KAUNO TECHNOLOGIJOS UNIVERSITETAS
INFORMATIKOS FAKULTETAS
PROGRAMŲ INŽINERIJOS KATEDRA

Milanas Skačkauskas

Paveldėtosios sistemos rekonstrukcija: metodai, modeliai ir realizacija

Magistro darbas

Darbo vadovas

doc. S. Vaičiulis

Kaunas, 2009

KAUNO TECHNOLOGIJOS UNIVERSITETAS
INFORMATIKOS FAKULTETAS
PROGRAMŲ INŽINERIJOS KATEDRA

Milanas Skačkauskas

Paveldėtosios sistemos rekonstrukcija: metodai, modeliai ir realizacija

Magistro darbas

Recenzentas
doc. A. Lenkevičius
2009-05-25

Vadovas
doc. S. Vaičiulis
2009-05-25

Atliko
IFM-3/2 gr. stud.
Milanas Skačkauskas
2009-05-25

Kaunas, 2009

Turinys

Reconstruction of legacy system: models, methods and implementation
Summary

The aim of this work is to made legacy system reconstruction analysis and based on the results implement flat owners information system reconstruction.

Reconstruction of legacy systems is very important for organizations, which spend too much time and money, trying to maintain these outdated systems business value. Another legacy systems renewal factor is that the organization is that the organization seeks to move the new Internet-based platforms like .NET, J2EE, and PHP. New technologies adaptation can reduce maintenance cost and make information system easier to maintain, making it easier to make changes or to adapt the market changes.

For system reconstruction realization was chosen reverse, data and forward reconstruction methods. Also was performed reconstructed system quality estimation in which was determined, that system match all customer requirements.

PAVEIKSLIUKŲ SĄRAŠAS

LENTELIŲ SĄRAŠAS

1. ĮVADAS

1.1. Santrauka

Paveldėtos sistemos buvo suprojektuotos taip kad veiktų ant centralizuotų sistemų. Paprastai jos yra monolitinės ir atlikus didelę eilę pakeitimų, jas tampa sudėtinga toliau palaikyti. Dėl naujų technologijų vystymosi, organizacijos siekia, kada jų paveldėtos sistemos būtų integruotos į naujas aplinką, kad būtų galima lengvai ir greitai atlikti verslo procesų pakeitimus. Yra keturi skirtingi modernizavimo metodai: pakeitimas nauja arba perkūrimas iš naujo, apgaubimas, perkėlimas ir rekonstrukcija. Tačiau, daugelis organizacijų nenori kurti savo paveldėtos sistemas nuo pradžios dėl aukštos kainos. Dauguma organizacijų naudoja apgaubimo metodą, kad modernizuotų paveldėtas sistemas. Deja, šis metodas neatlieka sistemos kokybės pagerinimo. Perkėlimas yra ganėtinai sudėtingas. Rekonstrukcija yra alternatyvus paveldėtų sistemų modernizavimo metodas. Tačiau, rekonstrukcija nepadės būsimam palaikymui, jei sistemų branduolys nebus pakeistas. Pastovūs pakeitimai vis dar bus būtini, jei nestabilios ir kritinės sistemų dalys nebus iš naujo suprojektuotos. Tam reikalui pagelbės tiesioginės inžinerijos procesas. Tiesioginė inžinerija yra tradicinis programinės įrangos išsivystymo procesas, apimdamas reikalavimus ir specifikaciją, projektą, kodavimą, ir testavimą.

Darbo metu buvo gilintasi į paveldėtų sistemų problemą, bei jų sprendimo būdus ir modernizavimą. Atlikus sistemos įvertinimą buvo pasirinkta programinės įrangos rekonstrukcija. Tolesnėje dalyje analizuojamas rekonstrukcijos proceso modelis, rekonstrukcijos metodai. Remiantis atliktos analizės rezultatais, buvo atlikta ir realizuota butų savininkų bendrijų informacinės sistemos rekonstrukcija pasirinkus apgrąžos, duomenų ir tiesioginės rekonstrukcijos metodus. Atlikus rekonstrukciją buvo atliktas rekonstruotos sistemos kokybės vertinimas siekiant nustatyti rekonstruotos sistemos kokybę. Vertinimo metu buvo nustatyta, kad sistema atitinka užsakovo keltus reikalavimus.

2. TERMINŲ IR SANTRUMPŲ ŽODYNAS

2.1 Lentelė. *Terminų ir santrumpų žodynas*

Santrumpa, terminas	Paaiškinimas
MVC (*angl. Model View Controller*)	– modelis-vaizdas-valdiklis šablonas
XML(*angl. eXtensible Markup Language*)	yra W3C rekomenduojama bendros paskirties duomenų struktūrų bei jų turinio aprašomoji kalba. Pagrindinė XML kalbos paskirtis yra užtikrinti lengvesnį duomenų keitimąsi tarp skirtingo tipo sistemų, dažniausiai sujungtų internetu.
OS (*angl. Operating System*)	– operacinė sistema
DB (*angl. Database*)	– duomenų bazė
UML (*angl. Unified Modelling Language*)	– unifikuota modeliavimo kalba UML
CGI (*angl. Common Gateway Interface*)	- protokolas, apibrėžiantis, kaip turi bendrauti WWW serveris ir jo vykdomos programos, skirtos iš naršyklės gautai informacijai apdoroti ir/arba dinaminiams puslapiams generuoti.
ASCII (*angl. American Standard Code for Information Interchange*)	- *(„Amerikietiškas informacijos mainų koduotės standartas“)* yra simbolių koduotė, pritaikyta anglų kalbos abėcėlei.
PHP (*angl. Hypertext Preprocessor*)	- plačiai paplitusi dinaminė interpretuojama programavimo kalba sukurta 1997 m. ir specialiai pritaikyta interneto svetainių kūrimui.
MySQL	- viena iš reliacinių duomenų bazių valdymo sistemų (*liet. santrumpa RDBVS, angl. – RDBMS*), palaikanti daugelį naudotojų, dirbanti SQL kalbos pagrindu.
HTML (*angl. Hyper text Markup Language)*	– *(„Hiperteksto žymėjimo kalba“)* tai kompiuterinė žymėjimo kalba, naudojama pateikti turinį internete. Kalbą standartizuoja W3 konsorciumas.
CSS (*angl. Cascading Style Sheets*)	– kalba, skirta nusakyti kita struktūrine kalba aprašyto dokumento vaizdavimą. Dažniausiai CSS aprašomas HTML dokumentų pateikimas, tačiau ją galima taikyti ir įvairiems kitiems XML dokumentams.
DBF	Vienas iš pagrindinių dBase duomenu bazių valdymo sistemos failų formatų. Yra plačiai panaudotas daugelyje kitų paraiškų, reikalaujančių paprasto formato, kad sukaupti suformuotus duomenis.

W3C (*angl. World Wide Web Consortium*)	- yra konsorciumas leidžiantis programinės įrangos standartus („rekomendacijas“, kaip jie jas vadina) žiniatinkliui.
BLL (*angl. Business Logic Layer*)	– veiklos logikos projektavimo sluoksnis
DAL (*angl. Data Access Layer*)	– priėjimo prie duomenų projektavimo sluoksnis
UI (*angl. User Interface*)	– vartotojo sąsaja
DBVS (*Duomenų Bazių Valdymo Sistema*)	- kompiuterinė programa ar programų paketas, skirtas duomenų bazės valdymui. Paprastai DBVS sugeba valdyti milžiniškus struktūrizuotų duomenų kiekius bei vienu metu palaiko daugelį lygiagrečiai dirbančių vartotojų.
WEB (*angl. World Wide Web*)	- yra sujungtų hiperteksto dokumentų sistema, prie kurių gaunama prieigą per internetą.
ISO 9126	- yra tarptautinis standartas programinės įrangos kokybės įvertinimui.

3. PAVELDĖTOSIOS SISTEMOS IR JŲ MODERNIZAVIMAS

3.1. Paveldėtos sistemos

3.1.1. Paveldėtos sistemos apibrėžimas

Daugelyje įvairių literatūros šaltinių aptinkami skirtingi paveldėtų sistemų apibrėžimai. Tačiau visų jų prasmė kalba apie tą pačią problemą. Taigi paveldėta sistema gali būti apibrėžta kaip “bet kokia informacinė sistema, kuri priešinasi pasikeitimui ir vystymuisi”[1].

Sistemos, kurios yra kuriamos specialiai tam tikrai organizacijai ir kurios gyvuoja pakankamai ilgai. Daugelis naudojamų sistemų buvo sukurtos labai seniai naudojant pasenusias technologijas, tačiau tos sistemos vis dar reikalingos normaliai organizacijos veiklai užtikrinti [2]. Todėl jos yra vadinamos paveldėtosiomis sistemomis, paveldo sistemos (*angl. legacy systems*).

Paveldėtos informacinės sistemos tipiškai formuoja informacijos srautą organizacijos viduje ir yra svarbiausia priemonė tam, kad užtikrintų informaciją apie jos verslą, bei procesus. Problemos vienoje iš šitų sistemų gali turėti rimtą verslo poveikį [3]. Šios paveldėtos sistemos šiuo metu kelia dideles ir aktualias problemas organizacijoms. Rimčiausios iš šių problemų yra:

- šios sistemos paprastai veikia ant atgyvenusios aparatūrinės įrangos, kuri yra lėta ir brangi palaikyti;
- programinės įrangos palaikymas yra apskritai brangus; klaidų susekimas yra brangus ir daug darbo reikalaujantis dėl dokumentacijos trūkumo ir bendro vidaus sistemos darbų supratimo trūkumo;
- integracijos pastangoms labai trukdo „švarių“ vartotojų sąsajų trūkumas;
- paveldėtos sistemos yra labai sudėtingos, o kartais neįmanomos vystyti.

I. Sommerville, šias sistemas apibūdina kaip sociotechninės sistemas, kurios buvo išvystytos, naudojant seną ar atgyvenusią technologiją [4]. Šios sistemos yra dažnai gyvybiškai svarbios organizacijai ar verslui, todėl jų atsikratymas dažnai įmonei yra per pavojingas. Paveldėtos sistemų palaikymas organizacijai kainuoja didelius kaštus, o tuo pačiu varžo naujų verslo procesų įgyvendinimą.

Paveldėtos sistemos yra apibūdinamos sekančiais atributais:

- svarbios organizacijų verslo procesams
- paprastai labai senos
- plačiai modifikuotos, ir tapę sudėtingomis
- pagrįstos sena technologija

- sistemos kūrėjų jau nebėra
- skurdi dokumentacija, arba jos visai nėra
- labai brangi priežiūra

3.1.2. Paveldėtų sistemų problema

Įmonė auga, kad naujos programinės įrangos kūrimas pralenkia mūsų gebėjimą palaikyti tai. Dideli programinės įrangos palaikymo kaštai, palieka mažą biudžeto dalį naujam vystymui. Vystant sistemą yra tikėtinas kaštų mažėjimas, bet tai yra sunku pasiekti kadangi palaikymo procesas yra neišvengiamas. Jei seksime šia tendencija, galų gale neliks kitų galimybių kaip vystyti naujas sistemas, ir mes įeisime į informacijos amžių [5].

Kodėl paveldėtoji sistema turi būti tobulinama?

Pateikiamos priežastys, kodėl paveldėtos sistemos turi būti tobulinamos:

- Programinė įranga turi būti pritaikyta, kad patenkintų naujų skaičiavimo aplinkų ar technologijos norus.
- Programinė įranga turi būti patobulinta, kad įgyvendintų naujus verslo reikalavimus.
- Programinė įranga turi būti išplėsta, kad bendradarbiautų su kitomis labiau šiuolaikinėmis sistemomis ar duomenų bazėmis.
- Programinė įranga turi būti iš naujo projektuota, kad įgyvendintumėme ją tinklo viduje.

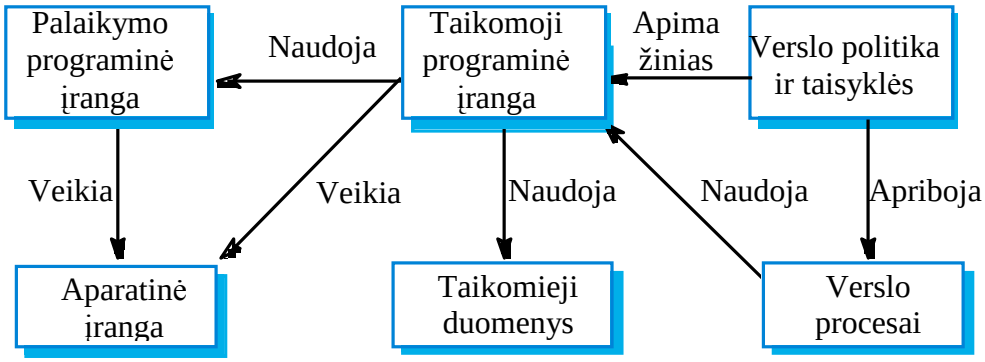
Galima išskirti paveldėtų sistemų tobulinimo sunkumus:

- Yra didelė rizika pakeisti paveldėtąją sistemą naujai sukurta sistema;
- paveldėtos sistemos retai turi pilną specifikaciją;
- verslo procesai naudoja paveldėtąją sistemą;
- naujos sistemos kūrimas gali būti nesėkmingas;
- sistemos turi keistis, kad išliktų naudingos;
- paveldėtų sistemų keitimas yra labai brangus;
- nėra vieningo programavimo stiliaus: skirtingos dalys buvo sukurtos skirtingų grupių;
- sistema galėjo būti sukurta pasenusia programavimo kalba;
- sistemos struktūra gali būti sugadinta;
- priemonės, skirtos atminčiai taupyti arba greičiui didinti, sumažino suprantamumą;
- failų struktūros gali būti nesuderinamos;

3.1.3. Paveldėtųjų sistemų struktūra

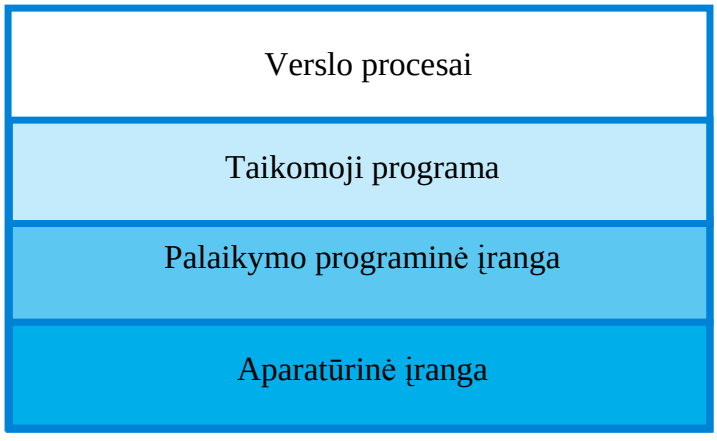
Paveldėta sistema yra sudaryta iš sekančių komponentų [4]:

1. Aparatinė įranga - gali būti atgyvenusi centrinio kompiuterio aparatine įranga.
2. Palaikymo programinė įranga - gali būti priklausoma gamintojų ar palaikymo programinės įranga, tokios kaip operacinė sistema ir kt..
3. Taikomoji programinė įranga - gali būti parašyta atgyvenusia programavimo kalba.
4. Taikomieji duomenys – duomenys kuriuos apdoroja sistema. Jie dažnai neužbaigti ir nenuoseklus.
5. Verslo procesai - gali būti suvaržyti programinės įrangos struktūros ir funkcionalumo.
6. Verslo politika ir taisyklės – paveldėtoji sistema gali priklausyti nuo verslo politikos ir taisyklių.



3.1 pav. *Paveldėtosios sistemos komponentai*

I. Sommerville teigia, kad visus šiuos komponentus galima suskirstyti į sekančius lygius kaip parodyta 3.2 paveikslėlyje.



3.2 pav. *Sluoksninis paveldėtosios sistemos modelis*

Kalbant apie sluoksniuotą paveldėtos sistemos modelio keitimą galime pasakyti:

- Teoriškai įmanoma pakeisti lygmenį sistemoje neliečiant kitų lygmenų
- Praktiškai tai yra neįmanoma

- Vieno lygmens keitimas įneša naujas priemones ir aukštesnieji lygmenys turi pasikeisti, kad galėtų šias priemones naudoti
- Programų keitimas gali sulėtinti sistemą, todėl gali reikėti pakeisti ir aparatūrą
- Dažnai neįmanoma prižiūrėti aparatūrinių sąsajų dėl jų nesuderinamumo

3.1.4. Paveldėtųjų sistemų įvertinimas

Organizacijos, kurios pasitiki turimomis paveldėtomis sistemomis turi pasirinkti jos vystymo strategiją:

- Visiškai atsisakyti sistemos ir verslo procesus pakeisti, kad ji nebūtų daugiau reikalinga;
- Tęsti sistemos palaikymą
- Rekonstruoti sistemą, kad pagerintų jos priežiūrą ir palaikymą
- Pakeisti seną sistemą nauja sistema

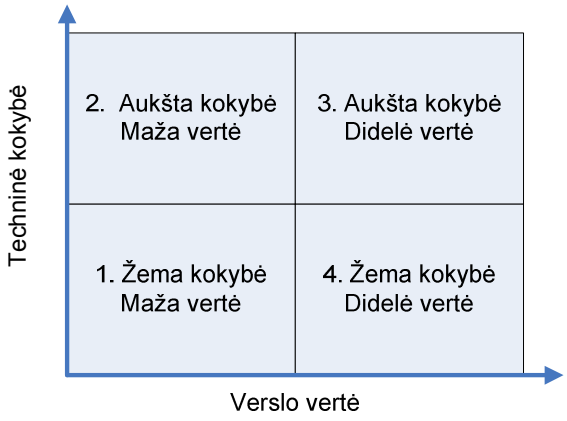
Organizacija turi pasirinkti strategija atsižvelgdama į sistemos kokybės ir jos vertė verslui santykį.

