TARTU ÜLIKOOL
Arvutiteaduse instituut
Informaatika õppekava

Kristjan Pekk

Maksu- ja Tolliameti sõnumivahetuse komponendi testimine

Bakalaureusetöö (9 EAP)

Juhendajad: Arne Ansper, MSc
Anne Villems, MSc

Tartu 2023

Maksu- ja Tolliameti sõnumivahetuse komponendi testimine

Lühikokkuvõte:
Selle töö eesmärk on kirjeldada testimise kogu protsessi Maksu- ja Tolliameti sõnumivahetuse komponendi testimise näitel. Töös kirjeldatakse testplaani koostamist, räägitakse testjuhtumite ja testraportite ülesehitusest ning tuuakse näiteid ka testimise käigus leitud vigadest.

Töös läbiviidud testimise tulemusena anti Maksu- ja Tolliametile üle toimiv sõnumivahetuse komponent, mille tööülesandeks on sõnumite vahendamine ühisdomeeni, rahvusliku domeeni ja sisedomeeni tollirakenduste vahel.

Kuna töös kasutatud materjali seas on ka ärisaladusi ning testitud komponent ise pole ja ei hakka kunagi tavalugejale kättesaadav olema, on siin töös antud komponendist vaid lühiülevaade ning üritatud võimalikult vähe tavalugejale kättesaamatule materjalile keskenduda.

Võtmesõnad:
Tarkvara testimine, testplaan, kasutusmall, testjuhtum, testraport

CERCS: P175 Informaatika, süsteemiteooria

Testing of the Estonian Tax and Customs Board's transitionhandler component

Abstract:
The purpose of this thesis is to provide an overview of the entire software testing process using the example of the Estonian Tax and Customs Board's transitionhandler component. The thesis describes the process of creating a test plan, discusses the structure of test cases and test reports and provides examples of errors found during the testing phase.

As a result of the testing done in this thesis, the Estonian Tax and Customs board was delivered a functioning transitionhandler component, the purpose of which is to exchange messages between the common domain, national domain and internal domain customs applications.

As the thesis contains references to trade secrets and the tested component is not and will never be accessible to the general reader, only a short overview of the structure of the component is provided in this work, with an attempt to focus on the inaccessible material as little as possible.

Keywords:
Software testing, test plan, use case, test case, test report

CERCS: P175 Informatics, systems theory

Sissejuhatus

Paralleelselt arendamisega toimub tarkvaraarendusprotsessis ka testimine, veendumaks tehtud töö korrektsuses ning leidmaks ja parandamaks selles leiduvaid vigu.

Sellel lõputööl on kaks eesmärki.
Esimeseks eesmärgiks on Maksu- ja Tolliameti sõnumivahetuse komponendi arendusprotsessi jälgimine, testdokumentatsiooni koostamine ning arendusprotsessi tulemi testimine.
Teiseks eesmärgiks on töö lugejale kogu testimise protsessist lühiülevaate andmine.

Töö esimene pool on teoreetilisem, selles räägitakse kogu testimise protsessist ning esitatakse ka testitava süsteemi lühikirjeldus. Süsteemi kirjeldus jääb põgusaks, kuna süsteemi eripäradele ning ehitusele siin töös ei keskenduta.

Töö teine pool on praktiline, selles kirjeldatakse, kuidas esimeses pooles mainitud testimistegevused läbi viidi. Praktilise poole võib omakorda jaotada kaheks: testimise planeerimine ja muude, praktilisemate testimistegevuste läbiviimine.

Kui algselt oli plaanis, et testija viib läbi kõik projektiga seotud testimistegevused (v.a üksus- ja integratsioonitestimine), siis suure töökoormuse ja soovi tõttu kliendile lubatud tähtaegadest kinni pidada otsustati, et see pole efektiivseim viis inimressursside jaotamiseks. Seetõttu tuli Cybernetica tarkvaraarendaja testimisel appi ja teostas jõudlustestimise ise.

Testimist toetab klient ka omapoolse testimismeeskonnaga. Suhtlusest ning töökorraldusest Cybernetica poolsete testijate ja kliendipoolsete testijate vahel räägitakse peatükis “Suhtlus kliendiga”.

Kuna paljud viited viitavad tavalugejale kättesaamatutele materjalidele (Cybernetica sisedokumendid jm ärisaladuste alla kuuluvad dokumendid), ei ole käesolevas lõputöös fookus süsteemi eripäradel ja ehitusel, vaid testimisprotsessil üldisemalt. Nii on see lõputöö kasulik ka neile, kellel nendele dokumentidele ligipääsu pole.

Mõisted ja terminid

AES (*Automatiseeritud Ekspordi Süsteem*) - tollisüsteem, mis on ette nähtud EL-ist väljaviidavate kaupade tollivormistuseks ja järelevalveks [30]

Arendaja või **tarkvaraarendaja** - isik, kes arendab tarkvara

Analüüsidokument - süsteemi ülesehitust ja/või toimimist kirjeldav dokument

Arhitektuuridokument - süsteemi põhikontseptsioone või -omadusi kirjeldav dokument

Automaattest - koodi poolt käivitatud ja jooksutatud test; test, mis ei nõua testijalt sisendit

CCN/CSI - üleeuroopaline maksu- ja tolliametite vaheline teabevahetusplatvorm [30]

EDIFACT - ÜRO elektronandmevahetuse standard halduse, kaubanduse ja veonduse tarbeks [30]

Funktsioon - spetsiifilist toimingut sooritav tarkvaramoodul, mis aktiveerub ta nime ilmumisel avaldises, saab vastu võtta sisendväärtusi ja väljastab üheainsa väärtuse

Gatling - jõudlustestimise tarkvara [14]

ieCA konverter - teenus, mis suudab EDIFACT vormingus faili XML vormingusse konverteerida

IMPULSS - impordi suuna tollideklaratsioonide töötlemise süsteem [31]

Java - objektkeel ja töötlusplatvorm porditavate programmide loomiseks Interneti hajuskeskkonnas

Järjekord või **sõnumitöötlusjärjekord** - töötlusjärjekorra mehhanism, mis võimaldab sõnumeid töökindlalt töödelda ja välja saata. [30]

Klass - andmetüüp programmikeeltes

Klient - süsteemi või teenuse tellija; selle töö kontekstis sõnumivahetuse komponendi tellija ehk MTA

Kohatäitesõne (ingl *placeholder string*) - sõne, mis asendatakse dünaamiliselt rakenduse poolt tegeliku sisuga

MTA - Eesti Maksu- ja Tolliamet

NCTS - Transiidi andmevahetuse süsteem

Perl - programmeerimiskeel, kasutuses süsteemiadministratsioonis, veebiarenduses jm

Postman - tarkvara rakendusprogrammiliideste arendamiseks ja testimiseks [13]

Pääsuõigus - subjektile antav luba juurdepääsuks teatavale objektile, teatavat tüüpi operatsiooni sooritamiseks

Rahvuslik domeen - MTA sisene ja MTA ning teiste riigiasutuste vaheline suhtlus ja seda toetav keskkond [30]

Rakendusprogrammiliides (ingl *api*) - reeglid ja vahendid rakendusprogrammi suhtluseks teiste komponentidega

REST - arhitektuuristiil hajussüsteemide loomiseks [30]

RMIT - Rahandusministeeriumi Infotehnoloogiakeskus

SQL (*Structured Query Language*) - päringukeel andmebaasiga suhtlemiseks

Sõne - määratletud klassi esindava keeleüksuse eksemplar tekstis

Sõnum - XML või EDIFACT vormingus dokument, mida eri riikide tollisüsteemid üksteisele saadavad

Tarne - iga etapi, vaheetapi või muu kliendiga kokkulepitud ajavahemiku tagant tehtud töö kliendile (ning kliendipoolsetele testijatele) üle andmine

Testija - isik, kes testib tarkvara tööd

Viga - süsteemi kasutamist või eesmärgipärast toimimist takistav olukord

Voog - kulgev aine- või muu jada kontekstist sõltuvas tähenduses; siin töös tegevuste / sammude jada

Välisdomeen - MTA ja ettevõtjate vaheline suhtlus ja seda toetav keskkond [30]

Ühisdomeen - EL liikmesriikide vaheline ja liikmesriikide ja kesksüsteemide vaheline suhtlus ja seda toetav keskkond [30]

XML - laiendatav märgistuskeel

X-tee - tehniline ja organisatsiooniline keskkond, mis võimaldab turvalist ja tõestusväärtust tagavat internetipõhist andmevahetust riigiasutuste vahel ja erasektoriga.

1. Testitava süsteemi lühikirjeldus

Sõnumivahetuse komponendi arhitektuuridokumendis [1] (lk 1) seisab, et sõnumivahetuse komponendi ehk üleminekukomponendi põhiülesandeks on sõnumite vahendamine välisdomeeni, rahvusliku domeeni ja ühisdomeeni rakenduste vahel, kasutades ieCA konverterit uue XML ja vana EDIFACT vormingu vahelisteks teisendusteks [1] (lk 1).

EDIFACT vormingult XML vormingu peale lähevad üle kõik Euroopa Liidu liikmesriikide tollirakendused. Euroopa Liit andis igale riigile sõnumite konverteerimiseks kolm varianti: [1]

1) Liikmesriik kasutab keskset ieCA konverterit
2) Liikmesriik kasutab endale paigaldatud ieCA konverterit
3) Liikmesriik teeb endale ise konverteri

Eesti otsustas 1. variandi kasuks ning kasutab nüüd sõnumivahetuse komponenti, et keskse ieCA konverteriga suhelda.[1]

Algselt pidi sõnumivahetuse komponent tegelema ainult konverteeritavate sõnumite vahendamisega. Projekti käigus lisandus komponenti ka sõnumivahetus teiste liikmesriikidega, kuna tehniline liidestus on sama. Seepärast jääb sõnumivahetuse komponent pärast üleminekuperioodi lõppu tööle ka liikmesriikide vaheliste sõnumite vahendajana, kuigi kesksed sõnumite konverteerimise teenused suletakse ning sõnumivahetuse komponent enam konverteeritavate sõnumite vahendamisega ei tegele.[1]

Sõnumivahetuse komponendi töö jagatakse kolme etappi [2] (lk 3-4):

1) **Pärandperiood** (*legacy period*) - selle perioodi vältel saadavad liikmesriigid ainult EDIFACT vormingus sõnumeid, komponent aitab neid vahendada kuid sõnumite konverteerimisega ei tegele
2) **Üleminekuperiood** (*transition period*) - selle perioodi vältel saadetakse nii vanas kui ka uues vormingus sõnumeid (sõltuvalt sellest, kas liikmesriik on juba uuele vormingule üle läinud). Sõnumivahetuse komponent peab sel perioodil aitama sõnumeid nii vanast

[1] Allikas: vestlus Cybernetica arendajaga

vormingust uude kui ka vastupidi teisendada (sest liikmesriigile, kes pole veel uuele vormingule üle läinud, tuleb saata vanas vormingus sõnumeid).

