VILNIAUS UNIVERSITETAS

MATEMATIKOS IR INFORMATIKOS FAKULTETAS

PROGRAMŲ SISTEMŲ KATEDRA

Senos verslui skirtos programinės įrangos priežiūra

Legacy Business Software Maintenance

Magistro baigiamasis darbas

Atliko: Adomas Bočkus (parašas)

Darbo vadovas: Doc. dr. Valdas Undzėnas (parašas)

Recenzentas: Doc. dr. Saulius Ragaišis (parašas)

Vilnius – 2010

SANTRAUKA

Šiame darbe yra išanalizuotos problemos, susijusios su senomis verslo programų sistemomis ir pasiūlyti būdai joms spręsti. Pateikiami modeliai ir standartai, kuriais remiantis atliekama tokių verslo sistemų priežiūra ir modernizacija. Taip pat pateikiama schema, įgalinanti efektyviau ir paprasčiau prižiūrėti sistemas, atlikti jose pakeitimus, transformuoti bei migruoti seną programinę įrangą į naują aplinką.

Raktiniai žodžiai: programų sistemų priežiūra, modeliai, senos programų sistemos, BizTalk, migracija, transformacija

SUMMARY

In this paper, problems, related with legacy business software maintenance, are discussed in details and there are suggested solutions to solve them. There are some models and standards which are used to maintain the software systems. Also, maintenance techniques are discussed, which simplify maintenance processes and make them more effective. There is suggested a schema fits to maintain, renew, migrate and monitor business legacy software systems**.**

Keywords: software maintenance, models, standards, legacy systems, BizTalk, migration, transformation

Turinys

Įvadas

Dažnas žmogus, paklaustas, kas yra programų sistema ar programinė įranga (toliau - PĮ), kokias funkcijas ji atlieka, kam ji reikalinga, arba kas yra PĮ priežiūra, geriausiu atveju atsakys tik bendrais bruožais, o dar blogiau, visiškai nieko apie tai nebus girdėjęs. Vaizdžiai kalbant apie PĮ, galėtume pasakyti, kad PĮ yra „širdis", be kurios organizacijos, įstaigos ar verslas praktiškai nefunkcionuotų. Šiandieniniame pasaulyje, vis daugiau procesų yra „nukraunama" nuo žmogaus rankų ir „uždedama" ant PĮ. Tai visiškai logiškas ir pagrįstas žingsnis, nes taip sparčiai besivystant naujovėms, augant industrijai, kylant verslo lygiui, žmogui tampa sunku susigaudyti, kas kur ir kaip vyksta, sunku aprėpti ir operuoti informacija, nepadaryti klaidos skaičiuojant, priimant sprendimus, atliekant kasdienines darbo veiklas. Dažniausiai tokiais atvejais į pagalbą „kviečiami" kompiuteriai ir PĮ.

Tačiau PĮ sensta, tampa nebereikalinga, nebenaudinga, trukdančia ar netgi lėtinančia jai patikėtos veiklos pažangą. Tokie objektai, kaip pastatai, tiltai ar keliai, dažniausiai naudojami ilgą laiką, nekeičiant jų pradinės naudojimo paskirties, tuo tarpu PĮ naudojimo paskirtis nuolat keičiasi ar plečiasi. PĮ naudojimo laikas retai kada būna didesnis kaip penkeri metai (moraliai pasensta) [GT03], kai, pavyzdžiui, pastatų, kelių naudojimo laikas gali būti daug dešimtmečių.

Užsakovas, pirkdamas PĮ, dažniausiai iki galo nežino, kokias konkrečiai funkcijas turi atlikti PĮ bei kokias veiklas ir procesus bus galima perduoti įsigyjamai PĮ. Tik pradėjus naudoti PĮ pastebima, kad PĮ trūksta vienokio ar kitokio funkcionalumo. Tačiau kas PĮ suteiks naujas funkcijas ir galimybes? PĮ dažniausiai tobulina tiekėjas/kūrėjas, bet, savaime aišku, už papildomą mokestį, kuris dažnu atvejų būna netgi didesnis už pačios PĮ įsigijimo kainą. Tokie PĮ tvarkymai užsakovui ir tiekėjui nėra malonus dalykas, bet, kaip rodo praktika, yra neišvengiami.

Senai PĮ egzistuoja daugybė apibrėžimų. Vienas iš jų seną PĮ apibūdina kaip PĮ, sukurtą naudojant senas technologijas, veikiančią senoje aparatinėje įrangoje, tokioje kaip didieji kompiuteriai, ir valdanti organizacijos pagrindinius verslo procesus. Be abejonės, tokia PĮ įranga organizacijai yra naudinga ir vertinga. Tam, kad organizacija galėtų konkuruoti, ji turi investuoti didelius pinigus į PĮ, jos priežiūrą, tobulinimą. Tačiau, po PĮ ilgo naudojimo laiko pastebima, kad investicijos į PĮ „virsta" į išlaidas, kurios iš esmės susideda iš visų žmonių, kurie dalyvauja kuriant, tobulinant, prižiūrint PĮ, atlyginimų.

Kadangi verslo pasaulis nuolat keičiasi, todėl organizacijos, norėdamos išsilaikyti rinkoje, privalo keisti, optimizuoti savo verslo procesus, pritaikyti juos prie pasikeitusios aplinkos. Kiekvienas verslo proceso pakeitimas savo ruožtu iššaukia ir naudojamos PĮ pakeitimus. Taigi, sena PĮ pradeda augti, darosi sudėtingesnė. Ant senos PĮ architektūros pradedami krauti nauji verslo

proceso sprendimai, PĮ tampa paini, dažniausiai nedokumentuota, prarandama kontrolė, nebesekamas PĮ dydis, vertė. Dėl darbuotojų kaitos ir prastos programinio kodo dokumentacijos, sena PĮ tampa sunkiai suprantama. Gali atsitikti netgi taip, kad nebus įmanoma atlikti jokio pakeitimo esamoje PĮ!

Organizacijose senai PĮ pradedama skirti vis mažiau dėmesio, lyginant su kitais organizacijos projektais. Organizacijos nenoriai skiria darbuotojus PĮ priežiūros darbų atlikimui. Tačiau, laikyti be priežiūros didelę vertę turinčią seną PĮ yra rizika, nes vieną dieną ši PĮ gali nustoti funkcionuoti, todėl organizacija turi pradėti skirti vis daugiau dėmesio senos PĮ priežiūrai. Buvo suprasta, kad organizacija, norėdama išsilaikyti rinkoje ir tęsti sėkmingą savo veiklą, turi savo dėmesį sutelkti į naudojamos PĮ efektyvumo ir funkcionalumo didinimą.

PĮ pakeitimus įtakoja daug veiksnių: tiekėjų kaita, kvalifikuotų darbuotojų trūkumas, technologijų ir teisinių aspektų kaita, veiksniai, kurie mažina organizacijos veiklumą, augina išlaidas, verčia organizaciją jaustis nestabilia rinkoje.

Prižiūrint PĮ iškyla daugybė klausimų, kaip efektyviai ir su mažiausiomis sąnaudomis atlikti PĮ atnaujinimo darbus? Žinoma, nėra vienintelio ir universalaus būdo, kuris tiktų visoms PĮ prižiūrėti, nes egzistuoja daugybė skirtingų PĮ, kurios turi būti skirtingai prižiūrimos**.**

Nagrinėjant verslo PĮ priežiūrą, kaip bebūtų keista, nėra pakankamai aišku, kaip reikėtų tai atlikti vadovaujantis standartais (IEEE-1219, ISO-12207). Nėra aiškiai aprašyta, kokie žingsniai turėtų būti atliekami atnaujinant verslo PĮ. Žinoma, skirtingai verslo PĮ yra keliami ir skirtingi priežiūros reikalavimai. Galbūt vienai PĮ prireiks mažai pastangų, kad ji atitiktų naujus reikalavimus, o kitą PĮ reikės pakeisti iš esmės (galbūt visai išmesti ir kurti naują).

Šiandien PĮ naudojama praktiškai visose gyvenimo srityse. Ji veikia ne tik bendrosios paskirties kompiuteriuose, kur sprendžiami pačio įvairiausio pobūdžio uždaviniai, bet ir daugelyje kitų gaminių, įskaitant šiuolaikinius automobilius, mobiliuosius telefonus, fotoaparatus, televizorius, moderniąją medicinos techniką, ryšių ir transporto sistemas, atomines elektrines, lėktuvus, raketas, liftus ir klausos aparatus. PĮ gamybos apimtys auga eksponentiškai. Trumpai tariant, šiandien nėra kitos veiklos srities, kuri būtų tokia svarbi pasaulio ekonomikai.

Plintant PĮ įvairiose šakose, jos buvo efektyviai panaudotos ir verslo procesuose. Netrukus buvo pastebėta, suprasta ir eksperimentiškai įrodyta, kad automatizavus verslo procesus, sumažėja teikiamų paslaugų ar prekių savikaina, greičiau ir efektyviau aptarnaujami klientai, suteikiamos paslaugos.

Vystantis pasaulio ekonomikai, keičiasi ir verslo poreikiai: atsiranda nauji kliento reikalavimai, prireikia naujų darbo metodų versle. Kitaip tariant, verslas adaptuojasi, prisitaiko prie nuolat

besikeičiančios aplinkos. Tam, kad verslas efektyviai funkcionuotų, atitinkamai turi turi būti keičiama ir naudojama PĮ.

Sena (liktinė) PĮ – tai PĮ, kuri yra gyvybiškai svarbi organizacijai ir kurią reikia prižiūrėti. Dažniausiai tokia PĮ būna sukonstruota prieš daugelį metų, naudojant technologijas, kurios šiuo metu jau yra pasenusios.

Gyvenant 21 amžiuje, yra verslo sričių, kurios vis dar naudoja seną PĮ, kuri šiandien dar veikia, tačiau ilgainiui gali būti jau nebetinkama naudojimui, nebeatitinkanti organizacijos poreikių, trukdanti verslo plėtrai**.**

Tyrimo objektas. Tyrimo objektas yra sena versle naudojama PĮ, jos priežiūra, būdai, kurie padeda pasiekti efektyvų senos PĮ palaikymą ir migraciją.

Dėl besikeičiančių verslo poreikių, turi keistis ir naudojama PĮ. Jei organizacija naudoja seną PĮ, iškyla daug klausimų, kaip pritaikyti jau egzistuojančią PĮ prie pakitusios naudojimo aplinkos. Vieni iš aktualiausių klausimų yra, kokių veiksmų turi ir privalo imtis PĮ prižiūrinti organizacija.

Darbo tikslai ir uždaviniai. Magistro darbo tikslas – pasiūlyti sprendimą kaip efektyviai prižiūrėti, transformuoti ir migruoti seną verslo PĮ, atstovaujant PĮ prižiūrinčioms kompanijoms; kaip, atsižvelgiant į senos PĮ reikšmę, svarbą ir struktūrą, atnaujinti PĮ komponentus.

Pagal ISO/IEC 14764 standartą, PĮ priežiūros veiklos yra:

- priežiūros proceso įgyvendinimas;
- PĮ problemų ir modifikacijos analizė;
- modifikavimas;
- peržiūra/priėmimas;
- migracija/transformacija;
- PĮ atsisakymas (naudojimo nutraukimas).

Magistro darbe koncentruojamasi ties PĮ migracijos/transformacijos užduotimi.

1. PĮ priežiūros apžvalga

1.1. PĮ priežiūros apibrėžimas

Dauguma šaltinių PĮ priežiūrą apibūdina kaip labai platų veiklų arsenalą, skirtą PĮ po jos pridavimo ir funkcionavimo pagal kliento poreikius [GT03, Pig97, TRB95]. Šios veiklos apima PĮ klaidų taisymą, efektyvumo kėlimą, PĮ galimybių pridėjimą ir naikinimą, PĮ pritaikymą prie naujų reikalavimų bei naudojimo aplinkos, naudojimo patogumą ir kitas PĮ kokybę atspindinčias komponentes. IEEE-1219 standartas [IEE90] PĮ priežiūrą apibūdina taip: PĮ priežiūra – tai procesas, kuris prasideda nuo PĮ pristatymo užsakovui ir kurio metu programinė įranga ar jos dalys yra taisomos tam, kad neliktų PĮ klaidų, padidintų našumą ar pritaikytų PĮ prie pakitusios aplinkos. Šiame apibrėžime atsispindi bendros PĮ priežiūrai reikalingos veiklos: viskas pradedama nuo tada, kai PĮ pasiekia klientą ir apima visas veiklas, kurios reikalingos tam, kad PĮ funkcionuotų pagal vartotojo poreikius.

Kai kurie autoriai nesutinka, kad PĮ priežiūra turi prasidėti tik tuomet, kai ji pristatoma užsakovui. Jie teigia, jog PĮ priežiūra turi prasidėti dar iki pradedant eksploatuoti ją. Schneidewind [Sch87] mano, jog požiūris į PĮ priežiūrą, kaip į veiklą sekančią po atidavimo klientui, „verčia" ją sunkiu darbu. Osborne ir Chikofsky [OC90] teigia, kad PĮ priežiūrą yra svarbu įkomponuoti į patį PĮ gyvavimo ciklą. Pigosky [Pig97] siūlo PĮ priežiūrą pradėti jau tuomet, kai pradedama kurti pati PĮ. Pagal jį, PĮ priežiūra – tai visuma veiklų, kurios reikalingos tam, kad PĮ tolimesnis tobulinimas būtų kuo pigiau atliekamas. Vienos veiklos yra atliekamos prieš PĮ atidavimą užsakovui, kitos po. Veiklas prieš PĮ atidavimą užsakovui sudaro planavimas veiklų, kurias reikės atlikti po pristatymo, palaikymas ir strategijų įvardinimas. Veiklas po PĮ pristatymo užsakovui sudaro PĮ modifikavimas, darbuotojų mokymai.

1.2. PĮ priežiūros modeliai

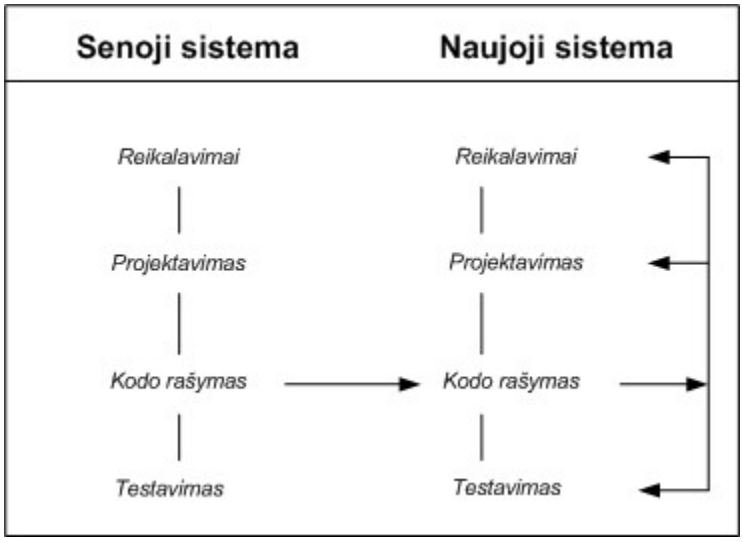
1.2.1. Greitojo taisymo modelis

Tipinė situacija, atliekant PĮ priežiūrą, yra pirmiausia dirbti su programos kodu, o tada atlikti atitinkamus įrašus ir modifikacijas dokumentacijoje. Tokiais veiksmais yra paremtas *Greito taisymo modelis (angl. quick-fix)* (žiūr. 1 pav.) [GT03]. Pakeitimai yra įgyvendinami pereinant nuo senos į naują PĮ versiją. Idealiu atveju, papildomai, kai tik yra pakeičiamas kodas, reikalavimų, projektavimo specifikacijos ar kiti su pakeitimu susiję dokumentai turi būti taip pat atnaujinami.

Dėl šio modelio lankstumo ir paprastumo, vartotojai tikisi, jog atlikti pakeitimą bus greita ir pigu. Modifikacijos labai dažnai atliekamos tiesiog vietoje, be atitinkamo plano, projektavimo,

poveikio analizės ir regresinių testų. Dėl laiko stokos ir finansinių resursų trūkumo neatnaujinami dokumentai galiausiai visai praranda vertę.

Toks modelis galėtų tikti organizacijoms, kurios turi mažai darbuotojų ir mažą biudžetą. Manoma, jog geriausiai šis modelis pasiteisintų, jei PĮ būtų prižiūrima vieno žmogaus: kadangi žmogus žino, ką ir kur pataisė, nereikia gaišti papildomo laiko prie dokumentavimo, planavimo ar projektavimo darbų. Tokiu atveju – tai pigus ir efektyvus sprendimas. Naudoti Greitojo taisymo modelį priverčia dažniausiai laiko stoka.



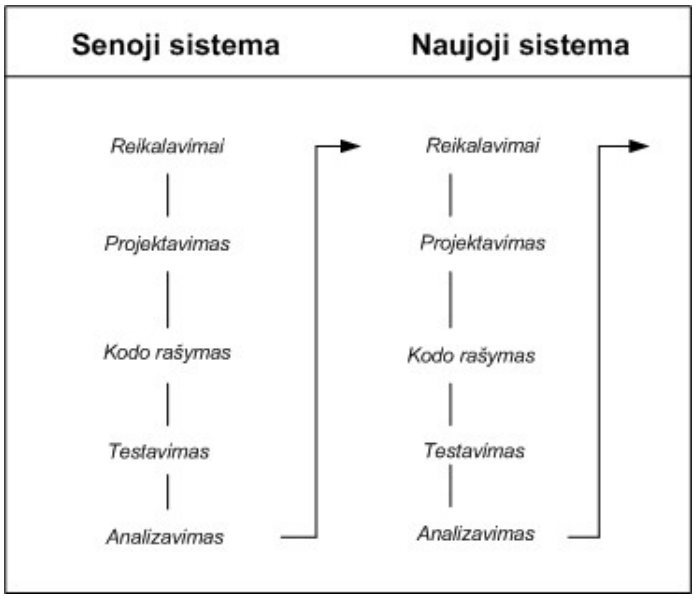
1 pav. Greitojo taisymo modelis [GT03]

1.2.2. Ciklinis tobulinimo modelis

Evoliucinis PĮ gyvavimo ciklo modelis siūlo alternatyvą Greito taisymo priežiūros modeliui [GT03]. Šis modelis pagrįstas idėja, kad PĮ reikalavimai vienu kartu nėra žinomi ir negali būti surinkti tam, kad pilnai būtų galima suprasti PĮ. Atitinkamai, PĮ turi būti plėtojama kiekvieną kartą iš naujo, surenkant reikalavimus bei pataisant esamas klaidas PĮ pagal vartotojo atsiliepimus. Tokio modelio pavyzdys galėtų būti ciklinis tobulinimo modelis, kuris PĮ atnaujinimus siūlo pradėti nuo paprastų reikalavimų įgyvendinimo ir baigti pilnai funkcionuojančia PĮ.

Ciklinis tobulinimo modelis (ang. Iterative enchancement model) (žiūr. 2 pav.) taip pat paaiškina ir PĮ priežiūros esmę. Nauja PĮ versija pradedama iš pradžių, t.y. iš naujo atliekama reikalavimų analizė, projektavimas, kodo rašymas, testavimas ir analizavimas, kurių metu pasiekiama aukštesnė PĮ kokybė. Trumpiau tariant, kiekvieną kartą PĮ yra tobulinama atsižvelgiant į senos PĮ versijos analizės rezultatus.

Visaggio [Vis99] palygino Greitojo taisymo ir Ciklinio tobulinimo modelius ir nustatė, jog organizacijose, kuriose naudojamas greitasis taisymo modelį, PĮ priežiūros lygis yra prastesnis nei organizacijose kuriose naudojamas ciklinis tobulinimo modelis. Taip pat eksperimentais įrodyta, kad organizacijos, naudojančios ciklinio tobulinimo modelį, reikalingus PĮ pakeitimus atlieka žymiai greičiau nei tos, kurios naudoja greitojo taisymo modelį.

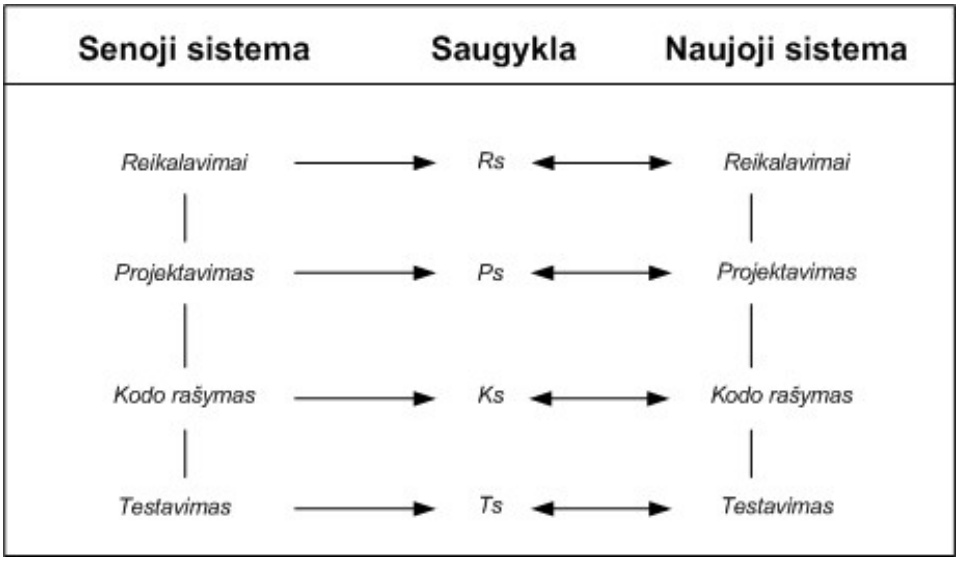


2 pav. Ciklinis tobulinimo modelis

1.2.3. Pakartotinio panaudojimo modelis

Basili pasiūlė modelį [Bas90], kuris paremtas pakartotinio panaudojimo principu, t.y. panaudoti jau sukurtos PĮ komponentus, testus bei kitą reikalingą medžiagą atnaujintos PĮ plėtojimui. Modelio autorius PĮ priežiūrą mato kaip tam tikrą pakartotinio panaudojimo atvejį PĮ plėtoti (žiūr. 3 pav.).

