LATVIJAS UNIVERSITĀTE
DATORIKAS FAKULTĀTE

INTEGRĀCIJAS TESTĒŠANAS EFEKTIVITĀTES UZLABOŠANAS VADLĪNIJU IZSTRĀDE

MAĢISTRA DARBS

Autors: **Agnese Semjonova**
Studenta apliecības Nr.: ab11241
Darba vadītājs: profesors Jānis Bičevskis

RĪGA 2017

ANOTĀCIJA, ATSLĒGVĀRDI

Maģistra darba ietvaros tiek izstrādātas vadlīnijas, kas jāņem vērā, izvēloties integrācijas testēšanas pieeju, lai integrācijas testēšanas posmā tiktu tērēti pēc iespējas mazāk resursu, bet tajā pašā laikā tiktu gūts maksimālais iespējamais guvums – nodrošināts kvalitatīvs gala produkts. Autore piedāvā integrācijas testēšanas metodi balstīt uz lietotāju stāstiem. Lai pārbaudītu vai balstīšanās uz lietotāju stāstiem sniedz kādu labumu, maģistra darba ietvaros tiek veikts pētījums, kura ietvaros tiek izstrādāts lietotāju scenāriju modelis (kas izriet no lietotāju stāstiem) konkrētam risinājumam, kas sastāv no vairākām dažādām sistēmām. Pēc tam tiek veikta analīze, balstoties uz vēsturiskiem datiem (atklātajām kļūdām), lai secinātu, kā mainītos rezultāti, ja tiktu izmantots lietotāju scenāriju modelis.

Pētījuma gala secinājums – lietotāju scenāriju modelis, konkrētā risinājuma ietvaros, diezgan būtiski uzlabotu gala piegādes rezultātu – 36% no visām neatklātajām kļūdām piecu piegāžu laikā tiktu atklātas, un divas no visām atklātajām kļūdām būtu iespējams atklāt vēl pirms koda piegādes. Tāpat, rezultāti atspoguļo, ka šīs kļūdas ļoti būtiski iespaido patērētāju, jo lietotāju scenāriju modelī tiek aplūkotas svarīgākās sistēmas darbības no lietotāja skatupunkta.

Atslēgvārdi: Integrācijas testēšana; testēšanas efektivitātes uzlabošana; lietotāju scenāriji; testēšanas tehnikas izvēle.

ANNOTATION, KEYWORDS

The aim of the master's thesis “Development of Guidelines for Integration Testing Efficiency Improvement” is to establish the guidelines – that should be taken into account when choosing integration testing approach – in order to spend as less as possible resources on integration testing phase, but at the same time would achieve maximum possible benefit – qualitative end product. The author offers to base integration testing on user stories. To check whether basing on user stories would provide some benefit, research is done as the part of Master's thesis. There is developed user scenario model for specific solution and analysis done based on historical data (errors detected) in order to conclude how would change situation if integration testing would be based on user stories.

After analysis the author concludes that user scenario model for particular solution would quite substantially improve the final delivery results - 36% of all undiscovered errors (occurred during the five latest releases) would be detected and two of the detected errors would be possible to detect even before software code is delivered to testers. Also, the results shows that the errors, that were not detected, very negative impacts consumer, because user scenario model deals with the most important system operations.

Keywords: Integration testing; improving the effectiveness of the test; user stories; choice of testing approach.

AUTOREFERĀTS

Maģistra darba ietvaros tiek izpētīta literatūra par integrācijas testēšanas pieejām, posmiem un procesiem. Balstoties uz izpētīto literatūru, darba otrajā nodaļā tiek izstrādātas vadlīnijas, pie kurām būtu jāpieturas, lai uzlabotu integrācijas testēšanas efektivitāti. Autore piedāvā testēšanas pieeju balstīt uz lietotāju stāstiem, un, lai pārbaudītu vai šādas pieejas izmantošana uzlabotu testēšanas efektivitāti, autore veic pētījumu, izmantojot konkrētu komplicētu risinājumu.

Autore izstrādā lietotāju scenāriju modeli – aktivitāšu diagrammas, kurās iekļauti lietotāju stāsti un no tiem izrietošie galvenie lietotāju scenāriji. Pēc modeļa izveides tiek veikta retrospektīva analīze, balstoties uz iepriekšējo piegāžu datiem – vēsturiski atklātajām kļūdām, un veikti secinājumi.

Rezultāti atklāj, ka lietotāju scenāriju modeļa izmantošana uzlabo integrāciju testēšanas kvalitāti darbā pētītajam risinājumam. Ievērojamu daudzumu kļūdu būtu iespējams atklāt, ja tiktu izmantots lietotāju scenāriju modelis kā arī tas palīdzētu ne tikai testēšanas posmā, bet arī izstrādes posmā. Piedāvātās metodes izmantošana veiksmīgi ieviesta konkrētā risinājuma integrāciju testēšanā un arī turpmāk to plānots izmantot.

Autore uzskata, ka šādas pieejas izmantošana varētu uzlabot integrāciju testēšanu arī citu risinājumu ietvaros, pirms tam izpētot lietotāju stāstus un ņemot vērā risinājumā iesaistīto sistēmu un integrāciju skaitu.

Autore uzskata, ka darbā piedāvātā metode, kā arī analīzes gaita ir detalizēti aprakstīta. Darbā tiek izmantoti dažādi literatūras avoti – gan grāmatas, gan zinātniski raksti, gan dažādi interneta avoti.

SATURS

Apzīmējumu saraksts 6
Ievads 7
1. Integrācijas testēšana 9
1.1 Integrācijas testēšanas plāns 12
1.2 Integrācijas testa specifikācija 13
1.3 Integrācijas testēšanas metodes un pieejas 14
1.3.1 Top-down integrācijas testēšanas tehnika 15
1.3.2 Bottom-up integrācijas testēšanas tehnika 15
1.3.3 Ad-Hoc integrācijas testēšanas tehnika 16
1.3.4 Backbone integrācijas testēšanas tehnika 16
1.3.5 Big Bang integrācijas testēšanas tehnika 16
1.3.6 Citi veidi 17
1.4 Integrācijas testēšanas organizācija 18
2. Vadlīniju izstrāde efektīvai integrācijas testēšanai 19
2.1 Kvalitatīva integrācijas testēšanas plāna izstrāde 19
2.2 Laika izmantošana, kamēr komponentes nav piegādātas 19
2.3 Testa scenāriju izstrāde 19
2.4 Uz lietotāju scenārijiem balstīts testēšanas modelis 20
3. Lietotāju scenāriju modeļa izstrāde un integrācijas testēšana atbilstoši vadlīnijām 22
3.1 Lietotāju scenāriju modeļa izstrāde 23
3.2 Lietotāju scenāriju modeļa izveides pamatojums 38
3.3 Analīze, balstoties uz vēsturiskiem datiem konkrētas sistēmas ietvaros 38
3.3.1 Kļūdu analīzes piemēri 39
3.3.2 Kvantitatīvie dati 42
3.3.3 Optimizēti regresa testa scenāriji 49
Rezultāti 51
Secinājumi 52
Izmantotā literatūra un avoti 53
Pielikumi 54

APZĪMĒJUMU SARAKSTS

Apzīmējums	Skaidrojums
IS	Informācijas sistēma
LV	Latvija
CRM	Klientu pārvaldības sistēma
PVS	Pasūtījumu veidošanas sistēma
NS	Norēķinu sistēma
TLK	Tīmekļa lapa klientiem
PAS	Pasūtījuma apstrādes sistēma
SmsS	Īsziņu sūtīšanas sistēma

IEVADS

Izstrādājot informācijas sistēmu, ir ļoti svarīgi nodrošināt sistēmas kvalitāti. Pat nelielas sistēmas kļūdas var izraisīt lielus zaudējumus uzņēmumam, kurš izmanto / izmantos sistēmu, tādēļ, testēšana ir ļoti svarīgs posms sistēmas izstrādē. Lai atklātu problēmas pēc iespējas ātrāk, testēšana tiek veikta dažādās fāzēs un var tikt iedalīta vairākās daļās – vienības jeb moduļu testēšana, integrācijas testēšana, sistēmtestēšana un akcepttestēšana. Integrācijas testēšana tiek veikta, lai noskaidrotu, ka savstarpēji moduļi sadarbojas bez kļūdām un izmaiņas vienā no moduļiem nav izraisījušas kļūdas citos moduļos. Darbā sīkāk tiks pētīta tieši integrācijas testēšana, jo bieži vien šim testēšanas posmam tiek tērēta salīdzinoši maza uzmanība (jo visas komponentes atsevišķi jau ir pārbaudītas), vai tieši pretēji – ļoti lieli resursi ieguldīti integrācijas testēšanā, atkārtojot sistēmtestā (atsevišķas komponentes testēšanā) veiktās pārbaudes.

Darba mērķis ir izstrādāt vadlīnijas, kas palīdzētu uzlabot integrācijas testēšanas efektivitāti. Autore piedāvā integrācijas testēšanas metodi balstīt uz lietotāju stāstiem.

Darba pirmajā daļā ir izpētīta literatūra par integrācijas testēšanas pieejām, posmiem un procesiem. Nodaļā tiek aplūkota integrācijas testēšanas plāna nozīme, integrācijas testa specifikācijas veiksmīga izveide, aplūkotas dažādas testēšanas metodes, to plusi un mīnusi, kā arī vairākas nianses integrācijas testēšanas organizācijai.

Darba otrajā daļā tiek izstrādātas vadlīnijas. Vadlīniju izstrāde tiek balstīta uz pirmajā nodaļā izpētīto literatūru kā arī izvirzīti autores ieteikumi. Vadlīnijās aprakstīti ieteikumi, kas jāņem vērā izstrādājot integrācijas testēšanas plānu, ieteikumi, kā efektīvi izmantot laiku gatavojot testa scenārijus un izpildot testus, kā arī kā veiksmīgi izstrādāt testa scenārijus un aprakstīta autores piedāvātā ideja par integrācijas testēšanas balstīšanu uz lietotāju stāstiem.

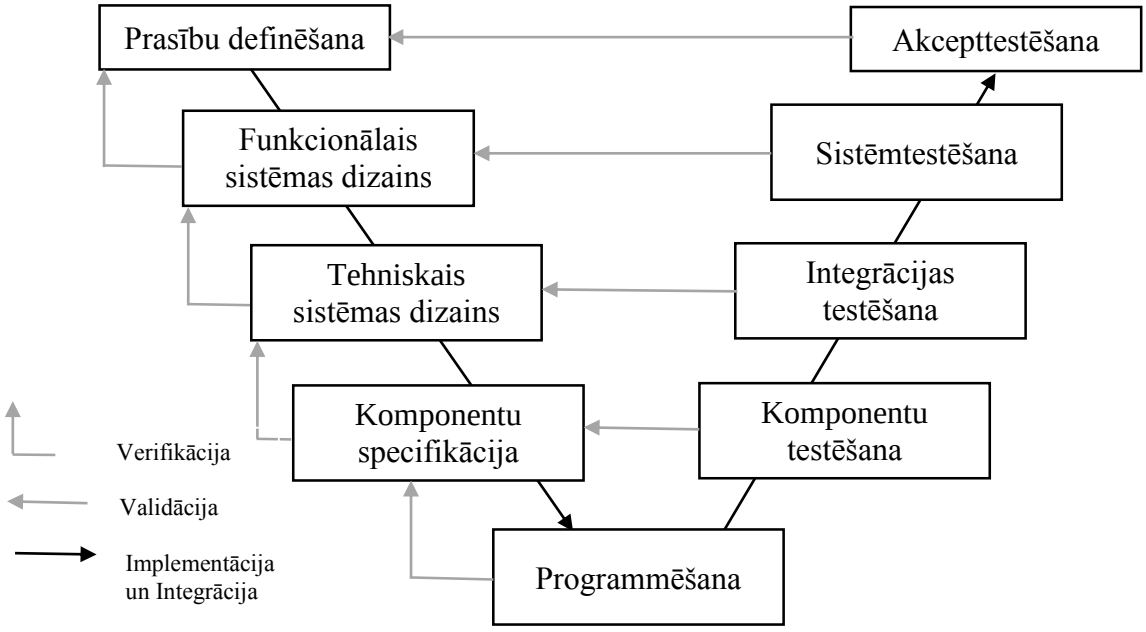
Darba galvenā ir trešā nodaļa, kurā tiek izklāstīts lietotāju scenāriju modeļa izveides pamatojums, izstrādāts lietotāju scenāriju modelis – vairākas aktivitāšu diagrammas ar paskaidrojumu, kuri scenāriji tiek iekļauti modelī, cik testa scenāriji izriet no konkrētās aktivitāšu diagrammas utml. Tālāk darbā tiek apkopotas iepriekšējo piegāžu kļūdas un veikta retrospektīva analīze, kuras mērķis ir noteikt, cik neatklātās kļūdas būtu iespējams atklāt, ja tiktu izmantots lietotāju scenāriju modelis, un cik kļūdas būtu iespējams atklāt ātrāk – vēl pirms programmatūras koda piegādes. Nodaļā

attēloti kļūdu analīzes piemēri, atspoguļoti kvantitatīvie dati un veikti secinājumi, kādus vēl ieguvumus sniegtu lietotāju scenāriju modeļa izstrāde un izmantošana.

Darba rezultāti atspoguļo, ka lietotāju scenāriju modelis, konkrētā risinājuma ietvaros, diezgan būtiski uzlabotu gala piegādes rezultātu – 36% no visām neatklātajām kļūdām piecu piegāžu laikā tiktu atklātas, un divas no visām atklātajām kļūdām būtu iespējams atklāt vēl pirms koda piegādes. Autore secina, ka lietotāju scenāriju modeli būtu ieteicams izmantot arī sistēmas arhitektiem un analītiķiem. Tiek secināts, ka šāda modeļa izmantošana palīdzētu ne tikai testēšanas posmā, bet arī sistēmas izstrādē. Tāpat, lietotāju scenāriju modeli būtu iespējams izmantot optimizētiem regresa testa scenārijiem.

1. INTEGRĀCIJAS TESTĒŠANA

Integrācijas testēšana ir testēšanas fāze, kuras laikā programmatūras un/vai aparatūras sastāvdaļas tiek kombinētas un testētas, lai pārliecinātos par to savstarpējo sadarbību atbilstoši izvirzītajām prasībām. [Webster's Dictionary, 2017]



1.1. att. "**V**" **modelis**

Kā minēts A. Spillner, T. Linz un H. Schaefer grāmatā "Software Testing Foundations", "V" modelis attēlo izstrādes procesu, un savstarpējās saistības starp aktivitātēm. Kā redzams, integrācijas testēšana seko pēc komponentu testēšana un ir saistīta ar tehnisko sistēmas dizainu (validācija). "V" modeļa kreisais zars attēlo izstrādes procesus, savukārt labais zars attēlo testēšanas procesu. [Spillner et. Al., 2014]

Nedaudz sīkāk par "V" modeļa kreiso zaru. Sistēmas izstrādes process sākas ar prasību definēšanu – tiek ievāktas prasības no klienta (gala patērētāja), specificētas un apstiprinātas. Pēc tam izstrādāts sistēmas funkcionālais dizains – prasības tiek pārvērstas funkcijās. Tālāk, tiek izstrādāts tehniskais sistēmas dizains – tiek definēti interfeisi un sistēmas arhitektūra. Katra apakšsistēma (pēc arhitektūras) var tikt izstrādāta atsevišķi. Pēc tam seko komponentu specifikācija – katrai apakšsistēmai (jeb komponentei) tiek definēta specifikācija par tās pienākumiem, struktūru, interfeisiem. Un kā beidzamais solis kreisajā zarā ir programmēšana – katra atsevišķā komponente (modulis, klase) tiek implementēta programmēšanas valodā. [Spillner et. Al., 2014]

“V“ modeļa kreisais zars attēlo testēšanas procesu, kas sākas uzreiz pēc programmēšanas soļa. Vispirms tiek veikta komponentu testēšana, kuras laikā tiek pārbaudīts vai katra komponente tikusi izstrādāta atbilstoši specifikācijai (attēlā redzama saite uz “komponentu specifikāciju“). Šajā posmā komponentes (moduļi, vienības, funkcijas vai klases) tiek testētas individuāli, izolēti no citām. Izolētība ir svarīga, lai izvairītos no kļūdām, kas radušās citu komponenšu dēļ. Ja tiek atklāta kāda kļūda, tad ir viennozīmīgi skaidrs, ka šī kļūda ir konkrētās komponentes ietvaros. Tā kā komponenšu testēšana ir zemākajā testu līmenī, uzreiz pēc programmēšanas, testēšana notiek ciešā saiknē ar izstrādātājiem. Parasti komponentu testēšanas laikā tiek atklāti defekti un kļūdas, kas saistītas ar nepareizām kalkulācijām, kļūdainiem vai trūkstošiem programmu ceļiem. Jāmin, ka komponenšu testēšanā tiek pārbaudīti arī tā sauktie negatīvie scenāriji, piemēram, kā sistēma reaģēs, ja tiks izmantots nepareizs datu tips, piemēram “char“ vietā tiks izmantots “int“. Tāpat, komponenšu testēšanas laikā tiek pārbaudīta efektivitāte (cik efektīvi tiek izmantoti datora resursi) un uzturamība (no angļu val. “maintainability“), kur tiek pārbaudītas īpašības, cik viegli vai sarežģīti ir veikts izmaiņas programmā vai turpināt uzturēt to. [Spillner et. Al., 2014]

Pēc komponentu testēšanas tiek veikta integrācijas testēšana, kas tālāk darbā tiks aplūkota detalizētāk.

