KAUNO TECHNOLOGIJOS UNIVERSITETAS
INFORMATIKOS FAKULTETAS
PROGRAMŲ INŽINERIJOS KATEDRA

Indrė Paulavičiūtė

ATVIRO KODO PRODUKTŲ KOKYBĖS UŽTIKRINIMO METODŲ TYRIMAS

Magistro darbas

Vadovas
doc. dr. Eimutis Karčiauskas

KAUNAS, 2006

KAUNO TECHNOLOGIJOS UNIVERSITETAS
INFORMATIKOS FAKULTETAS
PROGRAMŲ INŽINERIJOS KATEDRA

TVIRTINU
Katedros vedėjas
dr. Eduardas Bareiša
2006-05-29.

ATVIRO KODO PRODUKTŲ KOKYBĖS UŽTIKRINIMO METODŲ TYRIMAS

Informatikos inžinerijo`s mokslo magistro baigiamasis darbas

Kalbos konsultantė
Lietuvių k. kat. doc.
dr. J. Mikelionienė
2005-05-25

Recenzentas
L. Nemuraitė
2005-05-29

Vadovas
doc. dr. E. Karčiauskas
2005-05-25

Atliko
IFM 0/2 gr. stud.
I. Paulavičiūtė
2005-05-29

KAUNAS, 2006

RESEARCH IN QUALITY ASSURANCE METHODS FOR OPEN SOURCE PRODUCTS

SUMMARY

In this study, the criteria for quality assessment and problems of open source (OS) product quality assurance, based on these criteria, are discussed.

Broad use of open source products both in personal and business environment prove sufficient quality, despite of absence of traditional quality assurance methods for these products. There is no mature model for OS product quality evaluation - only various experimental models are applied (for example, OS project statistics criteria evaluation). OS product quality is also managed in unconventional methods, related to OS specific software development procedures (for example, volunteer control).

A generalized method for OS product quality evaluation, derived from a combination of various metrics, is proposed.

TURINYS

PAVEIKSLŲ SĄRAŠAS

LENTELIŲ SĄRAŠAS

1. Įvadas

Atviras kodas (AK) apibūdina gamybos ir vystymo technologiją, skatinančią galutinio produkto informacijos pasiekiamumą. Atviras kodas gali būti laikomas ir filosofija, ir pragmatiška metodologija. Atviro kodo programinė įranga (AKPĮ) tapo populiariausia AK taikymo sritimi.

AK nemokamą programinę įrangą, kartu su jos išeities kodais, galima laisvai naudoti, kopijuoti, platinti ir modifikuoti pagal poreikius. Atviro kodo stiprybė – galimybė sujungti ir gerinti plačios programuotojų bendruomenės sugebėjimus bei žinias.

Didelė atviro kodo kompanijų dalis naudoja atviro kodo įrankius, kurie skirti ne tik galutiniam vartotojui, bet kasdieniniam programuotojų darbui, t. y. įvairius transliatorius (*angl*. „compilers“), programų derintuvus (*ang*l. „debugers“), redaktorius (*angl*. „editors“). AKPĮ projektų sėkmė gali būti paaiškinama sparčiu kūrimu/tobulinimu, patikimumu, portatyvumu ir gaunamos programinės įrangos išplečiamumu. Šių savybių priežastys – programų kūrėjai yra ir jų vartotojai, išeities tekstai prieinami plačiajai visuomenei, programų kūrėjai yra PĮ kūrėjų bendruomenių grupių nariai.

Diskusija apie atviro kodo produktų kokybę tapo aktualia nuo pat šio judėjimo klestėjimo pradžios (aštuntasis dešimtmetis) [12]. Imta gilintis ir domėtis klausimais: kaip įmanoma kontroliuoti individualių savanorių ir galutinio produkto kokybę, kaip/ar savanoriai sugeba kurti kokybiškai gerus produktus, kaip kodas apsaugomas ir integruojamas, kaip priimami reikalingi valdymo sprendimai, kaip savanoriai suderina savo plačius interesus ir individualius tikslus?

