloost PopupTest on kujutatud Joonis 4, kus on näha defineeritud sammud, mida tööriist käivitab. Sammul nr. 1 käivitatakse rakendus, sammul 2 veendutakse et hüpikakna kuvamise nupp on olemas, seejärel vajutatakse nupule ja valideeritakse kaks korda ning suletakse aken nupuvajutusega. Valideerimisi saab lisada jooksvalt testi sisse. Testi kasutajaliides on mugav ja intuitiivne, teste on kerge hallata.



Joonis 4 Ranorex Studios realiseeritud testilugu PopupTest

Appium

Appiumiga automatiseerides kasutasin lehe objekti mustrit. Selleks kirjeldasin erinevad vaated paketis „page“ ning testid sellest eraldi Joonis 5.



Joonis 5 Appiumis kirjeldatud testifailide struktuur

Ülalnimetatud mustri kasutamine võimaldas testi ennast kirjeldada võimalikult lihtsalt ning arusaadavalt (Joonis 6). Iga testiklass päris klassi BaseUnitTest, kus oli defineeritud iga testi eel- ja järeltingimused. Testis endas jäi deklareerida vaated ning neile toetudes ehitada testid. Sellise ehitusega on vajadusel testid loetavad ka äripoole inimestele.

```
11  public class RegisterUserTest extends BaseUnitTest {
12
13      @Test
14      public void testRegisterUser() throws InterruptedException{
15          String username = "MaryS";
16          String email = "mary.strawberry@notexist.com";
17          String password = "12345";
18          String name = "Mary";
19          String programmingLanguage = "Java";
20          HomeScreen homeScreen = new HomeScreen();
21          UserRegistrationScreen userRegistrationScreen = new UserRegistrationScreen();
22          homeScreen.waitingButtonClick();
23          userRegistrationScreen.getProgressBar().isDisplayed();
24          userRegistrationScreen.waitUntilProgressbarDisappear();
25          userRegistrationScreen.loaded();
26          userRegistrationScreen.fillUsername(username);
27          userRegistrationScreen.fillEmail(email);
28          userRegistrationScreen.fillPassword(password);
29          userRegistrationScreen.fillName(name);
30          userRegistrationScreen.clickLanguageDropdown();
31          userRegistrationScreen.selectProgrammingLanguage(programmingLanguage);
32          userRegistrationScreen.clickAcceptAdds();
33          userRegistrationScreen.clickRegisterUser();
34          userRegistrationScreen.clickRegisterUser2();
35          homeScreen.loaded();
36
37      }
```

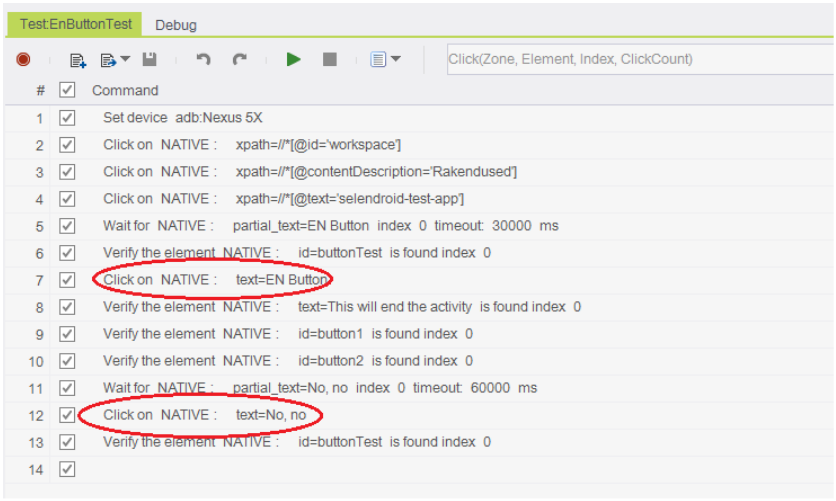
Joonis 6 Appiumis kirjeldatud koodinäide

SeeTest (recorded)

Kuna tegu on tasulise tootega, mis reklaamib oma salvestamise ja taasesitamise võimalust, siis katsetas autor just seda funktsionaalsust. Kirjutatud testide põhjal kujundatud arvamuse järgi jäi antud toode kõvasti alla Ranorexi sarnasele tööriistale. Esiteks oli salvestamine ise ebaintuitiivne. Valideerimisi saab lisada salvestamise ajal, kuid pärast salvestamist tuleb valideerimiskäsud ise õigesse asukohta lohistada. Toode ei jäta meelde, millal neid valideerimisi teostati. Järgmiseks miinuseks on asjaolu, et tööriist ei tuvasta elemente mõistlikul viisil, vaid otsib neid teksti (näiteks nupu nimi) järgi (Joonis 7) , kuigi elementidel on olemas identifikaatorid.