Pēc tam, kad integrācijas testēšana ir veiksmīgi pabeigta, tiek veikts sistēmtests – vai visa sistēma atbilst izvirzītajām prasībām. Kādēļ vēl pēc integrācijas testēšanas nepieciešams veikt sistēmtestu? Galvenais iemesls ir saistīts ar to, ka zemākos testa līmeņos testēšana tiek veikta pret tehnisko specifikāciju, t.i. no tehniskā skatupunkta, nevis no klienta / lietotāja skatupunkta. Sistēmtestā testētāji pārbauda, vai visas prasības ir ievērotas pilnībā un korekti. Pēc integrācijas testēšanas, sistēma ir pilnībā sakomplektēta jeb samontēta, un sistēmtests sistēmu pārbauda kā veselu kopumu. Pēc iespējas, tiek izmantota vide, kas ir līdzīgāka produkcijas videi. Šī testa laikā netiek izmantoti draiveri vai aizbāžņi (no angļu valodas “stubs”), un uz testa platformas tiek uzinstalēti aparatūras un programmatūras produkti, kas tiks izmantoti vēlāk. Bieži vien, lai taupītu laiku un izmaksas, netiek izmantota testa vide, bet gan klienta darbojošā vide. Galvenais iemesls, kādēļ nevajadzētu pieļaut šo kļūdu ir tāds, ka sistēmtestā tiks atklātas kļūdas un tas var atstāt ietekmi uz klienta vidi. Tāpat, testētājiem uz klienta vides būs ierobežotas piekļuves konfigurācijas iestatījumiem.

Noslēgumā, pēc sistēmtesta tiek veikta akcepttestēšana – līdzīgi kā sistēmtestā, tiek pārbaudīts vai sistēma atbilst izvirzītajām prasībām, bet atšķirībā no sistēmtesta

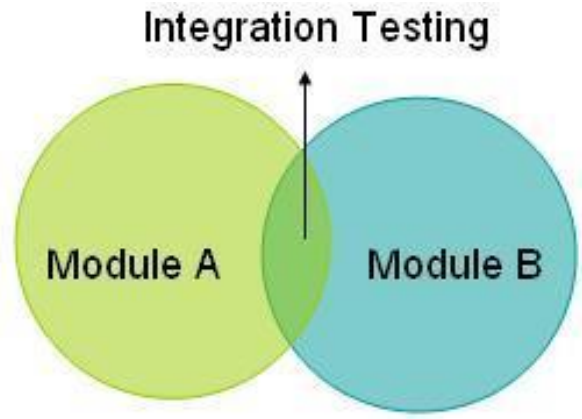
akcepttestēšanā tiek iesaistīts klients un tiek pārbaudīts vai no viņa skatupunkta izstrādātā sistēma atbilst prasībām. Tipiskākās formas, kādās tiek veikta akcepttestēšana:

1. Testēšana, lai noteiktu vai līgums ir izpildīts;
2. Lietotāju akcepttestēšana;
3. Operacionālā (akcept) testēšana
4. Alfa un Beta testēšana [Spillner et. Al., 2014]

Tālāk darbā tiks sīkāk aplūkota integrācijas testēšana, kas sekoja uzreiz pēc komponenšu testēšanas un tika izpildīta pirms sistēmtesta.

Pēc divu dažādu komponentu savienošanas, iesaka veikt integrācijas testēšanu. Viens no iemesliem, ko min Andreas Spillner grāmatā "Software Testing Foundations" [Spillner et. Al., 2014] ir tāds, ka realitātē sistēmas reti tiek izstrādātas no nulles. Biežāk tiek mainīta kādas eksistējošas sistēmas funkcionalitāte, paplašināta sistēma un/vai savienota ar citām sistēmām. Līdz ar to daudzas sistēmas komponentes ir komerciālas – nopirkts jau gatavs produkts, kurš pieejams tirgū. Šajos gadījumos eksistējošā jeb pamata funkcionalitāte reti tiek testēta, savukārt ar integrācijas testēšanu ir iespējams veikt testus, kas pārklāj arī sistēmas standarta funkcionalitāti.

Kā redzams attēlā 1.1, kad Modulis A tiek apvienots ar Moduli B, jāveic integrācijas testēšanu tām daļām, kuras ir kopīgas vai ietekmē viena otru. Šeit jāpiemin, ka integrāciju testēšanu būtu jāveic ja arī tajā gadījumā, ja kādā no moduļiem bijušas izmaiņas, kuras varētu ietekmēt otru moduli jeb sistēmu.



1.2. att. **Integrācijas testēšana - Moduļi**

Integrācijas testēšana parasti tiek veikta pēc vienību testēšanas.

Integrācijas testēšanas mērķis ir pārbaudīt funkcionalitāti, veiktspēju atbilstoši sistēmas prasībām, kas var rasties dažādu komponenšu savienošanas gadījumā.

Jāpiemin, ka reizēm problēmas var rasties jau mēģinot savienot vairākas komponentes – savienošana neizdodas, jo, piemēram, sistēmās tiek izmantoti dažādi formāti, trūkst kāda faila vai izstrādātāji nav korekti ņēmuši vērā arhitektūras plānojumu.

Protams, grūtāk ir konstatēt tās kļūdas, kuras iespējams atklāt tikai izpildot dažādus scenārijus, kas pārklāj integrēto sistēmas daļu. Šīs parasti ir kļūdas, kas saistītas ar datu apmaiņu un komunikāciju starp sistēmas daļām. Kā viena no kļūdas iespējām ir tāda, ka viena no komponentēm pārraida sintaktiski kļūdainus datus. Kā rezultātā, sistēmas komponente, kas datus saņem, reaģē ar kļūdas paziņojumu, pārtrauc darboties vai kā citādi reaģē uz to, ka nespēj apstrādāt ienākošos datus (piemēram, nekorekti formāti, protokola kļūdas). Šāda veida kļūdas nav iespējams atklāt ar sistēmas jeb atsevišķas komponentes testēšanu.

Amerikāņu zinātnieks R. Pressmans atzīmē, ka integrācijas testēšanu var iedalīt divos veidos – inkrementālā un neinkrementālā. Gadījumā, kad tiek izmantota inkrementālā metode, sistēmas komponentes tiek savienotas pakāpeniski, un šajā gadījumā testēšana notiek pēc katras komponentes pievienošanas. Savukārt neinkrementālās integrācijas testēšanas gadījumā, visas komponentes tiek savienotas un tikai tad tiek testēta sistēma. [Pressman, 2000]

1.1 Integrācijas testēšanas plāns

Viens no kritērijiem, kas ir būtisks efektīvai integrācijas testēšanai, ir optimāla integrācijas testēšanas plāna izstrāde. Bez šī plāna ir ļoti grūti kvalitatīvi organizēt testēšanas darbu.

Integrācijas testēšanas plāns tiek noteikts IS izstrādes sākuma jeb projektēšanas posmā - un nosaka stratēģiju, kāda tiks veikta integrācijas testēšana. (Testēšanas stratēģijas izvēle sīkāk tiks aplūkota 1.3 nodaļā). Integrācijas testēšanas plānā ir svarīgi ņemt vērā sistēmas struktūru, sistēmas moduļus, kā tie savā starpā tiek kombinēti, kā ietekmē viens otru. [Ould, Unwin, 1988]

Integrācijas testēšanas plāna mērķis ir definēt stratēģiju, kādā tiks veikta integrācijas testēšana – noteikt aparatūru, rīkus, procedūras, kādas tiks izmantotas testēšanas gaitā. Integrācijas testēšanas plānā būtu svarīgi definēt gala produktu integrācijas testēšanai, t.i. testējamo sistēmu vai testējamās apakšsistēmas, identificēt testa specifikāciju, atzīmēt katra integrācijas testa mērķi un secību, kādā būtu jāveic

testi. Plānā var iekļaut arī integrācijas testa specifikāciju aprakstus – katram testam atsauces uz sistēmas prasību dokumentiem, konfigurācijas piezīmes, kas aktuālas konkrētam testam un rīku un programmatūru sarakstu, kas tiks izmantoti testos.

Integrācijas testēšanas plānā ir jābūt definētam integrācijas testēšanas apjomam. Parasti tas tiek definēts līdz sistēmas projektēšanas posma beigām. Integrācijas testēšanas apjoms ir atkarīgs no vairākiem faktoriem:

- Sistēmas sarežģītība
- Aparatūras konfigurācija, kurā sistēma tiks palaista
- Moduļu testēšanas filozofija
- Rīku pieejamība

Ar integrācijas testēšanas plānu būtu svarīgi iepazīties:

- Vadībai, lai secinātu, kāds būs testēšanas grafiks, plāns
- Dizaina jeb projektēšanas komandai (arhitektiem, analītiķiem), lai noskaidrotu vai testa plāna stratēģija ir atbilstoša sistēmas dizaina prasībām
- Programmēšanas komandai, lai iepazītos ar plānoto integrācijas testēšanas apjomu
- Integrācijas komandai, lai iepazītos ar plānoto darbu sarakstu, apjomu un stratēģiju
- Akceptēšanas un kvalitātes komandai, lai pārbaudītu, vai testēšanas plāns atbilst izvirzītajām sistēmas un kvalitātes prasībām.

Integrācijas testēšanas plānam ir jābūt:

- Pilnīgam – jāpārklāj visas prasības un visus integrējamos moduļus jeb apakšsistēmas
- Īstenojamam - plānam jābūt izstrādātam atbilstoši pieejamajiem resursiem
- Konsekventam [Ould, Unwin, 1988]

1.2 Integrācijas testa specifikācija

Lai integrācijas testēšanu varētu saukt par efektīvu, testēšanas procesam ir jāgatavojas jau sākumā. Ir būtiski izstrādāt integrācijas testa specifikāciju (vai vismaz uzmetumu), kas vēlāk kalpos kā vadlīnijas veicot testa scenāriju izpildi.

Integrācijas testa specifikācija tiek veidota katram integrācijas testam. Tiek izstrādāti testa scenāriji, kur soli pa solim tiek definētas darbības un sagaidāmie rezultāti. Testa izpildes rezultātos tiek atspoguļoti reālie rezultāti un salīdzināti ar

sagaidāmajiem. No šiem rezultātiem ir iespējams secināt, vai tests ir bijis veiksmīgs, vai atklātas kādas kļūdas. [Ould, Unwin, 1988]

Bez testa specifikācijas, testa rezultātu analīze var prasīt daudz vairāk laika – esošie rezultāti jāsalīdzina ar prasību specifikāciju, analīzes un arhitektūras dokumentiem.

1.3 Integrācijas testēšanas metodes un pieejas

Kādā kārtībā būtu jāveic integrēto sistēmas komponenšu testēšana, lai sasniegtu visaugstāko efektivitāti – zemākas izmaksas un augstāks ieguvums? Viens no vissvarīgākajiem kritērijiem ir noteikt optimālu integrācijas stratēģiju projektam, ko parasti veic testa vadītāji.

Praksē problēmas rada tas, ka komponentes tiek izstrādātas dažādos laikos. Tās var būt dažas nedēļas, vai pat mēneši. Ne projekta vadītājs, ne testa vadītājs nespēj ietekmēt to, ka šajā laikā testētāju darbā ir dīkstāve – netiek aktīvi veikta testēšana, testētāji gaida, kad sistēmas izstrādātāju darbs būs pabeigts. Viena no stratēģijām, kas tiek izmantota, lai atrisinātu šo problēmu, ir Ad-Hoc stratēģija, kura tiks aplūkota sīkāk nedaudz vēlāk. Tomēr, bieži vien veicot integrācijas testus kādām atsevišķām sistēmas komponentēm, ir nepieciešams izmantot aizbāžņus, jeb angliski "stub" vai draiverus jeb angliski "drivers". Jāatzīst, ka jo ātrāk tiek veikta integrācijas testēšana (visas komponentes vēl nav pabeigtas), jo vairāk resursu un laika ir jāpavada pie aizbāžņu vai draiveru izstrādes. Testa vadītājiem ir jāizvēlas optimāla integrācijas testēšanas stratēģija, lai optimizētu darbu. Katram projektam stratēģija var atšķirties. Lietas, kas jāņem vērā izvēloties stratēģiju:

- Sistēmas arhitektūra – nosaka cik daudz un kuras komponentes eksistē visā sistēmā un kādas ir to savstarpējās atkarības
- Projekta plāns, kurš nosaka laiku, kurā jāiekļaujas komponenšu izstrādei un komponenšu testiem, kā arī laiku, kurā jāveic integrācijas testēšana.
- Testa plāns – nosaka kuri sistēmas aspekti tiks testēti, cik sīki tiks veiktas pārbaudes un kādos līmeņos.

Ņemot vērā visus šos punktus, testa vadītājam ir jāizlemj par testa stratēģiju, un, tā kā integrācijas testēšanas stratēģija ietekmē piegādes datumus, jākonsultējas ar projektu vadītāju. [Spillner et. Al., 2014]

Testa vadītāji var vadīties pēc vispārējām integrācijas testa stratēģijām. Tiek iedalīti vairāki pieeju jeb tehniku veidi kā veikt integrācijas testēšanu:

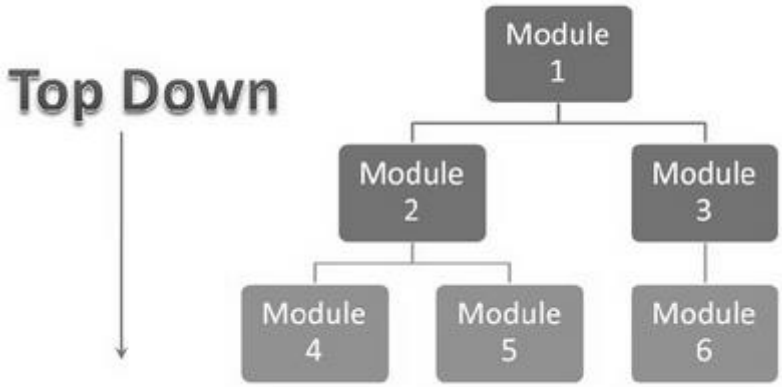
- “Top Down” jeb “No augšas uz leju” pieeja
- “Bottom Up” jeb “No lejas uz augšu” pieeja
- “Ad-Hoc” pieeja
- “Backbone” jeb “Mugurkaula” pieeja
- “Big Bang” tehnika

1.3.1 Top-down integrācijas testēšanas tehnika

Testēšana tiek veikta no augšas un apakšu (sk. attēls 1.2), sekojot datu plūsmai vai arhitektūras struktūrai (piemēram, sākot ar lietotāju saskarni). Apakšējās sistēma tiek aizvietotas ar aizbāžņiem, jeb angliski “stub”. [ISTQB, 2017] Pēc veiksmīgu testu veikšanas, tiek testēta komponente, kas atrodas līmeni zemāk, un augstākā līmeņa komponente kalpo kā draiveris.

Šīs stratēģijas galvenais ieguvums ir tāds, ka nav nepieciešami draiveri (vai nepieciešami ļoti vienkārši izstrādājami draiveri), jo augstākā līmeņa komponentes tiek testētas vispirms.

Šīs stratēģijas galvenais mīnuss ir tāds, ka nepieciešams izstrādāt aizbāžņus, kas bieži vien ir saistīts ar papildus augstākām izmaksām.



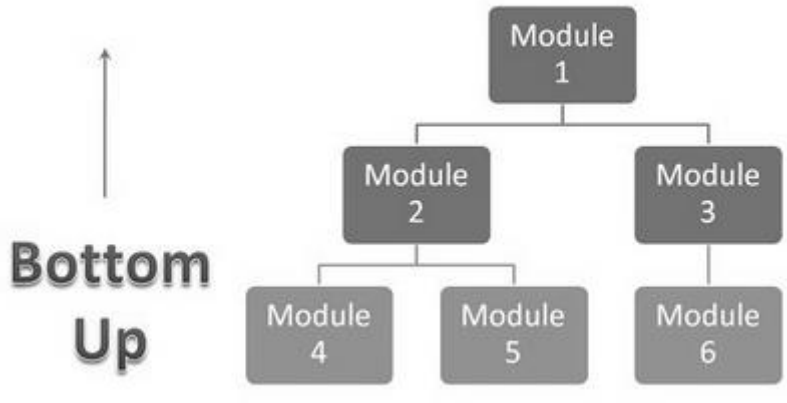
1.3. att. **Top-down integrācijas testēšanas tehnika**

1.3.2 Bottom-up integrācijas testēšanas tehnika

Pretēji iepriekšējai tehnikai, Bottom-up tehnikas gadījumā, testēšana tiek veikta no apakšas uz augšu (sk. attēls 1.3). Augšējās sistēmas tiek aizstātas ar draiveriem, “drivers” (palaiž).

Kā pluss tiek minēts tas, ka nav jāizstrādā sistēmas aizbāžņi.

Savukārt, kā galvenais mīnuss, ir tāds, ka šajā gadījumā diez gan daudz laika (resursu) prasa draiveru izstrāde kā arī galveno moduļu defekti jeb neatbilstības tiks atklātas tikai cikla beigās. [ISTQB, 2017]



1.4. att. **Bottom-up integrācijas testēšanas tehnika**

1.3.3 Ad-Hoc integrācijas testēšanas tehnika

Ad-Hoc stratēģijas gadījumā sistēmas tiek integrētas un testētas tādā secībā, kādā tiek piegādāts sistēmu kods. Tas nozīmē, tiklīdz atsevišķā sistēma jeb komponente ir izstrādāta un notestēta, tā tiek savienota ar otru gatavo sistēmu un tad tiek veikta integrācijas testēšana.