Šio modelio pradinė veikla - naujos PĮ reikalavimų analizė ir projektavimas, panaudojant senos PĮ atitinkamus reikalavimus, projektavimą, kodą ir testus.. Taip pat yra skatinamas kuo didesnis skaičius komponentų, kuriuos būtų galima panaudoti ateityje. Tokiu būdu sutaupomas PĮ kūrimo laikas, sumažėja resursų išlaidos, padidėja efektyvumas, nes komponentų veikimas jau būna patikrintas realiai veikiančioje PĮ.



3 pav. Pakartotinio panaudojimo modelis

1.3. PĮ priežiūros procesai

Aptarti keli autorių pasiūlyti PĮ priežiūros modeliai (Greitojo taisymo, Ciklinis, Pakartotinio pnaudojimo) susistemina PĮ priežiūrą į tarpusavyje susijusių veiklų seką ir apibrėžia eigą, pagal kurią šios veiklos turi būti atliekamos. Nors skirtingų modelių autoriai skirtingai įvardiją PĮ priežiūros stadijas, tačiau visi sutinka, kad yra veiklos, kurios sudaro PĮ priežiūros branduolį. Šios veiklos pagerina ir paspartina pačios PĮ supratimą, pasiūlyto pakeitimo poveikio analizę.

IEEE-1219 ir ISO-12207 standartai apibūdina priežiūros procesą kaip paskutinę dalį PĮ gyvavimo cikle [IEE98, ISO95].

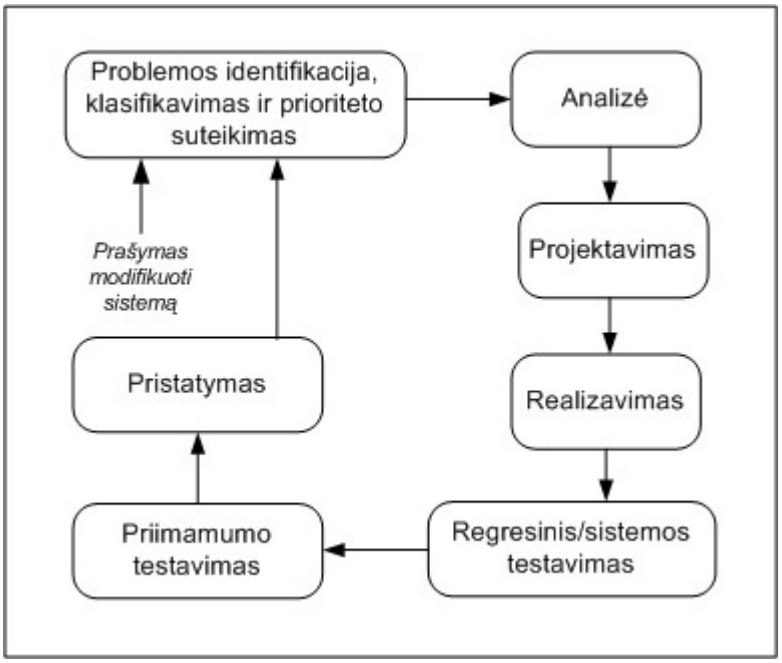
1.3.1. IEEE-1219 standartas

IEEE-1219 standartas [IEE98] PĮ priežiūrą skirsto į septynis žingsnius (žiūr. 4 pav.). Tam, kad būtų įvardinti žingniai ir jų eiliškumas, kiekvienam žingniui IEEE standartas nurodo įeigą ir išeigą, bei aprašo metrikas. IEEE-1219 priežiūros fazės:

- **problemos identifikacija, klasifikavimas ir prioriteto suteikimas.** Tai yra fazė, kurios metu yra svarstomas vartotojo prašymas modifikuoti PĮ. Taip pat programuotojai ar vadovai iškilusiai problemai priskiria prioritetą ir unikalų numerį. Į šią fazę yra įtraukiamos ir tokios veiklos, kurios padeda apsispręsti, ar prašymą modifikuoti (PĮ) atmesti, ar priimti, ir prašymą modifikuoti įtraukti į būsimų darbų planą;
- **projektavimas.** Pasiūlyta modifikacija šioje fazėje yra projektuojama. Šiam reikalui naudojama PĮ projekto dokumentacija, duomenų bazės. Šios fazės veiklas sudaro PĮ

modulių poveikio identifikavimas, modifikuojamo komponento dokumentavimas, naujų testų variantų sudarymas naujajai PĮ ir regresinių testų identifikavimas;

- **realizavimas.** Šią fazę sudaro kodavimas, atskirų dalių testavimas, naujo kodo integracija į PĮ, rizikos analizė ir peržiūros;
- **regresinis/sistemos testavimas.** Šioje fazėje yra testuojama visa PĮ tam, kad būtų įsitikinta naujojo modifikavimo teisingumu ir senų funkcijų išlaikymu. Taip pat yra ieškoma, ar nėra nenumatytų klaidų PĮ. Šioje fazėje yra užtikrinama, kad galima atlikti priėmimo testus;
- **priimamumo testavimas.** Šia faze yra siekiama ištestuoti pilnai PĮ įtraukiant tiek PĮ naudotojus, užsakovus ar trečiuosius asmenis (nurodytus užsakovo). Priimamumo testavimas apima funkcinius testus, regresinius testus.
- **pristatymas.** Tai tokia fazė, kurios metu modifikuota PĮ yra išleidžiama ir yra pasiruošusi dirbti. Šią fazę sudaro naudotojų informavimas, instaliavimo atlikimas bei apmokymai, senos versijos archyvavimas bei atsarginės kopijos išsaugojimas.



4 pav. IEEE-1219 standarto priežiūros procesas

1.4. Atvirkščioji inžinerija

Atvirkščioji inžinerija yra apibūdinama kaip procesas [WIKI2], kurio metu yra analizuojama PĮ tam, kad būtų išsiaiškinta, kokie komponentai ją sudaro, kaip jie tarpusavyje siejasi. Tai toks būdas, kurio tikslas yra išsiaiškinti, kokiais principais ir kaip PĮ veikia, nagrinėjant PĮ struktūrą,

funkcionavimą bei rezultatus. Šiuo būdu praktiškai kartojamas krioklio modelis iš galo: pradedama nuo rezultatų ir baigiama projektavimu bei reikalavimais.

Atvirkščioji inžinerija yra taikoma PĮ priežiūroje, tačiau atliekant tokį procesą taip pat galime atkurti prarastą informaciją, pažiūrėti į PĮ iš aukštesnio (abstrakcinio) lygmens, aptikti šalutinius PĮ poveikius, palengvinti pakartotinį panaudojamumą. IEEE-1219 [IEE98] standartas atvirkščiąją inžineriją siūlo naudoti kaip technologiją, padedančią suprasti PĮ turint tik programinį kodą. Taikant atvirkščiąją inžineriją galime aptiki svarbius komponentus, kuriuos galima panaudoti ateityje, atskleisti PĮ architektūrą, sumažinti, perkelti, išversti senos PĮ kodus. Atvirkščiosios inžinerijos principai taip pat sėkmingai yra taikomi verslo procesuose.

Atvirkščiajai inžinerijai kaip procesui yra sunku suteikti vienareikšmį apibrėžimą, nes jis nuolat kinta, greitai taikomas įvairiose srityse. Tradiciškai, atvirkščioji inžinerija yra dviejų žingsnių procesas: informacijos išgavimas ir išskyrimas.

IEEE-1219 standartas [IEE98] atvirkščiąją inžineriją rekomenduoja atlikti žingsniais: kodo analizė ir jo skaidymas į dalis, semantiškai apibrėžti šias dalis ir sukurti funkcines dalis, apibūdinti ryšius tarp tokių dalių, sudaryti šių dalių tarpusavio schemą, apibūdinti PĮ, sukurti PĮ architektūros schemą. Pirmieji trys žingsniai koncentruojasi ties atskiromis, mažomis PĮ dalimis, likusieji žingsniai skirti susidaryti bendram visos PĮ vaizdui.

1.5. Re-inžinerija

Atvirkščiaja inžinerija siekiama išsiaiškinti kaip ir kokiais principais veikia PĮ, o re-inžinerija – tai PĮ pertvarkymai, kurių metu PĮ pritaikoma prie naujų reikalavimų [WIKI3]. PĮ priežiūroje re-inžinerija naudojama tam, kad geriau būtų suprasta ir prižiūrima PĮ. Re-inžinerija – tai veikla, kuri padeda geriau suprasti PĮ ir/arba paruošia ar pagerina PĮ priežiūrą, pakartotinį panaudojimą bei pratęsia PĮ naudojimo laiką. Galima pasakyti, kad re-inžinerija leidžia į PĮ pažvelgti naujai: praktiškai re-inžinerija ir atvirkščioji inžinerija naudojamos kartu. Re-inžinerija yra naudojama dėl kelių reikšmingų priežasčių [Arn93]:

- re-inžinerija gali padėti sumažinti PĮ evoliucijos riziką;
- re-inžineriją naudojanti organizaciją gali geriau skirstyti PĮ skiriamas lėšas;
- re-inžinerija gali padėti lengviau modifikuoti PĮ.

Buvo įrodyta, kad PĮ migruojant iš vienos platformos į kitą, mažinant, verčiant ir minimizuojant priežiūros kainą, gerinant kokybę, naudojant re-inžinerijos metodą tai padaroma žymiai paprasčiau. Re-inžinerija ne tik atnaujina PĮ, bet ir suteikia naujų duomenų apie tai, kokius komponentus bus galima panaudoti ateityje [IEE98].

1.6. Priežiūros įrankiai

PĮ priežiūrai atlikti naudojami įrankiai. Jų gali būti įvairių: komerciniai, pačios PĮ priežiūra užsiimančios organizacijos sukurti įrankiai, kurie dažniausiai tinka tik vienai (prižiūrimai) PĮ priežiūros atlikimui. Pagal techninį lygį: duomenų bazės lygio, procesų lygio. Toliau trumpai, bendrais bruožais apie kelis dažnai naudojamus įrankius priežiūros atlikimui.

1.6.1. Visual Studio Code Analysis

Visual Studio grafinė vartotojo aplinka (*angl. IDE*) siūlo naudoti *Code Analysis* (toliau VSCA), skirtą kodo peržiūrai ir analizei. Tam, kad priežiūros efektyvumas nenukentėtų, rekomenduojama jau PĮ kūrimo fazėje pagalvoti apie būsimą PĮ priežiūrą. Šis įrankis (kaip priedas prie *Visual Studio*) sugeba išanalizuoti parašytą kodą, bei nuspręsti, kurios parašyto kodo vietos turėtų būti dar kartą peržvelgtos ir pataisytos. VSCA išanalizavęs kodą pateikią išsamią informaciją (žiūr. 5 pav.).

Code Metrics Results

Filter: None Min: Max:

Hierarchy	Maintainability Index	Cyclomatic Complexity	Depth of Inheritance	Class Coupling	Lines of Code
BusinessLayer (Release)	38	545	1	9	565
BusinessLayer	38	545	1	9	565
Address	37	265	1	7	275
Address(int, string, string)	76	1		0	4
Id.get() : int	98	1		0	1
LoadAddress(int) : Address	18	102		7	108
Save() : void	7	159		3	160
StreetAddress1.get() : string	98	1		0	1
StreetAddress2.get() : string	98	1		0	1
Customer	38	280	1	7	290
Address.get() : Address	98	1		1	1
Customer(int, string, string)	76	1		0	4
FirstName.get() : string	98	1		0	1
Id.get() : int	98	1		0	1
LastName.get() : string	98	1		0	1
LoadCustomer(int) : Customer	8	146		6	152
Save() : void	13	129		2	130
DataAccessLayer (Release)	95	6	1	2	6
MainApplication (Release)	84	10	7	5	16

5 pav. Visual Studio Code Analysis PĮ priežiūros įrankis

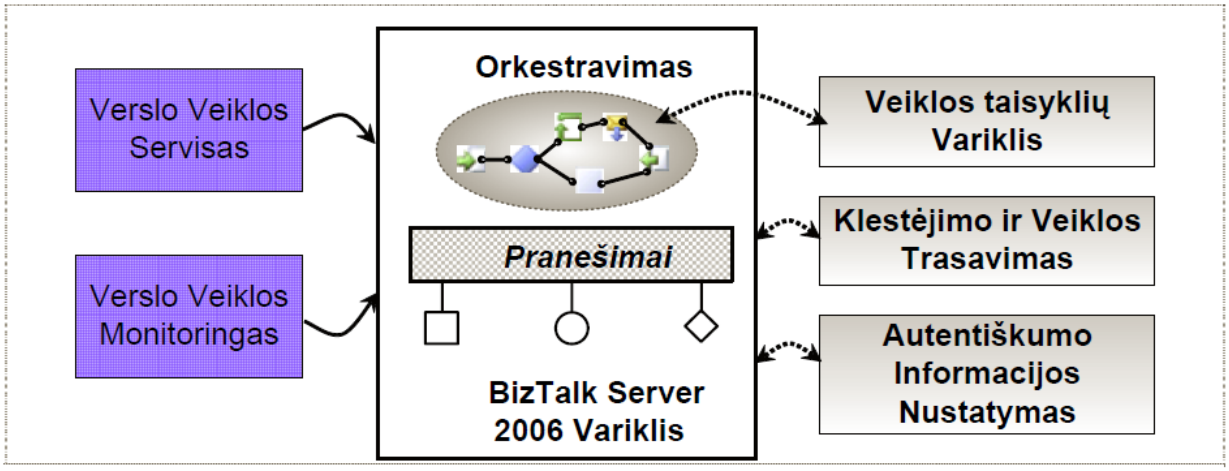
Svarbiausia VSCA išanalizuota metrika yra „*Maintainability Index*“, kuris parodo, kaip sudėtinga ateityje bus prižiūrėti sukurtą PĮ. 5 pav. matoma, kad metodas *LoadAddress* artėja prie kritinės prižiūrimumo ribos, o metodams *Save* ir *LoadCustomer* reikalingas ypatingas dėmesys priežiūros gerinimui. Įdiegus taip sukurtą PĮ, atlikti pakeitimą metoduose *Save* ir *LoadCustomer* bus sudėtinga, sugaišta daug laiko, o gal būt apskritai nebus įmanoma rasti klaidos ir ją ištaisyti. VSCA papildomai suteikia informacijos apie ciklų sudėtingumą, klasių paveldimumo lygį, kodo eilučių

skaičių ir kitą informaciją. Tokia išsami (bet neperkrauta) kodo analizė leidžia susidaryti pakankamą viziją kaip bus atliekama PĮ priežiūra po jos atidavimo klientui.

1.6.2. BizTalk

Microsoft kompanijos sukurtas *BizTalk* produktas yra skirtas įvairių skirtingų PĮ sujungimui į vieningą visumą. Kadangi verslo PĮ dažniausiai yra kuriamos keliais lygiais (duomenys, verslo logika, vartotojo lygis), natūralu, kad kiekvienas lygis gali būti sukurtas, naudojant skirtingus kūrimų įrankius bei technologijas. Pavyzdžiui, duomenys dažniausiai saugomi duomenų bazėse, verslo logiką atlikti gali web servisas, visas informacijos atvaizdavimas paliekamas taikomajai PĮ, programai-klientui. Visus šiuos tris lygius į visumą apjungia BizTalk sistema, kuri ir kontroliuoja visą duomenų judėjimą. *BizTalk Administration Console* – tai interfeisas skirtas administratoriams, kurie turi galimybę valdyti verslo procesus realiu laiku (stabdyti, paleisti, atjungti). Pavyzdžiui, reikalinga informacija apie pardavimų skaičių per savaitę. Informaciją norima gauti kiekvieną savaitę į savo pašto dėžutę. Tokiai situacijai naudojamas standartinis BizTalk SMTP adapteris, kuris suteikia galimybę siųsti ataskaitas pasirinktu formatu.

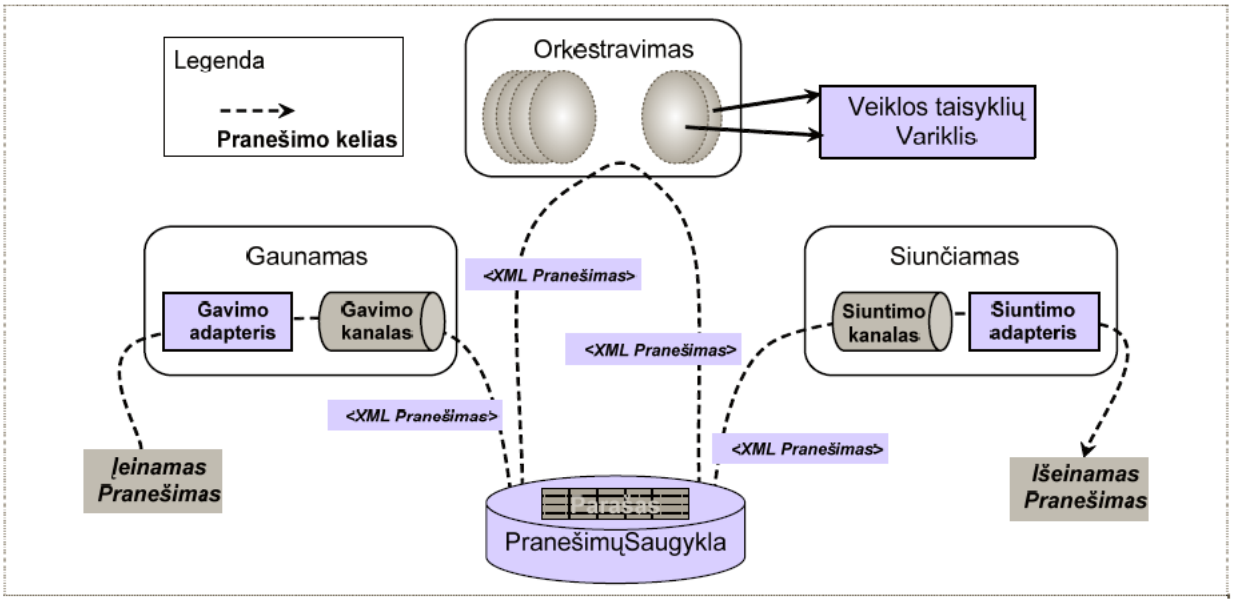
Microsoft BizTalk Server yra komercinis produktas naudojantis Microsoft .NET technologiją. Integruota su valdymo įrankiais, palaikančia SOA (*Service Oriented Architecture*) paradigma, kurios naudoja Web tinklo serverį, WS specifikaciją ir veiklos kontroliavimo portalą (BAM). Servisų orkestravimas (*Orchestration*, dažnai literatūros šaltiniuose dar vadinama kaip – servisų kompozicija) suteikia BizTalk serveriui sutelkti visą dėmesį ties sistema-su-sistema komunikacijai, kurią naudojant yra palaikomi verslo procesai, priklausantys nuo skirtingos programinės įrangos sujungimo [ES06]. Variklio komponentai pavaizduoti 6 pav.



6 pav. BizTalk Server 2006 variklio komponentai [Cha06]

BizTalk Server viduje yra naudojama XLANG kalba, kuri atvaizduoja darbo sekų konstravimą naudojant grafinę vartotojo sąsają. Palaiko XPATH (*XML Path Language*) reiškinius ir taip leidžiama .NET komponentams būti panaudotiems darbų sekų viduje. Programavimas vyksta .NET pagrindu, tačiau reikalingas ir XLANG supratimas [ES06].

Kad būtų leista vartotojams sukurti verslo procesą, kuris apima daugialypes taikomąsias programas, BizTalk Server 2006 variklis turi aprūpinti du pirminius dalykus: pirmas – apibrėžti ir realizuoti verslo proceso logiką; antra – mechanizmu palaikančiu ryšį tarp programos ir verslo proceso naudojimo. 7 pav. iliustruoja svarbiausius variklio komponentus, kurie sprendžią šias dvi problemas [Cha06].