3) **Lõplik periood** - selle perioodi ajal konverteerimist enam ei toimu. Sõnumivahetuse komponent jääb siiski sõnumite vastuvõtmise ja saatmisega tegelema [1] (lk 35).

Kuna sõnumivahetuse komponent pakub efektiivset asendust ka mitmele praegu sõnumite vahetamiseks kasutuses olevale süsteemile, jääb see kasutusse ka peale oma peamise ülesande - sõnumite konverteerimise - lõppemist [1] (lk 35).

2. Testimise planeerimine

Tarkvara testimise protsess algab testimise planeerimisest. Esmasel planeerimisprotsessil on kaks tulemit. Esimene, abstraktsem, on põhjalik arusaam testitavast süsteemist - sealhulgas selle ülesehitusest ning eeldatavast toimimisest.
Teine, käegakatsutavam, on testplaan, mis peaks sisaldama ka esimest ning kirjeldama eesootava testimise kulgu.

Testplaan pole testija igapäevatöös kasutatav dokument, pigem on see alusdokumendiks teistele testimisdokumentidele.

Kui testplaan on valmis, saab liikuda järgmiste planeerimistegevuste juurde.

Peale testplaani valmimist luuakse kasutusmallid, millega pannakse paika süsteemi ja kasutaja vaheline suhtlus. Kasutusmallid pole kasulikud vaid testijatele vaid kõigile, kes süsteemi arendamisel osalevad, aidates aru saada süsteemi toimimisest.

Kui kasutusmallid on valmis, koostatakse testjuhtumid, mille järgi testijad süsteemi testima hakkavad.

Testjuhtumite järgi läbi viidud testimise tulemused märgitakse testraportisse, dokumenteerimaks, milline oli süsteemi seis testimise hetkel.

2.1 Testplaan

Järgnevalt tuuakse Whittakeri [3] ja 99tests Software Testing [4] kogemustele tuginedes olulisemad andmed testplaani kohta.

Testplaani võib pidada kogu testimise alusdokumendiks, mis tutvustab põgusalt testitavat süsteemi ning annab ülevaate planeeritavatest testimistegevustest.

Testplaanis peaks kindlasti olema kirjeldatud testitava süsteemi omadused (ingl *attributes*), komponendid (ingl *components*) ja võimekus (ingl *capabilities*).

Süsteemi omadused iseloomustavad testitavat süsteemi, vastates küsimusele "milline see süsteem on?". Omaduste alla kuuluvad näiteks märksõnad *turvaline*, *kiire*, *lihtsasti kasutatav (kasutajasõbralik)* ja *efektiivne* [4].

Süsteemi komponendid on justkui süsteemi ehitusklotsid. Testplaanis ei pea (ja ei tohikski) need tehniliselt kirjeldatud olema. Siinkohal tuleks aga välja toomiseks valida peamised süsteemi osad, kõiki väiksemaid komponente pole süsteemi töö mõistmiseks vaja teada. Veebipoe puhul võiksid komponentideks olla näiteks kõikide saadavalolevate toodete loetelu, virtuaalne ostukorv ja otsinguriba, mille kaudu tooteid otsida.

Süsteemi võimekuste all peaks olema kirjeldatud süsteemi töökäik ning kasutusvõimalused. Need peaksid olema kirjeldatud kasutaja vaatepunktist ning ei tohiks seega liiga palju tehnilist informatsiooni sisaldada, pigem võiksid need olla stiilis "kasutaja saab veebilehe kaudu toitu koju tellida" või "kasutaja saab kulleri teenust viie palli süsteemis hinnata" [3, 4].

Kui testplaani koostajad ei suuda plaanis neid kolme punkti kokkuvõtvalt kirjeldada, tähendab see, et neil pole ilmselt süsteemist piisavalt head arusaama. Sellisel juhul tuleks testplaani koostamisega oodata ning enne testitavast süsteemist aru saada [3].

Testplaani ülesehitus pole aga kivisse raiutud - kuna testitavad süsteemid on nii ehituselt kui ka keerukuselt erinevad, ei saa iga testplaani koostamisel üht ja sama malli aluseks võtta, seega koostatakse testplaan testitava süsteemi järgi.

Lisaks võivad testimisega tegeleval firmal olla välja kujunenud ka oma tavad testplaani koostamiseks, mis selle firma tööd enim toetavad.

Kuna Cyberneticas kasutatakse siin peatükis kirjeldatust erineva struktuuriga testplaane ning ka sõnumivahetuse komponendi testplaan on koostatud Cyberneticas kasutatava struktuuriga, kirjeldatakse seda testplaani järgmises punktis lähemalt.

2.1.1 Sõnumivahetuse komponendi testplaan

Sõnumivahetuse komponendi testplaan süsteemi ülesehitust ei kirjelda, selleks on Cybernetica projektides kasutusel analüüsidokumendid. Selle asemel kirjeldab testplaan, **kuidas** süsteemi testitakse.
Selleks on testplaanis välja toodud, millised testimistegevused läbi viiakse ja millised süsteemi osad ja liidesed nendega kaetakse.
Samuti on testplaanis ülevaade ka testimiseks kasutatavatest tarkvaradest.

Sõnumivahetuse komponendi projektis on kasutusel sellise ülesehitusega testplaan [5]:

1) **Sissejuhatus** - kirjeldab paari lausega, mis on testitava süsteemi ülesanne.
2) **Testimise skoop ja ülevaade** - kirjeldab erinevaid testimistegevusi, mida süsteemi testimisel läbi viiakse.
3) **Testimiseks kasutatavad tehnoloogiad** - kirjeldab iga testimistegevuse kategooria jaoks kasutatavaid tehnoloogiaid.
4) **Ajakava** - sõnastab, millal klient süsteemi kasutusse võtta soovib / millal süsteem valmis olema peaks.
5) **Testimise tehised** - loetelu dokumentidest, mis iga etapi lõpus kliendile üle antakse.
6) **Vigade ja ettepanekute vormistamine** - kirjeldab, kuidas peaksid nii Cybernetica- kui kliendipoolsed testijad vearaporteid koostama.

Testplaan on kasulik ka neile, kes projektiga hiljem liituvad (nt uued arendajad või uued testijad), pakkudes esialgset ülevaadet projektis plaanitavast testimisest muid testimisdokumente lugemata, vähendades seega projekti sisseelamiseks kuluvat aega.

Kuigi testplaani igapäevaselt ei kasutata, on see siiski testimise alusdokument, mis tuleks hoida ajakohasena. See garanteerib, et testplaan täidab igal ajahetkel oma eesmärki, pakkudes ülevaadet plaanitavatest (või juba tehtud) testimistegevustest.

Kui testplaan on (vähemalt algsel kujul) valmis, saab asuda kasutusmallide loomise kallale.

3. Kasutusmallid

Russelli [6] sõnul kirjeldavad kasutusmallid süsteemi suhtlust kasutajaga, olles seega detailsemad ning tehnilisemad versioonid süsteemi võimekusest, mis testplaanis kirjeldatud on.

Üks kasutusmall sisaldab tavaliselt ühe süsteemis tehtava tegevuse kirjeldust. Näiteks võiks veebipoe üks kasutusmallidest kirjeldada toote lisamist ostukorvi. Selles kasutusmallis oleks kirjeldatud, kuidas kasutaja toodet ostukorvi lisada saab ja mis juhtub siis, kui toote lisamine ebaõnnestub (nt kui laos pole piisavalt kaupa või on kasutaja ületanud maksimaalse tellimuse koguse).

Võib juhtuda, et peale kasutusmallide koostamist avastatakse, et nendes on midagi ebaloogilist, siis võib neid hiljem ka muuta.

Kasutusmallides sisalduv info on voogudeks jaotamata, st seda pole võimalik ühe korraga täielikult testida. Lihtsaim näide sellest on olukord, kus kasutusmallis kirjeldatakse tegevuse õnnestumist ja ebaõnnestumist - kuna korraga ei saa mõlemad toimuda, tuleb selle kasutusmalli testimiseks teha mitu testi.

Kasutusmalli testimiseks vajalikud sammud pannakse kirja testjuhtumitesse. Testjuhtumitest räägitakse lähemalt järgmises peatükis.

4. Testjuhtumid

Almog ja Heart [7] väidavad, et testjuhtumid on testija igapäevases töös ühed tähtsamatest ning enim kasutatavatest dokumentidest, mis kirjeldavad detailselt läbiviidavaid teste. Testjuhtumid luuakse kasutusmallide põhjal, pannes kirja kõikvõimalikud kasutusmalli läbimise viisid (nii edukad kui ka mitteedukad). Erinevalt kasutusmallidest sisaldavad testjuhtumid ka testandmeid, näiteks kasutaja poolt sisestatavaid väärtuseid [7].

Testjuhtumid peaksid ideaalis ära katma kõik kasutusmallides kirjeldatud funktsionaalsused. Kuna kasutusmallid võivad olla mitme vooga, ei piisa ühe kasutusmalli katmiseks tihti vaid ühest testjuhtumist. Samas võib ühe testjuhtumiga katta ka mitu kasutusmalli (või osa mitmest kasutusmallist).

Cybernetica projektides on välja kujunenud kindel testjuhtumi struktuur, mida kasutatakse ka sõnumivahetuse komponendi projektis.

