Tallinna Ülikool

Digitehnoloogiate Instituut

Testimisprotsessi parendamine lõppkasutajate efektiivsema kaasamise läbi Statistikaameti näitel

Bakalaureusetöö

Autor: Andres Heiduk

Juhendaja: Inga Petuhhov

Autor:…..…… „ …… „2016

Juhendaja:……...…… „ …… „2016

Instituudi direktor:……..…… „ …… „2016

Tallinn 2016

Autorideklaratsioon

Kinnitan, et käesolev bakalaureusetöö on minu töö tulemus ja seda ei ole kellegi teise poolt varem kaitsmisele esitatud. Kõik töö koostamisel kasutatud teiste autorite tööd, olulised seisukohad, kirjandusallikatest ja mujalt pärinevad andmed on viidatud.

……. …….

(kuupäev) (autor)

Lihtlitsents lõputöö reprodutseerimiseks ja lõputöö üldsusele kättesaadavaks tegemiseks

Mina _____ (sünnikuupäev: _____)

(*autori nimi*)

1. annan Tallinna Ülikoolile tasuta loa (lihtlitsentsi) enda loodud teose

_____

_____

_____

(*lõputöö pealkiri*)

mille juhendaja on

_____,

(*juhendaja nimi*)

säilitamiseks ja üldsusele kättesaadavaks tegemiseks Tallinna Ülikooli Akadeemilise Raamatukogu repositooriumis.

2. olen teadlik, et punktis 1 nimetatud õigused jäävad alles ka autorile.

3. kinnitan, et lihtlitsentsi andmisega ei rikuta teiste isikute intellektuaalomandi ega isikuandmete kaitse seadusest tulenevaid õigusi.

Tallinnas, _____

(digitaalne) allkiri ja kuupäev

Sisukord

Sissejuhatus

Nähes, millise kiirusega toimub tehnoloogia areng ja märgates meid igapäevaselt ümbritsevate tehniliste seadmete hulga pidevat kasvu, on väga oluline, et tarbijateni jõuaks testitud, vigadeta töötav ja kvaliteetne tarkvara. Tarbijale on see kriteerium lausa iseenesest mõistetav, sest kes tahaks kulutada raha ebakvaliteetsele tootele. Oluline ei ole see aspekt mitte ainult kasutaja poolt vaadatuna, vaid ka seadme tarkvaratootjale endale. Ettevõtteid, kes tegelevad infotehnoloogia valdkonnas ja pakuvad erinevaid tarkvaraga seotud lahendusi või teenuseid, on palju. Sellest tulenevalt tekib väga tihe konkurents, kus ettevõtjad, nii suured kui ka väiksemad, tahavad oma klientuuri suurendada ja hästi läbi viidud testimine selle juures võib välja tuua puudused või vead süsteemis, millele ei osatud eelnevalt tähelepanu pöörata. See omakorda võib juba määrata lõppkasutajatele avalikustatava toote edukuse turul.

Alati ei piisa tarkvara testija teadmistest, et anda kinnitus töötavale funktsionaalsusele või rakenduse muule omadusele. Rakendus võib küll tööprotsessi läbi teha ilma vigadeta, aga see ei tähenda, et tööprotsess on õigesti läbitud. Veendumaks, et arendaja poolt loodud tarkvara või juba olemasolevale tarkvarale tellitud lisafunktsionaalsus töötab korrektselt, on vaja läbi viia tarkvara sisuline testimine. Hetkel Statistikaametil puudub testimise protsess, kus töö autor tarkvara testijana töötab, mis lõppkasutajaid süstemaatiliselt hõlmaks. Tihti peale on rakenduses testitavad äripoole protsessid väga keerulised ja seetõttu on lõppkasutajate testimisprotsessi kaasamise järele suur vajadus. Sisulise testimise käigus lõppkasutajad kinnitavad, kas tellitud tarkvara või selle komponent töötab nii, nagu äripool on seda ette näinud. See välistab näiliselt õigesti töötava, aga sisuliselt vigase rakenduse vastuvõtmise arendajalt.

Käesoleva bakalaureusetöö eesmärgiks on parendada testimisprotsessi Statistikaametis, kaasates selle jaoks efektiivsemalt lõppkasutajaid. Eesmärgi saavutamine on jaotatud kolme etappi. Esimeses etapis tutvustatakse erinevaid testimise võimalusi, mis on abiks olemasoleva Statistikaameti testimise protsessi kirjeldamiseks. Teises etapis kirjeldatakse ja määratakse hetkel Statistikaametis kasutusel olev testimise protsess koos protsessimudeliga, mis analüüsitakse ja mille käigus tuuakse välja protsessimudeli nõrgad küljed ja miks see mõjutab arendusprotsessi negatiivselt. Teises etapis analüüsitud ja selle tulemusel saadud järelduste põhjal kirjeldatakse kolmandas etapis parendatud testimisprotsess ning luuakse protsessimudel uuel kujul. Tuuakse välja, miks antud protsessimudel mõjub arendustööle positiivsemalt ning miks testimine sellisel kujul on tulemuslikum.

1 Tarkvara testimine

Järgnevas peatükis on esitatud tarkvara testimise definitsioon ja selle eesmärgid ning selgitatud, miks on vaja tarkvara testida. Samuti on välja toodud testimise põhimõtted, põhilised tasemed ja tüübid, mida arendusprotsessides järgitakse. Kasutatud on selleks ISTQB (*International Software Testing Qualifications Board*) organisatsiooni väljaannet testimise alustest ja põhitõdedest, millele ka edaspidi viidatakse. ISTQB näol on tegu 2002. aastal asutatud organisatsiooniga, mis nüüdseks on muutunud ülemaailmseks juhtfiguuriks tarkvara testimisega seotud suuna arendamisel.

1.1 Testimise mõiste ja eesmärgid

Tarkvara testimine on protsess, kus käivitatakse programm või komponent eesmärgiga leida selles vigu või defekte. Sellist tegevust võib kutsuda ka valideerimise protsessiks, kus veendutakse, et loodav tarkvaraline lahendus või programm vastab ärilistele ja tehnilistele nõuetele ning töötab ootuspäraselt. Põhjalikumal selgitamisel võib tarkvara testimise mõiste jagada kaheks osaks. Esimesel ehk järgneval juhul on esitatud definitsioon tarkvara testimisest, kui tegevusest (Graham, van Veenendaal, Evans, & Black, 2012):

1. **Protsess** – testimise puhul on tegu protsessi, mitte ühekordse tegevusega
2. **Kogu elutsükli tegevus** – testimine on protsess, mis toimub tsükliliselt läbi kogu tarkvaraarenduse protsessi. Testplaanide loomine juba arenduse varajases faasis ainuüksi analüüsi ja nõuete dokumentatsiooni põhjal võib aidata ära hoida võimalike vigade ja defektide sattumist komponenti või süsteemi. Selliste probleemide varane avastamine hoiab kokku oluliselt ressursikulu nii kliendile kui ka arendajale, sest arenduste hilisemas etapis osutub taoliste olukordade parandamine oluliselt kallimaks
3. **Staatiline testimine** – testimine, mis toimub ilma, et testitavat koodi käivitataks. Staatiline testimine toimub kontrollimise protsessis, mis hõlmab dokumentatsiooni ja lähtekoodi ülevaatamist
4. **Dünaamiline testimine** – dünaamilise testimise puhul käivitatakse lähtekood, et näha reaalseid tulemusi ja avastada võimalike vigu. Toimub andmete ja nende vormingu valideerimine, mis võib hõlmata näiteks komponendi testimist, integratsiooni testimist või süsteemi testimist

5. **Planeerimine** – peab planeerima, mida on vaja teha. Toimub testimisega seotud tegevuste juhtimine, vigade ja defektide raporteerimine ning ülevaate andmine testitava tarkvara seisundist
6. **Ettevalmistus** – peab otsustama, millist testi läbi viiakse. Selleks valitakse testimise tingimused ja moodustatakse testjuhud, mille põhjal hakatakse hiljem komponendi või süsteemi funktsionaalsuse käitumist testima
7. **Hindamine** – testitava tarkvara testimistulemuste hindamine ning nende võrdlemine testimise lõpetamise kriteeriumitega, mis aitab otsustada kas testimine on lõpuni viidud ja toode on läbinud kõik testid

