LATVIJAS UNIVERSITĀTE
DATORIKAS FAKULTĀTE

ZANE BIČEVSKA

VIEDĀS TEHNOLOĢIJAS UN TO EFEKTIVITĀTE

Promocijas darbs datorzinātņu doktora (Dr.sc.comp.)
zinātniskā grāda iegūšanai

Nozare: datorzinātnes
Apakšnozare: datu apstrādes sistēmas un datortīkli

Zinātniskais vadītājs:
Profesors, Dr. hab. dat. JURIS BORZOVS

RĪGA 2010

Satura rādītājs

ANOTĀCIJA

Promocijas darbs ir veltīts viedajām tehnoloģijām - idejai, kuras pamatā ir doma par "intelektuālu" komponenšu iekļaušanu programmatūrā. Viedās tehnoloģijas ir nefunkcionālo programmatūras īpašībām atbilstoši risinājumi, kas tiek iebūvēti programmatūrā un būtiski uzlabo programmatūras uzturēšanu, papildināšanu un izplatīšanu.

Analizējot pasaules vadošo programmatūras ražotāju radniecīgus pētījumus, darbā akcentēta viedo tehnoloģiju ideoloģiskā novitāte un praktiskā ievirze. Dots pārskats par viedo tehnoloģiju komponenšu praktiskām realizācijām, apkopotas reālos projektos gūtās atziņas.

Darbā autore definē kritērijus, kuri kalpo viedo tehnoloģiju pielietošanas efektivitātes novērtēšanai. Dots ieskats kvalitatīvo pētījumu ideoloģijā, kā arī aprakstīts autores veikts kvalitatīvs pētījums par viedo tehnoloģiju pielietošanu un iegūtie rezultāti.

Atslēgas vārdi: viedās tehnoloģijas, efektivitāte, programmatūras uzturēšana.

ABSTRACT

The paper proposes an idea of including the “intellectual” components into software solutions in order to improve the latter. Smart technologies are built-in components that are conform to non-functional requirements for better software maintenance, development and distribution.

The paper contains an overview of similar research trends of leading IT manufacturers. The innovative character and practical evidence of smart technologies are emphasized.

Based on the experience acquired during projects on software development and distribution, the author proposes criteria for evaluation of effectiveness that can be used to support decision making on implementing the smart technologies.

There is an insight into the principles of qualitative research methods as well as an overview of the qualitative research on smart technologies done by the author.

Keywords: smart technologies, effectiveness, software maintenance

IEVADS

Pēdējais ceturtdaļgadsimts informācijas tehnoloģiju nozares attīstībā raksturojas ar to, ka programmatūra kā produkts ir kļuvusi nesalīdzināmi apjomīgāka, daudzpusīgāka un sarežģītāka nekā tā bija IT pirmsākumos [1]. „Monolītās" (viengabalainās) informācijas sistēmas ir nomainījusi klientu-serveru arhitektūra, kuru savukārt pašlaik nomaina n-slāņu arhitektūra. Atsevišķi darbināmās (*stand-alone*) sistēmas tiek iekļautas integrētās un interaktīvās sistēmās (*enterprise application integration*). Programmatūra ir iekļauta gandrīz katrā iespējamā iekārtā (telefonos, televizoros, sadzīves tehnikā, automašīnās u.t.t.), līdz ar to pat vidusmēra cilvēks katru dienu saskaras ar neskaitāmām programmatūras darbinātām ierīcēm. Ikdienā lietotās interneta informācijas sistēmas, piemēram, *eCommerce* un *eMarketplace* risinājumi, ir tālu pārsniegušas vienkāršas informatīvās mājas lapas līmeni.

Sarežģītāka kļūst ne tikai lietojumprogrammu uzbūve; kritiskas kļūst arī informācijas sistēmu prasības – gan skaita, gan sarežģītības, gan veidu ziņā. Sistēmas prasību specificētāji vairs nedrīkst pilnībā koncentrēties tikai uz informācijas sistēmas funkcionālajām prasībām [2], kas nosaka konkrētās informācijas sistēmas darbības uzdevumus un funkcijas. Arvien lielāku lomu informācijas sistēmu dzīvē spēlē nefunkcionālās prasības – no konkrētās informācijas sistēmas primārajiem uzdevumiem neatkarīgas tehniskās īpašības, kā lietojamība, veiktspēja, drošība, uzturamība, mērogojamība, testējamība un citas.

Diemžēl bieži vien nefunkcionālo prasību identificēšanai un implementēšanai tiek piešķirta pakārtota loma - daļēji ierobežoto resursu dēļ, bet daļēji arī nezināšanas vai nevēlēšanās "lieki sarežģīt risināmo uzdevumu" dēļ. Praksē nereti nākas sastapties ar gadījumiem, kad funkcionālās prasības, kas attiecas uz informācijas sistēmā uzglabājamo informāciju, darījumprocesiem (*business processes*), datu struktūrām u.tml., ir sīki un detalizēti aprakstītas prasību specifikācijā, bet nefunkcionālās prasības – piemēram, operacionālā pieejamība, veiktspēja, sadarbības iespējas, mērogojamība, drošība u.c. - ne vien nav kvantitatīvi raksturotas, bet dažkārt pat nav identificētas un attiecīgi nav iekļautas prasību uzskaitījumā. Un tas izstrādē rada būtiskus riskus, jo tieši nefunkcionālo prasību izpilde daudzos projektos izrādās par "aisberga" neredzamo daļu, kas atstāj lielāku iespaidu uz izmaksām, sistēmas arhitektūru un projekta laika grafiku nekā pamata funkcionalitāte.

Papildus sarežģītību mūsdienu informācijas tehnoloģiju pasaulē ienes dažādo lietojumprogrammu savstarpējās sadarbības dimensija. Mūsdienās vairs tikpat kā nevaram runāt par atsevišķu informācijas sistēmu lietošanu, kas nodrošinātu konkrētu darījumprasību (*business requirements*) izpildi, bet saskaramies ar veseliem informācijas sistēmu kompleksiem, kas ietver dažādas, bieži vien savstarpēji nesaskaņotas programmatūras, aparatūras un datu pārraides komponentes lietojumu. Dažādo komponenšu attīstība notiek strauji un nesaskaņoti, tādējādi radot potenciālus "konfliktus" (nesaskaņotības, atteices u.tml.) starp informācijas sistēmu daļām.

Promocijas darbs iepriekš minētajām problēmām piedāvā principiāli jaunus risinājumus. Tiek piedāvāts izstrādājamā informācijas sistēmā iekļaut komponentus (programmatūru, metainformāciju, skriptus un citus artefaktus), kas vismaz daļu no informācijas sistēmas kvalitātes aspektiem nodrošina nevis ar lietotāja palīdzību, bet gan automātiski – ar programmatūras palīdzību. Piedāvātie risinājumi, turpmāk saukti par viedajām tehnoloģijām, raksturojas ar savu universālo pieeju, kas, atšķirībā no nereti lietotiem individuāliem risinājumiem vienas informācijas sistēmas vajadzībām, ļauj tos pielietot daudzās sistēmās.

Promocijas darbā aprakstīto pētījumu priekšmets ir viedās tehnoloģijas – nefunkcionālo programmatūras īpašību kopumam atbilstoši risinājumi jeb viedo tehnoloģiju komponentes, kas tiek iebūvētas programmatūrā un uzlabo vai būtiski atvieglo šādi aprīkotas programmatūras uzturēšanu, papildināšanu un izplatīšanu.

Pētījuma galvenais mērķis bija definēt un specificēt viedo tehnoloģiju īpašības, radīt idejisko un tehnoloģisko bāzi vispār pielietojamu viedo tehnoloģiju principiem atbilstošu programmatūras komponenšu izstrādei, kā arī pārliecināties par ieguvumiem, kurus sniedz viedo tehnoloģiju izmantošana reālos informācijas tehnoloģiju risinājumos. Šī mērķa sasniegšanai tika izvirzīti vairāki uzdevumi:

- radniecīgu ideju identificēšana un pieejamo praktisko realizāciju izpēte,
- viedo tehnoloģiju jēdziena definēšana ar konkrētu pielietošanas piemēru palīdzību,
- viedo tehnoloģiju realizācijas arhitektūras izstrāde;
- viedo tehnoloģiju idejai atbilstošu komponenšu specificēšana un izstrāde;
- viedo tehnoloģiju aprobācija reālos IT projektos;
- viedo tehnoloģiju pielietošanas efektivitāti raksturojošu kritēriju izvēle;
- kvalitatīvs pētījums par viedo tehnoloģiju pielietošanas efektivitāti.

Autores veiktie pētījumi ietvēra viedo tehnoloģiju koncepta attīstības pilnu ciklu: idejas identificēšanu, viedo tehnoloģiju koncepcijas izstrādi, viedām tehnoloģijām atbilstošu komponenšu prasību definēšanu, izveidoto komponenšu praktisku pielietošanu un viedo tehnoloģiju izmantošanas efektivitātes analīzi.

Autores veiktie viedo tehnoloģiju pētījumi seko aktuālai pētījumu tendencei, kur programmatūru vairs netiek apskatīta kā atsevišķs, izolēts objekts, kas tiek radīts „dzīvei" tam paredzētā vidē un apstākļos un darbojas saskaņā ar izstrādes laikā tam uzstādītajām funkcionālajām prasībām, bet gan daudz plašāk – kā „attīstīta virtuāla būtne", kam piemīt „izdzīvošanai" virtuālajā vidē nepieciešamās papildus īpašības. Arvien vairāk pētījumu un izstrāžu priekšplānā izvirzās humānā ideja par programmatūras pieskaņošanos pieejamajai videi. Tā sasaucas ar pēdējā laikā populāro „digitālās ekosistēmas" [3] ideju, kuras pamatā doma par saudzīgu, humānu attieksmi pret „vidi", resp., par sistēmu kompleksiem, kas pieskaņojas vides iespējām un pieejamajiem resursiem un veido ekoloģisku digitālo pasauli.

Viedo tehnoloģiju koncepcijas pamatā [4] ir doma par „gudru" programmatūru, kas, līdzīgi kā saprātīga būtne, spēj reaģēt uz iepriekš neprognozējamu vidi un notikumiem. Viedo tehnoloģiju principiem atbilstoša programmatūra spēj adekvāti apstrādāt gan ārējus notikumus (no ārējas vides nākošus impulsus, piemēram, infrastruktūras izmaiņas, ārēju sistēmu izmaiņas), gan iekšējus notikumus (programmatūras uzbūve, funkcionēšana). Turklāt svarīga ir ne vien spēja diagnosticēt vides raksturlielumus [5], bet arī saskaņā ar paredzamajām vides izmaiņu sekām sevi atbilstoši pieskaņot izmainītajai videi.

Viedo tehnoloģiju orientētā pieeja nosaka, ka informācijas sistēmas funkcionalitāte tiek papildināta un integrēta ar virkni papildus iespēju, kas atbalsta un atvieglo programmatūras lietošanu, uzturēšanu un attīstību. Izstrādājot šīs iespējas, konsekventi tiek izmantotas izstrādātāju zināšanas par attiecīgās informācijas sistēmas iekšējo uzbūvi, turklāt informācijas sistēmā iekļautās viedo tehnoloģiju komponentes nekad netiek „nojauktas", resp., paliek sistēmā visu tās lietošanas laiku.

Autores pētījumu rezultāti, kas atspoguļoti promocijas darbā, ir pārliecinoši:

- ir identificēts un definēts viedo tehnoloģiju jēdziens;
- ir izstrādāta un publicēta viedo tehnoloģiju koncepcija;
- ir izstrādātas vairāku viedo tehnoloģiju komponenšu prasību specifikācijas;
- saskaņā ar definētajām prasībām ir realizētas viedo tehnoloģiju komponentes;
- izveidotās viedo tehnoloģiju komponentes ir aprobētas pilotprojektos;
- versiju izplatīšanas komponente tiek reāli ekspluatēta pie vairāk kā 600 lietotājiem visā Latvijas teritorijā.

Autores tiešais ieguldījums ir viedo tehnoloģiju idejas izveide, koncepcijas radīšana, viedo tehnoloģiju komponenšu prasību identificēšana un specificēšana, kā arī pētījumi par dažādiem viedo tehnoloģiju pielietošanas aspektiem, jo īpaši – par viedo tehnoloģiju pielietošanas efektivitāti.

Autores veikto pētījumu rezultāti ir apkopoti sešos zinātniskos rakstos [4, 5, 6, 7, 8, 9], kas publicēti starptautiski atzītos izdevumos, un prezentēti četrās starptautiskās zinātniskās konferencēs:

- The 8th International Conference, PROFES 2007, Riga, Latvia, July 2-4, 2007
- The 8th International Baltic Conference on Databases and Information Systems (Baltic DB&IS'2008), Tallin, Estonia, June 2-5, 2008
- The 2nd International Multi-Conference on Engineering and Technological Innovation (IMETI 2009), Orlando, Florida, USA, July 10-13, 2009
- The 4th IFIP TC2 Central and East European Conference on Software Engineering Techniques (CEE-SET 2009), Krakow, Poland, Oktober 12-14, 2009

Promocijas darba teorētiskā daļa sagatavota Eiropas Reģionālās attīstības fonda atklātā projektu konkursa prioritātes Nr. 2 „Uzņēmējdarbības un inovāciju veicināšana" pasākuma Nr.2.5 „Atbalsts lietišķās zinātnes attīstībai valsts zinātniskajās institūcijās" aktivitātes Nr.2.5.1. „Atbalsts lietišķajiem pētījumiem valsts zinātniskajās institūcijās" Latvijas Universitātes īstenotā projekta „Jaunu tehnoloģiju izstrāde informācijas sistēmu izveidei un integrācijai" (līguma Nr. VPD1/ ERAF/ CFLA/ 05/ APK/ 2.5.1./ 000048/ 024) ietvaros.

Promocijas darba praktiskās daļas sagatavošanā izmantotas Latvijas IT uzņēmumā SIA „Datorikas institūts DIVI" izstrādātas komponentes un materiāli. Viedo tehnoloģiju idejas attīstītas un aprobētas ERAF līdzfinansētos projektos:

- „Gudrās programmatūras tehnoloģijas implementācija notikumu orientētās informācijas sistēmās" (realizēts 2006.-2008. gadā Valsts atbalsta programmas „Atbalsts jaunu produktu un tehnoloģiju attīstībai" apakšprogrammā „Atbalsts jaunu produktu un tehnoloģiju attīstībai")
- „Programmatūras paštestēšanas tehnoloģija" (tiek realizēts 2009.-2011. gadā Valsts atbalsta programmas „Uzņēmējdarbība un inovācijas" apakšprogrammā „Jaunu produktu un tehnoloģiju izstrāde").

Darba 1. nodaļa veltīta citās valstīs veiktu idejiski radniecīgu zinātnisku un praktisku pētījumu apskatam. Galvenā loma atvēlēta ASV korporācijas IBM izvirzītajam autonomisko sistēmu konceptam, jo tas savā vīzijā un ambīcijās ir vistuvāk viedo tehnoloģiju konceptam, kaut arī tālu pārsniedz atsevišķu programmatūras uzlabojumu līmeni. Nodaļas noslēgumā viedo tehnoloģiju ideja pozicionēta attiecībā pret iepriekš aprakstītajām citu pētnieku izstrādnēm.

Darba 2. nodaļā definēts viedo tehnoloģiju jēdziens, raksturotas viedo tehnoloģiju īpašības un atbilstošās komponentes. Nodaļā par katru no piedāvātajām viedo tehnoloģiju īpašībām dots to izvēles pamatojums (problēmas nostādne) un aprakstīti konceptuāli ieteikumi viedo tehnoloģiju īpašību implementācijai informācijas sistēmās. Izklāstā apzināti nav iekļautas ar implementāciju saistītas specifiskas detaļas, nav publicēti viedo tehnoloģiju komponenšu projektējumi u.tml. Autore stāstījumā abstrahējusies no konkrētu platformu un risinājumu tehniskām detaļām, koncentrējoties uz viedo tehnoloģiju ideju un pielietošanas aspektiem.

Tā kā viedo tehnoloģiju jēdziens un īpašības ir attīstīti un aprobēti, balstoties uz reālu klientu projektos gūtām atziņām un veidotām iestrādnēm, 3. nodaļā raksturoti četri projekti-pieredzes avoti un sniegts ieskats šo projektu ietvaros gūtajās atziņās.

4. nodaļā iekļauti autores definēti efektivitātes novērtēšanas kritēriji, kuri var tikt izmantoti, lai novērtētu, cik efektīva būs viedo tehnoloģiju iekļaušana informācijas sistēmā. Galvenā uzmanība veltīta investīciju pamatotības analīzei, jo investīciju atdeve ir galvenais katra komerciāla risinājuma dzinējspēks. Lai parādītu izvēlēto efektivitāti raksturojošo parametru pielietošanu, apkopoti un publicēti četros iepriekšējā darba nodaļā raksturotos projektos uzkrāti dati, kas tiek izmantoti rādītāju skaitlisko vērtību aprēķināšanā un no tām izrietošā efektivitātes novērtēšanā.

Darbu noslēdzošajā, 5. nodaļā aprakstīti kvalitatīvo pētījumu pamatprincipi, kas nosaka izvēlētā fenomena (šinī gadījumā – viedo tehnoloģiju) izpēti, balstoties uz skaitliski nelielu, taču kompetentu un idejiski bagātu respondentu padziļinātu aptauju. Nodaļā aprakstīts autores veikts kvalitatīvs pētījums, kura ietvaros gūta atbilde uz jautājumu – pie kādiem nosacījumiem viedo tehnoloģiju izmantošana ir lietderīga un perspektīva?

1. TEORĒTISKĀ BĀZE

Arvien labāku informācijas sistēmu paplašināšanas un uzturēšanas problēmu risinājumu meklējumos aktīvi darbojas pasaules vadošo programmatūras izstrādātāju pētījumu laboratorijas. Redzamākais ražotāju pārstāvis šajā jomā ir ASV korporācija *IBM*, kura izsenis izceļas ar pētniecisku sniegumu mūsdienīgu ideju ģenerēšanā un atbilstošo risinājumu izstrādē. Līdzīgā virzienā pēta arī *Microsoft, Hewlett Packard* un citu programmatūras ražotāju finansēti pētnieki.

Pasaules vadošo programmatūras izstrādātāju zinātnisko un praktisko pētījumu uzdevums ir rast un izstrādāt „intelektuālus" risinājumus, kas padarītu informācijas sistēmas elastīgākas, vieglāk uzturamas, stabilākas. Tā kā viedo tehnoloģiju koncepcijas attīstībai izvirzītie mērķi ir līdzīgi, lai arī izvēlētais risinājums ir atšķirīgs, turpmākajās nodaļās sniegts ieskats literatūrā atrodamo radniecīgo pētījumu idejās, akcentējot sasniegumus un identificētās problēmas.

1.1 Autonomiskās sistēmas

IBM iniciatīva [10], kas sasaucas ar viedo tehnoloģiju idejām, saukta par autonomisko skaitļošanu (*autonomic computing*), tika aizsākta 2001. gadā. Tās galvenais izvirzītais mērķis [11] ir veidot informācijas sistēmas, kas ir spējīgas pašas sevi vadīt (*self-management*), un tādējādi pārvarēt barjeru starp lietotājiem un arvien sarežģītāko informāciju tehnoloģiju pasauli.

Šajā pētījumu virzienā galvenais uzsvars tiek likts uz sadalītām sistēmām [12], kas spējīgas pieskaņoties neparedzamām izmaiņām, kuras sistēmu lietotājiem ir nemanāmas/ nezināmas. Autonomiskās sistēmas spēj pašas pieņemt lēmumus, balstoties uz augsta līmeņa nosacījumiem („likumiem"), turklāt tās nepārtraukti pārbauda sevi un optimizējas, automātiski pieskaņojoties mainīgiem apstākļiem.

Literatūrā [13] uz autonomisko skaitļošanu tiek piedāvāts raudzīties kā uz neatkarīgu ("autonomu") komponenšu, kuras sadarbojas savā starpā, grupu. Autonomiskās komponentes tiek piedāvāts aprakstīt ar divu pamata vadības plūsmu palīdzību (lokālā un globālā), kurām pievienotas papildus komponentes. Pašuzraudzības (*self-monitoring*) vajadzībām tiek izmantoti sensori; sistēmas pašsakārtošanās (*self-adjustment*) vajadzībām – t.s. „efektori".

Papildus ir nepieciešama zināšanu bāze, kur uzkrāt informāciju par vidi un informācijas sistēmas iekšējo uzbūvi, kā arī pārvaldīt attiecīgos nosacījumus.

Balstoties uz autonomiskās skaitļošanas koncepciju, pasaulē tiek veidoti arhitektūras satvari, kas nodrošina „pašregulēšanos". Jo īpaši pēdējā laikā šis virziens aktualizēts multi-aģentu sistēmu kontekstā [14].

Tomēr vairums šo pētījumu pārsvarā koncentrējas uz komerciālo aspektu – uz sagaidāmajiem pārvaldības izmaksu ietaupījumiem, kas rastos centralizētu vai klastera bāzētu sistēmu gadījumā. Sarežģītu programmatūras kompleksu papildināšana ar inovatīviem servisiem nav pētījumu galvenais uzdevums.

Zemāk dots *IBM* piedāvātās pieejas vispārīgs raksturojums.

1.1.1 Nepieciešamības pamatojums

Korporācija *IBM* problēmas nostādnes raksturojumā [15] norāda uz pēdējos 20 gados novērotu datorsistēmu iespēju pieaugumu, pat eksponenciālā apjomā. Turklāt tas attiecas ne vien uz infrastruktūras (datori, perifērija, tīkli u.t.t.) iespējām, bet arī uz programmatūru un informācijas pieejamību kopumā.

Tas viss veido nebijušas sarežģītības kompleksus, kuru pārvaldībai un uzturēšanai ir nepieciešams liels skaits augstas kvalifikācijas speciālistu. Kā norādīts [15], ASV jau tagad ir simtiem tūkstošu neaizpildītu IT speciālistu darba vietu. Un ir paredzams, ka šis skaitlis, pat ekonomiskās nedrošības periodā, nākošo 6 gadu laikā pieaugs par vairāk kā 100%.

[16] autori apgalvo, ka „skaitļošanas sistēmu sarežģītība tiecas uz cilvēka iespēju robežām", un norāda, ka visaptverošās (*pervasive*) skaitļošanas sapnis – triljoniem datorsistēmu, kas vienlaikus pieslēgtas internetam – ātri vien var pārvērsties par ļaunu „murgu". Tā kā dažādās informācijas sistēmas tiek arvien ciešāk integrētas, pat augsti kvalificētiem sistēmu arhitektiem ir grūtības identificēt un aprakstīt prasības sistēmu savstarpējai sadarbībai. Autori paredz vēl tālāku informācijas sistēmu sarežģītības palielināšanos, cilvēkam padarot par gandrīz neiespējamu veikt sistēmu instalēšanas, konfigurēšanas, optimizēšanas un uzturēšanas darbus. Kā vienīgo iespējamo izeju autori redz autonomiskās skaitļošanas ideju.

1.1.2 Uzstādītie mērķi

IBM iniciētā risinājuma primārie adresāti ir gala lietotāji, klienti. Atbildot uz jautājumu „Ko klienti sagaida no informācijas sistēmām?", korporācija uzsver gala lietotāju vēlmi strādāt intuitīvā manierē, nevēlēšanos būt iesaistītiem sistēmas uzturēšanā, vēlmi abstrahēties no sistēmas uzbūves un tehnisko ierobežojumu aspektiem.

Par nozīmīgāko pētījumu iedvesmas avotu *IBM* zinātnieki [15] min cilvēka centrālās nervu sistēmas darbību. Tai raksturīga spēja atbilstošā veidā reaģēt uz autonomu kontroles mehānismu sūtītiem signāliem. Nosūtītie ziņojumi regulē elpošanu, temperatūru, sirdspukstus bez cilvēka apzinātas līdzdalības. No šādas idejas realizācijas sagaidāmie ieguvumi ir acīmredzami – veidojas labi organizētu, „viedu" komponenšu tīkls, kas pats pilda nepieciešamās funkcijas un lietotājam nerada ne mentālas, ne fiziskas pūles.

Jaunā paradigma maina līdzšinējo tehnoloģiju laikmeta definīciju no skaitļošanas (datu apstrādes) uz datu izmantošanu. Pieeja datiem, kas izvietoti dažādos, sadalītos datu avotos, kombinācijā ar tradicionālajām centralizētajām datu glabāšanas ierīcēm dod lietotājiem iespēju piekļūt datiem, kad un kur tie to vēlas. Vienlaikus jaunais skatījums uz informācijas tehnoloģiju nozari pieprasa mainīt industrijas fokusu no ātruma un atmiņas apjoma kāpināšanas uz sadalīto tīklu attīstīšanu, kas būtu spējīgi pašorganizēties, un caurspīdīgumu pret lietotājiem

Saskaņā ar *IBM* piedāvāto jauno paradigmu datorsistēmām (tehnika, programmatūra, datu glabātuve, atbalsts) jānodrošina fundamentāli pamatprincipi, kas balstās lietotāju vajadzībās.

Elastība

Datorsistēmai jāspēj „izsijāt" jeb atlasīt vajadzīgos datus, neinteresējoties par konkrētajām platformām un iekārtām, uz kurām tā tiek darbināta vai dati ir izvietoti (*platform- and device-agnostic approach*).

Pieejamība

Autonomiskās sistēmas būtība ir vienmēr būt „ieslēgtai", resp., pieejamai lietošanai neatkarīgi no ārējiem un iekšējiem apstākļiem

Caurspīdība

Sistēma jābūt spējīgai pildīt tās uzdevumus un pieskaņoties lietotāju vajadzībām, neliekot lietotājam iedziļināties sistēmas darbības finesēs.

1.1.3 Raksturlielumi

Konceptuālā līmenī autonomisko sistēmu ideja ir viegli uztverama, taču galvenā sarežģītība slēpjas tehnoloģiskajā izpildījumā. Tā kā dažādu nefunkcionālu risinājumu informācijas tehnoloģiju pasaulē ir neiedomājami daudz, aktuāls ir jautājums, kādas pazīmes raksturo tieši autonomiskās sistēmas?

IBM piedāvā astoņus raksturlielumus, kuru esamība liecina par risinājuma atbilstību autonomisko sistēmu ideoloģijai.

- Autonomiska sistēma „pazīst pati sevi" - tās komponentēm piemīt sistēmas īpašības. Tā kā „sistēma" var eksistēt dažādos līmeņos, autonomai sistēmai jāietver zināšanas par tās sastāvdaļām, aktuālo stāvokli, kapacitātes ierobežojumiem un visām saitēm ar citām sistēmām. Autonomiska sistēma arī „zina", kādi ir tās rīcībā esošie resursi, kādus ārējus resursus tā var izmantot savai darbībai, kuros resursos tā var dalīties un kuriem jābūt izdalītiem tieši sistēmas vajadzībām.
- Autonomiska sistēma spēj sevi konfigurēt un pārkonfigurēt atbilstoši mainīgiem (pat neparedzamiem nākotnes) nosacījumiem. Sistēmas konfigurēšanai vai uzstādīšanai jānotiek automātiski, turklāt tai dinamiski jāpapildinās saskaņā ar mainīgo apstākļu prasībām.
- Autonomiska sistēma attīstībā neapstājas – tā allaž meklē savas darbības optimizācijas iespējas. Tā pati uzrauga savu uzbūvi un uzlabo darba plūsmas atbilstoši iepriekšdefinētiem sistēmas mērķiem.
- Autonomiska sistēma spēj veikt kaut ko līdzīgu „pašdziedināšanai" - tā spēj atjaunot sevi pēc standarta un ārkārtējiem notikumiem, kas negatīvi ietekmējuši sistēmas vai to daļu darbību. Tā spēj atklāt problēmas vai potenciālas problēmas, meklēt un atrast alternatīvus funkcionēšanas veidus, piemēram, izmantojot ārējus resursus vai pārkonfigurējoties.
- Virtuālā pasaule ir ne mazāk bīstama kā fiziskā, tādēļ autonomiska sistēma spēj sevi aizsargāt. Tā spēj konstatēt uzbrukumus un aizsargāt kopējo sistēmas drošību un integritāti.
- Autonomiska sistēma pazīst vidi, savu aktivitāšu kontekstu un atbilstoši reaģē. Tā ir spējīga atrast un izveidot likumus, saskaņā ar kuriem autonomiskā sistēma vislabāk var sadarboties ar blakus esošām ārējām sistēmām. Tā spēj izmantot

pieejamos resursus, pat vienoties par kopīgu resursu izmantošanu ar citām sistēmām, mainīt sevi un vidi ap sevi – vārdu sakot, adaptēt un adaptēties.

- Autonomiska sistēma nevar pastāvēt noslēgtā vidē. Tai jābūt darbspējīgai heterogēnā pasaulē un jāatbilst atvērtiem standartiem – tas nozīmē, ka autonomiskā sistēma, pēc definīcijas, nevar būt individuāls, no apkārtējās vides atrauts (*proprietary*) risinājums.
- Autonomiskā sistēma izmanto optimizētos resursus, taču neafišē risinājuma tehnisko sarežģītību. Tā vada informācijas tehnoloģiju resursus, lai samazinātu plaisu starp darījumu (*business*) vai lietotāju mērķiem un tiem atbilstoša informācijas tehnoloģiju nodrošinājuma radīšanas mērķiem, taču neiesaista lietotāju risinājuma implementēšanas procesā.

1.1.4 Pašorganizēšanās

[16] autori par būtiskāko autonomisko sistēmu izpausmi („esenci") uzskata pašorganizēšanos (*self-management*). Tieši šai īpašībai tiek piedēvēta spēja nodrošināt datorsistēmas, kuras darbināmas 24/7 („augu diennakti visu nedēļu", resp., nepārtraukti) režīmā, nepārslogojot sistēmu administratorus un lietotājus ar sistēmas funkcionēšanas un uzturēšanas niansēm.

Pašorganizēšanās tiek aprakstīta ar četru komponenšu palīdzību, no kurām katra nodrošina vienu vai vairākus astoņos iepriekš aprakstītajos pamatpostulātos deklarētos principus.

Paškonfigurēšana

Paškonfigurējošu sistēmu svarīgs elements ir cieša integrācija starp datortehnikas resursiem un informācijas sistēmas resursiem.

Datortehnikas komponentes un resursi var tikt automātiski konfigurēti un pārkonfigurēti sistēmas darbības laikā un vienlaicīgi. Šīs darbības var tikt ierosinātas vai nu gadījumos, kad operāciju izpildei saskaņā ar optimizācijas kritēriju nepieciešami papildus resursi, vai arī datortehnikas vai programmatūras kļūdu gadījumos.

Paškonfigurēšana ietver arī spēju pievienot un atslēgt datortehnikas resursu atkarībā no administrēšanas, datortehnikas resursu vadības vai servisa personāla informācijas sistēmu dotām komandām.

Pašdziedināšana

Pašdziedināšanas īpašības nodrošina autonomiskai sistēmai spēju konstatēt datortehnikas un programmatūras kļūdas un iespēju robežās novērst šo kļūdu radītās sekas. Tas ļauj avāriju gadījumā ar minimālām (vai pat nekādām) sekām atjaunot informācijas sistēmas normālu darbu.

Pašoptimizācija

Pašoptimizācijas īpašības ļauj informācijas sistēmai veikt autonomus veiktspējas vai resursu izmantošanas apjomu mērījumus un atbilstoši mainīt datortehnikas resursu konfigurāciju, lai uzlabotu veiktspēju

Pašaizsardzība

Pašaizsardzība ir informācijas sistēmas spēja aizsargāties pret iekšējiem un ārējiem integritātes, datu un programmatūras drošības draudiem.

Tabulā 1 [16] dots salīdzinošs pārskats par pašorganizēšanas koncepta komponenšu izpausmēm „parasto" un autonomisko sistēmu gadījumā.

Tabula 1 Pašorganizēšanās komponenšu izpausmes

Komponente	**Pašreizējā situācija**	**Autonomiskās sistēmas**
Paškonfigurēšana	Uzņēmumu datu centros ir vairākas platformas un vairāki piegādātāji. Sistēmu instalēšana, konfigurēšana un integrēšana prasa daudz laika, ir augsta kļūdu iespējamība	Komponenšu un sistēmu konfigurēšana tiek veikta automatizēti saskaņā ar augsta līmeņa likumiem. Pārējā sistēma automātiski un nevainojami pieskaņojas.
Pašdziedināšana	Sistēmām ir simtiem manuāli iestatāmu, nelineāri maināmu parametru, un to skaits arvien pieaug	Sistēmas un komponentes pastāvīgi meklē iespējas uzlabot to veiktspēju un atdevi
Pašoptimizācija	Problēmu noskaidrošana un novēršana lielu un sarežģītu sistēmu gadījumā var prasīt nedēļām programmētāju komandas laika	Sistēma automātiski konstatē un salabo atklātās datortehnikas un programmatūras problēmas
Pašaizsardzība	Uzbrukumu un bojājumu atklāšana un seko novēršana notiek manuāli	Sistēma automātiski aizsargājas pret ļaunprātīgiem uzbrukumiem. Tā izmanto savlaicīgās brīdināšanas metodi, lai demonstrētu problēmas un aizkavētu to atkārtošanos nākotnē.

1.1.5 Sagaidāmie ieguvumi

Ar autonomisko sistēmu izveidi *IBM* [10] saista cerības samazināt informācijas sistēmu sarežģītību, atslogot informācijas tehnoloģiju resursus, kā arī pacelt programmatūras izstrādes nozari pavisam citā dimensijā, kas balstītos uz lēmumu pieņemšanu augstā līmenī.

Īstermiņā ir cerības samazināt sarežģītu sistēmu uzturēšanas atkarību no cilvēku iejaukšanās sistēmu darbā, kas būtiski samazinātu produkcijas un uzturēšanas izmaksas. Ilgākā termiņā sagaidāmi ieguvumi no indivīdu, organizāciju un biznesa aktīvas sadarbības sarežģītu problēmu risināšanā.

Īstermiņa ieguvumi (saistīti ar informācijas tehnoloģijām)

- Pozitīva lietotāju pieredze, jo informācijas sistēma reālā laikā sniedz labāku atbalstu.
- Izmaksu ietaupījums (*scale to use*).
- Labāk izmantoti resursi (ātrdarbība, ietilpība, izmaksas.), optimizējot gan datortehnikas, gan programmatūras izmantošanu.
- Pilnībā izmantota procesora jauda (fonā darbināmiem procesiem), ieskaitot mājas datorus, kas savienoti vienotā datorsistēmā.
- Dabiskā valodā veidoti pieprasījumi garantē precīzākas un izsmeļošākas atbildes.
- Vienlaikus nodrošināta pieeja dažādu tipu datnēm. Atvērto standartu lietošana ļauj vākt datus no visiem iespējamiem avotiem un tos apkopot ļoti īsā laikā.
- Stabilitāte. Augsta pieejamība. Augsta drošība. Pateicoties pašdziedināšanas spējām, mazāk sistēmas un tīkla kļūdu.

Ilgtermiņa ieguvumi (saistīti ar augstāka līmeņa aspektiem)

- Pieejamos resursus ir iespējams novirzīt uz augstākas prioritātes uzdevumiem.
- Iespējams sasniegt vadības līmeņa tiešu komunikāciju starp pusēm (*end-to-end service*).
- Sadarbība un globālo problēmu risināšana. Sadalītās sistēmas ļauj nekavējoties dalīties ar resursiem un informāciju, lai risinātu sarežģītus matemātiskus uzdevumus.
- Masveida simulācijas iespējas, piemēram, laika apstākļu prognozēšanas vai medicīnas jomās, kur nepieciešami sarežģīti aprēķini 24/7 režīmā.

1.1.6 Izaicinājumi tālākiem pētījumiem

Autonomisko sistēmu izstrāde pat idejas autoram korporācijai *IBM* šķiet „Lielais izaicinājums" – problēma, kuras sarežģītības nozīmības pakāpe gan no tehniskā, gan sociālā viedokļa kļūst par atsevišķas zinātnieku kopienas intereses objektu.

Nenoliedzami autonomisko sistēmu izstrāde tās sarežģītības dēļ ir Lielā izaicinājuma nosaukuma cienīga. *IBM* uzsver, ka risinājuma meklēšana un atrašana nav vienas kompānijas (pat ne tik ievērojamas, kā *IBM* pati!) spēkos, tādēļ aicina pētījumos iesaistīt gudrākos prātus gan no praktiķiem, gan no zinātniskām aprindā, kā arī dažādas komerciālas un publiskas institūcijas, kuras saredz vajadzību un steidzamību atbilstošu pētījumu veikšanai.

Viens no izaicinājuma aspektiem ir tas, ka autonomiskas sistēmas izveide jāuztver kā vienota pieeja skaitļošanai. Turklāt sarežģītība neslēpjas pašā tehnikā - daudzajos gados, kopš pastāv skaitļošanas nozare, zinātnieki un inženieri ir ievērojami paplašinājuši sākotnēji uzstādītos mērķis attiecībā uz datortehnikas veiktspēju un ātrumu. Galvenā problēma ir atvērtu standartu un jaunu tehnoloģiju radīšanā, kas nepieciešami efektīvai savstarpējai sadarbībai, atbilstībai iepriekšdefinētiem darījumu likumiem, kā arī pašaizsardzībai un spējai atjaunoties ar minimālu atkarību no tradicionālā informācijas tehnoloģiju atbalsta personāla. No daudz plašākā skatījuma uz sistēmām izriet virkne izaicinājumu, kas kopīgi jārisina.

Konceptuālā līmenī jāmainās datorsistēmu definēšanas un projektēšanas pieejām:

- skaitļošanas efektivitātes mērīšanai jābalstās uz datiem, nevis uz procesora ātrumu;
- atsevišķi datori zaudē savu nozīmi, tie atdod pozīcijas sadalītām skaitļotāju grupām;
- datorikas nozares ekonomiskais aspekts daudz labāk atspoguļos datorsistēmu izmantošanu – to vislabāk raksturo tīmekļa bāzēta ārpakalpojuma (*e-sourcing*) jēdziens.

Balstoties uz autonomiskās skaitļošanas parametriem, mainās arī individuālo komponenšu funkcionalitāte, un tā ietvers:

- mērogojamu datu uzglabāšanas kapacitāti un procesora jaudu, lai nodrošinātu individuālo un daudzu autonomisko sistēmu vajadzības pēc resursiem;
- caurspīdīgu un pārskatāmu datu apstrādi un pārsūtīšanu starp dažādām iekārtām;
- uzlabotu mikroshēmu tehnoloģiju izstrādi, lai labāk sabalansētu atmiņu;
- tīklu pārraudzības funkciju uzlabošanu, lai nodrošinātu aizsardzību pret uzbrukumiem, detektētu potenciālos draudus un sasniegtu augstu lēmumu

pieņemšanas līmeni, kas pieļautu svarīgāko aktivitāšu datu apstrādes pārslēgšanu uz citām iekārtām;

- uzlabotu mikroprocesoru izveidi, kas ir spējīgi detektēt kļūdas un par tām ziņot.

Lai izveidotu autonomiskas sistēmas, pētniekiem jārisina virkne dažādas sarežģītības uzdevumu. *IBM* [15] norāda uz atsevišķiem izaicinājumiem, ar kuriem sastopas pētījumu ietvaros.

Sistēmas identitāte

Pirms sistēma sāk sadarboties ar citām sistēmai, tai paši ir jāzina savas pašas robežas. Kā lai izveidojam sistēmu, kas spēj pati sevi nodefinēt un pārdefinēt dinamiskās vidēs?

Saskarnes veidošana

Ja tiek lietotas vairākas platformas, sistēmu administratori spiestā kārtā sastopas ar veselu „zvēru dārzu" – neskaitāmiem standartiem, objektiem, izpausmēm, prasībām. Kā izveidot saskaņotas saskarnes un vienotus kontroles punktus, tai pat laikā pieļaujot heterogēnas vides?

Darījumprasību pārvēršana informācijas sistēmu prasībās

Sistēmas izstrādes rezultātam ir jābūt saprotamam un izmantojam no lietotāju viedokļa. Kā lai izveido lietotājiem saprotamu saskarni, kas nav sarežģīta un ļauj lietotājiem komunicēt ar informācijas sistēmu dabiskā veidā?

Sistēmiskā pieeja

Ar autonomisku komponenšu izstrādi vien nepietiek. Kā lai autonomiskās komponentes apvieno un iekļauj sistēmu kopās?

Standarti

Individuālo, specializēto (*proprietary*) risinājumu laikmets ir beidzies. Kā lai izveido un atbalsta atvērtus standartus, kas reāli darbojas?

Adaptīvie algoritmi

Lai nodrošinātu sistēmu spēju darboties mainīgās vidēs un ar mainīgiem notikumiem , ir nepieciešams izveidot un sistēmās iekļaut jaunas metodes. Kā lai izveido adaptīvus algoritmus, kas spētu pārņemt iepriekšējo pieredzi un izmantot to esošo sistēmu nosacījumu uzlabošanai?

