LATVIJAS UNIVERSITĀTE
DATORIKAS FAKULTĀTE

AUTOMATIZĒTA DOKUMENTĀCIJAS ĢENERĒŠANA „ISTEHNOLOĢIJAI”

BAKALAURA DARBS

Autors: Valdis Vizulis
Stud. apl. vv05002
Vadītājs: M. Dat. Jānis Iljins

RĪGA 2009

ANOTĀCIJA

Kopš 1995. gada bankās tiek lietots informatīvu sistēmu projektēšanas, izstrādes un uzturēšanas rīks „ISTehnoloģija", kas balstās uz metamodelī definēta sistēmas projektējuma interpretēšanu.

Bakalaura darba ietvaros izstrādāts risinājums, kā iespējams sasaistīt divas uz metamodeļa balstītas sistēmas – „ISTehnoloģiju" un atskaišu definēšanas un ģenerēšanas rīku „DIREPO" –, lai iegūtu automatizēti ģenerētu sistēmas dokumentāciju (tai skaitā datu bāzu projektējumu, lietotāja rokasgrāmatu u.c.) ar metamodeļu transformāciju palīdzību.

Bakalaura darbā ir apskatīti dokumentāciju ģeneratoru veidi, modeļu bāzēta arhitektūra, LVS standartā specificētie programmatūras projektu dokumenti un to nodaļas, kā arī novērtēta to ģenerēšanas iespējamība. Darba beigās apkopota pieredze un secinājumi, kas radās automatizēti ģenerējot dokumentāciju.

Atslēgvārdi

Sistēmas dokumentācija, metamodeļu bāzēta arhitektūra, dokumentāciju ģenerēšana.

ABSTRACT

Since 1995 the tool of designing, development and maintenance of informative system named „ISTehnologija", which is based on interpretation of system design defined in metamodel, is being used by banks.

In order to get automated system-generated documentation (including data base design, user manual, etc.), within the bachelor's work was developed a solution, how can be linked two systems based on metamodels – "ISTehnologija" and report defining and generation tool "DIREPO" – with the assistance of metamodels transformation.

Bachelor's work is looking at the types of documentation generators, model driven architecture, software project documents and sections specified by standard of LVS, as well as estimated the possibility to generate that. At the end of work summarized experience and conclusions that originated by automatically generating documentation.

Keywords

System documentation, model driven architecture, generation of documentation.

АННОТАЦИЯ

С 1995 года в банках используют средство проектирования, разработки и поддержки информативных систем "ISTehnologija", которое основывается на интерпретацию, проектированную на метамодели систему.

В пределах работы бакалавра разработано решение, как возможно связать две на метамодели основанные системы "ISTehnologija" и определения отчётов и приборов генерирования "DIREPO ", чтобы получить автоматическую систему генерирования документации (в том числе, описание базы данных, руководство пользователя и др.) с помощью трансформации метамодели.

В работе бакалавра рассмотрены виды документации генераторов, архитектура, основанная на модели, специфицированные в стандарте ЛГС документы проектов и их разделы, а также дана оценка возможности их генерирования. В конце обобщены опыты и выводы, которые возникли, автоматически генерируя документацию.

Ключевые слова

Системная документация, архитектура, основанная на метамодели, генерирования документации.

SATURA RĀDĪTĀJS

LIETOTIE APZĪMĒJUMI UN TERMINI

Apzīmējumi

IT – Informācijas tehnoloģijas;

PLD – programmatūras lietotāja dokumentācija;

PPS – programmatūras prasību specifikācija;

PTD – programmatūras testēšanas dokumentācija;

PPA – programmatūras projektējuma apraksts;

PVVP - programmatūras verifikācijas un validācijas plāns;

PKNP – programmatūras kvalitātes nodrošināšanas plāns;

MBA – modeļu bāzēta arhitektūra (angl. Model Driven Architecture).

Termini

Reversā inženierija (angl. Reverse engineering) – zema līmeņa informācijas, kas parasti saprotama tikai datoriem, pārveidošana augstāka līmeņa formātā, kas ir viegli uztverams cilvēkam [5.7.].

Dokumentācija - instrukcijas, specifikācijas, darbības apraksti, rokasgrāmatas un citi dokumenti, kas parasti ietilpst programmatūras vai aparatūras komplektā. Dokumentācija var būt arī elektroniskā formā [5.4.].

IEVADS

Jau kopš 1995. gada tiek lietota informatīvu sistēmu izstrādes un uzturēšanas metode un rīks „ISTehnoloģija". To izveidoja SIA „Datorikas institūts DIVI" un jau vairāk kā desmit gadus tā tiek veiksmīgi lietota banku un citu finanšu institūciju sistēmu izveidē. Sistēmas pamats ir bāzēts uz metamodeļa. Tai ir sava metadatu datu bāze, kas satur informāciju par sistēmas funkcionalitāti, kuru spēj „ISTehnoloģija" interpretēt. Šāda pieeja būtiski atvieglo jaunas funkcionalitātes pievienošanu sistēmai, kā arī izmaiņu ieviešanu sistēmas funkcionalitātē. Rezultātā būtiski tiek ietaupīts sistēmas attīstībai un uzturēšanai veltītais laiks. Detalizētāk par „ISTehnoloģiju" aprakstīts darba 3. nodaļā „„ISTehnoloģijas" apskats".

Mūsdienās programmatūras projekti nav iedomājami bez dokumentācijas, kas palīdz nodrošināt programmatūras projektu pārvaldību, kā arī programmatūras kvalitāti. Tādēļ arī uzņēmumā SIA „Datorikas institūts DIVI" programmatūras projektos tiek veidota un izmantota dažāda dokumentācija. Parasti šīs dokumentācijas veidošana ir diezgan laikietilpīgs process, kuru uzņēmuma darbinieki veic nelabprāt. Nereti retāk izmantojama dokumentācija tiek atstāta novārtā un sen jau vairs neatbilst aprakstāmajai programmatūrai. Tādēļ, lai ietaupītu uzņēmuma resursus, kuri nepieciešami dokumentācijas izstrādāšanai, tika nolemts izstrādāt rīku, kas spētu automatizēti uzģenerēt dokumentāciju „ISTehnoloģijai".

Lai šo uzdevumu varētu paveikt, vispirms tika apskatīti pašlaik pasaulē izmantoto dokumentāciju ģeneratoru veidi. Ņemot vērā, ka „ISTehnoloģija" balstīta uz metamodeļa, no kura iespējams iegūt vispusīgu informāciju par sistēmu, tika nolemts to izmantot dokumentācijas ģenerēšanā. Šāds dokumentācijas ģenerēšanas veids pasaulē ir ne visai plaši izplatīts un saucas „dokumentācijas ģenerēšana no sistēmas". Detalizētāk par dokumentāciju ģeneratoriem var iepazīties darba 1. nodaļā „Dokumentāciju ģeneratori".

Tālāk tika apskatīts, kādas dokumentācijas tiek pielietotas Latvijā un kuras no tām specificē LVS standarts, kā arī novērtēts, kādu dokumentāciju un kādas nodaļas iespējams automatizēti uzģenerēt no „ISTehnoloģijā" pieejamās informācijas. Tas parakstīts darba 4., 5. un 6. nodaļā.

„ISTehnoloģijas" sistēmās pastāv atskaišu ģenerēšanas rīks „DIREPO". Šis rīks līdzīgi kā pati „ISTehnoloģija" ir balstīts uz metamodeļa. Metamodelī tiek aprakstīta katra nepieciešamā atskaite un rīks, interpretējot metamodeli, atskaiti izveido.

Radās doma idejas dokumentācijas ģenerēšanai smelties no pasaulē plaši pazīstamās sistēmu pieejas MBA (detalizētāk darba 2. nodaļā), kas sistēmu izstrādes procesā izmanto

metamodeļus un transformācijas starp tiem. Līdzīgi kā MBA platformas neatkarīgo modeli transformējot uz platformas atkarīgo modeli iegūstam sistēmas realizāciju, „ISTehnoloģija" var kalpot kā informācijas avota modelis, bet atskaišu definēšanas un ģenerēšanas rīks „DIREPO" var tikt izmantots kā dokumenta metamodelis, rezultātā iegūstot sistēmas dokumentāciju. Tādējādi darba mērķis ir, nedaudz pielāgojot šos abus modeļus, izstrādāt transformāciju starp šiem modeļiem tā, lai rezultātā tiktu iegūta pēc iespējas lasāmāka dokumentācija (detalizētāk darba 7. nodaļā).

Rezultātā tika izstrādāta transformācija starp abiem metamodeļiem, kā arī izveidotas „DIREPO" atskaišu definīcijas trīs dokumentu (datu bāzu projektējums, lietotāja rokasgrāmata un nododamo vienumu saraksts) automatizētai ģenerēšanai. Dokumentus neizdevās uzģenerēt pilnībā, bet toties izdevās uzģenerēt to pamatnodaļas, kas sastāda lielāko daļu dokumenta (skat. 8. nodaļu).

1. DOKUMENTĀCIJU ĢENERATORI

Parasti, ja tiek runāts par dokumentāciju ģeneratoru, ar to tiek saprasts, ka runa ir par rīku, kas ģenerē dokumentāciju no speciāli komentēta koda. Bet mūsdienās tas nav vienīgais veids, kā tiek ģenerēta dokumentācija.

Dokumentācijas ģenerators šī darba ietvaros tiek pieminēts kā rīks, kas, izmantojot noteikta veida informācijas avotu, spēj automatizēti daļēji vai pilnībā uzģenerēt dokumentāciju. Pēc informācijas iegūšanas avotiem tos var iedalīt trīs dažādos veidos:

- ✓ no speciāli komentēta koda;
- ✓ no programmatūras koda;
- ✓ no sistēmas.

Ar visiem trim veidiem ir iespējams iegūt dažāda veida dokumentāciju, kas domāta programmētājiem, projektu vadītājiem, kā arī pasūtītājiem un sistēmas lietotājiem.

1.1. No speciāli komentēta koda

Šī veida dokumentācijas ģenerators ģenerē dažāda veida dokumentāciju no speciāli komentēta programmatūras koda, t.i., programmētājs, rakstot programmu, katrai klasei, funkcijai, metodei utt. nodefinē nepieciešamo informāciju attiecīgā programmas koda elementa aprakstīšanai, piem., padodamos parametrus, funkcijas tipu, aprakstu utt. Pēc tam dokumentācijas ģenerators meklē speciāli komentētos pirmkoda blokus un pārveido informāciju cilvēkam saprotamā formātā. Parasti šie ģeneratori neģenerē augsta līmeņa dokumentācijas, piem., programmatūras projektu dokumentācijas, bet gan tādas zemāka līmeņa dokumentācijas, kā lietojumprogrammas saskarnes dokumentācija. Šīs dokumentācijas ir saprotamas tikai IT nozares speciālistiem, jo satur pilnu programmatūras funkciju specifikāciju, kas nav saprotama un aktuāla parastiem sistēmas lietotājiem.

Šie ģeneratori atšķiras arī pēc programmēšanas valodu atbalsta iespējām un izejas formātiem, kādos tiek ģenerēta dokumentācija. Mūsdienās šādi dokumentāciju ģeneratori atbalsta visas plašāk izmantotās programmēšanas valodas, un arī izejas formāti ir ļoti dažādi un atkarīgi no attiecīgā rīka un operētājsistēmas, uz kuras tas darbojas [5.3.].

1.2.No programmatūras koda

Mūsdienās pastāv arī tāds dokumentācijas ģenerēšanas veids kā dokumentācijas ģenerēšana no koda jeb reversā inženierija. Reversā inženierija būtībā ir informācijas transformēšana no datoru valodas uz cilvēku valodu. Parasti reversās inženierijas rīki piedāvā

koda analīzi, piemēram, uzģenerējot programmatūras projekta datņu hierarhijas diagrammas, klašu izsaukumus, dažādas UML diagrammas, kas attēlo programmatūras koda struktūru utt. [5.8.]. Tāpat ir arī dokumentācijas ģenerēšana no programmatūras koda, kuras rezultātā no datoram saprotamās informācijas tiek uzģenerēts dokuments, kas ir viegli lasāms cilvēkam. Parasti tās ir dažādas diagrammas, kuras tiek iekļautas noteiktā vietā iepriekš sagatavotam dokumentam.

1.3.No sistēmas

Dokumentācijas ģenerēšana no sistēmas ir salīdzinoši maz izplatīts dokumentācijas ģenerēšanas veids. Šāda veida ģeneratori dokumentāciju ģenerē no sistēmā uzglabātās informācijas. Informācija tiek iegūta jau no sistēmā esošās informācijas, vai arī tā tiek attiecīgi pievienota sistēmai, t.i., katram sistēmas vienumam pievieno aprakstošu informāciju, kuru pēc tam ģenerators izmanto ģenerējot dokumentāciju.

1.4.Salīdzinājums

Katram no iepriekš minētajiem dokumentāciju ģeneratoru veidiem ir savi plusi un mīnusi, katrs ir pielāgojams attiecīgajai situācijai.

Dokumentāciju ģeneratori, kuri kā informācijas avotu izmanto speciāli komentētu kodu, parasti spēj uzģenerēt tikai zema līmeņa dokumentāciju, kas saprotama tikai nozares speciālistiem. Lai no programmatūras koda komentāriem uzģenerētā dokumentācija būtu saistoša parastam sistēmas lietotājam, programmētājam pirmkodā ir jāraksta arī informācija, ko parasti iekļauj augsta līmeņa dokumentācijā. Tas būtu ļoti neefektīvi, jo tādā veidā pasliktinātos koda pārskatāmība un kods būtu tikai daļa no lielā apjoma komentāriem, kas būtu veltīti dokumentācijai. Šādi būtu arī apgrūtināts informācijas izmaiņu veikšanas process, jo tad visas izmaiņas, kas saistītas ar dokumentāciju, būtu jāveic programmas pirmkodā. Tas pazeminātu arī darba efektivitāti, jo tas palielinātu varbūtību, ka vairākiem cilvēkiem būs nepieciešamība strādāt ar vienu un to pašu koda datni.

Tāda pati situācija ir arī ar dokumentāciju ģeneratoriem, kuri dokumentāciju ģenerē no programmatūras koda. Rezultātā iegūstamā dokumentācija ir zema līmeņa un saprotama tikai nozares speciālistiem. Parasti šī dokumentācija tiek izmantota analīzei. Atšķirībā no dokumentāciju ģeneratoriem, kuri ģenerē dokumentāciju no speciāli komentēta koda, šī veida dokumentāciju ģeneratori pārsvarā ģenerē dažādas diagrammas un statistiku, nevis tekstu.

Savukārt, izmantojot dokumentāciju ģeneratoru, kas ģenerē dokumentāciju no sistēmas, iespējams iegūt arī augsta līmeņa dokumentāciju, piem., lietotāju rokasgrāmatu, kas ir

saprotama arī parastam cilvēkam. Rezultātā iegūstamā dokumentācija, salīdzinot ar citiem dokumentāciju ģeneratoru veidiem, ir vairāk orientēta uz lietotājiem, jo sistēmā tiek glabāta informācija nevis par koda funkcijām, bet par sistēmas funkcijām. Šāda veida dokumentācijas veidošana ir arī ērtāka no lietotāju viedokļa, jo sistēma nodrošina saskarni, kur ievadīt dokumentācijai nepieciešamo informāciju. Lielākais mīnuss šāda veida dokumentācijas ģeneratoriem ir tāds, ka tas ir piesaistīts tikai viena veida vai pat vienai sistēmai. Lai ģenerētu šāda veida dokumentāciju, nepieciešama universāla un modulāra sistēma. Praktiski nav iespējams izveidot tādu dokumentācijas ģeneratoru, kas spētu ģenerēt dokumentāciju no dažādām sistēmām. Tāpēc pārsvarā šādi dokumentācijas ģeneratori nav publiski pieejami, bet tiek izstrādāti uzņēmumu iekšienē savu vajadzību nodrošināšanai.

Gan dokumentāciju ģeneratoriem, kas ģenerē dokumentāciju no speciāli komentēta koda, gan arī tiem, kas ģenerē dokumentāciju no sistēmas, ir nepieciešams pievienot specializētu informāciju, lai šo procesu varētu veikt. Lai uzģenerētu dokumentāciju no koda, nav nepieciešama papildus informācija, vienīgi programmas kods, kuru ģenerators izanalizē un attēlo diagrammu un statistikas veidā.

1.5.Secinājumi

Neatkarīgi no dokumentāciju ģeneratora veida tie visi ietaupa ļoti lielu laiku, kuru parasti nepieciešams patērēt programmatūras dokumentāciju izstrādāšanai un atjaunošanai. Lietotājam ir iespēja automātiski atjaunot izmaiņas dokumentācijā, kas veiktas sistēmā vai tās kodā. Visi trīs dokumentāciju ģeneratoru veidi kopā būtiski atvieglo programmatūras projektos izmantojamo dokumentāciju veidošanu, jo ar to palīdzību iespējams uzģenerēt gan zema līmeņa dokumentāciju, gan arī augsta līmeņu dokumentāciju.

Ņemot vērā to, ka dokumentāciju ģeneratori, kas ģenerē dokumentāciju no sistēmas, ir samērā ļoti maz izplatīti, tad šī ir tā joma, kurā iespējama dziļāka izpēte. Tāpēc šī darba ietvaros tiks apskatīts tieši šis dokumentācijas ģenerēšanas veids un pielietots uzņēmumā izstrādātajā sistēmā „ISTehnoloģija" (detalizētāku aprakstu skatīt nodaļā „„ISTehnoloģijas" apskats").

„ISTehnoloģija" ir jauns rīks, kas balstīts uz metamodeļiem, tāpēc automatizēta dokumentācijas ģeneratora izstrādāšanai tiks pielietota modeļu bāzēta arhitektūra, kuras pamatprincipi aprakstīti nākošajā nodaļā.

2. MODEĻU BĀZĒTA ARHITEKTŪRA

Pašlaik pasaulē popularitāti ir ieguvis jauns paņēmiens, kā veidot informatīvās sistēmas, kas balstās uz modeļiem un to transformācijām, – modeļu bāzēta arhitektūra. Arvien vairāk programmatūras izstrādes procesā tiek izmantota šī arhitektūra [5.2].

Modeļu bāzēta arhitektūra MBA (angl. Model Driven Architecture MDA) ir uzņēmuma *Object Management Group* (OMG) izstrādāts programmatūras izstrādes ietvars. MBA svarīgākā īpašība ir biznesa un lietojumprogrammas loģikas atdalīšana no realizācijas tehnoloģijas [5.6.].

MBA sastāv no šādiem galvenajiem moduļiem [1.1.]:

1) **platformas neatkarīgs modelis PNM (angl. Platform Independent Model PIM)**

   Šis modelis ir ar ļoti augstu abstrakcijas līmeni un pilnībā neatkarīgs no izstrādes tehnoloģijas. Tas apraksta sistēmas projektējumu un tiek veidots, raugoties no biznesprasību skatu punkta. Šajā modelī nav nozīmes, kādā programmēšanas valodā tiks izstrādāta sistēma vai uz kādas platformas tā darbosies. Nav svarīgi, kā tiks realizēta sistēma, bet gan, ko darīs sistēma.

2) **platformas specifisks modelis PSM (angl. Paltform Specific Model PSM)**

   MBA programmatūras izstrādes ciklā nākošais solis ir PNM transformēšana vienā vai vairākos platformas specifiskos modeļos (PSM). Katrs PSM apraksta vienu konkrētu sistēmas realizēšanas tehnoloģiju. Tas satur konkrētai platformai specifiskus terminus, piemēram, PSM relāciju datubāzes kontekstā iekļauj tādus terminus kā „tabula", „kolonna", „shēma" utt. Tikai tiem izstrādātājiem, kuriem ir zināšanas par attiecīgo platformu, ir skaidra PSM uzbūve. Katrs PNM var tikt transformēts uz vienu vai vairākiem PSM, t.i., katrai specifiskai tehnoloģijas platformai.

3) **kods**

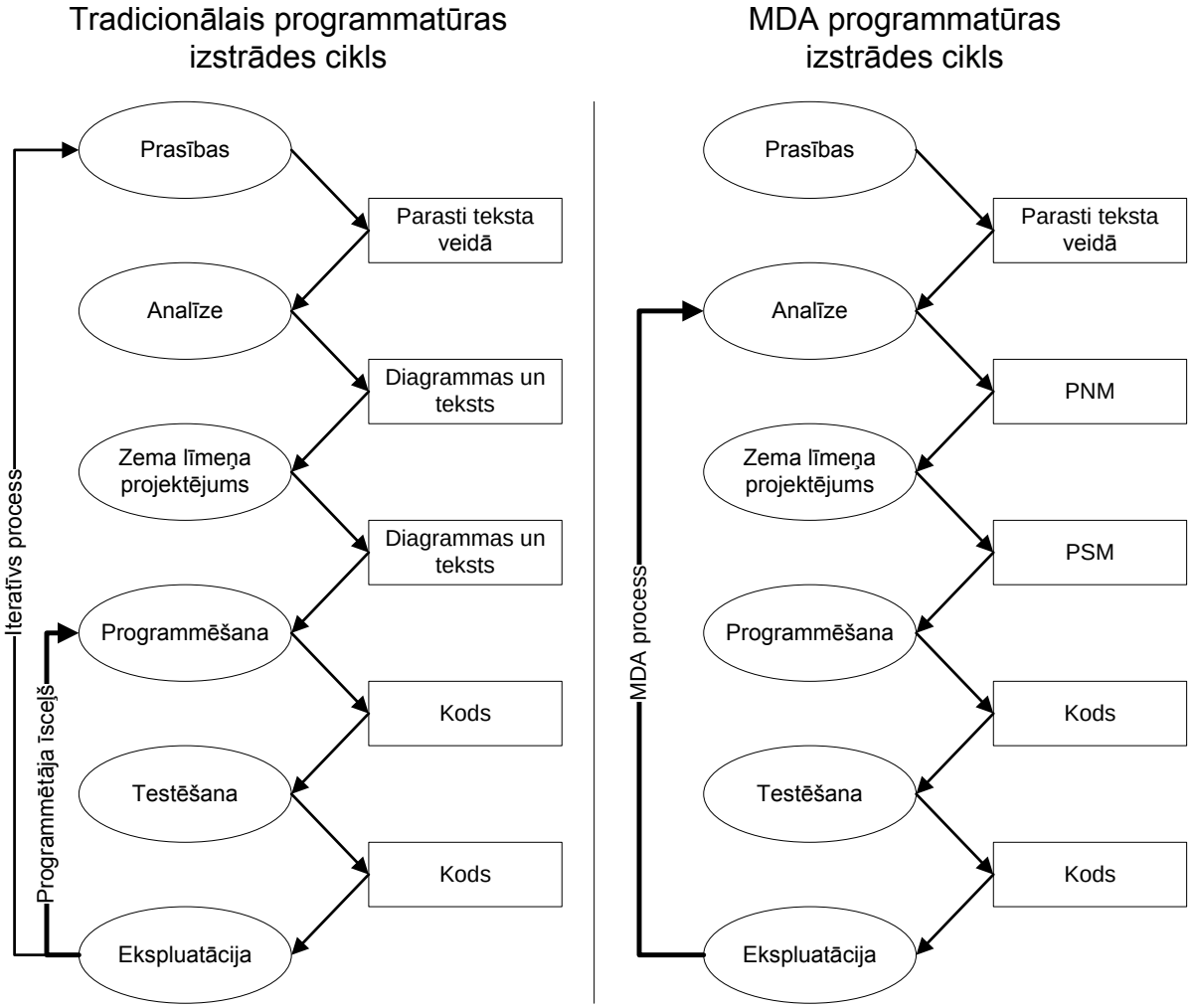
   Pēdējais MBA izstrādes ciklā ir katra PSM transformēšana kodā. Tā kā PSM ļoti tuvināts attiecīgajai tehnoloģijai, transformēšana ir relatīvi vienkārša.

Tātad MBA definē PNM, PSM, kodu un to, kā tie visi savā starpā sadarbosies. Izstrādājot sistēmu, vispirms jāizveido PNM, tad to ir jātransformē uz vienu vai vairākiem PSM, kurus savukārt pēc tam pārvērš kodā. Vissarežģītākais solis MBA izstrādes procesā ir PNM transformēšana uz PSM [5.1.].