Teine pool kirjeldab tarkvara testimise eesmärke ja põhjused ning miks tarkvara testimine vajalik on (Graham, van Veenendaal, Evans, & Black, 2012):

1. **Teha kindlaks, kas valminud tarkvaratoode rahuldab kirjeldatud nõuded.** Üks osa testimisest on suunatud kontrollimaks toote vastavust tehnilise kirjeldusega. Näiteks, vaadatakse üle, kas disain vastab nõuetele või kas kirjutatud lähtekood töötab nii nagu kirjeldatud. Kui toode vastab oma kirjeldatud tehnilistele nõuetele, on võimalik sellest teavitada võtmeisikud, kes omakorda saavad hinnata ja võtta vastu otsuse, kas toode on piisavalt kvaliteetne ning valmis kasutamiseks
2. **Näidata, et valminud tarkvaratoode täidab oma eesmärki.** See eesmärk erineb esimesest punktist vähesel määral, kuna kirjeldatud nõuded võivad olla valed või üldsegi täitmata. Antud testimise eesmärk on näha, kas tarkvara suudab täita oma ülesandeid piisavalt, et kasutajad saaks sellega läbi viia ettenähtud toiminguid. Ehk testimise käigus vaadatakse, kas tarkvara käitub nii, nagu kasutaja seda eeldab
3. **Vigade avastamine.** Antud testimise eesmärk on näidata võimalikke tarkvaratootega kaasnevaid riske, enne kui sellel lastakse minna kasutusesse – kas ja kui palju on vigu või takistusi tarkvaras, mis võivad ettenähtud tööprotsessi halvata. Samal ajal, aitab avastatud vigade parandamine tõsta ka toote kvaliteeti, mis omakorda maandab vigase tootega kaasnevaid riske. Vigade avastamisel on ka teine kasutegur – vea algpõhjuse analüüsi järeldustest on võimalik parendada arendusprotsessi ning seeläbi teha vähem vigu tulevikus. Arenduse lõpuetapis avastatud vigasid võib olla tunduvalt kulukam parandada ning seetõttu on varakult avastatud või üldse vigade tekke vältimine finantsiliselt ja ajaliselt soodsam nii arendajale, kui ka kliendile.

Välja toodud kaheosaline kirjeldus sobib iga taseme testimise juurde – alates komponendi testimisest kuni vastuvõtu testimiseni ning seda eeldusel, et arvestatakse erinevate testimise tasemete eesmärkidega. (Graham, van Veenendaal, Evans, & Black, 2012)

1.2 Testimise põhimõtted

Tarkvara testimine on loominguline ja intellektuaalselt väljakutsuv ülesanne. Kui testimisprotsessis järgitakse järgnevas loetelus toodud põhimõtteid, siis konkureerib see oma tähtsuselt mistahes tarkvara arendusetapi sammudega. (Ghahrai, 2008)

- Testimine näitab vigade olemasolu – mingi rakenduse või selle komponendi testimine võib küll esile tuua kindlad vead, aga see ei tähenda, et pärast leitud vigade parandamist on rakenduses need lõplikult elimineeritud. Sellest tulenevalt on tähtis, et loodaks testjuhtumeid, mis suudaks leida vigu võimalikult palju
- Ammendav testimine on võimatu – ainult juhul, kui testitav rakendus omab väga lihtsat loogilist struktuuri ja on piiratud sisendiga. Vastasel juhul on võimatu läbi testida kõikvõimalike kombinatsioone ja stsenaariume. Sel põhjusel on vajalik hinnata riske ja prioriteete, et keskenduda rakenduse kõige olulisematele aspektidele
- Varajane testimine – kahjuks on laialdaselt levinud käitumismuster, kus testimine lükatakse arendustsükli lõppu, mis võib luua edasilükkamisi toote üleandmises kasutajale või välja tuua takistavaid vigu rakenduses, mida on väga keeruline parandada. Seega, mida varem alustatakse testimisprotsessiga, seda paremini on võimalik ära kasutada olemasolevat aega. Varajast testimist saab alustada niipea, kui on valmis algne toode ning seda juba algse dokumentatsiooni kujul. Tänu sellele on leitud vigu võimalik parandada soodsamalt ja lihtsamalt
- Vigade kobar – testimise käigus on vaatluste põhjal võimalik näha, et enamus raporteeritavaid vigu viitavad kindlatele komponentidele süsteemis, ehk enamus vigu tulevad kindlatest kohtadest. Seda olukorda võib nimetatakse ka Pareto printsiibiks, kus ligikaudu 80% vigadest ilmnevad 20% rakenduse komponentides
- Pestitsiidide paradoks ehk immuunsus – kuna süsteem areneb pidevalt ja teadaolevad vead parandatakse ära, siis järjepidevalt ühtede ja samade testjuhtumite läbiviimisel on võimalus, et ühel hetkel ei suudeta enam defekte avastada. Igakord, kui paigaldatakse rakendusele parandustarne või lisandub uus funktsionaalsus, tuleb teha regressiooni

teste, veendumaks, et koos parandatud veaga ei ole tekkinud vigasid teistesse kohtadesse süsteemis. Samas tuleb rakenduse muutmise käigus regulaarselt üle vaadata ja muuta ka regressioontesti testjuhtumeid, et need oleks kohaldatavad ja suudaks võimalusel leida uusi vigu

- Testimine on kontekstipõhine – eritüüpi tarkvara või programme testitakse erinevatel meetoditel. Näiteks rakendus, mis oma töös käitleb delikaatset informatsiooni ja on kõrgete turvanõuetega, nõuab oluliselt põhjalikumat testimist, kui seda on tarvis mõnel mitteveebipõhisel arvutimängul
- Vigade puudumine soodustab eksikujutelma – kui testimise käigus ei leitud tarkvaras ühtegi viga, ei tähenda see veel, et toode on valmis kasutamiseks. Tuleb üle vaadata, kas loodud testplaanid ja –juhtumid olid suunatud leidmaks nii palju defekte, kui võimalik, või kas testjuhtumid loodi suunitlusega vastamaks kasutajapoolsetele nõuetele rakenduses

1.3 Testimise tasemed

Kogu arendus- ja hooldusprotsessi jooksul viiakse tarkvara testimine läbi kasutades tavaliselt mitut erinevat tasandit. Eraldiseisvaid tasemeid saab eristada lähtuvalt sellest, mis on läbiviidava testi eesmärk. Testimise eesmärk võib varieeruda, olenevalt vajadusest võidakse testida üksikut komponenti, integreeritud komponentide kogumit või hoopis tervikliku süsteemi. Testimise tasemed eristatakse eraldi kolme põhilise osana: ühiktestimine, integratsiooni testimine ja süsteemi testimine. Välja toodud kolm taset ei viita ühelegi protsessimudelile ja antud juhul ei ole üks tase teisest tähtsam või ebaolulisem. (Pierre & Fairley, 2014)

SWEBOK® (*Software Engineering Body of Knowledge*) puhul on tegu väljaandega, mis kirjeldab üheselt aktsepteeritavad arusaamad tarkvara arendusprotsessis. See on loodud koostöös mitmete professionaalsete kutseühingu liikmete ja tootjate vahel ning välja antud IEEE alamorganisatsiooni *IEEE Computer Society* poolt. Juhendit uuendatakse pidevalt ning 2013 aasta lõpus sai SWEBOK® V3 heakskiidu avaldamiseks. Samuti on SWEBOK® V3 tunnustatud rahvusvahelise ISO standardiga ISO/IEC TR 19759:2015 *Software Engineering – Guide to the software engineering body of knowledge (SWEBOK).*

Sarnaselt SWEBOK® V3 esitatud põhiliste testtasemetega, on järgmised kolm testimise taset esile toodud ka ISTQB organisatsiooni poolt:

Ühiktestimine. Tuntud ka komponendi testimisena, mille käigus tehakse kindlaks, kas mingi kindla funktsionaalsusega koodiosa terviklikkus lähtekoodis töötab ette nähtud kujul. Seda tüüpi teste viivad üldiselt läbi programmeerijad, kes on antud koodiosa kirjutanud, veendumaks, et spetsiifiline lõik koodist toimib nii nagu peab. Ühe funktsiooni jaoks võidakse kirjutada ka mitu testi, et välistada funktsiooni tõrked parandamatute juhtumite puhul. Olles läbi viinud ühiktestimise, ei tähenda see seda, et tarkvara on töökorras, kinnitatud saab ainult see, et kõik komponendid suudavad oma tööd teha üksteisest sõltumatult ja korrektselt. (Graham, van Veenendaal, Evans, & Black, 2012)