Tööriista kasutajaliideses ei ole võimalik käivitada näiteks paketti testidega, iga test tuleb lahti teha ning käivitada [34], mis on mõeldamatu suure hulga testide puhul.

Otsides tööriista, millel oleks kindlasti „salvesta ja taasesita“ funktsionaalsus ei soovitaks autor valida SeeTesti, kuna selles on antud funktsionaalsus on realiseeritud triviaalsel moel.



Joonis 7 SeeTest testi TestEnButton näide

5.1.3 Testide ajakulu

Tabel 10 võib näha, et vahendite paigaldamisele ja seadistamisele kulus märkimisväärselt erinev aeg, siiski on võimalik näha teatud seaduspärasusi.

- Vahendi seadistamisele kulunud aeg – kõige rohkem läks aega UI Automatori ja Calabashi seadistamisele. Osaliselt selle põhjuseks on UI Automatori puhul asjaolu, et seda seadistati esimesena, seega autoril oli siis teema automatiseerimise kogemust kõige vähem. Siiski oli tegu Java põhise lahendusega, millega on autor tuttav. Kõige enam võttis aega Calabashi seadistamine. Selle puhul oli raske leida juhendit, kuidas seda teha. Allalaetavate .exe formaadis tööriistadega oli ülesseadmine kõige kergem ja kiirem. Üldiselt seda parameetrit ei saa lugeda kaalukaks tööriista valimisel, kuna see on ühekordne ja kokkuvõttes pisike kulu.

- Vahendite esimese testi tööle saamise ajakulude vahe ei olnud enam nii suur. Kõige enam erines Calabash, kuna tegu oli autori jaoks uute tehnoloogiatega.

- Tähtis parameeter on testide loomisele kuluv aeg, kuna aeg tähendab raha kulu ning on tööandjale tähtis, et seda kasutataks võimalikult efektiivselt. Ka selles võrdluses paistavad esile esialgu vahendid, kus koodima ei pidanud. Suurem ajakulu tuleb eelkõige elementide käsitsi tuvastamise vajadusest. Siiski tuleb olla

ettevaatlik salvestatud testide osas, kuna nende haldamisele võib hakata kuluma rohkem aega kui koodi haldamisele [35] ning kokkuvõttes osutuda ebaefektiivseks. Siiski, nagu selgus koodinäidetest, on Ranorexi testi ehitamise loogika väga hea, ning seda võib kaaluda ka päris kasutuseks.

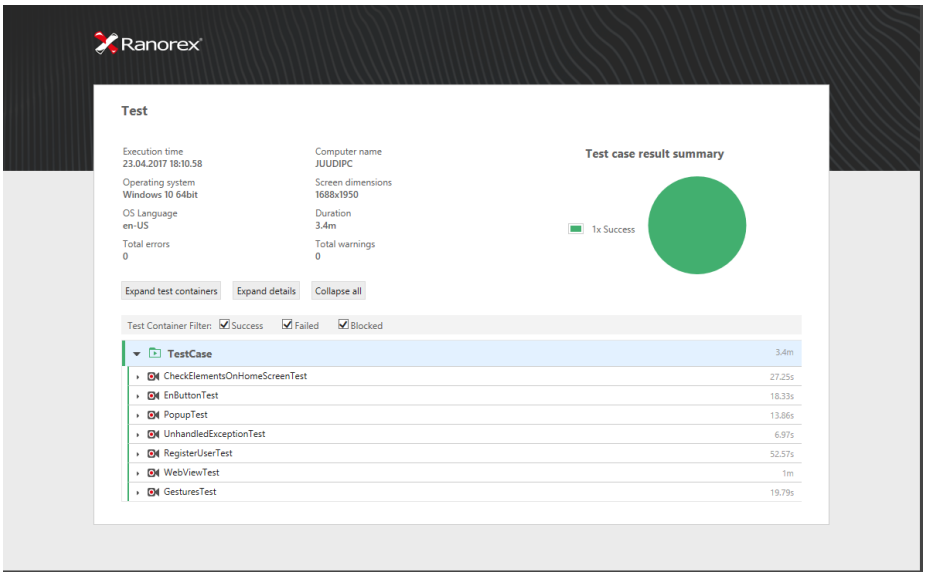
- Vähetähtis ei ole ka testide käivitamisele kuluv aeg. Kõige parema tulemuse saavutas Calabash, mille aeg on märkimisväärselt lühem, kui teistel tööriistadel. Järgmise koha saab UI Automator, mis on ühtlasi ka Javas kirjutatud testidest kõige kiirem. Ülejäänud kolm tööriista jäävad enam-vähem sarnasele tasemele.

