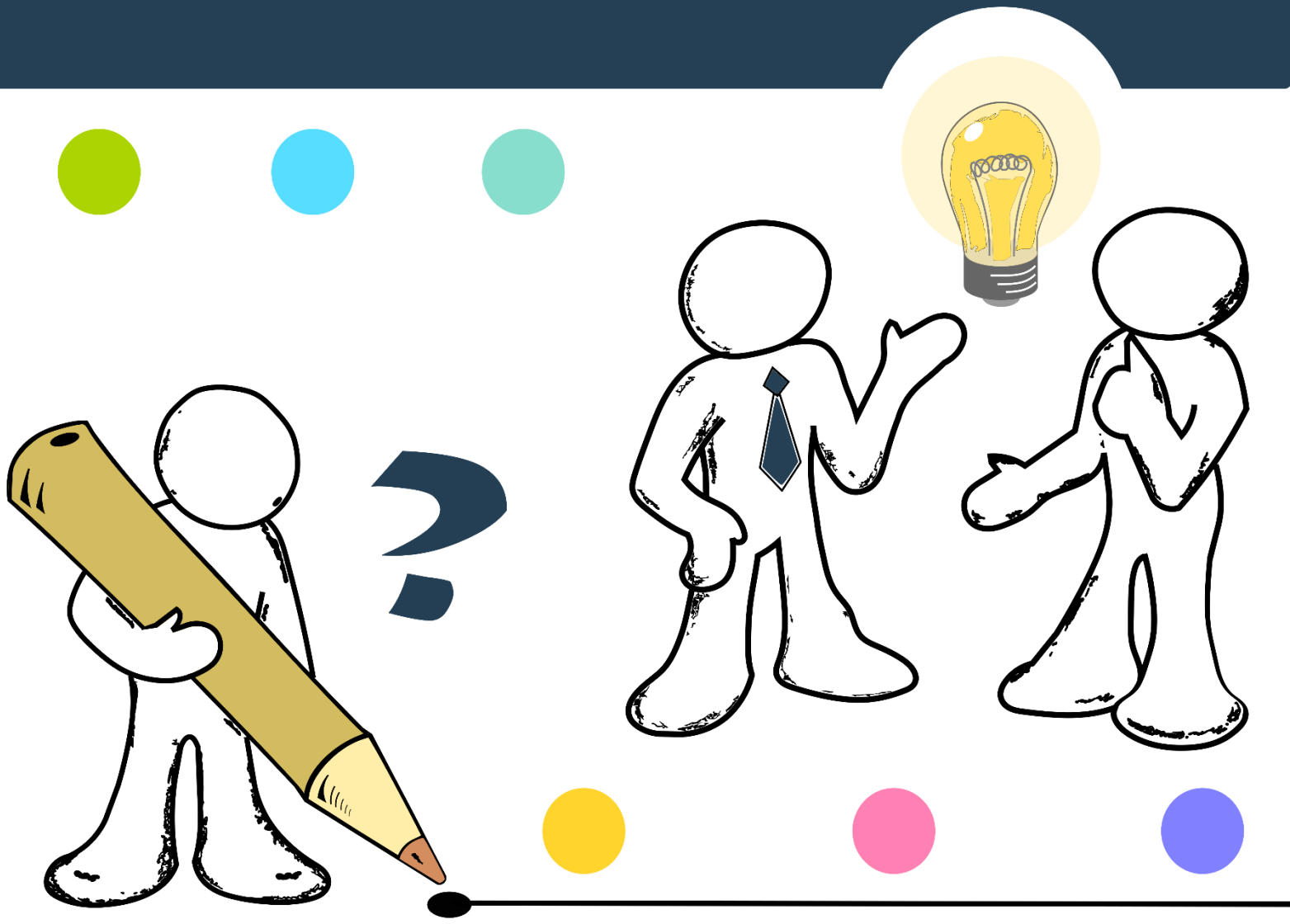
Ievads prasību inženierijā



Sergejs Kodors

Sergejs KODORS. 2019. *Ievads prasību inženierijā*. Rēzekne: Rēzeknes Tehnoloģiju akadēmija. 68 lpp.

Recenzenti:

- Dr. ing. sc. **Pēteris GRABUSTS** (Rēzeknes Tehnoloģiju akadēmija)
- Dr. ing. sc. **Imants ZAREMBO** (SIA "Soaphog")

Mācību līdzeklis sagatavots un izdots ar Rēzeknes Tehnoloģiju akadēmijas finansiālo atbalstu.

RĒZEKNES TEHNOLOĢIJU AKADĒMIJA

Publicēšanai rekomendējusi Rēzeknes Tehnoloģiju akadēmijas Studiju padome 2018. gada 11. septembrī.

Redaktore: **Vita Ansone**

Vāka autors: **Sergejs Kodors**

ISBN 978-9984-44-226-6

Saturs

Ievads

Informācijas sistēma ir strukturizēts informācijas tehnoloģiju un datu bāzu kopums, kuru lietojot, tiek nodrošināta noteiktu funkciju izpildei nepieciešamās informācijas ierosināšana, radīšana, apkopošana, uzkrāšana, apstrādāšana, izmantošana un iznīcināšana.

Bet sistēma nav vienkārši instruments, – tā ietekmē lietotāju. Kad ar sistēmu sāk strādāt cilvēks, viņam veidojas subjektīvs viedoklis par sistēmu, kas izpaužas, lietojot tādus sistēmas īpašību apzīmējumus kā pievilcīga, ērta, droša, lietderīga utt.

Sistēmas izstrāde ir līdzīga mākslai – nav pietiekami vienkārši realizēt sistēmu, lietotājam jābūt vēlmei strādāt ar to. Realizējot sistēmu, vienmēr jādomā par to, kā lietotājs strādās ar sistēmu, ko viņš vēlas un kā šī sistēma spēs apmierināt viņa prasības. Ja sistēma ir paredzēta vienam lietotājam, izstrādātājam jānodrošina komunikācija tikai starp sistēmu un lietotāju. Ja sistēmu izmantos vairāki lietotāji, plānošanas process kļūst daudz sarežģītāks, jo izstrādātājam arī jādomā par informācijas apriti starp vairākiem lietotājiem.

Ja sistēmas projektēšanā tiek pielietotas lietošanas gadījumu diagrammas, biznesa un sistēmas analītiķi apraksta sistēmu kā saites starp izpildītājiem un sistēmas funkcijām. Izpildītājam (angliski *actor*) saskaņā ar *Oxford* vārdnīcu ir divas semantiskās nozīmes:

- cilvēks, kurš filmējas vai uzstājas televīzijas raidījumā *(A person whose profession is acting on the stage, in films, or on television*);
- dalībnieks procesā vai darbībā (*A participant in an action or process*).

No šīs definīcijas var noprast, ka pirmais skaidrojums atbilst vārdam “aktieris”, otrais – “izpildītājs”. Taču, neskatoties uz to, ka tie ir dažādu nozīmju vārdi, informācijas sistēmu jomā starp tiem pastāv zināma sakarība, jo gala produkts veido virtuālo pasauli, kur sistēmas lietotāji spēlē savas lomas, veidojot informācijas apriti un savstarpējo komunikāciju. Rezultātā bezgalīgi dažādi sistēmas parametri var radīt dažādu cilvēka attieksmi pret sistēmu, tāpēc izstrādātājam jārada tāda harmonija, lai sistēma kā gala produkts darbotos saskaņoti un efektīvi un apmierinātu pēc iespējas visas lietotāju prasības.

Sistēmu projektēšanā eksistē dažādas pieejas, kas izpaužas programmatūras izstrādes metodoloģijās un modelēšanas valodās. Nav iespējams viennozīmīgi pateikt, kura no tām ir sliktāka vai labāka. Tāpēc jāvadās no šādiem aspektiem:

- vai izstrādātājs pārzina, spēj un prot pielietot noteiktu metodi/ rīku;
- vai izstrādātājam izdevās efektīvi realizēt vēlamo sistēmu.

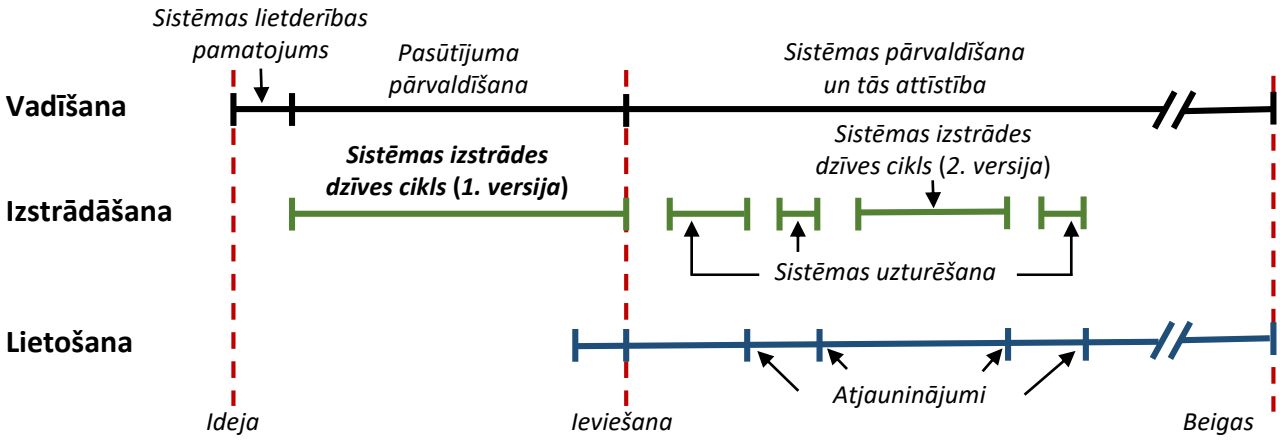
Metodoloģijas un modelēšanas valodas ir tikai izstrādātāja darba instrumenti. Pārzinot un spējot savlaicīgi un efektīvi pielietot tos, viņš var radīt vēlamo sistēmu. Turklāt efektīva sistēmas izstrādāšana paredz ne tikai teicama gala rezultāta sasniegšanu, bet arī ieguldīto līdzekļu, darba un laika apjoma lietderīgu izmantošanu.

Šīs grāmatas mērķis ir nevis iemācīt viennozīmīgi “pareizus” programmatūras prasību noteikšanas, aprakstīšanas un dokumentēšanas paņēmienus, bet dalīties pieredzē, lai topošais programmētājs spētu izveidot pamatu tālākai personīgai attīstībai un stimulētu labas prakses apgūšanu.

1. Sistēmas izstrādes dzīves cikls

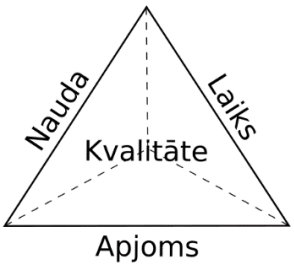
Sistēmas dzīves cikls ir sistēmas pastāvēšanas laiks no tās izstrādes procesa sākuma līdz brīdim, kad tā ir zaudējusi savu praktisko vērtību.

Sistēmas dzīves cikla pārvaldīšana izpaužas trijos aspektos (*Chappell*, 2008): vadīšana, izstrādāšana un lietošana. *1. attēlā* var redzēt sistēmas dzīves cikla pārvaldīšanas aspektus laikā.



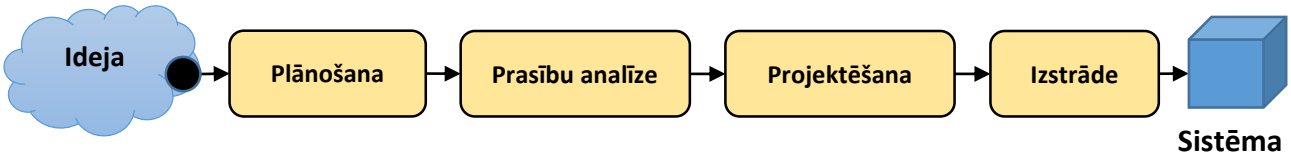
1. attēls. **Sistēmas dzīves cikla pārvaldīšanas aspekti laikā** (veidots pēc (*Chappell*, 2008))

Dzīves cikla sākumā ir ideja, kas tiek novērtēta kā jauna biznesa iespēja. Ja sistēmas lietderība ir pamatota un ir pieņemts lēmums izstrādāt to, projektam tiek izdalīts finansējums un tiek uzsākta sistēmas izstrāde. Sistēmas izstrādes pamatā ir projekta "trīsstūris", kas izpaužas kā līdzsvars starp apjomu, naudu un laiku (skat. *2. attēlu*). Lai veiksmīgi realizētu projektu, ir svarīgi pārvaldīt projekta apjomu, jo, pamainot vismaz vienu trīsstūra sastāvdaļu (naudu, laiku vai apjomu), uzreiz tiek aizskarti arī pārējie. Projekta apjomam ir arī zināms kvalitātes līmenis, kas tiek uzskatīts par neapstrīdamu (*Getchell u.c.*, 2002). Šo kvalitātes līmeni nosaka sistēmas prasības. Ja tās netiek izpildītas, sistēmu nevar uzskatīt par pilnvērtīgu vai pabeigtu, tāpēc kvalitāte ir ceturtā projekta "dimensija" (*Getchell u.c.*, 2002) (skat. *2. attēlu*).



2. attēls. **Projekta trīsstūris**

Sistēmas izstrādes dzīves ciklā parasti izdala četrus posmus (skat. *3. attēlu*) (*Dennis, Wixom & Roth*, 2012): (1) plānošanu, (2) prasību analīzi, (3) projektēšanu un (4) izstrādi. Katrs posms tiek raksturots ar vairākiem citiem procesiem, kas ir atkarīgi no izvēlētās izstrādes metodoloģijas un vēlamajiem gala rezultātiem.



3. attēls. **Sistēmas izstrādes dzīves cikls** (veidots pēc (*Dennis, Wixom & Roth*, 2012))

Plānošanas fāze ir ļoti svarīgs posms, lai noskaidrotu, ***kāpēc*** šī sistēma ir **vajadzīga** un ***kā*** izstrādātāju grupa plāno to **realizēt** (*Dennis, Wixom & Roth*, 2012). Šajā posmā tiek apzinātas biznesa prasības un uzdevumi, kā arī tiek noskaidroti šādi būtiski jautājumi:

- Vai sistēma ir lietderīga?
- Vai izstrādātāju grupa ir spējīga realizēt sistēmu?
- Vai plānojamā sistēma atbilst biznesa prasībām?

Prasību analīzē noskaidro, ***kas*** lietos sistēmu, ***kas*** jādara sistēmai, ***kur*** un ***kad*** sistēma tiks lietota. Šajā posmā tiek analizēta **eksistējošā sistēma** un noteikta **vēlamā sistēma**, tiek **vāktas un analizētas sistēmas prasības** (*Dennis, Wixom & Roth*, 2012).

Projektēšanas posmā plāno sistēmu atbilstoši prasībām, aprakstot sistēmas arhitektūru, datu bāzes struktūru, saskarnes, biznesa procesus utt.

Izstrādes posmā sistēma tiek izstrādāta, pārbaudīta un ieviesta ekspluatācijā.

Lai veiksmīgi pabeigtu projektu, ir svarīgi, lai pasūtītāja prasības atbilstu projekta realizēšanas iespējām. Tāpēc nepieciešams skaidrs priekšstats par gaidāmo sistēmu, jo, pat precīzi nosakot projekta apjomu, ir jābūt gatavam to koriģēt realizācijas laikā. Ja projekta apjoms vispār ir noteikts tikai aptuveni – labojumi var būt pārāk lieli, lai veiksmīgi vai vispār pabeigtu projektu.

2. Sistēmas izstrādes modeļi

Eksistē vairāki sistēmas izstrādes modeļi, kas nosaka sistēmas izstrādes aktivitātes, secību un uzdevumus, lai organizētu skaidru, saskaņotu, efektīvu un vadāmu izstrādes procesu. Grāmatā tiek apskatīti tikai četri modeļi – "Ūdenskrituma modelis", "Spirāles modelis", "Inkrementālais modelis" un *"Microsoft Solution Framework" (MSF)* – vairāku iemeslu dēļ.

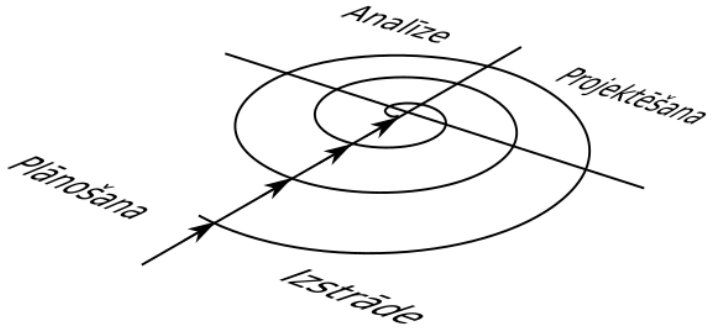
- *Ūdenskrituma modelis* palīdz paskaidrot sistēmas izstrādes galvenos posmus.
- *Spirāles modelis* liek ievērot nenoteiktības un neprecizitātes, kas rodas sistēmas izstrādes laikā.
- *Inkrementālais modelis* ļauj realizēt sistēmu pa daļām, nevis pārvaldīt visu sistēmu uzreiz.
- *MSF* apvieno labākās iepriekš minēto modeļu īpašības.

Ūdenskrituma modelis paredz sistēmas izstrādes procesu ar striktu secību, kad visi izstrādes posmi (plānošana, prasību analīze, projektēšana un izstrāde) tiek izpildīti viens pēc otra (skat. *3. attēlu*) (*Bell*, 2005; *Roger & Maxim*, 2014) bez iespējas atgriezties pie iepriekšējās fāzes, un katrs posms tiek raksturots ar konkrētu gala rezultātu. *Ūdenskrituma modelim* piemīt šādi trūkumi (*Roger & Maxim*, 2014):

- praksē projektus ļoti reti var realizēt secīgi;
- klientiem dažkārt grūti definēt visas savas prasības uzreiz;
- modelis neparedz iespēju mainīt vai labot iepriekšējos posmus;
- sistēma klientam nav pieejama līdz pēdējam realizācijas brīdim.

Tomēr tā ir laba izvēle, ja prasības ir skaidri un stingri noteiktas un izstrādes procesam jānotiek lineārā secībā (*Getchell* u.c., 2002; *Roger & Maxim*, 2014). Šim modelim ir izglītojošs raksturs, jo tam ir vienkārša struktūra (*Bell*, 2005), kas labi parāda izstrādes projekta realizācijas posmus.

Spirāles modelis ļauj izstrādāt sistēmu pakāpeniski, lai pārvarētu projekta neprecizitātes un neskaidrības. Visi izstrādes posmi tiek atkārtoti vairākas reizes (skat. *4. attēlu*), precizējot sistēmas prasības.



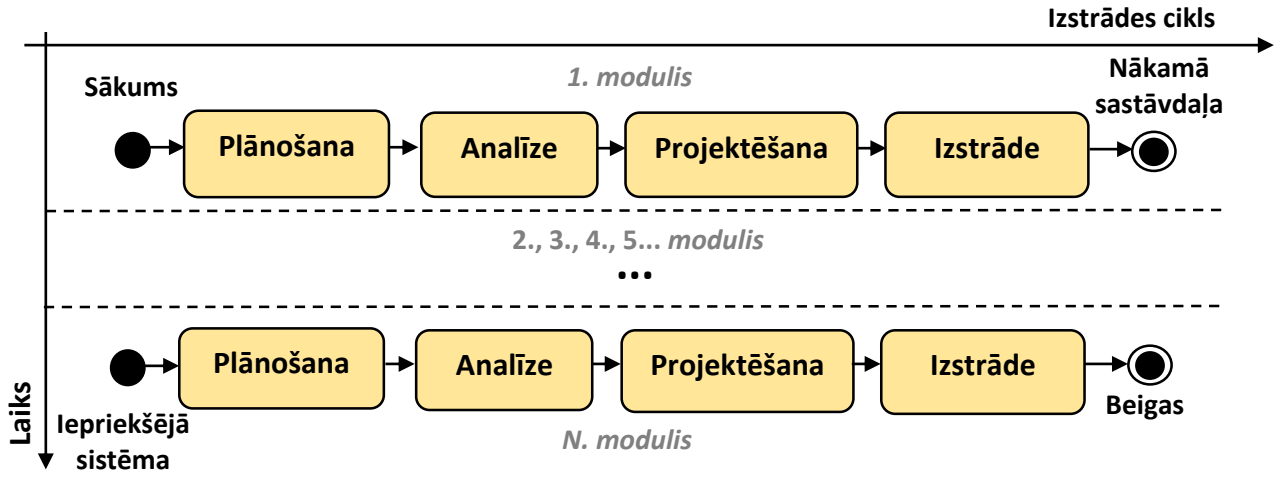
4. attēls. **Spirāles modelis**

Piemēram, pirmajā izstrādes ciklā tiek realizēts sistēmas prototips, kas pakāpeniski tiek uzlabots, kamēr netiek izstrādāta vēlamā sistēma. Šajā gadījumā pasūtītājs var savlaicīgi ieraudzīt sistēmu un precizēt savas prasības.

Salīdzinot ar *Ūdenskrituma modeli*, *Spirāles modelim* ir šādi trūkumi:

- sistēma tiek pārstrādāta vairākas reizes;
- modelis neļauj izstrādāt pārdomātu sistēmu, tāpēc sistēma var funkcionēt neefektīvi.