Integratsiooni testimine. Antud testimise puhul kontrollitakse koostöö tulemuslikkust erinevate tarkvarakomponentide vahel või loodud tarkvara võimet suhelda süsteemsete elementidega nagu operatsioonisüsteemi, failisüsteemi, riistvara ning liidestega süsteemis. Integratsiooni testimise tasemeid võib olla rohkem kui üks ning neid võidakse viia läbi erisuurusega testobjektide peal. (Graham, van Veenendaal, Evans, & Black, 2012)

Näiteid integratsiooni testimisest (Graham, van Veenendaal, Evans, & Black, 2012):

- Komponentide integratsiooni testimine, millega vaadatakse tarkvarasiseste komponentide omavahelist koostööd ja mida viiakse läbi alles pärast komponendi testimise etappi
- Süsteemi integratsiooni testimine, millega testitakse erinevate terviklike süsteemide või rakenduste omavahelist suhtlemist

Olenevalt sellest, kui suur on integreeritud süsteemi testimise skoop ehk ulatus, muutub keerulisemaks ka esile kerkivate vigade täpse põhjustaja kindlaks tegemine. See toob integratsiooni testimisele kaasa erinevaid lähenemisi, mis eelmainitud probleemi väldivad. Üheks äärmuslikuks variandiks on olukord, kus kõik süsteemi komponendid on integreeritud ühel ja samal ajal, pärast mida testitakse süsteemi juba tervikuna valmis kujul. Testimises nimetatakse seda „Suure Paugu“ teooriaks, kuna antud variandi kasutamine eeldab, et eelnevalt on kõik süsteemiga seotud komponendid kinnitatud valminuks, tänu millele ei ole testimisprotsessis tarvis kasutada simuleerivaid komponente, mis reaalsuses poleks veel programmeeritud. Negatiivne pool nimetatud variandi puhul on liigne ajakulu testimise läbiviimisel, sest süsteem tervikuna võib olla väga mahukas, mis omakorda teeb keeruliseks

veatekete allika tuvastamise. Lihtsustamaks integratsiooni testimist, tasub kaaluda teist äärmuslikku varianti – inkrementaalset ehk järk-järgult testimist. Inkrementaalse testimise puhul testitakse rakenduses valmivate komponentide omavahelist toimimist samm-haaval, mis lihtsustab vigade tekke allikate leidmist. Miinuseks siin juures see, et antud meetod võib samuti aeganõudev olla, sest valminud üksikute komponentide jaoks peab simuleerima teiste testimiseks vajalike osade olemasolu. (Graham, van Veenendaal, Evans, & Black, 2012)

Järk-järgult integratsiooni testimise siseselt on lisaks erinevaid meetodeid, sõltuvalt siis juba süsteemiarhitektuurist (Graham, van Veenendaal, Evans, & Black, 2012):

- Ülevalt-alla: testimine viiakse läbi ülalt alla, kus järgitakse süsteemiarhitektuuri kindlat töövoogu. Puuduvad komponendid või süsteemid on asendatud soovitud käitumist simuleerivate programmide poolt, mis annavad sisendi juba valmisolevale komponendile
- Alt-üles: testimine viiakse läbi alt üles poole, kus järgitakse samuti süsteemiarhitektuuri kindlat töövoogu. Puuduvad komponendid või süsteemid on asendatud juhtprogrammide poolt, mis simuleerivad puuduvate komponentide väljakutsumisi
- Funktsioonipõhine inkrementaalne testimine: komponentide integreerimine ja testimine on läbi viidud vastavalt sellele, kuidas on rakenduse funktsioonid nõuete spetsifikatsioonis kirjeldatud

Kõige mõistlikum oleks alustada integreerimist ja testimist nende komponentide vahel, mis teadaolevalt võivad hakata enim probleeme valmistama. Tänu sellele on võimalus, et integratsiooni testimise lõppfaasis esineb oluliselt vähem vigu. Järgides seejuures „Suure Paugu“ teooria asemel inkrementaalset testimise varianti, on võimalik vältida hiliste vigade tekkimise riske, kui testitakse juba terviklikult integreeritud süsteemi. (Graham, van Veenendaal, Evans, & Black, 2012)

Süsteemi testimine. Süsteemi testimise tasemesse on kaasatud kogu süsteemi, kui terviku käitumine. See võib sisaldada teste, mis põhinevad määratud riskidel ja/või rakenduse dokumentatsioonil, äriprotsessidel, kasutaja testjuhtumitel või teistel keerulisematel detailselt kirjeldatud testidel. Samuti võib esineda teste, mis vaatavad, kuidas ja kas valminud süsteem suudab ettenähtud kujul koostööd teha teiste süsteemsete elementidega vajalikus keskkonnas. Süsteemi testimise puhul on tegu peamise testtasemega, mida kasutatakse tarkvaraarenduse lõppfaasis. Veendutakse, et kliendile üleantav rakendus vastab dokumentatsioonis kirjeldatud

nõudmistele ja proovitakse leida veel tootes nii palju vigu, kui võimalik. Enamasti viiakse süsteemi testimine läbi spetsialiseerunud tarkvara testijate poolt, kes moodustavad arendustsüklis iseseisva testmeeskonna. Mõningates ettevõtetes või asutustes võidakse testimine läbi viia hoopis kolmandate osapoolte poolt, kelle teenus ostetakse eraldi sisse. (Graham, van Veenendaal, Evans, & Black, 2012)

Süsteemi testimise ajal tuleks läbi vaadata nii funktsionaalsed, kui ka mitte-funktsionaalsed rakenduse nõuded. Tüüpilisemad mitte-funktsionaalsed testid koosnevad jõudluse ja töökindluse katsetest, samuti võidakse kokku puutuda puudulike või dokumenteerimata nõuetega. Süsteemi funktsionaalsete nõuete testimine algab, valides dokumentatsiooni põhjal selleks kõige mõistlikumad ja asjakohasemad testimistehnikad (musta kasti meetod). Struktuuripõhiseid tehnikaid (valge kasti meetod) kaasatakse, kui on vajadus hinnata rakenduse mingisuguste elementide põhjalikkust, nagu näiteks avanevate menüüpuude struktuuri ja muude funktsioonide ülesehitust. Süsteemi testimine viiakse läbi arendaja vastutusel ning sellele vastavas testkeskkonnas. Testkeskkond peab olema võimalikult lähedases vastavuses selle keskkonnaga, kuhu hakatakse tarkvaratoodet valmis kujul paigaldama. Sellise testkeskkonna loomine aitab vältida konkreetseid keskkonna muudatustest tingitud võimalike vigade teket, mida vastasel juhul arendajapoolses testkeskkonnas ei pruugita avastada. (Graham, van Veenendaal, Evans, & Black, 2012)

Järgmine ning neljas testimise tase – vastuvõtu testimine, omab suurt tähtsust Statistikaameti olemasoleva testimisprotsessi kaardistamisel ning parendatava testimisprotsessi loomisel. Olulise ning neljanda testtasemena on see välja toodud ka ISTQB organisatsiooni poolt avaldatud raamatus ning SWEBOK® V3 kirjeldatud testtasemete all.