Tabel 10 Erinevate tööriistade kasutamisele kulunud aegade võrdlustabel

	UIAutomator	Ranorex Studio	Appium	SeeTest (recorded)	Calabash
Vahendi seadistamisele kulunud aeg	6 h	0.25 h	2 h	1 h	7 h
Vahendil „Hello world“ kirjutamise aeg	1 h	0.20 h	1 h	15 min	2 h
Vahendil 7 testiloo realiseerimise aeg	6 h	2h	12 h	2 h	6 h
Vahendil 7 testiloo käivitamise aeg samal seadmel	1 m 22 s	3 min 24 s	3 min 29 s	2 min 25 s	48 s

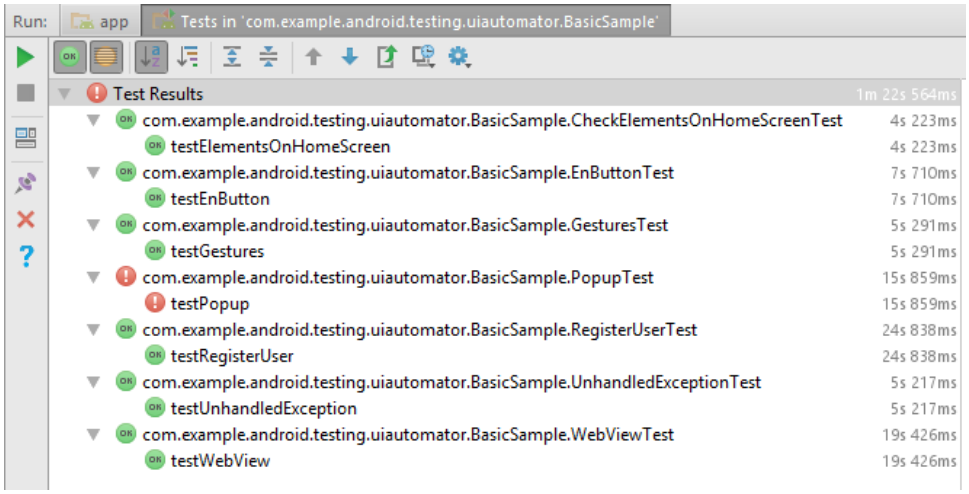
Järgnevalt esitatakse töös näited tööriistade kuvatud raportitest lühikese ülevaatega.

Joonis 8-lt võib näha Ranorex tööriistas testide käivitamise tulemust. Testid käivitatakse toote kasutajaliideses, samuti kuvatakse ka tulemus. Raportis on näha, millal ja mis parameetritega käivitati testid, kui kaua nad jooksid ning mitu testi läbisid või kukkusid läbi. Iga testi on võimalik eraldi avada täpsema info jaoks. Suuremahuliste testilugude jaoks on võimalik kasutada ka sorteerimist tulemuse järgi.



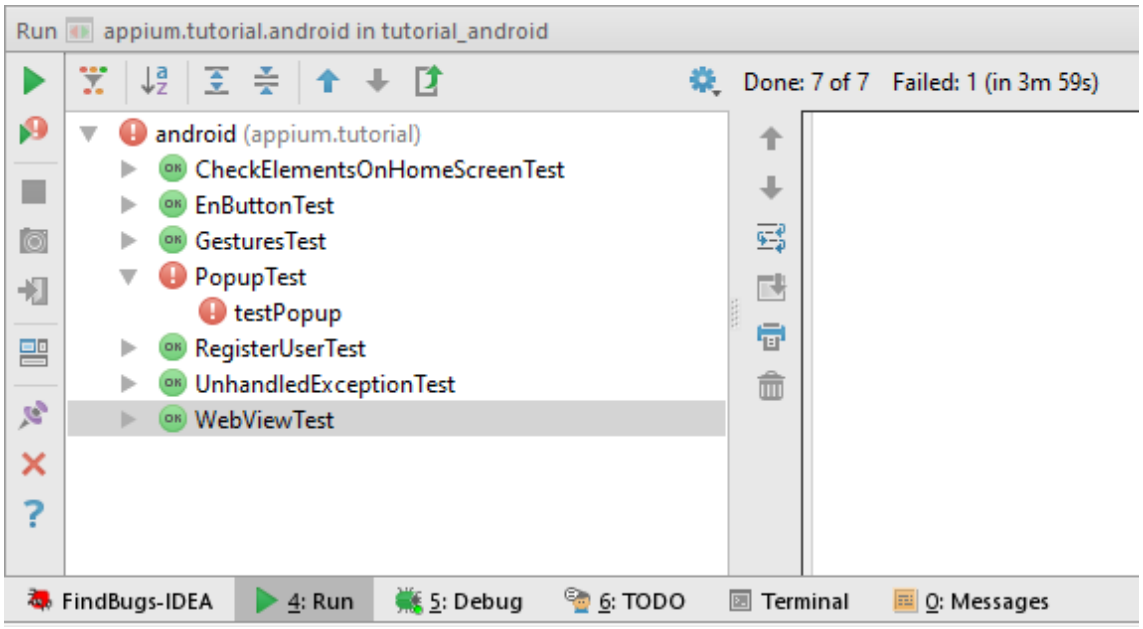
Joonis 8 Ranorex Studio testiraport peale testide käivitamist