Portfelio analizė nustato techninės kokybės metrikas ir verslo vertę sistemos rinkiniui ir įvertina šį rinkinį pagal tas metrikas [6].

Techninė kokybė yra sistemos gerumo matas lyginant sistemą su apibrėžtu techninių kriterijų rinkiniu. Pavyzdžiui, techninės kokybės kriterijai apima atnaujinimų išleidimo dažnumą, lengvumą atlikti sistemos pakeitimus, programinės ir aparatinė įrangos patikimumą, organizacinę infrastruktūrą, sistemos našumą, tikslumą, lengvą įsisavinimą.

Verslo vertė yra matas atitinkantis sistemos svarbumą organizacijai. Pavyzdžiui, verslo vertės kriterijai apima sistemos naudą, naudojimo lygį, tenkinamų verslo tikslų skaičių, sistemos vertę, vartotojų patenkinimą ir informacijos vertę, kurią programa ar sistema sukaupia.

Paveldėtos sistemos yra vertinamos pagal šituos matus ir atvaizduojamos ant portfelio analizės grafo, taip kaip parodyta 3.3 paveikslėlyje. Sistemos atvaizdavimas viename iš šitų kvadratų reikalingas tam, kad būtų išmatuojama techninė kokybė ir verslo vertė. Kvadratas kuriame sistema atsiduria siūlo tinkamą vystymo strategiją [7].



3.3 pav. *Portfelio analizės diagramas*

1 Kvadratas: Sistema turi mažą verslo vertę ir blogą techninę kokybę. Tokia sistema turi buti išmesta ir pakeista kitu nauju komerciniu paketu, dėl dviejų priežasčių. Pirma, todėl, kad ji turi žemą techninę kokybę, kuri turi būti pagerinta ar pakeista. Antra, todėl, kad ji turi mažą verslo vertę, sistema neteikia organizacijai jokių kritiškų paslaugų.

2 Kvadratas: Sistemos su aukšta technine kokybe ir žema verslo verte neturi būti perkuriamos iš naujo, modernizuojamos, ar keičiamos. Tokios sistemos turi būti ir toliau palaikomos.

3 Kvadratas: Aukštos kokybės sistemos su didele verslo verte turi būti ir toliau palaikomos. Tokios sistemos labai svarbios organizacijai. Dėl aukštos kokybės sistema lengvai prižiūrima.

4 Kvadratas: Sistema su aukšta verslo verte ir žema technine kokybe yra geriausia kandidatė atlikti modernizaciją ar pakeitimą.

Po parengiamosios portfelio analizės, turi būti apžvelgtos organizacinės ar išteklių svarstomos problemos. Šios problemos gali žymiai padidinti modernizavimo riziką. Tipiškos svarstomos problemos apima: organizacijos politiką, kainas, planą, laisvus darbuotojus, jų techninius įgūdžius, reikalingus atlikti modernizavimą, programuotojus, galinio vartotojo apmokymą, apmokymo personalą, modernizuotos sistemos priėmimą iš vartotojų pusės.

Verslo proceso įvertinimas

Norint atlikti verslo procesų įvertinimą reikia užduoti tokius klausimus:

1. Ar yra apibrėžtas proceso modelis ir ar jo yra laikomasi?
2. Ar skirtingos organizacijos dalys (padaliniai) naudoja skirtingus procesus tai pačiai veiklai?
3. Kaip procesas buvo adaptuotas?
4. Kokie yra sąryšiai su kitais verslo procesais?
5. Ar procesą palaiko paveldėtoji taikomoji programinė įranga?

Aplinkos įvertinimas

Siekiant įvertinti sistemos aplinką pirmiausia reikėtų atsakyti:

1. Koks tiekėjo patikimumas?
2. Ar tiekėjas vis dar egzistuoja? Ar sistemas palaiko kas nors kitas?
3. Koks sistemos gedimų lygmuo?
4. Ar aparatūra dažnai genda? Ar palaikančios programos dažnai “lūžta”?
5. Koks sistemos amžius?
6. Koks aparatūros ir programinės įrangos amžius?
7. Koks sistemos greitis?
8. Ar sistemos greitis tenkina sistemos vartotojus?

Vertinimo metodo rezultatas yra įgyti pakankamą paveldėtos sistemos supratimą. Suvokiant sistemą iš techninių, verslo, ir organizacinių perspektyvų galima pasirinkti tinkamai vystymosi strategiją. Todėl turime gauti atsakymus į vertinimo klausimus, tokius kaip [8]:

1. **Ar sistema yra kritiška organizacijai?** Atsakymas galėtų atskleisti, kad sistema nėra labai būtina verslo procesams. Tokiu atveju, nereikia svarstyti tolesnio sistemos vystymo.
2. **Kokie yra organizacijos verslo tikslai?** Vertintojai turi suprasti organizacijos verslo tikslus, kuriuos patenkina paveldėta sistema. Verslo tikslai kuria vystymosi reikalavimus.
3. **Kokie yra vystymosi reikalavimai?** Vystymosi reikalavimai yra gaunami iš verslo tikslų ir sistemos įvertinimo. Turi būti išsiaiškinta, ar egzistuojanti sistema gali patenkinti siekiamus reikalavimus.
4. **Kokia numatoma sistemos gyvavimo ciklo trukmė?** Sistemos gyvenimo trukmę diktuoja faktoriai tokie kaip programinės įrangos ir aparatinės įrangos tinkamumas. Kai aparatinė įranga ar palaikymo programinė įranga tampa atgyvenusi, sistemos naudojimo trukmė ribota.
5. **Kokia reikalaujama sistemos gyvavimo trukmė?** Jei pastebima, kad sistema bus naudojama ir yra reikalinga trumpam laikui, beprasmiška patirti išlaidas siekiant vystyti sistemą. Priešingai, jei numatoma, kad sistema ilga laiką palaikys organizacijos verslo procesus, tada itin reikalinga imtis sistemos modernizavimo.
6. **Kokia yra techninė sistemos būsena?** Jei taikomoji programinė įranga, itin blogoje techninėje būsenoje, tokią sistemą bus sudėtinga suprasti ir brangu palaikyti.

7. **Ar organizacija pritaria sistemos pakeitimams?** Organizacijos požiūris į pakeitimus yra kritiškas ir labai svarbus siekiat sėkmingai išvystyti projektą.

Taip pat turi būti atliktas taikomųjų programų įvertinimas**.**

Suprantamumas: Ar sunku suprasti sistemos išeities kodą? Ar naudojamos sudėtingos valdymo struktūros? Ar kintamieji turi prasmingus vardus?

Dokumentavimas: Ar sistema dokumentuota? Ar sistemos dokumentacija yra pilna, išsami ir atnaujinama?

Duomenys: Ar yra išreikštas sistemos duomenų modelis? Ar duomenys dubliuojami?

Greitis: Ar sistemos vartotojus tenkina greitis?

Programavimo kalba: Ar programavimo kalba vis dar naudojama? Ar yra jai šiuolaikinių kompiliatorių?

Konfigūravimo valdymas: Ar sistemos versijos yra valdomos konfigūravimo sistemos?

Testiniai duomenys: Ar yra sistemos testiniai duomenys?

Personalo patyrimas: Ar yra žmonių, susipažinusių su sistema? Galinčių ją prižiūrėti?

3.2. Paveldėtų sistemų modernizavimas

Nuo to momento kai programinė įranga yra išleista, prasideda lenktyniavimas su laiku ir jos senėjimu. Frazė, kad "paveldėtas kodas yra kodas, parašytas vakar" yra vis labiau ir labiau teisingas. Technologijų išsivystymo tempo didėjimas, taip pat didina ir technologijų senėjimo tempą. Programinės įrangos modernizacija bando plėtoti paveldėtą sistemą, ar sistemos elementus, kai tradicinės evoliucinės veiklos, tokios kaip palaikymas ir tobulinimas, negali daugiau pasiekti ar palaikyti pageidaujamų sistemos savybių.

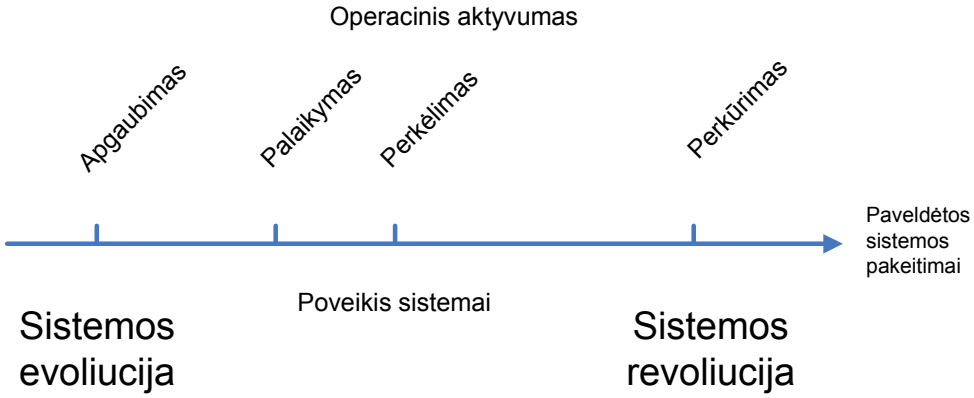
Programinės įrangos modernizacija yra daug jėgų ir žinių iš programinės įrangos inžinierių reikalaujantis procesas. Daug modernizuojamų projektų neįgyvendinama. „*Standish*“ grupės tyrimai rodo, kad 24 procentai projektų yra išvis nepabaigti, 44 procentai nebaigė projektų laiku, viršijo biudžetą, ir/ar neįgyvendino reikalaujamų funkcijų ar projekto tikslų, tuo metu, kai tiktai 32 procentai baigia laiku ir su planuotu biudžetu, bei lauktu funkcionalumu [9].

3.2.1. Susidorojimas su paveldėtomis sistemomis

J. Bisbal išskiria 4 pavedėtų sistemų problemos sprendimo metodus: apgaubimas, palaikymas, perkėlimas (migracija) ir perkūrimas [7]. 3.4 paveikslėlyje šie metodai pateikti pagal poveikį paveldėtoms sistemoms. Palaikymas čia minimas tiktai išbaigtumui, kadangi tai yra kiekvienos sistemos gyvavimo ciklo dalis, ar ji būti paveldėta ar ne. Iš tikrųjų, jei

programinės įrangos sistema gali būti palaikoma tam tikro priimtino biudžeto rėmuose, tokia paprastai nelaikoma paveldėta sistema [10].

Comella-Dorda išskiria 3 svarbius modernizavimo metodus: palaikymas, keitimas kita sistema ir rekonstrukcija [11].



3.4 pav. *Metodai, kaip susidoroti su paveldėtomis informacinėmis sistemomis*

3.4 pav. iliustruoja dvi skirtingas svarstomas problemas dėl šitų metodų. Pirma, jų poveikis paveldėtoms sistemoms, su perkūrimo pasirinkimu, kuris priveda prie daugumos svarbių pakeitimų - sistemos revoliucijos. Antra, faktas, kad jie formuoja nuolatinį metodų spektrą, o ne keturias visiškai atskiras alternatyvas. Atsižvelgiant į konkrečią paveldėtos sistemos problemą, ne visada galima apsispręsti, ar sprendimas, pritaikytas, priklauso tiktai vienai iš keturių alternatyvų, ar ir kitoms.

Pavyzdžiui, apgaubimas apibrėžiamas kaip palaikymo veikla, kuri nusitaiko į komponentų sukūrimą, neatsižvelgiant į apgaubto komponento pakeitimus. Perkėlimas gali apimti sistemos apgaubimo dalį, kitos dalies priežiūrą. Perkūrimas dar kitų dalių, ir taip toliau. Turi būti pažymima, kad, nors šitie metodai yra apibūdinti kaip pritaikyti sistemos lygmenyje, iš tikrųjų tikėtina, kad jie būtų pritaikyti sudedamajame lygmenyje. Todėl, skirtingai naudojamos veiklos (kaip parodyta 3.4 pav.) gali būti pritaikytos skirtingiems sistemos komponentams [7].

Metodai, analizuojami šiame skyriuje, gali būti padalyti į dvi kategorijas. Pirma, tie, kuriems reikalinga, kad paveldėta sistema turi būti išjungta kuriam tai laikui, kol vystoma ją pakeisianti nauja sistemą. Ir galiausiai, tie metodai, kurie pripažįsta, kad kritinės paveldėtos sistemos negali būti išjungtos kuriam tai reikšmingam laikui ir turi būti aprūpintos mechanizmu, kad sistema liktų veikianti atliekant perkėlimą [13]. Antros kategorijoje, būdas kritinėms sistemoms palaikyti yra pateikti atitinkami metodai (pavyzdžiui „tarpininku pagrįstas“ (*angl. gateway-based*) prieš „be tarpininko“ (*angl. gateway-free*) metodą). Metodai pasirenkami atsižvelgiant į sistemos sunkumą ir metodo lankstumą.

3.2.2. Paveldėtų sistemų modernizavimo metodų klasifikacija

Pirmiausia turi būti atliktas „paradigmos pakeitimo“ sprendimas, todėl, kad tai apibrėžia svarbiausią sprendimą, kurį organizacija turi priimti planuodama paveldėtos sistemos modernizacijos strategiją. „Keitimo paradigma“ kalba, jog mes turime keisti seną procedūrinę programavimo kalbą į objektiškai orientuotą projektą ar komponentu pagrįstą projektą. „Paradigma nekeisti“ neketina likti su einamuoju aplinkos modernizavimo sprendimu. Jei joks paradigmos pakeitimas nėra nuspręstas tada, mes esame apriboti metodų, kurie gali visi būti skirstyti kategorijomis, pvz. kaip „apgaubimas“ .

Viso yra 5 paveldėtų sistemų modernizavimo būdai remiantis įmonės reikalavimais ir poreikiais, todėl gali būti pasirinktas vienas ar kelių būdų kombinacija.

Paveldėtų sistemų modernizavimo būdai:

- Apgaubimas
- Palaikymas
- Perkėlimas (migracija)
- Perkūrimas iš naujo
- Rekonstravimas

3.2.3. Apgaubimo metodas

Apgaubimas apima paveldėtos sistemos egzistuojančių duomenų apsupimą taikomąja sistema, ir siekia, kad paveldėtai sistemai suteiktų „naują ir pagerintą“ išvaizdą ar patobulintas funkcijas [10]. Apgaubtas komponentas veikia kaip serveris, įvykdydamas tam tikrą funkciją, reikalaujamą išorinio kliento, kuris neturi žinoti, kaip paslauga yra įgyvendinta. Apgaubimas turi leisti pakartotinai panaudoti sistemos komponentus, kuriais organizacija pasitiki, kurie yra ištestuoti ir kurie veikė paveldėtoje sistemoje kelis metus.

Apgaubimo metodas yra padalytas į tris alternatyvius metodus ([13][14]).

- „Vartotojų sąsajos apgaubimas“, kur paveldėtos sistemos sąsaja yra perkuriama į šiuolaikinę grafinę sąsają.
- „Duomenų apgaubimas“, kur yra išvystytos naujos sąsajos paveldėtų duomenų struktūrai. Taip nauja sistema gauna prieigą prie senų duomenų. Šis metodas galėjo būti įgyvendintas per „duomenų bazės tarpininką“, „XML integraciją“ ir „duomenų replikavimo“ metodus.
- „Funkcijos apgaubimas“, kur netik duomenys, bet ir verslo logika, užkoduota sena programine paveldėta kalba, yra apgaubiama ir yra pasiekiama per kitų programų sąsają. Tai gali būti padalijama į tris sekančias poklases: „komponento apgaubimas“,

„objektiškai orientuotas apgaubimas“ ir „CGI (*angl. Common Gateway Interface*)„ integracija.

Populiariausias apgaubimo metodas yra „Vartotojų sąsajos apgaubimas“. Tokio metodo realizavimas yra pigus, bet efektyvus. Tačiau nepaisant grafinės sąsajos privalumų tai, yra vis dar didžiąja dalimi trumpalaikis sprendimas. Dažniausiai vartotojo sąsaja, sistemos veiklos logika bei DB sluoksniai yra tarpusavyje susipynę, o ne atskiri.

Komponentų apgaubimas paveldėtoje sistemoje neatkreipia dėmesį į daugelį rimtų problemų, su kuriomis susiduria sistemos. Problemos: vartotojo sąsajos perkrovimas, nesugebėjimas vystytis, kad aprūpintų naujomis funkcijomis, bei aukštos palaikymo kainos yra ignoruojamos. Paveldėtos sistemos apgaubimas specialia programine įranga prideda funkciškai pernelyg gausų sluoksnį, kuris turės būti palaikytas.