Kuna see struktuur on välja kujunenud Cybernetica siseselt ning pole identne Almog ja Heart [7] poolt kirjeldatuga, kirjeldatakse sõnumivahetuse komponendi testjuhtumeid järgmises punktis lähemalt.

4.1 Sõnumivahetuse komponendi testjuhtumid

Sõnumivahetuse komponendi testjuhtumid on Almog ja Heart [7] poolt kirjeldatule väga sarnased. Ainus selge erinevus on testandmete puudumine - arendajapoolsete keskkondade testandmed pole enamasti kliendipoolsetes keskkondades kasutatavad.

Lisaks kaotab kliendipoolne testimine mõtte, kui testijatele kõik vajalikud tegevused ja andmed ette anda - kuna testija töö pole pelgalt tuim testimine vaid ka arendajate poolt tehtu analüüsimine ja testimise planeerimine, on oluline, et iga testija omapoolse analüüsi teeks, nii on suurim tõenäosus, et kõik võimalikud vead (ka need, mis teistel kahe silma vahele on jäänud) üles leitakse.

Sõnumivahetuse komponendi projektis on kasutusel sellise ülesehitusega testjuhtumid [8]:

1) **Testjuhtumi kood** - kujul TCxx, kus TC tähistab testjuhtumit (ingl *Test Case*) ja x tähistab numbrit 0-9. Kui tegu pole põhivooga, lisatakse testjuhtumi koodile ka täht a - z, tähistamaks ja eristamaks laiendvoogusid.
2) **Lipp "Põhivoog/Laiendvoog"** - annab teada, kas tegu on põhivoo või selle laiendiga.
3) **Kirjeldus** - testjuhtumi lühikirjeldus, enamasti ühelauseline.
4) **Tegija** - roll / rollid, kes testi tegevustes osaleb / osalevad.
5) **Eeltingimused** - testi läbiviimiseks vajalikud eeltingimused (nt vajalikud pääsuõigused).
6) **Õnnestumise järeltingimused** - tingimused, mille täitmisel saab testjuhtumi õnnestunuks lugeda.
7) **Viide kasutusmallile** - link kasutusmalli(de)le, mille alusel testjuhtum koostati.
8) **Testjuhtumi sammud** - jaotatud tegevus-tulemus paarideks. Igas paaris on kirjeldatud tegevus, mida tegema peab ning sellele tegevusele oodatav tulemus.

Lisas 1 on näiteks toodud kuvatõmmis ühest sõnumivahetuse komponendi testjuhtumist.

Oluline on, et testjuhtumid kataks võimalikult suure osa süsteemi tööst. See tähendab, et iga tegevus, mida süsteemis teha on võimalik, peaks vähemalt ühes testjuhtumis läbi tehtama. Nii on suurim tõenäosus, et ka väikseim süsteemi vigane käitumine põhjustab testjuhtumi ebaõnnestumise ja on seega tuvastatav.

Iga väiksema tegevuse jaoks pole siiski arukas eraldi testjuhtumit luua, kuna siis kuluks kõikide testjuhumite läbi tegemise peale liiga kaua aega. Selle asemel peaks testjuhtumis testitama üht kasutaja tegevuse voogu, mille käigus tehakse mitu väiksemat tegevust.

Testjuhtum saab kas õnnestuda või ebaõnnestuda. Testjuhtumi saab õnnestunuks lugeda ainult siis, kui kõik õnnestumise järeltingimused on täidetud, kõigil muul juhtudel (ka siis, kui tegu on väikese erinevusega) loetakse testjuhtum ebaõnnestunuks.

Testjuhtumite tulemuste kirja panemiseks kasutatakse testraporteid, mida kirjeldatakse järgmises peatükis lähemalt.

5. Testraport

Testjuhtumite tulemused kantakse testraportisse. Reicherti [9] sõnul on raporti koostamisel oluline meeles pidada, kelle jaoks raport luuakse. Kindlasti pole mõtet seda liiga paljude andmetega üle ujutada, aga vigaste situatsioonide kordamiseks vajalikud andmed (koos vigade kirjeldustega) peaksid kindlasti välja toodud olema [9].

Kuna testimine toimub pidevalt, luuakse iga uue testimisprotsessi tulemusest uus testraport. Seega annab testraport ülevaate süsteemi toimimisest kindlal aja- või arendusprotsessi hetkel.

Sarnaselt testjuhtumitele on sõnumivahetuse komponendi projektis kasutusel kindla struktuuriga testraport, mida kirjeldatakse järgmises peatükis lähemalt.

5.1 Sõnumivahetuse komponendi testraport

Sõnumivahetuse komponendi testraportid [10] on Reicherti [9] poolt kirjeldatule väga sarnased, kuid kuna sõnumivahetuse komponendi testraport luuakse peamiselt kliendile ülevaate andmiseks, siis leitud vigade taastekitamiseks vajalikke andmeid testraportisse ei panda - kui Cybernetica testimismeeskond leiab vea, pole kliendipoolsel testimismeeskonnal vaja seda sama viga korrata proovida.

Testraportisse liigsete andmete lisamise asemel luuakse vea leidmisel Cybernetica sisekeskkonda vastav tööülesanne, kus on ka vajalikud andmed vea kordamiseks. Selline töövoog tagab, et kliendil on pidev ülevaade süsteemi arendusprotsessist kuid samas piisavalt vähe andmeid, et detailidest mitte segadusse sattuda.

Sõnumivahetuse komponendi projektis on kasutusel sellise ülesehitusega testraportid [10]:

1) **Testjuhtumi kood** - kopeeritakse testjuhtumist.
2) **Testjuhtumi lühikirjeldus** - kopeeritakse testjuhtumist.
3) **Staatus** - lipp võimalike väärtustega OK või NOK (ingl *Not OK*), märgib, kas testjuhtum oli edukas või mitte.
4) **Märkused** - siia lahtrisse pannakse leitud vea korral vea lühikirjeldus.

Lisades on testraportist ka kuvatõmmised. Lisas 2 on kuvatõmmis edukalt läbitud testist ja lisas 3 kuvatõmmis ebaõnnestunud testist.

Testraporti puhul on oluline anda selle vaatajale lihtne ja kiire ülevaade, kas ja millised testid ebaõnnestusid ehk milline oli süsteemi seis selle testimise hetkel.

Kiireima ülevaate saab “staatus” veergu jälgides - sealt saab teada, kas leidub mõni testjuhtum, mis ebaõnnestus [10].

Testraporti kõige olulisemaks osaks võib pidada aga “märkused” veergu [10].

Ebaõnnestunud testjuhtumi korral peab seda veergu lugedes saama lühikese (1-2 lauset) kokkuvõtte testjuhtumi läbimist takistanud veast. See annab kiire ülevaate süsteemi seisust ka neile, kes testjuhtumitega igapäevaselt ei tegele ja nende sisu täpselt ei tea.

Testimisprotsessi juures ei saa aga klienti kõrvale jätta, kuna tööde tellija soovib pidevat ülevaadet sellest, kuidas tema raha kasutatakse. Suhtlust kliendiga kirjeldatakse lähemalt järgmises peatükis.

6. Suhtlus kliendiga

Kuigi testraport annab süsteemi seisust hea ülevaate, ei ole see jooksvate muudatuste ja uuenduste kirjeldamiseks parim viis.

Seetõttu tehakse kliendile projektist regulaarselt ülevaateid. Cybernetica teeb klientidele tavaliselt ülevaate iga tarne lõpus. Sel juhul on tähtsaimateks ülevaadet pakkuvateks dokumentideks tarne kaaskiri [11], mis kirjeldab seda, mis antud tarnega tehtud ja üle antud sai ning testraport [10], milles on tarne seisuga süsteemis kõik vajalikud testjuhtumid läbi viidud.

Lisaks toetab Maksu- ja Tolliamet testimist ka omapoolse testimismeeskonnaga. Seega toimub testimine kahe meeskonnaga. Mõlemal meeskonnal on kasutuses eraldi keskkonnad, milles sama süsteemi üksteisest sõltumatult testida.

Omavaheliseks suhtluseks kasutatakse keskkonda Confluence [12]. Selles keskkonnas toimub ainult omavaheline suhtlus ning seal ei aruta kumbki pool oma meeskonna või ettevõtte siseteemasid.

Cybernetica testimismeeskond testib süsteemi selle arendamisega paralleelselt. Nii testitakse iga väiksemat süsteemi osa kohe peale selle valmimist. Sellise testimise käigus leitakse vigu enamasti testitava komponendi piires - st leitud vea parandamiseks piisab vaid testitava komponendi parandamisest.

Iga kindla ajavahemiku tagant paigaldatakse kliendi keskkondadesse testitava süsteemi uus versioon, mis on Cybernetica testimismeeskonna poolt üle testitud.

Kliendipoolne testimismeeskond testib peamiselt süsteemi üldpilti ja selle funktsionaalsusi. Sellise testimise käigus leitakse tavaliselt suuremaid vigu, mis hõlmavad tihti mitut komponenti korraga ning mille parandamiseks võib tarvis olla mõni varasemalt toimiv funktsionaalsus ümber teha. Selliseid suuremaid vigu on enamasti mitu korda vähem kui Cybernetica testimismeeskonna poolt leitud väiksemaid vigu.

Kuna süsteemi arendus jätkub ka siis, kui see ühel hetkel lõppkasutajatele kättesaadavaks tehakse, võib vea avastada ka kasutaja ise. Kuna Cybernetica töötajatel pole ligipääsu süsteemi kasutajate andmetele, tegelevad kasutajate poolt leitud vigadega kõigepealt kliendipoolsed testijad, kellel selleks vastavad ligipääsud on. Peale vea dokumenteerimist ja kordamiseks vajaminevate võltsandmete (mitte-päriselulised andmed mis kehtivad vaid testkeskkondades) loomist raporteerivad kliendipoolsed testijad vea Cybernetica testimismeeskonnale.

Kõik kliendi testimismeeskonna poolt leitud vead raporteeritakse eespool mainitud keskkonda Confluence [12]. Cybernetica meeskond enda poolt leitud vigu sinna keskkonda ei raporteeri, kuna leitud pisivigade hulk on nii suur, et keskkonna jälgimine muutuks keerukaks.