MBA priekšrocība ir tā, ka sistēmas specifikācija sastāv no dažādiem abstrakcijas līmeņiem. Spēja transformēt augsta līmeņa PNM uz PSM ļauj izstrādātājiem ar mazāku piepūli darboties ar daudz sarežģītākām sistēmām [1.1.].

2.1. Atšķirība starp tradicionālo izstrādes ciklu un MBA

Lai būtu skaidrs, kādas ir atšķirības starp tradicionālo un MBA programmatūras izstrādes ciklu, 2.1. attēlā parādītas abu šo ciklu shēmas. Kā redzams, abi šie cikli ir gandrīz vienādi. Abos ciklos ir vienādas fāzes, bet būtiskākā atšķirība ir pārejas starp šīm fāzēm. Ja tradicionālajā programmatūras izstrādes procesā pārejas starp fāzēm „Analīze", „Zema līmeņa projektējums" un „Programmēšana" nodrošināja cilvēks, izstrādājot dažādas diagrammas un aprakstus, tad MBA programmatūras izstrādes ciklā šie procesi tiek veikti, izmantojot modeļus PNM un PSM.



2.1. att. **Tradicionālā un MBA programmatūras izstrādes ciklu salīdzinājums [1.1.]**

Tradicionālajā izstrādes ciklā transformēšana no viena modeļa uz citu modeli, no modeļa uz kodu notiek manuāli. Katru reizi, kad ir izmaiņas kādā no modeļiem, nepieciešams

pārveidot arī transformēšanu. Tas ir ļoti laikietilpīgs process. Savukārt MBA šie transformēšanas procesi ir automatizēti, t.i., transformēšanai no PNM uz PSM un pēc tam no PSM uz kodu ir savi transformēšanas rīki. Ja PSM transformēšana uz kodu nav nekas jauns, jo ir ļoti daudz dažādu rīku, kas veic koda ģenerēšanu, tad PNM automatizēta transformēšana uz PSM ir jauna pieeja programmatūras izstrādē, kas būtiski atvieglo sistēmas izstrādi.

2.2.MBA priekšrocības

Tālāk uzskaitītas MBA priekšrocības attiecībā pret citiem programmatūras izstrādes procesiem:

- izstrādātājs var koncentrēties tikai uz PNM izstrādi, neuztraucoties par izstrādājamo tehnoloģiju [5.5.],
- transformēšana no PNM uz PSM jāizstrādā tikai vienreiz un tā var tikt pielietota, izstrādājot vairākas sistēmas,
- viss, kas tiek definēts iekš PNM, ir pilnībā pārnesams uz citu platformu,
- izmaiņas sistēmā tiek veiktas, izmainot PNM un pēc tam pārģenerējot PSM un kodu.

2.3.Secinājumi

Lai automatizētas dokumentācijas ģeneratora arhitektūru būtu iespējams izstrādāt, balstoties uz MBA principiem, nepieciešami vismaz trīs nosacījumi: PNM, PSM un transformēšanas rīks. Uzņēmumā tiek izmantots rīks „ISTehnoloģija" (skat. nodaļu „„ISTehnoloģijas" apskats"), kas ar metamodeli apraksta sistēmu un pēc tam to attiecīgi interpretē. Šis metamodelis ir pilnībā platformas neatkarīgs, jo apraksta tikai biznesa loģiku. No šī sistēmas metamodeļa tiks iegūta visa nepieciešamā informācija dokumentācijas ģenerēšanai.

Lai dokumentācijas ģenerators būtu pilnīgs, nepieciešams ne tikai apkopot informāciju, bet arī uzģenerēt noformētu dokumentu. Šim nolūkam nepieciešams metamodelis, kas apraksta noformētu dokumentu. Šim nolūkam uzņēmumā jau tiek izmantots rīks „DIREPO", kas izmanto metamodeli, kurš apraksta atskaiti jeb dokumentu.

Tātad tas nozīmē, ka ir izpildīti divi nosacījumi no trim – PNM ir realizēts „ISTehnoloģijā", bet PSM – „DIREPO". Bet, lai šo divu rīku modeļus varētu savienot, nepieciešams transformēšanas rīks, kas, izmantojot „ISTehnoloģijā" pieejamo informāciju, spētu to pārveidot par dokumentu, izmantojot „DIREPO". Šādu funkciju pilnībā nodrošina

pats rīks „DIREPO", jo šī rīka funkcionalitātē ietilpst ne tikai atskaišu ģenerēšana no dokumenta metamodeļa, bet arī atskaišu definēšana.

Tomēr ar pašreizējo „ISTehnoloģijas" un „DIREPO" piedāvāto funkcionalitāti nepietiek, lai uzbūvētu dokumentācijas ģeneratoru. Iepriekšējā nodaļā jau tika minēts, ka sistēmai, no kuras tiek ģenerēta dokumentācija, nepieciešams pievienot dokumentācijai nepieciešamo informāciju. Pašlaik izmantotā „ISTehnoloģija" nenodrošina iespēju uzglabāt dokumentācijai nepieciešamo informāciju, tāpēc darba mērķis ir izstrādāt moduli, kas veic šo funkciju, kā arī katram dokumentam izstrādāt savu „DIREPO" atskaites definēšanas šablonu, lai „DIREPO" spētu „ISTehnoloģijā" uzglabāto informāciju pārveidot par dokumentu.

3. „ISTEHNOLOĢIJAS” APSKATS

1995. gadā tika sākts darbs pie Latvijas Bankas vērtspapīru uzskaites sistēmas. Noskaidrojot pasūtītāja prasības, jau pašā sākumā tika secināts, ka veidojamā sistēma ir ļoti sarežģīta, jo lietotājiem nebija skaidras prasības attiecībā pret sistēmu, kā arī biznesa vide bija ļoti mainīga. Protams, sistēmas prasību sarežģītību noteica arī tas, ka pasūtītāju bizness strauji attīstījās, kas radīja jaunas izmaiņas vērtspapīru uzskaites sistēmā. Tādēļ jau pašā sākumā tika izlemts veidot sistēmu, kurā pēc iespējas vieglāk būtu veikt dažādas izmaiņas. Tā rezultātā radās sistēmas projektēšanas un konfigurēšanas rīks, kas ieguva nosaukumu „ISTehnoloģija”.

Šīs sistēmas pamats ir bāzēts uz metamodeļa. Tai ir sava metadatu datu bāze, kas satur informāciju par sistēmas konfigurāciju, kuru spēj interpretēt „ISTehnoloģija”. Metamodeļa izmantošanas galvenā priekšrocība ir tā, ka nav nepieciešamas ģenerēt kodu. Pietiek metamodelī definēto pareizi interpretēt, lai nodrošinātu darbu saskaņā ar biznesa loģiku.

„ISTehnoloģija” sākumā sastāvēja no sekojošiem moduļiem [2.2.]:

- ✓ uzdevumu modulis;
- ✓ organizāciju modulis;
- ✓ objektu modulis;
- ✓ procesu modulis;
- ✓ atskaišu modulis;
- ✓ darba vietu modulis;
- ✓ audita modulis.

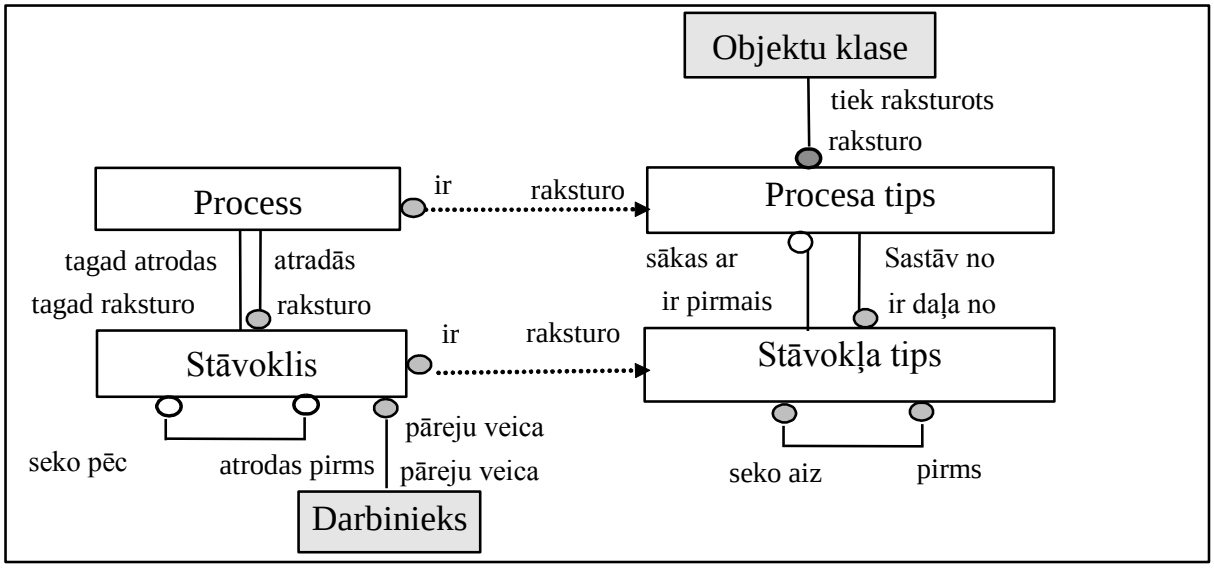
Attiecībā uz dokumentācijas ģenerēšanu „ISTehnoloģijai” jau sākumā Inga Medvedis izveidoja vienkāršu dokumentācijas ģeneratoru, kurš tika nosaukts par „SQLWord”. Lai uzģenerētu dokumentāciju, tas izmantoja „ISTehnoloģijas” metadatus. Šāds rīks noderēja dokumentācijas pirmreizējai ģenerēšanai, bet neguva atzinību atkārtotai un regulārai dokumenta atjaunošanai, jo viena dokumenta uzģenerēšana ar tā laika datoru resursiem aizņēma pārāk ilgu laiku un pēc dokumenta uzģenerēšanas cilvēkam tajā bija jāveic arī būtiskas izmaiņas. Tas nozīmēja, ja bija nepieciešams atjaunot sākotnēji uzģenerēto dokumentāciju, tad, ilgstoši noslogojot datoru, jāģenerē jauna dokumentācija un pēc tam no dokumenta iepriekšējās versijas jāievieto izmaiņas, ko veicis cilvēks. Tādēļ laika gaitā šis rīks dokumentācijas ģenerēšanai vairs netika izmantots.

3.1.„ISTehnoloģijas” moduļi

Šajā apakšnodaļā tiks apskatīti visi pašlaik „ISTehnoloģijā” izmantotie moduļi, kā arī aprakstīts agrāk ieviestais, bet pašlaik neizmantotais objektu modulis. Ar šī moduļa palīdzību iespējams aprakstīt sistēmu. Tā kā ģenerējamā dokumentācija tiešā veidā aprakstīs sistēmu, tad šis apraksts, kā viens no informācijas ieguves avotiem, būs nepieciešams dokumentācijas ģenerēšanā.

3.1.1. Procesu modulis

Procesu modulis nodrošina objektu stāvokļu pāreju definēšanu un maiņu. Procesa un stāvokļa tipi un to attiecības ļauj definēt stāvokļu pāreju procesu tipus. Savukārt procesi, stāvokļi un to attiecības ļauj izpildīt stāvokļu pāreju. Šis modulis uzglabā arī informāciju par to, kurš darbinieks un kad ir veicis noteikta stāvokļa pāreju [2.2.].



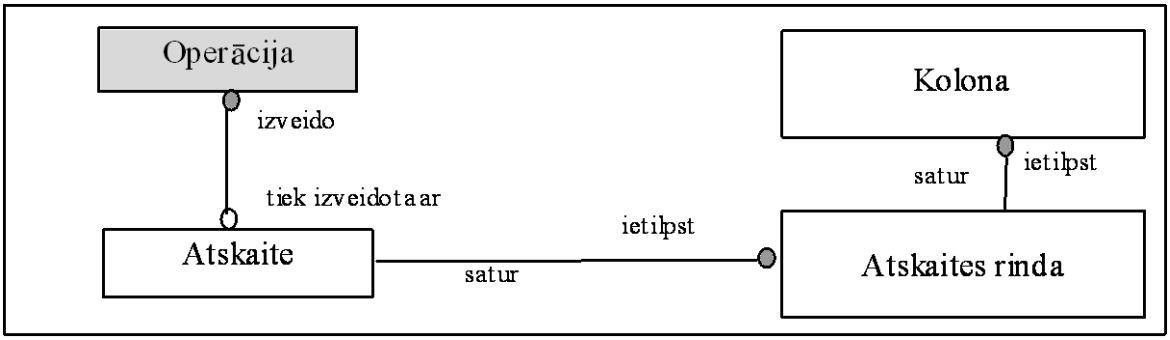
3.1. att. **Daļa no „ISTehnoloģijas” procesu moduļa [2.2.]**

Šāda procesu moduļa izmantošana ļauj izvairīties no dažādām kļūdām, kas var rasties nepareizu stāvokļu pāreju gadījumā, jo tas aizliedz šīs nepareizās stāvokļu pārejas, kā arī vienu stāvokļu pāreju vienlaicīgi ļauj veikt tikai vienam darbiniekam.

3.1.2. Atskaišu modulis

„ISTehnoloģijas” pirmsākumos atskaišu modulis izmantoja iepriekšminēto rīku „SQLWord”, kas noteiktā dokumenta sagatavē ievietoja attiecīgo informāciju, izmantojot SQL pieprasījumus. Jau vairākus gadus „ISTehnoloģijā” rīku „SQLWord” aizstāj daudz modernāks rīks ar nosaukumu „DIREPO”.

„DIREPO” ir universāls atskaišu definēšanas un ģenerēšanas rīks, kas spēj no citas sistēmas ER modeļa iegūt informāciju, kas nepieciešama atskaitei. Tāpat kā „ISTehnoloģija”, arī „DIREPO” ir balstīts uz metamodeļa, proti, tas apraksta, kā izskatīsies atskaite. Atšķirībā no „SQLWord” rīka „DIREPO” nav nepieciešams iepriekš sagatavots dokumenta šablons, jo visa atskaites informācija (arī par izkārtojumu) tiek glabāta datu bāzē.



3.2. att. **Daļa no „ISTehnoloģijas” atskaišu moduļa [2.2]**

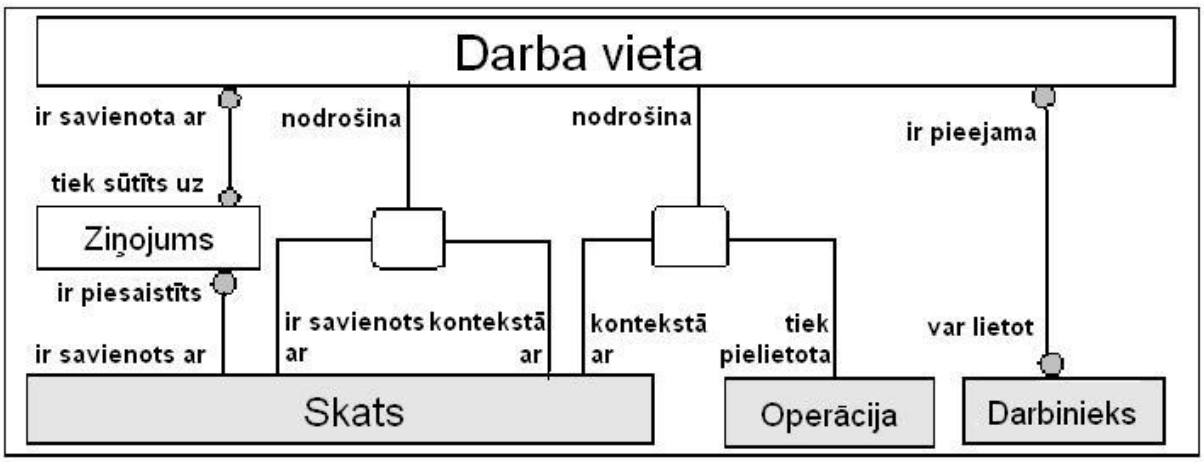
3.2. attēlā redzama atskaites veidošanas shēma, kas nedaudz pielāgota jaunajam atskaišu ģenerēšanas rīkam „DIREPO”. „ISTehnoloģijā”, izsaucot operāciju, kurai piesaistīts noteikts atskaites veids, tiek sameklēta atskaites definīcija, pēc kuras vadoties, tiek uzģenerēta atskaite.

Lai būtu pilnīgāka izpratne par „DIREPO” metamodeļa, kas apraksta dokumenta modeli, uzbūvi, 1. pielikumā pievienots „DIREPO” ER modelis.

3.1.3. Darba vietu modulis

Darba vietu modulis nodrošina darbinieku, sistēmas lietotāju tiesības un pienākumus.

Darba vietu moduļa shēma redzama 3.3. attēlā.



3.3. att. **Daļa no „ISTehnoloģijas” darba vietu moduļa [2.2]**

Darba vieta

Darba vietas entītijas atribūti ir nosaukums un komentārs. Tā nodrošina darba vietas atpazīšanu, kā arī lietotāju pieejas tiesības un pienākumus.

Ziņojums

Ziņojuma entītijas atribūti ir ziņojuma teksts, ikona, kas parādās pie ziņojuma teksta, un vaicājums. Vaicājumam var pievienot mainīgos. Vaicājumu izpilde notiek pēc noteikta laika intervāla. Ja vaicājums atgriež vismaz vienu vērtību, tad ziņojuma teksts un ikona tiek parādīta sistēmas lietotājam. Ziņojumus ir iespējams atslēgt un pēc tam atkal ieslēgt.

Ziņojumi ir piesaistīti darba vietai un skatam. Tas nozīmē, ka katrai darba vietai ir savi piedefinētie ziņojumi. Ziņojuma sasaiste ar skatu ir domāta tam, lai no ziņojuma varētu nokļūt uz skatu. Piemēram, parādās paziņojums „Brīdinājums: FX darījumiem nepieciešama apstiprināšana!". Tad lietotājs, veicot dubultklikšķi uz ziņojuma, var nokļūt pa taisno uz attiecīgo skatu, kur atrodas FX darījumi. Protams, skatam jābūt piesaistītam attiecīgajai darba vietai.

Skats

Katrai darba vietai ir piesaistīti iepriekš definēti skati. Katram skatam par apakšlīmeni var būt viens vai vairāki skati vai objekti, kas atbilst noteiktai objektu klasei. Objektam apakšlīmenī var būt tikai skats.

Operācija

Operācijas ļauj izpildīt noteiktas iepriekš definētas darbības ar skatiem un objektiem konkrētā darba vietā.

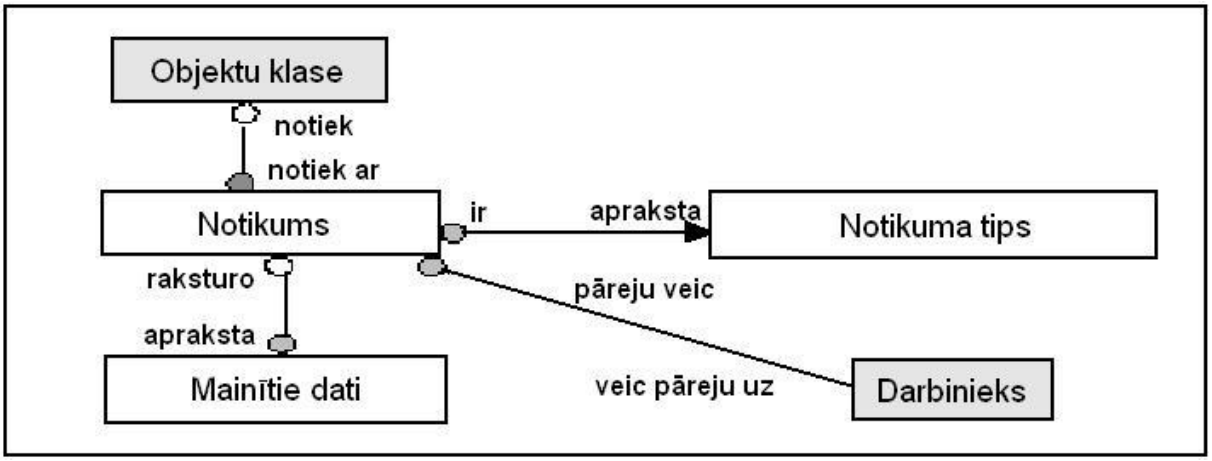
Darbinieks

Katram darbiniekam ir piesaistītas noteiktas darba vietas, kurās viņš drīkst darboties. Tas ļauj nodrošināt darbinieku pieejas tiesības un pienākumus.

3.1.4. Audita modulis

Audita modulis piereģistrē šādas darbības jeb notikuma tipus:

- ✓ Pieslēgšanās sistēmai
- ✓ Iziešana no sistēmas
- ✓ Datu skatīšanās
- ✓ Datu pievienošana
- ✓ Datu mainīšana
- ✓ Datu dzēšana



3.4. att. **Daļa no „ISTehnoloģijas" audita moduļa [2.2]**

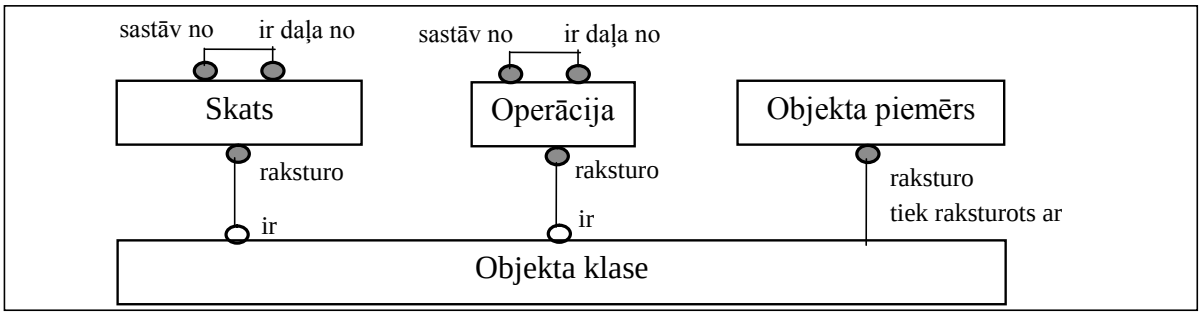
Visās reģistrēšanas darbībās darbību žurnālā tiek piereģistrēts arī lietotājs un viens no augstākminētajiem notikuma tipiem.

Piereģistrējot datu skatīšanos un mainīšanu, tiek noradīta arī datu objektu klase. Datu mainīšanai, piereģistrējot darbību žurnālā, tiek norādītas datu iepriekšējās vērtības.

Audita modulis nodrošina

3.1.5. Objektu modulis

Sākotnējā „ISTehnoloģijas" versijā bija iekļauts arī objektu modulis. Šo modeli bija paredzēts izmantot sistēmas prototipa veidošanā. Tas nodrošināja iespēju nodefinēt datu bāzes tabulas ar visām kolonnām, ierakstiem un to vērtībām, neveicot izmaiņas datu bāzes struktūrā, proti, datu bāzē glabājās modelis, kas aprakstīja objektu. Pievienojot jauna tipa objektus, nevajadzēja veidot jaunas datu bāzes tabulas. Pietika ar vairāku ierakstu ievietošanu tabulās, kas aprakstīja šo objektu.



3.5. att. **Daļa no „ISTehnoloģijas" objektu moduļa [2.2]**

Laika gaitā, praksē pielietojot „ISTehnoloģiju", pierādījās, ka šāds objektu modulis ir neefektīvs, jo pierādījās, ka datu bāzē izveidot objektus bija vieglāk nekā vispirms lietot objektu moduli un pēc tam pārveidot konfigurāciju [2.1.].

Savukārt, izstrādājot automatizētu dokumentācijas ģeneratoru „ISTehnoloģijai" un apskatot šī rīka vēsturisko objektu moduli, tika secināts, ka šāds modulis ļoti labi apraksta sistēmu. Šis modulis glabā informāciju par sistēmas tabulām un to struktūru, un, ņemot vērā, ka programmatūras projektu ietvaros nepieciešams ģenerēt arī tabulu struktūru aprakstošu dokumentāciju, šis modulis, protams, ar dažām izmaiņām noderēs sistēmas informācijas uzglabāšanai un iegūšanai. Detalizētāk ar jauno objektu moduli iespējams iepazīties 7.1.2. apakšnodaļā „„ISTehnoloģijas" objektu moduļa atjaunošana".

