VILNIAUS UNIVERSITETAS
MATEMATIKOS IR INFORMATIKOS FAKULTETAS
PROGRAMŲ SISTEMŲ KATEDRA

Mobiliųjų aplikacijų testavimas
Mobile Application Testing

Bakalauro darbas

Atliko: Steponas Ivašovas (parašas)

Darbo vadovas: lekt. Vytautas Valaitis (parašas)

Darbo recenzentas: Tomas Plankis (parašas)

Vilnius 2016

Santrauka

Šiame technologijų ir informacijos amžiuje turime didelę įvairovę mobiliųjų įrenginių platformų su joms skirtomis aplikacijomis ir dar begalė jų yra kuriamos, taigi savaime aišku , kad visos juos turi atitikti kokybės reikalavimus. Esant tokiai įvairovei yra neįmanoma visus variantus ištestuoti rankiniu būdu, todėl reikalingas automatinis testavimas. Automatinės testavimo priemonės kuriamos pagelbėti tiek aplikacijų kūrėjams, tiek vartotojams. Siekiant išsiaiškinti kokią automatizuotą testavimo įrangą geriausiai naudoti testuojant mobiliuosius įrenginius reikia išsiaiškinti pagal kokius kriterijus reikia ją atrinkti, o kad atrinkti kriterijus reikia pažinti mobiliojo testavimo ypatumus ir iššūkius.

Summary

In this age of technology and information we have large diversity of mobiles devices and each has its own different platform, furthermore each platform has its own specific applications, which are number grows in thousands by the day. It is necessary to ensure quality of applications and ensure their compatibility with all platforms for that we must use automated testing. To choose right testing tool we need to a set of criteria to characterize them, and to figure out what criteria are important we need to know ins and outs of mobile testing.

Turinys

Įvadas

„Mobilus įrenginiai sparčiai keičia darbalaukio kompiuterius ir tampa svarbi mūsų gyvenimo dalimi.“ [KAU15] Šiame amžiuje mobiliosios aplikacijos tampa dešiniąja mūsų ranka, tiek darbe, tiek ir asmeniniame gyvenime. Be jų nei vienas neįsivaizduojame normalios darbo dienos. Jos tampa plačiai naudojamos viešuosiuose ir privačiuose sektoriuose. Sparčiai besivystant technologijoms, kai rinką užvaldo išmanieji telefonai ir kuomet jais naudojantis patogu atlikti ne tik pagrindines funkcijas, bet ir naudotis kitomis teikiamomis galimybėmis: rašyti laiškus, pasirašyti dokumentus mobiliuoju parašu, kurti rašytinius darbus, atlikti sudėtingas finansines operacijas ir galų gale praleisti laisvalaikį. Todėl joms yra būtinas nuodugnus jų testavimas ir saugumas.

Šiuo metu naujuose mobiliuose telefonuose yra plačiai taikoma įvairi mobilioji įranga, kuri nelabai skiriasi nuo kompiuterinės, o programinės įrangos kūrimo procese didelę dalį užima jos testavimas. Šiais laikais nors pažengus technologijoms mobilių įrenginių automatinis testavimas vis dar yra galvos skausmas daugeliui programuotojų, o dar čia prisideda aplikacijų kūrimas ir papildomas testavimas. Programuotojai kurdami testavimo įrangą turi atsižvegti į mobiliųjų įrengimų darbo spartą, ekranų dydžius, platformų įvairovę, atminties kiekį, bei energiją, programavimo kalbą, aplinką Testavimo proceso automatizavimas leidžia sumažinti testų paleidimų kiekį ir trukmę, kas labai pagelbsti testuojant mobiliuosius įrengimus.

Kadangi turime didelę įvairovę mobiliųjų įrenginių platformų su joms kurtomis aplikacijomis, taigi turime ir daug automatizuotų testavimo būdų. Esant tokiai įvairovei sunku atsirinkti, kuri priemonė testavimo yra pati geriausia. Siekiant išaisktinti kokią automatizuotą testavimo įrangą naudoti testuojant mobiliuosius įrenginius reikia apžvelgti kokios platformos yra ir kokie automatiniai testavimai yra sukurti. Juos palyginti ir išrinkti geriausius.

Darbo tikslas atlikti mobiliųjų aplikacijų testavimo įrankių analizę atsižvelgiantis į objektų atpažinimo ir skripto testavimo ypatumus

Uždaviniai

Siekiant tinkamai išanalizuoti automatinius aplikacijų testavimo įrankius mobiliesiems įrenginiams yra iškelti šie uždaviniai:

1. Identifikuoti mobiliųjų aplikacijų testavimo ypatumus atsižvelgiantis į skirtingas mobiliuose įrenginiuose naudojamas operacines sistemas
2. Pasiūlyti kriterijus mobiliųjų aplikacijų testavimo įrankiams įvertinti
3. Atlikti mobiliųjų aplikacijų įrankių testavimo vertinimą atsižvelgiantis į testavimo skripto ypatybes ir objekto atpažinimo galimybes.
4. Pasiūlyti rekomendacijas mobiliųjų aplikacijų testavimo įrankių praplėtimui

Šie klausimai labai svarbūs tobulėjant išmaniesiems telefonams ir jų techninėms galimybėms bei didėjant jų pardavimams Lietuvoje ir pasaulyje. Taip yra todėl, kad kuriamos aplikacijos tampa funkcionalesnės, todėl tokių programų testavimas tampa sudėtingesnis. O kokybiškas testavimas yra pirmas žingsnis aukštos kokybės aplikacijų plėtros link. Verslo požiūriu aplikacijų testavimas turi būti atliktas kiek galima kruopščiau ir efektyviau. Kaip žinoma automatinių testų kūrimas sudaro didelę laiko dalį ir savaime aišku tas turi įtakos biudžetui.

1. Mobiliosios operacinės sistemos

Mobilioji operacinė sistema, taip yra vadinama operacinės sistemos naudojamos mobiliuose įrenginiuose tokiuose kaip išmanusis telefonas, planšetė, telefonas ir kiti mobilus įrenginiai. Tačiau taip jau susiklostė istoriškai, kad tokiems įregniniams kaip nešiojami kompiuteriai yra naudojamos paprastos operacinės sistemos, jie nelaikomi mobiliais įrengimais. Tolimesnis technologijų vystymasis privertė sukurti operacinių sistemų hibridus skirtus tik išmaniems telefonams, planšetėms.

Šiuo metu pasaulyje naudojamos ir dažniausiai sutinkamos mobiliuose įrenginiuose operacinės sistemos yra:

- Android
- iOS
- Windows Phone 8/ windows Mobile
- Blackberry
- Symbian
- Kiti (MeGo (Nokia))

Pateikiu lentelę (lentelė 1) su mobilių operacinių naudojimo procentine dalimi :

lentelė 1Mobiliųjų sistemų populiarumas [1]

Operacinė sistema	2011 metų rinkos dalis	2015 metų rinkos dalis
Android	38,90 %	43,80 %
Blackberry (OS)	14,20 %	13,40 %
Symbian	20,60 %	0,10 %
iOS	18,20 %	16,90 %
Windows Phone 7/8.1 Windows Mobile	3,80 %	20,30 %
Kiti (MeGo ir kt.)	4,30 %	5,50 %
Bendras	100,00 %	100,00 %

Kaip matome populiariausia mobili operacinė sistema yra tapusi Android, po jos seka Windows Phone sistemos. Kitos mobiliosios operacinės sistemos yra išstumiamos iš rinkos, arba dėl susdėtingo jų vartojimo, arba tiesiog nepratesiama mobiliųjų įrengimų gamyba. Taipogi pateikiu lentelėje (2 lentelė) populiariausių mobiliųjų operacinių palyginimą:

lentelė 2 Mobilių operacinių sistemų palyginimas

Savybės	Android	Windows Phone	iOS
Kompanija	Google	Microsoft	Apple
Paskutinė versija	6	8.1	9
OS šeima	Linux	Windows Mobile	Darwin
Programuojama kalba	C,C++, Java	C, C++	C, C++, Objective-C, Swift
Licenzijos	Atviro kodo	Uždaro kodo	Uždaro kodo
Leidimai diegti aplikacijas suteikiami	Prieš diegiant.	Diegimo metu.	Prieš paleidžiant aplikaciją.