Ņemot vērā visu augšminēto, *IBM* aicina visus zinātniekus un pētniekus, kuru intereses ir tuvas autonomisko sistēmu idejai, piedalīties ar savu artavu kopīgajā darbā pie jaunas paaudzes tehnoloģiju izstrādes. Diemžēl, lai arī aicinājums ir spēkā jau 10 gadus, rezultāti pagaidām ir pieticīgi.

1.2 Adaptīvās sistēmas

Atšķirībā no *IBM*, kas iepriekš aprakstītās idejas pētījumus un rezultātus aktīvi gan publicē dažādos izdevumos, gan arī prezentē konferencēs un semināros, korporācijas *Microsoft* radnieciski pētījumi ir daudz grūtāk pieejami. Ņemot vērā abu lielo korporāciju atšķirīgo biznesa pieeju, to varētu izskaidrot ar *Microsoft* vēlmi drīzāk gūt peļņu no dažādu inovāciju ieviešanas produktos, nekā baudīt pētnieciska uzņēmuma slavu.

Tomēr arī *Microsoft* laboratorijas informē par savu pētījumu virzieniem [17], un vienam no viedo tehnoloģiju virzienam radniecīgiem pētījumu virzieniem pievēršas ASI (*The Adaptive Systems & Interaction group*) pētījumu grupa.

ASI grupa nodarbojas ar mākslīgā intelekta un informācijas sistēmu savstarpējās sadarbības principu izpēti, lai uzlabotu datu apstrādes sistēmas un saskarnes. ASI grupas pētījumi ietver domāšanas, apmācīšanās un lēmumu pieņemšanas pamatu izpēti, adaptēšanās, meklēšanas un atrašanas problēmas, kā arī datora-cilvēka savstarpējās mijiedarbības jautājumus. Grupas dalībnieki veic pētījumus arī sarežģītas struktūras informācijas pārvaldības virzienā un izstrādā inovatīvus vizualizācijas un interaktivitātes risinājumus.

Kā ASI grupa norāda savā interneta vietnē [18], pētījumu spektrs ietver gan fundamentālus pētījumus, gan arī datu apstrādes sistēmu prototipu izveidi, lai demonstrētu jaunu funkcionalitāti un tās īpašības. ASI grupa *Microsoft Research* kontekstā ir vadošie lietotāju uzvedības modelēšanas un izpētes jautājumos, īpaši koncentrējoties uz lietotāju mērķiem un vajadzībām, kas rodas dažādu informācijas avotu izmantošanas gadījumos.

Bez augstāk minētajām tēmām ASI grupa nodarbojas arī ar informācijas savākšanas un pārvaldības jautājumu pētījumiem, iekļaujot arī automatizēto tekstu klasificēšanu un klasteru tehnoloģiju izmantošanu.

Turpmākajās apakšnodaļās īsi raksturoti ASI grupas pētījumu virzieni, kas sasaucas ar viedo tehnoloģiju ideju.

1.2.1 Lietotāju uzvedības modelēšana un intelektiskās saskarnes

Pētījumu mērķis ir uzlabot cilvēka-datora sadarbību, izmantojot arhitektūras, kas izmanto lietotāja interešu, vajadzību un mērķu izmantošanu sistēmas darbībā. Lai to panāktu, tiek izmantotas notikumu orientētas sistēmas, kas uzkrāj aktivitātes un ar tām saistīto informāciju.

Šajā pētījumu tēmā galvenais uzsvars tiek likts uz lēmumu-teorētiskā lietotāja modeļiem. Reprezentatīvi pētījumu rezultātā sasniegto rezultātu piemēri ir:

- *Lumiere* projekts - lietotāju pieprasījumu apkalpošanas tehnoloģija, kas *MS Office* produktu saimē nodrošina *Office Assistant* funkcionalitāti;
- Lookout – dažādu iniciatīvu sadarbības izpēte;
- *Attentional User Interface* (AUI) projekts veltīts apmācību un pierādījumu atrašanas principiem, kā arī ar attiecīgo informācijas un servisu priekšrocībām un izmaksām un arhitektūrai. AUI projekti ietver modeļus, teoriju un sistēmas, kas izveidotas uz Prioritāšu un ziņošanas platformas (*Priorities and Notification Platform*) un virkni psiholoģijas pētījumu par atšķirībām.

1.2.2 Psiholoģiskas studijas un sarunu balstītas sistēmas

ASI grupa ir galvenais *Microsoft* pētījumu centrs jautājumos par lietotāju izturēšanos. Pētnieki darbojas psiholoģiskās izpētes laboratorijās, un galvenie pētījumu virzieni ir empīriska cilvēka uztveres izpēte, ietverot uzmanības, meklēšanas, uztveres un personības aspektus.

Pētījumu fokuss ir vērsts uz centieniem izveidot alternatīvu informācijas vizualizācijas un navigācijas attēlojumus, kas labāk atbilstu cilvēka uztveres specifikai.

Microsoft cenšas piepildīt cilvēka seno sapni par informācijas sistēmām, kas nodrošinātu brīvu sarunu starp cilvēkiem un datoriem. Tam tiek veidota speciāla „sarunu balstīta" arhitektūra, un attiecīgie projekti orientējas uz atbilstošo principu, arhitektūru un atbalsta prototipu izstrādi.

1.2.3 Ievades iekārtas un sensori

Tiek izstrādātas jauna tipa ievades iekārtas un sensori, kas lietotājiem dotu jaunas iespējas tieši manipulēt ar vizuāliem objektiem un atbalstītu modelēšanas pasākumus ar papildus informāciju par kontekstu.

Pētījumi veltīti arī tādai novitātei kā peles un klaviatūras paplašināšana ar pieskārienu sensoriem, kas ļautu sistēmas vadīt ar lietotāju pieskārienu palīdzību.

1.2.4 Skaitļošana, sistēmas un datortīkli

Šis pētījumu atzars ietver elastīgas un autonomi darbināmas skaitļošanas metodes un lēmumu pieņemšanas teoriju, kas nodrošina „šauro vietu”(*bottlenecks*) atklāšanu, pieejamo resursu efektīvas izmantošanas organizēšanu, kā arī informācijas sistēmu funkcionalitātes optimizēšanas pasākumus. Tas tiek panākts, izmantojot varbūtiskus lietotāju modeļus un lēmumu-teorētiskās analīzes principus.

Citas šīs jomas pētījumu tēmas ir sakarības starp skaitļošanas sarežģītību un uztveri, kā arī metodes un principi par skaitļojamo tuvinājumu uztveri.

1.2.5 Diagnostika, problēmu novēršana

Diagnostikas jomā tiek veikti pētījumi par problēmu atklāšanas un klasificēšanas problēmām – sākot no programmatūras atkļūdošanas metodēm (*deubugging*) un beidzot ar programmatūras un datortehnikas atteikumu iemeslu detektēšanu.

Sadarbojoties ar *Microsoft Technical Support* grupu, ASI pētnieki ir izveidojuši risinājumus, kas brīvi pieejami internetā (*Microsoft Technical Support Troubleshooters*) un ir produktīvi lietojami problēmu diagnosticēšanai.

1.3 Citi pētījumi

Tā kā programmatūras un datortehnikas stabilitātes, paplašināmības, uzturamības u.t.t tēmas ir aktuālas, protams, daudzi interesenti un pētnieki veic attiecīgus pētījumus un publicē sasniegtos rezultātus gan akadēmiskajā, gan praktiskajā plāksnē.

Ne tuvu nepretendējot uz pilnīgu pārskatu pār pētījumiem, kas tiek veikti ārpus lielo ražotāju laboratorijām, turpmāk izlases kārtībā aprakstīti daži pētījumu virzieni, kas tiem izvirzīto mērķu un uzdevumu ziņā ir radniecīgi viedo tehnoloģiju mērķiem un uzdevumiem.

1.3.1 Uzticamā skaitļošana

Uzticamās skaitļošanas (*trustworthy computing*) jēdziens tiek pielietots, lai apzīmētu datorsistēmas, kas atbilst noteiktām drošības, uzticamības un pieejamības prasībām.

Atšķirībā no industrijā daudz labāk pazīstamā „drošās skaitļošanas"(*trusted computing*) jēdziena, kas drošu sistēmu apzīmē kā tādu, kas "drīkst pārkāpt drošības iestatījumus vai ierobežojumus" [19], uzticamā skaitļošana ir sistēma vai sistēmas komponente, kura „neizgāzīsies", resp., ar augstu varbūtību nekļūdīsies.

Galvenais jautājums, kas nodarbina pētniekus, ir jautājums, kā uzbūvēt uzticamas sistēmas no neuzticamām komponentēm? [20] norāda, ka uzbūvēt sistēmu, kas ir mazāk uzticama nekā neuzticamākā no tās komponentēm, nav liela māksla, taču izveidot uzticamu sistēmu no „parastām", resp., neuzticamām komponentēm ir izaicinājums.

[20] autori identificē divas aktivitāšu grupas, kādās būtu veicami pētījumi:

- sadalīt un replicēt;
- pārraudzīt, konstatēt, atbildēt.

Sadalīto sistēmu, infrastruktūras, resursu priekšrocība ir spēja balansēt noslodzes, savstarpēji pārņemt uzdevumus atteiču gadījumos, labāk izmantot pieejamos tehniskos līdzekļus. Taču to realizācijai nepieciešami labi izkopti replicēšanas un balansēšanas līdzekļi, resp., programmatūra, kas atbalsta sarežģītus, iepriekš neparedzamus notikumus un spēj nodrošināt sistēmas darba spējas arī ierobežotu resursu apstākļos. Par galvenajiem uzdevumiem autori izvirza uzticamības palielināšanu un drošības pastiprināšanu.

Uzraudzības funkcijas sevī ietver mehānismus problēmu konstatēšanai (detektēšanai), kā arī atbildes reakcijas. Pilns uzraudzības cikls nozīmē automatizētas rīkus/ komponentes, kas pastāvīgi seko līdzi izvēlētu kontrolvērtību vai ierīču statusam un problēmu gadījumā par

to informē uzturētājus. Atbildes reakcija sevī ietver ne vien fakta saņemšanas apstiprinājumu, bet konkrētas komandas situācijas risināšanai. Pārkonfigurēšana ir pati krasākā un iedarbīgākā no iespējamām reakcijām. Autori norāda uz nepieciešamo pragmatisma devu, kas nepieciešama, veidojot atbilstošos risinājumus, jo bieži vien pārkonfigurēšana nemaz nav nepieciešama vai arī nesasniegs savu mērķi.

1.3.2 Pašstabilizēšana

Kā savā tāda paša nosaukuma grāmatā [21] definē tās autors Dolevs, pašstabilizējošas ir sistēmas, kuras pašas spēj atgriezties sev vēlamā stāvoklī, neskatoties uz kļūdu parādīšanos.

Par pašstabilizēšanas idejas aizsācēju uzskatāms Edsgers Dijkstra, kurš 1973. gadā publicēja pašstabilizēšanās paradigmu. Viņš savā darbā piedāvāja jaunu pašstabilizēšanās konceptu, apskatot to no savstarpēji izslēdzošu algoritmu viedokļa. Diemžēl Dijkstras pienesums 10 gadus nebija plaši zināms, līdz 1983. gadā tika prezentēts konferencē „ACM Symposiom on Principles of Distributed Computing".

Idejas pamatā ir doma par sistēmu, kas savu darbu var uzsākt patvaļīgā stāvoklī un vienmēr pati saviem spēkiem var migrēt uz savai darbībai nepieciešamu stāvokli, tai skaitā atgūties no pārtraukumu un kļūdu radītām sekām.

Tas ir viens no aktīvākajiem sadalītās skaitļošanas (*distributed computing*) pētījumu laukiem. Vairāki sadalītie algoritmi, kas ir sekmīgi implementēti informācijas sistēmās, tikai pēc to ekspluatācijas uzsākšanas ir atzīti par piederīgiem pašstabilizējošo risinājumu klasei.

1.3.3 Pašatjaunošanās

Par to, ka pašorganizējošas programmatūras izstrāde ir aktuāla pētniecības tēma, liecina arī pavisam nesen (2009. gadā) publicēta informācija par Eiropā un ASV veiktiem tehnoloģiju izstrādes projektiem.

[22] informē par Eiropas pētnieku veikumu SELFMAN projektā, kura ietvaros izveidota sistēma pašvārvaldošu (*self-managing*) un pašatjaunojošu (s*elf-repairing*) interneta lietojumprogrammu izveidei. Saskaņā ar projekta koordinatora Pētera van Roja teikto, pētījuma mērķis bijis panākt, ka lielas interneta lietojumprogrammas ir spējīgas pašas sevi apkalpot.

SELFMAN projekta pētnieki ir identificējuši pašpārvaldošas informācijas sistēmas četras būtiskās īpašības: paškonfigurēšanās, pašsakārtošanās, pašdziedināšanās un

pašaizsardzība. Lai iekļautu šīs īpašības lietojumprogrammās, tika izveidota strukturēta trīs līmeņu pieeja.

Pirmais līmenis nosaka, ka lietojumprogrammas kā platformu izmanto strukturētu tīklu, kurā tās glabā norādes uz visām lēmuma pieņemšanas koka virsotnēm un saitēm, kuras satur informāciju par problēmu novēršanas risinājumiem.

Otrais risinājuma līmenis ietver replicētu datu uzglabāšanas sistēmu, nodrošinot, ka katra lēmumu pieņemšanas koka virsotne ir vienmēr sasniedzama un dati ir pareizi.

Trešajā līmenī tiek darbināts problēmu risināšanas mehānisms, kas nodrošina sarežģītu algoritmu izpildi, lai sistemātiski nodrošinātu trūkstošo elementu aizstāšanu un problēmu novēršanu.

Van Rojs ir pārliecināts, ka SELFMAN „pacels internetu jaunā līmenī".

Savukārt Masačūsetsas Tehnoloģiju institūta pētnieki profesora Martina Rinarda vadībā ir izveidojuši risinājumu ClearView [23], kas dažu minūšu laikā spēj programmatūrā konstatēt un novērst noteikta veida kļūdas. Profesors Rinards, kurš izveidoto risinājumu demonstrēja 2009. gada konferencē *The 22nd ACM Symposium on Operating Systems Principles*, par pētījuma mērķi nosauca „nenogalināmas un neievainojamas programmas" izveidi.

ClearView ir darbināma bez cilvēku atbalsta un bez pieejas uzraugāmās programmatūras pirmkodam. Novērojot uzraugāmās programmas „dabisko izturēšanos" un veidojot likumu kopu, ClearView spēj atklāt noteikta tipa kļūdas, ieskaitot ļaunprātīgus uzbrukumus. Risinājums identificē novirzes no normas, resp., uzkrāto uzvedības likumu pārkāpumus un piedāvā programmatūras ielāpus (*patches*), kas liek tai darboties saskaņā ar likumiem. Ielāpi tiek pielietoti binārā veidā, neiejaucoties programmatūras pirmkodā.

ClearView analizē iespējamos risinājumus, lai pieņemtu lēmumu, kurš no tiem ar lielāko varbūtību darbosies, automatizēti uzstāda risinājumu „kandidātus" un pārbauda (testē) to darbības efektivitāti.

Ja parādās likumi, kuri ir pretrunā ar iepriekš uzkrātiem vai ielāpi apdraud sistēmas darbību, ClearView šos elementus bloķē un meklē citu problēmas risinājumu. Pētnieki apgalvo, ka piedāvātais risinājums īpaši efektīvs ir gadījumos, kad vienā un tā pati programmatūra tiek darbināta uz veselas datoru grupas.

Testa ietvaros ClearView tika uzstādīts uz daudziem datoriem, kuros tiek darbināta Firefox lietojumprogramma, un neatkarīga lietotāju grupas 10 dažādos veidos uzbruka minētajai pārlūkprogrammai. ClearView sekmīgi bloķēja visus uzbrukumus, jo savlaicīgi konstatēja novirzi no normas un aizvēra lietojumprogrammu, pirms uzbrukums tai ko varēja nodarīt.

1.4 Salīdzinošais vērtējums

Kā redzams no iepriekšējās nodaļās dotā izklāsta, „pašuzlabojošas”/ „pašvadošas” u.tml. informācijas sistēmas ir aktuāla tēma ar komercializācijas potenciālu, un pie tās strādā pasaules vadošie programmatūras izstrādātāji. Tomēr sasniegtais vēl pagaidām tālu atpaliek no vadošo ražotāju ambīcijas revolucionēt visu informācijas tehnoloģiju nozari, tai skaitā datorsistēmu izstrādes paradigmas.

Tam, autoresprāt, ir vairāki iemesli:

- izvēlētās problēmas nostādnes universālais raksturs,
- plašais risināmo jautājumu aptvērums,
- atkarība no ražotāju komerciālajām interesēm.

IBM manifestā postulētā vēlme mainīt informācijas tehnoloģiju uzbūves principus un tādējādi iespaidot visu informācijas tehnoloģiju nozari kopumā ir ļoti vispārīga. Lai arī sasniedzamais mērķis ir labi saprotams, daudziem lietotājiem un speciālistiem pieņemams, tomēr tas ir arī ļoti visaptverošs – mērķa pilnīga sasniegšana būtu saistīta ar, ne vairāk un ne mazāk, pilnīgi patstāvīgi (bez cilvēku klātbūtnes un iejaukšanās!) darboties spējīgiem datorsistēmu kompleksiem.

Turklāt šī ambiciozā mērķa sasniegšanai vienlaikus un saskaņoti ir jārisina daudzi apakšuzdevumi, kas ietver ne vien komponentes un mehānismus informācijas sistēmu īpašību uzlabošanai, bet arī jo cieši saistīti ar datorsistēmu tehnisko nodrošinājumu – ar datortīkliem, datortehniku, sakaru līdzekļiem u.c.

Jo īpaši daudzo dažādu ražotāju izstrādātas programmatūras heterogenitātes dēļ pagaidām ir grūti iedomāties lietotāju un speciālistu pēkšņu konsolidēšanos uz vienotu platformu vai vismaz idejiski vienotiem sadarbības principiem. Lai arī uzstādījumu līmenī visi ir vienisprātis, ka informācijas tehnoloģiju nozarei nepieciešama principiāli jauna, standartizēti pielietojama „saziņas valoda”, reālitātē katrs ražotājs turpina attīstīt savus risinājumus, balstoties uz pierastajām shēmām un principiem, un nav ieinteresēts papildus investīcijās, lai savas izstrādnes padarītu pieejamas vispārīgiem tehnoloģiskas dabas uzlabojumiem.

Jo īpaši *Microsoft* risinājumi ir orientēti uz *Microsoft* ražotām vidēm - *MS Windows* operētājsistēmās darbināmiem produktiem, *MS SQL Server* bāzētām datu apstrādes sistēmām un *MS Internet Explorer* darbināmām interneta sistēmām. *IBM* kā datortehnikas un programmatūras ražotājs savukārt vairāk orientējas uz vienotu platformu, kura ietvertu pašu ražotās tehnikas komponentes un atbilstošus programmatūras risinājumus.

Tādēļ faktiski literatūrā programmatūras uzlabošanai (augstāk minētajā nozīmē) sastopamas divu veidu pieejas.

- Teorētiskā pieeja ietver vispārīgo principu identificēšanu, neiedziļinoties konkrētajā realizācijā. Šo varētu dēvēt arī par „top-down" pieeju, kur ideja tiek attīstīta ar pakāpeniskās detalizācijas līdzekļiem. Šajā grupā ietilpst *IBM* manifests, Dijkstras un Doļeva pētījumi.
- Praktiskā pieeja nodarbojas ar izvirzītās problēmas risināšanu konkrētai uzdevumu grupai, resp., meklē problēmas risinājumus pie noteiktiem ierobežojumiem. Šo pieeju varētu dēvēt par „bottom-up", jo rezultāti var tikt izmantoti par vispārināšanas bāzi, lai radītu universālākus risinājumus. Šajā grupā ietilpst, piemēram, iepriekšējā nodaļā aprakstītie SELFMAN un ClearView pētījumu projekti.

Viedo tehnoloģiju koncepts un ar to saistītie pētījumi iekļaujas praktiskās pieejas klasē, jo nepretendē uz pilnīgi universālu risinājumu „visām dzīves situācijām", taču vienlaikus ir pietiekami universāli, lai varētu tikt un tiktu reāli pielietoti veselām risinājumu grupām.

Atšķirībā no teorētiskās pieejas pārstāvjiem, kuri pārsvarā orientējas uz infrastruktūras vienādošanu vai pieskaņošanu programmatūras vajadzībām, viedo tehnoloģiju pētījumi nenodarbojas ar jautājumiem par datortehnikas uzlabošanu, datortīklu vai standarta programmatūras maiņu vai uzlabošanu, bet gan pie jau dotā nodrošinājuma cenšas atrast maksimāli labu (optimālu) risinājumu specifisku programmatūras nefunkcionālo īpašību uzlabošanai. Viedo tehnoloģiju pieeja cenšas „izdabāt" realitātē sastopamajai heterogēnajai videi, kurā (vairāk vai mazāk veiksmīgi) sadarbojas daudzu ražotāju dažādas informācijas sistēmas, kuras tiek darbinātas uz visdažādāko ražotāju datortehnikas.

2. VIEDĀS TEHNOLOĢIJAS UN TO KOMPONENTI

Ikviens sekmīgs programmatūras risinājums tiek lietots un uzlabots būtiski ilgāk nekā tas ticis radīts. Veiksmīgi izstrādāta sistēma tiek lietota daudzus gadus, un to laikā programmatūra tiek papildināta ar jaunām funkcijām, modificēta, uzlabota, pamazām to tuvinot visām lietotāju vēlmēm. Un ik reizi, kad programmatūrā vai vidē, kurā tā tiek darbināta, ir veiktas izmaiņas, aktualizējas vides un programmatūras saskaņotības pārbaudes jautājums.

Vēl pavisam nesenā pagātnē vides neatbilstības problēma tika risināta, automātiski nomainot vai ziņojuma veidā uzaicinot lietotāju manuāli nomainīt nepieciešamos iestatījumus vai veikt trūkstošo komponenšu uzstādīšanu vai atjauninašanu. Šī, sākotnēji it kā ļoti loģiskā, pieeja tomēr neattaisnojas gadījumos, kad lietotājs uz darba stacijas lieto vairākas lietojumprogrammas vienlaicīgi (un šodien tas tā ir gandrīz vienmēr!). Tādā gadījumā programmatūra vairs nedrīkst uzskatīt sevi par „vientuļu salu" un mainīt vidi atbilstoši savām vajadzībām, jo tas var negatīvi ietekmēt apkārtējo sistēmu darbspējas.

Tādēļ arvien vairāk pētījumu un izstrāžu priekšplānā izvirzās ideja par programmatūras pieskaņošanos pieejamajai videi. Šī problemātika ir netriviāli risināma, jo informācijas sistēmā jau tās sākotnējās izstrādes brīdī ir jāiekļauj ne tikai pasūtītā (lietotāja noteiktā) funkcionalitāte, bet arī papildus atbalsta mehānismi programmatūras lietošanai, uzturēšanai un attīstīšanai.

Viena no iespējamām augstāk minētās „inteliģentās" pieejas izpausmēm ir šajā darbā aprakstītā viedo tehnoloģiju ideja.

2.1 Viedo tehnoloģiju jēdziens

Viedo tehnoloģiju īpašības ir programmatūras nefunkcionālo īpašību kopa, kura nodrošina programmatūras spēju pieskaņoties mainīgai ārējai videi un adekvāti reaģēt uz neparedzētiem ārējiem un iekšējiem notikumiem, tādējādi atvieglojot programmatūras uzturēšanu, papildināšanu un izplatīšanu.

Programmatūras moduļus un apakšsistēmas, kas nodrošina viedo tehnoloģiju īpašības un parasti tiek iebūvētas programmatūrā, kurai nepieciešams piešķirt viedo tehnoloģiju īpašības, sauc par viedo tehnoloģiju komponentēm.

Vienkāršības labad turpmākajā izklāstā ar jēdzienu „viedās tehnoloģijas" atkarībā no konteksta tiek apzīmētas gan viedo tehnoloģiju komponentes, gan viedo tehnoloģiju īpašības.

Būtiskā viedās tehnoloģiju realizācijas vienojošā pazīme ir konsekventa zināšanu par konkrētās programmatūras iekšējo struktūras izmantošanai, lai izveidotu noteikta tipa informācijas sistēmu vai pat kādas konkrētas informācijas sistēmas vajadzībām atbilstošu individuālu viedo tehnoloģiju realizāciju.

Atšķirībā no programmatūras ražošanas nozarē izplatītās universālās pieejas, kuras pamatā ir standartizācijas centieni, viedo tehnoloģiju risinājumu uzdevums nav radīt visām uzdevumu klasēm unificēti pielietojamas komponentes, bet gan konkrētajam uzdevumu tipam vislabākās komponentes, lai konkrētā tipa risinājumi būtu labāk pilnveidojami, uzturami, attīstāmi.

Zemāk minētas tipiskas, empīrisku eksperimentu un praktisku pētījumu ietvaros identificētas viedo tehnoloģiju īpašības [8], taču tās, protams, nav vienīgās iespējamās.

- Dinamisks biznesa modelis – informācijas sistēmas funkcionalitāte mainās (tiek interpretēta) atkarībā no konkrēto informācijas sistēmu raksturojošiem, specifiskā valodā aprakstītiem darījumprocesiem.
- Iebūvēta versiju pārvaldība un datu sinhronizācija – automātiska programmatūras versiju izplatīšana no centrālās glabātuves uz lokālām darbstacijām, ieskaitot datu struktūru konversijas un datu transportu.
- Paštestēšana – programmatūras spēja pārbaudīt pašas integritāti un darbspējas, produkcijas vidē izpildot uzkrātus testpiemērus un informējot izstrādātājus par atklātajām neatbilstībām.
- Ārējās vides testēšana – automatizēta ārējās (produkcijas) vides parametru un komponenšu pārbaude un atkarībā no konstatētajiem rezultātiem iespēja pielāgot programmatūru specifiskām vides īpašībām un/vai informēt izstrādātājus par nepieciešamajām/veiktajām darbībām.
- Datu kvalitātes uzraudzība – pastāvīga datu glabātuvē uzkrāto datu iekšējās nepretrunības un pilnīguma kontrole (pārraudzība).
- Pieejamības testēšana – informācijas sistēmas nepārtrauktas pieejamības uzraudzība, ieskaitot ziņošanas mehānismu par programmatūras darbības statusu un identificētajām papildus nepieciešamajām komponentēm/ versijām.

- Drošības un slodzes testēšana – sistēmas drošības, veiktspējas un slodzes situācijas uzraudzība.

Viedo tehnoloģiju konceptā iekļaujas un dažādos avotos ir aprakstītas paštestēšana [6,9], ārējās vides testēšana [24], versiju atjaunošana [5, 7] u.c.

Protams, ne katru individuālas programmatūras nefunkcionālu īpašību var saukt par viedo tehnoloģiju principiem atbilstošu. Par viedo tehnoloģiju principiem atbilstošām būtu saucamas tādas komponentes, kas atbilst visām zemāk minētajām īpašībām.

- Risinājums sniedz būtisku pienesumu ar to aprīkotas programmatūras uzturēšanā, papildināšanā vai izplatīšanā.
- Risinājuma tehniskā realizācija balstās uz zināšanām/ informāciju par programmatūras risinājuma uzbūvi vai datu struktūrām.
- Risinājums ir pietiekami plašs un vispārināts, lai izveidotā komponente vai funkciju kopa varētu tikt izmantota citu, radniecīgu lietojumprogrammu aprīkošanā.

Turpmākajās apakšnodaļā raksturo izvēlētu viedo tehnoloģiju īpašību būtība un attiecīgo komponenšu principiālā realizācija.

2.2 Dinamisks darījummodelis

2.2.1 Problēmas nostādne un iespējamie risinājumi

Dinamiska, programmatūras darbību regulējoša biznesa modeļa nepieciešamība izriet no joprojām neatrisinātās problēmas par programmatūras specifikāciju aprakstošo modeļu un programmatūras realizācijas neatbilstību. Plaisa starp formalizētajiem programmatūras (vēlamās) funkcionalitātes aprakstiem (ja tādi vispār eksistē!) un reālo funkcionalitātes implementāciju informācijas sistēmās ir ievērojama.

Nu jau daudzu gadu garumā zinātnieki, kas nodarbojas ar modelēšanas tematiku, cenšas radīt procesu aprakstīšanas metodes, kurās būtu apvienoti, no vienas puses, formalizēts un viennozīmīgs procesu un funkciju apraksts un, no otras puses, lietotājam (pasūtītājam) intuitīvi saprotams un viegli modificējams attēlojums. Galvenais šo centienu mērķis ir radīt iespēju pasūtītājam un izstrādātājam „runāt vienā valodā", tādējādi savlaicīgi novēršot iespējamos pārpratumus komunikācijā un minimizēt šo pārpratumu rezultātā radušās sekas.

90-to gadu pētījumi un atbilstošie risinājumi balstās uz grafisko modelēšanas valodu un CASE (*Computer-Aided Software Engineering*) rīku [25] izmantošanu. Galvenais akcents tika likts uz nepārtrauktu programmatūras izstrādes dzīves ciklu, kura ietvaros secīgi tiek veikti konceptuālās modelēšanas, biznesa procesu modelēšanas, prasību detalizēšanas, projektēšanas pasākumi un gala rezultātā tiek ģenerēts attiecīgās informācijas sistēmas pirmkods. Par galveno CASE rīku izmantošanas sūtību tika uzskatīta iekšēji nepretrunīga sistēmas modeļa izveide, kurš kalpo par materiālu lietojamas lietojumprogrammas automatizētai izveidei. Par svarīgāko šādas pieejas rezultātu tika uzskatīta iespēja veidot programmatūru „bez programmēšanas", resp., visas programmatūras izstrādes aktivitātes attiecināt uz specificēšanu, nevis koda izstrādi.

CASE rīku orientētā pieeja, kur galvenais uzsvars tiek likts uz nepieciešamās funkcionalitātes realizācijas ieguvi automatizētā ceļā, ir tikusi realizēta atsevišķu komerciālu ražotāju risinājumos. Redzamākais piemērs ir korporācijas *Oracle* veidotā izstrādes rīku saime, kas satur iespēju no specificēšanas vidē *Oracle Designer* [26] aprakstīta modeļa veidot izstrādes vides *Oracle Developer* objektus. Pazīstamākais Latvijas zinātnieku rezultāts šajā jomā ir modelēšanas vides GRADE komponente, kas nodrošināja automatizētu C++ koda ģenerēšanu no GRAPES valodā aprakstītiem procesu modeļiem [27].

Pielietojot CASE rīkus reālos izstrādes projektos, izkristalizējās virkne trūkumu:

- lai izveidotie modeļi būtu izmantojami tik precīza veidojuma kā pirmkods ģenerēšanā, to detalizācija un sarežģītība krietni pārsniedza pasūtītāja (lietotāja) spējas pārbaudīt izveidoto modeļu atbilstību reālajām vajadzībām;
- aprakstošie modeļi raksturoja darījumprocesus, resp., precizēja funkcionālās prasības, novārtā atstājot tādus, ne mazāk būtiskus informācijas sistēmu aspektus kā lietotāju saskarnes ergonomika, lietojamība, uzturamība, veiktspēja un citas;
- CASE rīki spēja uzģenerēt lietojumprogrammu tikai ar tādām iespējām, kādas ir īstenotas CASE rīkos, resp., straujā tehnoloģiju attīstība un arvien jaunu objektu parādīšanās ātri vien radīja CASE rīkos iebūvēto iespēju atpalicību no tirgū esošām iespējām;
- izstrādātāju manuāli veiktās izmaiņas pirmkodā reti vai gandrīz nekad nebija sinhronizējamas ar aprakstošajiem modeļiem, līdz ar to neizbēgami veidojās tipiskā neatbilstība starp specifikāciju un pirmkodu.

CASE rīku koncepcijas ietvaros identificētās problēmas ir mēģināts risināt ar MDA (*Model-Driven Architecture*) [28] palīdzību. Šīs pieejas būtība ir atdalīt izstrādes vides specifiskās īpašības (realizācija) no informācijas sistēmas prasībām (sistēmanalīze/ modelēšana).

MDA pieeja piedāvā informācijas sistēmu veidot divos soļos.

1. Universālā modelēšanas valodā, piemēram, valodā UML (*Unified Modeling Language*) tiek veidots platformas neatkarīgs modelis PIM (*Platform Independent Model*) [29].
2. Tiek definēta vide, kurā veidojamā sistēma darbosies. Lietojumprogramma tiek ģenerēta no PIM, kas ļauj lietojumprogrammas izveidi veikt daudz elastīgāk un gala lietotājam draudzīgāku.

MDA pieeja šobrīd pasaulē tiek intensīvi attīstīta [30, 31], tomēr šai pieejai ir konstatētas arī problēmas [32]:

- informācijas sistēmu modelēšanai (PIM fāze) lietotās valodas, piemēram, UML ir universālas un grūti saprotamas lietotājiem, kam nav pietiekami dziļu informācijas tehnoloģiju nozares zināšanu;
- no PIM specifikācijām automatizēti ģenerētais pirmkods nespēj radīt no lietojamības un uzticamības viedokļa kvalitatīvu produktu;
- automātiski ģenerētas informācijas sistēmas nav ieteicams manuāli mainīt („pielabot"), jo tādējādi tiek sabojāta atbilstība starp specifikāciju un programmas pirmkodu; ja izmaiņas tomēr veiktas, tad mainītas specifikācijas gadījumā pirmkoda pārģenerēšana individuāli iestrādātās izmaiņas likvidē.

Līdz ar to izstrāde saskaņā ar MDA metodiku nereti aprobežojas ar pirmā soļa - PIM modeļa – izveidi UML valodā, no kuras informācija (prasību apraksts) ar dažādu rīku palīdzību vai manuāli tiek pārnesta uz programmatūru.

Viedo tehnoloģiju kontekstā dinamisks biznesa modelis nodrošina veidu, kā vadīt informācijas sistēmas darbību ar konfigurējošas informācijas palīdzību, turklāt nodrošinot pilnīgu atbilstību starp konfigurējošo informāciju, kas aprakstīta specifiskā modelēšanas valodā, un konkrētās informācijas sistēmas reālo funkcionēšanu. [33]

Atšķirībā no standartizētajām modelēšanas metodēm, kuru mērķis ir veidot vispār pielietojamus, universālus rīkus, viedo tehnoloģiju pieeja koncentrējas uz problēmas

risināšanu specifiskam problēmu apgabalam, izmantojot izstrādātāju zināšanas par konkrētā risinājuma uzbūvi.

Turpmāk aprakstīts dinamiska darījummodeļa bāzēts viedo tehnoloģiju komponentes risinājums notikumu orientētām sistēmām, kas praktiski realizēts SIA „Datorikas institūts DIVI", piesaistot ERAF līdzfinansējumu projektam „Gudrās programmatūras tehnoloģijas implementācija notikumu orientētās informācijas sistēmās", un kopš 2008. gada tiek izmantots komerciālu lietojumprogrammu veidošanai.

2.2.2 Notikumu orientētas informācijas sistēmas

Lai ilustrētu viedo tehnoloģijas īpašības, kas saistītas ar dinamiska biznesa modeļa izmantošanu, īsi jāraksturo notikumu orientētu sistēmu specifika, kas tiek izmantota viedo tehnoloģiju īpašības implementācijā.

Par notikumu orientētām turpmāk sauksim informācijas sistēmas, kurās nodrošināta iespēja uzkrāt ar noteikta veida (viena tipa) objektiem noteiktos laika brīžos notikušus gadījumus (notikumus) un tos raksturojošus lielumus (parametrus). Parasti notikumu orientētās sistēmās ir nodrošināta laikā secīgu notikumu reģistrācija un apstrāde atbilstoši dinamiskai darbu plūsmai.

Tipiski notikumu orientētu sistēmu piemēri ir dokumentu pārvaldības un lietvedības sistēmas, projektu pārvaldības sistēmas, klientu attiecību vadības sistēmas, dažādu pakalpojumu (piemēram, sociālo pakalpojumu) vadības sistēmas.

Notikumu orientētu sistēmu klase viedo tehnoloģiju ideju aprobācijai izvēlēta, lai nodrošinātu viedo tehnoloģiju īpašību pielāgošanu konkrētām iestrādnēm un sistēmu īpašībām, tai pašā laikā nezaudējot abstrakcijas līmeni un veidojot vispārināmu risinājumu.

Turpmāk īsumā aprakstīti notikumu orientētas sistēmas pamatjēdzieni.

Objekts

Notikumu orientētas sistēmas galvenais datu apstrādes elements (pamatvienība) ir objekts, kurš informācijas sistēmā tiek apstrādāts un kura izvēle ir atkarīga no notikumu orientētas sistēmas izmantotāju vajadzībām un darbības nozares. Datu uzkrāšana tiek veikta par izvēlēto objektu, līdz ar to objekts reprezentē notikumu orientētas sistēmas izveides un ieviešanas mērķi un uzdevumus.

Objekts var būt gan reāla materiālas dabas objekta attēlojums, gan arī abstrakta veidojuma izpausme.

Piemēram, materiālas dabas objekts ir persona, un ar šo objektu tiek reprezentētas konkrētas fiziskas būtnes. Līdz ar to notikumu orientētas sistēmas spēj nodrošināt, piemēram, darbu ar personu reģistru – informācijas sistēmu, kurā tie uzglabāti personu dati. Personu šādā informācijas sistēmā var interpretēt kā pacientu, vēlētāju, klientu, skolotāju utt. Ir iedomājami arī citi materiālas dabas objekta piemēri – prece, dokuments, automašīna utt.

Tipisks abstrakta objekta piemērs ir projekts – tādu resursu un aktivitāšu kopums, kuru lietotājs uzskata par pārraugāmu objektu. Tas var būt uzņēmuma iekšienē izveidots projekts vai atsevišķa struktūrvienība, tikpat labi arī budžetu pozīcijas u.tml.

Cits abstrakta objekta piemērs ir ģimene – fizisku objektu abstrakta kopa. Šī piemēra aspektā var apskatīt notikumu orientētas sistēmas būtisku īpašību – daudzlīmeņu objektus –, kur nodrošināta objektu apvienošana grupās, kur grupa var būt kāds cits notikumu orientētas sistēmas objekts. Attiecīgi konfigurējot daudzlīmeņa objektus, ir iespējams izveidot dažādas specifiskas darba vietas.

Objektus raksturo divas lielas datu grupas – pamatdati un papilddati. Pamatdati objektam vienmēr ir tieši viena grupa, bet papilddati var būt patvaļīgs skaits. Pamatdatiem ir stingri noteiktas datu struktūras, savukārt papilddati ir brīvi interpretējami un nepiesaistīti konkrētai datu struktūrai (hierarhiskām struktūrām, sarakstiem vai atomāriem datiem).

Pamatdatos parasti tiek glabāta objekta relatīvi nemainīgo, fiksēto datu kopa, savukārt papilddati ir gan kvantitatīvi, gan vēsturiski kvalitatīvi mainīga datu apakškopa.

Šāda objekta datu dalīšana divās daļās ļauj ērti regulēt piekļuves tiesības dažādiem lietotājiem, nodrošinot notikumu orientētas informācijas sistēmas atbilstību drošības un konfidencialitātes prasībām. Diferencēta lietotāju piekļuve dažādām datu grupām jo īpaši svarīga ir sistēmām, kurās norisinās darbs ar sensitīviem personu datiem un konfidenciāliem finanšu rādītājiem.

Notikums

Notikums raksturo funkcijas, kas notiek sistēmas objektiem, un atspoguļo mijiedarbību starp sistēmas objektiem. Notikumi ir, piemēram, personas koda piešķiršana, automobiļa īpašnieka maiņa, nodokļu paziņojuma sagatavošana un daudzi citi.

Ir iespējami kā iekšēji, tā arī ārēji notikumi, resp., notikuma jēdziens var pārklāt gan notikumu orientētas sistēmas lietošanu organizācijas iekšienē (nosūtījumi starp struktūrvienībām, darbu sadale starp darbiniekiem), gan tās pielietošanu sadarbībā ar ārējiem adresātiem (piemēram, ienākošie un izejošie dokumenti).

Notikumu identificē (viennozīmīgi nosaka) tā parametri (īpašības). Katram notikumam ir piekārtoti standarta parametri – laiks, vieta un tips –, un notikumiem ir iespējams piekārtot arī papildus īpašības. Notikuma laiks, atkarībā no notikuma veida (vienreizējs vai ilgstošs), tiek raksturots vai nu ar vienu rādītāju (norises datums, laiks), vai arī ar veselu laika periodu (sākums un beigas).

Notikumiem var piekārtot specifiskus parametrus (atkarībā no notikuma tipa). Piemēram, notikumam var būt raksturīgi ar to saistītie maksājumi un nodevas, par notikumu atbildīgais pārraudzības iestādes darbinieks-kontaktpersona u. tml.