Joonis 9 on kujutatud ekraanipilt UI Automatoris loodud testide käivitamise tulemus. Nagu pildilt võib näha, käivitatakse testid ühiktestidena, kuvatakse millised testid läbiti ning millised kukkusid läbi.



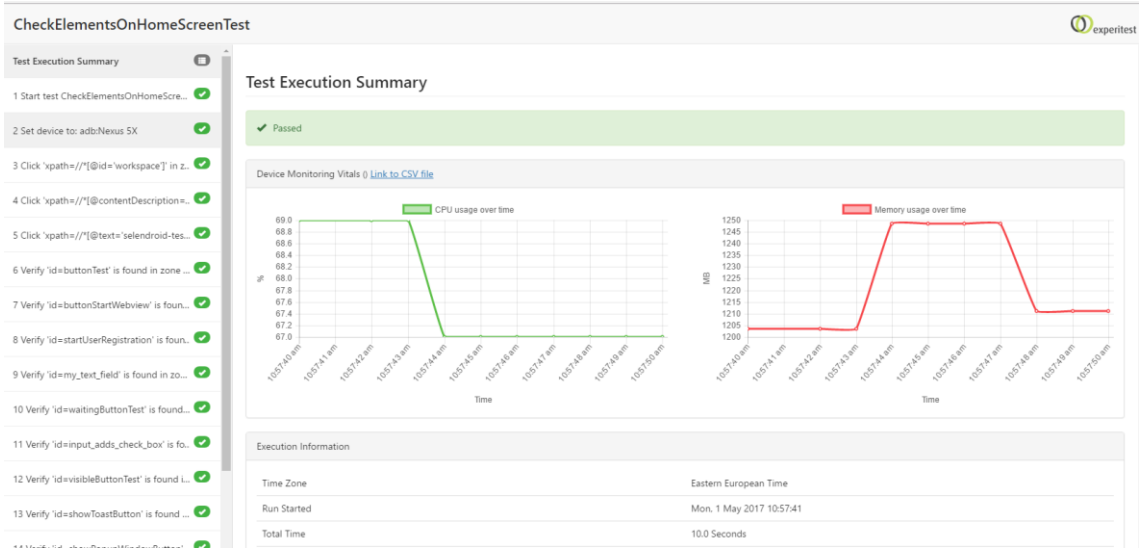
Joonis 9 UIAutomator-is testide kävitamise tulemused

Joonis 10 on kuvatud ekraanipilt Appiumi testide käivitamise tulemusest. Sarnaselt UI Automatorile on antud testid käivitatud IAK-st.



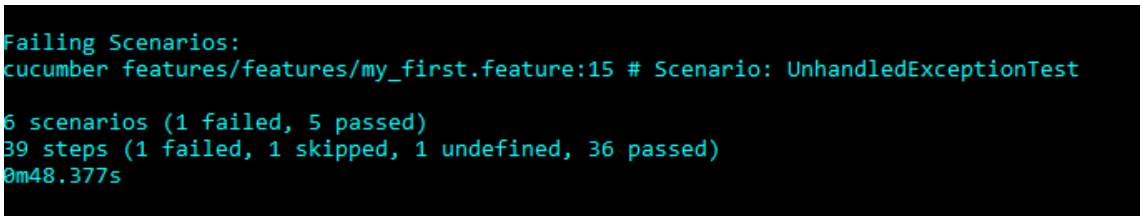
Joonis 10 Appiumis testide käivitamiseks kulunud aeg

Joonis 11-l on kujutatud SeeTest tööriista ühe testi käivitamise raport. Vasakul on näha sammud ning üldvaates on näha testi käivitamise aeg ning arvuti ressursside kasutus. Raport on ainult ühe testi kohta. Vaikimisi salvestab tööriist iga sammu kohta 1-2 ekraanipilti. Selle funktsiooni saab maha keerata deklareerides iga testi ees sellekohase parameetri, mis tähendab, et tööriist ei tee ühtegi pilti üldse. Raport on ainult ühe testi kohta korraga, kuna kasutajaliidese kaudu ei ole võimalik korraga käivitada mitu testi.