Vastuvõtu testimine. Kui arendaja poolt on süsteemi testimine läbi viidud ja kõik võimalikud vigade parandused on tehtud, tarnitakse valmis rakendus tellijale, et alustada kasutajapoolset vastuvõtu testimist. Vastuvõtu testimise käigus peab selgeks saama, kas tellitud tarkvara on võimalik kasutusele võtta ja kas, kui üldse, selle käigus kaasneb mingeid riske äriprotsessile. Testimise läbiviimiseks on vajalik testkeskkond, mis oma omaduste poolest on võimalikult sarnane tootekeskkonnaga, kuhu rakendus võimalusel hiljem paigaldatakse. Tootekeskkond on keskkond, kuhu rakendus paigaldatakse, kui leitakse, et see on valmis määratud sihtgrupile avalikuks kasutamiseks. (Graham, van Veenendaal, Evans, & Black, 2012)

Antud taseme eesmärgiks on luua kindlustunnet süsteemi, selle komponentide või kindlamate mitte-funktsionaalsete omaduste, nagu näiteks kasutatavuse suhtes. Enamasti keskendutakse

vastuvõtu testimise juures valideerimise tüüpi testimisele, kus põhirõhk ei ole rakenduses suunatud vigade leidmisele. Kuigi kirjeldatud testimise tase aitab suurel määral hinnata, kas rakendus on kasutajatele valmis avaldamiseks, siis ei pruugi olla ilmtingimata tegu viimase etapiga testimises. (Graham, van Veenendaal, Evans, & Black, 2012)

Vastuvõtu testimine jaguneb omakorda tasemeteks järgnevalt (Graham, van Veenendaal, Evans, & Black, 2012):

- Kasutajapoolne vastuvõtu testimine – põhirõhk on suunatud rakenduse funktsionaalsuse valideerimisele, kus äripoole kasutajad ja rakenduse haldur kinnitavad, kas tarkvara sisupool töötab eesmärgi kohaselt ning korrektselt
- Operatiivse valmisoleku testimine – valideeritakse, kas rakendus täidab nõuded administreerivate protseduuride tarvis. Enamasti viiakse läbi süsteemiadministraatorite poolt, kus vaadatakse, kas on võimalik edukalt läbi viia rakenduse varundamist, hooldusega seotud töid ja perioodilisi turvalisuse kontrolle ning kuidas tarkvara sellistes situatsioonides käitub
- Lepinguline vastuvõtu testimine – arendatud tarkvara testitakse kindlate kriteeriumite ja nõuete põhjal, mis olid enne töö alustamist lepingus märgitud
- Vastavuse testimine – vaadatakse, kas rakendus on kooskõlas määratud regulatsioonide ja õigusaktidega. Hõlmab näiteks avaliku sektoriga seotud teenuseid riigi valitsusalal või üheselt määratud standardsete kohustuslike ohutusnõuetega valdkondi

Kui süsteem on arendatud massidele kasutamiseks, mis puhul võib näiteks olla tegu kaubandusliku karbitootega, inglise keeles *COTS Software – Commercial Off The Shelf Software*, siis ei ole individuaalselt ja üksikute kasutajate poolt läbi viidud kasutaja vastuvõtu testimine enam praktiline. Vajalik on mastaapsem tagasiside juba olemasolevatelt või potentsiaalsetelt uutelt kasutajatelt enne, kui tarkvara läheb avalikku tootekeskkonda. Sellise süsteemi test jaotatakse tihti peale kaheks – alfa-testimine ja beeta-testimine. Alfa-testimine leiab aset arendaja keskkonnas, kuhu kutsutakse eelpool mainitud kasutajaid testimaks süsteemi. Arendajad seiravad kasutajad ning teevad märkmeid üles kerkivate probleemide puhul. Beeta-testimise puhul pakuvad arendajad kasutajatele loodud tarkvara allalaadimiseks või võrgus kasutamiseks, aga seda juba tegelikku töötamise käigus. Selle käigus kasutajad saadavad andmeid esile kerkivate probleemide kohta, mis omakorda tootja poolt analüüsitakse ja parandatakse. (Graham, van Veenendaal, Evans, & Black, 2012)

1.4 Testimise tüübid

Testimise tüübid võib võtta, kui kitsama suunitlusega meetodeid vastava testtaseme protsessi läbi viimiseks. Testimise tüüpe on mitu tükki ja erinevaid, sest ainuüksi funktsionaalsuse testimise rakendamine kas mingi komponendi või süsteemi peal ei pruugi olla piisav iga taseme jaoks, et katta ära kogu testimise skoobi eesmärk. Fokusseerides planeeritavas testimises kindlale eesmärgile, samal ajal valides selleks sobiva testimise tüübi, aitab see lihtsamini ja täpsemalt testitava objekti suhtes otsuseid vastu võtta ning neid vajadusel edastada. (Graham, van Veenendaal, Evans, & Black, 2012)

Testimise tüübi eesmärgiks on keskenduda ühele konkreetsele ülesandele, milleks võib olla (Graham, van Veenendaal, Evans, & Black, 2012):

- kindla funktsiooni testimine komponendis;
- mitte-funktsionaalse omaduse, nagu töökindluse ja kasutajamugavuse testimine;
- struktuuri ja arhitektuuri testimine rakenduse komponendis või süsteemis;
- muudatustega seotud testimine, kinnitamaks tehtud vigade paranduste toimimist;
- tahtmatute vigade otsimine (regressioontestimine).

Kuna testimise tüüpe on palju, siis on need jaotatud nelja põhilise testimise tüübi alla (Graham, van Veenendaal, Evans, & Black, 2012):

- funktsionaalne testimine;
- mitte-funktsionaalne testimine;
- struktuuripõhine testimine;
- muudatustega seotud testimine.

Funktsionaalne testimine – testimise tüüp, mis viitab kindlale tegevusele või funktsioonile rakenduses. Vaadatakse, kas teatud funktsionaalsus toimib nii, nagu see on dokumentatsioonis või testjuhtumis kirjeldatud. Funktsionaalsuse testi saab läbi viia kahel kujul: kirjeldatud tehniliste nõuete põhjal või äripoole protsesside alusel. Esimesel juhul tuvastatakse kriitilised funktsionaalsused ja selle põhjal luuakse ja rakendatakse testjuhtumid. See kindlustab kõige olulisemate testide läbiviimise süsteemis, enne kui suundutakse kõrvalisemate tegevuste ülevaatamisele, mida muidu võib-olla ei saakski testida. Teisel juhul koostatakse teststsenaariumid igapäevaselt läbitavate äripoole protsesside kohta, veendumaks, et rakendus teeb oma tööd ka sisuliselt õigesti. (Graham, van Veenendaal, Evans, & Black, 2012)

Mitte-funktsionaalne testimine – antud testimise eesmärk on veenduda süsteemi või selle komponentide kvaliteedi ehk mitte-funktsionaalsetes omadustes. Vaadatakse, kui hästi või kiiresti midagi tehtud saab ja seda omadust võidakse mõõta näiteks ajaliselt. Rahvusvaheline standardite organisatsioon ISO on defineerinud kindla kogumi rakenduse kvaliteedi omadustest (ISO/IEC 25010:2011). Nende määratlemine peegeldab suurt sammu konsensuse poole IT valdkonnas, mis käsitleb üldised mõisted tarkvara kvaliteedi omaduste kohta. Antud standard on saamas üha rohkem tunnustust ettevõtete poolt, mis suunab neid tarkvaraarendamisel ja selle testimisel kasutama ühtseid termineid kvaliteedi omaduste puhul ja seeläbi ka mitte-funktsionaalse testimise tüübi juures. (Graham, van Veenendaal, Evans, & Black, 2012)

Eelmainitud standardi ISO/IEC 25010:2011 väljaandes on kvaliteedinäitajaid välja toodud kokku kaheksa (International Organization for Standardization and International Electrotechnical Commission, 2011):

- **Funktsionaalne sobivus**, mis koosneb kolmest alamtüübist: funktsionaalne täiuslikkus, funktsionaalne õigsus ja funktsionaalne vastavus
- **Usaldusväärsus**, mis koosneb neljast alamtüübist: valmidus, kättesaadavus, rikkekindlus ja taastatavus
- **Kasutatavus**, mis koosneb kuuest alamtüübist: arusaadavus, õpitavus, operatiivsus, kasutajapoolse veatekke välditavus, kasutajaliidese esteetika ja juurdepääsetavus
- **Efektiivsus**, mis on koosneb kolmest alamtüübist: jõudlus, ressursside kasutamine ja mahutavus või läbilaskevõime
- **Hooldatavus**, mis koosneb viiest alamtüübist: modulaarsus, taaskasutatavus, analüüsitavus, muudetavus ja testitavus
- **Teisaldatavus**, mis koosneb kolmest alamtüübist: kohanemisvõime, paigaldatavus ja asendatavus
- **Turvalisus**, mis koosneb viiest alamtüübist: konfidentsiaalsus, terviklikkus, tunnustatavus, vastutus ja usaldatavus
- **Ühilduvus**, mis koosneb kahest alamtüübist: kooseksisteerimine ja koostalitusvõime