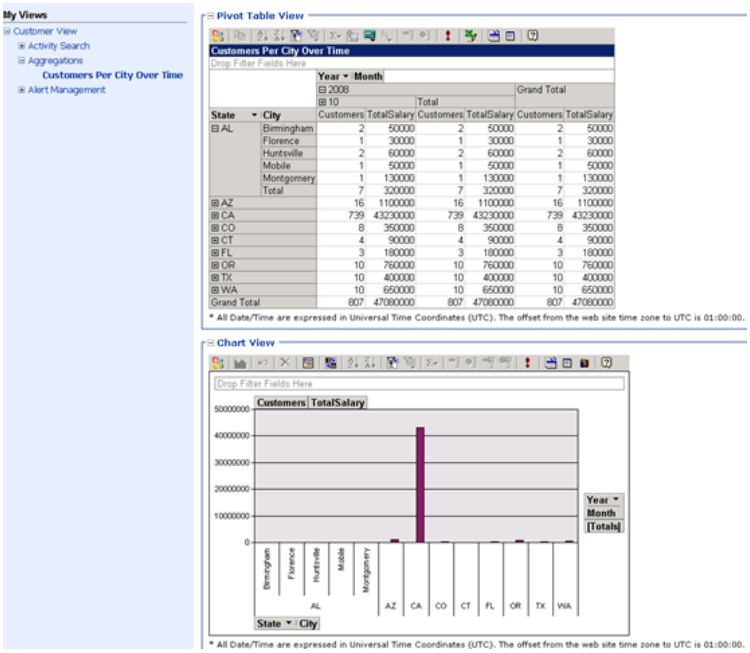
7 pav. BizTalk Server 2006 variklio architektūra

Kaip pavaizduota 7 pav., žinutė yra gaunama per gavimo adapterį. Skirtingi adapteriai tiekia skirtingus komunikacijos mechanizmus, tokiu būdu žinutė galėtų būti gaunama tiek per interneto paslaugas, tiek nuskaitant iš duomenų failų ar kažkokiu kitu būdu. Tada žinutė yra „apdirbama“ per gavimo kanalą. Šis kanalas gali turėti savyje įvairius komponentus, kurie žinutes paverčia iš paprasto formato į XML dokumentą, patvirtinant žinutę skaitmeniniu parašu ir kitokia informacija. Tada žinutė siunčiama į duomenų bazę, pavadintą *Pranešimų Saugykla*, kuri yra įgyvendinta panaudojant *MS SQL Server*. Ši logika taikoma verslo procesų įgyvendinimui su vienu ar daugiau orkestravimų. Kiekvienas orkestravimas kuria žinučių prenumeratą pagal tam tikrą rūšį, kurių jis nori gauti. Kai tinkama žinutė atvyksta į *Pranešimų Saugyklą*, tai ši žinutė yra išsisiunčiama laukiančiam orkestravimui, kuris atlieka veiksmus reikalingam verslo procesui įvykdyti. Rezultatas tipiškai yra

kita žinutė sukurta orkestravimo ir išsaugoma *Pranešimų Saugykloje.* Toliau žinutė siunčiama į siuntimo kanalą, kuris gali paversti iš XML formato (panaudoto BizTalk Server) į formatą reikalingą siunčiamu maršrutu, pridedant skaitmeninį parašą ar kitą informaciją. Tada žinutė yra išsiunčiama per siuntimo adapterį, kuris naudoja tinkamą mechanizmą susisiekimui su taikomąja programa, kuriai ši žinutė yra numatyta [Cha06].

1.6.2.1. BizTalk BAM

Verslo procesų stebėjimas *(Business Activity Monitoring – BAM)* – tai įrankių rinkinys, kuris suteikia galimybę valdyti PĮ, gauti informacinius pranešimus/signalus apie PĮ būklę, vykstančius procesus bei transakcijas, taigi galima imtis atitinkamų veiksmų problemos sprendimui. BAM suteikia galimybę realiu laiku stebėti duomenis, išrinkti juos SQL užklausomis, gauti ataskaitas. Atliekant duomenų užklausas, papildomai galima filtruoti pagal procesus, kurie jau baigė darbą, kurie yra pasiruošę, o kurie sustabdyti dėl neteisingų duomenų arba servisų nepasiekiamumo.

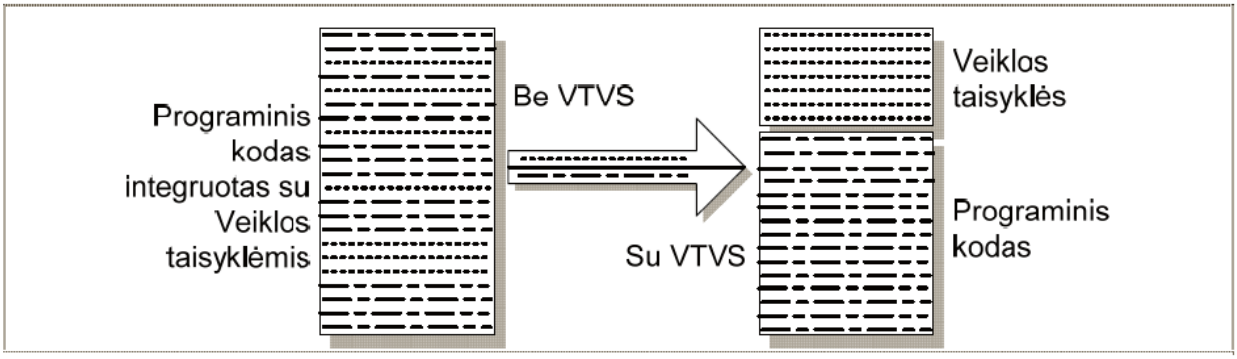


8 pav. BAM portalas

1.6.2.2. BizTalk verslo taisyklių valdymo sistema (VTVS)

Augant organizacijos valdymo lygiui, taip pat didėjant veiklos taisyklių skaičiui, atsirado poreikis kurti tam specifikuotą PĮ. Šios PĮ kūrėjų požiūriai yra panašūs: į vieningą ir lengvesnį organizacijos veiklos valdymą ir priežiūrą. VTVS yra programinė įranga, skirta apibrėžti verslo taisykles, padedanti išvystyti, valdyti ir palaikyti vidinę organizacijos logiką. VTVS yra kaip tiltas jungiantis verslą ir informacines technologijas, leidžianti analitikams kontroliuoti logiką ir net kodą

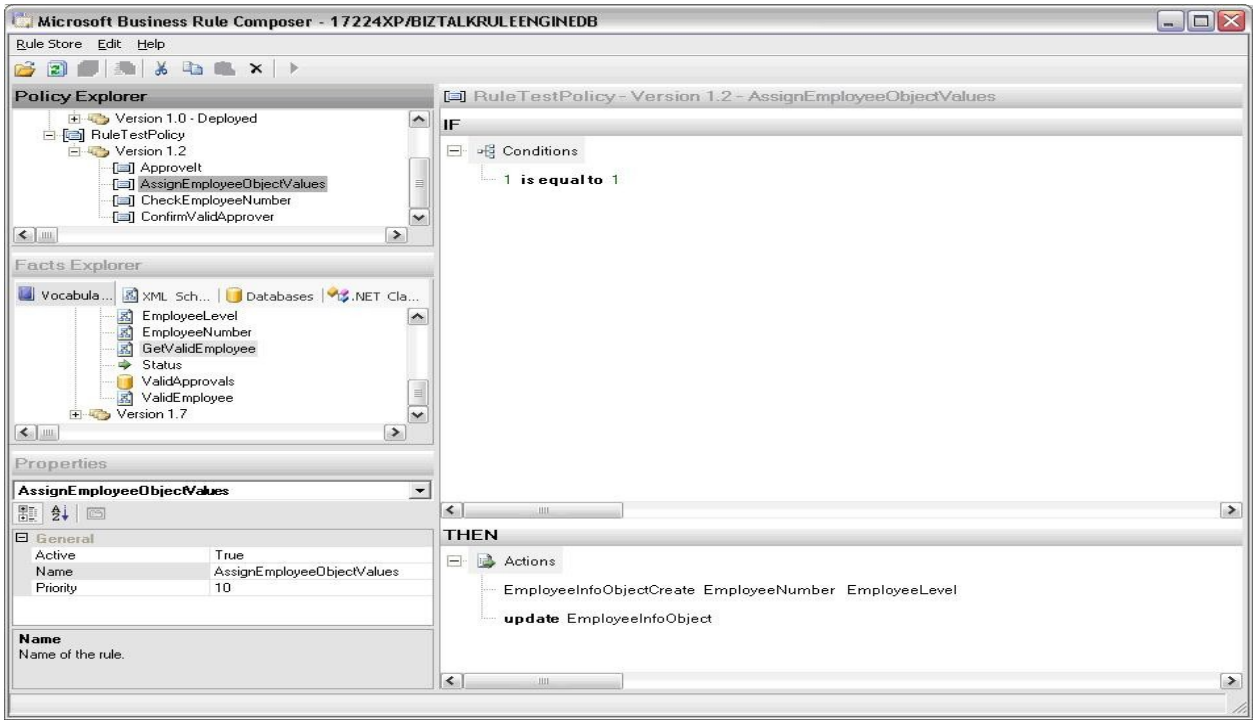
[DEV07, FS08, Gra05, RD05]. Dažniausiai organizacijos senose verslo PĮ verslo taisyklės yra turimos egzistuojančiame taikomajame kode ar procesų žinynuose, ar vadovuose, o žiūrint iš VTVS pusės: ši PĮ aprūpina reikalinga infrastruktūra tokia, kad vienoje vietoje yra sukuriamos ir palaikomos verslo taisyklės. Šios taisyklės yra kaip atskira programos dalis, kurią gali valdyti nebūtinai programuotojų grupė, o apmokytas organizacijos personalas.



9 pav. Skirtumas tarp sistemų, kurios yra kuriamos VTVS pagrindu

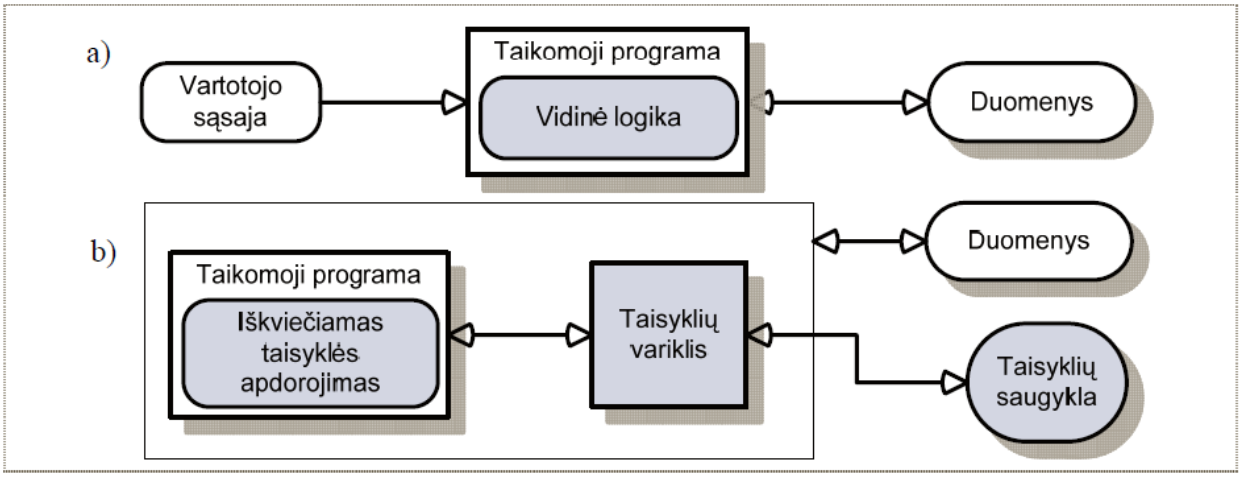
Taigi, naudojant VTVS, išplečiamas kuriamos ir esamos verslo PĮ funkcionalumas. Biztalk VTVS sudėtyje yra tokios pagrindinės dalys [DEV07, Gra05]:

- **taisyklių variklis** (*angl. Rule Engine*), kuris aprūpina reikalingą infrastruktūrą iškviečiant veiklos taisykles;
- **taisyklių saugykla** (*angl. Rule Repository*), kurioje yra saugomos taisyklės t.y. taisyklių rinkiniai. Taisyklių saugykla naudojasi taisyklių variklis;
- **taikomųjų programų kūrimo įrankis** (*angl. Business rule composer*) (žiūr. pav. 10): pagrįstas verslo taisyklėmis, kuris palaiko grafinę vartotojo sąsają.



10 pav. „Business rule composer“ įrankis

Įprastai verslo taisyklės sukuriamos ir užrašomos sąlyginiais sakiniais. VTVS tūri aprūpinti taisyklių: redagavimu, saugojimu ir vykdymu. 11 pav. pavaizduotos dvi architektūros: kur *a* dalyje yra pavaizduota tradicinė, o *b* – šiuolaikinė PĮ panaudojus VTVS [FS08].

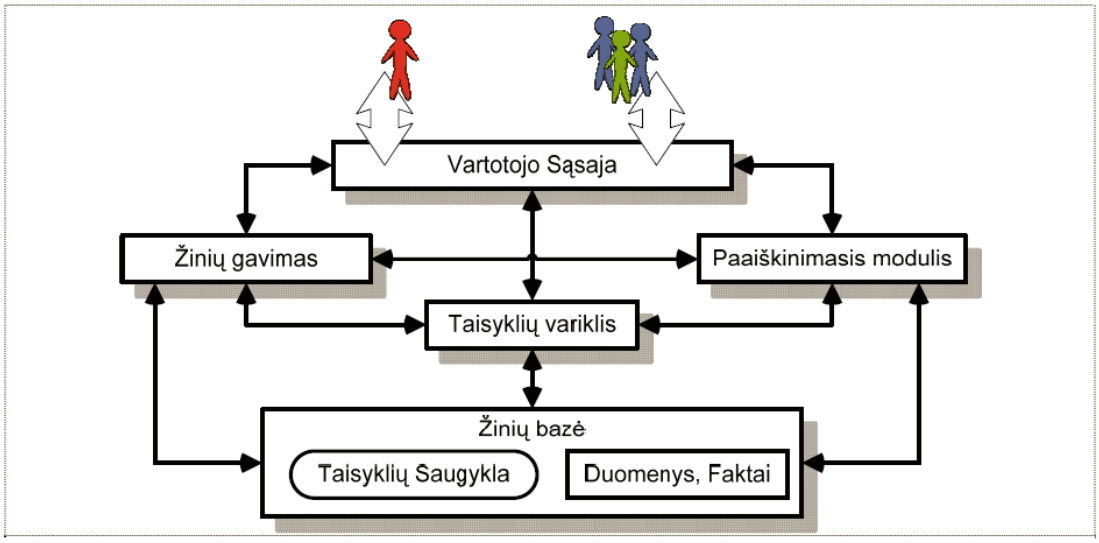


11 pav. Architektūra: a) tradicinė; b) panaudojus verslo taisykles

Kaip jau buvo minėta anksčiau, pagrindinis VTVS tikslas yra kurti, valdyti ir palaikyti organizacijos verslo struktūrą, taigi VTVS turi tokias ypatybes, pareigas ir galimybes [FS08, GRA05]:

- verslo taisyklių saugyklos kaupimas ir priežiūra, kurios pagalba yra valdoma organizacijos politika, logika ir procedūros;
- verslo taisyklių saugyklos sujungimas su taikomosiomis programomis tam, kad taisyklės būtų panaudotos priimant visus sprendimus;
- taisyklių užrašymo kalba yra artima gimtajai kalbai (šiuo atveju anglų kalbai) leidžianti analitikams keisti verslo taisykles ir taip mažinti klaidų skaičių;
- ne tik analitikams, tačiau ir organizacijos netechniniam personalui leidžiantys kurti suprantamas ir svarbiausias verslo taisykles, su minimaliomis PĮ kūrimo žiniomis;
- automatizuoti ir lengvinti vidinius ar išorinius organizacijos procesus;
- lengvas taikomųjų programų kūrimas, taip pat vartotojui aiškūs, suprantami ir logiški taikomosios programos dialogai ir langai;
- realiu atveju, pritaikius VTVS verslo PĮ, yra įmanoma taip sukonfigūruoti verslo PĮ, kad pačią PĮ priežiūrą atlieka klientas (užsakovas).

VTVS aprūpina vartotoją įrankiu, kuriuo yra valdomos taisyklės ir taisyklių rinkiniai: tekstinis ir grafinis redaktoriai yra kaip vartotojo grafinės sąsajos dalis. Taisyklės, ypač jų argumentai ar išvados, veikia duomenis. Duomenys kaupiami ir valdomi MS SQL Server duomenų bazėje. PĮ naudoja taikomąjį serverį, kuris atskiria vartotojų interfeisą, veiklos logika ir duomenų valdymą. Detalesnė VTVS architektūra pavaizduota 12 pav. [Lam07].



12 pav. Detalesnė VTVS architektūra

Taisyklių (žinių) inžinierius arba analitikas palaiko naujausias žinias PĮ, prideda naujas taisykles ir tikrina taisyklių rinkinius. Tikrinimas yra labai svarbus, nes taisyklių rinkiniai neturi turėti savyje taisyklių, kurios priveda prie prieštaraujančių rezultatų. Žinių ekspertas gali būti veiklos ekspertu, kuris tiesiogiai bendrauja su PĮ. Sistema tai pat gali būti panaudota kitos sistemos, pavyzdžiui, transakcijų automatizavimas atliekant pirkimo užsakymus [Lam07].

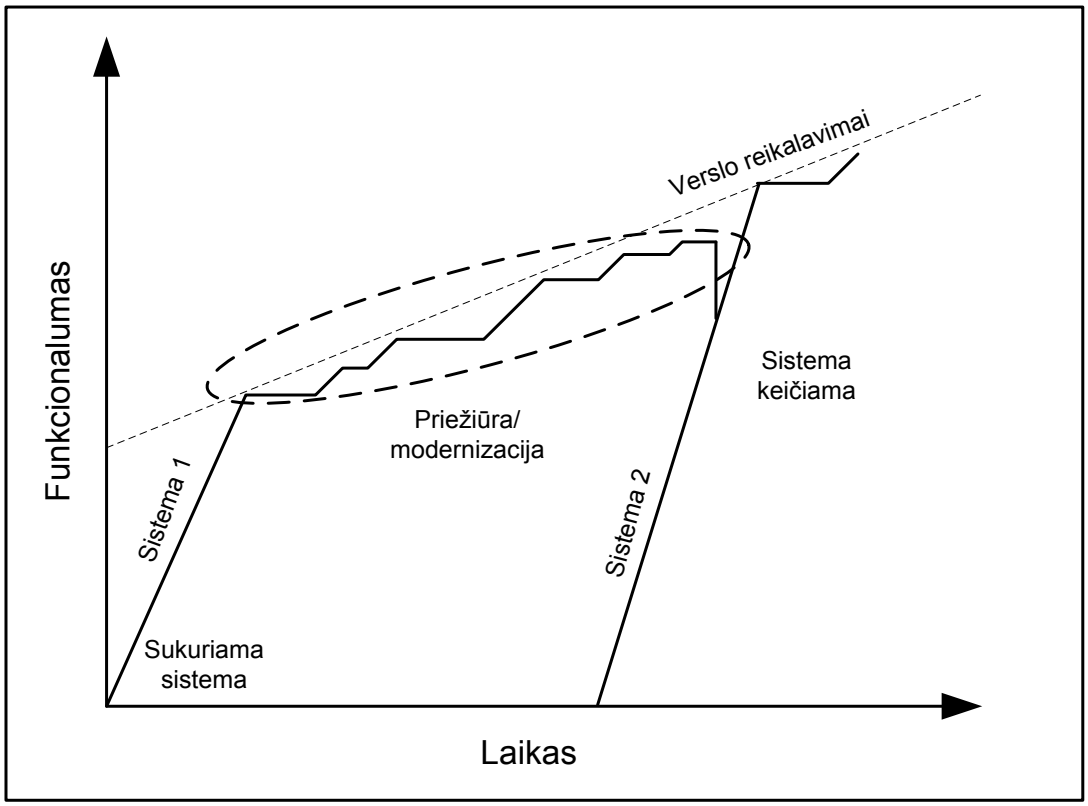
1 lentelė. Naudotojai ir pareigos išvystant VTVS

Naudotojai	Pareigos
IT specialistas	Kurti taisyklių formas, plėtoti taisyklių valdomą sistemą (TVS)
Analitikas	Taisyklių išvedimas, taisyklių formavimas, taisyklių pakeitimas naudojant TVS
Apmokytas vartotojas	Koordinuoti su įvairiais taisyklių savininkais, nustatyti jų prioritetus ir įgyvendinti taisyklių pakeitimus, valdyti taisyklių saugyklą
Vartotojas	Galutinis vartotojas, kuris tik naudojasi PĮ

1.7. Senos verslo PĮ evoliucionavimas

Senos PĮ evoliucionavimas apima didžiąją dalį PĮ plėtojimo veiklų pradedant veiklomis, kuriomis į duomenų bazės lentelę yra įterpiamas naujas laukas ir baigiant veiklomis, kurias atliekant iš naujo yra diegiama PĮ. Senos PĮ evoliucionavimas galėtų būti suskirstytas į dvi kategorijas: PĮ atnaujinimas ir PĮ pakeitimas. 13 pav. kylanti linija vaizduoja PĮ verslo reikalavimus, karpyta linija pavaizduotos PĮ galimybės. Galiausiai, kai PĮ toliau nebegali būti vystoma, ją turi pakeisti kita PĮ [GT03].

