TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond
Arvutitehnika instituut

ITI40LT

Laine Viisitamm 120406

SESSIOONIPÕHISE JA UURIVA TESTIMISE TESTIHALDUSVAHENDITE ANALÜÜS

Bakalaureusetöö

Juhendaja: Maili Markvardt
MSc
Lektor

Tallinn 2016

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: Laine Viisitamm

25.05.2016

Annotatsioon

Antud bakalaureusetöö on kirjutatud teemal "Sessioonipõhise ja uuriva testimise testihaldusvahendite analüüs".

Teema valik tekkis töö autoril olles ise testija tarkvara testimise teenust pakkuvas ettevõttes. Probleem ilmnes sessioonipõhise testimise juurutamisega ettevõtte siseselt, sest traditsiooniline testihaldusvahend ei sobinud uue testimise meetodi kasutamiseks.

Lõputöö eesmärgiks on leida nõuetele vastav sessioonipõhise ja uuriva testimise testihaldusvahend. Töös esitatakse kriteeriumid sobiliku vahendi valikuks, katsetatakse valitud vahendeid ning antakse soovitused sobilikuma vahendi kasutamiseks.

Lõputöö on kirjutatud eesti keeles ning sisaldab teksti 33 leheküljel, 5 peatükki, 7 joonist.

Abstract

Analysis of Session Based and Exploratory Test Management Tools

This bachelor thesis is written on the subject „Analysis of Session Based and Exploratory Test Management Tools“.

Subject selection emerged to the author while working in small software testing company FOB Solutions. The problem occurred when company decided to change testing method from traditional scripted testing to session based exploratory testing. While introducing new method to testers, it became aware that new method doesn’t meet the same test management tool criteria as it was for traditional test management tool.

The goal of this thesis is to find tool which would meet the criteria for session based exploratory testing. Thesis author used literary sources, published articles and experience in software testing for defining traditional testing, session based exploratory testing and also to define traditional test management tools. Taking that knowledge into an account, author defined problematic areas between testing methods and also why traditional test management tool doesn’t suit for session based exploratory testing.

Author made research based on criteria for new tool and soon it became aware that there aren’t many tools available to use. Three tools which were selected in this thesis are analyzed and compared against practical experience and theoretical requirements. Outcome from research revealed that only one of them is contemporary, but has problems to fill the criteria. Other tools analyzed met with most of the requirements, but were lacking in updates and support for the tool.

Recommendations which tool to use in which testing environment are made and also author is suggesting to continue this bachelor thesis further to develop tool for session based test management.

The thesis is in Estonian and contains 33 pages of text, 5 chapters, 7 figures.

Lühendite ja mõistete sõnastik

ad hoc testimine	planeerimiseta ja dokumentatsioonita testimine
scripted testing	traditsiooniline testimine kasutades testijuhtumeid
QA	*Quality Assurance*, kvaliteedi tagamine
test case	testijuhtum, tarkvara nõuetest samm sammult kirjutatud test
touring	heuristiline testimine ehk kus testija õpib rakendust, mis omakorda viib uute test ideedeni
manuaalne testimine	testijuhtumite läbi viimine käsitsi ehk inimese poolt
automatiseeritud testimine	testijuhtumid kirjeldatakse programmina inimese poolt ja neid viib läbi arvuti
FAQ	*Frequently Asked Questions*, korduma kippuvad küsimused, mis aitavad kasutajat küsimuste korral
avatud lähtekoodiga tarkvara	kasutajatele ja arendajatel vabalt kättesaadav tarkvara nii muutmiseks kui ka kasutamiseks

Sisukord

Jooniste loetelu

1 Sissejuhatus

Tänapäeva kiiresti arenevas tarkvaraarenduse maailmas on üha olulisemaks muutunud tarkvara testimine. Tarkvara testimist kummitavad erinevad müüdid, kuidas testimine on näiteks liiga kulukas või liiga ajanõudlik ja sellega nimekiri ei lõpe [1]. Nendest hoolimata on aga igal tõsiseltvõetaval tarkvaraarendajal *QA* meeskond, kes hoolitseb, et see, mida tehakse, vastab nõuetele. Nii nagu on erinevad tarkvaraarendus viise, on ka palju erinevaid viise, kuidas tarkvara testida, millel igaühel on omakorda mingi konkreetne väljund.

Antud uurimustöö eesmärk on võrrelda traditsioonilist testimist uuriva testimisega ja leida testihaldusvahend, millega saaks hallata uurivast testimisest tulenevaid sessioone. Traditsioonilise testimise ehk *scripted* testimise puhul on välja kujunenud kindel metoodika, kuidas antud testimist läbi viia ja hallata. Sessioonipõhise testimise puhul on olemas küll metoodika, aga puudub hea ja hinnatud testihaldusvahend. Eesmärgiks on leida tööriist, mis kataks ära kõik olulised aspektid, et tarkvara testimine, kui selline, ei kannataks, sest testmeetod on teine.

Käesoleva töö eesmärkide saavutamiseks on eelnevalt defineeritud teoreetilised alused nii testimeetodite puhul kui ka testihaldusvahendite puhul. Kaardistamisel on tuginetud kirjanduslikele allikatele ja kogemustele, mis käsitlevad tarkvara testimise meetodeid ja testihaldusvahendeid. Samuti on kasutatud interneti lehekülgedel avaldatud artikleid toetamaks praktilisi kogemusi tarkvara testimises.

Vajaliku tööriista leidmiseks kasutab töö autor peale nõuete esitamise ka katsetusi erinevates keskkondades, et hiljem igat tööriista põhjalikult analüüsida. Tööriistade leidmiseks kasutab autor erinevaid otsingumootoreid ja viiteid tööriistadele erinevatelt teoreetilistelt kirjandusallikatelt.

Töö sisu hakkab peatükist 2, kus on vaatluse all töö teoreetilised osad. Teoreetilistes osades tutvustatakse testmeetodeid ja testihaldusvahendeid. Samuti pannakse paika, millised probleemid tekivad ning millised on nõuded vajalikule tööriistale. Peatükis 3 on

välja toodud tööriistad ning põhjalik analüüs tööriistadele esitatud nõuete kaetuse kohta. Peatükis 4 annab töö autor soovitusi tööriista kasutusele võtuks arvestades ka töö iseloomu.

2 Testimise meetodid ja testihaldusvahendid

Tarkvara testimine on muutunud tänapäeva maailmas üheks lahutamatuks osaks tarkvaraarenduse käigus. Testimise meetodeid on erinevaid, mis kõik täidavad erinevaid eesmärke. Sellest hoolimata on tarkvara testimise eesmärk ikkagi verifitseerida, et tarkvara töötab nii nagu see oli disainitud ja loomulikult ka avastamaks potentsiaalseid riske [2] . Kaks laialdaselt levinut testimise meetodit on traditsiooniline testimine ehk *scripted testing* ja uuriv testimine.

Suurt rolli tarkvara testimise puhul mängivad testihaldusvahendid, mida kasutatakse dokumenteerimiseks. Dokumenteerimise all mõistame testijuhtumite, testi tulemuste ja üleüldise töökäigu tegevuste haldamist. Selliseid haldusvahendeid on turul palju ja selle õige valimine võib olla üpriski keeruline. Üldiselt on tööriistad grupeeritud vastavalt mõnele testimise meetodile või piirkonnale tarkvaras, mida soovitakse testida [3]

Valides õiget testihaldusvahendit, tuleb arvestada mitmete faktoritega. Kas on võimalik kasutada tööriista, mida kasutatakse ka tarkvara arendamiseks? Kui ei ole võimalik kasutada sama tööriista, siis kas on võimalik tulevast testihaldusvahendit integreerida tarkvaraarendustööriistaga? Need on mõned paljudest küsimustest, mis peaksid saama vastused enne, kui võetakse vastu otsus mõnda tööriista kasutada. Sellekohaseid uurimusi on tehtud ja on olemas ka konkreetsed juhendid, kuidas identifitseerida tööriistaga lahendatavat probleemi, kuidas võrrelda erinevaid tööriistu, kuidas valida need õiged ja lõpuks juba mõne konkreetse tööriista proovimine [4] . Antud juhendit kasutab autor ka selles töös, leidmaks tööriista, mis vastaks seatud nõuetele. Järgnevates peatükkides tutvustab autor, mis on traditsiooniline testimine, uuriv ja sessioonipõhine testimine ning mis on nende erinevused.

2.1 Traditsiooniline testimine

Traditsiooniline testimine ehk *scripted testing* on esindatud nii manuaalses testimises kui ka automatiseeritud testimises. Mõlema variandi puhul on vundament sama ehk kasutades mõnda testihaldusvahendit, kirjutatakse tarkvara nõuetest välja testijuhtumid, mida

hiljem kasutatakse tarkvara testimisel. Automatiseerimise puhul on eesmärgiks peamiselt, et teste saab pidevalt jooksutada inimese sekkumiseta ja tihtipeale on nende teostamine kiirem kui manuaalse testimise puhul [5] . Antud uurimustöös ei ole aga automatiseeritud testimine fookuses, sel põhjusel ka siin rohkem ei peatuta.

Manuaalse testimise puhul on oluliseks erinevuseks automatiseeritud testimisest, et seda saab teha ainult inimtestija poolt. Nagu eelpool mainitud on vundamendiks testijuhtumid, mis kirjutatakse tarkvara disainist ja nõuetest. Testijuhtumi eesmärk on kontrollida tarkavara käitumise õigsust tarkvara disaini ja nõuete suhtes.

Et testimisega oleks kaetud suurem osa tarkvarast, kasutatakse testjuhtumite kirjutamisel kahte erinevat detailsusastet. Esimene ja ilmselt ka levinum meetod on, kus kirjutakse kõik tegevused välja samm-sammult ehk olukord, kus testija rolliks on täita konkreetseid samme nendest kõrvale kaldumata. Selline praktika aga ei pruugi alati tagada võimalikult maksimaalset tarkvara kaetust nõuete suhtes. Teine vähem formaalsem meetod on, kus testijale antakse küll sammud, aga need pole kirjutatud otse dokumentatsioonist, järgides samm-sammult meetodit. Teisisõnu on testijuhtum kirjutatud kasutades mingisugust üldist stsenaariumit, mis omakorda jätab testijale nii öelda vabad käed nõude kontrollimiseks [6]

Et teooria oleks arusaadavam, toob autor praktilise näite. Näiteks on vaja testida nutitelefoni rakenduses, kas kasutaja saab rakendusse sisse logida. Kasutades esimest, formaalsemat meetodit, sisestab testija kasutajatunnuse või emaili ja salasõna ning saab oodatava tulemuse, ehk kasutaja on neid samme läbides saanud rakendusse sisse logida. Kasutades aga vähem formaalset meetodit, antakse testijale üldine stsenaarium kontrollimaks, kas rakendusse saab sisse logida. Testija võib siinkohal tekitada erinevaid olukordi või anda rakendusele sisendeid, mida formaalse meetodi puhul poleks kasutatud. Erinevate olukordade ja sisendite all peab autor silmas näiteks interneti kadumist, sisse logimise ajal ekraani sulgemist jne. See kõik taandub individuaalse testija kogemustele tarkvara testimise ja konkreetse valdkonna või tehnoloogia (nt mobiilirakendused, veebilehed, spetsiaaltarkvara vms) vallas. Standardne testijuhtum on ka väga konkreetselt ära defineeritud testimise dokumenteerimise standardis IEEE829 [7]

2.2 Sessioonipõhine ja uuriv testimine

Sessioonipõhine ja uuriv testimine on üksteisele toetuvad testimismeetodid. Kui tarkvara testimisel otsustatakse kasutada uurivat testimist traditsioonilise testimise asemel, on hea praktika ka testimise dokumenteerimine.