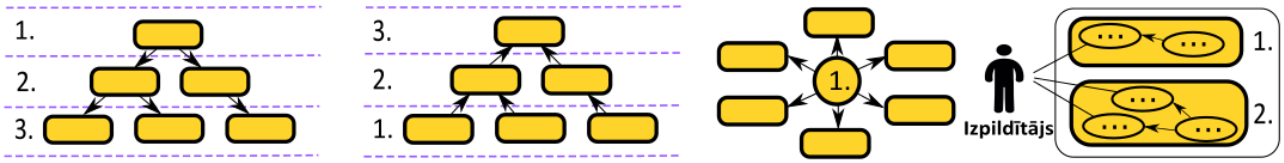
Inkrementālais modelis piedāvā izstrādāt sistēmu pa daļām (skat. *5. attēlu*) (*Bell*, 2005). Šī pieeja ir efektīva lielajos projektos, kad visas prasības ir skaidri un precīzi nodefinētas, tomēr sistēma jārealizē ātri un pakāpeniski (*Roger & Maxim*, 2014).



5. attēls. **Inkrementālais modelis**

Eksistē dažādi veidi, kā organizēt sistēmas moduļu izstrādi (*Bell*, 2005):

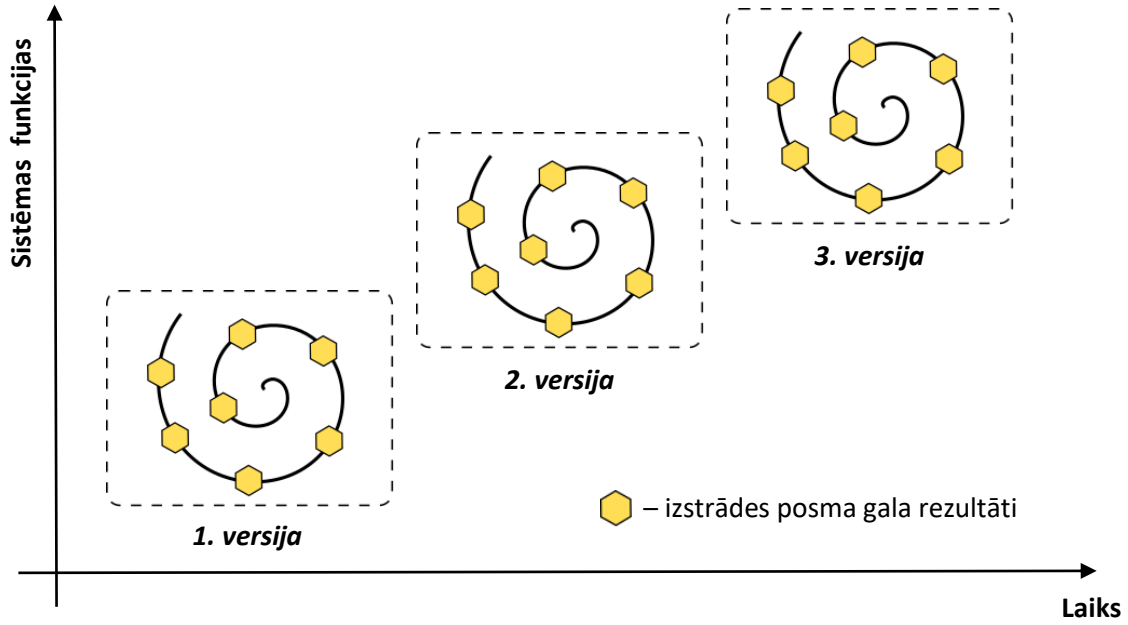
- "no augšas uz leju" – realizācija sākas ar komponentu hierarhijas plānošanu. Sistēmas izstrāde ir līdzīga prototipēšanai – sākumā tiek izstrādāti augšējā slāņa komponenti, pēc tam to sastāvdaļas, pakāpeniski papildinot sistēmu ar funkcijām. Šī pieeja ļauj pasūtītājam novērtēt visu sistēmu uzreiz (skat. *6. a) attēlu*);
- "no apakšas uz augšu" – realizācija arī balstās uz komponentu hierarhiju, tomēr komponentu izstrāde sākas no mazākām sastāvdaļām, kas vēlāk tiek apvienotas kopā (skat. *6. b) attēlu*);
- izstrādājot "no vidus" – realizācijas pamatā ir arhitektūra ar pamatelementu, kas tiek realizēts pirmais un vēlāk tiek paplašināts (skat. *6. c) attēlu*);
- izmantojot lietošanas gadījumus – sistēma tiek pakāpeniski izstrādāta, realizējot lietošanas gadījumu grupas (funkciju kopas), (skat. *6. d) attēlu*).



6. attēls. **Inkrementālais modelis**

Kā šī modeļa trūkumu var minēt to, ka ne visos gadījumos var realizēt sistēmas arhitektūru, kas atbilstu inkrementālajam modelim, izņemot izstrādi ar lietošanas gadījumu izmantošanu. Pakāpeniska lietošanas gadījumu realizācija ļauj pasūtītājam ātri novērtēt sistēmas funkcijas un savlaicīgi ieviest izmaiņas. Realizācija ar moduļiem ļauj tos konstruēt arī paralēli, tomēr paralēla izstrāde paredz riskus un problēmas savienošanas fāzē.

***Microsoft Solution Framework* (*MSF*)** (*Getchell* u.c., 2002) saskaņā ar *MSF* balto grāmatu apvieno labākās *Ūdenskrituma* un *Spirāles modeļa* īpašības. No *Ūdenskrituma modeļa* ir pārmantoti gala rezultāti, kas nosaka pāreju no viena posma uz otru, piemēram, programmatūras prasību specifikācija. No *Spirāles modeļa* – pastāvīga prasību precizēšana, kas veido sinerģiju starp izstrādātāju un pasūtītāju, jo pasūtītājs piedalās izstrādē visās projekta stadijās. *MSF* rekomendē izstrādāt sistēmas prasības vairākās versijās, sākot realizāciju no galvenajām funkcijām (skat. *7. attēlu*). Versiju galvenā priekšrocība ir iespēja izmantot tās biznesa procesā, kamēr visa sistēma vēl nav pabeigta.



7. attēls. **Versiju izstrāde, pielietojot *Spirāles* un *Ūdenskrituma modeli***
(veidots pēc (*Getchell* u.c., 2002))

No katra modeļa var mācīties labu praksi, kas jāievēro sistēmas realizācijas procesā:

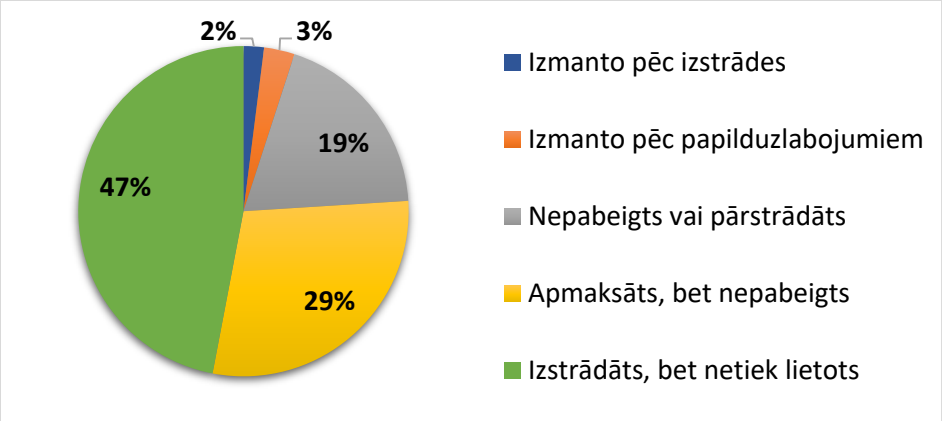
- *Ūdenskrituma modelis* – lai sistēmas izstrāde nebūtu haotiska (būtu pārvaldāma), jāizvēlas tāds izstrādes modelis, kas ļauj skaidri izprast katra posma uzdevumus. Lai pārliecinātos, ka posms ir pabeigts, jābūt definētam konkrēta posma gaidāmajam rezultātam.
- *Spirāles modelis* – lai realizētu vēlamo sistēmu un likvidētu pārpratumus, sistēmas prasības pastāvīgi jāsaskaņo ar pasūtītāju, pakāpeniski izstrādājot un demonstrējot sistēmu.
- *Inkrementālais modelis* – sistēma jārealizē pa daļām – pārvaldīt mazas daļas ir daudz vienkāršāk nekā visu sistēmu uzreiz. Lietošanas gadījumu diagramma ir labs instruments, lai identificētu sistēmas daļas.
- *MSF* – versiju izstrāde ir laba pieeja, kā apvienot visas iepriekš minētās rekomendācijas, ļaujot pasūtītājam daudz ātrāk iegūt izstrādes rezultātus un peļņu no tā. Tā arī motivē labākai sadarbībai.

3. Prasību analīze – veiksmīgas sistēmas pamats

Sistēmu var uzskatīt par veiksmīgi pabeigtu, ja tiek izpildīti divi nosacījumi:

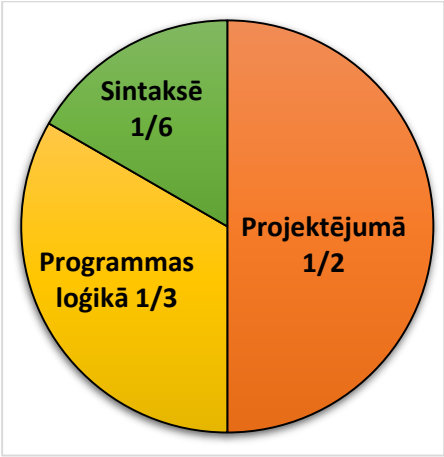
- sistēma strādā korekti, un visas obligātās funkcijas ir realizētas;
- sistēma atbilst pasūtītāja prasībām un vēlmēm.

Izstrādājot lielas sistēmas, apmēram 29 % projektu netiek pabeigti vispār, 47 % – tiek piegādāti pasūtītājam, bet praktiski netiek izmantoti (skat. *8. attēlu*) neveiksmīgas prasību analīzes dēļ (*Bell*, 2005).



8. attēls. **Projektu īstenošanas statistika** (veidots pēc (*Bell*, 2005))

Ja kļūdas netiek atklātas un izlabotas uzreiz, to labošana nākamajos posmos kļūst dārgāka. Vislielākais kļūdu skaits tiek pieļauts sistēmas projektējumā (sistēmas arhitektūra, lietotāju saskarnes, biznesa procesi utt.), bet projektējuma kļūdu labošanas process ir dārgāks nekā programmas koda atkļūdošana (skat. *9. attēlu*) (*Bell*, 2005).



9. attēls. **Projekta kļūdu proporcionālais sadalījums** (veidots pēc (*Bell*, 2005))

Projekta izstrādes laikā vissarežģītākais un vissvarīgākais ir saprast, kāda sistēma jārealizē, jo, izvēloties nepareizu mērķi, gala rezultātā var būt tikai divi varianti: nepabeigta sistēma vai veiksmīgi pabeigta nevajadzīga sistēma.

4. Prasību analīzes process

Lai sistēma atbilstu lietotāja / pasūtītāja prasībām un vēlmēm, jāveic detalizēta prasību analīze. Tās rezultāti tiek apkopoti programmatūras prasību specifikācijā, kas ir pamatdokuments, lai (*Getchell* u.c., 2002; *PMI*, 2015):

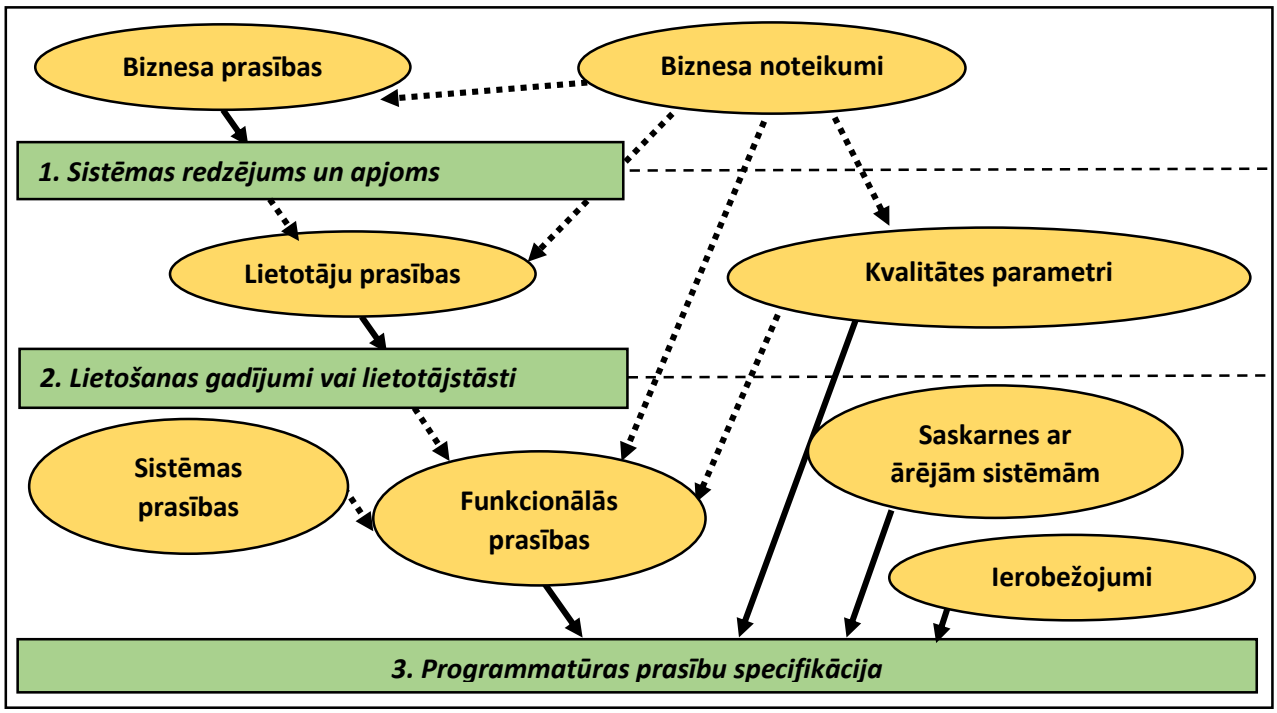
- saskaņotu ar pasūtītāju, kāda sistēma tiks izstrādāta;
- aprakstītu, kāda sistēma jāizstrādā;
- novērtētu darbietilpību;
- saskaņotu izstrādātāju darbu;
- izstrādātu pārējo dokumentāciju (lietotāju instrukcijas, programmatūras projektējuma aprakstu utt.).

Sistēmas prasība ir apgalvojums, kas apraksta kādu būtisku sistēmas īpašību vai funkciju.

Visas sistēmas prasības var iedalīt trīs līmeņos (*Wiegers & Beatty*, 2013).

1. ***Biznesa prasības*** nosaka, ko pasūtītājs vēlas iegūt, radot šo sistēmu – tie ir biznesa uzdevumi, kas jāatrisina izstrādājamajai sistēmai.
2. ***Lietotāja prasības*** ir darbi un uzdevumi, ko var veikt lietotājs, pielietojot sistēmu biznesa procesu izpildē.
3. ***Funkcionālās prasības*** ir detalizēts apraksts, kā sistēma apstrādās ievades procesu un kāds tam būs rezultāts.

Kamēr izstrādātāju grupa plāno un projektē sistēmu, sastādot programmatūras prasību specifikāciju, ļoti svarīgi saglabāt funkciju atbilstību sistēmas prasībām (*Getchell* u.c., 2002). Tāpēc, lai izstrādātu programmatūras prasību specifikāciju, kas atbilstu pasūtītāja vajadzībām, un realizētu atbilstošas sistēmas funkcijas, jāsāk ar biznesa prasību analīzi, pakāpeniski pārvēršot tās par sistēmas funkcionālajām prasībām (skat. *10. attēlu*).



10. attēls. **Programmatūras prasību specifikācijas saturs** (veidots pēc (*Wiegers & Beatty*, 2013))

Papildus katra sistēma paredz **nefunkcionālās prasības**, kas var ietekmēt kādas funkcionālās prasības realizācijas veidu (skat. *10. attēlu*).

Nefunkcionālās prasības veido (*Wiegers & Beatty*, 2013):

- biznesa noteikumi – organizācijas kultūra, korporatīvā politika, valsts regulas un industrijas standarti;
- kvalitātes parametri – lietotājiem un izstrādātajiem svarīgas sistēmas īpašības, kā veiktspēja, drošība, pieejamība un pārnesamība;
- saskarne ar ārējām sistēmām – sistēmas savienojamība ar citām ārējām sistēmām.

Vispirms jānosaka biznesa prasības – kāpēc sistēma ir vajadzīga? Kādas problēmas sistēmai jāatrisina? Kādas jaunas iespējas jāveido? – pakāpeniski pārvēršot biznesa prasības par sistēmas funkcijām. Ja sāk ar sistēmas funkcijām un pēc tam tās pielāgo biznesa prasībām, pastāv risks, ka sistēmu nevarēs izmantot.

5. Prasību vākšanas metodes

Visus projektus var sadalīt divās kategorijās: produkts un pasūtījums. Produkta gadījumā izstrādātājs pats izvirza sistēmas prasības ar mērķi izstrādāt sistēmu, ko kāds vēlēsies nopirkt. Pasūtījuma gadījumā ir otrādi – prasības definē pasūtītājs, izstrādātājiem tās jāsaprot un jārealizē sistēma.

Eksistē dažādas prasību vākšanas metodes, katrai no tām ir savi plusi un mīnusi. Produkta ražošanai raksturīga ideju meklēšana, SVID analīze, apspriedes darba grupā, prototipēšana, aptaujas/ statistika un sadarbība ar ekspertiem. Pasūtījumam labāk noderēs intervijas, dokumentācijas analīze, darba un biznesa procesu novērošana, personīgā pieredze, piedaloties biznesa procesos, un prototipēšana.

Iesācējiem būs noderīgas trīs metodes:

- eksistējošo sistēmu analīze;
- intervijas;
- prototipēšana.

Eksistējošo sistēmu analīze ir būtiska, jo informācijas tehnoloģija tiek plaši pielietota cilvēku ikdienā, gandrīz visās tautsaimniecības jomās tiek kaut kas izmantots no tās, tāpēc, veicot līdzīgu sistēmu analīzi, var iegūt pieredzi un idejas. Izstrādājot produktu, jārēķinās ar līdzīgiem tirgū esošajiem produktiem, nosakot sava un konkurentu produkta vājās un stiprās puses. Šim mērķim var noderēt SVID analīze (stipro un vājo pušu, iespēju un draudu analīze), kas paredz aizpildīt tabulu ar četrām nodaļām (skat. *1. tabulu*). SVID tabulā jāatzīmē gan iekšējie, gan ārējie faktori. Ja sistēma tiek realizēta pēc pasūtījuma, pasūtītājs var minēt līdzīgas sistēmas un paskaidrot, kāpēc eksistējošās sistēmas viņu neapmierina. Tirgū esošās sistēmas var izmantot ideju ģenerēšanai un labu/ sliktu piemēru analīzei.

1. tabula

SVID tabula

Stiprās puses	**Vājās puses**
[Projekta pozitīvās īpašības, kas palīdz sasniegt mērķi]	[Projekta trūkumi, kas traucē sasniegt mērķi]
Iespējas	**Draudi**
[Ārējie faktori, kas sekmē mērķa sasniegšanu]	[Ārējie faktori, kas apdraud vai traucē mērķa sasniegšanai]

Pasūtījuma gadījumā ***intervija*** ir intuitīvs risinājums. Intervija paredz piecus posmus (*Dennis, Wixom & Roth*, 2012): sagatavot jautājumus, izvēlēties respondentu, sagatavoties intervijai, veikt intervēšanu un dokumentēt interviju.

Visi jautājumi var tikt iedalīti trīs grupās:

- *aizvērtie jautājumi* – respondentam jāatbild konkrēti uz jautājumu, piemēram, "Kādu informāciju glabās sistēma?";
- *atvērtie jautājumi* – jautājums izraisa diskusiju, piemēram, "Ar kādām problēmām Jūs saskarieties, veicot savus darba pienākumus?";
- *precizējošie jautājumi* – nepieciešams precizēt iepriekšējo atbildi vai ir radušās neskaidrības atbildē. Tādā gadījumā var palīdzēt šādi jautājumi - ierosinājumi: "Kāpēc?", "Piemēram!" vai "Lūdzu, detalizētāk!".

Lai sagatavotu strukturētu interviju ar aizvērtajiem jautājumiem, nepieciešams daudz vairāk laika nekā sagatavot nestrukturētu interviju ar atvērtajiem jautājumiem, tāpēc iesācēji izvēlas vieglāko ceļu, cerot, ka

viss notiks vienkārši. Tomēr tas ir neapdomāts risinājums, jo pirmajā reizē neiegūtā informācija būs jānoskaidro nākamajā reizē, bet daudziem cilvēkiem nepatīk, ja atkārtoti jautā par vienu un to pašu tēmu.

Pirms tikšanās respondents jau iepriekš ir jāiepazīstina ar intervijas mērķi (nevis ar katru jautājumu, bet ar intervijas tēmām), lai respondents varētu sagatavoties (atrast materiālus, paraugus; sagatavot prezentāciju; uzaicināt citu personu, kas spētu atbildēt uz jautājumiem utt.).