3.2.4. PĮ palaikymas

Palaikymas yra augantis ir pasikartojantis procesas, kuriame sistemai yra atliekami maži pakeitimai. Šie pakeitimai yra dažniausiai yra klaidų, trikdžių taisymas ar maži funkciniai patobulinimai, kurie neapima pagrindinių struktūrinių pakeitimų [15]. Palaikymas privalo dalyvauti bet kokios sistemos vystyme, bet turi būti apribotas. Šitas apribojimas apima:

- Pranašumai gauti pritaikius naujas technologijas yra rimtai suvaržyti, nes patobulinimai, kaip pvz.: įgyvendinta paskirstyta architektūra ar realizuota grafinė vartotojo sąsaja dažniausiai nėra laikoma palaikymo operacijomis.
- Paveldėtų sistemų palaikymo kainos laukui bėgant gali didėti. Rasti senos technologijos ekspertus tampa vis labiau ir labiau sudėtinga ir brangu.
- Paveldėtos sistemos pakeitimus pritaikyti prie naujų verslų poreikių tampa vis labiau ir labiau sudėtinga, kadangi daugelio mažų atskirų pakeitimų poveikis gali būti didesnis negu jų suma.

Sistemos palaikymas yra kiekvienos sistemos gyvavimo ciklo dalis. Ir nepriklauso nuo to ar sistema yra paveldėta ar ne. Todėl savo prasme palaikymas nėra suprantamas kaip sistemos modernizavimo metodas.

3.2.5. Perkėlimo metodas

Situacijose, kur apgaubimas nėra tinkama alternatyva dėl jos apribojimo, turi būti svarstomas paveldėtos sistemos perkėlimas, kitaip vadinamas migracija. Metodas reikalauja daug didesnių pastangų, bet, jei pritaikomas sėkmingai - gaunama ilgalaikė ir didesnė nauda (pvz. daugiau lankstumo, geresnis sistemos supratimas, lengvesnė priežiūra, mažesni kaštai.

Nors paveldėtos informacinės sistemos perkėlimas yra svarstoma verslo problema, buvo išvystyti keli visapusiai perkėlimo metodai. Yra galimas bendras perkėlimo metodas,

tinkamas visoms paveldėtoms sistemoms. Tačiau, yra būtinas visapusių nurodymų komplektas, kaip atlikti perkėlimą. Prieš pradedant projekto perkėlimą, turi būti intensyviai studijuojama kaip spręsti paveldėtos sistemos problemas.

Perkėlimo metodas gali būti padaloma į dvi klases kaip aprašyta toliau:

- pirmoji klasė yra „komponento perkėlimas“, kai didelės paveldėtos sistemos yra suskaidomos į nepriklausomus komponentus ir kiekvieną komponentas yra perkeliamas atskirai [5]. Vyksta periodas, kur ir paveldėta, ir nauja platforma turi būti naudojama ir dirbti kartu. Yra dvi strategijos: „laipsniškas sąveikavimas“ ir „lygiagretus veikimas“. Abiem strategijoms reikia duomenų, kurie bus bendri ir pasiekiami per duomenų bazės tarpininką, ar kopijuojami abejoms platformoms, ar padalyti į atskiras nepriklausomas sritis, kurios bus perkeltos palaipsniui į naują platformą ([5][11]).
- antras perkėlimo metodas yra „sistemos perkėlimo“ metodas, kuriame visa paveldėta sistema ir duomenys yra perleidžiami naujai platformai vienu žingsniu.

Sėkmingai įgyvendinus šį metodą gaunama ilgalaikė ir didesnė nauda nei apgaubimas. Tačiau toks modernizavimo metodas yra gana sudėtingas. Taikant komponento perkėlimo metodą duomenų padalijimas nėra lengvais pritaikomas paveldėtoms duomenų bazėms.

Taikant visos sistemos perkėlimą, kai sistemos duomenys vienu žingsniu yra perleidžiama naujai platformai, kyla didelė rizika, kad nesėkmės atveju organizacija gali netekti svarbių duomenų ir jos darbas gali sustoti. Tada reikėtų pasirūpinti kaip sugrįžti prie senosios sistemos.

3.2.6. Pakeitimo metodas

Sistemos pakeitimas metodas yra tinkamas, kai paveldėtos sistemos neatitinka verslo poreikių ir kai modernizacija nėra galima arba neefektyvi atsižvelgiant į reikalaujamus kaštus [7]. Senos sistemos pakeitimas nauja gali būti vieninteliu pasirinkimu, kai rekonstrukcijos modernizacijos metodai negali būti pateisinti.

Pakeitimas sukelia didelę riziką, kuri turi būti įvertinta prieš pasirenkant šį metodą:

- IT personalas, vykdantis palaikymo darbus, galbūt nėra pažįstamas su naujomis technologijomis.
- Atlikus sistemos pakeitimą kita, reikalingas išsamus naujos sistemos testavimas. Paveldėtos sistemos yra paprastai gerai ištestuotos ir suderintos. Nėra jokios garantijos, kad nauja sistema bus tokia pat stabili ar atitiks reikalaujamas funkcijas kaip sena.

3.2.7. Perkūrimo ir rekonstrukcijos metodai

Programų rekonstrukcija, reinžinerija (*angl. re-engineering*) apima paveldėtų (egzistuojančių, liktinių) programų analizę ir perkūrimą į naują formą ir naujos formos realizavimą [16].

Rekonstrukcijos sąvoka, standžiai susieta su paveldėtomis informacinėmis sistemomis, ir su perkūrimo procesais. Šis terminas yra paprastai naudojamas vietoje rekonstrukcijos ([17] [18]). Tokiu atveju atsiranda nesusipratimo šitų dviejų sąvokų dėl terminologijos standartizacijos trūkumo, kuri bus panaudota palyginti naujoje disciplinoje. Perkūrimas yra suprantamas esamos sistemos pokytis, kad atkurtų ją naujoje formoje. Todėl perkūrimas galų gale priveda prie beveik užbaigto paveldėtos sistemos realizavimo iš naujo ([11][19]). Sukurta sistema gali ar galbūt neveiks ant skirtingos aplinkos. Saviti rekonstrukcijos tikslai išvengia užbaigto paveldėtos sistemos perkūrimo, pakartotinai panaudodami realizaciją, projektą, specifikaciją, reikalavimus, kiek galima daugiau. Be to, nauja sistema, kylanti iš rekonstrukcijos proceso, atliks perėjimus ant skirtingos aplinkos (galbūt tiktai skirtinga programavimo kalba, ar galbūt visiškai nauja architektūra ir technologija). Jei didžioji dalis paveldėtos sistemos turi būti išmetama, nėra pakartotiniai panaudojami komponentai), inžinierius susidurs su projekto perkūrimu, o ne projekto rekonstrukcija.

Rekonstrukcijos tikslas yra suvokti paveldėtąją sistemą (reikalavimus, specifikaciją, sistemos architektūrą, realizaciją) ir perrašyti naujoje formoje, atliekant sistemos pakeitimus siekiant pridėti ar pagerinti sistemos funkcionalumą, našumą ar realizaciją.

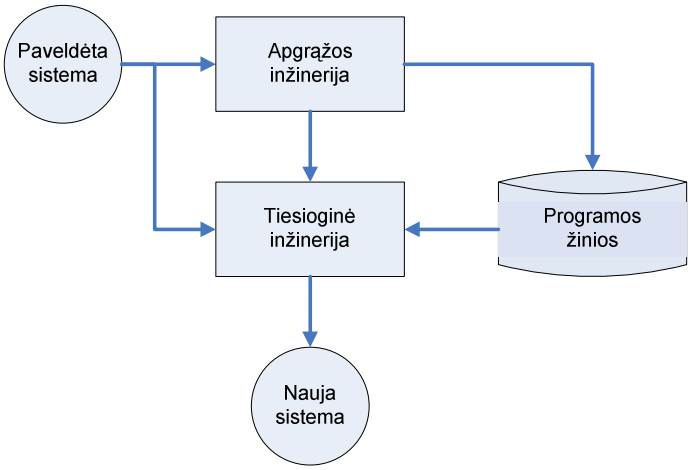
Anksčiau analizuoti metodai turi trūkumų. Jų pagrindinis tikslas yra modernizuoti paveldėtą sistemą nekeičiant jos funkcionalumo, tiesiog pritaikant ją prie naujos aplinkos ar pagerinant palaikomumą ar charakteristikas.

Toliau nagrinėsime rekonstrukcijos procesą plačiau apžvelgdami jo metodus.

3.3. Paveldėtų sistemų rekonstrukcija

Programinės įrangos palaikymas yra neišvengiamas procesas atliekant programos vystymą. Adaptyvus palaikymas yra procesas, panaudotas, kad pritaikytų programinę įrangą prie naujų aplinkų ar naujų reikalavimų dėl besiplėtojančio poreikio naujų verslo reikalavimų, platformų, naujų operacinių sistemų, naujos programinės įrangos, ir vystymosi. Programos turi vystytis greitai, kad atitiktų verslo reikalavimų pasikeitimus, nestatydamos į pavojų sistemos kokybės. Rekonstrukcija, taip pat žinoma kaip PĮ suremontavimas (*angl. renovation*) ir pataisymas, atstatymas (*angl. reclamation*), prasideda nuo sistemos analizės ir sistemos adaptacijos, pagrįstos žiniomis, gautomis iš naujos aplinkos analizės [18]. Iš esmės,

rekonstrukcija gali būti apibrėžta kaip apgrąžos inžinerija po kurios seka tiesioginė inžinerija [20]. Apgrąžos inžinerija yra procesas analizuojantis sistemą, kurio tikslas programos supratimas. Tiesioginis projektavimas yra tradicinis programinės įrangos išsivystymo procesas, apimantis reikalavimus ir specifikaciją, projektą, kodavimą, ir testavimą. Rekonstrukcija kaip rodytas 3.5 iliustracijoje panaudoja apgrąžos inžinerijos techniką, kad gautų žinias sistemų ir paskui panaudotų šią informaciją, kad padėtų sukurti sistemą naujoje formoje atliekant tiesioginės inžinerijos fazes.



3.5 pav. *Rekonstrukcijos procesas*

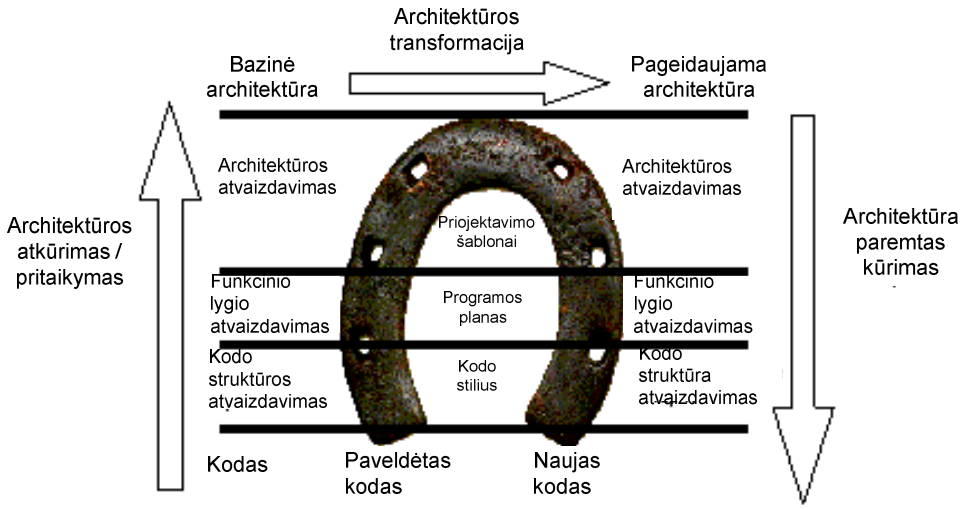
Paveldėtos sistemos rekonstrukcija apima didelę dalį projektavimo iš naujo. Kitaip sakant: rekonstrukcija nėra tik kodo transformacija. Kad integruotume kodo lygmenį ir architektūrinį rekonstrukcijos vaizdą, yra įvedamas „pasagos“ (*angl. horseshoe*) modelis ([18][21]). Šis konceptualus modelis atskiria skirtingus analizės lygmenis ir aprūpina rekonstrukcijos pagrindą kiekviename lygmenyje, o ypač transformacijai į architektūrinį lygmenį.

Rekonstrukcijos pagrindą sudaro trys procesai:

1) paveldėtos sistemos analizė,

2) loginė transformacija,

3) naujos, rekonstruotos sistemos išsivystymas.



3.6 pav. *Sistemos renovacijos pasagos modelis*

Pirmoje veikloje verslo logika turi būti ištraukta per gilią paveldėtos sistemos analizę. Procesas gali būti suskirstytas į dvi sritis „duomenų bazės apgrąžos inžinerijos“ ir „procedūros apgrąžos inžinerijos“. Duomenų bazės apgrąžos inžinerija gali būti atlikta panaudojant duomenų bazės apgrąžos inžinerijos įrankius. Procedūros apgrąžos inžinerijai reikalinga esamos sistemos architektūros analizė ir supratimas (suvokimas) [22]. Kad galėtume atlikti architektūrinius pakeitimus, kad patenkintume naujus reikalavimus ir pašalintume sistemos trūkumus.

Iš esmės, pirma veikla atkuria architektūrą, ištraukdama artefaktus iš programos kodo [23]. Ši atkurta architektūra yra analizuojama, kad nustatoma, ar ji atitinka esamo projekto architektūrą. Atkurta architektūra taip pat įvertinama keliomis kokybės metrikomis, tokiomis kaip našumas, keičiamumas, saugumas, ar patikimumas.

Antra veikla yra architektūrinė transformacija. Šiuo atveju, „kaip – yra“ (*angl. „as-is“*) architektūra yra atkuriama ir paskui iš naujo perprojektuojama, kad pataptų reikalaujama nauja architektūra. Tuomet vėl įvertinama ar atitinka sistemos kokybės tikslus ir kitus organizacinius ir ekonominius apribojimams.

Trečia pasagos veikla yra architektūra paremtas kūrimas pagal pageidaujamą architektūrą. Šiame procese, išsprendžiamos svarstomos problemos, ir yra pasirenkamos tarpusavio ryšio strategijos. Paveldėtos sistemos kodo lygmens artefaktai yra perrašomi, kad atitiktų šią naują architektūrą [24].

Pasagos modelis skiria tris skirtingus lygmenis. Pirmas, ar pagrindinis, yra kodo lygmuo, kuris apima programos tekstą ir artefaktus tokius kaip abstraktūs sintaksės medžiai ir srauto diagramos. Antras yra funkcijos lygmuo, kuris apibūdina santykį tarp programos funkcijų, duomenų, ir failų. Trečiasis yra koncepcijos lygmuo, kuriame susibūrime į grupes ir funkcijos, ir kodo lygmens artefaktai yra surinkami į architektūrinio lygmens komponentų

struktūrą. Veiklos esančio aplink pasagą atvaizduoja rekonstrukcijos abstrakčią formą. Praktiškai yra du papildomi sutrumpinimai, kurie sutrumpina pasagą, ir taip leidžia pereiti iš esamos, į pageidaujamą sistemą. Šitie pasagos kelio "sutrumpinimai" gali pateikti pragmatinius pasirinkimus, pagrįstus technologiniais, organizaciniais, ar su procesu susijusiais apribojimais.

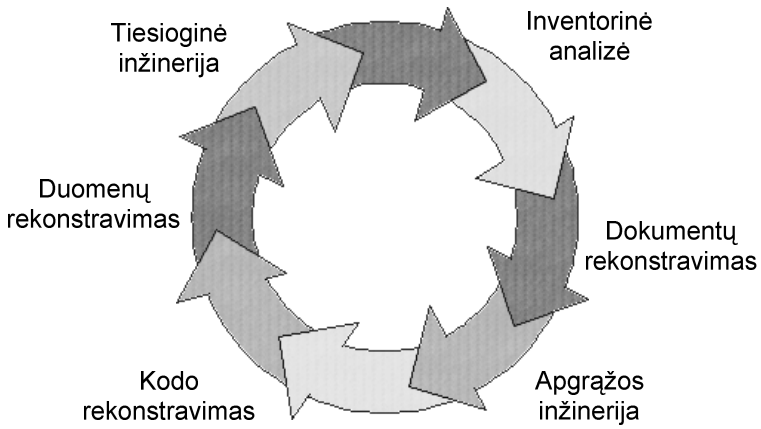
Specialiai sudėtingų sistemų architektūros rekonstrukcijai, kad suprastume atkurtą architektūrą, yra naudojamas rankinis sistemos modeliavimas, siekiant atlikti architektūrinę transformaciją su naujais architektūriniais kokybės reikalavimais.

Kitas svarbus rekonstrukcijos aspektas yra testavimas. Be deramos testavimo aplinkos, rekonstrukcijos procesas yra neefektyvus ir kartais nesėkmingas.

3.4. Rekonstrukcijos proceso modelis

Programos rekonstrukcijos modelis yra sudarytas iš sekančių metodų [25]:

- Inventorinė analizė
- Dokumentų rekonstravimas
- Apgrąžos inžinerija
- Kodo rekonstravimas
- Duomenų rekonstravimas
- Tiesioginė inžinerija



3.7 pav. *Rekonstrukcijos proceso modelis*

Svarbu paminėti, jog rekonstrukcijos modelis nenusako nuo kurio proceso reikia pradėti.

3.4.1. Inventorinė analizė

Inventorinė analizė kitaip dar vadinama resursų analize. Pagrindinai jos tikslai yra įvertinti analizuoti programos atributus: programos dydį, amžių, verslo kritiškumą bei kt. Rūšiuodami informaciją pagal svarbumą verslui, ilgaamžiškumą, atliktas palaikymo procedūras mes ruošiame sistemą rekonstrukcijai. Svarbu paminėti, kad sistemos resursų

analizė turi būti atliekama reguliariai, kadangi bėgant laikui sistemos vertė verslui gali pasikeiti.