4. DOKUMENTĀCIJU VEIDI UN STANDARTI

Mūsdienās visās nozarēs un darbības sfērās katram projektam vai oficiālai darbībai ir savi standarti, pēc kuriem ieteicams vadīties. Informāciju tehnoloģiju nozarē standarti ieņem svarīgu lomu, jo tie palīdz specificēt četras programmatūras izstrādē svarīgas sfēras: funkcionalitāti, kvalitāti, laiku un budžetu. Standartos aprakstītās prasības attiecībā uz dokumenta saturu un noformējumu nav obligātas, bet ieteicamas, tāpēc katrs dokumenta sastādītājs to var pielāgot savām vajadzībām. Dokumentiem, kas veidoti, balstoties uz standartu, ir vienota struktūra, kas nodrošina informācijas kodolīgumu, lasāmību un neatkārtošanos.

4.1.Latvijas Valsts standarti

Apakšnodaļa izstrādāta, balstoties uz [1.2.].

Latvijā IT standartus izstrādā SIA „Latvijas standarts" (LVS), tāpēc turpmāk darbā dokumentācijas kontekstā tiks apskatīti tieši LVS standarti.

Izstrādājot programmatūru, LVS iesaka vadīties pēc šādiem Latvijas Valsts standartiem [1.2.]:

Programmatūras dokumentēšana dažādās tās attīstības stadijās:

- Informācijas tehnoloģija. Programmatūras lietotāja dokumentācija (LVS 66:1996)

  Standarts nosaka programmatūras lietotāja dokumentācijas (PLD) formātu un minimālās prasības. PLD ir palīgs lietotājam darbā ar sistēmu. Tas iekļauj vairākus dokumentus, bet Latvijas standarts ir izstrādāts sekojošām lietotāju dokumentācijām:

  - Lietotāja ceļvedis programmatūras instalēšanā;
  - Lietotāja ceļvedis programmatūras darbināšanā;
  - Lietotāja ceļvedis programmatūras pārvaldīšanā.

- Informācijas tehnoloģija. Programmatūras prasību specifikācijas (PPS) ceļvedis (LVS 68:1996)

  Standarts ir ceļvedis lietotāja prasību specificēšanā. Izmantojot šo standartu, var izstrādāt pilnīgu un noteiktu PPS. Šajā dokumentā tiek aprakstītas detalizētas prasības katram programmatūras vienumam. Pēc šī dokumenta turpmākajā izstrādes gaitā tiek veikta programmatūras

testēšana. Šis dokuments tiek izstrādāts ciešā sadarbībā ar pasūtītāju un tiek sagatavots jebkura programmatūras projekta izstrādes gaitā.

- Informācijas tehnoloģija. Programmatūras testēšanas dokumentācija (LVS 70:1996)

  Standarts nosaka, kā ir veidojama programmatūras testēšanas dokumentācija (PTD) un kāda informācija tai ir jāsniedz dokumenta lasītājam. PTD nosaka, kā tiks veikta sistēmas testēšana. Šī dokumentācija apraksta arī testēšanas kalendāro plānu, testēšanas rezultātus un to novērtējumu. Šī dokumentācija iekļauj šādus sistēmas testēšanu aprakstošus dokumentus:

  - Programmatūras testēšanas apraksts;
  - Testu projektējuma specifikācija;
  - Testpiemēru specifikācija;
  - Testēšanas procedūras specifikācija;
  - Testēšanas žurnāls;
  - Problēmu ziņojumi;
  - Testēšanas (kopsavilkuma) pārskats.

- Informācijas tehnoloģija. Ieteicamā prakse programmatūras projektējuma aprakstīšanai (LVS 72:1996)

  Programmatūras projektējuma aprakstam (PPA) standarts apraksta ieteicamo praksi dokumenta veidošanā. PPA ir attēlojums, kā PPS iekļautās prasības tiks realizētas izvēlētajā programmatūras realizācijas vidē. Tas ir palīgs analīzē, plānošanā, implementēšanā un lēmumu pieņemšanā.

- Informācijas tehnoloģija. Sistēmas darbības koncepcijas apraksts (LVS 75:1996)

  Darbības koncepcijas apraksts ir pirmais dokuments, uz kura pamata pasūtītājs un izstrādātājs vienojas par pamatprasībām, kurai izstrādājamai sistēmai jāatbilst.

- Programmatūras izstrādes procesa dokumentēšana dažādās tās attīstības stadijās:
  - Informācijas tehnoloģija. Programmatūras projekta pārvaldības plāns (LVS 67:1996)

Standarts apraksta dokumenta formātu un saturu. Programmatūras projekta pārvaldības plāns ir pamatdokuments programmatūras projekta pārvaldē. Tajā ir jānosaka tehniskās un pārvaldošās projekta funkcijas, aktivitātes un uzdevumi, kas nepieciešami projekta līgumā noteikto programmatūras projekta prasību apmierināšanai.

- Informācijas tehnoloģija. Programmatūras konfigurācijas pārvaldības plāns (LVS 69:1996)

  Programmatūras konfigurācijas pārvaldības plāns identificē konfigurāciju, kā tiks organizēta konfigurācijas vadība, stāvokļu uzskaite, auditēšana un apskate, kā arī produkta izlaides organizēšana. Standarts nosaka šī dokumenta saturu un formātu.

- Informācijas tehnoloģija. Programmatūras verifikācijas un validācijas plāns (LVS 71:1996)

  Standarts apraksta minimālās prasības programmatūras verifikācijas un validācijas plāna (PVVP) izstrādāšanai un tā saturam. PVVP specificē, kā tiks noteikts, vai produkts attiecīgajā izstrādes fāzē atbilst iepriekšējā fāzē noteiktajām prasībām un vai produkts atbilst projekta sākumā specificētajām prasībām.

- Programmatūras kvalitātes nodrošināšanas pasākumu veikšana:
  - Informācijas tehnoloģija. Programmatūras kvalitātes nodrošināšanas plāns (LVS 65:1996)

    Standarts nosaka programmatūras kvalitātes nodrošināšanas plāna (PKNP) formātu un minimālās prasības. PKNP apraksta, kā tiks nodrošināts, ka programmatūra atbilst izvirzītajām tehniskajām prasībām. Šim dokumentam standarta pielietošana ir obligāta, ja tiek izstrādāta programmatūra, kuras kļūda dotu triecienu drošībai vai varētu būt cēlonis lieliem finansiāliem vai sociāliem zaudējumiem.

  - Informācijas tehnoloģija. Programmatūras vienībtestēšana (LVS 73:1996)

    Standarts specificē programmatūras vienībtestēšanu. Šis process ietver testēšanas plānošanu, testu kopas iegūšanu un testējamās programmatūras vienības mērīšanu, salīdzinot testējamo vienību ar tai noteiktajām prasībām. Mērīšana nozīmē datu paraugu izmantošanu, lai izpildītu

vienību un salīdzinātu vienības faktisko uzvešanos ar to uzvedību, kas specificēta vienības prasību dokumentācijā.

- Informācijas tehnoloģija. Programmatūras apskate un auditēšana (LVS 74:1996)

  Standarts definē apskates un auditēšanas procesus, kas izmantojami gan kritiskai, gan nekritiskai programmatūrai, kā arī specifiskās procedūras, kuras nepieciešamas apskates un auditēšanas izpildei.

4.2. Uzņēmumā SIA „Datorikas institūts DIVI" izmantotā dokumentācija

Informācijas tehnoloģijas ir salīdzinoši jauna nozare Latvijā, tāpēc pagaidām standarti nav izstrādāti pilnīgi visiem dokumentu veidiem. Tiem, kuriem nav noteiktu Latvijas standartu, par pamatu tiks ņemta uzņēmumā SIA „Datorikas institūts DIVI" pieņemtā prakse dokumentu veidošanā.

Uzņēmums SIA „Datorikas institūts DIVI" nodarbojas ar programmatūras izstrādi un, kā jebkurā nopietnā IT uzņēmumā, arī šajā dokumentu izstrādē tiek lietoti standarti. 4.1. tabulā attēlota uzņēmumā izmantotā dokumentācija, kas tiek izmantota programmatūras projektu dažādos posmos. Katram dokumentam ir norādīts standarts, pēc kura izstrādāts dokuments. Tiem dokumentiem, kam nav noteikta Latvijas standarta, norādīts avots, pēc kura tas izstrādāts.

4.1. tabula

Uzņēmumā SIA „Datorikas institūts DIVI" izmantotā dokumentācija

Projekta sagatavošanas un uzturēšanas dokumenti	
Sākotnējie projekta priekšlikumi	Uzņēmumā pieņemtā prakse
Līgums ar pasūtītāju	Uzņēmumā pieņemtā prakse
Līguma izpildes kalendārais plāns	Uzņēmumā pieņemtā prakse
Projekta plānošanas dokumenti	
Programmatūras (Sistēmas) izstrādes plāns	Uzņēmumā pieņemtā prakse
Projekta pārvaldības plāns	LVS 67:1996
Programmatūras konfigurācijas pārvaldības plāns	LVS 69:1996
Projekta iekšējo pārbaužu plāns	Uzņēmumā pieņemtā prakse
Programmatūras (kvalifikācijas) testēšanas plāns	LVS 70:1996
Programmatūras instalēšanas plāns	Uzņēmumā pieņemtā prakse
Projekta specifikācijas	
Sistēmas/apakšsistēmas (ar aparatūru) prasību specifikācija	LVS 68:1996
Programmatūras prasību specifikācija	LVS 68:1996
Saskarņu prasību specifikācija	LVS 68:1996

Programmatūras versijas apraksts	Uzņēmumā pieņemtā prakse
Projektējumu dokumentācija	
Programmatūras projektējuma apraksts	LVS 72:1996
Datu bāzu projektējuma apraksts	LVS 72:1996
Lietotāja dokumentācija	
Lietotāja rokasgrāmata	LVS 66:1996
Programmatūras instalēšanas apraksts	Uzņēmumā pieņemtā prakse
Testēšanas dokumentācija	
Programmatūras testēšanas apraksts	LVS 70:1996
Testpiemēru specifikācijas	LVS 70:1996
Problēmu ziņojumi	Kļūdu pārvaldības rīks „Bugzilla”
Cita dokumentācija	
Nododamo vienumu saraksts	Uzņēmumā pieņemtā prakse

5. UZŅĒMUMĀ IZMANTOTĀS DOKUMENTĀCIJAS AUTOMATIZĒTAS ĢENERĒŠANAS IESPĒJAMĪBAS NOVĒRTĒJUMS

Iepriekšējā nodaļā tika uzskaitīti daudzi dažādi dokumenti un to standarti, kurus izmanto programmatūras projektos, bet tikai daļu no šiem dokumentiem ir iespējams automatizēti uzģenerēt. Tāpēc ir svarīgi novērtēt, kuru no iepriekšminētajām dokumentācijām iespējams automatizēti uzģenerēt.

Automatizētas dokumentācijas ģenerēšanas iespējamība tiks novērtēta tikai tiem dokumentiem, kurus izmanto uzņēmumā. Tāpēc turpmākajā dokumentu analīzē neparādīsies pilnīgi visi iepriekš uzskaitītie dokumenti, bet gan tikai tie, kas tiek izmantoti uzņēmuma programmatūras projektu ietvaros.

Kā redzams 4.1. tabulā, vispirms programmatūras projekts sākas ar projekta sagatavošanas un uzturēšanas dokumentiem. Šajos dokumentos tiek aprakstīta vispārīga informācija par projektu kopumā, nosacījumi par projekta izstrādi un iekļaušanos noteiktā laika posmā. Tālāk programmatūras projekta gaitā tiek izstrādāti dažādi projekta plānošanas dokumenti, kas pilnībā raksturo attiecīgā programmatūras projekta izpildes un pārbaudes gaitu. Šie dokumenti minimāli raksturo pašu sistēmu. Tie apraksta projekta izpildes nosacījumus un termiņus, nevis izstrādājamo sistēmu, tāpēc šāda veida dokumentācijas ģenerēšana no sistēmā pieejamās informācijas nav iespējama.

Programmatūras projektu ietvaros ciešā sadarbībā ar pasūtītāju tiek izstrādātas arī dažādas projekta specifikācijas, kurās sīki un detalizēti tiek specificētas pasūtītāja izvirzītās prasības attiecībā uz izstrādājamo sistēmu. Jau fakts, ka šie dokumenti tiek izstrādāti sadarbībā ar pasūtītāju, pasaka priekšā, ka šāda veida dokumentos ietverto informāciju nav iespējams uzģenerēt, jo informācija tiek iegūta nevis no sistēmas, bet no pasūtītāja.

Uzņēmumā neatņemama programmatūras projektu dokumentācijas sastāvdaļa ir projektējumu dokumentācija, kas satur divus atsevišķus dokumentus: programmatūras projektējuma aprakstu un datu bāzu projektējuma aprakstu. Pēdējo aprakstu LVS standarts apraksta kā programmatūras projektējuma aprakstā iekļaujamo nodaļu, bet uzņēmumā ir izstrādājusies prakse šos dokumentus atdalīt to lielā apjoma dēļ. Tāpēc turpmāk PPA tiks apskatīts, kā dokumentācija, kurā nav nodaļas par datu bāzu projektējumu.

Programmatūras projektējuma apraksts ir salīdzinoši liels dokuments, kas programmētājiem saprotamā valodā pilnībā apraksta sistēmas uzbūvi. Lai dators spētu

uzģenerēt dokumentāciju, ir ļoti svarīgi, lai ģenerējamajam dokumentam būtu noteikta struktūra. PPA pārsvarā sastāv no cilvēka valodā rakstītām nodaļām, kurām nav šīs noteiktās struktūras. Cita situācija ir ar datu bāzu projektējuma aprakstu. Šajā dokumentā ir vienota struktūra, un tas satur informāciju, no kuras lielāko daļu iespējams iegūt no datu bāzes pārvaldības sistēmas tabulām un „ISTehnoloģijas" objektu moduļa.

Nedaudz savādāka situācija ir ar lietotāja rokasgrāmatu. Šis dokuments satur informāciju, kas ir strukturizēta, tiešā veidā apraksta sistēmu un precīzi atbilst sistēmai, tāpēc šāda veida dokumentāciju ir iespējams automātiski uzģenerēt no sistēmā pieejamās informācijas. Tā kā „ISTehnoloģija" neuzglabā nekādu informāciju, kas saistīta ar tās instalēšanu, bet toties ir iespējams piesaistīt informāciju, kas saistīta ar pašu sistēmas darbināšanu un pārvaldību, tad automatizēti tiks ģenerēta tāda lietotāja rokasgrāmata, kas satur lietotāja ceļvedi gan programmatūras darbināšanā, gan arī pārvaldīšanā. Informāciju šī dokumenta ģenerēšanai iespējams iegūt no sistēmas projektējuma jeb „ISTehnoloģijas" metamodeļa.

Testēšanas dokumentācija ir svarīga sastāvdaļa katra programmatūras projekta izstrādes dzīves ciklā, jo ar to palīdzību sistēmā tiek apzinātas, reģistrētas un labotas kļūdas. Šādu dokumentu ģenerēšana nav šī darba mērķis, jo informācija netiek glabāta sistēmā, kā arī pasaulē eksistē dažādas programmas kļūdu reģistrēšanai un testpiemēru specificēšanai.

Nododamo vienumu saraksts tiek nodots pasūtītājam, kad ir noslēdzies kāds projekta posms un pasūtītājam tiek nodoti dažādi projekta gaitā izstrādātie vienumi. Ja ņem vērā „ISTehnoloģijas" uzbūvi, kurā katra programmatūras izpilddatne tiek izsaukta, izmantojot kādu no operācijām, tad nododamā vienuma daļu, kas saistīta ar programmatūras vienumiem, iespējams pilnībā uzģenerēt.

5.1.Secinājumi

Bakalaura darba ietvaros tiks apskatīta tādas dokumentācijas ģenerēšana, kurai nepieciešamo informāciju pilnībā vai daļēji iespējams iegūt no sistēmas vai kuru bez būtiskas sistēmas darbības koncepcijas mainīšanas iespējams pievienot sistēmai.

Kā iepriekš tika noskaidrots, tad turpmāk bakalaura darbā tiks apskatīta šādu trīs dokumentu automatizēta ģenerēšana „ISTehnoloģijai":

- Datu bāzu projektējuma apraksts;
- Lietotāja rokasgrāmata;
- Nododamo vienumu saraksts.

6. ĢENERĒJAMĀ DOKUMENTĀCIJA

Iepriekšējā nodaļā tika noskaidrots, kurus uzņēmumā izmantotos dokumentus iespējams automatizēti uzģenerēt. Lielām informācijas sistēmām šie dokumenti ir ļoti apjomīgi un bieži vien satur informāciju, kuru dažādu iemeslu dēļ nav iespējams automatizēti uzģenerēt. Tādēļ šajā nodaļā tiks noskaidrots, kādu dokumentā iekļaujamo informāciju iespējams automātiski uzģenerēt un kādu – nav.

6.1.Datu bāzu projektējuma apraksts

Datu bāzu projektējuma aprakstam Latvijā nav izstrādāts atsevišķs standarts, bet ieteicamās prasības attiecībā uz programmatūras datu bāzu projektējuma aprakstu ir standartizētas iekš programmatūras projektējuma apraksta (PPA) LVS standarta [4.1.]. Ja datu bāzu projektējuma apraksts ir pietiekoši neliels, tad to parasti iekļauj PPA, kā atsevišķu nodaļu, bet, ņemot vērā, ka „ISTehnoloģijai" datu bāzes lielums lielā mērā ir atkarīgs no biznesa loģikas un parasti šāda veida sistēmas izmanto sarežģītu un bieži vien mainīgu informācijas sistēmu veidošanai, datu bāzu apraksta apjoms ir salīdzinoši liels. Tāpēc „ISTehnoloģijai" datu bāzu projektējuma apraksts tiek veidots kā atsevišķs dokuments, vadoties pēc LVS [4.1.] noteiktajiem standartiem.

6.1.1. Dokumenta saturs

Ņemot vērā to, ka uzņēmumā ir pieņemta prakse datu bāzu projektējuma aprakstu veidot kā atsevišķu dokumentu, bet LVS standarts to apraksta kā atsevišķu nodaļu, tad dokumenta ievads tiek veidots pēc uzņēmumā pieņemtās prakses, bet dokumenta pamatteksts tiek veidots, balstoties uz LVS standartu PPA izstrādei.

Dokumenta ievadā tiek iekļautas šādas apakšnodaļas:

- Auditorijas raksturojums;
- Dokumenta nolūks;
- Problēmu ziņošana;
- Lietotie apzīmējumi un terminoloģija.

Šīs apakšnodaļas satur informāciju, kas attiecas uz dokumenta uzbūvi un to lasītājiem, bet nesatur informāciju par sistēmu. Tāpēc šī nodaļa ir jāpievieno dokumenta sastādītājam bez automatizētas ģenerēšanas.

Tālāk dokumentā seko pamatteksts, kas apraksta datu bāzi. Balstoties uz LVS standartā iekļautā dokumenta satura piemēru, datu bāzu projektējumā tiek iekļautas tās nodaļas, kas attiecas uz datiem (skat. 6.1. tabulā).

6.1. tabula

Nodaļas, kas attiecas uz datiem

3.3. Datu dekompozīcija
3.3.1. Pirmās datu entītijas apraksts
3.3.2. Otrās datu entītijas apraksts
3.3.3. ...
4.3. Datu atkarības
6.2. Datu detalizētais projektējums
6.2.1. Pirmās datu entītijas detalizējums
6.2.2. Otrās datu entītijas detalizējums
6.2.3. ...

Vadoties pēc standartā noteiktās dokumenta struktūras un iekļaujamās informācijas, uzņēmumā pieņemts veidot datu bāzu projektējumu ar šādām trīs nodaļām:

1. Datu bāzes tabulas
2. Datu atkarības
3. Datu detalizētais projektējums

Šīs trīs nodaļas iekļaujamās informācijas ziņā attiecīgi atbilst 6.1. tabulā aprakstītajām nodaļām.

Nodaļā „Datu bāzes tabulas" tiek uzskaitītas visas datu bāzē esošās tabulas. Tas tiek darīts, izveidojot tabulu, kurā vienā kolonnā tiek rakstīts tabulas nosaukums, otrā – kolonnu skaits, bet trešajā – tabulas apraksts. Šīs nodaļas automatizēta ģenerēšana ir iespējama tādā gadījumā, ja sistēmā tiek glabāts katras tabulas apraksts, jo tabulu nosaukumus un kolonnu skaitu iespējams iegūt no datu bāzes, bet tabulu paskaidrojošie apraksti ir jāveido cilvēkiem.

Savukārt nodaļā „Datu atkarības" tiek iekļauts datu bāzes ER modelis. Mūsdienās datu bāzu pārvaldības sistēmas piedāvā iespēju automātiski ģenerēt ER modeļus no sistēmā pieejamās informācijas. Tātad šīs nodaļas satura automatizēta ģenerēšana ir iespējama tikai tad, ja starp tabulām ir izveidotas visas ārējās atslēgas, pēc kurām datu bāzu pārvaldības sistēma nosaka atkarības starp tabulām.

Kā pēdējā un vislielākā nodaļa, kas tiek iekļauta dokumentā, ir „Datu detalizētais projektējums". Šajā nodaļā tiek aprakstīta katra datu bāzes tabula atsevišķi. Katrai tabulai tiek raksturota tās struktūra, primārās un unikālās atslēgas, ierobežojumi, tabulas, uz kurām tabulai ir reference, kā arī tabulas, kurām ir reference uz šo tabulu. Tāpat kā gadījumā ar ER modeļa ģenerēšanu, arī tabulu raksturojošo informāciju iespējams pilnībā uzģenerēt, izmantojot datu

bāzu pārvaldības sistēmu un tās tabulas. Protams, ar nosacījumu, ja visa iepriekšminētā informācija ir ievadīta datu bāzu pārvaldības sistēmā. Tomēr ar datu bāzu pārvaldības sistēmas piedāvāto informāciju nepietiek, jo no tās nevar iegūt paskaidrojošu informāciju, kādam nolūkam kalpo katrs objekts. Tāpēc daļu informācijas nepieciešams ņemt arī no „ISTehnoloģijas" projektējuma.

6.1.2. Secinājumi

LVS standarts pilnībā nenosaka, kādas nodaļas iekļaujamas datu bāzu projektējuma aprakstā, ja tas tiek veidots kā atsevišķs dokuments. Tas nosaka tikai dokumenta pamattekstā iekļaujamo informāciju un tās struktūru, tāpēc dokumenta ievads tiek veidots pēc uzņēmumā pieņemtās prakses.

Dokumentā iespējams uzģenerēt tieši pamattekstu, kas no dokumenta kopējā apjoma sastāda apmēram 90 procentus. Šāda dokumenta automatizēta ģenerēšana ir ievērojams ieguvums, jo šie dokumenti lielām sistēmām ar lielām datu bāzēm ir ļoti apjomīgi un to izstrādāšana un pēc tam aktuālās informācijas uzturēšana ir ļoti laikietilpīgs process. Tas arī samazina kļūdu iespējamību dokumenta saturā.

Datu bāzu projektējuma aprakstam tiks ģenerētas šādas nodaļas:

6.2. tabula

Datu bāzu projektējuma apraksta ģenerējamās nodaļas

Nodaļa	**Piezīmes**
Titullapa	
Satura rādītājs	Tiks ģenerēts, izmantojot teksta redaktora iespējas
Datu bāzes tabulas	Tiks ģenerēts, izmantojot datu bāzu pārvaldības sistēmu
Datu atkarības	
Datu detalizētais projektējums	

6.2. Lietotāja rokasgrāmata

Ņemot vērā to, ka „ISTehnoloģija" tiek pielietota salīdzinoši lielām sistēmām, lietotāja rokasgrāmatas apjoms noteikti pārsniegs 8 lappušu apjomu. Tāpēc tiks apskatītas tās nodaļas, kuras LVS standarts iesaka iekļaut dokumentos, kas pārsniedz 8 lpp.