Lentelėje matome kas kuria mobilias operacines sistemas, kokios versijos yra galutinės šiuo metu rinkoje, kokia kalba rašomos ir kada gaunami leidimai diegti aplikacijas.

Dabar trumpai apžvelgsiu kokios aplikacijos populiariausios mobiliuose telefonuose pagal operacinę sistemą:

- *Apple iOS yra naudojama iPhone ir iPad.* Ji turi aplikacijų pardavimo „turgų“ (angl. App store). Daugiau nei 92% iPhone vartotojų kasdien naudojasi aplikacijomis skirtomis jų telefonui. Populiariausios - informacinės, žaidimų, parduotuvių aplikacijos. Pasirinkus sau patogią aplikaciją, klientas gali padidinti savo mobiliojo telefono galimybes ir gauti daugiau naudos iš mobiliojo telefono. Visos mokėjimo operacijos, susijusios su iPhone aplikacijomis, vykdomos per App Store, kuris priklauso kompanijai Apple. Kasdien dešimtys milijonų iPhone ir iPad naudotojų įvykdo įvairiausius veiksmus: perka prekes/paslaugas arba perka pačias aplikacijas, atnešdami pelną aplikacijų savininkams.
- *Android naudoja Samsung, LG, Huwei.* Tai yra kas dešimtas išmanusis telefonas. Iš viso išmanieji telefonai užima 1/3 mobiliųjų telefonų rinkos ir šis skaičius sparčiai auga, 80 procentų šių telefonų turėtojų naudojasi įvairiomis aplikacijomis. Prognozuojama, kad būtent Android OS po kelerių metų užims dominuojančias pozicijas rinkoje.

Informacinės populiariausios aplikacijos šiai operacinei sistemai yra: promo aplikacijos; žaidimai ir pramogų sistemos; Kompleksiniai sprendimai: integracija su internetine svetaine, su CRM ir ERP sistemomis, su buhalterinės apskaitos programomis; Verslo valdymo sistemos; Internetinių svetainių mobiliosios versijos (ypač aktualu naujienų portalams). Būtent šių aplikacijų populiarumą lemia, tai kad būtent ši mobili operacinė sistema jau užima didžiąją rinkos dalį ir ja naudojamasi darbuose.

- *WinCE/Windows Mobile naudoja Nokia.* Nors ši operacinė jau galima sakyti buvo išėjusi iš rinkos, tačiau esamais duomenimis Windows phone 7 naudojo apie 2 % išmaniųjų telefonų, o atsiradus windows phone 8.1 operacinei sistemai sulig kiekviena diena išlėto šis procentas auga. Prognozės kūrėjų yra itin optimistinės. Šiuo metu - tai vienintelė operacinė sistema, kuri realiai gali konkuruoti ir į šoną nustumti iOS ir Android. Paskutinėje "AppNation" konferencijoje, kurioje dalyvavo aplikacijų kūrėjai ir marketingo specialistai iš viso pasaulio, buvo pažymėta kad Windows Phone 8.1 pagal populiarumą tarp mobiliųjų paltformų pakilo į trečią vietą, aplenkdama Blackberry. Daugelis analitikų prognozuoja kad Windows phone 8.1 taps populiariausia mobiliąja platforma 2015.
- *Blackberry produkcija* - verslo klasės telefonai ir planšetiniai kompiuteriai, naudojantys Blackberry OS. Blackberry naudotojai - tai pasiturintys arba vidutines pajamas gaunantys žmonės. Oficiali Blackberry aplikacijų parduotuvė - Blackberry App World http://appworld.blackberry.com, iš kurios kasdien parsisiunčiama virš 6 000 000 programų.

2. Automatizuotas testavimas

„Mobilios aplikacijos sparčiai tampa vis labiau sudėtingesnės, taip padidiną funkcinio testavimo kiekį, kad susidoroti su šia problema, testavimo organizacijos nuolatos ieško alternatyvų tradiciniam rankiniam testavimui.“ [SUD13]

Automatizuotas testavimas – tai testavimas, kuris gali būti atliekamas kasdien ar ištisą parą automatizuotai, be rankinių veiksmų vedant klaidų taisymo registrą. Testuotojas reikalingas tik rezultatams išanalizuoti, kai jau yra pranešimai apie klaidas ar iškilusias problemas. Taigi kas gali būti padaryta automatiškai per vieną parą, neverta daryti dvi darbo dienas testuotojui sėdint šalia testuojamojo įrenginio.

Siekiant išsiaiškinti kas tiksliai yra automatizuotas testavimas visų pirma išsiaiškinsime kokie įrankiai naudojami jam atlikti ir kaip jis veikia. Taigi automatizuoto testavimo įrankiai yra suskirstyti į tris kategorijas:

- „unit test”, tam tikrų programos modulių testavimas atskirai, stebint sistemos parametrus. Ji išmatuoja reikšmes ir praneša apie esamas sistemos problemas
- „įrašymo-pakartojimo” testavimas vykdomas įrašant vartotojo veiksmus (tipiškai kai dirba su vartotojo sąsaja). Šie veiksmai yra koreguojami ir pagal poreikį pakartojami.
- apkrovos testavimo įrankiai vykdomi tuo pačiu metu atkartojant daugybės vartotojų veiksmus, kur pasekoje kartojimo tikrinamas sistemos apkrovimas.

Tačiau automatinis testavimas turi tiek privalumų, tiek ir trūkumų. Taigi automatinio testavimo privalumai:

- Leidžia išvengti žmogiškojo faktoriaus.
- Tą patį testą galima atlikti ilgai (defektų atkartojimas, regresinis testavimas) nereikalaujant testuotojo buvimo šalia.
- Galima testuoti tą patį scenarijų su daug skirtingų reikšmių, taip užtikrinant aukšos kokybės testavimą.
- Ilgo testavimo metu galima aptikti netikėtas klaidas.

O trūkumai būtų šie:

- Reikalinga didelė pradinė investicija (laiko, resursų).
- Ne visada atsiperka. Tarkim testo kūrimas nepavyks ar jo nereikės vykdyti daug kartų, tokiu nereikėtų automatizuoti.
- Aptinka ne visus defektus .

Automatizuoti testus verta, kai testas bus vykdomas daug kartų. Nepamirškime jog automatizuotas testas yra skirtas patikrinti ne logiką, o funkcionalumą palaikantį kodą.

Pagrindiniai automatizuotų testų kūrėjai:

1. MERCURY™
2. COMPUWARE
3. IBM
4. EMPIRIX
5. SEGUE

Tačiau mobiliųjų aplikacijų automatizuotus testus kuria ir daug kitų kompanijų, tokių kaip:

- GorilaLogic
- Ranorex
- IMPETUS
- Keyton system
- Mobile labs
- ir kt.

Automatinių testų kūrimo kompanijos konkuruoja tarpusavyje dėl automatizuotų testų patogumo, našumo, tikslesnio klaidų radimo ir testų ataskaitų informatyvumo.

Nepaisant to, kad jos kuria testavimo sistemas skirtingoms platformoms ir testavimas vykdomas skirtingais būdais, rezultatas vienas ir tas pats – efektyvus klaidų radimas ir patogus testavimas.

3. Mobiliojo testavimo iššūkiai

Mobiliųjų aplikacijų testavimas skiriasi nuo tradicinių darbalaukio aplikacijų testavimo daugelyje sričių. Šiame skyriuje aptarsime pagrindinius testavimo iššūkius mobiliųjų aplikacijų testavime ir kodėl tradiciniai testavimo įrankiai ir metodai netinka mobiliosioms aplikacijomis :

3.1 Mobilusis ryšis

Vienas svarbiausių mobilių aplikacijų testavimo iššūkių yra skirtingų mobilių įrenginių ryšis su skirtingais ryšio tinklais bei įrenginiais. Skirtingai nei darbalaukio aplikacijos, kurios naudoja statiškus tinklo ryšius, mobiliosios aplikacijos jungiasi prie mobiliųjų ryšių, kurie gali skirtis greičiu, saugumu ir patikimumu. Įprasti mobiliųjų tinklų tipai yra 2G, 3G, 4G ir daugelis belaidžių (angl. wireless) tinklų. Mobiliosios aplikacijos stipriai priklauso nuo mobiliųjų tinklų, o tai gali neigiamai paveikti mobiliųjų aplikacijų patikimumą, greitį, saugumą bei aplikacijos funkcionalumą. Dėl šių nepastovumų svarbu ištestuoti aplikacija skirtinguose aplinkose :

- aplinka, kurioje yra pastovus ryšis su mobiliuoju tinklu,
- aplinka, kurioje yra kintantis ryšis su mobiliuoju tinklu,
- aplinka, kurioje nėra ryšio.