Rinkoje atviro kodo programinė įranga konkuruoja dėl vartotojų dėmesio su firminės mokamos programinės įrangos produktais. Net jei AK produktas ekvivalenčios kokybės, be to – nemokamas (nereikia pirkti licenzijų, AK įrangą visada galima išbandyti prieš pritaikant ją savo darbams ar modifikuoti pritaikant specifiniams poreikiams), jis vis tiek turi kovoti dėl „kritinės rinkos dalies“. AKPĮ labai naudinga ne tik individualiam vartotojui. AK plitimas teikia privalumų valstybės ekonomikai.

Atviro kodo aplinkoje negalioja komercinės programinės įrangos kūrimo principai. Tradiciškai buvo laikomasi nuomonės, kad gerą programų kokybę gali užtikrinti tik hierarchinė organizacija, tačiau atviro kodo produktų jau užimta didelė rinkos dalis rodo, kad ši aplinka taip pat potencialiai tinkama kokybiškiems produktams kurti. Formalizuotą kokybišką kūrimo procesą ir centralizuotus valdymo sprendimus (formalūs vartotojo reikalavimai ir projektavimo specifikacijos), kuriais tradiciškai valdoma projekto kokybė, pakeičia kitos produkto kokybiškumą

skatinančios priemonės. Kokybei teigiamą įtaką daro atviro kodo ideologija bei kūrimo specifika – naudojama modulinė programinės įrangos (PĮ) struktūra, adaptyvus sistemos modelis ir dažni atnaujinimai, PĮ mutacija, aktyvus AK bendruomenės indėlis (resursų ir atsakomosios reakcijos atžvilgiu). Svarbus ne komercinės PĮ kokybę įtakojantis aplinkos aspektas – AKPĮ charakterizuojanti personalizacija, misijos ir opozicijos jausmas. Dėl griežtų formalių taisyklių nebuvimo tradiciniai kokybės užtikrinimo, valdymo ir įvertinimo metodai atviro kodo projektams paprastai negali būti taikomi, o pats kokybės užtikrinimo procesas yra probleminis. Neegzistuoja bendras brandaus AK kokybės vertinimo modelis ar bendri kokybės valdymo metodai – bandoma kurti ir taikyti eksperimentines kokybės užtikrinimo strategijas bei produktų kokybės įvertinimo modelius.

Nors atviro kodo aplinkoje neįmanoma taikyti tradicinių komercinės PĮ kokybės užtikrinimo metodų, sėkmingas AK produktų konkuravimas su komercine įranga įrodo, kad atviro kodo produktų kokybė tenkina jų naudotojus.

Šio darbo tikslas yra ištirti pakankamos atviro kodo produktų kokybės priežastis, kokybės gerinimo procesą, kuriamus eksperimentinius kokybės vertinimo modelius ir atviro kodo PĮ kokybės užtikrinimo strategijas bei metodus (ir didelės apimties, ir mažos bei vidutinės apimties AK projektuose). Tiriamas eksperimentinių kokybės modelių tinkamumas ir patikimumas AK produktų kokybės vertinimui. Darbe pateiktos problemos, kurios egzistuoja bandant įvertinti atviro kodo produktų kokybę ir valdyti atviro kodo produktų kokybės užtikinimą.

Darbe pasiūtas neformalus eksperimentinis integruotas kokybės vertinimo modelis taikomas magistrinio darbo projekto realizavimui panaudotiems AK produktams. Modelis tiriamas vertinant naudojamų kokybės vertinimo kriterijų teisingumą, korektišką prognozavimą, pritaikomumą AKPĮ projektams ir paprastumą.

2. Analitinė dalis

Atviro kodo produktų ir komercinės programinės įrangos kūrimo procesai smarkiai skiriasi. Įprastinės, gerai ištirtos, komercinės programinės įrangos vertinimo metodikos bei kokybės užtikrinimo strategijos iš principo negali būti taikomos AK aplinkoje. Tačiau AK produktų populiarumas privačiame ir organizaciniame sektoriuose rodo vartotojus tenkinančią AK produktų kokybę. Daug tyrėjų analizuoja veiksnius, darančius įtaką AK kokybei. Siekiant valdyti AK produktų kokybę, jiems bandoma pritaikyti eksperimentus kokybės vertinimo kriterijus. Kuriami individualūs, konkretiems produktams taikytini kokybės valdymo metodai [10].