Reikia nuspręsti, kas turi būti daroma su PĮ, kokių veiksmų reikia imtis tam, kad išlaikytume senos PĮ vertę organizacijoje. Ar ją reikia toliau plėtoti/moderninti ar jau pakeisti nauja PĮ? Tam, kad priimtume reikalingą sprendimą, organizacija turi įvertinti turimą seną PĮ, nustatyti atitinkamą evoliucionavimo strategiją ir išanalizuoti kiekvienos strategijos įtaką verslui.



13 pav. Programinės įrangos evoliucionavimas [SPL03]

1.8. PĮ atnaujinimas

PĮ atnaujinimas, tai tokia PĮ priežiūra, kai programinei įrangai yra atliekami dideli pakeitimai: perprojektuojama, daug komponentų perrašoma nauja programavimo kalba, migruojami seni duomenys ir kitos veiklos. Žinoma, tokie darbai atliekami atsižvelgiant į esamus ir pasikeitusius organizacijos verslo poreikius. Dažniausiai PĮ modernizaciją atlikti paskatina esamos PĮ nelankstumas, negalėjimas toliau diegti naujų pakeitimų, vientisumo nebuvimas PĮ, pažeidžiamumas [BLW+99].

1.8.1. Modernizacijos rūšys

1.8.1.1. Baltos dėžės modernizacija

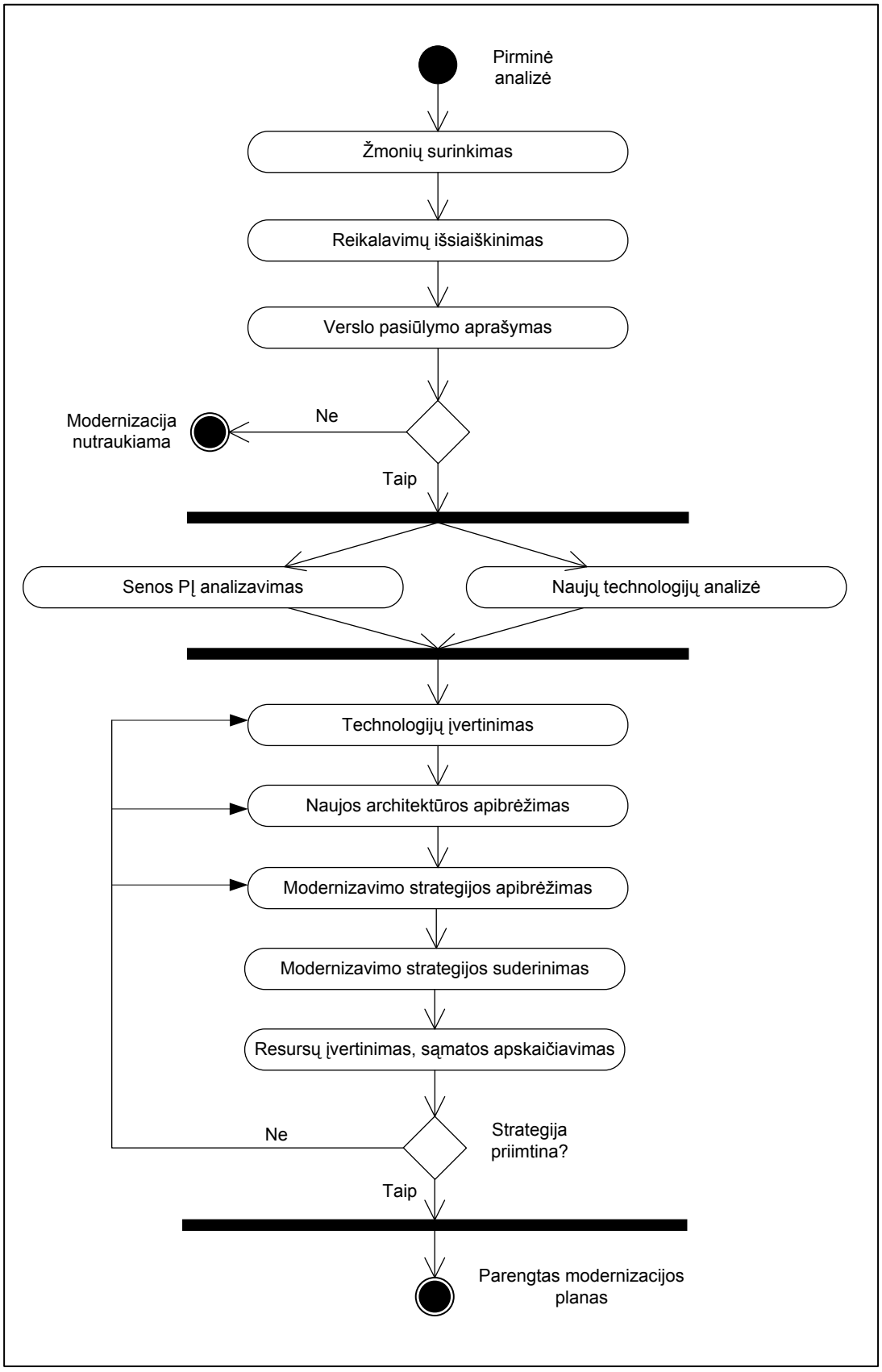
Tai toks PĮ atnaujinimas, kuriam atlikti reikalingos žinios apie esamos PĮ vidinę struktūrą ir jos veikimą. Tam, kad būtų suprasta, kaip veikia sena PĮ, reikia atlikti detalią kodo analizę, kurios metu galima susidaryti bendrą vaizdą apie PĮ atliekamus veiksmus bei esamą struktūrą [TRB95]. Kai kodas yra pilnai suprastas, tada galima atlikti kodo arba PĮ restruktūrizavimo darbus. Tokie restruktūrizavimo darbai dažniausiai pagerina PĮ kokybę, padidina spartą, padaro PĮ lengviau prižiūrimą.

1.8.1.2. Juodos dėžės modernizacija

Tai toks PĮ atnaujinimas, kuriam atlikti reikalingi tik įėjimo ir išėjimo duomenys. Į tai, kaip PĮ veikia, nereikia kreipti dėmesio. Tai lengvesnis modernizavimo būdas, lyginant su baltos dėžės atnaujinimu. Atnaujinant seną verslo PĮ, juodos dėžės modernizavimo metodu, sena PĮ yra paslepiama po nauja-tarpine PĮ. Tokiu būdu yra paslepiama dalis senos PĮ. Toks apjungimas – tai viena iš reinžinerijos užduočių. Jos metu yra nagrinėjami tik senos PĮ išorės interfeisai nekreipiant dėmesio į vidines PĮ detales. Tačiau, kaip rodo praktika, atliekant modernizavimą šiuo būdu, papildomai prireikia ir baltos dėžės metodo [PDH+99].

1.9. Senos PĮ atnaujinimo planas

14 pav. pavaizduota senos verslo PĮ atnaujinimo plano sudarymo diagrama. Ovalai vaizduoja veiklas, rodyklės – parėjimus nuo vienos veiklos prie kitos. Senos verslo PĮ atnaujinimas pradedamas investiciniais klausimais ir baigiamas modernizavimo plano pateikimu, bei jo įgyvendinimu.



14 pav. Senos verslo PĮ atnaujinimo plano sudarymas

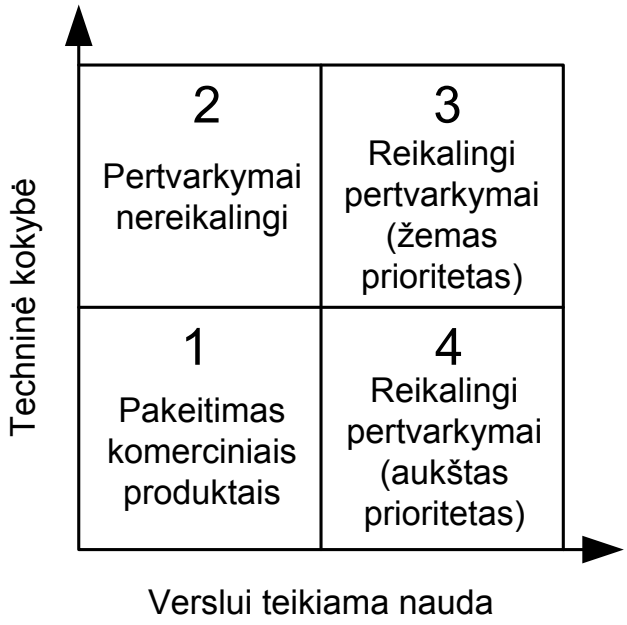
1.9.1. Pirminė analizė

Pirminės analizės metu nustatomi senos PĮ techniniai matai, PĮ vertė, vykdant verslo procesus [War99].

Techninė PĮ kokybė yra vertinama tokiais matais kaip PĮ atnaujinimo dažnumas, paprastumas atlikti pakeitimus PĮ, techninės įrangos (ant kurios veika PĮ) patikimumas, organizacijos infrastruktūra, PĮ našumas, tikslumas, lengvumas naudotis PĮ, galimybė apmokyti naujus darbuotojus dirbti su sena PĮ.

Iš verslo pusės, ar sena PĮ yra svarbi organizacijai, gali būti pamatuojama šiais kriterijais: PĮ teikiama nauda organizacijai, verslo PĮ naudojimo lygis organizacijoje, pasiektų verslo tikslų skaičius, naudotojų pasitenkinimas esama PĮ, informacijos, kurią saugo PĮ, svarbumas.

Sena verslo PĮ gali būti pamatuojama pagal šiuos faktorius ir taip nusprendžiama PĮ svarba. Pagal tai, į kurį kvadratą PĮ patenka (žiūr. 15 pav.), galima nuspręsti, kas bus daroma su sena PĮ.



15 pav. Senos PĮ pakeitimų nustatymo schema [SPL03]

Kvadratas nr. 1. Verslo PĮ, patekusi į šią grupę, turinti menką įtaką verslui ir prastą kokybę, turi būti pakeista naujais produktais. Kadangi PĮ yra prastos kokybės, ji turi būti pakeista arba bandoma šios PĮ kokybę pagerinti. PĮ, turinti mažą įtaką verslui, neatliekanti pagrindinių verslo procesų turėtų būti taip pat pakeičiama.

Kvadratas nr. 2. PĮ, turinčios gerą kokybę, bet neturinčios didelės įtakos verslo procesams. Tokia PĮ nereikalauja re-inžinerijos, atnaujinimo darbų, modernizacijos ar pakeitimo.

Kvadratas nr. 3. PĮ, turinčios aukštą kokybę ir atsakingos už pagrindinius verslo procesus, turėtų nuolat evoliucionuoti, jai turėtų būti atlikti minimalūs pakeitimai, tam, kad PĮ išlaikytų savo vertę.
Kvadratas nr. 4. Verslo PĮ, kuri vykdo pagrindinius organizacijos procesus, bet esanti prastos kokybės, turi būti pakeičiama arba jai atliekami modernizavimo darbai.

Aprašytame verslo sprendime (dokumente) turi atsispindėti kainos-privalumų analizė, kokie bus finansiniai pranašumai, kuo modernizavimas bus naudingas. Jeigu organizacija turi pakankamai resursų modernizuoti seną verslo PĮ, tai dar nereiškia, kad PĮ modernizacija ir turi būti atliekama.

1.9.2. Reikalingų asmenų nustatymas

PĮ atnaujinimui reikalingi darbuotojai - tai asmenys ar jų grupės, kurie glaudžiai susiję su atnaujinama verslo PĮ [SPL03]:

- *galutiniai PĮ vartotojai* – tai žmonės, kurie tiesiogiai dirba su verslo PĮ. Šie žmonės gali puikiai aprašyti, ko trūksta senoje PĮ, kaip ji turėtų veikti, kad dirbti su ja būtų patogiau ir efektyviau;
- *PĮ kūrimo komanda*. Ją sudaro projektuotojai, programuotojai, testuotojai, analitikai, projekto vadovai, architektai. Šie žmonės gali ir neturėti žinių apie seną verslo PĮ, kaip ji veikia, iš ko susideda, bet turi turėti žinių apie naujas technologijas, mokėti jomis naudotis. Kadangi organizacijose architektai dažniausiai dirba prie kelių projektų vienu metu, bendrauja tiek su projektų vadovais, tiek su programuotojais, todėl yra svarbu, kad senos PĮ modernizavimo projektas turėtų atskirą architektų komandą, kuri 100 procentų dirbs atnaujinant senos PĮ architektūrą. Darbo komandoje turėtų būti bent jau po vieną programuotoją, testuotoją, analitiką, architektą ir projektų vadovą. Efektyviausiam darbų atlikimui pageidautina, kad žmonės visą laiką (100%) skirtų PĮ atnaujinimo darbams. Laikas, kiek reikia dirbti, priklauso nuo atnaujinamos PĮ dydžio ir sudėtingumo;
- *senos PĮ priežiūros personalas* – tai programuotojai ir inžinieriai, kurie prižiūri seną PĮ nuo pat jos įdiegimo. Jie turi daug patirties ir jų žinios atliekant modernizavimo darbus yra labai naudingos.

1.9.3. Reikalavimų išsiaiškinimas

Vartotojų, sisteminiai ir nefunkciniai [SPL03] reikalavimai modernizuotai PĮ yra formuluojami atsižvelgiant į vartotojus, esamą PĮ, atliekant verslo procesų reinžineriją:

- *reikalavimai atsižvelgiant į seną PĮ*. Didžioji dalis reikalavimų ateina iš senos PĮ, kurioje jau yra veikiantys verslo procesai. Iš senos PĮ dažniausiai kyla priežiūros palengvinimo reikalavimai, lengvesnis naudojimasis PĮ ar spartesnis PĮ veikimas;

- *reikalavimai, ateinantys iš pasikeitusių verslo procesų*. Tam, kad efektyviai atlikti modernizaciją, reikia išsiaiškinti, kokie procesai jau veikia senoje PĮ, kuriuos reikia pakeisti, kuriuos tobulinti, o kurių visai atsisakyti. Formuluojant tokius reikalavimus reikia nepamiršti ir tai, kad pakeitimai verslo procesuose gali paveikti PĮ. Tai apsunkina atnaujinimo darbus;
- *vartotojų formuluojami reikalavimai*. Vartotojai dažnai pateikia reikalavimus, kurie jų manymu palengvins kasdieninį darbą su esama PĮ. Tačiau gali atsitikti ir taip, kad gali būti pateikiami ir individualūs reikalavimai. Pavyzdžiui, reikalavimas įdėti klaviatūros mygtukų kombinaciją, kurią paspaudus atsidaro meniu;
- *technologiniai aspektai*. Dažnai yra norima naujos technologijos privalumus panaudoti kaip reikalavimus moderninant PĮ. Kai kurie netgi mano, kad naujos technologijos gali būti lėtesnės, su jomis parašytas programinis kodas bus sunkiai prižiūrimas arba sunkiai naudojamas. Tai teisingas požiūris, nes negalime atmesti varianto, kad naujos technologijos gali sukurti ir naujas problemas;
- *apribojimai*. Apribojimo pavyzdžiais galėti būti nustatytas modernizavimo biudžetas, laiko sąnaudos, prisirišimas prie standartų.

1.9.4. Verslo pasiūlymo aprašymas

Verslo scenarijuje, kurį sudaro dažniausiai analitikai, turi būti detaliai aprašoma, kokių pajamų tikimasi, kokia bus kaina, aprašyti techniniai ir valdymo planai. Verslo pasiūlyme turi būti aprašyta probleminė sritis, siūlomas sprendimas, nauda bei rizikos:

- *probleminė sritis*. Turi būti aprašyta esama situacija ir pabrėžiami faktai, kodėl PĮ yra neefektyvi, pažeidžiama ar neatitinka reikalavimų;
- *sprendimas*. Verslo scenarijuje turi būti aprašytas nedetalizuotas problemos sprendimo būdas. Pavyzdžiui, sprendimas migruoti seną PĮ į naują aparatūrinę įrangą. Sprendime turėtų būti nurodytas ir laiko planas, iki kada tai turi būti padaryta, kokia bus kaina;
- *rizikos*. Turi būti aprašomos rizikos, kas gali nutikti, kokie galimi nenumatyti atvejai išryškės atnaujinus PĮ. Tai padės efektyviau pasirinkti modernizavimo strategiją;
- *nauda*. Čia aprašoma, kokia bus nauda, jei bus įgyvendintas problemos sprendimas. Galima aprašyti naudingumo intervalais: mažiausia nauda, tikėtina ir maksimali nauda verslui.

Formuluojant verslo scenarijų reikia atsižvelgti ir į galimas prielaidas. Prielaidos turėtų būti dokumentuojamos, kad esant reikalui, galėtume peržiūrėti ir atlikti patikslinimus verslo scenarijuose. Pavyzdžiui, galima prielaida, kad išliks tie patys darbuotojai, kurie naudojasi PĮ arba nebus pasikeitusios technologijos, kuriomis ruošiamasi modernizuoti seną PĮ.

Egzistuoja du būdai kaip galima pasiekti norimą kainos lygį. Tai išlaidų mažinimas ir išlaidų vengimas. Išlaidų mažinimas reikalauja veiksmų, kuriuos atlikus sumažėja išlaidos. Išlaidų vengimas atsispindi veiksmuose, kuriuo atlikus bus išvengiama nereikalingų išlaidų ateityje. Kadangi į PĮ atnaujinimo ir modernizavimo darbus įeina investavimas į apmokymus, architektūros pakeitimą, darbuotojų samdymas/atleidimas, reikia atlikti teikiamos naudos – išlaidų analizę. Šios analizės metu yra lyginama senos PĮ modernizavimo kaina su po modernizavimo atlikimo būsima nauda.

Senos verslo PĮ atnaujinimo privalumai lyginant su naujos kūrimu:

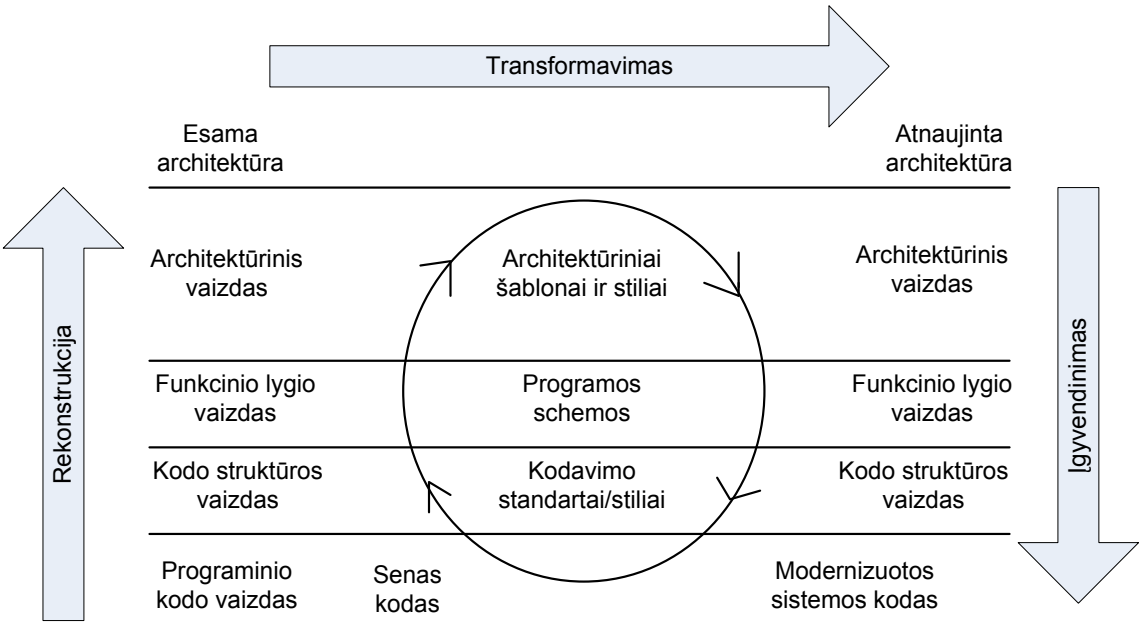
- *mažesnė kaina*. Atnaujinti verslo PĮ dažniausiai yra pigiau nei kurti naują;
- *mažesnė rizika*. Palaipsniui atnaujinant PĮ mažėja rizika, kad PĮ neveiks;
- *galimybė panaudoti darbuotojus, dirbančius su sena PĮ*. Šie darbuotojai jau turi patirtį, todėl nereikia ieškoti naujų darbuotojų. Seni darbuotojai turi galimybę tobulėti, kai modernizuojama senoji PĮ;
- *naujų verslo taisyklių aptikimas*. Dažnai būna taip, kad atnaujinant PĮ aptinkama kodo vietų, kuris būna suprogramuotas ekstremaliai situacijai. Tarkim, kas atsitiks, jei PĮ "nulūš" vykdant kliento užsakymą;
- *prieinamumas*. Pažingsninis atnaujinimas sumažina PĮ neveikimo riziką lyginant su naujos PĮ diegimu į gamybinę aplinką;
- *galutinio vartotojo pasitenkinimas*. Taupomas vartotojo laikas išmokstant naudotis atnaujinta PĮ.