Pēc intervijas visas atbildes obligāti jādokumentē, lai vēlāk varētu pārskatīt savākto informāciju. Dokumentētās atbildes ir jāiedod pārbaudīt respondentam, lai izvairītos no nepareizi saprastām vai pārprastām domām. Interviju dokumentēšanai var izmantot 2. tabulā doto struktūru.

2. tabula

Intervijas lapas struktūra (*Dennis, Wixom & Roth*, 2012)

Respondents	[Vārds Uzvārds] [Amats] [Struktūrvienība]
Intervētājs	[Vārds Uzvārds]
Intervijas uzdevumi	[Jautājumi, kas bija jānoskaidro intervijas laikā]
Intervijas kopsavilkums	[Kodolīgs atbilžu formulējums]
Atvērtie jautājumi	[Kādi jautājumi/ neskaidrības radās intervijas laikā vai pēc tās]
Detalizētas piezīmes	[Detalizēta intervijas dokumentācija]

Prototipēšana paredz vienkāršotas sistēmas izstrādi, lai lietotājs pēc iespējas ātrāk varētu novērtēt sistēmu, dažādus tās variantus un dot savu atsauksmi (*Dennis, Wixom & Roth*, 2012). Vienkāršākais prototipa veids ir saskarņu uzmetumi jeb horizontālā prototipēšana, kas tiek koncentrēta tā, lai pasūtītājs varētu novērtēt sistēmas ārējo izskatu, pieejamo funkciju daudzumu, opciju izvietojumu un navigāciju (*Wiegers & Beatty*, 2013). Tomēr mūsdienās eksistē rīki, kas ļauj veidot interaktīvus variantus un veikt koncepcijas pārbaudi (vertikālā prototipēšana), lai novērtētu sistēmas funkcionalitāti un uzdevumu veikšanas ātrdarbību salīdzinājumā ar veco risinājumu (*Wiegers & Beatty*, 2013). Izveidotais prototips ļauj lietotājam novērtēt sistēmu darbībā, kas dod iespēju noteikt, vai visi dati ir paredzēti, lai izpildītu uzdevumu, vai datu ievadīšanas process notiek korekti, vai risinājums ir labāks utt. (*Roger & Maxim*, 2014).

Iesācēja minimālais prasību vākšanas metožu komplekts var izskatīties šādi:

- **eksistējošu sistēmu analīze** – esošās sistēmas ir labu un sliktu risinājumu pieredze, kas ļauj mācīties no citiem un attīstīt jaunas idejas;
- **intervija** – cilvēkam ir raksturīgi neskaidrības gadījumā jautāt citiem, tomēr iesācējiem īpaši cītīgi jāplāno intervijas struktūra un jāgatavo jautājumi;
- **prototipēšana** – sistēmas prototipa izveide ļauj būtiski novērst neskaidrības un saskaņot izstrādātāja un lietotāja sistēmas domas/ vēlmes.

6. Biznesa prasības un sistēmas redzējums

Viens no veidiem, kā noteikt sistēmas biznesa prasības, ir *PIECES* modelis (*Gupta*, 2005). Saskaņā ar to tiek rastas atbildes par to, kā sistēma uzlabo uzņēmējdarbību atbilstoši katrai kategorijai.

P *Veiktspēja* (*performance*) – nepieciešamība paātrināt ražošanu vai pakalpojuma izpildes laiku (palielināt caurlaidspēju vai samazināt apstrādes laiku).

I *Informācija un dati* (*information and data*) – nepieciešamība uzlabot vai radīt jaunas informācijas pārvaldi.

E *Ekonomika* (*economics*) – nepieciešamība samazināt vai labāk pārvaldīt uzņēmējdarbības izdevumus; palielināt pārdošanas apjomus vai ienākt jaunā tirgū.

C *Pārvalde un drošība* (*control and security*) – nepieciešamība radīt vai uzlabot ražošanas, kvalitātes, darba, piekļuves un tiesību vai citu procesu pārvaldi vai uzskaiti.

E *Produktivitāte* (*efficiency*) – nepieciešamība uzlabot darbinieku vai iekārtu darba efektivitāti.

S *Pakalpojums* (*service*) – nepieciešamība radīt jaunu vai uzlabot eksistējošā pakalpojuma kvalitāti.

Lai korekti noformulētu biznesa prasības un uzdevumus, lieto *SMART* modeli (*PMI*, 2015).

S *Konkrēts* (*specific*) – formulējumam jābūt skaidram, kodolīgam un saprotamam.

M *Izmērāms* (*measurable*) – rezultātam jābūt pārbaudāmam.

A *Sasniedzams* (*achievable*) – prasībai/ uzdevumam jābūt izpildāmam.

R *Atbilstošs* (*relevant*) – prasībai/ uzdevumam jāatbilst biznesa mērķim.

T *Savlaicīgs* (*time-bound*) – prasībai jābūt aktuālai.

Kad biznesa problēmas un vajadzības ir noteiktas, rodas sistēmas redzējums, kas atbilst biznesa prasībām. Lai īsi aprakstītu sistēmas redzējumu, var aizpildīt 3. tabulas anketu (*Wiegers & Beatty*, 2013).

3. tabula

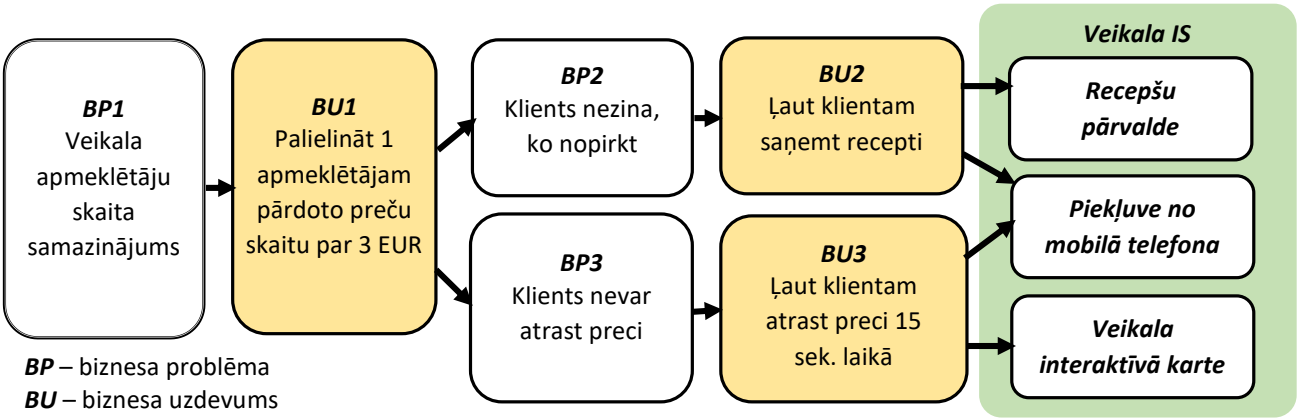
Īss redzējuma apraksts

Kas	[Produkta nosaukums]
Ir	[Produkta kategorija]
Kam	[Mērķauditorija]
Kāpēc	[Lietotāju vajadzības un vēlmes]
Tāpēc	[Ko piedāvā produkts]
Neskatoties, ka	[Konkurējošie produkti vai risinājumi]
Jo	[Šī produkta priekšrocības un atšķirības]

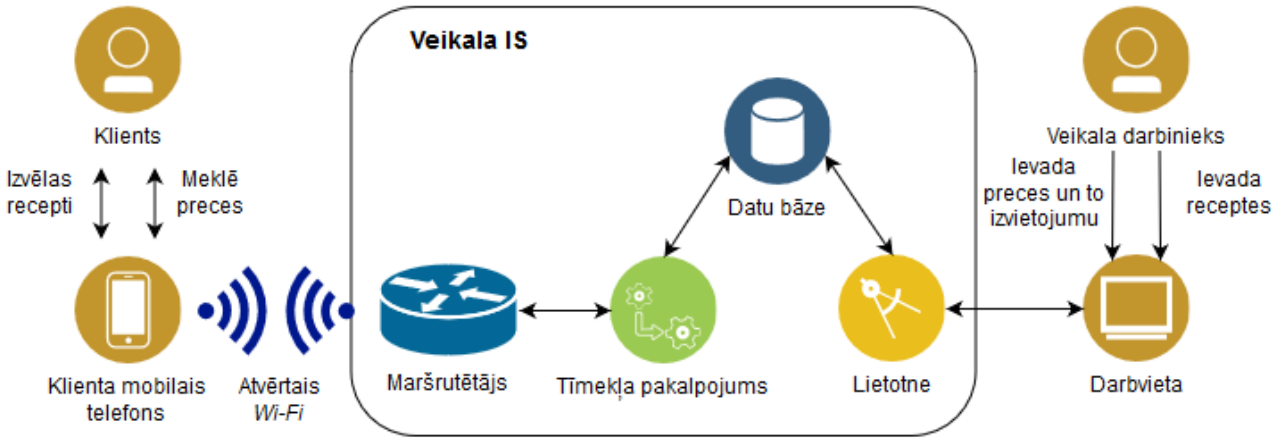
Mērķauditorija anketā ir cilvēku, uzņēmumu vai sistēmu grupas, kas ir ieinteresētas izstrādājamās sistēmas tapšanā un kas reglamentē biznesa prasības. Tomēr sistēma pati par sevi nevar eksistēt, kādam jāuztur un jālieto tā. Personas, speciālisti vai citas sistēmas, kas lieto šo sistēmu, veido lietotāju grupu, kas ir mērķauditorijas pārstāvji. Sākotnējā posmā ir svarīgi identificēt visus lietotājus, viņu īpašības, darba specifiku un vidi, jo tas būtiski var izmainīt risinājumu, piemēram, darbinieks var strādāt pie darbgalda vai ievadīt informāciju āra apstākļos. Otrkārt, lietotāji ir intervijas objekti, kas palīdz noskaidrot lietotāju prasības.

Lai labāk paskaidrotu sistēmas redzējumu, var uzzīmēt mērķa modeli un sistēmas uzmetumu.

- **Mērķa modelis** – diagramma, kas attēlo biznesa problēmas, uzdevumus un rīkus. Šī diagramma palīdz labāk apzināt biznesa prasības un noteikt darba apjomu un sistēmas sastāvdaļas (*PMI*, 2015). Piemēru var redzēt *11. attēlā.*
- **Sistēmas uzmetums** – diagrammas mērķis ir attēlot sistēmas koncepciju: galvenās sastāvdaļas, citas sistēmas un lietotājus. Piemērs ir dots *12. attēlā*.



11. attēls. **Pārtikas veikala mērķa modelis** (veidots pēc (*PMI*, 2015))



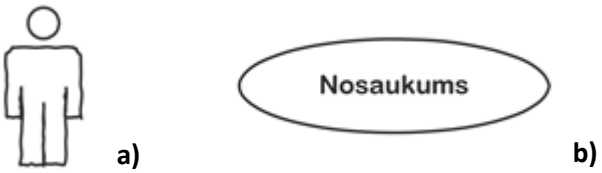
12. attēls. **Pārtikas veikala sistēmas uzmetums atbilstoši mērķa modelim**

Sistēmas redzējumu var pielietot programmatūras prasības specifikācijas ievada daļas noformēšanai (nodaļa "Darbības sfēra" un "Vispārējais apraksts").

7. Lietotāju prasību analīze

Lietotāju prasības ir uzdevumi, ko var veikt sistēmas lietotāji, lai izpildītu biznesa prasības. *Lietošanas gadījumi* (*use-cases*) ir rīks, kuru izgudroja Ivars Jakobsons (*Ivar Jackobson*) (*Kettenis*, 2007), lai aprakstītu lietotāja veicamās darbības sistēmā. Tas pieder pie objektorientētās modelēšanas, ir iekļauts sistēmu modelēšanas valodā *UML* (*Unified Modeling Language*) (*Dennis, Wixom & Roth*, 2012) un veido sistēmas analīzes un projektēšanas pamatu, izstrādājot tādas sistēmas sastāvdaļas kā lietotāja saskarnes, datu un procesu modeli (*Cockburn*, 2000). Lietošanas gadījumi ir jaudīgs rīks, lai aprakstītu biznesa lietotnes, bet nav piemērots lielu datu krātuvju vai lietotņu ar intensīviem skaitļošanas procesiem aprakstīšanai (*Dennis, Wixom & Roth*, 2012) – gadījumos, ja nav vai ir minimāla lietotāja darbība ar sistēmu.

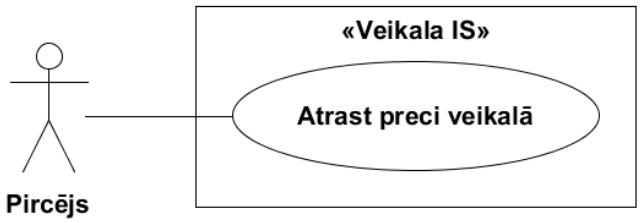
Lietošanas gadījumu modelēšanas pamatā ir divi pamatelementi: izpildītājs (*actors*) un lietošanas gadījumi (*use-cases*) (*Kettenis*, 2007). Izpildītājus apzīmē ar abstraktu cilvēka figūru (skat. *13.a) attēlu*), bet lietošanas gadījumu – kā ovālu ar veicamā uzdevuma nosaukumu (skat. *13.b) attēlu*).



13. attēls. **Lietošanas gadījumu diagrammas pamatelementi: a) izpildītājs; b) lietošanas gadījums**

Runājot par izpildītājiem, ir jāsaprot lietotāja loma (*user role*), nevis konkrēta persona vai darba amats, neskatoties uz to, ka tie var sakrist (*Dennis, Wixom & Roth*, 2012; *Kettenis*, 2007). Piemēram, cilvēks ar amatu "Administrators" reģistrējas cita uzņēmuma izglītības sistēmā, lai attālināti apgūtu mācību kursu, tad viņam tiks piešķirta loma *students* ar ierobežotu sistēmas funkciju daudzumu. Bet savā uzņēmumā viņš var uzturēt informācijas sistēmu un vienlaicīgi būt *sistēmas administrators* tajā. Izpildītāja lomas var pārņemt individuāli cilvēki, grupas, organizācijas, ierīces, datori vai citas sistēmas (*Cockburn*, 2000).

Lietošanas gadījumi ir stāsti, ko sistēmas lietotāji **var darīt** ar sistēmu (*Cockburn*, 2000), tāpēc lietošanas gadījumu diagramma ir lasāma un atbilst visiem gramatikas likumiem. Piemēram, *14. attēla* gadījums ir lasāms kā "Pircējs var atrast preci veikalā", kur asociācijas līnija apzīmē vārda *var* uzdevumu, veidojot šādu teikuma struktūru **[Loma/ izpildītājs]** ***var*** **[Lietošanas gadījums]**.



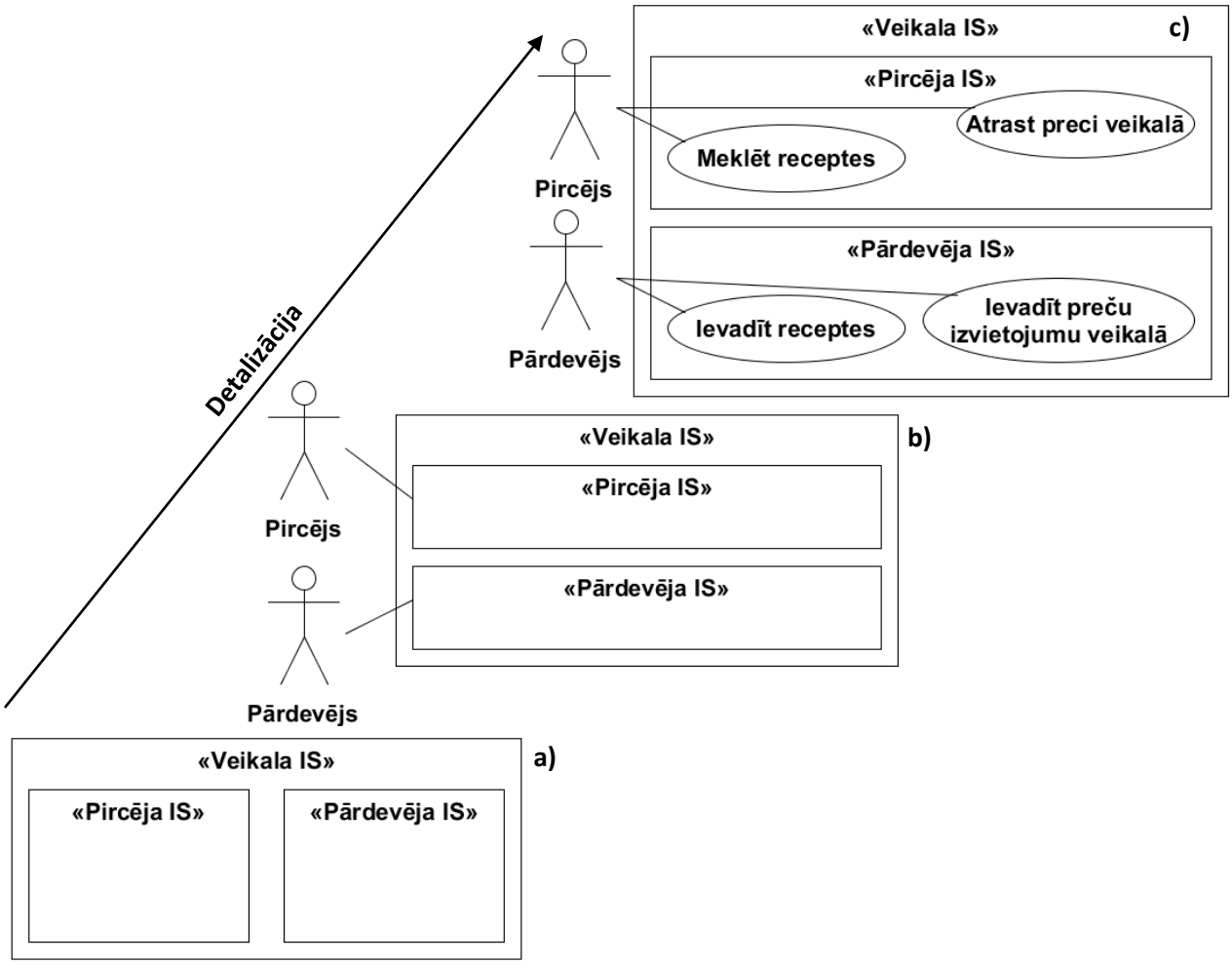
14. attēls. **Lietošanas gadījuma piemērs**

Lai sastādītu lietošanas gadījumu diagrammu, var secīgi atbildēt uz šādiem jautājumiem:

1) No kādām apakšsistēmām/ moduļiem sastāv sistēma?
2) Kas var lietot sistēmu?
3) Ko var darīt [izpildītājs] ar sistēmu?

Apskatot *6. nodaļas* piemēru ar *veikala IS* (skat. *12. attēlu*), atbildes uz iepriekš minētajiem jautājumiem varbūt šādas:

1) "No kādām apakšsistēmām sastāv veikala IS?" Sistēma sastāv no pircēja IS un pārdevēja IS (skat. *15.a) attēlu*).
2) "Kas var lietot sistēmu?" Sistēmu var lietot pircējs un pārdevējs (skat. *15.b) attēlu*).
3) "Ko var darīt pircējs ar sistēmu?" (skat. *15.c) attēlu*):
   - *pircējs* var meklēt receptes;
   - *pircējs* var atrast preci veikalā.
4) "Ko var darīt pārdevējs ar sistēmu?" (skat. *15.c) attēlu*):
   - *pārdevējs* var ievadīt receptes;
   - *pārdevējs* var ievadīt preču izvietojumu veikalā.



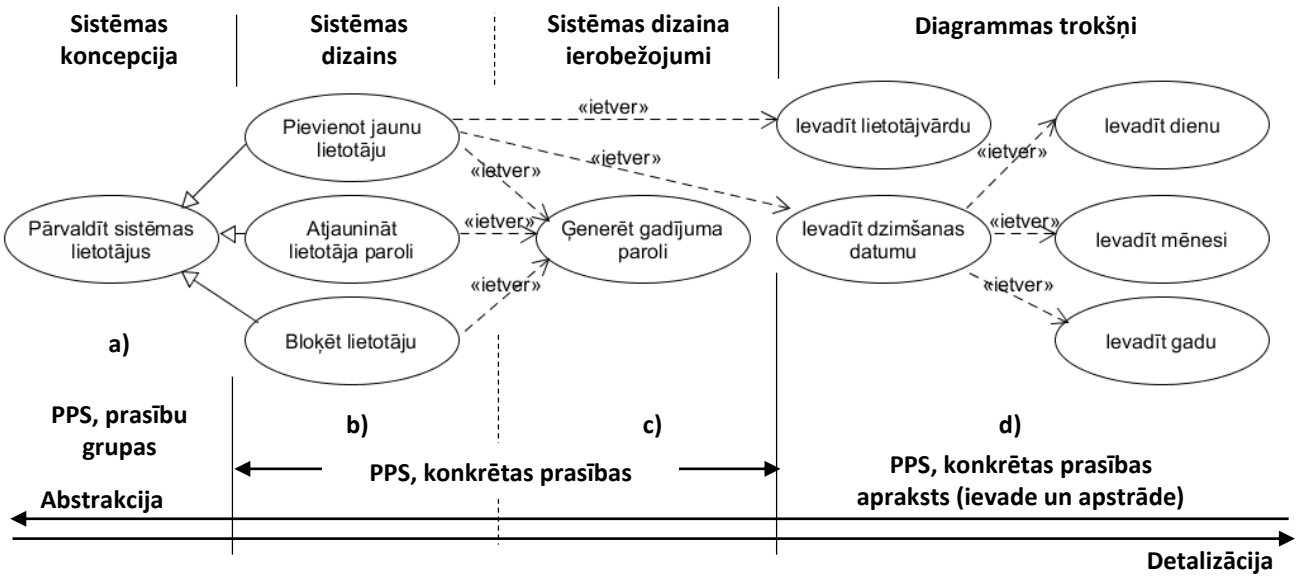
15. attēls. **Lietošanas gadījumu diagrammas zīmēšanas secība: a) apakšsistēmas; b) izpildītāji; c) lietošanas gadījumi**

Lietošanas gadījumi ir aktivitātes, lai izpildītu konkrētu uzdevumu, kas sastāv no konkrētu darbību secības (*Dennis, Wixom & Roth*, 2012; *Kettenis*, 2007) – sākuma (ievades), apstrādes procesa un beigām (izvades). Aktivitāte ir process, tāpēc lietošanas gadījuma nosaukumam jābūt struktūrai – **[darīt] [ko]** (*"doing something"*) (*Kettenis*, 2007), kas kodolīgi un skaidri formulē veicamo uzdevumu.