Šīs stratēģijas galvenais pluss ir tāds, ka tiek ietaupīts laiks, jo uzreiz, kad divas sistēmas komponentes ir gatavas, tās tiek integrētas un veikta integrācijas testēšana.

Kā galvenais mīnuss ir tāds, ka ir nepieciešams izstrādāt gan sistēmas aizbāžņus, gan draiverus, kas, kā jau tiek minēts, bieži vien ir saistīts ar augstām izmaksām.[Spillner et. Al., 2014]

1.3.4 Backbone integrācijas testēšanas tehnika

Tiek izveidots skelets jeb mugurkauls, kurā komponentes pakāpeniski tiek integrētas.

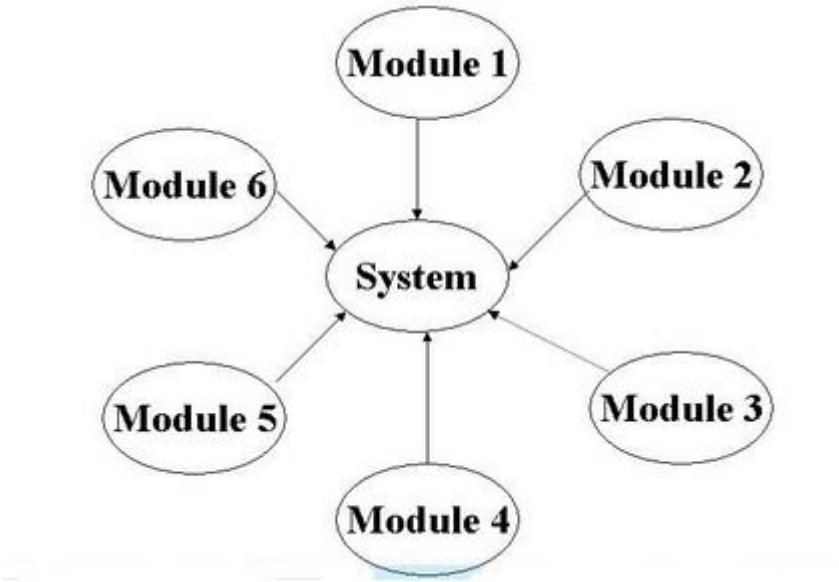
Šīs stratēģijas pluss ir tāds, ka komponentes var tikt integrētas jebkādā secībā.

Kā galvenais mīnuss ir tāds, ka nepieciešams izveidot skeletu, kas ir darbietilpīgs process. [Spillner et. Al., 2014]

1.3.5 Big Bang integrācijas testēšanas tehnika

Big Bang tehnikā (sk. attēls 1.4) visas sastāvdaļas, moduļi ir integrēti vienlaicīgi, pēc kā tiek pārbaudīts viss risinājums kopumā. Big Bang testēšanas priekšrocība ir tā,

ka integrācijas testēšana sākas tikai tad, kad viss ir pabeigts. Savukārt galvenais trūkums ir saistīts ar to, ka integrācijas testēšana ir parasti laikietilpīga un parasti rada problēmas tas, ka kļūdas tiek atklātas ļoti vēlu. [ISTQB, 2017]



1.5. att. **Big Bang integrācijas testēšanas tehnika**

1.3.6 Citi veidi

Teorijā tiek pieminēti tādi veidi kā Sandwich Testing – kas ir apkopojums "Bottom-up" un "Top-Down" tehnikai. Vēl tiek pieminēts arī Risky hardest veids, kura ietvaros integrācijas testēšana tiek sākta riskantākajiem, sarežģītākajiem moduļiem, programmatūrām.

Atsaucoties uz A. Spillner, T. Linz un H. Schaefer grāmatu "Software Testing Foundations", Top-Down vai Bottom-Up stratēģijas var tikt izmantotas "tīrā veidā" tikai sistēmās, kas ir stingri hierarhiskas. Realitātē, tas ir ļoti reti, tādēļ praksē parasti izvēloties turēties pie šīm stratēģijām, tās tiek miksētas savā starpā.

Autors arī min, ka vajadzētu izvairīties no "Big-Bang" stratēģijas, kura bieži tiek izmantota gadījumos, kad trūkst izvēlētās testa stratēģijas. Šajā gadījumā tiek zaudēts laiks, kurš varētu tikt izmantots testēšanai kā arī kļūdas tiek atklātas kopā, un ir grūti tās izlabot laicīgi un pabeigt veiksmīgi testus. [Spillner et. Al., 2014]

1.4 Integrācijas testēšanas organizācija

Integrācijas testēšanu iesaka veikt komandai, kas ir neatkarīga no programmēšanas komandas. Komunikācijai jābūt formālai un kontrolētai – visas pieteiktās kļūdas jārisina caur noteiktu procedūru. [Ould, Unwin, 1988]

Integrācijas testēšanas gaitu iesaka veikt pa punktiem – sagatavošanās testa veikšanai, testa izpildi, rezultātu pārbaudi. Tas nodrošinās, ka visas soļu savstarpējās atkarības tiek saglabātas. Būtiski, lai testa rezultāti tiktu saglabāti un būtu pieejami citām ieinteresētajām pusēm. [Ould, Unwin, 1988] Ļoti būtiski ir pārliecināties, ka vienībtestēšana ir notikusi veiksmīgi, pirms tiek uzsākti integrācijas testi. [Software Testing Fundamentals, 2017]

Integrāciju testēšana jāveic testa vidē, kurā būtu iespējams fiksēt atklātās kļūdas un veikt sistēmas labojumus. Tāpat, iesaka izmantot programmatūras konfigurācijas pārvaldības sistēmu, lai sekotu, ka visas komponentes ir korektas versijas. [Software Testing Fundamentals, 2017]

Shilpa C. Roy iesaka:

1. Izprast lietojumprogrammas arhitektūru.
2. Identificēt moduļus.
3. Izprast, kam paredzēts katrs modulis.
4. Izprast, kā dati tiek nodoti no viena moduļa citam.
5. Izprast, kā dati tiek ievadīti un saņemti sistēmā (lietojumprogrammas ieejas un izejas punkts).
6. Nodalīt komponentes, kas atbilst testēšanas vajadzībām.
7. Identificēt un radīt testa apstākļus.
8. Izmantojot vienu nosacījumu, veidot testa piemērus. [Roy, 2017]

2. VADLĪNIJU IZSTRĀDE EFEKTĪVAI INTEGRĀCIJAS TESTĒŠANAI

Izpētot dažādus avotus par integrācijas testēšanu, tās organizēšanu, stratēģijas izvēli, autore secina, ka galvenās problēmas ir saistītas ar nepilnīgu sagatavošanos integrācijas testēšanai kā arī laika trūkumu. Tālāk nodaļā autore apkopos ieteikumus un izstrādās vadlīnijas, pēc kurām būtu jāvadās, veicot integrācijas testēšanu.

2.1 Kvalitatīva integrācijas testēšanas plāna izstrāde

Kā pirmais autores ieteikums būtu laika un resursu plānošana kvalitatīva integrācijas testēšanas plāna izstrādei. Integrācijas testēšanas plānā ļoti būtiski ir izvēlēties testa stratēģiju, testēšanas apjomu, identificēt testa specifikāciju, atzīmēt katra integrācijas testa mērķi un secību, kādā būtu jāveic testi, iepazīstināt ar plānu visas ieinteresētās puses un projekta laikā sekot līdzi plānam.

2.2 Laika izmantošana, kamēr komponentes nav piegādātas

Otrais ieteikums būtu laiku, kas tiek pavadīts gaidot, kamēr sistēmas komponenšu izstrādātāji būs pabeiguši programmēšanas vai konfigurācijas darbus un kad komponente tiks notestēta atsevišķi, pavadīt lietderīgi – integrācijas testētāji pa šo laiku var iepazīties ar integrācijas testēšanas plānu un izstrādāt testa scenārijus, ņemot vērā izvēlēto testa stratēģiju. Tāpat, šajā laikā var tikt izstrādāti sistēmas aizbāžņi vai draiveri, ja plānots, ka tādus testos būs nepieciešams izmantot. Autore vēlas atzīmēt, ka šajā posmā svarīgi nodrošināt labu komunikāciju starp testēšanas vadītāju un integrācijas testētājiem (jautājumos par testa stratēģiju, plānu), integrācijas testētājiem un arhitektiem un analītiķiem (jautājumos par izstrādājamo sistēmu), integrācijas testētājiem un programmētājiem (jautājumos par problemātiskām vietām, piegādēm).

2.3 Testa scenāriju izstrāde

Trešais ieteikums ir saistīts ar testa scenāriju izstrādi. Testa scenāriji jāizstrādā apdomīgi, ņemot vērā, cik lielu apjomu būs iespējams pārbaudīt, tajā pašā laikā pārbaudot visas svarīgās daļas. Kā viens no variantiem, ko autore iesaka izmantot, ir izmantot dažādus resursus (darbiniekus) testa scenāriju izveidošanā un izpildē. Tādā veidā tiks izslēgta iespējamība, ka testa scenāriji izstrādāti pavirši, izlaižot svarīgas daļas, jo apzināti vai neapzināti darbinieks vēlējies samazināt sava darba apjomu. Otrs

ieguvums – ar prasībām būs iepazinušies divi darbinieki, un testa scenāriju izstrāde un izpilde nebūs aplūkota tikai no vienas puses. Integrācijas testēšanā, ļoti iespējams, būtu noderīgi iepazīties ar sistēmtesta (komponenšu) testa scenārijiem, lai tie nepārklātos. Vēl viens papildus ieteikums varētu būt saistīts ar balstīšanos uz pieredzi – uz vēsturiski atklātajām kļūdām, statistiku. Būtu nepieciešams pievērst paaugstinātu uzmanību tām sistēmas daļām, tiem moduļiem, kur visbiežāk novērotas kļūdas.

2.4 Uz lietotāju scenārijiem balstīts testēšanas modelis

Izpētot literatūru par integrācijas testēšanas metodēm, ņemot vērā metodes plusus un mīnusus, autore piedāvā izstrādāt lietotāju scenāriju modeli, uz kuriem tiktu balstīta integrācijas testēšana. Kelly Waters rakstā "User Stories Should Be *Testable*" [Waters, 2008] tiek noteikts, ka galvenais nosacījums lietotāju stāstam ir tāds, ka tam jābūt "notestējamam". Lietotāju stāsts var apvienot vairākus "notestējamus" lietotāju scenārijus (no sistēmas puses), tādēļ autore piedāvā izstrādāt modeli, kurā būtu apkopoti lietotāju stāsti (un no tiem izrietošie scenāriji). Šādas metodes izmantošanai vajadzētu palīdzēt testētājiem ietaupīt laiku (pēc iespējas apvienojot vairākus scenārijus) un pievērst lielāku uzmanību galvenajiem svarīgākajiem un biežākajiem) scenārijiem.

Šāda modeļa izstrādei būtu nepieciešams apkopot lietotāju scenārijus pārskatāmā modelī, kas pārklātu visas IS komponentes. Šāda modeļa izstrāde palīdzētu integrācijas testētājiem aptvert visus sistēmas atzarus kā arī izslēgt tos scenārijus, kurus sistēmā ir iespējams veikt, bet kas netiek, un netiks izmantots gala lietotājiem. Šajā gadījumā tiks patērēts laiks un resursi lietotāju scenāriju apkopošanā, bet ilgtermiņā tas palīdzēs ietaupīt resursus scenāriju veidošanas posmā kā arī izpildē.

Tāpat būtu vērtīgi sīkāk aplūkot scenārijus – veidojot UML aktivitāšu diagrammas. Ļoti iespējams, ka šāda diagramma ļautu vēl pirms izstrādes atklāt potenciālās kļūdas, kas varētu būt saistītas gan ar pašu izstrādi, gan pašu scenāriju un izvirzītajām prasībām.

Autore uzskata, ka integrācijas testēšanas fāze ir viena no pēdējām fāzēm, kuras laikā ir iespējams pārbaudīt sistēmu visās tās komponentēs (pirms akcepttestēšanas, kas pēc autores uzskatiem ir jau par vēlu, jo tad kļūdu un nepilnību atrašana saistās ar daudz lielākām izmaksām) un ir svarīgi aplūkot sistēmu no lietotāja skatupunkta, jo tieši šādi lietotājs (pasūtītājs) izmantos IS.

Pēc lietotāju scenāriju modeļa izstrādes, autore piedāvā integrācijas testēšanu veikt pēc plāna:

1. Integrācijas testētāji iepazīstas ar integrācijas testēšanas plānu (paralēli sistēmas komponenšu izstrādei un komponenšu testiem)
2. Integrācijas testētāji, izmantojot izstrādāto lietotāju scenāriju modeli, izstrādā detalizētākus testa scenārijus (ar konkrētiem parametriem un sagaidāmajiem rezultātiem), kuros tiek pārbaudītas tās sistēmas daļas, kurās bijušas izmaiņas (paralēli sistēmas komponenšu izstrādei un komponenšu testiem)
3. Uzsākt testus – sākot ar tiem scenārijiem, kurus ir iespējams pilnība izpildīt (visas komponentes piegādājušas izmaiņas). Analizējot konkrēto situāciju – izvērtēt vai ir vērts izstrādāt draiverus vai aizbāžņus, gadījumos, kad kāda sistēmas komponenšu izstrāde un testēšana aizkavējas, vai gaidīt, kamēr tā tiek piegādāta. Šajā solī būtu svarīgi, ka integrācijas testētāji ir kvalificēti un paši spējīgi pieņemt adekvātus lēmumus (iespējams, ņemot vērā cilvēcisko faktoru, šis lēmums būtu jāpieņem integrācijas testa vadītājam).

3. LIETOTĀJU SCENĀRIJU MODEĻA IZSTRĀDE UN INTEGRĀCIJAS TESTĒŠANA ATBILSTOŠI VADLĪNIJĀM

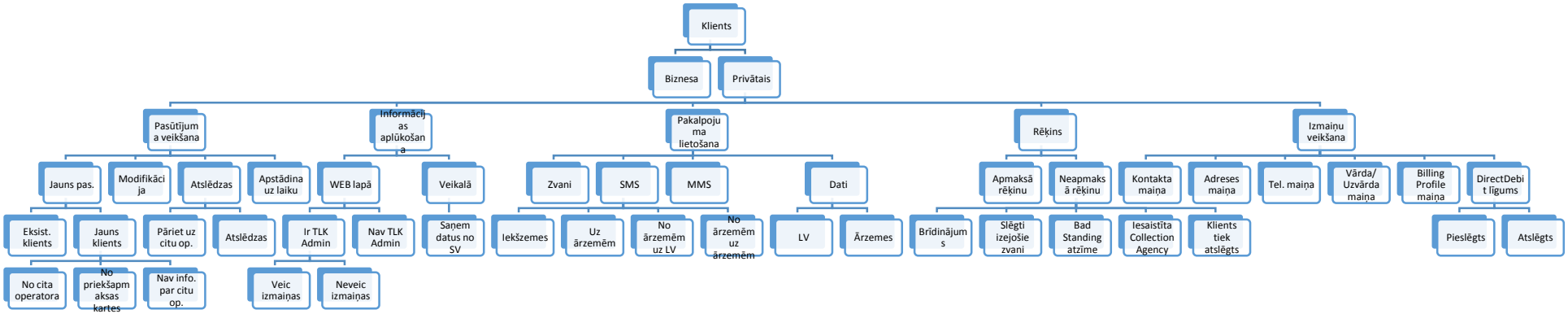
Lai pārbaudītu, vai iespējams praksē ievērot 2. nodaļā izvirzītās vadlīnijas, kā arī secinātu, vai sekošana izvirzītajām vadlīnijām sniedz kādu labumu, autore veiks pētījumu, izmantojot komplicētu IS risinājumu, kas sastāv no vairākām dažādām sistēmām (komponentēm). Konkrētais risinājums ietver tādas komponentes kā klientu pārvaldības sistēma (turpmāk tekstā CRM), pasūtījumu veidošanas sistēma (izmantots veikalos) (turpmāk tekstā PVS), norēķinu sistēma (turpmāk tekstā NS), tīmekļa lapa klientiem (TLK) un citas svarīgas sistēmas komponentes.

Vispirms tiek izveidots lietotāju scenāriju modeļa uzmetums (attēls 3.1), kurš iekļauj galvenos lietotāju stāstus un no tiem izrietošās testa scenāriju kopas un apakškopas. Tiek secināts, ka visu lietotāju scenāriju pamatā ir klients – biznesa vai privātpersona. Tālāk tiek iedalītas 5 galvenās jomas:

- Pasūtījumu veikšana;
- Informācijas aplūkošana;
- Pakalpojuma lietošana;
- Rēķins;
- Izmaiņu veikšana

Sīkāk modelī tiek iekļautas detaļas, kas saistītas ar pašu scenāriju un sistēmas funkcionalitāti – tiek atsevišķi atzīmēti scenāriji, kur pasūtījums ir saistīts ar jauna klienta reģistrāciju, modifikāciju, pakalpojuma atslēgšanu, atsevišķi tiek nodalīta informācijas aplūkošana no WEB lapas un informācijas pieprasīšanu veikalā. Līdzīgā veidā tiek apkopoti scenāriji, kas izriet pēc rēķina apmaksas vai neapmaksāšanas (noteiktā laika periodā), un tiek atdalīti vairāki veidi, kādas izmaiņas pēc lietotāju scenārijiem ir iespējams veikt klientam.