Uurivat testimist on palju võrreldud *ad hoc* testimisega, mida tuntakse kui laiska ja hooletut testimist [8] *Ad hoc'*i puhul ei planeerita testimist ja ei koostata dokumentatsiooni [9] mis ongi peamine erinevus uurivast testimisest. Erinevaid stiilinäiteid uurivas testimises:

- Kõike tavalisem stiil, mida kasutatakse, on **intuitiivne**, kus testija teeb midagi kogemuste ja teadmiste baasil.
- Teine populaarne ja laialt kasutatav stiil on ***touring,*** mida teostatakse peamiselt mõne uue funktsionaalsuse testijuhtumi kirjutamisel. Sellist stiili kasutavad ka automatiseerijad, et õppida süsteemi tundma.
- Keerulisem stiil on **süstemaatiline** testimine, kus eesmärgiks on, et uurivad testimised oleksid korratavad ja järjepidevad.
- **Regressioonitestimine**, mis on esindatud ka traditsioonilises testimises. Erinevuseks on, et uuriva testimise puhul kasutatakse märksõnu, traditsioonilise testimise puhul aga testijuhtumeid.
- Kasutatakse ka automatiseerimise tööriistu, mille eesmärkideks on kas testimise kiirendamine või mõne olukorra simuleerimine.

Uuriv testimine on meetod ja mitte tehnika, mis tähendab, et uurivas testimises saab kasutada erinevaid testimise tehnikaid nagu näiteks funktsionaalne testimine, turvatestimine või kasvõi jõudluse testimine [10] Siinkohal on välja toodud mõned traditsioonilise testimise ja uuriva testimise tugevused ja nõrkused **Error! Reference source not found.**Traditsioonilise testimise tugevused:

- Kogemusteta testijal on lihtsam verifitseerida süsteemi nõudeid jooksutades testijuhtumeid ette antud sammudega. Nii ei teki olukorda, kus testija testis valet nõuet.
- Testijuhtumite tulemustest sõltuvalt on võimalik hinnata tarkvara kvaliteeti.
- Parem ressursi valik, saab kasutada automatiseeritud testimist lisaks manuaalsele testimisele.

- Testimise jälgitavus on lihtsam, sest samasid testijuhtumeid kasutakse uuesti ja uuesti.

Traditsioonilise testimise nõrkused:

- Testijuhtumite pidev uuendamine ja haldamine võib osutada ajamahukaks.
- Vähene testijate testimisoskuste kasutamine ja vabaduse võtmine, kasutades samm-sammult meetodit testijuhtumites.
- Disainimine ja dokumenteerimine nõuab märkimisväärset pingutust ja aega, mis omakorda vähendab aega testimiseks.
- Automatiseeritud testijuhtumid on sõltuvuses inimese poolt tehtud vigadele – kui automatiseerija kirjutab programmikoodi mõnele testijuhtumile, võib viga olla programmeeritud koodis ja mitte testijuhtumis endas tarkvara vastu.
- Testijuhtumite kvaliteet sõltub testijate oskustest ja kogemustest.

Uuriva testimise tugevused:

- Kriitiliste vigade kiirem avastamine.
- Vähem nii aja- kui ka rahakulukas, sest ei pea haldama testijuhtumeid. Jääb rohkem aega testimiseks.
- Testijate testimisoskuste ja teadmiste maksimaalne ära kasutamine. Testijal on vabad käed uurimaks süsteemi ja avastamaks probleeme.
- Testimise tulemuste arutamine teise testijaga, mis võib tuua välja uusi riske testitavas süsteemis, mida poleks avastatud traditsioonilises testimises.
- Parem süsteemi analüüs, sest rakendust testitakse lõppkasutaja seisukohast nii nagu seda teeks potentsiaalne rakenduse kasutaja.

Uuriva testimise nõrkused:

- Testimise kvaliteet sõltub testija oskustest ja kogemustest.
- Raske hinnata, kas kõik funktsionaalsused ja piirkonnad said testitud.
- Testimise progressi on keeruline jälgida.
- Probleemi uurimine võtab rohkem aega, kuna ei järgita otseselt dokumentatsiooni, vaid märksõnu.
- Vähese dokumenteerimise tõttu on keeruline hinnata, kui palju süsteemist sai testitud, raske on hinnata testikatet.

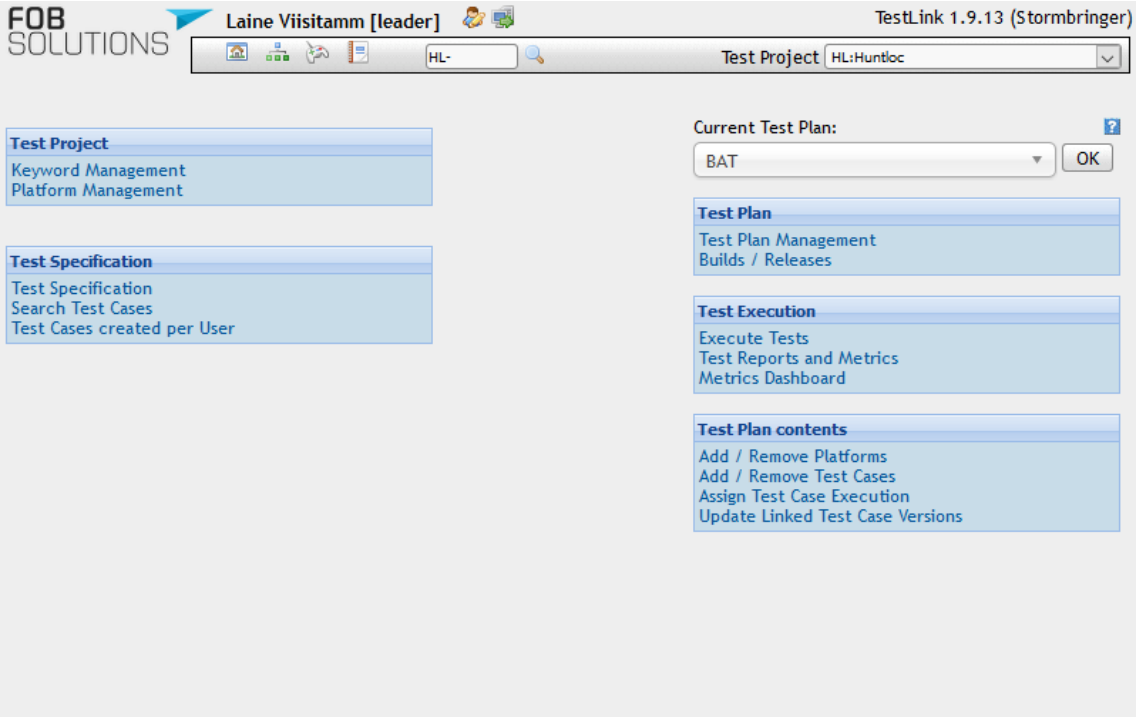
Et uuriv testimine oleks rohkem formaalsem, struktureeritum ja läbipaistvam, kasutatakse sessioonipõhist testimise meetodit uuriva testimise dokumenteerimiseks. Sessioonipõhisest testimistest on kirjutanud allikad [12] ja [13] ning järgnev kokkuvõte on koostatud nendele tuginedes. Sessioonipõhine testimine on loodud James ja Jon Bach'i poolt, et jälgida, mis on testitud ja kuidas. Sessiooni ajal ei tohiks olla mingeid segamisi nagu näiteks emailid, telefoni kõned jne. Sessioonide haldamiseks peab olema loodud kas sessiooni leht või siis mõni programm, mis täidab etteantud eesmärgi. Sessiooni pikkus peaks jääma 60ne või 120ne minuti vahele. Kui sessioon on lõppenud, vaadatakse leiud üle, kaasates sinna kolleege ja testijuhte. Sellised ülevaatused toovad välja uusi aspekte ja ideid, mida hiljem saab kasutada uutes testsessioonides. Et sessioon oleks organiseeritum, on välja pakutud kolm erinevat faasi:

1. Testimise tegevused – süsteemist vigade ja probleemide otsimine kasutades uurivat testimist.
2. Vigade uurimine ja raporteerimine – vead, milleni testija on jõudnud ja nende vigade raporteerimine veahaldus keskkonda.
3. Sessiooni seadistamine – siia kuulub kõik, mida on vaja, et saaks sessiooni teostada, alustades dokumentatsiooni lugemisega ja lõpetades seadmete ja töökeskkondade konfigureerimisega.

2.3 Traditsioonilised testihaldusvahendid

Kuna traditsiooniline testimine on olnud pikalt ja väga laialdaselt levinud meetod, siis on ka nõuded antud programmile või vahendile põhjalikult kaardistatud. Erinevad programmid pakkuvad erinevaid võimalusi, alustades testijuhtumite haldamisega ja lõpetades veahaldusega, et kogu arendus ja testimine tehtaks samas keskkonnas. Kuna erinevaid programme on palju, mis kokkuvõttes üksteisest ei erinegi, võetakse fookusesse avatud lähtekoodiga veebipõhine programm Testlink [14] mida autor ise on kasutanud.

Antud teema käsitlemisel kasutab autor programmi manuaali [15] ja kogemusi, mis on ajaga tekkinud. Puustruktuur, kust kogu testihaldusvahend hakkab lahti hargnema, on testiprojekt. Testiprojekt sisaldab endas testiplaani, testispetsifikatsioone, testide teostamist ja testi raporteid ning meetrikaid (Joonis 1).

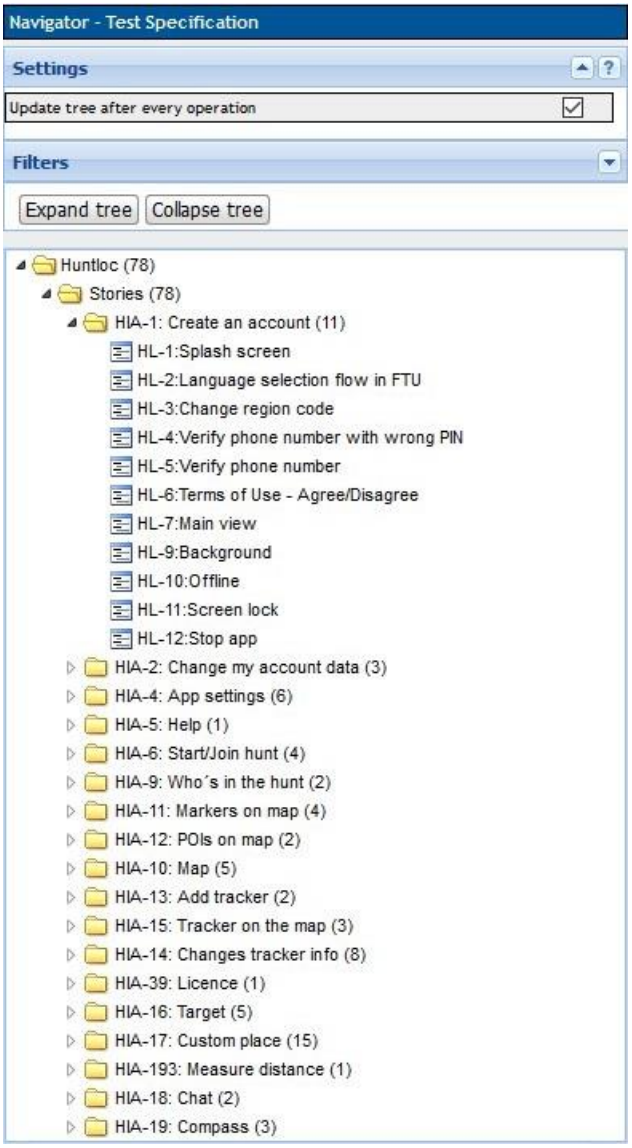


Joonis 1. Testlink projekti avakuva

Testiprojekti lisatakse testijad, kes hiljem hakkavad teste teostama. Testiplaani alla kuulub informatsioon projekti kohta, mis sisaldab projekti kokkuvõtet; detaile, mida peab testima ja mida ei pea testima; kriteeriume; testimise keskkondi; testimise vahendeid; riske ja abistavat dokumentatsiooni. Olulise informatsioonina toob töö autor eraldi välja punktid, mis kuuluvad testiplaani koostamisse:

- Tarkvara versioon – selgitab, millise tarkvara versiooni vastu testitakse.
- Testikaustad ja testijuhtumid – tulenevad testispetsifikatsioonidest.
- Platvorm – kus testimist teostatakse. Platvormide kaetus on eelnevalt defineeritud tarkvara nõuetes.

Teste on oma hallatavuse ja struktureerituse mõttes võimalik organiseerida testikaustadena, mille sees võivad asuda teised testikaustad ja/või testijuhtumid (Joonis 2).