6.2.1. Dokumenta saturs

6.3. tabulā redzams LVS standarta noteiktais lietotāja rokasgrāmatas saturs. Šajā tabulā pievienotas arī tās nodaļas, kuras lietotāja rokasgrāmatā tiek iekļautas, balstoties uz uzņēmumā pieņemto praksi.

6.3. tabula

Iekļaušanas prasības

Titullapa	O
Ierobežojumi	O
Garantijas	R
Satura rādītājs	O
Ilustrāciju saraksts	N
Ievads	
Auditorijas raksturojums	O
Darbības sfēra	U
Tehniskais un programmnodrošinājums	U
Lietojamība	O
Dokumenta nolūks	O
Dokumenta lietošanas apraksts	O
Saistītie dokumenti	R
Pieņemtie apzīmējumi un vienošanās	O
Problēmu ziņošana	O
Dokumenta pamatteksts	
Sistēmas vispārīgs apraksts	U
Izziņu veida	1
Instruktīva veida	1
Kļūdu situācijas	R
Pielikumi	N
Atsauces	O
Skaidrojošā vārdnīca	O
Alfabētiskais priekšmetu rādītājs	2

O - Obligāts, jāiekļauj, ja informācija eksistē.

N - Neobligāts.

R - Reference jeb atsauce: informācija tieši iekļaujama atsevišķā nodalījumā vai arī norādāma vieta, kur dokumentu komplektā tā būtu atrodama.

U – Nodaļa, kura tiek iekļauta, balstoties uz uzņēmumā pieņemto praksi.

*** -** Norāda saistību ar citiem sējumiem.

**** -** Obligāti jābūt vismaz kādā no sējumiem, kur norāda arī uz citos sējumos ietverto informāciju.

1 - Ikvienā dokumentā ir pamatteksts; katrā dokumentu komplektā jābūt auditorijai nepieciešamajiem instrukciju veida un izziņu veida dokumentiem.

2 - Priekšmetu rādītājs ir obligāts dokumentiem ar apjomu 40 lpp. un vairāk [4.2.].

Titullapā iekļaujamā informācija ir viegli iegūstama, un līdz ar to iespējams to automatizēti uzģenerēt, jo vienīgā informācija, kas mainās, ir dokumenta versija un datums. Pārējā informācija ir statiska un jāģenerē tikai pirmajā reizē.

Savukārt nodaļās „Ierobežojumi" un „Garantija" iekļaujamā informācija ir nenoteikta, proti, tāda, kuru nevar automatizēti uzģenerēt, jo parasti tiek rakstīta brīvā tekstā un ir atkarīga no attiecīgās sistēmas un tās versijas.

Dokumentā obligāti iekļaujamais satura rādītājs un neobligātais ilustrāciju saraksts ir pilnībā automātiski uzģenerējams, jo arī pašā dokumentā parasti šāda veida informācija tiek veidota automātiski, izmantojot teksta redaktora iespējas.

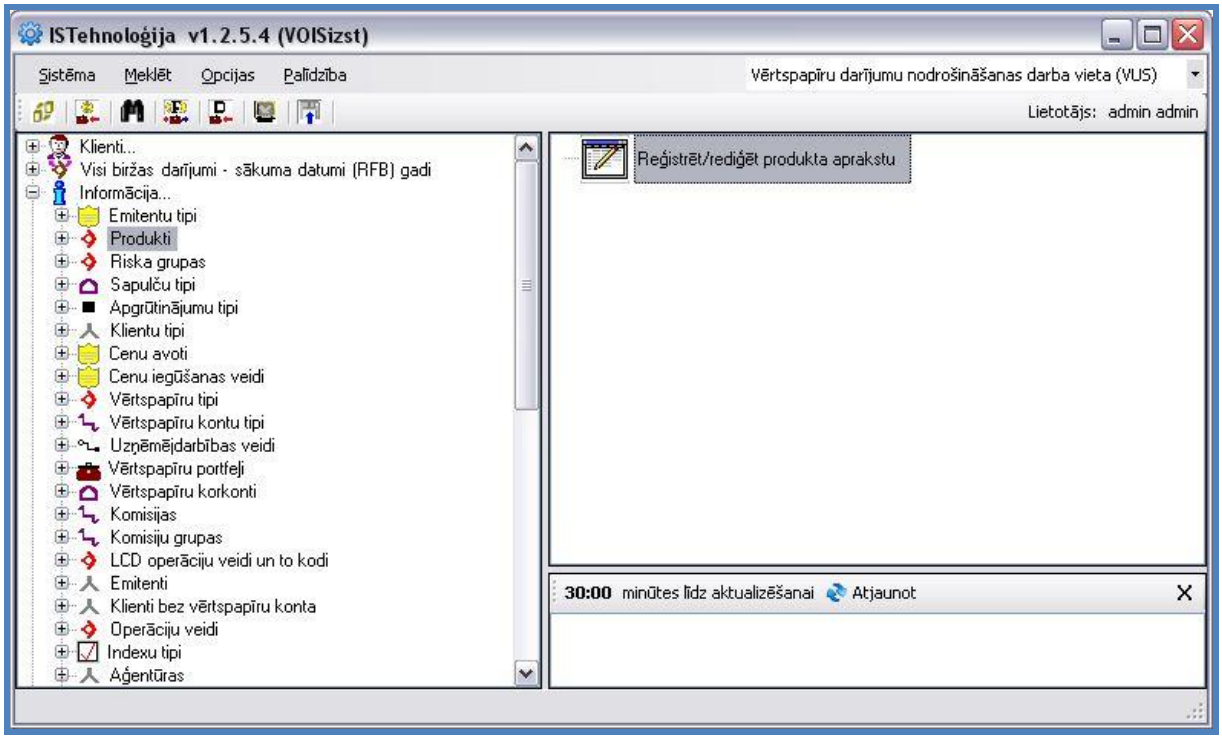
Kā nākošo LVS standarts specificē nodaļu „Ievads". Šajā nodaļā visas iekļaujamās apakšnodaļas satur informāciju, kuru neuzglabā pati sistēma, tādēļ šāda veida informāciju nav iespējams uzģenerēt no sistēmas. Šī informācija dokumenta sagatavotājam būs jāiekļauj atsevišķi pēc dokumenta uzģenerēšanas. Tāda pati situācija ir arī ar nodaļu „Sistēmas vispārīgs apraksts". Šajā nodaļā tiek aprakstīta sistēmas uzbūve un darbības pamatprincipi „cilvēka valodā", brīvā tekstā. Šī darba mērķis nav ģenerēt šāda veida informāciju, bet gan tādu, ko var iegūt no sistēmas.

Tālāk standarts nosaka, ka dokumentā jāiekļauj izziņu vai instruktīva veida lietotāja rokasgrāmatas pamatteksts. Šajā nodaļā iekļaujamo informāciju iespējams automatizēti uzģenerēt, pateicoties „ISTehnoloģijas" specifiskajai uzbūvei. Proti, „ISTehnoloģijā" visa informācija, kas saistās ar sistēmas vienumiem, tiek glabāta datu bāzē un attiecīgi interpretēta. Tādējādi datubāzē katram vienumam iespējams pievienot informāciju, kas to raksturo un kuru iespējams iekļaut šajā nodaļā.

6.1. att. **Fragments no pašlaik uzņēmumā izmantotās lietotāja rokasgrāmatas satura**

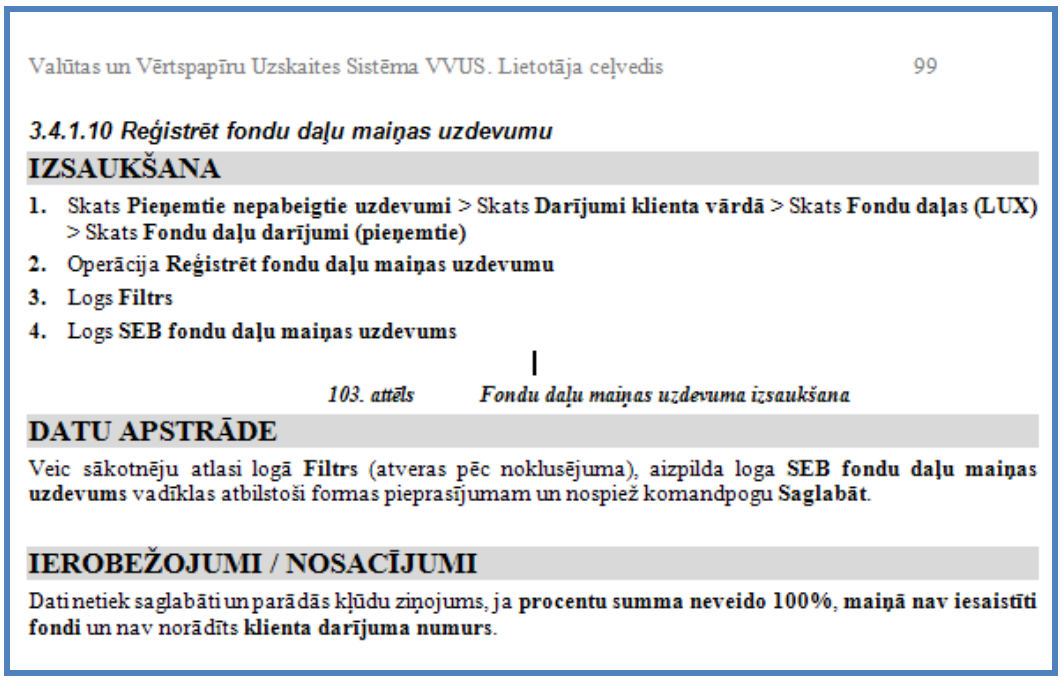
Ja apskatāmies uz pašlaik uzņēmumā izmantoto lietotāja rokasgrāmatas saturu (skat. 6.1. attēlu) un salīdzinām ar „ISTehnoloģijas" skatu un operāciju kokiem (skat. 6.2. attēlu), varam saskatīt acīmredzamas līdzības. Dokumenta satura 1. līmeņa virsraksti atbilst „ISTehnoloģijas" darba vietām, proti, dokumentā katra nodaļa atbilst katrai sistēmā pieejamajai darba vietai. Savukārt 2., 3., 4. utt. (atkarībā no operācijas dziļuma) līmeņa virsraksti atbilst „ISTehnoloģijas" skatu kokam, bet pēdējā līmeņa virsraksti (turpmāk operāciju apraksti) atbilst „ISTehnoloģijas" skatu operācijām. Piemēram, virsraksts *„3.2 Informācija"* atbilst skatam „*Informācija*", kas atrodas vērtspapīru darījumu nodrošināšanas

darba vietā, virsraksts *„3.2.1 Produktu reģistrs”* atbilst skata „*Informācija*” apakšskatam „*Produkti*”, bet virsraksts „*3.2.1.1 Reģistrēt/rediģēt produkta aprakstu*” atbilst operācijai „*Reģistrēt/rediģēt produkta aprakstu*”, kas piesaistīta apakšskata „*Produkti*” objektam. Tas nozīmē, ka nodaļas zemākā līmeņa virsraksti atbilst „ISTehnoloģijas” skata operācijām, augstākā līmeņa virsraksti atbilst sistēmas darba vietām, bet pārējie virsraksti atbilst skatiem.



6.2. att. **„ISTehnoloģijas” skatu un operāciju koki**

Kā redzams, tad dokumenta virsrakstus ir iespējams uzģenerēt no sistēmas, jo tiem ir noteikta struktūra, kas atbilst „ISTehnoloģijai”. Bet kā ir ar saturu katram operācijas aprakstam? Vai to ir iespējams uzģenerēt no sistēmas? Atbilde ir: „Jā!”, jo arī katras operācijas aprakstam ir noteikta struktūra. Par piemēru no dokumenta tiks ņemts operācijas „*Reģistrēt fondu daļu maiņas uzdevumu*” apraksts, kas attēlots 6.3. attēlā.



Valūtas un Vērtspapīru Uzskaites Sistēma VVUS. Lietotāja ceļvedis 99

3.4.1.10 Reģistrēt fondu daļu maiņas uzdevumu

IZSAUKŠANA

1. Skats **Pieņemtie nepabeigtie uzdevumi** > Skats **Darījumi klienta vārdā** > Skats **Fondu daļas (LUX)** > Skats **Fondu daļu darījumi (pieņemtie)**
2. Operācija **Reģistrēt fondu daļu maiņas uzdevumu**
3. Logs **Filtrs**
4. Logs **SEB fondu daļu maiņas uzdevums**

103. attēls Fondu daļu maiņas uzdevuma izsaukšana

DATU APSTRĀDE

Veic sākotnēju atlasi logā **Filtrs** (atveras pēc noklusējuma), aizpilda loga **SEB fondu daļu maiņas uzdevums** vadīklas atbilstoši formas pieprasījumam un nospiež komandpogu **Saglabāt**.

IEROBEŽOJUMI / NOSACĪJUMI

Dati netiek saglabāti un parādās kļūdu ziņojums, ja **procentu summa neveido 100%**, **maiņā nav iesaistīti fondi** un nav norādīts **klienta darījuma numurs**.

6.3. att. **Operācijas „Reģistrēt fondu daļu maiņas uzdevumu" apraksts bez attēla**

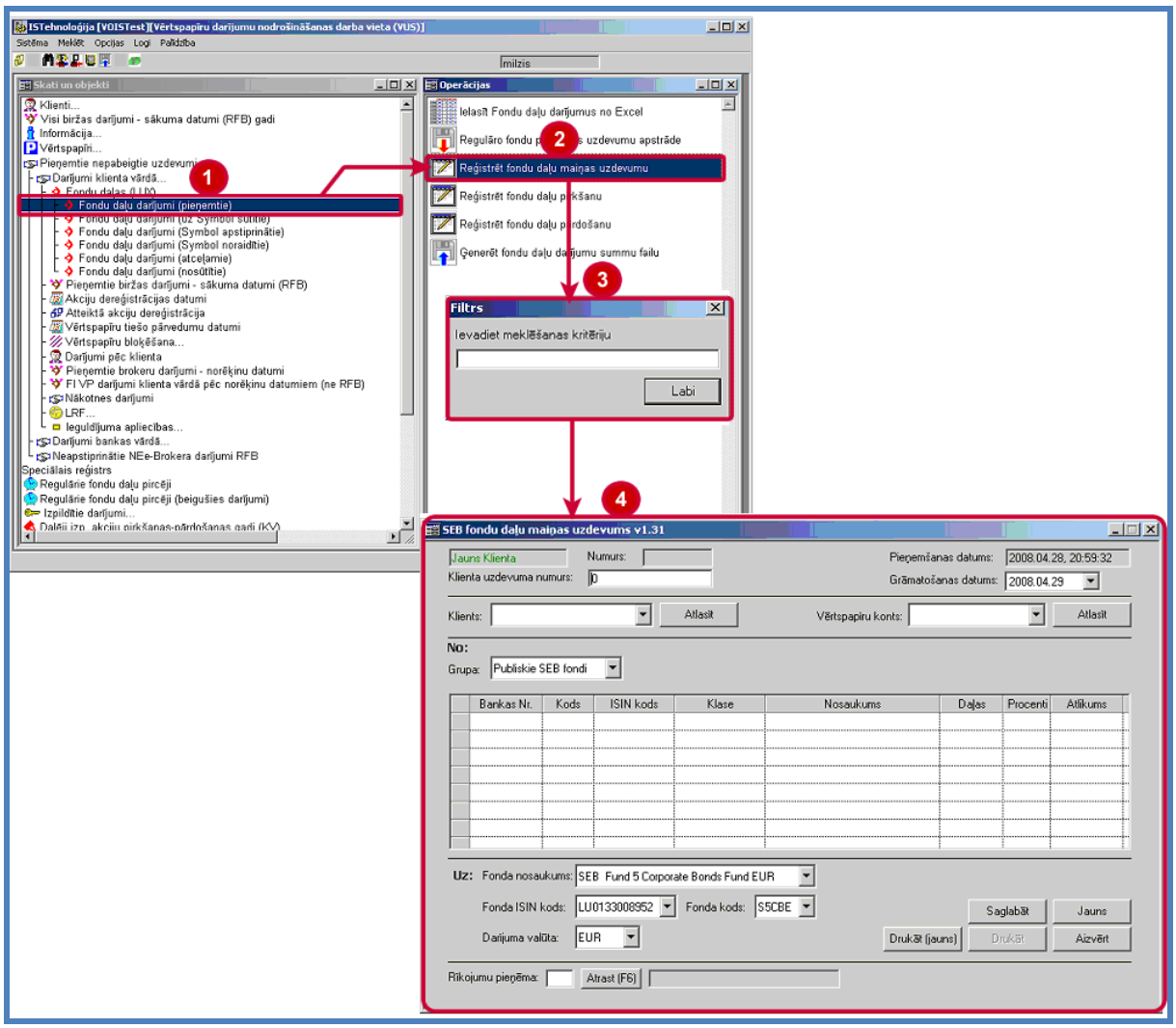
Šajā piemērā redzama operācijas apraksta struktūra (attēls, kur attēlota operācijas izsaukšana, izņemts pārskatāmības dēļ). Apraksts iekļauj šādas 3 sadaļas:

1. Izsaukšana;
2. Datu apstrāde;
3. Ierobežojumi/nosacījumi.

Sadaļā „Izsaukšana" tiek aprakstīts, kā iespējams izsaukt attiecīgo operāciju. Šāda veida informāciju ir iespējams uzģenerēt, jo „ISTehnoloģijā" katra operācija ir kā atsevišķs objekts datu bāzē un šis objekts satur arī informāciju, kā to izsaukt.

Savukārt sadaļas „Datu apstrāde" un „Ierobežojumi/nosacījumi" tiek rakstītas brīvā formā, bez noteiktas struktūras. Tāpēc tieši tekstu nevar uzģenerēt, bet var pie operācijas apraksta pievienot laukus, kas saturēs šo brīvā tekstā ievadīto informāciju.

Papildus tekstuālajai informācijai operācijas apraksts satur arī vienu vai vairākus attēlus (skat. 6.4. attēlā), kas lasītājam sniedz informāciju, kā izsaukt attiecīgo operāciju. Tā, kā šādi attēli tiek iegūti no sistēmas un sistēma arī prot atrast ceļu uz šo operāciju, tad arī šāda veida vizuālu informāciju ir iespējams automatizēti iekļaut dokumentā.



6.4. att. **Operācijas „Reģistrēt fondu daļu maiņas uzdevumu" izsaukšana.**

Tātad, kā noskaidrojām, dokumenta pamattekstu ir iespējams uzģenerēt, bet standarts bez jau apskatītajām nodaļām specificē arī šādas nodaļas:

- Kļūdu situācijas
- Pielikumi
- Atsauces
- Skaidrojošā vārdnīca
- Alfabētiskais priekšmetu rādītājs.

Šīs visas nodaļas ir sistēmu un dokumentu raksturojošas, bet netiek uzglabātas sistēmā. Vienīgā nodaļa, kuru ir iespējams uzģenerēt, ir „Alfabētiskais priekšmetu rādītājs", bet tas ir izdarāms ne tik daudz ar sistēmas palīdzību, kā ar teksta redaktora palīdzību.

6.2.2. Secinājumi

Pārskatot gan LVS standarta ieteikumus, gan arī uzņēmumā pieņemto praksi attiecībā uz lietotāja rokasgrāmatu, var secināt, ka dokumentu ir iespējams uzģenerēt. Protams, ne pilnībā, bet gan lielāko daļu. Tās ir nodaļas, kuras ir ar noteiktu struktūru un nesatur tekstus, kas rakstīti brīvā formā. Lielākais ieguvums ir iespēja uzģenerēt lietotāja rokasgrāmatas pamattekstu, kas dokumenta sastādītājiem aizņem ļoti daudz laika, jo pie lielām sistēmām lietotāja rokasgrāmatas apjoms pārsniedz pat 200 lappuses.

Lietotāju rokasgrāmatai tiks ģenerētas šādas nodaļas:

6.4. tabula

Lietotāja rokasgrāmatas ģenerējamās nodaļas

Nodaļa	**Piezīmes**
Titullapa	
Satura rādītājs	Tiks ģenerēts, izmantojot teksta redaktora iespējas
Ilustrāciju saraksts	
Dokumenta pamatteksts	
Instruktīva veida	Tiks ģenerēti visi operāciju apraksti
Alfabētiskais priekšmetu rādītājs	Tiks ģenerēts, izmantojot teksta redaktora iespējas

6.3. Nododamo vienumu saraksts

Nododamo vienu sarakstam nav noteikta standarta, pēc kura būtu ieteicams vadīties. To, kam jābūt iekļautam nododamo vienumu sarakstā un kādam tam jāizskatās, nosaka pati organizācija un/vai pasūtītājs atkarībā pēc konkrētā produkta vai uzņēmuma standartiem. Attiecībā uz šo dokumentu tiks apskatīta uzņēmumā SIA „Datorikas institūts DIVI" pieņemtā prakse.

6.3.1. Dokumenta saturs

Nododamo vienumu saraksts parasti satur tos vienumus, kas ir izstrādāti noteikta līguma vai vienošanās ietvaros un tiek nodoti pasūtītājam. Pirmajā reizē nododamo vienumu saraksts satur pilnīgi visus dokumentus, bet, ja, piemēram, tiek noslēgta papildus vienošanās, tad nododamo vienumu saraksts satur tos dokumentus, kuri ir jauni vai mainījušies kopš iepriekšējās vienumu nodošanas.

Dokuments satur šādas nodaļas:

1. Titullapa
2. Izmantotie saīsinājumi
3. Nodevumu saturs

4. Dokumenti
5. Programmatūras izejas tekstu datnes
6. Atskaites
7. Programmatūras skripti

Šī dokumenta titullapa nesatur nekādu specifisku informāciju, bet gan uzņēmuma logo, dokumenta nosaukumu , vietu un gadu, kurā izstrādāts dokuments. Pēc titullapas tiek iekļauta versiju lapa, kurā tiek iekļauts dokumenta nosaukums, versija, identifikators un izdošanas datums, un dokumentārā lapa. Šīs trīs lapas ir iespējams uzģenerēt, jo tās satur tehnisku, strukturētu un statisku informāciju.

Nodaļas „Izmantotie saīsinājumi" uzģenerēšana nav iespējama, vismaz ne šī darba ietvaros, jo sistēmā netiek glabāta informācija, kas saistīta ar saīsinājumiem un to atšifrējumiem.

Dokumenta 2. nodaļā „Nodevumu saturs" tiek uzskaitīti nodevumu veidi, kas arī atbilst tālākajām nodaļām. Šāda veida informāciju iespējams uzģenerēt, balstoties uz nākošo nodaļu iekļaušanu vai neiekļaušanu dokumentā.

Tālāk dokumentā seko visi nodevumu veidi, piem., dokumenti, programmatūras izejas tekstu datnes, atskaites un programmatūras skripti. Šīm visām nodaļām ir vienota struktūra. Katra nodaļa satur tabulu, kurā ir saraksts ar visiem attiecīgā veida nodevumiem. Katram nodevumam tiek iekļauts nosaukums, versijas numurs un apraksts. Tomēr ne visus nodevumu veidus iespējams automatizēti uzģenerēt, jo informācija par tiem netiek glabāta sistēmas projektējumā. Vienumu veidi, kurus iespējams iegūt no sistēmas projektējuma, ir programmatūras vienumi un „DIREPO" atskaites, kuras ir piesaistītas kādam no sistēmas projektējumā iekļautajiem skatiem.

6.3.2. Secinājumi

Nododamo vienumu saraksts ir salīdzinoši neliels dokuments ar ļoti strukturētu informāciju, kura tiek glabāta datu bāzē, tāpēc šo dokumentu ir iespējams daļēji uzģenerēt. Pēc dokumenta uzģenerēšanas cilvēkam būs jāpievieno nodaļa „Izmantotie saīsinājumi", kas nav obligāta un dažos gadījumos var netikt iekļauta, un nodaļas, kurās uzskaitīti nododamie dokumenti un programmatūras skripti.