Remiantis testavimo procedūrų sunkumais ir reikalavimais, skirtingi testavimo būdai yra rekomenduojami. „Dėl mobiliosios aplikacijos patikimumo, greičio, saugos ir teisingo funkcionalumo stipraus priklausymo nuo turimo ryšio, turi būti atlikti išsamūs funkciniai testai skirtinguose ryšio scenarijuose ir tinkluose“ [MFE12].

3.2 Vartotojo sąsajų įvairovė

Mobiliosios operacinės sistemos turi skirtingus vartotojo sąsajas, kurios yra apibrėžtos taisyklėmis ir gairėmis. Sąsajos elementų išsidėstymas ir panaudojimas yra patikrinamas patvirtinimo procese įkeliant mobiliąsias aplikacijas į rinką. Nesuderinamumas su taisyklėmis ir gairėmis gali atidėlioti publikavimo procesą, padidinti programavimo ir testavimo kainą. Skirtingi ekrano dydžiai taip pat gali paveikti mobiliosios aplikacijos išvaizdą ir naudojamumą (angl. usability). „Skirtingi mobilieji įrenginiai gali skirtingai reaguoti į tą patį aplikacijos kodą, o tai turi būti ištestuota testuojantis grafinę vartotojo sąsają.“ [MFE12].

3.3 Resursų apribojimai

Mobiliosios aplikacijos naudoja mobiliųjų įrenginių resursus, kurie yra smarkiai apriboti. Nepaisant spartaus mobilių įrenginių vystymosi, svarbu kad resursų sunaudojimas būtų stebimas ir kontroliuojamas visais laikais. Mobiliųjų įrenginių resursai yra šie : procesorius, RAM, atmintis, lietimosi ekranas (angl. touch screen), baterija bei įvairus moduliai ir sensoriai. Perdėtas resursų naudojimas gali sumažinti mobiliojo įrenginio greitį, o tai gali sukelti sutrikimus mobiliojoje aplikacijoje. Testavimo proceso metu, resursų suvartojimas turi būti nuolatos stebimas.

3.4 Resursų valdymas

Mobiliesiems įrenginiams veikti reikalinga energija. Visi mobiliojo įrenginio resursai ir veiklos naudoja energiją, bet ne vienodai. GPS sensoriai, duomenų persiuntimas bei video medžiagos redagavimas yra veiklos, kurios suvartoja daugiau energijos nei kitos veiklos. Šios veiklos naudoja kelis mobiliojo įrenginio resursus arba reikalauja nuolatinio duomenų ryšio, kuris ir yra pagrindinė priežastis, kodėl yra suvartojama daug energijos. „Skirtingos veiklos turi skirtingą įtaką mobiliajam įrenginiui ir turi būti stebimos testavimo proceso metu.“ [MFE12].

3.5 Kontekstas

Kontekstas tai bet kokia informacija, kuria galima panaudoti charakterizuojant subjekto situaciją. Subjektas tai asmuo, vieta arba objektas, kuris yra aktualus vartotojui sąveikaujant su aplikacija. Kontekstas gali apriboti arba praplėsti mobiliosios aplikacijos operacijas arba funkcionalumą naudojantis duomenis paimtus iš aplinkos, kurioje ji randasi. Mobiliosios aplikacijos gali būti skirtinguose kontekstuose su skirtingais duomenimis. Tai sukuria unikalius iššūkius testavimo procese. Kontekstą suprantančios mobiliosios aplikacijos adaptuojasi ir evoliucionuoja naudojant gautą informaciją iš aplinkos. Ši evoliucija gali vykti realiu laiku nenutraukiantis ar sustabdantis mobiliosios aplikacijos operacijų. Mobiliųjų aplikacijų patikimumas priklauso nuo konteksto prisitaikymo valdymo. „Kad užtikrinti aplikacijų operacijų teisingumą reikalingi kontekstui specifiniai testai ir testų apimties kriterijai.“ [MFE12].

3.6 Mobiliųjų įrenginių įvairovė

Šiais laikais yra daug mobiliųjų įrenginių, pagamintų skirtingų tiekėjų, kurie turi skirtingas techninės ir programinės įrangos savybes. Variacijų skaičius tik išauga jei priskaičiuosime, tai kad kiekvienas įrenginys turi modifikuotą mobilią operacinę sistemą. Tiekėjai modifikuoja operacines sistemas, tam kad vartotojui suteiktų geresnę patirti naudojant įrenginį, arba tam kad padidint įrenginio funkcionalumą. Dėl šių variacijų mobiliosios aplikacijos gali veikti ir elgtis skirtingai. Mobiliųjų įrenginių įvairovė taip pat gali padidint testavimo proceso kainą ir laiką. Norint ištestuoti visus įrenginius, pirkimo ir išlaikymo mobilių įrenginių kainos būtų milžiniškos. Taip pat išauga testavimo sudėtingumas, dėl reikalingo laiko testavimui. „Testavimo technikos, kurios padidina įrenginių įvairovės apimtį, bet sumažina skaičių ištestuotų mobiliųjų įrenginių, yra optimaliausios.“ [MFE12].

3.7 Vartotojo patirtis

Vartotojo patirtis tai vartotojo nuovoka ir įspūdžiai prieš naudojimą, naudojimosi metu ir baigus naudoti mobiliąją aplikaciją. Dažnai vartotojas įvertina aplikaciją pasitelkus savo naudojimo patirtimi, todėl vartotojo patirtis yra kritinė norintis užtikrini mobiliosios aplikacijos sėkmę. Vartotojo patirties adekvatumas negali būti tiesiogiai ištestuotas dėl subjektyvios kilmės šio proceso, bet galima patikrinti individualių segmentų teisingumą ir nustatyti suderinamumą su gerom praktikom. Vartotojo patirtis tai elementų ir veiklų patvirtinimas suprojektuotame grafinės vartotojo sąsajos srityse, aplikacijos sąveika ir naudojamumas (angl. usability). Grafinės sąsajos suprojektavimas yra įvertinamas remiantis tinkamu ir logišku išsidėstymo naudojimu, navigacija tarp ekranu, grafinių elementų išsidėstymu, tekstu ir jo šriftu.

3.8 Liečiamieji ekranai

Mobilieji įrenginiai naudoja liečiamus ekranus kaip pirminė priemonė sąveikauti su vartotoju. Liečiamieji ekranai įgalina parodyti ir įvesti duomenis kaip individualias reikšmes arba kaip duomenų grupe. Vartotojas sąveikauja su liečiamuoju ekranu prisilietus, prisilietimas gali būti vienas arba sudėtinis. Taip pat yra gestai, į kuriuos įeina skirtingos sekos ir kombinacijos prisilietimų. Gestai suteikia papildomą funkcionalumą mobiliosioms aplikacijoms, o tai sukuria papildomų iššūkių testavimo procese. Liečiamųjų ekranų testai remiasi rodomų duomenų teisingumu bei ekrano reagavimu. Liečiamojo ekrano reagavimas yra laiko tarpas tarp prisilietimo prie ekrano momento ir momento kada prisilietimas yra

atpažintas, kuris sukelia ekrane atnaujinimą. Ekrano reagavimas taip pat priklauso nuo mobiliojo įrenginio resursų.

3.9 Naujos programinės kalbos ir operacinės sistemos

Programavimo kalbos skirtos mobiliosioms aplikacijoms buvo suprojektuotos palaikyti mobilumą, resursų valdymą ir naujas grafines vartotojo sąsajas. Tradicinės testavimo technikos neatsižvelgia į programavimo kalbų veikimo būdą mobiliosiose operacinėse sistemose, todėl jos turi būti pakoreguotos atitinkamai. Kad išanalizuoti kodą yra svarbu žinoti programavimo kalbos specifines savybes ir kaip jos veikia. Mobiliosios operacinės sistemos yra naujos ir tik dalinai patikimos. „Įvairios patikimumo problemos yra pataisomos naujose versijose operacinės sistemos, kurios dažnai yra bet ne visada suderinamos su senesnėmis versijomis.“ [MFE12]. Pavyzdžiui, mobilieji įrenginiai naudojantis Microsoft mobile operacinė sistema Windows Phone 7 nebuvo atnaujinti kai buvo išleista nauja operacinė sistemos versija Windows Phone 8.