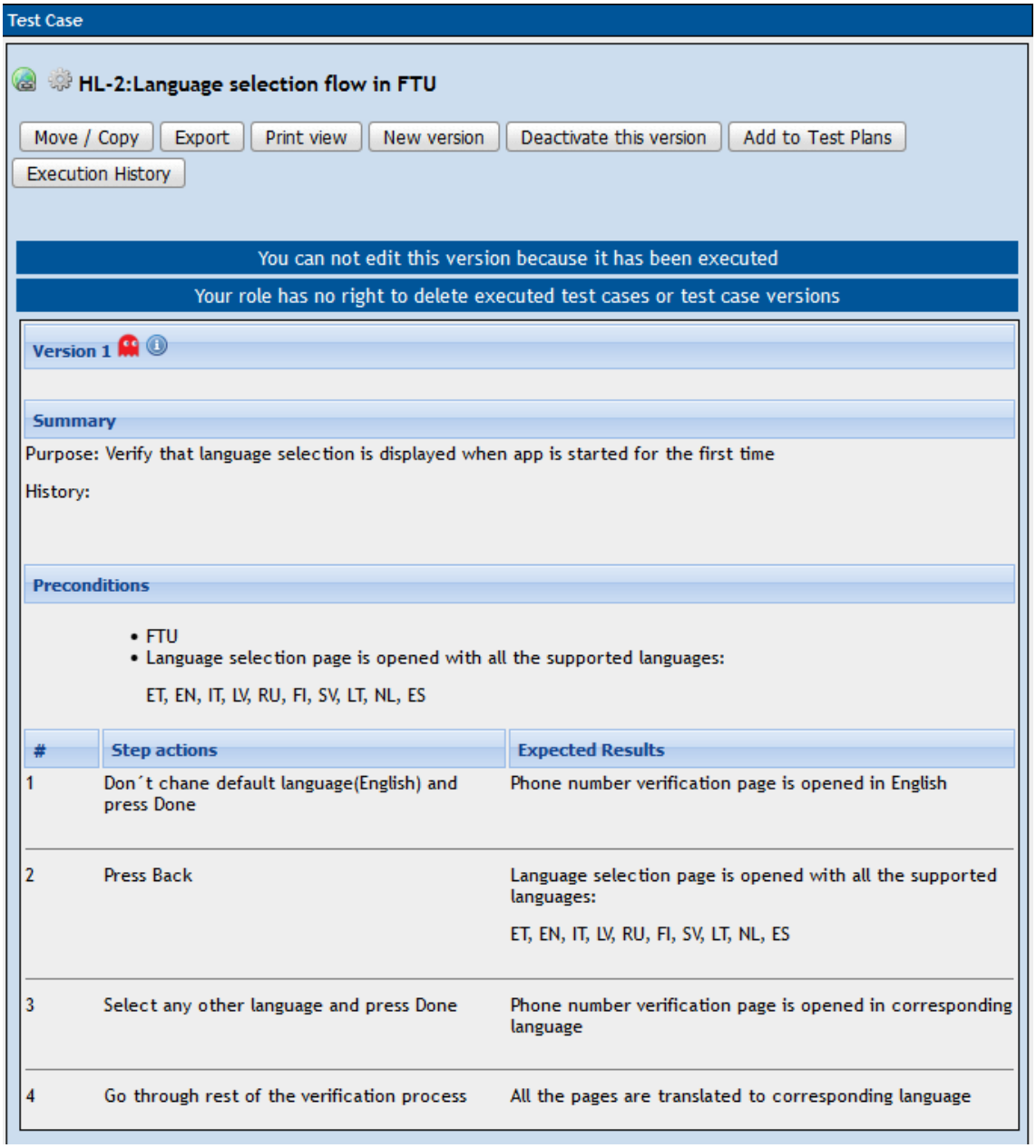


Joonis 2. Testikaustad ja kaustade sees olevad testijuhtumid Testlinkist

Kasutades sellist puustruktuuri, pannakse paika kogu testitav süsteem. Kaustadesse lisatakse testijuhtumid, mis on juba mõne konkreetse nõude kontrollimiseks. Testijuhtum (Joonis 3) võimaldab detailselt välja kirjutada kõik sammud ja nende oodatavad tulemused nii nagu testimise dokumenteerimise standard ette näeb **Error! Reference source not found.**Testijuhtum sisaldab endas järgnevaid komponente:

- Identifikaator – unikaalne numbrikombinatsioon, mille süsteem lisab igale uuele testijuhtumile.
- Pealkiri – testijuhtumit iseloomustav mõne sõnaline fraas.
- Kokkuvõte – lühike ja ülevaatlik testijuhtumi kirjeldus.
- Sammud – testitava süsteemi nõuetest välja kirjutatud tegevused.

- Oodatavad tulemused – testitava süsteemi nõuetest välja kirjutatud sammude tulemused.



Joonis 3. Testijuhtumi näide Testlinkist

Lisaks neile on veel võimalus lisada pilte, seada testijuhtumile tähtsust või prioriteeti ning on ka võimalik määrata, kas testijuhtum on manuaalseks kasutamiseks või automatiseeritud testi kasutamiseks.

2.4 Probleemide defineerimine

Nagu eelmises peatükis 2.3 kirjutatud, on traditsioonilise testihaldusvahendi puhul nõuded paigas, kuidas testimist sooritada, hallata ja mõõta. Sama on vajalik ka sessioonipõhise testimise puhul, sest testimise eesmärk, kui selline, ei muutu.

Võrreldes kahte erinevat testimise meetodit ja kui nüüd üritaks selle najalt mõtteliselt kasutada traditsioonilist testihaldusvahendit sessioonipõhises uurivas testimises, olemegi probleemi ees. Sessioonide haldamiseks on vajalik tööriist, kus saab hallata uurivaid testisessioone ja läbi viia erinevaid testimisi. Traditsioonilises testimises olid fookuses testijuhtumid, uurivas testimises aga testisessioonid. Testisessiooni ei saa hallata kui testijuhtumeid, sest ei testita tarkvara dokumentatsiooni vastu, kasutades samm-sammult meetodit. Testijuhtumi ja sessioonilehe sisendid on erinevad, mistõttu on vajalik teistsugune tööriist teistsuguste nõuetega.

1. Traditsioonilisel testihaldusvahendil on palju lisafunktsionaalsust, mis enamasti ei leia kasutust. Tulevane tööriist võiks olla lihtne ja loogiline nii funktsionaalsuse kui ka visuaalse poole pealt.
2. Testijuhtumeid ei kasutata uurivas testimises – probleemiks on selle asendus sessioonilehega otsitavas tööriistas.
3. Uurival testimisel puudub konkreetne mõõde testikatteks. Traditsioonilise testimise puhul oli see kergesti mõõdetav läbi testijuhtumite. Vajalik on leida viis mõõtmaks testikatet, mis on oluline kriteerium tarkvarale esitatud nõuete suhtes.
4. Uurivas testimises puuduvad testijuhtumid – probleemiks on tarkvara piirkondade defineerimine enne testimist.

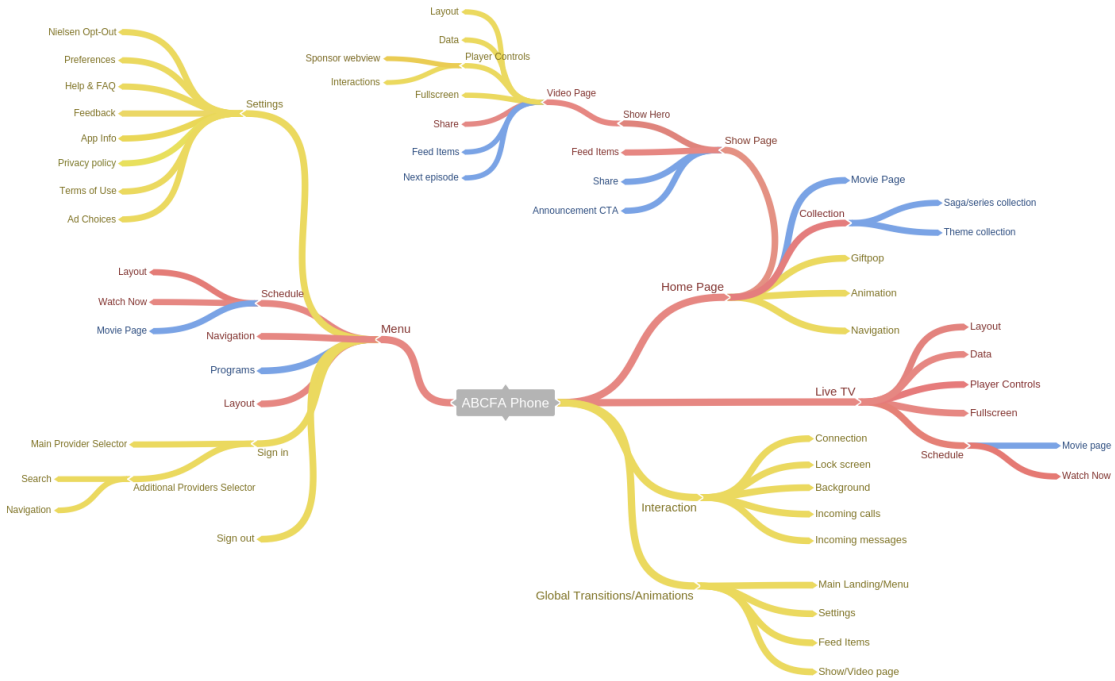
2.5 Nõuded vajalikule tööriistale

Et leida see õige tööriist, on vaja kaardistada, missugused on nõuded vajalikule testihaldusvahendile. Vajalik tööriist on paljuski sarnane traditsioonilisele testihaldusvahendile, erinevusega et enam ei ole testijuhtumeid, millest lähtuda. Algatuseks toob autor välja samasused traditsioonilise testihaldusvahendiga:

- Testiprojekt – vajalik defineerimaks, millise projektiga on tegemist.
- Testplaan – näitab, millise sprindi vastu testitakse. Väga oluline regressiooni testimise puhul, sest see aitab jälgida muutusi tarkvaras.

- Raportid ja meetrika – mõõtmaks testikatet ja võrdlemaks testiplaane, eriti oluline regressiooni testimise puhul.
- Platvormid – millistel platvormidel testimist läbi viiakse.
- Tarkvara versioon – millise tarkvara versiooni vastu testitakse.
- Kasutajad – testimise ja progressi jälgimiseks meeskonnas, eriti vajalik regressiooni testimise puhul.

Et oleks võimalik saada parem pilt testiprojektist, oleks vajalik visuaalne ülevaade kogu tarkvarast. Selle lahendamiseks võib kasutada näiteks mõttekaarti ehk *mind map*'i, kus saab välja kirjutada või joonistada kogu tarkvara puu. Mõttekaarti võib võrdsustada traditsioonilises testimises testi kaustade ja testijuhtumitega. Mõttekaardile (Joonis 4) kirjutatakse tarkvarast välja piirkonnad märksõnadega, mida testija kasutab uurivaks testimiseks. Antud olukorras kasutas autor keskkonda Coggle [16]



Joonis 4. Mõttekaart visualiseerimaks tarkvara märksõnadega

Tulevasel tööriistal peab olema võimalus lisada sessiooni lehti ehk *test sheets*'e. Testi lehtedel omakorda peaks olema võimalus lisada ülal väljatoodud nõuded ehk siduda testiplaaniga, platvormiga, tarkvara versiooniga ja kasutajatega. Neile lisaks peab olema võimalus sisestada:

- Vead – kuhu saab sisestada raporteeritud vead, mis leiti tarkvarast. Ideaalne oleks, kui tulevane tööriist ja veahaldussüsteem integreeruksid omavahel, selleks et tekitada sild kahe tööriista vahele.
- Tegevused – testija kirjutab, mida täpsemalt testitakse.
- Leiud – kõik leiud tarkvarast, kas siis konkreetsed vead või probleemid, mis vajavad rohkem uurimist.
- Aeg – testi lehel peab olema võimalus valida või sisestada aeg, mis kulus sessiooni teostamiseks.
- Ülevaatus – hilisem sessiooni analüüs leidude ja/või ideede kohta.

