Tartu Ülikool

Loodus- ja täppisteaduste valdkond

Tehnoloogiainstituut

Rauno Tamm

Microsoft Dynamics NAV tarkvara automaattestimise juhtumiuuring ettevõttes Columbus Eesti AS

Magistritöö (30 EAP)

Arvutitehnika eriala

Juhendajad:

doktorant Rivo Lemmik

vanemteadur Heiki Kasemägi

Tartu 2017

Resümee/Abstract

Microsoft Dynamics NAV tarkvara automaattestimise juhtumiuuring ettevõttes Columbus Eesti AS

Käesoleva töö peamise eesmärgina on kajastatud majandustarkvara konsultatsiooniettevõtte tegevusest lähtuvalt juhtumiuuringut, kus põhjalikumalt on keskendutud automaattestimise töövahendite rakendamisele ERP-tarkvara MS Dynamics NAV'i arendus- ja juurutusprotsessis.

Kirjanduse ülevaate osas on kirjeldatud teisi analoogseid automaattestimise tööriistakomplekte ja kajastatud MS Dynamics NAV'i majandustarkvara automaattestimise tööriistakomplekti tööpõhimõtet. Samuti on käsitletud tarkvara arendusprotsessis kasutatavaid automaattestimise ja juurutusmetoodikaid.

Juhtumiuuringu raames on keskendutud ERP-majandusinfosüsteemi automaattestimise funktsionaalsuse rakendamisele ja käsitletud MS Dynamics NAV'i standardfunktsionaalsuse arendamisel tarkvara automaattestimise tööriistakomplekti kasutamist. Juhtumiuuringu praktilise töö käigus on realiseeritud MS Dynamics NAV'i prototüüplahendus ja uuritud automaattestimise tööriistakomplekti toimivust erinevates simuleeritud veaolukordades.

CERCS: P170

Märksõnad: tarkvara testimine, automaattestimine, juhtumiuuring, ERP, majandustarkvara, Microsoft Dynamics NAV

Case study of Microsoft Dynamics NAV automated testing at Columbus Eesti AS

As the main objective of this research a case study is examined, which is based on the activity of a business software consultation company and that concentrates more thoroughly on implementing test automation tools in the development and deployment process of ERP software MS Dynamics NAV.

In the overview of literature section similar test automation toolkits are described, as well as the working principle of MS Dynamics NAV business software's test automation toolkit. Also the test automation and deployment methodologies used in software development process are described.

Within the case study the focus is on the implementation of the functionality of test automation of ERP business information system, and the usage of test automation toolkit in the development of MS Dynamics NAV standard functionality is described. During the practical phase of the case study the prototype solution of MS Dynamics NAV was executed and functioning of the test automation toolkit in different simulated error conditions was researched.

CERCS: P170

Keywords: software testing, automated testing, case study, ERP, business software, Microsoft Dynamics NAV

Sisukord

Jooniste loetelu

Tabelite loetelu

Lühendid

API	(*Application Programming Interface*) rakendusliides ehk API-liides
C/AL	(*Client/Server Application Language*) klient/server-rakenduse keel
C/SIDE	**(***Client/Server Integration Development Environment*) klient/server integratsiooni arenduskeskkond
CU	(*Cummulative Update*) tarkvara kumulatiivne uuendus
ERP	(*Enterprise Resource Planning*) ettevõtte majandustarkvara
HTTP(S)	(*Hypertext Transfer Protocol Secure*) turvaline hüperteksti edastusprotokoll
KMD	käibedeklaratsioon
MS	Microsoft
ODATA	(*Open Data Protocol*) avatud andmeprotokoll
REST	**(***Representational State Transfer*) tarkvaraarhitektuuri stiil
SEPA	(*Single Euro Payment Area*) ühtne euromaksete piirkond
SOAP	(*Simple Object Access Protocol*) lihtne objektipöördusprotokoll
SP	(*Service Pack*) tarkvara hoolduspakett
UI	(*User Interface*) kasutajaliides
W1	(*World Wide*) MS Dynamics NAV'i tarkvara rahvusvaheline väljalase
WSDL	(*Web Services Description Language*) XML-põhine keel veebiteenuste kirjeldamiseks

1 Sissejuhatus

Tarkvara testimine on asendamatuks osaks tarkvaraarenduses ja lahutamatuks osaks kogu juurutusprotsessis [1]. Tarkvaraarenduse protsessis on testimine üheks töömahukamaks ja kulukamaks etapiks, moodustades ligikaudu 50% kogu arenduse kulust ja mahust [1-4]. Testimine on üks põhitegevusi agiilse tarkvaraarenduse meetodi puhul [2]. Hiljutised uurimused näitavad, et 2013 aastal oli ülemaailmne kulu koodivigade leidmiseks ja parandamiseks kasvanud 312 miljardi dollarini aastas [3]. Kulu vähendamiseks ja efektiivsuse parandamiseks tuleks käsitsi testimine asendada automatiseeritud testimisega [1].

Tarkvara testimise eesmärgiks on tuvastada defekte ja ootamatusi tarkvara käitumises [3,5]. Testimise protsessi saab läbi viia käsitsi ja/või automatiseeritult [3,5]. Käsitsi testimise teostamisel võtab testija üle lõppkasutaja rolli, käivitades testitava tarkvara testi eesmärgil [3,5]. Selline tegevus on ajakulukas ja tulemus ebastabiilse kvaliteediga [4]. Automatiseeritud testimise korral kasutab testija mõnda teist tarkvara (nn test tööriista), mida tavaliselt testimise tarbeks täiendatakse testkoodiga [3,5]. Tuginedes hiljutistele uurimustele, on testimise automatiseerimise eelisteks paranenud koodikvaliteet, võimalus käivitada vähema ajaga rohkem teste ning lihtsamini kasutada testkoodi [3,5]. Peamised puudused on seotud testkoodi automatiseerimise ettevalmistamise kulude ja haldamisega [3,5]. Testimise metoodika oleneb suuresti vastavalt valitud arendusmetoodikast [2]. Agiilse metoodika puhul on see pidev arendusprotsessi osa, mistõttu kasutatakse valdavalt just automaatteste [2].

Maailmas on toimunud kiire kasv automatiseeritud testimise tööriistade kasutamisel ning üha suuremas mahus arendatakse automaattestimise töövahendeid erinevatele platvormidele [1]. Tarkvaralahendused katavad järjest olulisemat rolli keerulistest süsteemides, eriti kõrgtehnoloogilistes rakendustes, mis on seotud transpordi, finantsjuhtimise, kommunikatsiooni, biomeditsiini jne valdkondadega [6]. Nimetatud süsteemide jaoks on efektiivne toimimine ja veakindlus, mida tagatakse vastava tarkvara arhitektuuriga, väga olulised [6]. Peamine probleem on selles, et tarkvaralahenduste keerukus on hüppeliselt kasvanud ning selle vead võivad avalduda alles tarkvara elutsükli käigus [6]. Seetõttu tuleb

tarkvara kvaliteediprobleeme, sealhulgas testimise protseduure, lahendada kvaliteedikontrolli töövahenditega [6].

Käesolev töö keskendub standardfunktsionaalsuse regressioontestimisele arendusprotsessis läbi standardtestide rakendamise ning ei käsitle põhjalikult uute automaattestide loomise tegevusi standardfunktsionaalsusest eraldiseisvate erilahenduste arendamisel.

Käesoleva magistritöö eesmärgiks on kajastada majandustarkvara konsultatsiooni ettevõtte tegevusest lähtuvalt juhtumiuuringut, mis keskendub automaattestimise töövahendite rakendamisele ERP-tarkvara MS Dynamics NAV'i arendus- ja juurutusprotsessis. Automaattestimise töövahendite kasutamisel on eesmärgiks saavutada ettevõte majandustarkvara arendus- ja juurutusprojektide teostamisel suurem efektiivsus ja stabiilsema kvaliteediga tulemus. Eesmärk on saavutada arendustööde testimisel ajakulu kokkuhoid 20% ulatuses võrreldes traditsioonilise testimisega.

2 Ülevaade analoogsetest lahendustest

Automaattestimise rakendamisel kasutatakse valdavalt kolmanda osapoole tarkvara, mis on spetsiaalselt loodud testide loomiseks ja automatiseerimiseks. Olemas on erinevaid automaattestimise tööriistu, mis on loodud laiatarbe arendusvahendite jaoks, näiteks Java ja .NET keskkonnad, kuid ERP-spetsiifiliste automaattestimise tööriistade valik on enamasti kitsalt piiratud konkreetse ERP-toote põhiselt.

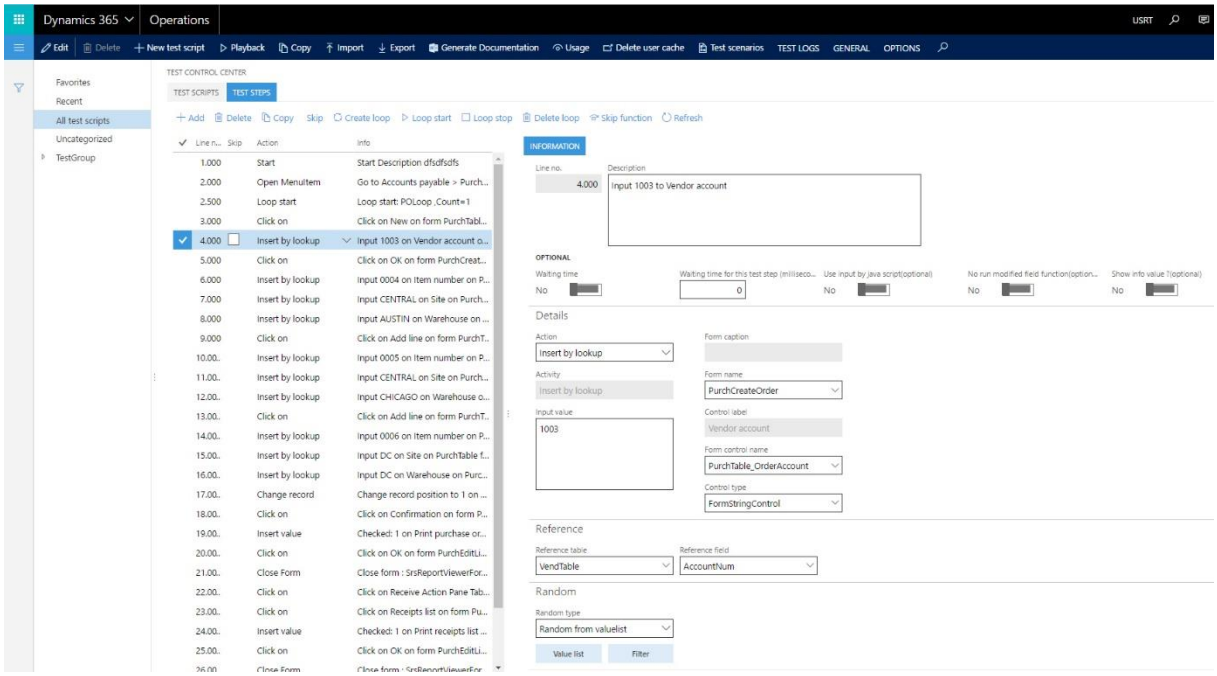
2.1 AXeptance

AXeptance on tarkvaralahendus, mille abil saab luua ja teostada automaattestimist MS Dynamics AX (versioonid 2009, 2012) ja MS Dynamics 365 for Operations toodetes [7]. AXeptance on täielikult integreeritud MS Dynamics'isse ning kasutab selle toote sisest objektide struktuuri, omades juurdepääsu kõikidele elementidele nagu tabelid ja klassid. AXeptance simuleerib reaalset kasutajasuhtlust ja klienditegevusi kasutajaliideses [7-8]. Tegemist on tasulise tarkvaralahendusega, mida pakub ettevõte Rödl & Partner. Tarkvara on saadaval kahes versioonis [7]:

- AXeptance Basic – tarkvara baasversioon, mis lubab funktsionaalsuse ja regressiooni testimist ühe kasutajasessiooni piires [7];
- AXeptance Enterprise – tarkvara versioon, kus lisaks baasversiooni funktsionaalsusele võimaldatakse koormustestide kasutamist üle kogu võrgu mitme kasutajasessiooniga [7].

AXeptance kasutab testskripte, mis on salvestatud MS Dynamics'is, kasutades Test Script Recorder'i vormi. Testskripti loomise ajal salvestatakse taustal kasutaja kõik tegevused, mis äriprotsessi teostamisel sooritati. Seda koos viitega mõjutatud süsteemi objektidele [7-8].

Test Control Center on testskriptide teek, kus saab salvestatud testskripte redigeerida, käivitada või koostada teststsenaariume (joonis 2.1) [7]. Test Control Center pakub suurel hulgal funktsioone, mille abil saab lähtuvalt äriprotsessi testnõuetest redigeerida testskripte [7-8].



Joonis 2.1: Testimise sammud MS Dynamics 365 for Operations'i tootes

2.2 SoapUI

Tegemist on maailmas laialdaselt kasutatava avatud lähtekoodiga API-liidese testimise tööriistaga, mõeldud eelkõige veebipõhiste rakendusliideste testimiseks [9]. SoapUI loodi 2006. aastal, kui polnud olemas ühtegi avatud lähtekoodiga rakendusliidese testimise tööriista [9]. Tarkvara on saadaval kahes versioonis: SoapUI OpenSource ja SoapUI NG Pro [10]. Tööriist on graafilise liidesega, mis võimaldab lihtsalt käivitada automatiseeritud funktsionaalsuse, turvalisuse, regressiooni, ja koormusteste [11]. Toetatud on peamised tehnoloogiad [10]:

- SOAP/WSDL;
- REST;
- HTTP(S).

SoapUI on „Šveitsi armeenuga" veebipõhiste rakendusliideste funktsionaalse ja regressioontestimise jaoks. Rohke funktsionaalsusega platvorm võimaldab põhjalikult kontrollida ja täiustada testitavate rakenduste ja teenuste kvaliteeti. Testide koostamiseks ei ole vaja tarkvaraarendaja kompetentsi, vaid see on lihtne ja intuitiivselt arusaadav tegevus [11].

2.3 Visual Studio Test Professional

Visual Studio Test Professional on Microsofti arenduskeskkonna versioon, mis sisaldab testimise tööriistakomplekti, võimaldades käsitsi testimise haldust ja automatiseerimist [12]. Tarkvara sisaldab endas testide haldamise moodulit MS Test Manager, mis võimaldab käitada, salvestada ja automaatselt korrata manuaalseid teste [13]. Täielikult seadistatav testide käitaja salvestab teostatud sammud ja iga sammu staatuse [12].

MS Test Manager on integreeritav Team Foundation Server'iga, mille eesmärk on parandada projekti ülevaadet läbi integreeritud algkoodi repositooriumi. Team Foundation Server'i ja Visual Studio Online'i liidestatus võimaldab keskselt koordineerida kõiki testimistegevusi ning saada tervikliku ülevaate projektile seatud testide kriteeriumitest ja sooritustest [13].

Alates versioonist Visual Studio 2010 kasutatakse Test Manager'i tööriista abivahendina ERP-tarkvara MS Dynamics AX testimisel ja testide automatiseerimisel. Kuna ERP-tarkvara MS Dynamics NAV'i arenduskeskkonda ei ole seni veel üle viidud Visual Studio platvormile ja baseerub tänini C/SIDE keskkonnal, siis ei ole NAV'i arendamisel Test Manager'i tööriista kasutamine tehnoloogiliselt võimalik.

3 Tarkvara automaattestimine

Traditsiooniline käsitsi testimine on rutiinne tegevus, mille automatiseerimine võimaldab tagada rakenduse püsiva kvaliteedi. Võrreldes käsitsi testimisega, kus peamisteks eelisteks on kasutajakogemuse hindamine inimese poolt ja teatud testitüüpide korral odavam maksumus, siis automaattestimise suurimaks eeliseks on palju lühema aja jooksul testidega kaetava koodi oluliselt suurem maht.