4. Mobiliųjų aplikacijų automatinio testavimo įrankių kriterijai

Atsižvelgus į išanalizuotą medžiagą apie mobiliųjų aplikacijų testavimo iššūkius buvo suformuluoti šie kriterijai norint įvertinti testavimo įrankį.

4.1 Palaikomos OS

Labai svarbu, kad įrankis galėtų testuoti keletą skirtingų OS, kad nereikėtų investuoti į skirtingus įrankius, kurie gali kainuoti papildomo laiko, o jei įrankis nėra atviro kodo tai ir pinigų. Pagrindinės mobiliosios OS, kurios užima didžiąja mobiliųjų aplikacijų rinkos dalį yra Andorid, iOS, Windows Mobile bei Blackberry.

4.2 Finansavimas

Dažnai atviro kodo įrankiai naudojimui reikalauja žinių programavime, todėl testavimo specialistų gali būti sudėtingiau surasti arba reikia papildomai investuoti į apmokymus. Taip pat atviro kodo įrankius prižiūri žmonių bendrijos ir vis daugiau žmonių jungiasi prie įrankio tobulinimo, tačiau dėl nestruktūrizuotos organizacijos dažnai naujuose leidimuose senos aptiktos problemos gali paliktos neištaisytos, tokiu atveju galima bandyti vartotojui bandyti išspręsti problemą ir prisidėti prie įrankio kūrimo arba laukti kol kas nors kitas tai padarys. Komerciniai įrankių kaina varijuoja ir kai kurie įrankiai dažnai nėra vieno karto įsigijimas ir juos reikia nuolat atnaujinti, o dėl spartaus mobiliųjų aplikacijų vystymosi įrankius būtina

nuolat naujinti norint neatsilikti ir vykdyti kokybiškus testus. Skirtingai nei nuo atviro kodo, komerciniai testavimo įrankiai dažnai turi patogias vartotojui grafines sąsajas, todėl sumažina, o kai kurie įrankiai visai panaikina poreikį žinoti programavimą.

4.3 Aplikacijų tipas

- Gimtosios aplikacijos (angl. native applications) – „tai aplikacijos sukurtos specifinei platformai naudojantis tos platformos programos kūrimo rinkinį (angl. Source development kit(sdk)), įrankiais ir programavimo kalbomis, kurias dažniausiai suteikia platformos prekiautojas.“ [SUD13] Šios aplikacijos gali maksimaliai išnaudoti visas įrenginio savybes, o tai dažnai praturtina vartotojo patirtį, o taip pat gimtosios aplikacijos gali veikti be interneto ryšio, tačiau šios aplikacijos reikalauja daugiau laiko ir apsieina brangiau, todėl kad kiekvienai skirtingai platformai reikalinga skirtinga aplikacijos versija. Palyginus su web aplikacijomis atnaujinimas gimtųjų aplikacijų yra varginantis ir reikalauja daug pastangų. Gimtosios aplikacijos geriausiai panaudojamos į vartotoją sutelktose aplikacijose, kurios reikalauja aukštos kokybės grafinės sąsajos, tokiuose kaip žaidimai. Gimtųjų aplikacijų testavimas yra skirtas užtikrinti mobiliųjų aplikacijų kokybę skirtinguose platformose. „Testai sutelkti į funkcionalumą ir aplikacijos elgesį (įskaitant įrenginiui specifines funkcijas tokias kaip gestai), serviso kokybę, naudojimo kokybę, apsaugos kokybę ir privatumą.“ [JAY12]
- Mobiliosios web aplikacijos (angl. web applications) serverio aplikacijos parašytos serverio technologijomis tokiomis kaip PHP, Node.js, ASP.NET, kurios gali apdoroti HTML ir būti atvaizduoti juos mobiliojo įrenginio arba atsiustuose naršyklėse. Web aplikacijos yra dažnai suderinamos su skirtingomis naršyklėmis ir šios aplikacijos gali būti atnaujintos sekundžių bėgyje, ir taip pat dėl jų parašymo lengvumo šios aplikacijos yra labai efektyvios laiko ir kainos atžvilgiu. Bet kadangi šios aplikacijos priklauso nuo ryšio jų veikimas gali būti lėtas ir taip pat turi ribotą duomenų saugojimo galimybes, galų gale web aplikacijos turi labai ribotą prieigą prie įrenginio ypatybių, tokių kaip kamera, adresų knygelė ir t.t. Web aplikacijos geriausiai panaudojamos verslui skirtomis aplikacijoms, bendro naudojimo aplikacijoms. Web aplikacijų testavimas siekia užtikrinti mobilių aplikacijų kokybę naudojantis skirtingomis interneto naršyklėmis ant skirtingų mobilių įrenginiu. „Skirtingai nei gimtosios

aplikacijos, web aplikacijos dažniausiai vartotojui suteikia prieigą prie funkcijų, kurias teikia vidinis (angl. back-end) serveris. Todėl priedui prie funkcionalumo, elgesio, serviso kokybės, saugos, bei privatumo, mobiliųjų web aplikacijų testai taip pat reikalauja testų ryšiui.“ [JAY12]

- Hybridinės aplikacijos (angl. Hybrid applications) - gimtųjų ir web aplikacijų derinys, jos yra rašomos bendra programavimo kalba tokia kaip JavaScript, todėl šios aplikacijos veikia ant skirtingų platformų, hibridinės aplikacijos pasileidžia ant įrenginio ir naudojantis naršyklės varikliu gali apdoroti html ir javascript lokaliai. HTML5 palaikymas praturtina aplikaciją bei pagerina naudojimo kokybę. Dėl to kad šioms aplikacijoms nereikia rašyti atskirto kodo, kad veiktu ant skirtingų platformų tai sutaupo laiko ir išteklių, taip pat šios aplikacijos turi nuo ryšio nepriklausomą duomenų saugojimą. Hybridinės aplikacijos yra apribotos vietinės naršyklės veiksnumu ir neturi pilno prieinamumo prie techninės įrangos, tokios kaip kamera, akselerometras, bluetooth. „Šios aplikacijos geriausiai panaudojamos į vartotoją sutelktuose aplikacijose su vidutinės kokybes grafine sąsaja, o taip pat verslo aplikacijose, kurios reikalauja prieigos prie mobiliojo įrenginio ypatybių.“ [JAY12]

Testavimo įrankis idealiai turėtų susitvarkyti su visais trimis tipais aplikacijų. Taip pat, įrankis turėtų suprasti niuansus ir turtingesnį funkcionalumą gimtųjų aplikacijų, tam kad būtų galima ištestuoti jas atitinkamai

4.4 Integracija

Modernios programinės įrangos vystymosi ciklas priklauso nuo daugelio įrankių. Ypatingai aktualu judriose (angl .agile) komandose, kur rolės dažnai sutampa, tai yra programuotojai testuoja tam tikrus testavimo atvejus, o testavimo specialistai kartais prisideda prie programavimo. Žmonės visada nori dirbti su įrankiais, kurie yra patogus naudoti. Programuotojai nenori palikti savo programavimo aplinkos, kad atlikti testus, tas pats galioja testavimo specialistams, kurie nori dirbti taip pat su jiems pažystamais įrankiais. Vis populiaresnis ir dažniau pasitaikantis naudojamas judrus procesas bei nuolatinis apsikeitimas programinė įranga reikalauja, kad testavimo įrankiai turėtų galimybę integruotis su programavimo įrankiais. Ši integracija leidžia pasiekti aukšto lygio produktyvumą ir padeda paspartinti programinės įrangos pristatymą kitai komandai, išlaikant tos komandos įrankių pasirinkimą.