3.4.2. Dokumentacijos rekonstravimas

Šio proceso metu atliekamas sistemos dokumentacijos stovio įvertinimas. Dažniausia ji yra nepilna, skurdi arba jos išvis nėra. Jeigu ji egzistuoja - būna palyginti nusistovėjusi, daugiau nebenaudojama. Atliekant sistemos rekonstrukciją dokumentacija yra atnaujinama, po kiekvienos atliktų pakeitimų iteracijos, bei užbaigiama. Dokumentacijos atkūrimas yra labai daug laiko reikalaujantis procesas.

3.4.3. Apgrąžos inžinerija

Atvirkštinė inžinerija (*angl. reverse engineering*) yra programų analizės procesas, kuris siekia sukurti programos atvaizdavimą aukštesniame abstrakcijos lygmenyje negu išeities kodas. Metodo tikslas panaudoti įrankius ir metodus siekiant automatiškai išgauti informacijai apie egzistuojančios programos duomenis ir architektūrą. Gauti rezultatai dažniausia yra atvaizduojami pasitelkiant UML diagramas. Šio proceso metu vykdomas projekto ir specifikacijos atkūrimas bei sukuriamas sistemos dokumentas kurį sudaro:

- Programos struktūros diagramos (grafai)
    - Kreipinių į procedūras/funkcijas
    - Valdymo srautų diagramos
- Duomenų struktūros diagramos
- Patobulintos grafinės sąsajos

3.4.4. Kodo rekonstravimas

Tai dažniausiai naudojamas būdas. Proceso metu analizuojama programos išeities kodas. Dažnai naudojami reorganizavimo įrankiai, tam, kad būtų galima atlikti automatinį reorganizavimas. Toliau yra apžvelgiami bandymų rezultatai, atnaujinama kodo dokumentacija.

3.4.5. Duomenų rekonstravimas

Duomenų rekonstravimas dažniausiai prasideda nuo apgrąžos inžinerijos kai būna apžvelgti ir apibrėžti duomenų modeliai ir struktūra. Šio proceso metu atkuriama duomenų architektūra. Yra nustatomi duomenų objektai bei požymiai. Duomenų architektūra turi didelę įtaką sistemos architektūrai ir algoritmams, kuriuos ji įgyvendina. Duomenų rekonstravimui reikalingi architektūros ar programos kodo pakeitimai.

3.4.6. Tiesioginė inžinerija

Tiesioginė inžinerija (*angl., forward enginering, forward re-engineering, renovation re-factoring*) – procesas, kai programų inžinerijos principai, koncepcijos ir metodai taikomi tam, kad perkurti egzistuojančią sistemą iš naujo [26].

Idealiu atveju šis procesas turi būti atliktas automatizuotai. Paveldėta sistema būtų analizuojama, pertvarkoma ir paskui atkuriama naujoje formoje, bei įvertinus atitiktų geriausius kokybės aspektus. Šiuo metu tokio mechanizmo nėra.

Tiesioginė inžinerija dar vadinama suremontavimu ar atstatymu [26]. Jos metu ne tik išgaunama projekto informacija iš egzistuojančios programinės įrangos, bet taip pat ši informacija yra panaudojama, kad pagerinti sistemos kokybę. Tai - ne tik senos programos naujo ekvivalento sukūrimas, tai taip pat naujų reikalavimų (vartotojų, technologinių) integravimas į rekonstrukciją. Perkurta programa išplečia senos programos taikymo galimybes, pagerinant bendrą sistemos našumą bei kokybę [27].

4. SISTEMOS REKONSTRUKCIJOS REALIZACIJA

Šiame skyriuje pateikiamos esminės rekonstruotos butų savininkų bendrijų informacinės sistemos specifikacijos dalys: sistemai keliami funkciniai ir nefunkciniai reikalavimai, sistemos architektūra, realizacijos ypatumai.

4.1. Projekto tikslai

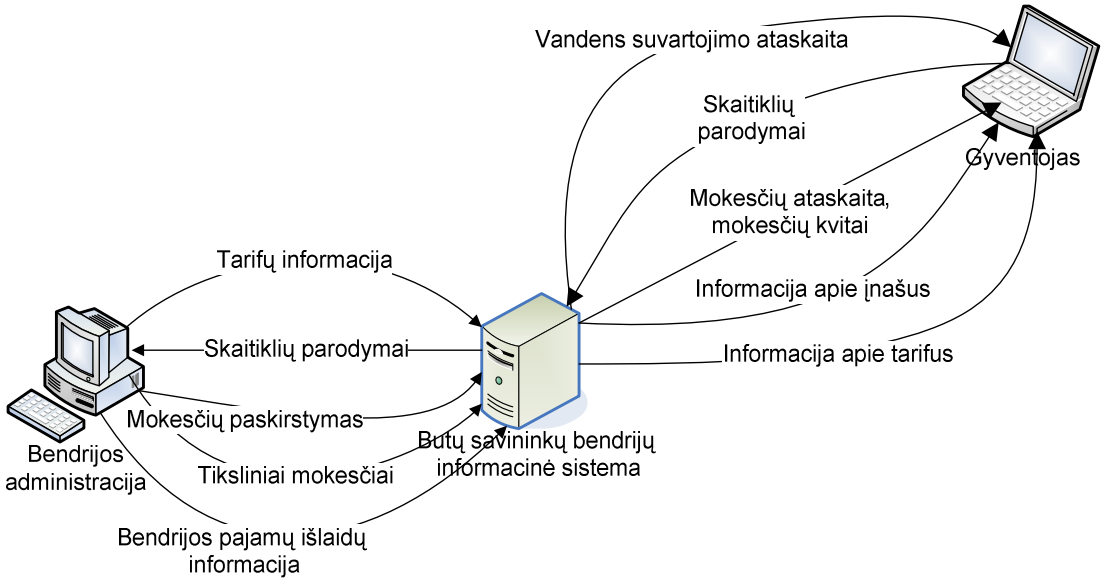
Projektas yra skirtas rekonstruoti butų savininkų bendrijų informacinę sistemą atliekančią bendrijos pajamų-išlaidų apskaita. Pagal mokesčių gavėjų pateiktas sąskaitas ir gyventojų pateiktus vandens skaitiklių rodmenis mokesčiai turi būti paskirstomi bendrijos gyventojams, pateikiami mokesčių kvitai apmokėjimui. Bendrijos gyventojai skaitiklių rodmenis pateikia internetu arba atskiru pranešimu, paskaičiuotus mokesčius gauna internetu arba atskiru pranešimu.

Projekto tikslas yra rekonstruoti paveldėtą butų savininkų bendrijų informacinę sistema su nauja technologija, atlikti funkcinius pakeitimus, kad palengvintų bendrijos duomenų kaupimą, saugojimą, valdymą bei apskaitą.

Užsakovui, butų savininkų bendrijai „Tuopa" bus pateikta išbaigta sistema, kuri bus parankiu technologiniu įrankiu, padėsiančiu žymiai produktyviau bei paprasčiau atlikti butų savininkų mokesčių apskaitą.

4.2. Veiklos kontekstas

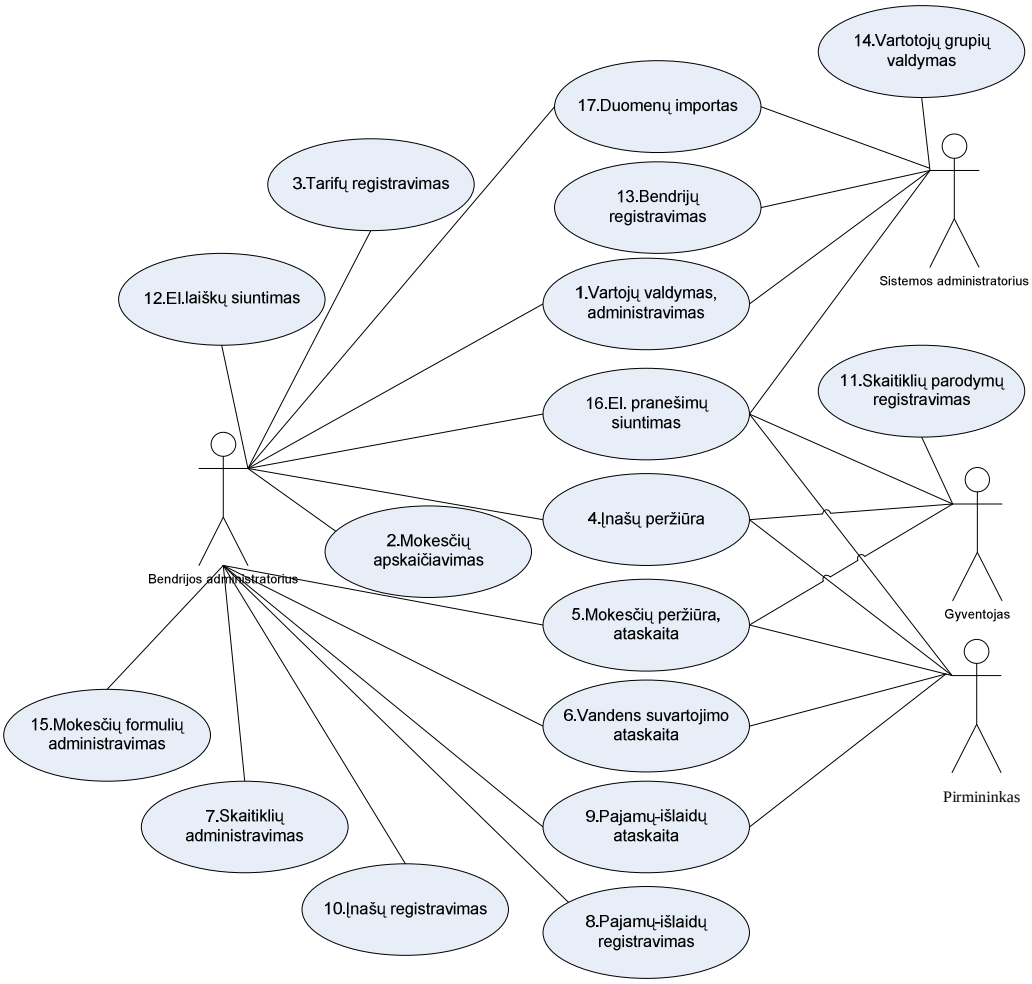
Žemiau, 4.1 paveikslėlyje pateikiamas butų savininkų bendrijų informacinės sistemos veiklos kontekstas. Kontekste atsispindi sistemos įėjimo ir išėjimo duomenys, galime susidaryti sistemos funkcinių reikalavimų bendrą vaizdą.



4.1 pav. *Sistemos veiklos kontekstas*

4.3. Sistemos panaudojimo atvejai

Žemiau, 4.2 paveikslėlyje pavaizduota panaudos atvejų diagrama, kuri atspindi ribas tarp kuriamos sistemos bei vartotojų.



4.2 pav. *Panaudos atvejų diagrama*

4.3.1. Aktoriai:

1. **Sistemos administratorius:** sistemos administratorius, gali atlikti su pačios sistemos administravimu susijusias operacijas: vartotojų valdymą (trinti redaguoti, įvesti naują), vartotojų grupių valdymą, registruoti naujas bendrijas sistemoje, bei importuoti duomenis.
2. **Bendrijos administratorius:** Bendrijos administratorius, kuris turi visas teises savo bendrijoje. Gali atlikti vartotojų valdymą (trinti redaguoti, įvesti naują), tarifų įvedimą, mokesčių apskaičiavimą, gauti gyventojų įnašų, mokesčių, vandens suvartojimo ataskaitą, registruoti bendrijos pajamas bei išlaidas, registruoti gyventojų įnašus, gauti bendrijos pajamų išlaidų ataskaitą, administruoti bendrijos mokesčius.
3. **Bendrijos pirmininkas** – vartotojo tipas skirtas bendrijos pirmininkui, ar kitam asmeniui kuris turi teises tik peržiūrėti bendrijos informaciją, bet negali jos keisti, trinti,

ar vykdyti kitų administravimo operacijų.

4. **Gyventojas:** bendrijos narys, buto savininkas. Prisijungęs prie sistemos gali registruoti skaitiklių parodymus, gauti ataskaitą apie savo mokesčius, įnašus, vandens sunaudojimą.

4.3.2. Panaudos atvejai:

1. **Vartotojų valdymas, administravimas**: registruojama nauji vartotojai, redaguojama esamo vartotojo informacija arba šalinami vartotojai iš sistemos.
2. **Mokesčių apskaičiavimas:** apskaičiuojami ir paskirstomi mokesčiai pagal esamus mėnesio tarifus. Mokesčių apskaičiavimas gali būti vykdomas pagal gyvenamą plotą, gyventojų skaičių, butus ir kitus parametrus.
3. **Tarifų registravimas:** įvedami bei esant reikalui koreguojami tarifai ir mokesčių skaičiavimo parametrai.
4. **Įnašų peržiūra:** pateikiama gyventojų arba konkretaus asmens įnašų atskaita, pasirinktam laikotarpiui.
5. **Mokesčių peržiūra, ataskaita:** pateikiama gyventojų arba konkretaus asmens mokesčių atskaita, pasirinktam laikotarpiui.
6. **Vandens suvartojimo ataskaita:** pateikiama gyventojų arba konkretaus asmens vandens suvartojimo atskaita, pasirinktam laikotarpiui.
7. **Skaitiklių administravimas:** registruojami nauji skaitikliai, skaitikliai priskiriami gyventojams.
8. **Bendrijos pajamų-išlaidų registravimas:** registruojamos bendrijos gauto pajamos, arba išlaidos.
9. **Pajamų-išlaidų ataskaita:** sudaroma ataskaita apie pajamų ir išlaidų balansą pasirinktam laikotarpiui.
10. **Įnašų registravimas:** atliekamas gyventojų įnašų, už paskaičiuotus mokesčius registravimas.
11. **Skaitiklių parodymų registravimas:** registruojami gyventojų skaitiklių parodymai.
12. **El. laiškų siuntimas:** sistema siųs pranešimus elektroniniu paštu visiems pamiršusiems užregistruoti vandens skaitiklių ar sumokėti mokesčių.
13. **Bendrijos registravimas**: sistemoje registruojama nauja bendrija. Sistemoje gali būti vykdoma kelto bendrijų apskaita.
14. **Vartotojų grupių valdymas** : naujų vartotojų grupių įtraukimas, redagavimas, teisių priskyrimas.
15. **Mokesčių formulių administravimas** : galimybė keisti mokesčių apskaičiavimo formules.

16. **El. pranešimų siuntimas:** sistemos vartotojai gali siųsti pranešimus vieni kitiems arba bendrijos administracijai.
17. **Duomenų importas:** bendrijos duomenų importas iš paveldėtų duomenų bazių. Realizuotas duomenų importas iš DBF formato.

4.4. Funkciniai reikalavimai

Šiame skyriuje pateikiami sutrumpintas sistemos funkcinių reikalavimų sąrašas:

1. Sistemoje turi buti 4 kategorijų vartotojai (sistemos administratorius, bendrijos administratorius, pirmininkas ir gyventojas).
2. Sistema turi apskaičiuoti ir paskirstyti mokesčius gyventojams pagal esamus mėnesio tarifus.
3. Sistemoje turi būti galimybė keisti tarifų dydžius. Pagal kuriuos skaičiuojami mokesčiai.
4. Sistema turi teikti ataskaitas apie gyventojų įnašų informaciją.
5. Sistema turi teikti ataskaitas apie gyventojų mokesčių informaciją.
6. Sistema turi teikti ataskaitas apie vandens suvartojimą.
7. Sistemoje turi būti galimybė įvesti tikslinį mokestį, kuris paskirstomas visiems bendrijos gyventojams.
8. Sistema turi registruoti bendrijos pajamas bei išlaidas.
9. Sistema generuoti ataskaita apie bendrijos pajamų išlaidų balansą.
10. Sistema turi registruoti gyventojų įnašus.
11. Sistema turi leisti gyventojams registruoti skaitiklių parodymus.
12. Sistemoje turi būti galimybė siųsti elektroninius pranešimus registruotiems bendrijos gyventojams.
13. Gyventojai turi turėti galimybę gauti savo sumokėtų įnašų informaciją.
14. Gyventojas turi turėti prieigą prie savo mokesčių informacijos.
15. Bendrijos administratorius turi galimybę priskirti skaitiklius gyventojams.
16. Sistemoje gali būti užregistruotos kelios bendrijos.

4.5. Nefunkciniai reikalavimai

Šiame skyriuje pateikiami sistemos nefunkcinių reikalavimų sąrašas.