Visus lietošanas gadījumus pēc abstrakcijas īpašībām var sagrupēt trīs līmeņos (*Cockburn*, 2000):

- **kopsavilkuma līmenis** (*summary-level*) – lietošanas gadījums iekļauj sevī vairākus izpildes uzdevumus. Tas apraksta uzdevumu kontekstu, bet nenosaka konkrētus uzdevumus, ko var realizēt sistēmā, vairāk definējot biznesa uzdevumu vai lietotāju darbības un pienākumus. Piemēram, "Pārvaldīt sistēmas lietotājus" (skat. *16.a) attēlu*) neizsaka konkrēti, ko var darīt izpildītājs sistēmā ar lietotājiem, lai konstatētu, ka prasība ir izpildīta;
- **lietotāja mērķa līmenis** (*user goal level*) – galvenais intereses objekts, sastādot diagrammu. Šim līmenim pieder lietošanas gadījumi jeb aktivitātes, kuras veicot, lietotājs spēj izpildīt uzdevumu un ir apmierināts ar rezultātu. Piemēram, "Pievienot jaunu lietotāju", "Bloķēt lietotāju", "Atjaunināt lietotāja paroli" (skat. *16.b) attēlu*) utt. Katrai funkcijai ir konkrēts rezultāts, kuru vēlas sasniegt lietotājs, izpildot šo aktivitāti;
- **apakšfunkciju līmenis** (*subfunctions level*) – visas vienkāršās un standarta darbības, kas lietotājam jāveic, lai izpildītu uzdevumu, tomēr atsevišķi šīs funkcijas nedod vēlamo rezultātu. Piemēram, "Ģenerēt gadījuma paroli" (skat. *16.c) attēlu*) ir apakšfunkcija, ko var iekļaut lietotāja uzdevumā "Pievienot jaunu lietotāju" un "Atjaunināt lietotāja paroli", kad jauna parole, kas nav zināma administratoram, tiek aizsūtīta uz lietotāja e-pastu, un "Bloķēt lietotāju", kad parole netiek izsūtīta. Tomēr pati par sevi funkcija "Ģenerēt gadījuma paroli" nav lietderīga, jo ģenerējot paroli pat 100 reižu, lietotājs negūst nekādu labumu.

Papildinot A.Kokburna (*A.Cockburn*) klasifikāciju, var izdalīt vēl kategoriju **"Primitīvo funkciju līmenis"** (skat. *16.d) attēlu*), kam pieder tādas pašsaprotamas un vienkāršas funkcijas kā "Ievadīt lietotājvārdu", "Ievadīt datumu" utt., kuru zīmēšana pārslogo diagrammu un paslēpj sistēmas galveno funkcionalitāti. Šīs funkcijas veido lietošanas gadījuma scenārija apakšpunktu. Tomēr tādu aktivitāšu izdalīšana ir obligāta, ja darbība paredz nestandarta risinājumu, piemēram, funkcija "Ievadīt paroli" var būt organizēta kā šablona zīmēšana (*pattern password*), bet tādā gadījumā pareizāk ir pārsaukt funkciju, precizējot veicamo uzdevumu. Šajā gadījumā to var nodēvēt kā "Atbloķēt sistēmas funkciju", pārvietojot lietošanas gadījumu no kategorijas "Primitīvo funkciju līmenis" uz "Apakšfunkcijas līmeni".



16. attēls. **Lietošanas gadījumu līmeņi un attēlojums PPS: a) prasību apkopojums; b) sistēmas funkcijas; c) papildu un apakšfunkcijas; d) darbības**

Lietošanas gadījumi, kas attiecas uz lietotāja mērķa līmeni, ir svarīgākie, jo to prasības un funkcijas nosaka sistēmas inženiertehnisko risinājumu, no kā ir atkarīgs, vai sistēma tiks praktiski lietota, vai nē. Savukārt apakšfunkciju līmeni veido lietošanas gadījumi, ko definē izstrādātāji sistēmas plānošanas procesā, izstrādājot kopīgas funkcijas vai piešķirot sistēmai netiešus ierobežojumus, jo retos gadījumos pasūtītājs var zināt, piemēram, ka bloķēt lietotāju vajag ar gadījuma paroles ģenerēšanu (skat. *16.c) attēlu*). Tas ir

izstrādātāju piedāvājums. Tomēr, pasūtītājs var arī ietekmēt ierobežojumus, izsakot savus projekta ierobežojumus (PPS nodaļas "Vispārējie ierobežojumi", "Pieņēmumi un atkarības" un "Projekta ierobežojumi"). Primitīvo funkciju līmenis jau veido troksni diagrammā, jo tiek attēlotas pārāk smalkas un vienkāršas darbības, kuras paslēpj galvenās sistēmas funkcijas. Šīm darbībām jābūt attēlotām lietošanas gadījuma scenārijā, kas apraksta konkrētu darbību procesu aktivitātes ietvaros (PPS konkrētās prasības "ievade" un "apstrāde"). Kopsavilkuma līmenī vairākas prasības sagrupē kopā vienkāršākai pārskatīšanai, meklēšanai, lietotāju identificēšanai vai izstrādāšanas darba sadalīšanai (PPS nodaļas "Konkrētas prasības" apakšnodaļas).

Vispār eksistē dažādas pieejas, kā sagrupēt lietošanas gadījumus (*Kettenis*, 2007):

- pēc izpildītājiem;
- pēc kopsavilkuma lietošanas gadījuma;
- pēc sistēmas moduļiem;
- pēc sistēmas versijas;
- pēc izstrādes grupas;
- kombinējot metodes.

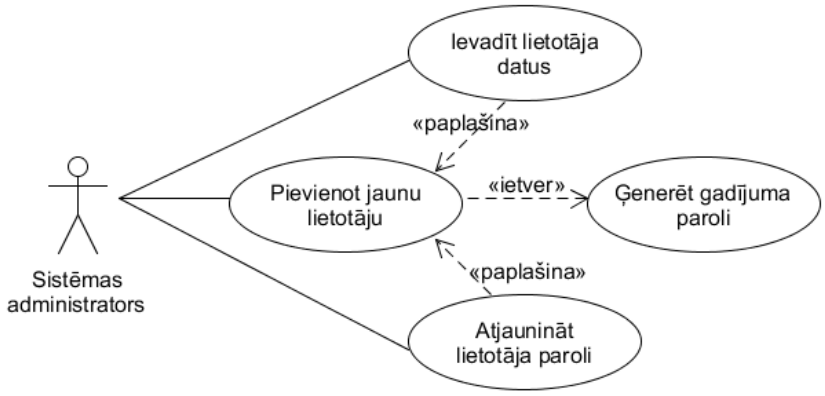
Kad lietošanas gadījums ir identificēts, tas tiek analizēts un aprakstīts. Dokumentācijai var pielietot šādu 4. tabulas struktūru (*Cockburn*, 2000; *Dennis, Wixom & Roth*, 2012; *Kettenis*, 2007; *PMI*, 2015).

4. tabula

Lietošanas gadījuma apraksts

Nosaukums	[Attēlo lietošanas gadījuma mērķi]
ID	[Unikāls numurs, lai īsi un viennozīmīgi veikt atsauces]
Apraksts	[Īss lietošanas gadījuma apraksts]
Izpildītāji	[Kuri izsauc lietošanas gadījumu]
Nepieciešamība	[Paskaidrojums, kāpēc šī funkcija ir nepieciešama]
Trigeris	[Notikums vai darbību secība, kas izsauc šo funkciju]
Priekšnosacījumi	[Nosacījumi, kam jābūt izpildītiem, lai šo funkciju varētu izsaukt]
Pamatscenārijs	[Apstrādes un darbību secība, kas notiek lietošanas gadījuma ietvaros]
Pēcnosacījumi	[Kam jābūt pēclietošanas rezultātā]
Alternatīvi scenāriji	[Alternatīvi scenārija izpildes varianti]
Izņēmuma scenārijs	[Scenāriji, kas notiek noteikta izņēmuma gadījumā]

Lietošanas gadījuma apraksta kodols ir scenārijs, kas apraksta lietotāja un sistēmas darbību secību. Scenārijam jābūt viegli lasāmam un skaidram, tāpēc labs stils paredz izvairīšanos no scenārijiem, kam soļu skaits ir lielāks par 9, katras darbības formulēšanu kā teicienu ar šādu struktūru **[kas] [dara] [ko]** (*Kettenis*, 2007). Lietošanas gadījuma "Pievienot jaunu lietotāju" scenārija piemērs dots *5. tabulā*; scenārijs tiek atbalstīts ar lietošanas gadījuma diagrammu (skat. *17. attēlu*).



17. attēls. **Lietošanas gadījums "Pievienot jaunu lietotāju" un ar to saistītās aktivitātes**

5. tabula

Scenārija piemērs

Lietošanas gadījums **"Pievienot jaunu lietotāju"**
Pamatscenārijs 1. *Sistēmas administrators* ievada lietotāja e-pastu. 2. Sistēma pārbauda, vai ievadītais e-pasts ir reģistrēts sistēmā, **ja** eksistē, tiek izsaukts **izņēmums *"Lietotājs jau reģistrēts sistēmā"***. 3. *Sistēmas administrators* ievada lietotājvārdu un lomu. 4. **Ja nepieciešams**, *sistēmas administrators* ievada lietotāja datus **(skat. *P1*).** /* Piemērā *P1* ir lietošanas gadījuma "Ievadīt lietotāja datus" unikālais identifikators */. 5. *Sistēmas administrators* apstiprina ievadīto informāciju. 6. Sistēmas ģenerē gadījuma paroli **(skat. *P2*).** /* Piemērā *P2* ir lietošanas gadījuma "Ģenerēt gadījuma paroli" unikālais identifikators */. 7. Sistēma pārbauda, vai e-pasts eksistē, ja neeksistē, tiek izsaukts **izņēmums *"E-pasts neeksistē"***. 8. Sistēma izsūta vēstuli ar lietotājvārdu un ģenerētu paroli uz norādīto e-pastu. 9. Sistēma paziņo, ka lietotājs ir veiksmīgi reģistrēts sistēmā.
Izņēmuma scenārijs **"Lietotājs jau reģistrēts sistēmā":** sistēma atver aktivitāti ***"Atjaunināt lietotāja paroli"*** **(skat. *P3*)**. **"E-pasts neeksistē":** sistēma paziņo *sistēmas administratoram*, ka ievadītā e-pasta adrese neeksistē.

Šajā piemērā parādās divas lietošanas gadījumu saites: "ietver" un "paplašina". Lai labāk pamanītu saišu formulējumus scenārija ietvaros, teksts ir izdalīts ar gaiši brūnu krāsu (skat. *5. tabulu*). Kāds princips ir šīm saitēm? Saite **"ietver" ir obligāta scenārija darbība, kas jāizpilda**. Tekstā to var identificēt pēc atsauces **"(skat. [*ID*])"**, protams, var rakstīt pilnu funkcijas nosaukumu *ID* vietā, tomēr, labojot funkcijas nosaukumu, katru reizi būs jālabo arī atsauces. *ID* parasti atbilst funkcijas secības numuram, kas paliek nemainīgs. Savukārt, **"paplašina" notiek izņēmuma gadījumā, ja izpildās kāds nosacījums**. Saiti "paplašina" raksturo teikums ar **saikli "JA"**. Praktiski to realizē vai kā lietotāja ievadi, *if-else* pārbaudi, vai kā *try-catch* konstrukciju. Lietošanas gadījumi, kas pieder grupai "ietver" vai "paplašina", tiek realizēti vai kā atsevišķas lietotnes saskarnes (*UI forms*), lietotāja kontroles (*user controls*) vai lietotnes funkcijas (*functions*).

Reālā dzīvē, kad cilvēks apraksta aktivitāti (lietošanas gadījumu) intervijas laikā, viņš apraksta sava darba gaitu kā viņš strādā un izpilda darba pienākumus. Ja sistēmas mērķis ir realizēt jaunu servisu, tad ieinteresētā puse, visticamāk, stāstīs, kādi cilvēki figurēs procesā un ko viņi darīs. Sistēmas darbības scenāriju sastāda

izstrādātājs, pamatojoties uz izstāstīto. *6. tabulā* ir attēlots, kā biznesa scenārijs tiek pārrakstīts sistēmas scenārijā.

6. tabula

Biznesa procesa realizācija sistēmā

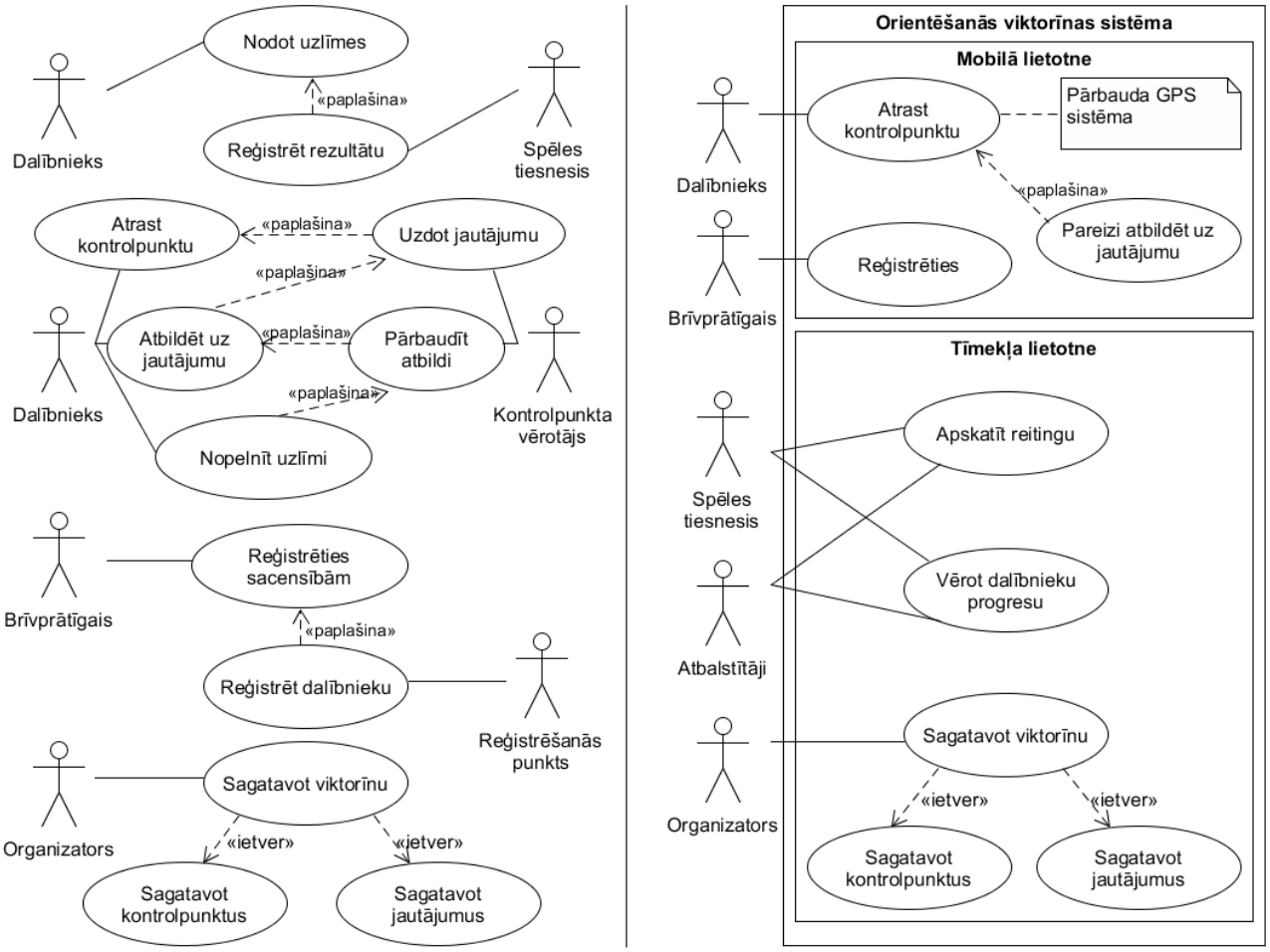
Biznesa scenārijs	Sistēmas scenārijs
Process: "Novērtēt tūrisma objektu"	**Lietošanas gadījums:** "Aizpildīt anketu"
Izpildītājs: eksperts	**Izpildītājs:** *eksperts*
	Priekšnosacījums: *analītiķis* izveidoja tūrisma objektu kategorijas aptaujas anketu
Pamatscenārijs 1. Eksperts saņem aptaujas anketu no vadītāja. 2. Ja nepieciešams, eksperts sazinās ar tūrisma objekta īpašnieku/ pārvaldnieku un sarunā apmeklējuma laiku. 3. Eksperts brauc apsekot vietu. 4. Eksperts piefiksē tūrisma objekta *GPS* koordinātes. 5. Eksperts novērtē tūrisma objektu. 6. Eksperts aizpilda anketu. 7. Eksperts izdrukā anketu. 8. Eksperts atdod anketu vadītājam.	**Pamatscenārijs** 1. *Eksperts* atver tūrisma objekta anketu. 2. Sistēma pārbauda, vai eksistē lokāla kopija. Ja ir, tiek izsaukts izņēmums "Eksistē lokāla kopija". 3. *Eksperts* ievada *GPS* koordinātes manuāli. Ja *eksperts* atrodas attiecīgajā vietā un ierīce satur *GPS* uztvērēju, var ievadīt automātiski. 4. *Eksperts* aizpilda anketas laukus. 5. *Eksperts* izvēlas "Iesniegt anketu", "Saglabāt melnrakstu" vai "Atcelt". 6. Ja, saglabājot anketas datus, sistēma konstatēja Interneta savienojuma pārtraukumu, tiek izsaukts izņēmums "Pazuda savienojums".
	Izņēmuma scenārijs **"Eksistē lokāla kopija":** sistēma pārbauda, vai ir reģistrēta anketa: • ja anketa ir iesniegta, sistēma dzēš lokālo datni; • ja anketa nav iesniegta, anketas laukos tiek ievadīti dati. **"Pazuda savienojums":** sistēma saglabā lokālo kopiju.

Sastādot lietošanas gadījumu diagrammu, jāatceras, ka lietošanas gadījumus var pielietot divu dažādu uzdevumu izpildē (*Kettenis*, 2007):

1) lai aprakstītu biznesa procesus;
2) lai aprakstītu funkcionālās prasības un dokumentētu sistēmas projektējumu.