Priedui prie integracijos su programavimo įrankiais, testavimo įrankis taip pat turi turėti galimybę integruotis su testavimo valdymo įrankiais arba projekto valdymo įrankiais. Integracija su testavimo valdymo įrankiu leidžia prižiūrėti testavimo procesą, organizuoti automatinius arba rankinius testus, gauti išsamius klaidų pranešimus ir pan. Integracija su projekto valdymo įrankiu leidžia visoms suinteresuotoms šalims dalintis informaciją, peržiūrėti vienas kito darbą, priskirti užduotis, matyti rezultatus ir suprasti bendra projekto būseną. Šios savybės leidžia nustatyti kur reikia papildomų pastangų ir matyti ar projektas bus įvykdytas numatytų laiku. Papildoma sritis integracijos gali būti tarp testavimo ir stebėjimo įrankiais. Pavyzdžiui, jei stebėjimo duomenis parodo, kad tam tikrą OS versija nuolatos lūžinėja ir dauguma vartotojų teikia jai prioritetą, tuomet programavimo ir testavimo komandos žino, kad sekančioje projekto iteracijoje ši problema turi būti dėmesio centre. Panašiai, jeigu programavimo ir testavimo komandos žino, kad tam tikras įrankis turėjo problemų testavimo srityje, tuomet komandos gali stebėti iki kokio lygio šios problemos paveikia vartotojus patikrinus stebėjimo duomenis.

4.5 Skripto kalba

Renkantis įrankį būtina žinoti kuo tikėtis ir ką reik mokėti, testavimo įrankiai naudoja skirtingas skripto kalbas, tokias kaip javascript, vb, kai kurie įrankiai savo unikalias kalbas būdingas tik tam įrankiui, pvz. Sensetalk naudojama eggPlant įrankio.

4.6 Jailbreaking

Jailbreak(iOS) arba ‘rooting(‘Android’) tai procesas kai įsilaužiama į mobilųjį įrenginį, kad instaliuoti bei vykdyti aplikacijas, kurios yra uždraustos priešingu atveju operacinės sistemos arba įrenginio prekiautoju. „Įrankiai, kurie naudoja jailbroken įrenginius gali nepalaikyti naujausių operacinių sistemų versijų ir taip gali būti nesuderinami su mobiliosiom įrenginio tvarkymo (angl. Mobile Device Management(MDM)) taisyklėmis, reikalingomis verslo aplikacijoms.“ [PRA11]

4.7 Testavimo būdas

Visi šie budai buvo sudaryti atsižvelgiantis į mobiliojo testavimo iššūkius, bet nei vienas iš jų negali būti universalus, todėl norint pasiekti geriausių testavimo rezultatų svarbu optimizuoti bei subalansuoti testavimą naudojantis skirtingais testavimo būdais. Yra iš viso keturi testavimo būdai :

- Testavimas emuliatoriais - Mobiliojo įrenginio testavimas emuliatoriumi dar kitaip vadinamu įrenginio simuliatoriumi tai testavimas kai yra sukuriama mobiliojo įrenginio virtualios mašinos versija ant personalinio kompiuterio. Emuliatorius yra dažnai įtrauktas į mobilios platformos programinės įrangos kūrimo komplektą (pvz. Android SDK). Tai yra gana pigus sprendimas, nes nereikalauja turėti testavimo laboratorijos ir nereikia pirkti arba nuomoti tikro mobilaus įrenginio, tačiau šiuo būdu galima ištestuoti tik apribotą kiekį sistemos funkcionalumo. Nepaisant to kad šis testavimo būdas yra pigus, jis turi kelis apribojimus – pvz. šis būdas turi sunkumų patvirtinant visas gestų rūšys, todėl kad dauguma emuliatorių palaiko gana ribotą gestų ir įrenginiui specifinių funkcijų. Tai pat šis būdas yra apribotas testuojant serviso kokybę. Kad įveikti šias problemas, galima sukurti šių testų simuliatorių, kuris atkartotu mobilaus kliento operacijas ir galėtų palaikyti daugiau nei viena mobilųjį klientą. Bet vis dėl to net ir šis būdas turi apribojimų tvirtinant įrenginiui specifinius mobilaus serviso funkcijas. „Taip pat šis būdas nepalaiko skirtingų mobiliųjų įrenginių bei platformų ar naršyklių, todėl kad emuliatoriai paprastai yra skirti specifiniam įrenginiui arba platformai.“[GAO14]
- Testavimas realiu mobiliuoju įrenginiu - Testavimas naudojantis tikru mobiliuoju įrenginiu reikalauja įkurti testavimo laboratorija ir nusipirkti tikrus mobiliuosius įrenginius, o tai yra daug brangiau nei testavimo būdas naudojantis emuliatoriumi, tačiau šiuo būdu galima ištestuoti visus įrenginiui specifines funkcijas, elgesį ir serviso kokybės parametrus. „Didžiausias šio būdo iššūkis tai nesugebėjimas automatizuoti testų, o tai smarkiai apriboja galimybės kovojant su sparčiais pasikeitimais mobiliųjų įrenginių ir jų platformų. Taip pat šis būdas yra apribotas testuojant serviso kokybę, todėl kad didelės skalės testai reikalauja daug mobilių įrenginių, o tai dažniausiai nėra įmanoma įmonėmis.“[GAO14]
- Cloud testatimas - Testavimas naudojantis cloud yra dažnai palaikomas testavimo paslaugų prekiautojais. Šio testavimo būdo principas yra įrengti mobilių įrenginių cloud, kuris galėtų palaikyti didelės skalės testavimą. „Palyginus su kitais testavimo būdais, šis būdas yra daug pigesnis nei realaus mobiliojo įrenginio testavimas didelėje skalėje ir taip pat daug efektyvesnis testuojant įvairias skirtingas veiklas ant mobiliųjų įrenginių.“[GAO14]

- Minios testavimas - tai testavimas pasamdžius grupė testavimo inžinierių arba bendruomenę potencialių vartotojų (pvz. uTest). Šiuo metu šios paslaugos prekiautojai palaiko primityvų testo valdymą, testavimo servisą ir klaidų pranešimus. Dauguma mobiliųjų testavimo operacijų yra valdomos dažnai improvizuotai ir su labai apribotu testavimo įrankiu naudojimu. Šis būdas siūlo testavimą realiose „laukinėse“ sąlygose (angl. in the wild), be poreikio investuoti į testavimo laboratorija arba pirkimo mobilių įrenginių, tačiau rizikuojama gauti žemos kokybės rezultatus, taip pat šis būdas dažnai negarantuoja testus būti atliktus pagal numatytą tvarkaraštį.

Ieškokite įrankių, kurie duoda galimybę naudoti emuliatorių vieniem testams, pavyzdžiui kodo testavimui, baziniam funkcionalumui, o kitiem testams realius įrenginius, ypatingai tiems testams, kurie reikalauja aukšto lygio tikslumo, pavyzdžiui testai su nepalankiomis sąlygomis kaip beveik išsekusi baterija, nepastovus ryšis.

4.8 Atpažinimo būdas

Vartotojo sąsajos elementų atpažinimo technika naudojama skirtingose testų automatizavimo įrankiuose yra svarbus veiksnys renkantis tinkamiausią įrankį testuojantis mobiliąsias aplikacijas, atpažinimo būdai yra šie :

- Tikrąjį objektų atpažinimas naudojantis įrankiai atžymi elementus esančius ant mobiliojo įrenginio ekrano ir pakeičia juos į objektus, kurios įrankis gali manipuliuoti. Šis testavimo būdas nepriklauso nuo ekrano dydžio ar rezoliucijos ir taip pat skriptai gali buti pakartotinai naudojami skirtingiems testams. Ši savybė yra ypatingai svarbi Android įrenginiams, kur yra daug skirtingų ekrano dydžių ir rezoliucijų. Kai kurie įrankiai naudojantis šią technologija reikalauja, kad būtų atlikti pakeitimai aplikacijos kode, o kiti ne.
- Vaizdais remiantis testavimo įrankiai pasikliauja testavimo įrankio testo įrašymo metu padarytomis ekrano nuotraukomis ir vėliau kuria automatizuotus testavimo skriptus pasinaudojus tomis nuotraukomis ir ekrano elementų koordinatėmis. Pavyzdžiui, mygtuko ant ekrano paspaudimas yra pasiekiamas paspaudus koordinates. Šis būdas yra nepriklausomas nuo aplikacijos tipo ir suteikia naudingus vaizdo atpažinimo pajėgumus, tačiau dažnai reikalauja jailbroken arba rooted įrenginių ir skriptus perrašytus kiekvienam skirtingam įrenginiui su skirtingu ekrano dydžiu arba rezoliucija dėl elementų

koordinačių pasikeitimo. Jailbroken įrenginiai tipiškai nėra suderinami su mobiliojo įrenginio verslo valdymo saugos taisyklėmis, todėl įrankiai, kuriems reikia jailbroken įrenginių gali būti netinkami testuojant verslo aplikacijas.

