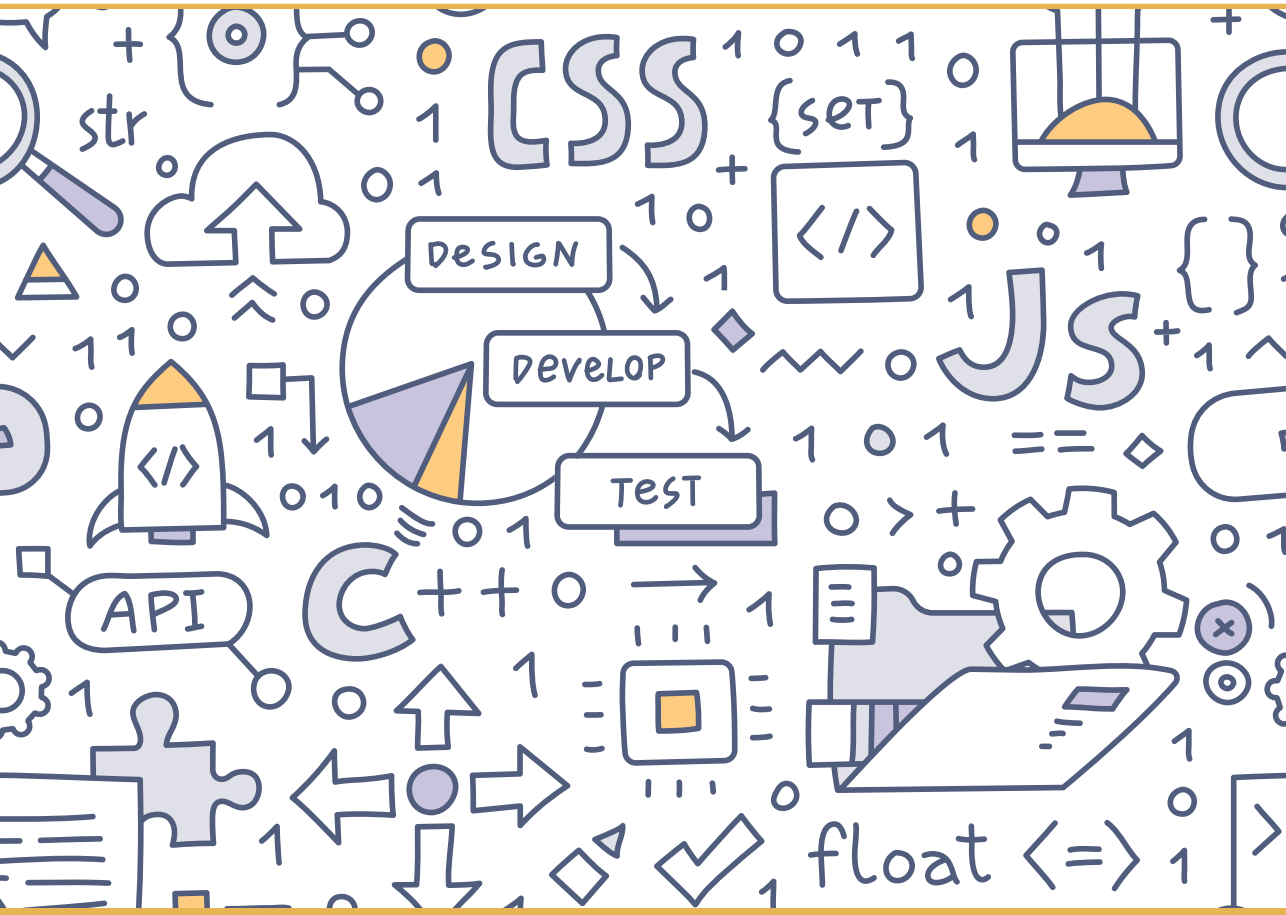
RĪGAS TEHNISKĀ UNIVERSITĀTE

Kristaps Babris

DIVPUSLOŽU MODEĻA TRANSFORMĀCIJĀ SAKŅOTA TĪMEKĻA LIETOTNES LIETOTĀJA SASKARNES PROTOTIPA ĢENERĒŠANA

Promocijas darba kopsavilkums



RTU Izdevniecība
Rīga 2025

RĪGAS TEHNISKĀ UNIVERSITĀTE
Datorzinātnes, informācijas tehnoloģijas un enerģētikas fakultāte

Kristaps Babris
Doktora studiju programmas "Datorzinātne un informācijas tehnoloģija" doktorants

DIVPUSLOŽU MODEĻA TRANSFORMĀCIJĀ SAKŅOTA TĪMEKĻA LIETOTNES LIETOTĀJA SASKARNES PROTOTIPA ĢENERĒŠANA

Promocijas darba kopsavilkums

Zinātniskā vadītāja
profesore *Dr. sc. ing.*
OKSANA ŅIKIFOROVA

RTU Izdevniecība
Rīga 2025

Babris K. Divpusložu modeļa transformācijā sakņota tīmekļa lietotnes lietotāja saskarnes prototipa ģenerēšana. Promocijas darba kopsavilkums. – Rīga: RTU Izdevniecība, 2025. – 44 lpp.

Publicēts saskaņā ar promocijas padomes "RTU P-07" 2024. gada 7. oktobra lēmumu, protokols Nr. 24-5.

Vāka attēls no *www.shutterstock.com*.

https://doi.org/10.7250/9789934371301
ISBN 978-9934-37-130-1 (pdf)

PROMOCIJAS DARBS IZVIRZĪTS ZINĀTNES DOKTORA GRĀDA IEGŪŠANAI RĪGAS TEHNISKAJĀ UNIVERSITĀTĒ

Promocijas darbs zinātnes doktora (*Ph. D.*) grāda iegūšanai tiek publiski aizstāvēts 2025. gada 3. februārī Rīgas Tehniskās universitātes Datorzinātnes, informācijas tehnoloģijas un enerģētikas fakultātē, Zunda krastmalā 10, 204. auditorijā.

OFICIĀLIE RECENZENTI

Profesors *Dr. habil. sc. ing.* Jānis Grundspeņķis,
Rīgas Tehniskā universitāte

Profesors *Dr. sc. ing.* Gatis Vītols,
Latvijas Biozinātņu un tehnoloģiju universitāte, Latvija

Profesors *Dr. hab. Leszek Maciaszek*,
Honorary Research Fellow, Macquarie University – Sydney, Austrālija
Emeritus Professor, Wrocław University of Economics and Business, Polija

APSTIPRINĀJUMS

Apstiprinu, ka esmu izstrādājis šo promocijas darbu, kas iesniegts izskatīšanai Rīgas Tehniskajā universitātē zinātnes doktora (*Ph. D.*) grāda iegūšanai. Promocijas darbs zinātniskā grāda iegūšanai nav iesniegts nevienā citā universitātē.

Kristaps Babris ……. (paraksts)
Datums: ……

Promocijas darbs ir uzrakstīts latviešu valodā, tajā ir ievads, četras nodaļas, secinājumi, literatūras saraksts, 76 attēli, 45 tabulas, viens pielikums, kopā 196 lappuses, neieskaitot pielikumus (kopā ar informācijas avotu sarakstu un pielikumiem 229 lapaspuses). Literatūras sarakstā ir 253 nosaukumi.

SATURS

IEVADS

Programmatūras izstrāde kā inženierijas nozare ir samēra jaunā, salīdzinot ar būvniecību, medicīnu, matemātiku, fiziku u. tml., tai kopš 1969. gadā konstatētās programmēšanas krīzes (*Aspray, Keil et al.*, 1999) un dibināšanas par inženierijas nozari promocijas darba izstrādes noslēguma gadā ir 55 gadi. Šajā laikā ir veikti vairāki pētījumi par programmatūras izstrādes metodoloģiju, realizācijas risinājumu, tehnoloģiju un arhitektūras veidu attīstīšanā. Mūsdienās ir nostabilizējies efektīvas programmatūras izstrādes pamatkoncepciju kopa, kas ir spējās metodoloģijas *(Calvary, Coutaz et al.*, 2003; *Al-Saqqa, Sawalha et al.*, 2020), automatizācija (*Narang & Mittal*, 2022; *Yigitbas, Jovanovikj et al.*, 2020), modulāra un elastīga arhitektūra (*Akiki, Bandara et al.*, 2014; *Mbuga, Korongo et al.*, 2022), satvaru izmantošana, efektīva testēšana un ieguldījumi izstrādātāju rīkos (*Akiki, Bandara et al.*, 2014; *Mbuga, Korongo et al.*, 2022). Automatizācijas ieviešana atkārtotiem uzdevumiem, piemēram, testēšanai, būvēšanai un izvietošanai, ievērojami paātrina izstrādes procesu (*Nikiforova, Babris et al.*, 2021).

Mūsdienu programmatūras risinājumi tiek veidoti no trīs aspektiem.

1. Biznesa loģikas komponenti, kuros ir realizēti lietojumprogrammas darbības noteikumi un procesi, kas veido sistēmas pamatu.
2. Dati nodrošina informāciju, kas ir būtiska gan analīzei, gan operatīvai darbībai, ļaujot pieņemt pamatotus lēmumus un uzlabot uzņēmuma procesus.
3. Lietotāja saskarne nodrošina lietotāju pieredzi un mijiedarbību ar sistēmu.

Lietotāja saskarne ir programmatūras kritiskais komponents, jo darbojas kā tilts starp lietotāju un programmatūru, padarot sistēmas realizācijas iespējas pieejamas un izmantojamas galalietotājiem. Lietotāja saskarne (angļu val. *User Interface, UI*) ir jebkura lietojumprogrammas galvenā sastāvdaļa *(Nguyen, Vu et al.*, 2018), un tai ir izšķiroša nozīme lietojumprogrammas mijiedarbībai ar lietotāju. Tomēr lietotāja saskarne nav neatkarīga no tā lietošanas konteksta, kas definēts kā lietotājs, platforma un vide *(Calvary, Coutaz et al.*, 2003; *Yigitbas, Jovanovikj et al.*, 2020). Faktiski lietojumprogrammas biznesa loģika var būt viena, bet lietotāja saskarne var būt realizēta un piemērota katrai atsevišķai platformai (mobilās ierīces, dažādas operētājsistēmas utt.). Turklāt visaptveroša dažādu jomu digitalizācija nosaka paaugstinātas prasības informācijas sistēmu veidošanas ātrumam, kad vairs nepastāv problēma piedāvāt programmatūras inženierijas risinājumus jebkurā jomā, taču galvenais uzdevums ir piedāvāt to ātrāk par konkurentiem. Un tajā pašā laikā mūsdienu lietotāja saskarnes kļūst aizvien sarežģītākas, jo nepieciešams atbalstīt dažādus heterogēnus lietošanas kontekstus un vairs nav efektīvi nodrošināt tikai vienu lietojamu lietotāja saskarni visiem gadījumiem. Tomēr regulāra šāda kontekstu jeb platformu izmaiņu veikšana manuāli rada nozīmīgas izmaksas, variācijas un sarežģīti pārvaldāmus projektus (*Akiki, Bandara et al.*, 2014).

RISINĀŠANAI IZVĒLĒTĀ PROBLĒMA

Problēma, kas definēta risināšanai šajā promocijas darbā, ir saistīta ar nepieciešamību piedāvāt lietotāja saskarnes komponentu izstrādes pieeju, kas nodrošinātu automatizētu un līdz ar to ātru lietotāju saskarnes komponentu iegūšanu, ņemot vērā zināšanas par problēmas vidi,

kas turklāt būs neatkarīgas no platformas un realizācijas detaļām, tas ir, nodrošināta pietiekami augsta problēmvides zināšanu abstrakcija. Šādām prasībām atbilstošs risinājums var būt modeļvadāmā izstrāde (angļu val. *Model-Driven Development, MDD*), kas paredz, ka problēmvides zināšanas ir atspoguļotas modeļa veidā no platformas neatkarīgajā līmenī un ir nodrošināts formālisms šī modeļa transformācijai platformai specifiskajā līmenī. Modeļvadamā izstrāde sāka attīstīties kopš 2001. gada (*Lycett, Marcos et al.*, 2007) ar ideju pārveidot programmatūras izstrādes procesu par pilnīgi automatizētu, bet, neskatoties uz acīmredzamajām modeļvadāmas izstrādes priekšrocībām (piemēram, modelējot var ieraudzīt sistēmas kopskatu un sākotnējo reprezentāciju, veikt tajā labojumus, panākot pilnīgu atbilstību problēmvides zināšanām, un tikai tad papildināt sistēmas modeli ar platformas detaļām), šīs izstrādes pieejas ieviešana līdz visu programmatūras izstrādes aspektu automatizācijai nav nonākusi.

Mūsdienās modeļvadāma izstrāde nodrošina sistēmas statisku artefaktu (piemēram, klašu definīcijas no *ER* diagrammām) ģenerēšanu, savukārt pie dinamiskiem elementiem ir jāpiestrādā. Tas pats attiecas arī uz lietotāja saskarni, kur kvalitatīvi izveidotam modelim var daļēji uzģenerēt formas un to saturu, bet pilnvērtīgu lietotāja saskarnes prototipu risinājumi, kas būtu gatavi nodošanai realizācijā, patlaban vēl nav izstrādāti un ieviesti praksē.

Pēdējā laikā sāk attīstīties mēģinājumi lietot ģeneratīvo mākslīgo intelektu (*Stige, Zamani et al., 2023*; *Silva (da), Martin et al.*, 2011) arī lietotāju saskarnes elementu ģenerēšanai. Lai gan ģeneratīvie rīki ir izcili dažos radošos uzdevumos, lietotāja saskarnes dizaina niansētais un ļoti precīzais raksturs tiem rada grūtības. Ģenerētie lietotāja saskarnes varianti ir mākslīgā intelekta algoritmu lietošanas rezultāta "fantāzijas", kam vajag ļoti precīzi definēt prasības un zināšanu bāzi ar zināšanu likumiem, kas pēc būtības jau ir topošās lietotāja saskarnes skices un prasa specifiskas prasmes, lai darbotos ar ģeneratīvā mākslīgā intelekta rīkiem (*Alfaridzi & Yulianti*, 2020; *Riccio, Jahangirova et al.*, 2020). Tā rezultātā ģenerētā lietotāja skiču apraksta definēšana (*Kompaniets, Lyz et al.*, 2020; *Johnson, Gross et al.*, 2009) un iegūtā rezultāta verificēšana prasa pārmērīgus resursu to veidošanai, testēšanai un atkļūdošanai (*Alfaridzi & Yulianti*, 2020; *Riccio, Jahangirova et al.*, 2020). Projektētāju loma patlaban joprojām ir būtiska, lai nodrošinātu precizitāti, funkcionalitāti un uz lietotāju vērstus aspektus, kas ir būtiski efektīvai lietotāja saskarnes izstrādei (*Newmanm & Landay*, 2000; *Li, Cao et al.*, 2023), un lietotāja saskarnes skiču veidošana joprojām ir izaicinājums dizaineriem, lai pārvarētu plaisu starp koncepciju un radīšanu (*Nikiforova, Zabiniako et al.*, 2021b).

Savā veidā ģeneratīva mākslīga intelekta izmantošanu lietotāja saskarnes ģenerēšanai arī var saukt par modeļos sakņotu lietotāja saskarnes veidošanu, jo ģeneratīvā mākslīga intelekta algoritmi par avota modeli izmanto zināšanas par problēmas vidi un rezultātā kā mērķa modeli piedāvā variantus lietotāja saskarnes skicēm (*Sharp, Rogers et al.*, 2019; *Planas, Daniel et al.*, 2021). Problēmvides modeļu izveide un transformācija, kas balstīta metamodelēšanas principos, var dot iespēju realizēt lietotāja saskarnes satura automātisku ģenerēšanu no problēmvides apraksta, ja šis apraksts ir izveidots formāli (*Joo, 2019*; *Beek & McIver*, 2021), piemēram, ar modeli, kas ir pilnīgs un konsekvents. Lai programmatūras galaprodukts būtu kvalitatīvs, sākotnēji ir jāizveido kvalitatīvi modeļi, kas ietver pilnīgu un visaptverošu informāciju par problēmvidi (*Osis & Asnina*, 2011).

PĒTĪJUMA OBJEKTS, PRIEKŠMETS UN HIPOTĒZE

Pētījuma objekts šajā promocijas darbā ir tīmekļa lietotnes lietotāja saskarne, jo īpaši tīmekļa lietotnes kontekstā, kur lietotāji sagaida ātru, intuitīvu un patīkamu pieredzi, saskarnes nozīme ir vēl lielāka. **Pētījuma priekšmets** ir lietotāja saskarnes prototipa automatizēta ģenerēšana, kur lietotāja saskarne ir veidota, ņemot vērā problēmvides zināšanas, kas ir attēlotas modeļa veidā un iever pilnīgu un nepretrunīgu informāciju par topošas programmatūras lietošanas loģiku. Ir plānots, ka promocijas darbā piedāvātā risinājuma rezultātā no problēmvides modeļa ģenerētie lietotāja saskarnes prototipi ir gan pietiekami vienkārši, gan atkārtoti lietojami un, balstoties modeļvadāmās izstrādes principos (*Beltramelli, 2018*; *Hussmann, Meixner et al.*, 2011), ir iespējams ģenerēt jaunus prototipus, veicot izmaiņas problēmvides modelī. Lai šāds risinājums darbotos pareizi, ir nepieciešams izvēlēties notāciju avota modelim, kas ietver pilnīgas un nepretrunīgas zināšanas par problēmvides loģiku. Un, zinot sagaidāmā mērķa modeļa saturu un notāciju, ir iespējams definēt transformācijas likumus, kas sakņosies avota un mērķa modeļa metamodeļos. Par avota modeli risinājuma izstrādē ir izvēlēts divpusložu modelis (*Nikiforova, Kozacenko et al.*, 2015), tā pilnīgums, nepretrunīgums un lietojamība jau ir pierādīti iepriekšējos pētījumos un publikācijas (*Nikiforova & Pavlova*, 2011; *Bajovs, Nikiforova et al.*, 2013; *Kozačenko*, 2014; *Nikiforova & Gusarovs*, 2020; *Gusarovs*, 2020) dažādu programmatūras artefaktu izgūšanā. (*Gusarovs*, 2020) ir pierādīts, ka:

1) divpusložu modelis var kalpot kā modelis, kas ietver pietiekamu informāciju problēmvides atbalsta programmatūras sistēmas ģenerēšanai;
2) biznesa procesu modelēšana ir viena no ierastajām programmatūras inženierijas aktivitātēm, līdz ar to nepieciešamība veidot divpusložu modeli, kas ietver biznesa procesa modeli un atbilstošo konceptu modeli, neprasa papildu resursus un izmaiņas tradicionālajā izstrādē.

Attiecīgi darbā ir izvirzīta šāda **hipotēze**: ja biznesa loģiku atbalstošo pirmkodu var ģenerēt no divpusložu modeļa (*Gusarovs*, 2020), tad no divpusložu modeļa ir iespējams ģenerēt arī lietotāja saskarnes prototipu, kas pēc būtības ir biznesa loģikas darbības rezultāta attēlojums informācijas sistēmas lietotāja pusē.

PROMOCIJAS DARBA MĒRĶIS UN UZDEVUMI

Promocijas darba **mērķis** ir izstrādāt pieeju lietotāja saskarnes prototipa automātiskajai ģenerēšanai no divpusložu modeļa, kurā ir definēti problēmvides loģikas scenāriji. Turklāt ir plānots, ka iegūtais lietotāja saskarnes prototips tiek definēts modeļa veidā, kas turpmāk ir transformējams kādā no tīmekļa lietotnes priekšgala komponenšu satvariem.

Mērķa sasniegšanai definēti vairāki uzdevumi.

1. Izpētīt esošos lietotāja saskarnes izstrādes metodes un paņēmienus, lai identificētu potenciālos pamatelementus un formālismu, ko var izmantot lietotāja saskarnes automātiskajā ģenerēšanā.
2. Apkopot informāciju par lietotāja saskarnes elementu daudzveidību, lai definētu lietotāja saskarnes elementu taksonomiju un šablonus, kas kalpos par pamatu mērķa metamodeļa izstrādē.