Kõikide vigadega, mis ühisesse suhtluskeskkonda raporteeritakse, tegeleb edasi Cybernetica testimismeeskond, kes veendub, et viga on korratav ka nende testimiskeskkonnas ning suunab selle edasi Cybernetica arendusmeeskonnale parandamiseks.

7. Testimine

Kokku kulus selle töö raames läbi viidud testimisele ligikaudu 250 töötundi (siia pole sisse arvestatud arendajate poolt läbi viidud testimist).

Testimist alustati testplaani [5] koostamisest. Sellele kulus ligikaudu 65 töötundi.
Testplaani koostamisest räägitakse lähemalt järgmises punktis.

Selles peatükis antakse ülevaade ka erinevatest läbi viidud testimise liikidest ning selgitatakse, miks neid antud projektis läbi viia oli vaja.

Viimasena räägitakse siin peatükis testimise tulemustest. See osa on jagatud alapeatükkideks ning iga peatükk räägib ühe tarne raames tehtud testimisest. Nii on tulemusi lihtsam lugeda ning samas annab see ka parema ülevaate testija töövoost.

Algselt oli plaanitud, et kõik testimistegevused (v.a üksus- ja integratsioonitestimine) viib läbi testija, kuid ajapuuduse tõttu pakkus testimise käigus üks arendajatest oma abi ning viis jõudlustestimise ise läbi.
Ajapuudusele tuli vastu ka klient, kes ei nõudnud erandkorras iga tarne kohta testraportit.

7.1 Testplaani koostamine

Testplaani [5] koostamisel anti testijale paar väiksemat suunist kuid üldjoontes vabad käed. See tegi ühest küljest ülesande huvitavamaks, andes võimaluse erinevaid lähenemisi proovida ning metoodikad ise valida.

Teisest küljest lisas see palju pinget - tuli otsustada, kuidas testida süsteemi, mille ehitusest veel palju ei teatud ning kasutada selleks vahendeid, mida varem kasutatud poldud (testimistarkvarad Postman [13] ja Gatling [14]).

Lisaks ei saanud unustada ajapiirangut - testimise alusdokumendina oli testplaan vaja valmis saada enne teiste testimistegevuste alustamist.

7.1.1 Testplaani 1. versioon

Ajapiirangu tõttu valiti alguses näiliselt kiireim lahendus - testplaani koostamine varasemate testplaanide alusel. Kuna varasemalt oldi Cybernetica poolt arendatud süsteemide AES [15] ja IMPULSS [16] testplaanidega kokku puututud, valiti aluseks just need. Testplaanid olid üksteisele väga sarnased ning see tekitas eksliku arusaama, et sõnumivahetuse komponendi testplaani saab koostada nende kahe süsteemi testplaanide alusel.

Testplaani esimene versioon valmis kõigest ühe tööpäevaga. Tulemuse valideerimiseks näidati testplaani ka vanemtestijale, kes tõi välja, et AES [15] ja IMPULSS [16] süsteemide testplaanid on sarnased, kuna nende süsteemide ülesehitus ja liidesed on sarnased.

Tagasiside esimesele versioonile oli konstruktiivne ning peamise mõttena jäi kõlama see, et varasemaid testplaane aluseks võtta ei saa, tuleb otsast alustada.

7.1.2 Testplaani 2. versioon

Vanemtestija tagasisidet arvesse võttes alustati testplaani koostamist uuesti, seekord puhtalt lehelt.

Kuna uue versiooni koostamiseks uusi ideid polnud, suunduti süsteemi arhitektuuridokumenti [1] lugema. See tundus küll alguses segane, kuid segaste kohtade üle arendajatega arutades saadi süsteemi kohta piisavalt teada, et valmis teha testplaani teine versioon.

Testplaani teist versiooni esitleti sõnumivahetuse komponendi meeskonna koosolekul.
Algselt oli plaanitud testplaani koosolekul põgusalt kirjeldada, kuid märgates selles vigu, andsid arendajad koheselt tagasisidet ning muude teemade, mis koosoleku plaanis olid, arutamiseks aega ei jäänudki. Kuna testplaani koostamisel poldud süsteemi arhitektuurist korrektselt aru saadud, soovitasid arendajad antud tagasiside põhjal uue testplaani luua. Seega alustati testplaani kolmanda versiooniga.

7.1.3 Testplaani 3. versioon

Kuna tagasiside testplaani teisele versioonile sai kirja pandud koosoleku ajal, kasutati kirjapanemisel palju lühendeid ning kogu konteksti kirja ei pandud.

Testplaani teise versiooni esitlemise ja kolmanda versiooniga alustamise vahele jäi aga terve töönädal, mil tegeleti teiste, tol hetkel pakilisemate projektide testimisega. Seega polnud märkmetest enam nii palju kasu, kuna neid analüüsiti liiga hilja, kui kontekst oli juba unustatud. Siiski saadi märkmete põhjal aru, millised testplaani osad tuleks kindlasti ümber teha ning suund kolmanda versiooni alustamiseks oli käes.

Meeskonna koosolekutel aktiivsem osalemine, arendajate nõuanded ja arhitektuuridokumendi [1] põhjalikum uurimine andsid süsteemist palju parema arusaamise ning aitasid uue versiooniga alustada.

Testplaan edenes vaevaliselt, kuid siiski kindla tempoga. Enim kasu oli kahest testplaanile pühendatud koosolekust (üks sõnumivahetuse komponendi meeskonnaga, üks vanemtestijaga).

Kolmanda versiooni valmimiseks kulus kokku ligikaudu viis tööpäeva. Vanemtestija vaatas testplaani üle ning otsustas, et see on kliendile esitlemiseks valmis.

7.1.4 Testplaani tutvustamine kliendile

Paar päeva peale kolmanda versiooni valmimist toimus koosolek kliendiga, mille käigus plaaniti testplaani [5] tutvustada. Kuna kliendil oli sellel kohtumisel mitu testplaaniga mitteseonduvat sisulist küsimust, jäi sellel kohtumisel aega väheks, seega otsustati, et testplaani tutvustatakse järgmisel kohtumisel.

Tekkinud lisaaega kasutati, et tutvuda testimistarkvaraga Postman [13], millega süsteemi läbi rakendusprogrammiliidese testima plaaniti hakata. Kuigi selle tarkvara kasutamise iseseisvalt õppimine pole keerukas, jagasid arendajad ja teised testijad nõuandeid, tänu millele saadi esimesed testid Postmaniga edukalt tehtud.

Järgmisel kliendikohtumisel jõuti testplaani [5] esitleda. Kuna kliendil küsimusi polnud, läks esitlus kiirelt ja sujuvalt.

Küll aga paluti, et testplaanis [5] mainitaks ka mittefunktsionaalseid nõudeid ning testimisel tehtaks läbi ka mittefunktsionaalsed testjuhtumid [17]. Kuna mittefunktsionaalsed testjuhtumid pole projektipõhised, sai nende koostamisel aluseks võtta juba varasemalt teiste projektide tarbeks loodud mittefunktsionaalseid nõuded [18].

Mittefunktsionaalset testimist ja muid testimisliike, mida selle süsteemi testimisel läbi viiakse, kirjeldatakse järgmises peatükis.

7.2 Testimise liigid

Selleks, et kõiki süsteemi omadusi testida, tuleb rakendada erinevaid testimise liike.

Cybernetica projektides on juba varasemalt välja kujunenud, milliseid testimise liike läbi viiakse, seega ei pidanud eraldi välja mõtlema, **milliseid** testimise liike läbi viia, vaid **kuidas** neid läbi viia.

Oli teada, et sõnumivahetuse komponendil hakkab olema kasutajaliides, seega tuleks seda kindlasti testida. Kuna kasutajavaade on väikesemahuline, otsustati (vanemtestija nõuandeid kuulda võttes), et selleks pole automaatteste vaja - kasutajaliidest saab ka käsitsi testida.

Komponent peab olema korraga võimeline sõnumeid vastu võtma ning nende lugemist võimaldama 27 riigile, seega tuleks testida korraga paljude sõnumite liikumist - st süsteemi jõudlust.

Sõnumivahetuse komponent peab suhtlema teiste tollirakendustega (AES ja NCTS) ning CCN / CSI liidesega. Seega testitakse integratsioonitestimise raames nende liidestega ühenduse saamist.

Suurim osa testimisest on funktsionaalne testimine. Selles kategoorias testitakse rakenduse erinevate funktsioonide toimimist - st kas rakendus ikka teeb seda, mida ta tegema peaks?

Testplaanis on ka mittefunktsionaalne testimine. See lisati sinna hiljem kliendi soovil.
Mittefunktsionaalsed nõuded [18] on RMITi poolt varasemalt koostatud ning kirjeldavad nõudeid, millele peavad kõik Cybernetica poolt MTAle valmistatavad projektid vastama.

Lisaks eelnevalt mainitule koostati ka üksustestid - programmeerimiskeeles Java kirjutatud automaattestid, mis testivad väiksemate koodiosade tööd.

7.2.1 Üksustestimine

Üksustestimise eesmärgiks on testida väiksemate koodiosade (nt funktsioon või klass) tööd. Kuna üksustestide loomiseks on vaja koodi mõista, koostavad neid Cyberneticas arendajad. Üksustestid koostatakse tavaliselt juba arenduse käigus, vahetult enne/peale testitava koodi kirjutamist. Seega on üksustestimine esimene testimisliik, mida projektis läbi viiakse.

Kuna üksusteste on väga palju (sest testitavat koodi on palju), ei viida neid käsitsi läbi. Selle asemel luuakse üksustestid automaattestide kujul. See tähendab, et need testid käivituvad automaatselt peale testitava koodi muutmist ning annavad seega ka kohest tagasisidet.

Kuna Cyberneticas ei osale testijad üksustestimise protsessis, siis sellel teemal siin töös pikemalt ei peatuta.

7.2.2 Kasutajaliidese testimine

Kuna Cybernetica projektides luuakse kasutajaliides ühena esimestest süsteemi osadest, on kasutajaliidese testimine tavaliselt esimene testimisliik, mida testija Cybernetica projektides läbi viia saab.

Kasutajaliidese testimise käigus kontrollitakse, et kasutaja saaks läbi kasutajaliidese süsteemi probleemideta kasutada.