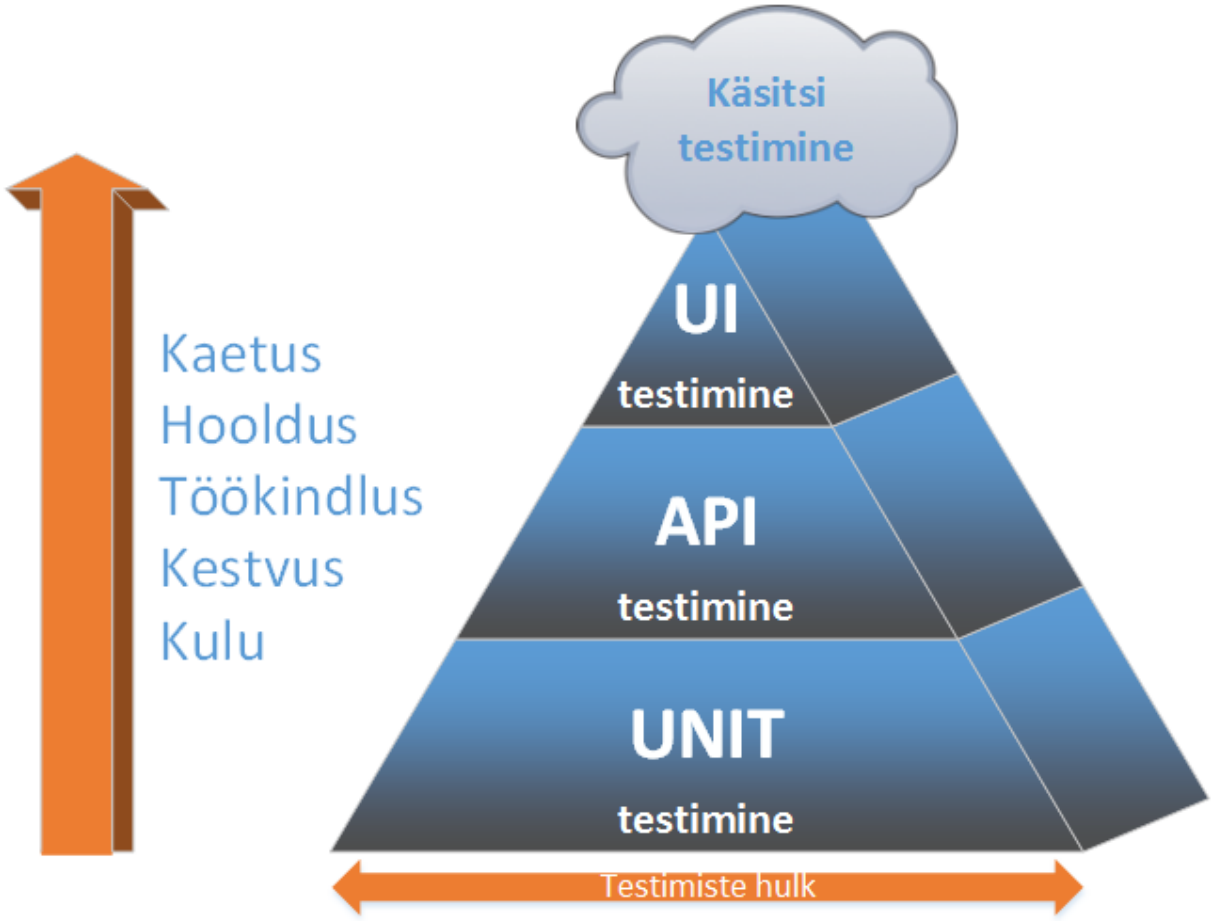
3.1 Automaattestimise metoodikad

Tavapäraselt luuakse uuele funktsionaalsusele erinevatel testitasemetel automaatsed testid. Laialdaselt on levinud tarkvara arendusprotsessis automaattestimise eraldamine kolmeks tasemeks [14]:

- kasutajaliidese testimine (UI *testing*);
- rakendusliidese testimine (API *testing*);
- ühiktestimine (*unit testing*).

Joonisel 3.1 on välja toodud automaattestimise püramiid, mis kirjeldab erinevate testimise tasemete mahtu ja vormi regressioontestimisel. Võimalikult madalal testitasemel proovitakse katta võimalikult suur osa funktsionaalsusest. Mida rohkem tipu poole, seda kulukamaks ja ajamahukamaks testimine läheb. Püramiidi tipus paikneb käsitsi testimine, mille maht peaks olema võimalikult väike. Suurem osa rakenduse testimisest peaks olema kaetud alumistes “automaatkihtides” [14-15].

Regressioontestimise peamiseks eesmärgiks on veenduda, et funktsionaalse muudatuse tagajärjel kõik tarkvara osad endiselt korrektselt toimivad ja et arenduse mõjupiirkonnas olev standardfunktsionaalsus vastab endiselt nõuetele. Regressioontestimine on parim meetod majandustarkvara testimise automatiseerimiseks, kuna tarkvara standardfunktsionaalsuse kohta on kirjeldatud testilood, mida saab korduvalt käitada peale iga muudatuse teostamist.



Joonis 3.1: Automaattestimise püramiid

Ühiktestimine – tarkvara testimise alumine tase, kus testitakse üksikute komponentide või süsteemide toimimist vastavalt nõuetele [16-17]. Selle eesmärk on testida, kas iga tarkvara üksus töötab nii nagu peab [16]. Tegemist on kõige väiksema testitava osaga tarkvarast [16]. Üldjuhul on sellel mõni üksik sisend ja tavaliselt üks väljund/tulemus [16]. Seda, mida „üksuse“ all mõeldakse, pole üheselt defineeritud [17].

Rakendusliidese testimine – tarkvara testimise keskmine tase, kus testitakse API-keskset arhitektuuri kui rakenduse ning andmebaasi vahele on loodud äriloogikat haldav teenuskiht. Rakendusliidese testid võimaldavad mugavalt testida API-kihis paiknevat äriloogikat ning integratsiooni andmebaasiga [15]. Selles kihis toimub valdavas osas funktsionaalne testimine ja seda kihti on kõige efektiivsem katta automaattestidega [18].

Kasutajaliidese testimine – tarkvara testimise ülemine tase, kus tegeletakse lõppkasutajatele mõeldud rakenduste kasutajaliideste testimisega. Kasutajaliidese testimisel võib väliseid teenuseid simuleerides keskenduda ainult testitavale rakendusele. [15]

3.2 Majandustarkvara juurutusmetoodika

Tarkvaraarenduse metoodikaid on maailmas mitmeid, neist levinuimad on koskmudel (*waterfall*) ja agiilne (*agile*) meetod [19].

MS Dynamics Sure Step on Microsofti-poolne metoodika, mida kasutatakse MS Dynamics'i toodete, sealhulgas NAV'i juurutusprojektide läbiviimiseks [20]. Microsoft ERP-partneritele suunatud metoodika kirjeldab protsesse ja põhimõtteid, mis on vajalikud MS Dynamics'i toodete korrektseks juurutamiseks [20].

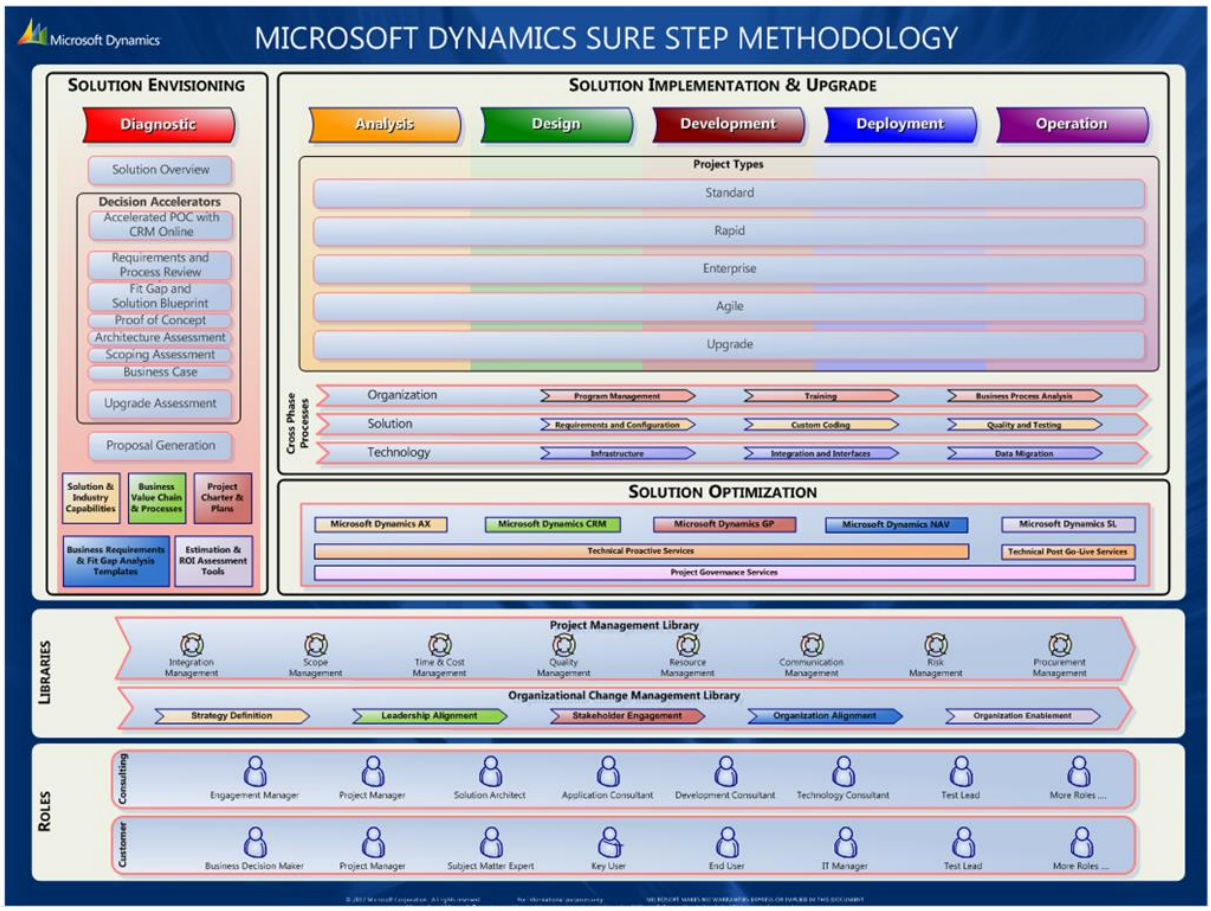
MS Dynamics Sure Step põhiprotsess (*masterflow*) on interaktiivne ja dünaamiline tegevuste skeem (joonis 3.2) [20], mis illustreerib kindlas loogilises järgnevuses sammud, mida on vaja läbida, et majandustarkvara juurutada [20]. Põhiprotsess on kohandatav vastavalt projekti keerukusele ja kliendi poolt projektile esitatavatele nõuetele [20]. Iga sammu juures on määratletud [20]:

- tegevuse eesmärk;
- ettevõtte poolt tegevuse eest vastutaja ja tema roll;
- kliendi poolt tegevuse eest vastutaja ja tema roll;
- tegevuse väljunddokumendid.

Sure Step põhiprotsessis jagatakse tegevused faasidesse, kus iga faasi väljund on sisendiks järgmise faasi tegevustele [20]:

- diagnostikafaas (*diagnostic*);
- analüüsifaas (*analysis*);
- disainifaas (*design*);
- arendusfaas (*development*);
- juurutusfaas (*deployment*);
- käivitusfaas (*operations*).

MS Dynamics Sure Step agiilne projektitüüp eirab agiilse tarkvaraarenduse manifestis üht kõige olulisemat punkti - varased ja sagedased tarned kliendile, käivitusfaas toimub ikkagi *big bang* viisil. Agiilne projektitüüp tähendab sisuliselt ainult arendusfaasi teostamist agiilsel meetodil ning üldine ülesehitus jääb ja põhineb koskmeetodil. Disaini- ja arendusprotsess toimub sprintidena, millele järgneb klassikaline juurutusfaas ja käivitusfaas.



Joonis 3.2: Microsoft Dynamics Sure Step põhiprotsess

3.3 Majandustarkvara MS Dynamics NAV'i ülevaade

MS Dynamics NAV (varem Navision) on ERP-tarkvara, mis on mõeldud väikese ja keskmise suurusega ettevõtetele [21]. Toode on osa Microsoft Dynamics'i perekonnast. Rohke funktsionaalsuse ja kohandamisvõimega majandustarkvara võimaldab ettevõtetel hallata oma äri [21], sealhulgas finantsi, tootmist, müüki, tarnet, projektijuhtimist, teenuseid jm.

Tarkvara ajalugu ulatub 1983. aastasse, kui Taanis loodi ettevõte PC&C (Personal Computing & Consulting) [22]. 1984. aastal andis ettevõte välja tarkvara PC-Plus, mida võib kaudselt pidada Navisioni eelkäijaks [22]. Päris esimene versioon Navision tarkvarast ilmus 1987. aastal, kandes siis nime Navigator ja olles üks esimesi klient-server arhitektuuriga majandusinfosüsteeme [22]. Tarkvara edust tingituna muutis 1995. aastal PC&C oma nime Navision Software A/S'is [23]. Sel aastal anti välja ka esimene Microsoft Windows operatsioonisüsteemil töötav tarkvaraversioon, mis kandis nime Navision Financials. Uuele platvormile üleminekuks kirjutati lähtuvalt Microsoft Windows 95 disainijuhendist täielikult ümber senine rakendus, äriloogika poolt muutmata [22]. 2000. aastal ühines Navision Software

A/S teise Taani majandustarkvara arendava ettevõttega Damgaard A/S. Ühinenud firma nimeks sai NavisionDamgaard A/S, mis 2001. aastal muudeti Navision A/S [22]. Koos jõuti välja anda paar versiooni, mis kandsid nime Navision Solutions ja Navision Attain [22].

2002. aastal ostis Microsoft ära Navision A/S ettevõtte, liites selle Microsoft Business Solutions'i tootegrupiga ja muutes tarkvara nime Microsoft Business Solutions Navision'iks [22]. Peale 2005. aasta brändi- ja tootenime muutust sai majandustarkvara nimeks Microsoft Dynamics NAV [22].

Tabel 3.1 annab ülevaate majandustarkvara peamistest versioonidest, mis on Microsofti poolt tänaseks päevaks välja antud.

Tabel 3.1: MS Dynamics NAV'i majandustarkvara versioonid

Versiooni nimetus	Ilmumise aasta
Navision Attain 3.60	2002
Microsoft Business Solutions Navision 3.70	2003
Microsoft Business Solutions NAV 4.00	2005
Microsoft Dynamics NAV 5.00	2007
Microsoft Dynamics NAV 2009 ("6.00")	2008
Microsoft Dynamics NAV 2009 SP1 ("6.00")	2009
Microsoft Dynamics NAV 2009 R2 ("6.00")	2010
Microsoft Dynamics NAV 2013 ("7.00")	2012
Microsoft Dynamics NAV 2013 R2 ("7.10")	2013
Microsoft Dynamics NAV 2015 ("8.00")	2014
Microsoft Dynamics NAV 2016 ("9.00")	2015
Microsoft Dynamics NAV 2017 ("10.00")	2016

Vaatamata sellele, et majandusinfosüsteem on aastate jooksul kandnud erinevaid nimesid ja välja antud erinevate ettevõtete poolt, on selle algse funktsionaalsuse tuumiklahendus jäänud suures osas samaks, millele on aja jooksul lisandunud uusi äriloogika mooduleid ja tehnoloogilise platvormi uuendusi.

Lahendus hõlmab muu hulgas järgmist [24]:

- finantsjuhtimine;
- müügi, turunduse ja teenushaldus;
- projektihaldus;
- tarneahela haldus;

- tootmise planeerimine, juhtimine;
- ost ja varude juhtimine;
- kliendisuhete juhtimine (CRM);
- personalijuhtimine;
- logistika haldus;
- ärianalüüs/aruanded.

Toote funktsionaalsus jaguneb kaheks paketiks [24], mis on detailselt kirjeldatud Lisas 1:

- startpakett (*Starter Pack*);
- laiendatud pakett (*Extended Pack*).

Toode on lokaliseeritud paljude erinevate regioonide jaoks ja on kasutatav ka eesti keeles ning sisaldab kohalikust seadusandlusest tulenevat vajalikku funktsionaalsust. Dynamics NAV 2017 EE (Eesti) lokalisatsiooni funktsionaalsuse moodulis sisaldub [24-25]:

- eestikeelne kasutajaliides;
- eestindatud demokeskkond;
- versioonivahetuse vahendid;
- eesti kohaliku funktsionaalsuse abiinfo eesti ja inglise keeles;
- kohalikust seadusandlusest ja turunõudlusest tulenev lisafunktsionaalsus (näiteks SEPA, KMD).