3. Bagātināt divpusložu modeļa notāciju ar transformācijas likumiem nepieciešamajiem elementiem, kas ir procesu metadati, un izveidot divpusložu modeļa (kā avota modeļa) metamodeli.
4. Izveidot formātu avota/mērķa modeļa saturam, lai definētu transformācijas likumus, kur par avota modeli tiek izmantots problēmvides divpusložu modelis un par mērķa modeli tiek sagaidīts lietotāja saskarnes prototips šīs problēmvides atbalsta tīmekļa lietotnei.
5. Demonstrēt izstrādāto risinājumu lietotāja saskarnes prototipa ģenerēšanai ar reālu problēmvides atbalsta tīmekļa lietotnes piemēru un pārbaudīt tā atbilstību sagaidāmajam rezultātam.
6. Novērtēt un definēt secinājumus par pētījuma rezultātiem.

PĒTĪJUMU METODES

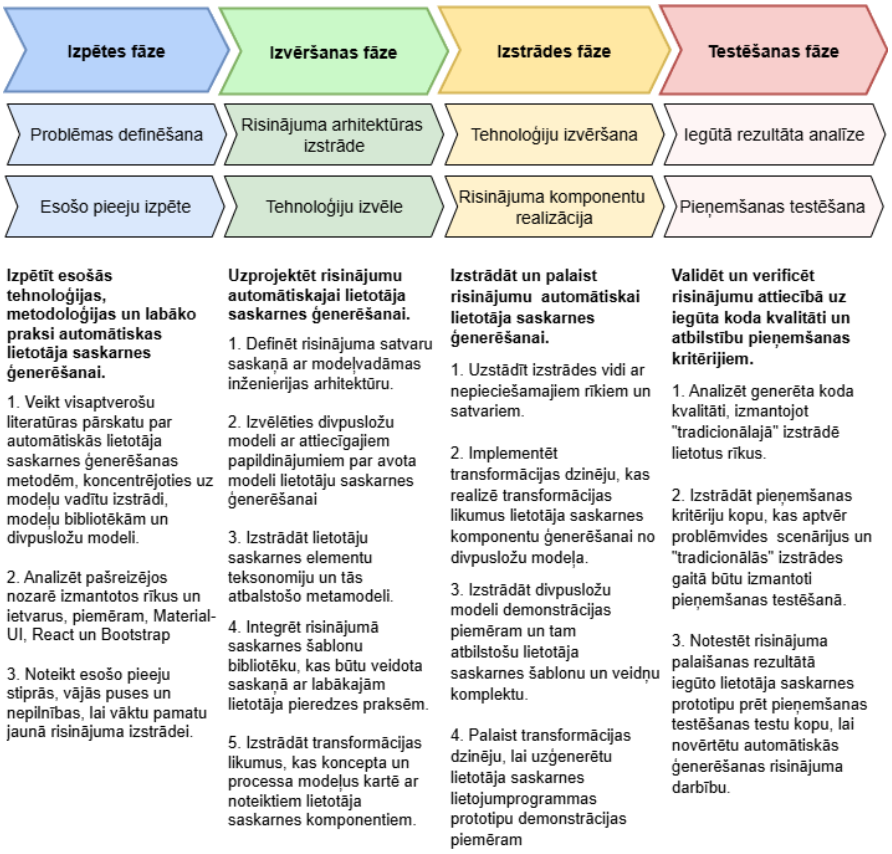
Uzdevumu izpildē kā pētījuma metodes promocijas darba izstrādes gaitā tiek izmantotas zinātniskās literatūras teorētiskā izpēte un saistīto pētījumu analīze, kas ietver esošo zinātnisko rakstu, grāmatu, monogrāfiju un citu avotu apkopošanu un analīzi, lai gūtu padziļinātas zināšanas par pētījuma objektu un esošajiem risinājumiem.

Promocijas darba izstrādes gaitā veiktā pētījuma metodoloģija ir definēta saskaņā ar programmatūras izstrādes labākās prakses fāzēm, kas ir izpēte, izvēršana, izstrāde un testēšana un detalizēti ir parādīta 1. attēlā (*Kruchten P.*, 2003). Izpētes fāze ietver iepriekš publicētu materiālu, piemēram, zinātnisko rakstu, grāmatu, monogrāfiju u. c., apkopošanu un izpēti, lai iegūtu vispusīgākas zināšanas par pētniecības objektu un pieejamajiem risinājumiem. Izvēršanas fāzes rezultātā ir piedāvāts risinājumu balstīt modeļvadāmas programmatūras inženierijas pamatkoncepcijās – modeļos, metamodeļos un modeļu transformācijās. Par avota modeli transformāciju definēšanai ir izmantots divpusložu modelis, un mērķa modelis ir veidots, pamatojoties uz darba autora veidoto tīmekļa lietotnes lietotāja saskarnes elementu taksonomiju. Transformācijas likumu izpildes gaitā ir izmantotas lietotāju saskarnes šablonu bibliotēkas veidnes un tajās definētais pirmkods, kas notiek saskaņā ar "*Model-ToText*" transformācijas nosacījumiem objektu vadības grupas definētajā kontekstā. Pēc būtības atbilstošo lietotāja saskarnes elementu ģenerēšana no divpusložu modeļa seko grafu transformācijas teorijai, kur gan divpusložu modelis, gan tā transformācijas rezultāts – lietotāja saskarnes prototips – var tikt apskatīti kā grafi ar attiecīgajiem elementiem (virsotnēm) un pārejām starp tiem (šķautnēm).

Piedāvātā risinājuma izstrādē ir izmantotas sistēmu analīzes, informācijas analīzes un sintēzes, sistēmu modelēšanas, grafu teorijā sakņotās modeļu transformācijas, noteiktas situācijas izpētes (angļu val. *Case study*) metodes un paņēmieni. No praktiskā skatpunkta risinājums tiek izstrādāts, izmantojot prototipēšanas metodi, un tas ietver tādas aktivitātes kā problēmas un iespējamo risinājumu alternatīvu analīze, eksperimentu veikšana un to rezultātu analīze. Rezultātā izstrādes fāzē ir implementēts iepriekš definēto transformācijas likumu dzinējs, kas ir aprobēts, izmantojot nelielu problēmvides fragmentu.

Risinājuma validācija ir veikta divos aspektos. Pirmkārt, modeļu transformācijas rezultātā iegūta koda kvalitātes analīze ir veikta ar *JSLint*, *JSHint* un *ESLint* palīdzību, kas dod iespēju pārliecināties, ka iegūtais rezultāts ir atbilstošs programmēšanas praksēm un vadlīnijām.

Otrkārt, ir veikta modeļu transformācijas rezultātā iegūta tīmekļa prototipa testēšana, lietojot standarta (*ISO/IEC/IEEE 29119*) pieņemšanas testēšanas metodi. Šīs validācijas metodes ietvaros ir izstrādāta potenciālās tīmekļa vietnes pieņemšanas (angļu val. *acceptance*) kritēriju kopa, pret ko būtu validēta "manuāli" izstrādāta tīmekļa vietne "tradicionālas" izstrādes procesā. Palaižot šo testa kopu modeļu transformācijas rezultātā iegūtam prototipam un pārbaudot, ka automātiski iegūti tīmekļa priekšgala (angļu val. *front-end*) komponenti atbilst pieņemšanas kritērijiem, kam atbilstu arī "manuālas" izstrādes rezultāts, ir iespējams pārliecināties, kas ar piedāvāto automatizāciju ir panākts izstrādes ātrums, nezaudējot rezultāta kvalitāti, jo tie paši kritēriji arī ir atbalstīti.



1. att. Promocijas darba izstrādes gaitā veiktā pētījuma detalizācija.

ZINĀTNISKĀ NOVITĀTE

1. Veikta divpusložu modeļa analīze un vērtētas iespējas izgūt lietotāja saskarnes prototipus no tā. Veikta divpusložu modeļa priekšrocību un trūkumu analīze lietotāja saskarnes elementu ģenerēšanas kontekstā un piedāvāts divpusložu modeļa uzlabojums, lai būtu iespējams ģenerēt transformācijas likumu mērķa elementus, kas ir definēti lietotāja saskarnes taksonomijas veidā.
2. Izstrādāts risinājums un atbilstoša transformācijas likumu kopa, kas ļauj ģenerēt lietotāja saskarnes prototipus no divpusložu modeļa. Promocijas darba gaitā izstrādātais risinājums ir aprobēta *JavaScript* bibliotēkas *React.js* prototipu ģenerēšanai *Material Design* satvara komponentos.
3. Definēti formāti avota (divpusložu) un mērķa (lietotāja saskarnes prototipa) modeļu uzglabāšanai un apstrādei, kā arī abu modeļu metamodeļi, ko turpmāk var izmantot citos pētījumos, kur būs nepieciešamība šos modeļu elementus lietot avota vai mērķa līmenī.
4. Izstrādātas rekomendācijas Rīgas Tehniskajā universitātē radītā *BrainTool* rīka uzlabošanai un turpmākai attīstībai.

PĒTĪJUMA PRAKTISKĀ NOZĪME

Promocijas darba praktiskā nozīmība ir izstrādātais risinājums realizācijas potenciālā ģenerēt lietotāja saskarnes prototipus no divpusložu modeļiem. Darbā ir definēta visa nepieciešama avota un mērķa modeļa specifikācija un transformācijas likumi, kas turpmāk ir lietojami piedāvāta risinājuma atbalsta prototipa realizācijā. Darbā piedāvātais risinājums var būt lietderīgs programmatūras izstrādātāju grupām, kas darbojas ar dažāda veida spraudņu/programmu izstrādi, kas atbalsta programmatūras izstrādes procesu, piemēram, integrētu izstrādes vižu, lietotāja saskarnes izstrādes rīku un sistēmu modelēšanas rīku izstrādātājiem. Kā arī izstrādāta risinājuma demonstrācija divpusložu modeļa veidošanai izmantojot *BrainTool* rīku, kas šajā promocijas darbā ir papildināts ar lietotāja saskarnes ģenerēšanu, parādītā rīka potenciālu turpmākai tā attīstībai un, iespējams, komercializācijai.

DARBA AUTORA PUBLIKĀCIJAS

Pētījuma rezultātu aprobācija ir atspoguļota astoņās publikācijās starptautiskos un Latvijas Zinātnes padomes atzītos zinātniskos izdevumos.

1. Babris K., Ņikiforova O., Sukovskis U. Brief Overview of Modelling Methods, Life-Cycle and Application Domains of Cyber-Physical Systems. Applied Computer Systems, 2019, Vol. 24, No. 1, pp. 1–8, DOI: 10.2478/acss-2019-0001 (*Web of Science)*.
    - Autora personīgais ieguldījums: saistīto darbu analīze, salīdzināšanas kritēriju izstrāde un vērtējumu apkopojuma veikšana.
2. Ņikiforova O., Babris K., Kristapsons J. Survey on Risk Classification in Agile Software Development Projects in Latvia. Applied Computer Systems, 2020, Vol. 25, No. 2, pp. 105–116. DOI: 10.2478/acss-2020-0012 (*Web of Science)*.
    - Autora personīgais ieguldījums: saistīto darbu analīze, riska veidu identificēšana, secinājumu izstrāde.

3. Ņikiforova, O., Babris K., Madelāne L. Expert Survey on Current Trends in Agile, Disciplined and Hybrid Practices for Software Development, Applied Computer Systems, Vol. 26, No. 1, 2021, pp. 38–43. DOI: 10.2478/acss-2021-0005 (*Web of Science)*.
    - Autora personīgais ieguldījums: saistīto darbu analīze, programmatūras izstrādes procesa prasību un īpašību definēšana, secinājumu izstrāde.
4. Ņikiforova O., Zabiniako V., Kornienko J., Rizhko R., Babris K., Gasparoviča-Asīte M. Efficiency Monitoring of Engineering System Designer Work Based on Multi-System User Behavior Analysis with AI/ML Algorithms, IEEE 62nd International Scientific Conference on Power and Electrical Engineering of Riga Technical University (RTUCON), 2021, pp. 1–6, DOI: 10.1109/RTUCON53541.2021.9711720 *(Scopus)*.
    - Autora personīgais ieguldījums: eksperimentu veikšana, sistēmas modeļa sākotnējas versijas validēšana.
5. Ņikiforova O., Zabiniako V., Kornienko J., Rizhko R., Babris K., Nikulsins V., Garkalns P., Gasparoviča-Asīte M. Solution to On-line vs On-site Work Efficiency Analysis on the Example of Engineering System Designer Work, Applied Computer Systems, Vol. 26, No. 2, 2021, pp. 87–95, DOI: 10.2478/acss-2021-0011 *(Scopus)*.
    - Autora personīgais ieguldījums: eksperimentu veikšana, sistēmas modeļa gala versijas validēšana.
6. Nikiforova O., Babris K., Mahmoudifar F. Automated Generation of Web Application Front-End Components from User Interface Mockups, Proceedings of International Conference on Software Technologies, SCITEPRESS Digital Library, 2024, pp. 100–111, DOI: 10.5220/0012759500003753 *(Scopus)*.
    - Autora personīgais ieguldījums: saistīto darbu analīze, avota un mērķa metamodeļa izstrāde.
7. Ņikiforova O., Babris K., Guliyeva A. Definition of a Set of Use Case Patterns for Application Systems: A Prototype-Supported Development Approach, Applied Computer Systems, Applied Computer Systems, Vol. 29, No. 1, 2024, pp. 59–67, DOI: 10.2478/acss-2024-0008 (*Web of Science)*.
    - Autora personīgais ieguldījums: praktiskā piemēra izstrāde, eksperimentu veikšana, lietošanas gadījumu šablonu apkopojums un strukturēšana.
8. Babris K., Nikiforova O. Towards Automated UI Mockup Generation from Two-Hemisphere Problem Domain Models: A Conceptual Framework and Approach, Proceedings of 19th Iberian Conference on Information Systems and Technologies, 2024, pp. 1–6, pieņemts publicēšanai, tiks indeksēts *Scopus*.
    - Autora personīgais ieguldījums: saistīto darbu analīze, avota un mērķa metamodeļa izstrāde, risinājuma koncepcijas izstrāde.
9. Babris K., Nikiforova O. From Models to Interfaces: Leveraging the Two-Hemisphere Model for Automated UI Generation, IEEE 65th International Scientific Conference on Information Technology and Management Science of Riga Technical University (ITMS), Riga, Latvia, 2024, pp. 1–6, DOI: 10.1109/ITMS64072.2024.10741944 (*Scopus*).

- Autora personīgais ieguldījums: saistīto darbu analīze, risinājuma un aprobācijas piemēra izstrāde.

Promocijas darba galvenie rezultāti prezentēti trīs starptautiskās zinātniskās konferencēs.

1. CISTI'2024 – 19th Iberian Conference on Information Systems and Technologies, – 2024, 25.–28.06.2024. – Salamanka, Spānija, "Towards Automated UI Mockup Generation from Two-Hemisphere Problem Domain Models: A Conceptual Framework and Approach".
2. ICSOFT 2024 – 19th International Conference on Software Technologies, 08.–10.07.2024. – Dižona, Francija, "Automated Generation of Web Application Front-end Components from User Interface Mockups".
3. ITMS'2024 – The 65th International Scientific Conference on Information Technology and Management Science of Riga Technical University, 03.–04.10.2024. – Rīga, Latvija, "From Models to Interfaces: Leveraging the Two-Hemisphere Model for Automated UI Generation".

AIZSTĀVĒŠANAI IZVIRZĪTĀS TĒZES

1. Izmantojot RTU izstrādāto divpusložu modeli, ir iespējams ģenerēt tīmekļa lietotnes lietotāja saskarnes prototipus, izmantojot modeļvadāmas izstrādes pamatkoncepcijas, kas ir modeļi un modeļu transformācijas.
2. Tīmekļa lietotnes lietotāja saskarnes prototipa automātiskā iegūšana no biznesa līmeņa modeļiem paātrina tīmekļa lietotnes izstrādes procesu, nezaudējot rezultāta kvalitāti.

PROMOCIJAS DARBA STRUKTŪRA

Promocijas darba pirmajā nodaļā apskatīti lietotāja saskarnes veidi un brieduma līmeņi pēc analoģijas ar spēju brieduma līmeni, novirzīts fokuss uz promocijas darba pētījuma priekšmetu, proti, lietotāja saskarnes prototipu, kā arī ieskicēts lietotāja saskarnes prototipa izstrādes process dažādos abstrakcijas līmeņos. Šajā nodaļā sniegts pilnīgs kopsavilkums par rīkiem un tehnoloģijām, kas veido pašreizējo lietotāja saskarnes attīstību, kā arī uzsvērta viegli saprotamas un lietojamas saskarnes izveides nepieciešamība. Par formālismu lietotāja saskarnes prototipa ģenerēšanas risinājuma izstrādē izvēlēti modeļvadāmas inženierijas pamatprincipi, kas ir avota un mērķa modeļi un to metamodeļi, kas aprakstīti promocijas darba otrajā nodaļā. Otrajā nodaļā paskaidrota arī modeļvadāmās izstrādes būtība, kā, veidojot abstraktus modeļus, ir iespējams pārveidot modeli par citu modeli vai pirmkodu. Risinājumā izmantotie un otrajā nodaļā aprakstītie metamodeļi ir pamats modeļu transformācijas likumu izveidei, kas definēti trešajā nodaļā. Nodaļā izskaidrots metodiskais veids, kā izveidot šos transformācijas likumus, lai garantētu, ka tie ir precīzi, elastīgi un pielāgojami dažādām lietotāja saskarnes veidošanas vajadzībām. Promocijas darba ceturtā nodaļa veltīta izstrādātā risinājuma aprobācijai un novērtēšanai, kas balstās sagaidāmajā rezultātā, t. i., lietotāja saskarnes prototipa izstrādē noteiktai problēmas videi un tā salīdzināšanā ar risinājuma lietošanas ietvaros izveidoto prototipu. Darba nobeigumā definēti secinājumi par iegūtajiem rezultātiem un nākotnes pētījuma virzieni.

1. LIETOTĀJA SASKARNES IZSTRĀDE

Šajā nodaļā tiek apskatīts lietotāja saskarnes (angļu val. *User Interface, UI*) izstrādes process, ietverot tā elementus un svarīgākās nianses. Lietotāja saskarne ir vide, kurā lietotājs mijiedarbojas ar sistēmu, saņemot atgriezenisko saiti saprotamā veidā. Šī promocijas darba kontekstā lietotāja saskarne ir domāta kā tīmekļa lietotnes lietotāja saskarne. Tā apvieno vizuālo dizainu, mijiedarbības dizainu un informācijas arhitektūru un ir lietojumprogrammas priekšgala komponents (*Nguyen, Vu et al.*, 2018; *Alfaridzi & Yulianti*, 2020).

Lietotāja saskarnes izstrāde ietver vairākus posmus no skices līdz augstas precizitātes prototipam, un šajā procesā tiek iesaistīti dažādi speciālisti. Šis process ir darbietilpīgs un ietver daudz manuālu un laikietilpīgu darbību, lai sasniegtu galaproduktu (*Bajammal, Davood et al.*, 2018). Prototipēšanas procesā koncepcijas tiek pārveidotas no melnrakstiem par lietojamiem prototipiem. Tas ir atkārtojams un laika ziņā ietilpīgs process, kas prasa vairākas pārstrādes, līdz tiek sasniegta nepieciešamā brieduma pakāpe (*Suleri, Pandian et al.*, 2019).