Reikalavimai sistemos išvaizdai

Bendri sistemos vartotojo sąsajai keliami reikalavimai:

1. Paprasta, nesudėtinga
2. Aiški navigacija

3. Neįkyri
4. Sąveikaujanti
5. Neapkrauta
6. Profesionali išvaizda

Reikalavimai panaudojamumui

7. Paprastas naudojimas (sėkmingas pasinaudojimas, be apmokymų, be aprašo)
8. Darbo našumo padidinimas įdiegus sistemą
9. Galimybė dirbti su viena ranka
10. „Karštieji“ (*angl. hotkeys*) klavišai

Reikalavimai vykdymo charakteristikoms

11. Užduočių vykdymas spartus, kitaip sistema pradės erzinti vartotoją
12. Tikslumas
13. Resursų panaudojimo efektyvumas
14. Patikimumas
15. Išplečiamumas

Reikalavimai veikimo sąlygoms

16. Perkėlus sistemą į kita aplinką, nesutrinka sistemos darbas
17. Neprarandami duomenys nutrukus interneto ryšiui

Reikalavimai sistemos priežiūrai

18. Mažos priežiūros sąnaudos
19. Sistemos veikimas kitose operacinėse sistemose
20. Sistemos pritaikymas kitoms kalboms

Reikalavimai saugumui

21. Konfidencialumas – prie sistemos duomenų gali prieinami tik teisėtų asmenų;
22. Vientisumas – į sistemą įvesti duomenys neiškraipomi;
23. Pasiekiamumas - galimybė pasinaudoti duomenimis teisėtiems vartotojams.

Kultūriniai-politiniai reikalavimai

24. Sistema gali būti pritaikyta kitoms kalboms

Teisiniai reikalavimai

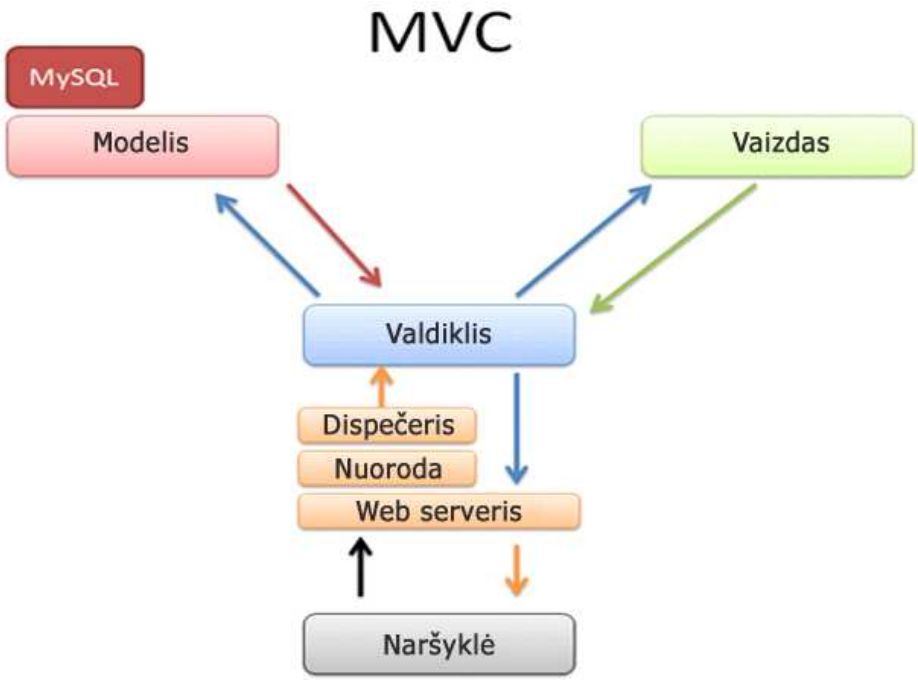
25. Sistema turi veikti pagal Lietuvos Respublikoje galiojančius įstatymus

4.6. Architektūros specifikacija

4.6.1. Bendras architektūros vaizdas

Butų savininkų bendrijų informacinės sistemos kūrime panaudotas trijų architektūrinių lygių modelis. Sistemoje pritaikytas dažniausia projektavime naudojamas Modelis-Vaizdas-Valdikis (*angl. Model-View-Controller - MVC*) šablonas. Šis šablonas skirtas atskirti verslo logika (*Model*) nuo vaizdavimo logikos(*View*). Valdiklis (*Controller*) yra atsakingas, kad kartu sujungti modelio ir vaizdo komponentus [29].

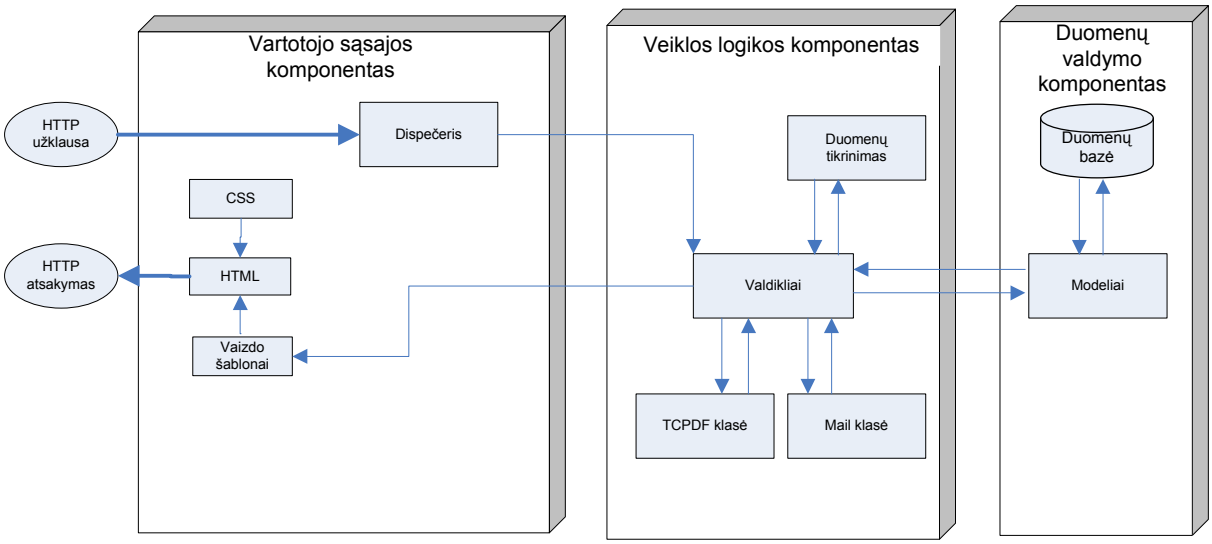
MVC šablonas garantuoja atskyrimą tarp skirtingų sluoksnių. Atskirdamas vaizdo komponentą galima keisti sistemos išvaizdą nelendant į visą biznio logiką. Kadangi modelis yra visiškai atskirtas, tokiu būdu jis gali būti išbandytas nepriklausomai nuo vartotojo sąsajos. Valdiklio logika yra tokia, jog turi sujungti modelį ir vaizdą į vieną sistemą. MVC architektūra duoda sistemai patogų palaikomumą.



4.3 pav. *Modelis-Vaizdas-Valdiklis šablonas*

4.6.2. Sistemos architektūra

Žemiau, 4.4 paveikslėlyje yra atvaizduoti pagrindiniai sistemos komponentai:



4.4 pav. *Komponentų išdėstymo diagrama*

Vartotojo sąsajos komponentas – komponentas atsakingas už duomenų atvaizdavimą vartotojui. Vartotojo sąsajos komponentas pateikia užklausos rezultatą HTML kodo pavidalu. Jis yra sudarytas iš dispečerio klasės, kuri pagal nuorodą nukreipia į tam tikrą veiklos logikos komponentą, bei vaizdų. Kiekvienai operacijai yra sukuriamas vaizdo šablonas, kuris atvaizduoja rezultatus gautus iš valdiklio ir paverčia juos į HTML žymėjimo kalbą. HTML išvestis yra siunčiama į kliento interneto naršyklę ir ten duomenys atvaizduojami, panaudojant CSS stilių valdymo kalbą.

Veiklos logikos komponentas - komponentas atsakingas sistemos valdymą, duomenų atvaizdavimo pateikimą pagal atitinkamus vartotojo veiksmus: mygtuko paspaudimus, teksto įvedimus ir t.t. Veiklos logikos komponentas sudarytas iš valdiklių, duomenų tikrinimo klasės, „*TCPDF*“ klasės kuri skirta generuot užklausas PDF formatu, bei „*Mail*“ klasės, kuri atsakinga už elektroninių laiškų siuntimą vartotojams. Komponentas iš vartotojo sąsajos gavęs veiksmus juos apdoroja. Visi komunikacijos veiksmai su duomenų baze apdorojami per tam tikras modelio klases.

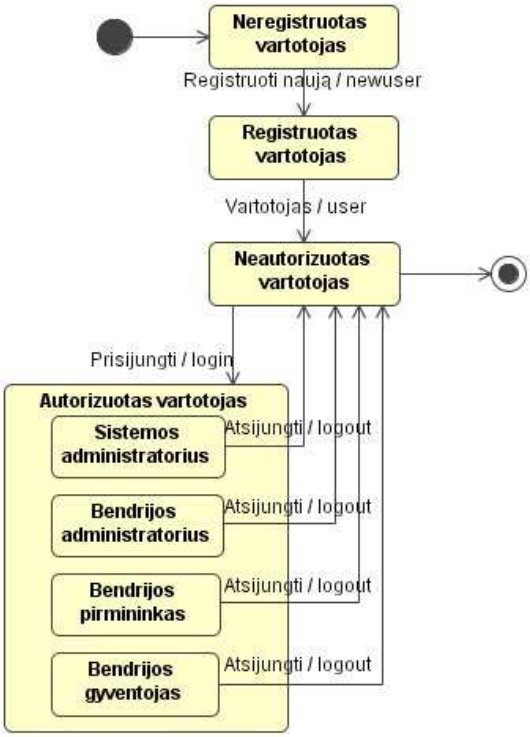
Duomenų valdymo komponentas – komponentas atsakingas už veiksmus su duomenimis. Komponentą sudaro duomenų bazė bei modelių klasės. Kiekvienas duomenų objektas turi savo modelį. Duomenų valdymo komponentas sąveikauja su veiklos logikos komponentu. Pagal atitinkamus vartojo užklausas tam tikras valdiklis kreipiasi į modelį su užklausa vykdyti vieną iš *CRUD* (***C****reate*, ***R****ead*, ***U****pdate*, ***D****elete*) operacijų ir gražinti rezultatus.

4.7. Sistemos dinaminis vaizdas

Šiame skyriuje pateikiamos esminės sistemos būsenų, veiklos ir sekų UML diagramos.

4.7.1. Būsenų diagramos

Žemiau esančiose diagramose atvaizduojamos vartotojo prisijungimo būsenų diagrama.

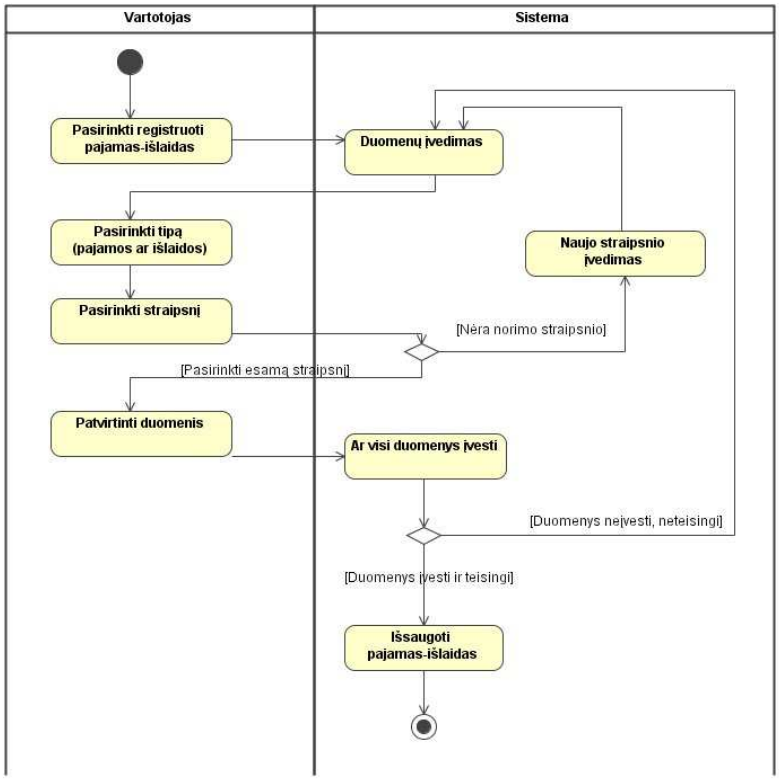


4.5 pav. *Vartotojų prisijungimo būsenų diagrama*

Diagramoje atvaizduotos vartotojų būsenos ir kaip jos keičiasi atliekant įvairius veiksmus. Apibendrintoje „Autorizuotas vartotojas“ būsenoje matosi visi sistemos vartotojų tipai pagal lygmenis.

4.7.2. Veiklos diagramos

Žemiau esančiame paveikslėlyje pavaizduota bendrijos pajamų arba išlaidų registravimo veiklos diagrama. Ši diagrama atvaizduoja tiek bendrijos išlaidų tiek ir bendrijos pajamų registravimą, kadangi pasirenkamas įvedamos informacijos tipas.

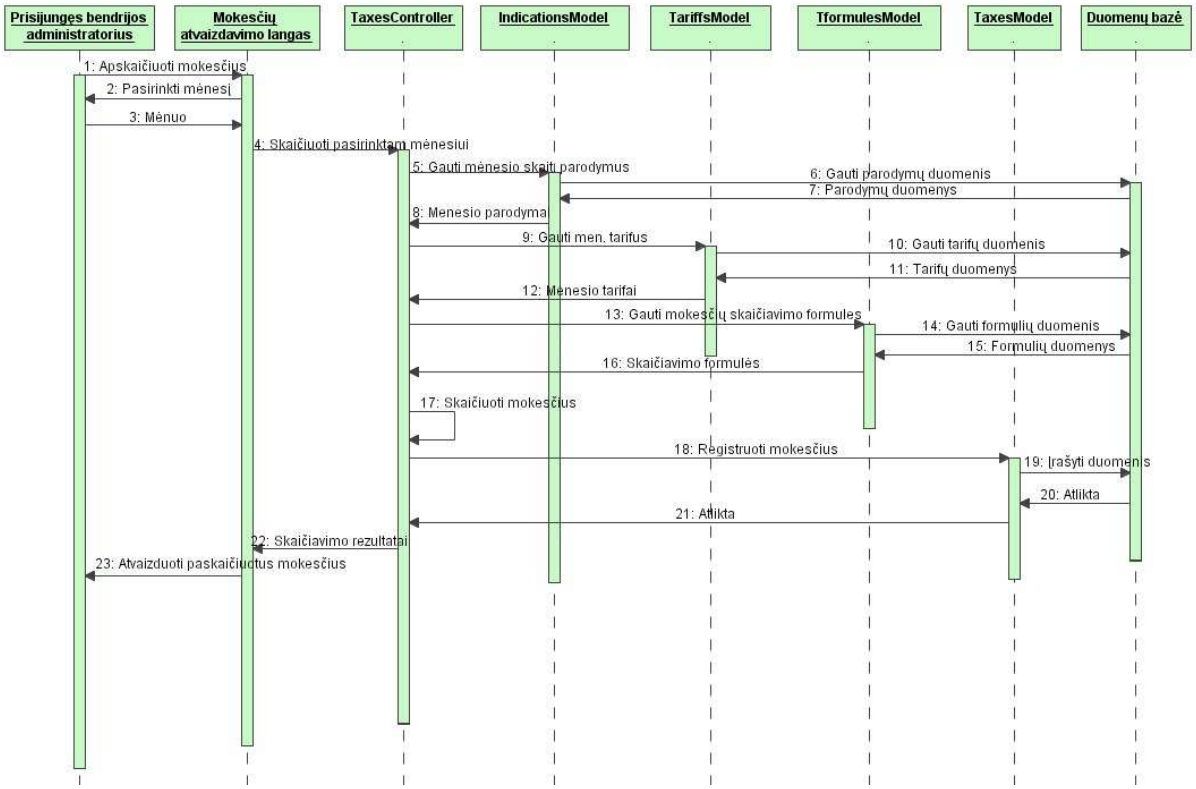


4.6 pav. *Bendrijos pajamų-išlaidų registravimo veiklos diagrama*

Iš diagramos matome įvedant duomenis vartotojas turi nurodyti duomenų tipą bei straipsnį, kuris nusako kokia buvo tų pajamų ar išlaidų paskirtis (pvz. „Už šilumą“, „Kanceliarinės prekės“ ir t.t.). Jeigu nėra norimo straipsnio – galima sukurti naują. Atlikus duomenų įvedimą sistema tikrina ar visi duomenys įvesti ir ar jie teisingi (tikrinamas duomenų tipas, formatas). Jei duomenys teisingi - tuomet registruojami duomenų bazėje, jei ne - turime juos pataisyti duomenų įvedimo lange.

4.7.3. Sekų diagramos

Čia pateikiama viena iš sistemos panaudojimo atvejų sekų diagramų. Kadangi sistemoje yra naudojamas MVC projektavimo šablonas, likusios sąveikos diagramos savo struktūra yra panašios. Žemiau, 4.7 paveikslėlyje pateikiama mokesčių apskaičiavimo sekų diagrama.