1.9.5. Senos PĮ analizavimas

Kada yra sukuriamas ir patvirtinamas verslo pasiūlymas, lygiagrečiai galime pradėti vykdyti dvi veiklas: senos PĮ analizė ir naujų technologijų analizė į kurią bus orientuota atnaujinta PĮ. Vykdant senos PĮ modernizaciją, reikia atlikti tris pagrindinius procesus:

- atkurti senos PĮ loginį vaizdą (atskirti, kas su kuo susiję);
- transformuoti esamus loginius vienetus į naujus, patobulintus;
- realizuoti patobulintą logiką.

Priklausomai nuo to, ką siekiama modernizuoti, keičiasi ir šių procesų analizavimo lygis.



16 pav. Senos PĮ transformavimo mechanizmas [ULR02]

1.9.5.1. Programinio kodo transformacija

Jeigu senos PĮ kodas yra prastai struktūrizuotas, aprašytas ir nei vienas programuotojas negali paaiškinti, kas ir kaip yra suprogramuota, geriausiai tokį kodą keisti nauju. Jei bus paliktas senas kodas, koks jis yra, ir pridėta papildomai pakeitimų, tai ateityje šis kodas išaugs, taps sunkiai suprantamas ir sunkiai prižiūrimas. Tam, kad gerai būtų suprastas programinis kodas, galima atlikti tokias veiklas [ULR02]:

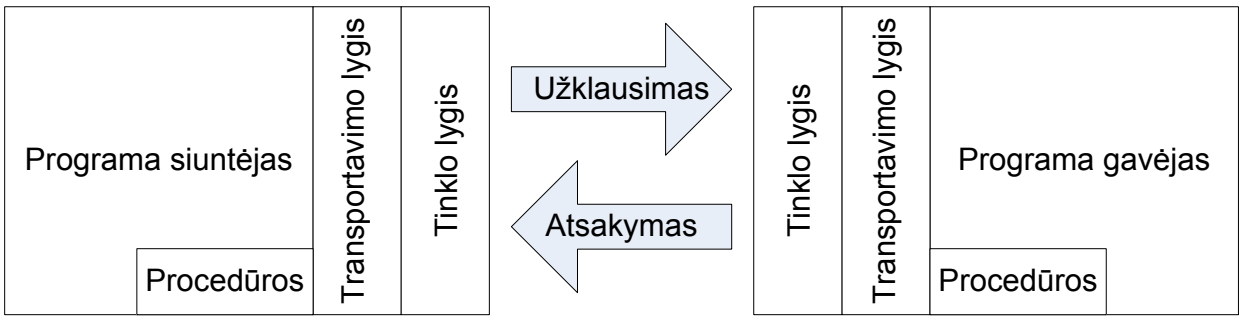
- **nuodugnus kodo skaitymas**. Programuotojai, užsiimantys šia veikla, atidžiai eilutė po eilutės nagrinėja programinį kodą. Tokia veikla nėra tinkama didelei PĮ, nes tyrimais įrodyta, kad gerus sugebėjimus turintis programuotojas, sėkmingai gali peržiūrėti apie 50000 kodo eilučių. Jeigu programą sudaro daugiau kodo, informacijos kiekis programuotojui tampa sunkiai valdomas;
- **statinė kodo analizė**. Ši veikla apima veiksmus, kuriais iš programinio kodo norima sugeneruoti įvairias ataskaitas, grafus, duomenų ir valdymo srautus, struktūrines diagramas. Dauguma atvirkščiosios reinžinerijos įrankių suteikia galimybę atlikti statinę kodo analizę;
- **dinaminė analizė**. Šios analizės metu yra apžvelgiama, kas vyksta, kai programa veikia realioje aplinkoje. Dinaminė analizė gali padėti programuotojams suprasti PĮ, kuri yra dinamiškai konfigūruojama. Tokių PĮ pavyzdžiais gali būti realaus laiko skaičiavimo PĮ ar kliento/serverio programos.

1.9.6. Naujų technologijų analizė

1.9.6.1. Komunikacijos modeliai

Šiandieninėje rinkoje egzistuoja gausybė komercinių produktų, kurie skirti kurti išskirstytas, transakcijomis paremtas verslo valdymo sistemas. Šie produktai (technologijos) dažniausiai yra paremti sinchronišku nutolusių (esančių kitame kompiuteryje) procedūrų iškvietimu arba pranešimų eilėmis (angl. *queues*). Sinchronišką tokių procedūrų iškvietimą turinti PĮ dažniausiai yra siejamas su išskirstytu-transakciniu modeliu (angl. *distributed-transaction model*), o pranešimų eilėmis paremta PĮ vadinama eilių-transakcijų modeliu paremta programine įranga. Dažniausiai yra sunku nuspręsti, kurį modelį geriau pasirinkti vienoje ar kitoje situacijoje, nes abu modeliai turi savo privalumų ir trūkumu. Modelio pasirinkimas nėra lengvai nuspėjamas ir intuityvus, todėl reikalinga kruopšti PĮ ir modelio analizė [Hou99].

Išskirstytas komunikavimas suteikia galimybę PĮ veikti skirtinguose kompiuteriuose apsikeičiant duomenimis tiesiogiai arba netiesiogiai panaudojant komunikavimo mechanizmą. Modeliuose, kurie suteikia galimybę komunikuoti sinchroniškai (žiūr. pav. 17), PĮ – siuntėjas (angl. *sender*) yra užsiblokavęs tol, kol negauna atsakymo iš PĮ – gavėjo (angl. *receiver*). Toks komunikavimo mechanizmas puikiai veikia LAN tinkle, ten, kur duomenų perdavimo greitis yra didelis. Sinchroninis komunikavimas turėtų būtų atsargiai naudojamas tinkluose, turinčiuose mažą duomenų pralaidumą, pavyzdžiui, internete. Problemos gali kilti dėl to, kad gavėjas arba siuntėjas gali užsiblokuoti dėl lėto duomenų perdavimo.

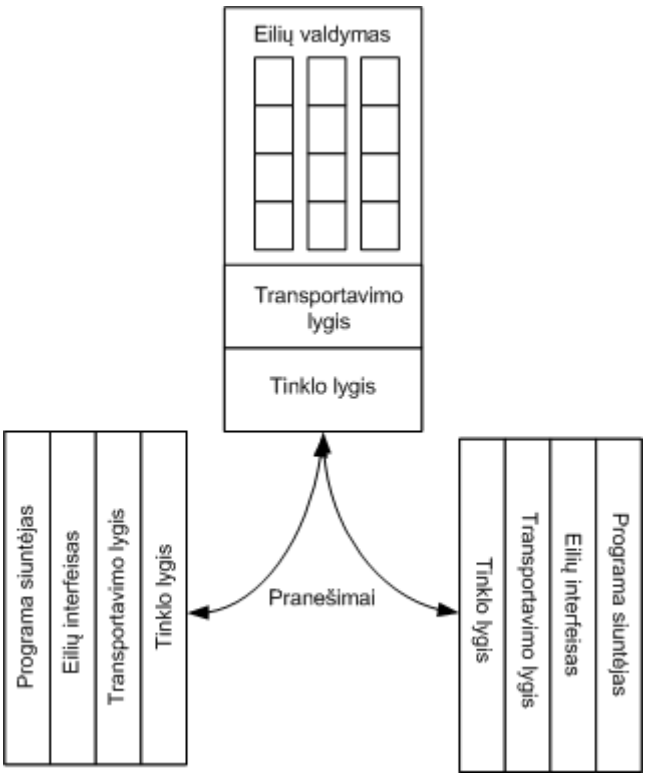


17 pav. Sinchroninis nutolusių procedūrų kvietimas [Hou99]

Naudojant asinchroninį komunikavimo būdą, PĮ procesai nesiblokuoja. Pranešimas gavėjui yra išsiunčiamas, bet atsakymo apie tai, ar sėkmingai gavėjas pranešimą priėmė, gali būti gautas vėliau. Panaudojant šį komunikavimo mechanizmą, programuotojai gali kurti tokią PĮ, kuri po išsiuntimo gali papildomai vykdyti kitus veiksmus, vykdyti kitas operacijas.

Tiesioginio komunikavimo metu, iš PĮ yra reikalaujama tiesioginio ryšio užmezgimo su gavėju. PĮ-gavėjas turi būti prieinamas užmezgant tiesioginį komunikavimo ryšį. Kai tik siuntėjas susijungia su gavėju, yra sukuriamas tarp jų kanalas, kuriuo pasinaudojant galimas apsikeitimas duomenimis. Papildomai, tiek gavėjas, tiek siuntėjas apsikeičia susijungimo būsenomis, pagal kurias galima spręsti, kiek laiko ar kokie duomenų kiekiai bus perduodami.

Skirtingai nuo tiesioginio (angl. *direct*) ryšio, netiesioginis komunikavimo būdas, toks kaip pranešimų-eilių mechanizmas (žiūr. pav. 18), nereikalauja PĮ – gavėjo prisijungimo, kai yra siunčiamas pranešimas. Siuntėjas dažniausiai padeda pranešimą į eilę, iš kurios vėliau gavėjas galės pasiimti ir atlikti reikiamas operacijas. Jeigu pranešimai nebūtų padedami į eilę, gavėjas nežinotų, ar apskritai buvo kreipiamasi į jį, ir tokiu būdu komunikacija neįvyktų. Nors toks komunikavimo būdas nereikalauja gavėjo prieinamumo, tačiau pats pranešimo padėjimas į eilę yra sinchroninis, todėl tuo metu, kada pranešimas yra siunčiamas, siuntėjas užsiblokuoja. Pranešimai šiuo komunikacijos būdu yra siunčiami į vieną pusę, asinchroniškai, nefiksuojant būsenų, todėl šis komunikavimo būdas reikalauja, kad siuntėjo būsena būtų įrašyta į siunčiamą pranešimą.

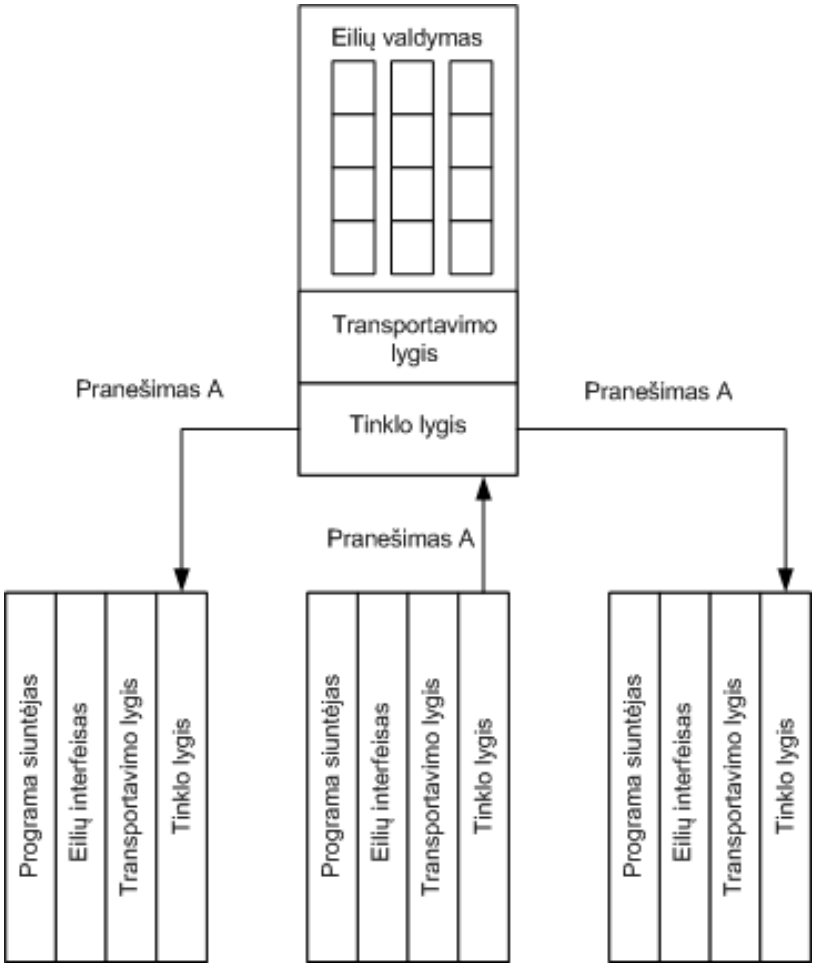


18 pav. Pranešimų-eilių mechanizmas [RD05]

Išsiųsti/užsakomi pranešimai yra pravaizduoti 19 pav. Principas yra labai panašus į pranešimų-eilių modelį, tik skirtumas tas, kad pranešimai iš eilių gali būtų siunčiami keliems gavėjams. Šiame

modelyje, PĮ-siuntėja išsiunčia pranešimą į eiles ir tuo pačiu metu pažymi, kokio pranešimo laukia. Tiek siuntėjui, tiek laukiančiajam nereikalaujama žinoti apie kitos pusės egzistavimą, vietą, apkrovą ar būseną kurioje esama.

Komunikacijos technologijos pasirinkimas priklauso nuo verslo PĮ reikalavimų. Pasirinkimui didelę įtaką turi verslo PĮ architektūra bei būdas, kaip yra atliekamos transakcijos. Trumpai tariant, sinchroninis nutolusių procedūrų kvietimas geriausiai tinka tuomet, kai tiek siuntėjas, tiek gavėjas yra pasiekiami tiesiogiai, bei kada programa siuntėjas negali toliau tęsti darbo, kol negavo atsakymo iš gavėjo. Pranešimo-eilių bendravimo modelis tinkamas tuomet, kai siuntėjas gali būti ne visada pasiekiamas, bei kada nereikalaujama staigaus, greito atsakymo iš gavėjo.



19 pav. Išsiųstų/užsakomų pranešimų mechanizmas [RD05]

Atitinkamai PĮ, kuri naudoja sinchroninį komunikavimo būdą, tuo pačiu naudojasi ir išskirstytų transakcijų modeliu, tuo tarpu PĮ, naudojanti eilių komunikavimo mechanizmą, naudojasi ir eilių transakcijos modeliu. Šių dviejų tipų transakcijos privalumai ir skirtumai yra pateikiami 2 ir 3

lentelėse. Pagrindinis išskirstytos transakcijos modelio privalumas yra lengvas panaudojimas ir pritaikymas verslo procesuose bei galimybė supaprastinti klaidų aptikimą ir taisymą programinėje įrangoje. Pagrindine neigiama savybe yra laikoma siuntėjo užblokavimas laukiant atsakymo iš gavėjo. Eilėmis paremtos transakcijos privalumas yra asinchroninis komunikavimas, todėl PĮ, kuri naudoja šį transakcijų mechanizmą, yra lankstesnė, turi daugiau galimybių būti plėtojamai. Pagrindinis šio transakcijos modelio minusas yra sudėtingas būsenos atstatymo, klaidų taisymo įgyvendinimas. Renkantis transakcijų modelį verslo PĮ, iš esmės reikia atkreipti dėmesį į tai, kaip lengvai bus galima atstatyti pradinę PĮ būseną prieš transakcijos pradžią [ULR02].

2 lentelė. Išskirstytų transakcijų (sinchroninio) modelio privalumai ir trūkumai [SPL03]

Privalumai	**Trūkumai**
Laiko patikrinta, „brandi“ technologija	Siuntėjas yra „užrakinamas“, kol negauna atsakymo iš gavėjo
Sudėtingas transakcijas yra lengva realizuoti	Labai prisirišama prie serverio, nes tik su vienu serveriu galima bendrauti
Klaidų aptikimas ir ištaisymas yra lengvas	Reikalauja, kad visos PĮ dalys, tinklas, tarpiniai komponentai veiktų ir būtų pasiekiami.

3 lentelė. Eilių transakcijų modelio privalumai ir trūkumai [SPL03]

Privalumai	**Trūkumai**
Siuntėjas nėra blokuojamas, kol laukia atsakymo iš gavėjo	Gali būti sunkiai realizuojamos sudėtingos transakcijos
Mažiau priklausoma nuo kitos PĮ ir tinklo	Gavėjas nežino, kada į eilę atėjo jam skirtas pranešimas
Suteikiama galimybė lygiagrečiai atlikti kitas operacijas	
Galimybė atšaukti siuntimą išmetant jį iš eilės, keisti prioritetą, jei gavėjas, jo dar nepaėmė	

Kadangi abu modeliai turi savų privalumų bei neigiamų savybių, yra sudėtinga vienareikšmiškai pasakyti, kokios transakcijos turėtų būti naudojamos vienoje ar kitoje verslo PĮ. Gairės, renkantis transakcijų mechanizmą PĮ galėtų būti tokios [Cha98]:

- modelio pasirinkimas yra įtakojamas technologijos, kurią naudoja PĮ. Taigi, kaip pagrindiniu kriterijumi renkantis transakcijų modelį galime laikyti galimybę įgyvendinti pasirinktą transakcijų modelį, naudojant technologijas, kuriomis yra sukurta PĮ;
- jei PĮ, kuri pradeda transakciją turi laukti atsakymo iš serverio tam, kad galėtų pratęsti savo pačios darbą, geriausia naudoti išskirstytas transakcijas, todėl kad šias transakcijas yra lengviau panaudoti lyginant su eilių transakcijomis. Tačiau, jei PĮ, pradedanti transakciją, nereikalauja staigaus atsakymo iš gavėjo ir gali atlikti toliau savo darbus, geriausia rinktis eilių transakcijas;
- jei keli serveriai gali dažnai „nulūžti“, būti nepasiekiamais ar tarp PĮ yra lėtas susisiekimo ryšys, patariama naudoti eilių transakcijų modelį;
- jei egzistuoja keletas serverių, kurie sėkmingai gali susitvarkyti su transakcijomis, geriau naudoti eilių transakcijų modelį, kadangi apkrovos balansas lengviau valdomas ir diegimas panaudojant būtent šį transakcijos modelį;
- jei sunkiai sekasi realizuoti PĮ grąžinimą į pradinę būseną pieš atliekant transakciją, vertėtų susimąstyti apie išskirstytų transakcijų modelį;
- atliekant verslo PĮ atnaujinimo darbus, kartais gali pasitaikyti tokių situacijų, kai vienas PĮ komponentas naudos išskirstytas transakcijas, o kitas eilių transakcijų mechanizmą, nes abu modeliai turi savo privalumų.

1.10. Senos PĮ pakeitimas

PĮ pakeitimas reikalauja naujai sukurti PĮ. Senos PĮ pakeitimas atliekamas tokiu atveju, kai atnaujinimo darbų kaštai yra didesni nei pakeitimo kaštai [BJG+97]. Pakeitimo praktika yra taikoma tik tuo atveju, kai nepadeda nei baltos, nei juodos dėžės modernizavimo metodai.

PĮ gali būti pakeičiama visa iš karto arba palaipsniui. PĮ yra keičiama palaipsniui tada, kai ji turi labai glaudžius ryšius ir stipriai susijusi su kita išorine PĮ. Keičiant esamą PĮ nauja, turi būti įvertintos tokios rizikos:

- PĮ prižiūrintis personalas gali sunkiai adaptuotis prie naujos PĮ bei naujų technologijų;
- atliekant senos PĮ keitimą, yra reikalaujama atlikti nuodugnesnius testus lyginant su senos PĮ modernizavimo testavimu, kadangi sena PĮ dažniausiai jau būna gerai ištestuota ir gerai priderinta prie esamų verslo poreikių;

- nėra garantijų, kad nauja PĮ bus lankstesnė ar geriau funkcionuos už seną PĮ.

Atliekant senos verslo PĮ atnaujinimo darbus būtina įvertinti riziką. Tai tokia PĮ inžinerijos praktika, kurią atliekant galima pamatuoti, kas gali nutikti blogai, kokie rizikos veiksniai yra svarbūs ir kokių veiksmų reikia imtis norint juos sumažinti. Rizikos valdymas yra vienas iš spiralinio PĮ vystymo modelio, kurį pristatė Barry Boehm [Boe88], elementų.

1.11. Metrikos

Modernizuojant PĮ, metrikos yra naudojamos tam, kad pamatuotume, kaip išaugo verslo nauda, sparta ar vartotojų pasitenkinimas. Kadangi vadovybė dažniausiai nori matyti tik skaičius tai galimos metrikos pavaizduotos 4 lentelėje.

4 lentelė. Tikslų – metrikų lentelė [SPL03]

Tikslas	Galima metrika
Mažesnė priežiūros kaina	Vidutinis laiko tarpas problemos uždarymui; Mažesnis darbuotojų kiekis;
Naujo funkcionalumo pridėjimas	Naujų funkcijų kiekis.
Padidintas našumas	Užbaigtų operacijų kiekis per nustatytą laiką.
Perkodavimas nauja programavimo kalba	Skirtingomis kalbomis parašytų modulių kiekis.
Egzistuojančių objektų panaudojimas	Objektų skaičius naudojamas kituose produktuose.
Kelių skirtingų PĮ apjungimas	Naudotojų apmokymo laikas.