Lietotāja saskarnes izstrādē tiek izmantoti dažādi precizitātes līmeņi, kas atspoguļo projektēšanas procesa dažādus posmus un detalizācijas pakāpes (*Stompff & Smulders*, 2015). Sākot no skicēm, kas ir zemas precizitātes zīmējumi, līdz struktūrskicēm, kas nodrošina saskarnes strukturālu izklāstu bez detalizēta vizuālā stila, dizaina process ietver vairākus posmus (*Rivero, Rossi et al.*, 2011). Maketi piedāvā augstas precizitātes statiskus vizuālos attēlojumus ar detalizētu stilu un krāsām, savukārt prototipi nodrošina interaktīvu lietotāja saskarnes simulāciju, piedāvājot reālistisku lietotāja pieredzi (*Rudd, Stern et al.*, 1996; *Virzi, Sokolov et al.*, 1996). Galalietotāja saskarne ir pilnībā izstrādāta un ieviešanai gatava versija ar visaugstāko precizitātes līmeni (*Rudd, Stern et al.*, 1996). Katrs šis līmenis palīdz dizaineriem izstrādāt, uzlabot un pārbaudīt dizaina koncepcijas dažādos posmos, sākot no sākotnējās idejas līdz galīgajai lietotāja saskarnes ieviešanai. Skices un struktūrskices sniedz pamata vizuālo attēlojumu un funkcionalitātes plānu, savukārt maketi un prototipi piedāvā detalizētāku un reālistiskāku attēlojumu, palīdzot vizualizēt un novērtēt galadizainu pirms tā ieviešanas (*Riaz, Arshad et al.*, 2022; *Silva (da), Martin et al.*, 2011; *Newmanm & Landay*, 2000). Prototipu veidošanā var izmantot izpētes, eksperimentālo vai evolucionāro pieeju, kas katra atšķiras ar precizitāti un mērķiem un piedāvā atšķirīgu detalizācijas līmeni un orientāciju (*Nacheva*, 2017).

Lietotāja saskarnes izstrādei ir trīs galvenās pieejas – manuālā, pusautomātiskā un automātiskā, katrai no tām ir savas priekšrocības un trūkumi (*Molina, Meliá et al.*, 2002). Manuālā izstrāde, lai arī ļoti elastīga un pielāgojama, prasa ievērojamu laiku un pūles, kā arī tai ir augsta kļūdu iespējamība (*Molina, Meliá et al.*, 2002). Pusautomātiskā izstrāde izmanto rīkus, kas balstīti dizaina modeļos, lai atvieglotu procesu un samazinātu izstrādes laiku, tomēr šāda pieeja var radīt grūtības uzturēšanā un atjaunināšanā, kad nepieciešamas izmaiņas dizainā vai rodas jaunas tehnoloģijas (*Pelechano, Pastor et al.*, 2002). Automātiskā izstrāde nodrošina ātrus rezultātus un var ievērojami samazināt projekta izmaksas, jo tiek izmantota kodu ģenerēšana, tomēr šī pieeja nav piemērota visām sistēmām, īpaši tām, kurām nepieciešama specifiska optimizācija vai nav izstrādātas šabloniskas pieejas (*Sunitha & Samuel*, 2019; *Pastor, Abrahao et al.*, 2001). Kopumā katra pieeja ir piemērota dažādām situācijām, un izvēle

starp tām ir atkarīga no projekta prasībām, dizainera pieredzes un konkrētās problēmas specifikas. Šī promocijas darba fokusā ir lietotāja saskarnes automātiskā izstrāde.

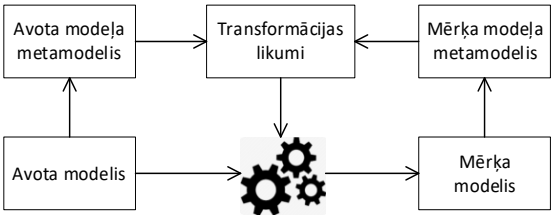
Lietotāja saskarnes prototipu ģenerēšanas risinājumi ietver dažādus rīkus un pieejas, kas palīdz automatizēt dizaina procesu, lai samazinātu tā ietekmi uz izstrādi (*Nikiforova, Babris et al.*, 2020). Lai gan ir daudz rīku, kas atbalsta dažādus programmatūras prototipu veidošanas aspektus, trūkst vienotas sistēmas, kas aptvertu visu procesu no sākuma līdz beigām (*Suleri, Pandian et al.*, 2019). Kā arī ir nepieciešami labi definēti, nobrieduši un pielāgojami lietotāja saskarnes izstrādes procesi, kas atbilst organizācijas mērķiem (*Gilbert, Fischer et al.*, 2021).

Spēju brieduma modeļa integrācijas (angļu val. *Capability Maturity Model Integration, CMMI*), modeļa brieduma indeksa (angļu val. *Model Maturity Index, MMI*) un lietotāja saskarnes brieduma līmeņi sniedz veidus, kā izmērīt un uzlabot procesus, modeļus un saskarnes. *CMMI* līmeņi liecina par procesu briedumu organizācijā, sākot no sākotnējā līdz optimizētam līmenim (*Hinderks, Mayo et al.*, 2022). *MMI* līmeņi parāda modeļu attīstību projektā, savukārt lietotāja saskarnes brieduma līmeņi atspoguļo saskarnes dizaina progresu no skicēm līdz pilnveidotām un interaktīvām saskarnēm. Lai gan katrs no šiem elementiem aptver savu jomu, kopumā tie visi virzās uz lielāku automatizāciju un optimizāciju, kas palīdz uzlabot galaprodukta kvalitāti un lietojamību.

Esošo risinājumu analīze lietotāja saskarnes projektēšanā ļauj secināt, ka eksistējošie pētījumi skar tikai atsevišķus lietotāja saskarnes aspektus un dod iespēju daļēji ģenerēt lietotāja saskarnes prototipus (*Nikiforova, Babris et al.*, 2024a). Lai gan lietotāja saskarnes ģenerēšana ir ievērojami attīstījusies, pašreizējā stāvoklī automatizācijas rīkiem ir ierobežojumi darbā ar sarežģītām jaunāko saskarņu vajadzībām, piemēram, adaptīvu dizainu, vairāku platformu piemērotību un mainīgu saturu. Bieži dizaineriem ir jāiejaucas manuāli, lai risinātu ārkārtas situācijas un apstiprinātu, ka izveidotās saskarnes atbilst noteiktām lietotāja prasībām un lietojumprogrammu scenārijiem (*Diehl, Martins et al.*, 2022). Joprojām ir sarežģīti pārliecināties, ka automātiski izveidotās lietotāja saskarnes ir labas kvalitātes un tās var pareizi izmantot. Lai gan daudziem rīkiem ir iebūvētas validācijas pārbaudes un lietojamības noteikumi, joprojām ir vajadzīgas spēcīgas testēšanas un novērtēšanas metodes, lai uzzinātu par problēmām, kas saistītas ar lietošanas vienkāršību, pieejamības šķēršļiem, kā arī dizaina atšķirībām. Lietotāja saskarnes ģenerēšanas metodes, kurās tiek izmantots mākslīgais intelekts, bieži vien nepiedāvā caurspīdīgumu vai interpretējamību (*Alfaridzi & Yulianti*, 2020; *Riccio, Jahangirova et al.*, 2020). Uzticēšanās pieejai un spēja izprast ģenerēšanas algoritmu rezultātus var būt šķērslis lietotāja saskarnes dizaineriem. Mākslīgā intelekta vadīta lietotāja saskarnes ģenerēšana kļūst arvien izplatītāka (*Stige, Zamani et al.*, 2023), tāpēc pieaug arī bažas par ētiku, piemēram, objektivitātes trūkumu algoritmos, rezultāta precizitāti, privātuma riskiem un iespējamo projektētāju nomaiņu. Dizaineriem un izstrādātājiem ir rūpīgi jādomā par automatizētas lietotāja saskarnes izveides sekām, kā arī jāatrod veidi, kā novērst jebkādu radīto kaitējumu. Kad šie šķēršļi būs pārvarēti, automātiskā lietotāja saskarnes ģenerēšana var būtiski ietekmēt to, kā tiek izstrādāta programmatūra, jo dizaineriem būs vieglāk izveidot lietotāja saskarnes, kas ir efektīvākas un uz cilvēku centrētas.

2. MODEĻVADĀMAS IZSTRĀDES ELEMENTI PIEDĀVĀTAJĀ RISINĀJUMĀ

Analizējot saistītos pētījumus, kur ir piedāvāti dažādi risinājumi, kā var automatizēt lietotāja saskarnes projektēšanu, ir secināts, ka modeļi un metamodeļi ir pietiekams formālisms lietotāja saskarnes projektēšanā, jo lietotāja saskarnes elementu kopu var aprakstīt kā modeli (*Escalona, García-Borgoñón et al.*, 2021), līdz ar to – lietot modeļvadāmas izstrādes principus, kur pamatkoncepti ir modeļi, metamodeļi un transformācijas likumi (2.1. att.).



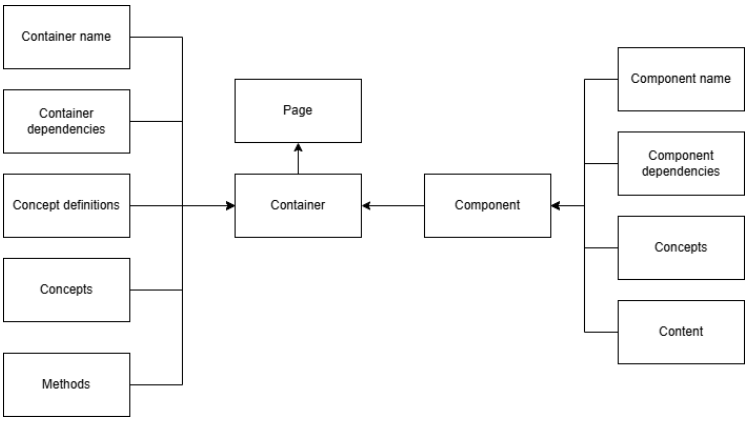
2.1. att. Modeļvadamās izstrādes konceptuālā shēma (adaptēts no (*Kleppe, Warmer et al.*, 2003)).

Modeļvadāmas izstrādes pamatā ir programmatūras koda automātiskā ģenerēšana no problēmvides modeļiem, kur pamatā ir problēmvides analīze un modelēšana, meta modelēšana, modeļvadāmā koda ģenerēšana, šablona valodas un problēmvidē balstītu ietvaru izstrāde (*Völter & Bettin*, 2004). Lietotāja saskarnes projektēšanas kontekstā ir izmantojami četri modeļvadāmās izstrādes principi.

1. Tiek veidots modelis vai modeļi, kas apraksta lietotāja saskarnes scenārijus un ietver pilnīgu un nepretrunīgu informāciju par problēmvides zināšanām un kontekstu, tā saucamo avota modeli.
2. Tiek savākta visa nepieciešamā informācija par lietotāja saskarnes formām, to saturu un pārejām starp tām, kas ir attēlota mērķa modelī.
3. Avota un mērķa modeļi dod iespēju definēt avota un mērķa metamodeļus, kas tiek izteikti, lietojot *UML* vai kādu problēmvides specifisko valodu.
4. Tiek definēti transformācijas likumi, kas dod iespēju iegūt mērķa modeli, izmantojot avota modelim iepriekš definētus transformācijas likumus (*Babris, Nikiforova et al.*, 2019).

Transformācijas rezultātu var izmantot tādā veidā, kā tas ir iegūts pēc transformācijas likumu lietošanas vai manuāli papildinātu. Lai transformācijas rezultāts būtu lietojams, tam ir jābūt palaižamam kodam vai importējamam kādā no vidēm, kurā ir iespējams transformācijas rezultātu apstrādāt. Šajā nodaļā aprakstīti lietotāja saskarnes elementi, lai būtu iespējams veidot mērķa modeli un mērķa metamodeli transformācijas likumu definēšanai, kā arī tiek aprakstīts divpusložu modelis un tā adaptācija transformācijas likumu definēšanai, lai rezultātā var iegūt lietotāja saskarnes prototipus.

2.2. attēlā redzams lietotāja saskarnes metamodelis *React.js* / *Redux.js* tīmekļa lietotnes organizēšanai automātiskas lietotāja saskarnes izveidei. Metamodelis koncentrējas uz to, kā lietotāja saskarnes konteineri un komponenti ir saistīti un atkarīgi viens no otra. Šis metamodelis parāda, kādi elementi ir sagaidāmi risinājuma rezultātā, lai automātiski izveidotu *React.js* / *Redux.js* tīmekļa lietotni, nodalot funkcionalitāti konteineros un komponentos, kā arī *Redux.js* iekļaušana datu pārvaldīšanai un *XState* procesu organizēšanai. Šāda pieeja piedāvā elastīgu un mērogojamu struktūru lietotāja saskarnes automātiskai ģenerēšanai.



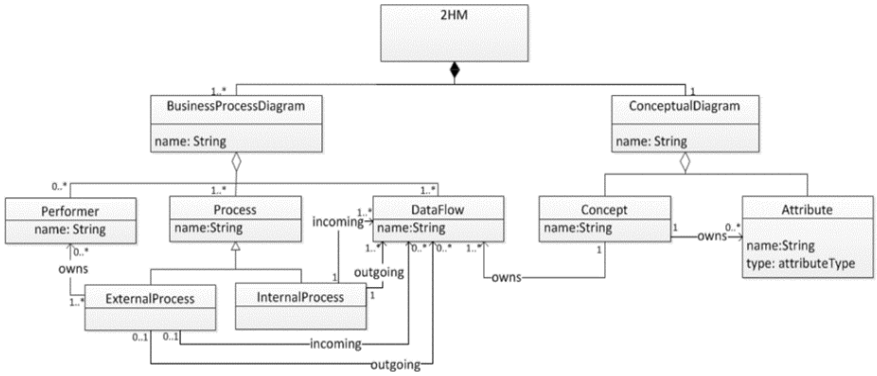
2.2. att. Lietotāja saskarnes metamodelis izstrādājamā risinājuma ietvaros.

Metamodelī tīmekļa lietotne tiek sadalīta lapās (angļu val. *Pages*), konteineros un komponentos. Lapas sastāv no konteineriem, kas apstrādā esošo procesa stāvokli, definē konceptus un metodes. Komponenti apzīmē atsevišķus lietotāja saskarnes elementus, kas saņem datus un metodes no saviem konteineriem, izmantojot īpašības (angļu val. *Properties*). Konteiners ir definēts ar nosaukumu, atkarību sarakstu, koncepta definīcijas sadaļu, kurā ir arī mainīgie un metodes. Tādā pašā veidā tiek definēti komponenti, kuriem ir nosaukumi, atkarības, koncepti un pats galvenais – saturs, kas ietver izskatu un darbību. Savienojumi starp lapām, konteineriem un komponentiem nodrošina to, ka viss ir iestatīts modulāri, lai atvieglotu sistēmas uzturēšanu. Šis metamodelis palīdz pārveidot biznesa procesu modeļus funkcionālās lietotāja saskarnēs, kartējot entītijas ar *React.js* komponentiem. Šī strukturētā sistēma palīdz viegli izveidot dinamiskas lietotāja saskarnes ar stāvokļiem, saglabājot skaidru pienākumu sadali un palielinot elastību.

Divpusložu modeļvadāma pieeja (*Nikiforova*, 2009) ir viena no modeļvadāmas programmatūras izstrādes pieejām, kas tiek izstrādāta un attīstīta Rīgas Tehniskajā universitātē un pirmo reizi bija publicēta 2004. gadā (*Nikiforova & Kirikova*, 2004). Atšķirībā no citām modeļvadāmām pieejām, kas bija attīstītas tajā laikā (*Nikiforova, Kozacenko et al.*, 2015), divpusložu modeļvadāmā pieeja piedāvāja balstīt programmatūras izstrādi divos savstarpēji saistītos modeļos, proti, procesu modelī, kas aprakstīja problēmvides funkcionēšanu, un konceptu modelī, kas attēloja problēmvidē esošā datu struktūras. Un tieši sasaiste starp šiem diviem modeļiem nodrošināja automātisku problēmu analīzes informāciju transformēt

programmatūras sistēmas projektēšanas modeļos, kas bija veidoti dažādu *UML* diagrammu veidā. Šī transformācija tādējādi piedāvāja plašāku projektēšanas artefaktu kopumu turpmākai koda ģenerēšanai uzreiz no problēmvides modeļiem. Divpusložu modeļvadāma pieeja bija aprobēta vairākās problēmvidēs (*Nikiforova, Sukovskis et al.*, 2015; *Nikiforova, el Marzouki et al.*, 2017; *Nikiforova & Gusarovs*, 2020; *Nikiforova, Iacono et al.*, 2020) un lietota vairākos promocijas darbos kā transformāciju pamatojuma formālisms (*Pavlova*, 2008; *Gusarovs*, 2020; *Marzuoki (el)*, 2021).

Divpusložu modeļa metamodelis redzams 2.3. attēlā. Ir redzams, ka divpusložu modelis pēc būtības ietver konceptuālu diagrammu un vienu vai vairākas biznesa procesu diagrammas. Šo divu modeļu sasaiste ir attēlota, saistot datu struktūru no konceptuālās diagrammas ar datu plūsmu elementu procesa diagrammā (*Nikiforova*, 2002). Katra datu plūsma ir precīzi saistīta ar vienu konceptu, lai gan katrs koncepts var būt saistīts ar vairākām procesu plūsmām. Šī saikne veido pamatu atbildības noteikšanai objektu klasēm turpmāko transformāciju laikā.



2.3. att. Divpusložu modeļa metamodelis (aizgūts no (*Kozačenko*, 2014)).

Darbā ir veikts lietotāja saskarnes elementu apkopojums, kas ļauj identificēt saskarnes prototipam nepieciešamu elementu kopu kā mērķa kopu. Turklāt iepriekšējo pētījumu esamība par lietotāja saskarnes metamodeļa izstrādi un lietojumu transformāciju definēšanai dažādos abstrakcijas līmeņos dod papildu pārliecību par šajā promocijas darbā izvēlētā problēmas risināšanas ceļa korektumu. Esošie metamodeļi bija pamats izveidot vienotu lietotāja saskarnes metamodeli, ko var izmantot ne tikai šī promocijas darba gaitā izstrādātajā risinājumā, bet to var turpmāk izmantot arī citi pētnieki (*Babris & Nikiforova*, 2024a).

Divpusložu modeļa praktiskās lietošanas ietvaros iepriekšējos pētījumos izstrādāts metamodelis jau ietver pilnīgu elementu kopu, kas potenciāli ir izmantojama transformāciju definēšanā lietotāja saskarnes elementu ģenerēšanai. Lai būtu iespējams izmantot divpusložu modeļa lietošanas formālismus saskarnes prototipa automātiskajā ģenerēšanā, nākamajā nodaļā aprakstīta transformācijas likumu veidošana sagaidāmajos mērķa modeļa elementos saskaņā ar tā metamodeli.

3. TRANSFORMĀCIJAS LIKUMU DEFINĒŠANA

Modeļvadāmas izstrādes lietojumprogrammu programmatūras inženierijas izaicinājumi iesākās 21.gadsimta sākumā ar ļoti pievilcīgu ideju izveidot no platformas neatkarīgu modeli programmatūras izstrādei, un tā transformēšana no neatkarīgas platformas par platformas specifisku modeli vēl vairāk veicināja izmantot automātisku koda ģenerēšanu (*Suleri, Pandian et al.*, 2019; *Stahl, Völter et al.*, 2006). Ideja bija daudzsološa, taču dažādu iemeslu dēļ utopiska (*Hailpern & Tarr*, 2006; *Thomas*, 2004). Tomēr tajā piedāvātais formālisms (*Trehan, Chapman et al.*, 2015) ir visai noderīgs uzdevumos, kad pilnīgu un konsekventu modeli, lietojot metamodelēšanas principus, var pārveidot par tā zināšanām, pasniegtajām citā formātā (*Nikiforova & Gusarovs*, 2020; *Domingo, Echeverría et al.*, 2020).

Iepriekšējā nodaļā aprakstīts izveidotais mērķa modeļa metamodelis, saskaņā ar kuru lietotāja saskarnes prototipā sagaidāmie komponenti ir iedalīti trīs grupās (*Leuthold*, 2010; *Koch & Mandel*, 1999):

1) koncepta elementi;
2) navigācijas elementi;
3) prezentācijas elementi.