3.1 Lietotāju scenāriju modeļa izstrāde

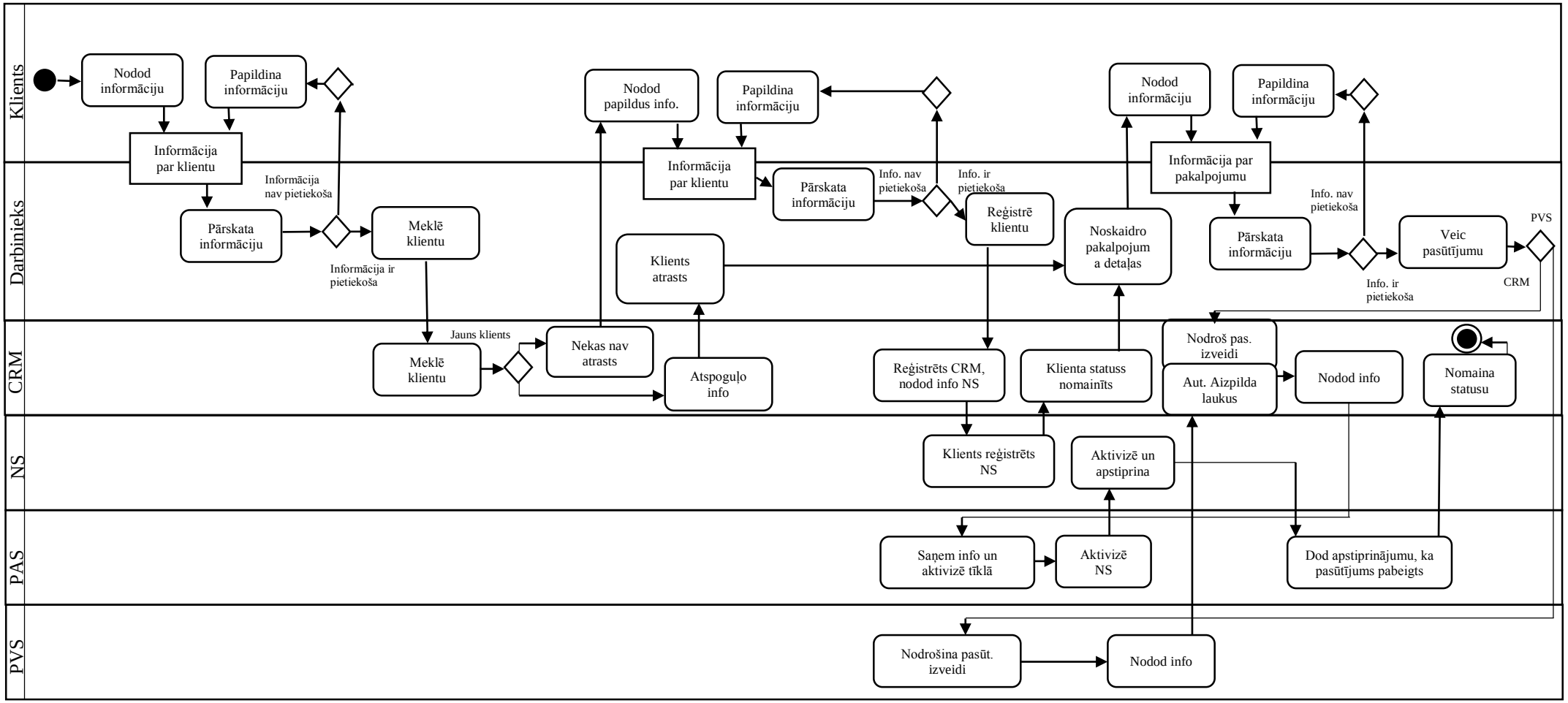


3.1. att. **Lietotāju scenāriju pilnais modelis IS**

Pēc 3.1 attēlā pieejamās informācijas, ir iespējams sīkāk nošķirt lietotāju scenārijus un aplūkot tos sīkāk, veidojot UML aktivitāšu diagrammas. Kā pirmo, sīkāk aplūkojamo lietotāju scenāriju kopu, autore izvēlējās vienu no populārākajiem scenārijiem – jauna pieslēguma izveidošana jaunam (izslēdzot gadījumu, kad klients pāriet no cita operatora vai priekšapmaksas kartes) vai eksistējošam klientam kā arī pieslēguma modificēšana. Tāpat, šī aktivitāšu diagramma attēlo pakalpojuma apstādināšanu uz laiku (esošs klients).

Aktivitāšu diagramma (sk. 3.2. att.) sākas ar notikumu, ko izraisa klients – klients sazinās ar klientu apkalpošanas centru un nodod informāciju. Vispirms darbiniekam nepieciešams apstrādāt klienta informāciju, secināt vai zvanītājs ir eksistējošs klients vai jauns potenciālais klients. Gadījumā, ja klients netiek atrast CRM sistēmā, tad darbinieks veic klienta reģistrēšanu (sākumā tiek reģistrēts CRM savukārt CRN nodod klienta datus NS). Kad tiek noskaidrots zvanītāja statuss, darbinieks var uzklausīt zvanītāju un izveidot pasūtījumu (gan jaunam, gan eksistējošam klientam). Pasūtījums tiek veidots izmantojot speciālu sistēmu PVS, kura satur biznesa loģiku un neļauj darbiniekam veikt pasūtījumu, kas nav atbilstošs uzņēmuma piedāvājumiem vai otrs variants – veidot pasūtījumu uzreiz CRM sistēmā. Pēc pasūtījuma izveidošanas PVS nodod pasūtījuma detaļas CRM sistēmā, kura automātiski aizpilda visus laukus, ņemot vērā informāciju no PVS. Abos gadījumos, pēc tam automātiski pasūtījums tiek iesniegts apstrādei PAS sistēmā, kura aktivizē pakalpojumu tīklā un NS. Kad saņemts apstiprinājums, ka klients ir aktivizēts NS, PAS sistēma dod apstiprinājumu CRM par pasūtījuma veiksmīgu izpildi, un CRM nomaina statusu.

Lai pārbaudītu visus iespējamos variantus (izietu visus aktivitāšu diagrammas ceļus), nepieciešams veikt tikai trīs testa scenārijus. Aptuvenais laiks, lai izpildītu scenārijus – 90 minūtes (30 minūtes katrs scenārijs).



3.2. att. **Aktivitāšu diagramma – Jauna pakalpojuma pieslēgšana, pakalpojuma modificēšana eksistējošam vai jaunam klientam**

Turpinot ar pasūtījuma izveidošanas atzaru, autore sīkāk aplūko gadījumu, kad klients vēlas atslēgt savu pakalpojumu (sk. 3.3 att.).

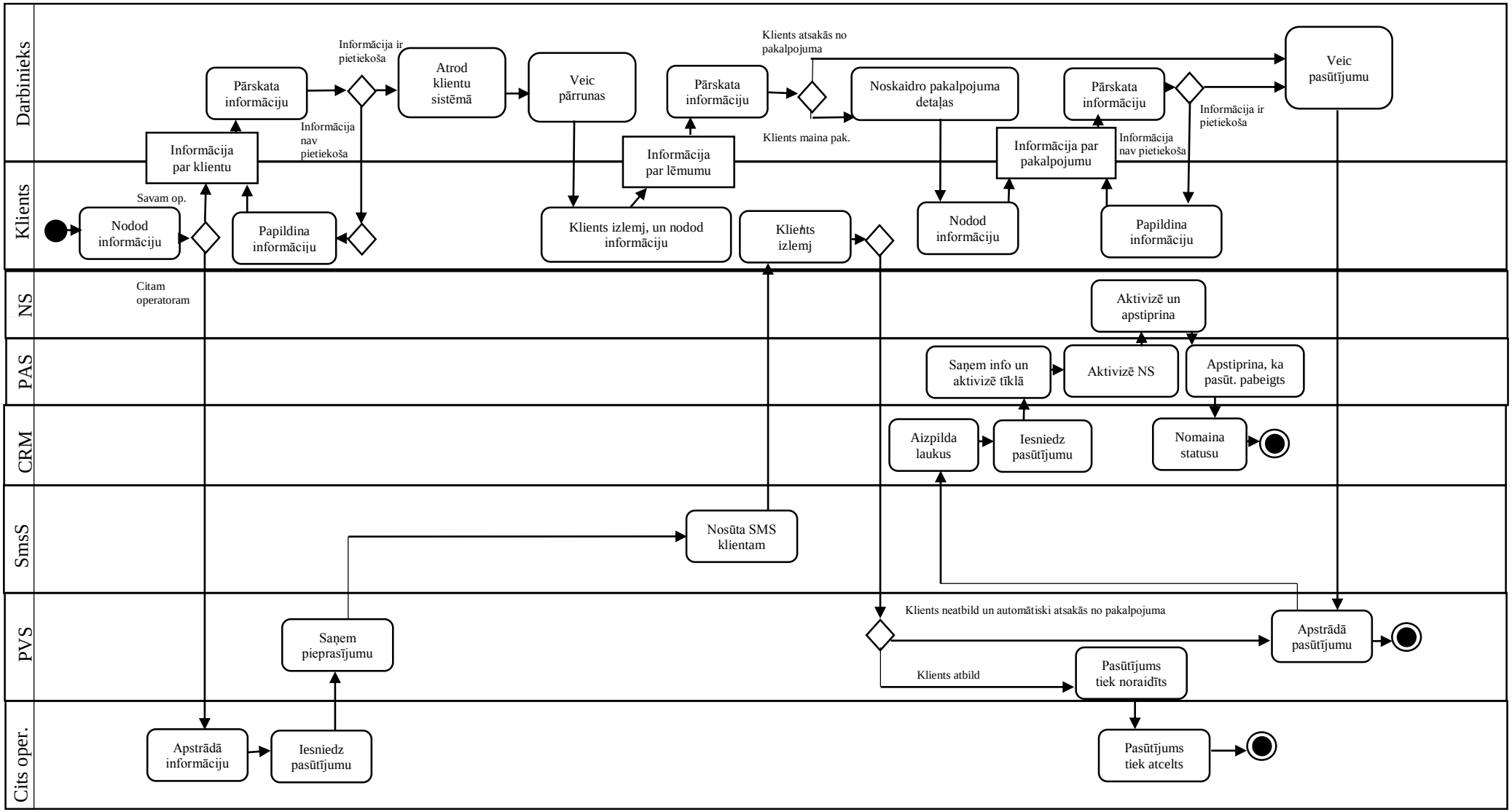
Tiek izveidota aktivitāšu diagramma, kas attēlo lietotāju scenāriju, kad klients sazinās ar klientu apkalpošanas centru, nodod informāciju par sevi un informāciju par pieslēgumu, no kura vēlas atteikties. Tāpat šajā diagrammā aplūkota situācija, kad informācija par atteikšanos no pakalpojuma saņemta automātiski informācijas sistēmā no cita pakalpojuma sniedzēja.

Aktivitāšu diagramma sākas ar notikumu (sk. 3.3. att.) sākas ar notikumu, ko izraisa klients – klients sazinās ar klientu apkalpošanas centru vai citu operatoru un nodod informāciju par vēlmi mainīt operatoru.

Gadījumā, ja klients sazinās ar klientu apkalpošanas centru, darbinieks atrod klientu sistēmā, veic pārrunas, kuras rezultātā klients izvēlas – atteikties no pakalpojuma vai mainīt pakalpojuma detaļas. Abos gadījumos rezultātā tiek veidots jauns pasūtījums, kurš tiek reģistrēts CRM sistēmā. CRM sistēma automātiski nodod informāciju PAS, kura veic izmaiņas tīklā un NS.

Gadījumā, ja klients ir sazinājies ar citu operatoru, tad cits operators iesniedz pieprasījumu, kas automātiski nonāk PVS. Pēc tam SmsS (SMS sūtīšanas sistēma) nosūta īsziņu klientam, kuram ir iespēja izlemt vai atteikties no pakalpojuma vai tomēr neatteikties. Gadījumā ja klients atbild apstiprinoši – pasūtījums tiek atcelts, bet gadījumā, ja klients neatbild – automātiski tiek iesniegts pasūtījums CRM sistēmā, kurš tiek apstrādāts.

Autore secina, ka šī modeļa pārbaudei nepieciešams veikt četrus testa scenārijus – divus scenārijus sava operatora ietvaros un divus – cita operatora. Aptuvenais kopējais scenāriju izpildes laiks ir 120 minūtes (25 minūtes cita operatora scenārija gadījumā; 35 minūtes, ja iesaistīts tikai savs operators).



3.3. att. **Aktivitāšu diagramma – Klients vēlas atteikties no pieslēguma**

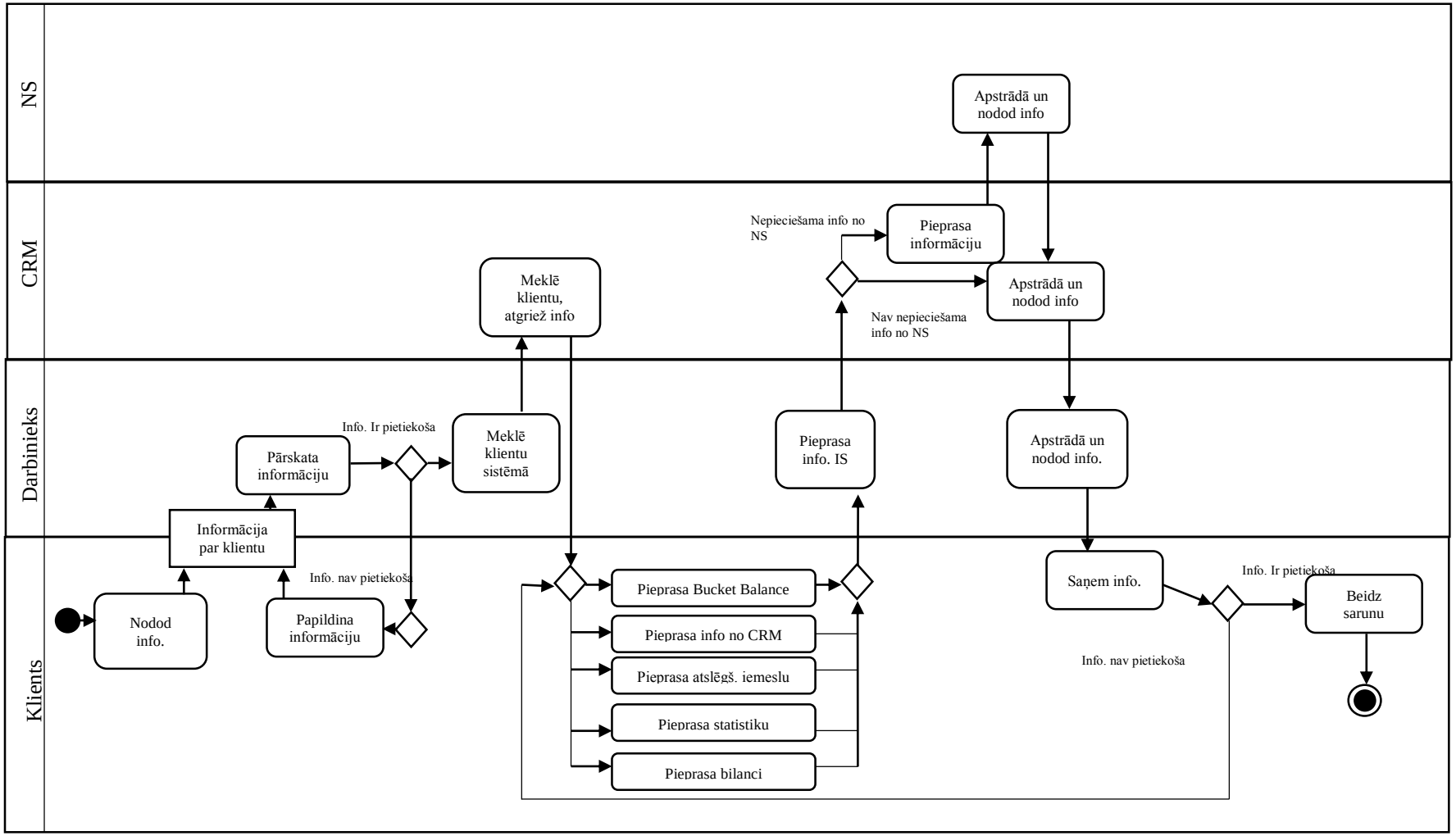
Līdzīgā veidā autore izanalizē un izveido diagrammu Informācijas aplūkošanas scenārijam. Kā tika attēlots 3.1 modelī – informācija var tikt iegūta divos veidos – apmeklējot veikalu (vai sazinoties telefoniski) vai apmeklējot WEB vietni – TLK.

Vispirms tiek izveidots modelis scenārijam, kad klients griežas pie klientu apkalpošanas darbinieka, lai noskaidrotu sev interesējošo informāciju (sk. 3.4 att.).

Aktivitāšu diagramma sākas ar notikumu, ko izraisa klients. Darbinieks tālāk pārskata informāciju. Gadījumā, ja informācija nav pietiekoša, klients papildina informāciju, bet gadījumā, ja ir pietiekoša, meklē klientu uzņēmuma IS, precīzāk CRM sistēmā. Kad klients ir atrasts, klients pieprasa informāciju - pieprasa informāciju par tekošo bilanci, Bucket Balance, informāciju no CRM (piemēram, pieslēgtie telefona numuri), informāciju par statistiku vai lietojumu.