Svarbu žinoti, kad visi šie pirminiai metrikų įvertinimai yra apytikriai ir dažniausiai naudojami tam, kad būtų gautas apmokėjimas iš finansuotojo (užsakovo). Vėliau, modernizavimo eigoje, gali prireikti tikslesnių vertinimų, paskaičiavimų.

Kadangi priežiūros standartai ir modeliai konkrečiai neapibrėžia ir nėra aišku kaip konkrečiu atveju reikia atlikti senos verslo PĮ priežiūrą, atnaujinimą, transformaciją bei migraciją, toliau darbe nagrinėjama kaip tą galima atlikti.

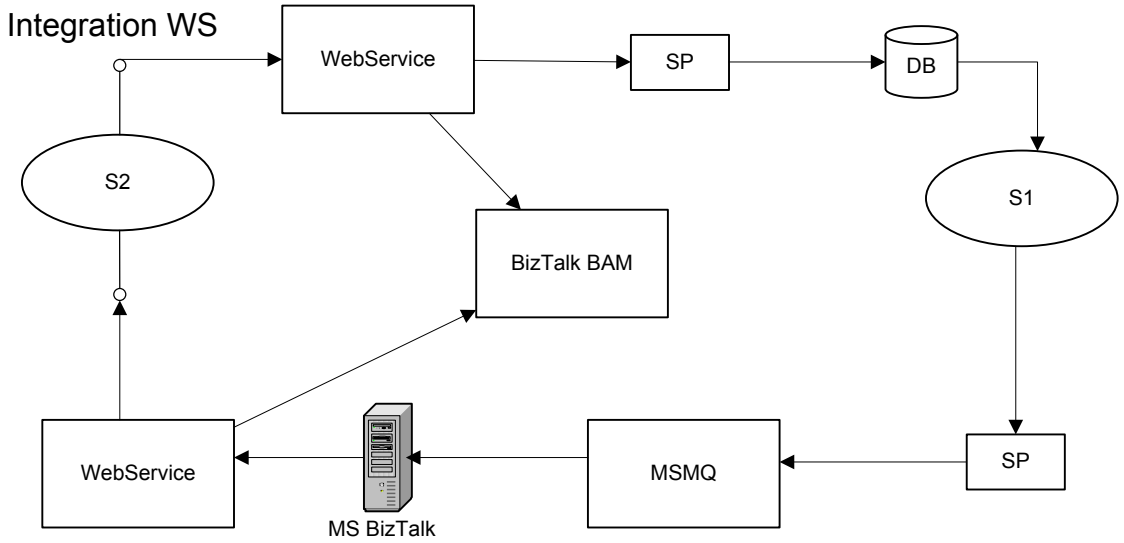
2. Senos PĮ atnaujinimas

2.1. Biztalk panaudojimas PĮ modernizacijoje

Dažniausiai sena verslo PĮ naudoja daug papildomų, trečiųjų šalių PĮ, tokių kaip CRM ir užsakymų valdymų. Ši PĮ dažniausiai naudoja skirtingas technologijas pradedant COBOL ir baigiant C# bei Java, ir veikia skirtingose operacinėse sistemose. Tuo pačiu metu daugelis procesų yra priklausomi nuo žmogaus veiksmų, tokių kaip telefono skambučiai, faksai, elektroniniai laiškai. Tampa labai sudėtinga stebėti, kas vyksta verslo PĮ einamuoju metu tokioje sudėtingoje aplinkoje. Didėjant verslo reikalavimams ir plečiantis rinkai, atsiranda būtinybė adaptuotis prie pakitusios aplinkos ir priimti teisingus sprendimus. Pasinaudojant BAM galimybėmis, lengvai galima stebėti kas vyksta PĮ, reaguoti į situaciją pasikeitus PĮ būsenai, stebėti visą veiksmo procesą nuo pradžios iki galo.

Tam, kad įsitikintume, kad BizTalk gali būti naudojamas senų verslo PĮ priežiūrai atlikti, panagrinėkime pavyzdį. Turime seną verslo valdymo PĮ, kuri apdoroja duomenis apie klientų užsakymus. Ši PĮ (S1) yra sunkiai palaikoma, reikalauja daug darbuotojų įsikišimo, todėl nuspręsta šią PĮ atnaujinti iš pagrindų. Kadangi tai yra svarbi PĮ visai organizacijai ir be jos organizacija negalėtų vykdyti savo veiklos, PĮ atnaujinimai turės būti diegiami palaipsniui, žingsnis po žingsnio, kad galutiniai PĮ vartotojai nepajustų staigaus pasikeitimo. Kliento komunikacija su PĮ S1 vyksta per grafinį vartotojo interfeisą. Kadangi S1 neturi galimybės stebėti, kaip gi vyksta ir kokioje būsenoje yra kliento užsakymas, užsakyta paslauga, perduodamas apmokėjimas, reikia prie šios PĮ prijungti komponentą, kuris atliktų visus PĮ monitorinimo (stebėjimo) darbus. PĮ (S2), kuri iš esmės atliks tą patį ką ir sena PĮ S1, tačiau bus našesnė ir efektyviau, be darbuotojų papildomo įsikišimo, vykdys procesus. PĮ S1 su išorine PĮ bendrauja tik duomenų bazės lygyje, todėl komunikacijai reikia sukurti procedūras, laikinas lenteles, kuriose bus laikomi tiek įeinantys, tiek išeinantys duomenys. S2 su išorine PĮ bendrauja per integruotą servisą. Kadangi perėjimas nuo S1 prie S2 reikalauja laiko, todėl tam tikrą laiką, kol nebus pabaigtas migravimo projektas, S1 ir S2 veiks lygiagrečiu režimu, tai yra, suvedus užsakymą S1, užsakymas atsispindės ir S2, ir atvirkščiai. S1 ir S2 nėra identiškos, todėl tam, kad konvertuotume duomenis iš S2 į S1, panaudosime tarpinius web servisus, kurie atliks konvertavimo darbus ir savyje laikys visą papildomų duomenų ištraukimo ir apjungimo logiką. Komunikacijai iš kitos pusės, iš S1 į S2, galime panaudoti Biztalk serverį, kuriuo lengvai bus galima suintegruoti abi S1 ir S2 sistemas, stebėti, kaip vyksta užsakymo procesai, kokia yra S1 būsena. Iš

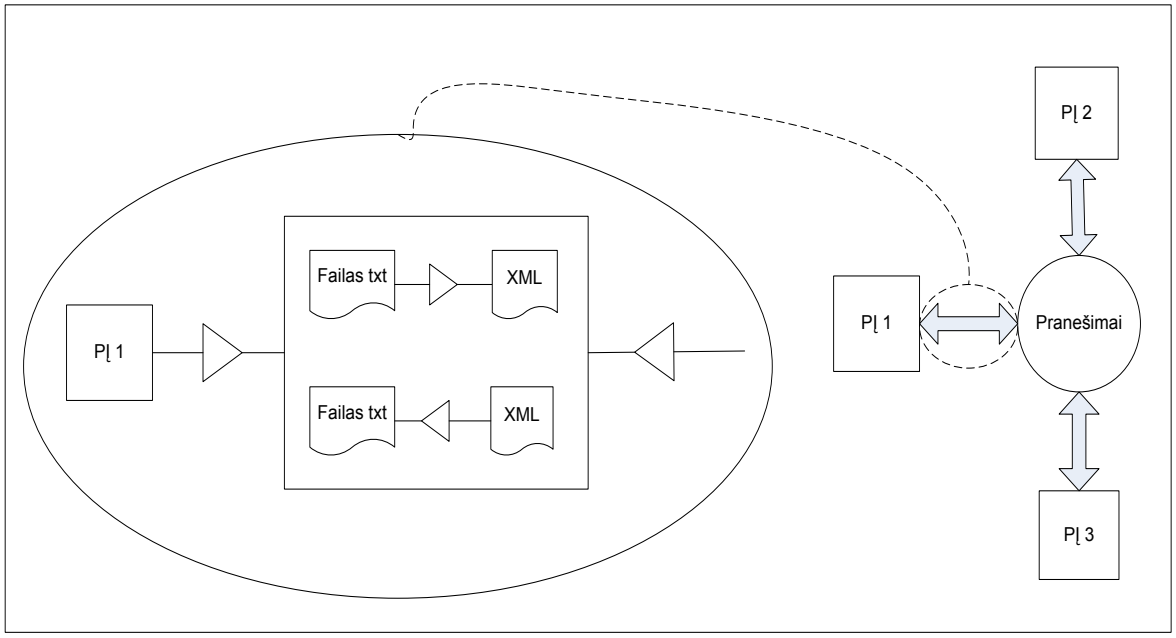
esmės, pastatydami Biztalk kaip tarpininką tarp S1 ir S2, mes galima laimėti tiek PĮ integracijoje, tiek PĮ stebėjime. Galimas tokios situacijos sprendimas pavaizduotas 20 pav.



20 pav. PĮ integracinė ir monitorinimo architektūra

Komponentas BizTalk BAM būtų naudojamas kaip PĮ stebėjimo taškas, kuriame atsispindėtų tiek sistemos S1 tiek S2 būsenos. Kada S2 bus pilnai paruošta naudojimui, tada bus galima visiškai atsisakyti S1, papildomų servisų, adapterių, trigerių.

Kadangi dauguma senų PĮ duomenų įvedimui ir išvedimui naudoja failus, kuriuose duomenys yra atskirti kableliais, brūkšniais ar kokiu kitu skiriamuoju ženklu, tai tam, kad apjungtume tokias sistemas su kita, naujasne PĮ, reikalingi adapteriai. Toks adapteris galėtų atrodyti taip kaip pavaizduota 21 pav. Į adapterį ateina kableliais ar kitais simboliais atskirti duomenys faile, o išeina XML tipo failas su perduodamais duomenimis. Analogiškai vyktų ir su XML tipo failais: atėjus tokiam failui, jis konvertuojamas į failą, kuriame duomenys atskiriami specifiniais skiriamaisiais ženklais.



21 pav. PĮ apjungimo schema panaudojant adapterį

2.2. Transformavimo strategija

Kai jau yra išsiaiškinta tiek senos verslo PĮ, tiek atnaujintos PĮ architektūra, galima pradėti galvoti apie tai, kaip bus pereinama nuo senos architektūros prie naujosios. Architektūros transformavimas yra pagrindinis tikslas, atnaujinant seną verslo PĮ. Atliekant architektūros transformaciją, dažniausiai sena PĮ yra skaidoma į gabalus, mažesnes dalis, kurias galima papildomai pertvarkyti ir paruošti perkėlimui. Iš esmės yra atliekama senos PĮ dekompozicija, išardymas. Vienas iš sudėtingiausių dalykų atliekant tokią architektūros transformaciją, yra suskaldytų komponentų ribų valdymas, tai yra, reikia griežtai kontroliuoti, iki kokio lygio yra skaidoma sena PĮ, ar suskaldyti „gabalai“ nebus per didelis, ar per maži. Tam, kad sujungtume kelias dalis į vieną visumą, yra naudojami adapteriai.

2.2.1. Architektūros transformacijos

Šios transformacijos yra aukščiausio lygio transformacijos, kadangi atkuriama visa esama PĮ architektūra. Kai pilnai yra atkuriama PĮ architektūra, galima pradėti spartos didinimo, galimybės modifikuoti, saugumo, atsparumo ar kitas analizes. Šios transformacijos metu, esamai PĮ architektūrai yra atliekama reinžinerija bei įvertinama, kokie yra PĮ kokybės tikslai. Tobulinimo proceso metu, architektūra yra peržiūrima, paruošiamas atnaujinimo planas ir realizuojamas programinis kodas. Toks tobulinimo procesas yra labai panašus į naujos PĮ kūrimo procesą, tik esminis skirtumas yra tas, kad reikia atsižvelgti į esamos senos PĮ įvairius apribojimus. Svarbu tiksliai identifikuoti senos PĮ komponentus ir tiksliai žinoti, kur jų vieta bus modernizuotoje PĮ. Jei modernizuojant PĮ

nusprendžiama, kad reikės panaudoti jau esamus senos PĮ komponentus, yra svarbu išsiaiškinti pilnai tokių komponentų funkcionalumą. Esant reikalui, galimos ir šių komponentų modifikacijos. Tam, kad būtų atkurta senos PĮ architektūrą, reikalinga nuodugni senos PĮ analizė. Tokio proceso rezultatas – PĮ architektūros vaizdas. Tokį architektūrinį vaizdą naudoja programuotojai, testuotojai, architektai.

Architektūriniai sprendimai nėra tiesiogiai vaizduojami ir pastebimi programiniame kode – jie yra suvokiami kaip atskirų komponentų sujungimas. Tais komponentais gali būti funkcijos, klasės, failai, objektai.

PĮ architektūra vaidina pagrindinį vaidmenį kuriant PĮ, o ypač atnaujinant-modernizuojant seną verslo PĮ. Dažniausiai sutinkamas atvejis, kai modernizuojant verslo PĮ, yra sudaromos kelios architektūros. Iš kelių architektūrų yra pasirenkama tinkamiausia pagal situaciją (sudėtingumą, PĮ spartą, realizacijos laiką). Vykdant nuoseklią modernizaciją, kiekviename žingsnyje (etape) yra sudaroma architektūra, kuria vadovaujamasi toliau modernizuojant PĮ. Kiekvieno etapo pradžioje yra lyginamos sena ir naujoji architektūra, žiūrima, ką reiktų tobulinti, ką išmesti, ką pakeisti. Atitinkamai atnaujinama ir architektūros dokumentacija, kurioje atsispindėtų naujausi PĮ pasikeitimai.

PĮ architektūra yra pagrindinis mokymosi dokumentas. Kadangi PĮ architektūra aprašo visą PĮ vaizdą, iš ko ji sudaryta, kaip ji sąveikauja su išore, tai ji puikiai tarnauja kaip edukacinė priemonė, pavyzdžiui, atėjus naujam darbuotojui į komandą. Taip pat, architektūra naudojama, kai bendraujama tarp PĮ priežiūros personalo, architektų, programuotojų, testuotojų ir kitų su modernizuojama PĮ susijusiu personalu [ULR02].

2.2.2. Architektūros transformavimo strategija

Vienas iš būdų, kaip parengti transformavimo strategiją, yra atsakyti į klausimus:

- Kaip bus migruojamas programinis kodas?
- Kada bus migruojami duomenys? Prieš kodą ar po?
- Ar bus reikalingos sinchroniškai, paraleliai veikiančios operacijos?

5 lentelėje pavaizduoti galimi atsakymai į šiuos klausimus. Pasirenkant atsakymus į pateiktus klausimus yra formuojami transformavimo strategijos pagrindai. Žinoma, atsakymai turi būti priimami bendrai, nes reikia atsižvelgti į visų (programuotojų, vadovų, architektų ir k.t.) nuomonę formuojant transformavimo strategiją.

5 lentelė. Iteracinio modernizavimo pasirinkimai

Klausimas	Galimas pasirinkimas
Kodo migracija	Paremtas vartotojo transakcijomis

	Paremtas funkcijomis
Duomenų migracija	Prieš kodo migraciją Tuo pat metu kaip migruojamas kodas Po kodo migracijos

2.2.2.1. Kodo migracija

Kodas gali būti dalinamas ir migruojamas keliais būdais. Vienas iš būdų, perkelti pagrindinę programos giją naudojantį kodą. Tai sudėtingas uždavinys, nes reikalauja PĮ suprasti baltos dėžės principu.

Galimas kitas migracijos būdas, tai migruoti visus programos elementus panaudojant juodos dėžės principą.

Naudotojo transakcijos (angl. *user transactions*) – vartotojo arba išorinės PĮ užklausos, kurios atliekamos per eilę programinių elementų – gali būti panaudojamos kaip funkcionalumas, kurį reikia sumigruoti. Šios transakcijos taip pat gali būti panaudojamos kaip įeities parametrai atliekant tiek statinę, tiek dinaminę analizę. Stebėjimo programos gali identifikuoti programos elementus, kodo vietas, kurios yra vykdomos atliekant transakcijas. Tačiau tokios programos negarantuoja, kad bus parodytas visas transakcijos kodas. Dažniausiai trūksta informacijos tada, kai kode yra klaidų gaudymas (*exception handlling*). Migruojant transakcijomis paremtą kodą, privalumas yra tas, kad atnaujintoje PĮ vartotojas gali jau atlikti tam tikras transakcijas.

Tokios migracijos trūkumas yra tai, kad vienu metu gali tekti migruoti labai didelius kiekius kodo, o tai labai sudėtingas arba net neįmanomas uždavinys.

2.2.2.2. Duomenų migracija

Duomenų migravimo metu nėra garantijos, kad lengvai bus apjungiami seni ir nauji duomenys: duomenų bazių lentelės gali būti padalintos arba sugrupuotos, naujos sukurtos, senos ištrintos. Tokie pertvarkymai komplikuoja duomenų apsikeitimą tarp sistemų.

Yra trys duomenų migracijos pasirinkimai: prieš, tuo pat metu arba po kodo migracijos.

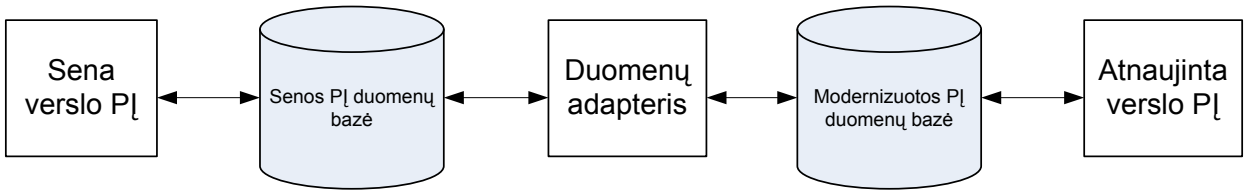
- **Duomenų migravimas prieš kodo migravimą.** Atliekant pirmiau duomenų migraciją, žymiai supaprastėja kodo migracija. Kodas gali būti plečiamas jau remiantis naujais duomenimis, nereikalauja senų duomenų apjungimo. Taip pat sumažėja rizika, kad sena PĮ nustos funkcionuoti, o duomenys bus sugadinti. Tokios migracijos neigiamomis savybėmis galima įvardinti senos PĮ duomenų lygio pertvarkymus, kuriuos reikia atlikti norint migruoti duomenis iš senos PĮ į naująją. Taip pat laiko bei resursų padidėjimas bus žymiai didesnis

atliekant pakeitimus senoje PĮ. Migruoti duomenis prieš kodą pasiteisina tik tuomet, jei turima didelė patirtis ir geras senos PĮ supratimas.

- **Duomenų ir kodo migracija**. Migruoti duomenis ir kodą vienu metu atrodo pats tiesiausias ir aiškiausias kelias. Teoriškai, tai pats pigiausias variantas, nes reikalauja mažiausiai pertvarkymų. Tačiau tai tik teorija. Praktika rodo, kad sunkiausia yra tais atvejais, kada sunku atskirti duomenų arba kodo elementą iš senos sistemos. Atliekant tokį žingsnį dažniausia išauga modernizacijos kaštai net gi keliais kartais. Tam, kad apsisaugotume nuo neplanuotų tokio žingsnio padarinių, galime migraciją atlikti mažais žingsniais: pasiimti keletą duomenų lentelių, jas struktūrizuoti modernizuotoje duomenų bazėje, o tada iškarti imtis kodo, kuris naudoja šias lenteles, migracijos. Nereiktų pamiršti tai, kad gali prireikti ir papildomo duomenų adapterio.
- **Duomenų migracija po kodo migracijos**. Šis migravimo variantas reikalauja kurti atnaujintos PĮ komponentus, atitinkančius senos PĮ duomenų struktūrą. Taip pat gali prireikti duomenų ir kodo adapterių, kurie lėtina visos PĮ darbą.

2.2.3. Duomenų adapteriai

Duomenų adapteriai yra reikalingi tuomet, kada programinis kodas ir duomenys yra migruojami vienu metu, sinchroniškai. Šie adapteriai sinchronizuoja duomenis tarp senos ir naujos verslo PĮ duomenų bazių jungdami seną duomenų bazės schemą su nauja (pav. 22).



22 pav. Duomenų adapteris

2.2.4. Duomenų replikacija

Duomenų replikavimas – tai procesas, kurio metu duomenų bazės lentelės yra kopijuojamos ir tvarkomos keliose duomenų bazėse. Tuomet, kai viena duomenų bazė tampa neprieinama, naudotojai toliau gali atlikti užklausas ar vykdyti duomenų atnaujinimo operacijas kitose bazėse.

Tam, kad galėtume efektyviau atlikti duomenų replikavimą, galime naudoti skriptus, įrankius ETL (*extraction, transformation and loading*), duomenų bazės trigerius.

2.2.5. Skriptai

Skriptai – vieni iš paprasčiausių būdų replikuotų duomenų sinchronizavimui. Panaudojant skriptus, duomenys gali būtų transformuojami iš vienos PĮ struktūros į kitos PĮ struktūrą. Skriptai dažniausiai yra paleidžiami darbui periodiškai arba rankiniu būdu, taip pat skriptai dažniausiai nereikalauja ypatingų pakeitimų PĮ.