Par avota modeli ir izvēlēts divpusložu modelis, kas iepriekš jau tika lietots projektēšanas un programmatūras komponentu automātiskajai ģenerēšanai.

Šajā nodaļā definēta transformācijas likumu kopa avota modeļa pārveidošanai par mērķa modeli saskaņā ar mērķa metamodelī definētām elementu grupām. Turklāt attiecībā uz avota modelī trūkstošiem elementiem ir pētītas citu formālismu lietošanas iespējas, lai noklātu pilnīgu mērķa modeļa elementu iegūšanu.

3.1. Koncepts

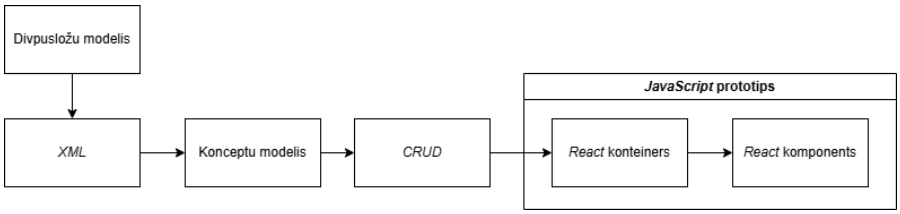
Konceptuālās projektēšanas soļa aktivitātes ir klašu noteikšana, atribūtu un operāciju precizēšana, hierarhisko struktūru definēšana un apakšsistēmu noteikšana. Lai sasniegtu šo mērķi, tiek izmantotas labi zināmās objektorientētās modelēšanas metodes.

Šajā darbībā definētās klases un asociācijas tiek izmantotas arī navigācijas projektēšanas laikā, lai iegūtu tīmekļa lietotnes struktūras saites. Attiecības tiks izmantotas, lai iegūtu saites. Klases un asociācijas var organizēt grupās. Klases ir aprakstītas, izmantojot atribūtus un darbības, un attēlotas grafiski (*Koch & Mandel*, 1999). Konceptuālās izstrādes soļa rezultāts ir apkopots konceptuālā modelī, kas ietver klases un asociācijas starp klasēm, kas modelē problēmas jomu. Šajā darbā koncepts kā pamats tiek izmantots no divpusložu modeļa ģenerētais koncepta modelis, kas ļauj atrast sakarību starp dažādiem datu modeļiem.

Faktiski šī informācija ļauj izveidot ko līdzīgu *ER* diagrammai, no kuras var ģenerēt lietotāja saskarnes elementus. Koncepta atribūtu kartēšana uz lietotāja saskarnes elementiem nodrošina, ka saistītā informācija tiek parādīta un ar to var pareizi mijiedarboties lietotāja saskarnē. Ne visai informācijai no koncepta modeļa ir jābūt redzamai lietotājam, faktiski ir nepieciešams norādīt, kādi lauki ir pieejami lietotājam un kādiem ir jābūt paslēptiem. Turklāt šai lauku

atribūtu pievienošanai ir jābūt pārvaldāmai, jo, piemēram, rediģēšanas vai formas ievades laikā lauks var tikt slēpts, savukārt lasīšanas režīmā šim laukam un tā saturam jābūt redzamam.

Datu ievades un izvades identificēšanai var palīdzēt lietotāja saskarnes šabloni, kuros jau būtu zināšanas par to, kā ir jāattēlo. Neizmantojot lietotāja saskarnes šablonus, var izgūt lietotāja saskarnes prototipu tādā veidā, kā tas attēlots 3.1. attēlā.



3.1. att. Koncepta skatījuma izgūšana no divpusložu modeļa.

Pēc eksperimentiem ar konceptu tika noskaidrots, ka divpusložu modelī (*BrainTool* rīka ietvaros ir tikai astoņi tipi) ir ierobežots koncepta lauku tipu skaits, kas nozīmē, ka šobrīd nav iespējams izmantot pilnīgu lietotāja saskarnes elementu kartēšanu. Var secināt, ka divpusložu modelis ir jāpapildina ar citiem lauku tipiem. Pozitīvi ir tas, ka konceptā var diezgan viegli norādīt uz saistītajiem ierakstiem. Šādā gadījumā var noderēt lietotāja saskarnes šabloni, kuros jau būtu norādīti izkārtojuma un formu elementi un tie tikai būtu jāsavieno ar koncepta lauku tipiem un nosaukumiem. Faktiski, līdzīgi kā citā pētījumā (*Mahatody, Ilie et al.*, 2021), būtu nepieciešams papildināt koncepta lauku veidus ar, piemēram:

1) *Text*, lai noteiktu gara teksta ievades opciju;
2) *Enum*, kas palīdzētu definēt īsus izkrītošos sarakstus;
3) *Date* un *Datetime*, kas palīdzētu izveidot datumu un laika laukus (piemēram, no datubāzes).

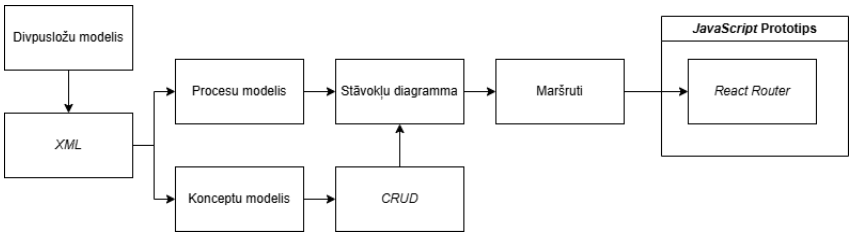
Iespējams, izmantojot lietotāja saskarnes šablonus ar jau definētu koncepta sasaisti ar lietotāja saskarnes elementiem, šī problēma būtu atrisināta. Piemēram, lauku slēpšanu un parādīšanu lasīšanas un rakstīšanas režīmā varētu atrisināt, izmantojot noteiktu šablonu katrā gadījumā. Darba turpinājumā tiek aprakstīta ar navigāciju saistītā informācija.

3.2. Navigācija

Navigācijas izstrādei tiek izmantota stāvokļa diagramma (*Sunitha & Samuel*, 2019), kas transformācijas ceļā tiek izgūta no divpusložu modeļa, līdzīgi kā citos pētījumos, piemēram, izgūstot stāvokļa diagrammas no prasību specifikācijas (*Pimentel, Castro et al.*, 2014; *Briand, Labiche et al.*, 2005). Lietotāja saskarnes navigāciju var attēlot kā stāvokļa diagrammu, pa kuru pārvietojas lietotājs, izmantojot navigāciju.

Stāvokļa diagrammas ir iespējams ģenerēt arī no *UML* secību diagrammām (angļu val. *Sequence Diagram*), izmantojot, piemēram, grafu transformācijas (*Grønmo & Møller-Pedersen*, 2011). No stāvokļa diagrammām ir piedāvājuši ģenerēt lietotāja saskarni arī citi autori (*Horrocks*, 1999; *Wellner*, 1989; *Harel*, 1987), izmantojot stāvokļus dažādās lietotāja

saskarnes izšķirtspējās – sākot no lietotāja saskarnes pamatelementu, piemēram, pogu darbības, beidzot ar kopējās lietotāja saskarnes navigācijas shēmām. Promocijas darbā stāvokļa diagrammas tiek izmantotas kā pamats kopējam navigācijas modelim, pa kuru var pārvietoties galalietotājs, neiedziļinoties katra lietotāja saskarnes elementā (3.2. att.).



3.2. att. No divpusložu modeļa izgūstamo stāvokļu diagrammu kopējā shēma.

Ņemot par pamatu konceptuālo modeli, balstoties navigācijas plānojumā, tiek veidots navigācijas modelis, kas ir skatījums uz konceptuālo modeli. Šo navigācijas modeli izgūst divos etapos (*Koch & Mandel*, 1999):

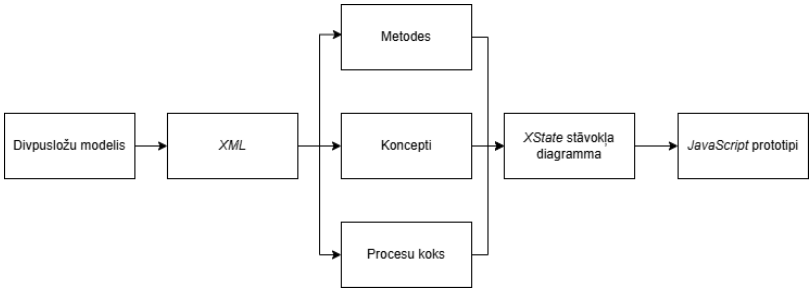
1) kādi objekti ir potenciāli sasniedzami, izmantojot navigāciju;
2) kā šie objekti tiek sasniegti.

Navigācijas modeļa izveidošanai ir nepieciešams izmantot zināšanas no citiem objektiem, lai veiktu objektu grupēšanu (*Koch & Mandel*, 1999). Modeļvadāmo pieeju mērķis ir automātiski ģenerēt kodu no modeļiem. Tādējādi katrā modeļvadāmā metodoloģijā darbs notiek ar modeļiem un transformācijām. Šiem modeļiem ir jāatbilst attiecīgajiem metamodeļiem. Savukārt transformācijas tiek aprakstītas, izmantojot transformācijas valodas (*Gharaat, Sharbaf et al.*, 2021).

Lai veiktu transformāciju uz stāvokļa diagrammu, ir nepieciešams noteikt, kādos stāvokļos sistēma var atrasties. Pēc dotā divpusložu modeļa procesu piemēra var secināt, ka sistēma var atrasties četros stāvokļos. Katrā stāvoklī (izņemot stāvokli "*Add Item to Cart*") ir sava vizuālā reprezentācija, tas nozīmē, ka ir nepieciešams papildināt divpusložu modeļa notāciju ar to, kā process ir jāinterpretē – vai nu tas ir stāvoklis ar savu skatījumu, vai arī tas ir starpstāvoklis, kas faktiski jāinterpretē kā notikums. Turklāt "*Add Item to Cart*" no lietotāja saskarnes pēc būtības ir process jeb notikums, tāpēc nepieciešams to izlaist un analizēt nākamo procesu, šajā gadījumā "*Show Cart*". Ja starpstāvoklis var virzīties tālāk uz vairāk nekā vienu procesu, tad tas ir jāņem vērā, projektējot divpusložu modeļa procesu modeli, lai nebūtu lieki jāpapildina notācija. Diagrammā var redzēt divpusložu modeļa procesa ceļu, izmantojot piedāvāto interneta veikala piemēru. Šī stāvokļa diagramma ir izgūta, izmantojot *BrainTool* kā divpusložu modeļa veidošanas rīku un *XState* bibliotēku, kartējot divpusložu modeļa procesus, izmantojot iepriekš minēto pieeju.

3.3. attēlā redzama kopējā shēma stāvokļa diagrammas izgūšanai no divpusložu modeļa. Pirmajā solī tiek izgūta *XML* datne no *BrainTool* rīka, kurā ir izstrādāts divpusložu modelis. Tālāk tiek izgūti dati no *XML* datnes un normalizēti tā, lai varētu izgūt procesu koku (kādi stāvokļi ir pieejami un kur var nokļūt tālāk), koncepti, kas tālāk tiek nodoti stāvokļa diagrammai

kā konteksts un metodes, ja tādas ir izstrādātajam divpusložu modelim. Metodes tiek attiecīgi pievienotas stāvokļa diagrammai pie konteksta. Stāvokļa diagrammas kontekstam tiek izveidoti katram konceptam izstrādātie atribūti un pievienotas vērtības pēc atribūtu datu tipiem.



3.3. att. Stāvokļa diagrammas izgūšana no divpusložu modeļa.

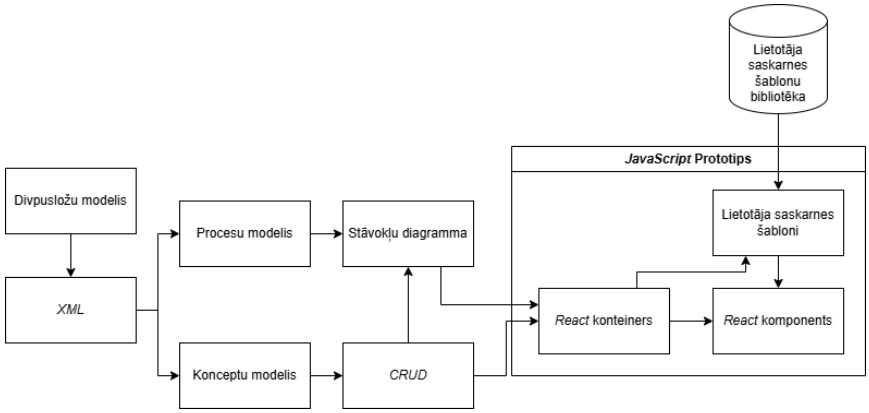
Divpusložu modeļa procesa modelis pēc būtības ir ļoti tuvs stāvokļu diagrammai, jo tas ietver stāvokļus (procesus), pārejas starp tiem (savienojumi), bet neietver informāciju, kas norādītu, vai esošais stāvoklis ir lapa, ko nepieciešams demonstrēt lietotājam, vai arī no lietotāja saskarnes skatpunkta ignorējams stāvoklis. Autors darbā apvieno divpusložu modeli ar *XState* stāvokļa diagrammām un *React Router* deklaratīvai maršrutēšanai *React.js* lietojumprogrammās. Būtībā var izmantot arī citas bibliotēkas un satvarus – tas nemaina principa būtību. Šāda pieeja strukturē lietotāja mijiedarbību, jo *XState* definē lietotāja mijiedarbību ar skaidriem stāvokļiem un pārejām, kā arī deklaratīva maršrutēšana ar *React Router* vienkāršo lietotāja saskarnes navigāciju, pamatojoties uz *URL* izmaiņām, un sniedz turpmākas deklaratīvas iespējas izmantošanu izstrādē. Šāda metode var apstrādāt gan funkcionālās (lietotājam neredzamas), gan prezentācijas daļas (lietotājam redzamie stāvokļi), tomēr šajā gadījumā ir nepieciešams papildināt divpusložu modeli ar iespēju noteikt, vai esošais process ir jāparāda lietotājam, vai nē. Lielām lietojumprogrammām stāvokļu pārvalde var kļūt sarežģīta un, iespējams, nepieciešams padomāt, kā šos stāvokļus, maršrutus un pārejas veidot modulārākas.

Divpusložu modeļa procesiem nevar norādīt, vai esošais process ir stāvoklis ar savu prezentācijas veidni, vai tas ir starpstāvoklis (pāreja), tāpēc divpusložu modeļa procesu nepieciešams papildināt ar tipu, kas no lietotāja saskarnes puses noteiktu, vai šim procesam ir sava veidne, ko nepieciešams attēlot, vai arī tas ir process, kas lietotājam nav jāredz. Par šādas iespējas piedāvāšanu tiek diskutēts nākamajā apakšnodaļā, kur tiek apskatīta prezentācijas izgūšana.

Divpusložu modeli var nepapildināt ar šādu tipa atribūtu, ja nākamajā apakšnodaļā – prezentācijas apakšnodaļā – var noteikt, vai esošais process ir demonstrējams lietotājam, vai arī tam nav jābūt redzamam lietotājam. Šāda pieeja varētu atvieglot divpusložu modeļa izstrādi un uzturēšanu.

3.3. Prezentācija

Prezentācijas projektējums iekļauj abstrakta lietotāja saskarnes modelēšanu, parādot, kā lietotājam tiek demonstrēta navigācijas struktūra (*Koch & Mandel*, 1999). Prezentācijas izstrāde nosaka veidu, kā parādīsies navigācijas mezgli, kā atlasīt lietotāja saskarnes objektus, lai aktivizētu navigāciju, un noteikt, kuras saskarnes transformācijas notiks. Viena un tā pati navigācijas struktūra var nodrošināt dažādas prezentācijas atkarībā no mērķa platformas ierobežojumiem un izmantotās tehnoloģijas (*Koch & Mandel*, 1999). Lietotāja saskarnes šabloni, kurus nosaka pēc stāvokļa nosaukuma, iespējamiem nākamajiem stāvokļiem un datu modeļa (koncepta), demonstrē reaģējošu lietotāja saskarni (angļu val. *Responsive User Interface*), kas tiktu vienlīdz labi attēlota gan tīmekļa lietotnēs, gan mobilajās lietotnēs. Demonstrējamo elementu izmērs un pozīcija tiek noteikta pēc *Material-UI* un lietotāja saskarnes šabloniem. Etiķešu un teksta lauku pārus, režģi (angļu val. *Grid*), atsevišķu etiķešu vai pogu izlīdzināšanu utt. iespējams demonstrēt ar iepriekš definētiem lietotāja saskarnes šabloniem. 3.4. attēlā redzama kopējā shēma – saskarnes skatījumu izgūšana no divpusložu modeļa, kas ietver gan saskarnes satura elementus (konceptus), gan navigācijas elementus.



3.4. att. No divpusložu modeļa izgūstamo skatījumu kopējā shēma.

Pirmajā solī tiek izgūta *XML* datne no *BrainTool* rīka, kurā ir izstrādāts divpusložu modelis. Tālāk tiek izgūti dati no *XML* datnes un procesa un koncepta modeļi. No koncepta modeļa tiek izveidota *CRUD* kartēšana un paralēli, izgūstot stāvokļa diagrammu, var izveidot saprotamus nosaukumus konteineriem un komponentiem. *React.js* konteiners atbild par datu un procesu kartēšanu *React.js* komponentam, kas faktiski ir statisks skatījums, kas tikai attēlo datus, bet visus notikumus apstrādā *React.js* konteiners. Izmantojot esošo stāvokļu diagrammu un koncepta modeli, ar *CRUD* kartēšanu var izveidot lietojamu lietotāja saskarnes prototipu, tomēr ir iespēja vēl uzlabot šos skatījumus, pievienojot jaunus elementus, lai iespējotu attēlot sarežģītas un ar elementiem bagātākas lietotāja saskarnes. Jaunais elements, kas ir parādījies šajā shēmā, ir nepieciešamība pēc lietotāja saskarnes šabloniem. Eksistē vairākas pieejas saskarnes šablonu veidošanā. Lai varētu piemērot lietotāja saskarnes šablonus izstrādājamā

projekta procesiem, nepieciešams identificēt šos šablonus un kartēt ar procesiem. Šajā darbā autors apskata literatūrā sastopamos veidus, kā pusautomātiski vai automātiski piemērot izstrādes šablonus. Tiek pieņemts, ka izstrādes šablonu definēšanā un izvēlē ir līdzība ar lietotāja saskarnes šablonu definēšanu.

Šajā darbā tiek izvēlēta teksta klasifikācijas pieeja kā atlases metode lietotāja saskarnes šabloniem ar zemu izmaksu priekšapstrādei, tādējādi tā ir ātrāka, vienkāršāka un lētāka nekā citas pieejas. Piedāvātā metode tika novērtēta, izmantojot trīs dizaina modeļu grupas un pietiekami daudz reālu dizaina problēmu aprakstu (*Hasheminejad & Jalili*, 2012).

Tāpat kā programmatūras šablonu izmantošana koda ģenerēšanā uzlabo ģenerētā koda kvalitāti (*Sunitha & Samuel*, 2019), tā arī, izmantojot lietotāja saskarnes šablonus, tiek uzlabota lietojamības kvalitāte ar cilvēkiem pierastiem maršrutiem un prezentāciju.

Prezentācijas plānojuma uzdevums ir iepriekšējos soļos izgūto navigācijas objektu prezentācijas definēšana, kas izpaužas kā statiskais un dinamiskais prezentācijas modelis (*Koch & Mandel*, 1999). Statiskais prezentācijas modelis katram navigācijas objektam piesaista vismaz vienu prezentācijas objektu, dinamiskais prezentācijas modelis apraksta šo prezentācijas objektu uzvedību (*Koch & Mandel*, 1999). Lietotāja saskarnes ģenerēšanas procesā, izmantojot noteiktus šablonus, var iegūt arī *CRUD* darbības (*Nasiri, Rhazali et al.*, 2023), tādējādi katrs process un koncepts tiek attiecīgi pārveidots par lietotāja saskarnes elementu (vai elementu kopu) (*Antović, Vlajić et al.*, 2012).