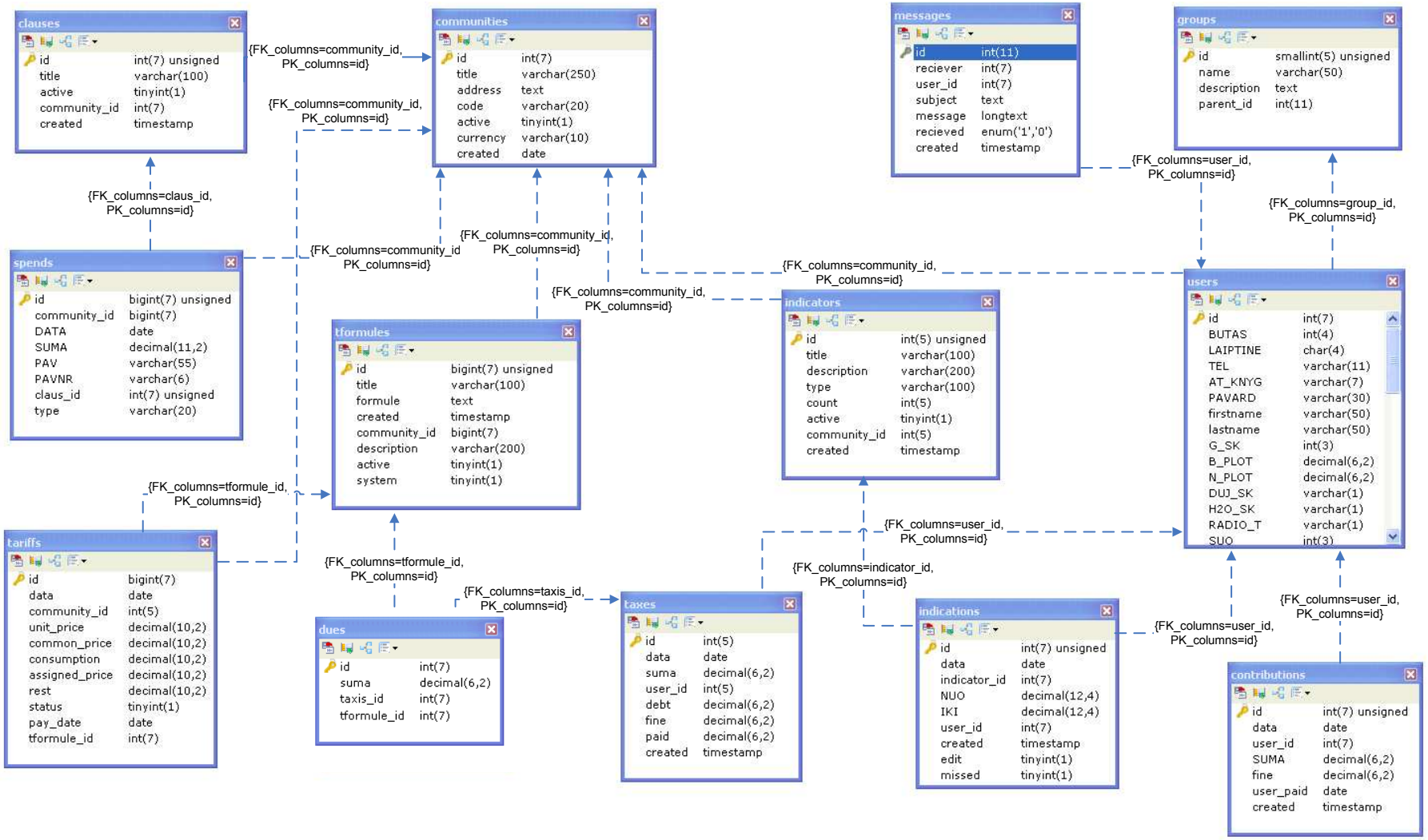


4.7 pav. *Mokesčių apskaičiavimo sąveikos diagrama*

Iš sekų diagramos matome, visų mokesčių apskaičiavime dalyvaujančių klasių tarpusavio bendravimo eiliškumą. Diagramoje matoma pagrindinis mokesčių valdiklis kuris skaičiuoja mokesčius prieš tai kreipdamasis į kitus modelius, kad gautų skaitiklių parodymų, mėnesio tarifų informaciją. Matome, kad atitinkamas duomenų objekto modelis yra tarsi tarpininkas, kuris bendrauja su duomenų baze. Atlikus skaičiavimus gauti mokesčiai registruojami duomenų bazėje ir atvaizduojami vartotojui.

4.8. Duomenų vaizdas

Duomenų bazei panaudota MySQL realiacinė duomenų bazės valdymo sistema. Viso duombazę sudaro 13 lentelės. Lentelės yra susietos išoriniai raktais. Buvo siekiama kuo mažiau keisti lentelių laukų pavadinimus, kad kuo lengviau butų atlikti sistemos duomenų perkėlimą. Senieji laukų pavadinimai lentelėje atvaizduoti didžiosiomis raidėmis. Visi nauji laukai atvaizduoti mažosiomis. Žemiau, 4.8 paveikslėlyje pateikiamas butų savininkų bendrijų informacinės sistemos duomenų bazės modelio vaizdas.



4.8 pav. *Duomenų bazės modelis*

Duomenų bazių esybių aprašas pateikiamas sekančioje 4.1lentelėje:

4.1 Lentelė. *Duomenų bazės modelio esybės*

Lentelės pavadinimas	Aprašymas
comunities	Saugoma informacija apie bendrijas
users	Saugoma informacija apie vartotojus
messages	Saugomos vartotojų siųsti pranešimai
groups	Saugoma vartotojų grupių informacija
contributions	Saugoma įnašų už mokesčius informacija
indicators	Saugoma informacija apie vartotojų skaitiklius
indications	Saugoma visų skaitiklių parodymų informacija
taxes	Saugoma gyventojų paskaičiuotų mokesčių informacija
dues	Saugoma atskirų mokesčių informacija
tformules	Saugomos mokesčių paskaičiavimo formulių informacija
tariffs	Saugoma mėnesinių tarifų informacija
clauses	Saugomi bendrijos išlaidų ar pajamų straipsnių informacija
spends	Saugoma bendrijos išlaidų ar pajamų informacija

4.9. Praktinis darbo rezultatas

Projekto vykdymo metu buvo sukurta programinės įrangos techninė dokumentacija. Visą dokumentų sąrašą sudaro:

- projekto paraiškos dokumentas;
- projektavimo metodologijos ir technologijos analizė;
- detalus projekto planas;
- reikalavimų specifikacija;
- architektūros specifikaciją;
- detalios architektūros specifikacija;
- programinės įrangos licencija;
- testavimo planas;
- kokybės analizės dokumentas;
- vartotojo dokumentacija.

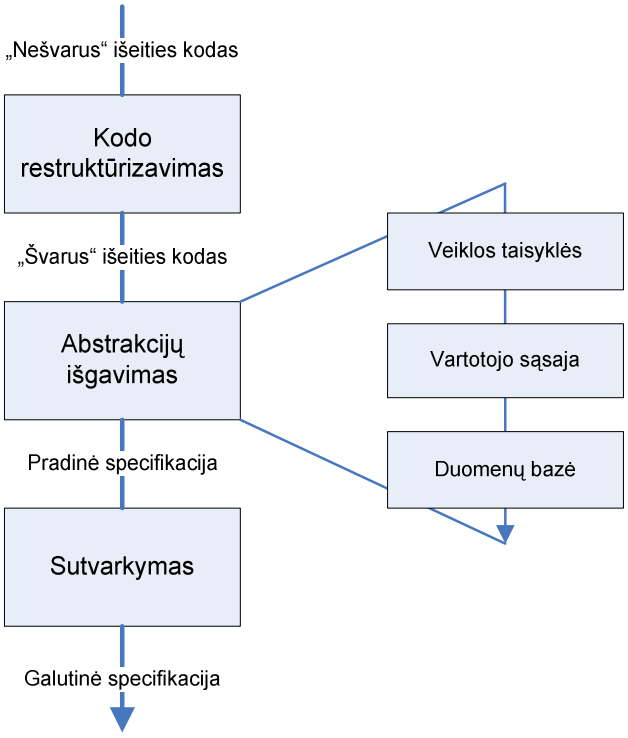
5. PAVELDĖTOS SISTEMOS REKONSTRUKCIJA REMIANTIS APGRĄŽOS IR TIESIOGINĖS INŽINERIJOS PRINCIPAIS

Kadangi paveldėta sistema neturėjo dokumentacijos, pirmiausia buvo analizuojamas programos kodas, kad iš sistemos realizacijos gauti sistemą abstrakčiame lygmenyje nei išeities kodas. Tam buvo pasinaudota apgrąžos rekonstrukcijos metodu.

5.1. Apgrąžos rekonstrukcija

Atliekant rekonstrukciją buvo siekiama išanalizavus sistemos kodą gauti sistemos struktūrą bei veiklos taisykles. Dažniausiai tam pasitelkiami automatizuoti įrankiai. Įrankio kuris analizuotų paveldėtą FoxPro programavimo kalbą rasti nepavyko. Todėl rekonstrukcija buvo atliekama rankiniu būdu gilinantis į sistemą, siekiant ją suvokti. Kadangi sistema neturėjo jokios dokumentacijos šis procesas buvo itin sudėtingas bei kainavo daug resursų.

Atliktą apgrąžos rekonstrukciją galima atvaizduoti sekančia iliustracija:



5.1 pav. *Apgražos rekonstrukcijos procesas*

Prieš atliekant apgrąžos inžinerijos veiksmus, „nešvarus“ FoxPro kodas buvo restruktūrizuojamas, kad būtų matomos programinės konstrukcijos. Toks pertvarkymas daro programos kodą lengviau skaitomą ir sukuria pagrindą apgrąžos inžinerijos veiksmui, kuris eina po to.

Analizuojant iš išeities kodo buvo bandoma ištraukti reikšmingas veiklos taisykles, vartotojo sąsają, duomenų struktūras ir duomenų bazę, kuri yra panaudota.

5.1.1. Veiklos taisyklių suvokimas

Kad suprastume sistemos procedūras ir veiklos taisykles, kodas buvo analizuojamas įvairiuose abstrakcijos lygmenyse: komponente, struktūroje, ir sakinyje.

Buvo stengiamasi suprasti visą sistemos funkcionalumą prieš einant prie sekančių apgražos inžinerijos žingsnių. Buvo gilintasi į sistemos veikimo kontekstą, nustatyti sistemos komponentai. Detaliai aprašomi apibrėžiant funkcijas kurias atlieka.

Analizuojant kiekvieno komponento vidų buvo identifikuojami atskiri kodo skyriai:

1. Skyriai, kur duomenys paruošiami veiksmams atlikti;
2. Duomenų apdorojimo skyriai;
3. Rezultatų pateikimo skyriai ar duomenų eksportas.

Kaip pavyzdys, žemiau (*5.2 pav.)* yra pateikiamas įnašų registravimo kodo fragmentas. Kuris buvo suskirstytas į atskirus skyrius.

```
PROCEDURE INASREG
PUSH KEY
SELECT 0
USE inasai ALIAS INASAI ORDER but_men
...
DO      WHILE      BUTAS+METAI+MENUO=M.BUTAS+M.MET+M.MEN      .AND.
DATA<=M.TDATA .AND. .NOT. EOF()
M.INESTA = M.INESTA+KOM_M
M.I_DEL = M.I_DEL+DELSP
M.I_UZ_EL = M.I_UZ_EL+EL_M
M.I_UZ_TEL = M.I_UZ_TEL+TEL_M
M.D_SK = IIF(DATA-M.DATAA>0, DATA-M.DATAA, 0)
M.SKIRT = IIF(M.SKOLA1>0, M.SKOLA1, 0)
M.K_DELSP = M.K_DELSP+M.SKIRT*M.D_SK*P_PROC/100
M.SKOLA1 = M.SKOLA1-KOM_M
M.DATAA = DATA
SKIP
ENDDO
...
ACTIVATE WINDOW LDARB1
CLEAR
L1E = 2
L1ST = 1
@ L1E, L1ST SAY ------------------------------------------------'
@ ROW()+1, L1ST SAY 'butas³              |                     |'
@ ROW()+1, L1ST SAY 'Komunaliniai        |    món. ³           |'
@ ROW()+1, L1ST SAY 'Delspinigiai        |                     |'
@ ROW()+1, L1ST SAY 'Viso:               |                     |'
@ ROW()+1, L1ST SAY '-------------------------------------------'
STORE 0 TO KOM_M, M.DELSP
@ L1E-1, 5 SAY 'Data:' GET M.DATA
READ
IF LASTKEY()=27
RELEASE WINDOW LDARB1
EXIT
```

Duomenų paruošimo skyrius

Duomenų apdorojimo, atliekamų skaičiavimų skyrius

Rezultatų pateikimo, atvaizdavimo skyrius

5.2 pav. *Programos kodo suskirstymas*

Atliekant sistemos kodo analizę buvo bandoma nustatyti sistemos veiklos taisykles. Veiklos taisyklės išgavimui buvo atliekami sekantys žingsniai [30]:

1. Ieškoma sintaksiškai mirusi logika – tai kodo segmentas, kuris niekada nėra vykdomas, nepaisant duomenų reikšmių.
2. Ieškoma semantiškai mirusi logika – kodo segmentas, kuris niekada nevykdomas dėl nustatytų reikšmių.
3. Įvedimo/ išvedimo logika - kodo dalys, susijusios su tiesioginiu duomenų skaitymu, rašymu, iškvietimu.
4. Ataskaitų išvedimo logika - randamos vietos, kuriose duomenys išvedami į ekraną ar ataskaitų formas.
5. Klaidų apdorojimo logika - randamos privalomai apdorojamos klaidos, iššaukiančios išimčių pranešimus ar darbo nutraukimą.
6. Nereikalinga logika - randamos besikartojančios(perteklinės) sąlygos.

Ištrauktos ir sutvarkytos verslo taisyklės buvo dokumentuotos. Didelėms sistemoms, apgražos inžinerija yra apskritai vykdoma, naudojant pusiau automatizuotus metodus. Naudojami CASE įrankiai, kad „gramatiškai nagrinėtų“ paveldėto kodo semantiką.

5.1.2. Duomenų suvokimas

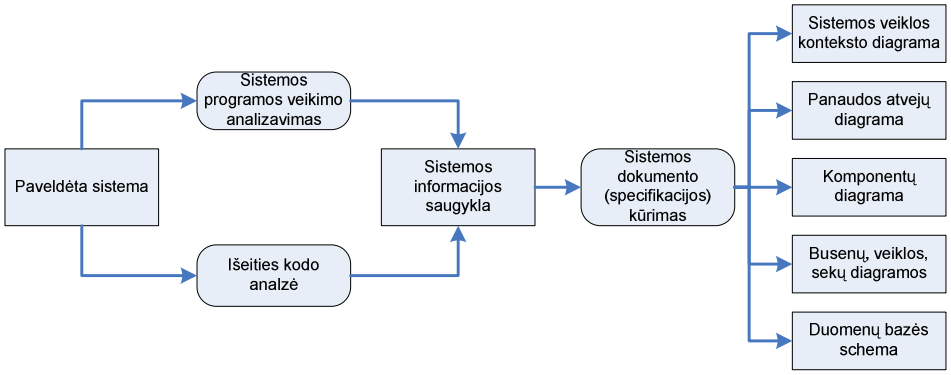
Norint pritaikyti naujas duomenų bazės valdymo sistemas pirmiausia reikia suprasti paveldėtus duomenis. Buvo nustatyta failinė duomenų struktūra. Paveldėti duomenys buvo saugomi DBF failų formate. DBF duomenų failas yra centralizuota dBase duomenų bazės lentelė. Toks failas yra sudarytas iš dviejų dalių. Tai yra dvejetainių (*angl. binary)* ir ASCII duomenų mišinys. Antraštę (*angl. Header*) sudaro dvejetainiai duomenys, o įrašai saugomi ASCII kodavimu. Kiekvienas duomenų objektas buvo saugojamas atskirame faile. Tokia duomenų bazę, galima sakyti, sudaro „laisvosios“ lentelės. Jos siejamos tam tikru duomenų identifikatoriumi, bet jokių raktų neturi.

Failinė duomenų struktūra buvo konvertuota į MySQL duomenų bazę pasinaudojant automatizuotus įrankius. Buvo nustatyta vidinė duomenų struktūra, duomenų atributai, tipai.

5.1.3. Vartotojo sąsajos supratimas

Vartotojo sąsajos ir elgesio supratimui, buvo analizuojam paveldėta sistema manipuliuojant jos vartotojo sąsaja. Buvo siekiama sukurti sistemos elgesio modelį, perprasti navigaciją, „karštuosius“ (*angl. hotkeys*) klavišus. Siekiant sukurti elgesio modelį buvo reikalinga papildoma informacija, kuri buvo ištraukta iš kodo.

Apibendrinta apgrąžos inžinerijos veikla, siekiant gauti sistemos dokumentą gali būti atvaizduojama sekančiu paveikslėliu:



5.3 pav. *Apibendrinta apgražos inžinerijos veikla*

Buvo sukurtos sistemos struktūros, panaudos atvejų, būsenų, veiklos, sekų diagramos, identifikuoti duomenų objektai bei ryšiai tarp jų.