3.3.1 MS Dynamics NAV'i tehniline arhitektuur

MS Dynamics NAV oli kuni 2009. aasta versioonini kahekihilise (*two-tier*) arhitektuuriga [26]. Kahekihilise ehk klient-server mudeli korral ei pääse kasutaja otse andmetele ligi, vaid talle kuvatakse andmed klientprogrammi kaudu, mis suhtleb andmebaasiserveriga [27]. Klassikaline kahekihiline arhitektuur leiab kasutust ka uuemate versioonide puhul, kuid seda vaid tarkvara haldamisel ja arendamisel [26].

MS Dynamics NAV'i uuemad versioonid on arhitektuurilt kolmekihilise (*three-tier*) ülesehitusega, sisaldades endas järgnevaid tasandeid [28]:

- **esitluskiht** ehk **kliendikiht** (*presentation layer, client tier*) – kasutajaliides, mille kaudu toimub ligipääs süsteemile, võimaldades ühendumiseks erinevaid klientrakendusi ning andmeliideseid (SOAP, ODATA);
- **äriloogika kiht** ehk **serverikiht** (*business logic layer, server tier*) – MS Dynamics NAV'i rakendusserver, mis vahendab andmeid esitluskihi ja andmekihi vahel; selles kihis toimub vastavalt äriloogikale andmete töötlemine;
- **andmekiht** (*data layer, data tier*) – MS SQL'i andmebaasiserver, koos MS Dynamics NAV'i andmebaasi komponentidega.

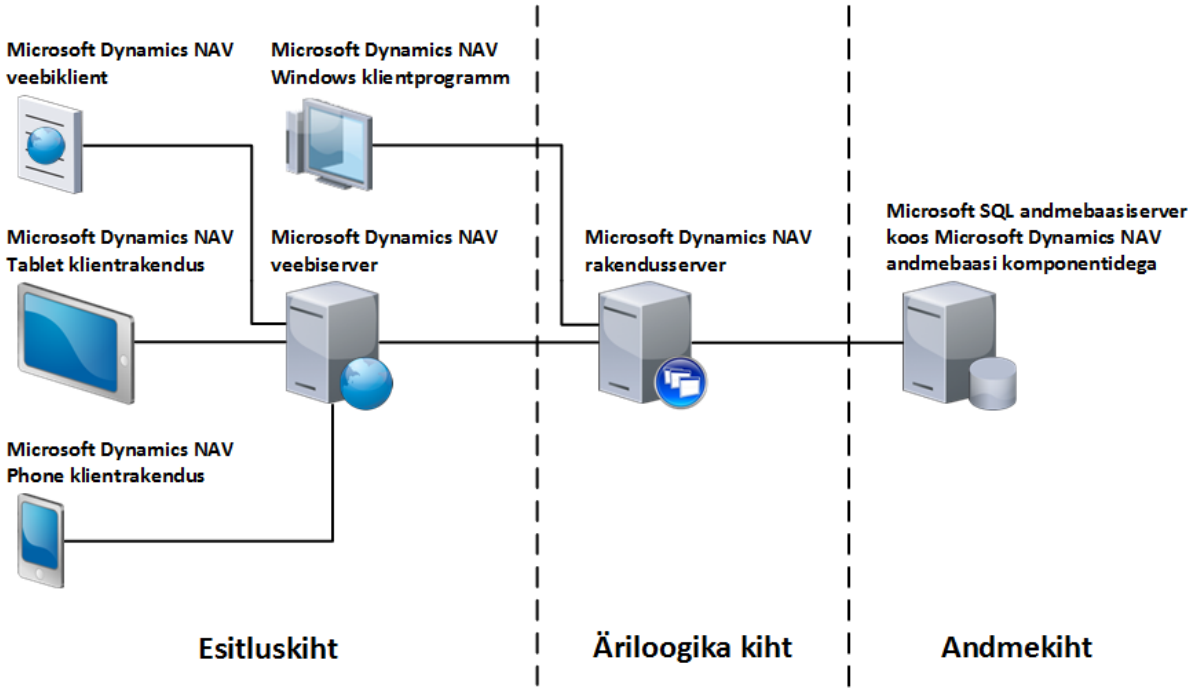
Kui kahekihilises süsteemis toodi kasutaja arvutist eraldiseisvasse serverisse vaid andmed, siis kolmekihilises süsteemis on eraldi kihti viidud ka protsessid [27]. Näiteks arve konteerimise protsessi puhul teostati kahekihilises süsteemis konteerimine kasutaja arvutis, nüüd toimub see äriloogika kihis ja kasutajale kuvatakse tema arvutisse lihtsalt tulemust [27].

Kolmekihiline arhitektuur toetab erinevaid tarkvarapaigalduse viise, seda lähtuvalt loodava lahenduse eesmärgist [28]. Järgnevalt on välja toodud 4 levinumat paigalduse meetodit:

- Kõik kolm kihti paigaldatud ühte serverisse. Sellise lahenduse plussideks on eelkõige töökindlus ja mugavus, kuna tõrkeid võrguühenduse ja/või serveri turvasätetega esineb vähem ja keskkond on kergemini hallatav [28]. Kasutatakse enamasti demo- ja/või arenduskeskkondade puhul [28].
- MS Dynamics NAV Server'i teenus ja keskkonna andmebaas on ühes serveris, klientprogramm teises serveris ja/või kasutaja seadmes [28]. Kasutatakse vähem ressursinõudlikumate lahenduste korral, kus näiteks ettevõttesisesest poliitikast tulenevalt võidakse klientprogrammi kasutada eelkõige terminal serveri kaudu.
- MS Dynamics NAV Server'i teenus ja klientprogramm ühes serveris ja/või kasutaja seadmes, keskkonna andmebaas teises serveris [28]. Kasutatakse ressursinõudlikumate lahenduste korral, kus andmebaasiserver võib samaaegselt kanda mitut rolli (näiteks aruandlus- ja/või analüüsilahenduse tarbeks) ja töödeldavad andmemahud on suuremad.
- Kõik kolm kihti ja selle komponendid on erinevates serverites [28]. Kasutatakse keerukamate ning ressursinõudlikumate lahenduste korral, kus üheskoos võib olla mitu MS Dynamics NAV'i instantsi.

Joonisel 3.3 on välja toodud kolmekihilise arhitektuuriga lahendus, kus kõik kolm kihti asuvad eraldi serverites ja/või seadmetes. Kasutatakse ühte MS SQL'i andmebaasiserverit, kus asub keskkonna andmebaas ja kuhu on paigaldatud MS Dynamics NAV'i andmebaasi komponendid.

Andmebaasiserver on seotud eraldiseisva MS Dynamics NAV'i rakendusserveriga. Rakendusserveris oleva keskkonnaga ühendumiseks kasutatakse kasutajate tööjaamadest MS Dynamics NAV'i Windows'i klientprogrammi või ühendutakse üle MS Dynamics NAV'i veebiserveri. Veebiserveri kaudu ühendumist kasutavad veebiklient ja mobiilsed klientrakendused.



Joonis 3.3: MS Dynamics NAV 2017 kolmekihiline arhitektuur

3.4 MS Dynamics NAV'i rakenduse testimine

MS Dynamics NAV 2017 sisaldab endas järgnevat tehnilist funktsionaalsust, mida kasutatakse rakenduse testimiseks ja mis on vajalikud rakenduse automaattestide ülesehitamisel [29]:

- testi koodiüksused (*test codeunits*);
- testitäitja koodiüksused (*test runner codeunits*);
- testi leheküljed (*test pages*);
- kasutajaliidese töötlejad (UI *handlers*);
- ASSERTERROR käsklus (ASSERTERROR *statement*).

3.4.1 Testi koodiüksused

Testfunktsioonid kirjutatakse testi koodiüksustes C/AL koodina. Kui testi koodiüksus töötab, täidab see *OnRun*-funktsiooni ja seejärel igat koodiüksuses sisalduvat testfunktsiooni. Vaikimisi käsitletakse igat testfunktsiooni eraldi andmebaasitoiminguna, kuid kasutades *TransactionModel Property*-atribuuti testfunktsioonides ja *TestIsolation Property*-atribuuti testitäitja koodiüksustes, saab kontrollida tehingute iseloomu. Vaikimisi kuvatakse testi koodiüksuste tulemused sõnumiaknas, kuid kasutades *OnAfterTestRun*-päästikprotsessi (*trigger*) testitäitja koodiüksuses, on võimalus need salvestada [29].

Testfunktsiooni tulemus on kas positiivne (SUCCESS) või negatiivne (FAILURE). Kui kontrollitavas koodis või testkoodis esineb viga, siis on tulemuseks FAILURE ja veateade sisaldub tulemuste logifailis. Isegi kui ühe testfunktsiooni tulemuseks on FAILURE siis järgmine testfunktsioon täidetakse sellest hoolimata [29].

Testi koodiüksustes sisalduvad testfunktsioonid on järgnevat tüüpi [29]:

- testfunktsioon (*test function*);
- töötlejafunktsioon (*handler function*);
- tavapärane funktsioon (*normal function*).

3.4.2 Testitäitja koodiüksused

Testitäitja koodiüksuseid kasutatakse testi koodiüksuste täitmiseks ning teiste testi haldus-, töötlus- ja aruandlusmeetmete raamistikus. Integreerumine testi haldusraamistikuga võimaldab protsessi automatiseerida ja teste käitada järelevalveta [29].

Testitäitja koodiüksused sisaldavad järgmisi päästikprotsesse [29]:

- *OnBeforeTestRun*-päästikprotsess;
- *OnAfterTestRun*-päästikprotsess.

Neid päästikprotsesse kasutatakse eeltöötluse ja järeltöötluse sooritamiseks, näiteks lähtestamiseks või testitulemuste logimiseks. *OnBeforeTestRun*-päästikprotsessi rakendamisel käivitatakse see enne iga testfunktsiooni käivitamist. *OnAfterTestRun*-päästikprogrammi rakendamisel käivitatakse see iga testfunktsiooni järgselt ning samas pärsitakse tulemuste sõnumi automaatset kuvamist [29].

OnBeforeTestRun- ja *OnAfterTestRun*-päästikprotsessid käivituvad alati eraldi tehinguna, sõltumata *TestIsolation Property*'i väärtusest, *TransactionModel Property*'i väärtusest või test-funktsiooni tulemustest. Päästikprotsesside kasutamine ei ole kohustuslik ja vaikimisi pole need testide täitja koodiüksustes saadaval. Nende päästikprotsesside kasutamiseks tuleb käsitsi uus funktsioon lisada ja õige signatuur määratleda [29].

3.4.3 Testi leheküljed

Testi lehekülg jäljendab tegelikku lehekülge, kuid ei too esile kasutajaliideseid klient arvutis. Testi lehekülg võimaldab koodi testida C/AL abil, simuleerides kasutaja ja lehekülje vahelist interaktsiooni [29].

Olemas on kahte tüüpi testi lehekülgi [29]:

- *TestPage*, mis on tavaline lehekülg ja võib olla mistahes liiki leht; see hõlmab endas lehe osi või alamlehti;
- *TestRequestPage*, mis kujutab endas aruande päringu lehekülge (*request page*).

Punktmärgistust (*dot notation*) kasutades pääseb ligi leheväljadele ja lehe või väljade atribuutidele. C/AL funktsioonide abil saab testi lehekülgi avada ja sulgeda, sooritada leheküljel toiminguid ning ringi navigeerida [29].

3.4.4 Kasutajaliidese töötlejad

Automatiseeritud testide loomiseks on vaja luua olukordi, kus kontrollitav kood nõuab kasutajapoolset interaktsiooni. Nõutud kasutajaliidese asemel käivitatakse kasutajaliidese-töötleja. Kasutajaliidesetöötleja väljund on sama kui kasutajaliidese väljund. Näiteks testfunktsioon, mille tüübiks on *ConfirmHandler*, täidab CONFIRM-funktsiooni. Kui kontrollitav kood nõuab CONFIRM-funktsiooni, siis selle asemel rakendatakse *ConfirmHandler*-funktsioon. Eeldatava küsimuse kuvamist CONFIRM-funktsioonis kinnitatakse koodi kirjutamisega *ConfirmHandler*-funktsioonis [29]. Tabel 3.2 [29] kirjeldab saadaval olevaid kasutajaliidese töötlejaid.

Tabel 3.2: Kasutajaliidese töötlejad

Funktsiooni tüüp	Otstarve
MessageHandler	Töötleb MESSAGE-käske
ConfirmHandler	Töötleb CONFIRM-käske
StrMenuHandler	Töötleb STRMENU-käske
PageHandler	Töötleb konkreetseid lehti, mida ei käivitata modaalselt
ModalPageHandler	Töötleb konkreetseid lehti, mis käivitatakse modaalselt
ReportHandler	Töötleb konkreetseid aruandeid
RequestPageHandler	Töötleb konkreetse aruande päringulehte

Iga töödeldava lehekülje jaoks luuakse konkreetne töötleja ja iga töödeldava aruande jaoks eraldi aruandetöötleja. Kui testi koodiüksus testitäitja koodiüksuses käitatakse, siis iga töötlejata kasutajaliides selle testfunktsioonides põhjustab tõrke. Testitäitjat kasutamata kuvatakse iga töötlejata kasutajaliides tavapäraselt [29].

3.4.5 ASSERTERROR märksõna

Rakendust testides tuleb kindlaks teha, et kood toimib ootuspäraselt nii edu kui tõrke korral. Neid nimetatakse positiivseteks ja negatiivseteks testideks. Selleks, et kindlaks teha, kuidas rakendus toimib tõrgete korral, kasutatakse ASSERTERROR märksõna. ASSERTERROR märksõna määrab, kas märksõnale järgnevas lauses võib käitusajal oodata tõrget. Kui ASSERTERROR märksõnale järgnev lause põhjustab vea, siis liigutakse ilma probleemideta edasi testfunktsiooni järgmise lause täitmise juurde. Kui ASSERTERROR märksõnale järgnev lause pole vigane, siis põhjustab ASSERTERROR märksõna ise tõrke ning testfunktsioon annab tulemuseks FAILURE [29].

3.5 MS Dynamics NAV'i automaattestimise ülevaade

Enne MS Dynamics NAV'i rakenduse kasutuselevõttu tuleb testida selle funktsionaalsust, eriti just teostatud lisaarenduste võimalike kaasmõjude osas. Testimine toimub iteratiivse mudeli kohaselt, seega on oluline luua korratavaid katseid ja soovitatavalt peaks neid olema võimalik automatiseerida [29].

ERP-tarkvara Dynamics NAV'i automaattestimise töövahendi testi koodiüksused rakenduvad ainult äriloogika kihis, mistõttu peab arvestama, et automaattestimine ei kata kasutajaliide kihis erinevate kasutajaliidese komponentide testimist.

Dynamics NAV'i rakenduse testtööriist on Microsofti poolt väga jõudsasti arendatav komponent. Esimene väljalase sellest ilmus NAV 2009 SP1 versioonile, olles üsna piiratud ja sisaldades vaid 19 peamist süsteemi funktsionaalsust katvat testi koodiüksust ja nendes kokku 226 testfunktsiooni (tabelis 3.3) [30].