Struktuuripõhine testimine – testimise tüüp, milles kasutatakse peamiselt valge kasti meetodit, sest testijal on tarvis näha, mis toimub funktsiooni tegevuse ajal selle sees, mitte ainult visuaalses kasutajaliideses. Et struktuuri testimist edukalt läbi viia, tuleb testijal omandada teadmised rakenduse lähtekoodist, et seejärel tuvastada, kuidas tarkvara on

arendajate poolt rakendatud ja toimima pandud. (Graham, van Veenendaal, Evans, & Black, 2012)

Muudatustest tingitud testimine – kui arendatavas tarkvaras avastatakse viga, siis selle kohta tehakse eelnevalt kokkulepitud keskkonnas vastav teavitus, misjärel viib arendaja rakenduses tekkinud vea suhtes sisse paranduse ning see paigaldatakse uuesti rakenduse testkeskkonda. Kuna veaparandusega on süsteemi sisse viidud muudatus, tuleb testijal antud funktsioon uuesti läbi testida, et kinnitada veaparandamise tulemuslikkus. (Graham, van Veenendaal, Evans, & Black, 2012)

2 Testimisprotsess Statistikaametis

Järgnevas peatükis on toodud ülevaade Statistikaametist, testimisprotsessist Statistikaametis ning mille juurde on lisatud ka joonis hetkel kasutusel olevast protsessimudelist koos kirjeldusega. Samuti on välja toodud olemasoleva protsessi mõju arendustele.

2.1 Ülevaade Statistikaametist

Statistikaamet on riigiasutus rahandusministeeriumi haldusalas, kelle põhiülesandeks on pakkuda objektiivset ja usaldusväärset informatsiooni Eesti keskkonnas esinevate trendide kohta (Eesti Statistika, 2016).

Et statistilised andmed jõuaks vajajateni, on Statistikaameti siseselt tarvis teha statistikatöid, läbi mille jõuavad andmed väljundandmebaasi. Statistiliste andmete käitlemine toimub digitaalselt ning, et Statistikaameti töö sujuks mugavamalt, toimub kasutusel olevate rakenduste uuendamine ja parenduste arendamine. Võib esineda ka suuremaid üleminekuid mingilt vanemalt tarkvaralt uuemale, aga seda erandkorras, regulaarselt selliseid toiminguid ei tehta. Süsteemisisene arendustöö ehk rakendustes tehtavad muudatused ja parendused toimuvad majaväliselt, kus Statistikaamet on klient ja arendustöid viivad ettevõtted läbi allhankena.

Üldine arendusprotsess ei saa toimuda ilma loodud tarkvara testimiseta ning selline kohustus on ka Statistikaametis. Statistikaametis töötab kaks tarkvara testijat, kaasa arvatud autor, kelle ülesandeks on majaväliste ettevõtete poolt arendatud tarkvara esmane testimine. Esmane ülevaatus ja testimine toimub Statistikaameti rakenduse testkeskkonnas. Testimise käigus veendutakse, et loodud või parendatud tarkvara funktsionaalsed ja mittefunktsionaalsed omadused on kooskõlas analüüsis kirjeldatuga või selles tehtud muudatustega. Tarkvara testijate otsusest sõltub, kas rakenduse uus või parendatud versioon on piisavalt hea Statistikaameti lõppkasutajate jaoks tootekeskkonda paigaldamiseks või mitte. Enne tootekeskkonda jõudmist paigaldatakse rakendus eeltootekeskkonda, mis on olemuselt identne tootekeskkonnaga, aga kus lõppkasutajad ei tee veel tegelikku tööd. Eeltootekeskkonna ja testkeskkonna erinevus seisneb andmete mahus, kus viimases on neid oluliselt vähem, mistõttu mõningaid andmemahust sõltuvaid veaparandusi ei ole võimalik alati Statistikaameti testijatel testkeskkonnas üle kontrollida.

Rakenduste arendamise puhul on Statistikaametis tegu tavaliselt kas mingisuguste alamfunktsionaalsuste lisamisega kindlale komponendile rakenduses või täiesti uue komponendi arendamisega tervele süsteemile. Kuna arendustööd tellitakse eraldiseisvatelt ettevõtetelt, ei ole võimalik testimisprotsessiga alustada juba tarkvara arendamise käigus. Statistikaameti tarkvara testijatel tuleb läbi viia funktsionaalne testimine ja vastuvõtu testimine alles pärast seda, kui lisafunktsionaalsus või komponent on valmis kujul testkeskkonda paigaldatud ja eeldatavasti arendaja poolt juba esmase testimise läbinud. Kui arendaja töökeskkonnas läbis loodud või muudetud tarkvara testimisprotsessi edukalt, ei tähenda see seda, et samamoodi läheb ka Statistikaameti poolt määratud töökeskkondades. Samasugune olukord kehtib ka rakenduses parandatud vigade puhul. Kui eelnevalt leitud veale on arendaja poolt tehtud parandused, siis tuleb antud parandatud funktsionaalsus või komponent uuesti läbi testida, veendumaks, et tehtud muudatused toimivad ja on õiged.

2.2 Kasutuses olev testimisprotsess

Käesolevas peatükis on etappide kaupa kirjeldatud, milline on praegu testimise protsess Statistikaametis. Olemasolev protsess algab siis, kui arendajal on seoses uue funktsionaalsuse tellimuse valmimisega või raporteeritud ja parandatud vigadega parendatud versioon rakendusest valmis.

Hetkel Statistikaametis kasutusel olev testimise protsess on jaotatud kokku seitsmesse erinevasse etappi ning seda järgmises järjekorras:

1. Arendajal on rakenduse uus versioon valmis ja rakenduse omanik Statistikaametist ning teenusepakkuja RMIT (Rahandusministeeriumi Infotehnoloogiakeskus) on teavitatud

Kui arendaja on valmis saanud rakenduse uue versiooni, mis võib sisaldada uusi funktsionaalsusi või kindlat arvu parandatud vigu, teavitatakse sellest vastava rakenduse toimimise eest vastutavat isikut Statistikaametis ehk rakenduse omanikku ja rakenduste haldusteenusepakkujat RMIT, kes otsustavad millal uus pake paigaldatakse.

2. Rakenduse omanik tellib teenusepakkujalt RMIT rakenduse versiooni paigaldamist Statistikaameti testkeskkonda

Pärast uue versiooni märkmetega tutvumist analüüsitakse, kui kriitilise tähtsusega on selles sisalduvate vigade parandused või uued funktsionaalsused igapäevase töö tegemiseks. Olles eelnevalt kirjeldatu suhtes vastu võtnud otsuse, tellitakse teenusepakkuja käest rakenduse uusima versiooni paigaldus Statistikaameti vastavasse testkeskkonda.

3. RMIT paigaldab rakenduse uue versiooni ning teavitab sellest Statistikaametit (rakenduse omanik, testijad)

Pärast rakenduse uue versiooni edukat paigaldust, teavitab teenusepakkuja RMIT nii rakenduse omanikku kui ka Statistikaameti testijaid sellest e-kirja teel. Rakenduse uue versiooni esmane paigaldus tehakse Statistikaameti vastavasse testkeskkonda.

4. Testija kontrollib rakenduse uue versiooni märkmetest, kas see sisaldab ainult vigade parandusi või on selles ka uusi funktsionaalsusi

Uue tarne paigaldus võib sisaldada ainult vigade parandusi, kuid ka uusi funktsionaalsusi rakenduse komponentides. Kuna uue funktsionaalsuse puhul puuduvad ülesanded või veateated raporteerimise veebikeskkonnas JIRA, siis uues funktsonaalsuses sisalduvate probleemide korral tuleb vajadusel vastavad ülesanded alles luua.

5. Kui tarnes on ainult uus funktsionaalsus, siis testija vaatab selle üle ja raporteerib leitud vea või probleemid JIRAs ülesandena arendaja nimele

Seotud neljanda sammuga, kus uues funktsionaalsuses sisalduvate võimalike probleemide esile kerkimisel on tarvis luua uued ülesanded. Kui arendaja kinnitab, et viga või probleem on nende poolne, mitte seotud rakenduse väära kasutamisega lõppkasutaja poolt, võetakse JIRAs Statistikaameti testija koostatud ülesanne tööplaani. Konkreetse vea või probleemide puhul liigub Statistikaameti jaoks testimise protsessi töötsükkel taas esimesse etappi.