3.4. Transformācijas likumu veidošanas princips

Nepieciešams izveidot transformācijas likumus, lai saskaņotu procesu nosaukumus ar šabloniem no lietotāja saskarnes šablonu bibliotēkas. Transformācijas likumu mērķis ir automātiski ieteikt lietotāja saskarnes šablonus no šablonu bibliotēkas, paļaujoties uz procesu nosaukumiem, kas norāda gan šablona nosaukumu, gan tā funkciju.

Pirmkārt, ir nepieciešams identificēt atslēgvārdus (*Haz, Funabiki et al.*, 2024). Divpusložu modeļa procesa nosaukums tiek izmantots, lai izgūtu lietotāja nodomu. Piemēram, procesa nosaukums "*Show Cart*", ietver gan "*Show*", gan "*Cart*", no kā var secināt, ka šablons ir saistīts ar preču grozu un attēlošanas funkciju. Otrkārt, identificētie atslēgvārdi ir jākartē ar lietotāja saskarnes šabloniem. Identificētie divpusložu modeļa procesa nosaukumu atslēgvārdi tiek kartēti ar lietotāja saskarnes šablonu bibliotēku šabloniem. Šī bibliotēka asociē atslēgvārdus vai funkcionalitāti ar noteiktiem lietotāja saskarnes šabloniem. Piemēram, "*Create*" var tikt asociēts ar formas šablonu, savukārt "*Item*" – ar specifisku datu ievades formu apakšsablonu lietotāja saskarnes šablonu bibliotēkā. Treškārt, pēc izvēles var ņemt vērā arī kontekstu jeb koncepta elementu nosaukumus, lai precīzāk kartētu lietotāja saskarnes šablonus. Piemēram, "*Search Items*" var piedāvāt "*Search Bar*" šablonu, taču, ja process ietver filtrēšanu pēc noteiktiem kritērijiem, tad precīzāks "*Filter Panel*" šablons iederētos labāk. Šo transformēšanas likumu efektivitāte lielā mērā ir atkarīga gan no procesa nosaukumu, gan lietotāja saskarnes šablonu bibliotēkas visaptverošuma un skaidrības. Lai formāli strukturētu nepieciešamos veicamos soļus, tiek piedāvāts lietotāja saskarnes šablonu procesu definēt kā kortežas kopu.

Faktiski var būt vairākas situācijas, kurās nepieciešams atrast precīzāko šablonu. Šablonu bibliotēkai jābūt precīzi definētai ar skaidriem aprakstiem un piemēriem katram šablonam, jo, izmantojot teksta klasifikācijas metodi, tiks izvēlēti šabloni, balstoties atslēgvārdos *(Hasan, Sanyal et al.*, 2017). Savukārt transformācijas likumus var pielāgot atkarībā no lietojumprogrammas problēmvides un funkcijām. Tālākajos pētījumos var izskatīt mašīnmācīšanās paņēmienus, lai uzlabotu šablonu atlases precizitāti, analizējot iepriekšējo kartējumu un lietotāju mijiedarbību.

Atslēgvārdu identifikācija un atbilstības noteikšana ir galvenā ideja, lai procesu nosaukumus pārvērstu lietotāja saskarnes šablonos. Izmantojot atslēgvārdu atbilstību, var kartēt atslēgvārdus no procesa nosaukuma ar iepriekš definētiem šabloniem bibliotēkā. Ja, pamatojoties uz atslēgvārdiem, ir vairākas iespējamās atbilstības, lietotāju saskarnes šablonu bibliotēka var piedāvāt šablonu prioritātes līmeņus. Augstākas prioritātes modeļi parasti ir specifiskāki, un tiem ir prioritāte. Šāda pieeja palīdz piedāvāt primāro lietotāja saskarnes šablonu, kas atbilst procesa funkcionalitātei. Piemēram, "*Create Customer Account*" kā sākotnējā opcija tiek ieteikta "*User Registration Form*". Efektivitāte lielā mērā ir atkarīga gan no procesu nosaukumu, gan atslēgvārdu precizitātes un detalizācijas, kas saistīti ar lietotāja saskarnes šabloniem lietotāja saskarnes šablonu bibliotēkā. Izmantojot šo pieeju, var veikt lietotāja saskarnes dizaina automatizāciju, pamatojoties uz procesu definīcijām, kas savukārt var paātrināt lietotāja saskarnes izstrādi. Turklāt šāda pieeja veicina lietotāja saskarnes šablonu konsekvenci visā lietojumprogrammā. Tomēr par trūkumu ir jāatzīst tas, ka sarežģītos procesos var būt nepieciešama manuāla iejaukšanās, lai izvēlētos vispiemērotāko šablonu un šīs pieejas efektivitāte ir atkarīga no procesu nosaukumu un lietotāja saskarnes šablonu bibliotēkas kvalitātes. Lai sāktu transformācijas procesu, ir nepieciešams divpusložu modelis. Visbiežāk tas tiek veidots *BrainTool* rīkā un ietver procesa modeli un koncepta modeli. *BrainTool* piedāvā eksportēt šo modeli uz *XML* datnes formātu. Šo *XML* datni apstrādājot, tiek iegūts gan procesu modelis, gan koncepta modelis ārpus *BrainTool* rīka apstrādājamā formā.

Koncepta modelis tālāk tiek kartēts ar *CRUD*, lai izgūtu datu modeļa entītijas un izlemtu, kādiem parametriem ir jābūt nodotiem. Iegūtie objekti un mainīgie tālāk tiek kartēti ar demonstrācijas datiem, lai prototipos būtu aizpildīti skatījumi un varētu redzēt, kā aptuveni izskatīsies gatava lietotāja saskarne. Mainīgie ar iestatītā vērtībām tālāk tiek definēti *React.js* konteinerā. Savukārt no procesu modeļa tiek izgūta stāvokļa diagramma, kas tālāk tiek kartēta ar *React Router* maršrutiem. Maršruti tiek definēti atsevišķi kopā ar *React.js* konteinera izsaukumiem, lai norādītu, kāds konteiners pie kāda maršruta izsaukuma tiek parādīts galalietotājam. Kad ir sagatavoti gan mainīgie, gan maršruti, tiek izveidots *React.js* konteinera komponents, kas tiek izvēlēts, pamatojoties uz procesa modeļa vienuma nosaukumu, izmantojot teksta klasifikācijas metodi – izgūts no lietotāju saskarnes šablonu bibliotēkas. Tādējādi tiek izgūts *JavaScript* prototips no divpusložu modeļa. Modeļu transformācijas apvienošana ar jaunākajām priekšgala tehnoloģijām, piemēram, *React.js*, paātrina izstrādes laiku un uzlabo vispārējo programmatūras kvalitāti. Divpusložu modelis sniedz izstrādātājiem noderīgu struktūru procesu un konceptu kartēšanai ar priekšgala saskarnēm. Tas savukārt var palīdzēt programmatūras projektus izstrādāt ātrāk, izpildot lietotāju prasības un biznesa mērķi – mazināt atšķirības starp tehniskajām funkcijām un galalietotāja pieredzi.

4. RISINĀJUMA APROBĀCIJA UN NOVĒRTĒŠANA

Promocijas darba gaitā izstrādātā risinājuma pirmais solis ietver izpratni par konkrēto problēmvidi, kurai tiek izstrādāts lietotāja saskarnes prototips, un avota modeļa (divpusložu modeļa) izstrādi. Tālāk izveidotais avota modelis tiek apstrādāts ar transformācijas likumu dzinējam, rezultātā iegūstot lietotāja saskarnes prototipu kā ekrānformu pirmkoda failu kopu, kas implementē problēmvides funkcionalitāti un lietotāja mijiedarbību problēmvidē. Automātiskā lietotāja saskarnes prototipu ģenerēšana kā dizaina pamatu izmanto iepriekš definētus lietotāja saskarnes šablonus. Lietotāja saskarnes šablonos ir izveidoti dizaina risinājumi izplatītiem lietotāja saskarnes elementiem un funkcijām, veicinot intuitīvu un lietotājam draudzīgu saskarni (*Nikiforova, Babris et al.*, 2024b). Iepriekš definētu šablonu izmantošana var paātrināt prototipu veidošanas procesu, salīdzinot ar lietotāja saskarnes izveidi no jauna. Lietotāja saskarnes šabloni nodrošina konsekvenci starp prototipiem un lietotāja mijiedarbībā visā lietojumprogrammā. Iegūtā transformācijas rezultāta atbilstība sagaidāmam rezultātam apliecina risinājuma darbspēju.

Lai validētu izstrādāto risinājumu iegūtā rezultāta kvalitātes ziņā, ir nepieciešams analizēt transformācijas rezultātu divos aspektos.

1. Iegūtā koda kvalitāte, kas attiecas uz koda vispārējo efektivitāti un uzturēšanu. Tas attiecas ne tikai uz koda pareizu darbību, bet arī uz to, cik labi tas ir uzrakstīts, strukturēts un dokumentēts (*Nikiforova, Zabiniako et al.*, 2021a). Kodam jābūt viegli saprotamam ikvienam izstrādātājam, kurš pārzina izmantoto programmēšanas valodu. Tas ietver pareizas atkāpes, jēgpilnus mainīgo nosaukumus un komentārus, kas izskaidro sarežģītu loģiku, ja tādi ir nepieciešami.
2. Iegūtā tīmekļa lietojuma atbilstība problēmvides biznesa procesam un sagaidāmiem lietošanas scenārijiem, kas attiecās uz koda funkcionēšanu. Lietojuma funkcionēšanu ir iespējams pārbaudīt ar pieņemšanas testēšanas metodi (*ISO/IEC/IEEE 29119*, 2013), ar ko var apstiprināt, vai iegūtais prototips darbojas korekti un atbilst problēmvidē identificētiem biznesa procesiem.

Šajā apakšnodaļā ir demonstrēta gan izstrādātā risinājuma lietošana, izmantojot nelielu abstrakta aprobācijas piemēru, kas apliecina risinājuma darbspēju, gan aprakstīta arī transformācijas rezultāta pieņemšanas testēšana, kas validē risinājuma lietošanas rezultātu.

4.1. Problēmas vides apraksts un sagaidāmais rezultāts

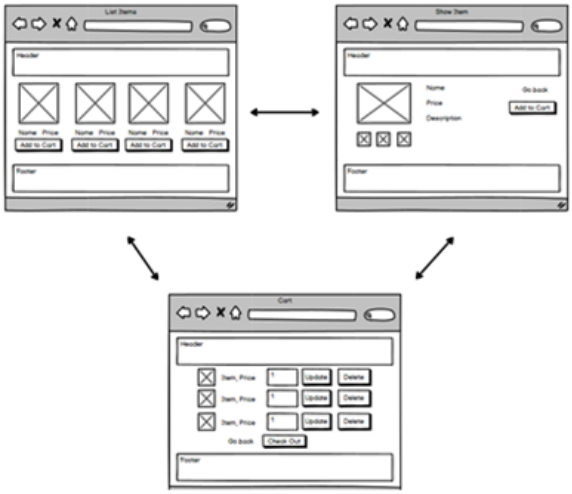
Lai demonstrētu piedāvāto pieeju, tika izvēlēts piemērs. Šajā gadījumā piemērs ir interneta veikala pārdodamais produkts – moduļa veidā izstrādāta sistēma, ko var viegli uzstādīt uz izvēlētajām platformām, neveicot kādas specifiskas pielāgošanas procesus. Produktu var iegādāties vai nu uz mēnesi, vai arī abonēt uz gadu, turklāt jāparedz iespēja šo abonementu automātiski pagarināt, ja ir izvēlēta tāda opcija. Šajā procesā lietotājs var reģistrēties, pieteikties, apskatīt preces, pievienot tās grozam un izveidot pasūtījumu. Piedāvātais process iezīmē tipisku (*Fernández, Liu et al.*, 2021) lietotāja mijiedarbību ar interneta veikalu. Process var sākties no reģistrācijas, pieteikšanās un produktu saraksta apskates. Pēc preču pārlūkošanas

iespējams preci pievienot iepirkumu grozam. Iepirkuma grozu var pārvaldīt, noņemt preces vai mainot to daudzumu. Pēc tam ir iespējams noformēt pasūtījumu. Process sīkāk aprakstīts (*Babris & Nikiforova*, 2024b).

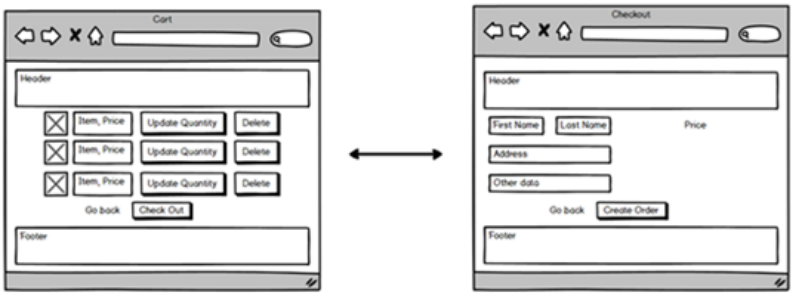
1. Lietotāja reģistrēšana – lietotājs, apmeklējot interneta veikala vietni, aktivizē pogu "Reģistrēties" un tiek novirzīts uz reģistrācijas formu, kurā viņš ievada savus personas datus, piemēram, vārdu, e-pasta adresi, paroli. Pēc nepieciešamās informācijas aizpildīšanas lietotājs iesniedz formu. Sistēma izveido jaunu lietotāja kontu.
2. Lietotāja pieteikšanās, ja lietotājam jau ir konts. Viņš var pieteikties, noklikšķinot uz pogas "Pieteikties". Pieteikšanās formā lietotājs ievada savu e-pasta adresi un paroli. Pēc iesniegšanas sistēma pārbauda akreditācijas datus un, ja tie ir pareizi, piešķir piekļuvi lietotāja kontam.
3. Preču pārlūkošana – lietotājs var pārlūkot interneta veikala preču katalogu. Lietotājs var izmantot meklēšanas funkcionalitāti, lai atrastu konkrētus vienumus, kas viņus interesē. Katrai precei tiek parādīts tās nosaukums, cena un apraksts, un ir iespēja preci pievienot grozam.
4. Preču pievienošana grozam. Kad lietotājs atrod preci, ko vēlas iegādāties, viņš noklikšķina uz pogas "*Pievienot grozam*". Atlasītā prece tiek pievienota lietotāja iepirkumu grozam, kurā redzams visu līdz šim pievienoto preču kopsavilkums. Lietotājs var pielāgot katras preces daudzumu grozā vai izņemt preces pavisam.
5. Groza pārvaldīšana. Groza skatā lietotājs var redzēt visas pievienotās preces, kā arī to daudzumus un cenas. Lietotājs var atjaunināt preču daudzumu, pielāgojot daudzuma lauku, vai noņemt preces, noklikšķinot uz pogas "Noņemt".
6. Norēķinu process. Pēc izvēles pabeigšanas lietotājs pāriet uz norēķināšanās skatījumu, noklikšķinot uz pogas "Norēķināties". Lietotājam tiek parādīta norēķinu skatījuma forma, kurā viņš apstiprina savu piegādes adresi. Kad nepieciešamā informācija ir iesniegta, lietotājs iesniedz norēķinu formu un sistēma apstrādā pasūtījumu.
7. Pēc pasūtījuma veikšanas lietotājs tiek pāradresēts uz savu profilu.

Lietotāji pāriet no preču kataloga, lai skatītu detalizētu informāciju par konkrētām precēm, ko vēlas apskatīt. Tas ļauj lietotājiem viegli izpētīt produkta informāciju, piemēram, aprakstus, attēlus un specifikācijas. Pēc preces informācijas pārbaudes un lēmuma par pirkšanu lietotāji var netraucēti pāriet uz iepirkumu groza zonu, lai pievienotu izvēlētās preces. Kā arī var no preču kataloga uzreiz pievienot preci grozam. 4.1. (a) attēlā redzama preču kataloga, preču vienuma un preču groza shēma un tas, kā var pārvietoties pa šiem elementiem.

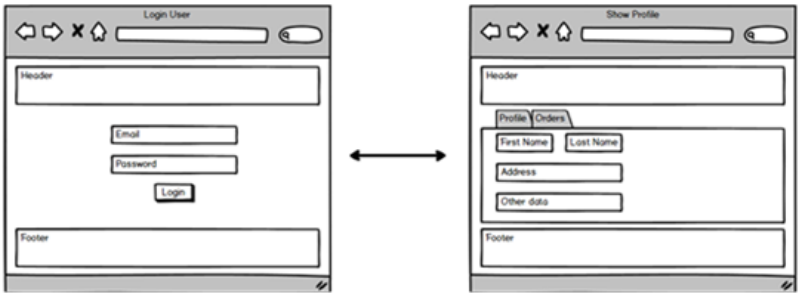
Savukārt no preču groza var nokļūt vai nu uz norēķina sadaļu, kur nepieciešams ievadīt rekvizītus, kas nepieciešami rēķina sagatavošanai, izvēlēties maksājuma veidus utt., vai arī atpakaļ – uz preču katalogu. 4.1. (b) attēlā redzama preču groza un norēķina shēma. Pabeidzot pasūtījumu, lietotājs tiek novirzīts uz lietotāja profilu (4.1. (c) att.). Uz lietotāja profilu lietotājs tiek novirzīts arī tādā gadījumā, ja ir notikusi lietotāja pieteikšanās sistēmā. Atslēdzoties no sistēmas, lietotājs tiek novirzīts uz pieteikšanās skatījumu.



(a) Preču kataloga, vienuma un preču groza shēma.



(b) Preču groza un norēķinu shēma.

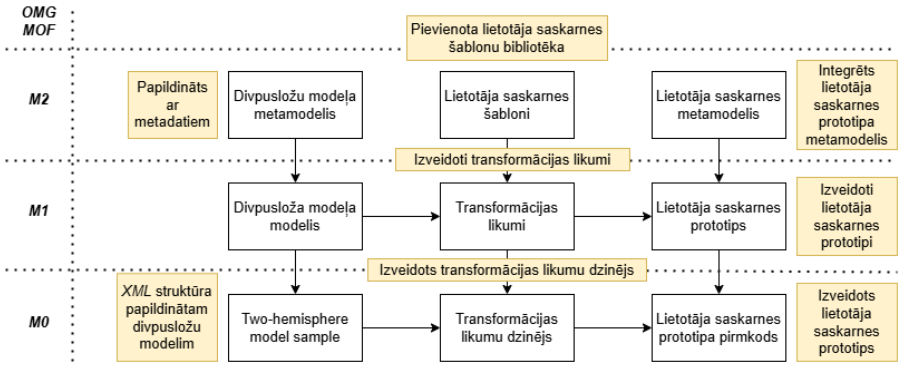


(c) Lietotāja pieteikšanās un lietotāja profila shēma.

4.1. att. Ekrānu mijiedarbības skices.

4.2. Risinājuma konceptuālā shēma un galvenie komponenti

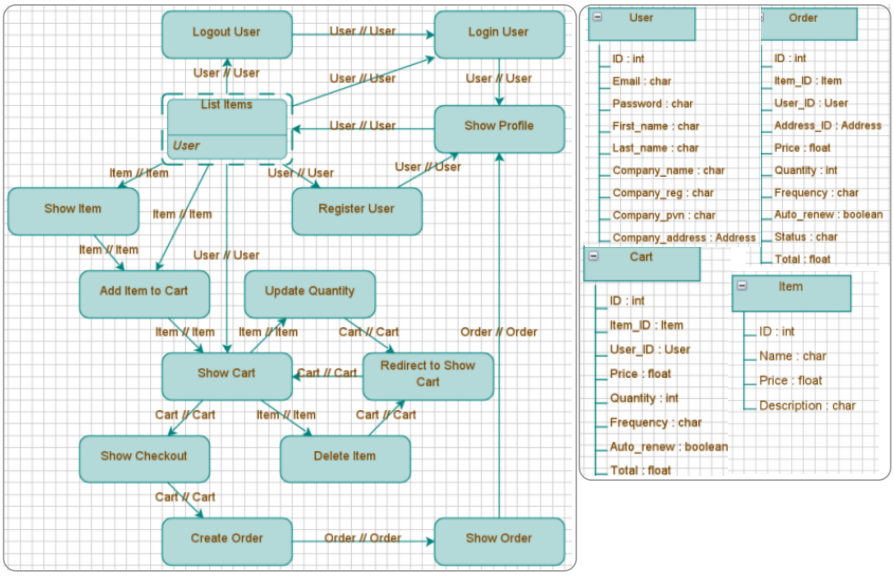
Promocijas darba gaitā izstrādātais lietotāja saskarnes prototipa ģenerēšanas risinājums redzams 4.2. attēlā, attēlojot risinājuma komponentes saskaņā ar objektu vadības grupas metaobjektu spējas (angļu val. *Meta Object Facility, MOF*) ietvaru (*https://www.omg.org/mof/*).