Kad tiek aprakstīti biznesa procesi, izpildītāji ir amati, cilvēku grupas, tehnika un informācijas sistēmas iesaistītās biznesa procesos, bet lietošanas gadījumi – reāli procesi. Biznesa procesu lietošanas gadījumu

diagrammu parasti veido, lai iepazīstinātu ar biznesa modeli. Izstrādājot sistēmu, jāatceras, ka tā ir sociotehniskas vides sastāvdaļa, tāpēc jāņem vērā, kā notiks sadarbība ar sistēmu. Dažkārt jāmaina biznesa procesi, lai vienkāršotu sistēmas realizāciju un paaugstinātu sistēmas lietošanas efektivitāti. *18. attēlā* ir dotas biznesa procesu un sistēmas lietošanas gadījumu diagrammas.



a) Biznesa procesu lietošanas gadījumu diagramma **b) Sistēmas lietošanas gadījumu diagramma**

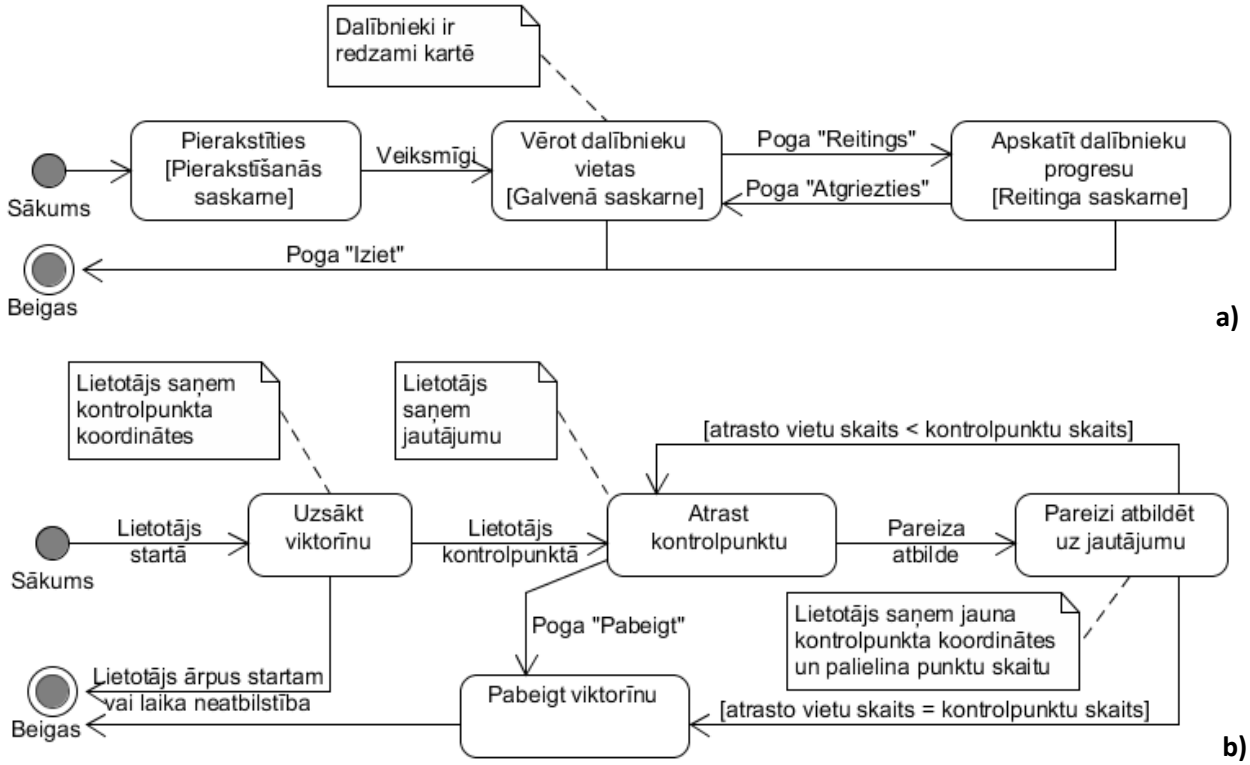
18. attēls. **Biznesa lietošanas gadījumu diagrammas attēlojums sistēmā**

Pirmajā gadījumā (skat. *18.a) attēlu*) ir aprakstīti biznesa procesi, kā tie notiek dotajā brīdī vai kā tos sistēmas pasūtītājs redz nākotnē. Otrajā diagrammā (skat. *18.b) attēlu*) ir parādīts izstrādātāju piedāvājums – var redzēt, ka pirmā un otrā varianta biznesa procesi atšķiras, jo sistēma pārņem un automatizē dažus uzdevumus, izslēdzot liekus elementus un vienkāršojot procesus. Piemēram, reģistrācija ir aizvietota ar pašreģistrāciju, viktorīnas atbilžu un kontrolpunktu pārbaudi veic sistēma, arī sakrāto punktu skaita aprēķināšana notiek automātiski.

Šī salīdzinājuma mērķis (skat. *18. attēlu*) bija paskaidrot, ka, noskaidrojot lietotāju biznesa procesus, izstrādātāja uzdevums ir ne tikai realizēt sistēmu, bet ar sistēmas palīdzību uzlabot biznesa procesus kopumā, kas paredz arī priekšlikumus par biznesa procesu izmaiņām, ievērojot jaunās sistēmas priekšrocības. Protams, tādi priekšlikumi sākumā jāsaskaņo ar pasūtītāju, nostiprinot tos programmatūras prasību specifikācijā.

Lietotāja prasības ir saistītas ar izpildītāju, tāpēc tās paredz ievadizvades operācijas. Ja sistēmu lieto cilvēks, ievadizvade notiek ar grafiskās saskarnes vai vadības paneļa formu starpniecību; ja sistēmu izmanto cita

sistēma, notiek datu pārraide. Ja sistēma ir saistīta ar apjomīgu datu pārvaldi (tai ir daudz grafisko saskarņu), ir jāattēlo pāreju starp saskarnēm. Līdzīga situācija rodas arī gadījumā, ja sistēmai ir sazarota un sarežģīta loģika. Te var pielietot stāvokļu diagrammas. *19. attēlā* ir dots piemērs ar vairākām lietotāju saskarnēm (skat. *19.a) attēlu*) un piemērs ar sazarotu loģiku (skat. *19.b) attēlu*). Abos piemēros jāpievērš uzmanība, ka lietošanas gadījumi un sistēmas funkcijas atbilst stāvokļu nosaukumiem.



19. attēls. **Stāvokļu diagramma: a) attēlo saites starp saskarnēm; b) attēlo saites starp funkcijām**

Lietošanas gadījumu diagrammas ir efektīvs veids, kā noteikt un kodolīgi aprakstīt biznesa sistēmas prasības.

Sastādot lietošanas gadījuma diagrammu, jāseko, kas tieši tiek aprakstīts – biznesa procesi vai sistēmas funkcijas. Biznesa procesus parasti apraksta intervējamās personas, definējot lietotāja prasības, sistēmas funkcijas rodas prasību analīzes procesā, kad izstrādātāji plāno sistēmu.

Sistēmas lietošanas gadījumu diagrammas elementus un to scenāriju aprakstus var tieši izmantot programmatūras prasību specifikācijā.

Ja sistēmas prasības programmatūras prasību specifikācijā nav skaidri definētas, realizējot sistēmu, izstrādātājam rodas barjera “Ko izstrādāt?”, ko var ietekmēt arī prasmju, zināšanu un iemaņu trūkums.

Nobeigums

Grāmatas saturs primāri ir saistīts ar programmatūras izstrādi. Tās mērķis ir paskaidrot programmatūras izstrādes iesācējiem, ka pirms sistēmas realizācijas jāsaprot, kāda sistēma tiks realizēta, noskaidrojot visas sistēmas prasības un funkcijas. Bez skaidra sistēmas redzējuma, kas parasti tiek dokumentēts kā programmatūras prasību specifikācija, izstrādātājam var rasties barjera – "Ko realizēt?".

Lai palīdzētu iesācējam vienkāršot un strukturēt prasību vākšanas un analīzes procesu, grāmatā:

- dots teorētisks ieskats par sistēmas dzīves ciklu;
- aprakstītas četras programmatūras izstrādes metodoloģijas;
- apskatīts prasību vākšanas, analīzes un dokumentēšanas process;
- detalizēti apspriestas lietošanas gadījumu diagrammas.

Grāmatā ir dots "Ūdenskrituma modeļa", "Spirāles modeļa", "Inkrementālā modeļa" un *Microsoft Solution Framework* metodoloģijas apskats. Iesācējam jāuztver šīs izstrādes metodoloģijas kā rekomendācijas:

- izstrādes process nedrīkst būt haotisks, un katra izstrādes posma beigās jābūt konkrētam rezultātam;
- izstrādājot sistēmu, jābūt gatavam pārmaiņām un modifikācijām, pastāvīgi saskaņojot izstrādājamo sistēmu ar pasūtītāju, lai realizētu vēlamo sistēmu un nenovirzītos no tās tik stipri, ka nekāda modifikācija jau nespēs izlabot situāciju;
- uzreiz realizēt un apzināt visas prasības ir grūti, tāpēc lietderīgi ir sagrupēt prasības un realizēt sistēmu pa daļām;
- prasību grupēšana pa versijām ir efektīvs veids, kā organizēt programmas izstrādes procesu pa daļām, jo versija ir pabeigts un pielietojams produkts.

Lietošanas gadījumu diagrammas tika detalizēti apskatītas grāmatā, jo tās ir efektīvs veids, kā analizēt lietotāja prasības, lai vēlāk vienkārši tās izmantotu programmatūras prasību specifikācijā.

Literatūra

Bell, D. (2005). *Software Engineering for Students: A Programming Approach*. Addison-Wesley Longman Ltd.

Chappell, D. (2008). *What Is Application Lifecycle Management?* Ielādēts no

http://www.davidchappell.com/WhatIsALM--Chappell.pdf

Cockburn, A. (2000). *Writing Effective Use Cases. Addison-Wesley Professional*.

Dennis, A., Wixom, B., & Roth, R. (2012). *Systems Analysis and Design. Wiley*.

Oxford. English Oxford Living Dictionaries. *Definition of actor in English.* Pieejams

https://en.oxforddictionaries.com/definition/actor

Getchell, S., Hargrave, L., Haynes, P., Lubrecht, M., Kazmi, P., Oikawa, R., . . . Short, M. (2002). *Microsoft Solutions Framework White Paper: MSF Process Model v. 3.1.* Ielādēts no

https://www.microsoft.com/en-us/download/details.aspx?id=13870

Gupta, P. (2005). *Structured System Analysis and Design.* Laxmi Publications.

Kettenis, J. (2007). *Getting Started With Use Case.* Ielādēts no

http://www.oracle.com/technetwork/testcontent/gettingstartedwithusecasemodeling-133857.pdf

OMG. (2011). *OMG Unified Modeling LanguageTM (OMG UML), Superstructure.* Ielādēts no

https://www.omg.org/spec/UML/2.4.1/Superstructure/PDF

PMI. (2015). *Business Analysis for Practitioners: A Practice Guide.* Project Management Institute.

Roger S., P., & Maxim, B. (2014). *Software Engineering: A Practitioner's Approach.* McGraw-Hill Education.

Wiegers, K., & Beatty, J. (2013). *Software Requirements. Microsoft Press.*

Pielikumi

“Kalkulators”

Programmatūras prasību specifikācija

(Versija 1.0.0.)

2019

Saturs

1. Ievads

1.1. Nolūks: šī programmatūras prasību specifikācija ir izstrādāta studentiem, lai viņi iegūtu priekštatu par programmatūras prasību specifikācijas struktūru un iemācītos izstrādāt datorprogrammas atbilstoši specifikācijai.

1.2. Darbības sfēra: programma “Kalkulators” ir paredzēta vienkāršu aritmētisko operāciju (saskaitīšanas, atņemšanas, reizināšanas un dalīšanas) veikšanai un starprezultāta iegaumēšanai.

1.3. Definīcijas, akronīmi un saīsinājumi:

- *ID* – unikālais identifikators;
- lietošanas gadījuma diagramma – diagramma, kas attēlo sistēmas lietotājus un viņu mijiedarbības ar sistēmu;
- PPS – programmatūras prasību specifikācija;
- OS – operētājsistēma.

1.4. Saistība ar citiem dokumentiem:

“LVS 68:1996 INFORMĀCIJAS TEHNOLOĢIJA. PROGRAMMINŽENIERIJA. PROGRAMMATŪRAS PRASĪBU SPECIFIKĀCIJAS CEĻVEDIS”.

1.5. Pārskats:

***2. nodaļā* “Vispārējais apraksts”** ir aprakstīta:

1) *produkta perspektīva* – produkta apraksts nākotnes rakursā un saistībā ar citiem produktiem vai projektiem;

2) *lietotāja raksturiezīmes* – vispārējas produkta lietotāja raksturiezīmes, kuras ietekmē specifiskās prasības;

3) *produkta funkcijas* – kopsavilkums par funkcijām, kuras izpilda programmatūra;

4) *kopsavilkums* – lietošanas gadījumu diagramma, kas kodolīgi attēlo produkta funkcijas un lietotājus.

***3. nodaļā* “Vispārējie ierobežojumi”** ir aprakstīti projekta ierobežojumu cēloņi.

***4. nodaļā* “Konkrētās prasības”** ir dots detalizēts funkciju apraksts.

***5. nodaļā* “Ārējās saskarnes prasības”** ir aprakstītas lietotāja saskarnes prasības un attēlota saskarnes struktūra.

***6. nodaļā* “Projekta ierobežojumi”** ir aprakstīti konkrēti produkta realizācijas ierobežojumi.

2. Vispārējais apraksts

2.1. Produkta perspektīva: programma “Kalkulators” ir neatkarīga un pašpietiekama programma. Produktam nav paredzētas vairākas sastāvdaļas. Produktam ir viena ārējā saskarne, kuru var apskatīt *5. nodaļā*. Produkts satur vienkāršas aritmētiskās operācijas funkcijas, tomēr nākotnē produktam ir paredzētas kāpināšanas aprēķināšanas un kvadrātsaknes izvilkšanas funkcijas.

2.2. Lietotāja raksturiezīmes: programma ir paredzēta vienam lietotājam, kurš saprot latviešu valodu. Lietotāja vecuma vai izglītības ierobežojumi nav paredzēti.

2.3. Produkta funkcijas:

Programmai “Kalkulators” ir astoņas funkcijas:

(*K1*) “Ievadīt skaitli” – funkcija ļauj ievadīt skaitļus, ar kuriem tiks veikta aritmētiskā operācija.

(*K2*) “Saskaitīt vērtības” – funkcija ļauj saskaitīt kopā divas lietotāja ievadītās vērtības.

(*K3*) “Atņemt vērtību” – funkcija ļauj atņemt no pirmās ievadītās vērtības nākamo.

(*K4*) “Reizināt vērtības” – funkcijas ļauj sareizināt pirmo ievadīto vērtību ar otro.

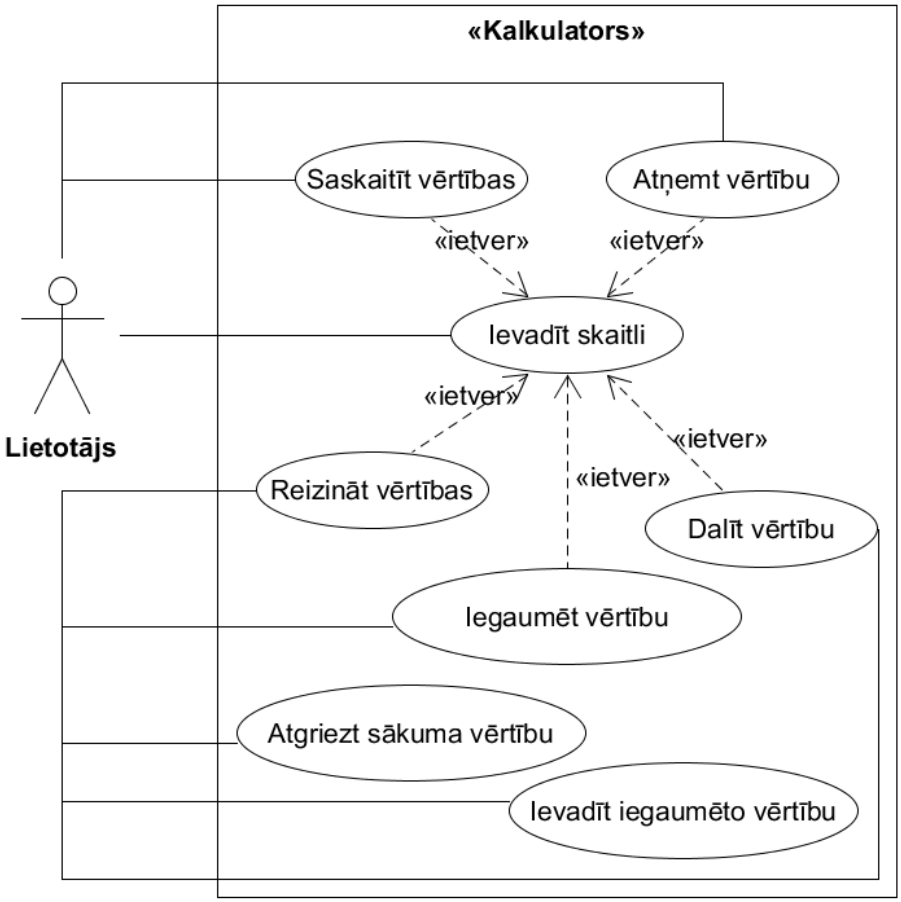
(*K5*) “Dalīt vērtību” – funkcija ļauj sadalīt pirmo ievadīto vērtību ar otro, paredzot pārbaudi “dalīt ar nulli”.

(*K6*) “Iegaumēt vērtību” – funkcija ļauj iegaumēt vienu vērtību, kuru vēlāk varēs izsaukt, neievadot to atkārtoti.

(*K7*) “Atgriezt sākuma vērtību” – funkcija iestata vērtību *nulle* ievadlaukā.

(*K8*) “Ievadīt iegaumēto vērtību” – funkcija automātiski ievada iepriekš iegaumēto vērtību.

2.4. Kopsavilkums: funkciju pārskatāmībai ir izveidota programmas lietošanas gadījuma diagramma (*1. attēls*).



1. attēls. **Programmas “Kalkulators” lietošanas gadījuma diagramma**

3. Vispārējie ierobežojumi

Projekta ierobežojumi: ievērojot, ka šī PPS ir izstrādāta mācību nolūkos studentiem, projekta realizācijas tehnoloģiju izvēle ir ierobežota mācību kursa ietvaros.

4. Konkrētās prasības

4.1. Funkcija “Ievadīt skaitli”

ID: K1

Ievads: funkcija ļauj ievadīt skaitļus, ar kuriem tiks veikta aritmētiskā operācija.

Ievade: lietotājs ievada skaitli, izmantojot pogas ‘Viens’, ‘Divi’, ‘Trīs’, ‘Četri’, ‘Pieci’, ‘Seši’, ‘Septiņi’, ‘Astoņi’, ‘Deviņi’, ‘Nulle’, (skat. *5. nodaļu*). Poga ‘Komats’ paredzēta, lai ievadītu daļskaitli; ‘Plus/mīnus zīme’ – lai mainītu pozitīvo skaitli uz negatīvu, un otrādi.

Apstrāde:

Ja laukā ‘Skaitlis’ vērtība ir nulle, pogas ‘Viens’, ‘Divi’, ‘Trīs’, ‘Četri’, ‘Pieci’, ‘Seši’, ‘Septiņi’, ‘Astoņi’, ‘Deviņi’, ‘Nulle’ aizvieto nulli ar atbilstošu ciparu. Piemēram, ja skaitlis ir nulle un lietotājs nospieda pogu ‘Trīs’, laukā jāparādās vērtībai ‘3’.

Ja lauka ‘Skaitlis’ vērtība nav nulle, pogas ‘Viens’, ‘Divi’, ‘Trīs’, ‘Četri’, ‘Pieci’, ‘Seši’, ‘Septiņi’, ‘Astoņi’, ‘Deviņi’, ‘Nulle’ esošajai vērtībai pievienos pa labi izvēlēto ciparu. Piemēram, ja skaitlis ir 3 un lietotājs nospiež pogu ‘Seši’, rezultātā jāparādās skaitlim ‘36’.

Ja lauka ‘Skaitlis’ simbolu daudzums ir septiņi, tad papildsimbolu pievienot nevar. Piemēram, ja ir ievadīts skaitlis –123,56 un lietotājs nospiež pogu ‘Nulle’, skaitlis paliks nemainīgs – –123,56.

Ja lauks ‘Skaitlis’ satur komatu, vēl viens komats netiek pievienots.

Ja lauks ‘Skaitlis’ satur vērtību nulle, plus vai mīnus zīme netiek pievienota.

Ja lauks ‘Skaitlis’ satur pozitīvu vērtību, lielāku par nulli, pa kreisi no vērtības tiek pievienots mīnuss. Piemēram, ja skaitlis ir 36, tad jābūt –36. Ja lauks ‘Skaitlis’ satur negatīvu vērtību, tad mīnus zīme tiek dzēsta. Piemēram, ja skaitlis ir –36, tad jāparādās skaitlim ‘36’.

Izvade: ievadītā vērtība tiek attēlota laukā ‘Skaitlis’.

4.2. Funkcija “Saskaitīt vērtības”

ID: K2

Ievads: funkcija ļauj saskaitīt kopā divas lietotāja ievadītās vērtības.

Ievade:

1) lietotājs ievada pirmo skaitli, pielietojot funkciju “Ievadīt skaitli” (*K1*);
2) nospiež pogu ‘Summa’ (skat. *5. nodaļu*);
3) ievada otro skaitli, pielietojot funkciju “Ievadīt skaitli” (*K1*);
4) nospiež pogu ‘Vienāds’ (skat. *5. nodaļu*).

Apstrāde:

1) teksta rinda ar pirmo skaitli tiek konvertēta par daļskaitli;
2) teksta rinda ar otro skaitli tiek konvertēta par daļskaitli;
3) pirmais daļskaitlis tiek saskaitīts ar otro daļskaitli.

Izvade: laukā ‘Skaitlis’ tiek attēlota daļskaitļu summa.

4.3. Funkcija “Atņemt vērtību”

ID: K3

Ievads: funkcija ļauj atņemt no pirmās ievadītās vērtības nākamo.

Ievade:

1)lietotājs ievada pirmo skaitli, pielietojot funkciju “Ievadīt skaitli” (*K1*);
2) nospiež pogu ‘Mīnuss’ (skat. *5. nodaļu*);
3) ievada otru skaitli, pielietojot funkciju “Ievadīt skaitli” (*K1*);
4) nospiež pogu ‘Vienāds’ (skat. *5. nodaļu*).

Apstrāde:

1) teksta rinda ar pirmo skaitli tiek konvertēta par daļskaitli;
2) teksta rinda ar otro skaitli tiek konvertēta par daļskaitli;
3) otrais daļskaitlis tiek atņemts no otrā daļskaitļa.

Izvade: laukā ‘Skaitlis’ tiek attēlota daļskaitļu starpība.

4.4. Funkcija “Reizināt vērtības”

ID: K4

Ievads: funkcijas ļauj sareizināt pirmo ievadīto vērtību ar otro.