2.1. Bendrosios atviro kodo idėjos

Kuriant atviro kodo programinę įrangą, pirminę programinės įrangos versiją sukuria programuotojas arba įmonė ir pateikia jos išeities tekstą – naudoti ir modifikuoti – plačiajai visuomenei (paprastai – programų kūrėjų bendruomenei).

Pateikus išeities tekstus, iš atitinkamų kontrolierių programuotojai įsigyja atviro kodo licenciją, suteikiančią vartotojui keturias teises (turėti programos išeities tekstų kopiją ir ją transliuoti, modifikuoti bei toliau platinti). Tradicinė programinės įrangos licenzija įpareigoja vartotoją mokėti už tai, kad naudoja programos kodą. Įprasta, bet ne visada, kad atviro kodo licencija įpareigoja programuotoją, keitusi pradinį atviro kodo programos tekstą, nusiųsti jį atgal programos autoriui, kad potencialiai būtų galima prijungti pakeitimus prie oficialios produkto versijos. Toks procesas kartojamas neribotą laiką – kol sumažėja domėjimasis programa arba firma nusprendžia programos tekstus įslaptinti, o galutinį programos variantą pardavinėti (nors tai atsitinka itin retai)

Atviras kodas – monopolių programinės įrangos rinkoje pabaiga. Tą patį darbą dirbančios įmonės ne švaisto resursų konkurencijai, o jungia savo jėgas. Išsiplėtus internetui tarptautinė kooperacija tapo kasdienybe [1].

Nikolai Bezroukov sukūrė alternatyvią atviro kodo teoriją, vadinamą „atviras kodas kaip mokslinis tyrimas“ (*angl.* “Open Source as academic research”). Ši teorija skelbia, kad internetas smarkiai sumažina įvairios programinės įrangos teikimo (pvz., operacinių sistemų, kompiliatorių, paslaugų programų) kainą. Galima sukurti begalę praktiškai nieko nekainuojančių tos

pačios programos kopijų. Tai didžiausias skirtumas tarp AKPĮ ir visų kitų plataus vartojimo prekių. Šis skirtumas ignoruojamas tradiciniame PĮ platinimo modelyje. Pagal N. Bezroukov, PĮ kūrimas panašus į taikomos teorijos kūrimą. Jis programos kūrimą klasifikuoja prie mokslinės veiklos. Antras didelis skirtumas tarp plataus vartojimo prekių ir PĮ, tai lengvai įgyvendinamos modifikacijos (dar viena analogija su mokslu). Ir moksle, ir programavime atviro kodo aplinkoje, žmonės įprastai dirba ne dėl finansinių priežasčių, o siekdami įgyvendinti savo svajonę. Interneto pagalba gali būti lengvai kuriamos virtualios mokslinės bendruomenės spręsti klausimams ar problemoms. Virtualios kūrėjų komandos, panašios į istorinę mokslininkų bendruomenę, struktūra ir atsakomybės gali būti dinamiškas procesas [2].

2.2.Atviras kodas – alternatyva komerciniams produktams

Daug tyrinėtojų gilinosi į atviro kodo programinės įrangos projektų ir jų kūrėjų bendruomenių esmę bei bruožus. Egzistuoja keletas sėkmingo projekto faktorių ir potencialūs apribojimai AKPĮ inžinerijos procesams.