Skriptai turi ir trūkumų, pavyzdžiui, gali būti sudėtinga nustatyti, kokie buvo atlikti duomenų pakeitimai yra naujausi sinchronizuojant duomenis kelioms sistemoms vienu metu. Galimas sprendimas – panaudoti laiko žymes (*timestamp*).

2.2.6. ETL

ETL (*Extraction, Transformation and Loading*) – tai įrankiai, kurie skirti duomenų ištraukimui, duomenų konvertavimui (pavertimui) į reikiamą formatą naudojant įvairias taisykles, kombinuojant duomenis, ir patalpinimui kitoje duomenų bazėje. ETL įrankiai gali būti naudojamas atskirai, kaip vienkartinėms operacijoms atlikti, bei būti viduriniąja grandimi tarp PĮ klientinės dalies ir duomenų bazės. Nors pagrindinė šių įrankių užduotis yra duomenų talpinimas, tačiau ETL įrankiai puikiai gali būti panaudojami ir duomenų replikavimo procese tarp senos ir modernios verslo PĮ. Kai kurie šios klasės įrankiai suteikia galimybę duomenų apsikeitimui į abi puses, tai yra, tiek iš senos PĮ į atnaujintą, tiek iš atnaujintos į seną. Nors šie įrankiai gali puikiai tarnauti duomenų replikavimui, tačiau tai yra papildomas komponentas PĮ, kuris verčia PĮ sudėtingesne. Taip pat kai kurie ETL įrankiai gali būti nesuderinami su sena PĮ.

2.2.7. Duomenų bazių trigeriai

Duomenų bazių trigeriai – tai kodas, kuris yra vykdomas duomenų bazėje prieš prasidedant ar pasibaigus kokiam nors įvykiui. Trigeriai gali būti panaudojami sinchronizuojant duomenis tarp senos ir naujosios duomenų bazių. Atliekant sinchronizaciją panaudojant trigerius, trigeriai pradeda veikti po duomenų atnaujinimo (*post-update*) arba prieš duomenų įkėlimą (*post-insert*). Trigeriuose užprogramuota logika sinchroniškai arba asinchroniškai iššaukia pakeitimus kitoje PĮ. Trigeriai gali būti suprogramuoti tiek senoje PĮ, tiek naujoje, arba abiejose kartu. Naudojant trigerius, privalumas yra tas, kad duomenys yra keičiami iš karto (tiesiogiai) po kokio nors įvykio. Trigeriai gali būti paprasčiausiai atjungti, kai sena duomenų bazė migruojama ir nebenaudojama. Trigerių neigiama savybe galėtume laikyti tai, kad juos naudojant yra papildomai apkraunama visa duomenų bazių valdymo sistema. Taip pat gali būti sudėtinga atlikti duomenų apjungimą tarp dviejų duomenų bazių, ypač jei duomenų struktūra turi būti pakeista vienoje iš duomenų bazių arba duomenų bazės yra skirtingų tipų.

Naudoti trigerius kaip adapterius yra patogiausia tuomet, kai sena ir naujoji duomenų bazės priklauso tai pačiai duomenų bazių valdymo PĮ. Tokiu atveju apjungimo taisyklės yra duomenų bazėje , o ne kažkur papildomuose įrankiuose. Tokiu būdų padidėja efektyvumas. Jei duomenų bazės priklauso skirtingoms duomenų bazių valdymo PĮ, trigerių pagalba galima iškviesti skriptus arba ETL įrankius.

2.2.8. Duomenų priėjimo sluoksnis

Duomenų priėjimo sluoksnis – tai duomenų vaizdavimo ir pateikimo tam tikru formatu būdas klientinei programai. Pavyzdžiui, duomenų priėjimo sluoksnis klientinei programai gali pateikti tinklinę duomenų bazę kaip releacinę arba atvirkščiai.

Duomenų sluoksnio privalumas yra tas, kad esamiems duomenims senoje PĮ nereikalinga duomenų replikacija, nes visos duomenų operacijos atliekamas per senos PĮ duomenų vaizdą. Nors dauguma duomenų priėjimo sluoksnio įrankių suteikia tik skaitymo priėjimą (*read-only*), tačiau yra ir tokių, kurie leidžia atlikti rašymo veiksmą į seną duomenų bazę.

Šio sluoksnio trūkumu galėtume laikyti tai, kad į vieną duomenų elementą yra prieinama iš skirtingų taškų. Tokio priėjimo rezultatas – duomenų integralumo pažeidimas.

Duomenų priėjimo sluoksnis turėtų būti naudojamas tik kaip vienos iteracijos sprendimas atliekant PĮ modernizaciją, nes hierarchinės ir releacinės duomenų bazių apjungimas yra neoptimalus dėl pernelyg didelio šių duomenų bazių tipų skirtumų.

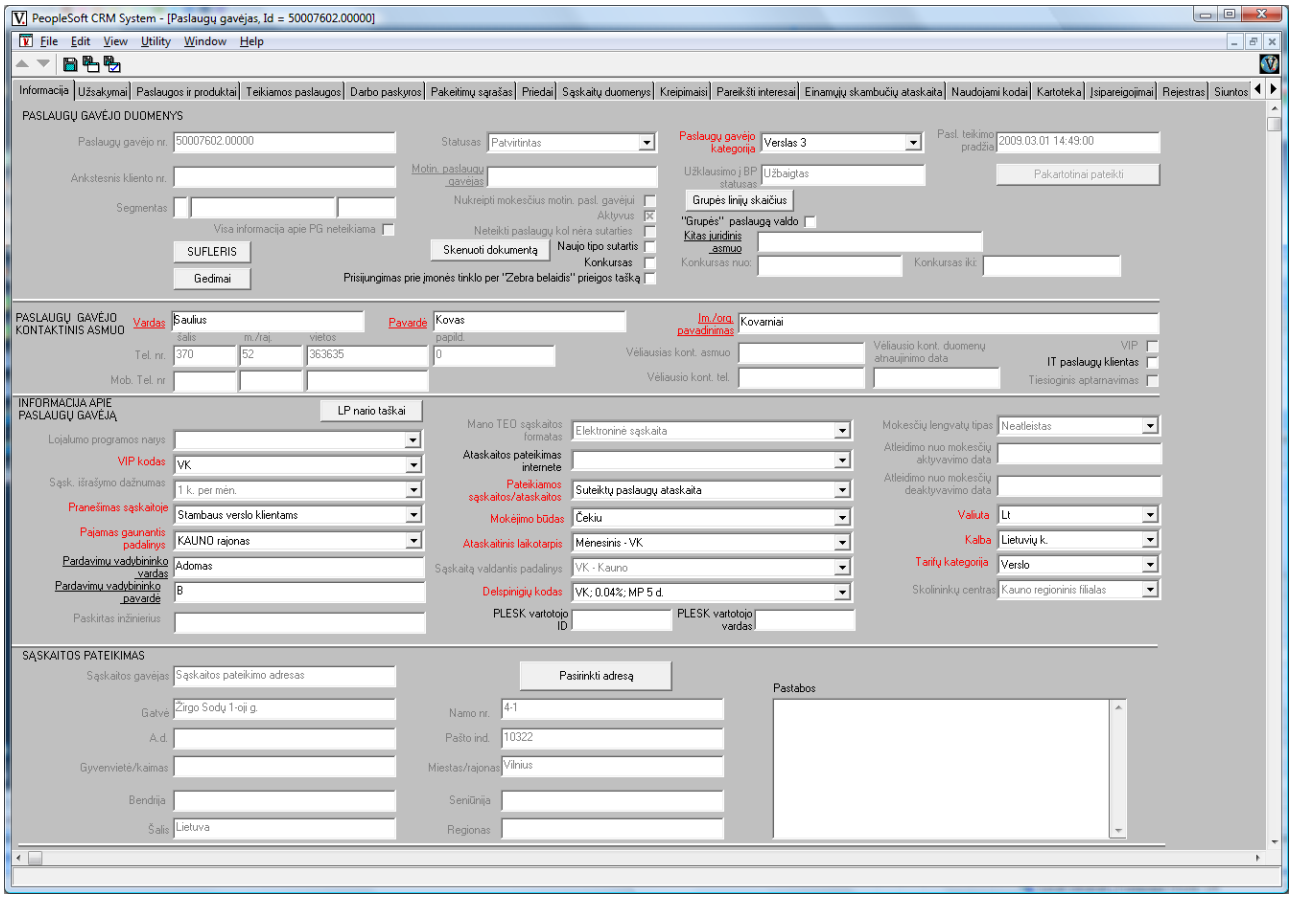
6 lentelė. Duomenų apjungimo būdų palyginimas

Būdas	Rezultatas	Privalumai	Trūkumai
Skriptai	Sinchronizuoja replikuojamus duomenis	Paprasta įdiegti; nereikalauja didelių pertvarkymų	Sudėtingą prižiūrėti duomenų susijungimą netrivialiais atvejais
ETL	Sinchronizuoja replikuojamus duomenis	Įrankių palaikymas	Galimas neefektyvumas; sudėtinga stebėti; įrankių nepakankamumas
Duomenų bazių trigeriai	Sinchronizuoja replikuojamus duomenis	Tai nėra papildomas sistemos komponentas; lengvas senos sistemos išjungimas	Papildomai apkraunama duomenų bazių valdymo sistema; sudėtinga įgyvendinti
Duomenų priėjimo	Senus duomenis	Nereikalingas	Neefektyvu; du

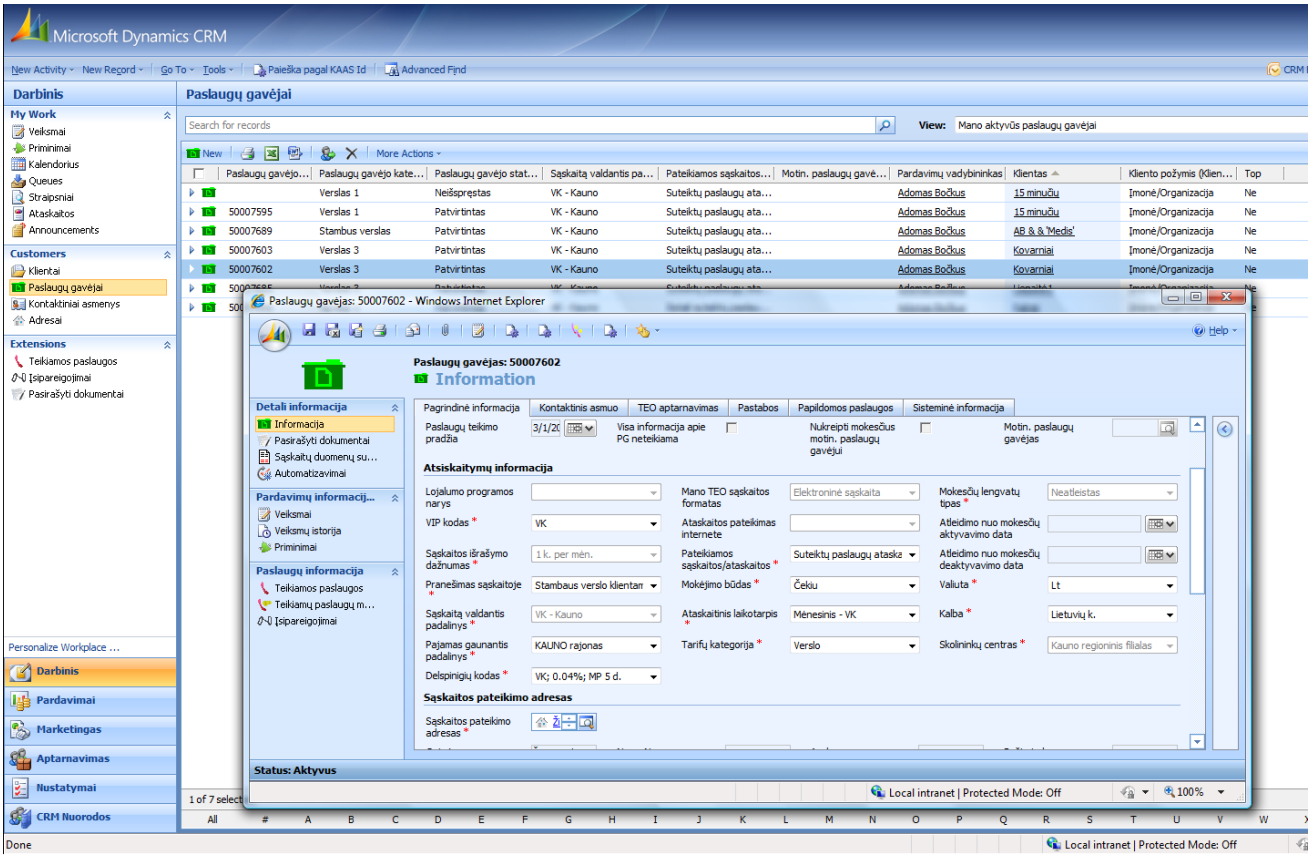
sluoksniai	vaizduoja reikiamu formatu	duomenų replikavimas	duomenų priėjimo taškai

2.3. Praktinis pritaikymas

20 pav. pavaizduotos schemos praktiniam įgyvendinimui, S1 PĮ buvo „pastatyta“ ant MS SQL pagrindo ir visa S1 logika suprogramuota procedūrose, o vartotojo grafinis interfeisas – Vantive 8.2 (23 pav.) (toliau sistema S1 vadinama tiesiog Vantive). Sistemai S2 pasirinkta MS Dynamics CRM platforma (24 pav.). Kadangi Vantive norima atsisakyti, duomenis, esančius senoje Vantive sistemoje reikia perkelti į naująją MS Dynamics CRM (toliau sistema S2 vadinama CRM).



23 pav. Vantive 8.2 langas su kliento informacija



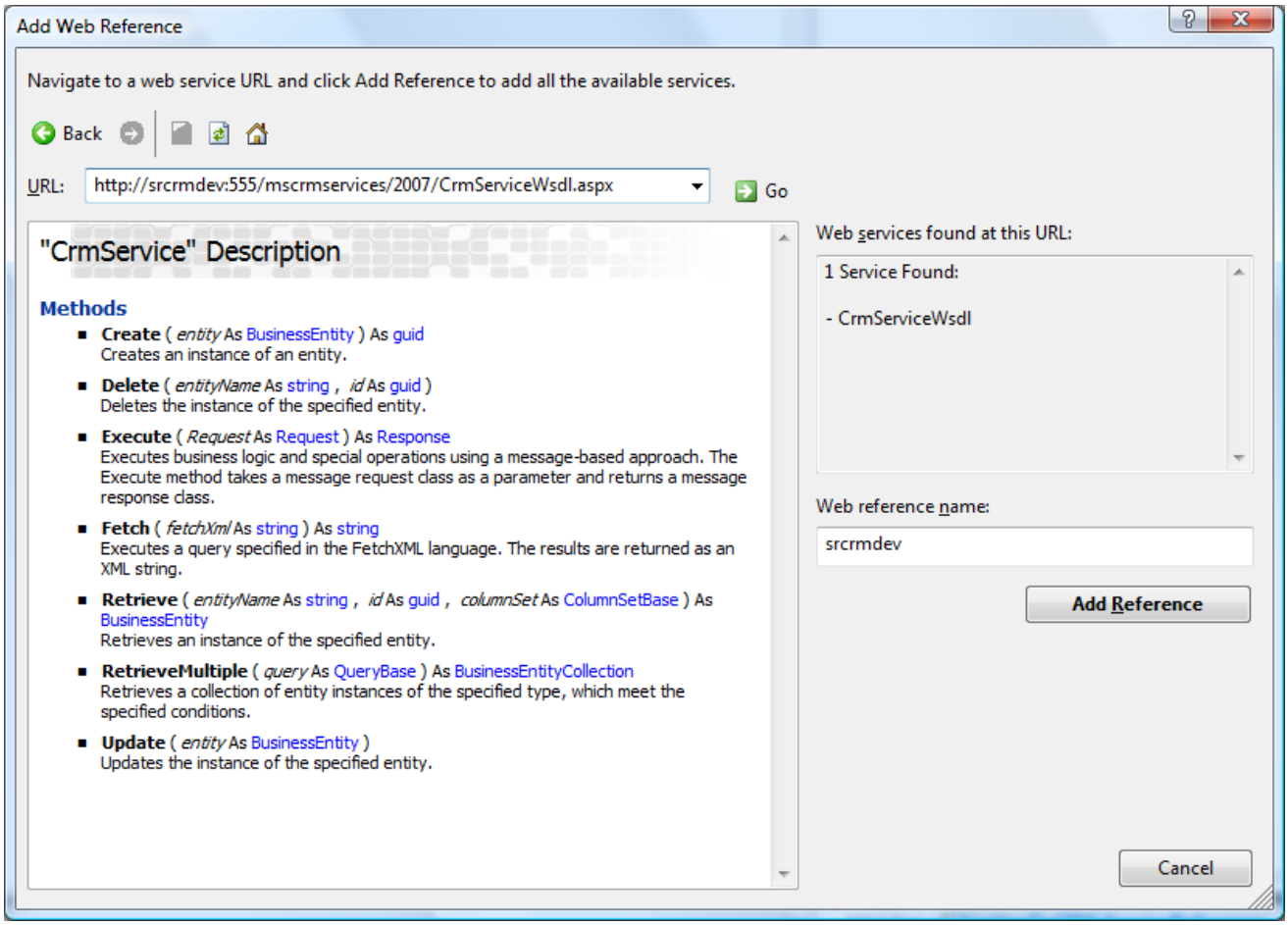
24 pav. MS Dynamics CRM langas su kliento informacija

Duomenų perkėlimui buvo pasirinkta kliento (*angl. Account*) informacija. Vantive sukuriama procedūra *ExtractAccountInfo* (*stored procedure*), kuri grąžina visą perkėlimui reikalingą informaciją apie konkretų klientą. Taip pat sukuriama integracinė lentelė *LTMesages_Old*, kurioje bus saugomi įrašai apie Vantive sistemoje įvykusius veiksmus, tokius kaip pavyzdžiui naujo kliento įvedimas, esamo kliento modifikavimas arba esamo kliento ištrynimas iš sistemos. Sukuriami ir trigeriai (*post-update*), kurie bus atsakingi už kliento informacijos patalpinimą lentelėje *LTMesages_Old* bei MSMQ eilėje. Kai tik Vantive įvyksta veiksmas, trigeris paima kliento kodą, prideda koks tai buvo veiksmas (insert – I, update – U, delete - D) ir patalpina į lentelę LTMessages_Old bei padeda į MSMQ eilę. Eilė pasirinkta tam, kad būtų išlaikytas įvykių eiliškumas, kad neįvyktų taip, kad įvykus dviems įvykiams vienam po kito Vantive sistemoje, susimaišytų įvykių eiliškumas.

Pradedamas konfigūruoti BizTalk serveris. Jis nustatomas taip, kad imtų įrašus iš MSMQ eilės, paverstų paimtą įrašą į XML pranešimą, kurį panaudotų kviesdamas .NET pagrindu parašytą web servisą VantiveToCRM (pavadintas dėl aiškumo – duomenų perkėlimui iš Vantive į CRM).

Suprogramuojamas web servisas VantiveToCRM. Šis web servisas atliks adapterio funkciją, duomenų konvertavimą. Iš anksto yra žinoma kokios duomenų struktūros reikės naujajai CRM

sistemai, todėl šis web servisas išsitraukia duomenis tiesiogiai iš Vantive sistemos panaudodams ExtractAccountInfo procedūrą, „sukelia“ gautą kliento informaciją į CRM kliento objektą ir visą šį objektą perduoda į CRM sistemą per CRM sistemos integracinį servisą (25 pav.). CRM integracinis servisas yra skirtas komunikacijai su išorine PĮ. Per šį servisą galima tiesiogiai komunikuoti su MS Dynamics CRM ir taip valdyti sistemą.



25 pav. MS Dynamics CRM integracinis web servisas

Norint kurį laiką naudotis abiejomis sistemomis vienu metu, reikia kad jos dirbtų lygiagrečiai, tai yra atlikus veiksmus vienoje sistemoje, analogiški veiksmai turi atsispindėti ir kitoje sistemoje. Pavyzdžiui, atnaujinant kliento informaciją CRM sistemoje, kliento informacija turi atsivaizduoti ir Vantive. Šiam tikslui pasiekti sukuriamas dar vienas tarpinis .NET web servisas CRMtoVantive, kuris bus atsakingas už duomenų perkėlimą iš CRM į Vantive sistemą, bei procedūra, kuri perduodamus duomenis įrašytų į atitinkamas kliento lenteles Vantive sistemoje. CRMtoVantive web servisas yra iškviečiamas panaudojant CRM įskiepį (angl. Plug-in) ir gauna objektą iš CRM sistemos. Pagal tai, kokia operacija yra vykdoma, web servisas atitinkamai kviečia procedūras deleteAccount, InsertAccount, UpdateAccount Vantive sistemoje. Kadangi duomenų struktūra CRM ir Vantive

sistemose skiriasi, tai CRMtoVantive web servisas taip pat atlieka adapterio funkciją paversdamas CRM duomenis Vantive sistemai suprantamais duomenimis.