Kasutajaliidese testimist võib pidada kõige algajasõbralikumaks testimistegevuseks, kuna töö toimub rakenduse kasutajaliideses, mis peab olema arusaadav ka tavakasutajale.

Kasutajaliidese testimisest saab tuua järgmise näite.

Nõuetes seisab “sõnumivahetuse komponendi sõnumite arhiivi leheküljel peab olema võimalus sõnumeid nende tüübi järgi filtreerida”.

Selle nõude testimiseks:

1) Ava sõnumite arhiivi vaade.
2) Ava vaates sõnumite tabeli filtreerimise menüü.
3) Vajuta avanenud menüüs "sõnumi liik" *dropdown*-menüü peale.
4) "Sõnumi liik" *dropdown*-menüü ei avane, seega pole võimalik sõnumeid nende tüübi järgi filtreerida.
5) Loo arendajale ülesanne, kus kirjeldad, kuidas sellise olukorra tekitasid ja kuidas "sõnumi liik" *dropdown*-menüü selles olukorras töötama peaks.
6) Oota, kuni arendaja ülesande valmis saab ja läbi seejärel samad sammud uuesti.

Kuna sõnumivahetuse komponendil on neli erinevat kasutajaliidese vaadet mis pole väga mahukad [1] (lk 27-31), otsustati arendajate nõul, et neid testitakse käsitsi, kuna automaattestide tegemine oleks sellest palju suurem ajakulu olnud.

Sõnumivahetuse komponendi kasutajaliidese vaated [1] (lk 27-31):

1) **Sõnumite arhiivivaade** - Läbi selle vaate saab otsida, vaadata, alla laadida ja uuesti saata sõnumivahetuse komponendi poolt vahendatud sõnumeid.

2) **Töötlusjärjekorra vaade** - Selles vaates on näha sõnumeid, mis on hetkel töötluses. Kuna sõnumi töötlemine käib kiiresti, on seal valdavalt nähtavad ainult sõnumid, mille töötlemine ebaõnnestus. Ebaõnnestunud sõnumid saavad lühikese ooteaja (mida kasutaja soovi korral muuta saab) ning jäävad järjekorda, kust on neid võimalik ka käsitsi eemaldada.

3) **Liikmesriikide olekuinfo vaade** - Siin vaates kuvatakse infot liikmesriikide olekute kohta - millised funktsionaalsused on hetkel kättesaadavad jne. See vaade on informatiivne, kasutaja siin kuvatavat infot muuta ei saa. Küll aga saab siit vaatest üles laadida katkestuse sõnumi (selle tulemusel lisatakse valitud riigile katkestus, mis takistab vastavalt katkestuse tüübile osade sõnumite saatmist).

4) **Olekuinfo laadimiste logi vaade** - Siin kuvatakse iga olekuinfo sõnumi kohta kirje koos töötlemise staatusega (kas sõnumi töötlemine õnnestus või ebaõnnestus).

Lisades 4-7 on igast kasutajaliidese vaatest ka kuvatõmmis.

Kui arenduse käigus järk-järgult süsteemile funktsionaalsusi lisatakse, testitakse neid võimalusel läbi kasutajaliidese. Nii saab mitu testimisliiki kombineerida ning testimisprotsessi kiiremaks ja mugavamaks muuta.

7.2.3 Integratsioonitestimine

Integratsioonitestimise käigus kontrollitakse komponentidevaheliste liideste vastavust tarkvara disainile ning kõigi komponentide ühilduvust üksteisega.
Kuna integratsioonitestimise ajaks peaksid süsteemi komponendid korrektselt töötama, teostatakse see peale üksustestimist.

Liidestest testitakse:

1) Sõnumivahetuse komponendi poolt kasutatavaid Eesti tollirakenduste REST-liideseid.
2) CCN / CSI liidest (ainult ühenduse saamine).
3) X-tee liidest.
4) ieCA konverterit (ühendust testitakse läbi CCN / CSI liidese).

Integratsioonitestid luuakse üksustestidele sarnaselt automaattestidena.

Kuna ka integratsioonitestide loomise ja haldusega tegelevad arendajad ning testijad selles protsessis ei osale, ei peatuta nendel siin töös pikemalt.

7.2.4 Funktsionaalne testimine

Funktsionaalse testimise raames kontrollitakse süsteemi vastavust funktsionaalsetele nõuetele.
Funktsionaalse testimise käigus leitakse vastus küsimusele “kas süsteem teeb seda, mida ta tegema peaks?”. Võimalusel testitakse kõiki funktsionaalsuseid eraldi.

Funktsionaalse testimise raames kaetakse:

1) Sõnumite vastuvõtmine
2) Sõnumite salvestamine
3) Sõnumite töötlemine
4) Sõnumite saatmine
5) Sõnumijärjekordade pikkuse monitoorimine
6) Sõnumite saatmisel tekkinud vigade monitoorimine

Funktsionaalne testimine toimub kas läbi süsteemi poolt pakutava rakendusprogrammiliidese või kasutajaliidese. Võimalusel eelistatakse mugavuse tõttu kasutajaliidese abil testimist.

Süsteemiga läbi rakendusprogrammiliidese suhtlemiseks kasutatakse testimistarkvara Postman [13].

7.2.5 Mittefunktsionaalne testimine

Mittefunktsionaalse testimise eesmärgiks on kontrollida süsteemi vastavust mittefunktsionaalsetele nõuetele [18]. Lisaks kontrollitakse ka süsteemide kasutatavust, stabiilsust ja tõrkekindlust.

Mittefunktsionaalsed nõuded vastavad küsimusele "Kuidas peab tarkvara vajalikke funktsioone täitma?" [5].

Mittefunktsionaalsete nõuete aluseks on RMIT poolt kirjeldatud ristfunktsionaalsete ja tehniliste nõuete ning IT-profiili dokument projekti arenduse alguses eksisteeriva versiooniga [18].

Kokku on MTA projektidesse loodud 176 mittefunktsionaalset nõuet [18].

Projektijuhi hinnangul olid selle projekti jaoks aktuaalsed vaid 31 [17], seega ei lisandunud ülemäära palju teste.

Mittefunktsionaalse testimise raames testiti näiteks kasutajaliidese vastavust uuematele standarditele, süsteemi taaskäivitamiseks kuluvat aega ja süsteemis jooksvate testide tulemuste kuvamist [17].

Mõnede mittefunktsionaalsete testjuhtumite läbiviimine käis väga kiiresti (näiteks veateadete arusaadaval kujul kuvamist testiv testjuhtum), kuid osad nõudsid kõvasti rohkem aega ja uurimist. Näiteks kulus testjuhtumi, mille raames testiti andmebaasi jõudlusnäitajaid, läbimiseks kaks tööpäeva ning oleks ilmselt kauemgi läinud, kui arendaja appi poleks tulnud.

7.2.6 Jõudlustestimine

Jõudlus- ehk koormustestimise eesmärgiks on veenduda, et testitav süsteem toimiks nii tavapärase kui ka kõrgenenud koormuse all ootuspäraselt [19].

Sõnumivahetuse komponendi jõudlustestimiseks mõõdetakse, kui palju sõnumeid suudab komponent kindlas ajavahemikus saata ja kõigi testsõnumite saatmiseks kulunud aega [19].

Prognoositavalt kasvab sõnumivahetuse komponendist läbiliikuvate sõnumite arv iga aasta ca 5%, kasvades 2023 aastal prognoositud 3,211,815 sõnumilt aastaks 2029 lausa 4,304,139 sõnumile [19].

Algselt oli plaanis, et jõudlustestimise viivad läbi testijad, kasutades selleks testimistarkvara Gatling [14].

Kahjuks ei jõudnud testijad ajapuuduse tõttu ise jõudlustestimist läbi viia, seega paluti abi arendajalt, kes seda varem teistes projektides teinud on.

Arendaja otsustas Gatling tarkvara kasutamise asemel testimiseks programmeerimiskeeles Perl kirjutatud skripti teha, kuna see oli tema jaoks mugavam ja kiirem lahendus [19].

7.3 Testjuhtumite koostamine

Testjuhtumid [8] koostati komponendi kasutajaliidese testimiseks, funktsionaalseks testimiseks ja mittefunktsionaalseks testimiseks.

Kasutajaliidese testimise ja funktsionaalse testimise testjuhtumid koostati kasutusmallide [20] põhjal.

Testjuhtumite koostamisel loodi iga voo (st grupi tegevuste, mida ühe korraga teha ei saa) kohta eraldi testjuhtum.
Lisaks otsiti kasutusmallidest ühiseid osasid mille saaks ühe testjuhtumiga ära katta. Tänu ühiste osade leidmisele loodi kaks testjuhtumit, mis mõlemad sisaldasid endas vooge kahest kasutusmallist [8, 20].

Kõikide kasutusmallide põhjal testjuhtumeid aga ei loodud. Nimelt olid mitmed kasutusmallid kaetud üksus- ja integratsioonitestidega ning kooskõlas projektijuhiga otsustati, et neid pole lisaks testjuhtumitega katta vaja.

Kokku loodi kasutusmallide põhjal 32 testjuhtumit [8].

Kasutusmallide põhjal koostatud testjuhtumitele lisandusid ka mittefunktsionaalsed testjuhtumid, mis olid juba varem koostatud (iga mittefunktsionaalse nõude kohta oli loodud selle nõude testimiseks vajalik testjuhtum) ning mille sisu sõnumivahetuse komponendi projekti jaoks ei kohandatud.

Küll aga otsustati, et kõik 176 mittefunktsionaalset nõuet [18] pole sõnumivahetuse komponendi projekti jaoks aktuaalsed ning koos projektijuhiga välja valiti 31 nõuet, mida tuleks kindlasti testida [17].

Kui testjuhtumid olid valmis, sai asuda nende alusel testima. Läbi viidud testimist ja selle tulemusi kirjeldatakse järgmises peatükis.

7.4 Testimise tulemused

Testimise kirjeldamiseks jagatakse testimine selle järgi osadeks, millise tarne raames see läbi viidi.

Ideaalis pannakse iga tarnega kaasa ka ajakohane testraport [10] ja tarne kaaskiri [11], mis kirjeldab lühidalt selle tarne käigus tehtut.