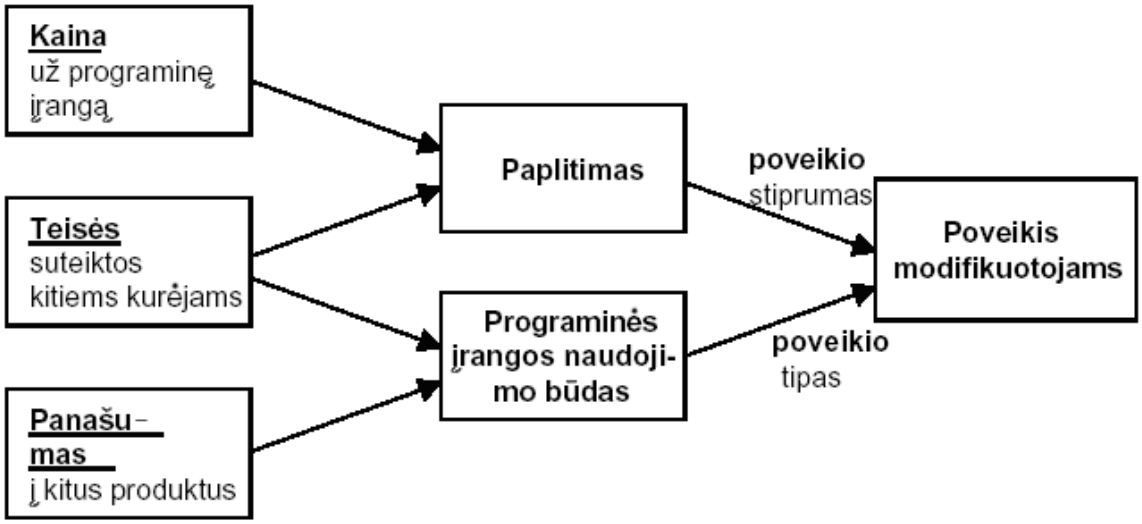
AKPĮ projektų sėkmė gali būti paaiškinama sparčiu projektų kūrimu/tobulinimu, jų patikimumu, portatyvumu ir sukurtos programinės įrangos išplečiamumu. Tokias savybes įtakoja išeities tekstų prieinamumas plačiajai visuomenei, programų kūrėjų priklausomybė PĮ kūrėjų bendruomenių grupėms, bei tas faktas, kad programų kūrėjai yra ir jų vartotojai [4].

AKPĮ projektų ištakos – asmeniniai poreikiai. Tokie poreikiai pritraukia kitų vartotojų-programuotojų dėmesį ir įkvepia juos prisidėti prie projekto. Kadangi programuotojai taip pat ir PĮ vartotojai – jie labai gerai supranta reikalavimus programai. Taip išvengiama daugiareikšmiškumo, dažnai lydinčio vartotojo poreikių identifikavimo procesą. Išeities tekstai atviri visiems individams, norintiems rasti, ištaisyti klaidas, patobulinti programą. Vartotojai, radę klaidą, gali ištaisyti ją patys, nelaukti, kol ją ištaisys programuotojai. Galima pridėti norimą specifinę programos funkciją – net jei pirminiai PĮ kūrėjai nemanė, kad ją verta realizuoti. Klaidos taisomas ir naudingos savybės vystomos labai greitai. AKPĮ bendruomenėje nariai keičiasi informacija apie programines klaidas, tikintis, kad kiti nariai jas ištaisys.

AKPĮ, priklausanti nuo savo nauda suinteresuotų programuotojų, gali būti mažiau tinkama kurti srities, su kuria programuotojai susiduria retai, programas.

Dėl labai didelio AK projektų skaičiaus ir propaguojamos laisvo žinių dalinimosi filosofijos, iš egzistuojančių projektų lengvai kuriami nauji, o kombinuojant egzistuojančių projektų resursus, pasiekiami nauji tikslai [6].

SourceForge.net – didžiausias pasaulyje atviro kodo programinės įrangos kūrimo internetinis portalas, teikiantis nemokamą vieta internete (*angl.* „free hosting“) dešimtims tūkstančių projektų. Sourceforge.net tikslas – praturtinti AK bendruomenę, suteikiant programuotojams galimybę centralizuotai kontroliuoti ir valdyti AK programinės įrangos kūrimą. Be to, AK bendruomenei ir puslapyje „priglaustiems“ projektams siūloma visa aibė paslaugų (pvz., metrikos).



1 pav. AK programinės įrangos ypatybių sąveika

Didelio paplitimo ir fakto, jog AKĮ (atviro kodo įrangą) gali modifikuoti visi, rezultatas (vienas dažniausiai minimų AKĮ privalumų) – su įranga dirba, tobulina, ieško klaidų ir išplečia daug kitų vartotojų/programuotojų, ne tik originalios versijos kūrėjų. Taigi prie AKĮ kūrimo prisidedanti įmonė ne tik lengviau, greičiau patobulina savo produktus, bet ir sumažina palaikymo išlaidas, laikantis prielaidos, jog įranga bus pakankamai domimasi ir tarp potencialių jos naudotojų bus kvalifikuotų, ją modifikuoti sugebančių žmonių.