Nododamo vienumu sarakstam tiks ģenerētas šādas nodaļas:

6.5. tabula

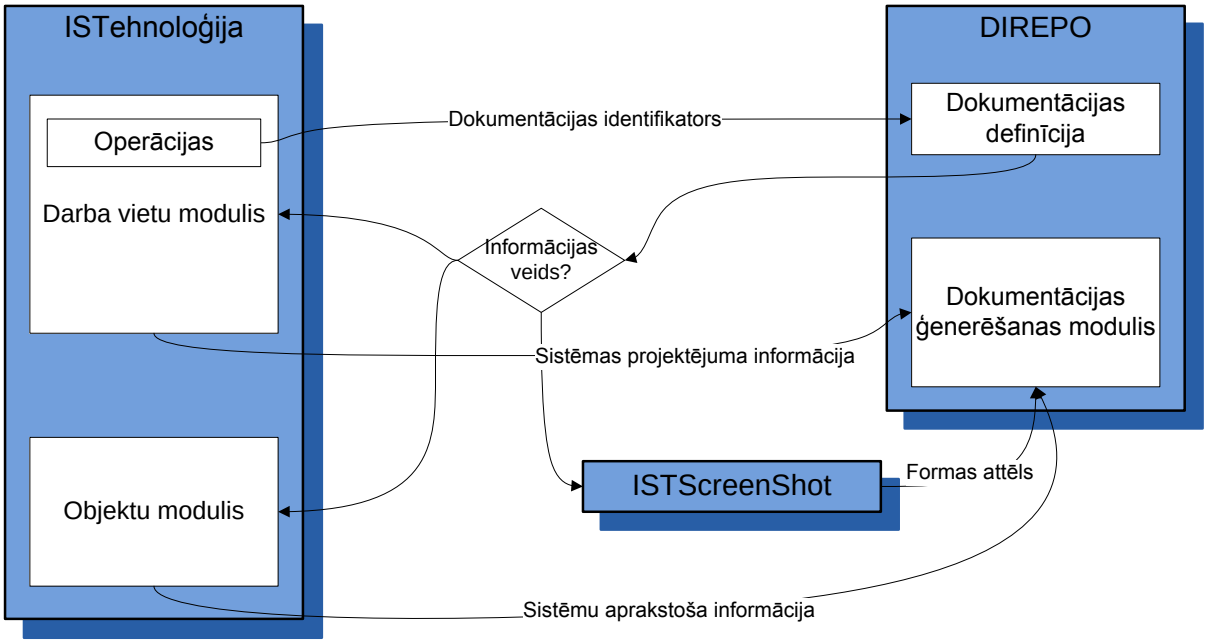
Nododamo vienumu saraksta ģenerējamās nodaļas

Nodaļa	**Piezīmes**
Titullapa	
Nodevumu saturs	Uzskaita tālāk sekojošās nodaļas
Programmatūras izejas tekstu datnes	Tiek iekļautas atkarībā no sistēmas
Atskaites	

7. DOKUMENTĀCIJAS ĢENERĒŠANA „ISTEHNOLOĢIJAI”

Kā iepriekš tika noskaidrots, tad dokumentācijas ģeneratoru „ISTehnoloģijai” visērtāk ir veidot, balstoties uz modeļu bāzētās arhitektūras principiem. Tādēļ ir nepieciešamas vismaz trīs lietas, kas MBA kontekstā ir PNM (modelis, no kā tiek ņemta informācija), PSM (modelis, kurā tiek likta informācija) un transformācija, bet dokumentācijas ģeneratora kontekstā tās ir attiecīgi:

1. **Sistēmas modelis** - „ISTehnoloģija”, kurai nepieciešams atjaunot objektu moduli, kas glabā informāciju par sistēmu, un nedaudz pielāgot darba vietu moduli;
2. **Dokumenta modelis** - Atskaišu definēšanas un ģenerēšanas rīks „DIREPO”, kas nepieciešamības gadījumā jāpielāgo dokumentācijas ģenerēšanai, jo līdz šim šis rīks tika izmantots tikai atskaišu ģenerēšanai;
3. **Transformācija** - Nepieciešams noskaidrot, kā iegūt dokumentācijai nepieciešamo informāciju no „ISTehnoloģijas” (gan tekstu, gan attēlus) un kā, izmantojot šo informāciju, ar „DIREPO” palīdzību katram dokumentam izveidot savu atskaites definēšanas šablonu.



7.1. att. **Automatizēta dokumentācijas ģeneratora „ISTehnoloģijai” uzbūve**

7.1. attēlā redzama automatizēta dokumentācijas ģeneratora „ISTehnoloģijai” vispārīga shēma. Šajā shēmā ar tumšāku taisnstūri apzīmētas neatkarīgās lietojumprogrammas, bet ar baltajiem taisnstūriem – attiecīgās lietojumprogrammas daļas, kas tieši tiek izmantotas

dokumentācijas ģenerēšanas procesā. Ar bultām attēlota informācijas plūsma. Šajā shēmā redzams, ka dokumentācijas ģenerēšana sākas no „ISTehnoloģijas" operācijas, kuru izsaucot, „DIREPO" tiek padots dokumentācijas definīcijas identifikators. Tālāk atkarībā no atskaites definīcijas no „ISTehnoloģijas" tiek vairākkārtīgi pieprasīta informācija. Dažādi informācijas veidi tiek iegūti no dažādiem avotiem. Informācija, kas saistās ar tehnisko sistēmas uzbūvi, piemēram, datu bāzi, lietojumprogrammām utt., tiek iegūta no „ISTehnoloģijas" objektu moduļa, bet informācija, kas saistās ar biznesa loģiku, tiek iegūta no darba vietu moduļa. Savukārt „ISTehnoloģijas" formu attēlus dokumentācijas ģenerators iegūst ar rīka „ISTScreenShot" palīdzību.

Turpmākajās trīs apakšnodaļās aprakstīts, kā realizējams sistēmas un dokumenta modelis, kā arī transformācija starp tiem.

7.1. „ISTehnoloģijas" pielāgošana dokumentācijas ģenerēšanai

„ISTehnoloģijai" dokumentācijas ģeneratora modelī ir informācijas avota loma. Tas nozīmē, ka dokumentācijas ģenerators pilnībā visu informāciju, kas nepieciešama dokumentācijai, iegūs no „ISTehnoloģijas". „ISTehnoloģija" līdz šim nav tikusi lietota dokumentācijas ģenerēšanai, tāpēc tās metamodelī nepieciešamas izmaiņas. Ja darba vietu moduli nepieciešams tikai papildināt, tad objektu modulis ir pilnībā jāatjauno, balstoties uz vēsturiski izzudušā moduļa bāzes.

7.1.1. Darba vietu moduļa pielāgošana

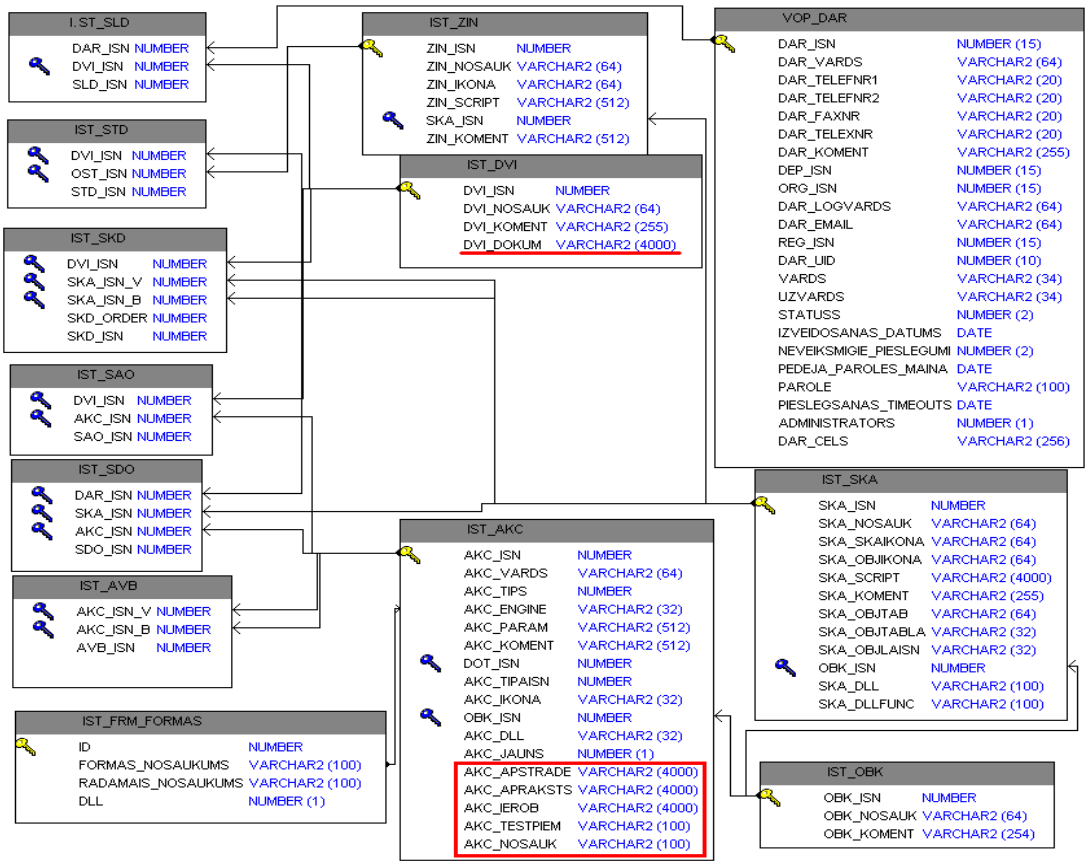
„ISTehnoloģijas" darba vietu moduļa pielāgošana nepieciešama, lai dokumentācijas ģenerators varētu iegūt informāciju dokumentācijas uzģenerēšanai. Darba vietu modulis (skat. 3.3. att.ēlā) iekļauj darba vietas, skatus un operācijas, uz kuriem pārsvarā balstās pati sistēma un no kuriem tiek ņemta informācija dokumentācijas sastādīšanai. Tomēr ar pašlaik modulī iekļauto informāciju nepietiek, lai uzģenerētu pilnīgas dokumentācijas, jo trūkst apraksta cilvēku valodā. Tādēļ 3.3. att.ēlā redzamajai entītijai „Darba vieta" tiks pievienots atribūts „*Dokumentācijas teksts*" un entītijai „Operācija" tiks pievienoti šādi atribūti: „*Formas nosaukums*", „*Formas apraksts*", „*Formas apstrāde*", „*Formas ierobežojumi/Nosacījumi*" un „*Testpiemēra nosaukums*".

Atribūts „*Dokumentācijas teksts*" aprakstīs darba vietu cilvēku valodā. Piemēram, ja sistēmā tiek piereģistrēta jauna darba vieta „*Redaktora darba vieta*", tad šajā atribūtā var norādīt šādu dokumentācijai nepieciešamu tekstu: „*Šī darba vieta domātā priekš lietotājiem, kas veic dažādus rediģēšanas darbus, bet ne administrēšanas darbus.*".

Operācijas raksturojošie atribūti nepieciešami, lai aprakstītu operācijas izsaukto formu. Gan lietotāja rokasgrāmatā, gan nododamo vienumu sarakstā nepieciešama informācija, kas apraksta operācijas izsaukto formu. Šo informāciju nodrošina atribūti:

- „*Formas nosaukums*" - satur formas augšpusē esošo nosaukumu;
- *„Formas apraksts"* – formāli apraksta formas nepieciešamību un pielietojamību;
- *„Formas apstrāde"* – formāli apraksta, kā notiek formas apstrāde. Šī informācija tiek neizmainīta ievietota lietotāja rokasgrāmatas attiecīgajā apakšnodaļā;
- *„Formas ierobežojumi/Nosacījumi"* – formāli apraksta, kādi ir ierobežojumi formas aizpildē un apstrādē, kā arī, kādiem nosacījumiem jāizpildās formas veiksmīgai pielietošanai. Šī informācija tiek neizmainīta ievietota lietotāja rokasgrāmatas attiecīgajā apakšnodaļā.

Savukārt atribūts „*Testpiemēra nosaukums*" nepieciešams dokumentācijas ģenerēšanas procesā, jo, izmantojot šajā atribūtā iekļauto informāciju, tiks iegūti attēli un pēc tam ievietoti dokumentā.



7.2. att. **Daļa no „ISTehnoloģijas" darba vietu moduļa detalizētās ER diagrammas**

Šādā veidā formāli, cilvēku valodā aprakstot gan operācijas, gan darba vietas, dokumentācijas ģenerēšanas laikā iespējams iegūt nosacīti precīzu informāciju par katru objektu. 7.2. attēlā redzama daļa no „ISTehnoloģijas" darba vietu moduļa detalizētās ER diagrammas, kurā atzīmēti pievienotie atribūti.

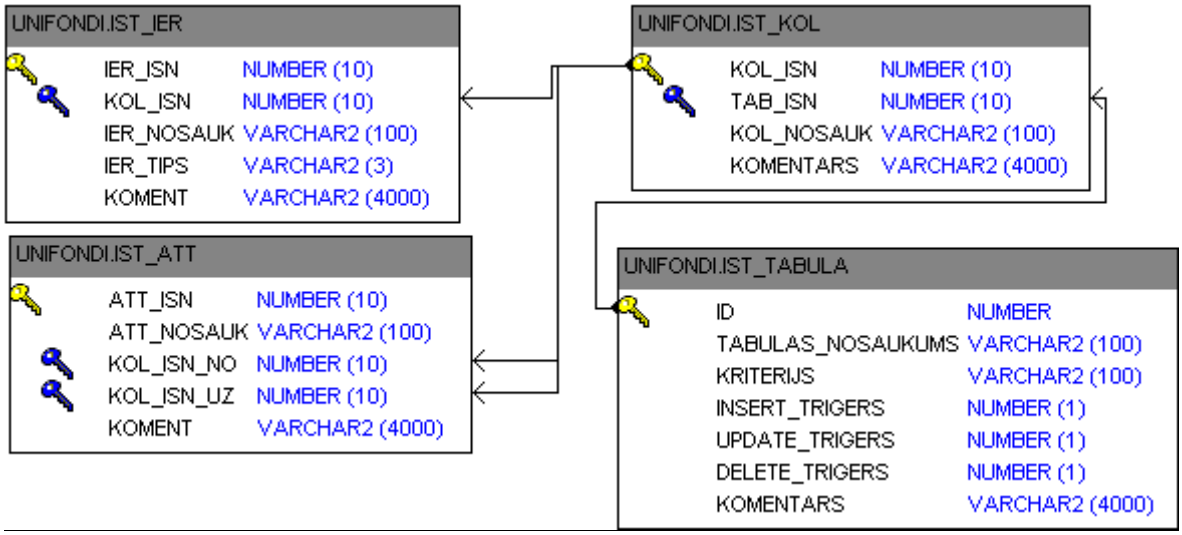
Kā iepriekš tika minēts, tad šādi atribūti nodrošina informācijas avotu dokumentācijas ģeneratoram. Tomēr tas nav vienīgais veids, kā izmantot šos atribūtus. Tos var izmantot arī kā informācijas avotu sistēmas palīdzības (angl. Help) sastādīšanai. Jo ar šiem atribūtiem par darba vietām, operācijām un formām būtu pieejama informācija, ar kuru lietotājs varētu iepazīties, strādājot ar sistēmu.

7.1.2. „ISTehnoloģijas" objektu moduļa atjaunošana

Nodaļā „„ISTehnoloģijas" apskats" kā viens no sistēmas moduļiem tika apskatīts arī pašlaik neizmantotais objektu modulis, kas ļoti labi aprakstīja sistēmu. Ar šī moduļa palīdzību bija iespējams katrai tabulai, kolonnai, attiecībai, ierobežojumam utt. pievienot to aprakstošu informāciju cilvēku valodā. Šāda veida sistēmas apraksts ļoti noder tādas zema līmeņa dokumentācijas ģenerēšanai kā datu bāzu projektējums. Tāpēc dokumentācijas ģeneratorā šis modulis pildīs nevis objektu definēšanas funkciju, bet gan sistēmas aprakstīšanas funkciju.

Ņemot vērā to, ka šis modulis tika veidots pirms vairāk nekā 10 gadiem un kopš tā laika nav būtiski mainījies, tam ir nepieciešamas būtiskas izmaiņas, kas spētu pielāgot šo moduli mūsdienu datoru resursiem, programmēšanas tehnoloģijām un, protams, arī dokumentācijas ģenerēšanai.

Lai būtu vieglāk izprast jaunā objektu moduļa struktūru, 7.3. attēlā parādīta tā detalizētā ER diagramma. Atšķirībā no vecā „ISTehnoloģijas" objektu moduļa jaunais objektu modulis vairāk balstīts uz sistēmas aprakstīšanu, nevis uz objektu aprakstīšanu. Līdz šim „ISTehnoloģijas" datu bāzē bija saglabājusies tikai tabula „IST_TABULA". Šīs tabulas uzdevums ir nodrošināt tabulas trigeru konfigurēšanu (norāda, kādus trigerus ieslēgt, kādus - izslēgt). Tā kā šī tabula satur visu sistēmas tabulu nosaukumus, tad tā tika ņemta par pamatu, veidojot jauno objektu moduli. Tā rezultātā šai tabulai tika pievienots atribūts „Komentārs", kurā tiek glabāts formāls tabulas apraksts.



7.3. att. **Jaunā objektu moduļa detalizētā ER diagramma**

Objektu modulis apraksta datu bāzes tabulas un to attiecības, tāpēc loģiski ir tas, ka tajā tiek iekļautas arī šādas tabulas:

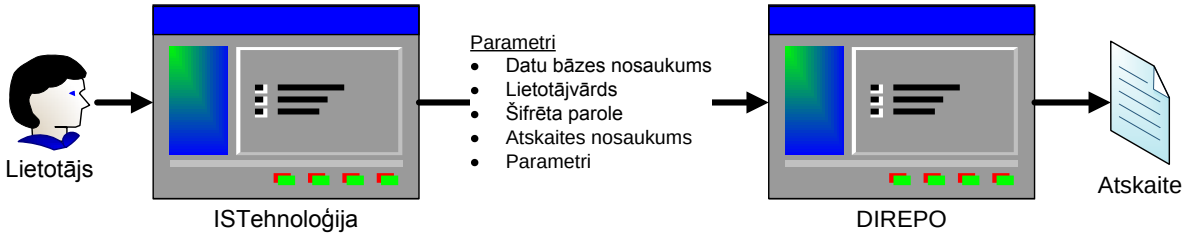
- „IST_KOL" – tiek glabātas visas tabulu kolonnas;
- „IST_IER" – tiek glabāti ierobežojumi (piem., primārās, unikālās atslēgas u.c.);
- „IST_ATT" – tiek glabātas attiecības starp tabulām.

Ņemot vērā, ka datu bāzu projektējumā jāiekļauj tabulu, kolonnu, ierobežojumu un citu datu bāzes aprakstošu informāciju, tad objektu modulis ir veidots kā datu bāzes metamodelis. Datu bāzu pārvaldības sistēmās, nodefinējot datu bāzi ar visām tabulām, kolonnām, ierobežojumiem un saitēm, visa informācija arī tiek glabāta datu bāzes metamodelī jeb sistēmas tabulās. Tādēļ būtībā sanāk, ka objektu modulis daļēji dublē informāciju, kas ir pieejama datu bāzes pārvaldības sistēmas tabulās, tikai objektu modulim katrai tabulai ir atribūts „Komentārs", kas cilvēku valodā apraksta attiecīgo datu bāzes objektu. Iemesls šādai informācijas dublēšanai ir lietotāju tiesību ierobežojumi, kas varētu neļaut lietotājam iegūt informāciju no datu bāzu pārvaldības sistēmas tabulām.

Faktiski no objektu modulī un datu bāzu pārvaldības sistēmā pieejamās informācijas iespējams uzģenerēt gandrīz visu datu bāzu projektējumā iekļaujamo pamatinformāciju. Tomēr dokumentācijas ģenerēšanā objektu modulis nedos nekādu ieguldījumu, ja tajā iekļauto informāciju regulāri neatjaunos. Tāpēc pēc būtiskām izmaiņām sistēmas projektējumā, nepieciešams atjaunot informāciju objektu modulī un datu bāzu pārvaldības sistēmā, lai uzģenerēta dokumentācija būtu tikpat aktuāla kā sistēmas projektējums.

7.2. „DIREPO” pielāgošana dokumentācijas ģenerēšanai

Jau pašlaik „DIREPO” nodrošina savietojamību ar „ISTehnoloģiju”, t.i., no „ISTehnoloģijas”, izmantojot operācijas, iespējams izsaukt „DIREPO” atskaiti, kas iepriekš ir nodefinēta šajā rīkā.



7.4. att. **„DIREPO” atskaites izsaukšana no „ISTehnoloģijas”**

Balstoties uz šo „ISTehnoloģijas” funkcionalitāti, tiks veikta dokumentācijas ģeneratora transformēšanas daļa. Tomēr starp dokumentācijas ģenerēšanu un atskaišu ģenerēšanu ir nelielas atšķirības. Tāpēc šajā apakšnodaļā tiks apskatīts, kādas atšķirības ir starp atskaišu un dokumentācijas ģenerēšanu, kā arī rasti risinājumi „DIREPO” pielāgošanai dokumentācijas ģenerēšanai.

7.2.1. Atšķirības starp atskaišu un dokumentācijas ģenerēšanu „DIREPO”

Galvenās atšķirības starp atskaišu un dokumentācijas ģenerēšanu jeb funkcijas, kuras līdz šim „DIREPO” nenodrošināja un nepieciešams pievienot, ir šādas:

- Dokumentācijā nepieciešams iekļaut dažādus ģenerēšanas laikā veidotus attēlus, nevis izmantot jau gatavus iepriekš noteiktus attēlus;
- Atskaitēm parasti netiek veidotas atsevišķas lapas un nodaļas;
- Atskaitēm netiek veidoti satura rādītāji, jo tās parasti nav tik liela apjoma kā dokumentācijas.
- Dokumentācijās visas nodaļas tiek numurētas, izmantojot automātisko numurēšanu.

7.2.2. Risinājumi

No iepriekšējā apakšnodaļā aprakstītajām atšķirībām jeb lietām, kas varētu neļaut uzģenerēt pilnīgu dokumentāciju ar atskaišu ģeneratora palīdzību, visbūtiskākā ir attēlu iegūšana dokumentācijas ģenerēšanas laikā. Šim nolūkam tika izstrādāts atsevišķs rīks „ISTScreenShot”, kas nodrošina šo funkciju un ar kuru detalizētāk var iepazīties 7.3.1. apakšnodaļā „ISTScreenShot”.

Lai „DIREPO” iegūtu „ISTScreenShot” izveidoto attēlu, „DIREPO” atskaišu definēšanas valodā tika izveidota jauna universāla globālā funkcija:

@CallApp(appPath, appPars)

, kur *appPath* ir ceļš līdz izsaucamajai lietojumprogrammai un *appPars* ir norādītās lietojumprogrammas parametru simbolu virkne, kurā katrs parametrs atdalīts ar atstarpi. Ar šīs funkcijas palīdzību iespējams izsaukt jebkuru lietojumprogrammas izpilddatni, padodot tai nepieciešamos parametrus. „DIREPO” gaida līdz izsauktās lietojumprogrammas darbības beigām un tikai tad turpina tālāk ģenerēt dokumentu. Šajā gadījumā kā lietojumprogramma jānorāda „ISTScreenShot”, bet kā parametrs jāpadod fotografējamās „ISTehnoloģijas” formas testpiemēra identifikators. Šī funkcija izsauc „ISTScreenShot”, kas izpilda norādīto testpiemēru un saglabā formas attēlu noteiktā mapē. Šī mape atrodas tajā pašā mapē, kurā atrodas pats rīks. Tālāk, izmantojot testpiemēra identifikatoru, iespējams identificēt izveidoto attēlu un ievietot dokumentā tāpat, kā tas tiek darīts atskaišu ģenerēšanas gadījumā.