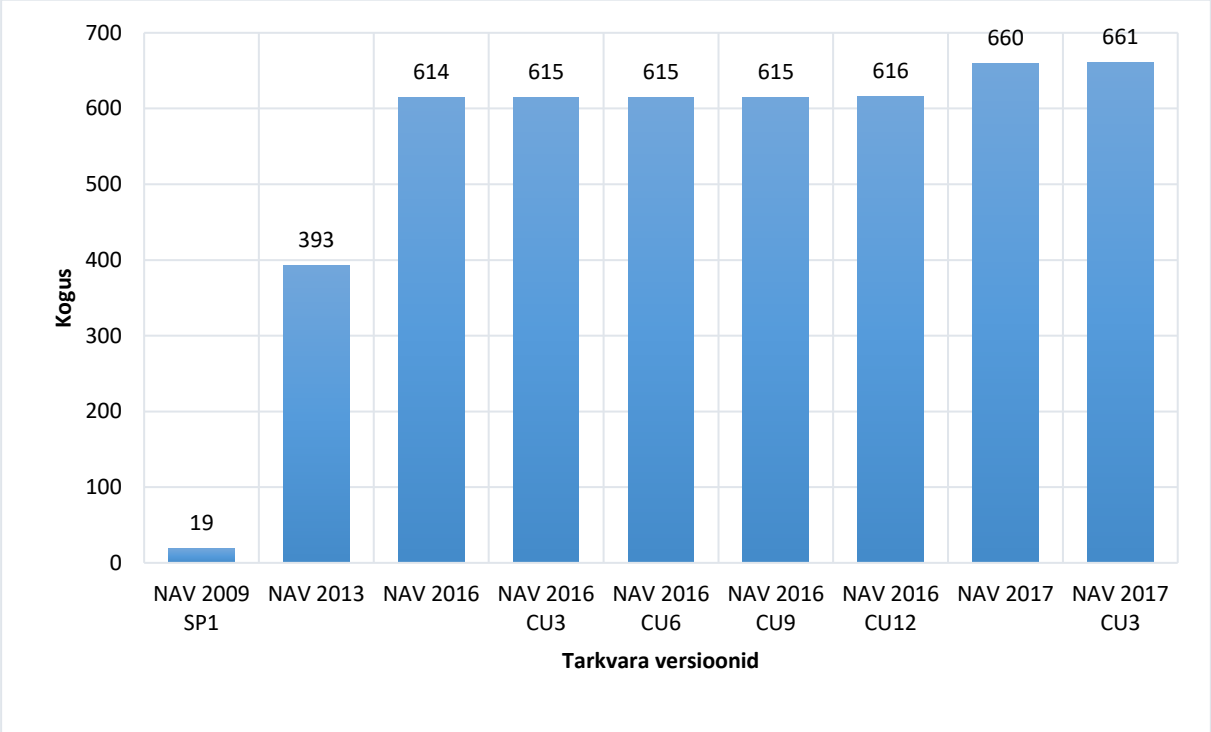
Tabel 3.3: MS Dynamics NAV 2009 SP1 versiooni testi koodiüksused

ID	Nimi
134325	Purchase Quote
134326	Purchase Blanket Order
134327	Purchase Order
134328	Purchase Invoice
134329	Purchase Return Order
134330	Purchase Credit Memo
134377	Sales Blanket Order
134378	Sales Order
134379	Sales Quote
136101	Service Order
136102	Service Contract
136103	Service Item
136300	Job Consumption Basic
137001	Online Inventory Adjustment
137007	Inventory Costing
137010	Inventory Revaluation I
137011	Inventory Revaluation II
137012	Costing Sales Returns I
137013	Costing Sales Returns II

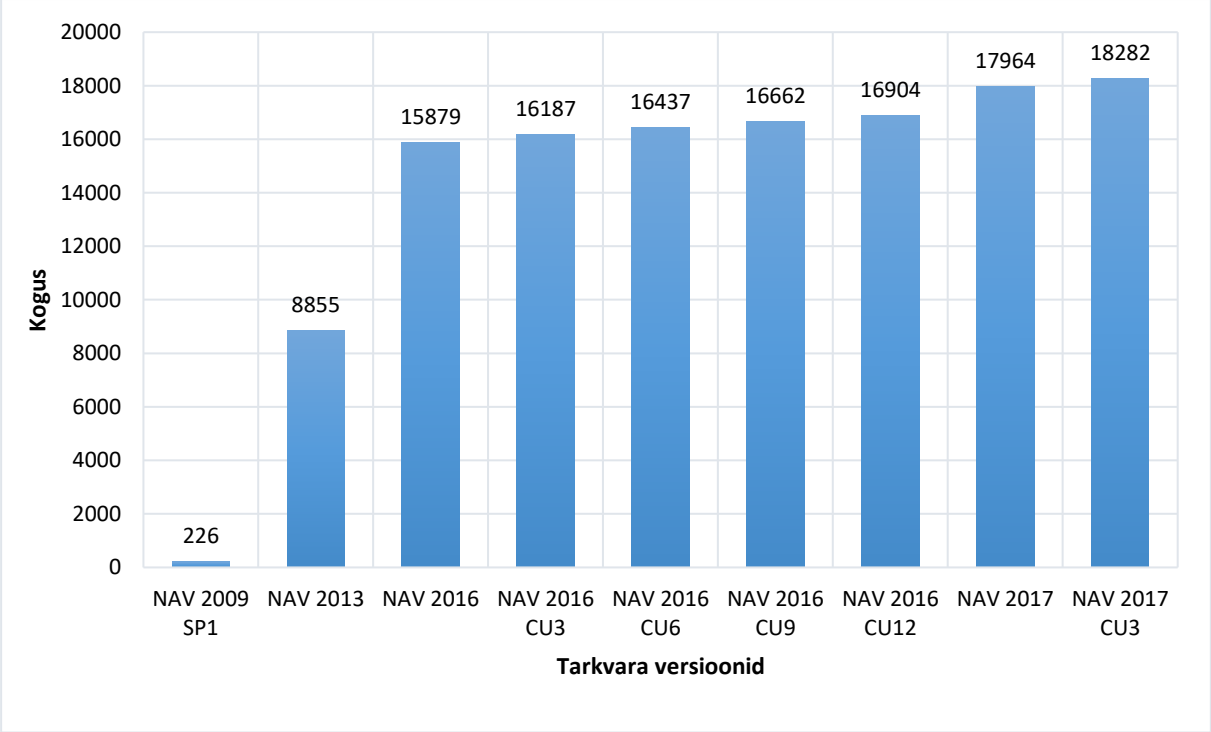
Järgmine versioon, milleks oli NAV 2013, sisaldas juba 393 testi koodiüksust ja 8855 testfunktsiooni. Vahepealsetest versioonidest (NAV 2013 R2 ja NAV 2015) jäeti rakenduse testtööriist Microsofti poolt välja. Sellise otsuse põhjuseks võib pidada asjaolu, et nende versioonide puhul läbis MS Dynamics NAV'i tarkvara olulise uuenduskuuri, seda eelkõige äriloogika poolelt [31].

Kui vanemate NAV 2009 ja NAV 2013 versioonide puhul oli tegu eraldi allalaetavate ja Microsofti poolt veel ametlikult mittetoetatud tööriistadega [32], siis alates NAV 2016 versioonist on rakenduse testtööriist juba tarkvara installatsioonimeediaga kaasas ja ka ametlikult Microsofti poolt toetatud [31].

Joonisel 3.4 on välja toodud testtööriista W1 versioonis sisalduvate testi koodiüksuste arv ja joonisel 3.5 vastavalt koodiüksustes sisalduvate testfunktsioonide arv.



Joonis 3.4: Testi koodiüksuste arv MS Dynamics NAV W1 versioonides



Joonis 3.5: Testfunktsioonide arv MS Dynamics NAV W1 versioonides

4 Juhtumiuuring

Juhtumiuuring keskendub ERP-majandusinfosüsteemi automaattestimise funktsionaalsuse rakendamisele, kus käsitletakse MS Dynamics NAV'i tarkvara automaattestimise tööriista testüksuste kogumikku standardfunktsionaalsuse ulatuses. Aluseks on võetud W1 lokaliseerimata versiooni ingliskeelse kasutajaliidesega ja rahvusvahelise baasseadistusega demoettevõtte andmestik.

4.1 MS Dynamics NAV'i testkeskkonna kirjeldus

Juhtumiuuringu tarbeks on autori poolt loodud testkeskkond, mis on paigaldatud Hyper-V virtuaalplatvormile. Tegemist on *all-in-one* keskkonnaga, mis sarnaneb arhitektuuri ja seadistuse poolest klientidele loodavate standardlahendustega, sisaldades ühes serveris koos nii MS Dynamics NAV'i klienttarkvara ja rakendusserveri komponente kui MS SQL-andmebaasiserveri tarkvara. Kuivõrd loodud keskkond on identne klientidele loodavate lahendustega, siis on samalaadse automaattestimise funktsionaalsuse rakendamine võrreldav.

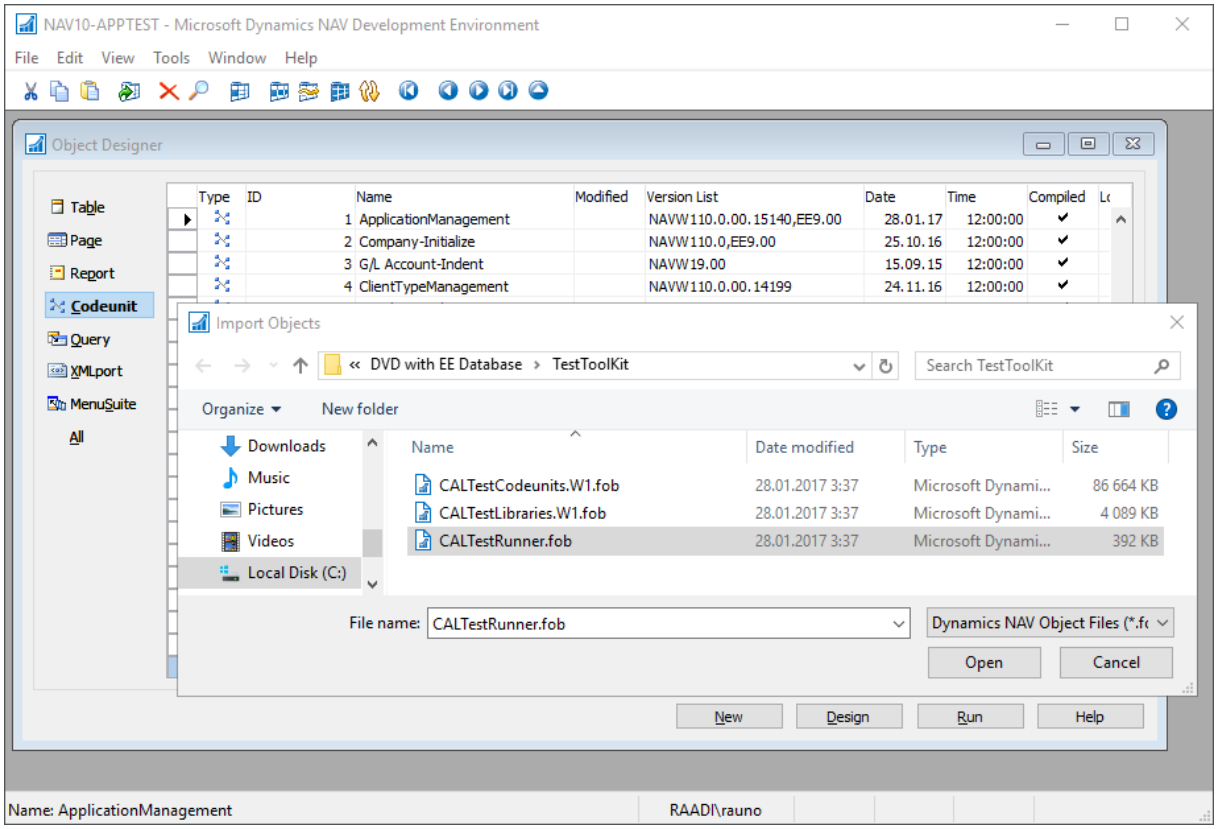
Loodud testkeskkonna seadistamisel ja tarkvara valikul on lähtutud MS Dynamics NAV 2017 süsteeminõuete dokumendist ning kasutatud võimalikult värskeid tarkvaraversioone. Serveri operatsioonisüsteemiks on MS Windows Server 2016 Standard, andmebaasitarkvaraks MS SQL Server 2016 Standard ja kontoritarkvaraks MS Office Professional 2016. Serverisse on paigaldatud majandustarkvara MS Dynamics NAV 2017 CU3 ja lisaks põhikomponentidele on paigaldatud järgnevad lisakomponendid:

- **Administration Tool** – administreerimise tööriist, mida kasutatakse instantside konfigureerimiseks ja haldamiseks [28];
- **Development Environment (C/SIDE)** – arenduskeskkond, mida kasutatakse rakenduse arendamiseks ja haldamiseks [28].

Ülejäänud MS Dynamics NAV'i komponendid pole automaattestimise vaates vajalikud ning jäeti paigaldamata.

4.2 MS Dynamics NAV'i automaattestimise paketi paigaldamine

Rakenduse testimise komplekt (*application test suite*) sisaldab endas aplikatsiooniteste, tööriistu testide haldamiseks ja käivitamiseks ning täiendavaid abifunktsioone (*helper functions*) grupeerituna teekideks [33]. Kuna MS Dynamics NAV'i demorakenduses ei sisaldu automaattestimise funktsionaalsust, siis on MS Dynamics NAV'i installatsioonimeedialt imporditud rakenduse testimise tööriistakomplekt (Test Toolkit). Automaattestimise funktsionaalsust sisalduvate objektide importimiseks on kasutatud MS Dynamics NAV Development Environment'i arendus- ja halduskeskkonda (joonis 4.1).



Joonis 4.1: Objektide importimine MS Dynamics NAV'i arenduskeskkonnas

Installatsioonimeedia **TestToolkit** kataloogist on imporditud kolm .fob laiendiga faili, mis sisaldavad endis objekte automaattestimise tarbeks (tabel 4.1) [34].

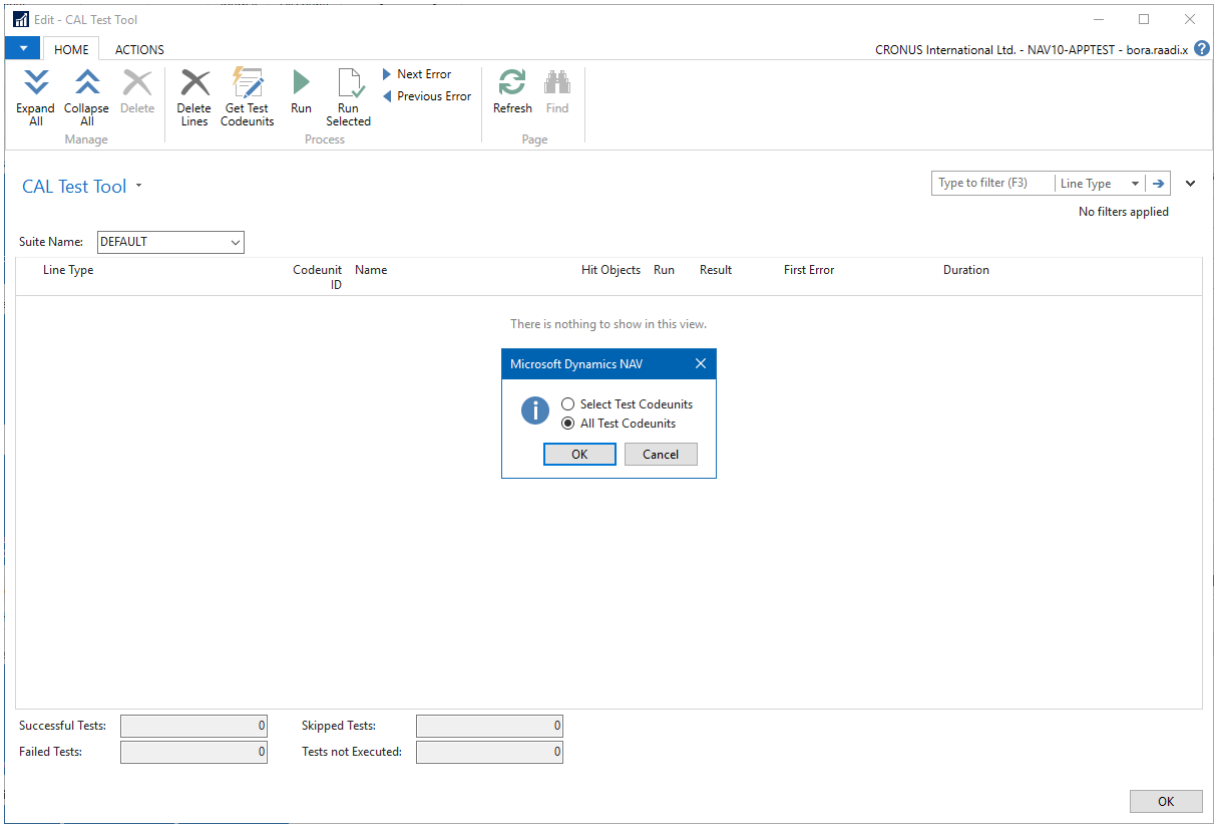
Tabel 4.1: Rakenduse testimise tööriistakomplekt

Faili nimi	Kirjeldus
CALTestCodeunits.W1.fob	Sisaldab testfunktsioonidega koodiüksuseid, mis aitavad kontrollida rakenduse erinevaid funktsionaalseid piirkondi. Antud .fob laiendiga failitestid sõltuvad testteekidest, mis sisalduvad *CALTestLibraries.W1.fob* failis. Rakenduse testi koodiüksuste

	kasutamiseks on eelnevalt vaja importida objektid kõigist kolmest .fob laiendiga failist.
CALTestLibraries.W1.fob	Sisaldab koodiüksuseid üldiste ja rakenduse põhiste funktsioonidega, mis aitavad vähendada testkoodi kordusi.
CALTestRunner.fob	Sisaldab objekte testide läbiviimiseks, sealhulgas „CAL Test Tool“ lehekülge ja muid objekte, mis toetavad rakenduse testimist.