4.2. att. Risinājuma konceptuālā shēma lietotāja saskarnes prototipa ģenerēšanai no divpusložu modeļa.

Teksta blokos bez iekrāsojuma redzami risinājuma komponenti. Komentāriem uz dzeltenā fona aprakstīti veiktie papildinājumi, modifikācijas vai jaunizstrādājumi, kas esošajiem artefaktiem ir veikti promocijas darba gaitā. Risinājuma centrāla komponente ir transformācijas likumi, kas definē to, kā no divpusložu modeļa elementiem tiek veidoti lietotāja saskarnes elementi, ievērojot attiecīgos lietotāja saskarnes šablonus. Risinājumā tie realizēti *PHP* programmēšanas valodā. Transformācijas likumi izmanto lietotāja saskarnes šablonus, ko ir iespējams izvēlēties atkarībā no procesa metadatiem. Transformācijas likumu dzinējs (programmatūra) palaiž transformācijas likumus divpusložu modeļa *XML* datnei un rezultātā izveido tīmekļa lietotnes priekšgala komponenšu pirmkodu *JavaScript* programmēšanas valodā. Divpusložu modeļa metamodelis un lietotāja saskarnes metamodelis apraksta avota un mērķa modeļu saturu un struktūru transformācijas likumu definēšanai. Divpusložu modelis, salīdzinot ar tā sākotnējo versiju, ir papildināts ar procesu metadatiem. Lietotāja saskarnes metamodelis ir veidots, integrējot esošos saskarnes metamodeļus un pārstrukturējot tos attiecībā uz darbā veikto lietojumsistēmas priekšgala komponentu kartēšanu un sagaidāmo tīmekļa lietotnes lietotāja saskarnes elementu kopu (aprakstīta *UML* klašu diagrammas veidā). Lietotāja saskarnes prototips pēc būtības ir strādājoša tīmekļa lietotne kā mijiedarbojošās ekrānformas palaistas tīmekļa pārlūkā. Un transformācijas rezultāts ir tīmekļa lietotnes priekšgala komponenšu pirmkods, kas ir palaižams kādā integrētajā izstrādes vidē ar realizācijai izvēlēta satvara atbalstu (promocijas darbā demonstrācijas piemērā ir izmantotas *React*.js, *Redux.js*, *XState* bibliotēkas).

Risinājuma demonstrācijai un aprobācijai 4.3. attēlā parādīts divpusložu modelis izvēlētai problēmvidei, *BrainTool* rīkā uzdodot attiecības starp attiecīgajām procesu plūsmām un konceptiem. Par problēmvidi ir izvēlēts tipisks interneta veikals, ko var lietot dažu produktu izvēlei un abonēšanai, tam nav kategoriju hierarhiju vai citādu kompleksu klasifikācijas sistēmu. Galvenā prasība ir, lai potenciālais vai esošais uzņēmuma lietotājs varētu apmaksāt sistēmas abonementu par viņam izvēlētu laika periodu.



4.3. att. Problēmvides divpusložu modeļa procesi (pa kreisi) un koncepti (pa labi).

Divpusložu modelis ir veidots *BrainTool* rīkā, tāpēc informāciju par tajā esošajiem konceptiem, procesiem un to savstarpējām saistībām ir iespējams transformēt apstrādājamā formā, t. i., eksportēt attiecīgo *XML* (paplašināmā iezīmēšanas valoda) datni.

XML ir iezīmēšanas valoda, kas nosaka noteikumu kopumu dokumentu kodēšanai formātā, kas ir lasāms gan cilvēkiem, gan mašīnlasāms. To parasti izmanto strukturētu datu glabāšanai hierarhiskā formātā. Divpusložu modeļa avota modelis, parasti ir *XML* datne, kas attēlo procesus un konceptus. Šajā divpusložu modelī avota modeļa *XML* datnes struktūra definē elementus un atribūtus, ko var izmantot lietotāja saskarnes komponentu ģenerēšanā, kā arī to īpašības, attiecības un uzvedību. Divpusložu modelī *XML* elementi varētu aprakstīt entītijas, funkcijas, saites un darbības, kas tālāk tiktu izmantotas darbā piedāvātajā risinājumā. No divpusložu modeļa ir iespējams ģenerēt arī secību diagrammas (*Nikiforova, Kozacenko et al.*, 2013) un *UML* klašu diagrammas (*Nikiforova & Pavlova*, 2011), kas liecina par to, ka divpusložu modelis ir pietiekami funkcionāls, lai no tā varētu ģenerēt lietotāja saskarnes elementus.

4.3. Mērķa modeļa pirmkoda datne un datu formāts

Risinājuma izejas dati tiek pasniegti kā *React.js JavaScript* pirmkods. Katram satvaram vai bibliotēkai var tikt izstrādāti savi modeļa transformēšanas likumi, lai iegūtu pilnvērtīgu lietotāja saskarni. Ņemot vērā to, ka viena no detalizētākajām lietotāja saskarnes modelēšanas pieejām ir *IFML* (angļu val. *Interaction Flow Modeling Language, IFML*), arī piedāvātajā risinājumā tiek aizgūti tādi elementi kā *Container* un *Component*, lai atdalītu funkcionālos un konteksta elementus no prezentācijas elementiem. Piemērojot šādu arhitektūru, var iegūt *React.js* un *Redux.js* kodu.

React.js un *Redux.js* tīmekļa lietojumprogrammā ir daudzas daļas, kas palīdz organizēt un kontrolēt lietotāja saskarni, kā arī stāvokļa pārvaldību. Piemēram, <*ComponentName*> ir tehniskais nosaukums *React.js* komponentam, ko var izmantot atkārtoti dažādās projekta vietās. Šis nosaukums ir atbildīgs par konkrētas lietotāja saskarnes daļas rādīšanu ekrānā, vienlaikus kontrolējot, kā šis apgabals darbojas. Elements ar nosaukumu <*ContainerDependencies*> ietver norādes uz izmantojamiem komponentiem vai konteineriem, tas var ietvert arī norādes uz kādu noteiktu bibliotēku lietošanu. Elements <*ContainerConceptDefinitions*> ietver galvenās datu struktūras vai entītijas, ko apstrādā tīmekļa lietojumprogramma. Izmantojot demonstrācijas datu ģeneratoru, var aizpildīt šīs datu struktūras tādā veidā, lai lietotājam šos datus būtu viegli uztverams un radītu prototipu, kas būtu maksimāli tuvs reālai lietotāja saskarnei. Elements <*ContainerConcepts*> ir iepriekš definēto <*ContainerConceptDefinitions*> elementu saraksts, kas tiek nodots komponentam, no konteinera definīcijas. Faktiski <*ContainerConcepts*> ir <*ContainerConceptDefinitions*> vienkāršojums. Pēdējais elements <*ContainerMethods*> var ietvert dažādas metodes vai funkcijas, kas pieder šim konteinera komponentam. Parasti šīs metodes apstrādā tādas lietas kā lietotāja darbības un datu apstrāde, kā arī navigāciju (4.4. att.).

```
import React from 'react';
import { connect } from 'react-redux';
import { faker } from '@faker-js/faker';
import { goNext, getStateMeta } from "./shared"

<ContainerDependencies>

import <ComponentName> from "./<ComponentName>"


function mapStateToProps(state, props) {
    const currentState = props.stateActor.getSnapshot();
    const meta = getStateMeta(currentState);

    <ContainerConceptDefinitions>

    return {
        title: meta.title,
        <ContainerConcepts>
    };
}

function mapDispatchToProps(dispatch, props) {
    return {
        goNext: (nextStateName) => {
            return goNext(nextStateName, props.stateActor)
        },
        <ContainerMethods>
    }
}

export default connect(
    mapStateToProps,
    mapDispatchToProps
)(<ComponentName>);
```

4.4. att. Minimālā konteinera veidne.

Arī komponenta gadījumā <*ComponentName*> ir komponenta tehniskais nosaukums un elements ar nosaukumu <*ContainerDependencies*>, līdzīgi kā <*ContainerDependencies*> ietver norādes uz izmantojamiem komponentiem vai konteineriem, kā arī uz šablonu bibliotēkām vai citām nepieciešamajām atkarībām. Elements <*ComponentConcepts*>, tāpat kā <*ContainerConcepts*>, ir iepriekš definēto konceptu elementu saraksts, kas tiek nodots komponentam no konteinera definīcijas. Komponentu šādā veidā var lietot atkārtoti arī citās projekta daļās (4.5. att.).

```
1   import * as React from "react";
2   import { useNavigate } from "react-router-dom";
3
4   <ComponentDependencies>
5
6   export default function <ComponentName>(props) {
7       const navigate = useNavigate();
8       const { goNext, title <ComponentConcepts>} = props;
9
10      return (<ComponentContent>)
11  }
```

4.5. att. Minimālā komponenta veidne.

Pēdējais elements <*ComponentContent*> ir vissvarīgākais elements, jo ietver lietotājam attēlojamo elementu kopumu, kas definēts lietotāja saskarnes šablonā.

Sakarā ar to, ka tiek ģenerēti prototipi, bez datiem nebūs pilnīgas iespējas galalietotājam saprast, kā rezultāti izskatīsies lietotāja saskarnē. Lai risinātu šo problēmu, tiek piedāvāts izmantot demonstrācijas datus, kas nenes nekādu informāciju, bet tiek izmantoti tikai, lai lietotājam demonstrētu maksimāli reālu prototipu.

4.4. Avota un mērķa modeļu kartēšana

Avota un mērķa modeļa kartēšana ar *React Router*, konteineriem un komponentiem nozīmē arhitektūras modeļa attēlojuma pārveidošanu maršrutu un komponentu izkārtojumā *React.js* tīmekļa lietojumprogrammā.

React Router ir rīks, kas palīdz savienot katru konteineru un komponentu ar noteiktu maršrutu *React.js* tīmekļa lietojumprogrammā. Tādēļ nepieciešams definēt maršrutus, kas atbilst dažādiem divpusložu modeļa procesiem.

Veicot šīs darbības, izstrādātāji var kartēt divpusložu modeļa procesu plūsmu ar *React Router* maršrutētāju, konteineriem un komponentiem. Tādējādi tiek izveidots vienots, funkcionāls tīmekļa lietotnes prototips, ar ko lietotājs var mijiedarboties. Kā arī nepieciešams kartēt divpusložu modeļa konceptus ar *React.js* konteineriem un komponentiem, lai varētu zināt, kādi datu modeļi nepieciešami katrā konteinerā, lai varētu veikt to definēšanu. Kā var ievērot, tad, ja mainīgā tips ir masīvs, tad divpusložu modeļa koncepts ir jāizmanto daudzskaitlī un pie transformācijas tiek pārvērts ar objektu masīvu. Savukārt, ja sagaidāmā mainīgā tips ir objekts, tad mainīgais tiek saukts vienskaitlī un atstāts tāds, kāds ir divpusložu modeļa koncepta nosaukums. Mainīgo tipi tiek definēti pie lietotāja saskarnes šablona, tas tiek darīts manuāli, veidojot šablonu.

4.5. Sagaidāmās lietotāja saskarnes salīdzināšana ar transformācijas rezultātu

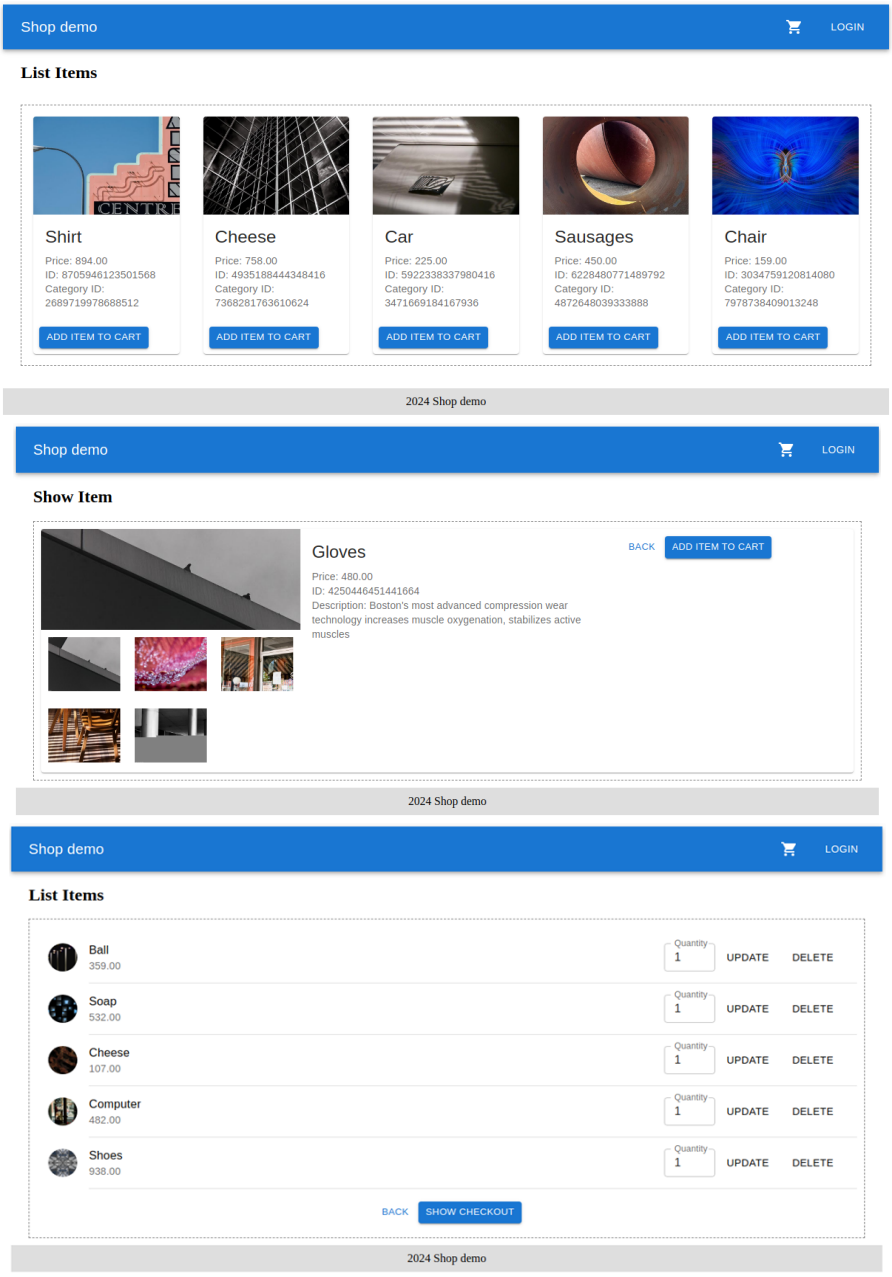
Ja visa iepriekš minētā procedūra norit sekmīgi, iespējams atvērt norādīto adresi un aplūkot iegūto rezultātu. Kā sākuma skatījumam ir jāparādās "*List Items*" (Preču katalogs).

Sadaļa "*List Items*" (4.6. att. augšējā ekrānbilde) ir pamatdaļa, kas izveidota, lai parādītu pieejamos vienumus un palīdzētu lietotājiem tos pārlūkot. Šī sadaļa, kas tiek automātiski izveidota saskaņā ar lietotāja saskarnes šabloniem un dizaina vadlīnijām, garantē vienveidību un draudzīgumu lietotājam visā iepirkšanās saskarnē. Šajā sadaļā tiek rādīti produktu sīktēli, kas sakārtoti režģa vai saraksta skatā, kur katrs sīktēls apzīmē vienu vienumu, ko lietotājs var iegādāties. Lietotāji var pārvietoties pa katalogu un apskatīt preces vienumu vai arī uzreiz to pievienot grozam. Skatījumu "*Show Item*" (4.6. att. centrālā ekrānbilde) bieži dēvē par produkta informācijas lapu vai produkta lapu, un tā ir būtiska interneta veikala sastāvdaļa. Tā ir īpaša vieta, kur lietotāji var izpētīt detalizētu informāciju par konkrētu produktu pirms pirkuma lēmuma pieņemšanas. Produkta informācijas lapā ir redzama produkta bilde kopā ar sīktēlu galeriju, preces apraksts, cena un iespēja to pievienot grozam. Sadaļā "*Show Cart*" (4.6. att. apakšējā ekrānbilde) katrai grozā esošajai precei ir pievienots tās sīktēla attēls, nosaukums, vienības cena, izvēlētais daudzums. Blakus katrai precei ir interaktīvie elementi, kas ļauj klientiem atjaunināt daudzumu vai pilnībā izņemt preci no groza. Kad klienti pielāgo daudzumus vai dzēš preces, grozs tiek dinamiski atjaunināts, lai atspoguļotu šīs izmaiņas, nodrošinot, ka kopējās izmaksas tiek precīzi aprēķinātas reāllaikā.

Promocijas darbā tiek piedāvāts validēt lietotāja saskarnes prototipu ģenerēšanu, salīdzinot ar manuālu rezultātu, interneta veikala kontekstā. Validācija balstās faktā, ka problēmvidei ir izstrādāts sagaidāmais rezultāts un tas ir salīdzināts ar transformācijas rezultātu, izmantojot darbā aprakstīto algoritmu. Darbā ir definēti pieņemšanas kritēriji, kas ir identificēti, analizējot procesu modelī esošo funkcionalitāti. Automātiski ģenerētam lietotāja saskarnes prototipam validācija koncentrējas uz apstiprināšanu, vai izstrādātie prototipi pareizi attēlo noteiktās funkcionālās prasības un lietotāju mijiedarbību. Tiek apskatīta arī ģenerētā koda kvalitāte, pārbaudot to ar tādiem rīkiem kā *JSLint*, *JSHint* un *ESLint* palīdzību. Darbā tiek apskatīti arī citi būtiski faktori, piemēram, lietojamība, veiktspēja un drošība.

Modeļvadāmas pieejas ļauj programmatūras izstrādātājiem koncentrēties uz augsta līmeņa dizaina jautājumiem. Tomēr joprojām tiek diskutēts, piemēram, cik efektīvas būs automatizētās koda ģenerēšanas metodes sarežģītos projektos vai kā šīs metodes darbosies dažādās programmēšanas valodās (*Uyanık & Sayar*, 2024).

Risinājuma aprobācija un pieņemšanas testēšana dod iespēju apgalvot, ka divpusložu modelis piedāvā vērtīgu sistēmu lietotāja saskarņu strukturēšanai. Transformācijas process, ko vada lietotāja saskarnes šabloni un metamodeļi, pārvērš šo modeli konkrētos lietotāja saskarnes komponentos un interaktīvos prototipos. Izmantojot šo metodi, lietotāja saskarnes prototipus var izveidot tā, lai tas labi atbilstu sistēmas pamata arhitektūrai. Rezultātā automātiski ģenerētais lietotāja saskarnes prototips ir atbilstošs sagaidāmai funkcionalitātei, kas promocijas darbā ir pārbaudīts ar pieņemšanas testēšanu.



4.6. att. Uzģenerēta lietotajā saskarnes prototips
ar kataloga, preču vienuma un preču groza šablonu.