Ir iespējams veidot arī, tā saucamos, „saliktos" notikumus. Ja augstāk aprakstītie notikumi ir saistīti tikai ar vienu sistēmas objektu, tad saliktie notikumi aptver veselu objektu kopu (piemēram, notikums *Vēstule-atteikums visiem pretendentiem*).

Darba plūsma

Pati būtiskākā notikumu orientētas sistēmas īpašība ir darba plūsmu (*workflow*) funkcionalitātes nodrošinājums. Tā padara sistēmu par konfigurējamu un piešķir notikumu orientētai sistēmai spēju mainīties laikā arī bez izstrādātāju piedalīšanās.

Katram pasūtītāja darījumprocesam ir iespējams aprakstīt un definēt atbilstošu darba plūsmu, norādot tās sastāvdaļas – aktivitātes, iesaistīto lietotāju lomas un atbildības, kā arī iesaistītos dokumentus. Darba plūsmas katrā aktivitātē (fāzē) tiek apstrādāti noteikti iesaistītā objekta dati.

Darba plūsma paredz lietotāju aktivitātes, kas ļauj uzkrāt datus par objekta dzīves ciklu, kas vēlāk būtu pieejami vēsturisko datu analīzei. Katra lietotāja aktivitāte saistībā ar konkrēto objektu tiek interpretēta ar notikumu jēdziena palīdzību.

Stāvoklis (objekta fāze)

Objektam virzoties pa dzīves ciklu (darba plūsmu), tas secīgi iziet vienu vai vairākas fāzes. Objektam, atkarībā no tā tipa, var būt piekārtotas dažādas fāžu secības (pēc būtības – iepriekšdefinētas darbu plūsmas).

Fāzes tiek nodrošinātas ar stāvokļa jēdziena palīdzību. Stāvoklis ir noteiktu īpašību, kvalitātes rādītāju un laika nosacījumu kopums, kurā var atrasties objekts. Objekts sava dzīves cikla laikā „staigā" pa stāvokļiem. No stāvokļa uz stāvokli objektu pārvieto notikums vai notikumu kopums (piemēram, darbinieks sagatavo vēstuli, tā tiek nodota vadītājam, nonākot stāvoklī *Apstiprināmās vēstules* u.t.t.).

Katram stāvoklim ir piekārtotas jeb definētas tiesības saskaņā ar lietotāju lomām.

Stāvoklim ir iespēja definēt termiņus – pieļaujamus laika intervālus (darba vai kalendārās dienās), cik ilgi objekts var atrasties šajā stāvoklī. Termiņi nodrošina automātiskās atgādināšanas un plānošanas iespējas.

2.2.3 Darījumprocesu modelēšanas valoda BiLingva

Lai atvieglotu notikumu orientētu sistēmu specifikāciju izstrādi, konkrētajai informācijas sistēmu klasei ir izveidota domēna specifiska valoda [34] BiLingva [33], kura lietotājiem ar zināšanām konkrētajā domēnā ļauj saskaņot no informācijas tehnoloģiju specifikas neatkarīgus procesu modeļu aprakstus un radīt rezultātu, kas ir pietiekami formalizēts, lai būtu izmantojams automatizētai konfigurējošas informācijas radīšanai.

Ir jāuzsver, ka Bilingva gan nepretendē uz universālas modelēšanas valodas statusu, taču tā ir pietiekami plaša, lai varētu tikt pielietota konkrētam domēnam – notikumu orientētiem projektu pārvaldības risinājumiem.

Tradicionālās prasību specificēšanas valodās [35] informācijas apstrādes modelēšanai mēdz lietot divu veidu aprakstus.

- Stāvokļu pārejas diagrammās tiek aprakstīti identificēti apstrādājamie objekti un tiem piekārtoti stāvokļi. Informācijas apstrāde stāvokļu pārejas diagrammu gadījumā tiek uztverta kā objektu pārvietošanās starp stāvokļiem, ko izraisa objektiem pielietotās darbības (funkcijas). Līdz ar to stāvokļu pārejas diagrammā (grafā) par virsotnēm kalpo abstrakti objekta stāvokļi, kurus savieno stāvokļu maiņu izsaucošas darbības reprezentējoši orientēti loki [36].
- Procesu diagrammās tiek identificētas informācijas sistēmas darbības (funkcijas) un to izpildes secība, kādā tiek nodrošināta informācijas apstrāde. Informācijas sistēmas darbība procesu diagrammu gadījumā tiek uztverta kā ieejas ziņojumu pārstrāde par izejas ziņojumiem. Procesu diagrammu orientētu pieeju atbalsta DFD (*Data Flow Diagram*) ar daudzām modifikācijām, Grade BM (*Grade Business Modeling*), UML aktivitāšu diagrammas un citi.

Informācijas sistēmu modelēšanā var izmantot paralēli abas pieejas – lai precizētu informācijas sistēmas vēlamo darbību no dažādiem aspektiem - , taču abu veidu diagrammas katra ir autonoms objekts, kas paskaidro informācijas sistēmu no cita skatupunkta.

BiLingva koncepcijas autori [33] iet netradicionālu ceļu un piedāvā izmantot abu augšminēto pieeju „hibrīdu" – uz stāvokļiem bāzētas un procesu modelēšanas iespējas tiek lietotas vienas valodas ietvaros vienlaicīgi.

Tādēļ BiLingva grafiskajā reprezentācijā tiek izmantotas divu tipu virsotnes - objektu stāvokļi un procesi. Virsotnes savieno orientēti loki, kuri apzīmē darbības, kas vai nu maina objektu stāvokļus (notikumi), vai arī izsauc procesu izpildi. BiLingva atļauts lietot nosacījumus, kas nav sastopami stāvokļu pārejas diagrammās un DFD, bet ir sastopamas blokshēmās un biznesa procesu modelēšanas valodās.

BiLingva atbalsta pakāpeniskās detalizācijas metodi, dodot modelētājam iespēju pēc saviem ieskatiem detalizēt darbības un stāvokļus nepieciešamajā detalizācijā. Atšķirībā no CASE rīkiem, kuri pieprasīja noteikta līmeņa detalizāciju, lai uzkrāto informāciju varētu izmantot pirmkoda ģenerēšanai, BiLingva orientēta uz procesu aprakstīšanu, izmantojot iespējami maz dažādus diagrammu veidus un runājot lietotājiem pierastos jēdzienos.

BiLingva koncepcijas autori norāda uz pieejas priekšrocībām, salīdzinot ar stāvokļu pārejas diagrammu vai aktivitāšu diagrammu lietošanu:

- vienlaikus iespējams detalizēt gan objektu stāvokļus, gan darbības;
- zarošanās gadījumos, kas tiek attēloti kā izeja no procesiem, pie darbībām var tieši norādīt nosacījuma pārbaudes rezultātu.

2.2.4 BiLingva redaktors

Darījumprocesu modelēšana BiLingva valodā, protams, nav pašmērķis. Darījumprocesu modelis tiek veidots, lai varētu izstrādāt un praktiskai lietošanai nodot programmatūru. Līdz ar to rodas jautājums par pareizo metodiku, kas ļautu modelēšanas ietvaros gūto informāciju par darījumprocesiem bez ievērojamiem zudumiem transformēt darbojošās informācijas sistēmā.

BiLingva bāzētās pieejas autori piedāvā informācijas sistēmas veidošanu dalīt trīs savstarpēji saistītu aktivitāšu grupās.

BiLingva grafiskā redaktora izstrāde

Lai nodrošinātu BiLingva valodā aprakstītu procesu modeļu specificēšanu, ar modeļu transformāciju palīdzību, kas paredzētas domēna specifisku valodu rīku [37] būvei, jāizveido nepieciešamie grafiskie redaktori. BiLingva redaktors paredzēts ne vien izveidotā procesu modeļa vizuālai attēlošanai, bet arī modeļa iekšējās nepretrunības pārbaudei un aprakstu saglabāšanai vienotā objektu repozitorijā.

Izmantojot modelēšanas rīku būves platformu [38], valodas BiLingva grafiskais redaktors projektu izvērtēšanas sistēmu klases vajadzībām tika izveidots divu mēnešu laikā, turklāt atzīstamā līmenī. Izveidotais rīks piedāvā glītu diagrammu vizuālo noformējumu un ērtu diagrammu izveides („uzzīmēšanas") vidi.

Tādējādi tika atrisināta viena no lielākajām domēna specifisko valodu lietošanas problēmām – atbalsta rīku eksistence. Domēna specifisku valodu rīku būves pamatprincipi ir aprakstīti [38] .

Darījummodeļa izveide

Ar grafiskā redaktora palīdzību tiek modelēti – diagrammu formā aprakstīti - konkrētās informācijas sistēmas darījumprocesi.

Modelēšanas rezultāts tiek uzkrāts BiLingva redaktora repozitorijā, kurā bez paša modeļa būtiskās informācijas tiek glabāta papildus informācija, kas nepieciešama grafiskā redaktora darbībai un ērtai lietošanai.

Konkrētās informācijas sistēmas izstrāde

Tā kā piedāvātā pieeja neizvirza uzdevumu automātiski ģenerēt modelim atbilstošu informācijas sistēmu, tā tiek izstrādāta konkrētam problēmu apgabalam ar tradicionālajām metodēm, taču programmatūrā tiek iebūvēta spēja izpildīt BiLingva valodas konstrukcijas, nolasot nepieciešamo informāciju no speciālas datu bāzes, kas ar modeļa repozitoriju tieši nav saistīta.

Izveidotais modelis informācijas sistēmas darbībā var tikt izmantots vairākos veidos:

- lai parādītu objektus sadalījumā pa stāvokļiem,
- lai objektam konkrētā stāvoklī ļautu pielietot tikai modelī definētās darbības/ pārejas.

Lai nodrošinātu modelī uzkrātās informācijas izmantošanu informācijas sistēmas veidošanā, programmistiski ir jāatrisina modeļa objektu identificēšanas problēma. Modeļa objektu identificēšanai katram objektam tiek piekārtots atribūts, kurā glabājas modeļa unikālais identifikators (numurs). Šo numuru objektam piešķir pati informācijas sistēma, kad tajā ievada konkrētā objekta datus – tā faktiski ir tabulas, kurā glabājas modeļa objekti, primārā atslēga.

Šāda pieeja, protams, uzliek zināmus ierobežojumus – mainot esošos modeļus, ir jāseko līdzi, kā mainīsies modeļa objekti un kā būtu jāmainās to identifikatoriem. Izmainot kādu modeļa objektu, atsevišķos gadījumos var saglabāt esošo identifikatoru, citos - ir jāveido pilnīgi cits modeļa objekts arī informācijas sistēmā. Tāpat arī ir jāievēro, ka modeļa objektu mainot informācijas sistēmas datu bāzē, atbilstošā informācija jāmaina arī modelī.

Lai biznesa modeļa informāciju varētu izmantot informācijas sistēmā, ir nepieciešams pārnest informāciju no grafiskā redaktora repozitorija uz informācijas sistēmas datu bāzi. To var darīt manuāli vai automatizēti (ar speciālas programmas palīdzību).

Veicot datu pārnešanu manuāli, izstrādātājs “ar roku” aizpilda tabulu datus par modeļa objektiem un to saitēm, kā arī modelī aizpilda informāciju par modeļa objektu identifikatoriem, ko piešķir informācijas sistēma.

Tas ir diezgan vienmuļš, ilgs un nogurdinošs process, kā arī ir pastāv diezgan lielas kļūdas iespējas. Daudz efektīvāk ir veikt modeļa datu pārnešanu ar automātiska rīka palīdzību.

Šāds modeļu pārnešanas rīks no modeļu repozitorija uz informācijas sistēmas datu bāzi tika izveidots. Šis rīks, izmantojot grafiskā redaktora nodrošinātās datu izguves metodes, piekļūst modeļa repozitorijam, nolasa modeļa informāciju un pārnes to uz informācijas sistēmas datu bāzi.

Rīka pamata funkcijas ir šādas:

- nolasīt modeļa informāciju;
- atrast modeļa objektus ar esošiem identifikatoriem informācijas sistēmas datu bāzē un veikt nepieciešamās izmaiņas;
- izdzēst no informācijas sistēmas datu bāzes objektus, kuri nav modeļa repozitorijā;
- pievienot jaunos modeļa objektus informācijas sistēmas datu bāzei;
- jauno objektu identifikatorus ierakstīt modeļa repozitorijā atbilstošo objektu attiecīgajos atribūtos.

Atšķirībā no CASE rīku pieejas, kad no specifikācijas automātiski tiek veidots izpildāms programmatūras kods, BiLingva rīki nodrošina nepieciešamās informācijas (objektu datu, parametru un nosacījumu) izvietošanu repozitorijā, no kurienes specifiski konvertēšanas rīki to var pārvērst konkrētajai informācijas sistēmai nepieciešamos konfigurējošos datos.

Tādējādi ir panākts, ka informācijas sistēma darbojas atbilstoši grafiskā valodā BiLingva izveidotam modelim, un tā kvalitāte – lietojamība, drošība, veiktspēja un citas - ir atkarīgas no pašas informācijas sistēmas īpašībām, nevis no hipotētiska ģeneratora iespējām ģenerēt kvalitatīvu lietojumprogrammu.

2.3 Paštestēšana

2.3.1 Problēmas nostādne

Tradicionālajos programmatūras izstrādes cikla modeļos [35] testēšana parasti tiek apskatīta no diviem aspektiem:

- programmatūras testēšana, zinot tās uzbūvi (t.s. "baltās kastes princips"), ko parasti veic paši izstrādātāji;
- programmatūras funkcionālā testēšana, pārbaudot programmatūras atbilstību prasībām (t.s. "melnās kastes princips"), ko parasti veic neatkarīga testētāju grupa.

„Baltās kastes" testēšanas principa pamatā ir redzējums, ka izstrādātāji kā programmatūras koda autori vislabāk zina programmatūras uzbūvi "no iekšienes" un līdz ar to vislabāk var pārbaudīt, vai programmatūras testēšanas ietvaros atbilstoši kādam noteiktam testēšanas kritērijam (pārklājums, moduļi, saskarnes utt.) pārbaudīti visi attiecīgie programmatūras komponenti [39].

Savukārt programmatūras prasības, saskaņā ar kurām ir veidota programmatūra (ja vien tās ir formalizēti pierakstītas, protams!), ir vispārpieejama informācija, kas ļauj pārbaudīt programmatūras darbības pareizību, nezinot un pat neiedziļinoties konkrētās programmatūras uzbūvē. Tā ir funkcionālā testēšana pēc "melnās kastes principa", kuru parasti uztic neatkarīgiem testētājiem, jo tiek uzskatīts, ka programmētāji var subjektīvi interpretēt specifikācijas (ir jau to darījuši, veidojot attiecīgo programmatūras produktu!), ir pārāk nekritiski pašu izstrādāto produktu testētāji un līdz ar to nespēj programmatūras pārbaudi veikt nepieciešamajā līmenī.

Augšminētā tradicionālā pieeja testēšanai labi darbojas jaunu produktu izstrādes gadījumā, kad tiek lietoti lineāri programmatūras izstrādes modeļi ar relatīvi nemainīgām prasībām. Tad ir svarīgi uz konkrētu brīdi (parasti tā ir programmatūras "palaišana" ekspluatācijā) radīt maksimāli kvalitatīvu programmatūru, kas ar augstu varbūtību būs darbspējīga mērķa vidē un strādās atbilstoši tai uzstādītajām prasībām. Gadījumā, ja sistēma un programmatūra tiek izstrādāta, pakāpeniski paplašinot funkcionalitāti, vai arī jau ekspluatācijā esošai programmatūrai prasību izmaiņu rezultātā nepieciešami jauni laidieni un versijas, aktuālākā problēma ir regresa testēšana. Tā prasa arvien atkārtot jau iepriekšējos

laidienos notestētas programmatūras atkārtotu testēšanu. Vēl sarežģītāka ir situācija, kad programmatūra attīstās garākā laika periodā saskaņā ar kādu no pēdējā laikā īpaši iecienītajiem spējās programmatūras izstrādes dzīves cikla modeļiem [40].

Tradicionālā pieeja ir efektīva, ja konkrētās programmatūras izstrādātājs ir „valdnieks" pār vidi, kurā programmatūra tiek ekspluatēta, resp., pār ekspluatācijā lietoto standarta produktu versijām un uz tiem pašiem resursiem darbinātajiem individuālajiem programmproduktiem. Tādā gadījumā izstrādātāji un testētāji pēc sekmīgas testēšanas var būt pārliecināti, ka testēšanas vidē novērotā programmatūras darbība tāda pati būs arī ekspluatācijas vidē.

Tomēr mūsdienās situācija praksē ir sarežģītāka. Reālu sistēmu gadījumā vairs nevaram runāt par atsevišķu informācijas sistēmu lietošanu, kas nodrošina konkrētu biznesa prasību izpildi, bet gan jārunā jau par heterogēniem informācijas sistēmu kompleksiem. Dažādo kompleksu komponenšu attīstība notiek strauji un nesaskaņoti, tādējādi radot konfliktus starp informācijas sistēmu komponentēm, negaidītus infrastruktūras radītus efektus u.tml.

Viedo tehnoloģiju [6] principiem atbilstoši informācijas tehnoloģiju risinājumi, kur iekļauta paštestēšana, ir viena no iekšējo programmatūras stabilitāti nodrošinošām metodēm, kas samazina augšminētos riskus.

Ir jāuzsver, ka paštestēšana neaizvieto tradicionālo skatījumu uz programmatūras testēšanas nepieciešamību un paņēmieniem, bet gan papildina tos. Kā jebkura programmatūras iespēja, arī paštestēšanas funkcija ir ne vien jāizstrādā, bet rūpīgi jānotestē, un produkts, kurš aprīkots ar paštestēšanas iespēju, pakļaujams rūpīgai testēšanai. Paštestēšanas funkcija programmatūras produktam piešķir papildus kvalitātes pārbaudes mehānismu – spēju mērķa vidē izpildīt kritisko funkcionalitāti pārbaudošus testa piemērus un automatizēti salīdzināt iegūtos rezultātus ar etalonu vērtībām.

2.3.2 Paštestēšanas komponenti

Paštestēšanas, tāpat kā pārējo viedo tehnoloģiju principiem atbilstošu programmatūras produktu īpašību, pamatā ir ideja par programmatūru, kura līdzīgi kā saprātīga būtne, spētu atbilstoši reaģēt uz iepriekš neprognozējamu vidi. Turklāt svarīga ir ne vien spēja diagnosticēt vides raksturlielumus [41], bet arī saskaņā ar paredzamajām vides izmaiņu sekām sevi atbilstoši pieskaņot izmainītajai videi.

Datortehnikas nozarē testi, kuri pārbauda datortehnikas darbības korektumu tūlīt pēc iekārtas ieslēgšanas, ir pierasta lieta, taču programmatūras nozarē tā ir novitāte. Viedo tehnoloģiju principiem atbilstošas programmatūras gadījumā ar paštestēšanas funkciju sapratīsim programmatūras spēju produkcijas vidē automātiski izpildīt iepriekš sagatavotus testus savas darba spējas pārbaudei.

Paštestēšanas funkcionalitāte satur divas pamata komponentes:

1. kritiskās funkcionalitātes testa piemēri un etalonu vērtības (testu dati);
2. risinājumā iebūvēts automātiskās testēšanas mehānisms (programmatūra).

Kritiskā funkcionalitāte ir to sistēmas funkciju kopums, bez kura programmatūra nav lietojama tai paredzēto uzdevumu izpildei. Kritiskā funkcionalitāte iekļauj sistēmas pamata funkcijas, parasti tās ir funkcionālās prasības, kā, piemēram, aprēķini, darba plūsmas utt. Parasti kritiskā funkcionalitāte neietver nefunkcionālās prasības, kā lietošanas ērtība, veiktspēja un citas.

Paštestēšanas funkcionalitātes implementācijas ietvaros izstrādātāji sagatavo pilnu funkcionālo testu kopu, kas ietver visu sistēmas darbībai kritisko funkciju pārbaudi, un noteiktā veidā noglabā tos izstrādātās programmatūras datu bāzē (failos).

Bez tam izstrādātāji programmatūrā iekļauj automātisku testu izpildes mehānismu, kas iekļauj testa piemēru izpildes rezultātā iegūto rezultātu salīdzināšanu ar etalona vērtībām. Kā būs redzams tālākajās nodaļās, paštestēšanas realizācija prasa tradicionālo testēšanas atbalsta rīku iespējas ieintegrēt veidojamā sistēmā.

Galvenā paštestēšanas īpatnība ir spēja programmatūru testēt produkcijas vidē jebkurā laikā. Līdz ar to, veidojot paštestēšanas mehānismu, izstrādātājiem jāgarantē, ka informācija reālajās datu bāzēs rakstīta netiek, bet tās var izmantot lasīšanas režīmā. Tādējādi sistēmas testēšanu var veikt mērķa vidē, netraucējot tās lietošanu, un katrā testēšanas reizē tiek veikta uzkrātajiem testiem atbilstoša regresā testēšana.

Paštestēšana ir paredzēta sistēmas pārbaudei produkcijas vidē, un tā tiek veikta visā sistēmas lietošanas laikā. Pēc katras sistēmas izmaiņas tiek papildināta testu kopa, tā nodrošinot paštestēšanas ietvaros pārbaudāmās kritiskās funkcionalitātes pārklājumu.

Paštestēšanas idejas ir attīstījušās daudzu gadu garumā, sākot ar senu publikāciju lieldatoru ērā [42] un attīstoties līdz iebūvēto testu (*built-in-test*) idejām komponentu bāzētai programmatūrai [43].

Galvenā paštestēšanas atšķirība no minētajos darbos diskutētajiem risinājumiem ir realizācijas mehānismā. Paštestēšanas funkcionalitāte līdzinās regresa testēšanas atbalsta rīku iespējām, taču ir iebūvēta testējamās informācijas sistēmās.

2.3.3 Paštestēšana realizācija

Paštestēšanas īstenošanai programmatūrā ir jāiekļauj iespējas, kuras parasti realizē rīkos, kurus lieto tradicionālajā testēšanas procesā: testu uzkrāšana, testu atspēlēšana, testu izpildes rezultātu salīdzināšana ar etalona vērtībām un citas [43].

Galvenā atšķirība ir apstāklī, ka testēšanas atbalsta rīkus var lietot tikai testa vidē. Tie nav pielietojami ražošanas vidē, jo grūti veikt pietiekamu testēšanu, neveicot ierakstus datu bāzēs, taču tas ražošanas vidē nav pieļaujams. Īpaši svarīgi tas ir situācijās, kad datu bāzē glabājas sistēmas darbībai būtiski parametri, piemēram, kodifikatori, konfigurācijas dati un citi. Tad testa vide un ražošanas vides var būtiski atšķirties, un tāpēc ražošanas vidē radušās problēmu situācijas grūti atkārtot testa vidē.

Lai risinātu šīs problēmas, ir vai nu jārūpējas par testēšanas vides izveidošanu lietotāju datoros, kas tehniski ir darbietilpīgi un licenču izmaksu dēļ – dārgi, vai arī jāpārnes ražošanas vides dati uz testa vidi, lai pārbaudi varētu atkārtot un izprast ražošanas vidē atklātās problēmas. Īpaši sarežģīti novērst ražošanas laikā atklātās problēmas ir gadījumos, kad programmatūras izstrādātāji ir nodevuši to pasūtītājam (izpildījuši savu kontraktu) un uzturēšanas darbus veic citi speciālisti.

Citādu pieeju piedāvā paštestēšana – testēšanas iespēja tiek iekļauta programmatūras funkcionalitātē tās izstrādes laikā, ir pieejama visā programmatūras dzīves cikla laikā – sākot no izstrādes un turpinot to visu uzturēšanas laiku. Turklāt programmatūra, kurā ir iekļauta paštestēšanas iespēja, var tikt testēta gan testa vidē, gan ražošanas vidē, kur testēšanas atbalsta rīki, kā likums, nav pielietojami. Tādēļ paštestēšanas mehānisms izstrādātājiem ir jāiebūvē pašā programmatūrā, kuru paredzēts pārbaudīt paštestēšanas režīmā.

Lai sistēma spētu veikt paštestēšanu, ir jāsagatavo:

1. pārbaudes testi kritiskai funkcionalitātei (pilna testu kopa);
2. programmatūrā jāīsteno testu automātiska izpilde, ieskaitot iegūto rezultātu salīdzināšanu ar etalona vērtībām.

Tā kā paštestēšanu jāspēj veikt produkcijas vidē, testu piemēri nedrīkst ietekmēt produkcijas datu bāzi.

Lai izveidotu paštestēšanas otro komponenti – testu automātiskās izpildes iespēju un iegūto rezultātu salīdzināšanu ar etalona vērtībām – jāņem vērā zemāk aprakstītie principi.

2.3.4 Režīmu vadība

Tā kā ar paštestēšanas mehānismu aprīkotā informācijas sistēma var tikt lietota gan produkcijas, gan dažādos testēšanas režīmos, tad ir lietderīgi ieviest parametrus, kuru vērtības nosaka izpildes režīmu (*mode*) un informē par informācijas sistēmas statusu (*info*).

Sistēmā būtu jāparedz vismaz trīs izpildes režīmi: produkcijas, paštestēšanas un testu uzkrāšanas režīmi (mode={production, self-test, create-test}).

Pēc noklusēšanas informācijas sistēma tiek lietotas produkcijas režīmā, nepieciešamības gadījumā lietotājs var pārslēgties uz testēšanas režīmu.

Produkcijas režīmā nekādas testējošas darbības netiek veiktas, tomēr sistēmas arhitektūrā ir paredzēts, ka visas informācijas lasīšanas-rakstīšanas darbības (informācijas lasīšana no ekrāna formām un datu bāzēm), kā arī vadības izvēles darbības (izvēlnes un spiedpogu darbības) ir koncentrētas atsevišķos moduļos.

Tas palīdzēs modificēt informācijas lasīšanas-rakstīšanas darbības lietošanai testēšanas režīmā. Produkcijas režīmā ir lietderīgi uzstādīt pazīmi, kas, gadījumā, ja noticis sistēmas darbības incidents, nosūta atbildīgam personālam atbilstošu ziņojumu un papildus informāciju.

Testēšanas režīmā tiek veikta automatizēta programmatūras pārbaude, kuras laikā tiek izpildīti uzkrāšanas režīmā reģistrēti vai specifiskā testpiemēru specificēšanas valodā aprakstīti testa piemēri.

Visas apstrādājamās informācijas lasīšanas komandas sistēmā ir modificētas tā, ka informācija tiek lasīta nevis no reālās datu bāzes, bet gan no testu bāzes vai failiem (*test_Base*). Arī sistēmas darbības rezultāti tiek rakstīti nevis produkcijas datu bāzes tabulās, bet gan saglabāti rezultātu bāzē vai failos (*result_Base*).

Pēc sistēmas darbības rezultātu iegūšanas tiek salīdzināti testu piemēros faktiski iegūtie rezultāti (*result_Base*) ar etalona vērtībām, kuras ir noglabātas testu uzkrāšanas laikā (*etalon_Base*). Nesakritību gadījumā tiek izdoti atbilstoši ziņojumi.

Augstāk raksturotā informācijas lasīšanas-rakstīšanas darbību modifikācija ir specifiska katrai konkrētai sistēmai, un tās implementācija prasa papildus darbu.

Uzkrāšanas režīms ir paredzēts testa piemēru un etalona vērtību fiksēšanai un noglabāšanai testu bāzē, kur tie būs pieejami izmantošanai testēšanas režīmā.

Ja iestatīts uzkrāšanas režīms, informācijas sistēma protokolē visas nepieciešamās informācijas lasīšanas-rakstīšanas un vadības darbības, ierakstot tās testu bāzē (*test_Base*). Sistēmas darbības rezultāti netiek rakstīti produkcijas datu bāzē, bet saglabāti rezultātu bāzē vai failos (*result_Base*). Aprakstītā testu uzkrāšanas darbība un uzkrājamās informācijas apjoms ir specifiski katrai konkrētai sistēmai.

Uzkrāšanas režīma iekļaušana sistēmā nav obligāta, jo testi testu bāzē var tikt ierakstīti ar citiem līdzekļiem. Tomēr pilnīgi automatizētai paštestēšanas veikšanai tā ir augstākā mērā lietderīga. Papildus jāmin, ka šis režīms var tikt izmantots, lai lietotājs, atklājot informācijas sistēmas darbības kļūdu, varētu šo darbību virkni, kas ved pie programmatūras nekorektas darbības, ierakstīt testu datu bāzē un nodot to uzturēšanas personālam kopā ar incidenta ziņojumu.

Informējošais parametrs *info* ir neatkarīgs no izpildes režīma *mode*. Tas nosaka izpildes gaitā protokolēšanas failā izvadāmās informācijas apjomu, kuru pēc tam var analizēt, lai noskaidrotu programmatūras darbības pareizību (info = {high, middle, low}).

Kad programmatūras tiek darbināta noteiktā režīmā, būtisko darbību rezultātus un programmas izpildes trasi atbilstoši parametra informējošā parametra vērtībai ieraksta protokolēšanas failā. Tiek veidots savdabīgs darbību žurnāls, kurā uzkrātā informācija var tikt izmantota ražošanas vidē atklāto problēmu cēloņu noskaidrošanai un analīzei.

Vairums programmēšanas vides piedāvā līdzīgas mainīgo vērtību un programmas izpildes trases protokolēšanas iespējas audita žurnālos. Tomēr šajā gadījumā var tikt izvadīta problēmas apgabalam specifiska un būtiska informācija dažādās detalizācijas pakāpēs.

Lai nodrošinātu iespēju pilnīgi automatizēti vienā seansā izpildīt paštestēšanu uz daudziem testiem, ir lietderīgi paredzēt testu bāzē iekļaut vadības tabulu vai failu (*test_Plan*), kas satur izpildāmo testu vārdus. Paštestēšanas dzinis šajā gadījumā izpildīs visus *test_Plan* norādītos testus, tādējādi panākot iespēju veikt selektīvu paštestēšanu.

2.3.5 Testa piemēru aprakstīšana

Līdzīgi kā ārēji regresās testēšanas rīki, paštestēšanas risinājums sniedz lietotājam mehānismu testu piemēru veidošanai, bet nenosaka (neveido) testa piemēru vērtības, kā tas notiek, piemēram, pilnu testa kopu ģenerēšanas gadījumā [44].

Tā kā paštestēšanas mehānisms ir iebūvēts pašā programmatūrā, tad ir pilnīgi iespējams, ka tajā ir paredzēta un realizēta tikai ierobežota apgabala testa vērtību ierakstīšana. Paštestēšanas risinājumā varētu būt, piemēram, paredzēts nevis uzkrāt pa vienai atsevišķai ekrāna ievades lauka vērtībai, bet gan visu vienas aizpildītas ekrāna formas vērtību kopu saglabāšanas operācijas brīdī. Ir iespējams, ka uzkrāti un par testa piemēriem kalpo programmatūrā dinamiski izveidotie SQL pieprasījumi, kuri tad paštestēšanas režīma ietvaros tiks izpildīti uz testa bāzes.

Tieši šajā īpatnībā arī slēpjas paštestēšanas patiesais spēks – testēšanai sagatavot un testēt ir iespējams tieši tos objektus (datu kopas), kuri informācijas sistēmas darbam ir būtiskie. Tā kā paštestēšanas funkcijas programmatūrā iebūvē paši izstrādātāji atbilstoši pasūtītāja dotam norādēm, tad jau savlaicīgi ir zināms, kuri objekti sistēmas darbā ir svarīgākie un kādā veidā var pārliecināties par kritisko funkciju korektu izpildīšanos.

Piemēram, reālā SIA „Datorikas institūts DIVI" produktā Vērtspapīru un valūtas operāciju uzskaites sistēmā VVUS par vienu no būtiskajām pazīmēm, kā pārbaudīt vērtspapīru un/ vai valūtas darījumu pareizību, ir aplūkot attiecīgo aprēķinu rezultātā no ievadītajiem parametriem izveidotos grāmatojumus. Regresu rīku gadījumā šādu pārbaudi būtu grūti veikt, jo izveidotie grāmatojumi tiek noglabāti specifiskā starptabulā un līdz ar to attiecīgās etalona vērtības nav atrodamas kādā ekrāna formā. Turpretī attiecīgais paštestēšanas mehānisms ļauj kā testa piemēru noglabāt visu vērtspapīra/ valūtas operācijas ievades formas aizpildījumu kā vienotu testa piemēru un attiecīgo grāmatojumu uzskatīt par etalona vērtību, ar kuru paštestēšanas režīmā salīdzināt iegūtos grāmatojumus.

Arī lietojumprogrammās, kuras ir notestējamas, balstoties uz ekrānu formās atrodamajām ievades un aprēķinu vērtībām, ir iespējama no konteksta atkarīga (selektīva) testēšana, kuras ietvaros tiek pārbaudītas būtiskās informācijas sistēmas īpašības un funkcijas. Ekrāna formas elementu apstrādes procedūrās tiek iebūvēta vērtību uzkrāšanas un paštestēšanas iespēja, tādējādi veidojot savdabīgus vadības kontroles punktus (*testpoint*), ar kuru palīdzību tiek nodrošināta secīga testpiemērā iekļaujami vērtību ielasīšana un

aprēķināšana un starp kuriem paštestēšanas režīmā vadība tiek nodota informācijas sistēmai darbam „parastajā" veidā.

Samazinot starp diviem kontroles punktiem ielasāmo ekrāna formas lauku skaitu līdz 1, tiek iegūts savdabīgs atkļūdošanas mehānisms (*debugger*), kas automatizēti iziet vērtību ievades un aprēķināšanas soļus un ļauj lietotājam izsekot neprecizitāšu/ kļūdu paradīšanās vietu un izpausmes. Darbinot informācijas sistēmu „pa soļiem", var iegūt interesantu paštestēšanas realizācijas „blakusproduktu" – iespēju automatizēti demonstrēt informācijas sistēmas funkcionalitāti, kas labi var tikt izmantots attālinātās apmācībās vai demonstrācijās bez cilvēka klātbūtnes.

Testa piemēru pierakstīšanai ir nepieciešama specifiska valoda, kurā vispārīgi var tikt aprakstīti testa piemērus raksturojoši parametri. Katru testa piemēru raksturo vismaz šādi lielumi:

1. testa piemēra identifikators (unikāls visas testu kopas ietvaros) un konkrētās testa piemēra sastāvdaļas identifikators (unikāls konkrētā testa piemēra ietvaros);
2. testa piemēra uzkrātās (uzkrāšanas režīmā ierakstītās) vērtības;
3. etalona vērtības (ja nav jau uzkrātas, bet tiek definētas);
4. testa datu bāzes versija, uz kuru testa piemērs izpildāms;
5. iepriekšdefinēti nosacījumi, kuru izpildīšanas/ neizpildīšanās gadījumā testa piemēra izpildes gaitā piemērojamas papildus darbības.

Tā kā vērtību ievade „tiek pārķerta" tieši no pirmkoda, tad testa piemēros tiek fiksētas un uzglabātas tikai to objektu vērtības, kuros uzkrāšanas režīmā tiek veiktas lietotāju (testa piemēru definētāju) ievades darbības un attiecīgi aprēķini. Nav nepieciešama visu ekrāna vadīklu un/ vai visu datu bāzes darbību reģistrēšana.

Testa piemēru noglabāšanai vajadzīgas specifiskas datu struktūras. Pirmā doma, kas rodas, ir ievietot testa piemērus konkrētās informācijas sistēmas datu bāzē, taču šāda pieeja ir jūtīga pret platformas (izstrādes vides, datu bāzu pārvaldības sistēmas u.tml.) izmaiņām un ir gandrīz nederīga produktiem, kas ir darbspējīgi uz dažādām platformām. Tādēļ testu glabāšanai labāk izvēlēties kādu universālu formātu – teksta (virknes) failus vai XML struktūras.

2.3.6 Datu bāzes versiju vadība

Veidojot testa piemēru kopu, paštestēšanas rīkiem jābūt spējīgiem ne vien reģistrēt lietotāja ievadītās testa piemēru vērtības, bet arī „izdomāt", kādas sekas šo vērtību ievade un tai sekojošie aprēķini varētu atstāt uz testa un produkcijas datu bāzēm.

Ir iespējami arī tādi testu piemēru, kuru izpildei testa datu bāzē ir nepieciešamas specifiskas vērtības (tipisks piemērs- klasifikatori) vai to izpildē jāvar apiet „normālo" programmas darījumloģiku (tipisks piemērs ir modālās ekrāna formas).

Atšķirībā no regresās testēšanas rīkiem, kad lietotājs (testa piemēru veidotājs) var būt drošs, ka testu uzkrāšanas darbības neko ļaunu produkcijas datu bāzei nodarīt nevar, paštestēšanas gadījumā riski ir pavisam citi. Līdz ar to, krājot testu piemērus, ir jāpieraksta ne vien testējamie ievadlauki, to struktūras un vērtības, bet arī ar testa bāzu versijām un saturu saistītā specifiskā informācija.

Veidojot paštestēšanas implementāciju, būtiski ir iestrādāt testa/ produkcijas datu bāzu versiju pārvaldības mehānismus, nodrošinot datu bāzes satura rezerves kopiju veidošanu, pārslēgšanos starp dažādām testa datu bāzes versijām, datu (daļēju) atjaunošanu no rezerves kopijām, testa datu (daļēju) dzēšanu u.tml.

2.4 Ārējās vides analīze

2.4.1 Problēmas nostādne

Viens no lielākajiem informācijas sistēmu draudiem ir neprognozējama ārējā vide. Jo īpaši jau iepriekš raksturotā dažādo platformu heterogenitāte un straujā tehnoloģiju attīstība padara informācijas sistēmu uzturēšanu un attīstīšanu par lielu izaicinājumu.

Katrs programmatūras izstrādātājs ir saskāries ar situāciju, ka jau pat neilgu laiku pēc informācijas sistēmas piegādes pasūtītājam un ekspluatācijas uzsākšanas tirgū parādās jaunas standarta komponentes, tai skaitā pat operētājsistēmu versijas, kurām izveidotā informācijas sistēma nav vai ir tikai daļēji derīga, un tāpēc ir problemātiski nodrošināt informācijas sistēmas darbaspējas.

Lai tomēr nodrošinātu izstrādātās lietojumprogrammatūras izmantojamību, praksē tiek lietotas divas pieejas. Pirmajā gadījumā programmatūras piegādātājs lietotājam diktē noteikumus, pie kuru izpildīšanās tiek grantēta programmatūras darbība atbilstoši tai

uzstādītajām prasībām. Šis princips konsekventi tiek lietots programmproduktu izplatīšanā, kad tiek noteiktas minimālās prasības datortehnikai un bāzes programmatūrai. Diemžēl minētā pieeja vāji vai vispār neaizsargā lietotāju no kolīzijām, kas rodas, piemēram, savstarpēji konfliktējošu programmproduktu versiju vai citu blakus faktoru dēļ. Otrs šādas pieejas trūkums ir tirgus ierobežojumi, kas rodas, jo lietotāji nespēj vai nevēlas izpildīt (varbūt arī, ka nesaprot, kā izpildīt) ražotāju uzstādītās minimālās prasības.

Otrs ceļš ir programmatūras papildināšana ar dažādu versiju spraudņiem, resp., uzlabošana, lai būtu nodrošināta programmatūras darbaspēja visdažādākajās (maksimāli daudzās) vidēs un to kombinācijās. Fiziski tas nozīmē izstrādāt moduļus visām zināmajām platformām, un katras jaunas vides parādīšanās gadījumā papildināt programmatūru ar jaunu „pieslēgumu".

Viens no atslēgas vārdiem te ir platformu neatkarība, kas nosaka to, ka programmatūra ir darbināma vairākās operētājsistēmās, uz dažādām datu bāzu vadības sistēmu platformām, ar dažādām pārlūkprogrammām utt.

Faktiski, izstrādājot „universāli lietojamu" programmatūru, ir izvēle – vai nu katrai no sagaidāmajām vidēm izveidot savu tehnisko risinājumu, kas ir savietojams ar konkrētās platformas prasībām, vai tehniskajā risinājumā lietot paņēmienus un izstrādes rīkus, kas aptver visu mērķa platformu atbalstīto iespēju apakškopu.

Pirmā pieeja ir ļoti resursu ietilpīga, jo faktiski nozīmē atbalstīt visu plašo iespējamo vidu klāstu, kas plaši izplatīta programmprodukta gadījumā nozīmē lielu darbietilpību un augstas prasības pret reakcijas ātrumu.

Otrās pieejas raksturīgs piemērs ir SQL pieprasījumu valodas universālās apakškopas lietojums datu pieprasījumos, lai nodrošinātu pieprasījumu izmantojamību ar dažādām datu bāzu pārvaldības sistēmām.

Neatkarīgi no izvēlētās pieejas šis ceļš ir dārgs, jo bieži vien prasa ar potenciālo ieguvumu (apmierināts klients) nesamērīgu ieguldījumu (ļoti plašu tehnisko funkcionalitāti).