Ievade:

1)Lietotājs ievada pirmo skaitli, pielietojot funkciju “Ievadīt skaitli” (*K1*);
2) nospiež pogu ‘Reizināt’ (skat. *5. nodaļu*);
3) ievada otro skaitli, pielietojot funkciju “Ievadīt skaitli” (*K1*);
4) nospiež pogu ‘Vienāds’ (skat. *5. nodaļā*).

Apstrāde:

1) teksta rinda ar pirmo skaitli tiek konvertēta par daļskaitli;
2) teksta rinda ar otro skaitli tiek konvertēta par daļskaitli;
3) pirmais daļskaitlis tiek reizināts ar pirmo daļskaitli.

Izvade: laukā ‘Skaitlis’ tiek attēlota daļskaitļu reizinājums.

4.5. Funkcija “Dalīt vērtību”

ID: K5

Ievads: funkcija ļauj sadalīt pirmo ievadīto vērtību ar otro, paredzot pārbaudi “dalīt ar nulli”.

Ievade:

1)lietotājs ievada pirmo skaitli, pielietojot funkciju “Ievadīt skaitli” (*K1*);
2) nospiež pogu “Dalīt” (skat. *5. nodaļu*);
3) ievada otru skaitli, pielietojot funkciju “Ievadīt skaitli” (*K1*);
4) nospiež pogu “Vienāds” (skat. *5. nodaļu*).

Apstrāde:

1) teksta rinda ar pirmo skaitli tiek konvertēta par daļskaitli;
2) teksta rinda ar otru skaitli tiek konvertēta par daļskaitli;
3) ja otrais daļskaitlis nav vienāds ar nulli, pirmais daļskaitlis tiek dalīts ar otro daļskaitli.

Izvade:

Laukā “Skaitlis” tiek attēlots daļskaitļu dalījums.

Ja otrais skaitlis bija nulle, laukā ‘Skaitlis’ tiek attēlota nulle un parādās ziņojumu logs ar tekstu “Kļūda: dalīt ar nulli nedrīkst!”.

Ja pirmais skaitlis bija nulle, bet otrais – cita vērtība, laukā ‘Skaitlis’ tiek attēlota nulle un parādās ziņojumu logs ar tekstu “Bezgalība!”.

4.6. Funkcija “Iegaumēt vērtību”

ID: K6

Ievads: funkcija ļauj iegaumēt vienu vērtību, kuru vēlāk varēs izsaukt, neievadot to atkārtoti.

Ievade:

1)lietotājs ievada skaitli, pielietojot funkciju “Ievadīt skaitli” (*K1*);
2) nospiež pogu ‘Saglabāt’ (skat. *5. nodaļu*).

Apstrāde:

1) teksta rinda ar skaitli tiek konvertēta uz daļskaitli;
2) daļskaitlis tiek iegaumēts sistēmā.

Izvade: laukā ‘Skaitlis’ tiek attēlota nulle. Laukā ‘Saglabātā vērtība’ tiek attēlota jaunā iegaumētā vērtība.

4.7. Funkcija “Atgriezt sākuma vērtību”

ID: K7

Ievads: funkcija iestata vērtību ‘nulle’ ievadlaukā.

Ievade: lietotājs nospiež pogu ‘Nodzēst’ (skat. *5. nodaļu*).

Apstrāde: tiek mainīta lauka ‘Skaitlis’ vērtība.

Izvade: laukā ‘Skaitlis’ tiek attēlota nulle.

4.8. Funkcija “Ievadīt iegaumēto vērtību”

ID: K8

Ievads: funkcija automātiski ievada iepriekš iegaumēto vērtību.

Ievade: lietotājs nospiež pogu ‘Ievadīt’ (skat. *5. nodaļu*).

Apstrāde:

1) iegaumētais daļskaitlis tiek konvertēts uz teksta rindu;
2) teksta rindas garums tiek samazināts līdz septiņiem simboliem.

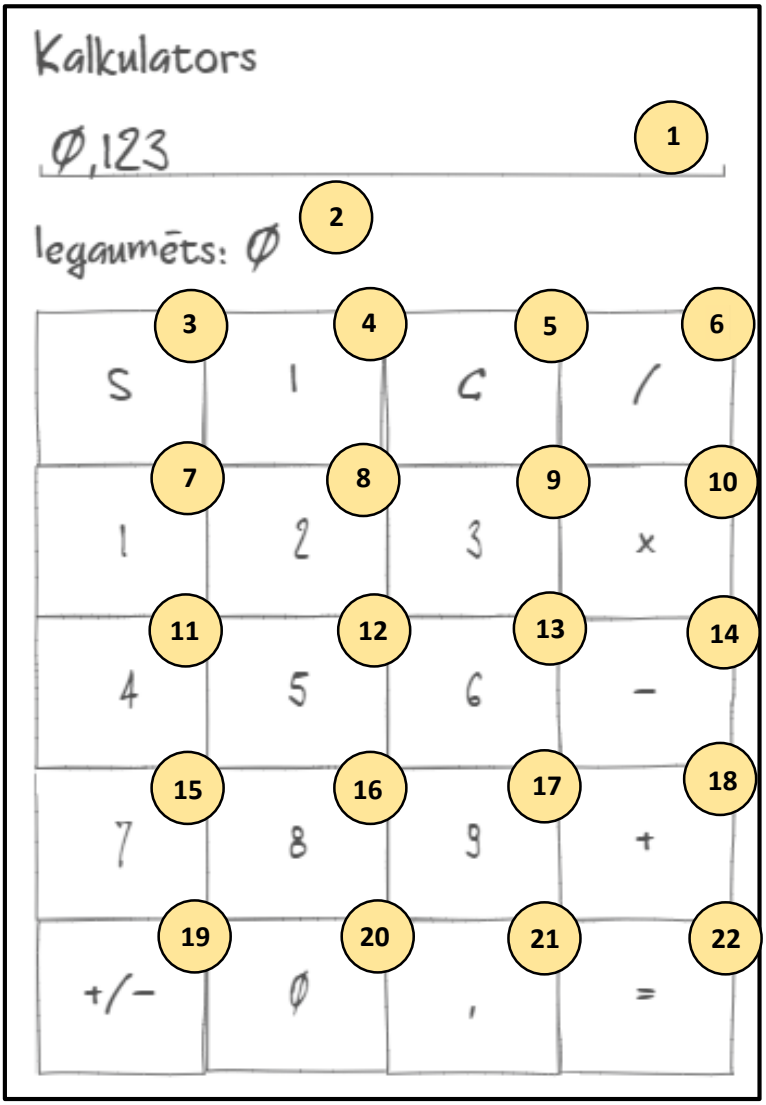
Izvade: laukā ‘Skaitlis’ tiek attēlota iegaumētā vērtība.

5. Ārējās saskarnes prasības

5.1. Lietotāja saskarne: produkts paredz vienu lietotāja saskarni.

5.2. Ekrāna formāti: minimālais ekrāna izmērs ir 10” ar minimālo izšķirtspēju – 800x600 punkti.

5.3. Saskarnes struktūra: skat. *2. attēlu*.



2. attēls. **Programmas “Kalkulators” saskarne**

5.4. Saskarnes elementi:

1. Lauks “Skaitlis”;
2. Lauks “Saglabātā vērtība”;
3. Poga “Saglabāt”;
4. Poga “Ievadīt”;
5. Poga “Nodzēst”;
6. Poga “Dalīt”;
7. Poga “Viens”;
8. Poga “Divi”;
9. Poga “Trīs”;
10. Poga “Reizināt”;
11. Poga “Četri”;
12. Poga “Pieci”
13. Poga “Seši”;
14. Poga “Mīnuss”;
15. Poga “Septiņi”;
16. Poga “Astoņi”;
17. Poga “Deviņi”;
18. Poga “Summa”;
19. Poga “Plus/ mīnus zīme”;
20. Poga “Nulle”;
21. Poga “Komats”;
22. Poga “Vienāds”.

6. Projekta ierobežojumi

6.1. Aparatūras ierobežojumi: programmai jāstrādā korekti uz “*Windows 10”* OS un uz jaunākām versijām.

6.2. Citas prasības: veicot tehnoloģiju izvēli, jāievēro mācību kursā ietvertās programmatūras izstrādes tehnoloģijas, lai realizētu projektu.

7. Prasību tabula

ID	**Prasības**	**Lpp.**
K1	Ievadīt skaitli	6.
K2	Saskaitīt vērtības	7.
K3	Atņemt vērtību	7.
K4	Reizināt vērtības	7.
K5	Dalīt vērtību	8.
K6	Iegaumēt vērtību	8.
K7	Atgriezt sākuma vērtību	9.
K8	Ievadīt iegaumēto vērtību	9.

Tīmekļa spēle
“Krustiņi un nullītes”

Programmatūras prasību specifikācija

(Versija 1.0.0.)

2019

Saturs

1. Ievads

1.1. Nolūks: šī programmatūras prasību specifikācija ir izstrādāta Rēzeknes Tehnoloģiju akadēmijas studentiem, lai viņi iegūtu priekštatu par programmatūras prasību specifikācijas struktūru un iemācītos izstrādāt datorprogrammas atbilstoši tai.

1.2. Darbības sfēra: tīmekļa spēle “Krustiņi un nullītes” ir izklaides programma vienam dalībniekam, kas ir pieejama internetā. Tā ir balstīta uz spēli “Krustiņi un nullītes”, tikai satur specifiskas modifikācijas. Spēles laikā lietotājs cīnās ar datoru par to, kurš pirmais izveidos rindu no 3 vienādiem simboliem. Spēlē uzvar tas, kurš pirmais to izdara 2 reizes pēc kārtas.

1.3. Definīcijas, akronīmi un saīsinājumi:

- *ID* – unikāls identifikators;
- lietošanas gadījuma diagramma – diagramma, kas attēlo sistēmas lietotājus un funkcijas;
- PPS – programmatūras prasību specifikācija;
- vietrādis *URL* – standartizēta resursa (dokumenta vai attēla) adrese internetā;
- stāvokļu diagramma – diagramma, kas attēlo pārejas starp funkcijām.

1.4. Saistība ar citiem dokumentiem

“LVS 68:1996 INFORMĀCIJAS TEHNOLOĢIJA. PROGRAMMINŽENIERIJA. PROGRAMMATŪRAS PRASĪBU SPECIFIKĀCIJAS CEĻVEDIS”.

1.5. Pārskats

***2. nodaļā* “Vispārējais apraksts”** ir aprakstīta:

1) *produkta perspektīva* – produkta apraksts nākotnes skatījumā un saistībā ar citiem produktiem vai projektiem;
2) *lietotāja raksturiezīmes* – produkta lietotāja vispārīgas raksturiezīmes, kas ietekmē specifiskās prasības;
3) *produkta funkcijas* – kopsavilkums par funkcijām, ko veic programmatūra;
4) *kopsavilkums* – lietošanas gadījumu diagramma, kas attēlo produkta funkcijas un lietotājus.

***3. nodaļā* “Vispārējie ierobežojumi”** ir aprakstīti projekta ierobežojumu cēloņi.

***4. nodaļā* “Pieņēmumi un atkarības”** ir norādīti pieņēmumi par lietotāja darba ierīces aprīkojumu.

***5. nodaļā* “Konkrētās prasības”** ir detalizēti aprakstītas funkcijas.

***6. nodaļā* “Ārējās saskarnes prasības”** ir aprakstītas lietotāja saskarnes prasības un attēlota saskarņu struktūra.

***7. nodaļā* “Projekta ierobežojumi”** ir aprakstīti konkrēti produkta realizācijas ierobežojumi.

2. Vispārējais apraksts

2.1. Produkta perspektīva: tīmekļa spēle “Krustiņi un nullītes” ir produkts, kas ir atkarīgs no pārlūkprogrammas. Produktam nav paredzētas vairākas sastāvdaļas. Produktam ir divas ārējās saskarnes, kuras var apskatīt *6. nodaļā* – viena paredzēta spēles sākšanai, otra veido spēles laukumu. Perspektīvā divi spēlētāji varēs sacensties savā starpā un tiks krāti statistikas dati par spēles rezultātiem.

2.2. Lietotāja raksturiezīmes: produkts ir paredzēts mobilo telefonu, planšetdatoru, portatīvo un stacionāro datoru lietotājiem. Spēle tiek izstrādāta Latvijas tirgum, tāpēc lietotājam jāsaprot latviešu valoda. Vecuma vai izglītības ierobežojumi nav paredzēti, tāpēc tiek pieņemts, ka lietotājam nav priekšzināšanu par spēles mehānismu.

2.3. Produkta funkcijas

Tīmekļa spēlei “Krustiņi un nullītes” ir vienpadsmit funkcijas:

(*K1*) “Izlasīt instrukciju” – funkcija ļauj lietotājam izlasīt spēles noteikumus.

(*K2*) “Sākt spēli” – funkcija ļauj lietotājam sākt jaunu spēli.

(*K3*) “Izvēlēties spēles laukuma izmēru” – funkcija ļauj izvēlēties spēles laukuma izmēru.

(*K4*) “Sākt spēļu ciklu” – funkcija ļauj uzsākt 3 spēļu ciklu.

(*K5*) “Veikt spēles gājienu” – funkcija ļauj lietotājam aizpildīt spēles laukuma rūtiņu ar krustiņu vai nullīti.

(*K6*) “Pārbaudīt, vai ir līnija” – funkcija pārbauda, vai ir izveidota līnija no 3 simboliem.

(*K7*) “Uzvarēt spēļu ciklā” – funkcija tiek izsaukta, kad lietotājs ir izveidojis rindu no krustiņiem vai nullītēm. Tiek atjaunots spēles rezultāts, un notiek pārbaude, vai lietotājs uzvarēja.

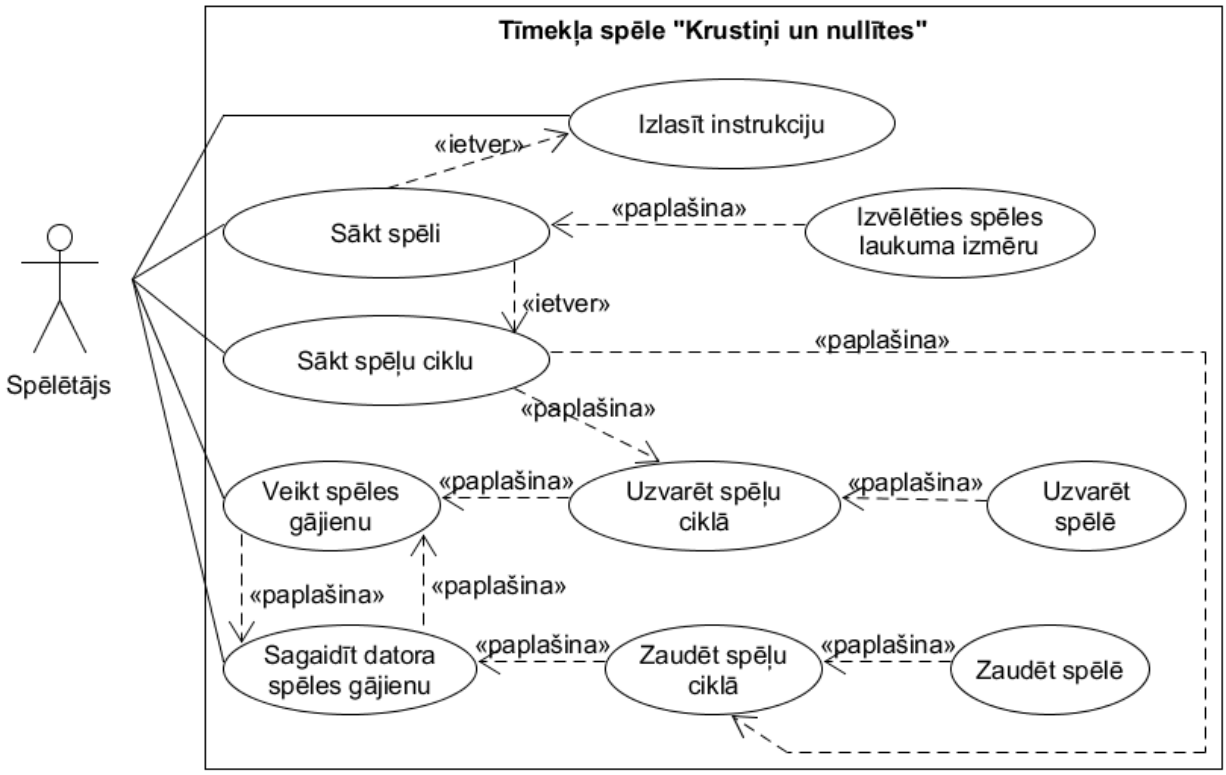
(*K8*) “Uzvarēt spēlē” – funkcija tiek izsaukta, ja lietotājs uzvar divas reizes pēc kārtas. Programma apsveic lietotāju ar uzvaru.

(*K9*) “Sagaidīt datora spēles gājienu” – funkcijas uzdevums attēlot un imitēt pretinieka gājienu.

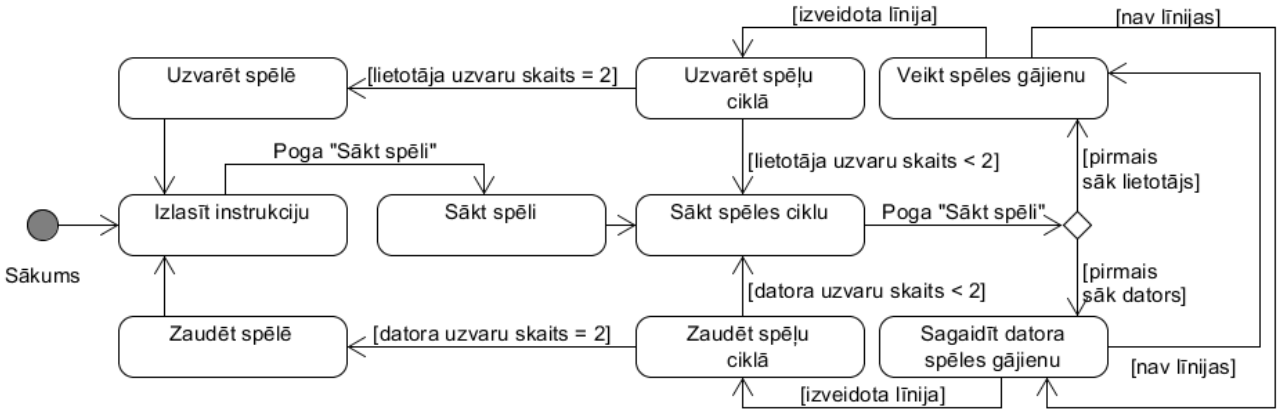
(*K10*) “Zaudēt spēļu ciklā” – funkcija tiek izsaukta, kad datora gājienā tiek izveidota līnija no 3 simboliem.

(*K11*) “Zaudēt spēlē” – funkcija tiek izsaukta, ja dators uzvar divas reizes pēc kārtas. Programma paziņo, ka uzvarēja dators.

2.4. Kopsavilkums: funkciju pārskatāmībai ir izveidota programmas lietošanas gadījuma diagramma (*1. attēls*). Stāvokļu diagrammā ir attēlotas pārejas starp funkcijām (*2. attēls*).



1. attēls. **Tīmekļa spēles “Krustiņi un nullītes” lietošanas gadījuma diagramma**



2. attēls. **Tīmekļa spēles “Krustiņi un nullītes” stāvokļu diagramma**

3. Vispārējie ierobežojumi

Projekta ierobežojumi

Lietotne ir paredzēta lietošanai tīmekļa vidē: mobilajos telefonos, planšetdatoros, portatīvajos un stacionārajos datoros.

Tiek pieņemts, ka primāri nepieciešams atbalstīt *Chrome* pārlūkprogrammas lietotājus.

Ievērojot, ka šī PPS ir izstrādāta mācību nolūkos Rēzeknes Tehnoloģiju akadēmijas studentiem, projekta realizācijas tehnoloģiju izvēle ir ierobežota studiju kursa ietvaros.

4. Pieņēmumi un atkarības

Tiek pieņemts, ka lietotājam ir pārlūkprogramma un interneta savienojums.

5. Konkrētās prasības

5.1. Funkcija “Izlasīt instrukciju”

ID: K1

Ievads: funkcija ļauj lietotājam izlasīt spēles noteikumus.

Ievade: lietotājs atver pārlūkprogrammu un ievada tīmekļa spēles vietrādi *URL*.

Apstrāde: lietotājam tiek attēlota spēles sākšanas saskarne (skat. 6.2. nodaļu)

Izvade: “Spēles sākšanas saskarne” (skat. 6.2. nodaļu).

5.2. Funkcija “Sākt spēli”

ID: K2

Ievads: funkcija ļauj lietotājam sākt jaunu spēli.

Ievade:

Saskarne: “Spēles sākšanas saskarne” (skat. 6.2. nodaļu).

Lietotāja darbības:

1) lietotājs apmeklē spēles tīmekļa vietni un izlasa spēles noteikumus (*K1*);
2) ja vēlas lielāku spēles laukumu, maina tā izmēru (*K3*);
3) nospiež pogu “Sākt spēli” (skat. 6.2. nodaļu).

Apstrāde:

1) tiek izveidots spēles laukums ar izmēru *N x N*, kur *N* – izvēlētais spēles laukuma izmērs (pēc noklusējuma 3x3);
2) tiek iestatīts lietotāju un datora uzvaru skaits, kas vienāds ar nulli;
3) nejauši tiek izvēlēts spēlētāja gājiena apzīmējums “krustiņš” vai “nullīte”, bet datoram tiek piešķirts otrais gājiena apzīmējums;
4) nejauši tiek izvēlēts, kurš veic pirmo gājienu: lietotājs vai dators;
5) tiek uzsākts pirmais spēļu cikls (funkcija *K4*).