Tālāk darbinieks pieprasa šo informāciju no CRM sistēmas, kas izanalizē, vai nepieciešama papildus informācija no NS, vai nē. Gadījumā, ja nepieciešama papildus informācija no NS, CRM sistēma pieprasa informāciju un pēc saņemšanas apstrādā. Savukārt, ja papildus informācija nav nepieciešama, CRM sistēma apstrādā un attēlo informāciju. Tālāk darbinieks apstrādā un nodod informāciju klientam. Ja klientam jautājumu vairāk nav, tad saruna tiek beigta, bet, ja klients vēlas noskaidrot vēl ko papildus, tad darbinieks atkārtoti pieprasa informāciju no IS un nodod to tālāk klientam.

Autore secina, ka šī modeļa pārbaudei nepieciešams izpildīt sešus testa scenārijus (pārbauda katru klienta pieprasījumu atsevišķi un vienā no scenārijiem tiek pārbaudīti vairāki klienta pieprasījumi). Autore uzskata, ka aptuvenais viena scenārija laiks ir 20 minūtes – kopā 120 minūtes.



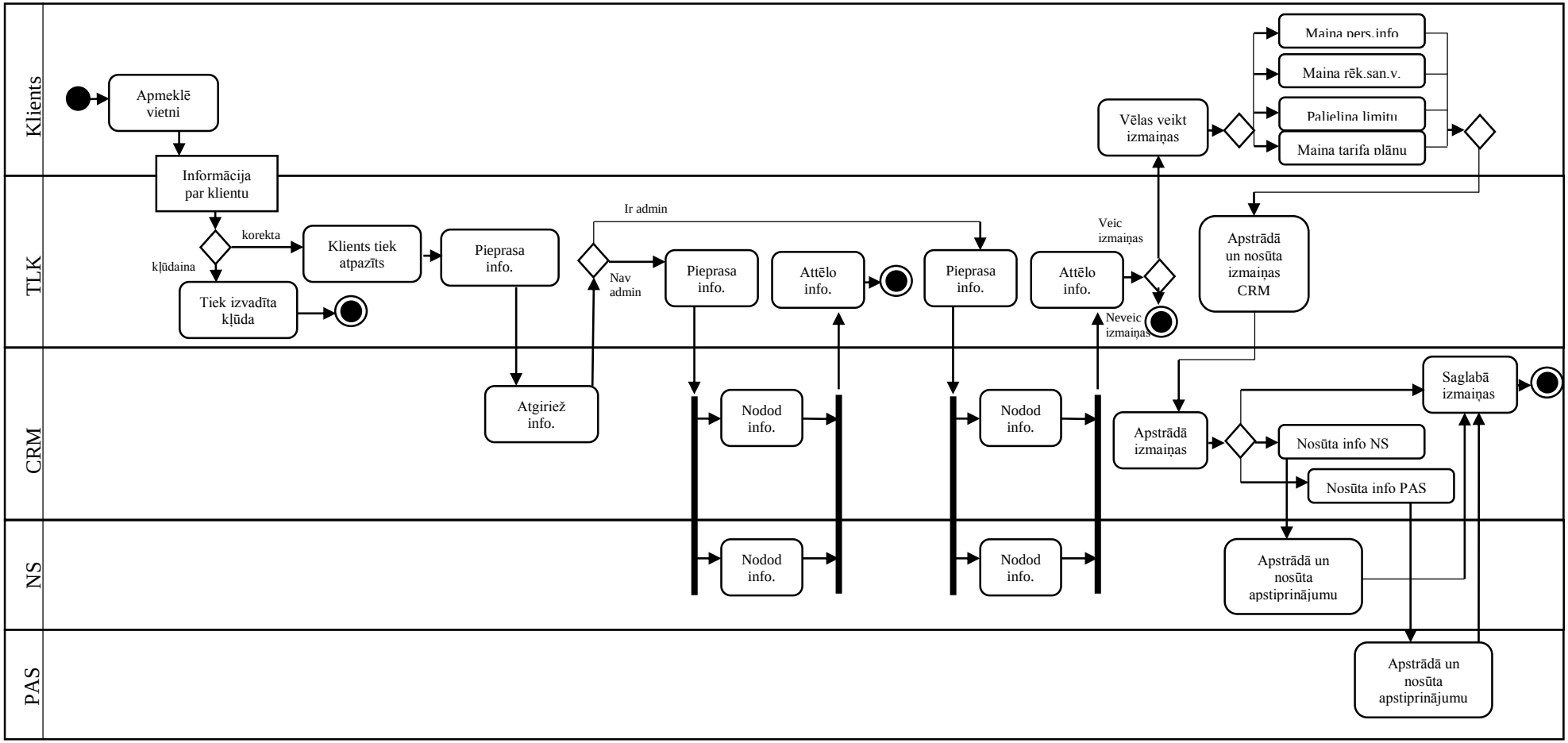
3.4. att. **Aktivitāšu diagramma – Klients saņem informāciju no klientu apkalpošanas daļas**

Tālāk tiek izveidota aktivitāšu diagramma otram lietotāju scenāriju veidam (sk. 3.5 att.) – gadījumam, kad klients izmanto WEB pārlūku, piekļūst TLK sistēmai un aplūko informāciju.

Aktivitāšu diagramma sākas ar tīmekļa vietnes apmeklēšanu, ko veic klients. Tad klients norāda informāciju par sevi un TLK veic analīzi vai informācija (pieslēgšanās informācija) ir korekta vai kļūdaina. Gadījumā, ja norādītā informācija ir kļūdaina, sistēmas izvada kļūdu. Savukārt, ja informācija ir korekta, klients tiek atpazīts un tiek pieprasīta aktuālā informācija no CRM. (TLK pati neuzglabā informāciju). CRM sistēma atgriež informāciju un TLK izanalizē vai klients ir TLK admin (papildus tiesības) vai nav. Atkarībā no šīm tiesībām, TLK attēlojamā informācija atšķiras, tāpat, gadījumā, ja klients ir TLK admin, tad klients var veikt arī izmaiņas, piemēram, nomainīt kontakttālruni.

Gadījumā, ja klientam ir administratora tiesības, klients var izvēlēties veikt vai neveikt izmaiņas. Gadījumā, ja klients neveic izmaiņas, lietotāju scenārijs tiek pabeigts savukārt, ja klients vēlas veikt izmaiņas, klients tās veic TLK vietnē, vietne apstrādā un nodod izmaiņas CRM sistēmai, kas apstrādā informāciju un analizē vai to tālāk nepieciešams nodot NS vai PAS.

Lai tiktu pārbaudīts šis modelis, nepieciešams izpildīt 7 testa scenārijus (kļūda no TLK; nav admin; ir admin, bet neveic izmaiņas; maina personīgo informāciju; maina rēķina saņemšanas veidu; palielina limitu; maina tarifa plānu). Aptuvenais laiks vienam scenārijam 20 minūtes. Kopā aptuveni 140 minūtes.



3.5. att. **Aktivitāšu diagramma – Klients saņem informāciju no TLK**

Kā nākamais tiek aplūkots pakalpojuma lietošanas modelis (sk. 3.6 att.). Šī modeļa ietvaros klients ar aktīvu pieslēgumu izvēlas lietot (vai nelietot) dažādus pakalpojumus – veikt zvanus, sūtīt SMS / MMS, lietot datus.

Tā kā atkarībā no pieslēguma veida un klienta pastāv dažādi ierobežojumi, pēc zvana uzsākšanas tīklā tiek veikta pārbaude, lai noteiktu vai konkrētajam klientam konkrētajā brīdī pakalpojums būs pieejams. Rezultātā zvans tiek noraidīts vai nodrošināts. Zvanu iespējams beigt divos veidos – klients pats pārtrauc zvanu vai arī zvana laikā tiek sasniegts iepriekš noteiktais limits, kā rezultātā SmsS sistēma nosūta brīdinājuma SMS, PAS uzliek konkrētam klientam ierobežojumu, un tad tīklā zvans tiek pārtraukts. Pēc zvana beigšanas (abos gadījumos) tīkls ģenerē failus, kas satur informāciju par veikto zvanu, un nosūta to NS, kas apstrādā failus.

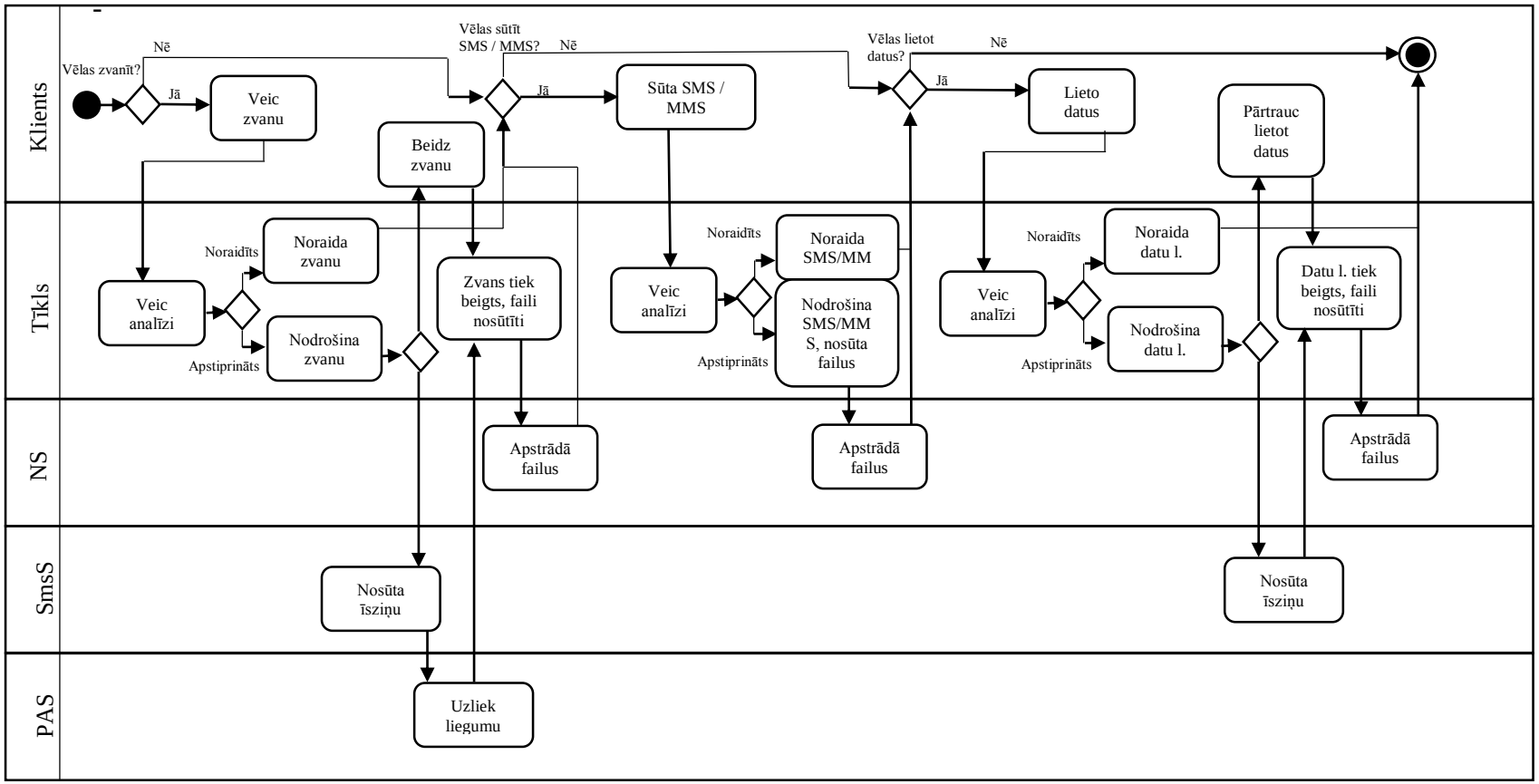
Sūtot SMS / MMS, tīklā tiek veikta analīze vai konkrētajam klientam ir ļauts sūtīt SMS / MMS. Attiecīgi pēc analīzes tīklā pakalpojums tie nodrošināts vai noraidīts. Pēc SMS / MMS sūtīšanas nodrošināšanas, tīkls ģenerē un nosūta failus NS, kas apstrādā failus.

Līdzīgā veidā notiek datu lietošana. Tīklā tiek veikta analīze, vai konkrētam klientam datu lietošana ir ļauta. Rezultātā datu sesijas uzsākšana tiek noraidīta vai nodrošināta. Līdzīgi kā zvanu gadījumā, datu sesiju iespējams beigt divos veidos – klients pats pārtrauc lietot datus vai arī datu lietošanas laikā tiek sasniegts iepriekš noteiktais limits, kā rezultātā SmsS sistēma nosūta brīdinājuma SMS un tīklā datu lietošana tiek pārtraukta. Pēc zvana beigšanas (abos gadījumos) tīkls ģenerē failus un nosūta to NS, kas apstrādā failus.

Autore secina, ka šajā gadījumā būtu jāizpilda četri testa scenāriji:

1. Zvans, kas tiek pārtraukts sasniedzot limitu, noraidīta SMS, nodrošināts datu lietojums, kas tiek pārtraukts, jo klients pārtrauc sesiju;
2. Noraidīts zvans, noraidīta SMS, noraidīti dati;
3. Nodrošināts zvans, kas tiek pārtraukts, jo klients beidz zvanu, nodrošināta SMS (dati netiek izmantoti);
4. Datu lietojums, sasniedzot limitu (zvans, SMS netiek izmantoti)

Aptuvenais laiks viena scenārija izpildei 50 minūtes, kopā 200 minūtes.



3.6. att. **Aktivitāšu diagramma – Pakalpojumu lietošana**

Nākamais (sk. 3.7 att.) autores aplūkotais ir rēķina izveides un apmaksāšanas modelis.

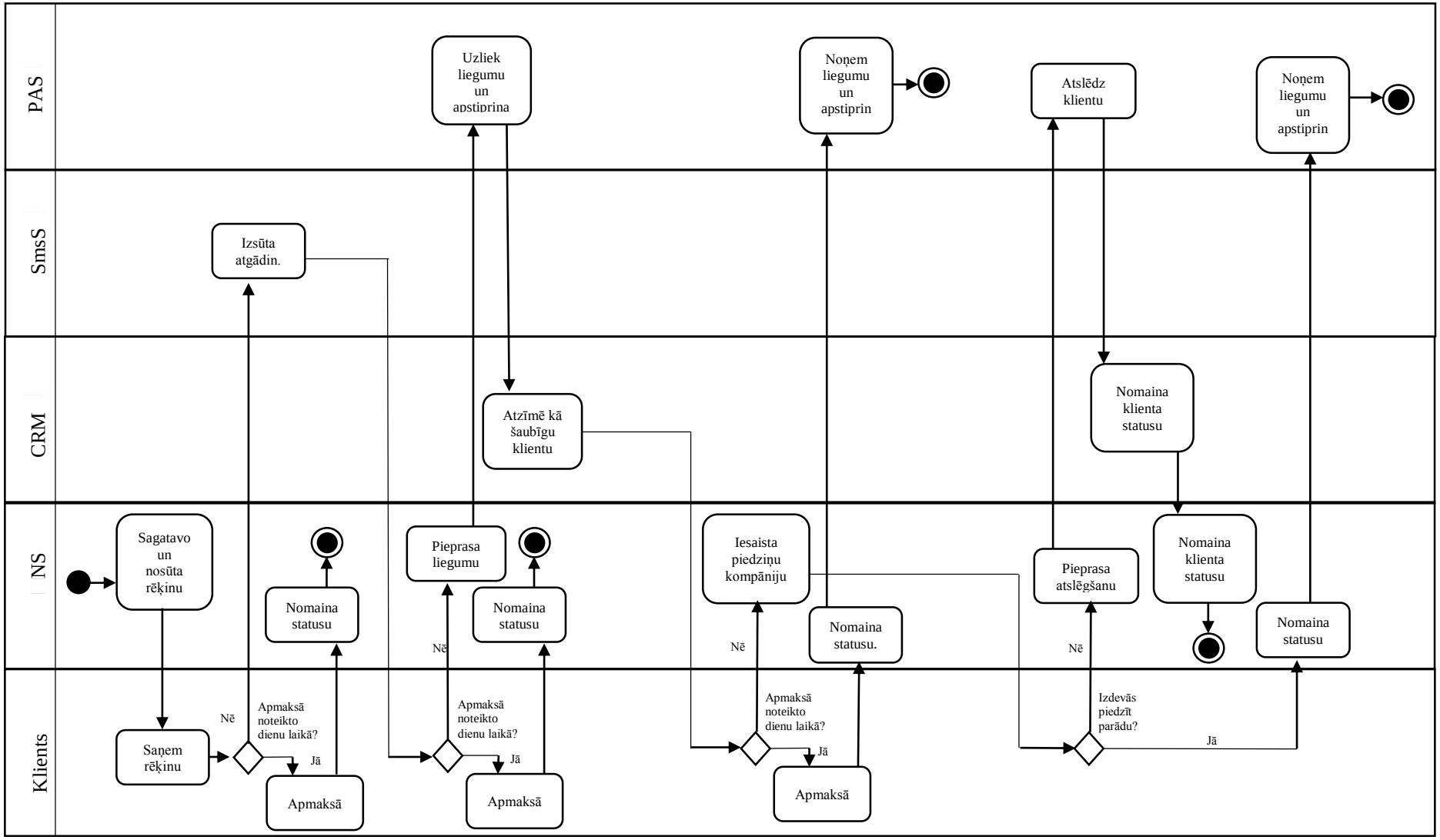
Vispirms NS izveido rēķinu (parasti tas notiek mēneša 1. datumā), tad nosūta to klientam. Pēc tam, kad klients ir saņēmis rēķinu, tiek gaidīts, kad rēķins tiks apmaksāts. Gadījumā, ja rēķins netiek apmaksāts noteiktu dienu laikā, SmsS automātiski nosūta atgādinājuma SMS klientam par rēķina apmaksu. Savukārt, ja rēķins tiek laicīgi apmaksāts, NS nomaina rēķina statusu.