Selle lõputöö raames toimunud testimise ajavahemikku jäi 6 tarnet:

1) 26. mai tarne [21]
2) 10. juuni tarne [23]
3) 17. juuni tarne [24]
4) 1. juuli tarne [25]
5) 12. juuli tarne [27]
6) 26. augusti tarne [28]

Iga tarne eel oleks pidanud koostama ka tarnega kaasa mineva testraporti, kuid ajapuuduse tõttu jõuti testraportid vaid kahe tarne kohta koostada [10].

Lisaks testjuhtumite [8] alusel testimisele vaadati enne iga tarnet rakendus ka üldisema pilguga läbi, klikkides kõikvõimalikud nupud ja valikud läbi ning veendudes, et kõik tarnevalmis funktsionaalsused töötavad ning ükski tarnest kõrvale jääv funktsionaalsus rakenduse tööd ei häiri.

7.4.1 26. mai tarne

Esimese tarne [21] valmimise ajaks polnud suur osa süsteemist veel valmis, neljast planeeritud vaatest oli realiseeritud kaks - sõnumite arhiivivaade ja töötlusjärjekorra vaade.

See tähendas, et läbi sai teha vaid 14 testjuhtumit 32st võimalikust [8].

26. mai tarne seisuga sai testida [21]:

1) Sõnumite otsimist ja filtreerimist
2) Rakenduse täisvaadet
3) Sõnumi detailvaadet
4) Sõnumi kordussaatmist
5) Sõnumi töötlusaja muutmist
6) Sõnumi töötlusjärjekorrast eemaldamist

Testimisel leiti kolm viga.

Esimese veana avastati, et sõnumi detailvaatest sõnumi XML faili allalaadimisel salvestati arvutisse alati tühi fail.

Teine ja kolmas viga ilmnesid ühe ja sama testjuhtumi käigus, kui testiti sõnumi kordussaatmist. Nimelt ei olnud sõnumi kordussaatmise vaates näha sõnumi XML kuju, st saadetava sõnumi sisu lahter oli tühi.

Lisaks sai sõltumata "saadetava sõnumi sisu" lahtrisse sisestatud XML-ist kordussaatmisel veateate: "Sõnumi saatmine ebaõnnestus!".

Mõlemad leitud vead raporteeriti arendajatele ning tarne testraportis [22] märgiti kaks testjuhtumit, mille käigus vead leiti, ebaõnnestunuks. Testjuhtumite märkmete alla raportis lisati ka põgus kirjeldus leitud vigadest [22].

7.4.2 10. juuni tarne

Teine tarne [23] oli väiksemamahuline - uusi funktsionaalsusi, millega uusi testjuhtumid läbi viia saaks, ei lisandunud.

10. juuni tarne seisuga testiti (lisaks eelnevale) [23]:

1) Säilitustähtaja ületanud sõnumite automaatset kustutamist
2) Alla 3-tähemärgiliste otsingute keelamist (sõnumite filtreerimisel)

Selle tarne raames tehtut testides leiti vaid üks pisem viga. Nimelt loeti tühikut tähemärgiks ning seega oli kahe tähe ja ühe tühiku pikkune sõne (või kolme tühiku pikkune sõne) süsteemi jaoks piisav, et selle alusel otsing teostada. Leitud viga oli madala prioriteediga, seega parandati see alles augusti alguseks.

Kuna tarne testimisel jäi aega üle, uuriti põhjalikumalt eelmise tarnega tehtud sõnumite arhiivivaadet ning avastati, et sõnumite järjestamine nende ID alusel ei toimi. Arendaja sai selle kiirelt parandatud.

Testimise käigus tuli välja, et paar sisulist viga on ka analüüsidokumentides. Testija parandas need ise ära ning ei hakanud nende parandamiseks eraldi ülesannet tegema.

Kuna uusi testjuhtumeid, mida läbi viia saaks ei lisandunud ning eelmise tarne käigus leitud vead polnud selleks hetkeks veel parandatud, otsustati kliendi nõusolekul selle tarnega testraportit kaasa mitte panna (kuna testraporti seis oleks olnud täpselt sama).

7.4.3 17. juuni tarne

Ka kolmanda tarnega [24] ei lisandunud funktsionaalsusi, millega uued testjuhtumid läbi viia.

17. juuni tarne seisuga oli testitav (lisaks eelnevale) [24]:

1) Saadetava sõnumi suuruse kontroll
2) X-tee kaudu saadetavate sõnumite lisadokumentide salvestamine ja faililaiendi kontroll
3) Kasutajaseaded, sh tabeli veergude valimine ning nende järjekorra muutmine ja salvestamine

Seda tarnet testides avastati, et osad logifailid on täis, st saavutanud maksimaalse ettenähtud mahu ning testimise jaoks vajalikke sõnumeid enam ei logita. Tuli välja, et üks eelneva testimise käigus käsitsi andmebaasi lisatud sõnum oli logidesse väga palju veateateid tekitanud.

Ajutise parandusena aitas sõnumi andmebaasist kustutamine ning kuni arendaja seda viga parandas, testiti muid ülesandeid.

Sarnaselt eelmisele tarnele otsustati ka selle tarnega testraportit kaasa mitte panna.

7.4.4 1. juuli tarne

Kuigi neljandat tarnet [25] testides sai läbi viia juba 18 testjuhtumit 32st [8], lisandus rohkem kui vaid nelja testjuhtumi jagu. Nimelt parandati ka eelnevad vead, näiteks sõnumi saatmise testimisel leitud vead.

1.juuli tarne seisuga oli testitav (lisaks eelnevale) [25]:

1) Seirelehed
2) Filtrite vaatamine
3) Tabeli seaded ja nende muutmine
4) Töötlusjärjekorra tabeli automaatne uuendamine katkestuste üleslaadimisel

Kuna tegu oli mahukama tarnega, oli rohkem, mille hulgast vigu otsida ja seega ka suurem tõenäosus vigu leida. Nii juhtus, et sellest tarnest leiti lausa viis viga.

Esimese veana avastati, et sõnumi detailvaates ei kuvata sõnumi sisu, selle asemel kuvatakse kas tühja elemendiplokki või lausa veateadet. See viga sai lahenduse umbes kuu aega hiljem, kui viga enam korrata ei suudetud, seega loeti viga parandatuks.
Selle kindla ülesandega seoses poldud ühtegi muudatust hiljuti tehtud, seega parandas vea mõne muu ülesande käigus tehtud täiendus.

Teisena leiti, et sõnumite töötlusjärjekorra tabeli seadete menüüs kuvatakse kohatäitesõnesid, mitte tegelikku sisu.

Selle tarne lõpuks olid välja valitud ka mittefunktsionaalsed testjuhtumid [17], mille läbiviimisel avastati veel kolm viga.

Esimesena avaldus erisümbolite andmebaasis moonutatud kujul kuvamine.
Lisaks ei kuvatud käsitsi andmebaasi sisestatud sõnumites mõningaid sümboleid (nt $\rightarrow$,∞, $\sum$ ja $\prod$) üldse.

Teine viga leiti seirelehti testides.
Kui mõni automaattest, mille staatust seirelehtedel nägema peaks, ebaõnnestub, tuleb selle staatust ka koheselt uuendada. Selle testimiseks tekitati olukord, kus üks neist testidest ebaõnnestus ning kuna ka peale pooletunnist ootamist polnud seirelehti uuendatud, raporteeriti see olukord veana.

Viimase veana raporteeriti, et konfiguratsioonimuudatused ei jõustu ilma rakenduse restardita. Selle testimiseks muudeti rakenduse seadetest maksimaalset kuvatava sõnumite hulka ja veenduti, et kasutajaliideses kuvatakse endiselt vana limiidi piires sõnumeid.

Kasutusmallide alusel loodud testjuhtumite ja mittefunktsionaalsete testjuhtumite tulemused raporteeriti ühisesse testraportisse [26].

7.4.5 12. juuli tarne

Kuna 12. juuli ja eelneva tarne vahel oli vaid 12 päeva, ei lisandunud viienda tarnega [27] palju muudatusi.

Suurim lisandunud funktsionaalsus oli veasõnumite aruanne. Seda testiti tarnele eelneval päeval ning avastati, et aruandes kuvatakse vigaseid kirjeid. Arendaja tuvastas tarne hommikul loetud minutitega, et viga tuleneb vigasest SQL päringust ja parandas selle.

Kuna ühtki varem leitud viga enam korrata polnud võimalik ja uusi vigu ei leitud, sai selle ülesande sulgeda.

Muud selle tarne muudatused olid seotud pigem parandustega: parandati seirelehtedel testi staatuse uuendamine ja asendati sõnumite töötlusjärjekorra tabeli seadetes kohatäitesõned.

Kuna testijal polnud piisavalt aega testraporti koostamiseks, otsustati selle tarnega testraportit kaasa mitte panna.

7.4.6 26. augusti tarne

Viimane, 26. augusti tarne [28] oli ühtlasi ka mahukaim tarne. Lisaks algselt plaanitule tarniti sellega ka need muudatused, mis eelnevatest tarnetest välja olid jäänud. Näiteks tõsteti sinna tarnesse nii palju katkestustega seotud parandusi ja muudatusi, et ühel hetkel otsustati tarne nimetuseks panna “katkestuste tarne” [28].

26. augusti tarne seisuga oli testitav (lisaks eelnevale) [28]:

1) Liikmesriikide olekuinfo vaade
2) Olekuinfo laadimiste logi vaade
3) Katkestuste haldus (sh nende üleslaadimine, töötlemine ja kordussaatmine)
4) Kõikide failide salvestamine eri rakenduste jaoks eraldi kaustadesse
5) Säilitustähtaja ületanud katkestuste automaatne kustutamine

Kuna selle tarnega lisandus lausa kaks vaadet, mida kasutajaliideses testida sai, oli see tarne ka testimise mahu poolest suurim. Sellegipoolest leiti seda tarnet testides vaid neli viga.

Esimesed kaks viga avastati katkestuse sõnumi kasutajaliidesest üleslaadimisel.