Nākošā problēma, ar kuru jāsaskaras, pielāgojot „DIREPO” dokumentācijas ģenerēšanai, ir dokumentu ģenerēšana uz vairākām lapām. Dokumentācijas parasti sastāv no titullapas, satura rādītāja, ievada un citām nodaļām, no kurām katra tiek veidota uz atsevišķas lapas un kuru struktūras būtiski atšķiras. Šai problēmai tika atrasts salīdzinoši vienkāršs risinājums. „DIREPO” atskaites definēšana notiek pa rindām un kolonnām. Tas nozīmē, ka katrai rindai dokumentā atbilst viena rinda atskaites definīcijā. Tātad, ja dokumentā nepieciešama jauna lapa, tad vienkārši rindas īpašībās jānorāda parametrs „*Lappuses pārtraukums*”, kas nozīmē, ka rinda tiks pārnesta jaunā lapā.

Liela apjoma dokumentācijās neatņemama sastāvdaļa ir satura rādītājs. Satura rādītāja izveidošanas funkciju nodrošina pats teksta redaktors, piem., Microsoft Word vai OpenOffice Writer, bet, lai to izdarītu, nepieciešams katra līmeņa nodaļām veidot vienotu stilu. „DIREPO” ir iespēja veidot šādus vienotus stilus, tomēr tas joprojām neatrisina satura rādītāja ģenerēšanas problēmu, jo „DIREPO” katru atskaites definīcijas rindu ģenerē secīgi vienu pēc otras. Tas nozīmē, ja satura rādītājs (kā tas parasti arī ir) atrodas pirms nodaļām, kuras tajā tiek iekļautas, tad atskaišu ģenerators, nonākot pie satura rādītāja veidošanas rindas, vēl nezinās, kādas būs nākošās nodaļas, jo tās vēl nav izveidotas, un attēlos tukšu satura rādītāju. Tāpēc tika izstrādāta vēl viena funkcija bez parametriem:

@InsertContents()

, pēc kuras izsaukšanas atskaišu ģenerators atliek tās izpildi, līdz tiek uzģenerēta pēdējā dokumenta rinda. Tiklīdz tas tiek izdarīts, tiek izpildīta šī funkcija, kas, izmantojot teksta

redaktora iespējas, no uzģenerētā dokumenta nodaļām uzģenerē satura rādītāju, kuru ievieto attiecīgajā rindā.

„DIREPO" nodrošina iespēju veidot vienotus stilus, kurus būtu iespējams izmantot satura rādītāja veidošanā, tomēr, definējot šos stilus, nav iespējams uzstādīt automātisko numurēšanu. Nepietiek ar to, ka automātiskā numurēšana tiek pievienota vienam noteiktam stilam. Nepieciešama iespēja definēt arī numurēšanas apakšlīmeņus. Tas nozīmē, ka katram stilam jābūt iespējai nodefinēt apakšlīmeņa stilu, t.i., ja dokumentā parādās pirmā līmeņa stils ar numurēšanu un pēc tam 2. līmeņa stils ar numurēšanu, tad arī numurēšanai jābūt otrajā līmenī.

7.3.„ISTehnoloģijas" – „DIREPO" transformācija dokumentācijas ģenerēšanai

Dokumentācijas ģeneratora transformācijas daļā ietilpst viss, kas atrodas starp „ISTehnoloģiju" un „DIREPO". Lai notiktu veiksmīga „ISTehnoloģijā" pieejamās informācijas transformācija uz „DIREPO" un pēc tam uz dokumentu, nepieciešams formu attēlu fotografēšanas rīks, kas spētu atrast noteiktu „ISTehnoloģijas" formu un pēc tam nofotografētu. Tāpēc šajā apakšnodaļā tiks apskatīts šim nolūkam izstrādātais rīks „ISTScreenShot". Tomēr ar šī rīka palīdzību nepietiek, nepieciešams arī izstrādāt „DIREPO" atskaišu šablonus katrai dokumentācijai, lai „DIREPO" spētu „ISTehnoloģijā" pieejamo informāciju pārvērst dokumentācijā.

7.3.1. „ISTScreenShot"

„ISTScreenShot" ir dokumentācijas ģenerēšanai pielāgots „Test Automation FX" rīks. „Test Automation FX" ir lietotāja saskarnes testēšanas rīks, kas ļauj testētājam ierakstīt un pārvaldīt lietotāja saskarnes testus. Šī rīka galvenā priekšrocība ir tā, ka tas atdala lietotāja saskarni no testēšanas loģikas un ģenerē .NET kodu, kuru pēc tam testētājs var pielāgot savām vajadzībām [5.9.].

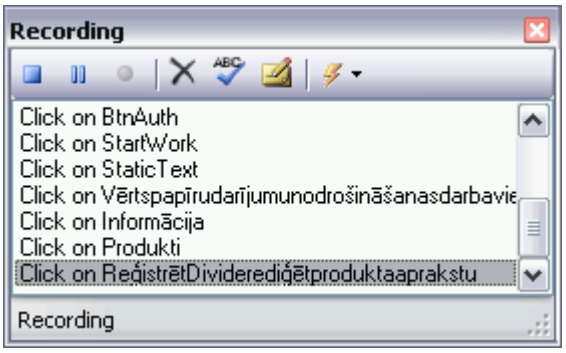
Dokumentācijas ģenerēšanai šis rīks tiek izmantots nevis testēšanai, bet „ISTehnoloģijas" formu attēlu iegūšanai, jo tas nodrošina noteiktas formas atvēršanu un arī atsevišķu lietotāja saskarnes elementu attēlu saglabāšanu.

Lai šo testēšanas rīku būtu iespējams izmantot „ISTehnoloģijas" ekrāna formu attēlu iegūšanai un pēc tam ievietošanai dokumentā, vispirms nepieciešams nodefinēt nepieciešamās formas izsaukumu. Piemēram, ja dokumentācijā nepieciešams ievietot „ISTehnoloģijas" skata „*Informācija*" apakšskata „*Produkti*" operācijas „*Reģistrēt/rediģēt produkta aprakstu*" formas

attēlu, tad šīs formas izsaukums ir jāieraksta rīkā „ISTScreenShot" kā testpiemērs. Šādi jādara katrai formai, kuru nepieciešams nofotografēt.

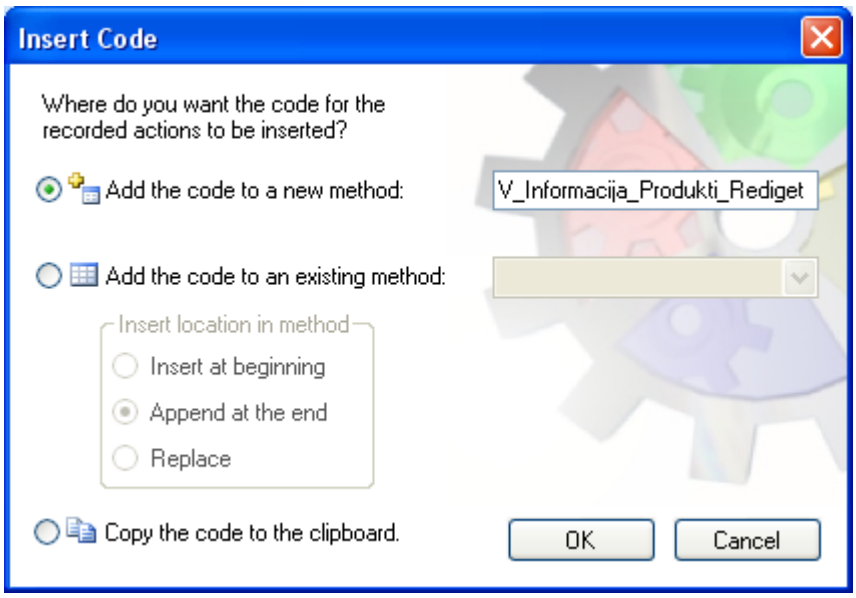
7.3.1.1. Testpiemēra definēšana un izsaukšana formas attēla iegūšanai

Šajā apakšnodaļā tiks aprakstīts, kā jau esošam projektam pievienot jaunu testpiemēru, ar kuru iespējams iegūt formas attēlu. Testpiemērs tiks veidots „ISTehnoloģijas" skata „*Informācija -> Produkti*" operācijas „*Reģistrēt/rediģēt produkta aprakstu*" izsauktajai formai. Ja ir vēlme iepazīties ar Visual Studio noklusētajā testēšanas projektā iekļautajām datnēm, kuras tika pielāgotas „ISTehnoloģijas" formu attēlu iegūšanai, tās ir pievienotas 2. pielikumā.



7.5. att. **Test Automation FX lietotāja saskarnes darbību ierakstītājs**

Vispirms, izmantojot Microsoft Visual Studio programmēšanas vidi, nepieciešams ierakstīt darbības (skat. 7.5. attēlā), kā atvērt formu, kurai nepieciešams iegūt attēlu. Kad ir atvērta vajadzīgā forma, tad nepieciešams iziet ārā no „ISTehnoloģijas", lai pēc testa beigām uz darba stacijas nepaliktu liekas darbojošās lietojumprogrammas. Pēc šo darbību veikšanas jāpārbauda, vai gadījumā nav ierakstījušās liekas darbības, kuras nav nepieciešamas formas izsaukumā, un jāpārtrauc ierakstīšana, nospiežot pogu „*Stop*". To izdarot, parādīsies logs, kurā jāizvēlas sadaļa „*Add the code to a new method*" un jāievada testpiemēra identifikators:



7.6. att. **„Test Automation FX" testpiemēra saglabāšana**

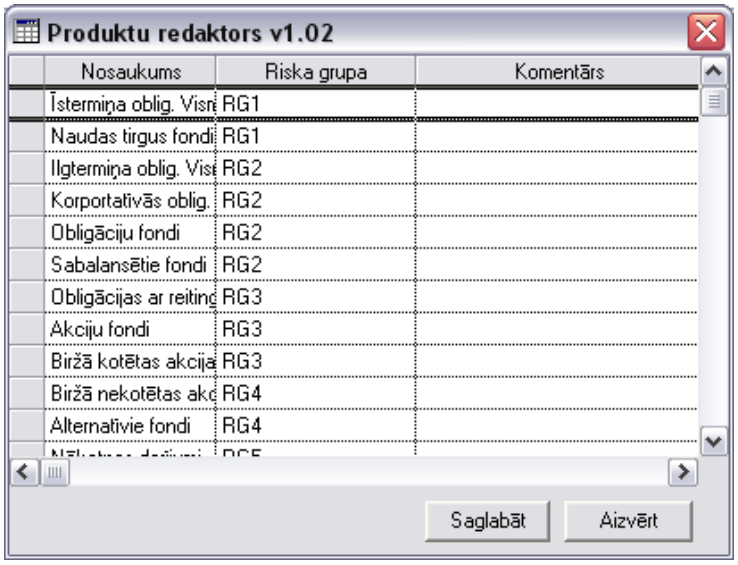
Pēc iepriekšminēto darbību veikšanas ir veikta formas atvēršanas scenārija ierakstīšana, bet vēl nenotiek formas attēla fotografēšana un saglabāšana. Tāpēc tiks izmantota iespēja rīkā „Test Automation FX" mainīt testpiemēra kodu. Formas izsaukšanas scenārija ierakstītājs ir automātiski uzģenerējis kodu, kas redzams 7.7. attēlā bez pelēkā fona. Lietojumsaskarnes rīks katram objektam veic arī fotografēšanu. Tas nozīmē, ka katram objektam (formas, pogas, rīkjoslas un citi lietotāja saskarnes elementi), kas piedalās formas izsaukšanas scenārijā, ir savs attēls. Šajā gadījumā ir nepieciešams tikai viens attēls – izsaucamās formas attēls. Tāpēc Microsoft Visual Studio programmēšanas vidē formas „ISTScreenShot.cs" dizaina skatā attēlotajā objektu kokā jāatrod fotografējamās formas objekts, kas šajā piemērā ir „Produkturedaktorsv102Window". Tālāk funkcijā „VDNDV_Informacija_Produkti_Rediget" šim objektam jāizsauc metode, kas saglabā šī objekta attēlu (skat. 7.7. attēlā rindu ar pelēko fonu). Kā bildes saglabāšanas ceļu jānorāda mape „Bildes", un bildes nosaukumam jābūt vienādam ar testpiemēra nosaukumu, lai pēc tam „DIREPO varētu šo bildi atrast.

```
[UITest()]
public void VDNDV_Informacija_Produkti_Rediget()
{
    ISTexe.DoubleClick(30, 27);
    StaticText.Click(240, 5);
    LBVOP_SEB_VVUSizst_20081205INI.Click(149, 2);
    BtnOK.Click();
    Username.Click(13, 12);
    Keyboard.SendKeys("admin[TAB]");
    Keyboard.SendKeys("admin[TAB]");
    BtnAuth.Click();
    StartWork.Click();
    Informācija.Click(8, 8);
    Produkti.Click(47, 6);
    ReģistrētDivideredīģētproduktaaprakstu.DoubleClick(75, 21);
    Produkturedaktorsv102Window.SaveImage("Bildes\\VDNDV_Informacija_Produkti_Rediget.bmp");
    ButtonAizvērt.Click();
}
```

7.7. att. **„Test Automation FX" testpiemēra definēšanas pirmkods**

7.3.1.2. Testpiemēra izsaukšana

„Test Automation FX" visus testpiemērus nokompilē un ievieto vienā izpilddatnē ar nosaukumu „*ISTScreenShot.exe*". Šo izpilddatni iespējams izsaukt, padodot izsaucamā testpiemēra identifikatoru kā parametru (piem., *ISTScreenShot.exe VDNDV_Informacija_Produkti_Rediget*). Tā rezultātā mapē „Bildes" tiek saglabāts formas attēls:



7.8. att. **„ISTScreenShot" nofotografētais formas attēls**

Tāda pati testpiemēra izsaukšana notiek arī „DIREPO". Lai iegūtu kādu no „ISTehnoloģijas" formu attēliem, izpilddatnei *„ISTScreenShot.exe"* tas nosūta formas izsaukšanas testpiemēra identifikatoru ar *@CallApp* funkcijas palīdzību.

7.3.2. Dokumentu šabloni

Katram dokumentācijas veidam ir sava struktūra un informācija, kuru nepieciešams iekļaut dokumentā. Tāpēc tālāk tiks aprakstīts, kā „DIREPO" iespējams nodefinēt atskaiti, kas, atlasot informāciju no „ISTehnoloģijas" modeļa, spēj uzģenerēt dokumentāciju. Šajā apakšnodaļā tiks apskatītas tikai tās dokumentācijas daļas, kuras līdz šim ir izdevies uzģenerēt.

Svarīgākā atskaites definīcijas daļa tiks attēlota tabulā, kur katra tabulas rinda atbilst vienai rindai dokumentā. Šajā tabulā tiks attēlota tekstuāla informācija un datu bāzes pieprasījumi, bet netiks attēlots katras rindas formatējums un tukšās rindas. Tas nozīmē, ja tabulas numerācijā iztrūkst kāds skaitlis, tad tā rinda ir tukša un veido dokumentā tukšu rindu. Svarīgāko rindu definīcijas tiks paskaidrotas detalizētāk.

Rīks „DIREPO" iekļauj arī iespēju veidot ciklus, tāpēc rindas, kas atrodas cikla nosacījumā un ķermenī, tiks attēlotas ar nelielu atstarpi no kreisās malas. Cikla nosacījuma rinda tiks iezīmēta treknrakstā, lai lasītājam būtu vieglāk orientēties. Ja kādā rindiņā parādās mainīgais *:blockpar[i]*, kur *i* var būt no *1* līdz *n* atkarībā no ciklu skaita, tad šis mainīgais atgriež to vērtību, kas atlasīta pieprasījuma, kas atrodas cikla nosacījumā, pirmajā kolonnā. Piemēram, ja atskaites definīcijā tiek nodefinēts otrs cikls un cikla nosacījumā ir pieprasījums, kas pirmajā kolonnā atgriež kādu vērtību, tad šo vērtību cikla ķermenī iespējams iegūt ar mainīgā *:blockpar2* palīdzību.

7.3.2.1. Datu bāzu projektējuma automatizēta ģenerēšana

Datu bāzu projektējumā lielākā daļa informācijas tiek ņemta no datu bāzes pārvaldības sistēmas tabulām. Tā ir informācija par „ISTehnoloģijas" tabulām, kolonnām, primārajām atslēgām, unikālajām atslēgām, pārbaudēm un saitēm starp tabulām. Šo visu datu bāzes objektu apraksts tiek ņemts no „ISTehnoloģijas" objektu moduļa, jo sistēmas tabulas nesatur formālu informāciju cilvēku valodā.

7.1. tabulā redzama datu bāzu projektējuma definīcija „DIREPO" vidē. Sākumā dokumentā nepieciešams iekļaut visu datu bāzes tabulu sarakstu, kur katrai tabulai pievienots kolonnu skaits un formāls apraksts, kurā tiek paskaidrots, kādam nolūkam tiek izmantota attiecīgā tabula. Tāpēc 5. rindā vispirms tiek izdrukāts tabulas virsraksts, bet pēc tam 6. rindā – tiek izpildīts pieprasījums, kas no datu bāzu pārvaldības sistēmas un objektu piemēru moduļa tabulām atlasa nepieciešamo informāciju.

Tālāk seko datu bāzes tabulu detalizēts apraksts, tāpēc nepieciešams cikls (10. rinda), kas apstaigā iepriekšējā tabulā uzskaitītās tabulas, kuras tiek ņemtas no objektu moduļa tabulas „IST_TABULA”. 16. rindā tiek izdrukātas visas tabulas kolonnas, to datu tipi, obligātums un apraksts. 21. rinda izvada primārās atslēgas, unikālās atslēgas un pārbaudes, ja tādas eksistē. Savukārt 26. un 31. rindā tiek drukāta informācija par saitēm starp tabulām, attiecīgi - tabulas, uz kurām ir saite, un tabulas, kurām ir saite uz attiecīgo tabulu. Visās šajās rindās pamatinformācija tiek iegūta datu bāzu pārvaldības sistēmas tabulām, bet formāls apraksts, kas tiek ievietots komentāru kolonnā, tiek iegūts no objektu moduļa atribūtiem.

7.1. tabula

Datu bāzu projektējuma definīcija „DIREPO” vidē

Nr.	Datu avots	
1	Datu bāzes apraksts	
3	Sistēmas datu bāze sastāv no šādām tabulām:	
5	Tabulas nosaukums;Kolonnu skaits;Komentārs	
6	select t.tabulas_nosaukums, count(k.column_name), t.komentars from ist_tabula t, user_tab_cols k where t.tabulas_nosaukums = k.table_name(+) group by t.tabulas_nosaukums, t.komentars order by 1	
8	Tālāk dokumentā tiks apskatīta katra datu bāzes tabula detalizēti. Katrai tabulai tiks aprakstītas kolonnas, primārās atslēgās, unikālās atslēgas, pārbaudes, saites uz citām tabulām, kā arī tabulas, kurām ir saites uz attiecīgo tabulu.	
	10	**select tabulas_nosaukums from ist_tabula order by 1**
	11	select '2.', '', 'Tabulas ' || :blockpar1 || ' apraksts' from dual
	13	select '2.', '', '.1.', 'Tabulas struktūra' from dual
	15	Nr.;Kolonna;Datu tips;NULL?;Komentārs
1. cikls	16	select distinct k.column_id, k.column_name, decode(k.data_type, 'varchar2', k.data_type || '(' || k.data_length || ')', 'number', decode(k.data_precision, null, 'integer', k.data_type || '(' || k.data_length || ',' || k.data_scale || ')'), 'char', k.data_type || '(' || k.data_length || ')', k.data_type) type, decode(nullable, 'y', 'jā', 'n', 'nē', '') nullable, ik.komentars from user_tab_cols k, (select kol.kol_nosauk, kol.komentars from ist_kol kol, ist_tabula t where kol.tab_isn = t.id and t.tabulas_nosaukums = :blockpar1) ik where k.table_name = :blockpar1 and k.column_name = ik.kol_nosauk (+) order by 1
	18	select '2.', '', '.2.', 'Primārā atslēga, unikālās atslēgas, pārbaudes' from dual
	20	select 'Identifikators', 'Kolonna', 'Tips', 'Komentārs' from user_constraints where table_name = :blockpar1 and constraint_type not in ('r', 'c') and rownum = 1
	21	select uc.constraint_name, ucc.column_name, decode(uc.constraint_type, 'P', 'Primārā atslēga', 'U', 'Unikālā atslēga', 'C', 'Pārbaude', '') c_type, i.koment from user_constraints uc, user_cons_columns ucc, ist_ier i where ucc.constraint_name = uc.constraint_name and uc.table_name = :blockpar1 and uc.status = 'enabled' and uc.constraint_type not in ('R', 'C') and uc.constraint_name = i.ier_nosauk (+) order by 2, 3
	23	select '2.', '', '.3.', 'Tabulas, uz kurām ' || :blockpar1 || ' ir reference' from dual
	25	select 'Identifikators', 'Kolonna (No)', 'Tabula', 'Kolonna (Uz)', 'Komentars' from user_constraints c, user_constraints r, user_cons_columns cc, user_cons_columns rc where c.constraint_type = 'R' and c.r_owner = r.owner and c.r_constraint_name = r.constraint_name and c.constraint_name = cc.constraint_name and c.owner = cc.owner and r.constraint_name = rc.constraint_name and r.owner = rc.owner and cc.position = rc.position and c.table_name = :blockpar1 and rownum = 1
	26	select c.constraint_name, cc.column_name, r.table_name, rc.column_name, a.koment from user_constraints c, user_constraints r, user_cons_columns cc, user_cons_columns rc, ist_att a where c.constraint_type = 'R' and c.r_owner = r.owner and c.r_constraint_name =

		r.constraint_name and c.constraint_name = cc.constraint_name and c.owner = cc.owner and r.constraint_name = rc.constraint_name and r.owner = rc.owner and cc.position = rc.position and c.table_name = :blockpar1 and c.constraint_name = a.att_nosauk (+) order by c.table_name, c.constraint_name, cc.position
	28	select '2.', '', '.4.', 'Tabulas, kuras referencējas uz ' || :blockpar1 from dual
	30	select 'identifikators', 'tabula', 'kolonna', 'komentars' from user_constraints c, user_constraints r, user_cons_columns cc, user_cons_columns rc where c.constraint_type = 'R' and c.r_owner = r.owner and c.r_constraint_name = r.constraint_name and c.constraint_name = cc.constraint_name and c.owner = cc.owner and r.constraint_name = rc.constraint_name and r.owner = rc.owner and cc.position = rc.position and r.table_name = :blockpar1 and rownum = 1
	31	select c.constraint_name, c.table_name, cc.column_name, a.koment from user_constraints c, user_constraints r, user_cons_columns cc, user_cons_columns rc, ist_att a where c.constraint_type = 'R' and c.r_owner = r.owner and c.r_constraint_name = r.constraint_name and c.constraint_name = cc.constraint_name and c.owner = cc.owner and r.constraint_name = rc.constraint_name and r.owner = rc.owner and cc.position = rc.position and r.table_name = :blockpar1 and c.constraint_name = a.att_nosauk (+) order by c.table_name, c.constraint_name, cc.position

Lai „DIREPO" no šādas dokumenta definīcijas spētu uzģenerēt pilnīgu un aktuālu dokumentāciju, nepieciešams, lai arī datu bāzu pārvaldības sistēmā un objektu piemēru modulī būtu aktuāla informācija. Daļu no datu bāzes projektējuma definēšanas rezultātā iegūtā dokumenta iespējams apskatīt 3. pielikumā. Šajā paraugā redzams, ka nav ievadīta komentāru informācija, kā arī datu bāzu pārvaldības sistēmā vairākām tabulām nav izveidotas saites un primārās atslēgas, toties ir iespēja iepazīties ar uzģenerētā dokumenta struktūru.

Ar šādu datu bāzes projektējuma definīciju „DIREPO" spēj uzģenerēt 60 lappušu dokumentāciju, kurā aprakstītas 109 „ISTehnoloģijas" tabulas aptuveni 5 minūšu laikā. Jāņem vērā tas, ka pagaidām datu bāzē vēl nav savadīta visa dokumentācijai nepieciešamā informācija un ka tas ir atkarīgs no datora resursiem. Tomēr 60 lappušu dokumenta iegūšana 5 minūšu laikā ir ievērojams ieguvums, it īpaši, ja ņem vērā faktu, ka dokumentācijas sastādītājs šo pašu informāciju pats meklē datu bāzē un pēc tam raksta dokumentā. Protams, tas gan nav pilnībā gatavs dokuments, kuru uzreiz var izmantot, bet, nedaudz papildinot, tas, salīdzinoši daudz īsākā laikā un patērējot daudz mazāk resursus, ir gatavs lietošanai.