6. Kui tarnes on olemas parandused, siis testija vaatab need üle ja haldab parandustega seotud ülesanded JIRAs

Seotud samuti neljanda sammuga protsessis, kus lisaks uute võimalike probleemide kohta ülesannete loomisel JIRAs, on tarvilik hallata ka juba olemasolevaid probleeme. Rakenduse uues versioonis parandatud vigade taasesinemise korral suunatakse JIRA veateade tagasi

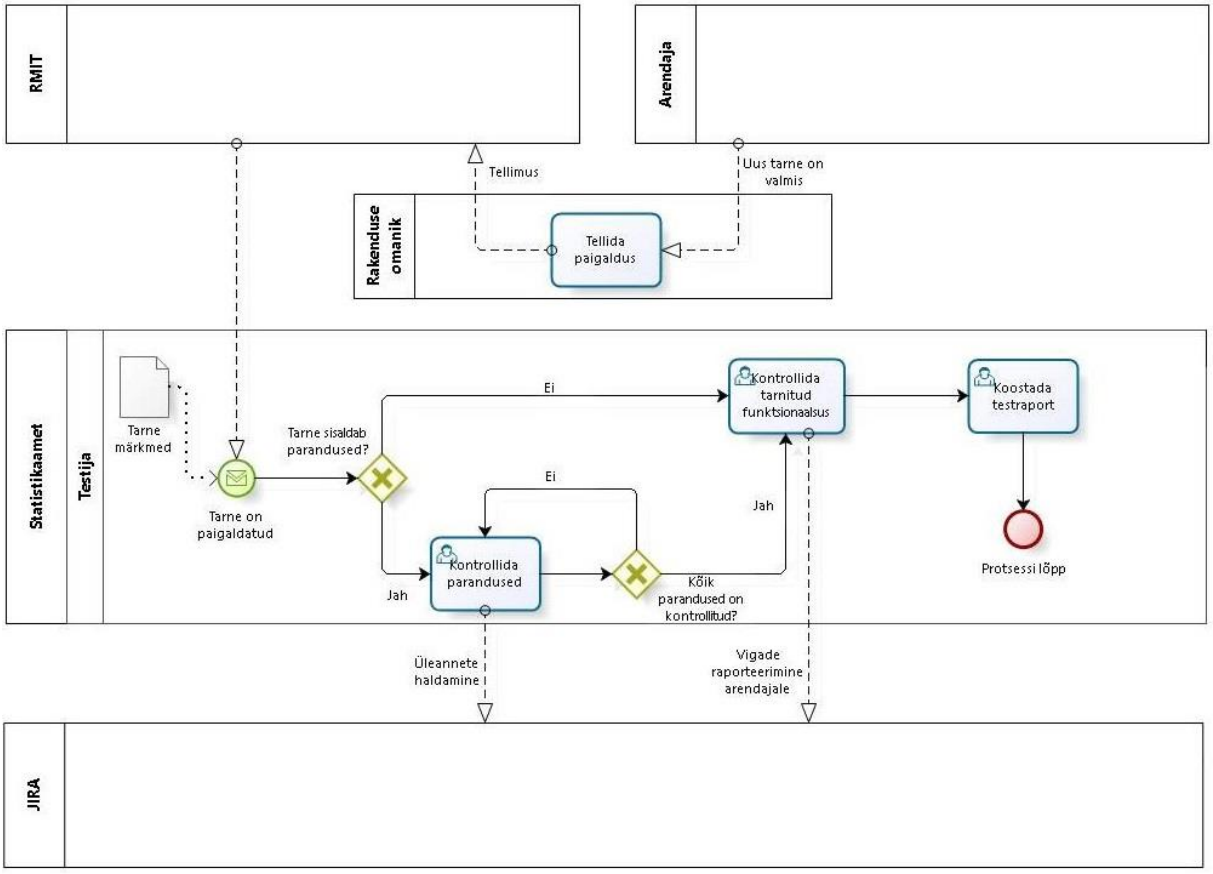
arendajale, kes lisab ülesande uuesti oma tööplaani. Jätkuvalt esinevate vigade puhul liigub Statistikaameti jaoks testimisprotsessi töötsükkel tagasi esimesse etappi.

7. Kui funktsionaalne testimine ja vastuvõtu testimine Statistikaameti testkeskkonnas on lõppenud edukalt, paigaldatakse rakenduse uus versioon eeltootekeskkonda ja tootekeskkonda

Kui testkeskkonda paigaldatud rakenduse uue versiooniga seotud funktsioonid ja muudatused on testitud ja eelnevalt saadud vigade asemel käitub rakendus ootuspäraselt, tellitakse teenusepakkuja RMIT käest rakenduse uue versiooni paigaldus Statistikaameti eeltoote- ja tootekeskkonda. Pärast seda jõuavad arendaja poolt tehtud lisafunktsioonid ja/või muudatused süsteemis lõppkasutajateni.

2.3 Protsessimudel

Joonisel 1 on kujutatud Statistikaameti testimisprotsessi BPMN mudel alates hetkest, kui arendaja on valmis saanud uue versiooni rakendusest. Eristatavad on kolm põhilist osapoolt: teenusepakkuja RMIT, arendaja ning Statistikaamet. Rakenduse omanik kuulub samuti Statistikaameti alla, aga protsessi sisu arusaadavamaks visualiseerimiseks on teda joonisel kujutatud eraldi. Veebirakendus JIRA ei ole joonisel osapoolena, vaid täidab keskset sisemiste ülesannete haldamise keskkonna rolli, olles abiks kogu protsessi läbiviimisel RMITi, arendaja ja Statistikaameti poolt.



Joonis 1 – Statistikaameti kasutusel olev testimise BPMN protsessimudel

2.4 Protsessi mõju arendusele

Kahes eelmises peatükis kirjeldatud ning hetkel Statistikaametis kasutatava testimisprotsessi puhul on lõppkasutajatel vähene roll uute arenduste või veaparanduste testimisel. Testimisprotsess lõpeb seitsmenda punktiga, kus testijad otsustavad, kas paigaldatud rakenduse uus funktsionaalsus või veaparandused töötavad. Lõppkasutajate jaoks on uute funktsionaalsuste või parandatud vigade mõjud tuntavad alles siis, kui rakenduse uus versioon on jõudnud tootekeskkonda, kus juba toimub ka reaalne töö tegemine. Sellist protsessi järgides jõuavad tehtud arendused kiiresti tootekeskkonda.

Eelnevalt kirjeldatud protsessi järgides võib tunduda, et vead parandatakse kiiresti ja uued funktsionaalsused toimivad nendele seatud eesmärkide kohaselt. Küll aga võib oma funktsionaalsusi täitev komponent töötada sisuliselt valesti. Lihtsa näitena saab tuua olukorra, kus arendaja võib funktsionaalsuse eesmärgist valesti aru saada ning koostatud testjuhtum või lisakirjeldus dokumentatsioonis on juba algusest peale vigane. Sellest tulenevalt, kui testija testib mingit funktsionaalsust, mis annab kindla väljundi, võib esmapilgul tunduda, et uus funktsionaalsus või parandatud viga on testi läbinud. Küll aga äripoolelt vaadatuna võib sama funktsionaalsus täita vale eesmärki, anda vigase väljundi ning seeläbi luua ka rakenduse muudes kohtades potentsiaalseid uusi vigu. See omakorda tekitab kasutajate jaoks vigu ja probleeme, millele reageeritakse enneaegselt, ilma, et saadaks aru, kus vea põhjus tegelikult asub.

Seega, olemasoleva protsessiga kaasnevad Statistikaameti rakendustes järgmised probleemid:

- näiliselt töökorras rakendus, mis täidab vale eesmärki lõppkasutaja jaoks;
- tootekeskkonda paigaldatakse tarkvara, mis sisaldab vigu;
- lõppkasutajate töö on tootekeskkonnas rohkete vigade tõttu häiritud;
- lõppkasutajatel kulub häiritud tööprotsessi tõttu rohkem aega töötegemisele.