- Optinis simbolių atpažinimas arba (angl. Optical character recognition((OCR)) tai atpažinimo būdas, kai iš aplikacijos grafinės sąsajos yra surenkamas tekstas ir gautos reikšmės yra naudojamos rašant testavimo skriptus. Šis būdas gali susidoroti su skirtingomis ekrano rezoliucijomis, orientacijomis ir dydžiais. Tačiau šis būdas gali patvirtinti tik teksto elementus matomus ekrane. Jei tekstas pasikeičia arba yra ištrinamas iš aplikacijos, šis vartotojo elementas yra labai sunkiai atpažindamas. Dar vienas šio būdo trukumas yra jo lėtas veikimas.

4.9 Testavimo įrankiai

Palyginimui dėl jų populiarumo ir gerų atsiliepimų buvo atrinkti šie įrankiai ir buvo suvesti į lentelę (lentelė 3) :

Pav. 1 eggPlant logotipas

Eggplant (Pav. 1) (TestPlant) tai testavimo įrankis naudojantis vaizdo atpažinimo technika. Eggplant naudoja skripto kalba ‘SenseTalk’, kuri leidžia profesionalioms testavimo komandoms nesunkiai automatizuoti užduotis paprastai ir greitai.

Pav. 2 Robotium logotipas

Robotium (Pav 2) (Robotium Tech) yra atviro kodo testavimo įrankis naudojamas kurti išsamius ir lanksčius juodos dėžės testus Android gimtosioms aplikacijoms. Su robotium pagalba galima rašyti funkcinius, sisteminius ir prieinamumo testų scenarijus. Kadangi tai atviro kodo programa, kiekvienas norintis gali ją patobulinti.

Pav. 3 Appium logotipas

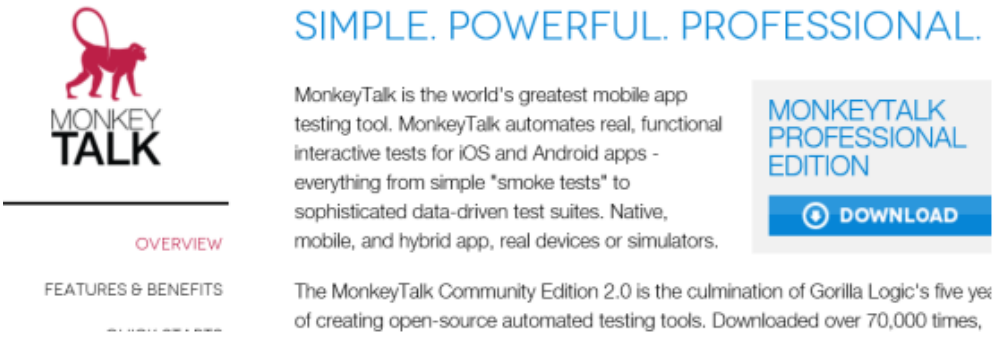
Appium (Pav. 3) (*GitHub project, Sauce Lab*), tai atviro kodo automatizuota mobiliųjų aplikacijų testavimo programa skirta iOS ir Android platformoms. Ši testavimo programa parašyta naudojant iOS ir Android SDK. Appium testavimą atlieka naudojant mobiliame esamą interneto naršyklę, bei WebDrive protokolą. Appium supranta Safari (ji yra iOS operacinėje sistemoje) ir Chrome kuri yra Android operacinėje sistemoje).

Pav. 4 Sikuli logotipas

Sikuli (Pav 4)(Sikuli) testavimo įrankis gali automatizuoti bet ką kas yra matoma ant ekrano. Sikuli naudoja vaizdo atpažinimo techniką atpažinti ir kontroliuoti grafinės vartotojo sąsajos komponentus. Šis įrankis naudingas tokias atvejais kai nėra lengvo priėjimo prie grafinės vartotojo sąsajos išeities kodo.

Pav. 5 Ranorex logotipas

Ranorex (Pav. 5) (Ranorex) – palaiko tik naujausias versijas ir jai paleisti yra nesudėtinga. Tai automatizuota testavimo programa, kuri sukuria vientisą testavimo veikimą darbastalio, žiniatinklio ir mobilių aplikacijų. Dėl RanoreXPath technologijos ji turi glaudžia sąsaja su vartotoju ir patogia vartotojui aplinką



Pav. 6 MonkeyTalk logotipas

MonkeyTalk (Pav. 6)(GorillaLogic) - tai yra automatinis realaus laiko interaktyvus testavimo įrankis, kuris atlieka testavimą nuo paprastų užduotėlių iki sudėtingų duomenų tikrinimų. Tai sudėtinga testavimo programa, kuri susideda iš trijų pagrindinių vienetų: MonkeyTalk IDE, MonkeyTalk Agent ir Monkey Script.

lentelė 3 Testavimo įrankių palyginimas

Testavimo įrankis	Palaikomos OS	Finansavimas	Aplikacijų tipas	Skripto kalba	Jail breaking	Integracijos galimybės	Indentifikacijos budas	Testavimo būdas
Robotium	Android	Atviro kodo	Native, Hybrid, Web	Java	Nereikalingas	Jenkins, Eclipse	Tikrasis	Emulatorius, Realus įrenginiai, Cloud
EggPlant	Android, iOS, Blackberry, Windows mobile	Licencijuojama	Native, Hybrid, Web	Sensetalk	Nereikalingas	QC Visual Studio, Jenkins	Vaizdo, OCR	Emuliatoriai, Realus įrenginiai
Ranorex	Android, iOS, Windows mobile	Licencijuojama	Native, Hybrid, Web	C#, VB	Nereikalingas	Jenkins, QC, Visual Studio	Tikrasis	Emuliatoriai, Realus įrenginiai
Monkey talk	Android, iOS	Licencijuojama	Native, Hybrid, Web	MonkeyTalk, Javascript	Nereikalingas	Jenkins, Eclipse	Tikrasis, OCR	Emuliatoriai, Realus įrenginiai
Appium	iOS, Android, Blackmerry	Atviro kodo	Native, Hybrid, Web	Java, C#, Ruby, Python, Perl, PHP , Javascript	Nereikalingas	Jenkins, Eclipse	Tikrasis	Emuliatoriai, realus įrenginiai, Cloud
Sikuli	Android, iOS	Atviro kodo	Native	Jython	Reikalingas	Jenkins, Eclipse	Vaizdo, OCR	Emulatoriai

5. Testavimo įrankių įvertinimas

Šiam išsamesniam įvertinimui buvo parinkti trys testavimo įrankiai eggPlant, Appium bei MonkeyTalk, pasinaudojus pirminiu palyginimu (lentelė 3) buvo atmestas Sikuli dėl to, kad jis reikalauja Jailbreakingo testuojant aplikacijas, o Robotium dėl jo panašumo į Appium, tik apribotam Android aplikacijoms. Verta paminėti Ranorex, kuris galėtų būti potencialus kandidatas į geriausią testavimo įrankį, tačiau nebuvo galimybės jo patikrinti dėl licencijos. Vertinime buvo atsižvelgiama į testavimo skriptų galimybes ir ypatybes, objektų atpažinimo metodą, bei testavimo rezultatų pateikimą :

5.1 EggPlant testavimo įrankio vertinimas

Išeities kodo pakeitimas nereikalingas, todėl kad eggPlant dirba tiesiogiai su grafinę vartotojo sąsaja. Kad dirbti su realiais įrenginiais tereikia turėti papildoma VNC programą ant abiejų įrenginių, kompiuterio ir mobilaus renginio, su kurios pagalba galima būtų komunikuoti tarpusavyje.

Su eggPlant yra du būdai kurti testavimo skriptus. Pirmas būdas yra prijungti savo įrenginį arba emuliatorių prie IDE ir pradėti įrašinėti savo veiksmus. EggPlant įrašo vartotojo veiksmus ir išsaugo paspaustus vaizdus bei veiksmus, kurie atliekami ant to vaizdo. Šiame kontekste vaizdas tai yra įvairūs elementai randantis grafinei vartotojo sąsajoje. Kitas būdas kurti skriptus yra savarankiškai ir vėliau juos vykdyti ant įrenginio arba emuliatoriaus.