Joonis 11 SeeTest tööriistas testi raport

Joonis 12 on kujutatud Calabash testide käivitamise tulemus. Testid tuli käivitada konsooli kaudu ning tulemused kuvatakse seal samas. Antud joonisel on näha, et läbikukkunud stsenaarium oli UnhandledExceptionTest, mida tööriist ei suutnud läbi teha.



Joonis 12 Calabash tööriistas testi raport

5.1.4 Tulemused

UI Automator

Tegu on kergesti õpitava tööriistaga, tänu selle intuitiivsusele. Testid kirjutatakse kasutades JUnit raamistikku, mis teeb testide haldamise ja käivitamise kergeks ja suurele osale ka tuttavaks. Tööriist on laialt levinud, seega suur osa probleeme, millega tuleb kokku puutuda, on juba kellegi poolt lahendatud ning abi leiab suuremale osale küsimustest. Võimaldab ligipääsu rakendusevälistele protsessidele ja on võimalik kasutada seadme füüsilisi nuppe. Javas kirjutatud testidest on kuni kaks korda kiirem võrreldes teiste tööriistadega. Tegemist on Androidi enda tootega, mis annab suure plussi Androidi testimisele, kuid miinusena see töötabki ainult Androidiga.

Ranorex

Tegu on laiapõhjalise kommertstootega, mis sobib pigem suurtesse ja pikaajalistesse projektidesse. See ei ole tööriist, millega kirjeldada minimaalne kogus teste ning need ära unustada. Investeering sellesse tähendab jätkusuutlikku ning läbimõeldud testimis-protsessi. Samuti peaksid sellele tööriistale tähelepanu pöörama need, kes soovivad rakendada järjepidevalt salvesta ja taasesita meetodit. Praktiline osa näitas selle tugevusi, sama väidavad ka testijate ja kvaliteedispetsialistide kommuunid [36].

Appium

Tasuta tööriistadest on see kõige universaalsem, kuna ühelt poolt on kasutajal väga suur vabadus valida teostamise tehnoloogiaid. Teisalt aga kasutab Appium Selenium WebDriverit, mis tähendab olulisel määral stabiilset tehnoloogiat, kuna on W3C poolt standardi kehtestamise protsessi sammus „*Candidate Recommendation*“ [37]. Samuti toetab Appium nii Androidi kui iOS ja kõiki rakenduste tüüpe. Kuna on vabadus valida, mis keeles või tehnoloogias teste kirjutatakse, siis on nendega opereerimine kasutajale ilmselt ka tuttav ja seega mugav.

SeeTest

Kaaludes salvesta ja taasesita meetodit, siis antud tööriist ei ole autori hinnangul selleks sobiv. Salvestaja on sobilik ainult näidistesti loomiseks, et saada tööle esimene test. Samuti ei soodusta kasutajaliides testide haldamist ega käivitamist. Samas on tootja lubanud hulga muid omadusi, mis on mingil määral unikaalsed ning spetsiifilised. Näiteks kaamera või optilise sümbolituvastuse kasutamine.

Calabash

Tööriist, mis toetab sama tootja kommertstoodet testi-pilve. Toetab nii Android kui iOS ning erinevat tüüpi rakendusi, seega on tehnilised võimalused tagatud üsna laiale tarbijale. Testide loomisel kasutatakse Cucumberi ning Rubyt. See on toote nii poolt- kui ka vastuargument. Kui pakutavad tehnoloogiad tunduvad käepärased, siis on see suurepärane vahend, mis praktilises katses näitas ka kõige paremat testide läbijooksmise aega, olles kuni kuus korda kiirem teistest toodetest.

5.2 Analüüs

Töö käigus kogus autor kokku erinevad tööriistad mobiilirakenduste automatiseerimiseks ning lõi nendest ülevaate, seejärel teostas praktilise katse ning tõi välja tulemused.

Mobiilirakenduse automatiseerimistööriista valides tuleks esmalt vastata järgmistele küsimustele:

- Milliseid operatsioonisüsteeme peab tööriist toetama? – Kui rakendus on kättesaadav erinevate operatsioonisüsteemidega seadmetel, kas automatiseerida ainult ühe või mõlema platvormi jaoks, ning kas see kulu on mõistlik. Tuleks arvestada teadmisega, et hetkel hõlmab Android ca 87% ja iOS 13% koguturust.