REZULTĀTI UN SECINĀJUMI

Programmatūras inženierijā lietotāja saskarne kļuva par tā būtisku daļu pēc programmatūras inženierijas krīzes 1969. gadā. Šī situācija atklāja nopietnas programmatūras izstrādes grūtības, piemēram, projektu budžeta pārtēriņu, grūtības nodot projektus laikā vai vispār tos pabeigt. Problēma ir tajā, ka, programmām kļūstot sarežģītākām un grūtāk apstrādājamām, izstrādātājiem kļūst vitāli svarīgi ne tikai tās izveidot, bet arī nodrošināt lietotājiem veidus, kā efektīvi mijiedarboties ar izstrādātajiem projektiem. Tādējādi labi izveidota lietotāja saskarne ir kļuvusi par vienu no galvenajām atbildēm uz šīm problēmām, uzlabojot lietojamību – atvieglojot sarežģītas programmatūras sistēmas, izmantojot intuitīvus projektēšanas principus. Lietotāja saskarnes dizainam ir liela nozīme lietotāju apmierinātības uzlabošanā, apmācību un atbalsta izmaksu samazināšanā, padarot lietotāja saskarnes vieglāk saprotamas un lietojamas. Efektīvs lietotāja saskarnes dizains ar vienkāršiem izstrādes procesiem samazina lietotāju kļūdas un nodrošina to, ka uzdevumus var veikt ātri un precīzi. Veicinot labāku saziņu starp lietotājiem un izstrādātājiem, lietotāja saskarnes projektējums padara vienkāršāku tādas programmatūras izveidi, kas precīzāk atbilst lietotāju prasībām. Turklāt modernās lietotāja saskarnes izstrādes metodes ļauj pielāgoties dažādām vajadzībām, izmantojot atkārtotu uz lietotājiem vērstu izstrādi un dizainu, piemēram, lietotāja saskarnes šablonus. Būtībā, koncentrējoties uz lietotāja saskarnes dizainu, tiek risinātas daudzas problēmas, kas saistītas ar programmatūras inženierijas krīzi.

Vēl viens svarīgs aspekts programmatūras inženierijā ir programmatūras komponenšu ātra izstrāde, nezaudējot kvalitāti. Līdz ar to promocijas darbs ir veltīts modeļvadāmās inženierijas pamatkoncepcijās sakņota risinājuma izstrādei, kas dod iespēju automātiski ģenerēt tīmekļa lietotnes priekšgala komponentu pirmkodu, kas, no vienas puses, paātrina lietotāja saskarnes komponenta izstrādi, no otras puses, nodrošina transformācijas rezultāta kvalitāti.

Promocijas darba izstrādes laikā definētie **uzdevumi ir pilnībā izpildīti**.

1. Izpētītas lietotāja saskarnes izstrādes metodes un paņēmieni, lai identificētu potenciālos pamatelementus un formālismus, ko var izmantot lietotāja saskarnes automātiskajā ģenerēšanā.
2. Apkopota informācija par lietotāja saskarnes elementu daudzveidību, lai definētu lietotāja saskarnes elementu taksonomiju un šablonus, kas būs pamats mērķa metamodeļa izstrādē.
3. Divpusložu modeļa notācija tiek bagātināta ar transformācijas likumiem nepieciešamajiem elementiem – procesu metadatiem.
4. Izveidots avota un mērķa modeļa saturs, lai definētu transformācijas likumus, kur par avota modeli tiek izmantots problēmvides divpusložu modelis, savukārt par mērķa modeli tiek sagaidīts lietotāja saskarnes prototips šīs problēmvides atbalsta tīmekļa lietotnei.
5. Izstrādāts risinājums lietotāja saskarnes prototipu ģenerēšanai, kas ir demonstrēts, izmantojot reālas problēmvides atbalsta tīmekļa lietotnes piemēru, un ir pārbaudīta tā atbilstība sagaidāmajam rezultātam.
6. Veikta risinājuma novērtēšana un definēti secinājumi par pētījuma rezultātiem.

Promocijas darba **galvenais rezultāts** ir piedāvātais risinājums, kas nodrošina tīmekļa lietotnes lietotāja saskarnes ģenerēšanu no divpusložu modeļa, izmantojot modeļvadāmas izstrādes pamatkoncepcijas, kas ir metamodelēšana, modeļi un to transformācijas.

Promocijas darba gaitā apskatīts lietotāja saskarnes izstrādes svarīgums mūsdienu lietojumprogrammās. Tas aptver dažādus lietotāja saskarnes izstrādes likumus un metodes, uzsverot prasību pēc lietotājiem viegli saprotamām un lietojamām saskarnēm. Pētīti arī pašreizējie stili un tehnoloģijas lietotāja saskarnes izstrādē, uzsverot to ietekmi uz lietotāja pieredzi un programmatūras lietderību.

Tālākā darba gaitā runāts par modeļavadāmās izstrādes potenciālu lietotāja saskarnes izstrādes uzlabošanā. Modeļvadāmas pieejas pamatideja ir izveidot abstraktus modeļus, ko var automātiski pārveidot izpildāmā kodā. Mērķis ir palielināt efektivitāti, samazināt kļūdas un garantēt vienveidību dažādās programmatūras platformās. Promocijas darbā sniegts arī detalizēts skaidrojums par rīkiem un struktūrām, kas palīdz modeļvadāmai izstrādei.

Promocijas darbā detalizēti aplūkota modeļvadāmās pieejas būtība, definēti transformācijas likumi, kas transformē augsta līmeņa modeļus reālās implementācijās, sniegts arī izklāsts par to, kā izveidot šos likumus un pārliecināties, ka tie ir pareizi, elastīgi un var tikt pielāgoti dažādām lietotāja saskarnes vajadzībām. Darbā parādīti daži transformācijas likumi, to paraugi, kas parāda, kā tie tiek izmantoti praktiskās situācijās. Turklāt tiek arī runāts par grūtībām un galvenajām metodēm šo likumu realizēšanā un apstrādāšanai, lai iegūtu labākos rezultātus.

Pēdējā nodaļa veltīta ieteiktā modeļvadāmās lietotāja saskarnes izstrādes risinājuma aprobācijai un validācijai. Tajā izskaidrots darbā piedāvātā risinājuma lietošanas process un veikta tā novērtēšana, sniegti testu rezultāti, kas parāda, cik precīzs ir piedāvātā risinājuma rezultāts attiecībā uz sagaidāmo. Darba noslēgumā apkopotas visas atziņas, akcentējot risinājuma priekšrocības un iespējamos ierobežojumus. Tas noslēdzas ar ieteikumiem par gaidāmo izpēti un progresu modeļvadāmas lietotāja saskarnes projektēšanas jomā.

Promocijas darbā veikts detalizēts pētījums par lietotāja saskarnes izstrādes pilnveidošanu, izmantojot modeļvadāmu metodi. Pētījumā piedāvāts risinājums, kas apvieno abstraktu modelēšanu ar reālu ieviešanu, izmantojot precīzus transformācijas likumus, lai uzlabotu programmatūras lietotāja saskarņu izstrādi, viendabīgumu un lietošanas vienkāršību.

Promocijas darba rezultāti

1. Apkopota informācija par tīmekļa lietojumu priekšgala programmēšanas satvaros esošajiem elementu klāstiem, kas bija pamatojums izveidot lietotāja saskarnes elementu taksonomiju transformācijas likumu mērķa elementiem.
2. Noteikti lietotāja saskarnes metamodeļa parametri un izveidots metamodelis, kuru izmantojot veikt transformācijas no divpusložu modeļa, papildinot procesu ar lietotāja saskarnes šabloniem.
3. Veikta divpusložu modeļa notācijas papildināšana procesu modeļa metadatos, lai marķētu procesus, kas jāpakļauj lietotāja saskarnes izveides procesam un kas neattiecas uz prototipa izveidi, kā arī piedāvāts ieviest papildu koncepta modeļa elementu atribūtu datu tipus.

4. Definēti transformācijas likumi, kas, izmantojot kopā ar lietotāja saskarnes šablonu bibliotēku un šablonu atlases metodi, var tikt palaisti, lai automātiski ģenerētu lietotāja saskarnes prototipus.
5. Piedāvāts risinājums automātiskajai lietotāja saskarnes ģenerēšanai, kas ietver komponentus saskaņā ar modeļvadāmās inženierijas pamatprincipiem.
6. Demonstrētas piedāvātā risinājuma praktiskās lietošanas iespējas, izmantojot to programmatūras sistēmas priekšgala komponentu izveidošanai.
7. Ieskicētas divpusložu modeļa atbalsta rīka *BrainTool* attīstīšanas un bagātināšanas iespējas ar tādām funkcijām kā lietotāja saskarnes šabloniem, procesu izvēles prognozēšanu, saistot to ar lietotāja saskarnes šablonu bibliotēkas elementu saiknēm, utt.

Paveiktais darbs un iegūtie rezultāti dod pamatojumu apgalvot, ka darba ievadā definētas tēzes ir apstiprinātas.

1. No RTU izstrādātā divpusložu modeļa ir iespējams ģenerēt tīmekļa lietotnes lietotāja saskarnes prototipus, izmantojot modeļvadāmas izstrādes pamatkoncepcijas, kas ir modeļi un modeļu transformācijas. Par to liecina izstrādātais risinājums un tā demonstrācija, izmantojot aprobācijas piemēru.
2. Tīmekļa lietotnes lietotāja saskarnes prototipa automātiskā iegūšana no biznesa līmeņa modeļiem paātrina tīmekļa lietotnes izstrādes procesu, nezaudējot rezultāta kvalitāti. Par to liecina transformācijas rezultāta analīze gan iegūtā koda kvalitātes ziņā, gan iegūtā tīmekļa lietojuma atbilstība problēmvides pieņemšanas kritērijiem.

Balstoties promocijas darba izstrādes gaitā veiktajos pētījumos un iegūtajos rezultātos, ir definēti vairāki secinājumi.

1. Mūsdienās modeļvadāmajā inženierijā ir ierasts izmantot *UML* valodu artefaktu definēšanā, tomēr tā ir vairāk gatavu risinājumu aprakstam un var būt sarežģīti izprotama problēmvides ekspertiem. Tomēr, veicot pētījumu, var secināt, ka divpusložu modelis ir vairāk piemērots lietotāja saskarnes prototipu ģenerēšanai.
2. Divpusložu modeļiem ir jābūt kvalitatīviem un strikti definētiem, kā arī lietotāja saskarnes šabloniem un to bibliotēkai ir jābūt kvalitatīvām. Savukārt pats šablonu kods ir pareizi jāuzraksta, lai būtu iespējams ģenerēt lietotāja saskarnes prototipus.
3. Divpusložu modeļa notācija gandrīz pilnībā nodrošina lietotāja saskarnes ģenerēšanu, tomēr ir nepieciešams papildināt notāciju: koncepta modelī – ar papildu lauku tipa vērtībām; procesu modelī – ar to, vai konkrēts process ir lietotājam redzams, vai nē.
4. Var papildināt *BrainTool* ar lietotāja saskarnes šabloniem un procesu prognozēšanu. Piemēram, izveidojot jaunu procesu, tiktu piedāvāts automātiski izveidot tam sekojošus procesus, ja šāda informācija glabātos lietotāja saskarnes šablonu bibliotēkā.

Praktiskā nozīme. Izstrādāto risinājumu ir ieteikts lietot tīmekļa lietotņu lietotāja saskarņu izstrādei, automātiski ģenerējot priekšgala komponentu pirmkodu prototipiem. Šīs pieejas mērķis ir risināt programmatūras izstrādes izplatītas problēmas, piemēram, projektu budžeta

pārtēriņu un projektu realizēšanas termiņu problēmas, izmantojot modeļvadāmās inženierijas principus, lai uzlabotu izstrādes efektivitāti un nodrošinātu tās kvalitāti.

Definējot mērķa metamodeli un transformācijas likumus, risinājums ļauj automātiski ģenerēt lietotāja saskarnes prototipus no augsta līmeņa abstraktiem modeļiem. Tas samazina manuālās programmēšanas piepūli, paātrina izstrādes procesu un samazina cilvēku kļūdas.

Standartizētu šablonu un transformēšanas likumu izmantošana nodrošina, ka ģenerētās lietotāja saskarnes ir konsekventas dizainā un atbilst paraugpraksei, tādējādi nodrošinot augstāku kvalitāti un vieglāk uzturamu kodu.

Darbā piedāvātā risinājuma mērķis ir racionalizēt lietotāja saskarnes prototipu izstrādes procesu, padarot to ātrāku, efektīvāku un spējīgu radīt augstas kvalitātes, uz lietotāju orientētas saskarnes. Šī pieeja ne tikai risina dažus ieilgušos programmatūras inženierijas izaicinājumus, bet arī veido pamatu turpmākiem sasniegumiem modeļos balstītā lietotāja saskarnes izstrādē.

Modificējot esošo divpusložu modeļa notāciju un adaptējot to transformācijām lietotāja saskarnes prototipu pirmkodā, promocijas darbā izstrādāto prototipu ģenerēšanas algoritmu pēc tā implementācijas attiecīgajā atbalsta rīkā var ieviest industrijā, jo algoritms dot iespēju no problēmvides ekspertiem saprotamā modeļa iegūt lietotāja saskarnes prototipu pirmkodu.

Pētījuma gaitā definētie spriedumi un iegūtie rezultāti var būt interesanti arī plaša loka speciālistiem gan industrijā, gan, turpinot zinātniskos pētījumus, modeļvadāmās programmatūras jomā. Promocijas darbā izstrādātais risinājums nodrošina programkoda inženierijas uzdevuma risināšanu vienā virzienā no problēmvides modeļa uz kodu. Piedāvātā risinājuma arhitektūra ļauj pētīt arī koda reinženierijas problēmas risināšanu, tas ir, nodrošināt iespēju uzturēt kodā veiktās izmaiņas attiecībā uz modeļa saturu. Šajā promocijas darbā koda reinženierijas problēmas risināšana ir ārpus pētījuma tvēruma.

Vēl viens nākotnes pētījuma virziens šim darbam būtu esošā rīka – *BrainTool* – papildināšana ar darbā apskatīto pieeju vai arī jauna redaktora izveide promocijas darba pieejas realizēšanai lietotājam draudzīgā vidē. *BrainTool* rīku varētu papildināt ar iespēju lietotājam, veidojot divpusložu modeļa procesus, automātiski piedāvāt, kādi būtu nākamie procesi un attiecīgi – lietotāja saskarnes šabloni. Papildus tam varētu papildināt lietotāja saskarnes šablonu sarakstu ar definīcijām un dažādiem lietotāju saskarnes satvariem, lai varētu ģenerēt prototipus plašākā tehnoloģiju spektrā.

Nozīmīgāks pētījums būtu izmantot mākslīgā intelekta piedāvātās iespējas, lai automātiski izveidotu un savstarpēji savienotu nepieciešamos šablonus no lietotāja saskarnes bibliotēkas aprakstu, tādējādi samazinot manuālo darbu lietotāja saskarnes šablonu izstrādei.

Ņemot vērā to, ka patlaban tiek plaši attīstīti dažādi lietotāja saskarnes satvari, tādi kā *Material Design*, *Ant Design* utt., būtu vērtīgi veikt pētījumus, kā var izmantot šo satvaru semantiku un citas pieejas, lai automātiski izveidotu vēlamos lietotāja saskarnes elementus, tikai norādot, kādu lietotāja saskarnes satvaru izmantot.

BIBLIOGRĀFIJA

Akiki P., Bandara A., Yu Y. (2014) Adaptive Model-Driven User Interface Development Systems. ACM Comput. Surv. 47, 1, Article 9 (July 2014), 33 pages. DOI: https://doi.org/10.1145/2597999.

Alfaridzi M. D., Yulianti L. P. (2020) UI-UX Design and Analysis of Local Medicine and Medication Mobile-based Apps using Task-Centered Design Process, 2020 International Conference on Information Technology Systems and Innovation (ICITSI), 2020, pp. 443–450, DOI: https://doi.org/10.1109/ICITSI50517.2020.9264947.

Al-Saqqa S., Sawalha S., Abdel-Nabi H. (2020) Agile Software Development: Methodologies and Trends. International Journal of Interactive Mobile Technologies (iJIM), 14, 246, DOI: https://doi.org/10.3991/ijim.v14i11.13269.

Antović I., Vlajić S., Milić M., Savic D. (2012) Model and software tool for automatic generation of user interface based on use case and data model, IET Software, 6, DOI: https://doi.org/10.1049/iet-sen.2011.0060.

Aspray W., Keil R., Parnas D. (1999) History of Software Engineering.

Babris K., Nikiforova O. (2024a) Towards Automated UI Mockup Generation from Two-Hemisphere Problem Domain Models: A Conceptual Framework and Approach, Proceedings of 19th Iberian Conference on Information Systems and Technologies (CISTI 2024, June 25–28), 2024 (accepted for publication) (SCOPUS).

Babris K., Nikiforova O. (2024b) From Models to Interfaces: Leveraging the Two-Hemisphere Model for Automated UI Generation, 2024 IEEE 65th International Scientific Conference on Information Technology and Management Science of Riga Technical University (ITMS), Riga, Latvia, 2024, pp. 1–6, submitted for publication (SCOPUS).

Babris K., Nikiforova O., Sukovskis U. (2019) Brief Overview of Modelling Methods, Life-Cycle and Application Domains of Cyber-Physical Systems. Applied Computer Systems, 2019, Vol. 24, Issue 1, pp. 1–8, DOI: https://doi.org/10.2478/acss-2019-0001 (Web of Science).

Bajammal M., Davood M., Ali M. (2018) Generating reusable web components from mockups. In Proceedings of the 33rd ACM/IEEE International Conference on Automated Software Engineering (ASE '18). Association for Computing Machinery, New York, NY, USA, 601–611, DOI: https://doi.org/10.1145/3238147.3238194.

Bajovs A., Nikiforova O., Sējāns J. (2013) Code Generation from UML Model: State of the Art and Practical Implications. Scientific Journal of Applied Computer Systems, 14, 7–18, DOI: https://doi.org/10.2478/acss-2013-0002.

Beek (ter) M. H., McIver A. (2021) Formal methods: practical applications and foundations. Form Methods Syst Des 58, 1–4, DOI: https://doi.org/10.1007/s10703-021-00380-6.

Beltramelli T. (2018) Pix2code: Generating Code from a Graphical User Interface Screenshot. In Proceedings of the ACM SIGCHI Symposium on Engineering Interactive Computing Systems (EICS '18). Association for Computing Machinery, New York, NY, USA, Article 3, 1–6, DOI: https://doi.org/10.1145/3220134.3220135.

Briand L. C., Labiche Y., Lin Q. (2005) Improving statechart testing criteria using data flow information, 16th IEEE International Symposium on Software Reliability Engineering (ISSRE'05), Chicago, IL, USA, 2005, pp. 10–104, DOI: https://doi.org/10.1109/ISSRE.2005.24.

Calvary G., Coutaz J., Thevenin D., Limbourg Q., Bouillon L., Vanderdonckt J. (2003) A Unifying Reference Framework for multi-target user interfaces, Interacting with Computers, Volume 15, Issue 3, June 2003, pp. 289–308, DOI: https://doi.org/10.1016/S0953-5438(03)00010-9.

Diehl C., Martins A., Almeida A., Silva T., Ribeiro Ó., Santinha G., Rocha N., Silva AG. (2022) Defining Recommendations to Guide User Interface Design: Multimethod Approach. JMIR Hum Factors. 2022 Sep 30, 9(3), e37894, DOI: http://doi.org/10.2196/37894.

Domingo A., Echeverría J., Pastor O., Cetina C. (2020) Evaluating the Benefits of Model-Driven Development: Empirical Evaluation Paper. DOI: https://doi.org/10.1007/978-3-030-49435-3_22.

Escalona M. J., García-Borgoñón, L., Koch, N. (2021) Don't Throw your Software Prototypes Away. Reuse them! In Insfran, E., González, F., Abrahão, S., Fernández, M., Barry, C., Linger, H., Lang, M., Schneider, C. (Eds.), Information Systems Development: Crossing Boundaries between Development and Operations (DevOps) in Information Systems (ISD2021 Proceedings). Valencia, Spain: Universitat Politècnica de València.