Tomēr pat platformu neatkarība neglābj piegādātājus no programmatūras uzstādīšanas problemātikas – joprojām liels laika un darba apjoms tiek tērēts kolīziju iemeslu noskaidrošanai, konsultācijām programmatūras un jauninājumu uzstādīšanas jautājumos, arī programmatūras uzlabošanai, lai tā būtu darbināma specifiskā vidē.

2.4.2 Ārējās vides analīzes risinājuma sastāvdaļas

Iedvesma ārējās vides analīzes īpašībai ņemta no dabas - no novērojuma, ka katra dzīva būtne spēj analizēt vidi, kurā viņa atrodas, un cenšas piemēroties dzīvei tajā.

Prakse rāda, ka jo īpaši sadalītu sistēmu gadījumos (arī interneta bāzētās informācijas sistēmām) bieži sastopamas neatbilstības starp lietojumprogrammas prasībām un vides uzstādījumiem, kas izsauc nepatīkamas situācijas.

Lai nodrošinātu programmatūras spēju piemēroties ārējai videi, būtiska ir programmatūrā iebūvētā ārējās vides analīzes funkcija. Ārējās vides analīzes process sevī ietver noteiktu pazīmju analīzi, salīdzinot pazīmju vērtības ar vēlamajām (etalonu), un konkrētas pieskaņošanās darbības atbilstoši iepriekšdefinētai lēmumu pieņemšanas plūsmai.

Ārējās vides analīze ietver mehānismus operētājsistēmas, datubāzes pārvaldības sistēmas versiju, iestatījumu u.c. ārēju sistēmu parametru kontrolei, kā arī lēmumu pieņemšanas komponenti, kas, izmantojot konstatētās parametru vērtības, „pieņem" lēmumu par programmatūras reakciju uz konstatēto vides statusu. Ar viedām tehnoloģijām aprīkota programmatūra spēj analizēt apkārtesošās, informācijas sistēmas darbībai relevantās vides atbilstoši informācijas sistēmas prasībām un adekvāti reaģēt neatbilstību gadījumos, par to izdodot nepieciešamos ziņojumus lietotājam.

Kaut arī pirmajā acu uzmetienā šis uzdevums šķiet triviāls, praksē tieši ārējās vides analīze („cik lielā mērā vide atbilst prasībām?") un tai sekojošā lēmumu pieņemšana („kā vislabāk rīkoties, lai videi pieskaņotos?") ir vissmagākie uzdevumi. Piemēram, programmprodukta versijas numurs un daži uzstādījumi tikai daļēji nosaka to, vai konkrētais programmprodukts spēs pildīt savas funkcijas. Arī rīcības izvēle ir atkarīga no ārējiem apstākļiem, piemēram, vai ir pieejamas tiešsaistes pieslēgums nepieciešamajiem resursiem u.tml.

Detalizēts pētījums ārējās vides analīzes jautājumos ir atrodams [45], turpmāk īsi raksturotas galvenās attiecīgās komponentes sastāvdaļas un funkcijas.

Viens no risinājumiem [46], kas atvieglotu ārējās vides analīzes procesu, ir programmatūras papildināšana ar tā saucamo „prasību pasi", kurā tiek fiksētas visas prasības pret izpildes vidi: operētājsistēmas un citu komponentu versijas, detalizējot līdz objektu klasēm, *DLL*, *ini* failiem, *registry* ierakstiem, failu un mapju izvietojumiem, reģionālajiem un valodu iestatījumiem, darba staciju iestatījumiem utt.

Prasību pasi izveido programmprodukta izstrādātājs; viens no prasību pases iegūšanas ceļiem ir tās ģenerēšana no izstrādes vides, kur lietojumprogramma ir izstrādāta un ir darbības spējīga. Sagatavotā prasību pase tiek ieintegrēta attiecīgajā programmatūrā.

Pēc programmatūras uzstādīšanas ekspluatācijas vidē speciāls izstrādātāju sagatavots modulis salīdzina ekspluatācijas vides parametrus ar pasē norādītajiem (veic vides analīzi).

Lai arī no pirmā acu uzmetiena ārējās vides parametru atbilstības pasei pārbaude šķiet trivialitāte, tomēr tā ir saistīta ne vien ar tehniskām grūtībām (ne visas ārējās komponentes pakļaujas parametru nolasīšanai), bet arī ar grūti atbildamiem jautājumiem saturīgā plāksnē – kā būt drošiem, ka katrā laika momentā pase atbilst aktuālajai situācijai (piemēram, ka apritē nav jaunas, nepazīstamas vides)?

Ja konstatēta neatbilstība starp mērķa vides parametru vērtībām un prasību pasē norādītajām vērtībām, atkarībā no pieskaņošanās mehānisma, ir iespējamas dažādas reakcijas. Pati triviālākā reakcija ir brīdinājums par neatbilstību, iespēju robežās norādot nepieciešamās darbības, kas būtu veicamas, lai konstatēto neatbilstību novērstu („Lūdzu nomainiet reģionālos iestatījumus!”).

Augstāk attīstīti programmatūras risinājumi ir aprīkoti ar trūkstošo komponenšu savākšanas mehānismu (piemēram, automātiskā komponenšu atjaunināšana no ražotāja resursiem), kā arī spēj veikt vides pārkonfigurēšanu atbilstoši prasībām [47].

Tajā pašā laikā šādi automatizēti mehānismi, kas nodarbojas ar apkārtējās vides pieskaņošanu sev, var nopietni apdraudēt ārēju programmatūras risinājumu darbību. Tādēļ viens no iespējamiem risinājumiem ir risinājumā iestrādāt savietojamību pasi, kura, līdzīgi kā prasību pase, tiek piegādāta kopā ar programmproduktu un kurā tiek fiksētas ārējo sistēmu obligātās prasības un iespējamie savstarpēji konfliktējošie parametri.

2.5 Datu kvalitātes kontrole

Viedo tehnoloģiju principiem atbilstošas programmatūras svarīga sastāvdaļa ir diagnosticētās informācijas ziņošana izstrādātājiem. Primitīvākā ziņošanas mehānisma izpausme ir lietotāju darbību auditācijas un kļūdu reģistrācijas žurnāls, kuru veido un speciāli tam pilnvarotam lietotājam analīzei pieejamu padara ikviena mūsdienīga informācijas sistēma.

Taču viedo tehnoloģiju gadījumā būtiska ir programmatūras spēja informāciju ne vien pierakstīt (pasīvi piefiksēt), bet arī to mērīt, salīdzināt ar kvalitātes prasībām (metainformāciju) un darīt zināmu (aktīvi izplatīt) atbilstošam personālam.

Īpaši labi attīstītu risinājumu gadījumā programmatūrai pašai jācenšas (iespēju robežās) meklēt problēmas risinājumu. Šīs īpašības sasaucas ar ārējās stabilitātes nodrošināšanas pasākumiem, kuru ietvaros programmatūra atrod un uzstāda trūkstošās komponentes un ārējos dziņus.

Īpaša loma informācijas sistēmas ekspluatācijas procesā ir datu bāzē uzkrātās un uzkrājamās informācijas kvalitātei. Datu modelis tikai daļēji definē ierobežojumus un atkarības starp datu elementiem, kas tiek uzkrāti datu bāzē. Savukārt programmatūras procedūras, kas veic detalizētu ievadāmās informācijas kontroli, var izrādīties nepilnīgas.

Papildus grūtības rada notikumu fiksēšana, kas datu bāzē nonāk ar laika nobīdi pret notikuma faktisko iestāšanās momentu. Jo īpaši šādos gadījumos datu bāzē var nokļūt savstarpēji nesaderīgi un nekvalitatīvi dati.

Risinājums meklējams, informācijas sistēmā iekļaujot autonomu datu kvalitātes komponenti, kas regulāri pārbauda datu bāzes informācijas atbilstību kvalitātes un saderības nosacījumiem.

Lai gan datu kvalitātes atribūti un to mērīšana ir specifisks uzdevums katram konkrētam lietojumam [48], tomēr vispārīgā teorija [49,50] jau šobrīd ļauj definēt un mērīt datubāzē uzkrātās informācijas kvalitāti. Pieredze rāda, ka minētā datu kvalitātes kontrole ir īstenota atsevišķās sistēmās, piemēram, [51].

2.6 Automātiska versiju izplatīšana

2.6.1 Problēmas nostādne

Informācijas tehnoloģiju nozare raksturojas ar lietojamās infrastruktūras, programmatūras un tās izpildes vides daudzveidību, pie kam individuālās programmatūras pielāgošana darbam dažādās vidēs prasa ievērojamus resursus. Augstā konkurence nozarē, kas prasa kvalitatīvu un inovatīvu produktu savlaicīgu parādīšanos tirgū, izvirza programmatūras izstrādātājiem nopietnu problēmu: īsā laikā izstrādāt un piegādāt lietotājiem dažādās vidēs lietojamu programmatūru.

Nodaļā par ārējās vides analīzi diskutētās problēmas attiecas ne vien uz ārējās vides parametru kontroli un lēmumu pieņemšanu, bet arī uz risinājumu versiju izplatīšanu. Nehomogēnās vides apgrūtina attālinātu versiju izplatīšanu, jo sekmīgu versiju uzstādīšanu un atjaunināšanu apdraud konfliktējošas informācijas sistēmas (programmproduktu versijas) un platformu īpašības.

Kā jau minēts iepriekš, platformu neatkarība, kas paredz programmatūras darbināšanu dažādās vidēs, var prasīt nopietnus papildus resursus. Faktiski, cenšoties izstrādāt „universāli lietojamu" programmatūru, nonākam pie nepieciešamības katrai no vidēm veidot savu tehnisko risinājumu.

Universālu algoritmu un metadatu lietošana tikai daļēji var atvieglot „universāli lietojamas" programmatūras veidošanu, jo galarezultāts - izpildāmais kods – joprojām paliks platformas atkarīgs. Līdz ar to programmatūras izplatītāji un lietotāji ir spiesti rēķināties ar programmatūras uzstādīšanas un aktualizēšanas grūtībām – liels laika un darba apjoms tiek tērēts kolīziju un problēmu risināšanai.

Automātiskās versiju izplatīšanas īpašība atspoguļo viedo tehnoloģiju koncepcijas centienus uzlabot gan programmatūras, gan datu versiju izplatīšanas procesus, padarot tos automatizētus un „inteliģentus". Informācijas sistēma, kas aprīkota ar viedajām tehnoloģijām, spēj analizēt vidi, kurā tā tiek ievietota, no standarta un specifisko parametru viedokļa un veikt nepieciešamās darbības, lai informācijas sistēma tiktu nogādāta pie lietotājiem darbaspējīga un aktuāla.

Automātiskā versiju aktualizēšana sevī ietver gan ārējās, gan iekšējās stabilitātes pārbaudi, jo ir saistīta ar apkārtējās vides analīzi pirms versijas aktualizēšanas un uzstādītās versijas savas darbības pārbaudi pēc aktualizēšanas.

2.6.2 Versiju izplatīšanas sastāvdaļas

Atšķirībā no iepriekš raksturotajām viedo tehnoloģiju īpašībām, no kurām katra pilnībā veltīta kādam konkrētam programmatūras aspektam – ārējā vide, iekšējā konsistence, konfigurējams darījummodelis -, versiju izplatīšanas īpašība pēc būtības ir „pamata" komponenšu kombinācija, jo sevī ietver gan pašpārbaudi, gan ārējās vides analīzi u.c.

Faktiski versiju izplatīšanas problemātika sevī ietver šādas lielākas sastāvdaļas:

- programmatūras versijas uzstādīšana/ atjaunināšana ietver informācijas sistēmas izpildāmu moduļu un to darbībai nepieciešamo dziņu piegādi (transportu) un uzstādīšanu;
- datu struktūru sinhronizēšana ietver izveidoto/ atjaunināto datu struktūru veidojošo/ mainošo skriptu piegādi un izpildi;
- datu sinhronizēšana veic attālināti uzkrātu datu apkopošanu, resp., aktualizēšanu nepieciešamajā „virzienā" (daļa datu no centrāles uz lokālajām tabulām, daļa – no lokālajām uz centrālo).

Katra no augstāk minētajām sastāvdaļām ir saistīta ar plašu nepieciešamo aktivitāšu un iespējamo risinājumu klāstu. Jo īpaši datu sinhronizēšanas jautājumi jau tagad tiek labi risināti pasaules vadošo informācijas tehnoloģiju produktu ražotāju risinājumos.

Pilns automātiskās versijas aktualizēšanas cikls sevī ietver šādu viedo tehnoloģiju principiem atbilstošu mehānismu secīgu darbināšanu:

1. ārējās vides analīze,
2. esošās sistēmas drošības kopiju veidošana,
3. programmatūras versijas lejupielādēšana un uzstādīšana,
4. datu pārcelšana,
5. paštestēšana,
6. sistēmas atjaunošana neveiksmīgu darbību gadījumā.

Turpmākajā izklāstā īsi raksturota autores piedāvātā pieeja un pamatota atsevišķu izstrādņu nepieciešamība, salīdzinot ar jau esošajiem gatavajiem risinājumiem.

2.6.3 Programmatūras versiju izplatīšana

Veidojot programmatūras uzstādīšanas (instalēšanas) risinājumus, lietojumprogrammā ir jāiekļauj līdzekļi, kas bez lietotāju iejaukšanās spēj uzstādīt jaunas programmatūras versijas. Nepieciešamās šāda risinājuma sastāvdaļas :

1. automātiska ārējās vides atbilstības prasībām pārbaude,
2. automātiska programmatūras jaunās versijas lejupielāde un uzstādīšana,
3. automātiska personalizācija un konfigurējošo parametru pārnešana uz jauno versiju,
4. jaunās versijas paštestēšana, kas pārbauda informācijas sistēmas korektu darbību uz kritiskās funkcionalitātes,
5. drošības kopiju veidošana un sistēmas atjaunošana neveiksmīgu darbību seku novēršanai.

Ārējās vides analīze

Ārējās vides analīzes problemātika un risinājumi ir raksturoti iepriekšējā darba nodaļā pie attiecīgās viedo tehnoloģiju īpašības apraksta.

Jāuzsver, ka, veidojot programmatūras versiju izplatīšanas komponenti, jākontrolē ne vien „standarta", bet arī individuālie informācijas sistēmai pieejamo resursu parametri.

Pie standarta parametriem pieder uz konkrētā servera/ darba stacijas pieejamā operētājsistēma, datu bāzu vadības sistēma, pārlūkprogrammas u.tml. Un, lai arī to pārbaude ir salīdzinoši standartizēta, tomēr ir iespējami pavisam negaidīti efekti. Viens no tradicionāliem sarežģījumiem ir izkrītošo izvēlņu nobloķēšana, kas nepieciešami interneta bāzētu informācijas sistēmu darbā.

Pie specifiskajiem parametriem pieder konstatējums, vai uz konkrētā servera/ darba stacijas jau bijušas uzstādāmās sistēmas iepriekšējās (iespējams – bojātas) versijas, cik un kādas informācijas sistēmas tiek darbinātas paralēli konkrētajai u.tml

Specifisko parametru vērtības ļoti būtiski ietekmē versiju uzstādīšanas komponentes lēmumu pieņemšanas procesu un līdz ar to arī darbības efektivitāti. Konkrēts piemērs no SIA „Datorikas institūts DIVI" projektu pieredzes ir negaidīta antivīrusa programmas darbība, klasificējot viedo tehnoloģiju risinājumu, kas nodrošina jauninājumu automatizētu lejupielādi, par vīrusu un tā bloķēšana.

Programmatūras versiju uzstādīšana

Programmatūras versijas uzstādīšanas problēmas un risinājumi ir atkarīgi no izplatāmās programmatūras arhitektūras.

Tradicionāli par visvienkāršāk uzstādāmajām tiek uzskatītas centralizētās tīmekļa lietojumprogrammas, jo tās savam darbam izmanto standartizētus interneta pārlūkus. Praksē tomēr arī tīmekļa lietojumprogrammu versiju izplatīšanā sastopamas vismaz trīs problēmas:

- pārlūkprogrammu tikai daļēja savietojamība; interneta un dažādo pārlūkprogrammu attīstība ir dabisks process, kuru nespēj ietekmēt atsevišķu sistēmu izstrādātāji, līdz ar to ir praktiski neiespējami nodrošināt visu pasaulē pieejamo pārlūkprogrammu identisku darbību (kaut vai informācijas attēlošanu);
- decentralizēta datu uzglabāšana; uzņēmumi un valsts iestādes nereti visus savus datus glabā nevis vienotā centralizētā sistēmā, bet gan individuālos serveros; līdz ar to programmatūra vai tās daļas ir jāsagatavo un jāizplata izvietošanai uz daudziem, savstarpēji nesaistītiem serveriem;
- lokālo darbstaciju iestatījumu atšķirības; personalizācija kā lietotāju ērtības risinājums kļūst arvien populārāks un ir pretrunā ar datu centralizāciju; papildus grūtības var radīt dažādi reģionālie iestatījumi, kas nepieciešami dažādu sistēmu ekspluatācijā (piemēram, decimālais atdalītājs un datuma formāti dažādās sistēmās var būt atšķirīgi.).

Turpmākais izklāsts ir veltīts sadalītu klienta-servera lietojumprogrammu versiju izplatīšanai, jo attiecīgā problemātika te ir aktuālāka un piedāvātie risinājumi grūtāk implementējami nekā centralizētu tīmekļa lietojumprogrammu gadījumā.

Jaunu programmatūras versiju uzstādīšana notiek automatizēti caur internetu un ietver šādus secīgus soļus.

1. jaunākās versijas uzstādīšana uz servera,
2. jaunākās versijas parametru identificēšana,
3. programmatūras komponenšu versiju salīdzināšana,
4. jauno komponenšu lejupielāde un instalēšana.

Uz centralizētā servera tiek novietota programmatūras jaunākā versija kopā ar parametru datni, kurā ir norādīti sistēmas komponenšu jaunāko versiju numuri un darbināšanai nepieciešamie parametri.

Lai optimizētu versiju lejupielādi, sistēma tiek dalīta komponentēs, kas ir nosacīti neatkarīgi sistēmas apgabali, kuri var tikt aktualizēti neatkarīgi viens no otra. Katrai no sistēmas komponentēm parametru datnē tiek iekļauts ieraksts, kurā par jaunāko versiju tiek norādīts komponentes versijas numurs un (vajadzības gadījumā) izpildes parametri.

Ja kādas sistēmas komponentes jaunākā versija ir saistīta ar nepieciešamību atjaunināt arī citu sistēmas komponenti, tas tiek atspoguļots parametru datnē, paaugstinot jaunākās versijas numuru ne tikai vienai, bet abām komponentēm.

Brīdī, kad lietotājs no lokālās darba stacijas pieslēdzas sistēmai, programmatūrā iebūvēts mehānisms vēršas pie centrālās datu bāzes un lejupielādē aktuālās versijas parametru datni. Sistēmas komponenšu jaunāko versiju numuri tiek salīdzināti ar uz darba stacijas uzstādīto sistēmas komponenšu versiju numuriem. Tām komponentēm, kuru versiju numuri uz lokālās darba stacijas ir zemāki par jaunākajām versijām uz centrālā servera, ir nepieciešams lejupielādēt jaunākās versijas

Sistēma lejupielādē nepieciešamo komponenšu jaunāko versiju instalāciju pakotni un uzstāda versijas uz darba stacijas. Instalēšanas laikā tiek kontrolēta operētājsistēmas iestatījumu atbilstība parametru datnē norādītajām.

Kad instalēšana sekmīgi pabeigta, lokālās darba stacijas parametru datnē tas tiek atzīmēts, palielinot uzstādīto komponenšu versiju numurus uz uzstādīto versiju numuriem. Ja kaut kādu apstākļu dēļ jauno versiju lejupielāde vai instalēšana beigusies nesekmīgi, parametru datnē komponenšu versiju numuri palielināti netiek, resp., nākamajā sistēmas darbināšanas reizē komponenšu aktualizēšana tiks veikta atkārtoti.

Konfigurējošo datu migrācija

Datu migrācijas problemātika vispārīgam gadījumam ir apskatīta nākamajās apakšnodaļās, šeit jāmin specifisku datu pārnešanas jautājumi.

Problēmspecifiska datu migrācija ir nepieciešama gadījumos, kad katram programmatūras lietotājam sistēmā ir iespējama personalizācija. Ja sistēma ir veidota tā, ka personalizācija glabājas katrā individuālā darba stacijā, tad katra programmatūras versijas nomaiņa izsauc personalizācijas parametru migrāciju uz jauno versiju. Ja personalizācija ir centralizēta, tad šī migrācija ir jāveic servera līmenī. Tipisks personalizācijas piemērs ir lietotāju un tiesību vadība, bez kuras nevar iztikt gandrīz neviena sistēma.

Viedo tehnoloģiju risinājuma gadījumā personalizācijas datu (lietotāju un tiesību vadība) migrāciju automātiski veic atbilstoša sistēmas iekšējā komponente, atbrīvojot lietotāju no tam neizprotamu skriptu izpildes.

Paštestēšana

Programminženierijas praksē ir pieņemts, ka pirms jaunas versijas ieviešanas produkcijas vidē tā tiek testēta un akceptēta testa vidē. Tomēr, gadījumos, kad sistēma izmanto daudzas darba stacijas, daudzus dažādās vietās izvietotus serverus, nav garantijas, ka jaunā versija produkcijas vidē spēs strādāt korekti divu galveno iemeslu dēļ:

- uz katras darba stacijas un/ vai servera ir individuāli dati, kas var ietvert, piemēram, personalizācijas parametru vērtības, lietotāju tiesības un pat ļoti sarežģītas procesu definīcijas;
- katra darba stacija un/ vai serveris var būt aprīkoti ar atšķirīgu ārējo vidi - atšķirīgas operētājsistēmu, datu bāzu vadības sistēmu versijas, failu izvietojums, reģionālie iestatījumi un citas ārējās sistēmas, kas tiks darbinātas paralēli jaunajai versijai.

Līdz ar to svarīga versiju izplatīšanas sastāvdaļa ir iebūvēts paštestēšanas mehānisms.

Drošības kopijas un sistēmas atjaunošana

Automātiskā versiju aktualizēšana ir saistīta ar izmaiņām programmatūras komplektācijā, apkārtējās vides iestatījumos, kā arī datu bāzes struktūrās un pat datos. Līdz ar to neatņemama automātiskā versiju aktualizēšanas mehānisma sastāvdaļa ir drošības kopiju pārvaldība.

Katrai no informācijas sistēmas fiziskajām sastāvdaļām (programmatūrai, datu bāzei, vides iestatījumiem) ir jānodrošina šādas darbības:

- drošības kopijas izveidošana un noglabāšana,
- atjaunošana no drošības kopijām iepriekšējās situācijas atjaunošanai neveiksmju gadījumos.

Drošības kopiju izveidošana ir salīdzinoši vienkāršs uzdevums – pirms jaunu versiju uzstādīšanas speciālās mapēs tiek veidotas lietojumprogrammu failu kopijas, noglabāta datu bāzes rezerves kopija, kā arī aizpildīta ārējās vides iestatījumus raksturojoša datne (atbilstoša iepriekš aprakstītajai informācijas sistēmas iestatījumu pasei).

Atjaunošanas funkcionalitāte ir daudz sarežģītāks uzdevums, jo

- datortehnikas ierobežoto iespēju dēļ vēstures uzglabāšanas ilgums ir galīgs,
- ārējās vides atgriešana iepriekšējā stāvoklī ne vienmēr iespējama tikai standarta iestatījumu nomaiņas ceļā.

Svarīga atjaunošanas funkcionalitātes sastāvdaļa ir neveiksmju detektēšana, resp., viennozīmīgu pazīmju kopums, kas signalizē, ka programmatūras jaunās versijas uzstādīšana kaut kādu apstākļu dēļ nav izdevusies un ir nepieciešama iepriekšējās situācijas atjaunošana no drošības kopijām. Turklāt ierobežotā vēstures uzglabāšanas perioda dēļ ir svarīgi, lai izveidotā un atjaunošanai pieejamā drošības kopija būtu tikusi izveidota no funkcionēt spējīgas programmatūras versijas, nevis no iepriekšējā nesekmīgā uzstādīšanas mēģinājuma rezultātā radušās!

Neveiksmju analīze, jo īpaši pazīmju kopuma, kas liecina par programmatūras darbspēju vai darbnespēju, izveide ir atsevišķa pētījuma rezultāts. Vienkāršākajos pielietojumos parasti tiek pārbaudīta visu komponenšu esamība un lietojumprogrammas galvenā loga atvēršana bez nokļūšanas kļūdu apstrādes procedūrā (*error handling routine*).

SIA „Datorikas institūts DIVI" izveidotajā risinājumā atjaunošanas funkcionalitāte sevī ietver pēdējās sekmīgi uzstādītās versijas uzstādīšanas komplekta atkārtotu lejupielādi un instalēšanu, ieskaitot attiecīgo datu bāzes struktūru rekonstruēšanu. Diemžēl šajā pieejā ir apdraudēta tā uzkrāto datu daļa, kas nākusi klāt kopš pēdējās veiksmīgi uzstādītās versijas, resp., dati varētu būt tikuši uzkrāti ar kādas nepilnīgas starpversijas palīdzību, bet nevis modernizētā (papildinātā) datu struktūrā.

2.6.4 Datu struktūru pārcelšana

Izmaiņas programmatūrā bieži ir saistītas ar izmaiņām datu bāzu struktūrā, iestatījumos, apmaiņas formātos u.tml.

Datu pārcelšana kā automatizētās versiju izplatīšanas mehānisma sastāvdaļa var tikt apskatīta no diviem viedokļiem:

- datu struktūru pieskaņošana jaunai programmatūras versijai;
- periodiska datu sinhronizēšana starp lokālām darba stacijām, kas atsevišķos brīžos var strādāt pilnīgi autonomi, un centrālo datu krātuvi.

Datu struktūru pieskaņošana jaunai programmatūras versijai parasti tiek veikta ar speciālu skriptu palīdzību, kas tiek izplatīti reizē ar jauno programmatūras versiju un kas tiek darbināti tūdaļ pēc programmatūras uzstādīšanas. Bieži vien datu struktūru papildināšana ir saistīta ar vēsturisko datu papildināšanu ar jauniem (piemēram, noklusētajiem) vai datu migrēšanu (strukturālu pārveidošanu), taču arī tas ir salīdzinoši vienkāršs uzdevums.

2.6.5 Datu sinhronizēšana

Daudz sarežģītāks ir datu sinhronizēšanas uzdevums. Klientu-serveru arhitektūras lietojumprogrammās, kur tiek nodrošināts darbs gan autonomā režīmā uz lokālajām darba stacijām, gan tiešsaistes režīmā ar tiešsaistes pieslēgumu centrālajam serverim, datu sinhronizēšana ir centrālā problēma.

Datu sinhronizēšanas ietvaros ir jāatrisina ierakstu (objektu) viennozīmīgas identificēšanas jautājums, jānodrošina „atritināšanas" (*rollback*) iespējas, jāspēj automatizēti sinhronizēt konfliktējošus un pretrunīgus datus u.tml. .

Viens no risinājumiem, ko parasti izmanto, ir identifikatoru segmentu izdalīšana, resp., konkrētām darba stacijām tiek piešķirti unikāli identifikatoru segmenti, kuros ietilpstošie ieraksti tiek identificēti kā konkrētai instancei piederīgi.

Ja risinājums tiek darbināts tiešsaistes režīmā, visi ieraksti tiek glabāti centralizēti. Ja kaut kādu apstākļu dēļ tiešsaistes režīms nav pieejams vai netiek izmantots, sistēma automātiski pārslēdzas uz nesaistes (*offline*) režīmu un nodrošina datu glabāšanu lokālā datu

bāzes instancē. Pēc tiešsaistes režīma atjaunošanās, programmatūra automātiski veic datu sinhronizēšanu, balstoties uz ierakstu identifikatoriem un ierakstu izveidošanas/ mainīšanas laika zīmogu.

Sinhronizēšanas procesa laikā pēc ierakstu intervāla tiek identificēta datu rašanās instance (lokālā darba stacija), tad tiek salīdzinātas centrālajā datu bāzē noglabātās vērtības ar aktualizētajām vērtībām no lokālajām datu bāzēm un vajadzības gadījumā centrālajā datu bāzē tiek veikta neaktuālo vērtību nomainīšana pret aktuālajām.

2.7 Uzraudzības komponentes

2.7.1 Veiktspējas pārraudzība

Veiktspējas analīze īpaši aktuāla kļuvusi pēc portālu un tīmekļa lietojumprogrammu ienākšanas izstrādātāju un lietotāju ikdienā. Problēmas būtība ir tā, ka, izstrādājot portālu vai tīmekļa aplikāciju, izstrādātājiem nav iespējams pietiekami viennozīmīgi prognozēt lietotāju pieprasījumu intensitāti un veidus.

Problēmu cenšas atrisināt ar slodzes testiem, kur, izmantojot tīmekļa lapu pieprasījumu ģeneratorus, portālam piesūta iepriekš definētu pieprasījumu plūsmu un tad, mērot sistēmas veiktspējas parametrus pie dažādām slodzēm, cenšas prognozēt tīmekļa lietojumprogrammas veiktspēju produkcijas vidē. Tā kā šādi mērījumi parasti tiek veikti testa un nevis produkcijas vidē, bieži vien izstrādātāji informācijas sistēmas faktiskās "šaurās" vietas tā arī nespēj atklāt.

Viedo tehnoloģiju koncepts piedāvā programmatūrā iebūvētu veiktspējas kontroles un pārraudzības mehānismu. Izstrādātāji jau savlaicīgi paredz programmatūras instrumentāciju slodzes un laika parametru mērīšanai. Ikdienas ekspluatācijā šie mērītāji atrodas neaktīvā stāvoklī. Tos var aktivizēt sistēmas uzturēšanas personāls atbilstošās situācijās un tādējādi operatīvi saņemt precīzu informāciju par sistēmas problemātiskajām, tā sauktajām „šaurajām", vietām.

Kā papildus īpašību viedo tehnoloģiju programmatūras veiktspējas pārvaldības risinājumā var (un būtu vēlams) paredzēt atbilstošā personāla informēšanu par reakcijas laiku kritisko vērtību sasniegšanu.

2.7.2 Slepenības pārraudzība

Slepenība - iespēja lietot sistēmu tikai tiem, kam tā ir paredzēta - šobrīd ir viena no informācijas sistēmu kritiskajām prasībām.

Parasti slepenības prasības cenšas nodrošināt, izveidojot autentifikācijas (lietotāju identifikācijas) un pilnvarošanas (lietotāju tiesību) mehānismus. Tomēr šī pieeja ir visai sašaurināta, jo atbildība par slepenību tiek nodota sistēmas lietotāju ziņā. Kā zināms, daudzi slepenības apdraudējumi ir saistīti ar citiem faktoriem – operētājsistēmu un programmēšanas sistēmu arhitektoniskām nepilnībām, kā arī ar infrastruktūras ekspluatācijas kļūdām, par kurām lietotāji var nebūt pat informēti.

Viedās tehnoloģijas ietver papildus pasākumus slepenības nodrošināšanā. Tā, piemēram, ar viedām tehnoloģijām aprīkota programmatūra var atbalstīt paroļu un lietotāju vadības kontroli, atļaujot lietot tikai pietiekami „neuzminamas" paroles, var atgādināt par nepieciešamību savlaicīgi mainīt paroles un kontrolēt to unikalitāti. Viedo tehnoloģiju komponentes var informēt atbilstošo personālu un konkrētu lietotāju par ārējiem mēģinājumiem „uzlauzt paroles" vai piekļūt sensitīviem datiem. Šīs iespējas relatīvi viegli īstenot ar aģentu tehnoloģiju palīdzību [52].

SIA „Datorikas institūts DIVI" minētās iespējas ir realizējis Latvijas telekomunikāciju nozarei nozīmīgajā numuru saglabāšanas datu bāzes izstrādes projektā NUSA, kas katram interesentam atļauj noskaidrot drauga mobilo telefonu apkalpojošo operatoru, lai lēti varētu nosūtīt īsziņu. Toties operatora visu apkalpojamo numuru sarakstu atsevišķai personai saņemt nav atļauts. Numuru saglabāšanas programmatūra, kas aprīkota ar viedajām tehnoloģijām, spēj atklāt telefona numuru „skenēšanas" mēģinājumus, sastādīt operatora apkalpojamo telefonu sarakstu un nosūtīt īsziņu vai citādi ziņot par ekscesu atbildīgām personām. Līdzīgs risinājums pieejams pret pieslēgšanās paroles "skenēšanu".

2.7.3 Pieejamības pārraudzība

Sistēmas pieejamība, tāpat kā iepriekš minētā slepenība, ir kritiska informācijas sistēmas īpašība, Pieejamība ir iespēja lietot sistēmu tad, kad tas ir nepieciešams, bez pārrāvumiem un nevēlamiem traucējumiem.

Šīs prasības parasti cenšas nodrošināt, izveidojot atbilstošu infrastruktūru, nereti dublējot elektrības pieslēgumus, interneta pieslēgumus un pat serverus. Modernāku

risinājumu gadījumos tiek lietotas slodzes balansēšanas metodes [53]. Tās paredz izveidot attālinātu serveru un to pieslēgumu klasterus, kas atsevišķa servera atteikuma gadījumā automātiski spēj pārslēgties uz citiem serveriem. Tomēr sistēmas darbības pārtraukumi ir iespējami, par ko nekavējoties būtu ar īsziņu vai citādi jāinformē atbildīgās personas.

Neapšaubāmi, šādas iespējas ir iekļaujamas ar viedo tehnoloģiju īpašībām aprīkotos programmproduktos, un tās labi īstenojamas ar aģentu tehnoloģiju palīdzību [52].

Tomēr jāpiezīmē, ka mehāniska pārslēgšanās starp serveriem un personāla informēšana vēl neatrisina pašu sarežģītāko problēmu – datu sinhronizāciju starp vienlaicīgi darbinātiem serveriem, no kuriem viens ir pārtraucis savu darbību. Tas prasa papildus darbu un individuālu pieeju katram konkrētam gadījumam.

Pieejamības pārraudzība ir realizēta iepriekšminētajā Numuru piederības saglabāšanas datu bāzes projektā NUSA.

3. VIEDO TEHNOLOĢIJU APROBĀCIJAS

Viedo tehnoloģiju idejas rašanās un attīstības galvenais dzinulis jau no paša sākuma ir bijušas SIA „Datorikas institūts DIVI", kas ir viena no Latvijas progresīvākajām IT kompānijām, reālo projektu vajadzības. Pirmās viedo tehnoloģiju idejai pieskaitāmu īpašību realizācijas tapušas pirms vairāk kā 5 gadiem, tās vēlāk attīstītas, vispārinātas un implementētas plašāk izmantojamās iestrādnēs.

Pašlaik viedo tehnoloģiju principi dažādās automatizācijas pakāpēs un tehniskās realizācijās ir ieviesti un aprobēti daudzos SIA „Datorikas institūts DIVI" programmatūras izstrādes un ieviešanas projektos. Lai arī pagaidām praksē nav izdevies sastapt programmproduktus, kas būtu aprīkoti ar pilnu viedo tehnoloģiju īpašību kopu [4], tomēr, kā rāda pieredze, pat tikai atsevišķu īpašību iekļaušana programmatūrā var būt efektīva un nest reālu atdevi.

Lai radītu priekšstatu par viedo tehnoloģiju pielietošanas gaitā uzkrāto pieredzi un to pielietošanas aspektiem, šajā nodaļā īsumā raksturoti konkrēti SIA „Datorikas institūts DIVI" projekti, kuru ietvaros jau iepriekš izstrādāti programmprodukti tika papildināti ar atsevišķām viedo tehnoloģiju īpašībām, izveidojot jaunas, būtiski uzlabotas produktu versijas vai to komponentes.

3.1 Dinamisks biznesa modelis

2000. gadā SIA „Datorikas institūts DIVI" speciālisti uzsāka darbu pie t.s. „notikumu orientētās" informācijas sistēmas GVS izveides. Risinājuma uzdevums bija nodrošināt pašvaldības sociālās palīdzības notikumu (toreizējā leksikā – „gadījumu") uzskaiti. Veidojot GVS, izstrādātāji saskārās ar reālajā dzīvē bieži sastopamo problemātiku, ka dažādu pašvaldību dažādos sociālās palīdzības centros, lai arī tie darbojas saskaņā ar LR likumdošanas aktiem un nodrošina tos pašus pakalpojumus, darījumprocesi ir atšķirīgi. Tādēļ, lai sasniegtu plaši (ideālā gadījumā – visos Latvijas sociālās palīdzības centros) lietojamu programmatūras funkcionalitāti, bija nepieciešama "funkcionalitātes vispārināšana" - sistēmas atbalstīto procesu dinamiskas konfigurēšanas iespēju iekļaušana sistēmā.

Tas tika panākts, ieviešot notikuma jēdzienu. Atšķirībā no ierakstu orientētajiem reģistra tipa informācijas sistēmu risinājumiem, kuru "misija" ir uzkrāt tabulārus datus par sistēmas objektiem, notikumu orientēta sistēma papildus nodrošina arī lietotāju ar objektiem

veikto darbību uzkrāšanu un apstrādi. Bez "statiskās" datu lauku komponentes sistēma ieguva arī vēl "dinamisko", vēsturiskās ass komponenti. Informācijas sistēma kļuva ārēji konfigurējama – atkarībā no konkrētajam objektam piekārtotajiem iespējamajiem (pieļaujamiem) notikumiem mainījās arī datu kopa, kas par objektu uzkrājama, un pat apstrādes algoritmi. Tomēr iegūto risinājumu vēl nevarēja saukt par viedo tehnoloģiju principiem atbilstošu.

Notikuma jēdziens ir ļoti tuvs pārejas (*transition*) jēdzienam stāvokļu pārejas diagrammās; tas saturiski ir solis starp diviem dažādiem sistēmas objekta stāvokļiem. Līdz ar to dabiska bija izveidotā risinājuma koncepta paplašināšana ar stāvokļa jēdzienu. Atšķirībā no iepriekšējās realizācijas, kur notikums kalpoja konfigurēšanas vajadzībām, stāvokļa jēdziens kopā ar notikumu veido iespēju aprakstīt veselas darba plūsmas (*workflow*). Ar minētajām īpašībām aprīkotu programmproduktu SIA "Datorikas institūts DIVI" speciālisti nodēvēja par „notikumu orientētu sistēmu", un tieši šādā nozīmē šīs jēdziens lietots iepriekšējās šī darba nodaļās, kur raksturotas viedo tehnoloģiju īpašības un komponentes.

Pirmā notikumu orientētas sistēmas, kurā iekļauti abi – notikuma un stāvokļa – jēdzieni, implementācija tika izveidota Eiropas projektu pārvaldības risinājuma izstrādes projekta ietvaros, kas 2004. gada nogalē tika uzsākts pēc publiskā sektora klienta (par ERAF projektu ieviešanu un pārraudzību atbildīgas valsts aģentūras) pasūtījuma.

Vēlāk (laika posmā līdz 2008. gadam) risinājums tika pilnībā pārstrukturēts, izveidojot universālu darba plūsmu balstītu izpildes komponenti ("kodolu"), kas nodrošina patvaļīgas notikumu orientētas sistēmas darba plūsmu izpildi. Šo sistēmas attīstības brīdi var uzskatīt par viedo tehnoloģiju principu iekļaušanas sākumu. Bija izveidota informācijas sistēma, kas pēc savas konstrukcijas un uzbūves bija spējīga mainīties (gan no uzkrājamo datu, gan no sistēmā pieļauto darbību viedokļa) saskaņā ar ārpus sistēmas aprakstītu darījumprocesu modeli. Izveidotais risinājums sekmīgi tika ieviests vēl četrās radniecīgās valsts pārvaldes aģentūrās, kā arī izmantots trīs ar Eiropas struktūrfondu finansējuma pārvaldīšanu vispār nesaistītu notikumu orientētu informācijas sistēmu izveidē.

Kvalitatīvi jauns attīstības līmenis tika sasniegts 2009. gadā, kad augšminētā kodola programmatūra tika papildināta ar unikālu iespēju biznesa procesus raksturojošo informāciju automatizēti iegūt no darījumprocesu modelēšanas rīka repozitorijā atrodamā darījumprocesu apraksta. Jau iepriekšējos projektos sistēmas un lietotāju prasību noskaidrošanai un fiksēšanai tika izmantoti darījumprocesu grafiskie attēli, kuri tika veidoti īpaši konkrētajai sfērai raksturīgā domēnu specifiskā valodā (DSL jeb *domain specific language*). Tagad atbilstoši domēnu specifiskajai valodai ar rīku būves platformas GrTP palīdzību tika sagatavots speciāls

redaktors, kurā izstrādātāji sadarbībā ar klientu ne vien grafiskā formā varēja aprakstīt un saskaņot darījumprocesus, bet arī jau automātiski iegūt veidojamās informācijas sistēmas funkcionalitātes aprakstu. Kodola programmatūra bija spējīga no redaktora repozitorija izgūt darījumprocesu aprakstus un darboties saskaņā ar tiem.