Gadījumā, ja pēc atgādinājuma SMS saņemšanas noteiktu dienu laikā rēķins netiek apmaksāts, NS pieprasa PAS sistēmai uzlikt konkrētajam klientam liegumu. Pēc lieguma uzlikšanas klientam CRM sistēmā tiek uzlikta atzīme (kas palīdz darbiniekiem atšķirt regulārus maksātājus no nemaksātājiem). Ja rēķins pēc atgādinājuma SMS saņemšanas tiek apmaksāts, NS tiek nomainīts rēķina statuss.

Ja pēc lieguma uzlikšanas noteiktu dienu laikā rēķins (rēķini) netiek apmaksāti, tiek iesaistīta piedziņas kompānija. Savukārt, ja rēkins tiek apmaksāts, NS nomaina statusu un PAS sistēma noņem liegumu.

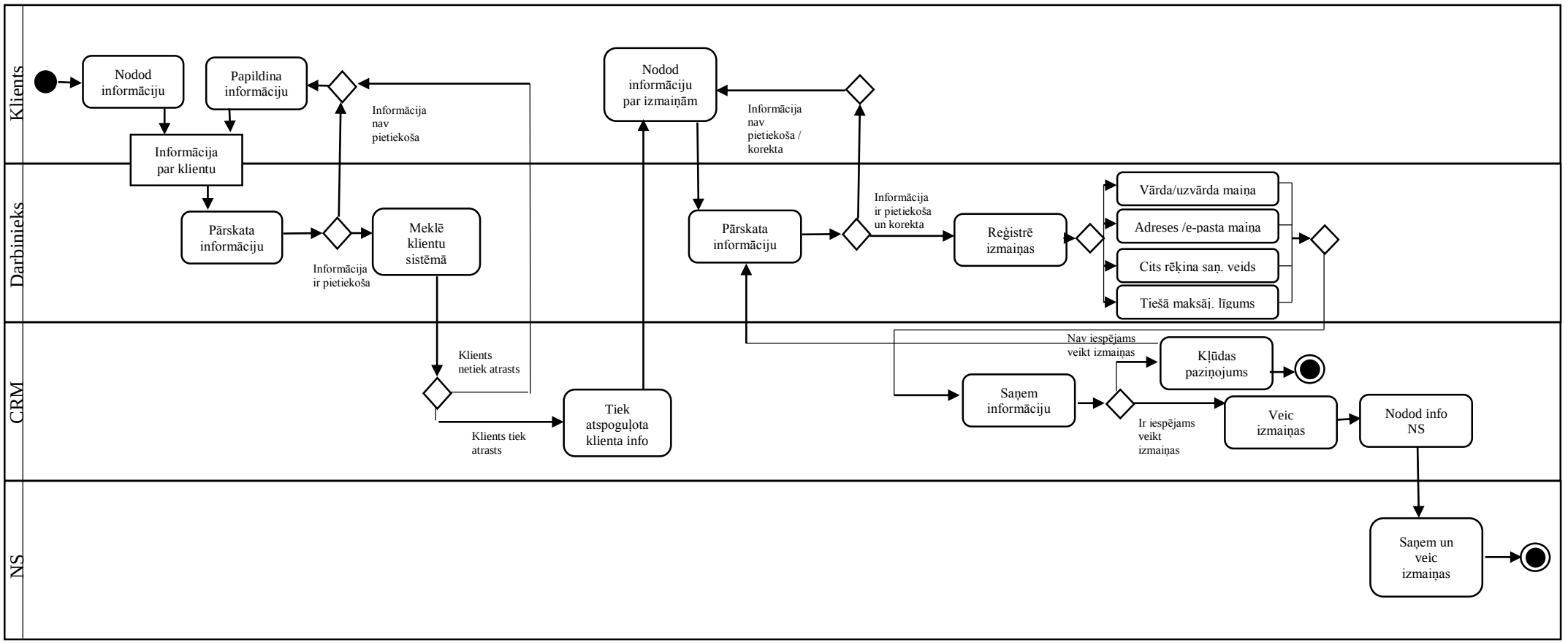
Gadījumā, ja piedziņu kompānijai nav izdevies piedzīt klienta parādu, NS pieprasa numura atslēgšanu, ko veic PAS. Pēc atslēgšanas klienta statuss tiek nomainīts (uz neaktīvs) gan CRM sistēmā, gan NS. Ja piedziņu kompānijai izdevies piedzīt parādu, rēķina statuss tiek nomainīts NS un PAS noņem liegumu.

Šī modeļa ietvaros būtu nepieciešams izpildīt 5 testa scenārijus – apmaksā rēķinu uzreiz; apmaksā pēc atgādinājuma; apmaksā pēc lieguma uzlikšanas; piedziņu kompānijai izdodas un neizdodas piedzīt parādu. Aptuvenais laiks, kas nepieciešams šo scenāriju izpildei – 150 minūtes.



3.7. att. **Aktivitāšu diagramma – Rēķina izveide un apmaksa**

Pēdējais tiek aplūkots izmaiņu veikšanas modelis (sk. 3.8 att.). Šajā modelī tiek aplūkots gadījums, kad klients vēlas veikt kādas izmaiņas, kas saistītas ar klienta profilu – vārda, uzvārda maiņa, rēķina saņemšanas veids, adreses vai citas kontaktinformācijas izmaiņas. Visas izmaiņas tiek apstiprinātas no klienta apkalpošanas speciālista puses, un tad, tiek veiktas izmaiņas informācijas sistēmā. Klientu apkalpošanas speciālists izmaiņas reģistrē CRM sistēmā, kas, balstoties uz sistēmas nodefinēto loģiku, pārbauda, vai izmaiņas iespējams veikt, un pozitīvā gadījumā, automātiski nodod informāciju maksājuma (NS) sistēmai. Šī modeļa pārbaudei būtu nepieciešams izpildīt 5 testa scenārijus, aptuvenais laiks izpildei – 100 minūtes.



3.8. att. **Aktivitāšu diagramma – Klienta informācijas rediģēšana**

3.2 Lietotāju scenāriju modeļa izveides pamatojums

Galvenais iemesls, kādēļ autore nolemj balstīties uz lietotāju scenārijiem ir saistīts ar testa scenāriju skaitu. Lai pārbaudītu visas sistēmas integrācijas, tiek veidoti ļoti daudz testa scenāriji, kuru sagatavošana un izpilde prasa ļoti lielu resursu ieguldījumu, kā arī laiku. Kā zināms, integrāciju testēšana ir viena no beigu fāzēm, kur laiks un resursi ir ļoti būtiski, tādēļ autore nolemj, ka būtu iespējams testēšanu balstīt uz lietotāju scenārijiem, tādejādi izslēdzot tos gadījumus, kuri pēc lietotāju scenārijiem netiek izmantoti (bet sistēmā ir iespējami). Kā tiek aprakstīts 3.1 nodaļā, izmantojot lietotāju scenāriju modeli, testu skaits ir salīdzinoši neliels – vidēji pieci testa scenāriji uz katru aktivitāšu diagrammu. Jāatzīmē, ka tiek apskatīti tikai gadījumi, ja sistēma tiek lietota pareizi.

Autore uzskata, ka, balstoties uz lietotāju scenārijiem, tiktu uzlabota integrācijas testēšanas kvalitāte. Ļoti iespējams, ka palielinātos atklāto kļūdu skaits, un, galvenais, būtu pārliecība, ka sistēma strādā labi, jo tiktu pārbaudīti visi svarīgākie sistēmas lietošanas gadījumi. Jāpiemin, ka testa scenāriji, kas izstrādāti balstoties uz lietotāju scenāriju modeli, varētu kalpot par pamatu akcepttestēšanai.

3.3 Analīze, balstoties uz vēsturiskiem datiem konkrētas sistēmas ietvaros

Līdzīgi kā Edgara Diebeļa promocijas darbā "Programmatūras paštestēšana" [Diebelis, 2012], lai secinātu vai lietotāju scenāriju modeļa izmantošana spētu uzlabot integrācijas testēšanas kvalitāti (veiktu piedāvātās metodes efektivitātes mērījumus), autore veic analīzi, balstoties uz vēsturiskiem datiem (atklātajām kļūdām) konkrētā risinājuma ietvaros. Gadījumā, ja integrāciju testēšana tiktu balstīta uz lietotāju stāstiem, nebūtu iespējams noteikt metodes ietekmi uz piegādēm, jo nebūtu iespējas salīdzināt situāciju, kāda tā būtu, ja piedāvātā metode netiktu izmantota.

Analīzes ieejas dati ir piecu piegāžu (2016. un 2017. gads) atklātās un neatklātās (kuras vēlāk atklājās piegādājot IS klientam) kļūdas.

Autores mērķis ir veikt kvantitatīvu analīzi, kuras ietvaros tiktu secināts:

- Cik kļūdas no katras piegādes būtu iespējams atklāt ātrāk, izmantojot lietotāju scenāriju modeli;

- Cik neatklātās kļūdas no katras piegādes būtu atklātas, balstot integrāciju testēšanu uz lietotāju scenāriju modeli.

3.3.1 Kļūdu analīzes piemēri

Kā iepriekš tika minēts, autore veic analīzi, balstoties uz iepriekšējās piegādēs atklātām un neatklātām kļūdām.

Pielikumā Nr. 1 tiek attēlotas pirmās piegādes kļūdas. Pirmajā tabulā tiek attēlotas kļūdas, kuras atklātas konkrētās piegādes ietvaros (un novērstas), savukārt otrajā tabulā attēlotas kļūdas, kuras atklātas tikai pēc sistēmas piegādes.

Viens no neatklāto kļūdu piemēriem ir gadījums, kad sistēma neļauj izveidot tiešā maksājuma līgumu CRM sistēmā. Tiešā maksājuma līguma gadījumā, klientam nav jāveic manuāls maksājums – jāapmaksā rēķins klientu apkalpošanas centrā vai izmantojot internetbanku, bet rēķina maksa automātiski tiek piemērota un atskaitīta no klienta norādītā konta. Šīs kļūdas ietvaros, CRM sistēmā veidojot jaunu līgumu, kurā tiek norādīta maksimālā summa, līguma numurs, aktivācijas datums un beigu datums, līguma statuss visiem līgumiem automātiski nomainās un 'neaktīvs' (arī līgumiem, kuriem aktivācijas datums < šodiena un beigu datums > šodiena), un attiecīgi līguma detaļas netiek nosūtītas uz NS.

Izmantojot 3.8 attēlā redzamo aktivitāšu diagrammu – Klienta informācijas rediģēšana, skaidri redzams, ka šī lietotāju scenāriju ietvaros tiktu pārbaudīts, ka ir iespējams veikt izmaiņas klienta kontā – pievienot tiešo maksājuma līgumu, kā rezultātā CRM sistēma saņemtu informāciju, veiktu izmaiņas, nodotu informāciju NS, kas saņemtu informāciju un veiktu izmaiņas. Rezultātā tiktu atklāta kļūda CRM sistēmā, kas saistīta ar tiešā maksājuma līguma izveidi.

Zemāk attēlota tabula (3.1) ar testa scenāriju soļiem, kas izriet no 3.8 attēlā attēlotās aktivitāšu diagrammas. Autore atzīmē, ka šajā piemērā netiek pievērsta uzmanība tehniskām detaļām, bet idejiski attēloti testa scenāriju soļi.

3.1. tabula

Testa scenārijs, pārbaudot tiešā maksājuma līguma funkcionalitāti

Kārtas nr.	**Darbība**	**Sagaidāmie rezultāti**
1. solis	Meklē iepriekš izveidotu klientu CRM sistēmā	Klients tiek atrasts, korekti attēlota informācija
2. solis	CRM sistēmā veic izmaiņas klienta kontā – pievieno jaunu tiešā	Tiešā maksājuma līgums pievienots veiksmīgi.

	maksājuma līgumu (sākuma datums < šodiena; beigu datums > šodiena).	CRM sistēma piešķir statusu "aktīvs".
3. solis	Veic pārbaudi NS.	NS informācija ir atjaunota. Maksājuma tips ir nomainīts uz "tiešo maksājumu". Informācija par līgumu ir korekta.

Kā otru neatklātās kļūdas piemēru, autore atzīmē kļūdu, kuras ietvaros, lietojot pakalpojumu (veicot zvanu) un, sasniedzot iepriekš definētu limitu, klients ir spējīgs turpināt veikt zvanus.

3.6 attēlā redzama aktivitāšu diagramma par pakalpojumu lietošanas scenāriju, kurā var redzēt, ka gadījumā, kad tīkls nodrošina zvanu, iespējami divi varianti – klients pats veic zvanu, vai, gadījumā, ja sasniegts klienta definētais limits, SmsS sistēma nosūta brīdinājuma SMS, un PAS sistēma uzliek liegumu.

Gadījumā, ja testēšanā tiktu izmantoti lietotāju scenāriji (un pārbaudīts scenārijs par pakalpojumu lietošanu), tiktu atklāts, ka SmsS sistēma nosūta īsziņu klientam, bet nenosūta pieprasījumu PAS sistēmai uzlikt liegumu, līdz ar to klientam ir iespēja veikt maksas zvanus arī ja jau ir sasniegts iepriekš definētais limits (kā rezultātā rēķinā summa būs lielāka par norādīto maksimumu konkrētā pakalpojuma ietvaros).

Zemāk attēlota tabula (3.2) ar testa scenāriju soļiem, kas izriet no 3.6 attēlā attēlotās aktivitāšu diagrammas (piemērs).

3.2. tabula

Testa scenārijs, pārbaudot limita sasniegšanas funkcionalitāti, veicot zvanu

Kārtas nr.	**Darbība**	**Sagaidāmie rezultāti**
1. solis	Veic zvanus esošam klienta, kuram nav uzlikts ierobežojums veikt zvanu, nav sasniegts limits, pieejami dati.	Zvans veiksmīgi nodrošināts.
2. solis	Sasniedz iepriekš noteikto limitu.	Saņemta SMS par brīdinājumu. Zvans tiek pārtraukts.

3. solis	Mēģina vēlreiz veikt zvanu.	Nav iespējams veikt zvanu, jo uzlikts liegums.
4. solis	Mēģina sūtīt SMS (bezmaksas).	SMS veiksmīgi nosūtīts.
5. solis	Mēģina sūtīt SMS (maksas).	SMS nav nosūtīts (noraidīts tīkla pusē).
6. solis	Lieto datus.	Datu lietojums veiksmīgs.
7. solis	Beidz lietot datus.	Datu sesija pabeigta. Faili saģenerēti.
8. solis	Pārbauda failus, kas saņemti no tīkla NS.	Faili satur korektu informāciju par veiktajiem zvaniem, SMS, datu lietojumu.

Viena no kļūdām, kas vienā no iepriekšējām piegādēm ir tikusi atklāta, bet autore uzskata, ka, izmantojot lietotāju scenāriju modeli, šo kļūdu būtu iespējams atklāt vēl pirms koda piegādes, ir saistīta ar jauna pasūtījuma izveidi. Jaunas prasības ietvaros tiek pieprasīts papildināt sistēmu ar jaunu lauku, kurā tiks norādīts iekārtas numurs (IMEI).

Veicot prasību analīzi un veidojot pieprasījumu programmētājiem, sistēmas analītiķi veido jaunu dokumentu, kurā tiek pieprasīts pievienot jauno lauku tikai CRM sistēmai. Izmantojot 3.2 attēlā pieejamo aktivitāšu diagrammu, redzams, ka pasūtījums var tikt veidots gan CRM sistēmā, gan arī PVS. Integrāciju testētāji gatavojoties jaunajai piegādei (testa scenāriji tiek gatavoti vēl pirms koda piegādes) būtu iepazinušies ar analītiķu veidotu dokumentu, un, izmantojot lietotāju scenāriju modeli, būtu pamanījuši iespējamo kļūdu – IMEI lauka norādīšana iespējama tikai veidojot pasūtījumu pa tiešo CRM sistēmā, nav iespējams šo lauku norādīt PVS.

Rezultātā būtiski tiktu ietaupīts laiks, jo kļūda dokumentācijā būtu, pamanīta vēl, pirms programmētāji piegādājuši kodu (iespējams, pat uzsākuši programmēšanas darbus), dokumentācija tiktu izlabota, nodota arī programmētājiem, kas atbild par PVS kodu.

Zemāk attēlota tabula (3.3) ar testa scenāriju soļiem, kas izriet no 3.2 attēlā attēlotās aktivitāšu diagrammas (piemērs).