Izvade: “Spēles saskarne” (skat. 6.3. nodaļu). Spēles progresa indikators ir 0 : 0.

5.3. Funkcija “Izvēlēties spēles laukuma izmēru”

ID: K3

Ievads: funkcija ļauj izvēlēties spēles laukuma izmēru.

Ievade:

Saskarne: “Spēles sākšanas saskarne” (skat. 6.2. nodaļu).

Trigeris: lietotājs no nolaižamā saraksta “Spēles laukuma izmērs” izvēlas laukuma lielumu (skat. 5.2. nodaļu).

Ievaddati: spēles laukuma izmērs – norāda spēles laukuma horizontālo un vertikālo rūtiņu skaitu. Izvēlei ir paredzēti četri varianti {3, 5, 7, 9}. Pēc noklusējuma ir vērtība “3”.

Apstrāde: sistēma iegaumē izvēlēto lielumu.

Izvade: “Spēles sākšanas saskarne” (skat. 6.2. nodaļu). Nolaižamajā sarakstā “Spēles laukuma izmērs” tiek attēlota izvēlētā vērtība.

5.4. Funkcija “Sākt spēļu ciklu”

ID: K4

Ievads: funkcija ļauj uzsākt jaunu 3 spēļu ciklu.

Ievade:

Saskarne: “Spēles saskarne” (skat. 6.3. nodaļu).

Trigeris:

- funkcija “Sākt spēli” (*K2*);
- funkcija “Uzvarēt spēles ciklā” (*K7*);
- vai funkcija “Zaudēt spēles ciklā” (*K10*).

Apstrāde:

1) spēles laukuma rūtiņas tiek attīrītas no krustiņiem un nullītēm;
2) spēles laukuma rūtiņām tiek piešķirta noklusējuma krāsa;
3) sistēma paziņo spēles numuru (skat. *3. b) attēlu*);
4) lietotājs nospiež pogu “Sākt spēli” (skat. *3. a)* attēlu);
5) ja spēli pirmais uzsāk dators, sistēma izsauc funkciju “Sagaidīt datora spēles gājienu” (*K9*);
   ja lietotājs – funkciju “Izpildīt spēles gājienu” (*K5*).

Izvade: “Spēles saskarne” (skat. 6.3. nodaļu).

5.5. Funkcija “Veikt spēles gājienu”

ID: K5

Ievads: funkcija ļauj lietotājam aizpildīt spēles laukuma rūtiņu ar krustiņu vai nullīti.

Ievade

Saskarne: “Spēles saskarne” (skat. 6.3. nodaļu).

Trigeris:

- funkcija “Sākt spēļu ciklu” (*K4*);
- vai funkcija “Sagaidīt datora spēles gājienu” (*K9*).

Lietotāja darbības: lietotājs izvēlas tukšu rūtiņu un ieklikšķina tajā ar peles kursoru.

Apstrāde:

1) sistēma gaida 5 sekundes, kamēr lietotājs ieklikšķinās tukšajā rūtiņā spēles laukumā. Atlikušais laiks tiek attēlots ar laika indikatoru. Ja lietotājs nepaspēj veikt savu gājienu 5 sekunžu laikā, sistēma pāriet pie datora gājiena (*K9*);
2) sistēma iegaumē rūtiņu, kas tika atzīmēta, un attēlo lietotāja simbolu (krustiņu vai nullīti) izvēlētajā šūnā;
3) sistēma pārbauda, vai ir izveidota līnija no lietotāja simboliem (*K6*);
4) ja līnija ir izveidota, tiek mainīts spēles rezultāts un paziņots par uzvaru spēļu ciklā (K7);
   ja līnija nav izveidota, spēles gājienu izdara dators (K9).

Izvade: “Spēles saskarne” (skat. 6.3. nodaļu). Spēles laukuma rūtiņai tiek piešķirts lietotāja simbols. Laika indikatora vērtība tiek atjaunota ik pēc 0,25 sekundēm.

5.6. Funkcija “Pārbaudīt, vai ir līnija”

ID: K6

Ievads: funkcija pārbauda, vai ir izveidota līnija no 3 simboliem.

Ievade

Trigeris:

- funkcija “Veikt spēles gājienu” (*K5*);
- funkcija “Sagaidīt datora spēles gājienu” (*K9*).

Ievaddati: funkcija saņem vērtības:

- simbolu, kas ir jāpārbauda (krustiņu vai nullīti);
- izvēlētās šūnas koordinātes (*x, y*), *x* – rindas numuru, *y* – kolonnas numuru.

Apstrāde:

1) sākot ar izvēlēto rūtiņu (*x, y*), sistēma pārbauda, vai ir izveidota līnija no 3 simboliem (horizontāli, vertikāli vai pa diagonāli);
2) funkcija atgriež vērtību, vai līnija ir izveidota, vai nē.

Izvade: vērtība, vai līnija ir izveidota, vai nē.

5.7. Funkcija “Uzvarēt spēļu ciklā”

ID: K7

Ievads: funkcija tiek izsaukta, kad lietotājs izveido rindu no krustiņiem vai nullītēm. Tiek atjaunots spēles rezultāts, un notiek pārbaude, vai lietotājs uzvarēja.

Ievade

Saskarne: “Spēles saskarne” (skat. 6.3. nodaļu)

Trigeris: funkcija “Veikt spēles gājienu” (*K5*).

Priekšnosacījums: lietotājs izveidoja līniju no 3 simboliem.

Apstrāde:

1) sistēma paziņo par lietotāja uzvaru spēļu ciklā;
2) sistēma palielina lietotāja uzvaru skaitu par vienu;
3) ja lietotājs spēļu ciklā uzvar divas reizes, viņam tiek piešķirta uzvara spēlē (*K8*);
   ja lietotājs neuzvar divas reizes, tad:
   4) sistēma maina spēles uzsākšanas kārtību: sāk lietotājs vai dators;
   5) sistēma uzsāk jaunu spēļu ciklu (*K4*).

Izvade

Pēcnosacījums: lietotāja uzvaru skaits spēļu ciklos ir palielināts par vienu.

*Saskarne: “*Spēles saskarne” (skat. 6.3. nodaļu). Spēles progresa indikators tiek atjaunots ar jaunu vērtību.

5.8. Funkcija “Uzvarēt spēlē”

ID: K8

Ievads: funkcija tiek izsaukta, ja lietotājs spēļu ciklā uzvar divas reizes. Programma apsveic lietotāju ar uzvaru.

Ievade

Saskarne: “Spēles saskarne” (skat. 6.3. nodaļu)

Trigeris: funkcija “Uzvarēt spēļu ciklā” (*K7*).

Priekšnosacījums: lietotājs spēļu ciklā uzvar divas reizes.

Apstrāde:

1) sistēma paziņo par lietotāja uzvaru spēlē (skat. *5. a) attēlu*);
2) sistēma atver sākuma saskarni ar instrukciju (skat. 5.2. nodaļu).

Izvade: “Spēles sākšanas saskarne” (skat. 6.2. nodaļu).

5.9. Funkcija “Sagaidīt datora spēles gājienu”

ID: K9

Ievads: funkcija tiek izsaukta, kad datora gājienā ir izveidota līnija no 3 simboliem.

Ievade

Saskarne: “Spēles saskarne” (skat. 6.3. nodaļu).

Trigeris: funkcija “Sagaidīt datora spēles gājienu” (*K9*).

Apstrāde:

1) sistēma izvēlas brīvu rūtiņu spēles laukumā un aizpilda to ar datora simbolu (rūtiņas izvēles algoritma realizāciju izdomā sistēmas izstrādātājs);
2) ja dators izveido rindu no trīs simboliem (*K6*), sistēma paziņo spēlētājam par sakāvi spēļu ciklā (*K10*); ja līnija nav izveidota, gājienu veic spēlētājs (*K5*).

Izvade: “Spēles saskarne” (skat. 6.3. nodaļu).

5.10. Funkcija “Zaudēt spēļu ciklā”

ID: K10

Ievads: funkcijas uzdevums attēlot un imitēt pretinieka gājienu.

Ievade

Saskarne: “Spēles saskarne” (skat. 6.3. nodaļu)

Trigeris: funkcija “Sagaidīt datora spēles gājienu” (*K9*).

Priekšnosacījums: sistēma izveido līniju no 3 simboliem ātrāk par lietotāju.

Apstrāde:

1) sistēma paziņo par lietotāja sakāvi spēļu ciklā;
2) sistēma palielina datora uzvaru skaitu par vienu;
3) ja dators spēļu ciklā uzvar divas reizes, lietotājam tiek piešķirta sakāve spēlē (*K11*);
   ja dators neuzvar divas reizes, tad:
   4) sistēma maina spēles uzsākšanas kārtību: sāk lietotājs vai dators;
   5) sistēma uzsāk jaunu spēļu ciklu (*K4*).

Izvade

Pēcnosacījums: datora uzvaru skaits spēļu ciklos ir palielināts par vienu.

Saskarne: “Spēles saskarne” (skat. 6.3. nodaļu). Spēles progresa indikators tiek atjaunots ar jaunu vērtību.

5.11. Funkcija “Zaudēt spēlē”

ID: K11

Ievads: funkcija tiek izsaukta, ja dators spēļu ciklā uzvar divas reizes. Programma paziņo spēlētājam par sakāvi spēlē.

Ievade

Saskarne: “Spēles saskarne” (skat. 6.3. nodaļu).

Trigeris: funkcija “Zaudēt spēļu ciklā” (*K10*).

Priekšnosacījums: lietotājs spēļu ciklā zaudē divas reizes.

Apstrāde:

1) sistēma paziņo spēlētājam par sakāvi spēlē;
2) sistēma atver sākuma saskarni ar instrukciju (skat. 6.2. nodaļu).

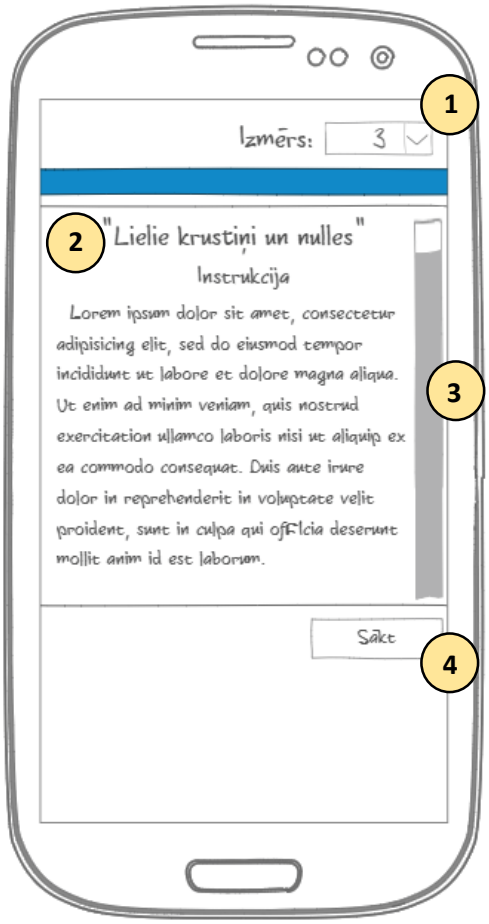
Izvade: “Spēles sākšanas saskarne” (skat. 6.2. nodaļu).

6. Ārējās saskarnes prasības

6.1. Ekrāna formāti: produkts atbalsta minimālo ekrāna izmēru 4” un minimālo izšķirtspēju ar 480x800 punktiem.

6.2. Lietotāja saskarne “Spēles sākšanas saskarne”

Saskarnes struktūra: skat. *3. attēlu.*



3. attēls. **Spēles sākšanas saskarne**

Saskarnes elementi:

1 – nolaižamais saraksts “Spēles lauka izmērs”; vērtību saraksts { 3, 5, 7, 9 };

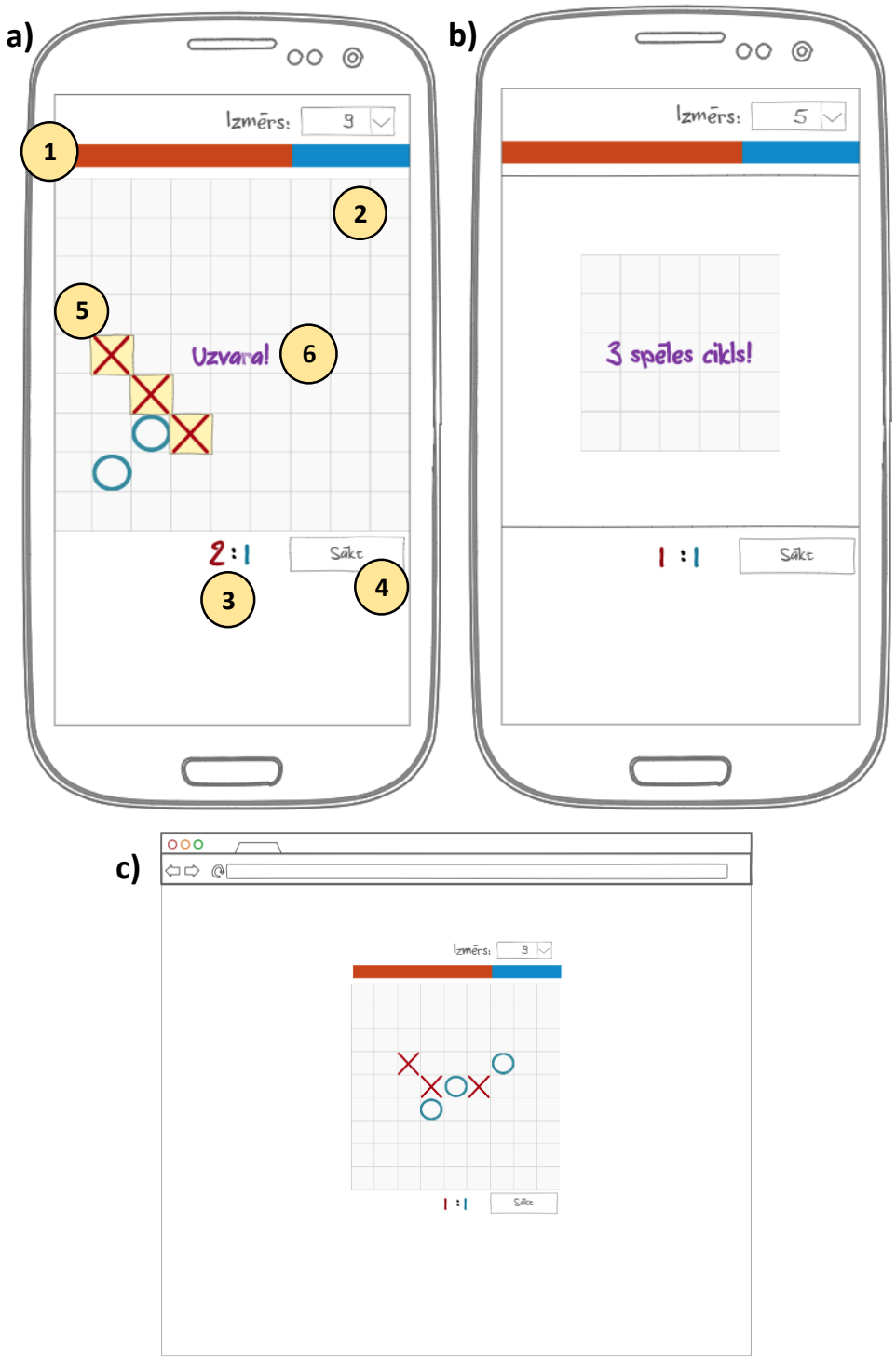
2 – teksta lauks “Spēles instrukcija”;

3 – ritjosla;

4 – poga “Sākt spēli”.

6.3. Lietotāja saskarne “Spēles saskarne”

Saskarnes struktūra: skat. *4. attēlu.*



4. attēls. **Spēles saskarne: a) spēles laukums 9x9, lietotāja uzvara un izveidota līnija; b) spēles laukums 5x5, spēles cikla sākums; c) gadījums ar lielu ekrāna izmēru**

Saskarnes elementi:

1 – laika indikators;
2 – spēles laukums;
3 – spēles progresa indikators;
4 – poga “Sākt spēli”;
5 – izveidota līnija no 3 simboliem;
6 – paziņojums.

7. Projekta ierobežojumi

7.1. Aparatūras ierobežojumi: programmai jāstrādā korekti ar pārlūkprogrammu *Chrome 66v.* vai jaunāku versiju.

7.2. Citas prasības: veicot tehnoloģiju izvēli, projektu realizēšanā jāievēro studiju kursā ietvertās programmatūras izstrādes tehnoloģijas.

8. Prasību tabula

ID	Prasības	Lpp.
K1	Izlasīt instrukciju	7.
K2	Sākt spēli	7.
K3	Izvēlēties spēles laukuma izmēru	8.
K4	Sākt spēļu ciklu	8.
K5	Veikt spēles gājienu	9.
K6	Pārbaudīt, vai ir līnija	9.
K7	Uzvarēt spēļu ciklā	10.
K8	Uzvarēt spēlē	10.
K9	Sagaidīt datora spēles gājienu	11.
K10	Zaudēt spēļu ciklā	11.
K11	Zaudēt spēlē	12.

3. pielikums

Mobilā lietotne
“Pirkumu saraksts”

Programmatūras prasību specifikācija

(Versija 1.0.0.)

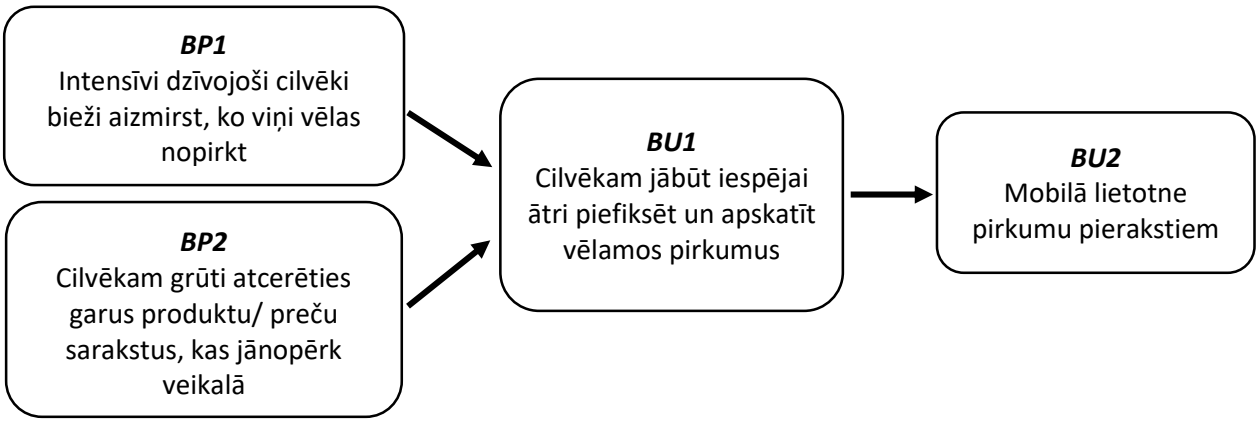
2019

Saturs

1. Ievads

1.1. Nolūks: šī programmatūras prasību specifikācija ir izstrādāta studentiem – topošajiem programmētājiem, lai viņi iegūtu priekštatu par programmatūras prasību specifikācijas struktūru un iemācītos izstrādāt datorprogrammas atbilstoši specifikācijai.

1.2. Darbības sfēra: mobilā lietotne "Pirkumu saraksts" ir elektroniska pierakstu grāmata, ar kuras palīdzību lietotājs var pierakstīt vēlamos produktus/ preces un vēlāk to apskatīt, apmeklējot veikalu. Sistēmas koncepcija dota *1. attēlā.*



1. attēls. **Mērķa modelis**

1.3. Definīcijas, akronīmi un saīsinājumi:

- *ID* – unikāls identifikators;
- lietošanas gadījuma diagramma – diagramma, kas attēlo sistēmas lietotājus un funkcijas;
- OS – operētājsistēma;
- PPS – programmatūras prasību specifikācija.

1.4. Saistība ar citiem dokumentiem

"LVS 68:1996 INFORMĀCIJAS TEHNOLOĢIJA. PROGRAMMINŽENIERIJA. PROGRAMMATŪRAS PRASĪBU SPECIFIKĀCIJAS CEĻVEDIS".

1.5. Pārskats

***2. nodaļā* "Vispārējais apraksts"** ir aprakstīta:

5) *produkta perspektīva* – produkta apraksts nākotnes skatījumā un saistībā ar citiem produktiem vai projektiem;
6) *lietotāja raksturiezīmes* – produkta lietotāja vispārīgas raksturiezīmes, kas ietekmē specifiskās prasības;
7) *produkta funkcijas* – kopsavilkums par funkcijām, ko izpilda programmatūra;
8) *kopsavilkums* – lietošanas gadījumu diagramma, kas attēlo produkta funkcijas un lietotājus.