Oluliseks nõudeks on ka testisessioonide hallatavus ja võimalus filtreerida neid hiljem, kasutades erinevaid tüüpe. Tüüpide all peab autor siinkohal silmas filtreerimist näiteks piirkonna, seadme ja teiste võimaluste järgi. Sessioonidele tagasi vaadates võib leida olulist informatsiooni näiteks regressiooni kohta ja saab jälgida tarkvara üldist arengut.

Nõuded, mis antud peatükis esitati, ei ole lõplikud ja võivad erineda. Antud lõputöös toetus töö autor nõudeid esitades erinevatele allikatele, kogemustele tarkvara testimises ja võeti arvesse ka töö iseloomu. Mõned nõuded, mis esitati ei pruugi olla ilmtingimata nõudeks kõikides projektides. Teiselt poolt võib nimekirjast aga hoopiski puududa nõue, mis on oluline mõnes teises projektis.

3 Uurimus turul olevate tööriistade kohta

Järgnevates peatükkides on välja toodud mõned tööriistad, mis sobiksid kõige paremini esitatud testihaldusvahendi nõuetele peatükis 2.5. Töö autor tutvustab põhjalikumalt neid tööriistu, mis vastasid osaliselt või täielikult haldusvahendile esitatud nõuetele.

Järgnevates peatükkides 3.1, 3.2 ja 3.3 on välja toodud kõik nõuded otsitavale tööriistale, mis seati peatükis 2.5. Autor teeb kokkuvõtva analüüsi iga nõude suhtes, kas nõue sai täidetud, osaliselt täidetud või nõue ei saanud täidetud.

3.1 Jira Capture

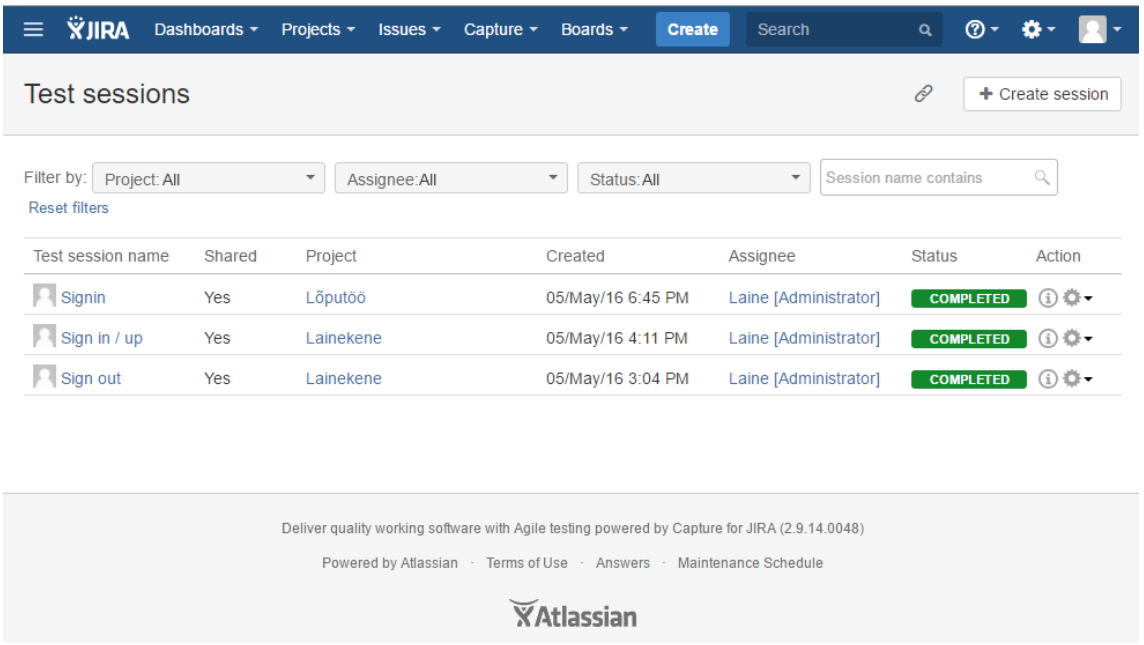
Jira Capture (Joonis 5) on Atlassiani perekonnast pärit tasuline sessioonide haldamiseks mõeldud *plugin*. Tööriista kirjeldamisel kasutab autor tööriista manuaali [18] ja katsetusi antud keskkonnas.

Esitatud nõuded ja nõuete analüüs tööriistale Jira Capture:

- Testiprojekt – võimalus lisada erinevaid projekte, nii tarkvaraarendus, veahaldus kui ka sessioonihaldus käib ühe projekti all.
- Testplaan – ei ole võimalus defineerida sprinte ja/või testiplaani.
- Raportid ja meetrika – puudub igasugune võimalus raportite ja meetrika väljastamiseks.
- Platvormid – puudub võimalus ise sisestada platvorm, veebi testimise puhul on võimalus teha kastikesse linnuke, mis võtab ise selle informatsiooni veebilehitsejast.
- Tarkvara versioon – ei ole võimalust eraldi sisestada.
- Kasutajad – võimalus lisada kasutajaid ja anda õigusi erinevatele projektidele.
- Mõttekaart – võimalus lisada Jira-sse tasuline *Mindmap*-i *plugin*.
- Vead – veahaldussüsteem olemas ja saab vigasid siduda sessioonidega.
- Tegevused – tavaline teksti sisestamine.
- Leiud – konkreetne väli puudub, saab lisada leiud tegevuste alla.

- Aeg – aeg hakkab jooksma, kui sessiooni alustatakse ja lõpeb, kui sessioon lõpetatakse, konkreetsemad aja näidud puuduvad sessiooni kohta.
- Filtreerimine – võimalus filtreerida sessioone projekti, sessioonile määratud isiku ja sessiooni staatuse järgi.
- Ülevaatus – puudub võimalus ülevaatuse dokumenteerimiseks, vajadusel saab kasutada märkmete võimalust sessiooni lehe allosas.

Lisana pakub Jira Capture võimalust veebilehitsejale lisada Capture pikendus, mis on väga mugav, kui testimise platvormiks on veeb. Saab väga lihtsalt alustada sessioone, läbi viia sessioone testija perspektiivist, lisada pilte leidudest ning isegi vigasid raporteerida.



Joonis 5. Jira Capture avavaade

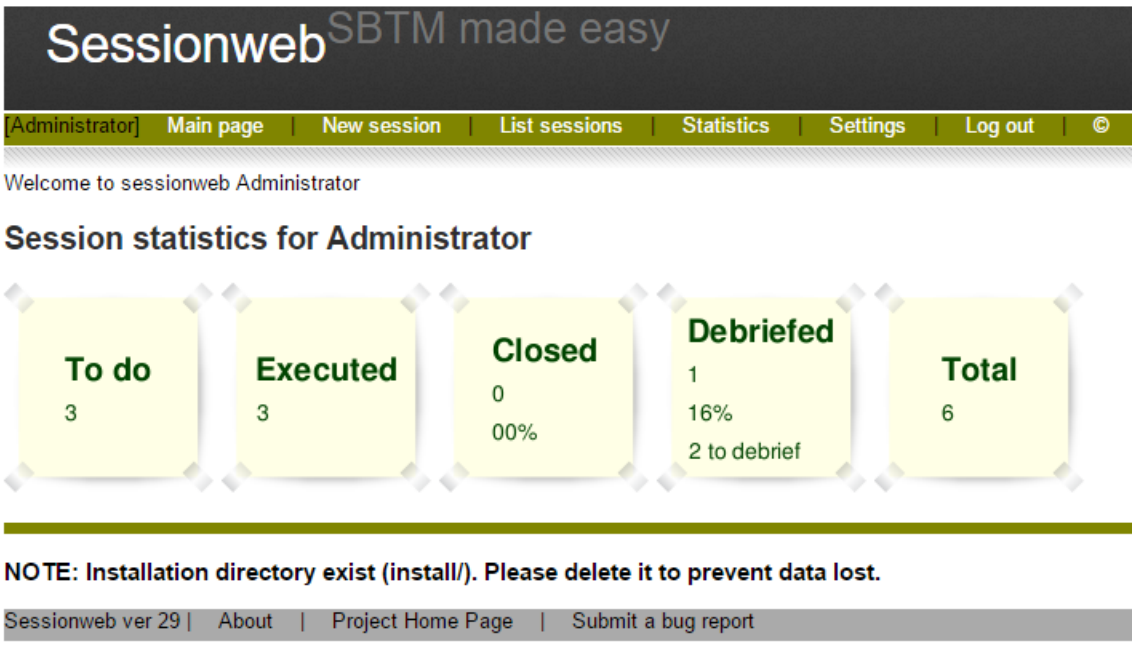
3.2 Sessionweb

Sessionweb [19] (Joonis 6) on avatud lähtekoodiga tasuta tööriist haldamaks sessioonipõhist ja uurivat testimist. Tööriista kirjeldamisel kasutab autor tööriista *FAQ* veebilehte [20] ja katsetusi antud keskkonnas. Järgnevas peatükis analüüsitakse tööriista vastavust nõuetele.

Esitatud nõuded ja nõuete analüüs tööriistale Sessionweb:

- Testiprojekt – tööriist ei võimalda luua erinevad projekte, mis tähendab, et erinevate projektide olemasolul on keskkonda vaja duplikeerida.

- Testplaan – on võimalus kasutada sprint välja.
- Raportid ja meetrika – tööriist näitab kolme erinevat statistikat testi sessioonide kohta: ajaline, progressiline ja piirkonnaline.
- Platvormid – võimalus lisada platvorme, kus testimist sooritati.
- Tarkvara versioon – võimalus lisada, aga sisend on kui tavaline teksti sisend ja sellest johtuvalt ei ole võimalik seda kasutada hiljem filtreeringus.
- Kasutajad – võimalus lisada kasutajaid ja anda neile erinevaid õiguseid.
- Mõttekaart – puudub võimalus mõttekaarti sisestada, peaks kasutama mõnda teist keskkonda.
- Vead – võimalus integreerida veahaldussüsteemiga läbi lähtekoodi (saab lisada veahaldussüsteemi hüperlingi ja hiljem keskkonnas lisada vea ID).
- Tegevused – tavaline tekstiredaktor, saab oma vajadustele vastavalt konfigureerida.
- Leiud – tavaline tekstiredaktor, saab oma vajadustele vastavalt konfigureerida.
- Aeg – sessiooni aega mõõdetakse protsentides ja meetrika sisaldab: testi ülesseadmine, testimine, veahaldus ja *opportunity* ehk ideed, mis tekkisid.
- Filtreerimine – võimalus sessioone filtreerida vastavate parameetritega: testija, sprint/testplaan, meeskond, piirkond ja sessiooni staatus.
- Ülevaatus – võimalus ülevaatus sessioonile, lisada märkusi ja muuta sessiooni staatust: mitte üle vaadatud (algne olek), üle vaadatud ja *closed* ehk suletud.