3.3. tabula

Testa scenārijs, pārbaudot pasūtījuma veidošanu no PVS

Kārtas nr.	Darbība	Sagaidāmie rezultāti
1. solis	Izmantojot CRM sistēmu, meklē neeksistējošu klientu.	Nekas netiek atrasts
2. solis	Izveido jaunu klientu.	Klients veiksmīgi reģistrēts CRM.
3. solis	Pārbauda, ka klienta dati veiksmīgi nodoti NS.	Klients veiksmīgi reģistrēts NS.
4. solis	Pārbauda klienta statusus CRM.	Klienta statuss ir korekts - "jauns"
5. solis	Izmantojot PVS, veido jaunu pasūtījumu.	Pasūtījums veiksmīgi izveidots PVS.
6. solis	Pārbauda pasūtījuma detaļas CRM.	CRM lauki automātiski aizpildīti, pasūtījuma detaļas ir korektas.
7. solis	Pārbauda pasūtījumu PAS.	Pasūtījums veiksmīgi iesniegts PAS un aktivizēts tīklā.
8. solis	Pārbauda pieslēgumu NS.	Klientam pieslēgts korektais pakalpojums NS.
9. solis	Pārbauda pasūtījumu CRM.	Pasūtījuma statuss – "izpildīts"

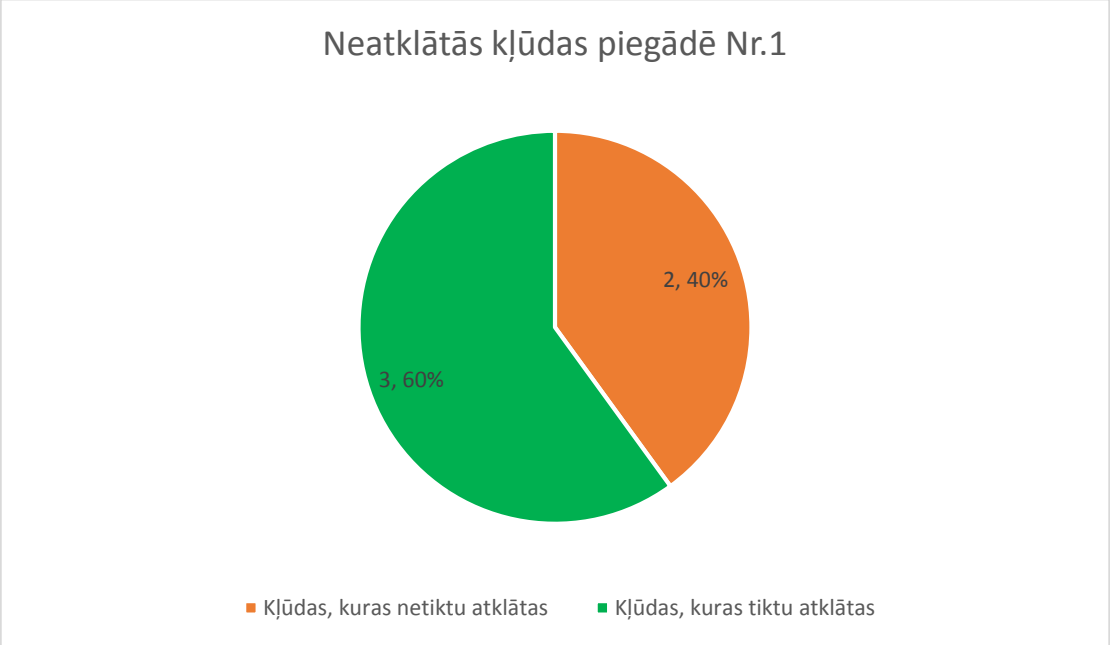
3.3.2 Kvantitatīvie dati

Līdzīgi kā 3.3 nodaļā aplūkotajos piemēros, autore veic analīzi kļūdām, kas atklātas un neatklātas iepriekšējās sistēmas piegādēs. Kopā tiek analizētas piecas sistēmas piegādes, katrai piegādei tiek analizēts gan neatklāto kļūdu saraksts (kļūdas atklātas tikai pēc ieviešanas produkcijā), gan atklāto kļūdu saraksts.

Apkopojot statistiku par kļūdām, kas figurē iepriekšējās piegādēs, autore secina, ka pirmajā piegādē tika atklātas 28 kļūdas, bet produkcijā tika ieviestas 5 neatklātas kļūdas,

otrajā piegādē tika atklātas 38 kļūdas un neatklātas 13 kļūdas, trešajā piegādē - 43 atklātas kļūdas un 7 neatklātas, ceturtajā - 49 atklātas un 5 neatklātas, savukārt piektajā piegādē - 25 atklātas un 9 neatklātas kļūdas.

Analizējot katru piegādi, tiek secināts, ka pirmajā piegādē visas kļūdas ir atklātas laicīgi, lietotāju scenāriju modeļa izmantošana šajā gadījumā nesniegtu ieguvumu. Tomēr, aplūkojot neatklātās kļūdas pirmajā piegādē, autore secina, ka, izmantojot lietotāju scenāriju modeli, trīs no piecām kļūdām būtu iespējams atklāt (sk. 3.9 att.).



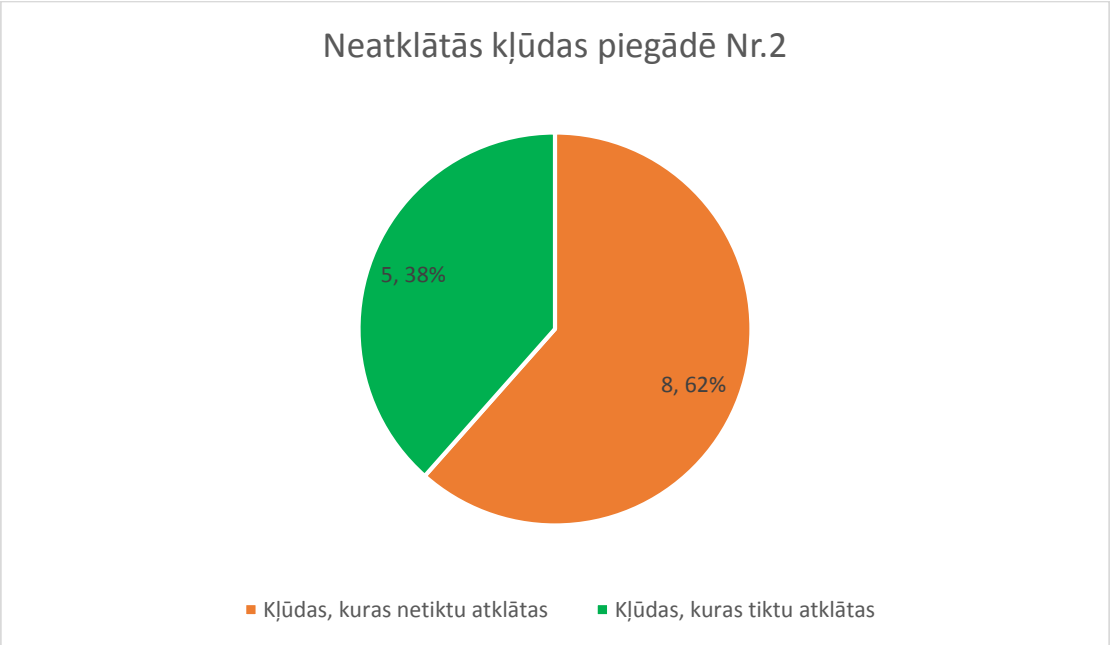
3.9. att. **Neatklāto kļūdu izmaiņas piegādē nr.1**

Aplūkojot otru piegādi, autore secina, ka vienu kļūdu būtu iespējams atklāt ātrāk, jo tā ir saistīta ar kļūdām analīzes procesā. Ja analīzes laikā tiktu aplūkots lietotāju scenāriju modelis, precīzāk, 3.3 att. attēlotā aktivitāšu diagramma, būtu iespējams pamanīt, ka pasūtījuma apstrāde, klientam atsakoties no pakalpojuma, tiek nodrošināta gan manuāli, darbiniekam veicot pasūtījumu no PVS, gan automātiski, cita operatora gadījumā. Rezultātā, analīzē tiktu iekļautas izmaiņas, kas jāveic sistēmā, ja pasūtījums ir automātiski ienācis sistēmā (jāpievieno papildus parametri).

Attēlā 3.10 redzams, ka uz kopējo atklāto kļūdu skaitu izmaiņas, izmantojot lietotāju scenāriju modeli, ir ļoti nelielas.

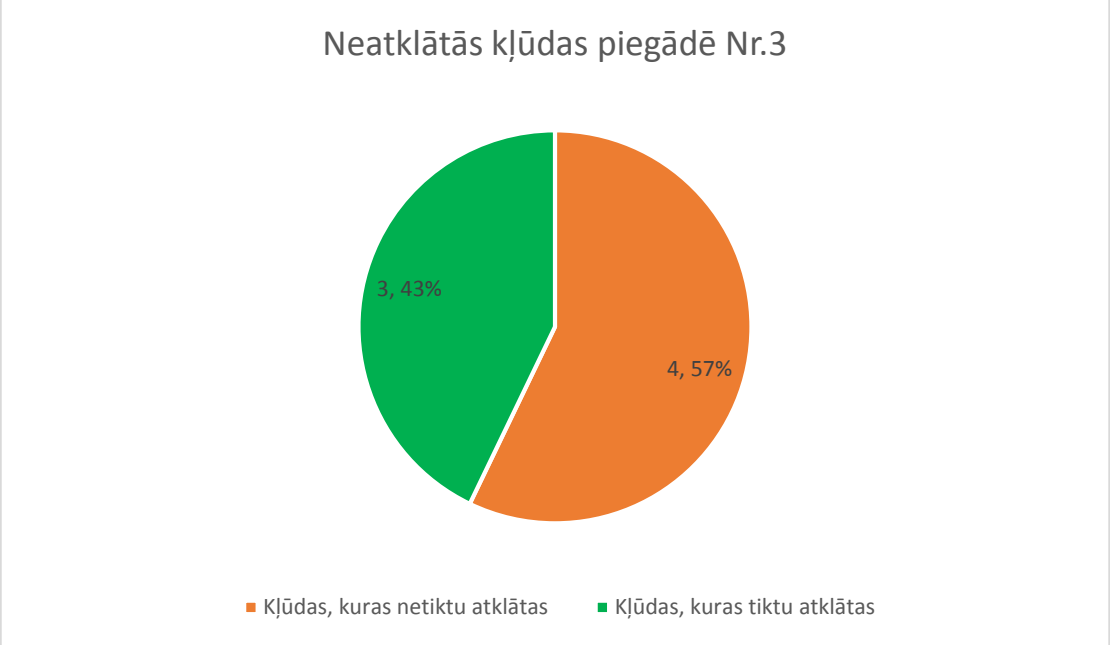
3.10. att. **Kļūdas, kuras būtu iespējams atklāt vēl pirms koda piegādes, izmaiņas 2. sistēmas piegādē**

Aplūkojot neatklātās kļūdas, autore secina, ka 5 no 13 kļūdām būtu iespējams atklāt. (kļūdas saistītas ar pasūtījuma izpildi PAS, klienta pašreizējās bilances aplūkošanu, jauna klienta reģistrāciju, statusa atjaunošanu PAS sistēmā un ierobežojumu iemesla noskaidrošanu).



3.11. att. **Neatklāto kļūdu izmaiņas piegādē nr.2**

Līdzīgi kā pirmajā piegādē, arī trešajā piegādē netiek pamanītas kļūdas, kuras būtu iespējams atklāt vēl pirms koda piegādes. Pēc trešās piegādes neatklāto kļūdu analīzes autore secina, ka trīs no septiņām kļūdām būtu iespējams atklāt (sk. 3.12 att.). Kļūdas saistītas ar tiešā maksājuma līguma pievienošanu, limita pārsniegšanu un statusu nomaiņu pasūtījuma laikā.

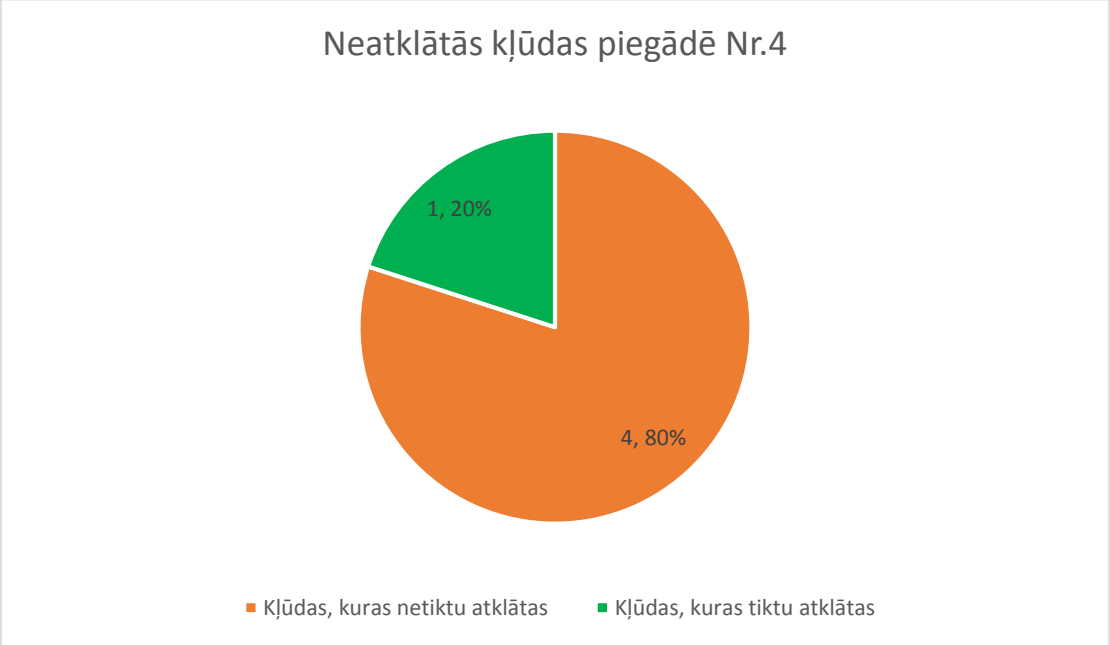


3.12. att. **Neatklāto kļūdu izmaiņas piegādē nr.3**

Analizējot ceturtajā piegādē atklātās kļūdas, vienu no kļūdām (sk. 3.13 att.) autore atzīmē, kā analīzes gaitā pieļautu, un secina, ka, izmantojot lietotāju scenāriju modeli, šo kļūdu būtu iespējams nepieļaut vai atklāt vēl pirms koda piegādes. Sīkāk kļūda aprakstīta 3.3.1 nodaļā.

3.13. att. **Neatklāto kļūdu izmaiņas piegādē nr.3**

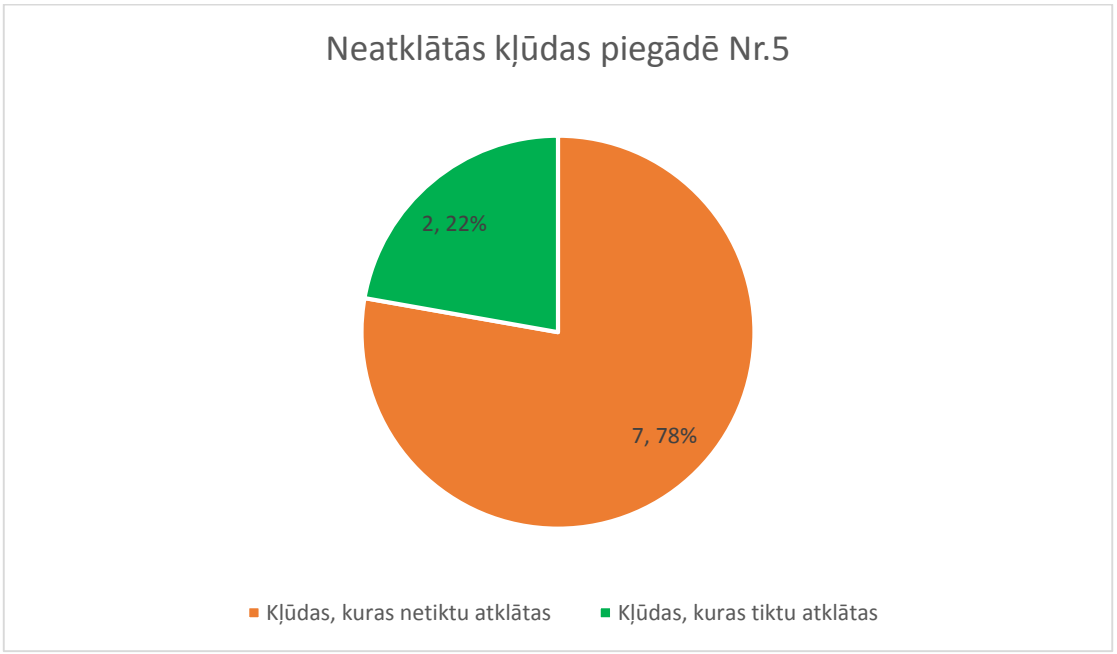
Ceturtajā piegādē viena no piecām neatklātajām kļūdām testēšanas laikā tiek atzīmēta kā kļūda, kuru būtu iespējams atklāt un novērst, izmantojot autores piedāvāto modeli.