Dažnai, apibūdinant komercinių ir atviro kodo produktų kūrimo specifikas, naudojamos metaforos „cathedral“ ir „bazzar“ („katedra“ ir „turgus“). „Turgaus“ metaforos taikymas ir pats „turgaus“ modelis susilaukia kritikos – jis laikomas prieštaringu [3].

Rinkoje atviro kodo programinė įranga konkuruoja su firminės mokamos programinės įrangos produktais dėl vartotojų dėmesio.

2.1 lentelė. Komercinių ir atviro kodo produktų kūrimas

	Komercinė programinė įranga	Atviro kodo programinė įranga
Modelis	„*cathedral* „	*„bazaar“*
Resursai	Žinomi	Nežinomi
Planavimo periodas	Visas projektas	Žingsnis po žingsnio
Tikslas	Įvykdyti kontraktą/ specifikacijas	Išspręsti problemą
Spaudimas	Stiprus	Silpnas
Progresas	Privatus	Viešas
Bendradarbiavimas	Betarpiškas	Interneto terpėje
Kokybės užtikrinimas	valdymas	Konkurencija, peržiūra

AKPĮ labai naudinga individualiam vartotojui. AK plitimas teikia privalumų valstybės ekonomikai. Todėl valstybinė AK skatinimo programa būtų teisinga ir reikalinga [16].

2.3.Atviro kodo programinės įrangos kūrimo aplinka

Tradiciškai, programinės įrangos kokybės pagrindas – tai formalizuotas ir kokybiškas PĮ kūrimo procesas, centralizuoti valdymo sprendimai („*from top to bottom*“). Visos šios savybės būdingos komercinėms programoms. Tačiau potencialiai kokybiškiems produktams kurti tinka ir tradicinė, ir atviro kodo aplinkos. AK nėra formalių vartotojo reikalavimų ir projektavimo specifikacijų, formalaus palaikymo. Ši aplinka skatina programinės įrangos kūrėjų konkuravimą ir inovacijas, palaiko individualų kūrybiškumą [8].

AK aplinkoje kuriamų produktų charakteristikos:

- masiškas iteratyvus programinio kodo peržiūrėjimas (paprastai tą atlieka patyrę programuotojai);
- modulinis dizainas (kiekvienas modulis gali būti modifikuojamas atskirai, taip sumažinant riziką indukuoti naujas klaidas);
- adaptyvus kuriamų sistemų modelis
    - dažnas naujų versijų pristatymas („*release*“);
    - PĮ evoliucija ir mutacija – greitas tobulinimas;

- aktyvios kūrėjų/naudotojų bendruomenės indėlis;
- potencialiai daugiau, nei komerciniuose analoguose, resursų;
- PĮ vartotojai kartu yra ir kvalifikuoti jos kūrėjai;
- modulinis projektavimas (atskirtos sąsajos ir funkcionalumo branduolys).

AK aplinkoje, bendrais interesais susiję bendruomenės nariai dažnai jungiasi į virtualias kūrėjų komandas. Komandos nariai sprendžia gailinasi ir sprendžia juos dominančias problemas. Tokios virtualios mokslinės komandos struktūra ir bendruomenės narių atsakomybės gali dinamiškai keistis. Nors apie internetu pagrįstų virtualių bendruomenių bendradarbiavimą žinoma nedaug, keletas akivaizdžių problemų, tai:

- svarbiausių kūrėjų pasitraukimas iš projekto prarandant interesą, dėl per didelės apkrovos ar pervargimo;
- konservatyvus požiūris į architektūrą (labai sunku pakeisti produkto architektūrą kai jo kūrimas jau prasidėjęs);
- komunikuojant elektroniniu paštu pasitaiko informacijos prasmės iškraipymų, gali būti sukeliami idėjiniai ginčai (*angl*. „flame wars“) [17].