Tam, kad atnaujinant to paties kliento informaciją abiejose sistemose vienu metu naujausi pakeitimai būtų abiejose sistemose, CRM ir Vantive reikia sukurti po vieną versijos lauką, kuris bus tikrinamas atėjus pranešimams iš kitos sistemos: jei pranešimo versija yra didesnė nei esama sistemoje – veiksmai atliekami, jei mažesnė – veiksmai atšaukiami. Tokiu būdu abiejose sistemose informacija bus naujausia.

Kad būtų galima stebėti kaip duomenys vaikšto nuo vienos sistemos prie kitos, panaudojamas Biztalk BAM komponentas-portalas, kuriame matomas duomenų judėjimas tarp sistemų. Stebėjimo taškais pasirenkami CRMtoVantive bei VantiveToCRM web servisai, kadangi juose suprogramuota didžioji dalis logikos, o tai reiškia, kad klaidos tikimybė būtent šiuose web servisuose yra didžiausia.

```
DirectEventStream d = new DirectEventStream(ConfigurationManager.AppSettings["BAMconnection"], 1);
d.BeginActivity(ConfigurationManager.AppSettings["ActivityName"], g);
d.UpdateActivity(ConfigurationManager.AppSettings["ActivityName"], g,
                    "TrackingID", 1122,
                    "OpportunityID", "80602751-9329-DE11-BDE0-0017A477040C",
                    "ProcessStart", 2009-12-11
                    );
```

26 pav. Tiesioginis BAM iškvietimas iš web serviso

Web servisai yra suprogramuoti ir tokiai situacijai, kad jeigu kliento duomenų nėra vienoje iš sistemų, jie pirmiau yra sukuriami. Tokiu būdu, naudojant sistemas gana ilgą laiko tarpą, duomenys patys migruoja nereikalaudami skirti papildomai laiko atskirai duomenų migracijai.

Analogiški veiksmai vykdomi ir su kitais duomenimis: užsakymais, adresais, ataskaitomis. Vantive sistemos atsisakymas vykdomas panaikinant tarpinius web servisus (VantiveToCRM ir CRMtoVantive) bei CRM sistemos įskiepių atsisakymu.

Ši sprendimo schema buvo išbandyta ir patikrinta su realiais duomenimis įmonėje „UAB TEO LT“. Sprendžiant iš vartotojų, PĮ administratorių atsiliepimų bei vadovų, toks sprendimas yra gerai veikiantis, patikimas ir efektyvus.

2.4. Resursų skaičiavimas

Kai jau yra sudaryti kodo bei duomenų migravimo planai, galime paskaičiuoti modernizacijos kainą. Kai tik bus įvertinta kaina, bus galima spręsti, ar verta vykdyti verslo PĮ modernizaciją.

2.4.1. Kainos apskaičiavimai

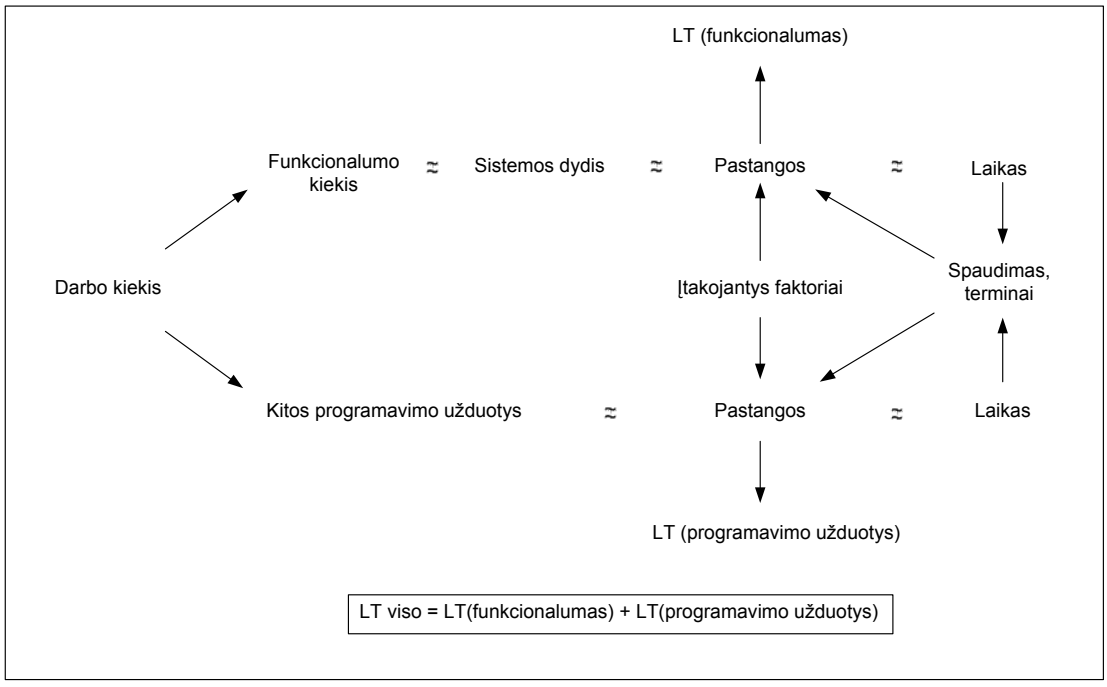
Pagrindinis kainos apskaičiavimo rodiklis yra darbas: kuo daugiau reikia padaryti, tuo daugiau tai kainuoja.

2.4.1.1. Funkcijomis paremtas paskaičiavimas

27 paveikle matome, kad visa PĮ atnaujinimo kaina yra proporcinga funkcionalumui, kurį reikia suprogramuoti. Šis funkcionalumas gali būti naujas, panaudojamas senas arba keičiamas. Į darbo paskaičiavimą taip pat yra įtraukiama analizė bei kodavimo darbai reikalingi seno funkcionalumo migravimui į naująją platformą.

Funkcionalumas gali būti skaičiuojamas įvairiai. Skaičiuojant funkcionalumą, svarbūs yra reikalavimų, funkcinių taškų, panaudos atvejų ir scenarijų matai. Funkcionalumas taip pat gali būti matuojamas ir kodo eilutėmis. Pavyzdžiui, remiantis istoriniais arba esamais duomenimis, organizacija gali nuspręsti, kad vienas funkcinis taškas yra lygus 89 kodo eilutėms.

Dauguma resursų skaičiavimo modelių naudoja būtent kodo eilučių metriką. Projektavimo metrikos, tokios kaip vartotojo interfeisas, komponentai, gali būti taip pat matuojami kodo eilutėmis. Kadangi PĮ dydis ir darbai, reikalingi PĮ plėtoti, yra tiesiogiai proporcingi, pagal PĮ dydį galima nuspręsti, kiek pastangų ir darbuotojų reikės numatytiems darbams atlikti. Šis ryšys apibūdinamas kaip produktyvumo indeksas: programuotojo darbo laikas per dieną padalinta iš kodo eilučių skaičiaus.



27 pav. Darbo konvertavimas į laiką ir pinigus

Jeigu galutinė PĮ atnaujinimo kaina yra priimtina, galima pradėti PĮ atnaujinimo darbus.

2.4.2. PĮ dydžio paskaičiavimai

2.4.2.1. Atnaujintos PĮ dydis

Vienas iš būdų, kaip galima nustatyti, kokio dydžio bus atnaujinta PĮ, tai remtis senos PĮ dydžiu. Tam tikslui galima panaudoti konvertavimo indeksą, tai yra, imti senos PĮ kodo eilutę, padauginti iš konvertavimo indekso ir gauti atnaujintos PĮ dydį. Pavyzdžiui, 1 COBOL programinio kodo eilutė atitinka 0,58 Java kodo eilutės. Dauguma konvertavimo indeksų yra apytiksliai, todėl praktiškai labai dažnai pasitaiko nukrypimai nuo paskaičiuojamų duomenų, todėl atliekant modernizaciją etapais, po kiekvieno etapo reikėtų perskaičiuoti ir patikslinti konvertavimo indeksą.

Atnaujintos PĮ dydis taip pat priklauso nuo esamo biudžeto (turimų lėšų atnaujinimams atlikti) ir funkcionalumo kiekio, kurį reikia perkelti iš senos verslo PĮ į naująją.

2.4.2.2. Adapteriai

Modernizacijos dydis ženkliai išauga, jei yra naudojami adapteriai. Kiekvienas atnaujinimo etapas reikalauja kurti naujus adapterius bei atsisakyti jau nebereikalingų.

Vienas iš būdų kaip pamatuoti adapterių dydį, tai sukurti prototipą, sudarytą iš tokių adapterių, ir pamatuoti kiekvieno adapterio dydį atskirai. Tokie sukurti prototipai taip pat leidžia patikrinti adapterių produktyvumą, patikrina, ar jie bus naudingi migracijai.

2.4.2.3. Augimo faktorius

Paskutinis veiksnys, į kurį reikia atkreipti dėmesį atnaujinant PĮ, tai augimo faktorius. Augimo (plėtimosi) faktorių nulemia papildomai prisidedantys reikalavimai verslo procesuose. Jeigu augimo faktorius sudaro daugiau nei 25% modernizavimo etapo kodo, tai tikrai nėra sveikintinas rezultatas.

Rezultatai ir išvados

Teoriškai pats efektyviausias senos PĮ atnaujinimo būdas – visos PĮ keitimas iš karto. Tokiam atnaujinimui nereikia papildomų adapterių, PĮ jungiančių sąsajų, supaprastėja testavimas. Tačiau praktiškai atnaujinti PĮ vienu kartu yra neįmanoma, nes dažnai atsitinka taip, kad kol sukuriama nauja PĮ, esami reikalavimai pasikeičia ir tokia PĮ nebetinkama naudojimui. Atnaujinama PĮ siejasi su daug kitų sistemų, kas apsunkina modernizaciją. Taip pat yra labai sudėtinga atkurti senos PĮ funkcionalumą, kurį ji vykdė kelis ar keliasdešimt metų. Atnaujinant PĮ dalimis, sumažėja PĮ neveikimo rizika, greičiau pastebimi neatitikimai, klaidos, kurias dar galima ištaisyti. Modernizuojant verslo PĮ palaipsniui, atsiranda galimybė pasimokyti iš praeitų modernizavimo žingsnių, nebekartoti jau padarytų klaidų.

Komponentais paremta modernizavimo strategija ateityje leidžia žymiai efektyviau atlikti pakeitimus PĮ, pakeisti ar perkelti į ją modernesnius komponentus. Tai paprastas ir efektyvus būdas senos PĮ atnaujinimui.

Biztalk – tai įrankis išskirstytų sistemų apjungimui. Šis įrankis nekuria duomenų, jis juos perkelia. Jeigu yra reikalinga priemonė, galinti lengvai susitvarkyti su duomenų pernešimu iš vienos PĮ vienokio formato į kitą PĮ su kitokiu formatu panaudojant transakcijas, galinti stebėti, valdyti mainų procesą, tuomet BizTalk – tinkamas pasirinkimas.

Valdyti verslo PĮ modernizacijos kompleksiškumą yra sudėtinga, nes atnaujinimo sudėtingumui įtaką turi verslo PĮ dydis, užsibrėžtas modernizacijos lygis, žmonių kiekis, senos ir naujos PĮ technologijos išmanymas ir kiti veiksniai. Vyresni organizacijos darbuotojai, turintys senesnių technologijų patirtį turėtų įgyti žinių apie naujas technologijas bei įrankius, o jaunesniems darbuotojams vertėtų pasisemti patirties ir gerų praktinių patarimų iš vyresniųjų, labiau patyrusių darbuotojų.

Senos verslo PĮ architektūros atnaujinimas – sudėtinga užduotis. Jai atlikti, reikalinga nuodugni PĮ analizė, pasitelkiant tokias technikas kaip atvirkščioji inžinerija. Nuo to, kaip sėkmingai pavyks atkurti PĮ architektūrą, priklausys tolimesni šios architektūros transformavimo darbai.

Migruoti programinį kodą ir duomenis gali būti nesudėtinga, jei laiku ir teisingai ši užduotis bus atliekama. Jei duomenys bus migruojami anksčiau už PĮ kodą, kodo migracija žymiai supaprastės. Jei kodas bus migruotas pirmiau nei duomenys, duomenų migracijai prireiks papildomų adapterių. Geriausia kodą ir duomenis migruoti vienu metu, žinoma, prieš tai atlikus duomenų replikavimą.

Duomenų bazių procedūros, trigeriai, web servisai bei skriptai puikiai gali pagelbėti atliekant PĮ atnaujinimo ir migracijos darbus. Naudojant šias priemones svarbu atkreipti dėmesį į šių priemonių programavimo sudėtingumą bei laiko sąnaudas.

Darbo rezultatas – sukurta schema, panaudojant web servisus, duomenų bazės procedūras, trigerius, adapterius, MSMQ bei BizTalk mechanizmą, kurį naudojant galima efektyviai atlikti senos verslo PĮ atnaujinimą, stebėjimą bei migraciją.

Šaltiniai

[Arn93] Arnold R. S. „A Road Map to Software Re-engineering Technology", Software Re-engineering – a tutorial, IEEE Computer Society Press, 1993, pp. 3-22

[ARTI] Art in soft. Prieiga per internetą: http://www.artinsoft.com/legacy-software-migration.aspx [Žiūrėta 2010 m. balandžio 22 d.]

[Bas90] Basili V. R. „Viewing Maintenance as Reuse-Oriented SoftwareDevelopment", IEEE Software, 1990, pp.19-25

[BJG+97] Bisbal, Jesus, Geirdre Lawless, Bing Wu, Jane Grimson, Vincent Wade, Ray Richardson and D. O'Sullivan „An Overview of Legacy Information System Migration" Proceedings of the 4th Asian-Pacific Software Engineering and International Computer Science Conference (APSEC 97, ICSC 97). Hong Kong, 1997, pp.529-530

[BLW+99] Bisbal J., D. Lawless, B. Wu and J.Grimson „Legacy System Migration: A Brief Review of Problems, Solutions and Research Issues" (07-TCD-CS-1998-38). Dublin, Ireland: Computer Science Department, Trinity College, 1999, pp. 5-6

[Boe88] Boehm B. "A Spiral Model of Software Development and Enhancement" Computer (May), 1988, pp. 61-72

[Boe91] Boehm B. "Software Risk Management: Principles and practices" IEEE Software 8, 1 (January), 1991, pp.32-41

[BS89] Beath C. N., Swanson E. B. „Maintaining Information Systems in Organizations", New York, 1989, pp.155-167

[Cha06] Chappell D. „Understanding BizTalk Server 2006", 2005. Prieiga per internetą: http://download.microsoft.com/documents/australia/windowsserversystem/biztalk2006/Understanding_BTS06.pdf [Žiūrėta 2009 m. sausio 10 d.]

[Cha98] Chappell David. "Microsoft Message Queue Is a Fast, Efficient Choice for Your Distributed Application". Microsoft Journal (July), 1998, pp.17-24

[DEV07] Devanathan, N. Brms and Business Users. 2007. Prieiga per internetą: http://hosteddocs.ittoolbox.com/ND061807.pdf [Žiūrėta 2009 m. gegužės 12 d.]

[ES06] Eswaran, Sharanya, et al. „Adapting and Evaluating Commercial Workflow Engines for e-Science". E-Science and Grid Computing. E-Science,06. Second IEEE International Conference, 2006 pp. 20-28

[FS08] Feng X., Subramanian M. „Incorporating Business Rule Engine Technology in Control Center Applications-Toward Adaptive IT solutions". Energy 2030 Conference, ENERGY, 2008, pp. 1-5

[Gil88] Gilb T., „Principles of Software Engineering Management", Addison-Wesley, Reading, MA, 1988, pp. 136-148

[Gra05] Graham I, „Service Oriented Business Rules Management Systems". 2005. Prieiga per internetą: http://researchlibrary.theserverside.net/viewer/viewDocument.do?accessId=7637562 [Žiūrėta 2009 m. gegužės 20 d.]

[GT03] Grubb P., Takang A. A., „Software Maintenance – Consepts and Practice"- Second Edition, World Scientific Publishing, 2003, pp. 175-182

[HM98] Hall E. M."Managing Risk", MA: Addison-Wesley, 1998, pp. 107-119

[Hou99] Houston Peter "Selecting Between Synchronous And Asynchronous alternatives" Microsoft, 1999 Prieiga per internetą: http://www.sei.cmu.edu/activities/cbs/mls/links.html#houston99 [Žiūrėta 2009 m. lapkričio 22 d.]

[HRH96] Higuera, Ronald P. And Yacov Y. Haimes "Software Risk Management" (CMU/SEI-96-TR-012). Pittsburgh, PA: Software Engineering Institute, Carnagie Mellon University, 1996. Prieiga per internetą: http://www.sei.cmu.edu/activities/cbs/mls/links.html#higuera96 [Žiūrėta 2009 m. lapkričio 22 d.]

[IEE90] IEEE Std. 610.12, „Standart Glossary of Software Engineerin Terminology", IEEE Computer Society Press, Los Alamitos, CA, 1990.

[IEE98] IEEE Std. 1219-1998, „Standart for Software Maintenance", IEEE Computer Society Press, Los Alamitos, CA, 1998.

[ISO95] ISO/IEC 12207, „Information Technology – Software Life Cycle Processes", Geneva, Switzerland, 1995.

[KW96] Karolak, D.W. "Software Engineering Risk Management". Los Alamitos, CA:IEEE Computer Society Press, 1996, pp. 93-105

[Lam07] Lammel U. Business Rules make Business more flexible. 3rd Conference on Baltic Business and Socio-Economic Development BBSED 2007. Prieiga per internetą: www.wi.hs-wismar.de/~laemmel/Forschung/Docs/laemmel_BBSED07.pdf [Žiūrėta 2009 m. sausio 10 d.]

[OC90] Osborne, W.M., Chikofsky, E. J., „Fitting Pieces to the Maintenance Puzzle", IEEE software, 1990, pp. 11-12

[OV99] Ozsu, M.T., and P. Valduriez. „Principles of Distributed Database Systems" Upper Saddle River, NJ: Prentice-Hall, 1999, pp. 55-73

[PDH+99] Plakosh, Daniel, Scott Hissam and Kurt Wallnau. "Into the Black Box: A Case Study in Obtaining Visibility into Commercial Software" (CMU/SEI-99-TN-010).

Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 1999 Prieiga per internetą: http://www.sei.cmu.edu/activities/cbs/mls/links.html#plakosh99 [Žiūrėta 2009 m. lapkričio 22 d.]

[Pig97] Pigosky T.M., „Praktical Software Maintenance – Best Practices for Managing Your Software Investment", John Wiley & Sons, New York, NY, 1997, pp. 18-23

[RD05] Rosenberg F., Dustdar S. „Towards a distributed service-oriented business rules system. Web Services",. Third IEEE European Conference on 2005, pp. 1-11

[Sch87] Schneidewind, N. F., „The State of Software Maintenance", IEEE Transaction on Software Engineering, 1987, pp. 15-18

[SEMD1] Semantic Designs. Prieiga per internetą: http://www.semdesigns.com/Products/Services/SoftwareModernization.html?Home=LegacyMigration [Žiūrėta 2010 m. balandžio 22 d.]

[SEMD2] Semantic Designs. Prieiga per internetą: http://www.semdesigns.com/Products/Services/LegacyMigration.html [Žiūrėta 2010 m. balandžio 22 d.]

[Sne84] Sneed H.M. "Software Renewal: A Case Study", IEEE Software 1, 3 (July), 1984 pp. 56-63

[SPL03] Robert C. Seacord, Daniel Plakosh, Grace A. Lewis „Modernizing Legacy Systems" 2003, pp. 162-188

[TRB95] Tilley, S. R. And D. B. Smith. „Perspectives on Legacy System Reengineering" Pittsburg, PA: Reengineering Center, software Engineering Institute, Carnegie Mellon University 1995. Prieiga per internetą: http://www.sei.cmu.edu/activities/cbs/mls/links.html#tilley95 [Žiūrėta 2009 m. lapkričio 22 d.]

[ULR02] William M. Ulrich „Legacy Systems: Transformation Strategies" Prentice Hall PTR, Upper Saddle River, NJ, USA, 2002, pp. 118-166

[Vis99] Visagio, G., „Assessing the Maintenance Process through Replicated Controlled Experiments", Elsevier Science Inc. New York, NY, USA, 1999, pp. 165-187

[VoCon] „Software Maintenance" Prieiga per internetą: http://cnx.org/content/m14717/latest/ [Žiūrėta 2010 m. balandžio 22 d.]

[War99] Warren I.. "The Renaissance of Legacy Systems: Method Support for Software Evolution" London: Springer-Verlag, 1999, pp. 236-255

[WIKI] Wikipedia – The Free Encyclopedia. Prieiga per internetą: http://en.wikipedia.org/wiki/Outsourcing [Žiūrėta 2010 m. balandžio 22 d.]

[WIKI2] Wikipedia – The Free Encyclopedia. Prieiga per internetą: http://en.wikipedia.org/wiki/Reverse_engineering [Žiūrėta 2010 m. gegužės 18 d.]

[WIKI3] Wikipedia – The Free Encyclopedia. Prieiga per internetą: http://wiki.answers.com/Q/What_is_the_difference_between_reverse_engineering_and_re-engineering&alreadyAsked=1&rtitle=Differnce_between_reverse_engineering_and_re_engineering [Žiūrėta 2010 m. gegužės 18 d.]