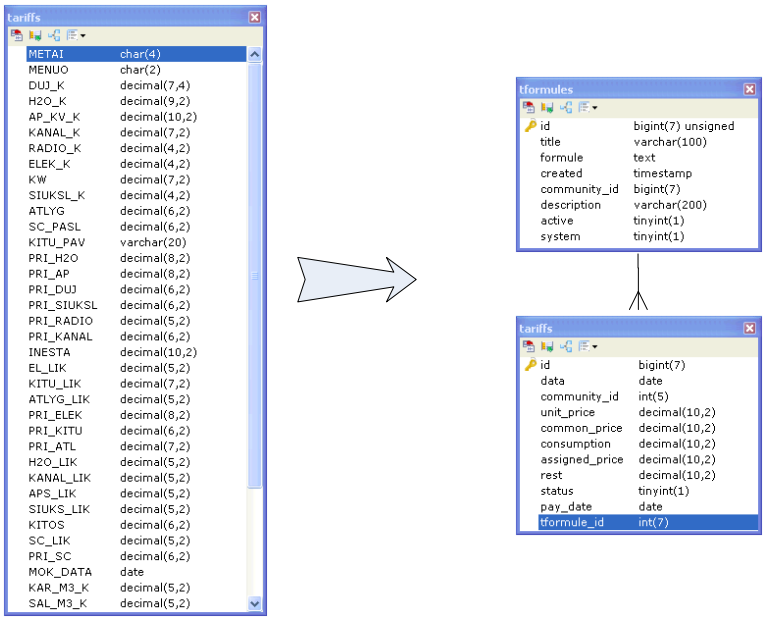
5.2. Duomenų rekonstrukcija ir migracija

Pasinaudojant vizualiu „*SQLYog Schema Designer*“ įrankiu buvo atlikta duomenų restruktūrizavimas. Rekonstruojant esamą duomenų bazės schemą, buvo identifikuojamos egzistuojančios duomenų esybės, duomenų objektai, bei ryšiai tarp jų.

Buvo atliekamas paveldėtos duomenų bazės normalizacija. Normalizuota duomenų bazė užima mažiau vietos ir veikia žymiai greičiau. Pagrindinis tikslas - duomenų dubliavimo sumažinimas ir anomalijų pašalinimas. Atlikus duomenų bazės normalizaciją, žymiai lengviau atnaujinti duomenis esančius lentelėse, mažai tikėtina, jog prisivels klaidų.

Žemiau, 5.4 paveikslėlyje pavaizduotas paveldėtos tarifų lentelės normalizavimas. Remiantis duomenų bazių normalinėmis formomis, buvo atlikta:

- Pašalinti dubliuoti laukai, kiekvienas įrašas turi vienodą laukų skaičių
- Visi laukai nepriklausomi
- Lentelėms sukuriami pirminiai raktai
- Pašalintas duomenų dubliavimas stulpeliuose
- Sukurti ryšiai tarp lentelių
- Lentelės susiejamos išoriniais raktais



5.4 pav. *Duomenų lentelės normalizavimas ir pertvarkymas*

Atlikus duomenų bazės optimizavimo veiksmus buvo sukurta posistemė, kurios paskirtis automatizuotai atlikti duomenų importą. Duomenų importas buvo atliekamas pagal tokias taisykles:

1. Pirmiausia importuojamos lentelės kurios yra sudarytos iš vieno duomenų objekto.
2. Lentelėms sukuriami raktai. Kiekvienam įrašui sukuriamas unikalus id, kuris vėliau naudojamas lentelių tarpusavio ryšiui.
3. Toliau importuojamos lentelės sudarytos iš sudėtinių duomenų, bei susiejamos raktais atsižvelgiant į atitinkamą duomenų identifikatorių.

Nors procesas buvo automatizuotas, buvo išvengta klaidų, duomenų praradimo bei dubliavimo. Priešingai, buvo išspręstos duomenų kodavimo problemos iškilusios keičiant sistemos aplinką. Nors duomenų bazės schema buvo pertvarkoma, buvo siekiama išlaikyti nepakitusių duomenų tipus tai palengvinant automatizuotą duomenų perkėlimo procesą, bei siekiant nepatirti duomenų tikslumo praradimo.

Iš to galima daryti išvadą, jog atliekant sistemos rekonstrukciją ir siekiant pagerinti sistemos charakteristikas atliktas duomenų restruktūrizavimas efektyvus. Tai ne tik sumažina duomenų apimtį, bet ir pagerina sistemos operacijų našumą atliekant veiksmus su dideliais duomenų kiekiais. Svarbu paminėti, kad atlikus pakeitimus duomenų bazėje, tai įtakoja ir architektūros pakeitimus.

5.3. Tiesioginė inžinerija

Šio proceso metu pagal sukurta specifikacijos dokumentą (žr. 4 skyrių) realizuojami reikalaujami sistemos pakeitimai, bei įtraukiamos naujos funkcijos. Sistema yra perkuriama

su nauja technologija bei nauja architektūra. Realizuojant rekonstruotą sistemą buvo remiamasi pagrindiniais tiesioginės rekonstrukcijos tikslais, kurie yra:

- Keisti sistemos architektūra, kad sistemos palaikymas ir priežiūra būtų palengvinta.

Tam pasirinkta struktūrizuota 3 lygių architektūra. Sistema, kuri paremta tokia architektūra turi vaizdinius, kurie atsakingi tik už puslapių atvaizdavimą. Šie puslapiai vykdo, verslo logikos (*BLL*) sluoksnyje, esančią programos veiksmų logiką. Šis sluoksnis, bendrauja su duomenų sluoksniu (*DAL*), kuris vykdo reikiamas operacijas su duomenimis. Visa informacija iš duomenų sluoksnio gauta informacija ir apdorota verslo logikos yra grąžinama į atvaizdavimo sluoksnį (*UI*).

Sistemoje pritaikytas MVC projektavimo šablonas (žr. 4.6.1). MVC yra populiarus tarp internetu paremtų programinės įrangos sistemų.

- Vystomos sistemos kokybė yra aukštesnė nei vidutinė bei aukštesnė nei buvusios sistemos.

Sistemoje panaudotos atviro kodo technologijos kaip PHP bei MySQL. PHP prieš kitas WEB paremtas technologijas pasižymi greičiu, daugiaplatformiškumu, įvairių DBVS palaikymu. MySQL pasižymi pranašumais mažoms ir vidutinėms duomenų bazių sistemoms. Naujoji sistema realizuota objektiškai orientuotoje aplinkoje. HTML atvaizdavimo bei CSS stilių kalba realizuota palaikant W3C žymėjimo standartus.

- Naujos sistemos pakeitimai bei naujos funkcijos lengvai išmokstamos ir perprantamos.

Rekonstruojant sistemą buvo išlaikytas esamas sistemos funkcionalumas, kuris buvo praplėstas, bei pridėtos naujos funkcijos. Kadangi sistemos vartotojo sąsaja yra paremta naršykle, vartotojams nėra sunku perprasti jos veikimą. Grafinėje sąsajoje naudojami standartiniai elementai (meniu, mygtukai, piktogramos, ir kt.)

- Prevencinis sistemos palaikymas.

Realizuota galimybė keisti sistemos konfigūraciją. Atsižvelgta apskaitos sistemos dažniausia pasitaikančios problemos, kaip mokesčių skaičiavimo taisyklių pakitimai, kiti apribojimai (pvz. PVM ir kt.). Tam realizuota mokesčių apskaičiavimo formulių bei įvairių koeficientų keitimas ir valdymas. Pertvarkius duomenų bazės struktūrą realizuota galimybė įtraukti naujus skaitiklius, naujus mokesčius.

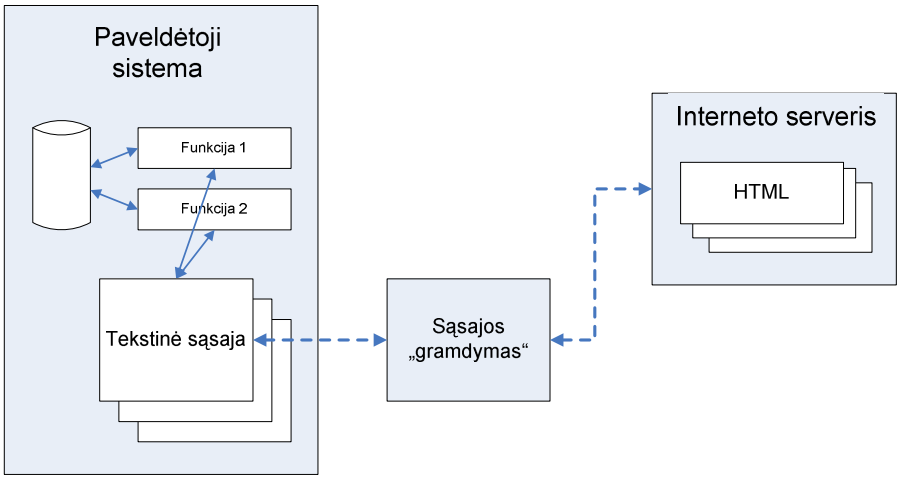
5.3.1. Tiesioginė vartotojo sąsajos inžinerija

Sistemos vartojo sąsaja yra matomiausia sistemos dalis. Todėl jai buvo skiriamas nemažas dėmesys. Modernizavus vartotojo sąsają yra pagerinamas sistemos panaudojamumas, be to atnaujinta sąsaja tai labai vertinama sistemos vartotojų.

Atsižvelgiant į pasirinktas technologijas bei sistemos architektūrą, sistemos vartotojo sąsaja yra paremta naršykle. Tokios sąsajos pranašumai:

- Grafinė vartotojo sąsaja
- Nuotolinis priėjimas prie sistemos
- Vartotojai gali prisijungti prie sistemos iš kitų, geografiškai nutolusių, kompiuterių.
- Nepriklauso nuo platformos.
- Mažesni apmokymo kaštai, nes vartotojas susipažinęs su žiniatinklio galimybėmis.
- Lengviau pasiekti sąsajos nuoseklumą.

Paveldėtos sistemos dažniausiai turi tekstu paremtas sąsajas. Pakeitus jas grafine vartojo sąsajas, kuri yra sukurta hiperteksto (HTML) kalba, vartotojas gali naudotis sistema interneto naršyklės pagalba. Šiam procesui yra naudojami automatizuoti įrankiai (*angl. scraping tools*). Tokie įrankis iš paveldėtos tekstinės sąsajos sugeneruoja naują sąsają, pavyzdžiui HTML formatu. Butų savininkų pavedėtos sistemos modernizavimui buvo naudojama „Screen Scraping Library" įrankis.



5.5 pav. *Vartotojo sąsajos modernizavimas*

Vartotojo požiūriu tokia vartotojo sąsają yra modernizuota, kadangi pateikiama šiuolaikine technologija. Svarbu paminėti, kad atnaujinta sistema neturėtų būti panaši į paveldėtą.

6. REKONSTRUOTOS SISTEMOS KOKYBĖS VERTINIMAS

6.1. Sistemos kokybės vertinimas

Atlikus sistemos rekonstrukcija buvo atliktas butų savininkų bendrijų informacinės sistemos kokybės vertinimas pagal ISO 9126 standartą. ISO 9126 - tarptautinis PĮ kokybės vertinimo standartas. Pagrindinis šio standarto tikslas yra apibrėžti gerai žinomus žmonių daromus nukrypimus, kurie gali nepalankiai įtakoti PĮ kūrimo projekto teikimą ir supratimą. Buvo atlikta pirmoji ISO 9126 standarto dalis, kurioje yra apibrėžtas struktūrizuotas PĮ kokybės charakteristikų ir subcharakteristikų rinkinys.

Žemiau pateiktoje lentelėje pavaizduoti kokybės kriterijų vertinimai. Sistemos kokybę vertino užsakovas, jos eksploatacijos metu, remiantis sistemos užsakovo iškeltais reikalavimais, stebint sistemos veikimą ir vartotojų darbą su sistema. Sistema buvo vertinama dešimtbalėje vertinimo sistemoje, kur 10- reiškia, jog užsakovas priekaištų neturi, 1-visiškai netinkama.

6.1 Lentelė. *Kokybės charakteristikų vertinimo rezultatai*

Kokybės charakteristika	Vertinimas	Charakteristikos paaiškinimas
Funkcionalumas: bendras įvertinimas 8,4		
Tinkamumas	9	Programos funkcijų, atliekančių reikiamas užduotis, pilnumas bei atitikimas reikalavimams. Funkcinių bei nefunkcinių reikalavimų atitikimas.
Tikslumas	9	Programos veikimas patiekiant teisingus arba sutartus rezultatus Sistemoje atliekami skaičiavimai tikslūs ir teisingi.
Sąveikumas/Bendradarbiavimas	6	Programos sąveika su kitomis sistemos galimybėmis.
Suderinamumas	9	Programos atitikimas įvairiems standartams: PĮ kūrimo standartams, įstatymams..
Saugumas	9	Uždraustas neautorizuotų vartotojų priėjimas prie programos ir jos duomenų. Šifruoti slaptažodžiai.
Patikimumas: bendras įvertinimas 8		
Brandumas	8	Sistemos klaidų ir sutrikimų dažnumas
Klaidų tolerancija	8	Iškilus sistemos klaidai kurioje nors dalyje, sistema nepakimba. Galima naudotis kitomis sistemos funkcijomis.
Atkuriamumas	8	Nutrūkus sistemos darbui ar įvykus klaidai sistemos duomenys neišgadinami, neprarandami. Atsitikus sistemos sutrikimams duomenys yra nesunkiai atstatomi.
Panaudojamumas: bendras įvertinimas 8,33		
Suprantamumas	9	Aiški vartotojo sąsaja, pagrįsta interneto naršykle.

		Panaudoti standartiniai elementai (meniu, mygtukai, įvedimo laukai, piktogramos).
Perprantamumas	8	Aiški sistemos navigacija, lengvai įsisavinamas veikimas.
Veiklumas	8	Sistemoje gausu informacinių pranešimų: atliktų veiksmų, iškilusių klaidų, perspėjimų bei paaiškinimų.
Našumas: bendras įvertinimas 8		
Operacijų greitis	8	Programos operacijų vykdymas yra greitas.
Resursų naudojimas	8	Standartiškai sistema naudoja mažai resursų. Sistema nereikalauja itin spartaus interneto ryšio.
Palaikomumas: bendras įvertinimas 8		
Analizuojamumas	9	Pastangų apimtis, reikalinga programos trūkumų arba defektų analizei, arba modifikuojamų programos dalių nustatymui.
Keičiamumas	9	Sistemos veikimo bei skaičiavimo operacijos gali būti konfigūruojamos ir keičiamos. Atsižvelgiant į vartotojo poreikius.
Stabilumas	7	Vienos sistemos dalies pakeitimai neiššaukia pakeitimų ar klaidų kitose.
Testuojamumas	7	Automatiniai sistemos testai apima didžiąją dalį sistemos funkcijų.
Pernašomumas: bendras įvertinimas 8,33		
Prisitaikomumas	10	Sistema veikia įvairiose operacinėse sistemose ir aplinkose
Įdiegiamumas	8	Sistemos įdiegimo sudėtingumo lygis ir pasiruošimas
Pakeičiamumas	7	Sistemos komponentai ir funkcijos gali būti panaudoti kituose sprendimuose.

Atlikus subcharakteristikų įvertinimą ir paskaičiavus kiekvienos iš charakteristikų vidurkį, galime paskaičiuoti sistemos bendrą kokybės įvertinimą:

$$BKĮ = \frac{\sum CKĮ}{CK}, \textit{ kur CKĮ – charakteristikos kokybės įvertinimas, CK – charakteristikų kiekis}$$

Atlikus skaičiavimus gauta, jog rekonstruotos sistemos įvertinimas 8,18. Galime teigti, kad sistema patenkina užsakovo reikalavimus.

6.2. Sistemos tobulinimo ir plėtojimo galimybės

Atsižvelgiant, kad sistema ir toliau bus naudojama, galime iškelti tam tikrus vartotojų reikalavimus ir tolesnes prioritetines butų savininkų bendrijų informacinės sistemos tobulinimo galimybes.

Vartotojo sąsajos tobulinimas

Atsižvelgiant į užsakovo atsiliepimus dėl sistemos navigacijos patobulinimo ateityje siekiama vartotojo sąsajoje atvaizduoti sistemos meniu „gylį“. Tai yra dirbant su sistema yra rodoma esama vieta, kelias sistemoje (pvz. įtraukiant nauja sistemos vartotoją rodomas kelias BSBIS > Vartotojai > Naujas vartotojas , kur kiekvienas vaizduojamas meniu lygis yra su nuoroda, kuri veda į ta lygį. BSBIS yra aukščiausias sistemos lygis dar vadinamas „root“). Taip vartotojas mato esamą vietą sistemoje, bei atliekamus veiksmus, žymiai lengviau pereiti į norimą meniu lygį. Toks sistemos kelio vaizdavimas yra labai populiarus ir vertinamas, bei pakelia sistemos panaudojamumo kokybės laipsnį.

Prijungimas prie elektroninės bankininkystės

Šiais laikais kai žmonės siekia taupyti ne tik pinigus, bet ir laiką, elektroninis sąskaitų apmokėjimas yra labai vertinamas. Žvelgiant į ateitį, sistemą susieti su bankininkystės sistemomis yra didelis sistemos patobulinimas. Tokiu būdu būtų galima suformuoti sąskaitas už įvairias paslaugas bei mokesčius ir jas apmokėti neišeinant iš namų - internetu.

Nauji sistemos moduliai

Siekiant praplėsti sistemos galimybės yra pasiūlymas įtraukti daugiau apskaitos modulių. Vienas iš tokių modulių vartotojų biudžeto apskaitą ir planavimas. Toks modulis vartotojams padėtų planuoti savo biudžeto lėšas, sulyginti planuotas išlaidas su esamomis. Didesnis naudingas modulių skaičius praturtina sistemą, bei pritraukia vartotojus savo naudingomis funkcijomis.