Tuli välja, et katkestuse faili üleslaadimisel ei kontrollita faililaiendit. Katkestuse fail tohib olla ainult XML formaadis, seega ei tohiks muid failitüüpe lubada üles laadida. Algselt peeti seda viga tõsiseks, kuna üles sai laadida ka programmifaili (.exe või .sh laiendiga faili), mis võinuks olla ka pahavara, kuid arendaja kinnitas, et Cybernetica süsteemid sellistel failidel käivituda ei laseks ja programmifail märgitaks üleslaadimisel vigaseks.

Siiski pole mõistlik kasutajal ükskõik mis faili üles laadida lubada, kuna heatahtlik kasutaja võib kogemata vale failitüübi üles laadida ning veateate puudumisel eeldada, et töövoog jätkub ja üles laetud katkestus aktiveerub soovitud ajahetkel. Seega lisati ka puuduv sellekohane veateade.

Katkestust üles laadides saab lisada ka märkuse/kommentaari. Pisema veana avastati, et kommentaari kirjutatud täpitähti kuvatakse andmebaasis arusaamatute sümbolitena.

Süsteemi konfiguratsioonifailis saab määrata, kui kaua tuleks katkestusi, mis on aegunud, säilitada. Need katkestused, mida selle järgi enam säilitama ei peaks, kustutatakse automaatselt. Kolmanda veana leiti, et aegunud katkestusi ei kustutata õigel ajal - katkestused, mis pidid testimise päeval kustuma, olid järgmisel hommikul veel andmebaasis olemas.

Neljas viga leiti olekuinfo laadimiste logi vaatest. Nimelt ei töötanud seal olevate sõnumite filtreerimine nende olekute põhjal - sõltumata valitud olekust tagastati kõikides olekutes sõnumid.

Ka selle tarnega otsustati ajapuuduse tõttu testraportit mitte kaasa panna. Viimased tarne testimise käigus leitud vead parandati 30. augustiks [28].

8. Arendusjärgne faas

Üleminekuperioodiks on Eesti planeerinud 1.06.2023 - 1.12.2023 [29]. Selle perioodi vältel tegeleb sõnumivahetuse komponent lisaks sõnumite marsruutimisele ka ieCA konverteri abil sõnumite konverteerimisega [2] (lk 3-4).

Kuigi peale üleminekuperioodi sõnumite konverteerimist enam ei toimu, jääb sõnumivahetuse komponent liikmesriikide vahelisi sõnumeid vahendama [1] (lk 32).

Selle lõputöö valmimise ajaks on sõnumivahetuse komponent kliendile üle antud.
See tähendab, et komponendi arendus on lõppenud ning plaanitavat lisaarendust pole. Kui aga peaks juhtuma, et tekib vajadus uute funktsionaalsuste järele või on soov olemasolevaid funktsionaalsusi muuta, siis on Cybernetica valmis komponenti edasi arendama.[2]

Omalt poolt on Cybernetica arendajad teinud ka lisapingutusi, et komponenti tulevikus uute süsteemidega võimalikult lihtne liidestada oleks.[2]

[2] Allikas: vestlus Cybernetica tarkvaraarendajaga

9. Kokkuvõte

Selles töös kirjeldati testimise kogu protsessi Maksu- ja Tolliameti sõnumivahetuse komponendi näitel.

Sellel tööl oli kaks eesmärki, millest mõlemad on saavutatud.

Esimeseks eesmärgiks oli, et kliendile antakse üle vigadeta ja põhjalikult läbi testitud süsteem. Klient võttis 30.08.2022 süsteemi vastu (st luges süsteemi üleandmiseks sobilikuks). Süsteemiga anti kaasa testimise dokumendid - testplaan, kasutusmallid, testjuhtumid ja testraportid - mille alusel saavad tulevikus testimist jätkata ka teised testijad.

Teiseks eesmärgiks oli anda selle töö lugejale ülevaade testimise kogu protsessist.
Kuna töös kasutatud materjali seas on ka ärisaladusi ning komponent ise pole ja ei hakka kunagi tavalugejale kättesaadav olema, anti siin töös küll komponendist lühiülevaade kuid üritati võimalikult vähe tavalugejale kättesaamatule materjalile keskenduda.

Kui seda tööd kunagi täiendada soovitakse, saaks siin kirjeldada ka tootevigu (st neid vigu, mis avalduvad süsteemi ametliku tööperioodi ajal) ja nende parandamist.
Oleks naiivne eeldada et neid vigu ei teki - alati leidub mõni viga mis on kahe silma vahele jäänud.

Sõltumata töö täiendamisest võiks see pakkuda ülevaadet testimise kogu protsessist, mis on küll iga süsteemi puhul erinev kuid mille üldpraktikad ilmselt suurt muutust läbi ei tee.

Lisaks võiks see lõputöö teistele testijatele näidata, et testimine pole alati sile tee ja ebaõnnestumised on täiesti loomulikud. Näiteks tuli sõnumivahetuse komponendi testplaani koostamisel luua lausa kolm erinevat versiooni, enne kui lõpliku testplaanini jõuti, aga see on osa protsessist, mis tagas lõppkokkuvõttes parema arusaama süsteemist.

Viidatud kirjandus

[3] Whittaker, James. The 10-Minute Test Plan. IEEE Softw. 29, 6 (November 2012), 70–77. DOI: https://doi.org/10.1109/MS.2012.25

[4] 99tests Software Testing, 2016. How to log bugs the Google way: The ACC methodology. https://medium.com/@99tests/how-to-log-bugs-the-google-way-the-acc-methodology-4128a586a112 (10.12.2016)

[6] Mike Russell, ESEP, Supporting Decision Makers with Use Cases; Case Study Results, Procedia Computer Science 153 (2019), 294-300,
DOI: https://doi.org/10.1016/j.procs.2019.05.082

[7] Almog D., Heart T. (2009) What Is a Test Case? Revisiting the Software Test Case Concept. Software Process Improvement. EuroSPI 2009. Communications in Computer and Information Science, vol 42. Springer, Berlin, Heidelberg.
DOI: https://doi.org/10.1007/978-3-642-04133-4_2

[9] Amy Reichert, 2021. How to write a test report for software testing https://searchsoftwarequality.techtarget.com/tip/How-to-write-a-test-report-for-software-testing (09.01.2022)

[12] Confluence, https://www.atlassian.com/software/confluence (21.05.2023)

[13] What is Postman?, https://www.postman.com/product/what-is-postman/ (10.05.2023)

[14] Gatling, https://gatling.io/docs/gatling/ (10.05.2023)

Viidatud Cybernetica või MTA dokumendid

[1] Cybernetica AS, Üleminekukomponendi arhitektuur, 2021 (viimane külastus 10.05.2023)

[2] Cybernetica AS, Üleminekukomponendi töö kirjeldus erinevate stsenaariumite puhul, 2022 (viimane külastus 10.05.2023)

[5] Cybernetica AS, Testplaan, 2022 (viimane külastus 10.05.2023)

[8] Cybernetica AS, Testjuhtumid, 2022 (viimane külastus 10.05.2023)

[10] Cybernetica AS, Testraportid, 2022 (viimane külastus 10.05.2023)

[11] Cybernetica AS, Tarne kaaskiri, 2022 (viimane külastus 10.05.2023)

[15] Cybernetica AS, Süsteemi AES testplaan - Etapp 1, 2021 (viimane külastus 10.05.2023)

[16] Cybernetica AS, Projekti Impulss testplaan, 2020 (viimane külastus 10.05.2023)

[17] Cybernetica AS, MFN vastavuse testjuhtumid, 2022 (viimane külastus 10.05.2023)
[18] RMIT, RMIT üldised ristfunktsionaalsed, mittefunktsionaalsed ja tehnilised nõuded 2.6.0, 2022 (viimane külastus 10.05.2023)
[19] Cybernetica AS, Koormustestide juhend, 2022 (viimane külastus 10.05.2023)
[20] Cybernetica AS, Kasutusmallimudel, 2022 (viimane külastus 10.05.2023)
[21] Cybernetica AS, 20220526 Sõnumivahetuskomponent, 2022 (viimane külastus 10.05.2023)
[22] Cybernetica AS, 2022-05-26 tarne testraport, 2022 (viimane külastus 10.05.2023)
[23] Cybernetica AS, 09.06 freeze, 10.06 tarne, 2022 (viimane külastus 10.05.2023)
[24] Cybernetica AS, 16.06 freeze, 17.06 tarne, 2022 (viimane külastus 10.05.2023)
[25] Cybernetica AS, 30.06 freeze, 01.07 tarne, 2022 (viimane külastus 10.05.2023)
[26] Cybernetica AS, 2022-07-01 testraport, 2022 (viimane külastus 10.05.2023)
[27] Cybernetica AS, 20220712 tarne, 2022 (viimane külastus 10.05.2023)
[28] Cybernetica AS, 20220826 Katkestuste tarne, 2022 (viimane külastus 10.05.2023)
[29] Cybernetica AS, Üleminekuperioodi ülevaade, 2022 (viimane külastus 10.05.2023)
[30] Cybernetica AS, Sõnastik, 2022 (viimane külastus 10.05.2023)
[31] Cybernetica AS, Süsteemide AES ja Impulss vaheline liides, 2022 (viimane külastus 13.05.2023)

Lisad

Lisa 1: sõnumivahetuse komponendi testjuhtum [8]

TC01 Tabeli seadete muutmine

Testjuhtum TC01	**Põhivoog**
Kirjeldus: Kasutaja saab valida tabelis kuvatavaid veerge ja muuta nende järjekorda	
Tegija: Ametnik	

Eeltingimused:
• Kasutaja asub kuval, millel on seadete muutmise võimekusega tabel
• Kasutajal on toiminguks vastavad õigused:
  • **th.official.messages.view**
  • **th.official.unavailability.view**	

Õnnestumise järeltingimused: • Kasutaja poolt tehtud muudatused on salvestatud kasutaja seadetesse	
Viide kasutusmallile: TUC03 Muuda tabeli seadeid (*põhivoog*)	
Tegevuse sammud	***Oodatud tulemus***