Dinamiskā procesa modeļa izstrādes projektu grupa (turpmāk saukta par Projektu A) uzskatāmi pierāda viedo tehnoloģiju pielietošanas efektu. Uz konkrēto brīdi Projekts A aptver apmēram 10 klientu projektus, un tajos ieguldītais darba apjoms pārsniedz 20 persongadus. Pirmajā mirklī tas šķiet daudz, taču pat virspusēji darbietilpības aprēķini (balstoties uz zināšanām par izveidoto informācijas sistēmu funkcionalitāti un lietotāju īpašībām) liecina: ja minētās sistēmas tiktu izstrādātas pēc "tradicionālās" pieejas - individuāla vajadzību izzināšana, specificēšana, projektēšana, izstrāde, testēšana, atgriezeniskā saite, izmaiņu pieprasījumi - , kopējā darbietilpība nepieciešamās kvalitātes un apjoma sistēmu izstrādei būtu pat līdz 8 reizēm lielāka! Darbietilpības ietaupījums tika panākts jo īpaši specificēšanas un prasību saskaņošanas fāzē, kā arī kodēšanas darbos – liela daļa funkcionalitātes bija realizējama ar konfigurēšanas palīdzību.

Ņemot vērā Projekta A specifiku, kas nosaka vajadzību sniegt klientiem pirmā līmeņa atbalstu (*first level support* jeb tiešais gala lietotāju atbalsts), viedo tehnoloģiju pielietošana sistēmas izveidē ir radikāli samazinājušas problēmgadījumu skaitu un atvieglojušas pieteikumu apstrādes procesu. Ja, piemēram, būtu bijis jāuztur sešas, idejiski līdzīgas, bet no procesu viedokļa atšķirīgas informācijas sistēmas, kaut vai likumdošanas izmaiņu iekļaušana risinājumos vien nozīmētu sešas koda caurskates ar atbilstošu kļūdas risku, nepieciešamību atkārtoti pārtestēt un lietotājiem saskaņot izmaiņas. Tāpat ievērojami sarežģītos konfigurāciju pārvaldība, izmaiņu ienešana sistēma, kas konkrētajā realizācijā nozīmē darba plūsmu apraksta palabošanu, būtu saistīta ar reālu implementācijas darbu ar visām no tā izrietošajām sekām.

Projekta A ietvaros gūtās atziņas.

1. Idejas par viedo tehnoloģiju pievienošanu sistēmai attīstījās pakāpeniski, līdz ar to nebija iespējama viedo tehnoloģiju iekļaušana sistēmas sākotnējās izstrādes laikā.
2. Viedo tehnoloģiju iestrādi sistēmā atviegloja domēnu specifiskās valodas un grafisko atbalsta rīku (redaktoru) eksistence. Ļoti vērtīga bija grafisko atbalsta rīku izstrādātāju sniegtā iespēja automatizēti izmantot redaktora repozitorijā uzkrātos datus.

3. Viedo tehnoloģiju principu implementācija programmatūrā prasa mazāk resursu nekā individuālu, lai arī radniecīgu informācijas sistēmu izstrāde saskaņā ar konkrēto lietotāju prasībām.
4. Izmantojot viedās tehnoloģijas, būtiski tiek ietaupīti resursi, kas saistīti ar izmaiņu izstrādi un ieviešanu, kā arī paaugstinās klientiem nodrošinātais servisa līmenis.
5. Viedo tehnoloģiju principiem atbilstošas programmatūras izstrādi un apkalpošanu nav iespējams nodrošināt, ja nav pieejams pietiekami kvalificēts izstrādātāju sastāvs.
6. Ārēju piegādātāju izstrādātāju rīku (konkrētajā gadījumā – biznesa procesu modelēšanas redaktora) izmantošana ir saistīta ar risku, ka ārējais piegādātājs nav ieinteresēts vai spējīgs turpināt sadarbību ilgstošā laika periodā. Tas var apdraudēt produkta, kurš uz ārēju izstrādātāju rīkiem balstīts, tālāku attīstību vai pat tikai uzturēšanu.

3.2 Versiju pārvaldība un vides testēšana

Projekts B tika aizsākts 2002. gadā, un tā sākotnējais uzdevums bija izveidot finanšu sfēras risinājumu pēc konkrēta publiskā sektora klienta (ministrija) prasībām. Projekta darba uzdevums bija izveidot programmatūru, kas nodrošina strukturētu finanšu pārskatu apkopošanu vairākos hierarhijas līmeņos.

Taču jau drīz pēc pirmās versijas izveides un piegādes klientu loks sāka paplašināties, jo risinājuma funkcionalitāte bija nepieciešama arī citiem publiskā sektora klientiem. Laika posmā no 2003. gada janvāra līdz 2004. gada janvārim izveidotās programmatūras klientu skaits izauga no apm. 100 līdz 500. Ņemot vērā augošo klientu pieprasījumu, sākotnējā individuālā informācijas sistēma tika pārveidota par programmproduktu trīs dažādās komplektācijās (*lite*, *medium*, *business*), turklāt funkcionalitāte tika sadalīta divos dažādos, arī neatkarīgi izmantojamos funkcionālos moduļos – pārskatu sistēmā un plānu sistēmā. 2006. gada nogalē risinājumam (dažādās tā komplektācijās) bija aptuveni 1500 lietotāju.

Līguma ietvaros uzņēmuma uzdevums bija ne vien izveidot un piegādāt klientiem programmatūru, bet arī izmitināt sistēmas centrālo datu bāzi savā datu centrā un sniegt pirmā līmeņa atbalstu lietotājiem. Sistēmai līdz ar to bija jābūt darba spējīgai vairāk kā 600 valsts

iestādēs visā Latvijas teritorijā un jānodrošina regulāra informācijas apkopošana dažādos periodiskos griezumos. Turklāt lietojuma specifika noteica, ka lielākā daļa lietotāju noteiktos laika periodos ar sistēmu strādā visi vienlaikus.

Lai nodrošinātu programmatūras darba spēju ne tikai tajos Latvijas punktos, kur ir pieejams tiešsaistes interneta pieslēgums, bet arī vietās ar nestabilu, neregulāru un pat vispār neeksistējošu interneta pieslēgumu, sistēmas arhitektūra bija balstīta uz klienta-servera principiem ar centralizētu datu bāzi un lietojumprogrammatūru, kura atkarībā no interneta pieslēguma pieejamības spēj darboties gan tiešsaistē (*online*), gan autonomi (*offline*).

Ņemot vērā lielo lietotāju skaitu un to ģeogrāfisko izkliedētību, kā arī stingros līgumu nosacījumus un ierobežotos sistēmas ieviešanai un uzturēšanai atvēlētos finanšu un darbaspēka resursus, izstrādātāji pieņēma lēmumu programmatūru aprīkot ar konkrētu viedo tehnoloģiju īpašību - automatizētu versiju izplatīšanu.

Programmatūrā tika iebūvēta dažādo sistēmas komponenšu (izpildāmie moduļi, konfigurējošie dati, veidlapas, atskaites u.c.) versiju pārvaldības modulis, kas tika izplatīts kopā ar sistēmu, līdz ar to spiestā kārtā pirmajā uzstādīšanas reizē tika instalēts uz klienta darba stacijas. Vēlākā ekspluatācijas laikā attiecīgais programmatūras mehānisms pirms katras sistēmas darba sesijas uzsākšanas pieslēdzās centrālajam serverim, secīgi pārbaudīja uz klienta darba stacijas uzstādīto sistēmas komponenšu esamību un versiju numurus, salīdzināja tos ar centrālajā serverī norādītajiem un neatbilstību gadījumā lejupielādēja un pieinstalēja jaunās/ uzlabotās komponentes uz klienta darba stacijas. Ja konkrētajam datoram nebija pieejams internets, programma šo atjaunošanas procedūru varēja veikt vēlāk, kad internets būs pieejams.

Izveidotais risinājums jau ar tā pirmo versiju bija būtisks atspaids gan izstrādes grupai, gan atbalsta speciālistiem, jo strauji samazinājās pieteikumu skaits, kuru iemesls bija nepilnīgi uzstādījušās, trūkstošas un neaktuālas komponentes. Ne vien samazinājās konsultāciju skaits par sistēmas uzstādīšanu, bet arī to ilgums, jo vairs nebija nepieciešams detalizēti pa telefonu skaidrot par dažādo komponenšu atrašanās vietu un jauno versiju lejupielādi un instalēšanu.

Pēc aptuveni gada tika pieņemts lēmums programmatūrai pievienot vēl vienu viedo tehnoloģiju komponenti – ārējās vides parametru automatizētās pārbaudes iespēju. Tās nepieciešamība izrietēja no jau sekmīgi produkcijā esošās versiju izplatīšanas komponentes, jo lietotāji bija pieraduši, ka programmatūra pati sevi atjaunina, tādēļ nespēja izprast, ka ir

situācijas, kad atsevišķu komponenšu uzstādīšana nav iespējama bez zināšanām par konkrētās darba stacijas aprīkojumu ar citām (standarta) informācijas sistēmām.

Ārējās vides pārbaudes modulis arhitektoniski tika veidots kā atsevišķs modulis, kurš ir pieinstalējams arī pēc pašas sistēmas uzstādīšanas un lietojams autonomi dažādu informācijas sistēmu kontekstā. Attiecīgais programmprodukts nodrošināja norādītu vides, kurā tas ir uzstādīts, parametru vērtību pārbaudi un salīdzināšanu ar iedefinētajiem nosacījumiem. Tipiskākie pārbaudes objekti bija operētājsistēmas versijas, MS Office biroja paketes versijas, datora reģionālie iestatījumi u.c.

Tā kā programmatūra, kas nodrošināja sistēmai uzstādītās funkcionālās īpašības, bija izstrādāta jau iepriekš, to modernizācijas ietvaros tika veikta programmatūras pārveide atbilstoši automātiskās versiju aktualizēšanas prasībām. Sistēmas pārveidei nepieciešamā papildus darbietilpība sastādīja apmēram 20% no sākotnējā produkta izstrādes darbietilpības, daļa no tās tika investēta apjomīgā testēšanā un koda apskates (*code review*) pasākumos. Lai pārliecinātos par izstrādāto mehānismu darbības pareizību, tika veikta programmatūras testēšana dažādās infrastruktūras konfigurācijās ar virtuālo mašīnu palīdzību.

Rezultātā bija izveidota uzlabota programmatūras versija, kura nodrošina pilnīgi automatizētu programmatūras uzstādīšanu attālināti, turklāt ļoti atšķirīgās mērķa vidēs: atšķirīgas komplektācijas datorsistēmās, dažādās operētājsistēmās, sadarbība ar dažādās ofisa programmatūras versijām utt. Ārējās vides parametru automatizētās pārbaudes iespēja tika veidota kā neatkarīgs, bet konfigurējams modulis, kurš pirms programmatūras izplatīšanas „izlūko" mērķa vidi un informē lietotājus un/ vai izstrādātājus par vides specifiku un atbilstību minimālām tehniskām prasībām.

Viedo tehnoloģiju īpašības iestrāde programmatūrā notika pakāpeniski – vairāk kā divu gadu garumā. Šajā laika posmā lietotāju skaits pieauga no apmēram 1500 2006. gadā līdz 3000 2008. gadā, un vidējais apgrozījums gadā pieauga no 0,3 Mio EUR 2006. gadā līdz 0,5 Mio EUR 2008. gadā.

Iegūtais rezultāts pārspēja gaidīto - automātiskās versiju aktualizēšanas un ārējās vides parametru automatizētās pārbaudes īpašības būtiski ir atvieglojušas sistēmas uzstādīšanu, uzturēšanu un samazinājusi lietotāju atbalstam nepieciešamos konsultantu resursus.

Projekta B ietvaros gūtās atziņas.

1. Viedo tehnoloģiju pievienošana sistēmai pēc tās sākotnējās izstrādes ir lietderīga, taču prasa vairāk resursu nekā būtu nepieciešami, ja viedo tehnoloģiju iespējas arhitektūrā tiktu paredzētas jau projektēšanas laikā.
2. Viedo tehnoloģiju principu implementācija programmatūrā prasa mazāk resursu nekā būtu nepieciešams pilna dažādo vides konfigurāciju spektra nodrošināšanai. Vienlaikus viedo tehnoloģiju izmantošana uzliek daudz mazāk ierobežojumu uz pieļaujamajiem izteiksmes līdzekļiem.
3. Būtiski tiek ietaupītas programmatūras uzstādīšanai veltāmās pūles (laiks, resursi) un paaugstinās klientiem nodrošinātais servisa līmenis.
4. Izmantojot viedās tehnoloģijas, iespējams nodrošināt programmatūras darbaspējas laikā mainīgā un no platformu un infrastruktūras viedokļa nehomogēnā vidē, tomēr viedo tehnoloģiju mehānismi regulāri ir jāadaptē atbilstoši ārējās vides (īpaši – standarta programmatūras) izmaiņām.
5. Ļoti svarīgi ir sistēmā iekļaut izsmeļošu ziņošanas mehānismu, kas izstrādātājus jau laikus informē par konstatētajām problēmām.
6. Pat lietotāju darbu atvieglojoši mehānismi sākumā lietotājiem speciāli jāizskaidro, jo tie ir pretrunā ar līdzšinējo lietotāju pieredzi, kad programmatūras uzstādīšana ir saistīta ar fizisku cilvēka parādīšanos.

3.3 Versiju pārvaldības un datu sinhronizēšana

Projekts C tika aizsākts 2008. gadā, un tā uzdevums bija izveidot vairāk nekā 10 gadus veca produkta pilnīgi jaunu versiju (pārrakstīt to). Tā kā produkta lietotāju skaits bija vairāki desmiti, jau jaunās versijas projektēšanas laikā tika pieņemts lēmums risinājumā iekļaut projekta B sekmīgo pieredzi.

Tādēļ papildus pamata funkcionalitātei tika plānots iekļaut arī automatizētās versiju izplatīšanas iespēju un paštestēšanas iespējas elementus. Tuvākas analīzes ietvaros tika konstatēts, ka nepieciešama arī datu sinhronizēšanas iespēja, kas ļautu efektīvi izplatīt centrālajā datu bāzē noglabātu lietojumprogrammas konfigurāciju un klasifikatorus, savukārt no lokālām datu bāzēm - savākt lietotāju ievadītus datus.

Modernizācijas darbi, ieskaitot pamata funkcionalitātes pilnīgu pārrakstīšanu, prasīja aptuveni 8 personmēnešus. No tiem apmēram 50% tika tērēti datu sinhronizēšanas iespējas adaptācijai (šīs viedo tehnoloģiju funkcijas iestrādnes jau bija uzņēmuma rīcībā). Tas apdraudēja projekta budžetu, kā rezultātā uzņēmums bija spiests atteikties no plānotās paštestēšanas mehānismu iekļaušanas risinājumā.

Tomēr produkts tika izveidots un pašlaik jau tiek aktīvi izplatīts tirgū. Izveidotais automatizētās programmatūras sākotnējās uzstādīšanas risinājums diemžēl nebija tik veiksmīgs kā cerēts, jo vairumā gadījumu vidu nesaderības un datu bāzes vadības sistēmas specifikas dēļ tomēr bija nepieciešama izstrādātāja aktīva iesaistīšanās uzstādīšanas procesā. Taču jau uzstādītu versiju gadījumā datu sinhronizācija un programmatūras jauninājumu izplatīšana strādāja nevainojami, patiešām atvieglojot uzturēšanu un samazinot lietotāju atbalstam tērējamos resursus.

Projektu rezultātu analīze, kas tika veikta pēc produkta pabeigšanas, parādīja, ka galvenos zudumus projektā C radīja izvēlētā tehniskā pieeja datu sinhronizēšanai. Projektā B versiju izplatīšana tika vadīta no specifiska konfigurācijas faila, kurā specifiskā valodā bija pierakstītas uzstādīšanas ietvaros veicamās pārbaudes un uzstādīšanas komandas un kuru programmatūra spēja adekvāti interpretēt. Projektā C savukārt bija izvēlēta uz konkrētās datu bāzes vadības sistēmas iespējām bāzēta versiju un datu sinhronizēšanas pieeja. Faktiski tika nodublēts konkrētajā datu bāzes vadības sistēmā jau iebūvēts datu izplatīšanas mehānisms, turklāt izvēlētās datu bāzes vadības sistēmas smagnējības dēļ ievērojami apgrūtināts arī no projekta B aizņemtais versiju izplatīšanas mehānisms.

Projekta C ietvaros gūtās atziņas.

1. Lēmums par to, kad viedo tehnoloģiju iespējas pievienot sistēmai (jau tās sākotnējās izstrādes laikā vai vēlāk), ir mazāk svarīgs nekā konkrētais tehniskais risinājums.
2. Veidojot viedo tehnoloģiju funkcijas, ieteicams izmantot pārbaudītus, pierastus, standartizētus, robustus arhitektūras un tehniskus risinājumus – tas samazinās vienlaikus risināmo problēmu skaitu un uzlabos izstrādes procesa vadāmību.
3. Pasūtītāji, lai arī atzinīgi novērtē viedo tehnoloģiju sniegtās iespējas, parasti nav gatavi par tām papildus maksāt.

4. Viedo tehnoloģiju iespējas komerciālos projektos atmaksājas, ja ar pasūtītāju ir ilgtermiņa sadarbība, ja lietotāju skaits ir liels un to darba staciju konfigurācijas ir atšķirīgas.

3.4 Paštestēšana

Lai arī paštestēšanas ideja SIA „Datorikas institūts DIVI" pieredzējušajiem kolēģiem (skatīt arī aiznākamajā nodaļā aprakstītā kvalitatīvā pētījuma atziņas) nav sveša, darbs pie konkrētas implementācijas SIA "Datorikas institūts DIVI" projektu ietvaros tika uzsākts tikai aizvadītā gada vidū, pateicoties inovāciju programmas ietvaros piesaistītajam ERAF līdzfinansējumam.

Paštestēšanas komponente (Projekts D), kas ir būtiska viedo tehnoloģiju īpašību saimes sastāvdaļa, tika veidota (un joprojām tiek turpināta veidot) pakāpeniski , vadoties no reālos projektos identificētām vajadzībām. Lai arī paštestēšanas idejiskā nostādne iesaistītajiem bija skaidra un arī formulēta jau vairākās zinātniskās publikācijās starptautiski atzītos izdevumos, pirmā realizācija tapa smagi un iegūtais rezultāts pagaidām nav komerciāli izmantojams.

Galvenais dzinējspēks paštestēšanas implementācijai konkrētā sistēmā bija SIA "Datorikas institūts DIVI" banku sektora klienta vēlme atvieglot piegādāto nodevumu testēšanas procesu un kopumā uzlabot programmproduktu kvalitāti. Banku sektora klientu prasības pret programmatūras piegādes un akcepttestēšanas procesu tradicionāli ir ļoti augstas, un katra jaunas versijas ieviešana no gala lietotāju viedokļa ir saistīta ar pilnīgu sistēmas funkcionalitātes pārtestēšanu. Lieki piebilst, ka tas ir smags un darbietilpīgs process, kurš diemžēl tomēr nesniedz pilnīgu garantiju par ieviestās sistēmas versijas pilnīgu atbilstību vajadzībām. Savukārt ārkārtas labojumu ("ielāpu") uzlikšana ekspluatācijā jau esošai sistēmai ir saistīta ar bankas augstākās vadības iejaukšanos un attiecīgo darbinieku reputācijas iedragāšanu. Tādēļ viena no bankām, kas nu jau vairāk kā 15 gadus ir SIA "Datorikas institūts DIVI" klients, izteica aktīvu vēlmi izmantotajā programmproduktā redzēt paštestēšanas iespējas.

Jau paštestēšanas komponentes plānošanas laikā gan ierobežoto resursu, gan tehnisko nosacījumu dēļ tika identificēti ierobežojumi, pie kuriem veidojamā paštestēšanas realizācija būtu darbināma:

- tika konstatēts, ka ir grūtības instrumentēt *Centura* vidē izveidotās sistēmas komponentes, tādēļ testa punktus tika paredzēts ievietot tikai .NET vidē izstrādātajās komponentēs;

- problēma ar savstarpēji saistītajiem mainīgajiem ekrāna formās (daži lauki tiek aprēķināti no citu vērtībām, turklāt iespējama savstarpēja pārrēķināšana vairākos "virzienos") tiek ignorēta, atstājot testu definētāja ziņā lauku aizpildīšanas secību un vērtības;
- datu bāzes sagatavošana (pārbaude, vai ir nodrošināti testa izpildei nepieciešamie sākumdati) netiek nodrošināta – pagaidām tiek pieņemts, ka testa definētājs veidos tādus testus, kuru ietvaros datu bāzē tiek sagatavota (definēta) vēlāku pārbaužu nodrošināšanai nepieciešamā pamata informācija (piemēram, klasifikatori, izvēles saraksti u.tml.).

Iegūtais prototips pašlaik ir validēšanas stadijā, tomēr jau tagad var tikt izdarīti pirmie secinājumi.

1. Paštestēšanas mehānismu iestrāde programmatūrā prasa ne vien labas zināšanas attiecīgā programmprodukta uzbūves jautājumos, bet arī darījumloģikā, jo testa punkti tiek ievietoti programmas kodā vietās, kas „atbildīgas" par kritiski svarīgo funkcionalitāti.
2. Paštestēšanas komponenti testu uzkrāšanas režīmā pilnvērtīgi varēs lietot tikai kompetenti lietotāji, kas spēj nodefinēt kritisko funkcionalitāti pārklājošu testu kopu, kas turklāt ir "sevī noslēgta", resp., aptver ne vien konkrētas funkcijas notestēšanu, bet arī tam nepieciešamo sākumdatu sagatavošanu.
3. Paštestēšanas realizācija prasa ievērojamu darbietilpību, un pasūtītāji, pat ja apzinās šādu programmatūras uzlabojumu potenciālos ieguvumus, nav spējīgi vai gatavi par šiem uzlabojumiem maksāt.

3.5 Pārskats par "gudro tehnoloģiju" projektu

3.5.1 Projekta pamatojums

Versiju izplatīšanas komponente SIA „Datorikas institūts DIVI" produktos un projektos pagaidām ir visplašāk izmantotā un līdz ar to arī visattīstītākā no viedo tehnoloģiju komponentēm.

Ņemot vērā Latvijā pastāvošās objektīvās problēmas ar lietotāju izglītības līmeni un pieejamo tehnisko infrastruktūru (jo īpaši – interneta pieslēgumu), tika konstatēts, ka sadalītu risinājumu izplatīšana un ieviešana visā valsts teritorijā ir ļoti apgrūtināta. Lai nodrošinātu

kvalitatīvu pakalpojumu, informācijas tehnoloģiju risinājumu piegādātājiem ir vai nu pašiem jānodarbojas ar programmatūras uzstādīšanu (ieskaitot vietējo datortīklu problēmu novēršanu, kas formāli vispār neattiektos uz programmatūras izplatīšanas sfēru) un lietotāju apmācīšanu, vai jānodrošina tiešās uzturēšanas serviss (*hotline*) ar gariem darba laikiem un augstas kvalifikācijas konsultantiem. Īpaša smaga ir attālinātās diagnostikas problēma – bieži vien telefoniski ir gandrīz neiespējami noskaidrot lietotāju pieteikto problēmu iemeslus, jo lielākoties to cēloņi slēpjas atšķirīgā tehniskās vides konfigurācijā un/vai datortīkla un interneta pieslēguma nestabilitātē.

Tādēļ 2006. gadā SIA „Datorikas institūts DIVI" iesniedza pieteikumu uz ERAF atbalstu inovāciju projekta izpildei, kas paredzēja inovāciju pielietošanu, lai uzlabotu mūsdienīgu informācijas tehnoloģiju risinājumu pieejamību Latvijas attālākajos reģionos un atpalikušajos segmentos ar viedo tehnoloģiju risinājuma palīdzību.

Inovāciju projekta ietvaros automātiskās versijas izplatīšanas viedo tehnoloģiju īpašību bija plānots ieviest notikumu orientētās sistēmās, kurās tiek nodrošināta laikā secīgu notikumu reģistrācija un apstrāde. Tika sagaidīts, ka izveidotā tehnoloģija atvieglos programmatūras izplatīšanu, uzstādīšanu un uzturēšanu, kā arī būtiski uzlabos programmatūras uztveramību un spēju pielāgoties konkrētā klienta infrastruktūrai. Par projekta mērķi tika izvirzīts, ka jaunās programmatūru versijas samazinās lietotāju atkarību no tehniskās infrastruktūras, palīdzēs ietaupīt līdzekļus programmu uzturēšanai un apkalpošanai, kā arī uzlabos lietotāju darba funkciju izpildes efektivitāti.

ERAF līdzfinansētā projekta „Gudrās programmatūras tehnoloģijas implementācija notikumu orientētās informācijas sistēmās", kas 2006.-2008. gadā tika realizēts Valsts atbalsta programmā „Atbalsts jaunu produktu un tehnoloģiju attīstībai" un tās apakšprogrammā „Atbalsts jaunu produktu un tehnoloģiju attīstībai", ietvaros minētā versiju izplatīšanas komponente tika ne vien specificēta un izstrādāta, bet arī aprobēta četrām sfērām, kurās SIA „Datorikas institūts DIVI" ir konkrēti klienti un projekti.

3.5.2 Projekta fāzes

Lai raksturotu viedo tehnoloģiju izstrādei nepieciešamo darbietilpību, zemāk dots konspektīvs projekta „Gudrās programmatūras tehnoloģijas implementācija notikumu orientētās informācijas sistēmās" plāns ar attiecīgo etapu darbietilpībām personmēnešos.

1. etaps: Sistēmanalīze (izpildes termiņš: 4 mēneši, darbietilpība: 23 personmēneši)

Sistēmanalīzes ietvaros tika specificētas veidojamā risinājuma darījumprasības un paralēli izstrādāts risinājuma projektējums. Etapa rezultātā tika izveidots prasību specifikācijas dokuments, kā arī ir izveidota sistēmas arhitektūra – nodefinēta datu bāzes struktūra, kā arī izveidota programmatūras izstrādes vide.

2. etaps: „Gudrās programmatūras" tehnoloģijas izstrāde (izpildes termiņš: 6 mēneši, darbietilpība: 70 personmēneši)

Šī etapa ietvaros tika izstrādāts „gudrās programmatūras" tehnoloģijas kodols, resp., modulis, kuru nākamajos projekta etapos pielāgos konkrēto risinājumu vajadzībām. Etapa rezultātā ir izveidots un notestēts strādājošs tehnoloģijas modulis, kurš nodrošina sinhronizācijas un versiju sinhronizēšanas mehānismus, kā arī iebūvētu konfigurācijas pārbaudi.

3. etaps: Adaptācija VAAD IS (izpildes termiņš: 5 mēneši, darbietilpība: 21 personmēnesis)

Šī etapa ietvaros iepriekšējā etapa rezultātā izstrādātais „gudrās programmatūras" tehnoloģijas kodols tika iekļauts Valsts augu aizsardzības dienesta informācijas sistēmā, jo īpaši akcentējot jauno versiju un datu automātiskās izplatīšanas iespēju. Etapa rezultātā bija izveidota un notestēta jauna (beta) VAAD IS versija, kurā bija realizētas viedās tehnoloģijas īpašības.

4. etaps: Aprobācija VAAD IS (izpildes termiņš: 4 mēneši, darbietilpība: 14 personmēneši)

Iepriekšējā etapa rezultātā izstrādātā VAAD IS jaunā (beta) versija tika aprobēta pie izvēlētiem lietotājiem – tika nodrošināta izveidotās sistēmas versijas kvalifikācijas testēšana, validācija, tika apkopoti un apstrādāti lietotāju problēmziņojumi (kļūdu pieteikumi,

ierosinājumi, izmaiņu pieprasījumi). Etapa rezultātā tika izveidota VAAD IS (gala) versija un nodrošināta tās ieviešana pie visiem VAAD IS lietotājiem.

5. etaps: Adaptācija sociālajai sfērai (izpildes termiņš: 9 mēneši, darbietilpība: 51 personmēnesis)

Šī etapa ietvaros „gudrās programmatūras" tehnoloģijas kodols tika izmantots, lai izveidotu būtiski uzlabotas versijas diviem sociālās sfēras risinājumiem – ACIS un GVS-2 -, jo īpaši akcentējot konfigurācijas pārbaudes īpašības un iekļaujot VAAD IS jaunās versijas aprobācijas ietvaros gūtās atziņas. Etapa rezultātā tika izveidotas un notestētas jaunas (beta) versijas ACIS un GVS-2, kurās realizētas viedās tehnoloģijas.

6. etaps: Aprobācija VAAD IS (izpildes termiņš: 3 mēneši, darbietilpība: 27 personmēneši)

Iepriekšējā etapa rezultātā izstrādātās jaunas (beta) versijas ACIS un GVS-2 tika aprobētas pie izvēlētiem lietotājiem. ACIS tika darbinātas divās izvēlētās sociālās aprūpes iestādēs, bet GVS-2 tika testēts trīs dažāda lieluma pašvaldību sociālajos dienestos. Etapa ietvaros tika apkopoti un apstrādāti lietotāju problēmziņojumi (kļūdu pieteikumi, ierosinājumi, izmaiņu pieprasījumi). Etapa rezultātā tika izveidotas ACIS un GVS-2 (gala) versijas un nodrošināta to izplatīšana visā LR teritorijā.

7. etaps: Adaptācija budžetu pārskatu sastādītājiem (pašvaldību grāmatveži) (izpildes termiņš: 6 mēneši, darbietilpība: 22 personmēneši)

Šī etapa ietvaros „gudrās programmatūras" tehnoloģijas kodols tika iekļauts Valsts kases noteikto budžetu pārskatu (gada, mēnešu, ceturkšņu) sastādīšanas risinājumā PBP, jo īpaši akcentējot versiju izplatīšanas īpašības un iekļaujot iepriekšējo aprobācijas etapu ietvaros gūtās atziņas. Etapa rezultātā tika izveidota un notestēta jauna PBP versija, kurā realizētas viedās tehnoloģijas.

8. etaps: Aprobācija pie budžetu pārskatu sastādītājiem (izpildes termiņš: 5 mēneši, darbietilpība: 17 personmēneši)

Iepriekšējā etapa rezultātā izstrādātā jaunas (beta) PBP versija tika aprobēta pie izvēlētiem lietotājiem trīs dažādās pašvaldībās – lielā pilsētā, rajona padomē un pagasta pašvaldībā. Etapa laikā tika apkopoti un apstrādāti lietotāju problēmziņojumi (kļūdu

pieteikumi, ierosinājumi, izmaiņu pieprasījumi). Etapa rezultātā izveidota PBP (gala) versija un nodrošināta tās izplatīšana visā LR teritorijā.

9. etaps: Adaptācija budžetu pārskatu sastādītājiem (ministriju grāmatveži) un finanšu plānotājiem (izpildes termiņš: 6 mēneši, darbietilpība: 20 personmēneši)

Šī etapa ietvaros „gudrās programmatūras" tehnoloģijas kodols tika iekļauts Valsts kases noteikto budžetu pārskatu (gada, mēnešu, ceturkšņu) un finanšu plānošanas risinājumā FIBU, jo īpaši akcentējot versiju izplatīšanas īpašības un iekļaujot iepriekšējo aprobācijas etapu ietvaros gūtās atziņas. Etapa rezultātā tika izveidota un notestēta jauna FIBU versija, kurā realizētas viedās tehnoloģijas.

10. etaps: Aprobācija pie budžetu pārskatu sastādītājiem un finanšu plānotājiem (izpildes termiņš: 5 mēneši, darbietilpība: 16 personmēneši)

Iepriekšējā etapa rezultātā izstrādātā jaunas (beta) FIBU versija tika aprobēta pie izvēlētiem lietotājiem divās dažādās valsts iestādēs – vienā ministrijā un vienā pakļautajā iestādē, kam savukārt ir pakļautas struktūrvienības. Etapa laikā tika apkopoti un apstrādāti lietotāju problēmziņojumi (kļūdu pieteikumi, ierosinājumi, izmaiņu pieprasījumi). Etapa rezultātā tika izveidota FIBU (gala) versija un nodrošināta tās izplatīšana visā LR teritorijā.

3.5.3 Rezultātu raksturojums

Augšminētā inovāciju projekta rezultāti ir pārliecinoši un ar augstu praktisku pielietojumu. Tika izveidots konkrēts risinājums, kas nodrošina viedo tehnoloģiju principus notikumu orientētās informācijas sistēmās, un organizēta izveidotā risinājuma plaša pielietošana Latvijā gan publiskajā, gan privātajā sektorā.

Izveidotais risinājums ļāva lietojumprogrammai (*desktop application*), izmantojot interneta pieslēgumu, pašai veikt programmatūras versiju un datu atjaunināšanu, kā arī pārbaudīt sev pieejamās vides atbilstību prasībām un veikt darbaspējas pārbaudi ar iebūvēta mehānisma palīdzību. Atšķirībā no „klasiskajām" tīmekļa lietojumprogrammām, kuras ir korekti darbināmas tikai tad, ja ir pieejams stabils interneta pieslēgums un jaudīgs centrālais serveris, ar viedo tehnoloģiju aprīkotās sistēmas spēj efektīvi darboties arī tad, ja interneta pieslēgums ir nestabils (ar pārrāvumiem) vai temporāls (uz noteiktu brīdi, piemēram,

iezvanpieejas gadījumā), turklāt piedāvājot lietojumprogrammatūrai raksturīgo izteiksmes bagātību.

Iegūtās atziņas apstiprināja augstāk raksturotajos projektos konstatēto.

1. Viedo tehnoloģiju implementācija ir iespējama tikai tad, ja tiek iesaistīti kvalificēti speciālisti, vēlams – ar iepriekšēju pieredzi viedo tehnoloģiju vai radniecīgu izstrādņu veidošanā.
2. Izvēlētajam tehniskajam risinājumam jābūt robustam un uzturamam, vēlams atteikties no konkrētām izstrāde vidēm raksturīgiem specifiskiem paņēmieniem.
3. Klientiem, kuru projektos izmantotas viedās tehnoloģijas, jāsniedz papildus informācija, lai paskaidrotu izmaiņas produkta uzturēšanas procesā un parādītu abpusējos ieguvumus.
4. Ieguldījumi, kas saistīti ar viedo tehnoloģiju implementāciju un ieviešanu, atmaksājas tikai ilgtermiņa sadarbības un savstarpējas uzticēšanās gadījumā. Galvenais dzinējspēks viedo tehnoloģiju ieviešanā tomēr ir pats izstrādātājs, nevis klienti.

4. EFEKTIVITĀTES NOVĒRTĒŠANA

Pieredze rāda [5], ka lēmumu par viedo tehnoloģiju iekļaušanu sistēmā var pieņemt ne tikai programmatūras sākotnējās izstrādes ietvaros, bet arī vēlākā izstrādes vai uzturēšanas posmā – piemēram, pirms kārtējās būtiski izmainītās versijas izlaišanas vai speciālu pasākumu ietvaros.

Tomēr, lai izšķirtos par viedo tehnoloģiju izmantošanu, izstrādātājiem ir nepieciešams veikt potenciālo ieguvumu/ zaudējumu analīzi, kā arī izvērtēt konkrētā produkta atbilstību virknei kritēriju, kas varētu apstiprināt vai noliegt viedo tehnoloģiju pielietošanas mērķtiecību konkrētajā gadījumā. Vārdu sakot, ir jāspēj kvantitatīvi un/ vai kvalitatīvi novērtēt (izmērīt) programmatūras uzlabošanas pasākumu sagaidāmo efektivitāti.

Turpmāk aprakstītas autores idejas, no kādiem aspektiem un ar kādiem parametriem ir jānovērtē viedo tehnoloģiju iekļaušanas informācijas sistēmā efektivitāte.

4.1 Pamatjēdzieni

Runājot par efektivitāti, bieži vien efektivitāte tiek jaukta ar produktivitāti un rezultativitāti, tādēļ pirmkārt ir jānoskaidro šī jēdziena patiesā būtība.

Efektivitātes (*effectiveness*) jēdziens literatūrā tiek skaidrots kā „spēja būt efektīvam; spējas sasniegt efektu kvalitāte" [54], arī kā „pakāpe, kādā programma vai serviss sasniedz tam uzstādītos mērķus" [55].

Atšķirībā no rezultativitātes (*efficiency*), kura apzīmē konkrētu rādītāju sasniegšanas pakāpi un pēc savas būtības ir tuva produktivitātes jēdzienam, efektivitāte ir objekta vai procesa spēja ilgtermiņā piepildīt iepriekš uzstādītus mērķus pie noteiktiem nosacījumiem. Abi jēdzieni, lai arī saturiski it kā tuvi, atšķiras ar kvalitātes aspektu – rezultativitāte pēc savas būtības tiek raksturota ar daudzumu laikā, kamēr efektivitāti drīzāk raksturo kvalitatīva ietilpība, spēja sasniegt izvirzītos mērķus optimālā veidā.

Efektivitāte drīzāk liecina par kādas iekārtas, pakalpojuma, īpašības utt. potenciālu, nevis par to reālajām īpašībām (faktiskajām vērtībām). Līdz ar to aktualizējas jautājums, kā efektivitāti novērtēt, kā prognozēt, piemēram, viedo tehnoloģiju iekļaušanas programmatūrā efektivitāti .

Novērtēšana literatūras avotos tiek aprakstīta kā „sistemātiska snieguma, nozīmīguma un vērtības pārbaude, salīdzinot ar standartu kopu" [56]. Efektivitātes novērtēšanas procesa priekšnoteikums līdz ar to ir standartu kopas esamība, kuras vērtības liecinātu par vēlamo efektivitātes pakāpi, ar kuru būtu salīdzināmas reālās vērtības.

Izplatītākā, taču ne vienīgā iespējamā novērtēšanas forma ir mērīšana - paņēmienu kopums, kas tiek pielietots, lai kvantitatīvi un/ vai kvalitatīvi novērtētu objekta vai procesa atbilstību noteiktiem kritērijiem un to vēlamajām vērtībām [57].

Efektivitātes novērtēšanas sagatavošana katrā konkrētā gadījumā ietver:

1. efektivitāti raksturojošu kritēriju izvēle (standartu kopa);
2. kritēriju optimālo vērtību izvēle (standartu kopas etalons);
3. novērtēšanas (mērīšanas) metodoloģija (procesa apraksts).

Efektivitātes novērtēšana (mērīšana) savukārt nozīmē:

1. izvēlēto kritēriju vērtību nolasīšana /aprēķināšana saskaņā ar novērtēšanas metodoloģiju;
2. iegūto vērtību salīdzināšana ar etalona vērtībām;
3. secinājumi no vērtību atbilstības/ neatbilstības vēlamajām (etalona) vērtībām.

Turpmākajās nodaļās tiks diskutēti konkrētos programmatūras izstrādes projektos empīriski identificēti viedo tehnoloģiju efektivitātes novērtēšanas kritēriji un to iespējamās vērtības, kas ļautu vēl pirms viedo tehnoloģiju iekļaušanas risinājumos novērtēt sagaidāmo pasākumu efektivitāti.

4.2 Ekonomiskie kritēriji

Katrā komerciālā projektā primārais ir potenciālās atdeves (peļņas) jautājums. Ikviens racionāls tirgus dalībnieks būs gatavs investēt tikai tad, ja nākotnē sagaidāmā naudas plūsma (*cash flow*), ieskaitot kapitāla lietošanas procentus, pārsniedz ieguldījumus, kas jāveic [58]. Tātad ir nepieciešams izveidot investīciju atgūšanas plānu, kurā sākotnējās investīcijas tiktu salīdzinātas ar nākotnes naudas plūsmu (*discounted future cash flows*).

Šādu lielumu ekonomikā sauc par ROI (*return on investment*), un tas ir plaši lietots rādītājs biznesa plānu rentabilitātes novērtēšanai, investīciju plānošanai u.tml. Tā kā ROI ir aprēķināms lielums (kurš gan, kā vēlāk būs redzams, balstās uz aptuvenām izejas vērtībām!), tas labi var kalpot par kritēriju ekonomiskās efektivitātes novērtēšanā.

ROI aprēķins balstās uz investīciju potenciāla novērtēšanu (sagaidāmās atdeves nozīmē) un salīdzināšanu ar veicamajām investīcijām. ROI tiek definēts kā attiecība starp vidējo sagaidāmo nākotnes naudas plūsmu projekta laikā un sākotnējām investīcijām [59].

4.2.1 Sākotnējo investīciju novērtēšana

Investīciju projektos, kur tiek iegādātas preces vai pakalpojumi, kuru apjoms, kvalitāte, un līdz ar to arī cena, ir novērtējami tirgus izpētes ceļā, sākotnējo investīciju apjoma novērtēšana nav ļoti sarežģīts uzdevums. Tas parasti ir noteikts naudas vienībās mērāms lielums, kurš raksturo naudas daudzumu, kas investoriem jāiegulda, lai projekts vispār varētu sākties.