***3. nodaļā* "Vispārējie ierobežojumi"** ir aprakstīti projekta ierobežojumu iemesli.

***4. nodaļā* "Pieņēmumi un atkarības"** ir doti mobilo telefonu parametri.

***5. nodaļā* "Ārējās saskarnes prasības"** ir aprakstītas lietotāja saskarnes prasības un attēlota saskarņu struktūra.

***6. nodaļā* "Projekta ierobežojumi"** ir aprakstīti konkrēti produkta realizācijas ierobežojumi.

2. Vispārējais apraksts

2.1. Produkta perspektīva: mobilā lietotne “Pirkumu saraksts” ir neatkarīga programma, kas sastāv no piecām saskarnēm (skat. *6. nodaļā*). Galvenās saskarnes ir “Pirkumu saraksts” un “Veikalu saraksts”. Pārējās saskarnes ir paredzētas, lai rediģētu pirkumu un veikalu datus. Nākamajā versijā ir paredzēts pievienot vizuālo un skaņas funkciju par pirkumu atgādināšanu. Lietotāji varēs apmainīties ar receptēm vai pārsūtīt pirkumu sarakstu, kā arī tiks vākta pirkumu un veikalu statistika, kuru izmantos mākslīgais intelekts, lai iegaumētu lietotāja preferences un piedāvātu priekšlikumus, pamatojoties uz tām.

2.2. Lietotāja raksturiezīmes: produkts ir paredzēts mobilo telefonu lietotājiem, kuri dzīvo Latvijā un saprot latviešu valodu. Vecuma vai izglītības ierobežojumi nav paredzēti, tomēr par primāro lietotāju grupu tiek uzskatīti cilvēki ar intensīvu dzīves ritmu, kas aizmirst par nepieciešamajiem pirkumiem, vai mājsaimnieces, kuras plāno savus pirkumus. Nākamajās versijās ir paredzētas tādas grupas kā “Draugi” un “Ģimenes locekļi”. Draugi varēs dalīties, piemēram, ar receptēm, bet ģimenes locekļi – pārsūtīt pirkumu sarakstus.

2.3. Produkta funkcijas

Mobilajai lietotnei “Pirkumu saraksts” ir deviņas funkcijas:

(*K1*) “Apskatīt pirkumu sarakstu” – funkcija ļauj lietotājam apskatīt produktus/ preces, ko viņš pievienoja pirkumu sarakstam.

(*K2*) “Pievienot pirkumu” – funkcija ļauj lietotājam pievienot produktu/ preci pirkumu sarakstam.

(*K3*) “Atzīmēt, ka nopirkts” – funkcija ļauj lietotajam atzīmēt, ka produkts/ prece ir nopirkta.

(*K4*) “Dzēst pirkumu” – funkcija ļauj izdzēst produktu/ preci no pirkumu saraksta.

(*K5*) “Filtrēt pirkumu sarakstu” – funkcija ļauj filtrēt pirkumu sarakstu pēc veikala.

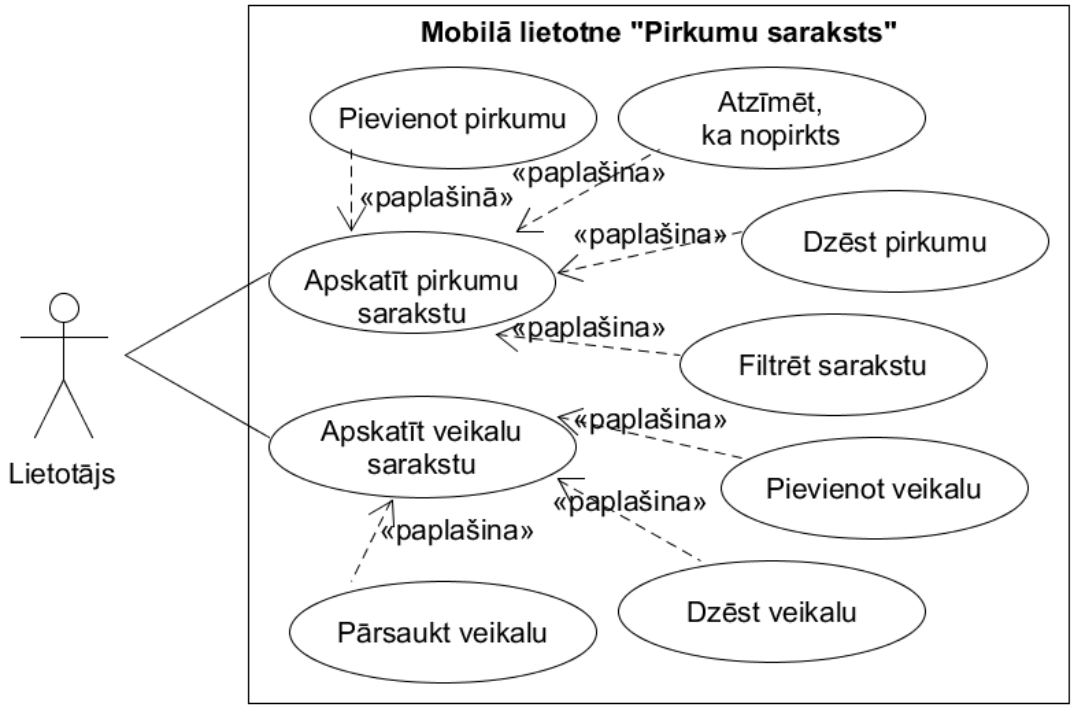
(*K6*) “Apskatīt veikalu sarakstu” – funkcija ļauj lietotājam apskatīt veikalu sarakstu.

(*K7*) “Pievienot veikalu” – funkcija ļauj pievienot veikalu vai tā kategoriju sarakstam.

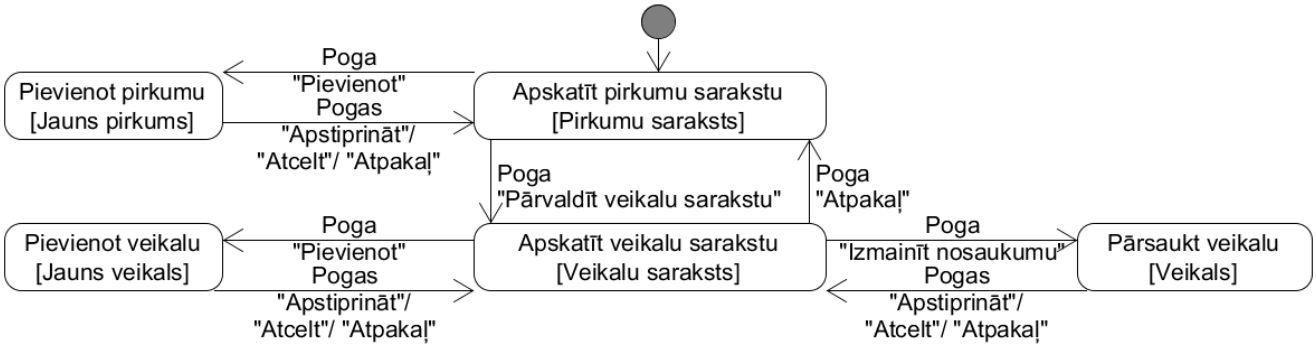
(*K8*) “Pārsaukt veikalu” – funkcija ļauj izmainīt veikala vai kategorijas nosaukumu.

(*K9*) “Dzēst veikalu” – funkcija ļauj nodzēst veikalu vai kategoriju veikalu sarakstā.

2.4. Kopsavilkums: funkciju pārskatāmībai ir izveidota programmas lietošanas gadījuma diagramma (*1. attēls*), bet, lai attēlotu pārejas un saites starp saskarnēm, ir pievienota stāvokļu diagramma (*2. attēls*).



1. attēls. **Mobilās lietotnes “Pirkumu saraksts” lietošanas gadījuma diagramma**



2. attēls. **Mobilās lietotnes “Pirkumu saraksts” funkciju un saskarņu saites**
(kvadrātiekavās ir saskarņu nosaukumi)

3. Vispārējie ierobežojumi

Projekta ierobežojumi

Lietotne ir paredzēta mobilajiem telefoniem. *Android* ir visbiežāk izmantojamā mobilo telefonu OS Latvijā.

Ievērojot, ka šī PPS mācību nolūkos ir izstrādāta studentiem, projekta realizācijas tehnoloģiju izvēle ir ierobežota mācību kursa ietvaros.

4. Pieņēmumi un atkarības

Tiek pieņemts, ka lietotājam ir mobilais telefons ar minimālo *Android* OS versiju *KitKat 4.4*.

5. Konkrētās prasības

5.1. Funkcija “Apskatīt pirkumu sarakstu”

ID: K1

Ievads: funkcija ļauj lietotājam apskatīt produktus/ preces, ko viņš pievienoja pirkumu sarakstam.

Ievade: lietotājs aktivizē programmu.

Apstrāde: tiek attēlota saskarne “Pirkumu saraksts” (skat. *6.2. nodaļu*).

Izvade: saskarne “Pirkumu saraksts” (skat. *6.2. nodaļu*).

5.2. Funkcija “Pievienot pirkumu”

ID: K2

Ievads: funkcija ļauj lietotājam pievienot produktu/ preci pirkumu sarakstam.

Ievade

Saskarne “Pirkumu saraksts” (skat. *6.2. nodaļu*).

Trigeris: poga “Pievienot”.

Apstrāde:

1) lietotājam tiek attēlota saskarne “Jauns pirkums” (skat. *6.4. nodaļu*);

2) lietotājs ievada pirkuma nosaukumu un izvēlas veikalu;

3) lietotājs var pievienot piezīmi;

4) lietotājs nospiež pogu “Apstiprināt”.

Izvade: ieraksts tiek saglabāts, un tiek atvērta saskarne “Pirkumu saraksts” (skat. *6.2. nodaļu*), kurā ir redzams pievienotais produkts/ prece.

5.3. Funkcija "Atzīmēt, ka nopirkts"

ID: K3

Ievads: funkcija ļauj lietotajam atzīmēt, ka produkts/ prece ir nopirkta.

Ievade

Saskarne "Pirkumu saraksts" (skat. *6.2 nodaļu*).

Priekšnosacījums: ir izvēlēts produkts/ prece sarakstā.

Trigeris: poga "Nopirkts".

Apstrāde

Ieraksts par produktu/ preci tiek dzēst no saraksta.

Izvade: saskarne "Pirkumu saraksts" (skat. *6.2. nodaļu*) tiek atjaunota, un tajā vairs nav nopirktā produkta/ preces.

5.4. Funkcija "Dzēst pirkumu"

ID: K4

Ievads: funkcija ļauj izdzēst produktu no pirkumu saraksta.

Ievade

Saskarne "Pirkumu saraksts" (skat. *6.2 nodaļu*).

Priekšnosacījums: ir izvēlēts produkts/ prece sarakstā.

Trigeris: poga "Nodzēst".

Apstrāde

Ieraksts par produktu/ preci tiek dzēsts sarakstā.

Izvade: saskarne "Pirkumu saraksts" (skat. *6.2. nodaļu*) tiek atjaunota, un tajā vairs nav nodzēstā produkta/ preces.

5.5. Funkcija "Filtrēt pirkumu sarakstu"

ID: K5

Ievads: funkcija ļauj filtrēt pirkumu sarakstu pēc veikala.

Ievade

Saskarne "Pirkumu saraksts" (skat. *6.2. nodaļu*).

Trigeris: nolaižamajā sarakstā "Veikali" ir izvēlēta kategorija/ veikals.

Apstrāde

Sistēma attēlo produktus, kas atbilst filtrēšanas nosacījumam (izvēlētajam veikalam/ kategorijai).

Izvade: saskarnē "Pirkumu saraksts" (skat. *6.2. nodaļu*) tiek attēloti tikai tie produkti/ preces, kas atbilst izvēlētajam veikalam/ kategorijai.

5.6. Funkcija “Apskatīt veikalu sarakstu”

ID: K6

Ievads: funkcija ļauj lietotājam apskatīt veikalu sarakstu.

Ievade

Saskarne “Pirkumu saraksts” (skat. *6.2. nodaļu*).

Trigeris: poga “Pārvaldīt veikalu sarakstu”.

Apstrāde

Sistēma atver saskarni “Veikalu saraksts”.

Izvade: saskarne “Veikalu saraksts” (skat. *6.3. nodaļu*).

5.7. Funkcija “Pievienot veikalu”

ID: K7

Ievads: funkcija ļauj pievienot veikalu vai tā kategoriju sarakstam.

Ievade

Saskarne “Veikalu saraksts” (skat. *6.3. nodaļu*).

Trigeris: poga “Pievienot”.

Apstrāde:

1) sistēma atver saskarni “Jauns veikals” (skat. *6.5. nodaļu*);

2) lietotājs ievada veikala nosaukumu vai kategorijas nosaukumu;

3) lietotājs nospiež pogu “Apstiprināt”.

Izvade: ieraksts tiek saglabāts, un tiek atvērta saskarne “Veikalu saraksts” (skat. *6.3. nodaļu*), un veikalu sarakstā ir pievienotais veikals/ kategorija.

5.8. Funkcija “Pārsaukt veikalu”

ID: K8

Ievads: funkcija ļauj izmainīt veikala vai kategorijas nosaukumu.

Ievade

Saskarne “Veikalu saraksts” (skat. *6.3. nodaļu*).

Priekšnosacījums: izvēlēts veikals/ kategorija.

Trigeris: poga “Izmainīt nosaukumu”.

Apstrāde:

1) sistēma atver saskarni “Veikals” (skat. *6.6. nodaļu*);

2) lietotājs ievada jaunu veikala vai kategorijas nosaukumu;

3) lietotājs nospiež pogu “Apstiprināt”.

Izvade: ieraksts tiek saglabāts, un tiek atvērta saskarne “Veikalu saraksts” (skat. *6.3. nodaļu*), kurā ir veikals/ kategorija ar jaunu nosaukumu.

5.9. Funkcija “Dzēst veikalu”

ID: K9

Ievads: funkcija ļauj nodzēst veikalu vai kategoriju no veikalu saraksta.

Ievade

Saskarne “Veikalu saraksts” (skat. *6.3. nodaļu*).

Priekšnosacījums: izvēlēts veikals/ kategorija.

Trigeris: poga “Nodzēst”.

Apstrāde

Ieraksts par veikalu tiek nodzēsts.

Izvade: saskarnē “Veikalu saraksts” (skat. *6.3. nodaļu*) vairs nav nodzēstā veikala/ kategorijas nosaukuma.

6. Ārējās saskarnes prasības

6.1. Ekrāna formāti: programma atbalsta minimālo 4” ekrāna izmēru, minimālo izšķirtspēju – 480x800 punkti un tikai vertikālo ekrāna izvietojumu.

6.2. Lietotāja saskarne “Pirkumu saraksts”

Saskarnes struktūru skatīt *2. attēlā.*

2. attēls. **Saskarne “Pirkumu saraksts” (sākuma saskarne)**

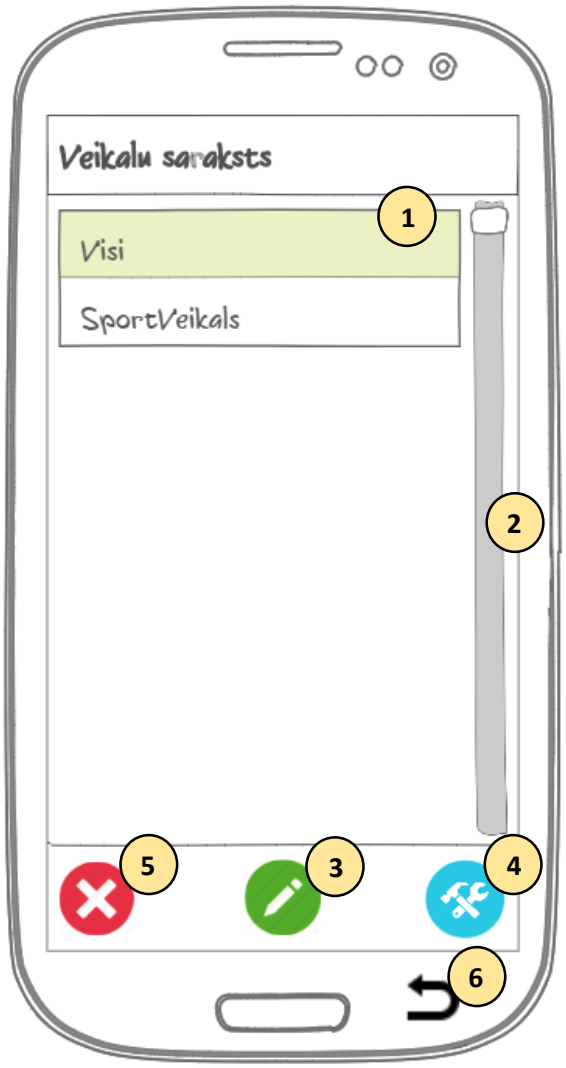
Saskarnes elementi:

1) saraksts “Pirkumu saraksts”;
2) nolaižamais saraksts “Veikali” (pēc noklusējuma ir kategorija “Visi”);
3) poga “Pārvaldīt veikalu sarakstu”;
4) ritjosla;
5) poga “Pievienot”;
6) poga “Nodzēst”;
7) poga “Nopirkts”;
8) poga “Atpakaļ”.

.

6.3. Lietotāja saskarne “Veikalu saraksts”

Saskarnes struktūru skatīt *3. attēlā.*



3. attēls. **Saskarne “Veikalu saraksts”**

Saskarnes elementi:

1) saraksts “Veikalu saraksts” (kategorija “Visi” eksistē pēc noklusējuma, to nevar nodzēst);
2) ritjosla;
3) poga “Pievienot”;
4) poga “Izmainīt nosaukumu”;
5) poga “Nodzēst”;
6) poga “Atpakaļ”.

6.4. Lietotāja saskarne “Jauns pirkums”

Saskarnes struktūru skatīt *4. attēlā.*

4. attēls. **Saskarne “Jauns pirkums”**

Saskarnes elementi:

1) ievadlauks “Pirkuma nosaukums” (satur ieteikumu “Nosaukums”);
2) nolaižamais saraksts “Veikali”;
3) ievadlauks “Piezīmes” (satur ieteikumu “Piezīmes”);
4) poga “Apstiprināt”;
5) poga “Atcelt”;
6) poga “Atpakaļ”.

.

6.5. Lietotāja saskarne “Jauns veikals”

Saskarnes struktūru skatīt *5. attēlā.*

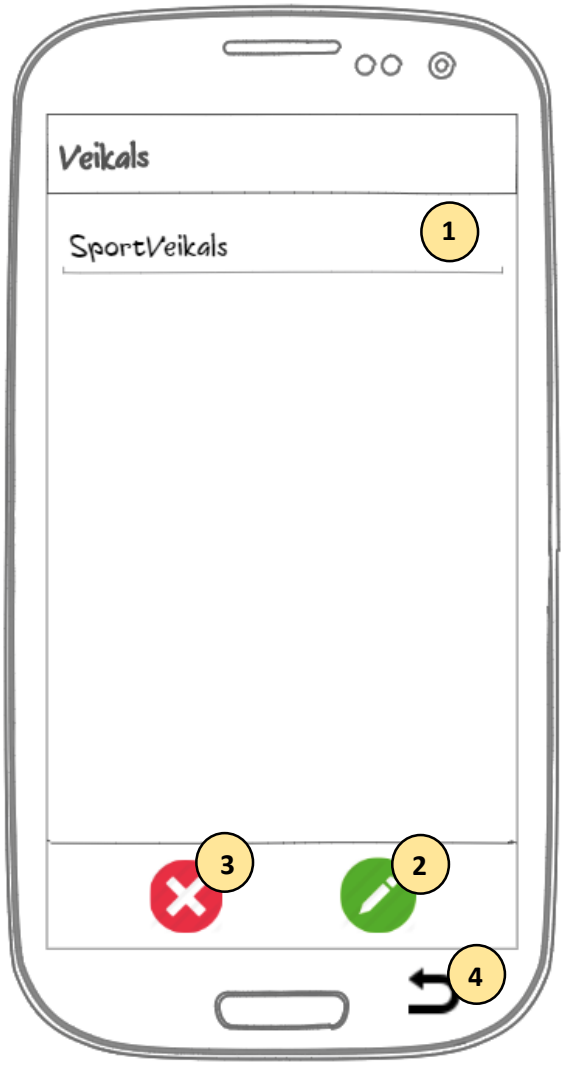
5. attēls. **Saskarne “Jauns veikals”**

Saskarnes elementi:

1) ievadlauks “Veikala nosaukums” (satur ieteikumu “Nosaukums vai kategorija”);
2) poga “Apstiprināt”;
3) poga “Atcelt”;
4) poga “Atpakaļ”.

6.6. Lietotāja saskarne “Veikals”

Saskarnes struktūru skatīt *6. attēlā.*



6. attēls. **Saskarne “Veikals”**

Saskarnes elementi:

1) ievadlauks “Veikala nosaukums” (satur ieteikumu “Nosaukums vai kategorija”);
2) poga “Apstiprināt”;
3) poga “Atcelt”;
4) poga “Atpakaļ”.

7. Projekta ierobežojumi

7.1. Aparatūras ierobežojumi: programmai ar *Android* OS versiju *KitKat 4.4* vai jaunāku jāstrādā korekti.

7.2. Citas prasības: veicot tehnoloģiju izvēli, jāievēro mācību kursā ietvertās programmatūras izstrādes tehnoloģijas, lai realizētu projektu.

8. Prasību tabula

ID	**Prasības**	**Lpp.**
K1	Apskatīt pirkumu sarakstu	6.
K2	Pievienot pirkumu	6.
K3	Atzīmēt, ka nopirkts	7.
K4	Dzēst pirkumu	7.
K5	Filtrēt pirkumu sarakstu	7.
K6	Apskatīt veikalu sarakstu	8.
K7	Pievienot veikalu	8.
K8	Pārsaukt veikalu	8.
K9	Dzēst veikalu	9.