7. IŠVADOS

1. Sistemos rekonstrukcija yra atliekama toms paveldėtoms sistemoms, kurios yra svarbios organizacijos verslui ir yra žemos kokybės, kitu atveju renkamasi kitas modernizacijos metodas, sistema ir toliau palaikoma arba išmetama.
2. Sistemos rekonstrukcija yra sudėtingas procesas, su daug kliūčių. Labai svarbi praktinė patirtis, kad padėtų organizacijai nepadaryti tų pačių klaidų.
3. Magistrinio darbo metu buvo pasirinkti trys rekonstrukcijos metodai: atvirkštinė rekonstrukcija, duomenų rekonstrukcija ir tiesioginė rekonstrukcija.
4. Apgrąžos rekonstrukcijos metu buvo sukurta sistemos dokumentacija, kuri yra labai svarbi jos vystyme ir reikalinga tolimesniame sistemos palaikyme ar tobulinime.
5. Nebuvo rasta įrankių galinčių automatizuotai analizuoti paveldėtą FoxPro išeities kodą ir gauti abstrakčią specifikaciją.
6. Remiantis apgrąžos ir tiesioginės rekonstrukcijos principais suformuluoti projektiniai, techniniai probleminių uždavinių sprendimai, bei rekonstruota butų savininkų bendrijų informacinė sistema. Jos architektūra suskirstyta į tris pagrindinius lygius: atvaizdavimo, valdymo ir priėjimo prie duomenų lygius. Sistemos elementai išdėstyti pasinaudojus „Modelis-Vaizdas-Valdiklis“ projektavimo šablonu.
7. Atlikus rekonstruotos sistemos kokybės įvertinimą nustatyta, kad sistema atitinka specifikacija ir užsakovo keliamus reikalavimus.
8. Atliekant sistemos palaikymą ir priežiūrą labai svarbu, jog būti laikomasi bendrų programų inžinerijos principų, kitaip sistemą vėl taps paveldėtąją.

8. LITERATŪRA

[1] Brodie M., Stonebraker M., „*Migrating Legacy Systems: Gateways, Interfaces and the Incremental Approach*“, Morgan Kaufmann Publishers, Inc. USA, 1995

[2] Bakehouse G.& Wakefield T., „*Legacy information systems*“, 2005. [Žiurėta 2009-04-15]. Prieiga per internetą: http://www.accaglobal.com/students/publications/student_accountant/archive/2005/54/2377837

[3] Bennett K., „*Legacy Systems: Coping with Success*“, IEEE Software, Jan 1995, p.19-23

[4] Sommerville I., „*Software Evolution and Reengineering*“, Software Engineering, 7th edition. Chapter 21, 2004

[5] Seacord RC, Plakosh D, Lewis GA. „*Modernizing Legacy Systems: Software Technologies, Engineering Processes, and Business Practices*“, Addison-Wesley, 2003.

[6] Ransom J., Sommerville I., and Warren I., 1998. „*A Method for Assessing Legacy Systems for Evolution*“, Proceedings of the Conference on Software Maintenance and Reengineering. Florence, Italy (8–11 March 1998), p. 128–134.

[7] Bisbal, J. Lawless, D. Bing Wu Grimson, J. Trinity Coll., Dublin, *Legacy information systems: issues and directions*, : Software, Volume: 16, p. 103-111, 1999

[8] Reengineering Center, Software Engineering Institute, *Perspectives on Legacy System Reengineering* , Carnegie Mellon University, 1995.

[9] The Standish Group, „*CHAOS Summary 2009 report*“. [Žiūrėta 2009-04-30]. Prieiga per internetą: http://www.standishgroup.com/newsroom/chaos_2009.php.

[10] Weiderman, Nelson H., John K. Bergey, Dennis B. Smith, and Scott R. Tilley. , *"Approaches to Legacy System Evolution"*, Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 1997

[11] Comella-Dorda S., Wallnau K., Seacord RC, Robert J. „*A Survey of Legacy System Modernization Approaches*“, Report CMU/SEI-2000-TN-003, April 2000, Software Engineering Institute, Carnegie Mellon University, Pittsburgh, PA 15213.

[12] Winsberg P., „*Legacy Code: Don’t Bag it, Wrap it*“, Datamation, 41(9), May 1995, p. 36-41

[13] Bisbal J., Lawless D., Wu B., Grimson J., „*Legacy Information System Migration: A Brief Review of Problems, Solutions and Research Issues*“, IEEE Software, 1999

[14] Sneed H.M. and Majnar R., „*A case study in software wrapping*," International Conference on Software Maintenance, Bethesda, MD, Nov1998, p. 86–93

[15] Kantogiannis K., „*Re-engineering of Legacy Systems*", Software Reenginiering – Research & Practice. [Žiurėta 2009-04-11]. Prieiga per internetą: http://www.telecom.ntua.gr/ees/downloads/pse/

[16] Byrne E. J., *"A conceptual foundation for software reengineering,"* Conference on Software Maintenance, Orlando, Florida, Nov 1992, p. 226–235.

[17] Meena Jha, Maheshwari P., „*Reusing Code for Modernization of Legacy Systems"*, Proceedings of the 13th IEEE International Workshop on Software Technology and Engineering Practice (STEP'05), 2005

[18] Philippus E., „*Working Model for the Modernization of Legacy Systems"*, ImprovemenT BV, 2008

[19] ESPRIT Project - Lancaster University, „*RENAISSANCE Project - Methods & Tools for the evolution and reengineering of legacy systems*". [Žiurėta 2009-04-25]. Prieiga per internetą: http://www.comp.lancs.ac.uk/computing/research/cseg/projects/renaissance/deldesc.html

[20] CHIANG C., „*Software Stability in Software Reengineering"*, IEEE 2007, p. 719-722

[21] Software Engineering Institute, „*Reengineering: The Horseshoe Model*. [Žiūrėta 2009-04-05]. Prieiga per internetą: http://www.sei.cmu.edu/reengineering/horseshoe_model.html

[22] Ping L., and Yang X., "*An evolutional model of legacy system reengineering*," 9th Joint International Computer Conference, Zhuihai, China, Nov 2003, p.160-164.

[23] Boucetta S., Hadjami H., Ghezala B., Kamoun F., „*Architectural Recovery and Evolution of Large Legacy Systems"*, Proceedings of the International Workshop on the Principles of Software, 1999

[24] Weiderman N., Northrop L., Smith D., Tilley S., Wallnau K., „*Implications of Distributed Object Technology for Reengineering"*, Technical Report CMU/SEI-97-TR-005, Carnegie Mellon University, June 1997

[25] Pressman, Roger S., „*Software engineering: a practitioner's approach"* / Roger S. Pressman.—5th ed., McGraw-Hill series in computer science, 2001, p. 805-816

[26] Baxter Ira D. and Mehlich M., „*Reverse engineering is reverse forward engineering"*, Science of Computer Programming Volume 36, Issues 2-3, March 2000, p. 131-147.

[27] Deursen A., Klint P., and Verhoef C., „*Research Issues in the Renovation of Legacy Systems*“, Proceedings of the Second Internationsl Conference on Fundamental Approaches to Software Engineering, 1999, p. 1-21.

[28] Fong J., „*Information Systems Reengineering and Integration*“, Springer Science, Second edition 2006

[29] *CakePHP web application framework model: MVC Model*. [Žiurėta 2009-04-15]. Prieiga per internetą: http://www.bhartisoftland.com/technologies-skill-sets/cake-php-developments.html

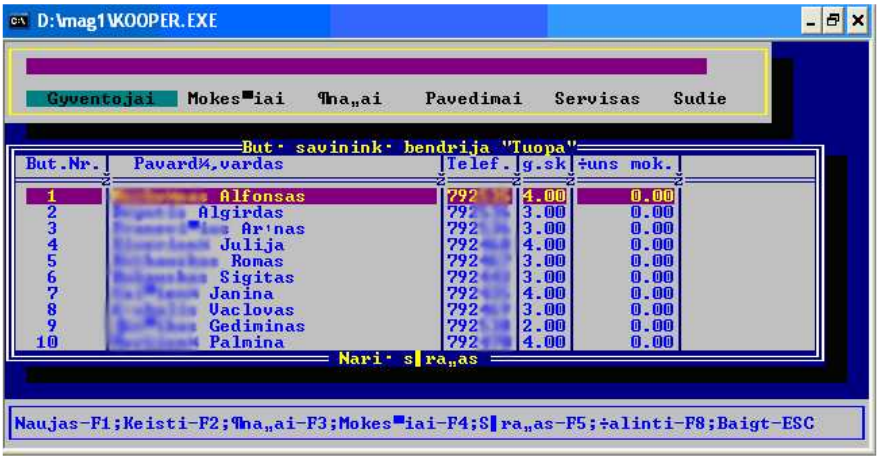
[30] Rataitė B., „*Verslo taisyklių išgavimo iš esamų sistemų būdai*“, 10-osios Lietuvos jaunųjų mokslininkų konferencijos „Mokslas – Lietuvos ateitis“ medžiaga, , Vilnius 2007, p. 270-276

[31] Kan S. H.,“*Metrics and Models in Software Quality Engineering*“, 2nd Edition, 2002 by Addison-Wesley Professional. [Žiurėta 2009-04-15]. Prieiga per internetą: http://www.informit.com/articles/article.aspx?p=30306

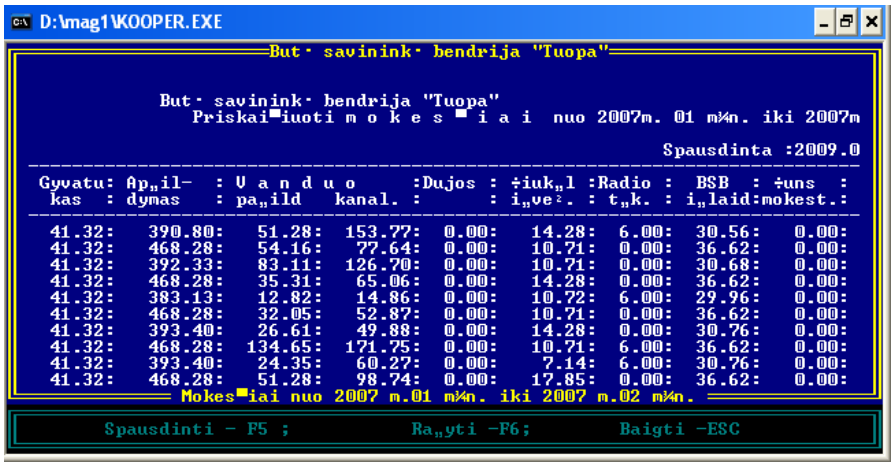
[32] Попов А. А., „Fox Pro 2.5/2.6 в DOS и WINDOWS“, [Knyga], ДЕСС КОМ, 1997, ISBN 5-93650-004-7

9. PRIEDAI

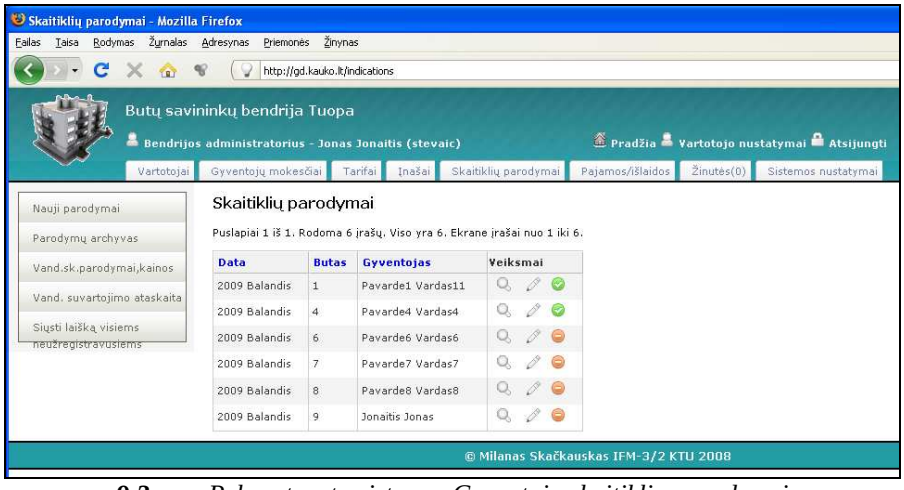
1 priedas. Paveldėtos ir rekonstruotos sistemos vartotojo sąsajos pavyzdžiai



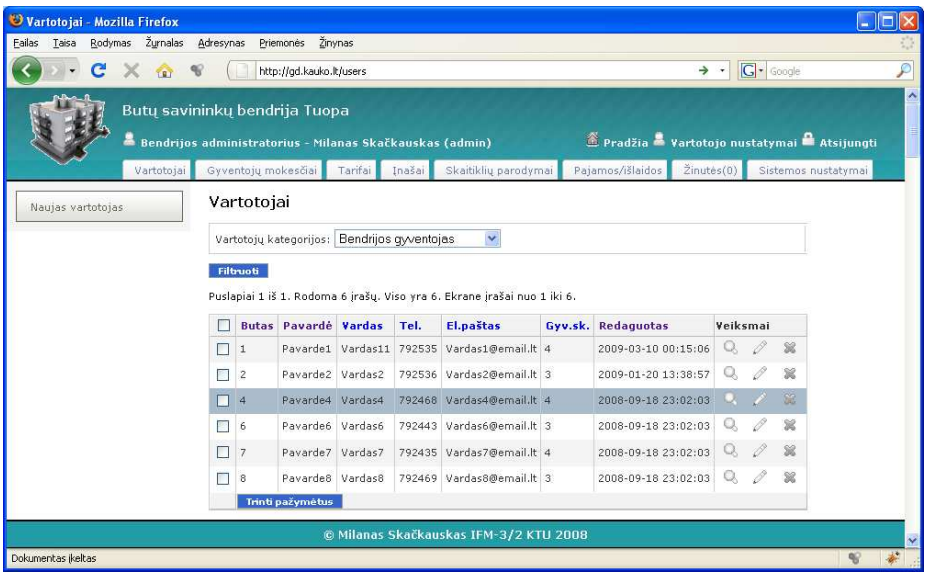
9.1 pav. *Paveldėtos sistemos gyventojų informacijos langas*



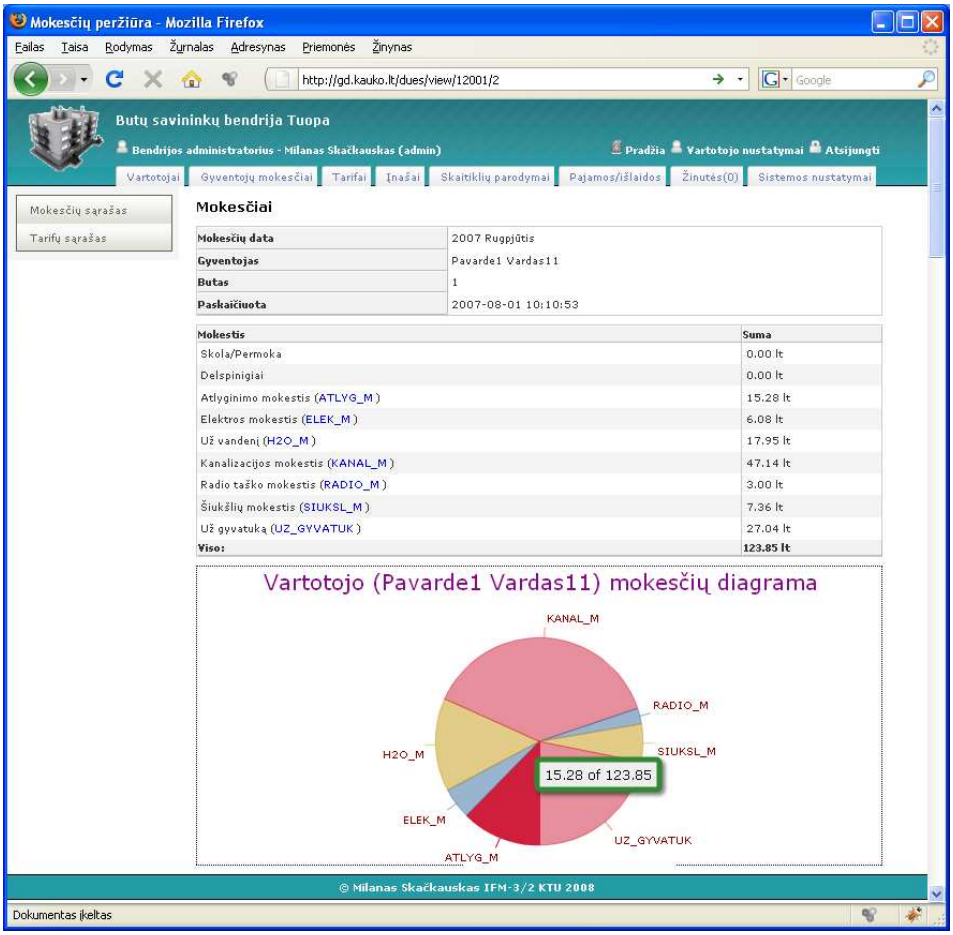
9.2 pav. *Paveldėtos sistemos gyventojų mokesčių ataskaita*



9.3 pav. *Rekonstruota sistema: Gyventojų skaitiklių parodymai*
(Raudoni žymi neužregistruotus, žali užregistruotus parodymus)



9.4 pav. *Rekonstruota sistema: Gyventojų valdymo sąsaja*



9.5 pav. *Rekonstruota sistema: Gyventojų mokesčių pateikimas*