Fernández E., Liu Y., Pan R. (2001) Patterns for Internet shops. Available: https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=8ad2c6f226600828dab4e5317ff713603e003383.

Gharaat M., Sharbaf M., Zamani B. et al. (2021) ALBA: a model-driven framework for the automatic generation of android location-based apps. Autom Softw Eng 28, 2, DOI: https://doi.org/10.1007/s10515-020-00278-3.

Gilbert D., Fischer H., Röder D. (2021) UX at the Right Level: Appropriately Plan the UX Expertise Using the PUXMM – A UX Maturity Model for Projects. i-com, Vol. 20, Issue 1, pp. 105–113, DOI: https://doi.org/10.1515/icom-2020-0029.

Grønmo R., Møller-Pedersen B. (2011) From UML 2 Sequence Diagrams to State Machines by Graph Transformation, Journal of Object Technology, Vol. 10, pp. 8:1–22, DOI: http://doi.org/10.5381/jot.2011.10.1.a8.

Gusarovs K. (2020) Metodes izstrāde koda ģenerēšanai no divpusložu modeļa, promocijas darbs, Rīgas Tehniskā universitāte, 2020, DOI: https://doi.org/10.7250/9789934225772.

Hailpern B., Tarr P. (2006) Model-driven development: The good, the bad, and the ugly, in IBM Systems Journal, vol. 45, no. 3, pp. 451–461, DOI: https://doi.org/10.1147/sj.453.0451.

Harel D. (1987) Statecharts: a visual formalism for complex systems, Science of Computer Programming, Volume 8, Issue 3, 1987, pp. 231–274, ISSN 0167-6423, DOI: https://doi.org/10.1016/0167-6423(87)90035-9.

Hasan H. M., Sanyal F., Chaki D., Ali Md. (2017) An empirical study of important keyword extraction techniques from documents. DOI: https://doi.org/10.1109/ICISIM.2017.8122154.

Hasheminejad S. M. H., Jalili S. (2012) Design patterns selection: An automatic two-phase method. Journal of Systems and Software, 85, 408–424, DOI: https://doi.org/10.1016/j.jss.2011.08.031.

Haz A. L., Nobuo F., Fajrianti E. D., Sukaridhoto S. (2024) A Study of Summarization and Keyword Extraction Function in Meeting Note Generation System from Voice Records. In Proceedings of the 2023 12th International Conference on Networks, Communication and Computing (ICNCC '23). Association for Computing Machinery, New York, NY, USA, 106–112, DOI: https://doi.org/10.1145/3638837.3638853.

Hinderks A., Mayo F. J. D., Thomaschewski J., Escalona M. J. (2022) Approaches to manage the user experience process in Agile software development: A systematic literature review, Information and Software Technology, Volume 150, 2022, 106957, ISSN 0950-5849, DOI: https://doi.org/10.1016/j.infsof.2022.106957.

Horrocks I. (1999) Constructing the User Interface with Statecharts (1st. ed.). Addison-Wesley Longman Publishing Co., Inc., USA, ISBN:978-0-201-34278-9.

Hussmann H., Meixner G., Zühlke D. (2011) Model-Driven Development of Advanced User Interfaces, DOI: https://doi.org/10.1007/978-3-642-14562-9.

ISO/IEC/IEEE 29119 Software Testing Standards, International Organization for Standardization (ISO), International Electrotechnical Commission (IEC), Institute of Electrical and Electronics Engineers (IEEE), 2013.

Johnson G., Gross M., Hong J., Do E. (2009) Computational Support for Sketching in Design: A Review. Foundations and Trends in Human-Computer Interaction, 2, 1–93, DOI: http://doi.org/10.1561/1100000013.

Joo H. (2017) A Study on Understanding of UI and UX, and Understanding of Design According to User Interface Change, in International Journal of Applied Engineering Research ISSN 0973-4562, vol. 12, no. 20, pp. 9931–9935, 2017.

Kleppe A. G., Warmer J., Bast W. (2003) MDA Explained: The Model Driven Architecture: Practice and Promise. Addison-Wesley Longman Publ., USA. ISBN: 978-0-321-19442-8.

Koch N., Mandel L. (1999) Using UML to design hypermedia applications. Technical Report 9901, Institut für Informatik, Ludwig-Maximilians-Universität, München, March 1999.

Kompaniets V., Lyz A., Kazanskaya A. (2020) An Empirical Study of Goal Setting in UX/UI-design, 2020 IEEE 14th International Conference on Application of Information and Communication Technologies (AICT), 2020, pp. 1–5, DOI: https://doi.org/10.1109/AICT50176.2020.9368570.

Kozačenko L. (2014) Divpusložu modeļa lietošanas analīze UML diagrammu ģenerēšanā, Maģistra c, Rīgas Tehniskā universitāte, 2014.

Kruchten P. (2003) The Rational Unified Process: An Introduction (3rd. ed.). Addison-Wesley Longman Publishing Co., Inc., USA, ISBN: 978-0-321-19770-2.

Leuthold, S. (2010) User Interface, Navigation Design and Content Representation: Three Perspectives on World Wide Web Navigation. Doctoral Thesis, University of Basel, Switzerland.

Li J., Cao H., Lin L., Hou Y., Zhu R., El Ali A. (2023) User Experience Design Professionals' Perceptions of Generative Artificial Intelligence, DOI: https://doi.org/10.48550/arXiv.2309.15237.

Lycett M., Marcos E., Storey V. (2007) Model-driven systems development: an introduction. European Journal of Information Systems, 16(4), 346–348. DOI: https://doi.org/10.1057/palgrave.ejis.3000684.

Mahatody T., Ilie M., Rapatsalahy M. A., Dimbisoa W. G., Ilie S. (2021) Metamodel based approach to generate user interface mockup from UML class diagram, Procedia Computer Science, Volume 184, 2021, Pages 779–784, ISSN 1877-0509, DOI: https://doi.org/10.1016/j.procs.2021.03.096.

Marzuoki (el) N. (2021) Model Composition in Multi-Modeling Approaches Based on Model Driven Architecture, PhD Thesis, la Faculte des Sciences Dhar El Maharaz Fes, Marocco.

Mbugua S. T., Korongo J., Samuel M. (2022) On Software Modular Architecture: Concepts, Metrics and Trends, International Journal of Computer and Organization Trends, vol. 12, no. 1, Jan–Apr. 2022, pp. 3–10, DOI: 10.14445/22492593/IJCOT-V12I1P302.

Molina P., Meliá S., Pastor O. (2002) User Interface Conceptual Patterns, 159–172, DOI: http://doi.org/10.1007/3-540-36235-5_12.

Nacheva R. (2017) Prototyping Approach In User Interface Development, 2nd Conference on Innovative Teaching Methods, Bulgaria, 28–29 June, 2017, 80–87.

Narang P., Mittal P. (2022) Performance Assessment of Traditional Software Development Methodologies and DevOps Automation Culture. Engineering, Technology & Applied Science Research, 12, 9726–9731, DOI: https://doi.org/10.48084/etasr.5315.

Nasiri S., Rhazali Y., Adadi A., Lahmer M. (2023) Generation of User Interfaces and Code from User Stories. In: Joshi, A., Mahmud, M., Ragel, R. G. (eds) Information and Communication Technology for Competitive Strategies (ICTCS 2021). Lecture Notes in Networks and Systems, vol. 400. Springer, Singapore. DOI: https://doi.org/10.1007/978-981-19-0095-2_38.

Newman M. W., Landay J. A. (2000) Sitemaps, storyboards, and specifications: a sketch of Web site design practice. In Proceedings of the 3rd conference on Designing interactive systems: processes, practices, methods, and techniques (DIS '00). Association for Computing Machinery, New York, NY, USA, 263–274. DOI: https://doi.org/10.1145/347642.347758.

Nguyen T. D., Vu P., Pham H., Nguyen T. (2018) Deep Learning UI Design Patterns of Mobile Apps, 2018 IEEE/ACM 40th International Conference on Software Engineering: New Ideas and Emerging Technologies Results (ICSE-NIER), Gothenburg, Sweden, pp. 65–68, DOI: https://doi.org/10.1145/3183399.3183422.

Nikiforova O., Gusarovs K. (2020) Anemic Domain Model vs Rich Domain Model to Improve the Two-Hemisphere Model-Driven Approach, Applied Computer Systems, 25(1), 51–56, DOI: https://doi.org/10.2478/acss-2020-0006.

Nikiforova O., Kirikova, M. (2004) Two-Hemisphere Model Driven Approach: Engineering Based Software Development. In: Persson, A., Stirna, J. (eds) Advanced Information Systems Engineering. CAiSE 2004. Lecture Notes in Computer Science, vol. 3084. Springer, Berlin, Heidelberg, DOI: https://doi.org/10.1007/978-3-540-25975-6_17.

Nikiforova O., Pavlova, N. (2011) Open Work of Two-Hemisphere Model Transformation Definition into UML Class Diagram in the Context of MDA. In: Software Engineering Techniques: Lecture Notes in Computer Science. Vol. 4980. Berlin: Springer Berlin Heidelberg, pp. 118–130. ISBN 9783642223853.

Nikiforova O. (2002) General Framework for Object-Oriented Software Development Process. Applied computer systems, Vol. 13, pp. 132–144, ISSN 1407-7493.

Nikiforova O. (2009). Two Hemisphere Model Driven Approach for Generation of UML Class Diagram in the Context of MDA. e-Informatica, 3, 59–72, https://www.e-informatyka.pl/attach/e-Informatica_-_Volume_3/eInformatica2009Art4.pdf.

Nikiforova O., Babris K., Kristapsons J. (2020) Survey on Risk Classification in Agile Software Development Projects in Latvia. Applied Computer Systems, Vol. 25, No. 2, pp. 105–116, ISSN 2255-8683, e-ISSN 2255-8691, DOI: https://doi.org/10.2478/acss-2020-0012 (Web of Science).

Nikiforova O., Babris K., Madelāne L. (2021) Expert Survey on Current Trends in Agile, Disciplined and Hybrid Practices for Software Development, Applied Computer Systems, vol. 26, no. 1, 2021, pp. 38–43, DOI: https://doi.org/10.2478/acss-2021-0005 (Web of Science).

Nikiforova O., Babris K., Mahmoudifar F. (2024a) Automated Generation of Web Application Front-End Components from User Interface Mockups, Proceedings of International Conference on Software Technologies (ICSOFT 2024, July 8–10), SCITEPRESS Digital Library, 2024, pp. 100–111, DOI: 10.5220/0012759500003753 (SCOPUS).

Nikiforova O., Babris K., Guliyeva A. (2024b) Definition of a Set of Use Case Patterns for Application Systems: A Prototype-Supported Development Approach. Applied Computer Systems, Vol. 29, No. 1, 2024, pp. 59–67, DOI: 10.2478/acss-2024-0008 (*Web of Science*).

Nikiforova O., El Marzouki N., Gusarovs K., Vangheluwe H., Bures T., Al-Ali R., Iacono M., Orue Esquivel P., and Leon F. (2017) The Two-Hemisphere Modelling Approach to the Composition of Cyber-Physical Systems" – Proceedings of International Conference on Software Technologies (ICSOFT 2017), 24–26 July, 2017, Madrid, Spain. SCITEPRESS Digital Library, pp. 286–293, DOI: https://doi.org/10.5220/0006424902860293.

Nikiforova O., Iacono M., El Marzouki N., Romanovs A. and Vangheluwe H. (2020) Enabling Composition of Cyber-Physical Systems with the Two-Hemisphere Model-Driven Approach, In: Multi-Paradigm Modelling Approaches for Cyber-Physical Systems. B. Tekinerdogan, D. Blouin, H. Vangheluwe, M. Goulão, P. Carreira, V. Amaral ed. 125 London Wall, London EC2Y 5AS, United Kingdom: Academic Press is an imprint of Elsevier, 2020. pp. 149–168, ISBN 978-0-12-819105-7 (Scopus).

Nikiforova O., Kozacenko L. and Ahilcenoka D. (2013) UML Sequence Diagram: Transformation from the Two-Hemisphere Model and Layout, Applied Computer Systems, vol. 14, no. 1, pp. 31–41, DOI: https://doi.org/10.2478/acss-2013-0004.

Nikiforova O., Kozacenko, L., Ahilcenoka, D., Gusarovs, K., Ungurs, D., Jukss, M. (2015) Comparison of the Two-Hemisphere Model-Driven Approach to Other Methods for Model-Driven Software Development, Scientific Journal of Riga Technical University: Applied Computer Systems, 18, 5–14, DOI: http://doi.org/10.1515/acss-2015-0013.

Nikiforova O., Sukovskis U., Gusarovs K. (2015). Application of the two-hemisphere model supported by BrainTool: Football game simulation. AIP Conference Proceedings, 1648, DOI: https://doi.org/10.1063/1.4912557.

Nikiforova O., Zabiniako V., Kornienko J., Rizhko R., Babris K., Gasparoviča-Asīte M. (2021a) Efficiency Monitoring of Engineering System Designer Work Based on Multi-System User Behavior Analysis with AI/ML Algorithms, 2021 IEEE 62nd International Scientific Conference on Power and Electrical Engineering of Riga Technical University (RTUCON), 2021, pp. 1–6, DOI: https://doi.org/10.1109/RTUCON53541.2021.9711720 (SCOPUS).

Nikiforova O., Zabiniako V., Kornienko J., Rizhko R., Babris K., Nikulsins V., Garkalns P., Gasparoviča-Asīte M. (2021b) Solution to On-line vs On-site Work Efficiency Analysis on the Example of Engineering System Designer Work, Applied Computer Systems, vol. 26, no. 2, pp. 87–95, DOI: https://doi.org/10.2478/acss-2021-0011 (SCOPUS).

Osis J., Asnina E. (2011) Model-Driven Domain Analysis and Software Development: Architectures and Functions. 1 edition. Hershey – New York, USA: IGI Global, 2011, 514 p. ISBN13: 9781616928742, DOI: http://doi.org/10.4018/978-1-61692-874-2.

Pastor O., Abrahao S., Molina J.C., Torres I. (2001) A FPA-like Measure for Object Oriented Systems from Conceptual Models, Current Trends in Software Measument, Ed. Shaker Verlag, pages 51–69, Montreal, Canada, 2001.

Pavlova N. (2008) Platformneatkarīga modeļa izstrādes pieeja modeļvadāmas arhitektūras ietvarā, promocijas darbs, Rīgas Tehniskā universitāte, 2008.

Pelechano V., Pastor O., Insfrán E. (2002) Automated code generation of dynamic specializations: an approach based on design patterns and formal techniques, Data & Knowledge Engineering, Volume 40, Issue 3, pp. 315–353, ISSN 0169-023X, DOI: https://doi.org/10.1016/S0169-023X(02)00020-4.

Pimentel J., Castro J., Mylopoulos J., Angelopoulos K., Souza V. S. (2014) From requirements to statecharts via design refinement. Proceedings of the ACM Symposium on Applied Computing, DOI: https://doi.org/10.1145/2554850.2555056.

Planas E., Daniel G., Brambilla M. et al. (2021) Towards a model-driven approach for multiexperience AI-based user interfaces. Softw Syst Model 20, 997–1009, DOI: https://doi.org/10.1007/s10270-021-00904-y.

Riaz S., Arshad A., Band S. S., & Mosavi A. (2022) Transforming Hand Drawn Wireframes into Front-End Code with Deep Learning, Computers, Materials & Continua, 72, 4303–4321, DOI: https://doi.org/10.32604/cmc.2022.024819.

Riccio V., Jahangirova G., Stocco A. et al. (2020) Testing machine learning based systems: a systematic mapping. Empir Software Eng 25, 5193–5254, DOI: https://doi.org/10.1007/s10664-020-09881-0.

Rivero J., Rossi G., Grigera J., Luna R. E., Navarro A. (2011) From Interface Mockups to Web Application Models, 6997, 257–264, DOI: http://doi.org/10.1007/978-3-642-24434-6_20.

Rudd J., Stern K., Isensee S. (1996) Low vs. high-fidelity prototyping debate. interactions 3, 1 (Jan. 1996), 76–85, DOI: https://doi.org/10.1145/223500.223514.

Sharp H., Rogers Y., Preece J. (2019) Interaction Design: Beyond Human Computer Interaction (5th Edition), ISBN: 9781119547259, Wiley, 2019.

Silva (da) T. S., Martin A., Maurer F., Silveira M. (2011) User-Centered Design and Agile Methods: A Systematic Review, 2011 Agile Conference, 2011, pp. 77–86, DOI: https://doi.org/10.1109/AGILE.2011.24.

Stige Å., Zamani E. D., Mikalef P., Zhu Y. (2023) Artificial intelligence (AI) for user experience (UX) design: a systematic literature review and future research agenda, Information Technology & People, DOI: https://doi.org/10.1108/ITP-07-2022-0519.

Stompff G., Smulders F. (2015) The Right Fidelity: Representations That Speed Up Innovation Processes. Design Manag J, 10, 14–26, DOI: https://doi.org/10.1111/dmj.12019.

Suleri S., Pandian V. P. S., Shishkovets S., Jarke M. (2019) Eve: A Sketch-based Software Prototyping Workbench. In Extended Abstracts of the 2019 CHI Conference on Human Factors in Computing Systems (CHI EA '19). Association for Computing Machinery, New York, NY, USA, Paper LBW1410, 1–6, DOI: https://doi.org/10.1145/3290607.3312994.

Sunitha E. V., Samuel P. (2019) Automatic Code Generation From UML State Chart Diagrams, in IEEE Access, vol. 7, pp. 8591–8608, 2019, DOI: https://doi.org/10.1109/ACCESS.2018.2890791.

Thomas D. (2004) MDA: Revenge of the Modelers or UML Utopia? IEEE Softw. 21, 3 (May 2004), 15–17, DOI: https://doi.org/10.1109/MS.2004.1293067.

Trehan V., Chapman C., Raju P. (2015) Informal and formal modelling of engineering processes for design automation using knowledge based engineering. J. Zhejiang Univ. Sci. A 16, 706–723, DOI: https://doi.org/10.1631/jzus.A1500140.

Uyanık B., Sayar A. (2024) Analysis and Comparison of Automatic Code Generation and Transformation Techniques on Low-Code Platforms. In Proceedings of the 2023 5th International Conference on Software Engineering and Development (ICSED '23). Association for Computing Machinery, New York, NY, USA, 17–27, DOI: https://doi-org.resursi.rtu.lv/10.1145/3637792.3637795.

Virzi R. A., Sokolov J. L., Karis D. (1996) Usability problem identification using both low- and high-fidelity prototypes. In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '96). Association for Computing Machinery, New York, NY, USA, 236–243, DOI: https://doi.org/10.1145/238386.238516.

Völter M., Bettin J. (2004) Patterns for Model-Driven Software-Development. Proceedings of the 9th European Conference on Pattern Languages of Programms (EuroPLoP '2004), Irsee, Germany, July 7-11, 2004, 525–560.

Wellner P. D. (1989) Statemaster: A UIMS based on statecharts for prototyping and target implementation. SIGCHI Bull. 20, SI (March 1989), 177–182, DOI: https://doi.org/10.1145/67450.67486.

Yigitbas E., Jovanovikj I., Biermeier K. et al. (2020) Integrated model-driven development of self-adaptive user interfaces. Softw Syst Model 19, 1057–1081, DOI: https://doi.org/10.1007/s10270-020-00777-7.

Kristaps Babris dzimis 1986. gadā Rīgā. Rīgas Tehniskajā universitātē ieguvis bakalaura (2015) un maģistra (2018) grādu datorsistēmās ar programmēšanas inženiera kvalifikāciju. Kopš 2012. gada strādā SIA "*IT Sapiens*", ieņemot IT struktūrvienības vadītāja amatu, un kopš 2016. gada strādā Rīgas Tehniskajā universitātē zinātniskā asistenta amatā. Zinātniskās intereses ir saistītas ar modeļvadāmas inženierijas principu lietošanu tīmekļa sistēmu izstrādē.