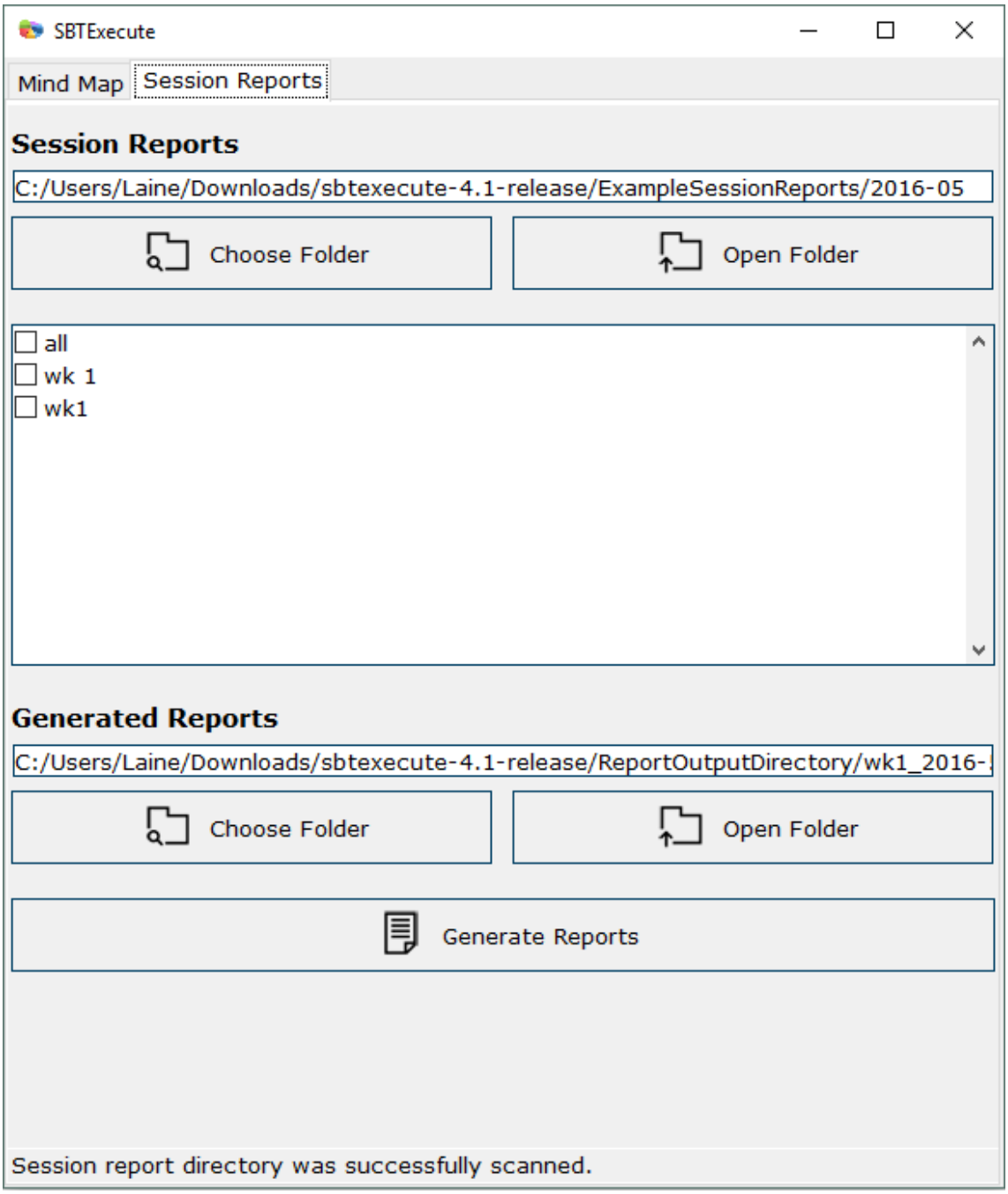
Joonis 6. Sessionweb-i avavaade

3.3 SBTExecute

SBTExecute [21] (Joonis 7) on tasuta suletud lähtekoodiga tööriist, haldamaks uurivast testimisest tulenevaid sessioone. Tööriista kirjeldamisel kasutab autor katsetusi antud keskkonnas. Järgnevas peatükis analüüsitakse tööriista vastavust nõuetele. Programm ise on Java põhjal üles ehitatud, sessiooni lehti hallatakse Excelis.

Esitatud nõuded ja nõuete analüüs tööriistale SBTExecute:

- Testiprojekt – võimalus kasutada tööriista erinevate projektide tarbeks. Selleks peab tekitama igale projektile uue kausta, kuhu raportid lähevad.
- Testplaan – võimalus kasutada kaustade loogikat defineerimaks testiplaani.
- Raportid ja meetrika – tööriist genereerib erinevat meetrikat sessiooni raportitest: kokkuvõte sprintidest, piirkondadest, vigade meetrika, vead tarkvaraversiooni vastu jne. Lisaks on võimalik näha protsentuaalselt aega, mis kulutati testi ülesseadmiseks, vigade raporteerimiseks ja testimiseks.
- Platvormid – puudub võimalus platvormi valikuks.
- Tarkvara versioon – võimalus lisada.
- Kasutajad – võimalus lisada kasutajaid *data* ehk andmebaasi Exceli lehele.
- Mõttekaart – olemas integratsiooni programmiga Xmind, mille saab koostada ja hiljem sessiooni raporti lehele üle kanda Exceli formaati.
- Vead – võimalus lisada vigasid raporti lehele, otsene integratsioon veahaldussüsteemiga puudub.
- Tegevused – tavaline teksti sisestamine Excelisse.
- Leiud – tavaline teksti sisestamine Excelisse.
- Aeg – sessiooni aega mõõdetakse protsentides ja meetrika sisaldab: testi ülesseadmine, testimine ja veahaldus.
- Filtreerimine – puudub võimalus filtreerida, sest iga sessioon asub eraldi Exceli lehel.
- Ülevaatus – puudub võimalus ülevaatuse dokumenteerimiseks.



Joonis 7. SBTExecute avavaade

3.4 Tööriistade võrdlus ja järeldused

Pärast mõne ajalist erinevate tööriistade kasutust selgus, et kõige paremini vastavad nõuetele Sessionweb ja SBTExecute tööriistad. Mõlema puhul said enamus nõuetest täidetud väiksemate mööndustega. Kahjuks on aga mõlemad tööriistad kõvasti ajale jalgu jäänud ja puudub igasugune tugi ning arendus. Jira Capture jäi nõuete täitmisel hätta, millest on kahju, sest tegu on kõige tänapäevasema tööriistaga. Järgnevates lõikudes teeb töö autor järeldused kõikide tööriistade kohta.

Sessionweb-i puhul jäid täitmata järgnevad nõuded:

- Testiprojekt
- Mõttekaart

Peale nende puuduste toob autor veel välja, et tarkvara versiooni sisestamiseks on loodud tavaline teksti sisend. Kahjuks sellised tekstiredaktori sisendid ei kajastu hiljem ülevaatlikus mõttes, mis omakorda loob olukorra, kus ei ole võimalik tagasi vaadata ja jälgida tarkvara arengut ning muutusi versiooni järgi. See muutub eriti oluliseks just pikemates projektides ehk projektid, mis kestavad aastaid. Lühemate projektide puhul saab kokkuleppel kasutada ka olemasolevat varianti ja tööriist nii öelda enda kasuks tööle panna.

Sessionweb-i puhul peab veel ära mainima, et mõned lehed olid katki ehk tööriist viskas php erroreid, mis tähendab, et kui omade jõududega neid korda ei tee, siis tööriist lihtsalt ei tööta nii nagu soovitud. Kuna tundub, et tööriistale uuendusi ega arendusi ei tehta, siis lahenduseks oleks vead ise ära parandada. Õnneks on tegu tööriista logi failidega, mis ei sega sessioonide haldamist. Tegemist on ju avatud lähtekoodiga tööriistaga, mis seda võimaldab. Sessionweb-i plussiks on aga see, et kogu keskkond on üles ehitatud loogiliselt ja nii nagu töö alguses teoreetilises osas välja toodud. Visuaalselt näeb ta välja võrdlemisi vana ja väsinud, mis tänapäeva maailmas võib saada takistuseks ja põhjuseks, miks tööriista mitte kasutusele võtta.

SBTExecute puhul jäid täitamata järgnevad nõuded:

- Platvormid
- Filtreerimine
- Ülevaatus

Täitmata nõuetest on üks olulisemaid puudusi platvormid ja seda just siis, kui testimine toimub kas veebis või siis nutitelefonis. Erinevatel platvormidel on omad iseärasused, mis ongi antud puuduse vajalikkus tarkvara testimisel.

Kui hinnata veel SBTExecute üleüldiselt, siis tegemist on tööriistaga, mis nõuab palju haldust ja tööd. Seda just alguses, et üles ehitada kogu testimise haldamise võrgustik. Võrgustiku all peab autor silmas kaustade loomist sessioonide ja meetrika tarbeks. Õnneks on tänapäeval olemas erinevaid pilveteenuseid pakkuvaid ettevõtteid, mis

tähendaks, et kogu sessiooni halduse saaks tervele testimise meeskonnale seada üheaegselt kättesaadavaks. Pikemas perspektiivis muutub selline süsteem aga ilmselt koormavaks ja kasutegur võib sootuks kaduda. Mõte, et hallata erinevaid sessioone kaustades, ei ole ühelegi kogenud testijuhile meeltmööda.

Positiivse poole pealt toob autor välja, et antud tööriistal on olemas integratsioon mõttekaardiga, mis on vägagi vajalik ja oluline nõue, et testimist läbi viia. Mõttekaardi integratsioon Excelisse tähendab, et peale visuaalse poole on võimalik lihtsalt hallata piirkondi, mis on testimisele suunatud ja seda erinevatel tasanditel.

Jira Capture puhul jäid täitmata järgnevad nõuded:

- Testiplaan
- Raportid ja meetrika
- Tarkvara versioon
- Leiud
- Ülevaatus

Võttes arvesse, et numbriliselt oli tööriistale esitatud nõudeid 13, siis nendest 5 jäi täielikult täitma. Osade nõuete suhtes oli täidetud mingisugune osa või autor hindas, et nõuet saab täita kasutades mõnda teist funktsionaalsust, mida tööriist pakub. Järeldus puhtalt nõuete täitmisest näitab, et tööriistal on suuri puudusi. Tundub, et tegu on tööriistaga, mida saab kasutada küll sessioonide läbiviimiseks, aga mitte nii väga praktiseerimaks sessioonipõhist uurivat tarkvara testimist pikemas perspektiivis. See tuleneb sellest, et tööriista võimalused peegeldavad testimist selle väga lihtsas mõttes.

Täidetud nõuete suhtes kõige suuremaks plussiks Jira Capture puhul on, et testiprojekti saab väga lihtsalt siduda veahaldusega. Mõlemad on samas keskkonnas ja selletõttu ei kaasne suurt ajakulu sessioonide haldamiseks ega ka vigade raporteerimiseks ja sidumaks tarkvara arendusega. Seotult eelnevaga on plussiks ka see, et kui ettevõte otsustab Jira Capture kasutusele võtta, siis ei saa takistuseks erinevate projektide olemasolu. Projekte saab lisada palju ja see on suur lisaväärtus ettevõttele, kus töös on mitmeid erinevaid projekte samal ajal.

Kui aga panna nõuded hetkeks teisejärguliseks, siis Jira Capture puhul on suureks plussiks, et tegemist on kaasaegse tööriistaga. Tootele on olemas tugi ja kui keskkonda

natuke aega kasutada, tundub see isegi lihtne ja loogiline. Suurima miinusena võib välja tuua, et tegemist on tasulise keskkonnaga. Et üldse saaks keskkonda kasutama hakata, on kõigepealt vaja osta Jira kasutusõigus. Seejärel on vaja osta Capture kasutusõigus ja kui veel kasutada mõttekaarti, on see ka vaja osta. Kõikide nende kasutusõiguste juures on veel üks aga – hind sõltub sellest, kui mitmele kasutajale tööriistad on mõeldud. Mida rohkem kasutajaid, seda suuremaks läheb ka hind. Jira hind sõltub peale kasutajate arvu ka sellest, kas tarkvara installeerimiseks kasutakse Jira pakutavat serverit või kasutakse enda serverit [22] Capture puhul algavad hinnad 10-st dollarist kuus 10-le kasutajale [23] Mõttekaardi, mida töö autor soovitab, hinnad algavad samuti 10-st dollarist kuust 10-le kasutajale [24] Õnneks on aga antud 30-ne päevane tutvumisperiood, mis tähendab, et kõiki neid keskkondi saab katsetada ja proovida enne otsuse langetamist tööriista kasuks.

Nii töö autori poolt kui ka ettevõtte siseselt leiti, et antud tööriistad ei vasta normidele ja ettevõtte siseselt kasutusele ei võeta. Sellest hoolimata oldi probleemi ees, sest testimist oli vaja dokumenteerida. Lahenduseks loodi Exceli fail [Lisa 1], kus testimisi teostada ja hallata. Kasutades erinevaid valemeid ja informatsiooni jagamist erinevate lehtede vahel, loodi ühtne ja loogiline süsteem haldamaks sessioone. Lahendus ei ole küll ideaalne, aga selle eest turvaline, täidab oma eesmärgi ning seda on lihtne hallata.