3 Statistikaameti parendatud testimisprotsess

Järgnevas peatükis on kirjeldatud esmalt Statistikaameti parendatud testimise protsessi, mille kohta on tehtud muudatused Statistikaameti arenduste töökorras ning mis on hetkel asutuse siseselt juurutamises. Sellele järgneb parendatud testimise BPMN protsessimudel ning viimasena on toodud välja, miks loodud mudel on Statistikaametile ja selle igapäeva töös rakendusi kasutavatele lõppkasutajatele parem ja efektiivsem.

3.1 Parendatud protsess

Uuendatava ja parendatud testimise protsessi puhul jäävad samaks hetkel veel kasutusel oleva protsessi esimesed 6 punkti, mistõttu ei kirjeldata neid eraldi allolevas loetelus. Muudatusi on kasutuses olevas protsessis tehtud alates seitsmendast sammust ning täiendavalt on juurde lisatud veel neli etappi. Muudetud ja lisatud etappide eesmärk on paremini kaasata lõppkasutajaid testimise protsessi, kuna sellel on otsene mõju nende töövahenditele.

Parendatud protsessi muudetud seitsmes etapp ja neli lisandunud etappi:

7. Kui funktsionaalne ja mitte-funktsionaalne testimine Statistikaameti testkeskkonnas on lõppenud edukalt, siis testija teavitab rakenduse omanikku ja rakenduse kasutajaid

Statistikaameti testijad on uute arenduste või parandatud vigade osas funktsionaalse ja mitte-funktsionaalse testimise lugenud lõppenuks, kus kõik funktsionaalsused toimisid vastavalt kaasa tulnud testjuhtumitele ja/või kirjeldatud kasutusloole. Järgnevalt teavitatakse sellest rakenduse omanikku ja rakenduse lõppkasutajaid.

8. Rakenduse omanik tellib teenusepakkuja RMIT käest rakenduse uue versiooni paigaldamise Statistikaameti testkeskkonnast Statistikaameti eeltootekeskkonda

Kui rakenduse omanik on otsustanud, millal on sobiv rakenduse eeltootekeskkonnas teha katkestus, siis teavitatakse sellest teenusepakkujat ning uus versioon paigaldatakse Statistikaameti eeltootekeskkonda.

9. Äripool valmistab oma testimise stsenaariumid ja nende järgi viiakse läbi kasutajapoolne testimine Statistikaameti eeltootekeskkonnas

Äripool, kuhu kuuluvad rakenduse omanik ja rakenduse lõppkasutajad, saavad Statistikaameti testijatelt teate, et tellitud uus funktsionaalsus või parandatud vead esialgsel testimisel töötavad. Sellele järgnevalt moodustatakse sisulist poolt hõlmavad testjuhtumid, mis kontrollivad, kas rakendus töötab ka sisuliselt õigesti. Lõppkasutajad viivad rakenduses läbi testjuhtumites kirjeldatud kasutuslood.

10. Kui kasutajapoolse testimise käigus on avastatud vigu või probleeme, siis kasutades vigade raporteerimise juhendit, kasutaja raporteerib vead JIRAs testija nimele

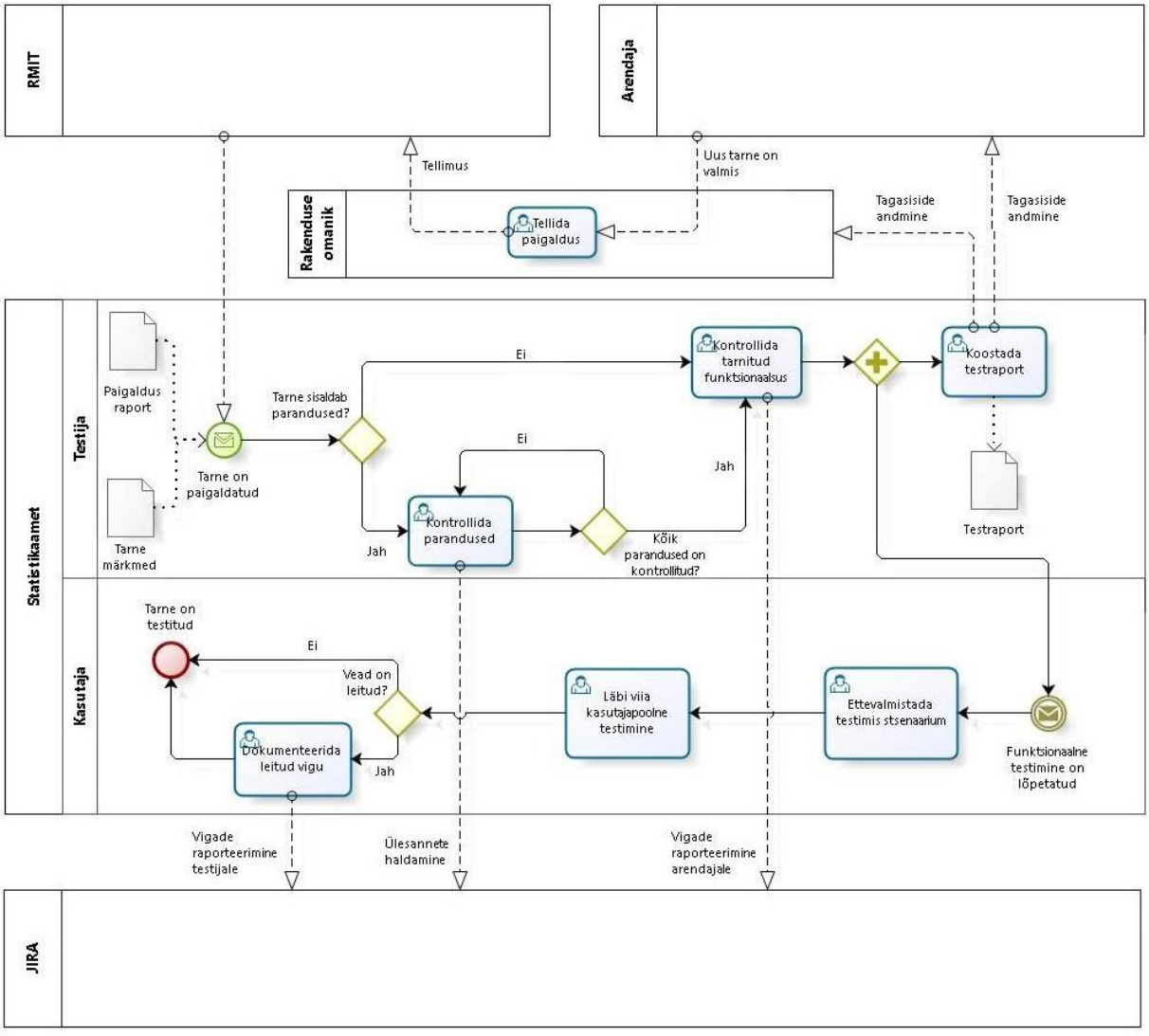
Kasutajapoolse testimise käigus võivad lõppkasutajad leida vigu või probleeme, mida Statistikaameti testijad ei avastanud või kerkib esile sisulisi vigu, millele ei osatud tähelepanu pöörata. Sellise olukorra puhul on tarvis moodustada raporteerimise veebikeskkonnas JIRA vastav ülesanne, kus on detailne kirjeldus probleemi kohta. See suunatakse edasi Statistikaameti testijale, kes kontrollib kirjeldatud situatsiooni täpsemalt ja kinnitab või lükkab ümber probleemi olemasolu, vajadusel suunab selle edasi arendaja nimele.

11. Kui rakenduse uues versioonis sisalduv funktsionaalsus ja/või kõik parandused on Statistikaameti testijate ja äripoole poolt testitud, võib arvestada rakenduses arendatud uue funktsionaalsuse või parandatud vigade testi lõppenuks

Kui rakenduse uus versioon on testitud ning testimise tulemus on testijate, rakenduse omaniku ja lõppkasutajate jaoks positiivne, tellitakse teenusepakkujalt RMIT rakenduse uue versiooni paigaldus Statistikaameti tootekeskkonda.