4.2.1 MS Dynamics NAV’i standardtestide esmane käitamine

Peale automaattestimise funktsionaalsuse lisamist on sooritatud esmane standardtestide käitamine. Seda saab teha MS Dynamics NAV’i Windows klientprogrammi kaudu „CAL Test Tool“ lehelt, mis eelnevalt imporditud objektidega testkeskkonda juurde lisati. Kuigi testi koodiüksused on imporditud, pole testfunktsioonid „CAL Test Tool“ lehel vaikimisi nähtavad. Selleks, et testfunktsioone kasutada, tuleb esmalt luua testkomplekt (*test suite*) ja sinna lisada soovitud testi koodiüksused (joonis 4.2).

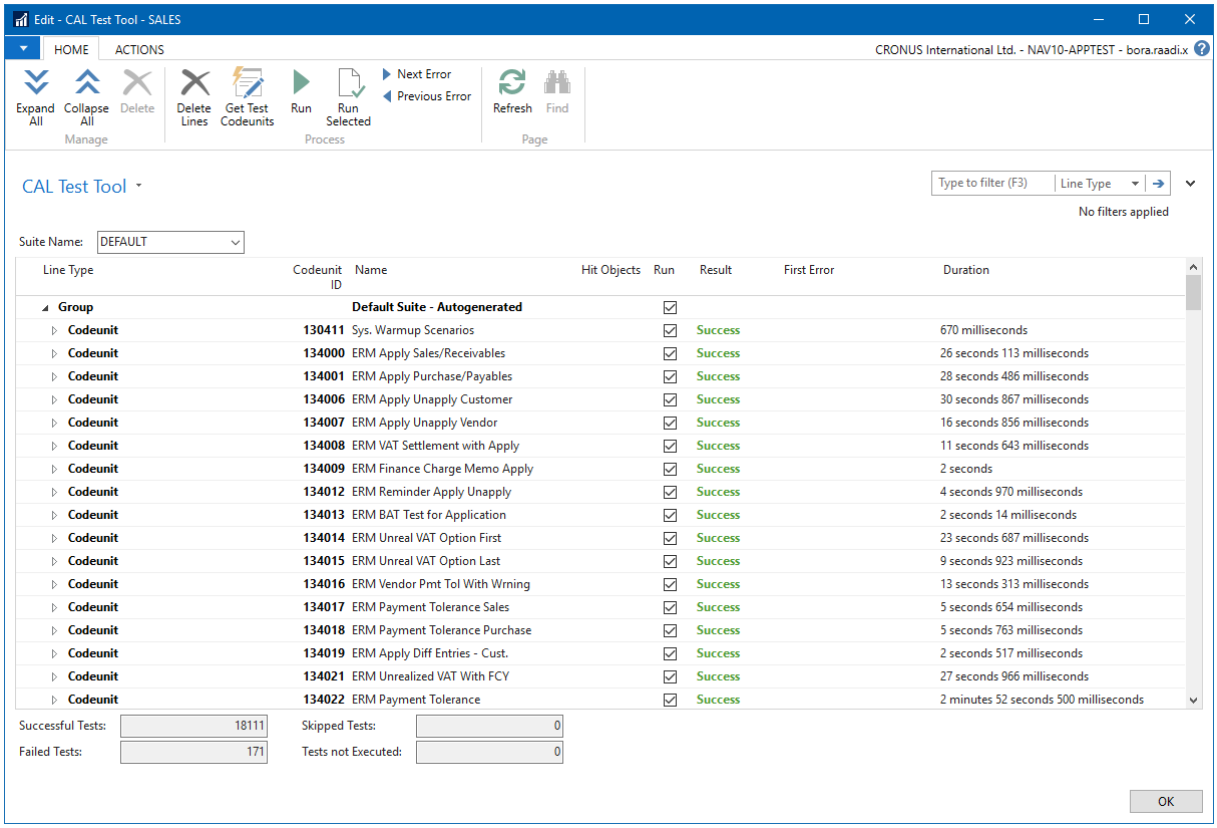


Joonis 4.2: Testi koodiüksuste lisamine testkomplekti

Esmasel testide käitamisel on soovitatav käivitada kõik testfunktsioonid, mis on rakenduse testtööriistaga kaasas. Kõiki testi koodiüksusi sisaldava testkomplekti nimeks sai vaikimisi „DEFAULT“, kuhu on lisatud kõik imporditud testi koodiüksused. Kõigi testfunktsioonide

käitamisega sooviti teada saada, mis tulemuse annab demoandmebaasiga standard keskkonnas automaattestimine.

Kõigi testide käitamine võttis kokku ligemale 5 tundi, kuid sõltuvalt keskkonna ressursist ja jõudlusest, võib see aeg olla märgatavalt lühem. Kokku sooritati esmasel testide käitamisel 18282 testi. Neist edukaid teste oli 18111 ja tõrkega lõppenud teste 171, mis teeb ebaõnnestunud testide koguprotsendiks alla 1% (joonis 4.3).



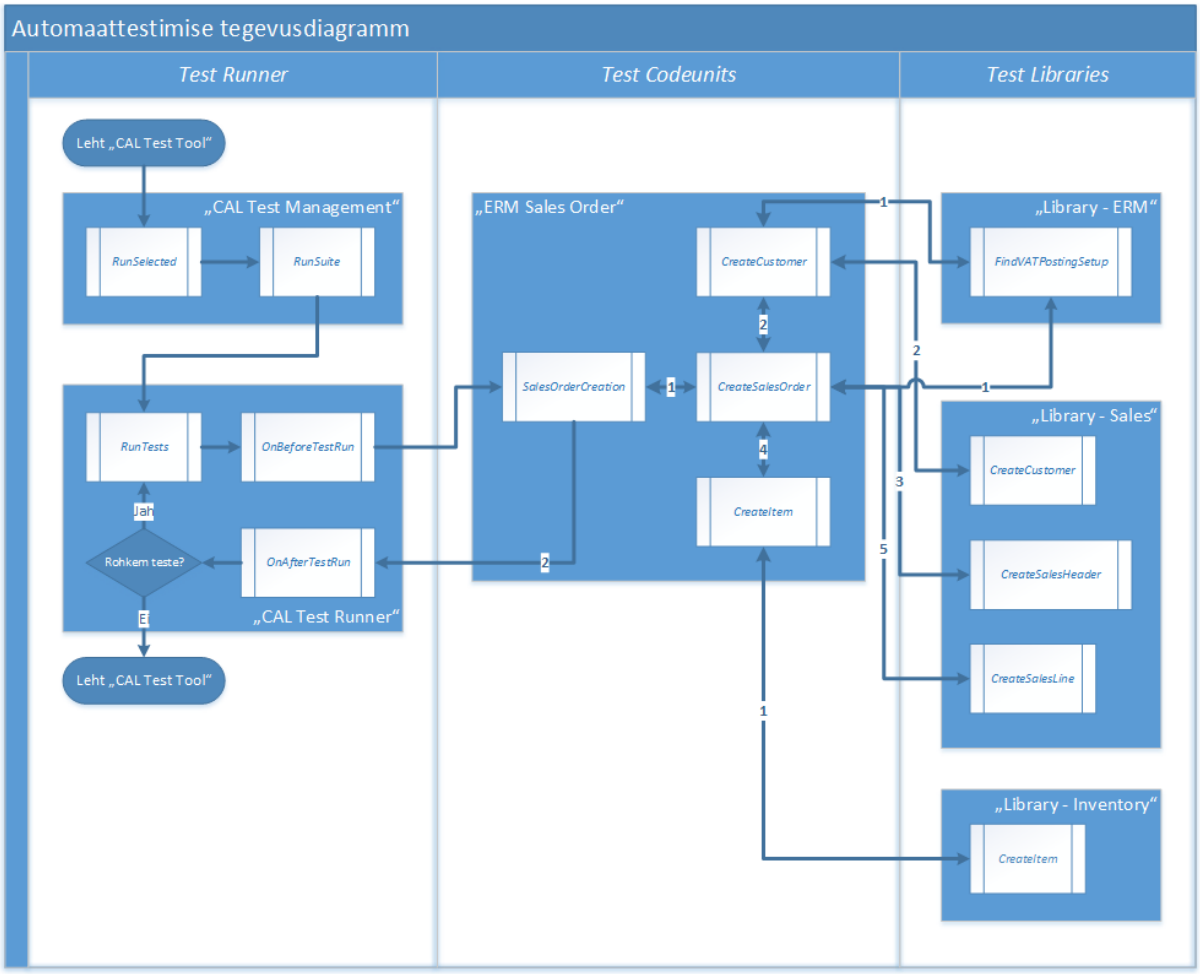
Joonis 4.3: Esmaste testide käitamise tulemus

4.3 MS Dynamics NAV'i automaattestimise tööpõhimõte

Allolev funktsionaalne kirjeldus ja tegevusdiagramm (joonis 4.4) iseloomustab MS Dynamics NAV'i testtööriista ülesehitust ja funktsionaalset liigendatust kooskõlas tabelis 4.1 kirjeldatud struktuurile. Funktsionaalse kirjelduse esitamisel on näitlikustatud ühe testi koodiüksuse üks testfunktsioon (*SalesOrderCreation*). Reaalne testprotsess läbib analoogselt, üle kõikide testi koodiüksuste, kõik valitud testfunktsioonid.

Aplikatsiooni testimise käigus teostab süsteem järgmised tegevused:

1. tavaline koodiüksus „CAL Test Management“ funktsioon *RunSelected* – valitud testfunktsioonide käivitamise ettevalmistamine;
2. tavaline koodiüksus „CAL Test Management“ funktsioon *RunSuite* – valitud testfunktsioonide käivitamine;
3. ***TestRunner*-alamliiki** koodiüksus „CAL Test Runner“ funktsioon *RunTests* – käivitab valitud testfunktsioonide koodiüksusi (Test-alamliiki koodiüksustes funktsioonid liigiga Test) ja kui käivitatavas koodiüksuses tekib äriloogika viga, siis *TestRunner*-koodiüksus jätkab tegevust:
    a. enne iga testi koodiüksuse käivitamist, käivitub automaatselt „CAL Test Runner“ **triger funktsioon** *OnBeforeTestRun*;
    b. peale iga testi koodiüksuse käivitamist, käivitub automaatselt „CAL Test Runner“ **triger funktsioon** *OnAfterTestRun*;
4. **Test-alamliiki** koodiüksus „ERM Sales Order“ **Test-liiki** funktsioon *SalesOrderCreation*:
    a. Test-alamliiki koodiüksus „ERM Sales Order“ tavaline funktsioon *CreateSalesOrder*;
    b. tavaline koodiüksus „Library - ERM“ funktsioon *FindVATPostingSetup*;
    c. Test-alamliiki koodiüksus „ERM Sales Order“ tavaline funktsioon *CreateCustomer*;
    d. tavaline koodiüksus „Library - ERM“ funktsioon *FindVATPostingSetup*;
    e. tavaline koodiüksus „Library - Sales“ funktsioon *CreateCustomer*;
    f. tavaline koodiüksus „Library - Sales“ funktsioon *CreateSalesHeader*;
    g. Test-alamliiki koodiüksus „ERM Sales Order“ tavaline funktsioon *CreateItem*;
    h. tavaline koodiüksus „Library - Inventory“ funktsioon *CreateItem*;
    i. tavaline koodiüksus „Library - Sales“ funktsioon *CreateSalesLine*.



Joonis 4.4: Automaattestimise tegevusdiagramm

4.4 Automaattestimise rakendamine arendusprotsessis

Test kood peab kindlaks tegema, et kontrollitav kood töötaks ootuspäraselt, seda nii tavaoludes kui tõrgete korral. Automaattestimisel on oht testitulemuste valeks tõlgendamiseks. Testi tulemused saavad olla nelja sorti:

- positiivne tulemus, mis kinnitab, et kontrollitud kood töötab tavakohaselt normaaltingimustes;
- negatiivne tulemus, mis kinnitab, et kontrollitud koodis esineb tõrkeid ja see ei tööta tavakohaselt;
- vale-positiivne (*false positive*) tulemus: testi tulemus on positiivne, kuid tegelikult oleks pidanud olema negatiivne, ehk koodis leiduvat viga ei suudeta tuvastada;
- vale-negatiivne (*false negative*) tulemus: testi tulemus on negatiivne, kuid tegelikult oleks pidanud olema positiivne, ehk koodis tuvastatakse viga, mida seal tegelikult ei ole.

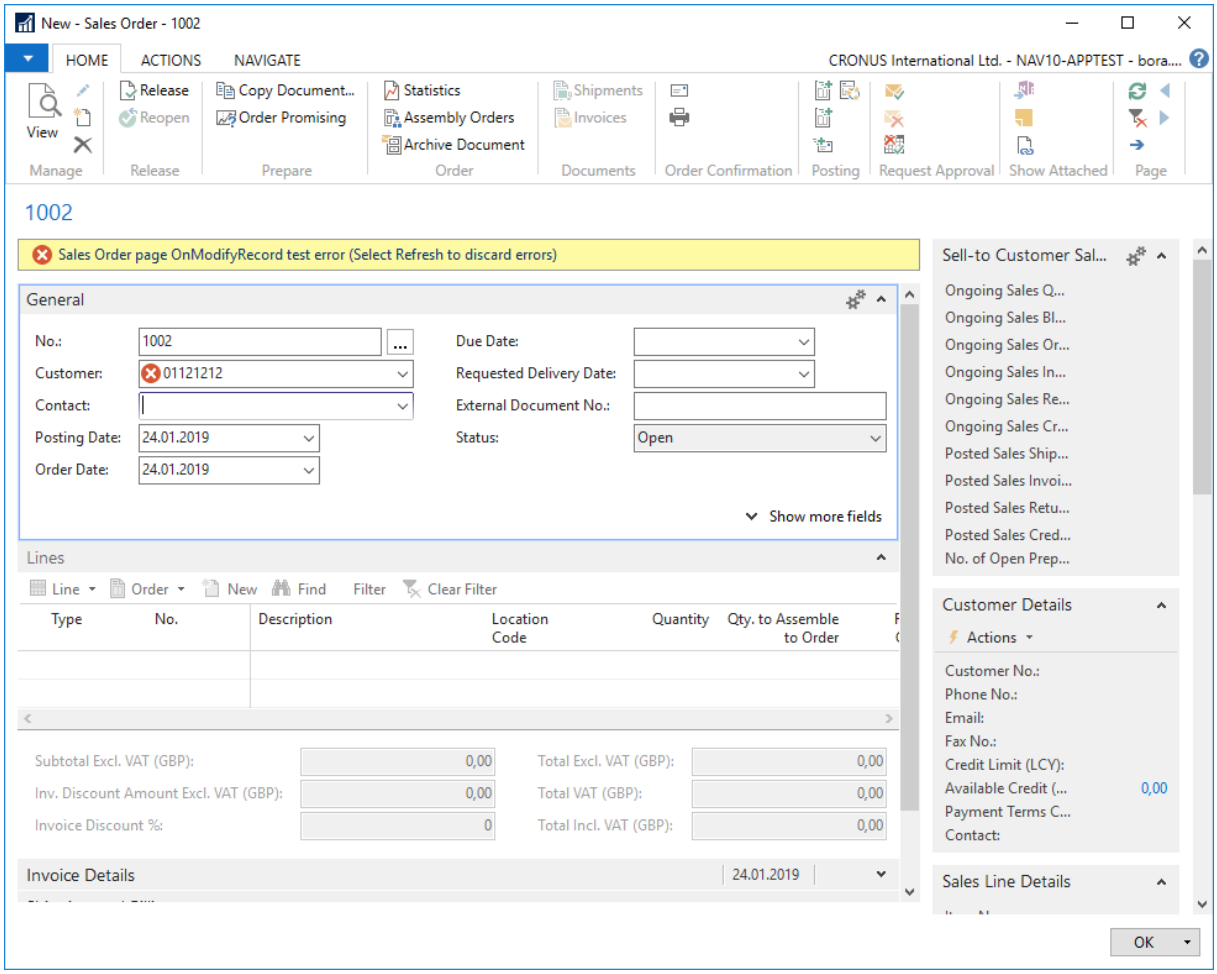
4.4.1 Arendusviga kasutajaliidese kihis

Kasutajaliidese kihis arendusvea tekitamiseks on lehe „Sales Order“ päästikprotsessi *OnModifyRecord* lisatud kood, mis ei lase muuta müügitellimuse kirjet:

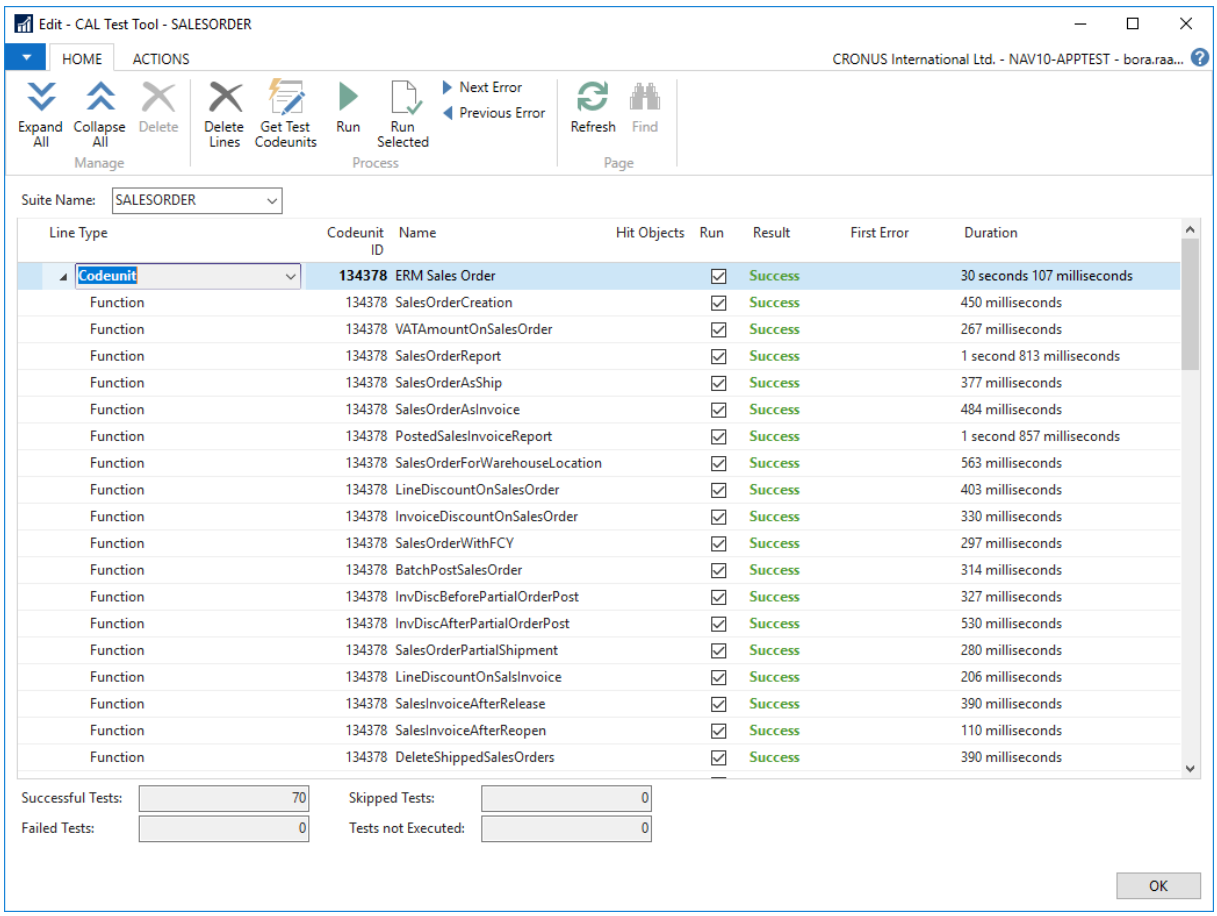
```
ERROR('Sales Order page OnModifyRecord test error'); //Test
```

Uue müügitellimuse sisestamisel kliendi valimisel saame veateate: „*Sales Order page OnModifyRecord test error*“, mis on oodatud tulemus (joonis 4.5).

Automaattestimise testi koodiüksuse „ERM Sales Order“ käivitamisel läbitakse kõik testfunktsioonid edukalt ja kasutajaliidese kihis olevat viga ei tuvastata, mis on vale-positiivne tulemus (joonis 4.6).



Joonis 4.5: Uue müügitellimuse sisestamisel saadav veateade: „*Sales Order page OnModifyRecord test error*“



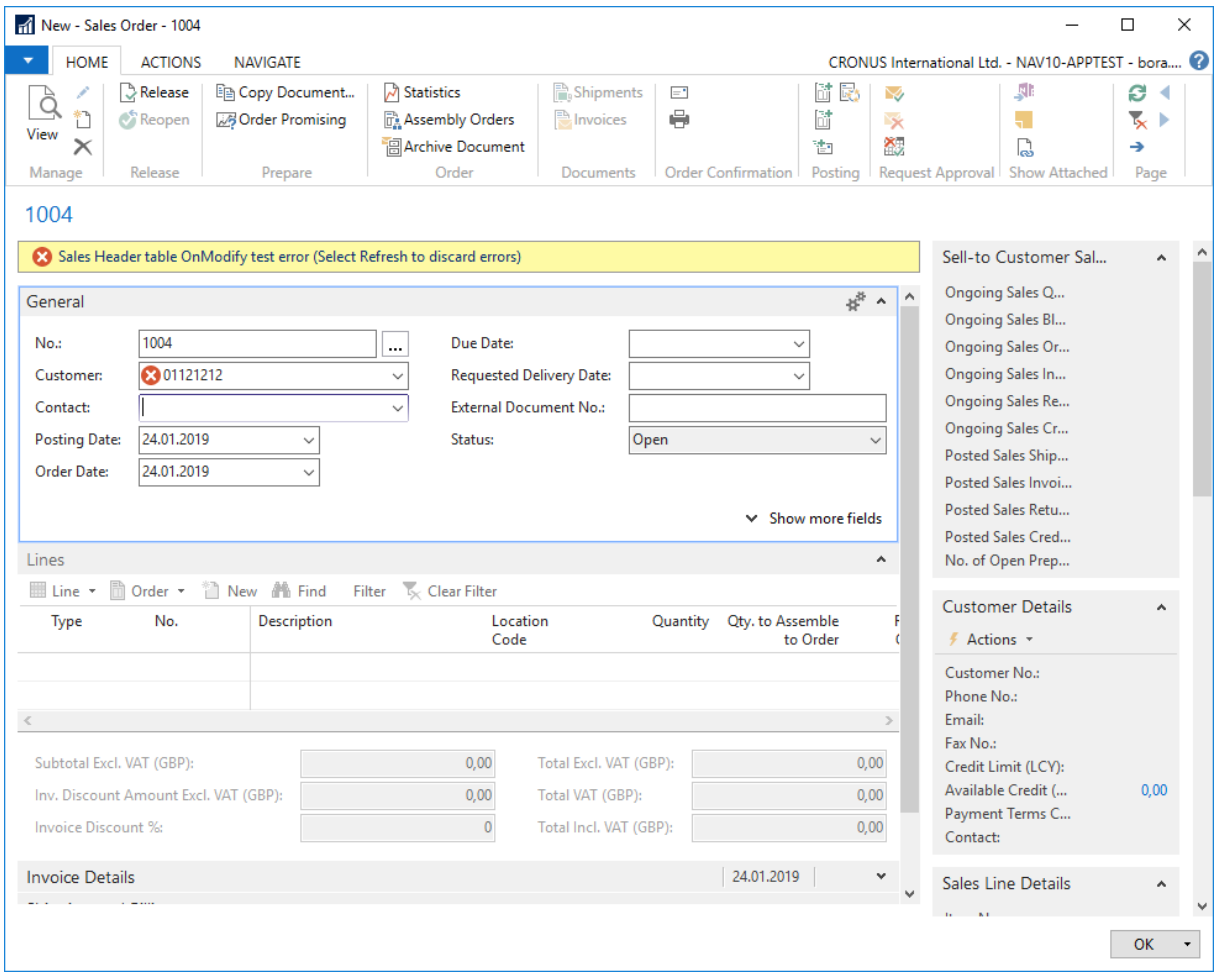
Joonis 4.6: Müügitellimuse automaattestimise tulemus peale kasutajaliidese kihis arendusvea tekitamist

4.4.2 Arendusviga äriloogika kihis

Äriloogika kihis arendusvea tekitamiseks on tabeli „Sales Header“ päästikprotsessi *OnModify* lisatud kood, mis ei lase muuta müügitellimuse kirjet:

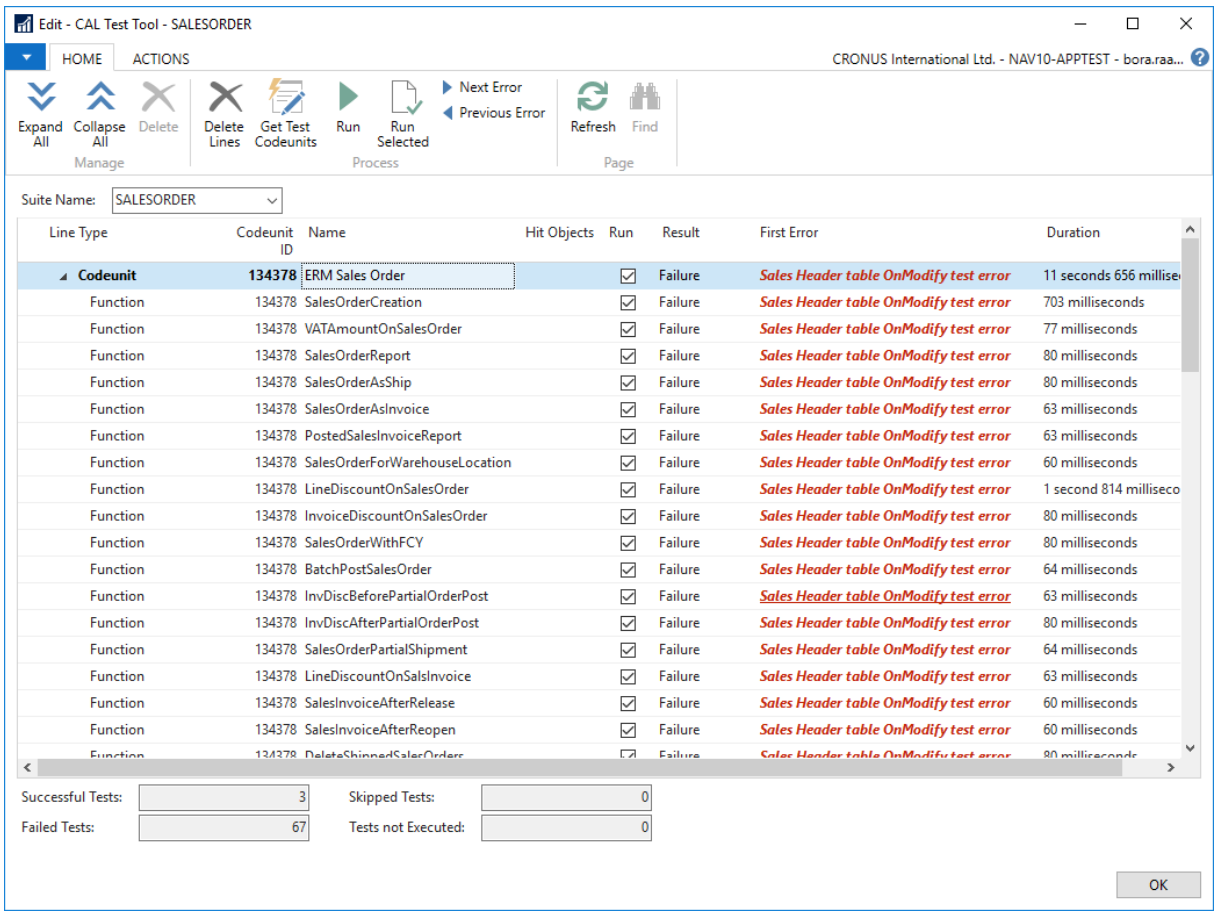
```
ERROR('Sales Header table OnModify test error'); //Test
```

Uue müügitellimuse sisestamisel kliendi valimisel saame veateate: „*Sales Header table OnModify test error*“, mis on oodatud tulemus (joonis 4.7).



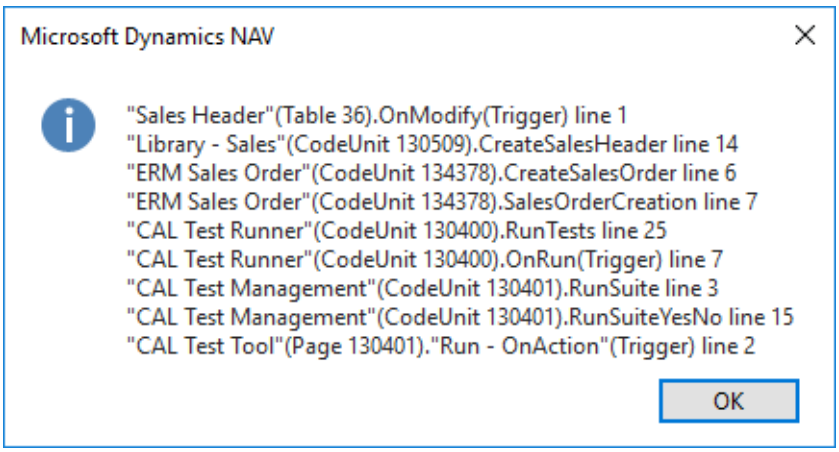
Joonis 4.7: Uue müügitellimuse sisestamisel saadav veateade: „*Sales Header table OnModify test error*“

Automaattestimise testi koodiüksuse „ERM Sales Order“ käivitamisel nurjuvad enamus testfunktsioone veaga: „*Sales Header table OnModify test error*“, mis on oodatud tulemus (joonis 4.8).



Joonis 4.8: Müügitellimuse automaattestimise tulemus peale äriloogikakihis arendusvea tekitamist

Automaattestimise tegevuste jada veaolukorra tekkimise hetkeni on joonisel 4.9.



Joonis 4.9: Automaattestimise tegevuste jada veaolukorrani

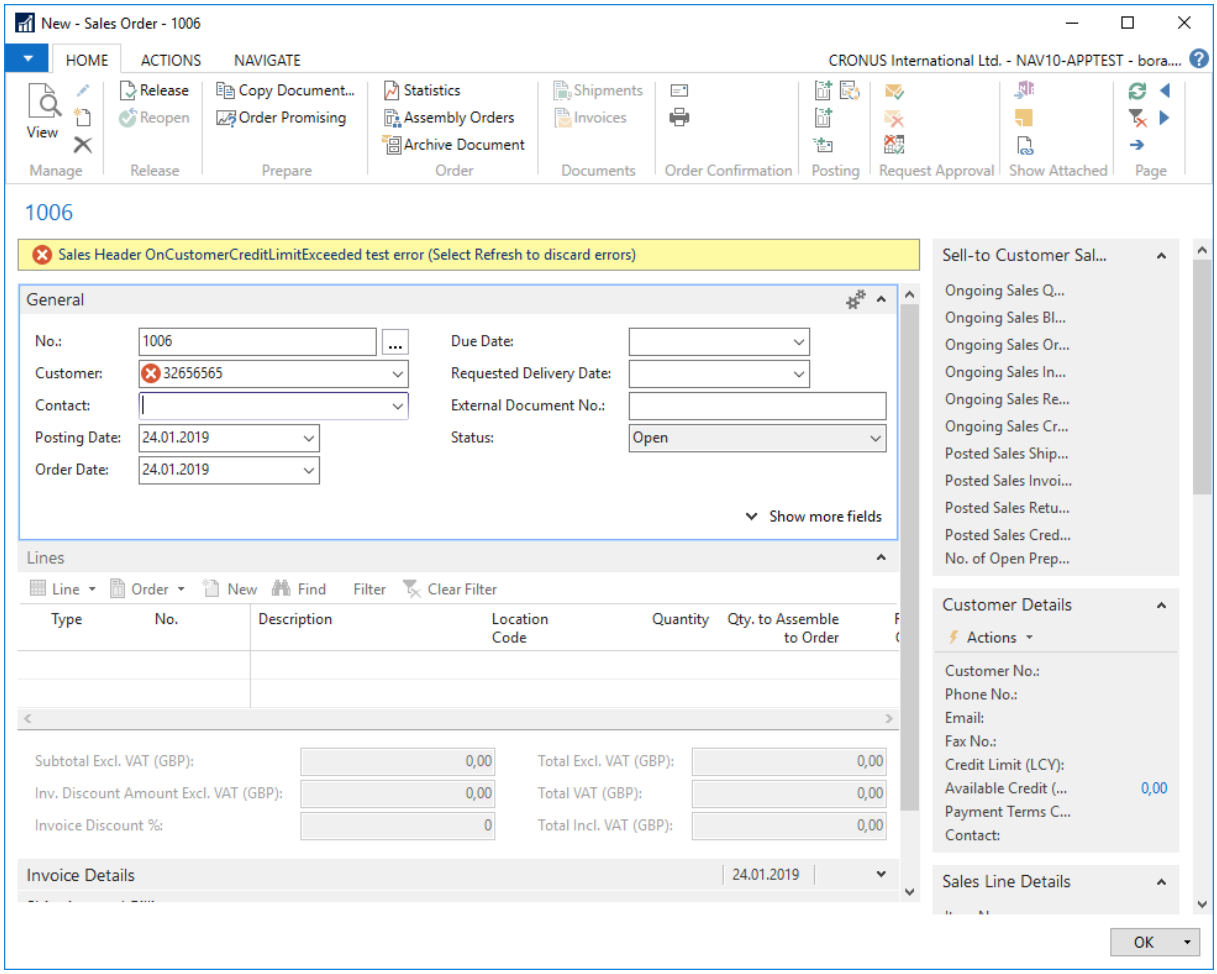
4.4.3 Andmekomplektist sõltuv arendusviga

Äriloogika kihis arendusvea tekitamiseks on koodiüksuse *Custom Functions*-päästikprotsessi [*EventSubscriber*] *OnCustomerCreditLimitExceeded* lisatud kood, mis ei lase muuta

müügitellimuse kirjet, kui tellimusele valitakse olemasolev klient, kelle krediidilimiit on ületatud:

```
ERROR('Sales Header OnCustomerCreditLimitExceeded test error');//Test
```

Uue müügitellimuse sisestamisel ja krediidilimiidi ületanud kliendi valimisel saame veateate: „*Sales Header OnCustomerCreditLimitExceeded test error*“, mis on oodatud tulemus (joonis 4.10).