3.14. att. **Neatklāto kļūdu izmaiņas piegādē nr.4**

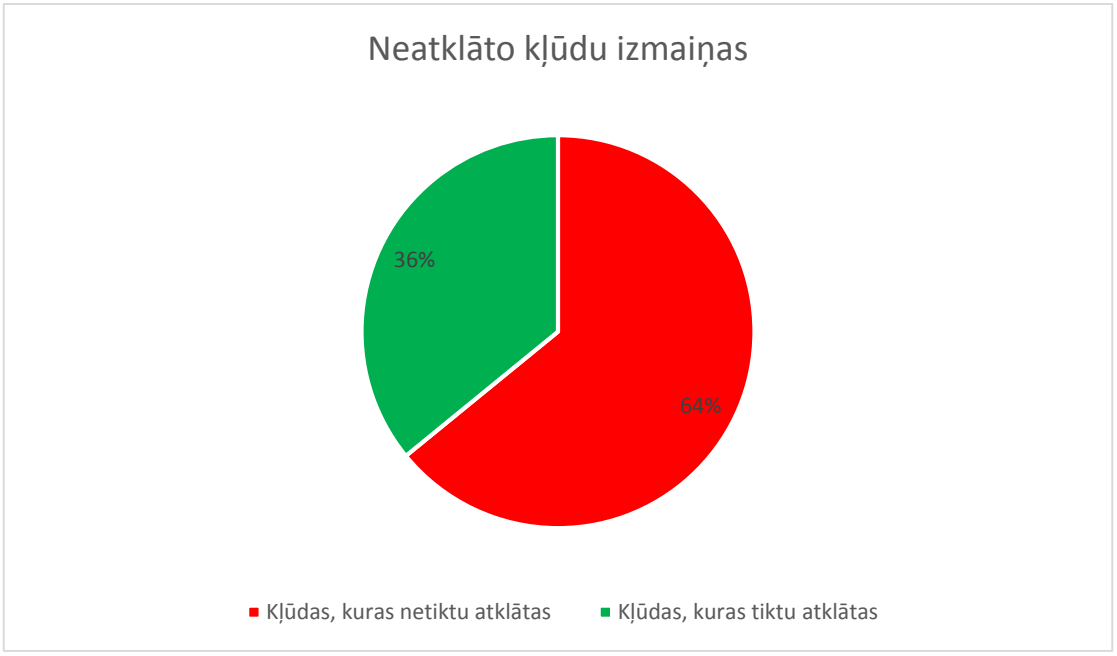
Aplūkojot pēdējā piegādē figurējošās kļūdas, autore veic secinājumu, ka šajā piegādē, izmantojot lietotāju scenāriju modeli, būtu iespējams samazināt neatklāto

kļūdu skaitu. Divas no deviņām kļūdām būtu iespējams atklāt, ja tiktu izmantots lietotāju scenāriju modelis.



3.15. att. **Neatklāto kļūdu izmaiņas piegādē nr.5**

Attēlā 3.16 uzskatāmi var redzēt, cik neatklātās kļūdas tiktu atklātas pēc autores veiktās analīzes, ja tiktu izmantots lietotāju scenāriju modelis. Kopā pa visām piecām piegādēm tika ieviestas 39 kļūdas, no kurām 14 būtu iespējams atklāt un novērst.



3.16. att. **Neatklāto kļūdu izmaiņas kopā pa piegādēm**

Pēc kļūdu analīzes autore secina, ka lietotāju scenāriju modelis diezgan būtiski uzlabotu gala piegādes rezultātu – 36% no visām neatklātajām kļūdām piecu piegāžu laikā tiktu atklātas.

Tāpat, autore secina, ka šīs kļūdas, kuras iepriekš netika atklātas, ļoti negatīvi iespaido patērētāju, jo lietotāju scenāriju modelī tiek aplūkotas svarīgākās sistēmas darbības.

Autore secina, ka lielākā daļa visu pārējo kļūdu, kuras nebūtu iespējams atklāt, izmantojot lietotāju scenāriju modeli, ir saistītas ar atsevišķu vērtību, produktu izmantošanu. Piemēram, viena no neatklātajām kļūdām bija saistīta ar pasūtījuma neveiksmīgu izpildi (sistēma PAS) gadījumā, ja tiek atslēgts konkrēts datu apjoma produkts. Autore uzskata, ka, manuāli veicot integrāciju testēšanu, nav iespējams pārbaudīt katru lauku, katru vērtību, katru produktu utml. Iespējams, izmantojot lietotāju scenāriju modeli un, automatizējot visus testa scenārijus, būtu iespējams pārbaudīt pilnībā visu sistēmu vai tuvoties tai.

Pārējās kļūdas, kuras nebūtu iespējams atklāt arī balstoties uz lietotāju stāstiem, ir saistītas ar dažādiem iemesliem, piemēram – kļūda, ka viena no sistēmām daļēji (PAS) neatbild; kļūda, ka tiek izsaukta nekorekta funkcija (no NS puses); iekšēja kļūda CRM sistēmā – nekorekts tulkojums utml.

Attēlā 3.17 autore attēlo atklāto kļūdu skaitu, kuru būtu iespējams atklāt ātrāk, ja integrāciju testēšana tiktu veikta, balstoties uz lietotāju scenāriju modeli.

3.17. att. **Neatklāto kļūdu izmaiņas kopā pa piegādēm**

Redzams, ka kļūdu skaits ir ļoti neliels – tikai divas no 183 kļūdām būtu iespējams atklāt vēl pirms koda piegādes, līdz ar to arī procentuāli skaits ir ļoti mazs.

Izanalizējot visas piegādēs atklātās kļūdas, autores secina, ka galvenais ieguvums no lietotāju scenāriju modeļa izmantošanas būtu saistīts ar kļūdām, kas pieļautas, veicot analīzi jaunām prasībām (abas kļūdas, ko autore novērtēja kā kļūdas, kuras būtu iespējams atklāt ātrāk, ir kļūdas, kas pieļautas veicot analīzi). Līdz ar to, lietotāju scenāriju modeli būtu ieteicams izmantot arhitektiem un analītiķiem, lai nepieļautu šāda veida kļūdas. Autore pieļauj, ka atklāto kļūdu skaits, kas būtu atklātas agrāk, izmantojot lietotāju scenāriju modeli, būtu lielāks, ja tiktu ņemts vērā, ka, izmantojot lietotāju scenāriju modeli, testa scenāriju apjoms ir samazinājies un testētājiem būtu iespējams ātrāk pārbaudīt galvenās sistēmas funkcijas. (Vairāk par laika patēriņa samazinājumu nākamajā nodaļā).

3.3.3 Optimizēti regresa testa scenāriji

Aplūkojot konkrētās sistēmas regresa testa scenārijus, autore secina, ka daļa no scenārijiem nav lietderīga – daļā no scenārijiem nav iesaistīta neviena integrācija (pārbaude notiek tikai vienas sistēmas ietvaros), līdz ar to tiek veikta komponenttestēšana integrācijas testēšanas vietā. Tāpat, daļa scenāriju pārklājas, un

daļu no scenārijiem būtu iespējams apvienot. Autore uzskata, ka lietotāju scenāriju modelis varētu būt lietderīgs, veicot integrācijas regresa testus.

Konkrētā risinājuma lietotāju scenāriju modeļa izstrādei tika patērētas aptuveni 25 stundas. Pašreiz katras piegādes ietvaros tiek pildīti integrācijas regresa testa scenāriji, kas vidēji aizņem 20 stundas. Saskaitot aptuveno prognozēto laiku, kas tiktu patērēts visiem 34 scenārijiem, kas izriet no lietotāju scenāriju modeļa, iegūstam 920 minūtes, kas būtu aptuveni 15,4 stundas. No šiem rādītājiem izriet, ka regresa testu izpildei nepieciešamais laiks tiktu samazināts par 4,6 stundām, līdz ar to pēc sešām piegādēm laika patēriņš, kas tika veltīts modeļa izstrādei būtu nosedzies. Jāatzīmē, ka šie rādītāji atbilst tikai šim konkrētam risinājumam, un mainīsies atkarībā no risinājumā iesaistītajām sistēmām, lietotāju scenārijiem, integrāciju skaitu.

REZULTĀTI

Maģistra darba ietvaros autore ir izpētījusi, izanalizējusi un apkopojusi informāciju par integrācijas testēšanas posmu. Pamatojoties uz šo informāciju, autore izstrādājusi vadlīnijas (2. nodaļa), kas saistītas ar testēšanas plāna kvalitāti, laika izmantošanu izstrādes fāzē, testa scenāriju izstrādi un lietotāju scenāriju modeļu izveidi, lai integrācijas testēšanas posms būtu efektīvāks un sniegtu lielāku guvumu.

Maģistra darba trešajā nodaļā ir veikts pētījums, izmantojot komplicētu IS risinājumu, kas sastāv no vairākām dažādām sistēmām (komponentēm) un tiek izmantots telekomunikāciju uzņēmumā. Pētījuma ietvaros ir izstrādāts lietotāju scenāriju modelis un veikta analīze, balstoties uz vēsturiski atklātām kļūdām (gan izstrādes, gan piegādes posmā), lai secinātu, kā mainītos rezultāti, ja tiktu izmantots lietotāju scenāriju modelis.

Rezultāti rāda, ka kopā pa visām piecām piegādēm tika ieviestas 39 kļūdas, no kurām 14 būtu iespējams atklāt un novērst, ja integrāciju testēšana tiktu balstīta uz lietotāju stāstiem. Tiek secināts, ka gandrīz visas pārējās kļūdas, kuras nebūtu iespējams atklāt, izmantojot lietotāju scenāriju modeli, ir saistītas ar atsevišķu vērtību, produktu izmantošanu.

Pēc rezultātu apkopošanas, tiek secināts, ka salīdzinoši neliels ir kļūdu skaits, ko būtu iespējams atklāt ātrāk, ja integrāciju testēšana tiktu veikta, balstoties uz lietotāju scenāriju modeli - tikai divas no 183 kļūdām būtu iespējams atklāt vēl pirms koda piegādes. Tiek secināts, ka galvenais ieguvums no lietotāju scenāriju modeļa izmantošanas būtu saistīts ar kļūdām, kas pieļautas, veicot analīzi jaunām prasībām. Tāpat tiek secināts, ka lietotāju scenāriju modeļa izmantošana uzlabotu rezultātus ne tikai testēšanas posmā, bet arī sistēmas izstrādē.

Pēc darbā veiktās analīzes, tiek secināts, ka, izmantojot piedāvāto metodi, būtu iespējams arī samazināt regresa testu izpildei nepieciešamo laiku.

SECINĀJUMI

Maģistra darbā izvirzītais mērķis ir sasniegts. Ir izstrādātas vadlīnijas, kuru mērķis ir uzlabot integrācijas testēšanas efektivitāti.

Darbā plānotie uzdevumi ir izpildīti. Ir izpētīta literatūra par integrācijas testēšanas pieejām, posmiem un procesiem un, uz šo informāciju balstoties, izstrādātas vadlīnijas. Ir veikts pētījums – izstrādāts lietotāju scenāriju modelis un veikta analīze, balstoties uz vēsturiskiem datiem, lai secinātu, kā mainītos rezultāti, ja testēšana tiktu balstīta uz lietotāju stāstiem.

Analīzes posmā tiek secināts, ka autores piedāvātās metodes izmantošana, uzlabotu integrāciju testēšanas efektivitāti risinājumam, uz kuru tika veikts pētījums. Līdz ar to tiek secināts, ka šī darba rezultāti un lietotāju scenāriju modelis ir izmantojams konkrētā risinājuma integrāciju testēšanā.

Autore uzskata, ka šādas pieejas izmantošana varētu uzlabot integrāciju testēšanu arī citu risinājumu ietvaros, pirms tam veicot analīzi lietotāju stāstiem un ņemot vērā risinājumā iesaistīto sistēmas, sistēmu un integrāciju skaitu. Līdz ar to, tiek secināts, ka darbs būtu noderīgs arī citu risinājumu integrāciju testēšanas plānošanā.

IZMANTOTĀ LITERATŪRA UN AVOTI

[Diebelis, 2012] Edgars Diebelis "Programmatūras paštestēšana", promocijas darbs doktora disertācijai, LU Datorikas fakultātes, Latvijas Universitāte, 2012., pieejams http://dspace.lu.lv/dspace/handle/7/4708

[ISTQB, 2017] ISTQB Certificationstudy material, pieejams http://istqbexamcertification.com/what-is-integration-testing/ skatīts 08.02.2017

[Ould, Unwin, 1988] Martyn A. Ould, Charles Unwin, *Testing in Software Development*, Cambridge, USA: Press Syndicate od the Unicersity of Cambridge, 1988., pieejams, https://books.google.lv/books?id=utFCImZOTEIC&pg=PA73&dq=integration+test&hl=en&sa=X&ei=4EpTVOvJMayu7Aak5YCIDA&redir_esc=y#v=onepage&q=integration%20test&f=false

[Pressman, 2000] Pressman R. S., *Software Engineering - A Practioner's approach 5th edition*, The McGraw-Hill Companies Inc., 2000.

[Roy, 2017] Shilpa C. Roy "What Is Integration Testing", 2017., pieejams http://www.softwaretestinghelp.com/what-is-integration-testing/

[Software Testing Fundamentals, 2017] Software Testing Fundamentals, pieejams http://softwaretestingfundamentals.com/integration-testing/, skatīts 08.02.2017

[Spillner et. Al., 2014] Andreas Spillner, Tilo Linz, Hans Schaefer, *Software Testing Foundations*, Rocky Nook, 2014., (35-59 lpp)

[Waters, 2008] Kelly Waters "User Stories Should Be *Testable*", 2008., pieejams http://www.allaboutagile.com/user-stories-should-be-testable/

[Webster's Dictionary, 2017] Webster's Dictionary, pieejams http://www.webster-dictionary.org/definition/integration%20testing, skatīts 08.02.2017

PIELIKUMI

1. pielikums. 1. piegādes kļūdas

Kļūdas nosaukums / īss apraksts	**Iespējams atklāt ātrāk?**
Kļūdas paziņojums tīklā "Neizdevās iesniegt pasūtījumu sistēmā"	Nē
Validācijas kļūda, veicot izmaiņas klienta kontā	Nē
Problēmas ar klienta adreses atjaunināšanu	Nē
Pašreizējās bilances pieprasīšana, neatgriež sagaidāmās vērtības	Nē
Notikumu kodi vēsturiskām darbībām nav korekti	Nē
Pieslēgumu skatā liekas opcijas	Nē
Īsziņa nav nosūtīta, statuss - gaidīšanā	Nē
Iekšēja sistēmas kļūda, pieprasot pašreizējo bilanci	Nē
PAS tukši lauki, aplūkojot pasūtījuma detaļas	Nē
Parametri netiek nosūtīti NS, pieslēdzot datu paketi	Nē
Neveiksmīga rēķina izveide: Produkts neeksistē NS	Nē
Pasūtījums neveiksmīgs: kļūda – trūkst klienta datu	Nē
Kļūda – nekorekts parametrs – atbilde, pieprasot pašreizējo bilanci	Nē
Pasūtījums neveiksmīgs ar konkrētu operācijas tipu: “pievienot dalībnieku”	Nē
Validācijas kļūda: Nepilnīgs saturs	Nē
Trūkstoša komponente ārvalstu datu paketes pieslēgšanai	Nē
Datu paketes bilance netiek atgriezta	Nē
Validācijas kļūda ārzemju datu paketēm	Nē
Kļūda, pieprasot soda naudu, līguma laušanas gadījumā	Nē
Interfeisa izlaišana, pieslēdzot ārzemju datu paketi	Nē
Tīklam netiek nosūtīts lauks – beigu_datums	Nē
Notikuma izveides kļūda – lauks neatbilst!	Nē
Apakškomponentēs tiek attēlotas nekorektas vērtības	Nē
Nav iespējams veikt meklēšanu pēc tel. nr.	Nē
Pieprasījums neveiksmīgs – sistēmas kļūda!	Nē
Kļūda, pārtērējot limitu	Nē
Klienta limits netiek veiksmīgi palielināts	Nē
NS ģenerē ierakstus dubultā, pārsniedzot limitu	Nē

Kļūdas nosaukums / īss apraksts	**Būtu iespējams atklāt?**
Neveiksmīga pakalpojuma atslēgšana	Jā
Kļūda mainot rēķina saņemšanas veidu	Jā
Pieslēdzot produktu “korekcija”, netiek sūtīt primārās vērtības lauks	Nē
Vērtība netiek nosūtīta pakalpojuma modificēšanas gadījumā	Jā
Atlaide netiek piemērota, ja pieslēgta atlaide uz konkrēto pieslēgumu	Nē

DOKUMENTĀRĀ LAPA

Maģistra darbs "Integrācijas testēšanas efektivitātes uzlabošanas vadlīniju izstrāde" izstrādāts LU Datorikas fakultātē.

Darba teksta galīgā versija izgatavota 13.05.2017

Ar savu parakstu apliecinu, ka pētījums veikts patstāvīgi, izmantoti tikai tajā norādītie informācijas avoti un iesniegtā darba elektroniskā kopija atbilst izdrukai.

Autors: _____

(Autora paraksts un datums)

Ar savu parakstu apliecinu, ka esmu lasījis augstāk minēto maģistra darbu un atzīstu to par **p i e m ē r o t u / n e p i e m ē r o t u** (nevajadzīgo svītrot) aizstāvēšanai Latvijas Universitātes datorzinātņu maģistrantūrā.

Darba vadītājs: _____

(Vadītāja paraksts un datums)

Darbs iesniegts **maģistratūras sekretariātā** _____.

(Iesniegšanas datums)

Ar šo es apliecinu, ka darba elektroniskā versija ir augšupielādēta LU informatīvajā sistēmā.

Studiju metodiķe: _____.

(Metodiķes paraksts)

Recenzents: _____

(Akad.amats, zin.grāds, vārds, uzvārds)

Darbs aizstāvēts maģistra gala pārbaudījuma komisijas sēdē

_____ prot. Nr. _____

(Darba aizstāvēšanas datums)

Komisijas sekretārs: _____

(Sekretāra paraksts)