7.3.2.2. Lietotāja rokasgrāmatas automatizēta ģenerēšana

Lietotāja rokasgrāmatā iekļaujamā informācija pilnībā tiek ņemta no „ISTehnoloģijas" projektējuma. Šī dokumenta šablona definīcija ir daudz sarežģītāka, jo, lai iegūtu nepieciešamo informāciju, to nepieciešams krietni pārveidot, lai iegūtu dokumentācijai derīgu struktūru.

7.2. tabula

Lietotāja rokasgrāmatas definīcija „DIREPO" vidē

Nr.	Datu avots
1	Lietotāja rokasgrāmata
3	Darba vietas apraksts veidots dalījuma pa moduļiem - 1. (augstākā) līmeņa nosaukumiem skatu un

	objektu hierarhiskajā struktūrā. Tālāks iedalījums uzskaita dotajam modulim saistošās operācijas. Ņemot vērā sistēmas specifiku, katrs operācijas apraksts ietver 3 daļas:				
5	IZSAUKŠANA - ietver navigācijas secību un ceļu līdz sistēmas datu apstrādes vienumam;				
6	DATU APSTRĀDE - ietver atvērtā sistēmas datu apstrādes vienuma vadīklu darbināšanas aprakstu;				
7	IEROBEŽOJUMI/NOSACĪJUMI - ietver sistēmas datu apstrādes ierobežojumu un/vai nosacījumu (ja tādi eksistē) aprakstu.				
	9	select to_number(dvi_isn) from ist_dvi where dvi_isn = 14			
	10	exec istkoks(to_number(:blockpar1));			
	11	select dvi_nosauk from ist_dvi where dvi_isn = :blockpar1			
	13	select dvi_dokum from ist_dvi where dvi_isn = :blockpar1			
		15	select t.ska_isn, d.skd_order from ist_ska t, ist_skd d where d.dvi_isn = :blockpar1 and t.ska_isn = d.ska_isn_b and d.ska_isn_v is null order by d.skd_order, t.ska_nosauk		
		16	select 'Skats "' || ska_nosauk || '"' from ist_ska where ska_isn = :blockpar2		
1. cikls	2. cikls	3. cikls	**18**	**select distinct iska.akc_isn from ist_ska t, ist_skd d, (select substr(a.rez, 0, instrc(a.rez, '//', 1,1)-1) ska, a.rez, a.akc_isn from (select distinct rez, akc_isn from ist_koks order by rez) a) iska, ist_akc ak where d.dvi_isn = :blockpar1 and t.ska_isn = d.ska_isn_b and d.ska_isn_v is null and t.ska_nosauk = iska.ska and t.ska_isn = :blockpar2 and iska.akc_isn = ak.akc_isn and mod(ak.akc_tips, 2) != 1 order by iska.akc_isn**
			19	select 'Operācija "' || akc_vards || '"' from ist_akc where akc_isn = :blockpar3
			21	select akc_apraksts from ist_akc where akc_isn = :blockpar3
			23	IZSAUKŠANA
			25	select replace(tab.rez, '//', '->') from (select t.ska_isn, t.ska_nosauk, iska.rez, d.skd_order, iska.akc_isn from ist_ska t, ist_skd d, (select substr(a.rez, 0, instrc(a.rez, '//', 1,1)-1) ska, a.rez, a.akc_isn from (select distinct rez, akc_isn from ist_koks order by rez) a) iska where d.dvi_isn = :blockpar1 and t.ska_isn = d.ska_isn_b and d.ska_isn_v is null and t.ska_nosauk = iska.ska order by iska.akc_isn, d.skd_order, t.ska_nosauk, iska.rez) tab where tab.akc_isn = :blockpar3 and rownum = 1
			26	@CallApp("ISTScreenShot.exe", "JaunaKontuGrupa874")
			27	select ‘\field{*\fldinst { includepicture „bildes/’ || akc_testpiem || ‘.bmp” * mergeformat \\d }}' from ist_akc where akc_isn = :blockpar3 and akc_testpiem is not null
			29	DATU APSTRĀDE
			31	select akc_apstrade from ist_akc akc where akc_isn = :blockpar3
			33	IEROBEŽOJUMI/NOSACĪJUMI
			35	select akc_ierob from ist_akc akc where akc_isn = :blockpar3

Dokumenta sākumā tiek izvadīta statiska informācija par dokumenta struktūru. Pēc tam sākas pirmais cikls, kas sagrupē informāciju pa darba vietām (skat. 7.2. tabulā 9. rinda). Nākošajā rindā tiek izsaukta procedūra „ISTKOKS”, kura noteiktai darba vietai katram 1. līmeņa skatam sameklē visas operācijas, kas piesaistītas 1. līmeņa skatam, apakšskatiem un to objektiem, un ceļu, kā nokļūt līdz šai operācijai. Šī informācija procedūras izpildes rezultātā tiek ievietota tabulā „IST_KOKS”.

15. rindā sākas otrais cikls. Šajā ciklā tiek izdrukāti visi attiecīgās darba vietas 1. līmeņa skati. Savukārt 18. rindā, izmantojot informāciju no tabulas „IST_KOKS”, tiek atlasīti visas noteiktas darba vietas un noteikta 1. līmeņa skata operācijas. Šajā rindā tiek uzsākts arī pēdējais, trešais, cikls, kas izdrukā visas iepriekšminētās operācijas. Šī cikla ķermenī tiek aprakstīta katra operācija. 25. rindā tiek ievietots ceļš līdz operācijai. Nākošajā rindā tiek

izsaukts „ISTehnoloģijas” formu fotografēšanas rīks „ISTScreenShot”, kā argumentu padodot testpiemēra identifikatoru. Pēc „ISTScreenShot” izpildes beigām 27. rindā tiek ievietots operācijas izsauktās formas attēls, ja tāds eksistē. Atlikušajās rindās tiek izdrukātas nodaļas „Datu apstrāde” un „Ierobežojumi/Nosacījumi”, kurās iekļautā informācija tiek iegūta no „ISTehnoloģijas” darba vietu moduļa.

Atšķirībā no datu bāzes projektējuma šī uzģenerētā dokumenta saturs nav tik ļoti atkarīgs no tā, vai ir savadīta aktuālā informācija, jo sistēmas projektējums vienmēr ir aktuāls, citādāk sistēma nedarbotos. Protams, aktuāli var nebūt darba vietu modulī pievienotie jaunie atribūti, kuri speciāli pievienoti dokumentācijas ģenerēšanai, jo bez šiem atribūtiem sistēma var darboties bez ierobežojumiem.

Lietotāja rokasgrāmatas (daļu iespējams apskatīt 4. pielikumā), kas sastāv no 214 lappusēm bez bildēm, uzģenerēšana aizņem aptuveni 7 minūtes. Tomēr, ja šī pati dokumentācija tiks ģenerēta ar bildēm, tad šis process varētu aizņemt diennakti vai pat ilgāk, jo, lai iegūtu katru bildi, nepieciešams atvērt formu un pēc tam aizvērt. Jāņem vērā fakts, ka, ja to darītu cilvēks, tas aizņemtu daudz vairāk laika, neskatoties uz to, ka attēlu automātiskai fotografēšanai nepieciešams nodefinēt testpiemērus, jo, atkārtoti ģenerējot dokumentāciju, testpiemēru definēšana vairs nav nepieciešama, tas jādara tikai jaunām formām.

7.3.2.3. Nododamo vienumu saraksta automatizēta ģenerēšana

Nododamo vienumu saraksts tiek veidots tikai no vienā tabulā pieejamās informācijas. Šī tabula ir „ISTehnoloģijas” darba vietu moduļa entītija „IST_AKC”. Šajā tabulā glabājas informācija par operācijām un to izsauktajām formām. Tādēļ 7.3. tabulā 8. rindā esošais pieprasījums atlasa un izdrukā visas tās programmatūras izpilddatnes, kuras tiek izsauktas ar skatam piesaistītu operāciju. Šajā pieprasījumā visas piesaistītas izpilddatnes (*.exe), tiek pārsauktas par datnēm ar paplašinājumu „*.*apt*”. Tas notiek tāpēc, ka „Centura SQLWindows/32" izpilddatnes tiek kompilētas no programmatūras koda datnēm, kuras tad arī tiek nodotas pasūtītājam.

13. rindā pieprasījums izdrukā visas tās atskaites, kas tiek izsauktas no vismaz vienam skatam piesaistītas operācijas. Tas, vai operācija izsauc atskaiti, tiek noteikts pēc izpilddatnes, kura atskaitēm vienmēr ir „DIREPO_IST.EXE”. Operācija kā parametru „DIREPO” padod atskaites nosaukumu, kuru jāģenerē, tāpēc atskaites nosaukums pieprasījumā tiek iegūts no operācijas atribūta „AKC_PARAM”.

7.3. tabula

Nododamo vienumu saraksta definīcija „DIREPO” vidē

Nr.	Datu avots
1	Nododamo vienumu saraksts
3	Tālāk ir uzskaitīti visi tie izejas teksta faili, tām formām, kuras ir piesaistītas kādam no sistēmā pieejamajiem skatiem.
5	Programmatūras izpilddatnes
7	Izejas teksta fails;Nosaukums;Apraksts
8	select distinct replace(upper(a.akc_engine), '.exe', '.apt'), a.akc_nosauk, a.akc_apraksts from ist_akc a, ist_sdo s where a.akc_isn = s.akc_isn and upper(a.akc_engine) like '%.EXE%' and upper(a.akc_engine) not like '%DIREPO_IST.EXE%' order by 1
10	„DIREPO” atskaites
12	Atskaite;Nosaukums;Apraksts
13	select distinct substr(a.akc_param, 0, instr(a.akc_param, ' ', 1, 1)-1), a.akc_nosauk, a.akc_apraksts from ist_akc a, ist_sdo s where a.akc_isn = s.akc_isn and upper(a.akc_engine) = 'direpo_ist.exe' order by 1

Šādi uzģenerēta dokumenta (skat. 5. pielukumu) aktualitāte, tāpat kā gadījumā ar lietotāja rokasgrāmatu, ir vienmēr augsta, jo visa informācija tiek ņemta no sistēmas projektējuma, bez kura sistēma nedarbojas. Vienīgi aprakstu aktualitāte ir atkarīga no ievadītās informācijas savlaicīguma.

Šāda četru lappušu dokumenta uzģenerēšanai nepieciešamas tikai 10 sekundes. Tomēr šis dokuments nav pilnīgs, jo vēl nepieciešams pievienot mūsdienu programmēšanas valodu pirmkoda datnes, programmatūras skriptus, kā arī dokumentācijas. Salīdzinot ar iepriekšējiem dokumentiem, arī šāda dokumentācijas automatizēta ģenerēšana ietaupa laiku, kas nepieciešams, lai sagatavotu visus programmatūras projektos izmantotos dokumentus.

8. REZULTĀTI

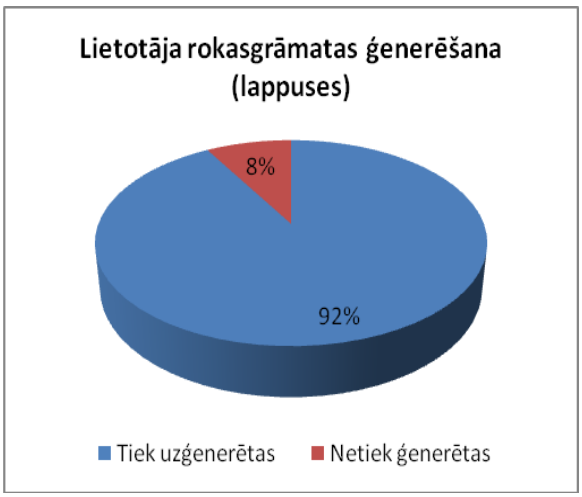
Pirms „ISTehnoloģijas” dokumentācijas ģeneratora izstrādes tika apskatīti dažādi dokumentācijas ģeneratoru veidi un nolemts veidot dokumentācijas ģeneratoru no sistēmas. Šādu izvēli nebija grūti izdarīt, pateicoties „ISTehnoloģijai”, kas balstīta uz metamodeļa.

„ISTehnoloģijas” pielāgošana dokumentācijas ģenerēšanai neradīja sarežģījumus, jo, veidojot objektu moduli, bija jau dažas iestrādnes, ko varēja aizņemties no iepriekšējo gadu pieredzes. Rezultātā „ISTehnoloģijas” metamodelī nekādas būtiskas strukturālas izmaiņas netika veiktas. Tika pievienoti tikai daži atribūti, kas neietekmē sistēmas funkcionalitāti, bet nodrošina informācijas avotu dokumentācijas ģeneratoram.

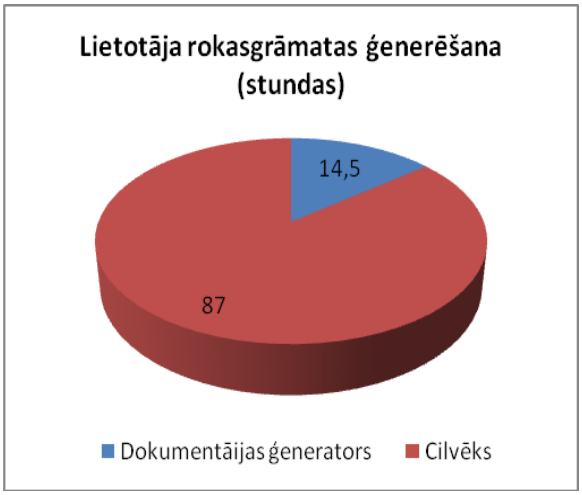
Savukārt „DIREPO” pielāgošana dokumentācijas ģenerēšanai bija sarežģītāka, jo šis rīks tika būvēts, kā atskaišu ģenerators, nevis kā dokumentācijas ģenerators. Tādēļ, definējot dokumentus „DIREPO”, tika konstatēti vairāki trūkumi „DIREPO” funkcionalitātē. Viens no tiem bija tāds, ka „DIREPO” definēšanas valodā nebija iespējas tekstu stiliem pievienot automātisko numurēšanu un satura rādītāju. Šāda funkcionalitātes ieviešana netika veikta, jo tas prasīja nopietnu teksta stilu definēšanas struktūras pārveidi. Toties veiksmīgi tika realizēta neatkarīgu lietojumprogrammu izsaukšana, kas šajā gadījumā bija „ISTehnoloģijas” formu fotografēšanas rīks „ISTScreenShot”.

Kas attiecas uz rīku „ISTScreenShot”, tad sākumā bija doma veidot jaunu rīku, kas spētu atvērt „ISTehnoloģijā” piesaistītu formu un pēc tam nofotografēt. Tomēr, apskatot dažādus pasaulē pielietotos rīkus, tika nonākts pie secinājuma, ka šajā situācijā ļoti noderīgs būtu kāds lietotāja saskarnes testēšanas rīks. Rezultātā, apskatot šādus rīkus, tika atrasts rīks „Test Automation FX”, kas spēj ierakstīt formas izsaukšanu un saglabāt tās attēlu. Tādējādi tika atrisināta „ISTehnoloģijas” formu attēlu iegūšanas problēma.

Dokumentu ģenerēšana vēl līdz galam nav pabeigta, jo ar pašlaik sadefinētajiem dokumentiem nav iespējams uzģenerēt pilnu dokumentu. Realizēta ir tā daļa, kas ģenerē visu dokumentu būtiskākās un apjomīgākās nodaļas, bet netiek ģenerēta titullapa, satura rādītājs, automātiskā nodaļu numurēšana un nodaļas, kas rakstītas brīvā tekstā. Datu bāzu projektējumā tiek ģenerēts gan tabulu saraksts, gan arī tabulu detalizētais apraksts. Lietotāja rokasgrāmatā tiek ģenerēta pamatstruktūra, tomēr, lai šādu dokumentu uzģenerētu pilnībā, nepieciešama informācijas ievade datu bāzē. Savukārt nododamo vienumu sarakstam tiek ģenerēts pilnībā viss nododamo programmatūras koda un atskaišu vienumu saraksts, bet pārējais jāpievieno klāt pēc dokumenta uzģenerēšanas.

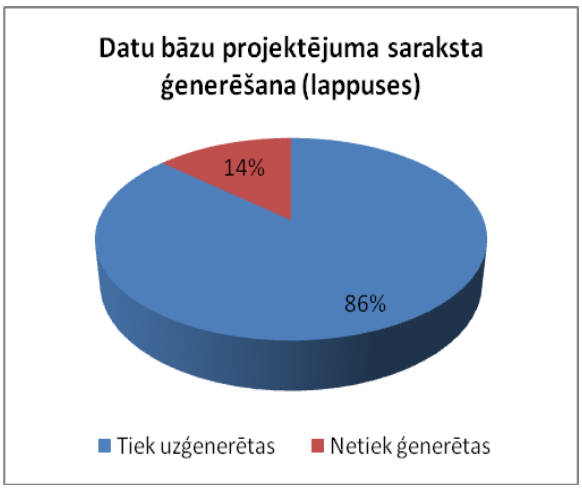


8.1. att. **Lietotāja rokasgrāmatas uzģenerējamo un neģenerējamo lappušu attiecība**

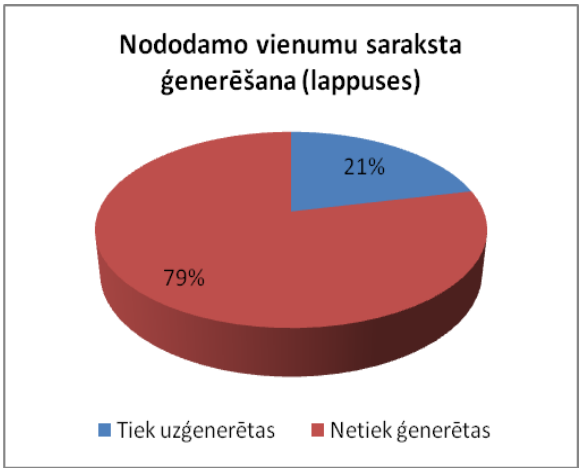


8.2. att. **Lietotāja rokasgrāmatas izstrādei nepieciešamās stundas**

Tomēr, neskatoties uz vēl nepabeigto dokumentācijas ģenerēšanu, līdz šim ir panākts, ka dokumentos tiek uzģenerētas pamatnodaļas, kuru rakstīšana patērē visvairāk laika. Ja pieņem, ka dokumentācijas ģenerēšanai visa nepieciešama informācija ir ievadīta, tad no pašlaik uzņēmumā izmantotās lietotāja rokasgrāmatas [3.1.], kuras apjoms ir 380 lappuses, ar līdz šim izstrādāto dokumentācijas ģeneratoru tiktu uzģenerētas 348 lappuses, kas kopumā sastāda 92% no dokumenta apjoma (skat. diagrammu). Lietotāja rokasgrāmatas izstrāde līgumā tika vērtēta ar 11 cilvēkdienām, līdz ar to vienas lappuses izstrādes laiks ir vidēji 15 minūtes. Tad uzģenerējamās 348 lappuses tiktu uzrakstītas 87 stundās, kas ir gandrīz 11 cilvēkdienas. Savukārt, ja pieņem, ka vienā lappusē vidēji ir viens attēls, kuru „ISTScreenShot" iegūst vidēji 2 minūšu laikā, tad vienu lappusi dokumentācijas sliktākajā gadījumā ģenerators veidotu vidēji 2,5 minūtes. Tas nozīmē, ka 348 lappuses tiku uzģenerētas 14,5 stundās, kas ir nepilnas 2 cilvēkdienas. Rezultātā, ģenerējot lietotāja rokasgrāmatu, tiek ietaupītas 72,5 stundas jeb aptuveni 9 cilvēkdienas.



8.3. att. **Datu bāzu projektējuma uzģenerējamo un neģenerējamo lappušu attiecība**



8.4. att. **Nododamo vienumu saraksta izstrādei nepieciešamās stundas**

Balstoties uz dokumentācijas izstrādāšanas novērtējumu līgumā, sanāk, ka vidēji datu bāzu projektējuma [3.3.] un nododamo vienumu saraksta [3.2.] vienas lappuses uzrakstīšanai cilvēkam nepieciešamas attiecīgi 20 un 15 minūtes, bet dokumentācijas ģeneratoram 0,005 un 0,0025 minūtes, tad cilvēkam tas prasītu attiecīgi 61,6 un 0,375 stundas, bet datoram attiecīgi 0,01541 un 0,0000625 stundas. Rezultātā, ģenerējot datu bāzu projektējumu, tiek ietaupītas 61,5 stundas, bet, ģenerējot nododamo vienumu sarakstu, - 0,374 stundas.

9. SECINĀJUMI

Izstrādājot bakalaura darbu par automatizētu dokumentācijas ģenerēšanu „ISTehnoloģijai", nonācu pie šādiem secinājumiem:

1. Mūsdienās pastāv vairāki dokumentācijas ģeneratoru veidi. Katrs no tiem paredzēts dažādu programmatūras izstrādes etapu un līmeņu dokumentācijas ģenerēšanai.
2. Dokumentācijas ģeneratori no sistēmas nevar būt universāli, jo tie ir konkrētai sistēmai specifiski, bet ģeneratori no programmatūras koda un speciāli komentēta koda ir salīdzinoši universāli un pielietojami vairākām programmēšanas valodām un sistēmām.
3. Modeļu bāzēta arhitektūra ir piemērota ne tikai lielu sistēmu izstrādē, bet arī nelielu projektu realizēšana. Šīs arhitektūras pamatprincipus iespējams pielāgot arī dokumentācijas ģeneratoram, tādējādi apvienojot divus neatkarīgus modeļus – biznesa sistēmas modeli un dokumenta modeli.
4. No realizācijas viedokļa modeļu bāzētajā arhitektūrā vissarežģītākā daļa ir transformācija, jo tajā ir jāsavieno divi neatkarīgi modeļi.
5. Latvijā programmatūras projektos izmantotās dokumentācijas specificē LVS standarts. Latvijā standarts nav izstrādāts pilnīgi visiem programmatūras projektos izmantotajiem dokumentiem.
6. Uzņēmumā tiek izstrādātas un pielietotas daudzas dokumentācijas – gan tās, kuras specificē LVS standarts, gan arī tās, kuras tiek veidotas pēc uzņēmumā pieņemtās prakses.
7. Lai būtu iespējams uzģenerēt dokumentāciju vai daļu no tās, tajā iekļautajai informācijai jābūt strukturētai un pieejamai sistēmā.
8. Ne visas nodaļas dokumentācijā iespējams uzģenerēt, jo dažas nodaļas satur nestrukturizētu, sistēmā neeksistējošu informāciju.
9. Dokumentācijas ģenerēšanai nepieciešams pielāgot darba vietu moduli un atjaunot objektu moduli.
10. Pašlaik „DIREPO" nepiedāvā vairākas funkcionalitātes, kuras nepieciešamas tieši dokumentācijas ģenerēšanā.
11. Formas attēlu fotografēšanas rīks „ISTScreenShot" ir atkarīgs no sistēmas. Ja sistēmā tiek veiktas izmaiņas, tās var ietekmēt rīka darbību.