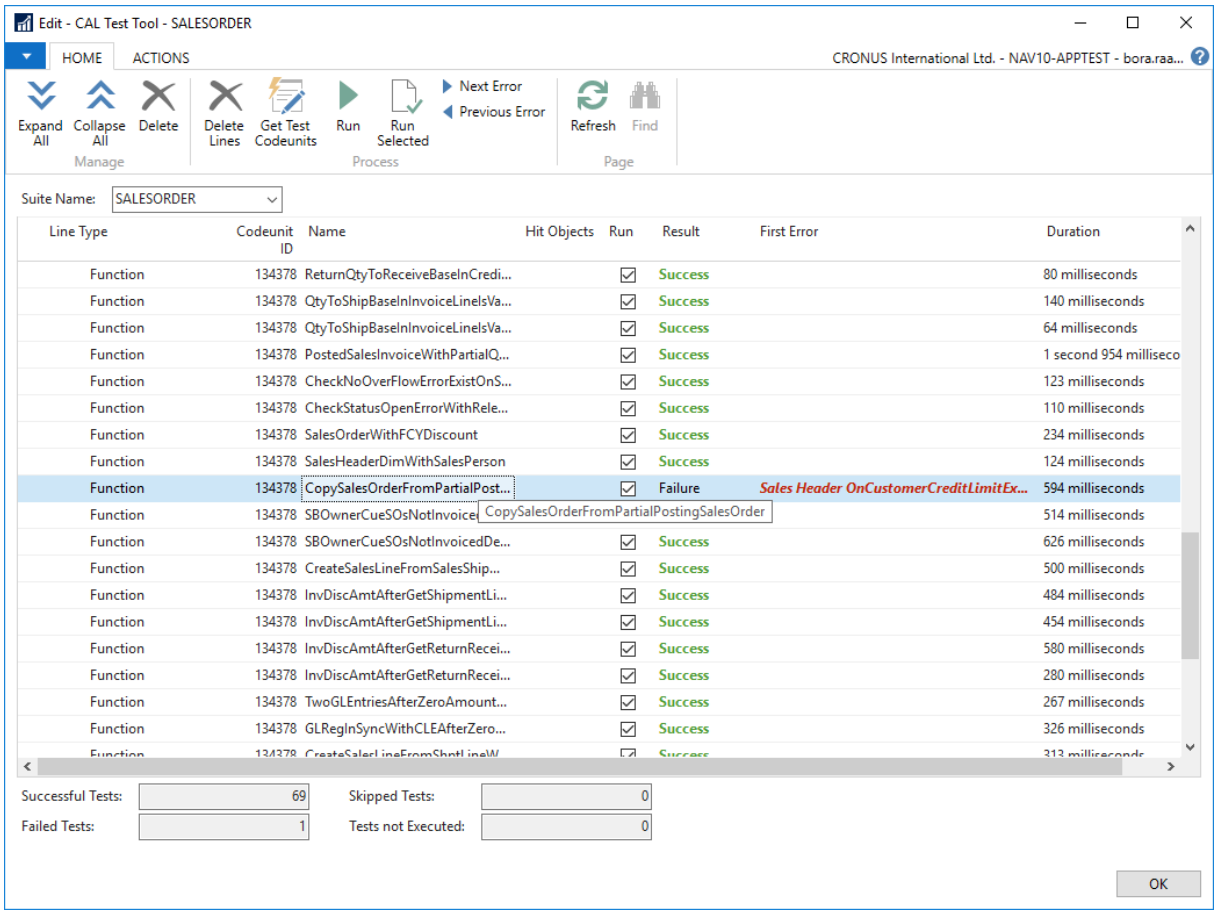


Joonis 4.10: Uue müügitellimuse sisestamisel saadav veateade: „*Sales Header OnCustomerCreditLimitExceeded test error*“

Automaattestimise testi koodiüksuse „ERM Sales Order“ käivitamisel läbitakse kõik test-funktsioonid edukalt, välja arvatud funktsioon *CopySalesOrderFromPartialPostingSalesOrder* (joonis 4.11).

Äriloogika kihis olevat krediidilimiidi ületanud kliendi viga ei tuvastata, mis on vale-positiivse tulemus. Samas tuvastatakse viga müügitellimuse kopeerimise tegevuses, mis ei ole otseselt

seotud kliendi krediidilimiidiga. Süsteemis toimus veaolukorra eskaleerumine vea tegeliku põhjusega kaudselt seotud teises äriprotsessis, mis on vale-negatiivne tulemus (joonis 4.11). Kasutajaliideses ei õnnestu vastavale kliendile müügitellimuse loomine, kuid automaattesti funktsioon *SalesOrderCreation* ei tuvasta seda tingimuslikult esinevat äriloogika viga.



Joonis 4.11: Müügitellimuse automaattestimise vale-negatiivne tulemus

4.4.4 Testi koodiüksuste täiendamine andmekomplektist tulenevalt

Vale-positiivsete ja vale-negatiivsete tulemuste vältimiseks on vajalik testtööriista täiendada. Testi koodiüksuse „ERM Sales Order“ tugifunktsioonide koodiplokis „Library - Sales“ tehtud järgmised täiendused:

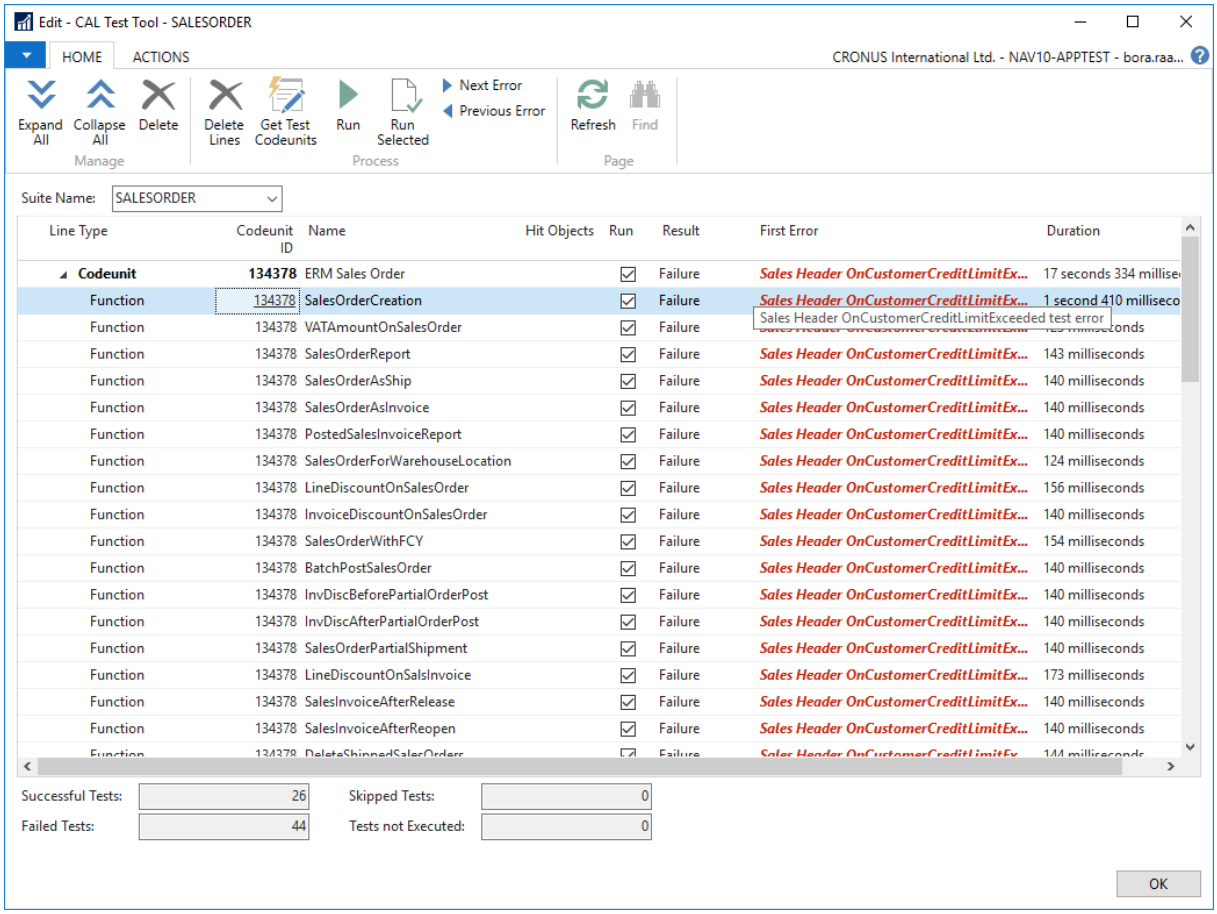
- *CreateCustomer*-funktsioonis lisatud uue kliendi loomisel krediidilimiidi määramine kliendi andmetesse:

```
//+Test
Customer.VALIDATE("Credit Limit (LCY)",1);
//-Test
```

- *CreateSalesLineWithShipmentDate*-funktsioonis lisatud krediidilimiidi kontrolli rakendamine peale müügirea loomist:

```
//+Test
SalesHeader.CheckAvailableCreditLimit;
//-Test
```

Automaattestimise testi koodiüksuse „ERM Sales Order“ käivitamisel nurjuvad enamus testfunktsioone veaga: „*Sales Header table OnModify test error*“, mis on oodatud tulemus (joonis 4.12).



Joonis 4.12: Testi koodiüksuste täiendamise järgselt müügitellimuse automaattestimise tulemus

5 Tulemuste analüüs

Juhtumiuuringu alusel saab järeldada, et automaattestimise rakendamine majandustarkvara arendusprotsessis on väga efektiivne. Seda peamiselt põhjusel, et testimine toimub regressioon-meetodil, kus eesmärk on tuvastada lisaarenduse mõjusid standardfunktsionaalsusele ja veenduda, et arenduse mõjul midagi mõnes teises funktsionaalses piirkonnas „katki“ ei läinud.

Võrreldes kogu standardfunktsionaalsuse käsitsi ületestimisega annab automaattestimise kasutamine väga olulise ajavõidu, kuna MS Dynamics NAV’i tarkvara tootja poolt on välja töötatud väga suurel hulgal tarkvara standardfunktsionaalsust katvaid testüksusi.

Tabel 5.1 kirjeldab käesoleva juhtumiuuringu näitel valitud funktsionaalses piirkonnas ühe testfunktsiooni (*SalesOrderCreation*) testimise ajakulusid, kui samad tegevused teostada käsitsi. Võrreldes automaatse ja käsitsi testimise ajakulusid näeme, et samade tegevuste teostamine automaattesti poolt võttis aega 450 ms (joonis 4.6) ja käsitsi 120 s, mis annab üle 250-kordse ajavõidu.

Tabel 5.1: Käsitsi testimise ajakulud

Automaattesti tegevus	**Käsitsi tegevus**	**Käsitsi (s)**
CreateCustomer	Uue kliendi kaardi loomine	30
CreateSalesHeader	Uue müügitellimuse dokumendi loomine	45
CreateItem	Uue kauba kaardi loomine	30
CreateSalesLine	Uue müügitellimuse rea lisamine	15
Kokku		**120**

Võttes arvesse automaattestimise rakendamisel üldiste tegevuste ajakulu ning funktsionaalse katvuse piiranguid saame järeldada, et töö eesmärgiks seatud testimise tervikprotsessis on võimalik saavutada 20% suurem efektiivsus kui rakendatakse automaattestimise vahendit.

5.1 Esilekerkinud probleemid ja nende lahendamine

Peamine MS Dynamics NAV'i automaattestimise töövahendi piirang on see, et teste ei ole võimalik käitada kasutajaliidese kihis, mistõttu on võimalikud vale-positiivsed tulemused (joonis 4.6). See piirang ei ole aga väga suureks probleemiks, kuna NAV'i arendusstandardi järgi ei realiseerita äriloogika reegleid kasutajaliidese kihi objektides. Testimisel tuleb selle piiranguga arvestada ja vajadusel teostada täiendavad testid käsitsi kasutajaliidese kihis.

Samuti on automaattestimisel lihtsalt võimalik äriloogika veaolukordade eskaleerumine teises äriprotsessis, mis kajastub vale-positiivsete ja vale-negatiivsete tulemustena (joonis 4.11). Seetõttu tuleb testitulemuste raportit tõlgendada kriitiliselt ja kogemustele tuginedes eristada valed tulemused õigetest. Samas saab kindlalt järeldada, et ka vale-negatiivse testitulemuse tegelikuks põhjuseks on süsteemi arendusviga mõnes teises äriprotsessis, mis tuleb üles leida ja parandada. Samuti on sellises olukorras mõistlik täiendada testtööriista funktsioone selliselt, et edaspidi samas olukorras vale-positiivseid ja vale-negatiivseid tulemusi vältida (joonis 4.12).

5.2 Tulevikuideed

MS Dynamics NAV'i automaattestimine on rakendatav arendusprotsessis ja töö järgmise etapina on plaanis arendada lokaliseeritud teststsenaariumite kogum, mis on kohaldatud rakendamaks ERP-tarkvara MS Dynamics NAV'i EE lokalisatsiooniga arendusprojektides.

Testtööriista lokaliseerimise tegevused on peamiselt seotud püsiandmete klassifikaatorite tähiste tõlgetega, näiteks W1-versioonis on kliendi konteeringurühma tähiseks „DOMESTIC" ja EE-versioonis vastavalt „KODUMAINE". Testtööriista teekide funktsioonid on otseselt seotud süsteemi baasandmete seadistusega, mistõttu on lokaliseeritud seadistuse korral vajalik teekide funktsioonide viimine kooskõlla süsteemi seadistusega.

Testtööriista lokaliseerimise eeltööna on vajalik prioritiseerida testfunktsioonid vastavalt äriloogikale, et lokaliseerida esmalt enim kasutatavat funktsionaalsust katvad testfunktsioonid. Teise eesmärgina on plaanis välja töötada ja juurutada arendusprotsessis vajalik metoodika ja standardid, et paralleelselt arenduste teostamisega realiseeritakse ka vastavad testfunktsioonid.

6 Kokkuvõte

Töö käigus teostatud juhtumiuuringu raames rakendati ERP-majandusinfosüsteemi MS Dynamics NAV'i standardfunktsionaalsuse arendusprotsessis tarkvara automaattestimise tööriistakomplekti. Juhtumiuuringu praktilise töö käigus realiseeriti MS Dynamics NAV'i prototüüplahendus ja uuriti automaattestimise tööriistakomplekti toimivust erinevates simuleeritud veaolukordades.

Juhtumiuuringu teostamise eesmärk sai täidetud ja töö käigus veenduti, et MS Dynamics NAV'i automaattestimise tööriistakomplekt toimib vastavalt eeldatud funktsionaalsusele. Tööriistakomplekti kasutamisel tuleb arvesse võtta, et funktsionaalse piiranguna toimib see vaid testitava rakenduse äriloogika kihis ning kasutajaliidese automaattestimine ei ole toetatud.