Kasutaja vajutab **seadete ikoonile**	• Süsteem leiab kasutaja varasemalt talletatud seaded
	• Süsteem avab flyout akna, milles kuvab kõik tabelis lubatavad veerud
	  • Veerud, mis on hetkel valitud (nähtavad), on linnutatud
	  • Veerud on kasutaja poolt valitud järjekorras
	  • Igal kuvataval real on
	  • märkeruut
	  • veeru nimetus
	  • asukoha muutmise ikoon (lohistamiseks)
	  • Lisaks kuvatakse nupp vaikeseadete taastamiseks
	• **Tabel seis vastab hetkel valitud seadetele**

Kasutaja teeb seadetes soovitud muudatused	• Süsteem talletab muudatused jooksvalt kasutaja seadetesse • Tabel muutub vastavalt muutustele seadetes

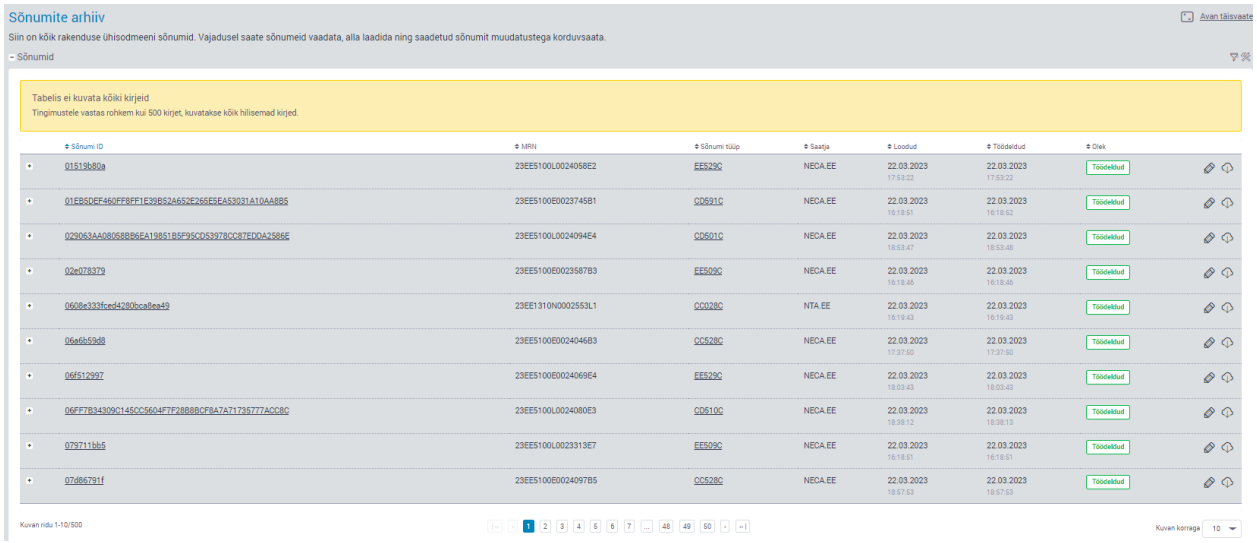
Lisa 2: testraport, OK staatusega test [26]

Kood	Testjuhtumi kirjeldus	Staatus	Märkused
TC01	Kasutaja saab valida tabelis kuvatavaid veerge ja muuta nende järjekorda	OK	

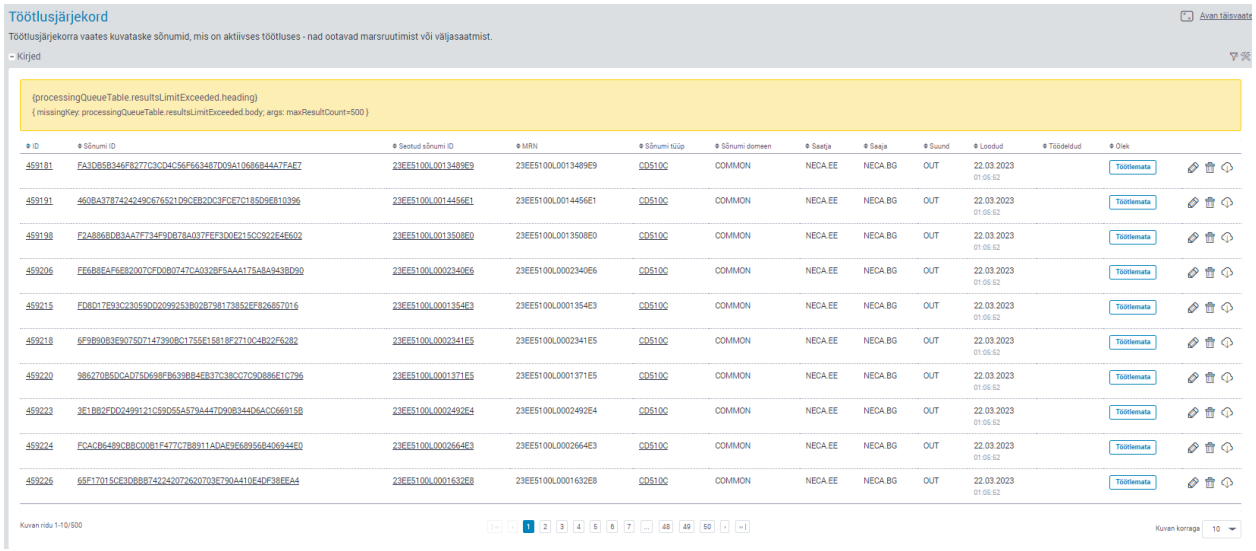
Lisa 3: testraport, NOK staatusega test [22]

TC07	Kasutaja vaatab sõnumi andmeid	NOK	sõnumi andmete vaatamisel saab kas veateate "sõnumi faile ei suudetud alla laadida" või kuvatakse andmete xml asemel tühi elemendiplokk

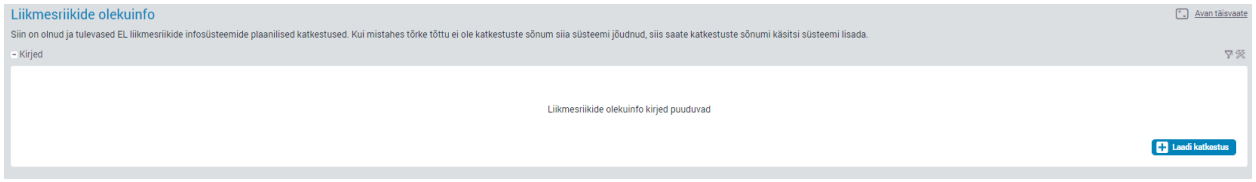
Lisa 4: Sõnumite arhiivivaade [1] (lk 28)



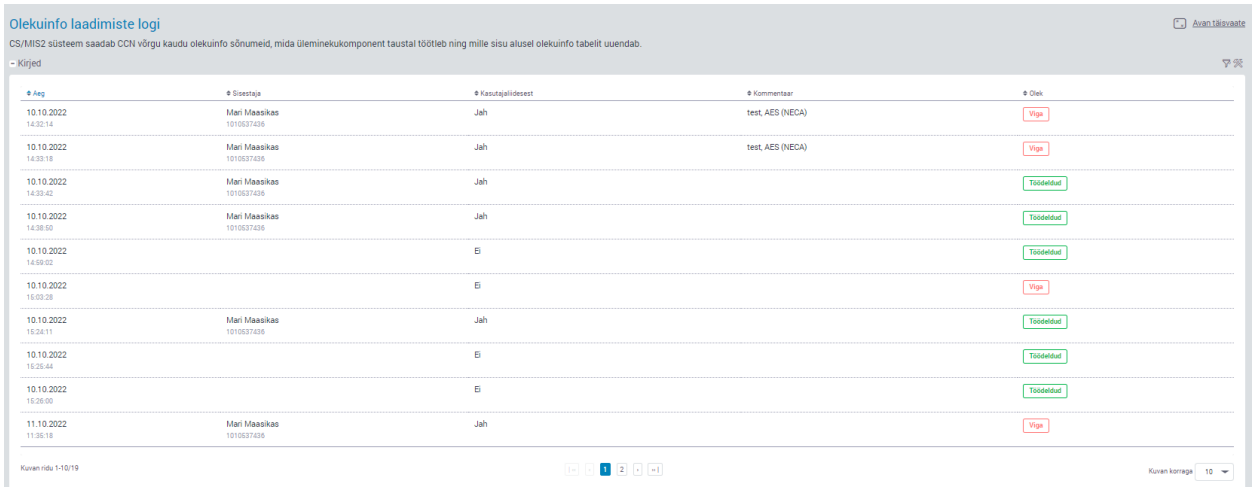
Lisa 5: Sõnumite töötlusjärjekorra vaade [1] (lk 29)



Lisa 6: Liikmesriikide olekuinfo vaade [1] (lk 30)



Lisa 7: Olekuinfo laadimiste logi vaade [1] (lk 31)



Lisa 8 : Litsents

Lihtlitsents lõputöö reprodutseerimiseks ja üldsusele kättesaadavaks tegemiseks

Mina, Kristjan Pekk,

1. annan Tartu Ülikoolile tasuta loa (lihtlitsentsi) minu loodud teose „Maksu- ja Tolliameti sõnumivahetuse komponendi testimine“, mille juhendajad on Arne Ansper ja Anne Villems, reprodutseerimiseks eesmärgiga seda säilitada, sealhulgas lisada digitaalarhiivi DSpace kuni autoriõiguse kehtivuse lõppemiseni.

2. Annan Tartu Ülikoolile loa teha punktis 1 nimetatud teos üldsusele kättesaadavaks Tartu Ülikooli veebikeskkonna, sealhulgas digitaalarhiivi DSpace kaudu Creative Commonsi litsentsiga CC BY NC ND 3.0, mis lubab autorile viidates teost reprodutseerida, levitada ja üldsusele suunata ning keelab luua tuletatud teost ja kasutada teost ärieesmärgil, kuni autoriõiguse kehtivuse lõppemiseni.

3. Olen teadlik, et punktides 1 ja 2 nimetatud õigused jäävad alles ka autorile.

4. Kinnitan, et lihtlitsentsi andmisega ei riku ma teiste isikute intellektuaalomandi ega isikuandmete kaitse õigusaktidest tulenevaid õigusi.

Kristjan Pekk
8.05.2023