See tähendab, et luues testid ka ainult Androidile tuleb märkimisväärne osa vigadest juba seal välja. Kõiki vigu ei olegi võimalik üles leida. Juhul kui tegu ei ole kriitilise rakendusega, siis see võib olla vastuvõetav. Samuti tuleb arvestada, et tööriistad, mis on loodud spetsiifiliselt mingi platvormi jaoks töötavad tõenäoliselt paremini sellega, kui üldiselt orienteeritud tööriistad.

- Milline on rakendus: ehe, hübriid või veebielementidega? – selles punktis järeleandmisi või kahtlusi peaks olema üsna vähe. Ilmselt automatiseerimis-tööriista valiku ajaks on rakendus juba valmis, seega tuleb lihtsalt jälgida, et rakenduse tüüp oleks toetatud valikusse kaasatud tööriistade poolt. Õnneks iga tüübi jaoks on valik olemas, muidugi kõige laiem on see ehedate rakenduste testimiseks, kuna seda toetavad kõik tööriistad.

- Kas funktsioon „Salvesta ja esita“ on tähtis? Kui vastate jah, siis miks? Kas plaanite arvestatava osa testidest luua just selle meetodiga? – antud lähenemisega on võimalik kokku hoida märkimisväärselt aega ning ka tööjõukulusid. Siiski tuleb sellise lähenemise osas olla kriitiline ning valida parim tööriist selle jaoks, vastasel juhul kokkuhoiud lihtsalt suunatakse testide haldamisse. Antud töös teostatud praktilise katse käigus leidis autor, et kõige parem tööriist antud meetodi jaoks oli Ranorexil. SeeTesti pakutavat salvestajat autor ei soovita, kuna testide loomine ei olnud mugav, lisaks testide haldamine ja käivitamine oli lahendatud samuti ebasoodsalt. Antud vahendi tööriist sobis pigem esimese testi loomiseks, et genereeritud koodi edasi arendada.

- Kas on olemas ligipääs lähtekoodile? – Kui arendatakse ja testitakse samas ettevõttes ei pruugi see punkt olla tähtis. Siiski on lähenemistes põhimõtteline vahe. Seega tuleks läbi mõelda, mis eesmärgid on testidel ning kes hakkab neid teste kirjutama. Valge karbi printsiipi toetavad vahendid on Robotium ja Espresso. Mõlemad tööriistad on mõeldud ainult Android platvormile. Ülejäänud tööriistad on mõeldud testimiseks ilma lähtekoodita.

- Kas on vahe, millises keeles arendada, mis tehnoloogiaid kasutada? – Ei ole vähetähtis, milline on automatiseerija kogemused ja eelistused. See ei saa olla põhiline argument, kuid võrdväärse valiku puhul võib olla määrav. Näiteks Calabashi saab kirjutada ainult Ruby-s. UI Automatoris, Espressos ja Robotiumis

saab kasutada ainult Javat. Ranorexis, SeeTestis ja Appiumis on võimalik valida kasutatavate tehnoloogiate seas.

- Kas olete valmis toote eest maksma? – Küsimus on aktuaalne, kuna turul on palju häid vabavaralisi tööriistu. Tasulise toote lisandväärtus peab olema tõesti tuntav, et otsustada selle kasuks. Antud võrdluses olid tasulised tooted Ranorex ning SeeTest. Ranorexil on väga hea salvesta ja esita funktsionaalsus, mida autor kiidab. SeeTest paistab silma keerukamate ja spetsiifilisemate lahendustega, nagu võimalus kasutada kaamerat, sõrmejäljelugejat. Sellised tööriistad ei sobi väikestesse projektidesse, kuna nendesse investeerimine ei tasu ilmselt ära.

Kombinatsioon vastustest antud küsimustele kitsendab valikut oluliselt. Ilmselt nii palju, et mõnes punktis tuleb teha järeleandmisi, näiteks otsustada, kas on tähtsam, et toode oleks tasuta, või toode, millel oleks „Salvesta ja taasesita“ funktsionaalsus tagatud parimal võimalikul viisil.

Järgmise sammuna on oluline kaardistada oma toote ja vajaduste eripärad. Hea on luua nimekiri märksõnadest, mis on kriitilise tähtsusega toote puhul: näiteks kaamera kasutamise võimalus, asukoha teesklemise võimalus, hüpik-akende olemasolu, võimalus rakenduseväliselt seadet juhtida, spetsiifilised liigutused, millele rakendus peaks reageerima. Need märksõnad tuleks sõelale jäänud töövahendite kontekstis läbi töötada ning otsida võimalikke kitsaskohti. Tööriistad on erinevad ning nende nõrgad küljed võivad avalduda ootamatutes kohtades.