Töö tulemusel saab järeldada, et automaattestimise rakendamine majandustarkvara arendusprotsessis lisab oluliselt efektiivsust ja aitab tagada tarkvaralahenduse töökindlust ning arendustööde kvaliteeti. Automaattestimine ei asenda traditsioonilist käsitsi testimist, kuid arendusprotsessis standardfunktsionaalsuse regressioontestimisel on mõistlik täiendava abivahendina kasutusele võtta.

Tulevikuplaanidena on eesmärk viia MS Dynamics NAV'i automaattestimise tööriista-komplekt vastavusse kohaliku MS Dynamics NAV'i EE (Eesti) lokalisatsiooniga ning juurutada majandustarkvara konsultatsiooni ettevõtte arendusprotsessis vajalik metoodika ja standardid, et automaattestimise tööriist leiaks kasutust igapäevases tööprotsessis.

Viited

[1] Anand, S., Burke, E. K., Chen, T. Y., Clark, J., Cohen, M. B., Grieskamp, W., & ... Zhu, H. (2013). An orchestrated survey of methodologies for automated software test case generation. *The Journal Of Systems & Software*, *86*1978-2001. doi:10.1016/j.jss.2013.02.061

[2] Jarosław, B., Patryk, C., Marcin, K., Iwona, K., & Marian, J. (2016). Highly Automated Agile Testing Process: An Industrial Case Study. *E-Informatica Software Engineering Journal, Vol 10, Iss 1, Pp 69-87 (2016)*, (1), 69. doi:10.5277/e-Inf160104

[3] Garousi, V., & Pfahl, D. (2016). When to automate software testing? A decision-support approach based on process simulation. *Journal Of Software: Evolution & Process*, *28*(4), 272-285. doi:10.1002/smr.1758

[4] Geetha Devasena, M. S., Gopu, G., & Valarmathi, M. L. (2016). Automated and Optimized Software Test Suite Generation Technique for Structural Testing. *International Journal Of Software Engineering & Knowledge Engineering*, *26*(1), 1-13. doi:10.1142/S0218194016500017

[5] Sahaf Z, Garousi V, Pfahl D, Irving R, Amannejad Y. (2014). When to Automate Software Testing? Decision Support based on System Dynamics – An Industrial Case Study. In *Proceedings of the 2014 International Conference on Software and System Process* (ICSSP 2014), 149-158. doi:10.1145/2600821.2600832

[6] Catelani, M., Ciani, L., Scarano, V. L., & Bacioccola, A. (2011). Software automated testing: A solution to maximize the test plan coverage and to increase software reliability and quality in use. *Computer Standards & Interfaces*, *33*(XVI IMEKO TC4 Symposium "Exploring New Frontiers of Instrumentation and Methods for Electrical and Electronic Measurements" and XIII International Workshop on ADC Modelling and Testing), 152-158. doi:10.1016/j.csi.2010.06.006

[7] AXeptance - Module Detail Description | AXeptance. (s.a.). Kasutatud 16. mai 2017, https://axeptance.com/details-axeptance-functional-benchmark-testing

[8] AXeptance - Solution for automated functional and load testing. (s.a.). Kasutatud 4. mai 2017, https://mbs.microsoft.com/Downloads/Customer/SureStep/SureStepOnlineUser-Guide.pdf

[9] SoapUI | The Leading Open Source API Testing Tool. (s.a.). Kasutatud 16. mai 2017, https://www.soapui.org/open-source.html

[10] Compare SoapUI and SoapUI NG Pro | SoapUI NG Pro. (s.a.). Kasutatud 16. mai 2017, https://www.soapui.org/professional/soapui-ng-pro/soapui-vs-soapui-ng-pro.html

[11] SoapUI Feature Set | Open Source API Testing Tool. (s.a.). Kasutatud 16. mai 2017, https://www.soapui.org/open-source/features.html

[12] Webster, L. (2017, veebruar 27). Software Testing for Professionals. Kasutatud 16. mai 2017, https://www.visualstudio.com/vs/test-professional

[13] ComponentSource Live Help. (s.a.). Kasutatud 16. mai 2017, https://secure.livechatinc.com/licence/6365471/open_chat.cgi?groups=11&embedded=1&session_id=S1494691275.80feacfa72&server=secure.livechatinc.com#https://www.componentsource.com/product/microsoft-visual-studio-test-professional/about

[14] Automation Testing Interview Questions and Answers for QAE Profile. (2016, märts 10). Kasutatud 16. mai 2017, http://www.techbeamers.com/software-automation-testing-interview-questions

[15] Perkmann, Henri. (2015). Veebirakenduse kasutajaliidese automaatne testimine väleda arendusprotsessi kontekstis – põhimõtted ja implementatsioon. (Tartu Ülikool.; Loodus- ja tehnoloogiateaduskond.; Tehnoloogiainstituut.;). Tartu: Tartu Ülikool

[16] Unit Testing –Software Testing Fundamentals. (s.a.). Kasutatud 16. mai 2017, http://softwaretestingfundamentals.com/unit-testing

[17] Ühiktestimine ehk Unit Testing — Java materjalid dokumentatsioon. (s.a.). Kasutatud 16. mai 2017, https://ained.ttu.ee/javadoc/unit_testing.html

[18] API Testing | Vector Software. (s.a.). Kasutatud 16. mai 2017, https://www.vectorcast.com/testing-solutions/api-testing

[19] Agile & Waterfall Methodologies – A Side-By-Side Comparison | Base36. (s.a.). Kasutatud 16. mai 2017, http://www.base36.com/2012/12/agile-waterfall-methodologies-a-side-by-side-comparison

[20] Microsoft Dynamics Sure Step 2012 User Guide. (2012). Kasutatud 4. mai 2017, https://mbs.microsoft.com/Downloads/Customer/SureStep/SureStepOnlineUserGuide.pdf

[21] Microsoft Dynamics NAV 2016. (s.a.). Kasutatud 4. mai 2017, https://msdn.microsoft.com/library/hh173988(v=nav.90).aspx

[22] The history of Dynamics NAV / Navision - NAV Wikipedia - Dynamics NAV Users. (s.a.). Kasutatud 4. mai 2017, https://dynamicsuser.net/nav/w/navdev/6/the-history-of-dynamics-nav-navision

[23] "The Navision Model" was born when IBM said no thanks | Hans Peter Bech. (s.a.). Kasutatud 4. mai 2017, http://www.hpbech.com/the-navision-model-was-born-when-ibm-said-no-thanks

[24] Microsoft Dynamics NAV – sisult ja tehnoloogiliselt kaasaegne ning kiiresti rakendatav majandustarkvara! - Raamatupidaja. (2016, märts 14). Kasutatud 25. aprill 2017, http://www.raamatupidaja.ee/uudised/2016/03/14/microsoft-dynamics-nav--sisult-ja-tehnoloogiliselt-kaasaegne-ning-kiiresti-rakendatav-majandustarkvara

[25] Dynamics NAV 2015 Eesti kohalik funktsionaalsus. (s.a.). Kasutatud 25. aprill 2017, http://www.itera.ee/lahendused/dynamics-nav-2015-eesti-kohalik-funktsionaalsus

[26] Dynamics NAV Architecture - NAV Wikipedia - Dynamics NAV Users. (s.a.). Kasutatud 4. mai 2017, https://dynamicsuser.net/nav/w/navdev/95/dynamics-nav-architecture

[27] Mis on 3-kihiline majandustarkvara arhitektuur ja miks on seda vaja? (2015, jaanuar 10). Kasutatud 4. mai 2017, http://www.itera.ee/2015/01/mis-3-kihiline-majandustarkvara-arhitektuur-ja-miks-seda-vaja

[28] Product and Architecture Overview. (s.a.). Kasutatud 6. mai 2017, https://msdn.microsoft.com/en-us/library/dd354965(v=nav.90).aspx

[29] Testing the Application. (s.a.). Kasutatud 4. mai 2017, https://msdn.microsoft.com/en-us/library/ee414224(v=nav.90).aspx

[30] Application Test Toolset for Microsoft Dynamics NAV 2009 SP1 - Microsoft Dynamics PartnerSource. (s.a.). Kasutatud 4. mai 2017, https://mbs.microsoft.com:443/partnersource/northamerica/deployment/data-sheets/MDNAVApplicationTest

[31] Vjeko. (2015, oktoober 8). What's New in NAV 2016: Application Test Toolkit. Kasutatud 4. mai 2017, http://vjeko.com/whats-new-in-nav-2016-application-test-toolkit

[32] waldo. (2013, märts 7). NAV 2013: Implementing the Application Test Toolset. Kasutatud 4. mai 2017, http://www.waldo.be/2013/03/07/nav-2013-implementing-the-application-test-toolset

[33] Andrei Panko. (2015, oktoober). What is the CAL Test Tool in Microsoft Dynamics NAV 2016. Kasutatud 4. mai 2017,

https://mbspartner.microsoft.com/secure/whatsnew/NAV2016/QuickToImplement/HOW_TO_use_CAL_Test_Tool_in_Microsoft_Dynamics_NAV_2016.docx

[34] Application Test Automation. (s.a.). Kasutatud 4. mai 2017, https://msdn.microsoft.com/en-us/library/dn951441(v=nav.90).aspx

Lihtlitsents lõputöö reprodutseerimiseks ja lõputöö üldsusele kättesaadavaks tegemiseks

Mina, Rauno Tamm

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

“ Microsoft Dynamics NAV tarkvara automaattestimise juhtumiuuring ettevõttes Columbus Eesti AS”

mille juhendajad on Rivo Lemmik ja Heiki Kasemägi

(a) reprodutseerimiseks säilitamise ja üldsusele kättesaadavaks tegemise eesmärgil, sealhulgas digitaalarhiivi DSpace’is lisamise eesmärgil kuni autoriõiguse kehtivuse tähtaja lõppemiseni;

(b) üldsusele kättesaadavaks tegemiseks Tartu Ülikooli veebikeskkonna kaudu, sealhulgas digitaalarhiivi DSpace’i kaudu kuni autoriõiguse kehtivuse tähtaja lõppemiseni.

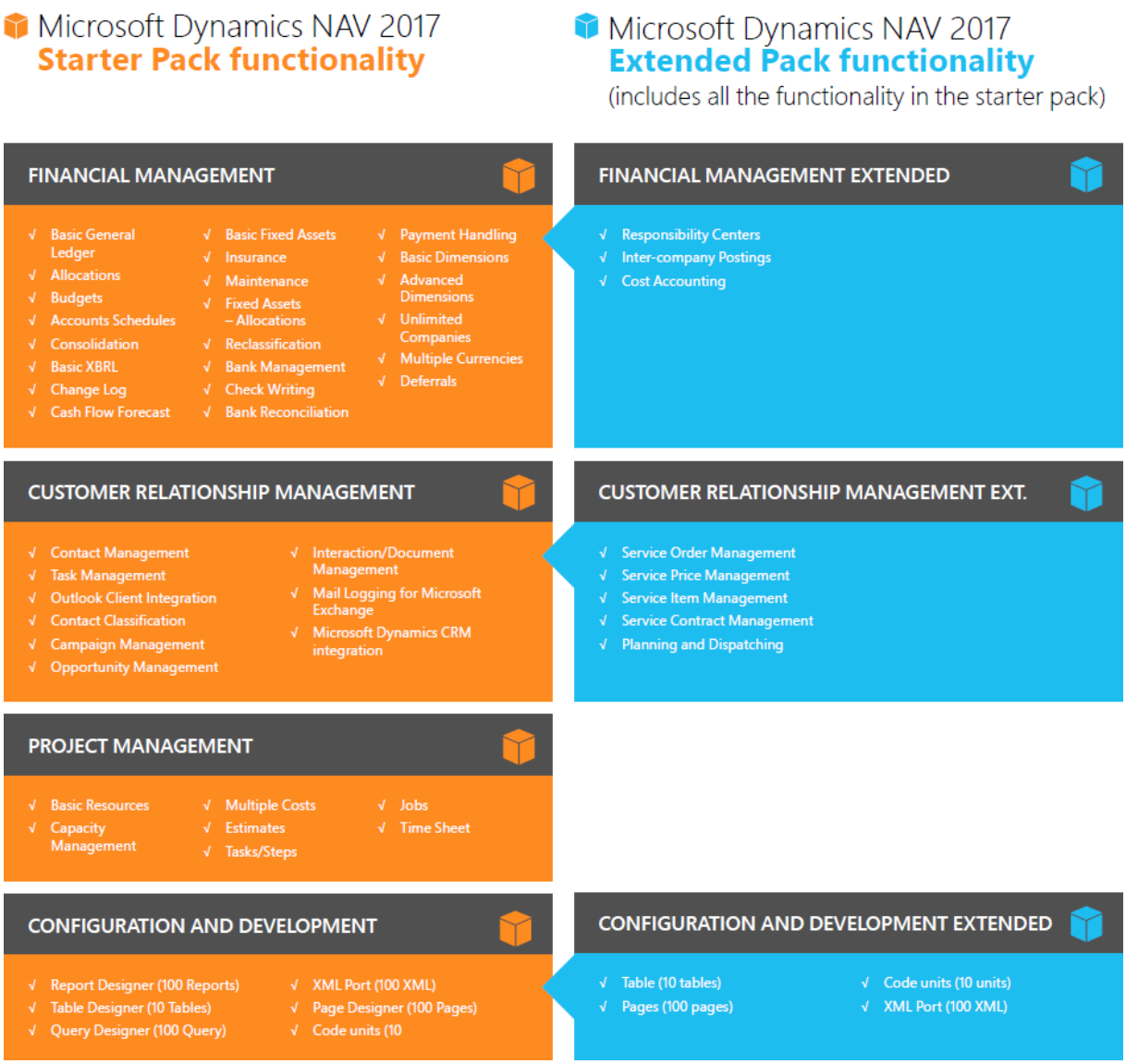
2. olen teadlik, et punktis 1 nimetatud õigused jäävad alles ka autorile;

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

Tartus, **17.05.2017**

Lisad

Lisa 1. Microsoft Dynamics NAV 2017 funktsionaalsus



Microsoft Dynamics NAV 2017
Starter Pack functionality

SUPPLY CHAIN MANAGEMENT

- √ Basic Receivables
- √ Sales Invoicing
- √ Sales Order Management
- √ Sales Invoice Discounts
- √ Alternative Ship-To Addresses
- √ Shipping Agents
- √ Sales Return Order Management
- √ Sales Line Discounting
- √ Sales Line Pricing
- √ Sales Tax
- √ Basic Payables
- √ Purchase Invoicing
- √ Purchase Order Management
- √ Purchase Invoice Discounts
- √ Requisition Management
- √ Alternative Order Addresses
- √ Purchase Return Order Management
- √ Purchase Line Discounting
- √ Purchase Line Pricing
- √ Drop Shipments
- √ Salespeople/ Purchasers
- √ Basic Inventory
- √ Multiple Locations
- √ Stock-Keeping Units
- √ Alternative Vendors
- √ Assembly
- Management
- √ Location Transfers
- √ Item Substitutions
- √ Item Cross References
- √ Nonstock Items
- √ Item Tracking
- √ Item Charges
- √ Bin
- √ Pick
- √ Analysis Reports
- √ Item Budgets
- √ Workflows
- √ Document Management, Document Capture and OCR
- √ E-services

HUMAN RESOURCE MANAGEMENT

- √ Basic Human Resources

SPROG

- √ Multiple Document Languages
- √ Multiple Languages

OTHER

- √ Subsidiary (Each)
- √ Intrastate
- √ Extended Text
- √ Job Queue
- √ Reason Codes
- √ Dynamics NAV Server

Microsoft Dynamics NAV 2017
Extended Pack functionality
(includes all the functionality in the starter pack)

SUPPLY CHAIN MANAGEMENT EXT

- √ Order Promising
- √ Calendars
- √ Campaign Pricing
- √ Cycle Counting
- √ Put Away
- √ Warehouse Receipt
- √ Warehouse Shipment
- √ Standard Cost Worksheet
- √ Warehouse Management Systems
- √ Internal Picks and Put Aways
- √ Automated Data Capture System
- √ Bin Setup

MANUFACTURING SOLUTIONS

- √ Production Orders
- √ Production Bill of Materials
- √ Version Management
- √ Agile Manufacturing
- √ Basic Supply Planning
- √ Demand Forecasting
- √ Basic Capacity Planning
- √ Machine Centers
- √ Finite Loading