4 Soovitused vahendite kasutusele võtuks

Üldist pilti vaadates võib öelda, et turul ei ole saadaval palju erinevaid tööriistu, mille seast valida. Tundub, et tööriistu hakati arendama peale seda, kui uuriv testimine kogus populaarsust. Hetkeseisuga on saadaval erinevaid tööriistu, aga paljud neist on ajale jalgu jäänud ja ei vasta enam kasvavatele nõuetele. Uurimusest selgus ka, et nii mõnigi tööriist on oma tegevuse lõpetanud ja pole enam saadaval. Näiteks ei ole enam saadaval uuriva testimise looja James ja Jon Bach'i poolt loodud tööriist Session Tester [25]

Enne kui neist tööriistadest mingisugune kasutusele võtta, soovitab töö autor kindlasti analüüsida, mida täpselt valitud vahend peab tegema ja kuidas on see hea ning kasulik. Antud lõputöös kasutas töö autor nõuete esitamisel uuriva- ja sessioonipõhise testimise teoreetilisi aluseid ning võeti arvesse ka töö iseloomu ettevõttes FOB Solutions, milleks on peamiselt testimine mobiili platvormidel. Töö iseloomust johtuvalt, sobib kõige paremini Sessionweb, mis võimaldab testimist läbi viia edukalt mobiili platvormidel. Tööriistal olid mõned väiksemad programmeerimise vead, mis aga ei sega otseselt sessioonide haldamist, sest vigased lehed on tööriista logi lehed. Et tööriist oleks usaldusväärsem, teeks autor ettepaneku arendusmeeskonnale vead tööriistas parandada ning samuti kohandada tööriista vastavalt töö iseloomule. Selle all peab töö autor silmas erinevate sisendite muutmisi ja lisamisi, mis puudutab just sessioonilehti. Vähem olulise muudatusena teeb töö autor ettepaneku ka disaini muuta või uuendada, et tööriist paistaks välja professionaalsem.

Juhul kui testimise keskkonnaks on veeb, siis oleks autori soovitus kasutusele võtta kas Jira Capture või Sessionweb. Capture puhul muudab tööriista edukaks kasutuseks selle keskkond ja ka ära mainitud *plugin* peatükis 3.1. Küll aga ei soovita töö autor võtta kasutusele Capture-it, kui soovitakse testimist põhjalikult dokumenteerida, sest sel juhul hakkavad testimise haldamist piirama puudulikud nõuded, mida käsitleti peatükis 3.1.

Kindlasti ei soovita töö autor kasutusele võtta SBTExecute tööriista. Kuigi enamus nõuded said küll kaetud, ei ole sessioonide haldamine kiire ja lihtne. Tööriista struktuur ei ole kasutajasõbralik ja pigem testijuhile lisatöö, kuigi tööriist peaks olema testimist

toetav. Kui sessioonide ja raportite haldamine üle viia kaustadelt näiteks veebikeskkonda, muutuks ka ilmselt tööriista kasutamine loogilisemaks. Aga selle tegemine nõuab lisa tarkvaraarendust ja ei tundu kuigi mõistlik.

5 Kokkuvõte

Käesoleva töö eesmärgiks oli uurida ja analüüsida sessioonipõhise tarkvara testimise testihaldusvahendeid.

Töö käigus selgitas autor erinevate testimismeetodite teoreetilisi aluseid. Võrreldi traditsioonilist testimist uuriva sessioonipõhise testimisega ja jõuti järeldusele, et mõlemal meetodil on omad plussid ja miinused. Teoreetilise poole pealt tutvustati ka traditsioonilise testimise testihaldusvahendit, et mõista paremini lahendatava probleemi sügavust.

Teoreetilistest alustest ja tarkvara testimises omandatud kogemustest koostas töö autor nõuded tulevasele tööriistale. Nõudeid tuli kokku 13 antud lõputöös, mis ei pruugi olla aga lõplik nimikiri, sest need võivad varieeruda nii projekti kui ka testimise iseloomust. Esitatud nõuetest sõltuvalt tegi töö autor põhjaliku analüüsi kolme tööriista kohta eraldi, et nende võrdlemine oleks läbipaistvam nii tööriistade endi suhtes kui ka tööriistadele esitatud nõuete suhtes.

Tööriistadele tehtud võrdlustest ja järeldusest selgus, et kõikidel analüüsitud tööriistadel olid oma head ja vead. Põhjalikumalt katsetusi läbi viies leidis tööautor, et kõige paremini sobiks kasutusele võtta Sessionweb ja seda just siis, kui tegu on testimisega mobiili platvormidel. Kõne all olevale tööriistale tehti ettepanekud, kuidas saaks tööriista parendada ja professionaalsemaks muuta.

Jira Capture-i ja SBTExecute puhul leiti, et mõlemal tööriistal on puudused, mis ei rahulda antud lõputöö probleemistikku ja pigem teevad sessioonide haldamist keerulisemaks. Jira Capture-i puhul jäid täitmata paljud nõuded ning suuremal ülevaatusel selgus, et sessioonide dokumenteerimine võib muutuda liiga keeruliseks. SBTExecute tööriistal olid enamus nõuded täidetud, aga vajaka jäi tööriista ülesehitusest, mis muudab sessioonide haldamise liiga ajakulukaks.

Võttes arvesse saadud lõputöö tulemusi, teeb tööautor soovituse antud teemat edasi uurida ning arendada välja uus kaasaegne tööriist.

Kasutatud kirjandus

[1] TutorialsPoint. Software Testing – Myths. http://www.tutorialspoint.com/software_testing/software_testing_myths.htm (07.03.2016)

[2] Shah, S.M.A., Gencel, C., Alvi, U.S., Peterson, K. Towards a hybrid testing process unifying exloratory testing and scripted testing – *Journal of Software: Evolution and Process,* 2014, 2, 220 – 250. http://onlinelibrary.wiley.com/doi/10.1002/smr.1621/full (05.03.2016)

[3] Black, R., van Veenendaal, E., Graham, D. (2012). Foundations of Software Testing ISTQB Certification. 3rd ed. Cengage Learning Emea.

[4] Traq Software Ltd. How to Evaluate Test Management Tools. http://www.testmanagement.com/Test_Tool_Evaluation.pdf (13.05.2016)

[5] Wikipedia.org. Test Script. https://en.wikipedia.org/wiki/Test_script (05.03.2016)

[6] Whittaker, J.A. (2010). Exploratory Software Testing : Tips, Tricks, Tours, and Techniques to Guide Test Design. 1st ed. United States of America : Pearson Education, Inc.

[7] IEEE. (2008). Standard for Software and System Test Documentation. http://ieeexplore.ieee.org/xpl/freeabs_all.jsp?arnumber=4578383 (14.03.2016)

[8] Bach, J. (2003). Exploratory Testing Explained. http://www.satisfice.com/articles/et-article.pdf (05.03.2016)

[9] Chhabra, N. Introduction to Adhoc Testing – *International Journal of Scientific & Technology Research*, 2012, 7. (http://www.ijstr.org/final-print/August2012/Introduction-To-Adhoc-Testing.pdf) (23.05.2016)

[10] Kohl, J. Demystifying Exploratory – *Stickyminds Magazine,* 2010, 2. https://www.stickyminds.com/better-software-magazine/demystifying-exploratory-testing (23.05.2016)

[11] Shah, S.M.A., Gencel, C., Alvi, U.S., Peterson, K. Towards a hybrid testing process unifying exloratory testing and scripted testing – *Journal of Software: Evolution and Process,* 2014, 2, 220 – 250. http://onlinelibrary.wiley.com/doi/10.1002/smr.1621/full (05.03.2016)

[12] Kohl, J. Documenting Exploratory Testing – *Stickyminds Magazine,* 2011, 3. https://www.stickyminds.com/better-software-magazine/documenting-exploratory-testing (27.02.2016)

[13] Bach, J. Session-Based Test Management – *Software Testing and Quality Engineering magazine,* 2000, 11. http://www.satisfice.com/articles/sbtm.pdf (27.02.2016)

[14] Testlink. Testlin Open Source Test Management. http://testlink.org/ (17.04.2016)

[15] Testlink. User manual. TestLink version 1.9. https://wiki.openoffice.org/w/images/1/1b/Testlink_user_manual.pdf (06.05.2016)

[16] IEEE. (2008). Standard for Software and System Test Documentation. http://ieeexplore.ieee.org/xpl/freeabs_all.jsp?arnumber=4578383 (14.03.2016)

[17] Coggle. https://coggle.it/ (17.04.2016)

[18] Atlassian. Capture for JIRA User Guide. https://confluence.atlassian.com/display/CAPTURE/Capture+for+JIRA+User+Guide (24.04.2016)

[19] Sessionweb. About sessionweb. http://www.sessionweb.org/ (02.05.2016)

[20] Sessionweb. FAQ. http://www.sessionweb.org/faq.php (02.05.2016)

[21] SBTExecute. SBTExecute 4.1. http://www.addq.se/test-och-krav/sbtexecute/ (21.05.2016)

[22] Atlassian. https://www.atlassian.com/software/jira/pricing?tab=host-in-the-cloud (21.05.2016)

[23] Atlassian Marketplace. Capture for JIRA. https://marketplace.atlassian.com/plugins/com.atlassian.bonfire.plugin/cloud/pricing (21.05.2016)

[24] Atlassian Marketplace. Yoikee Creator Templates by Mind. https://marketplace.atlassian.com/plugins/com.keinoby.confluence.plugins.yoikee-creator/server/pricing (21.05.2016)

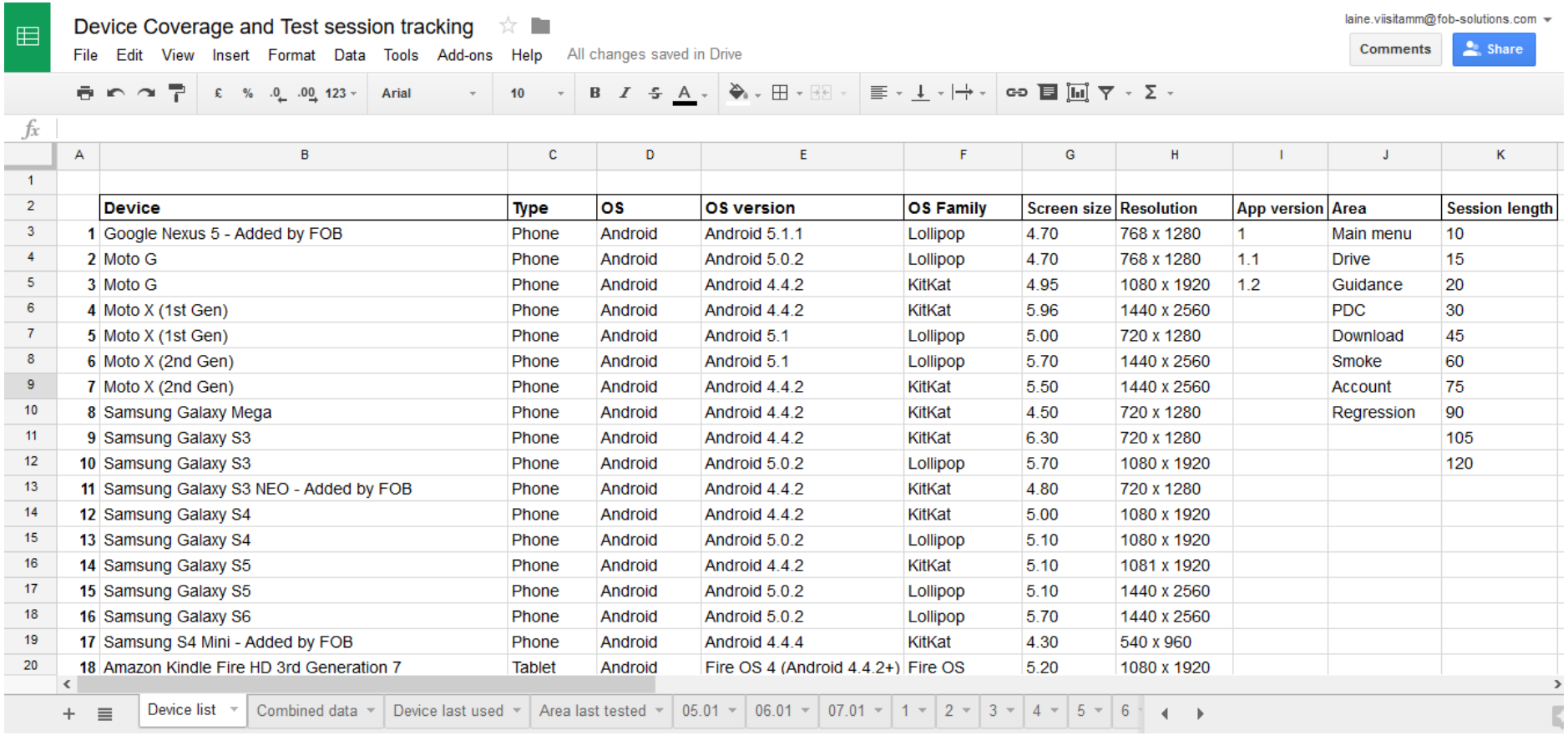
[25] Software informer. Looking for an OpenQA project? http://external.informer.com/sessiontester.openqa.org/ (21.05.2016)

Lisa 1 – FOB Solutions'i poolt välja töötatud Exceli fail haldamaks testi sessioone

Antud lisas tutvustab töö autor ettevõtte FOB Solutions-i poolt välja töötatud Exceli faili, mis on ettevõttes mitmes projektis hetkel kasutuses. Fail on loodud võttes arvesse töö iseloomu ja vajadusi. Projektid, kus seda Exceli faili kasutakse, hõlmavad endas testimist mobiili platvormidel.