EggPlant naudoja programavimo kalbą SenseTalk. Tai nesunkiai išmokstama paprasta skriptingo kalba. Ši kalba neturi tokių plačių galimybių kaip JavaScript ar Python, tačiau ji su užduotimis susitvarko efektyviai pasitelkus į pagalbą TestPlant vaizdo atpažinimo technologiją.

EggPlant testavimo įrankis neturi objekto atpažinimo. Šis įrankis naudoja vaizdo atpažinimą ir OCR, kuris atliekamas grafinio vartotojo sąsajoje, tai yra aukščiausiame aplikacijos lygyje. Šis įrankis dirba su padarytomis ekrano nuotraukomis, o ne objekto parametrais. Dėl to šis įrankis negali dirbti su paslėptais objektais, tačiau jis gali susidoroti su sudėtingomis grafikos problemomis, kur kiti automatizavimo įrankiai negali įvykdyti testo sėkmingai.

Duomenų valdomi testai yra galimi naudojantis įvairius sąrašo tipo failus, tai yra CSV, txt, ar XML failai, kurie suteikia duomenų rinkinį testavimo skriptams ir tada skriptai yra vykdomi kelis kartus su duotais duomenimis

Testavimo rezultatai yra pateikiami ataskaitoje. Ataskaita tai atskiras langas, kuris pateikia informacija apie testavimo skripto vykdymą. Testavimo atvejis, testavimo žingsniai, vykdymo laikas, klaidos ir ekrano nuotraukos yra pateikiami ataskaitoje. Peržvelgus rezultatus ataskaitoje, ją galima eksportuoti kaip registrų failą arba HTML dokumentą.

5.2 eggPlant privalumai

Naudotis eggPlant nereikia turėti ypatingų programavimo žinių, tereikia gerai pažinti aplikaciją ir testavimo atvejus, kad galima būtų panaudoti vaizdinius failus įtraukiantis juos į Sensetalk skriptus, taip pat neprivaloma žinoti aplikacijos išeities kodo ar programavimo kalbos, kuria buvo parašyta aplikacija. Taipogi SenseTalk tai lengvai nuskaitoma, nesunkiai išmokstama skripto kalba, kuri leidžia nesunkiai kurti testavimo skriptus.

5.3 eggPlant trūkumai

Nėra tikrojo objektų atpažinimo. Jei aplikacija naudoja paslėptus elementus ar struktūras, tai ši aplikacija negali būti ištestuota su eggPlant. Kitas trūkumas aplikacijos yra kiek ir privalumas tiek ir trūkumas tai SenseTalk, kuri yra gana siaura kalba ir kai kurių naujų savybių gali neturėti.

5.4 eggPlant rekomenadicijos

SenseTalk yra patogi ir suprantama skripto kalba, tačiau dažnai programuotojai norintis atlikti testus mieliau pasirinks jiems pažįstama kalba, todėl svarbu palaikyti daugiau skripto kalbų, ypatingai JavaScript. Neturėjimas galimybės pasinaudoti tikruoju objektų atpažinimo metodu suvaržo šio testavimo įrankio galimybes, gali iškilti atveju kai testai yra nepakankamai tikslus, taip pat vaizdo atpažinimas yra lėtesnis, todėl svarbu kad įrankis galėtų palaikyti tikrąjį objektų atpažinimą.

5.5 Appium testavimo įrankio vertinimas

Testuojant su Appium ant iOS įrenginių yra reikalingas modifikuotas išeities kodas, tai yra Apple programuotojo ID ir pakonfigūruotas sertifikatas, kuris turi buti įtrauktas į iOS aplikacijos kodą ir sukompiliuotas kartu su šia informacija. Testuojant and iOS simuliatoriaus, Android emuliatoriaus arba Android įrenginio kodo pakeitimas nereikalingas.

Testavimo skriptai gali būti sukurti dviem skirtingais būdais. Pirmas būdas tai maža programėlė appium.app turinti grafine vartotojo sąsaja, kuria pasinaudojus galima įrašyti

veiksmus atliktus aplikacijoje ir vėliaus juos perkelti į pasirinktą programavimo kalbą. Šiuos skriptus galima modifikuoti ir pernešti atgal į appium.app ir paleisti juos ant emulatoriaus arba mobilaus įrenginio. Kitas variantas tai parašyti testavimo atvejus savarankiškai pasinaudojus kodo redaktoriumi. Pabaigus rašyti skriptą jis yra pernešamas į appium.app ir paleidžiamas. Taip pat skriptai gali būti paleisti tiesiogiai iš terminalo nepasinaudojus grafine vartotojo sąsaja.

Visos žinomos programinės kalbos ir struktūros gali būti panaudotos rašantis appium skriptą. Pats Appium tai yra http serveris, kuris gali sukurti ir valdyti Selenium Webdriver sesijas. Selenium WebDriver tai yra HTTP API, kuris yra implementuotas į programinės kalbos kodą ir naudojamas kaip standartas naršyklių automatizavimui. Naudojantis šia technologija galima rašyti testus daugeliu kalbų, įskaitant Java, JavaScript, Python, Ruby, XCode ir kt. Kitas būdas yra testavimo skriptus įrašytus naudojantis appium.app eksportuoti į pasirinktą kalbą.

Appium naudoja tikrąjį objektų atpažinimo metodą. Objektai gali būti atpažinti ir patikrinti kelias skirtingas budais : pagal prieinamumo lygį, pagal elemento tipą, pagal hierarchiją arba pagal Android ID. Šie keturi metodai yra labai patogus ir yra optimalus sprendimas. Testuojantis skirtingas platformas yra svarbu, kad informacija apie tą patį objektą būtų identiška naudojantis Android arba iOS. Priešingu atveju reikės kurti du skirtingus skriptus kiekvienai platformai tam pačiam atvejui testuoti.

Kaip minėjau anksčiau Appium gali būti interpretuojamas kaip HTTP serveris, kuris gauna testavimo skriptus ir duomenis iš tam tikros programavimo kalbos ir tuomet gautus duomenis jis įvykdo testuojamame įrenginyje. Todėl duomenų valdomi testai yra įmanomi jeigu tai leidžia naudojama programavimo kalba

Appium rezultatus galima peržiūrėti tik derinimo registruose (angl. debugging logs) arba klaidų pranešimuose terminale. Jeigu appium.app yra panaudojamas testavimui vykdyti, klaida pasirodys programos grafinėje sąsajoje. SauceLabs siūlo galimybę susikurti paskyrą jų puslapyje ir prisijungti prie Appium projektų pasinaudojus šia paskyra. Jeigu testai yra vykdomi prisijungus, tuomet bus sukurtas registro failas SauceLabs paskyroje, kurioje galima rasti testavimo duomenis, žingsnius, ekrano nuotraukas ir klaidų pranešimus.

5.6 Appium privalumai

Appium didžiausias privalumas yra tai, kad jis suteikia galimybę testuoti tokią pačią aplikaciją, kuria vartotojas instaliuoja į įrenginį. Jokio išeities kodo pakeitimų atlikti testavimams nereikia.

5.7 Appium trūkumai

Appium didelis trūkumas yra tai, kad negalima testavimo skriptus vienu metu paleisti ant kelių įrenginių. Tai reiškia, kad visi skirtingi įrenginiai turi būti ištestuoti vienas po kito. Tai užima daug laiko ir jei naudoti įmonėje kainuoja pinigų ir resursų.

5.8 Appium rekomendacijos

Turėti galimybę testuoti skriptus keliuose įrenginiuose iš karto, praplėsti įrankį su OCR bei vaizdo atpažinimo technologijomis, praturtinti dokumentaciją. Taip pat pertvarkyti appium.app dizainą, padaryti jį patogesnį naujiems vartotojams, taikytis ne tik į aukštą funkcionalumo lygį bet ir į programos paprastumą.