12. Formas attēlu iegūšana ir laikietilpīgs process. Tomēr dators to izdara ievērojami ātrāk nekā cilvēks.
13. Vairākkārtīgi ģenerējot dokumentāciju, kurai nepieciešami formu attēli, tiek ietaupīts laiks uz formas fotografēšanas testpiemēru veidošanas.
14. Automatizēti ģenerēts dokuments nekad nebūs tik lasāms, kā cilvēka veidots dokuments.
15. Dokumentācijas definēšana „DIREPO" ir salīdzinoši sarežģīta, jo nepieciešams apkopot dažāda veida informāciju, kas iegūta no „ISTehnoloģijas".
16. Automatizēti ģenerējot dokumentāciju, no „ISTehnoloģijas" iespējams iegūt ļoti labi strukturētu un pārskatāmu dokumentāciju, kaut arī nav iespējams uzģenerēt to pilnībā.
17. Automatizēta dokumentācijas ģenerēšana ievērojami ietaupa patērēto laiku, jo tā paveic cilvēka „melno" darbu.
18. Ar bakalaura darbā izstrādātajiem dokumentāciju ģenerēšanas šabloniem iespējams ģenerēt lielāko daļu no lietotāja rokasgrāmatas, datu bāzu projektējuma un nododamo vienumu saraksta.
19. Datu bāzu projektējuma pamatinformāciju iespējams pilnībā uzģenerēt.
20. Automatizēti ģenerēta lietotāja rokasgrāmatu nav iespējams uzģenerēt pilnīgi precīzi, kā to ir izveidojis cilvēks. Toties tās uzģenerēšana atvieglo cilvēka darbu.
21. Sistēmas informāciju nepieciešams regulāri atjaunot, lai automatizēti uzģenerēta dokumentācija būtu aktuāla.

Nākotnē nepieciešams papildināt „DIREPO" funkcionalitāti, lai tas spētu ģenerēt pēc iespējas līdzīgākus dokumentus cilvēka veidotam dokumentam. Vēl nepieciešams papildināt „ISTScreenShot" rīka funkcionalitāti ar automātisku testpiemēru definēšanu. Protams, nepieciešams arī uzlabot dokumentācijas ģenerēšanas ātrdarbību un papildināt dokumentācijas ģeneratora ģenerējamo dokumentāciju klāstu ar citām dokumentācijām.

IZMANTOTĀ LITERATŪRA UN AVOTI

1. **Grāmatas**

1.1. **Kleppe A., Warmer J., Bast W.**, *MDA Explained: The Model Driven Architecture: Practice and Promise*. Addison Wesley, 2003.

1.2. **Šmite D., Dosbergs D., Borzovs J.,** *Informācijas un komunikācijas tehnoloģijas nozares tiesību un standartu pamati.* Latvijas Universitāte, 2005

2. **Publikācijas**

2.1. **Iljins J.**, Metamodel Based Approach to IS Development and Lessons Learned. Baltic DB&IS, 2008.

2.2. **Treimanis M.**, ISTehnology – Tehnology Based approach to Information System Development. University of Latvia, 1997.

3. **Iekšējā sistēmas dokumentācija**

3.1. DIVI.SEB.PJD.LLC.VVUS, *VVUS lietotāja rokasgrāmata*. Datorikas Institūts DIVI, 2008.

3.2. DIVI.SEB.PJD.NVS.VIE11, *Nododamo vienumu saraksts.* Datorikas Institūts DIVI, 2009.

3.3. DIVI.VVUS.PJD.DBP, *VVUS Datu bāzu projektējums.* Datorikas Institūts DIVI, 2009.

4. **LVS standarti**

4.1. Ieteicamā prakse programmatūras projektējuma aprakstīšanai, LVS 72:1996, Latvijas Valsts standarts.

4.2. Programmatūras lietotāja dokumentācija, LVS 66:1996, Latvijas Valsts standarts.

5. **Elektroniskie informācijas avoti**

5.1. *An introduction to MDA*. [tiešsaiste] – [atsauce 11.04.2009].
Pieejams Internetā: http://www.ibm.com/developerworks/rational/ library/3100.html.

5.2. *Committed Companies and Their Products*. [tiešsaiste]–[atsauce 11.04.2009].
Pieejams Internetā: http://www.omg.org/mda/committed-products.htm.

5.3. *Comparison of documentation generators*. [tiešsaiste] – [atsauce 03.04.2009].
Pieejams Internetā: http://en.wikipedia.org/wiki/Comparison_of _documentation_generators.

5.4. *Lielā terminu vārdnīca*. [tiešsaiste] – [atsauce 25.05.2009].
Pieejams Internetā: http://www.termini.lv/index.php.

5.5. *Model Driven Architecture (MDA) FAQ...*. [tiešsaiste] – [atsauce 11.04.2009]. Pieejams Internetā: http://www.omg.org/mda/faq_mda.htm#what%20is %20mda.

5.6. *OMG Model Driven Architecture*. [tiešsaiste] – [atsauce 11.04.2009]. Pieejams Internetā: http://www.omg.org/mda/.

5.7. *Reverse Engineering Resources*. [tiešsaiste] – [atsauce 10.04.2009]. Pieejams Internetā: http://www.backerstreet.com/cg/work.htm.

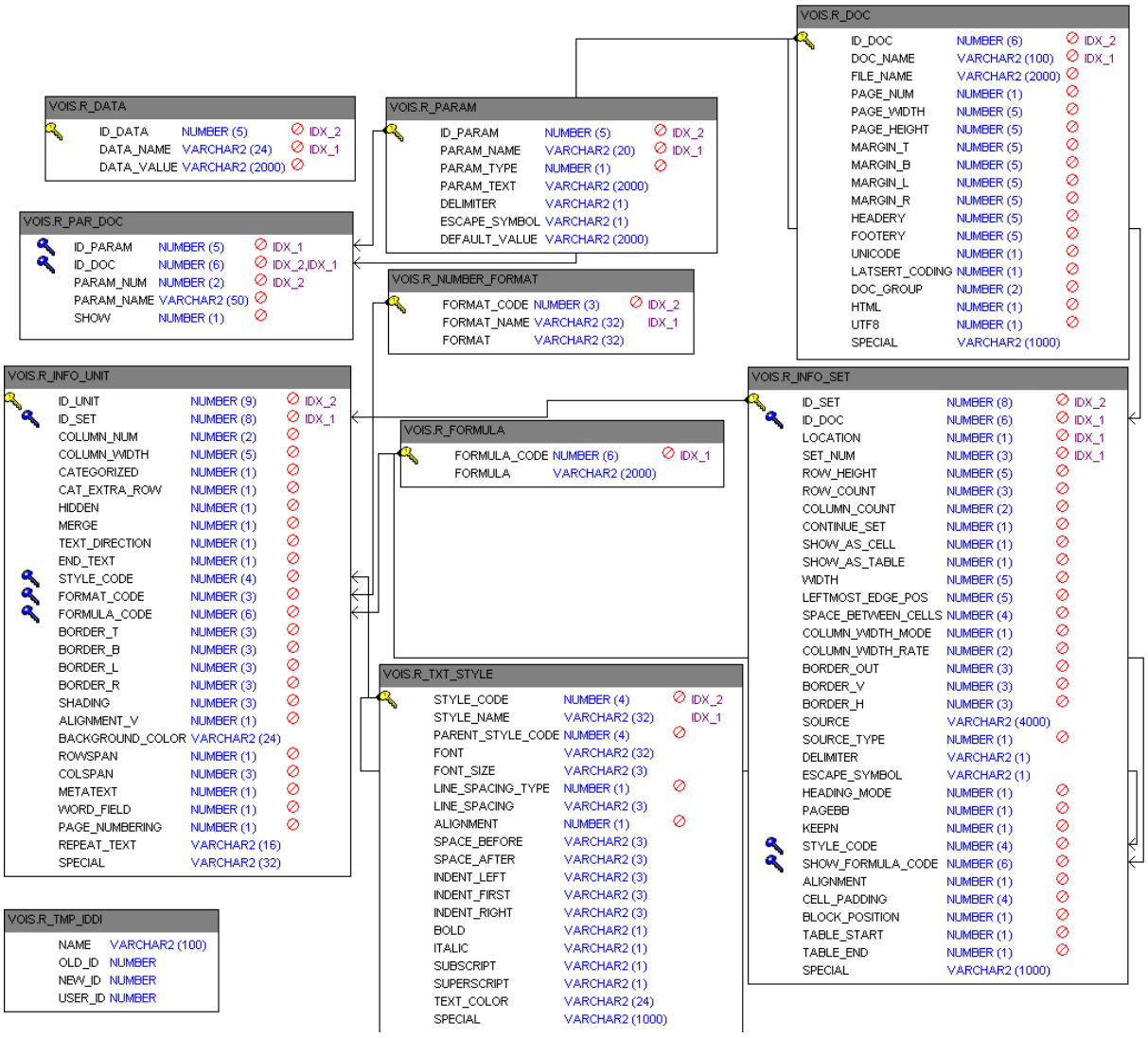
5.8. *Reverse Engineering Tools*. [tiešsaiste] – [atsauce 11.03.2009]. Pieejams Internetā: http://www.laatuk.com/tools/documentation_tools.html.

5.9. *Test Automation FX*. [tiešsaiste] – [atsauce 15.05.2009]. Pieejams Internetā: http://www.testautomationfx.com.

PIELIKUMI

1. pielikums

„ISTehnoloģijas" atskaišu moduļa „DIREPO" ER modelis



Daļa no „ISTScreenShot” koda

Program.cs

```
using System;
using System.Collections.Generic;
using System.Windows.Forms;
using TestAutomationFX.UI.TestRunner;

namespace ISTScreenShot
{
    static class Program
    {
        /// <summary>
        /// The main entry point for the application.
        /// </summary>
        [STAThread]
        static void Main(String [] Params)
        {
            TestApplication.Run(new ISTScreenShot(Params));
        }
    }
}
```

Program.cs

```
using System;
using System.ComponentModel;

using TestAutomationFX.Core;
using TestAutomationFX.UI;

namespace ISTScreenShot
{
    [UITestFixture]
    public partial class ISTScreenShot : UIMap
    {
        public ISTScreenShot(String [] Params)
        {
            InitializeComponent();

            if (Params.Length > 0)
            {
                switch (Params[0])
                {
                    case "VDNDV_Informacija_Produkti_Rediget":
                        VDNDV_Informacija_Produkti_Rediget();
                        break;
                    case "TEST2":
                        //Test2();
                        break;
                    default:
                        System.Windows.Forms.MessageBox.Show("Netika
atrasts šāds testpiemērs!");
                        break;
                }
```

```
            }
        }

        public ISTScreenShot(IContainer container)
        {
            container.Add(this);

            InitializeComponent();
        }

        [UITest()]
        public void VDNDV_Informacija_Produkti_Rediget()
        {
            ISTexe.DoubleClick(30, 27);
            StaticText.Click(240, 5);
            LBVOP_SEB_VVUSizst_20081205INI.Click(149, 2);
            BtnOK.Click();
            Username.Click(13, 12);
            Keyboard.SendKeys("admin[TAB]");
            Keyboard.SendKeys("admin[TAB]");
            BtnAuth.Click();
            StartWork.Click();
            Informācija.Click(8, 8);
            Produkti.Click(47, 6);
            ReģistrētDivideredižētproduktaaprakstu.DoubleClick(75, 21);

Produkturedaktorsv102Window.SaveImage("Bildes\\VDNDV_Informacija_Produkti_R
ediget.bmp");
            ButtonAizvērt.Click();
        }
    }
}
```

3. pielikums

Daļa no uzģenerētā „ISTehnoloģijas” datu bāzes projektējuma

Datu bāzes apraksts

Sistēmas datu bāze sastāv no šādām tabulām:

Tabulas nosaukums	Kolonnu skaits	Komentārs
IST_AGEN	6	
IST_AKC	17	
IST_ATT	5	
IST_AVB	3	
IST_DARBIBA	2	
IST_DOP	5	
IST_DOS	13	
IST_DOT	7	
IST_DVI	4	
IST_ERR	18	
IST_IER	5	Apzīmē kolonnas ierobežojumus, piemēram, pirmārā atslēga, unikāla atslēga, dažādus nosacījumus utt.
IST_INF	12	
IST_INF_SUT	9	
IST_KLUDA	3	
IST_KOL	4	Glabā informāciju par tabulas kolonnām
IST_KON	4	
IST_NOT	11	

[...]

VOP_ORG	26	Tabula "VOP_ORG" glabā informāciju par organizācijām.
VOP_PIL	5	

Tālāk dokumentā tiks apskatīta katra datu bāzes tabula detalizēti. Katrai tabulai tiks aprakstītas kolonnas, primārās atslēgās, unikālās atslēgas, pārbaudes, saites uz citām tabulām, kā arī tabulas, kurām ir saites uz attiecīgo tabulu.

2.1 Tabulas IST_AGEN apraksts

2.1.1. Tabulas struktūra

Nr.	Kolonna	Datu tips	NULL?	Komentārs
1	AGEN_ISN	NUMBER(22,0)	Nē	
2	AGEN_CONF	VARCHAR2(64)	Jā	
3	AGEN_ENGINE	VARCHAR2(254)	Jā	
4	AGEN_DLL	VARCHAR2(10)	Jā	
5	AGEN_PAR	VARCHAR2(255)	Jā	
6	AGEN_SEQ	NUMBER(22,0)	Jā	

2.1.2. Primārā atslēga, unikālās atslēgas, pārbaudes

Identifikators	Kolonna	Tips	Komentārs
SYS_C009567	AGEN_ISN	Primārā atslēga	

2.1.3. Tabulas, uz kurām IST_AGEN ir reference

2. 1 .4. Tabulas, kuras referencējas uz IST_AGEN

2. 2 Tabulas IST_AKC apraksts

2. 2 .1. Tabulas struktūra

Nr.	Kolonna	Datu tips	NULL?	Komentārs
1	AKC_ISN	INTEGER	Nē	
2	AKC_VARDS	VARCHAR2(64)	Nē	
3	AKC_TIPS	INTEGER	Nē	
4	AKC_ENGINE	VARCHAR2(32)	Jā	
5	AKC_PARAM	VARCHAR2(512)	Jā	
6	AKC_KOMENT	VARCHAR2(512)	Jā	
7	DOT_ISN	INTEGER	Jā	
8	AKC_TIPAISN	INTEGER	Jā	
9	AKC_IKONA	VARCHAR2(32)	Jā	
10	OBK_ISN	INTEGER	Jā	
11	AKC_DLL	VARCHAR2(32)	Jā	
12	AKC_JAUNS	NUMBER(22,0)	Jā	
13	AKC_APSTRADE	VARCHAR2(4000)	Jā	
14	AKC_APRAKSTS	VARCHAR2(4000)	Jā	
15	AKC_IEROB	VARCHAR2(4000)	Jā	
16	AKC_TESTPIEM	VARCHAR2(100)	Jā	
17	AKC_NOSAUK	VARCHAR2(100)	Jā	

2. 2 .2. Primārā atslēga, unikālās atslēgas, pārbaudes

Identifikators	Kolonna	Tips	Komentārs
SYS_C005147	AKC_ISN	Primārā atslēga	

2. 2 .3. Tabulas, uz kurām IST_AKC ir reference

Identifikators	Kolonna (No)	Tabula	Kolonna (Uz)	Komentars
SYS_C005692	DOT_ISN	IST_DOT	DOT_ISN	
SYS_C005693	OBK_ISN	IST_OBK	OBK_ISN	

2. 2 .4. Tabulas, kuras referencējas uz IST_AKC

Identifikators	Tabula	Kolonna	Komentars
SYS_C005656	IST_AVB	AKC_ISN_V	
SYS_C005657	IST_AVB	AKC_ISN_B	
SYS_C005755	IST_SAO	AKC_ISN	
SYS_C005727	IST_SDO	AKC_ISN	

2. 3 Tabulas IST_ATT apraksts

2. 3 .1. Tabulas struktūra

Nr.	Kolonna	Datu tips	NULL?	Komentārs
1	ATT_ISN	NUMBER(22,0)	Nē	
2	ATT_NOSAUK	VARCHAR2(100)	Nē	
3	KOL_ISN_NO	NUMBER(22,0)	Nē	

4	KOL_ISN_UZ	NUMBER(22,0)	Nē	
5	KOMENT	VARCHAR2(4000)	Jā	

2. 3 .2. Primārā atslēga, unikālās atslēgas, pārbaudes

Identifikators	**Kolonna**	**Tips**	**Komentārs**
IST_ATT_PK	ATT_ISN	Primārā atslēga	

2. 3 .3. Tabulas, uz kurām IST_ATT ir reference

Identifikators	Kolonna (No)	Tabula	Kolonna (Uz)	Komentars
SYS_C0011047	KOL_ISN_NO	IST_KOL	KOL_ISN	
SYS_C0011048	KOL_ISN_UZ	IST_KOL	KOL_ISN	

2. 3 .4. Tabulas, kuras referencējas uz IST_ATT

2. 4 Tabulas IST_AVB apraksts

2. 4 .1. Tabulas struktūra

Nr.	**Kolonna**	**Datu tips**	**NULL?**	**Komentārs**
1	AKC_ISN_V	INTEGER	Nē	
2	AKC_ISN_B	INTEGER	Nē	
3	AVB_ISN	INTEGER	Nē	

2. 4 .2. Primārā atslēga, unikālās atslēgas, pārbaudes

2. 4 .3. Tabulas, uz kurām IST_AVB ir reference

Identifikators	Kolonna (No)	Tabula	Kolonna (Uz)	Komentars
SYS_C005656	AKC_ISN_V	IST_AKC	AKC_ISN	
SYS_C005657	AKC_ISN_B	IST_AKC	AKC_ISN	

2. 4 .4. Tabulas, kuras referencējas uz IST_AVB

[...]

4. pielikums

Daļa no uzģenerētās „ISTehnoloģijas” lietotāja rokasgrāmatas

Lietotāja ceļvedis

Darba vietas apraksts veidots dalījuma pa moduļiem - 1. (augstākā) līmeņa nosaukumiem skatu un objektu hierarhiskajā struktūrā. Tālāks iedalījums uzskaita dotajam modulim saistošās operācijas. Ņemot vērā sistēmas specifiku, katrs operācijas apraksts ietver 3 daļas:

IZSAUKŠANA - ietver navigācijas secību un ceļu līdz sistēmas datu apstrādes vienumam

DATU APSTRĀDE - ietver atvērtā sistēmas datu apstrādes vienuma vadīklu darbināšanas aprakstu

IEROBEŽOJUMI/NOSACĪJUMI - ietver sistēmas datu apstrādes ierobežojumu un/vai nosacījumu (ja tādi eksistē) aprakstu.

Fondu darba vieta

Fondu darba vieta ir galvenā sistēmas darba vieta, kur notiek visas svarīgākās operācijas sistēmā

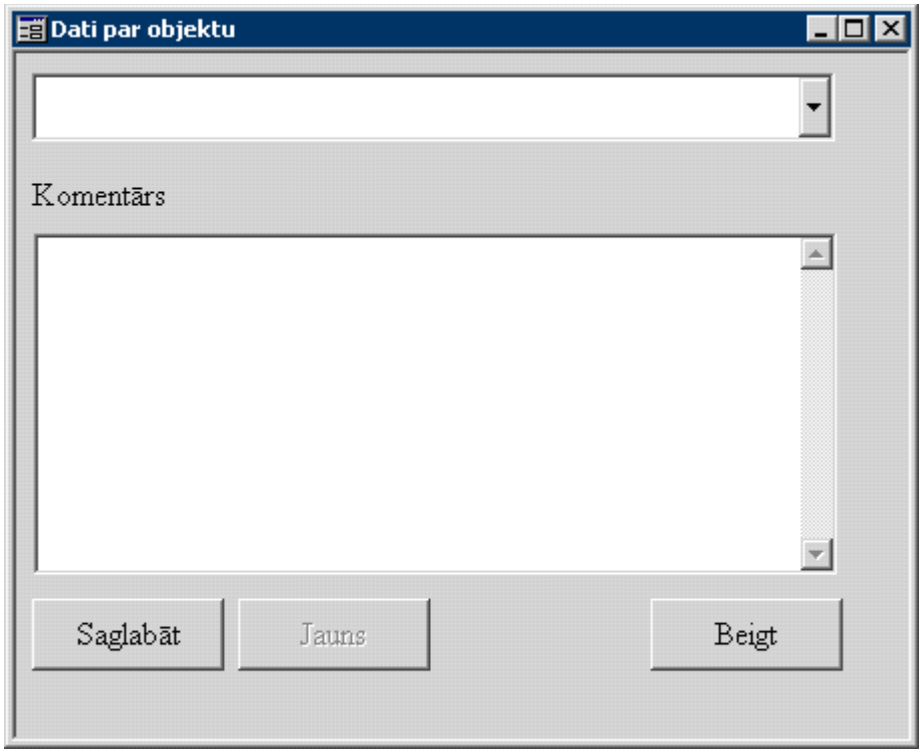
Skats "Portfeļi..."

Operācija "Jauna kontu grupa"

No sistēmas loģikas viedokļa kontu grupa ir identiska portfelim.

IZSAUKŠANA

Portfeļi...->Grāmatvedības portfeļi->Jauna kontu grupa



DATU APSTRĀDE

Aplikācijā "Dati par objektu" aizpilda vadīklas un datus saglabā ar komandpogu "Saglabāt".

IEROBEŽOJUMI/NOSACĪJUMI

Lai saglabātu komentāru, jāievada vismaz viens simbols, kas nav tukšums.

Operācija "Labot kontu grupu"

IZSAUKŠANA

Portfeļi...->Grāmatvedības portfeļi->Labot kontu grupu

DATU APSTRĀDE

IEROBEŽOJUMI/NOSACĪJUMI

Operācija "Jauns vērtspapīru konts"

IZSAUKŠANA

Portfeļi...->Grāmatvedības portfeļi->Jauns vērtspapīru konts

DATU APSTRĀDE

IEROBEŽOJUMI/NOSACĪJUMI

[...]

Daļa no uzģenerētā „ISTehnoloģijas" nododamo vienumu saraksta

Nododamo vienumu saraksts

Tālāk ir uzskaitīti visi tie izejas teksta faili, tām formām, kuras ir piesaistītas kādam no sistēmā pieejamajiem skatiem.

Programmatūras izpilddatnes

Izejas teksta fails	Nosaukums	Apraksts
AKCLINK.APT	Akciju piesaistīšana	
CNCOMPARELPS_VVUS.APT	LPS un VVUS salīdzināšana	
CNCOMPARELPS_VVUS_IIP.APT	LPS un VVUS salīdzināšana II	
CNDAL.APT	Dalītājs	
CNDALITAJSAKC.APT	Akciju sadalītājs	
CNDALITAJSFI.APT	FI dalītājs	
CNDALITAJSFX.APT	FX dalītājs	
CNDALITAJSIA.APT	IA dalītājs	
CNDALITAJSIAMAINA.APT	Maiņas darījumu dalītājs	
CNDARAKC.APT	Akciju darījumi	
CNDARAKCBLOK.APT	Bloķēto akciju darījumi	

[...]

"DIREPO" atskaites

Atskaite	Nosaukums	Apraksts
Dalu_pirpar	Daļu pirkšana un pārdošana	
Dalu_registra_dariijumi	Daļu reģistra darījumi	
Dalu_registra_dariijumi_excel	Daļu reģistra darījumi (Excel)	
Dalu_registrs_dienas_beigas	Daļu reģistrs dienas beigās	
Dalu_registrs_paplash	Daļu reģistrs (Paplašināts)	
Dar_Statistika	Darījumu statistika	
DepozDetalizeti	Depozīti	
DepozIIP	IIP depozīti	
Depozitu_parskats	Depozītu pārskats	
Depozitu_parskats_excel	Depozītu pārskats (Excel)	

[...]

Bakalaura darbs

Automatizēta dokumentācijas ģenerēšana „ISTehnoloģijai”

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

Autors: Valdis Vizulis _____
(Autora paraksts)

Ar savu parakstu apliecinu, ka esmu lasījis augšminēto bakalaura darbu un atzīstu to par **piemērotu/nepiemērotu** (nevajadzīgo svītrot) aizstāvēšanai Latvijas Universitātes datorzinātņu bakalaura studiju programmas gala pārbaudījuma komisijas sēdē.

Darba vadītājs(-ja): M. Dat. Jānis Iljins _____
(Vadītāja paraksts)

Darbs iesniegts Datorikas fakultātē 2009. gada 1. jūnijā.

Ar šo es apliecinu, ka darba elektroniskā versija ir augšupielādēta LU informatīvajā sistēmā.
Metodiķe: _____.
(Metodiķes paraksts)

Recenzents: M. Dat. Inga Medvedis

Darbs aizstāvēts bakalaura darbu gala pārbaudījuma komisijas sēdē

_____ prot. Nr. _____, vērtējums _____
(Darba aizstāvēšanas datums)

Komisijas sekretārs: _____
(Sekretāra paraksts)