Exceli failil on 4 lehte, mis sisaldavad endas erinevaid andmeid, mida omakorda kasutatakse edasi teistes lehtedes. Ülejäänud lehed failis on testisessioonide lehed. Esimesel lehel on kirja pandud kõik oluline seadmete ning tarkvara kohta, mis sisaldab:

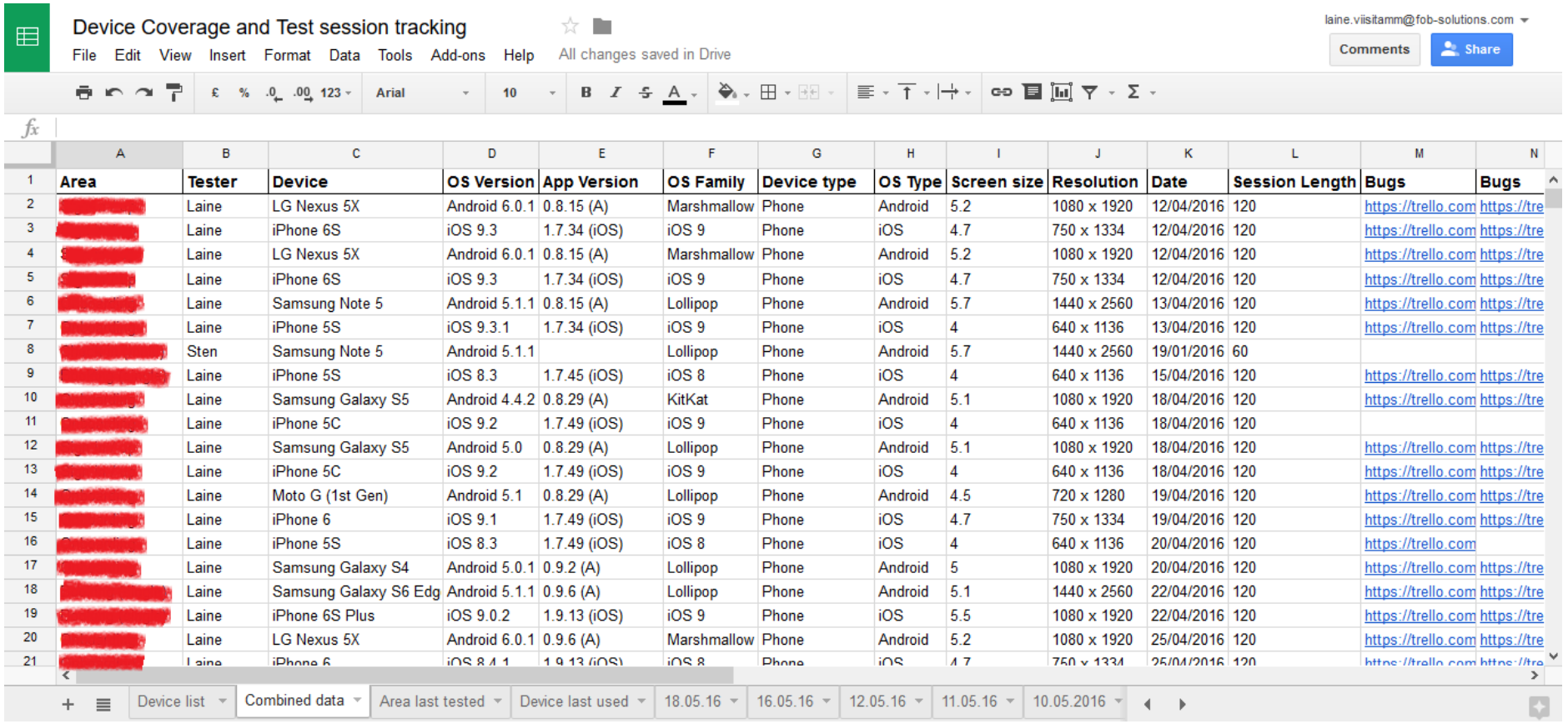
- Seadme nimi
- Seadme tüüp (telefon, tahvelarvuti)
- Seadme operatsioonisüsteem (Android, iOS, Windows Phone)
- Seadme operatsioonisüsteemi versioon (näiteks Android 4.4.2, 5.0, 6.0.1 jne, iOS 8.2, 9.3.2 jne, WP 8, 8.1, 10 jne)
- Seadme perekond (Androidi põhine peamiselt. Android on oma süsteemidel nimed pannud nagu KitKat, Lollipop, Marshmallow)
- Seadme ekraani suurus (oluline seadmepõhiste vigade avastamiseks)
- Seadme resolutsioon (oluline seadmepõhiste vigade avastamiseks)
- Testitava tarkvara versioon
- Tarkvara piirkonnad
- Sessioonipikkus



	Device	Type	OS	OS version	OS Family	Screen size	Resolution	App version	Area	Session length
1	Google Nexus 5 - Added by FOB	Phone	Android	Android 5.1.1	Lollipop	4.70	768 x 1280	1	Main menu	10
2	Moto G	Phone	Android	Android 5.0.2	Lollipop	4.70	768 x 1280	1.1	Drive	15
3	Moto G	Phone	Android	Android 4.4.2	KitKat	4.95	1080 x 1920	1.2	Guidance	20
4	Moto X (1st Gen)	Phone	Android	Android 4.4.2	KitKat	5.96	1440 x 2560		PDC	30
5	Moto X (1st Gen)	Phone	Android	Android 5.1	Lollipop	5.00	720 x 1280		Download	45
6	Moto X (2nd Gen)	Phone	Android	Android 5.1	Lollipop	5.70	1440 x 2560		Smoke	60
7	Moto X (2nd Gen)	Phone	Android	Android 4.4.2	KitKat	5.50	1440 x 2560		Account	75
8	Samsung Galaxy Mega	Phone	Android	Android 4.4.2	KitKat	4.50	720 x 1280		Regression	90
9	Samsung Galaxy S3	Phone	Android	Android 4.4.2	KitKat	6.30	720 x 1280			105
10	Samsung Galaxy S3	Phone	Android	Android 5.0.2	Lollipop	5.70	1080 x 1920			120
11	Samsung Galaxy S3 NEO - Added by FOB	Phone	Android	Android 4.4.2	KitKat	4.80	720 x 1280			
12	Samsung Galaxy S4	Phone	Android	Android 4.4.2	KitKat	5.00	1080 x 1920			
13	Samsung Galaxy S4	Phone	Android	Android 5.0.2	Lollipop	5.10	1080 x 1920			
14	Samsung Galaxy S5	Phone	Android	Android 4.4.2	KitKat	5.10	1081 x 1920			
15	Samsung Galaxy S5	Phone	Android	Android 5.0.2	Lollipop	5.10	1440 x 2560			
16	Samsung Galaxy S6	Phone	Android	Android 5.0.2	Lollipop	5.70	1440 x 2560			
17	Samsung S4 Mini - Added by FOB	Phone	Android	Android 4.4.4	KitKat	4.30	540 x 960			
18	Amazon Kindle Fire HD 3rd Generation 7	Tablet	Android	Fire OS 4 (Android 4.4.2+)	Fire OS	5.20	1080 x 1920			

Teisel lehel on *combined data*, mis kuvab kõiki sessioone kaasates sessiooni lehelt võetud andmeid ning sisendeid. Antud lehel on võimalik filtreerida testisessioone erinevate parameetrite järgi:

- Piirkond
- Testija
- Seade
- Operatsioonisüsteem
- Tarkvara versioon
- Operatsioonisüsteemi perekond
- Seadme tüüp
- Operatsioonisüsteemi tüüp
- Seadme ekraani suurus
- Seadme resolutsioon
- Kuupäev
- Sessiooni pikkus
- Vead