6 Kokkuvõte

Töö eesmärgiks oli anda ülevaade hetkel saadaolevatest Androidi mobiilirakenduste automatiseerimise tööriistadest, läbi töötada erinevad tooted, võrrelda neid algmaterjali ka praktilise katse käigus ning luua lähtematerjal vahendite tundmaõppimiseks ja nendes orienteerumiseks.

Töö tulemuseks võrreldi seitset automatiseerimisvahendit teoreetilise materjali põhjal ning praktilise katse käigus töötati viie tööriistaga. Katse tulemusena valmis igas vahendis näide kirjeldatud testilugude realiseerimiseks. Antud tööd analüüsiti ning võrdluse tulemusena toodi välja vahendite tugevad ja nõrgad küljed.

Antud töös keskenduti vahenditele, mis katavad Androidi ning lisaks ka teisi operatsioonisüsteeme. Tööst jäid välja ainult iOS platvormile orienteeritud tööriistad, mis võib ka üks võimalus, mida tulevikus täiendada. Samuti muutub antud valdkonnas olukord kiiresti, seega mõne aja pärast võivad tööriistad olla suurema funktsionaalsusega ning turul olla ka uusi tegijaid.

Püstitatud eesmärgid saavutati. Antud töö on suureks abiks alustajale, andes suuna tegutsemiseks ning säästes olulisel määral aega esmasele infokorjele.

Kasutatud kirjandus

[1] „Mobile marketing statistics 2017“. [Online]. Available at: http://www.smartinsights.com/mobile-marketing/mobile-marketing-analytics/mobile-marketing-statistics/. [Vaadatud: 22-märts-2017].

[2] Dan Morrill, „Announcing the Android 1.0 SDK, release 1 | Android Developers Blog“, 2008. [Online]. Available at: https://android-developers.googleblog.com/2008/09/announcing-android-10-sdk-release-1.html. [Vaadatud: 04-mai-2017].

[3] „IDC: Smartphone OS Market Share 2016, 2015“. [Online]. Available at: http://www.idc.com/promo/smartphone-market-share/os. [Vaadatud: 04-mai-2017].

[4] J. Gao, X. Bai, W.-T. Tsai, ja T. Uehara, „Mobile Application Testing: A Tutorial“, *Computer (Long. Beach. Calif).*, kd 47, lk 46–55, 2014.

[5] Z. Liu, B. Liu, ja X. Gao, „Test automation on mobile device“, *Proc. 5th Work. Autom. Softw. Test AST 10*, nr 2007, lk 1–7, 2010.

[6] C. Hu ja I. Neamtiu, „Automating gui testing for android applications“, *6th Int. Work. Autom. Softw. Test (AST 2011)*, nr Section 4, lk 77–83, 2011.

[7] S. R. Choudhary, A. Gorla, ja A. Orso, „Automated test input generation for android: Are we there yet?“, *Proc. - 2015 30th IEEE/ACM Int. Conf. Autom. Softw. Eng. ASE 2015*, lk 429–440, 2016.

[8] „UI/Application Exerciser Monkey | Android Studio“. [Online]. Available at: https://developer.android.com/studio/test/monkey.html. [Vaadatud: 22-märts-2017].

[9] D. Amal, A. R. Fasolino, P. Rio Tramontana, B. D. Ta, ja A. M. Memon,

„MobiGUITAR Automated Model-Based Testing of Mobile Apps“, *Software, IEEE*, kd 32, nr 5, lk 53–59, 2014.

[10] „GUITAR - A GUI Testing Framework download | SourceForge.net“. [Online]. Available at: https://sourceforge.net/projects/guitar/. [Vaadatud: 22-märts-2017].

[11] „DynoDroid“. [Online]. Available at: http://www.dynodroid.com/. [Vaadatud: 22-märts-2017].

[12] S. Gunasekaran ja V. Bargavi, „Survey on Automation Testing Tools for Mobile Applications“, *Int. J. Adv. Eng. Res. Sci.*, kd 2, nr 11, lk 2349–6495, 2015.

[13] C. H. Liu, C. Y. Lu, S. J. Cheng, K. Y. Chang, Y. C. Hsiao, ja W. M. Chu, „Capture-replay testing for android applications“, *Proc. - 2014 Int. Symp. Comput. Consum. Control. IS3C 2014*, lk 1129–1132, 2014.

[14] „TestAutomationPatterns - CAPTURE-REPLAY“. [Online]. Available at: https://testautomationpatterns.wikispaces.com/CAPTURE-REPLAY. [Vaadatud: 22-märts-2017].