Informācijas tehnoloģiju nozare turpretī izceļas ar lielu nenoteiktību, jo pakalpojumu cenu amplitūda ir ļoti plaša, ražojošā resursa atdeve ir grūti izmērāma un ļoti reti sagaidāmā darba apjoms ir pilnīgi precīzi paredzams, pirms darbs vēl uzsākts. Izstrādājot viedo tehnoloģiju komponentes un/vai integrējot tās ar esošām vai jaunos moduļos, faktiski vienīgā investīciju pozīcija ir izstrādātāju darbs, resp., investīciju novērtēšana būtu veicama, reizinot izstrādei/ pielāgošanai nepieciešamo darbietilpību ar attiecīgās vienības pašizmaksu.

Nepieciešamās darbietilpības novērtēšanai ar tikt izmantota grūti formalizējamā uz līdzīgos projektos gūtas iepriekšējās pieredzes balstīta pieeja vai kāda no industrijā pieņemtajām formalizētajām darbietilpības novērtēšanas metodēm. Viena no biežāk citētām metodēm ir COCOMO metode [60], kas implementēta daudzos un dažādos internetā pieejamos kalkulatoros.

Viedo tehnoloģiju gadījumā COCOMO metode noteikti nav pielietojama „plakani", resp., vērtējot darbietilpību, noteikti ir jāņem vērā vismaz šādi koriģējoši faktori:

- personāla iepriekšēja pieredze viedo tehnoloģiju īpašību implementācijā un/vai adaptācijā;
- pieejamās viedo tehnoloģiju iestrādnes, to kvalitāte;
- ar viedajām tehnoloģijām aprīkojamā produkta modularitāte;
- viedo tehnoloģiju implementācijai atvēlētais laika periods;
- pieejamā atgriezeniskā saite no lietotājiem un lietotāju tolerances līmenis.

Iepriekšēja pieredze vai iestrādnes viedo tehnoloģiju īpašību implementācijā būtiski paaugstina projekta izdošanās varbūtību un samazina projekta izstrādei nepieciešamo

(prognozēto) darbietilpību. Kā katrā izpētes tipa projektā, pirmā viedo tehnoloģiju implementēšanas reize neapšaubāmi ir saistīta ar kļūdām, nepieciešamiem pārveidojumiem, labojumiem u.t.t., kas pagarina projekta izpildes laiku un palielina darbietilpību.

SIA „Datorikas institūts" pieredze, kas izklāstīta iepriekšējā darba nodaļā, liecina par ciešu korelāciju starp iepriekšēju pieredzi līdzīga tipa izstrādnēs un projekta izdošanās varbūtību.

Svarīgs faktors, novērtējot nepieciešamās sākotnējās investīcijas, ir ar viedajām tehnoloģijām aprīkojamā programmprodukta piemērotība šiem uzlabojumiem. Tā kā viedās tehnoloģijas parasti tiek iebūvētas attiecīgajā programmatūrā, lielu lomu spēlē aprīkojamā produkta uzbūve, tai skaitā, modularitāte. Jo modulārāk ir veidots pats produkts, jo vienkāršāk ir tam pievienot (tajā iebūvēt) „tikai" vienu papildus īpašību. Ja viedo tehnoloģiju implementācija ir saistīta ar fundamentālām izmaiņām visā produktā, neveiksmes iespējamība ir augstāka un sākotnējie ieguldījumi ir jāprognozē lielāki.

Lai arī darbietilpības novērtēšanas metodes allaž nosaka arī optimālo projekta izpildes laiku kalendārā nozīmē, viedo tehnoloģiju implementēšanai parasti nav iespējams veltīt pietiekami daudz laika. Iemesls ir klienti, kuri konkrēto īpašību tieši nav pasūtījuši un līdz ar to nevar izprast šādu „iekšēju izstrādņu" nozīmi.

Iepriekšējā nodaļā izklāstītā pieredze projektā C rāda, ka nenoslīpēta risinājuma piegāde ne vien neuzlabo, bet pat apgrūtina produkta izplatīšanu un pasliktina lietotāju attieksmi pret produktu.

Ir svarīgi jau pašā sākumā apzināties, vai konkrētie lietotāji ir gatavi (spējīgi) pieciest īslaicīgas neērtības, ja tādas rastos, un aktīvi piedalīties produkta uzlabošanas procesā.

Ne visi klienti ir gatavi tikt iesaistītiem „eksperimentos", tādēļ pateicīgāki ir projekti, kur klients ir jau ilggadīgs un izveidojis zināmu lojalitāti pret konkrēto piegādātāju. Ja klients ir jauns vai tā tolerances līmenis pret jauninājumiem ir zems, ieviešanas procesā jārēķinās ar papildus kvalitātes nodrošināšanas pasākumiem, jābūt labi nostādītam krīžu menedžmentam, un tas nozīmē papildus izdevumus.

4.2.2 Sagaidāmās naudas plūsmas novērtēšana

Nākotnē sagaidāmās naudas plūsmas novērtēšana vienmēr ir nedrošs un uz pieņēmumiem balstīts pasākums, tomēr tā ir neizbēgama katra biznesa plāna sastāvdaļa.

Atšķirībā no „klasiskā gadījumā", kad investīcijas tiek veiktas, plānojot tiešu izveidotā produkta pārdošanu, kad atdeve ir izmērāma konkrētos ieņēmumos, viedo tehnoloģiju atdeves novērtēšana ir daudz sarežģītāka.

Protams, teorētiski ir iespējams mazvarbūtīgs gadījums, kad viedo tehnoloģiju izstrādi pasūta konkrēts pasūtītājs un par viedo tehnoloģiju izstrādi kontrakta ietvaros arī maksā. Tomēr parasti viedās tehnoloģijas produktiem tiek pievienotas pēc izstrādātāju / uzturētāju pašu iniciatīvas. Tādēļ pieredze rāda, ka viedo tehnoloģiju atdeve ir jāiekalkulē produkta cenā (ja tas ir iespējams) vai jāsadala solidāri starp iesaistīto projektu administratīvajiem izdevumiem.

Lai novērtētu no viedo tehnoloģiju ieviešanas sagaidāmās naudas plūsmas, ir iespējams lietot divu veidu pieejas:

- augšupejošā jeb izdevumu bāzētā (*bottom-up*);
- lejupejošā jeb noieta bāzētā (*top-down*).

Izdevumu bāzētā pieeja nosaka to, ka tiek novērtētas izmaksas, kas viedo tehnoloģiju izmantošanas gadījumā tiek ietaupītas („atkrīt"). Tipiskākie šādu izmaksu piemēri ir transports un personāls, kas līdz šim bija nepieciešami programmatūras uzstādīšanai, instalēšanas materiāli un dokumentācija, kas līdz šim tika izgatavota, daļa lietotāju atbalsta kapacitātes, daļēji - neatkarīgās testēšanas resursi u.tml.

Noieta bāzētā pieeja nosaka, ka tiek vērtēts, par kādu cenu un kādā apjomā tirgū salīdzinājumā ar līdzšinējo cenu būtiski uzlaboto produktu būtu iespējams pārdot. Sagaidāmā cenu diference liecina par viedo tehnoloģiju „pievienoto vērtību", izteiktu reālos noieta rādītājos.

Bieži praksē tiek lietota abu nosaukto pieeju kombinācija, kur abas veido savdabīgu sagaidāmo naudas plūsmas intervālu, kurā investīciju atdevei būtu jāiekļaujas.

4.2.3 ROI aprēķināšana

Lai salīdzinātu sākotnējās investīcijas, kas tiks veiktas laika periodā t_0, ar naudas plūsmām, kas atgriezīsies laika momentos t_1 ... t_n, ir nepieciešama nākotnes naudas plūsmu vērtības aprēķināšana uz laika brīdi t_0. To parasti veic, katra perioda naudas plūsmām aprēķinot to diskontēto tagadnes vērtību (*discounted present value*).

Diskontētā tagadnes vērtība noteiktai naudas plūsmai vienā nākotnes periodā ir izsakāma kā:

$$DPV = \frac{FV}{(1+i)^n} = FV(1-d)^n, \quad (1)$$

kur

- DPV ir nākotnes naudas plūsmas FV (vai FV, kas pieskaņota naudas saņemšanas aizkavējumam) diskontētā tagadnes vērtība;
- FV ir nākotnes periodā saņemamā naudas daudzuma nominālā vērtība;
- i ir procentu likme, kas atspoguļo kapitāla izmantošanas izmaksas, tajā var tikt iekļauta arī maksājuma nesaņemšanas pilnā apmērā risks;
- d ir diskonta likme, ko aprēķina kā i/(1+i), resp., tā ir procentu likme, kas pārrēķināta uz gada sākumu, nevis uz gada beigām;
- n ir laika periodu (gadu) skaits līdz nākotnes naudas plūsmu saņemšanas brīdim. [61]

Viedo tehnoloģiju iekļaušana produktā no ekonomiskā viedokļa ir izdevīga tad, ja ROI ir lielāks par 1, resp., ja sākotnējās investīcijas ir mazākas par nākotnes naudas plūsmu diskontēto tagadnes vērtību.

Taču nedrīkst aizmirst, ka salīdzināti tiek lielumi, kuri iegūti novērtēšanas ceļā. Kaut vai darbietilpības novērtēšanas metodes vien var būt ļoti neprecīzas [62]. Pieņemot lēmumu par vai pret viedo tehnoloģiju izmantošanu, ir jāņem vērā arī turpmākajās apakšnodaļās diskutētie citi kritēriji.

ROI aprēķini iepriekšējā nodaļā aprakstīto projektu gadījumā rāda, ka Projekti A un B no ekonomiskā viedokļa ir bijuši efektīvi, projekts C ir bijis neefektīvs, bet projekta D efektivitātes novērtēšanu pagaidām ir grūti veikt nepietiekamā datu materiāla dēļ.

4.3 Kvalitātes kritēriji

Kā jau iepriekš minēts, katrs racionāls tirgus dalībnieks ir ieinteresēts tirgū darboties tikai tad, ja sagaida par to atlīdzību, resp., ja pārskatāmā laika periodā ar pietiekami augstu varbūtību ir sagaidāma peļņa. Lai arī ilgtermiņā, protams, noteicošais vienmēr ir ekonomiskais aspekts, tomēr peļņas ģenerēšanai savu pienesumu var dot arī tādi, finanšu rādītājos grūti ietērpjami aspekti kā uzņēmuma reputācija, piedāvājuma portfelis u.tml.

Viedās tehnoloģijas ir tipisks gadījums, kad sākotnējās investīcijas viena konkrēta klienta un/ vai produkta kontekstā var nenest finansiālu atdevi, taču ieguldījumi ilgtermiņā atmaksājas pastarpināti – uzlabojot uzņēmuma imidžu, sagatavojot klientus cita līmeņa risinājumiem u.t.t.

Produkta un pakalpojuma kvalitāte ir cieši saistīta ar uzņēmuma reputāciju, tādēļ ilgtermiņā tas ir īpaši svarīgs jautājums. Un, lai arī galu galā piedāvājuma kvalitāte un reputācija atspoguļojas uzņēmuma darbības finanšu rādītājos, sagaidāmo kvalitatīvo atdevi no viedo tehnoloģiju implementēšanas ir grūti izmērīt. Pārsvarā kvalitātes kritēriju novērtēšana balstās uz pagātnes pieredzes vērtībām, kas uzkrāta līdzīgos projektos.

Autores piedāvātās kvalitātes novērtēšanas kritēriju grupas ir šādas:

- programmprodukta kvalitāte;
- programmatūras uzturēšanas kvalitāte;
- klientu apkalpošanas kvalitāte.

Kvalitātes kritēriji ir tie, kas varētu pamudināt izstrādātājus pievienot viedo tehnoloģiju iespējas „stratēģiski svarīgiem" produktiem, pat, ja tuvākajā vai vidēja termiņa laika periodā nav sagaidāma šo investīciju atdeve no ekonomiskā viedokļa. Izstrādātājs tādējādi varētu celt savu tēlu klientu acīs un iegūt labumu citā veidā – iegūstot peļņu nesošus jaunus produktus, jaunu tirgus daļu u.tml.

4.3.1 Programmprodukta kvalitāte

Programmprodukta kvalitāte parasti tiek mērīta kvantitatīvi, izmantojot testēšanas fāzē un uzturēšanas pasākumu ietvaros apkopotos datus. Visiecienītākais rādītājs ir konstatēto kļūdu un neprecizitāšu skaits noteiktā laika periodā, papildus analizēts tiek atklāto kļūdu smagums, kā arī kļūdu skaits tiek attiecināts pret programmprodukta pirmkoda rindiņu vai

metožu/ funkciju skaitu, tādējādi iegūstot relatīvus rādītājus, kuri izmantojami dažādu produktu kvalitātes salīdzināšanai. Autores piedāvātais rādītājs tiek aprēķināts, kļūdu skaitu attiecinot pret produkta izstrādē ieguldīto resursu – darbietilpību personmēnešos - , jo šīs dalītājs visprecīzāk raksturo produkta "izmēru".

Ja ar viedajām tehnoloģijām tiek papildināta jau ekspluatācijā esoša programmatūra, tad viedo tehnoloģiju pienesuma novērtēšanai būtu saprātīgi izmantot tieši tos pašus parametrus, par kuriem dati ir jau uzkrāti no tā perioda, kad sistēma vēl nebija aprīkota ar viedajām tehnoloģijām.

Savukārt, ja lēmums par viedo tehnoloģiju iekļaušanu informācijas sistēmā tiek pieņemts vēl sistēmas prasību specificēšanas vai projektēšanas fāzē, tad salīdzinājums nav iespējams, jo būtu neracionāli veidot laboratorijas eksperimentus, resp., vispirms izveidot programmatūru bez viedās tehnoloģijas, tad veikt mērījumus un tikai pēc tam aprīkot programmatūru ar viedajām tehnoloģijām, lai atkārtoti veiktu mērījumus un izdarītu secinājumus par viedo tehnoloģiju iekļaušanas efektivitāti vai neefektivitāti. Šādos gadījumos mērījumus var salīdzināt ar radniecīgu produktu, kas nav aprīkoti ar viedajām tehnoloģijām, rādītājiem.

Autores piedāvātās programmatūras kvalitātes mērīšanas metrikas ir:

- produkcijā esošā produktā atklāto kļūdu skaits laika periodā (ja iespējams - sadalījumā pa kļūdu smagumiem un cēloņu tipiem) pret produkta izstrādē ieguldīto darbietilpību,
- kļūdu cēloņu noskaidrošanai vidēji patērētais laiks,
- kļūdu novēršanai vidēji patērētais laiks,
- atkārtoti konstatēto kļūdu skaits pret kopējo kļūdu skaitu periodā.

Minētie rādītāji tika aprobēti iepriekšējā nodaļā aprakstītajos projektos, analizējot uzņēmumā uzkrāto informāciju par projektu problēmziņojumu apstrādi un darba laika uzskaites sistēmas datus. Lai arī analīzei nepieciešamā informācija visos uzņēmuma projektos saskaņā ar vadības rīkojumu tiek uzkrāta kopš 2002. gada, tomēr atsevišķi rādītāji nebija pieejami nepieciešamajā kvalitātē (piemēram, problēmziņojumu klasifikācija un detalizācijas pakāpe neatbilda analīzes vajadzībām). Šādos gadījumos attiecīgo projektu vadītāji

interpretēja pieejamos datus, nodrošinot apkopoti uzkrāto lielumu detalizēšanu sīkāk un, vajadzības gadījumā, trūkstošo datu iegūšanu no pieredzes vērtībām (novērtēšanu).

Projektā A bija pieejams plašs datu materiāls, jo dinamiska darījummodeļa bāzētais risinājums kopš 2004. gada aktīvi tiek izplatīts publiskā sektora klientiem, un viens no pielietojumiem ir tipveida, resp., vienotu risinājuma tehnisko kodolu ar atšķirīgi konfigurētiem darījumprocesiem izmanto seši publiskā sektora klienti – valsts iestādes. Izvērtējot datus, tika uzskatīts, ka visi seši klienti pieteikumus veikuši par vienu kopīgu produktu, lai gan, protams, juridiski katrs klients problēmziņojumus pieteica esošā izstrādes un/vai uzturēšanas līguma ietvaros.

Projekts B, lai arī no visiem aprakstītājiem ir ar visgarāko viedo tehnoloģiju pielietošanas pieredzi, varēja piedāvāt krietni pieticīgāka apjoma datu materiālu. Kaut arī kopš speciāla klientu atbalsta speciālista iekļaušanas projekta darba grupā situācija ir strauji mainījusies, jo īpaši pirmajos produkta izstrādes gados un tiem sekojošajā straujās tirgus iekarošanas laikā datu vākšana netika veikta ar nepieciešamo rūpību un precizitāti. Tādēļ rādītāju vērtību noteikšanā nācās lielā mērā balstīties uz projekta vadītāja novērtējumiem.

Projekts C, salīdzinot ar iepriekšējiem diviem, bija īsa termiņa un salīdzinoši nesen pabeigts, tādēļ par to bez grūtībām bija iespējams atrast nepieciešamos datus.

Savukārt projekts D pagaidām ir pētnieciskā stadijā, tādēļ tajā uzkrātie dati vēl nav izmantojami secinājumu izdarīšanai. Projektā D gūtās atziņas turpmākajā izklāstā nav izmantotas.

Atklāto kļūdu skaits

Projektā A saņemto (gan no ārējiem klientiem, gan iekšēji kvalitātes nodrošināšanas pasākumu ietvaros radušos) un apstrādāto problēmziņojumu skaits bija vidēji 820 problēmziņojumi gadā (dati no 2005. gada līdz 2009. gadam). Sadalot pēc pieteikumu tipa un to smagumiem, redzams, ka vairāk kā puse (!) no pieteikumiem jeb vidēji 56% saistīta ar dinamiskā darījummodeļa aspektiem – gan lietotāju pieteiktas problēmas ar darbu plūsmām, gan konsultācijas par korektu lietošanu, izmaiņu pieprasījumi darba plūsmu konfigurācijai, arī iekšējas testēšanas saskaņā ar darbu plūsmu bāzētiem scenārijiem rezultātā tapuši pieteikumi. Ja risinājums būtu būvēts pēc "klasiskās" pieejas, kad katram darījumprocesam tiek veidotas speciālas ekrāna formas un datu struktūras, lielākā daļa no šiem 56% pieteikumu būtu saistīti ar programmēšanas darbiem – specificēšana, kodēšana, testēšana, jaunas versijas

izveide un piegāde. Pārējie 44% problēmziņojumu tiešām bija saistīti ar programmatūras risinājumu – tās ir kļūdas, izmaiņu pieprasījumi, dublējošies pieteikumi u.tml. Par kļūdām (vieglākām vai smagākām) līdz ar to būtu dēvējami apmēram 25% pieteikumu jeb vidēji 200 kļūdas gadā.

Projektā A kodola programmatūras izstrādē ir ieguldīti apmēram 50 personmēneši. Līdz ar to aprēķināmais atklāto kļūdu koeficients pret darbietilpību ir gadā vidēji 4 atklātas kļūdas uz katru izstrādē ieguldīto personmēnesi.

Projektā B saņemto (galvenokārt no ārējiem klientiem) un apstrādāto problēmziņojumu skaits pārsniedza 2000 problēmziņojumus gadā (dati no 2004. gada līdz 2009. gadam). Sadalot pēc pieteikumu tipa un to smagumiem, redzams, ka "lauvas tiesa" jeb apmēram 70% pieteikumu ir lietotāju konsultācijas par risinājuma lietošanu, kas telefoniski vai elektroniski sniegtas daudzajiem klientiem Latvijas reģionos.

Pārējie 30% pieteikumi bija saistīti ar programmēšanas darbiem – specificēšana, kodēšana, testēšana, jaunas versijas izveide un piegāde. Precīzu datu trūkuma dēļ uzskatīsim, ka visi 30% no kopējā problēmziņojumu skaita jeb vidēji 600 gadā klasificējami kā kļūdas.

Jo īpaši versiju piegādē viedo tehnoloģiju pielietošana devusi būtisku pienesumu – saspringtākajās produkta attīstības fāzēs produktam tika izlaistas pat līdz 4 jauninājumu versijām nedēļā (!). Pie milzīgā klientu skaita jauninājumu izplatīšana ar šādu intensitāti nebūtu bijusi iespējama vispār.

Projektā B pamata programmatūras izstrādē ir ieguldīti apmēram 80 personmēneši. Līdz ar to aprēķināmais atklāto kļūdu koeficients pret darbietilpību ir gadā vidēji 7,5 atklātas kļūdas uz katru izstrādē ieguldīto personmēnesi.

Projektā C saņemto (galvenokārt no iekšējiem testētājiem) un apstrādāto problēmziņojumu skaits bija vidēji 78 problēmziņojumi gadā (dati no 2008. gada līdz 2009. gadam). Gandrīz visi pieteikumi bija attiecināmi tieši uz programmatūras darbību, jo lietotāju pieteikumi (pārsvarā konsultāciju pieprasījumi un izmaiņu pieprasījumi) tika reģistrēti atsevišķā reģistrā, kurš nav pilnīgs.

Projektā C programmatūras izstrādē ir ieguldīti 8 personmēneši. Līdz ar to aprēķināmais atklāto kļūdu koeficients pret darbietilpību ir gadā vidēji 9,75 atklātas kļūdas uz katru izstrādē ieguldīto personmēnesi.

Kļūdu cēloņu noskaidrošanai vidēji patērētais laiks

Analizējot projektos uzkrātos darba laika atskaites datus, rodas grūtības atdalīt kļūdu cēloņu noskaidrošanai un kļūdu novēršanai patērēto laiku, jo šādā sadalījumā dati nav tikuši uzkrāti. Toties salīdzinoši viegli ir iespējams noskaidrot vidējo laiku, kas patērēts no problēmpieteikuma reģistrācijas brīža līdz problēmpieteikuma "aizvēršanai", resp., iekšējās kvalitātes kontroles sekmīgas pabeigšanas un problēmpieteikuma atzīmēšanai par pabeigtu. Lai novērstu brīvdienu radītos datu kropļojumus, vidējie laiki tikai aprēķināti tikai darba laika ietvaros, resp., piektdiena pēcpusdienā ienācis problēmpieteikums attiecīgajā darba nedēļā tika risināts līdz darba nedēļas beigām, lai darbu atsāktu pirmdien darba laikā.

Projektā A vidēji no problēmpieteikuma reģistrācijas līdz tā slēgšanai tika patērētas 9 darba stundas, kas pirmajā acu uzmetienā liekas liels skaitlis. Projekta vadītāja šo skaitli skaidro ar 2 faktoriem – 1) pilns problēmpieteikuma apstrādes cikls ietver apjomīgu iekšējās loģikas pārtestēšanu (saskaņā ar attiecīgo dartba plūsmu aprakstiem), 2) problēmpieteikumu apstrāde un "aizvēršana" bieži vien tiek pakārtota versiju piegādes plānam resp., ne visi problēmpieteikumi uzreiz arī tiek apstrādāti, bet daži apzināti tiek atlikti uz vēlāku laiku (tiek iekļauti citās versijās). Kļūdu cēloņu noskaidrošanai no kopējā problēmpieteikumu apstrādes laika tiekot veltīti apmēram 40%.

Projektā B vidēji no problēmpieteikuma reģistrācijas līdz tā slēgšanai tika patērētas 3 darba stundas, kas ir būtiski mazāk nekā projektā A. Tas izskaidrojams ar projekta B specifiku – 1) produkta izstrāde un ieviešana tika veikta ļoti saspringtā režīmā, kas izskaidro straujo reakciju uz pieteikumiem, kā arī netika veikti tik apjomīgi kvalitātes nodrošināšanas pasākumi, kādi tiek veikti "parastos" projektos, 2) tā kā produktam bija ļoti daudz vienlaicīgo lietotāju, kļūdu gadījumā identisku kļūdu pieteikumu skaits bija ļoti liels, kas ietekmēja vidējo rādītāju. Kļūdu cēloņu noskaidrošanai, saskaņā ar projekta vadītāja vērtējumu, no kopējā problēmpieteikumu apstrādes laika tikuši veltīti apmēram 25%.

Projektā C vidēji no problēmpieteikuma reģistrācijas līdz tā slēgšanai tika patērētas 16 darba stundas, kas ir liels rādītājs. Projektā C tas bija pieļaujams, jo gandrīz nebija lietotāju pieteiktu kļūdu – lielākais vairums problēmpieteikumu bija iekšējas izcelsmes. Termiņu ziņā kritiskā projektā šādi apstrādes laiki, protams, nebūtu akceptējami. Kļūdu cēloņu noskaidrošanai projektā no kopējā problēmpieteikumu apstrādes laika vidēji tikuši veltīti apmēram 50%.

<u>Kļūdu novēršanai vidēji patērētais laiks</u>

Kļūdu novēršanai un ar to saistītajai kvalitātes kontrolei patērētais laiks procentuāli no kopējā laika, kas vidēji patērēts problēmpieteikumu apstrādei no to reģistrācijas līdz pabeigšanas brīdim, ir viegli aprēķināmi kā starpība starp kopējo vidējo problēmziņojumu apstrādes laiku un celoņu noskaidrošanai patērēto laiku. Līdz ar piedāvātais mērījums pēc šādas aprēķinu metodikas būtu neko neizsakošs. Tādēļ autore attiecīgo projektu vadītājus lūdza procentuāli novērtēt vidējo tieši izstrādei (kļūdas novēršanai un vienībtestēšanai, neieskaitot administratīvo un kvalitātes procesu izraisīto laika patēriņu) patērēto laiku.

Projektā A kļūdu novēršanai tieši tiek patērēti vidēji 15% no no kopējā problēmpieteikumu apstrādes laika. Šis rādītājs liecina, ka apmēram 45% no problēmpieteikumu apstrādei patērētā laika tiek tērēti kvalitātes nodrošināšanas, jaunu versiju izveides un izplatīšanas jautājumiem. Izmantojot viedo tehnoloģiju komponenšu – paštestēšanas un versiju izplatīšanas - sniegtās iespējas, projektā A būtu iespējams panākt labāku vidējo problēmpieteikumu apstrādes rādītāju.

Projektā B kļūdu novēršanai tieši tiek patērēti vidēji 60% no kopējā problēmpieteikumu apstrādes laika. Tas nozīmē, ka projektā B salīdzinoši ātri tiek noskaidroti kļūdu cēloņi, taču kļūdu novēršanas tiešajām darbībām vidēji ir lielāks īpatsvars nekā atbalstošajiem procesiem. Tas daļēji ir izskaidrojams ar projekta B straujo ieviešanu un lielo lietotāju skaitu, taču viens no iemesliem ir lielais pirmkoda apjoms un viena izstrādātāja kā galvenā zināšanu īpašnieka dominējošā loma projektā.

Projektā C kļūdu novēršanai tieši tiek patērēti vidēji 25% no kopējā problēmpieteikumu apstrādes laika, kas vērtējams kā optimāls lielums. Tajā pašā laikā no ekonomiskā un darbietilpības patēriņa viedokļa projekts C drīzāk vērtējams kā nesekmīgs viedo tehnoloģiju izveides projekts, kas parāda, ka kļūdu apstrādes ātrums un sadalījums aktivitāšu veidos nebūt automātiski neliecina par projekta komerciālo veiksmi.

Atkārtoti konstatēto kļūdu skaits pret kopējo kļūdu skaitu

Šis parametrs autorei rādītāju izvēles brīdī šķita visdaudzsološākais, jo varētu pierādīt vai apgāzt pieņēmumu, ka viedo tehnoloģiju (jo īpaši automatizētas paštestēšanas, par kuru gan pagaidām nav uzkrāts pietiekami apjomīgs datu materiāls) izmantošana samazina atkārtotu kļūdu parādīšanās iespējamību un biežumu, kā arī paātrina ielāpu izplatīšanas ātrumu.

Diemžēl analizējot pieejamos problēmpieteikumu datus, nācās vilties, jo no ziņojumu aprakstiem bija gandrīz neiespējami noteikt, kuri no problēmpieteikumiem saturiski attiecas uz jau iepriekš novērstu problēmu un kuri ir pēc būtības jauni. Nācās vērsties pie attiecīgo projektu vadītājiem ar lūgumu dalīties pieredzes vērtībās.

Projekta A vadītāja atzina, ka atkārtoto kļūdu skaits no kopējā kļūdu skaita ir salīdzinoši liels un sasniedz pat 10%. Daļēji tas izskaidrojams ar to, ka dažādi klienti vienu un to pašu kļūdu atklāj un piesaka dažādos laika periodos, daļēji tomēr ar to, ka izstrādātāji nepietiekami nopietni izturas pret testēšanu. Projekta vadītāja uzskata, ka rutīnas pārtestēšanu noteikti varētu uztucēt paštestēšanas komponentei, kā arī saredz ārējās vides analīzes komponentes izmantošanas potenciālās priekšrocības.

Projekta B vadītājs apgalvoja, ka līdz pat 75% no kopējā kļūdu skaita pēc savas būtības ir dublikāti, un tas ir saistīts ar milzīgo vienlaicīgo lietotāju skaitu, kas, paši būdami zem liela termiņu spiediena, nekavējoties piesaka katru sīkāko neprecizitāti, kas kavē viņu tiešo darbu izpildi. Projekta vadītājs to uztver par sava projekta specifiku. Viedo tehnoloģiju pienesums tiek sagaidīts no darījumbiznesa procesu bāzētās funkcionalitātes iespējām, kas pašlaik tiek iestrādātas produkta jaunākajā versijā.

Projekta C ietvaros atkārtotu kļūdu pieteikumu praktiski nebija. Tas ir vienkārši izskaidrojams ar to, ka problēmpieteikumi pamatā tika reģistrēti iekšēju pasākumu rezultātā, resp., testētāji un izstrādātāji zināja, kuras problēmas jau ir bijušas konstatētas un pirms katras jaunas versijas izgatavošanas varēja atļauties veltīt speciālu uzmanību iepriekš konstatēto un novērsto problēmpieteikumu atkārtotai pārbaudei.

4.3.2 Programmatūras uzturēšanas kvalitāte

No izstrādātāju viedokļa būtisks arguments par labu viedo tehnoloģiju izmantošanai ir programmatūras uzturēšanas aspekti. Jo īpaši ilgtermiņa kontraktu gadījumā, kad izstrādātājs līguma ietvaros nodrošina augsta līmeņa uzturēšanas servisu, ir vitāla interese programmatūru uzlabot ar iespējām, kas uzturēšanas procesu atvieglo.

Uzturēšanas kvalitāti autore piedāvā novērtēt ar uzturēšanai specifiskiem rādītājiem:

- kļūdu īpatsvars no kopējā lietotāju pieteikumu skaita periodā,
- jaunu versiju testēšanai patērētais laiks,
- jaunu versiju izplatīšanai (uzstādīšanai) patērētais laiks.

Produkta uzturēšanas periods no izstrādes perioda parasti atšķiras ar mazāku kodēšanas darbu intensitāti, toties aktualizējas izmaiņu pieprasījumu nepieciešamība un jānodrošina lietotāju atbalsts (līmenī, kādā tas paredzēts kontraktā). "Klasiskus" uzturēšanas projektus bieži raksturo samazināta darba grupa vai pat speciālas uzturēšanas struktūrvienības piesaistīšana, lai izstrādātājus būtu iespējams piesaistīt citu projektu darbiem. Taču bieži vien iepriekšējo produkta versiju uzturēšanas darbi tiek veikti paralēli tā paša produkta jaunas versijas izstrādei, un tikai reti projektiem ir piesaistīts speciāls klientu atbalsta speciālists.

Autores ideja, ieviešot programmatūras uzturēšanas kvalitāti raksturojošus rādītājus, bija ne vien pārliecināties par izvēlēto rādītāju vērtībām reālos projektos, bet arī pārbaudīt pieņēmumu, ka viedo tehnoloģiju pielietošana produktu izstrādē atvieglo arī šo produktu uzturēšanu.

Kļūdu īpatsvars

Gan projektā A, gan projektā B izstrādes darbi joprojām turpinās (kas parasti ir sekmīgu produktu pazīme!), līdz ar to ir netriviāli nodalīt uz uzturēšanu attiecināmos datus no uz izstrādi attiecināmiem datiem. Projektā A uzturēšana no izstrādes tika atdalīta ar pieņēmuma palīdzību – tika uzskatīts, ka pie katra klienta izstrādes darbi ir notikuši vienu gadu, bet pēc tam projekts pie attiecīgā klienta automātiski pārgājis uzturēšanas fāzē. Projektā B tas tika panākts ar versiju pārvaldības palīdzību – tika uzskatīts, ka visi problēmpieteikumi, kas attiecas uz 1. līdz 5. produkta versiju, uzskatāmi par uzturēšanai piekrītošiem. Projektā C izstrādes darbi ir paveikti, līdz ar to uz uzturēšanu ir attiecināmi visi saņemtie pieteikumi, taču pagaidām uzturēšanas periods (kopš 2009. gada nogales) ir pārāk īss, lai no uzkrātajiem datiem izdarītu nopietnus secinājumus.

Tas, ka kļūdu īpatsvars pret kopējo saņemto pieteikumu skaitu, kā arī absolūtais saņemto pieteikumu skaits ar katru gadu samazinās, atbilst katra izstrādātāja novērojumiem. Taču reizē ar pieteikto kļūdu skaitu krītas arī nepieciešamo atbalsta aktivitāšu skaits: tā kā lietotāji darbam ar produktu ir apmācīti, darījumprocesi ir nostabilizējušies, nepieciešamie izmaiņu pieprasījumi identificēti un, ja nav realizēti, tad vismaz atrasts cits ceļš, kā nepieciešamo funkcionalitāti iegūt, arī ar programmatūras kļūdām nesaistīto pieteikumu skaits ar gadiem krītas.

Projektā A 2006. gadā kļūdas sastādīja apmēram 20% no uz uzturēšanas procesiem attiecināmu pieteikumu, un 2009. gadā pie mazāka absolūtā pieteikumu skaita proporcija starp

kļūdu izraisītiem pieteikumiem un citu iemeslu radītiem pieteikumiem joprojām bija apmēram tāda pati. Viens no iemesliem, kāpēc strauji samazinājās absolūtais problēmpieteikumu skaits, bija kvalificēta pasūtītāja pārstāvja esamība klienta informācijas tehnoloģiju daļā, kas spēja lielu daļu konsultācijas darba uzņemties uz sevi, kā arī relatīvi stabils produkta lietotāju sastāvs. Savukārt nemainīgais kļūdu īpatsvars kopējā pieteikumu kopā izskaidrojams ar likumdošanas radītu izmaiņu ieviešanu produktā, kas tika veikta uzturēšanas darbu ietvaros.

Projektā B kļūdu īpatsvars kopējā pieteikumu skaitā pa uzturēšanas gadiem ir stipri mainījies – no sākotnēji gandrīz 30% 2005. gadā līdz mazāk kā 10% 2009. gada nogalē. Tas daļēji izskaidrojams ar versiju nostabilizēšanos - produkta 2., 3. un 5. versijas nebija "revolucionāras", resp., neietvēra būtiskas arhitektūras izmaiņas. Savukārt lietotāju sastāvs šajos gados kadru rotācijas dēļ ir būtiski mainījies, tādēļ lielu daļu pieteikumu sastāda konsultācijas par produkta izmantošanu, kā arī ar likumdošanas radītu izmaiņu un jauninājumu apguvi.

Jaunu versiju testēšanai patērētais laiks

Šis uzturēšanas kvalitātes rādītājs pamatā paredzēts paštestēšanas efektivitātes pētījumiem. Autores pieņēmums ir – parādīt, ka jaunas versijas notestēšanai patērētais laiks var tikt būtiski ietaupīts, ja tiek izmantoti paštestēšanas mehānismi. Tādēļ pašlaik aktīvi tiek uzkrāti dati projektā D – pēc paštestēšanas prototipa aprobācijas reālā projektā būs iespējams salīdzināt šajā projektā uzkrātos datus par versiju testēšanai patērēto laiku ar datiem, kas tiks iegūti paštestēšanas izmantošanas laikā.

Tomēr arī projektos, kuros jau uzkrāti dati, varētu būt interesanti noskaidrot – cik ilgs laiks paiet no piegādājamas versijas izveides līdz lēmuma pieņemšanai, ka versija ir piegādājam lietotājiem.

Projektā, kur ir salīdzinoši maz instalāciju punktu (pret, piemēram, projektu B), versiju testēšanai tiek veltīts salīdzinoši mazāks laiks – ja arī piegādātā versija saturēs kļūdas, nepieciešamo pārinstalāciju daudzums būs pārskatāms. Vidēji integrācijas testiem, tai skaitā kritiskās funkcionalitātes pārbaudei uz katru versiju tiek patērēta 1-2 persondienas. Tas nav daudz, taču ierobežotu resursu apstākļos projektu vadītāja neredz iespējas ieguldīt lielākus laika resursus. Problēmpieteikumu skaita pieaugums tūlīt pēc jaunas versijas piegādes liecina par to, ka pilna testēšana tomēr parasti nav veikta un lietotāji pilda savdabīgu "ārējo testētāju" funkciju. Projekta vadītāja atzīst, ka kritiskās funkcionalitātes automatizēta pārtestēšana

varētu risināt pieteikumu skaita palielināšanos dienās pēc jaunas versijas piegādes, taču pagaidām neredz finanšu iespējas šādu īpašību produktā ieviest (skatīt arī kvalitatīvā pētījuma atziņas nākamajā darba nodaļā).

Jaunu versiju izplatīšanai patērētais laiks

Ar jaunu versiju izplatīšanu minētā rādītāja noteikšanai tiek saprasts laiks, kas tiek tērēts no lēmuma par jaunas versijas piegādi līdz brīdim, kad uz visu adresātu (lietotāju) darbstacijām ir darbspējīga produkta versija.

Projektā A versiju izplatīšanai tiek patērētas vidēji 2 stundas – piegāde notiek attālināti, izmantojot klienta pusē nodrošinātās pieslēgšanās iespējas, taču manuāli. Tas paaugstina kļūdīšanās iespējamību, un projekta darbinieki atzīst – ja klients nepieļautu izstrādātāju darbības uz sava servera, piegādes laiki varētu palielināties par vismaz 4 reizēm.

Projektā B, kur versijas ar viedo tehnoloģiju komponenti tiek izplatītas jau kopš 2006. gada, par jaunām tiek sauktas versijas, kas radušās būtisku programmatūras uzlabojumu rezultātā. Līdz ar to jaunas versijas tiek izplatītas salīdzinoši reti (apmēram 3-4 reizes gadā), taču aktīvi tiek lietots ielāpu un uzlabojumu izplatīšanas mehānisms (ar tā palīdzību tiek papildinātas komponentes, mainīta konfigurācija, piegādāti jauni objekti, veiktas datu struktūru transformācijas utt.) Pateicoties noslīpētajam versiju izplatīšanas mehānismam, visi jauninājumi uz klientu darbstacijām tiek nogādāti nepilnas minūtes laikā, turklāt lietotāji katrā sava darba brīdī var būt droši, ka automatizēti saņems jauninājumus, ja vien vai tiklīdz būs pieejams internets.

Šis rādītājs labi parāda projekta C problēmas – versiju izplatīšana, neskatoties uz viedo tehnoloģiju pielietošanu, bija smagnēja. Sākotnējā versiju uzstādīšana bija saistīta ar obligātu ierašanos pie klienta, un klienti bija izkliedēti pa visu Latvijas teritoriju. Arī iecerētais atjaunināto versiju izplatīšanas mehānisms, vismaz sākumā, bija pārāk jūtīgs pret ārējās vides specifiku un interneta pieslēguma neesamību, ka vidēju uz katru otro instalācijas punktu nācās ierasties atkārtoti. Līdz ar to vidēji versijas sākotnējai uzstādīšanai tika tērēta puse līdz vesela diena uz instalācijas punktu, vēlāk šis rādītājs nokritās līdz vidēji 2 stundām. 2010. gada sākumā, kad tika atjauninātas produkta versijas, situācija šķiet uzlabojusies – jauninājumus izdodas gandrīz momentāni piegādāt 9 no 10 gadījumiem un atlikušo vienu apmēram 80% izdodas atrisināt attālināti.

4.3.3 Klientu apkalpošanas kvalitāte

Klientu apkalpošanas kvalitāte nosaka servisa līmeni, ko produkta atbalsta speciālisti var nodrošināt bez būtiskiem papildus finanšu ieguldījumiem - nepalielinot apkalpojošo darbinieku skaitu, būtiski nepārveidojot apkalpošanas procesus u.tml.

Šī kritērija izpildes pakāpi var mērīt ar kvalitatīviem paņēmieniem – aptaujājot klientus, izlases kārtībā analizējot saņemtos lietotāju pieteikumus u.tml.