Device Coverage and Test session tracking

File Edit View Insert Format Data Tools Add-ons Help All changes saved in Drive

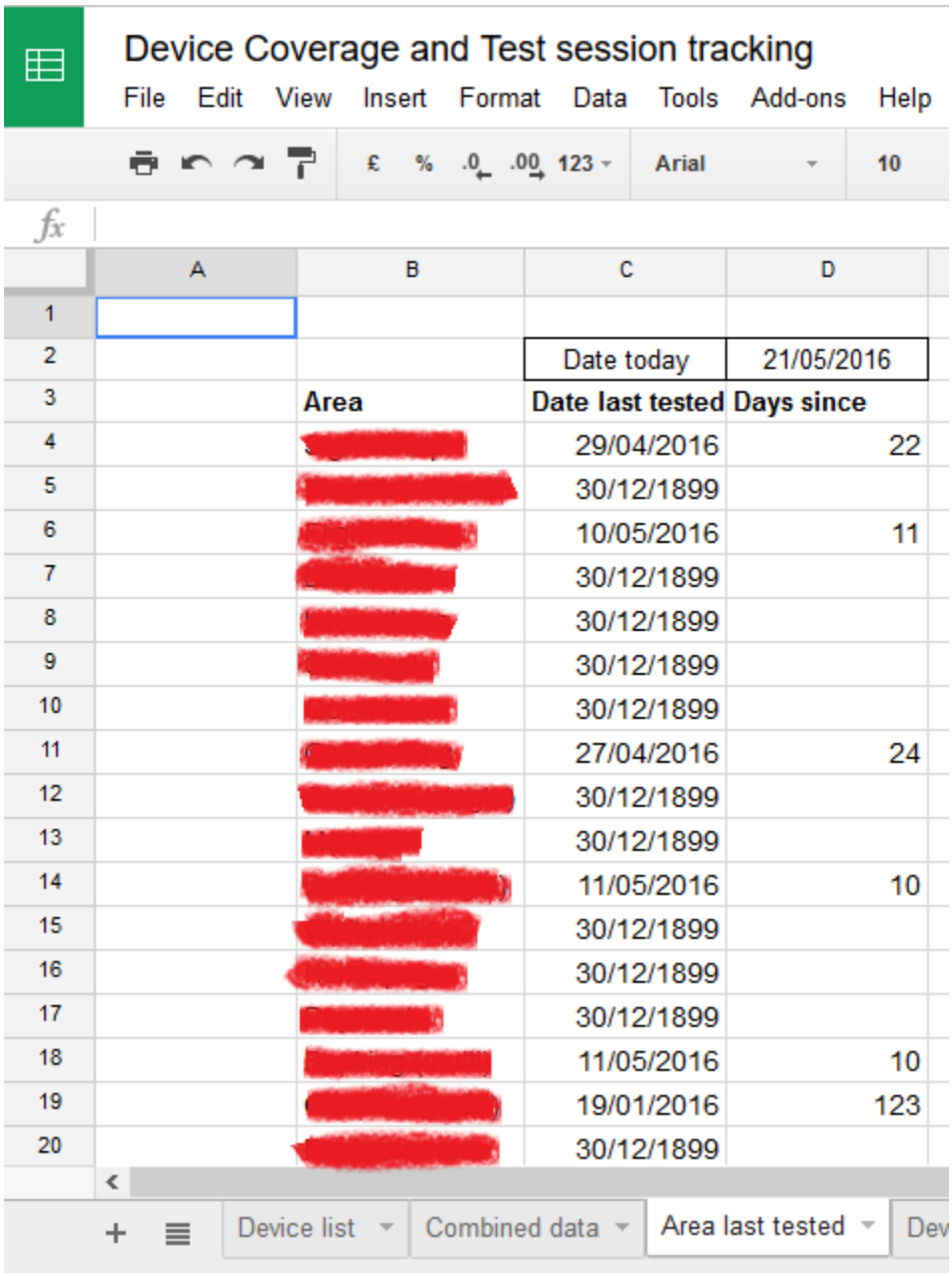
laine.viisitamm@fob-solutions.com

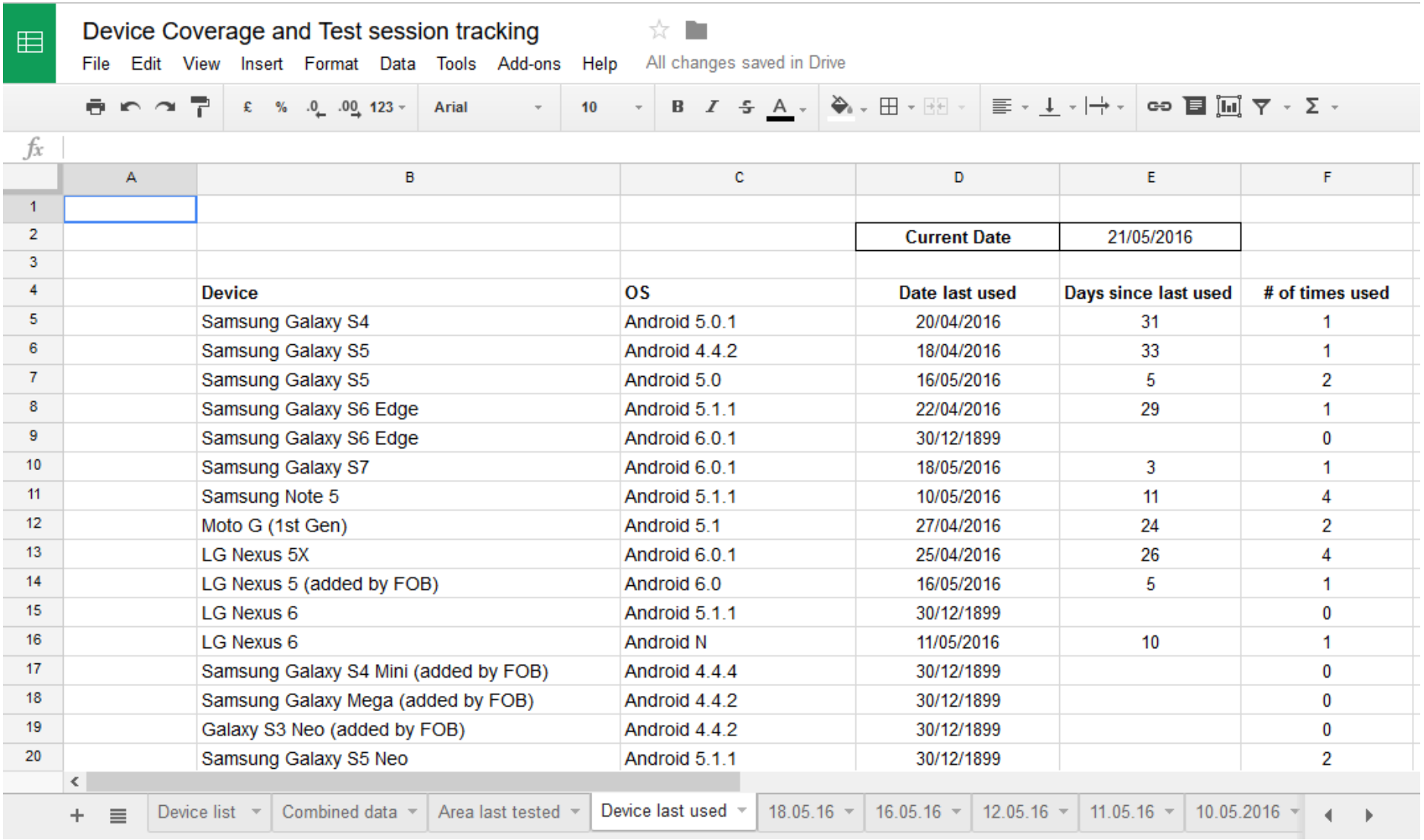
Comments Share

	A	B	C	D	E	F	G	H	I	J	K	L	M	N
1	Area	Tester	Device	OS Version	App Version	OS Family	Device type	OS Type	Screen size	Resolution	Date	Session Length	Bugs	Bugs
2	[redacted]	Laine	LG Nexus 5X	Android 6.0.1	0.8.15 (A)	Marshmallow	Phone	Android	5.2	1080 x 1920	12/04/2016	120	https://trello.com	https://tre
3	[redacted]	Laine	iPhone 6S	iOS 9.3	1.7.34 (iOS)	iOS 9	Phone	iOS	4.7	750 x 1334	12/04/2016	120	https://trello.com	https://tre
4	[redacted]	Laine	LG Nexus 5X	Android 6.0.1	0.8.15 (A)	Marshmallow	Phone	Android	5.2	1080 x 1920	12/04/2016	120	https://trello.com	https://tre
5	[redacted]	Laine	iPhone 6S	iOS 9.3	1.7.34 (iOS)	iOS 9	Phone	iOS	4.7	750 x 1334	12/04/2016	120	https://trello.com	https://tre
6	[redacted]	Laine	Samsung Note 5	Android 5.1.1	0.8.15 (A)	Lollipop	Phone	Android	5.7	1440 x 2560	13/04/2016	120	https://trello.com	https://tre
7	[redacted]	Laine	iPhone 5S	iOS 9.3.1	1.7.34 (iOS)	iOS 9	Phone	iOS	4	640 x 1136	13/04/2016	120	https://trello.com	https://tre
8	[redacted]	Sten	Samsung Note 5	Android 5.1.1		Lollipop	Phone	Android	5.7	1440 x 2560	19/01/2016	60		
9	[redacted]	Laine	iPhone 5S	iOS 8.3	1.7.45 (iOS)	iOS 8	Phone	iOS	4	640 x 1136	15/04/2016	120	https://trello.com	https://tre
10	[redacted]	Laine	Samsung Galaxy S5	Android 4.4.2	0.8.29 (A)	KitKat	Phone	Android	5.1	1080 x 1920	18/04/2016	120	https://trello.com	https://tre
11	[redacted]	Laine	iPhone 5C	iOS 9.2	1.7.49 (iOS)	iOS 9	Phone	iOS	4	640 x 1136	18/04/2016	120		
12	[redacted]	Laine	Samsung Galaxy S5	Android 5.0	0.8.29 (A)	Lollipop	Phone	Android	5.1	1080 x 1920	18/04/2016	120	https://trello.com	https://tre
13	[redacted]	Laine	iPhone 5C	iOS 9.2	1.7.49 (iOS)	iOS 9	Phone	iOS	4	640 x 1136	18/04/2016	120	https://trello.com	https://tre
14	[redacted]	Laine	Moto G (1st Gen)	Android 5.1	0.8.29 (A)	Lollipop	Phone	Android	4.5	720 x 1280	19/04/2016	120	https://trello.com	https://tre
15	[redacted]	Laine	iPhone 6	iOS 9.1	1.7.49 (iOS)	iOS 9	Phone	iOS	4.7	750 x 1334	19/04/2016	120	https://trello.com	https://tre
16	[redacted]	Laine	iPhone 5S	iOS 8.3	1.7.49 (iOS)	iOS 8	Phone	iOS	4	640 x 1136	20/04/2016	120	https://trello.com	
17	[redacted]	Laine	Samsung Galaxy S4	Android 5.0.1	0.9.2 (A)	Lollipop	Phone	Android	5	1080 x 1920	20/04/2016	120	https://trello.com	https://tre
18	[redacted]	Laine	Samsung Galaxy S6 Edg	Android 5.1.1	0.9.6 (A)	Lollipop	Phone	Android	5.1	1440 x 2560	22/04/2016	120	https://trello.com	https://tre
19	[redacted]	Laine	iPhone 6S Plus	iOS 9.0.2	1.9.13 (iOS)	iOS 9	Phone	iOS	5.5	1080 x 1920	22/04/2016	120	https://trello.com	https://tre
20	[redacted]	Laine	LG Nexus 5X	Android 6.0.1	0.9.6 (A)	Marshmallow	Phone	Android	5.2	1080 x 1920	25/04/2016	120	https://trello.com	https://tre
21	[redacted]	Laine	iPhone 6	iOS 8.4.1	1.9.13 (iOS)	iOS 8	Phone	iOS	4.7	750 x 1334	25/04/2016	120	https://trello.com	https://tre

Device list | Combined data | Area last tested | Device last used | 18.05.16 | 16.05.16 | 12.05.16 | 11.05.16 | 10.05.2016

Kolmandal ja neljandal lehel on andmed selle kohta, millal kasutati viimati mõnda piirkonda ning millal kasutati viimati mingit konkreetset seadet. Neist andmeist saab suunata testijat, millisele piirkonnale pole ammu tähelepanu pööratud ning millist seadet ei ole ammu kasutatud.





Device Coverage and Test session tracking

File Edit View Insert Format Data Tools Add-ons Help All changes saved in Drive

	Current Date	21/05/2016	
Device	**OS**	**Date last used**	**Days since last used**	**# of times used**
Samsung Galaxy S4	Android 5.0.1	20/04/2016	31	1
Samsung Galaxy S5	Android 4.4.2	18/04/2016	33	1
Samsung Galaxy S5	Android 5.0	16/05/2016	5	2
Samsung Galaxy S6 Edge	Android 5.1.1	22/04/2016	29	1
Samsung Galaxy S6 Edge	Android 6.0.1	30/12/1899		0
Samsung Galaxy S7	Android 6.0.1	18/05/2016	3	1
Samsung Note 5	Android 5.1.1	10/05/2016	11	4
Moto G (1st Gen)	Android 5.1	27/04/2016	24	2
LG Nexus 5X	Android 6.0.1	25/04/2016	26	4
LG Nexus 5 (added by FOB)	Android 6.0	16/05/2016	5	1
LG Nexus 6	Android 5.1.1	30/12/1899		0
LG Nexus 6	Android N	11/05/2016	10	1
Samsung Galaxy S4 Mini (added by FOB)	Android 4.4.4	30/12/1899		0
Samsung Galaxy Mega (added by FOB)	Android 4.4.2	30/12/1899		0
Galaxy S3 Neo (added by FOB)	Android 4.4.2	30/12/1899		0
Samsung Galaxy S5 Neo	Android 5.1.1	30/12/1899		2

Device list | Combined data | Area last tested | Device last used | 18.05.16 | 16.05.16 | 12.05.16 | 11.05.16 | 10.05.2016

Viimasena on sessiooni leht, mida testija kasutab sessiooni läbi viimiseks. Sessiooni leht võtab andmeid seadmete lehelt ja saadab läbiviidud sessiooni andmed hiljem *combined data* lehele. Sessiooni lehel on järgmised sisendid ja lahtrid, mida testija peab täitma:

- Tarkvara piirkond
- Testija nimi
- Seade
- Seadme operatsioonisüsteem
- Tarkvara versioon
- Kuupäev
- Sessiooni pikkus
- Vead
- Tegevused

Automatiseeritult tulevad ülejäänud andmed (OS perekond, seadme tüüp, operatsioonisüsteem, ekraani suurus ja seadme resolutsioon) seadme kohta, kui on ära valitud seade ja seadme operatsioonisüsteem.

	A	B
1	Area (TO BE SELECTED)	
2	Tester	
3	Device (TO BE SELECTED)	
4	OS version (TO BE SELECTED)	
5	App version (TO BE SELECTED)	
6	OS Family (AUTO-FILL)	
7	Device Type (AUTO-FILL)	
8	OS Type (AUTO-FILL)	
9	Screen size (AUTO-FILL)	
10	Resolution (AUTO-FILL)	
11	Date (TO BE SELECTED)	
12	Session length (TO BE SELECTED)	
13	Bugs	
14	Bugs	
15	Bugs	
16	Bugs	
17	Bugs	
18	Activities	
19	Findings	