3.2 Parendatud protsessimudel

Joonisel 2 on modelleeritud Statistikaameti testimisprotsessi BPMN mudel parendatud kujul. Joonisel on näha kolme põhilist osapoolt: teenusepakkuja RMIT, arendaja ning Statistikaamet. Testimise protsessi jaoks oluline muudatus on toimunud Statistikaameti poole peal, kus lisaks Statistikaameti testijale ja rakenduse omanikule on nüüd protsessi kaasatud ka lõppkasutaja. Lõppkasutaja omab koos Statistikaameti testijaga võrdväärse osatähtsusega rolli tarkvara vastuvõtu testimisel. Samuti on lisandunud lõppkasutaja poolt sisendi andmine JIRA kaudu, mis ühendab kõiki osapooli ning on keskseks halduskeskkonnaks sisevõrgus.



Joonis 2 – Statistikaameti parendatud testimise BPMN protsessimudel

3.3 Parendatud protsessi mõju arendusele

Uuendatud testimise protsess seob läbi vastuvõtu testimise lõppkasutajad oluliselt rohkem rakenduste uute versioonide testimisega. Lisatud funktsioonide keerukusest ja mahust või parandatud vigade arvust sõltub testimise ulatus ning sellele pühendatud ajakulu. See omakorda, võrreldes vana protsessiga, aeglustab teatud määral rakenduste versiooniuuenduste jõudmist tootekeskkonda. Teisalt aga, ja mis kõige olulisem, kaalub tervikliku ja täiuslikumas töökorras oleva tarkvara jõudmine tootekeskkonda oluliselt üle lisandunud ajakulu lõppkasutajate poolt läbi viidud vastuvõtu testimisel. See omakorda annab stabiilsust süsteemide töökindlusele ning potentsiaalset efektiivsust lõppkasutajate poolt tehtavale tööle ja selle tulemuslikkusele.

Statistikaameti rakenduste tootekeskkondades tuleb esile ka veel praegu kriitilisi või takistavaid vigu, mis aeglustavad oluliselt tööprotsessi või, mis ei luba töös kasutada olulisi funktsioone. Seetõttu on kasutajad sunnitud töö tegemiseks välja mõtlema erinevaid meetodeid, mistõttu kulub neil töö tegemiseks rohkem aega. Halvimal juhul peavad lõppkasutajad ootama seni, kuni probleemi põhjus saab välja selgitatud ning arendaja poolt parandatud. Rakenduse vigu sisaldava parandusversiooni ootamisele kuluv aeg võib olla lõpuks kordades suurem, kui see aeg, mis kulub lõppkasutajatel eeltootekeskkonnas vastuvõtu testimisele. Tänu lõppkasutajate rolli suurendamisele vastuvõtu testimise läbi, on eelnevalt mainitud probleeme ja takistusi võimalik tootekeskkondade igapäevatöös vältida. Suureneb võimekus avastada kriitilisi või takistavaid vigu juba eeltootekeskkonnas, mis tootekeskkonda jõudes võivad tekitada töös palju rohkem häireid, kui neid enne rakenduse uue või parandatud versiooni paigaldamist oli.

Uuendatud testimise protsessi juurutamiseks on Statistikaametis sisse viidud muudatusi arenduste töökorralduses. Eelnevalt kasutusel olnud testimise protsessi kirjeldav dokument on nüüdseks Statistikaameti kahe tarkvara testija, kellest üks on autor, poolt asendunud uuendatud testimise protsessi kirjelduse ja BPMN mudeliga. Samuti on kõikidele Statistikaameti rakendustele määratud kindlad omanikud, kelle poole pöördutakse rakenduse uue versiooni paigaldamisega seoses. Lõppkasutajate jaoks loodi ning lisati Statistikaameti siseveebi juhend, kuidas vastava rakendusega seotud küsimuste või probleemide korral raporteerimise veebikeskkonnas JIRA ülesannet luua.

Kokkuvõte

Käesoleva bakalaureusetöö põhieesmärgiks oli parendada Statistikaameti testimise protsessi lõppkasutajate efektiivsema kaasamise abil. Eesmärgi saavutamiseks sai läbi töötatud tarkvara testimisega seotud kirjandus, mille abil oli võimalik kindlaks teha olulisemad testimise põhimõtted, testimise tasemed ja testimise tüübid. Statistikaameti testimise parendatud protsessi loomiseks oli vaja eelnevalt kaardistada juba olemasolev protsess, mida analüüsides oli võimalik välja tuua selle protsessi vigu ja negatiivne mõju arendusprotsessile. Vana protsessi analüüsist lähtudes sai luua juba uue ning parendatud testimise protsessi Statistikaameti asutusesiseseks kasutamiseks.

Töö olulisemateks tulemusteks oli uue ja parendatud testimise protsessi loomine Statistikaametis. Kaardistamise käigus analüüsiti rakenduse uue versiooni liikumist alates arendaja teavitusest kuni uue versiooni tootekeskkonda paigaldamiseni teenusepakkuja poolt. Analüüsi käigus toodi välja kasutuses oleva protsessi negatiivsed mõjud, mis andsid aluse uue protsessi loomiseks. Uue loodud protsessi positiivne mõju toodi välja ning põhjendati, pärast mida asuti seda juurutama Statistikaameti siseselt.

Parendatud testimisprotsessi juurutamise esimesed tulemused näitavad, et Statistikaameti lõppkasutajad on võimelised tuvastama rakenduste uutes versioonides äripoole protsesside vigu. See omakorda kinnitab fakti, et Statistikaameti testijatel võivad jääda testkeskkonnas funktsionaalse ja mitte-funktsionaalse testimise läbinud rakenduste suhtes märkamata pealt näha töökorras, kuid sisuliselt vigased funktsioonid. Sellest tulenevalt on kindla rakendusega igapäevaselt töötavate lõppkasutajate kogemus ja detailne arusaam rakenduse sisulise poole testimisel suure tähtsusega. Tulevikus on plaan uuendatud testimise protsessi efektiivsust teistkordselt analüüsida ja hinnata ning vajadusel seda optimeerida.

Summary

Present Bachelor's thesis „Improving Software Testing Process Through Efficient Involvement of End Users. The Case of Statistics Estonia" primary goal was to improve Statistics Estonia's testing process through more efficient end users involvement. To achieve this goal there were literature related to software testing worked through, which helped to identify the key principes of testing, testing levels and testing types. Creating improved testing process for Statistics Estonia, existing process had to be mapped, which was analyzed and showed the disadvantages and negative effects on general developing process. Based on existing testing process analysis a new and improved testing process for Statistics Estonia was created.

Most important outcome of the work was the creation of a new and improved testing process in Statistics Estonia. During the mapping, a movement of a new version release of the software was monitored from the notification of the developer to the installation of the software to product environment by the service provider. The analysis showed the negative effects of the existing process that gave a foundation creating an improved testing process. Positive impact of the new process was brought out and justified, after which it was introduced within Statistics Estonia.

First signs of using the new testing process show that Statistics Estonia end users are able to discover bugs in a new version releases of software. This, in turn, confirms the fact that Statistics Estonia software testers may not discover all the bugs in testing environment during the functional and non-functional testing. The unnoticed bugs might be seemingly functional, but in the essence are fulfilling a faulty function. Therefore, users working daily with a certain application have an experience and detailed understanding of the merits of the software, which is a great value in acceptance testing. In the futuure there is a plan to examine improved testing process for a second time, evaluate and, if necessary, optimize it.

Kasutatud kirjandus

Eesti Statistika. (2016). Kasutamise kuupäev: 3. märts 2016. a., allikas Statistikaametist: http://www.stat.ee/statistikaametist

Ghahrai, A. (30. November 2008. a.). *Seven Principles of Software Testing*. Kasutamise kuupäev: 22. veebruar 2016. a., allikas Testing Excellence: http://www.testingexcellence.com/seven-principles-of-software-testing/

Graham, D., van Veenendaal, E., Evans, I., & Black, R. (2012). *Foundations of Software Testing.* London: Cengage Learning.

International Organization for Standardization and International Electrotechnical Commission. (2011). *ISO/IEC 25010:2011 Systems and software engineering -- Systems and software Quality Requirements and Evaluation (SQuaRE) -- System and software quality models.*

Pierre, B., & Fairley, R. E. (2014). *Guide to the Software Engineering Body of Knowledge.* IEEE Computer Society.