Klientu apkalpošanas efektivitāte raksturo atbalsta apjomu, ko ar konkrēto atbalsta personālu iespējams sasniegt. Tātad vai nu ar tiem pašiem atbalsta resursiem var apkalpot vairāk lietotāju, vai esošo lietotāju skaitu - labāk. Ja viedo tehnoloģiju ieviešanas rezultātā lietotāju skaitu neizdodas palielināt, tad par šī kritērija pozitīvu izpildi liecinātu tas, ka ar mazākiem atbalsta resursiem var apkalpot to pašu lietotāju skaitu, nepazeminot apkalpošanas kvalitāti.

Piedāvātie kvantitatīvie rādītāji (pie fiksēta apkalpojošā resursa) ir šādi:

- vidējais viena pieteikuma reģistrēšanas ilgums (ieskaitot kļūdas izpausmes izdibināšanu no lietotāja),
- pieteikumu skaits (procentos no kopējā), kuru izdodas atrisināt atbalsta personālam, nepiesaistot informācijas tehnoloģiju speciālistus.

No iepriekšējā nodaļā raksturotajiem projektiem tikai projektā B tiek nodarbināts pilnas slodzes klientu apkalpošanas darbinieks, kurš pēc izglītības nav informācijas tehnoloģiju speciālists. Pārējos projektos lietotāju apkalpošanu telefoniski vai elektroniski (projektā C – arī klātienē) veic attiecīgo produktu izstrādātāji. Līdz ar to augšminēto mērījumu veikšanai izmantojams ir tikai projekts B, kur turklāt ir nodrošināta ļoti precīza pieteikumu reģistrēšana un klientu atbalsta speciālists ir arī ieinteresēts pēc iespējas daudz pieteikumu atrisināt saviem spēkiem, neiesaistot izstrādātājus.

Pēc klientu atbalsta speciālista sniegtajiem datiem vidējais viena pieteikuma reģistrēšanas ilgums, ieskaitot kļūdas izpausmes izdibināšanu no lietotāja un, ja nepieciešams, ar viedajām tehnoloģijām aprīkotā produkta pierakstītās diagnosticējošās informācijas analīzi, ir pusstunda. Vidējo rādītāju "sabojā" specifiski gadījumi, kad problēmas/ defekta izpausme ir grūti konstatējama un no klienta nepieciešama papildus informācija. Savukārt pieteikumu skaits, kuru izdodas atrisināt atbalsta personālam, nepiesaistot informācijas tehnoloģiju speciālistus, ir augsts – tie ir vairāk kā 80% no lietotāju pieteikumiem.

4.4 Organizatoriskie kritēriji

Atšķirībā no iepriekš raksturotajiem kritērijiem, kas orientēti uz peļņas un reputācijas gūšanu, organizatoriskie kritēriji raksturo izstrādātāja organizatorisko un tehnisko gatavību veikt viedo tehnoloģiju komponenšu izstrādi un integrāciju gala produktos.

Viedo tehnoloģiju izstrāde nav iespējama bez kvalificēta sistēmas arhitekta un spēcīgiem izstrādātājiem. Viedo tehnoloģiju komponenšu būve prasa, no vienas puses, labas akadēmiskās un profesionālās zināšanas, no otras, precīzu sapratni par aprīkojamā produkta uzbūvi un funkcionalitāti.

Izstrādes procesā ir jāievēro disciplīna, jāprogrammē saskaņā ar vadlīnijām, nedrīkst lietot specifiskus paņēmienus, kas vēlākā uzturēšanā varētu radīt problēmas. Tie gan ir vispārīgi norādījumi, kuru izpilde nav raksturīga tikai viedo tehnoloģiju projektiem vien. Protams, izstrādes procesu raksturojošos kritērijus ir grūti kvantitatīvi mērīt, tādēļ parasti priekšroka tiek dota pieredzējušiem arhitektiem un izstrādātājiem, necenšoties kvantitatīvi novērtēt šo darbinieku efektivitāti.

Tā kā viedo tehnoloģiju implementācija ir saistīta ar būtiskām izmaiņām programmatūras uzbūvē, nav izslēgta iespējamo kļūdu varbūtība, tāpēc no organizatoriska viedokļa jābūt nodrošinātam labam krīzes menedžmentam neparedzētiem gadījumiem. Spēcīgs uzņēmuma menedžments ar stabilu pārliecināšanas potenciālu ir labākais risinājums šādos gadījumos.

Tā kā izveidotais gala risinājums pakļaujams skrupulozam testēšanas procesam, protams, izstrādātāja rīcībā jābūt pienācīgam tehniskam nodrošinājumam, tai skaitā virtuālajām mašīnām, jaudīgiem centrāliem serveriem u.tml.

Viedo tehnoloģiju pielietošanas mērķis ir ilgtermiņā maksimizēt sagaidāmos ieguvumus. Samazinātu izmaksu formā vai kvalitātes uzlabojumu dēļ – klientam ir jājūt, ka papildinājumi dod pievienoto vērtību.

Mārketinga uzdevums ir veicināt piesaisti uzlabotajam produktam, veicināt klienta pierašanu pie viedo tehnoloģiju īpašībām. Līdz ar to, izšķiroties par vai pret viedo tehnoloģiju pielietošanu, ir svarīgi apzināties pieejamo mārketinga spēku, kas spētu viedo tehnoloģiju izstrādnes „pārvērst naudā”.

Ja klients viedo tehnoloģiju iespējas nenovērtē vai arī nav orientēts uz ilgtermiņa sadarbību, visticamāk, viedo tehnoloģiju iekļaušana programmatūrā ir veltīgs resursu patēriņš.

4.5 Viedo tehnoloģiju pielietošanas ierobežojumi

Kā jau iepriekš minēts, viedo tehnoloģiju iekļaušana informācijas sistēmā ir saistīta ar papildus resursu patēriņu. Līdz ar to likumsakarīgs ir jautājums, kādos gadījumos un pie kādiem nosacījumiem papildus resursu piesaiste dod ekonomisko atdevi.

Saskaņā ar ekonomikā, jo īpaši kontraktu teorijā (*contract theory*), apskatītajām asimetriskās informācijas sadalījuma teorijām (*information asymmetry*) [63] darījumu partneri tirgū lēmumus pieņem, balstoties uz nepilnīgu informāciju. Tādējādi veidojas nevienlīdzīgs spēku sadalījums starp iesaistītajām pusēm un transakcijas tiek kropļotas. Veidojas interešu sadursme – labāk informētā puse cenšas gūt labumu no sev pieejamās informācijas, kamēr sliktāk informētā puse izmanto savas priekšrocības (pasūtītāja statuss, labāks finansējums u.tml.), lai ierobežotu otras puses vadošo lomu.

Viedo tehnoloģiju kontekstā interesants ir jautājums par ieinteresēto jeb labāk informēto pusi, kura varētu būt viedo tehnoloģiju ieviešanas dzinējspēks. Apskatīti tiks komerciāli projekti, resp., tālāko pārspriedumu pamatā ir tirgus ekonomika, kur katrs dalībnieks cenšas maksimizēt savu peļņu.

Vispārīgi raugoties, ierosme viedo tehnoloģiju izmantošanai varētu rasties divos dažādos veidos:

- pasūtītājs, apzinoties ar programmatūras uzturēšanu saistīto problemātiku, jau informācijas sistēmas iegādes brīdī vai tās uzturēšanas laikā apzināti pasūta viedo tehnoloģiju komponenšu iekļaušanu programmproduktā;
- piegādātājs, balstoties uz ilgtermiņa norunām vai cerot uz resursu ietaupījumu nākotnē, pats iekļauj viedo tehnoloģiju komponentes savos programmproduktos.

Pieņēmumu, ka viedo tehnoloģiju pielietošana varētu tikt vadīta no pasūtītāju puses, varētu pārbaudīt, analizējot konkrētā tirgū veiktos iepirkumus un to rezultātus. Tā kā programmproduktu iegādes nosacījumi pamatā ir komercnoslēpums, vienīgais (nosacīti) brīvi pieejamais informācijas avots ir publiskie iepirkumi. Latvijas Republikā publisko iepirkumu

norisi nosaka Publisko iepirkumu likums, kurš paredz iepirkuma datu pieejamību tirgus dalībniekiem, kā arī iepirkumu procedūru rezultātu paziņošanu procedūras dalībniekiem.

Autore, balstoties uz tirgū aktīva IT piegādātāja, kas specializējies uz publisko sektoru, SIA „Datorikas institūts DIVI" uzkrāto dažādu valsts iestāžu iepirkumu dokumentāciju pēdējos 5 gados, veica iepirkumu dokumentācijas analīzi ar mērķi konstatēt viedo tehnoloģijām atbilstošu prasību esamību iepirkuma dokumentācijā.

Tika konstatēts, ka pasūtītāji publisku iepirkumu prasībās visbiežāk iekļauj ar drošību un stabilitāti saistītas prasības – pieejamības pārraudzību, noslodzes pārraudzību un slepenības pārraudzību. Faktiski nekad netiek iekļautas tādas ar programmatūras uzturēšanu tieši saistītas īpašības kā vides testēšana vai paštestēšana. Atsevišķos gadījumos prasībās iekļauts dinamisks biznesa modelis, datu kvalitātes pārraudzība un versiju pārvaldības īpašības.

Šie novērojumi rāda, ka Latvijas publiskā sektora pasūtītājs pamatā neapzinās viedo tehnoloģiju priekšrocības un līdz ar to nepasūta atbilstošu īpašību iekļaušanu savās informācijas sistēmās. Un tas savukārt nozīmē, ka galvenais dzinējspēks viedo tehnoloģiju izmantošanā ir pats piegādātājs. Kā racionāls tirgus dalībnieks, piegādātājs ir ieinteresēts investēt tikai tad, ja pārskatāmā laika periodā ar augstu varbūtību var rēķināties ar investīciju atgūšanu un peļņu.

Praksē piegādātāji var piekopt divas stratēģijas:

- ar viedo tehnoloģiju iekļaušanu programmproduktā saistīto izmaksu iekļaušana prasītās funkcionalitātes darbietilpības novērtējumā (pasūtītājs par viedo tehnoloģijām maksā reizē ar pamata funkcionalitātes izstrādi);
- viedo tehnoloģiju izstrāde tiek veikta „uz pašu rēķina", investīcijas atgūstot no atbalstoši optimizētiem pakalpojumiem, kuru pārdotais apjoms pārsniedz to sniegšanai reāli patērēto laiku (pasūtītājs par viedajām tehnoloģijām maksā pastarpināti, saņemto pakalpojumu formā).

Pirmajā gadījumā pasūtītājs, visticamāk pašam to sākotnēji nemaz neapzinoties, par augstāku cenu nopērk labāku produktu nekā sākotnēji iecerējis. Šādā gadījumā investējis ir pasūtītājs, bet piegādātājs kā zināšanu nesējs ir ieguvis priekšrocības turpmākā produkta uzturēšanā. Punkta, pie kura investīcijas sāk atmaksāties (t.s. *break–even* punkta) meklēšana šādā gadījumā nav relevanta, jo piegādātāja darbs ir jau apmaksāts.

Savukārt, ja investīcijas veicis piegādātājs uz „paša iniciatīvas“ pamata (un publisko iepirkumu gadījumā tas tā ir bieži), svarīgi ir kritēriji, kas piegādātājam nodrošina investīciju atdevi.

Autore praksē ir identificējusi šādus, būtiskākos kritērijus, lai piegādātājs būtu ieinteresēts minētajās investīcijās:

- ilgtermiņa uzturēšanas kontrakti,
- produkts tiek darbināts ģeogrāfiski attālinātās vietās,
- produkts tiek darbināts atšķirīgās (nehomogēnās) vidēs,
- klients pieprasa augstu servisa līmeni (*first level support*).

5. KVALITATĪVAIS PĒTĪJUMS

5.1 Socioloģiskās teorijas nepieciešamība

Attīstot un aktīvi popularizējot viedo tehnoloģiju idejas, autore nereti saskārusies ar grūti atbildamo jautājumu – kā pierādīt, ka paveiktais ir uzskatāms par zinātnisku/ pētniecisku novitāti, kā „garantēt", ka viedo tehnoloģiju idejas un implementāciju rezultāti ir ne vien unikāli, bet arī industrijai nepieciešami. Vēlme rast apstiprinājumu izvēlētā ceļa pareizībai un pētījuma rezultātu nozīmīgumam ir labi saprotama. Viens no veidiem, kā gūt „objektīvu" skatu uz pētījumu un tā rezultātiem, ir veikt atbilstošu pētījumu.

Diemžēl Latvijas IT nozares zinātniskajos darbos un pētījumos rezultātu nozīmīguma pierādīšanai, kā likums, lieto „pseido-socioloģisku" pierādīšanas metodi, kura parasti sastāv no brīvi izvēlētu (aptaujai pieejamu) IT uzņēmumu pārstāvju anketēšanas (izvēle no iepriekš definētām atbildēm) un iegūto (šķietami reprezentatīvo!) aptaujas datu attēlošanas apkopotā formā, pievienojot pētījuma autoru skaidrojumu par anketēšanas rezultātiem. Autore ir iepazinusies ar vairākiem šādiem pētījumiem, tai skaitā ir bijusi iesaistīta „reprezentatīvajos pētījumos" kā respondente, un intuitīvi nespēja noticēt šādas metodes pielietojamībai viedo tehnoloģiju kontekstā.

No augšminētā loģiski izrietēja jautājums: Vai tiešām zinātniskā darbā izvirzītās hipotēzes iespējams pārbaudīt tikai un vienīgi kvantitatīva pētījuma ceļā?

Diskutējot šo problēmu ar socioloģijas zinātņu maģistri Lieni Odīti [64], kā arī iepazīstoties ar Lienes Odītes maģistra darbu [65], autore guva negaidītu apstiprinājumu savām šaubām. Socioloģijā tiek lietotas un akceptētas ne tikai kvantitatīvās (uz statistikas datiem balstītas) metodes, bet arī kvalitatīvas metodes, turklāt pētījumi var tikt veikti dažādu ideoloģiju satvaros. Vienkāršoti skaidrojot, kvantitatīvais pētījums noskaidro respondentu attieksmi pret jautājumu, kamēr kvalitatīvs pētījums analizē fenomena būtību.

Līdz ar to autorei radās interese iepazīties ar socioloģisko pētījumu pamatnostādnēm. Nākamajās apakšnodaļās sniegts ieskats autores izvēlētajā metodikā, uz kuru balstoties tika validēti viedo tehnoloģiju pielietojamības aspekti.

5.2 Fenomenoloģiskā pieeja

Tā kā izšķiršanās par vai pret viedo tehnoloģiju pielietošanu parasti notiek individuālā līmenī, resp., šis lēmums pārsvarā tiek pieņemts nelielā lēmēju lokā un balstoties uz dalībnieku subjektīviem priekšstatiem, individuālām izjūtām un preferencēm, būtiski ir izzināt potenciālo šādu lēmumu pieņēmēju refleksiju, pārdomas, ko iespējams panākt ar dziļo interviju palīdzību.

Tādēļ šāda veida pētījuma uzdevumu veikšanai piemērota ir fenomenoloģiskā pieeju, kuras pamatā ir uzskats, ka sociālā realitāte ir interpretēšanas procesa rezultāts.

Fenomenoloģiskā pieeja pārstāv uzskatu, ka pētnieka zinātniskie skaidrojumi rodas viņa paša pieredzes ietvaros un ka visi zinātniskie skaidrojumi kopumā šo sfēru paplašina, kas liek pētniekam iegūt jaunu pieredzi, uzdot jaunus jautājumus un radīt jaunus skaidrojumus.

Fenomenoloģija ir filozofijas skola, kuras pamatlicējs ir Edmunds Huserls [66]. Fenomenoloģija bija izplatīta visā Eiropā, un īpaši nozīmīgs šis virziens bija agrīnā eksistenciālisma laikā.

Huserls centās izveidot universālu filozofijas metodi, brīvu no aizspriedumiem, kas koncentrējas tikai un vienīgi uz fenomenu izpēti un aprakstīšanu. Parādības, kas nebija saredzamas vai nekavējoties uztveramas, no pētījuma tika izslēgtas. Galvenais uzsvars tika likts uz zināmo, neiedziļinoties jautājumā, kā tas kļuvis zināms.

Līdz ar to fenomenoloģiskā metode nav ne deduktīva, ne empīriska, kā tas pierasts dabas zinātnēs; tā sevī ietver objekta esamības konstatēšanu un tā nozīmes izskaidrošanu. Huserls fenomenoloģiskajā metodē saredzēja vīzijas iespēju, fenomena ideālu izskaidrojumu.

[67] fenomenoloģiju definē kā teoriju, kas matemātiski izsaka novērotā fenomena rezultātus, nepievēršot uzmanību fenomena fundamentālajai nozīmei. Zinātnē fenomenoloģijas jēdziens tiek izmantots, lai raksturotu zināšanu kopu, kas attiecināma uz empīriskiem novērojumiem par fenomenu, ir saskaņā ar fundamentālo teoriju, taču tieši neizriet no teorijas.

Dabaszinātnēs, jo īpaši fizikā, ir gadījumi, kad fenomenus aprakstošas teorijas nav iespējams izsecināt no kāda fundamentāla principa [68]. Tam var būt vairāki iemesli. Piemēram, nepieciešamā teorija vēl nav izveidota vai matemātiskais aparāts, lai aprakstītu novērojumus, ir pārlieku sarežģīts.

Šādos gadījumos dažkārt mēdz lietot vienkāršas algebriskas izteiksmes, lai modelētu novērojumus vai eksperimentālus rezultātus. Un, ja izvēlētie algebriskie modeļi ir pietiekami precīzi, bieži vien zinātniskā vide pieņem šos modeļus un lieto tos, lai gan, formāli raugoties, šie modeļi nevar tikt (vai vēl nav tikuši) atvasināti no attiecīgo zināšanu kopu aprakstošās fundamentālās teorijas.

Robežas starp teoriju un fenomenoloģiju, kā arī starp fenomenoloģiju un eksperimentu ir plūstošas. Atsevišķi zinātnes filozofi, jo īpaši Nansija Kārtraita uzskata, ka visi dabas fundamentālie likumi ir vairākkārtēji fenomenoloģiski vispārinājumi [69].

Statistikā jebkuru kvantificējošu datu apkopošana sevī ietver fenomenoloģisku soli. Lai apkopotu datus, ir jāizveido jautājumu kopa, kuras atbildes ir mērāmas un var tikt apkopotas loģiskā veidā. Turklāt izvēlētā jautājumu forma nedrīkstētu no statistiskā viedokļa izkropļot rezultātus. Ja nav nodrošināta jautājumu neitralitāte, tiks saņemtas ietekmētas un/vai savstarpēji nesalīdzināmas atbildes, tādēļ savāktie dati vispār nebūs izmantojami [70].

Laba aptauja ir tāda, kurā katram respondentam tiek dota iespēja atbildēt skaidri un viennozīmīgi, turklāt visi respondenti jautājumu saprot vienādi. Piemēram, uzdodot zemniekam jautājumu „Ar cik lielu risku jūs sastopaties savā zemnieku saimniecībā?" un par iespējamām norādot atbildes skalā no „nav riska" līdz „ļoti riskanti", apkopotie dati spiestā kārtā nebūs objektīvi, jo katra zemnieka izpratne par risku ir citāda. Visi zemnieki cieš no sausuma u.c. dabas stihijām, bet dažu skatījumā tās ir normāla lauksaimnieciskas darbības sastāvdaļa, kamēr citiem – negaidīta katastrofa.

Aptaujājot respondentus un kodējot (pierakstot) to sniegtās atbildes ar skaitliskām vērtībām, ir jāizkopj aptaujas tehnika, kas novērš kļūdu un pārpratumu rašanās iespējamību. Vārdu sakot, laba aptaujas izveides metodika ļauj pētniekam izveidot apmierinošu nozīmju „tiltu" starp loģiskiem un praktiskiem nosacījumiem, kas uzstādāmi anketētājam, statistiku klasifikācijas shēmu, respondentu apziņu un savāktajiem jēldatiem. Šī „tilta" atrašana prasa abstrakcijas procesu, kas ietver loģiskās sakarības, teoriju, eksperimentu un labu tiesu „mākslas", jo ir jāizveido atbilstoša saite starp lietoto valodu, starp-personu attiecībām starp aptaujātāju un respondentu, kā arī abu pušu veidu, kā idejas nozīme tiek atainota un izpausta valodā. Šim izziņas procesam ir neiespējami izveidot vispār pielietojamu standarta procedūru, bet gan tikai ieteikumus, kuru pielietošanai vienmēr būs nepieciešams „veselais cilvēka saprāts" [71].

5.3 Kvalitatīvās pētījumu metodes

Kvalitatīvie pētījumi ir pielietojami daudzās nozarēs un dažādiem pētījumu apgabaliem [72]. Kvalitatīvie pētnieki tiecas labāk izprast cilvēka izturēšanos un iemeslus, kas šo uzvedību vada un pamato. Kvalitatīvās metodes pēta lēmumu pieņemšanas „kāpēc?" un „kā?", ne tikai „kas?", „kad?" un „kur?". Līdz ar to kvalitatīvajos pētījumos ir nepieciešams mazāks skaits, taču mērķtiecīgi atlasītu datu, nevis liels skaits gadījuma tipa datu. [73]

5.3.1 Ieskats vēsturē

Kvalitatīvie pētījumi bija viena no pirmajām sociālo studiju formām (tipiski pārstāvji, piemēram, Broņislavs Maļinovskis vai Eltons Majo). 20. gadsimta 50. līdz 60.gados, kad kvantitatīvā zinātne sasniedza savas popularitātes augstāko virsotni („Kvantitatīvā revolūcija"), kvalitatīvo pētījumu virziens nozīmībā pabalēja un atzīšanu atguva tikai 70-tajos gados.

Līdz 1970. gadam jēdziens „kvalitatīvs pētījums" tika lietots vienīgi antropoloģijas vai socioloģijas nozaru apzīmēšanai. 20. gs. 70.-80. gados kvalitatīvā pētījuma metodi sāka pielietot citās nozarēs, un šī forma kļuva par būtisku pētījumu metodi izglītības, sociālā darba, vadības zinātņu, psiholoģijas, komunikāciju zinību un daudzās citās sfērās.

Kvalitatīvie pētījumi tika veikti, lai pētītu patērētāju uzvedību, kas ietekmē jaunu un esošu produktu pozicionēšanu/ popularizēšanu. Paralēli turpinājās diskusija par kvalitatīvo un kvantitatīvu pētījumu patieso vietu pētniecībā.

80. gadu beigās un 90-tajos gados, reaģējot uz kvantitatīvās pieejas aizstāvju kritiku, tika izveidotas jaunas kvalitatīvās metodes, kas risināja uzticamības un neprecīzas datu analīzes problemātiku. Šajā gadu desmitgadē reklāmas industrija piedzīvoja tradicionālo mediju ietekmes lejupslīdi, kas paaugstināja industrijas interesi par pētījumiem, kas varētu veicināt reklāmas efektivitātes pieaugumu.

Pēdējos 30 gados kvalitatīvo pētījumu nozīme ir augusi. Jau kopš 70-to gadu beigām daudzi vadošie zinātniskie izdevumi sāka publicēt zinātniskus rakstus par kvalitatīvajiem pētījumiem [74], izveidojās pat jauni izdevumi, kuri koncentrējās tikai uz kvalitatīvo pētījumu rezultātiem un metodēm. Šī tendence turpinās arī 21. gadsimtā ar arvien jauniem zinātniskiem izdevumiem.

5.3.2 Kvalitatīvās pieejas atšķirība no kvantitatīvās

Pirmkārt, kvalitatīvajos pētījumos gadījumi (pētāmie objekti) var tikt speciāli atlasīti atkarībā no tā, vai tie atbilst (neatbilst) noteiktām īpašībām vai kontekstam. Otrkārt, lielāku lomu spēlē paša pētnieka loma, viedoklis vai nostāja, jo kvalitatīvajos pētījumos pētniekam ir problemātiskāk (grūtāk) ieņemt neitrālu pozīciju. Tādēļ bieži vien kvalitatīvo pētījumu autoriem tiek uzstādīta prasība pētījumā skaidri atspoguļot savu lomu un nostāju.

Treškārt, tā kā kvalitatīvās datu analīzes process var tikt veikts ļoti dažādos veidos, iegūtie rezultāti mēdz atšķirties no kvantitatīvo pētījumu rezultātiem, īpaši attiecībā uz valodu, zīmēm, nozīmi. Kvalitatīvie pētījumi analīzes procesā lieto kontekstuālu pieeju, nevis izolētu un reducējošu. Tas tomēr nenozīmē, ka kvalitatīvie pētījumi tiktu atbrīvoti no katrā zinātnes virzienā pašsaprotamās vajadzības pēc skaidrības un sistemātiskas pieejas. Ļoti daudzas kvalitatīvo pētījumu metodes uzliek pētniekiem par pienākumu rūpīgi kodēt datus, tēmas, dokumentus un tos apkopot nepretrunīgā un uzticamā veidā.

Iespējams, pats tradicionālākais sadalījums starp kvalitatīvo un kvantitatīvo metožu pieejām noticis sociālajās zinātnēs. Tur kvalitatīvās metodes tiek uztvertas par izskaidrojošām („hipotēzes ģenerējošām"), bet kvantitatīvās metodes tiek lietotas hipotēžu pārbaudei. Kvalitatīvo metožu spēks ir spējā atbildēt uz jautājumu: „Vai mērījumi patiešām izmēra to, ko pētnieki uzskata, ka tie mēra?". Kvantitatīvie pētījumi tiek uzskatīti par reprezentatīvākiem, uzticamiem un precīziem, ja jāpierāda izvirzītas hipotēzes, ja lieto mērīšanas rīkus un lietišķās matemātikas iespējas. Kvalitatīvo pētījumu rezultātus savukārt ir grūti attēlot grafu vai matemātisku rādītāju formā.

Kvalitatīvie pētījumi bieži tiek lietoti, lai novērtētu pieeju un attīstību, jo tie uz svarīgiem jautājumiem spēj atbildēt efektīvāk nekā kvantitatīvās metodes. Jo īpaši svarīgi tas ir gadījumos, kad jāsaprot, kā un kāpēc zināms iznākums (rezultāts) iegūts, nevis – kas tika iegūts. Kvalitatīvie pētījumi spēj izskaidrot svarīgus aspektus – relevanci, negaidītas ietekmes, efektus u.tml. Piemēram, „Vai uz kaut ko liktās cerības ir pamatotas?", „Vai procesi funkcionēja, kā gaidīts?", „Vai atslēgas personas bija spējīgas tikt galā ar saviem pienākumiem?", „Kādi bija negaidītie efekti, ko konstatēja?"

Kvalitatīvo pētījumu priekšrocība ir iespēja sniegt daudz diversificētākas atbildes, kā arī parādās iespēja variēt aptaujas saturu, pat formu atkarībā no saņemtām atbildēm. Tā kā individuāli kvalitatīvie pētījumi ir dārgi un laikietilpīgi, daudzās nozarēs tiek lietotas jau

iepriekš izveidotas kvalitatīvās metodes, kas speciāli piemērotas konkrētu jautājumu apskatei un ir laika/ izmaksu ziņā pieticīgākas. Labi zināms šādas specifiskas formalizētas adaptācijas piemērs ir *Rapid Rural Appraisal* metode [75].

5.3.3 Datu savākšana

Lai savāktu nepieciešamos datus, kvalitatīvo pētījumu autori ir spiesti izmantot citas metodes – pamatotās teorijas pieeju, naratoloģiju, klasisko etnogrāfiju u.c. Kvalitatīvās metodes ir iekļautas arī citās metodoloģiskajās pieejās, piemēram, aktivitāšu pētniecībā vai aktiera-tīkla teorijā.

Datu savākšanas forma var ietvert intervijas, grupu diskusijas, novērojumus, piezīmju, tekstu, attēlu un citu materiālu apstrādi. Lai labāk apkopotu un attēlotu pētījumu rezultātus, kvalitatīvo pētīju ietvaros apkopotie dati tiek kategorizēti saskaņā iepriekš definētiem šabloniem. Parasti kvalitatīvo pētījumu autori vadās pēc šādām informācijas savākšanas metodēm:

- dalībnieku novērošana,
- nepiedalošos novērošana,
- piezīmes,
- refleksīvas pārdomas,
- strukturēta intervija,
- nestrukturēta intervija,
- dokumentu un materiālu analīze [76].

Dalības un novērošanas pieejas var būt ļoti atšķirīgas dažādos pētījumos. Dalībnieku novērošana ir ne vien vienkārša iepazīšanās, bet refleksīvās apmācības forma [77]. Novērojot dalībniekus, pētnieki kļūst par kultūras un grupas dalībniekiem un gūst iespēju pieskaņoties nepieciešamajām lomām, kā arī labāk saprast kultūras paradumus, motivāciju un emocijas.

Atsevišķas kvalitatīvās metodes pielieto fokusa grupu un atslēgas informantu pieeju. Fokusa grupa ietver speciāli organizētas diskusijas nelielās grupās par kādu noteiktu tēmu. Šī metode īpaši iecienīta ir tirgus pētījumu gadījumā vai lai pārbaudītu jaunu ideju dzīvotspēju.

Cita tradicionāla un specializēta kvalitatīvo pētījumu forma ir tā saucamā kognitīvā jeb pilota testēšana, kura tiek lietota kvantitatīvu aptauju izstrādē. Sagatavotie aptauju piloti tiek testēti uz izvēlētiem dalībniekiem, lai pārliecinātos par aptauju uzticamību un izmantojamību.

5.3.4 Datu analīze

Savākto datu analīzei un apkopošanai var tikt izmantotas dažādas datu analīzes tehnikas. Zemāk dots pārskats par zināmākajām.

Interpretējošā tehnika

Visbiežāk izmantotā pieeja, lai analizētu kvalitatīvos datus, ir novērotāja iespaidi. Tas nozīmē, ka pētnieks (novērotājs) caurskata datus, interpretē tos, pārveidojot iespaidos un tos atspoguļo strukturētā, dažkārt arī kvantitatīvā formā.

Kodēšana

Kodēšana ir viena no interpretējošām tehnikām, kas reizē kārto datus un pieļauj interpretāciju izmantošanu noteiktās kvantitatīvās metodēs. Vairums kodēšanas pieeju nosaka, ka analītiķis lasa datus un atzīmē tajos segmentus. Katrs segments tiek atzīmēts ar „kodu" – parasti ar vārdu vai kādu īsu frāzi, kas asociē konkrētos datus ar pētījuma mērķiem.

Kad kodēšana pabeigta, analītiķis veic kodu biežuma analīzi, apskata līdzības un atšķirības starp saistītiem kodiem, oriģinālajiem avotiem, kontekstu u.c.

Atsevišķos gadījumos labi strukturēti kvalitatīvie dati (piemēram, atvērtie jautājumi aptaujas anketās) jau kalpo par kodiem, tādēļ papildus segmentēšana vairs nav nepieciešama.

Mūsdienās kvalitatīvajai analīzei tiek izmantota specializēta programmatūra, kas analītiķa darbu būtiski atvieglo, jo spēj efektīvi uzglabāt datus, tajos meklēt, piekārtot kodus utt. Programmatūra arī sniedz efektivitātes uzlabojumus – ir iespējama rekursīva datu apstrāde, paralēlā caurskatīšana u.tml.

Kodēšanas metodēm bieži tiek pārmests, ka tās cenšas kvalitatīvus datus pārstrādāt kvantitatīvos un tādējādi laupa datiem to dažādību, bagātību un individuālo raksturu.

Rekursīvā abstrakcija

Dažas kvalitatīvo datu kopas var tikt analizētas bez kodēšanas. Tādos gadījumos parasti izmanto rekursīvās abstrakcijas metodi, kas paredz pakāpenisku datu kopu apvienošanu vairākos līmeņos. Kā gala rezultāts veidojas kompakts kopsavilkums, kurš būtu grūti uztverams, ja nebūtu notikuši iepriekšēja satura „destilēšana".

Rekursīvai abstrakcijai tiek pārmests, ka gala secinājumi ir vairākkārtīgi atvasināti no sākuma datiem, tādējādi it kā zaudējot svarīgu informāciju vai pārveidojot to. Analītiķi uz to

atbild ar rūpīgiem pierakstiem, pētījuma ietvaros dokumentējot eliminētos (dzēstos, nepārņemtos) datus un norādot uz atstātajiem datiem, kas tiks iekļauti gala apkopojumā.

Mehāniskās tehnikas

Mehāniskās tehnikas izmanto datoru sniegtās iespējas, lai apstrādātu lielas kvalitatīvu datu kopas. Pamata līmenī datori datos saskaita vārdus, frāzes, saistītas pazīmes utt. Iegūtais rezultāts, bieži saukts arī par satura analīzi, tiek tālāk izmantots sarežģītākā statistiskā analīzē.

Mehāniskās tehnikas ir īpaši piemērotas noteiktos gadījumos. Viens gadījums ir tad, kad kvalitatīvo datu kopa ir pārāk liela, lai cilvēks to spētu efektīvi apstrādāt, vai tad, ja apstrādes izmaksas ir pārāk lielas salīdzinājumā ar potenciāli iegūstamo informāciju. Otrs tipisks piemērs ir iestatītu „vadošo vērtību" meklēšana plašā datu kopā: meklēt publikācijās datus par noteiktiem klīnisko nāvju gadījumiem vai preses apskatos noteikta zīmola raksturojumus u.tml.

Mehāniskās tehnikas bieži tiek kritizētas, jo to rezultātus nav interpretējis cilvēks. Lai arī programmatūras pamatā ir speciālistu radīti, augsti attīstīti algoritmi, rezultāts pats var būt salīdzinoši mehānisks. Analītiķi uz pārmetumiem atbild, veidojot laboratorijas eksperimentus, kur veido salīdzinošus pētījumus mehāniski un ar cilvēku piedalīšanos.

5.3.5 Pieeju novērtēšana

Kvalitatīvo pētījumu divi centrālie aspekti ir ticamība un neatkarība. Ir dažādi ticamības pārbaudīšanas veidi: dalībnieku pārbaude, intervētāju apliecinājumi, pāru izvērtēšana, pagarināta apstrāde, negatīvo gadījumu analīze, apstiprinājumi u.c. Vairums no šīm metodēm ir izsmeļoši aprakstītas [78].

5.4 Kvalitatīvā pētījuma apraksts

5.4.1 Mērķis un uzdevumi

Šī darba ietvaros veidotā kvalitatīvā pētījuma mērķis bija pētīt viedo tehnoloģiju fenomenu.

Augstāk minētā mērķa sasniegšanai tika izvirzīti divi uzdevumi:

1. identificēt viedo tehnoloģiju (vai tām saturīgi radniecīgu risinājumu) izplatību (pielietošanas biežumu un intensitāti) Latvijas IT industrijas uzņēmumos;
2. izzināt un analizēt viedo tehnoloģiju pielietošanas robežas no ekonomiskajiem un organizatoriskajiem aspektiem.

Kvalitatīvajam pētījumam būtu jāsniedz atbildes uz jautājumiem par viedo tehnoloģiju izpausmēm (kādas īpašības var tikt uzskatītas par viedo tehnoloģiju principiem atbilstošām un kuras – nē?), par tēmas aktualitāti (vai un kādos gadījumos viedās tehnoloģijas būtu izmantojamas?), par izplatītākajiem (tipiskajiem) pielietošanas apgabaliem. Īpašs uzsvars pētījumā liekams uz reālas pieredzes rezultātā gūtām atziņām, uz IT profesionāļu-praktiķu novērojumiem un ieteikumiem.

Viedo tehnoloģiju (un to saturisko līdzinieku) izplatības identificēšana varētu sniegt izziņu par radniecīgu pētījumu un risinājumu esamību vai neesamību Latvijas IT industrijā. Ja pētījuma ietvaros izdotos konstatēt centienus uzlabot tirgū izplatītos programmatūras risinājumus, papildinot tos ar viedo tehnoloģiju īpašībām, tas, no vienas puses, liecinātu par zināmu industrijas tendenci, no otras - apstiprinātu izvēlētā pētījumu virziena aktualitāti.

Viedo tehnoloģiju pielietošanas robežu izzināšana, savukārt, varētu palīdzēt atbildēt uz pragmatisko jautājumu – kurā brīdī (kad?, kuros gadījumos?) šo tehnoloģiju pielietošana ir mērķtiecīga („jēgpilna", „racionāla")? Kas ir tie faktori, kas izstrādātājus mudina izmantot viedo tehnoloģiju idejas vai, tieši otrādi - attur no to izmantošanas? Kā jau diskutēts darba iepriekšējās nodaļās, par robežas noteicošiem aspektiem varētu tikt uzskatīti ekonomiskie un organizatoriskie, bet arī kvalitātes un reputācijas aspekti. Šo pieņēmumu pārbaude arī būtu pētījuma sastāvdaļa.

5.4.2 Respondentu izvēle

Jebkura pētījuma neatņemama sastāvdaļa ir uzstādītajam pētījuma mērķim atbilstošas (reprezentatīvas) pētījuma dalībnieku grupas (respondentu kopas) izvēle.

Tā kā kvalitatīvi pētīt kādu fenomenu iespējams tikai tad, ja aptaujāti tiek respondenti, kuru profesionālā sagatavotība un pieredze ļauj izprast fenomena pamatā liktos netriviālos pieņēmumus, par respondentiem varēja tikt izvēlēti trīs savstarpēji mazliet atšķirīgu grupu pārstāvji:

- informācijas tehnoloģiju profesionāļi, kas ir pazīstami ar viedo tehnoloģiju jēdzienu (tajā nozīmē, kādā tas diskutēts šajā darbā) un saskārušies ar šiem principiem atbilstošiem risinājumiem;
- informācijas tehnoloģiju profesionāļi, kuri gan tiešā veidā nav saskārušies ar viedo tehnoloģiju jēdzienu, bet kuru izglītība, profesionālā sagatavotība un darba pieredze ļauj izprast minēto jēdzienu un identificēt savā praksē sastaptu/ izveidotu analoģisku risinājumu piemērus;
- gala lietotāji, kuru priekšzināšanas un darba pieredze ļauj izprast viedo tehnoloģiju jēdzienu, vēlama būtu pieredze viedo tehnoloģiju vai tām radniecīgu risinājumu ieviešanā un/ vai izmantošanā.

Pirmās grupas dabiskākie pārstāvji ir uzņēmuma „Datorikas institūts DIVI" darbinieki, no kuriem daudzi tiešā vai netiešā bijuši iesaistīti viedo tehnoloģiju idejas attīstībā. Tā kā viedo tehnoloģiju fenomenu paredzēts pētīt kvalitatīva pētījuma veidā, kas paredz padziļinātu interviju ar katru no respondentiem, tas ierobežo potenciāli piesaistāmo respondentu loku uz ne vairāk kā 10 personām. Līdz ar to par respondentu kopu izvēlēsimies „Datorikas institūts DIVI" projektu/ biznesa centru vadītājus, no kuriem katram ir pakļauti 3-8 darbinieki. Pētījumā šīs respondentu grupas viedokļi varētu tikt uzskaitīti par savdabīgu „iekšējo indikatoru", resp., paust ražošanas procesā iesaistītu darbinieku attieksmi pret pašu uzņēmumā izauklētu inovāciju un tās pielietošanas iespējām.

Otrās grupas dalībnieki – citu Latvijas IT uzņēmumu pārstāvji. Pieņemot, ka programmatūras izstrādes optimizācija ir aktuāla visos ražošanas uzņēmumos, pētījuma uzdevums būtu identificēt radniecīgas izstrādes, kuras veidotas citu uzņēmumu projektu n pētījumu ietvaros, kā arī noskaidrot respondentu viedokli par vai pret šādu risinājumu izveidi un/ vai izmantošanu. Šīs respondentu grupas pārstāvju atlase ir netriviāls uzdevums, jo,

pirmkārt, jāspēj atrast patiesi kvalificētus cilvēkus, kas spējīgi viedo tehnoloģiju analogus arī identificēt un raksturot, otrkārt, jāpanāk respondentu gatavība dalīties savu uzņēmumu „biznesa noslēpumos", par kādiem nereti tiek uzskatīta specializētu tehnoloģiju izmantošana programmatūras izstrādes procesā. Pētījumā šīs respondentu grupas viedokļi kalpoja par „skatu no malas", resp., parādīja ārpus stāvošu profesionāļu attieksmi pret Latvijas informācijas tehnoloģiju uzņēmumā veidotām inovācijām.

Trešās no augšminētajām potenciāli aptaujājamām grupām – gala lietotāju pārstāvju – iesaistīšana pētījumā ir pamatota ar vēlmi uzklausīt inovāciju „patērētāju" viedokli, resp., noskaidrot, kā viedo tehnoloģiju idejas un pielietošanas aspektus uztver klienti, kuru projektos ar viedo tehnoloģiju iespējām aprīkotie programmprodukti tiks izmantoti.