[15] N. H. Saad ja N. S. A. A. Bakar, „Automated Testing Tools for Mobile Applications“, *IEEE*, 2014.

[16] L. Kasvand ja J. Ivask, „Automatiseerimisvahendi valik mobiilirakenduse testimiseks agiilses keskkonnas“, 2015.

[17] „The Hello World Collection“. [Online]. Available at: https://helloworldcollection.github.io/. [Vaadatud: 22-märts-2017].

[18] P. Singh Kochhar, F. Thung, N. Nagappan, T. Zimmermann, ja D. Lo, „Understanding the Test Automation Culture of App Developers“, 2015.

[19] „JUnit - About“. [Online]. Available at: http://junit.org/junit4/. [Vaadatud: 22-märts-2017].

[20] „Monkeyrunner | Android Studio“. [Online]. Available at:

https://developer.android.com/studio/test/monkeyrunner/index.html. [Vaadatud: 22-märts-2017].

[21] „Robotium Tech“. [Online]. Available at: https://robotium.com/. [Vaadatud: 22-märts-2017].

[22] T. Samuel ja D. Pfahl, „Problems and Solutions in Mobile Application Testing“, kd 5089, lk 217–232, 2016.

[23] J. A. Gerard Meszaros, Shaun M. Smith, „The Test Automation Manifesto“, *Extrem. Program. Agil. Methods - XP/Agile Universe 2003*, kd 2753, lk 73–81, 2003.

[24] „What is Software Testing?“ [Online]. Available at: http://istqbexamcertification.com/what-is-a-software-testing/. [Vaadatud: 09-apr-2017].

[25] C. Klonk, R. Copyrigh, F. Copyrigh, ja K. Ol, „Certified Tester Foundation Level Syllabys“. [Online]. Available at: http://www.istqb.org/downloads/send/2-foundation-level-documents/3-foundation-level-syllabus-2011.html4. [Vaadatud: 07-apr-2017].

[26] G. J. Myers, C. Sandler, ja T. Badgett, *The art of software testing*. John Wiley & Sons, 2012.

[27] James Bach, „What is Exploratory Testing?“ [Online]. Available at: http://www.satisfice.com/articles/what_is_et.shtml. [Vaadatud: 10-apr-2017].

[28] „Mobile App Test Coverage Model : LONG FUN CUP – The Pragmatist“. [Online]. Available at: https://testingideas.wordpress.com/2014/08/17/mobile-app-test-coverage-model-long-fun-cup/. [Vaadatud: 07-apr-2017].

[29] „What is automated software testing?“ [Online]. Available at: http://searchsoftwarequality.techtarget.com/definition/automated-software-testing. [Vaadatud: 10-apr-2017].

[30] AMIR GHAHRAI, „Why Would You Want To Automate a Test?“ [Online]. Available at: http://www.testingexcellence.com/why-would-you-want-to-automate-a-test/. [Vaadatud: 07-apr-2017].

[31] R. Ramler ja K. Wolfmaier, „Economic Perspectives in Test Automation: Balancing Automated and Manual Testing with Opportunity Cost“.

[32] „5 Best Automation Tools for Testing Android Applications — Software Testing Help“. [Online]. Available at: http://www.softwaretestinghelp.com/5-best-automation-tools-for-testing-android-applications/. [Vaadatud: 03-mai-2017].

[33] „UIAutomator not able to access android.widget.PopupWindow [37017411] - Issue Tracker“. [Online]. Available at: https://issuetracker.google.com/issues/37017411. [Vaadatud: 22-apr-2017].

[34] „Running Multiple Scripts Of A Single Test Suite | ExperiTest.com forums“. [Online]. Available at: https://forum.experitest.com/content/running-multiple-scripts-single-test-suite. [Vaadatud: 01-mai-2017].

[35] M. Leotta, D. Clerissi, F. Ricca, ja P. Tonella, „Capture-replay vs. programmable web testing: An empirical assessment during test case evolution“, *2013 20th Working Conference on Reverse Engineering (WCRE)*, 2013, lk 272–281.

[36] „Ranorex - Automated Testing Tool for Desktop, Web & Mobile Applications“. [Online]. Available at: http://www.methodsandtools.com/tools/ranorex.php. [Vaadatud: 04-mai-2017].

[37] „W3C Candidate Recommendation 30 March 2017“. [Online]. Available at: https://www.w3.org/TR/webdriver/. [Vaadatud: 04-mai-2017].

Lisa 1

Loodud automaattestid on kättesaadavad järgmisel aadressil:

https://github.com/salomevellemaa/test-automation-examples