5.9 MonkeyTalk testavimo įrankio vertinimas

Išeities kodas, kad paleisti testavimo skriptus nėra reikalingas, tačiau reikalingas pakeitimas išeities kode, kad aplikacija galėtų bendrauti su MonkeyTalk. Monkeytalk naudoja savo IDE, kuris yra paremtas Eclipse, prieš pradedant testavimą yra svarbu instaliuoti MonkeyTalk Agent į aplikaciją. Šis agentas reikalingas tam kad aplikacija galėtų bendrauti su MonkeyTalk IDE, tai galioja naudojantis ir emuliatoriumi ir tikru įrenginiu.

Testavimo skriptai yra kuriami dviem būdais. Pirmas būdas yra pasinaudojus įrašymo funkcija arba galima parašyti kodą pasinaudojus redaktoriumi ir išsaugoti šį failą su praplėtimu .mt. Šis testavimo skriptas yra įkeliamas į MonkeTalk IDE ir tuomet paleidžiamas. Įrašytas skriptas gali būti modifikuotas pasinaudojus IDE arba bet kokiu kitų teksto redaktoriumi ir vėliau įkeltas į IDE vykdymui.

MonkeyTalk naudoja MonkeyTalk programavimo kalba tai nesudėtinga funkcinė testavimui skirta kalba, kuri susideda iš paprasto rinkinio komandų skirtų vartotojo sąsajos skriptingui. Kad pasinaudoti šia kalba ypatingų programvimo žinių nereikia ,bet prireikus MonkeyTalk galima praplėsti JavaScript. Skriptai gali būti kuriami kaip paprasti teksto failai, arba automatiškai pasinaudojant įrašymo savybe.

Verta paminėti, kad MonkeyTalk agent tai yra nesudėtingas API, kurį su šiom tokiom pastangomis galima implementuoti į bet kuria standartinę programavimo kalbą kaip Java, C, C# ir kt.

MonkeyTalk kaip ir Appium turi panašius objekto atpažinimo metodus. Trys skirtingi budai skirti iOS ir keturi Android : pagal prieinamumo lygį, pagal elemento tipą, pagal hierarchiją, pagal Android ID. Kaip ir Appium objekto ypatybės turi buti vienodos abiejuose operacinėse sistemose, antraip nebus galima realizuoti testų ant skirtingų platformų.

Duomenų valdomi testai yra įmanomi panaudojantis CSV failus. Tas pats skriptas su skirtingomis reikšmėmis kiekvienoje iteracijoje vykdomas MonkeyTalk. Taip pat yra galimybė pakoreguoti numatomą rezultatą kiekvienam testavimo skripto vykdymui. Pasinaudojus šia savybe tą patį skriptą galima leisti tam pačiam testavimo scenarijui su skirtingais numatomais rezultatais, tai sutaupo laiką, nes nereikia kurti papildomų skriptų.

Rezultatus galima peržvelgti kaip HTML failą su ekrano nuotraukomis ir testavimo žingsniais paimtais iš testavimo registrų, arba galima sukurti xml failą ir importuoti į kokį nors xml peržiūros įrankį.

5.10 MonkeyTalk privalumai

MonkeyTalk privalumas yra tai, kad jis turi savo IDE kartu su testų įrašymo galimybėmis. Kita naudinga savybė yra tai kad MonkeyTalk kalbą galima papildyti naujomis komandomis.

5.11 MonkeyTalk trūkumai

MonkeyTalk trukūmas yra tai, kad šis įrankis silpnai palaiko HTML5 web aplikacijas, o taip pat jei aplikacija turi įdėtus į ją internetinius puslapius jie negali būti ištestuoti. Kitas trūkumas yra tai kad testus vykdyti iOS platformoje reikalingas Wi-Fi.

5.12 MonkeyTalk rekomendacijos

Praplėsti palaikomų operacinių sistemų palaikymą tokiomis operacinėmis sistemomis kaip Blackberry, Windows Phone. Patobulinti web elementų palaikymą, o taip pat rasti alternatyvų sprendimą MonkeyTalk Agent, kuris turi būti suinstaliuotas į aplikaciją, norint ją testuoti su MonkeyTalk, o tai nėra gera praktika. Ir taip pat turėti galimybę testuoti iOS pasinaudojus USB jungtimi.

Išvados

Taigi išanalizavus literatūrą, sudarius kriterijus ir įvertinus testavimo įrankius buvo buvo gauti šie rezultatai :

1. Identifikuota mobiliųjų aplikacijų testavimo ypatumai, nustatyta kad testavimą apsunkiną didelė įvairovė mobilių įrenginių ir jų naudojamos operacinės sistemos
2. Pasiūlytas kriterijų rinkinys leidžiantis įvertinti automatinius mobiliųjų testavimo įrankius.
3. Atliktas trijų testavimo įrankių išsamus vertinimas atsižvelgiant į testavimo skripto ypatybes bei objekto atpažinimo galimybes.
4. Pasiūlytos rekomendacijos testavimo įrankių patobulinimui.

Išnagrinėjus visus iškeltus klausimus tampa aišku jog mobilusis testavimas daug ko skiriasi nuo darbalaukio aplikacijų testavimo, norint nustatyti automatiniams mobiliųjų aplikacijų testavimo įrankiam kriterijų rinkinį yra svarbu pažinti mobiliųjų aplikacijų testavimo ypatybes bei problemas. Ypatingai verta atkreipti dėmesį į palaikomų operacinių sistemų kriterijų, dėl to kad norint turėti sėkmę mobiliųjų aplikacijų versle svarbu užtikrini aplikacijos veikimą populiariausiuose platformose, kitas išskirtinis kriterijus yra objektų atpažinimo būdas, kuris lemia testavimo įrankio tikslumą, našumą bei greitį, norint pasiekti geriausių rezultatų yra svarbu, kad įrankis palaikytų visus objekto atpažinimo būdus, o ypatingai tikrąjį objektų atpažinimo būdą, kuris yra sparčiausias ir tiksliausias, vienintelis jo trūkumas yra tai kad jis negali aptikti grafinės sąsajos defektų.

Iš šešių ištirtų testavimo įrankių geriausias yra Appium ir eggPlant. Pirmasis yra dėl to jog visapusiškai ištestuoja aplikacijas ir naudoja tikrąjį objektų atpažinimą, o pastarasis dėl labai patogios vartotojo sąsajos, nepaisant to kad eggPlant neturi tikrojo objekto atpažinimo, jis turi labai išdirbtą vaizdo atpažinimo technologiją, taip pat dėl grafinės įrankio sąsajos intuityvumo, bei skripto kalbos SenseTalk paprastumo įrankis yra greitai įsisavinamas, o tai leidžia kurti testus be jokių programavimo žinių.

Taigi atsižvelgiant į testavimo poreikius ir kvalifikacija yra svarbu parinkti tinkamą testavimo įrankį norint testus vykdyti efektyviai ir pelningai.

Literatūra

[KAK14] KAKADE, Harshal. SELECTING THE RIGHT MOBILE TEST AUTOMATION STRATEGY: CHALLENGES AND PRINCIPLES1 2014

[GAO14] GAO, Jerry, et al. Mobile application testing: a tutorial. Computer, 2014, 2: 46-55.

[MFE12] MUCCINI, Henry; FRANCESCO, Antonio Di; ESPOSITO, Patrizio. Software testing of mobile applications: Challenges and future research directions. In: Automation of Software Test (AST), 2012 7th International Workshop on. IEEE, 2012. p. 29-35.

[PRA11] PRADHAN, Tushar. Mobile Application Testing. TCS White papers,[online]. Available: http://www. tcs. com/SiteCollectionDocuments/White% 20Paper s/Mobility_Whitepaper_Mobile-Application-Testing_1012-1.pdf, 2011.

[SUD13] SUDHEER M, Practice Head - Mobility Testing and Automation, 2013

[KAU15] KAUR, Anureet. Review of Mobile Applications Testing with Automated Techniques. interface, 2015, 4.10.

[JAY12] JAYAMALRAO, Gopinath. Mobile-Testing-QAAC, 2012

[CAP13] Capgemini and Sogeti, The Fine Art of Mobile Testing, 2013

[UTE] uTest, The Essential Guide to Mobile App Testing

[MUL12] Mulikita, Melisa D., Mobile Application Testing, 2012

[HP09] HP, Functional Testing For Mobile Applications, 2009

[LG15] Feroz Pearl Louis,Gaurav Gupta , Mastering Mobile Test Automation, 2015

[1] http://www.thewindowsclub.com/idc-windows-phone-will-be-number-2-operating-system-worldwide-by-2015