Pētījums apzināti tika veidots, orientējoties uz Latvijas tirgu un tajā strādājošiem uzņēmumiem un profesionāļiem. Tika uzskatīts, ka Latvija informatizācijas jomā Eiropā neuzrāda būtiskas novirzes no citām valstīm speciālistu sagatavotības un inovāciju (tehnoloģisko jauninājumu) ieviešanas ziņā. Un, ja arī šāda novirze būtu bijusi, viedās tehnoloģijas kā Latvijā radusies un attīstīta ideja pirmkārt būtu jāvalidē Latvijas speciālistiem un lietotājiem.

5.4.3 Jautājumu grupas

Katra pētījuma neatņemama sastāvdaļa ir aptaujas plāna un jautājumu (anketas) sagatavošana. Atšķirībā no kvantitatīvajiem pētījumiem, kad respondentiem uzdodamie jautājumi un uz tiem iespējamās atbildes (atbilžu pieļaujamās vērtības) ir fiksēti, kvalitatīvā pētījuma „anketa" sastāv no atvērtā tipa jautājumiem un ir grupēti radniecīgās jautājumu kopās.

Viedo tehnoloģiju fenomena pētījuma anketā tika iekļautas šādas jautājumu/ tēmu grupas.

- Informācija par respondentu, viņa pieredzi, profesionālo profilu.
- Attieksme pret inovācijām kopumā, pieredze to izstrādē / pielietošanā.
- Viedās tehnoloģijas pamatidejas, galveno jēdzienu un nostādņu raksturojums (ja respondentam jau nav labi pazīstams).
- Attieksme pret viedo tehnoloģiju ideju kopumā, intuitīva ticība/ neticība tām.

- Diskusija par viedo tehnoloģiju pielietojamību dažādām projektu/ uzdevumu grupām.
- Tehniskas un organizatoriskas dabas apsvērumi, kas runā par vai pret viedo tehnoloģiju izmantošanu.
- Ekonomiskie kritēriji, kas varētu liecināt par viedo tehnoloģiju pielietošanas lietderīgumu.
- Ierosinājumi, atziņas, kopējais novērtējums.

Turpmāk īsumā ieskicēta paredzētā aptaujas norise sadalījumā pa augstāk minētajām tēmām.

Informācija par pretendentu pētījumā pilda konteksta izziņas lomu. Fiksējot respondenta profesionālās sagatavotības līmeni, pieredzi, gūto izglītību, tiek iegūti izmērāmi un salīdzināmi lielumi, kas varētu sniegt papildus informāciju respondenta sniegto atbilžu analīzes un apkopošanas procesā. Šīs aptaujas sadaļā gūtās atbildes ir sava veida „objektīvā" respondenta mēraukla.

Attieksme pret inovācijām kopumā, pieredze to izstrādē/ pielietošanā ļauj gūt priekšstatu par respondenta subjektīvajām īpašībām – gatavību pieņemt jauninājumus, personīgo attieksmi pret inovācijām, vēlmi būt iesaistītam inovāciju pārnesē. Atšķirībā no iepriekšējā sadaļā iegūtās informācijas, kas satur objektīvi pārbaudāmu informāciju, šajā aptaujas sadaļā gūtās atbildes satur „subjektīvo" respondenta raksturojumu.

Ja respondents nav bijis iesaistīts viedo tehnoloģiju idejas tapšanā un attīstībā, anketā nepieciešams iekļaut īsu idejas aprakstu, kā arī raksturot pazīmes, pēc kurām respondents, balstoties uz savu iepriekšējo pieredzi, varētu identificēt radniecīgas idejas un risinājumus. Ja respondents ar viedās tehnoloģijas ideju ir pazīstams, šī aptaujas sadaļa reducējas uz diskusiju par pamatnostādnēm un jēdzieniem. Neatkarīgi no respondenta sagatavotības šī aptaujas sadaļa tiek veidota organizēta kā diskusija, nevis jautājumu un atbilžu kopa.

Pēc iepazīšanās ar viedo tehnoloģiju ideju (vai, attiecīgi, - pēc diskusijas par viedo tehnoloģiju pamata nostādnēm) tiek noskaidrota respondenta attieksme pret viedo tehnoloģiju ideju kopumā. Vispārīgā līmenī ir iespējams noskaidrot ne vien intuitīvu uzticību vai neticību idejai, bet arī attieksmes pamatojumu un pat ieteikumus idejas tālākai vai citādai attīstībai.

Jautājumu kopā par viedo tehnoloģiju pielietojamību dažādām projektu un/ vai uzdevumu grupām tiek veidota, lai izzinātu viedo tehnoloģiju pielietošanas robežas. Īpaši

svarīgi ir noskaidrot tos tipiskos uzdevumus vai problēmas, kuru risināšanai viedo tehnoloģiju risinājums potenciāli būtu pielietojams vai arī jau ir sekmīgi ticis pielietots.

Izvērtējot viedo tehnoloģiju pielietošanas robežas, spiestā kārtā tiek apkopoti arī subjektīvi un objektīvi faktori, kas varētu ietekmēt lēmumu par vai pret viedo tehnoloģiju pielietošanu katrā konkrētā gadījumā. Pirmkārt, tie ir tehniskas un organizatoriskas dabas apsvērumi, resp., respondentiem tiek uzstādīts jautājumi par realizācijas diktētiem ierobežojumiem, par darba grupas sastāva aspektiem, par sadarbību ar klientu u.tml. Aptaujas nostādne šajā fāzē balstās uz ietekmējošu kritēriju identificēšanu, abstrahējoties no jautājuma par izmaksām, kas ar viedo tehnoloģiju implementāciju saistītas. Šī ir aptaujas kritiskākā sadaļa, jo varētu arī novest pie strupceļa - ja, piemēram, respondents nav spējīgs vai gatavs iedziļināties jautājumos par viedo tehnoloģiju realizācijas īpatnībām, nav domājis par organizatorisko aspektu u.tml.

Viens no aptaujas centrālajiem uzdevumiem ir noskaidrot ekonomiskos kritērijus, kas varētu kalpot par pamatu (sava veida etalonu) lēmuma pieņemšanas procesā par vai pret viedo tehnoloģiju izmantošanai konkrētā programmproduktā. Jautājumu izveides pamatā tika likti iepriekšējās šī darba nodaļās diskutētie efektivitātes kritēriji. Respondenti, kuri jau ir darbojušies ar produktiem, kuros iekļautas viedo tehnoloģiju īpašības, tika aicināti kvantitatīvi novērtēt savā praktiskajā pieredzē konstatētās „kritiskās" kritēriju vērtības, resp., noteikt vai novērtēt kritēriju vērtības, kuru iestāšanās gadījumā tiek pieņemts cits lēmums. Respondenti, kuriem nav praktiskas pieredzes ar viedo tehnoloģiju implementēšanai un/ vai izmantošanu, tika aicināti subjektīvi novērtēt piedāvāto ekonomisko kritēriju lietderību un to atbilstību tādam lielumam, uz kura vērtību tiek balstīta lēmuma pieņemšana.

Aptaujas noslēgumā tiek veidota rezumējoša diskusija, kuras ietvaros kopīgi ar respondentu tiek formulēts atzinums/ vērtējums par viedo tehnoloģiju pielietošanas aspektiem, apspriestas praktiskās pieredzes atziņas un akadēmiski ieteikumi viedo tehnoloģiju idejas tālākai attīstībai.

5.4.4 Pētījuma norise un iegūtie dati

Pētījums tika organizēts individuālu interviju formā, kur katrs respondents atbilstoši iepriekšējā apakšnodaļā aprakstītajam plānam 45-60 minūšu sarunā tika aicināts izteikt savu viedokli un dalīties pieredzē par viedo tehnoloģiju ideju un tās realizācijām. Zemāk interviju ietvaros iegūtā informācija ir konspektīvi atreferēta.

Informācija par respondentiem, viņu pieredzi, profesionālo profilu

Pavisam tika aptaujāti 24 respondenti – 9 no tiem bija „Datorikas institūts DIVI" projektu vadītāji, 5 bija citu Latvijas informācijas tehnoloģiju uzņēmumu pārstāvji un 10 bija informācijas tehnoloģiju risinājumu, kuros iekļautas viedās tehnoloģijas, lietotāji.

Aptaujājamo personu lokā bija 2 datorzinātņu doktori, 6 datorzinātņu maģistri, 5 datorzinātņu bakalauri, 2 profesionālie programmētāji, 1 informācijas tehnoloģiju nozarē strādājošs autodidakts ar nepabeigtu augstāko izglītību un 8 bija bez izglītības informācijas tehnoloģiju nozarē un bez darba pieredzes informācijas tehnoloģiju uzņēmumos.

No respondentiem 7 bija sievietes (no tām 1 ir informācijas tehnoloģiju speciāliste), pārējie – vīrieši (speciālistu sadalījums apmēram atbilst Latvijā *de facto* eksistējošai dzimumu dalībai informācijas tehnoloģiju profesijā). Jaunākais respondents bija 22 gadus vecs, vecākais – 65.

No aptaujātajiem informācijas tehnoloģiju speciālistiem lielākajai daļai bija pieredze akadēmiskā darbā (tikai 2 nebija); visi respondenti kādā savas dzīves posmā (vairums – joprojām) bija bijuši projektu vadītāju statusā reālos ražošanas projektos. Vairumam bija arī pieredze zinātniskā un/ vai pētnieciskā darbā. No respondentiem bez pieredzes informācijas tehnoloģiju risinājumu izstrādē visiem bija bijušas priekšzināšanas darbā ar programmatūras risinājumiem, resp., ar viedām tehnoloģijām aprīkots risinājums nebija pirmais risinājums, ar ko šie respondenti savā dzīvē saskārās.

Attieksme pret inovācijām kopumā, pieredze to izstrādē / pielietošanā

Visi aptaujātie informācijas tehnoloģiju nozarē nodarbinātie respondenti bija labi informēti par inovāciju nozīmi tautsaimniecībā; lielākā daļa paši ir piedalījušies zinātnes/ pētniecības sasniegumu pārneses (inovāciju) projektos. Respondentiem bez pieredzes informācijas tehnoloģiju risinājumu izstrādē, kā atklājās intervijā, bija visai miglains

priekšstats par inovācijām, līdz ar to autorei jau drīz pēc interviju sākuma radās šaubas par šajās intervijās gūto datu lietojamību.

Atšķirīga bija respondentu attieksme un aktivitātes līmenis attiecībā uz inovāciju projektiem, to rezultātiem un ieviešanu reālā dzīvē:

- 5 no aptaujātajiem pret inovāciju pielietošanu pauda neitrālu, pat vienaldzīgu attieksmi; zīmīgi ir tas, ka tieši šie respondenti paši nebija bijuši iesaistīti inovāciju tipa projektos.
- 4 no aptaujātajiem klasificējami kā "neaktīvi piekritēji", resp., kopumā pauda pozitīvu attieksmi, taču atzina, ka dažādu apstākļu dēļ (aizņemtība ražošanas darbā, intereses un zināšanu trūkums, negatīva pieredze sadarbībā ar zinātniskajām institūcijām, neticība Latvijas valstī notiekošajam) nav gatavi un ieinteresēti paši līdzdarboties inovāciju projektos vai izmantot inovāciju projektu sasniegumus savos ražošanas projektos.
- 3 respondenti atzina, ka ir pozitīvi noskaņoti pret inovācijām, tic to ilgtermiņa ietekmei uz tautsaimniecību un iespēju robežās savos ražošanas projektos izmanto kolēģu un pašu sasniegumus inovāciju sfērā. Tiesa, šīs grupas pārstāvji atzina, ka uz līdzdarbību inovāciju projektos viņus drīzāk mudina apkārtējā vide (kolēģi, kas jau nodarbojas ar pētniecību, papildus finansējuma pieejamība, citu projektu veiksmes stāsti) un personīgās ambīcijas apliecināties "augstākā līmenī", nekā patiesa zinātniska interese.
- Tikai 2 respondentus var uzskatīt par īstiem inovāciju "faniem", resp., viņi aktīvi seko līdzi savu pētījuma tēmu attīstībai pasaulē, interesējas par risinājumiem, tendencēm u.tml. Abi ir bijuši iesaistīti inovāciju projektos, sekmīgi tos izpildījuši un būtu gatavi iesaistītiem jaunos arī tad, ja tas nebūtu saistīts ar papildus finansējumu.

Interesanti ir, ka konkrētajā gadījumā neizdevās konstatēt nekādu sakarību starp respondenta formālās izglītības līmeni (zinātnisko grādu) un intereses pakāpi par inovācijām. Savukārt skaidri pamanāma bija korelācija starp iepriekšēju pieredzi pētnieciskos projektos un interesi par dalību jaunos inovāciju projektos.

Viedās tehnoloģijas pamatidejas, galveno jēdzienu un nostādņu raksturojums

Visi respondenti ar pieredzi informācijas tehnoloģiju jomā bija pazīstami ar viedo tehnoloģiju un tām radniecīgām idejām. 4 aptaujātie bija kaut kādā pakāpē bijuši iesaistīti konkrētu viedo tehnoloģiju realizāciju izstrādē. Līdz ar to šī aptaujas sadaļa konkrētajā segmentā papildus informāciju nesniedza.

Citādi bija ar respondentiem bez pieredzes informācijas tehnoloģiju jomā. Šiem dalībniekiem informācija par viedo tehnoloģiju (atgādinājumam – visi, paši to gan neapzinoties, ikdienā strādāja ar risinājumiem, kas aprīkoti ar viedām tehnoloģijām!) bija interesants jaunums. Šajā aptaujas sadaļā autore atmeta hipotēzi, ka no gala lietotājiem kā no tiešajiem viedo tehnoloģiju „patērētājiem" būtu iespējams iegūt interesantas atziņas. Tādēļ turpmākajā aptaujas norises atreferējumā šo aptaujas dalībnieku sniegtās atbildes vairs netika izmantotas.

Attieksme pret viedo tehnoloģiju ideju kopumā, intuitīva ticība/ neticība tām

Arī šajā sadaļā būtiski jaunu informāciju iegūt neizdevās, jo attieksme pret viedo tehnoloģiju ideju kopumā gandrīz pilnībā sakrita ar respondentu attieksmi pret inovācijām kā tādām. Gandrīz visi respondenti uzsvēra savu vēlēšanos (!) pilnībā ticēt viedo tehnoloģiju pielietošanas spēkam, taču gandrīz katrs varēja norādīt uz apsvērumiem, kas šai ticībai traucē. Biežāk minētie argumenti sasaucas ar nākošajās aptaujas sadaļās apskatītajām tēmām, tādēļ šeit netiek atkārtoti.

Diskusija par viedo tehnoloģiju pielietojamību dažādām projektu/ uzdevumu grupām

Šajā interviju sadaļā paustie viedokļi drīzāk klasificējami kā vingrinājums ar mērķi atrast ideālo pielietojuma sfēru. Visi respondenti gan bija vienisprātis, ka viedo tehnoloģiju kā programmatūras nefunkcionālo īpašību kopas pielietošanas aspekti nav tieši saistīti ar konkrētās programmatūras funkcionālajām iespējām vai ar biznesa sfēru, kurai programmatūra tiek izstrādāta. Drīzāk viedo tehnoloģiju pielietošanas iespējamību nosaka pašas programmatūras uzbūve un ārējie apstākļi.

Biežāk tika minētas šādas potenciālo pielietojumu jomas:

- darba plūsmu vai notikumu orientētas sistēmas,
- programmatūra, kas nodrošina kritiskus biznesa procesus,

- ģeogrāfiski sadalītas informācijas sistēmas ar lielu lietotāju skaitu,
- programmatūras, kam uzstādīta ļoti augstas pieejamības, slepenības, drošības prasības,
- produkti, kuru darbaspējas būtiski iespaido piegādātāja reputāciju,

Tehniskas/ organizatoriskas dabas apsvērumi par/ pret viedo tehnoloģiju izmantošanu

Intervijas daļā, kura attiecināma uz neekonomiskas dabas apsvērumiem, bija novērojamas ļoti atšķirīgas kvalitātes atbildes un dažāda ieinteresētības pakāpe.

Kopumā tendence tomēr liecināja – jo pozitīvāk respondents attiecas pret inovācijām kopumā, jo konkrētāki bija izmantotie argumenti. Kamēr aptaujājamie ar reālu inovāciju vai pat viedo tehnoloģiju implementācijas pieredzi dalījās interesantās un pat negaidītās atziņās, dažu respondentu atbildes bija neko neizsakošas.

Zemāk apkopotas izplatītākās un/ vai, autoresprāt, interesantākās atziņas.

- Nesaki klientam, ka tieši viņa projektā grasies ieviest jauninājumus! Turpretī tad, kad šie jauninājumi jau ieviesti, neaizmirsti ar tiem palepoties. Tikai retais interesējas par inovatīviem risinājumiem savos produktos – vairums koncentrējas uz sev nepieciešamās funkcionalitātes izstrādi, pārējam nepievēršot gandrīz nekādu uzmanību.
- Viedo tehnoloģiju komponenšu tehnisko realizāciju var uzticēt arī iesācējiem vai ne īpaši kvalificētiem darbiniekiem, taču svarīgi ir veikt nepārtrauktu izskaidrošanas un iedvesmošanas darbu visas komandas iekšienē. Pretējā gadījumā rezultāts pārvērtīsies par projekta vai izstrādes vadītāja un viena vadošā speciālista "neaizskaramu mākslas darbu", ko pārējie nesteigsies izmantot praktiskajās realizācijās. Nevienam neļauj izveidoto implementāciju vai idejas monopolizēt!
- Dalies sasniegumos ar kolēģiem no citām projektu grupām - ieteikumu veidā saņemtā atgriezeniskā saite un iegūtā reputācija atsvērs no projekta budžeta finansētās "nākotnes izstrādnes".
- Nepietiek tikai ar lielisku realizāciju - izveidotā risinājuma ieguvumi/ priekšrocības jāpadara saprotami gan sev, gan klientiem, gan dažkārt arī konkurentiem.
- Programmatūrai, kuru paredzēts aprīkot ar viedajām tehnoloģijām, tomēr jābūt nodrošinātai zināmai arhitektūras un uzbūves kvalitātei, pretējā gadījumā viedo

tehnoloģiju iekļaušana programmatūrā būs līdzvērtība šīs programmatūras pārrakstīšanai.

Ekonomiskie apsvērumi par/ pret viedo tehnoloģiju izmantošanu

Par galveno iemeslu viedo tehnoloģiju nepietiekamajai izplatībai vienbalsīgi tika minēts atbilstoša finansējuma trūkums.

No visiem aptaujātajiem speciālistiem tikai 2 (un tieši viņiem bija atbilstoša pozitīva pagātnes pieredze) uzskatīja, ka ilgtermiņa attīstības vārdā ir vērts veikt ieguldījumus viedo tehnoloģiju implementācijās, to apmaksājot no projektu vai uzņēmuma iekšējiem attīstības budžetiem. 3 respondenti uzskatīja, ka šāda veida investīcijas būtu "godīgi" jāsadala starp piegādātāju un pasūtītāju, un tūdaļ atzina, ka savā praksē īsti netic šādai varbūtībai. Pārējie izeju saskatīja ārēja finansējuma piesaistē, jo īpaši – Eiropas struktūrfondu līdzfinansējumā.

5.4.5 Pētījuma rezultāti

Pētījuma ietvaros savāktā informācija rāda diezgan viendabīgu viedokli – viedo tehnoloģiju ideja uzskatāma par mūsdienīgu un vajadzīgu, taču virkne subjektīvas un objektīvas dabas apstākļu traucē to plašākai izplatībai. Zemāk citētas plašāk izplatītās tēzes, kas reprezentē valdošo viedokli un lielā mērā sakrīt ar autores pieredzi, kas gūta viedo tehnoloģiju idejas attīstības ietvaros un inovāciju projektos kopumā.

- Viedo tehnoloģiju implementācijas pašreizējā to idejiskās attīstības stadijā ir pa spēkam tikai uzņēmumiem ar atbilstošas kvalifikācijas personālu un pietiekamu finansējumu.
- Viedo tehnoloģiju izmantošana ir lietderīga ilgtermiņa vai augstas reputācijas projektos. Ļoti reti viedo tehnoloģiju izmantošanas iniciators ir klients.
- Viedo tehnoloģiju sasniegumi ir ļoti atkarīgi no IT industrijas tendencēm, ražojumiem utt, tādēļ pastāv risks, ka izveidotās viedo tehnoloģiju komponentes "noveco" reizē ar programmproduktiem, kuru vajadzībām tās izstrādātas .
- Jāatsakās no ilūzijas, ka jebkādi tehniski risinājumi varētu atrisināt ("izglābt") objektīvi pastāvošās heterogēnās IT pasaules problēmas, tomēr viedo tehnoloģiju pielietošana noteiktos segmentos ar pietiekami augstu varbūtību var dot pienesumu, lai situāciju uzlabotu.

SECINĀJUMI

Apkopojot pētnieciskās darbības laikā iegūtās atziņas, reālos IT projektos gūto pieredzi un kvalitatīvā pētījuma rezultātus, var secināt:

- No viedo tehnoloģiju pielietošanas sagaidāmais efekts ir ievērojams. Ieguvums paredzams teju vai visos programmatūras dzīves cikla aspektos – jau sākot no programmatūras tehnisko īpašību uzlabojumiem līdz pat klientu apkalpošanas kvalitātes kāpumam.
- Lai arī pat tikai atsevišķu viedo tehnoloģiju īpašību pielietošana jau var nest būtisku atdevi gan materiālā plāksnē, gan imidža jautājumā, salīdzinoši augstās sākotnējās investīcijas, nedrošība ieguldījumu atguvē, neinformētība un piemērotas kvalifikācijas izstrādātāju trūkums IT tirgū ir būtiski šķēršļi viedo tehnoloģiju plašākai izplatībai.
- Viedo tehnoloģiju sekmīgas izmantošanas komerciālos projektos priekšnoteikumus var izpildīt tikai kvalificēts piegādātājs, kas ir spējīgs programmproduktu izstrādei pieiet inovatīvi, pat zinātniski. Izvērtējot viedo tehnoloģiju pielietošana lietderību, jāņem vērā ne tikai ekonomiskie aspekti vien, bet arī tādi grūti novērtējami faktori kā uzņēmuma reputācija, mārketinga kapacitāte, klientu atbalsta līmenis u.tml.
- Viedo tehnoloģiju iespējas reti tiek pieprasītas pēc pasūtītāju iniciatīvas. Pasūtītāji, lai arī atzinīgi novērtē viedo tehnoloģiju sniegtās iespējas, parasti nav gatavi par tām papildus maksāt. Riski, kurus uzņemas piegādātājs, iekļaujot programmatūrā viedās tehnoloģijas, attaisnojas ilgas pasūtītāja un piegādātāja sadarbības projektos vai standartizētu produktu gadījumos.
- Viedās tehnoloģijas ideja ir perspektīva, viedo tehnoloģiju implementācijas ir tehniski realizējamas un (pie noteiktiem nosacījumiem) izmantojamas komerciālos programmatūras izstrādes projektos.

NOBEIGUMS

Viedo tehnoloģiju ideja jau tagad ir sevi pierādījusi gan zinātniskās izpētes, gan praktisko pielietojumu jomā.

Zinātniskajā jomā paveiktais:

- vairāk kā desmit publikācijas starptautiski atzītos programminženierijas nozares izdevumos; autori: prof. J. Bičevskis, K. Rauhvargers, E.Diebelis, Z. Bičevska; īpašs uzsvars likts uz viedo tehnoloģiju konceptuālajām nostādnēm;
- dalība ESF finansētajā projektā „Jaunu tehnoloģiju izstrāde informācijas sistēmu izveidei un integrācijai", kas 2006. -2008. gadā realizēts Vienotā programmdokumenta 2. prioritātes „Uzņēmējdarbības un inovāciju veicināšana" 5. pasākuma „Atbalsts lietišķās zinātnes attīstībai valsts zinātniskajās institūcijās" 1. aktivitātes „Atbalsts lietišķajiem pētījumiem valsts zinātniskajās institūcijās" ietvaros;
- viens sekmīgi aizstāvēts maģistra darbs (V. Takeris);
- četri sekmīgi aizstāvēti bakalaura darbi (J. Ivanovs, K. Jekimova, A. Ermansons, V. Bartuševics).

Praktiskajā darbībā paveiktais :

- ERAF līdzfinansētā projekta „Gudrās programmatūras tehnoloģijas implementācija notikumu orientētās informācijas sistēmās" (realizēts 2006.-2008. gadā Valsts atbalsta programmas „Atbalsts jaunu produktu un tehnoloģiju attīstībai" apakšprogrammā „Atbalsts jaunu produktu un tehnoloģiju attīstībai") ietvaros izveidota un konkrētu klientu projektos aprobēta versiju automatizētās izplatīšanas komponente; iegūtie rezultāti tiek komerciāli izmantoti SIA „Datorikas institūts DIVI" produktos un klientu projektos;
- viedo tehnoloģiju principu iestrāde projektu pārvaldības risinājumā EiroSIS; iegūtie rezultāti tiek komerciāli izmantoti SIA „Datorikas institūts DIVI" projektos, kas tiek veikti Eiropas struktūrfondu pirmā un otrā līmeņa starpniekorganizāciju uzdevumā;
- numuru saglabāšanas datu bāzes izstrāde saskaņā ar labajiem viedo tehnoloģiju izveides principiem; rezultātus SIA „Datorikas institūts DIVI" O.Ozols, V.

Kuzmins, N. Blumbergs ar referātu „Testēšanas pieredze numuru saglabāšanas datu bāzes izstrādē" prezentējuši 7. Latvijas IT uzņēmumu konferencē „Testēšanas teorija un prakse" konferencē (2006);

- ERAF līdzfinansētā projekta „Programmatūras paštestēšanas tehnoloģija" (apstiprināts izpildei 2009.-2011. gadā Valsts atbalsta programmas „Uzņēmējdarbība un inovācijas" apakšprogrammā „Jaunu produktu un tehnoloģiju izstrāde") ietvaros izveidota un konkrētu klientu projektos aprobēta paštestēšanas komponente.

Kopumā viedo tehnoloģiju tematika vērtējama kā mūsdienīga, praktiski pielietojama, vienlaikus saglabājot pētniecībai raksturīgās abstrakcijas un vispārināšanas iespējas.

PATEICĪBAS

Lielu pateicību par atbalstu un saturisko pienesumu viedo tehnoloģiju idejas radīšanā un implementēšanā izsaku savam tēvam profesoram Jānim Bičevskim.

Kopā esam sagatavojuši un publicējuši piecas zinātniskas publikācijas, kā arī līdzdarbojušies nu jau četros Eiropas Savienības struktūrfondu līdzfinansētos pētniecības projektos.

Par uzmundrinājumu un organizatorisku atbalstu, kā arī vērtīgiem komentāriem visā doktorantūras studiju un promocijas darba izstrādes laikā lielu pateicību izsaku savam draugam profesoram Jurim Borzovam.

Liels paldies arī kolēģiem no SIA „Datorikas institūts DIVI", kuri tēmas attīstībā ienesa jaunas un neordinārās idejas, veica praktiskās izstrādnes, kā arī neskopojās ar pieredzes datiem, tai skaitā konkrētiem projektu rādītājiem.

Noslēgumā liels paldies manai ģimenei un draugiem par sapratni un atbalstu publikāciju un šī darba radīšanas laikā.

IZMANTOTĀ LITERATŪRA UN AVOTI

[1] **Donald G. Firesmith,** *Modern Requirements Specification*, Journal of Object Technology, 2003. [atsauce 28.09.2009].

Pieejams: http://www.jot.fm/issues/issue_2003_03/column6/

[2] http://en.wikipedia.org/wiki/Requirement [atsauce 13.03.2010]

[3] http://www.digital-ecosystems.org/ [atsauce 28.09.2009]

[4] **Bičevska Z., Bičevskis J.**, Smart Technologies in Software Life Cycle. In: Münch J., Abrahamsson P. (eds.): *Product-Focused Software Process Improvement*. 8th International Conference, PROFES 2007, Riga, Latvia, July 2-4, 2007, Lecture Notes in Computer Science, Vol. 4589, pp. 262–272,Springer, Heidelberg (2007)

[5] **Bičevska Z., Bičevskis J**., Applying of smart technologies in software development: Automated version updating. In: *Scientific Papers University of Latvia. Computer Science and Information Technologies*, Vol.733, pp. 24–37 (2008)

[6] **Bičevska Z., Bičevskis J**., Self-testing: A New Testing Approach. In: Haav, H.-M., Kalja, A. (eds.): *Proceedings of the Eighth International Baltic Conference on Databases and Information Systems (Baltic DB&IS'2008)*, Tallin, 2-5, June, 2008, pp. 179–189 (2008)

[7] **Bičevska Z., Bičevskis J**., Applying Self-testing: Advantages and Limitations. In: Hele-Mai Haav, Ahto Kalja(eds.): *Databases and Information Systems, Selected Papers from the Eighth International Baltic Conference*, IOS Press V. 187, pp. 192–202 (2009)

[8] **Bicevska Z.**, Applying Smart Technologies: Evaluation of Effectiveness, In: *Conference Proceedings of the 2nd International Multi-Conference on Engineering and Technological Innovation* (*IMETI 2009*), Orlando, Florida, USA, July 10-13, 2009 [atsauce 28.08.2009]

Pieejams: http://www.iiis.org/CDs2008/CD2009SCI/IMETI2009/PapersPdf/F788TV.pdf

[9] **Bicevska Z.**, Applying Smart Technologies: Evaluation of Effectiveness, In: *Conference Proceedings of the 4th IFIP TC2 Central and East European Conference on Software Engineering Techniques* (*CEE-SET 2009*), Krakow, Poland, Oktober 12-14, 2009

[10] http://www.research.ibm.com/autonomic/manifesto/ [atsauce 28.09.2009]

[11] http://en.wikipedia.org/wiki/Autonomic_Computing [atsauce 28.09.2009]

[12] **Giovanni Pacifici, Mike Spreitzer, Asser Tantawi, and Alaa Youssef,** Performance Management for Cluster Based Web Services, IBM TJ Watson Research Center, 2003 [atsauce 28.09.2009]

Pieejams: http://www.research.ibm.com/autonomic/research/papers/pacifici_TechReport.pdf

[13] **Xiaolong Jin and Jiming Liu**, From Individual Based Modeling to Autonomy Oriented Computation, In: Matthias Nickles, Michael Rovatsos, and Gerhard Weiss (editors), *Agents and Computational Autonomy: Potential, Risks, and Solutions*, pages 151–169, Lecture Notes in Computer Science, vol. 2969, Springer, Berlin, 2004. ISBN 978-3-540-22477-8.

[14] **Jose M. Vidal:** *Fundamentals of Multiagent Systems*, 2007 [atsauce 29.09.2009]

Pieejams: http://www.scribd.com/doc/2094479/Fundamentals-of-Multiagent-Systems

[15] http://www.research.ibm.com/autonomic/overview [atsauce 28.09.2009]

[16] **J. Kephart and D. Chess:** The Vision of Autonomic Computing, *Computer Magazine*, IEEE, 2003

[17] http://research.microsoft.com/en-us/research/ [atsauce 29.09.2009]

[18] http://research.microsoft.com/en-us/groups/adapt/ [atsauce 29.09.2009]

[19] **Anderson, Ross:** Trusted Computing' Frequently Asked Questions, 2003 [atsauce 28.09.2009]

Pieejams: http://www.cl.cam.ac.uk/~rja14/tcpa-faq.html

[20] Trust in Cyberspace, Computer Science and Telecommunications Board (CSTB), (1999) [atsauce 25.09.20009]

Pieejams: http://www.nap.edu/openbook.php?record_id=6161

[21] **Dolev Shlomi:** Self-Stabilisation, Massachusetts Institute of Technology, pp 195, 2000, ISBN 0-262-04178-2

[22] The Self-Managing, 'Unbreakable' Internet?, *ICT Result*s (09/30/09) [atsauce 20.03.2010]

Pieejams: http://cordis.europa.eu/ictresults/index.cfm?section=news&tpl=article&BrowsingType=Features&ID=90892

[23] **Naone Erica:** Software That Fixes Itself, *Technology Review* (10/29/09) [atsauce 20.03.2010] Pieejams: http://www.technologyreview.com/computing/23821/

[24] **Rauhvargers, K., Bicevskis, J**.: Environment Testing Enabled Software – a Step Towards Execution Context Awareness. In: Hele-Mai Haav, Ahto Kalja(eds.):

Databases and Information Systems, Selected Papers from the Eighth International Baltic Conference, IOS Press V. 187, pp. 169–179 (2009)

[25] **Thomas J. Bergin,** Computer-aided Software Engineering: Issues and Trends for the 1990s and Beyond. Idea Group Inc (IGI), 1993

[26] http//www.oracle.com/technology/products/designer/index.html [atsauce 25.08.2009]

[27] **A.Kalnins, J.Barzdins, A.Auzins, I.Etmane, A.Kalis, K.Podnieks, J.Tenteris, E.Vilums, A.Zarins**, Business Modeling Language GRAPES-BM and Related CASE Tools, Institute of Mathematics and Computer Science, University of Latvia & Riga Information Technology Institute (RITI) In: *Proceedings of the Second International Baltic Workshop "Databases and Information Systems"*, Tallinn, June 12-14, 1996, Vol.2, pp.3-16

[28] Object Management Group (OMG) Pieejams: http://www.omg.org [atsauce 25.08.2009]

[29] **Booch G., Jackobson I., Rumbaugh I**., *The Unified Modeling Language. Reference Manual*, 1999.

[30] http://www.omg.org/mda/committed-products.htm [atsauce 16.08.2009]

[31] http://www.omg.org/mda/products_success.htm [atsauce 16.08.2009]

[32] http://blogs.msdn.com/alan_cameron_wills/archive/2004/11/11/255831.aspx [atsauce 16.08.2009]

[33] **J. Ceriņa-Bērziņa, J. Bičevskis, Ģ. Karnītis,** Information systems development based on visual Domain Specific Language BiLingva. *Accepted for publication in the 4th IFIP TC2 Central and East European Conference on Software Engineering Techniques (CEE-SET 2009)*, Krakow, Poland, Oktober 12-14, 2009

[34] http://en.wikipedia.org/wiki/Domain_Specific_Language [atsauce 16.08.2009]

[35] **Roger S.Pressman**, Ph.D., *Software Engineering. A Practitioner's Approach*. 6th edition, 2004, ISBN: 007301933X

[36] http://yourdon.com/strucanalysis/wiki/index.php?title=Chapter_13 [atsauce 16.08.2009]

[37] **Kalnins A., Barzdins J., Celms E.,** Model Transformation Language MOLA. Lecture Notes in Computer Science, Springer, v. 3599, 2005, *Model-Driven Architecture, European MDA Workshop*, Revised Selected Papers, pp. 62-76.

[38] **Kalnins A., Barzdins J., Celms E.,** Efficiency Problems in MOLA Implementation. *19th International Conference OOPSLA'2004*, Vancouver, Canada, October 2004.

[39] **Beizer, B.** *Black-Box Testing Techniques for Functional Testing of Software and Systems*. John Wiley & Sons, 1995.

[40] Manifesto for Agile Software Development [atsauce 29.09.2009] Pieejams: http://agilemanifesto.org/

[41] **Rauhvargers K., Bicevskis, J.,** Automating the Software Environment Testing Process. In: *Proceedings of the 8th International Baltic Conference on Databases and Information Systems (Baltic DB&IS 2008*), June 2-5, Tallin, Estonia, pp. 155-166.

[42] **Бичевский Я., Борзов Ю**., Приоритеты в отладке больших программных систем, *Программирование*, №3, 1982

[43] **Chengying M., Yansheng L., Jinlong Z.,** Regression testing for component-based software via built-in test design. In: *Proceedings of the ACM symposium on Applied computing,* March 11 – 15, 2007, Seoul, Korea, pp. 1416 – 1421.

[44] **J. Bicevskis, J.Borzovs, U.Straujums, A.Zarins, E.F.Miller**, SMOTL - A System to Construct Samples for Data Processing Program Debugging, In: *Technical Note RN-415*, Software Research Associates, 1978

[45] **K. Rauhvargers**, *Programmatūras izpildes vides testēšana*, Promocijas darbs, 2009

[46] **Rauhvargers K., Bicevskis J.**, Towards a semantic execution environment testing model, In: *Scientific Papers University of Latvia. Computer Science and Information Technologies*, 2008, Vol.733, pp. 24–37

[47] **Beydeda, S**.: Research in Testing COTS Components - Built-in Testing Approaches. In: *ACS/IEEE 2005 International Conference on Computer Systems and Applications*. Bonn, Germany, 2005

[48] **Loshin D**., *Enterprise Knowledge Management: The Data Quality Approach*. Morgan Kaufman. 2001

[49] **L.I.Pipino, Y.W.Lee, R.Y.Wang**, Data quality assessment. *Communications of the ACM,* vol.45. pp 211-218, 2002

[50] **A.T.Berztiss**, Models for data quality management. *Proceedings of the 16th Conference on Advanced Information Systems Engineering (CaiSE'04)*, 2004

[51] http://www.lursoft.lv [atsauce 27.03.2007]

[52] **Alonso, Eduardo; Kudenko, Daniel; Kazakov, Dimitar** (Eds.), *Adaptation and Multi-Agent Learning Series*: Lecture Notes in Computer Science Vol. 2636, 2003, XIV, 323 p., ISBN: 978-3-540-40068-4

[53] http://en.wikipedia.org/wiki/Load_balancing_(computing) [atsauce 29.09.2009]

[54] http://www.thefreedictionary.com/effectiveness [atsauce 25.08.2009]

[55] www.scoea.bc.ca/glossary2001.htm [atsauce 25.08.2009]

[56] http://en.wikipedia.org/wiki/Evaluation [atsauce 25.08.2009]

[57] http://www.absoluteastronomy.com/topics/Quantitative [atsauce 25.08.2009]

[58] **Sullivan A., Sheffrin S. M**., *Economics: Principles in action*, Pearson Prentice Hall, 2003

[59] http://en.wikipedia.org/wiki/Rate_of_return [atsauce 25.08.2009]

[60] **Boehm B. et al**., *Software cost estimation with COCOMO II* (with CD-ROM). Englewood Cliffs, NJ: Prentice-Hall (2000)

[61] http://en.wikipedia.org/wiki/Discounted_Cash_Flow [atsauce 26.08.2009]

[62] **Schach S. R**., *Object-oriented and classical software engineering*. McGraw-Hill Professional (2004)

[63] **Macho-Stadler I., Perez-Castrillo J**., *An Introduction to the Economics of Information. Incentives and Contract*. Oxford University Press, 2001, 2nd ed.

[64] Intervija ar Mg. Soc.sc. Lieni Odīti, 27.06.2009

[65] **Liene Odīte**, *Skolu jaunatne un Internets: informācijas sabiedrības veidošanās Latvijā*, Maģistra darbs, LU, 1999

[66] http://encyclopedia2.thefreedictionary.com/Fenomenology [atsauce 28.09.2009]

[67] **Thewlis, J.** (Ed.) *Concise Dictionary of Physics*. Oxford: Pergamon Press, p. 248, 1973

[68] http://en.wikipedia.org/wiki/Phenomenology_(science)#Phenomenology_in_social_statistics [atsauce 28.09.2009]

[69] **Cartwright, Nancy**, intro., *How the Laws of Physics Lie*, Oxford U., 1984

[70] **Nicolaas J. Molenaar**, Non-Experimental Research on the Effects on the Wording of Questions in Survey Interviews,. *Quality & Quantity*, 16, no 2 (1982) 69-90

[71] **Stanley Payne**, *The Art of Asking Questions*. Princeton: Pinceton University Press, 1980

[72] **Denzin Norman K., Lincoln Yvonna S**. (Eds.) (2005), *The Sage Handbook of Qualitative Research* (3rd ed.). Thousand Oaks, CA: Sage. ISBN 0-7619-2757-3 ^ Taylor, 1998

[73] http://en.wikipedia.org/wiki/Qualitative_research [atsauce 28.09.2009]

[74] **Loseke Donileen R., Cahil Spencer E**., Publishing qualitative manuscripts: Lessons learned. In C. Seale, G. Gobo, J. F. Gubrium, & D. Silverman (Eds.), *Qualitative Research Practice: Concise Paperback Edition*, pp. 491-506. London, Sage, 2007. ISBN 978-1-7619-4776-9

[75] **Crawford I.M.,** *Food and agriculture organization of the united nations*, Rome, 1997, ISBN 92-851-1005-3

[76] **Marshall Catherine, Rossman Gretchen B.,** *Designing Qualitative Research*. Thousand Oaks, CA: Sage. 1998. ISBN 0-7619-1340-8

[77] **Lindlof T. R., Taylor B. C.,** *Qualitative communication research methods: Second edition*. Thousand Oaks, CA: Sage Publications, Inc, 2002. ISBN 0-7619-2493-0

[78] **Lincoln Y., Guba EG,** *Naturalist Inquiry*, Sage Publications, Newbury Park, CA, 1985