Pakankamai gerą AK produktų kokybę ypač skatina atviro kodo ideologija. Ji pozityviai veikia ne tik AK kūrėjų komandų darbo efektyvumą, bet ir kognityvų tarpusavio pasitikėjimą, bendravimo kokybę PĮ kūrėjų grupėje. AK bendruomenėje laikomasi sutartinių normų ir įsitikinimų, bendruomenės nariai propaguoja tam tikras AK vertybes. Vertybių, įsitikinimų ir normų visuma ir sudaro AK ideologiją [15].

2.3.1. Normos

Egzistuoja pageidaujamos AK bendruomenės narių elgesio normos, pagrindinės jų:

- atviro kodo ideologijos normos įpareigoja kūrėjus neskelti projekto į keletą (*angl*. „no forking“);
- kodo pakeitimus platinti tik tinkamais kanalais;
- gerbti kodo autorystę ir nenutrinti autorystės žymių.

2.3.2. Įsitikinimai

Atviro kodo bendruomenės nariai laikosi įsitikimų:

- AK produktai yra geresnės kokybės nei komerciniai jų analogai;

- nemokama programinė įranga ir laisvai pasiekiama informacija naudingesnė ir duoda geresnius darbo rezultatus;
- kuo daugiau žmonių dirba su projekto programiniu kodu, tuo efektyviau šalinamos jo klaidos;
- praktinis darbas svarbesnis už teorinį mokymąsi;
- statusas įgyjamas tada, kais bendruomenės nario nuopelnus pripažįsta AK bendruomenė.

2.3.3. Vertybės

Atviro kodo bendruomenėje propaguojama aibė vertybių:

- dalinimasis informacija;
- pagalbos kolegoms teikimas;
- gerbtinos bendruomenės narių techninės žinios ir sugebėjimai;
- savišvieta;
- geranoriškas bendradarbiavimas;
- reputacija bendruomenėje (reputacija susikuriama dalyvaujant projektuose – „*egoboo*“).

Dalis AK bendruomenės įsitikinimų mažina dalyvių indėlį į komandinį darbą ir atitinkamai neigiamai paveikia produkto užbaigimą [18].

2.4.Atviro kodo produktų kokybės vertinimas

AK programinės įrangos kūrimo modelio atsiradimas gali pakeisti požiūrį į kokybės užtikrinimą. Kad projekto kokybės užtikrinimo komanda galėtų veikti, reikalingi kokybės vertinimo įverčiai, kurių AK aplinkoje kuriant projektus taikyti neįmanoma. Yra keletas kitų kokybės įverčių, bet jų nedaug. Bandoma taikyti eksperimentinius vertinimo modelius, remiantis projektų statistikos kriterijais. Vartotojai linkę AKPĮ vertinti pagal metrikas. Neegzistuoja sukurto brandaus atviro kodo produktų kokybės vertinimo modelio ir kokybės užtikrinimo procedūrų. Mažesniuose projektuose kokybės užtikrinimo procesai chaotiški ir neorganizuoti. Remiamasi filosofiniu AK požiūriu – pakankamai didelis projektą peržiūrinčių bendruomenės narių skaičius padės rasti ir ištaisyti visas programines klaidas. Dideliuose ir plačiai naudojamuose produktuose, pvz., *„Gnome“*, *„Linux“*, *„Mozilla“, „Alache“* ir pan., kokybės užtikrinimo problemos vertinamos atsakingiau.

Suklestėjus AK judėjimui, aktualiausi tapo tokie klausimai:

- būdai, kuriais galima tikrinti AK produktų kokybę;
- faktoriai, veikiantys AK produktų kokybę;
- AK požiūris į kokybės užtikrinimą.

Nors AK projektams negalima taikyti kokybės kontrolės tikrąja to žodžio prasme, galima įvertinti projektus, taikant ISO/IEC 9126 standartą, pagal šešis pagrindinius atributus:

- **Funkcionalumas**
  - funkcionalumui labai naudingi atviri AK produktų standartai ir produktų sąveika;
  - produktai tinkami ir plačiai taikomi, pvz., AK operacinės sistemos ar programavimo kalbos (didelę rinkos dalį užėmusi alternatyvi komercinei įrangai AKPĮ, pvz. „*Linux*“ ar „*Apache,*“ o rinkos spragas užpildo naujo, vartotojo norus tenkinančio funkcionalumo AK aplikacijos ir įrankiai);
  - AK projektai retai detaliai specifikuojami (nepopuliarus darbas), todėl mažai specifinio funkcionalumo AK produktų (AK aplinka tinkamesnė ne specifinio (vertikalaus) funkcionalumo, o plataus (horizontalaus) funkcionalumo PĮ);
- **Patikimumas**
  - klaidos operatyviai randamos ir efektyviai taisomos (didelė bendruomenė), pvz., 50% „*Apache*“ tarnybinės stoties klaidų pataisoma per mažiau nei 24 valandas nuo klaidos pastebėjimo datos;
  - organizuotas komunikavimas tarp suinteresuotų projekto dalyvių – naudotojų ir kūrėjų;
  - programinio kodo pasiekiamumas daro AK produktus patikimesnius ir saugesnius, negu jų komerciniai analogai;
  - kūrimas pagrįstas savanorišku darbu, todėl visada rizikuojama, kad savanoriai paliks savo paketus – tai neigiamai veikia patikimumą.
- **Naudojamumas**
  - AK produktai dažniausiai skirti kūrėjams, o ne galutiniam vartotojui, todėl naudojamumas nėra prioritetinė produktų savybė;
  - naudojamumo klaidos sunkiai aprašomos ir taisomos;

- dažnai AK įrangą sudėtinga įdiegti;

- **Palaikomumas**
  - programinės įrangos kokybė prastėja, kai nėra jos palaikymo (uždaro kodo projektuose palaikymo fazei skiriama iki 90% projekto kainos), o AK projektuose negarantuojamas nei palaikymas, nei pageidaujamų funkcijų realizavimas; laikomasi nuomonės, kad palaikomumas gerėja, projektui pasiekiant aukštesnius brandumo lygmenis ir plečiantis kūrėjų bei vartotojo bendruomenėms [5];
  - rizikuojama, kad sumažėjus savanorių, prisidedančių prie projekto, palaikymas bus nutrauktas;
  - nedaug dokumentacijos, todėl palaikyti produktą sudėtinga;
  - paprastai vykdoma daug AKPĮ produkto versijų išlaidimo iteracijų, kiekvienoje iteracijoje pridedant naujo funkcionalumo.

- **Perkeliamumas**
  - ši savybė - prioritetas, kuriant atviro kodo PĮ (bendru atveju siekiama AKPĮ universalumo);

- **Efektyvumas**
  - nėra patikimų šio atributo vertinimo strategijų, produkto efektyvumas labai priklauso nuo vartotojo (pvz., *„Windows“* ir *„Linux“* tarpusavio palyginimas);

Bendruoju atveju, pagal ISO/IEC 9126 PĮ kokybės standartą, AK programinė įranga potencialiai gali būti kokybiška [13].

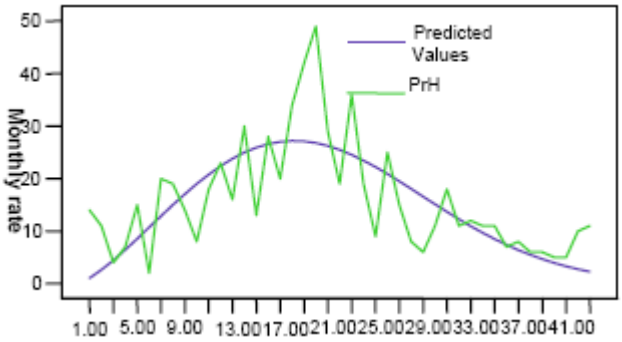
AK produkto kokybės reikalavimai, vystant produktus, keičiasi. Pirmiausia turi būti tenkinami funkciniai reikalavimai – vertinant, AK produktas lyginamas su komercinių produktų funkcionalumu arba naujais vartotojų poreikiais, jei analogiško komercinio produkto nėra. Vėliau svarbiu tampa patikimumas. Vartotojas gali rinkis senesnę, patikimesnę, stabilią AK produkto versiją, turinčią mažiau funkcionalumo. Ėmus produktą vartoti, svarbus palaikomumas (korekcinis, adaptyvus ir prevencinis). Korekcinis palaikomumas neigiamai veikia PĮ kokybę, kadangi programos kodas tampa sudėtingesnis. Prevencinis, iš anksto apgalvotas palaikomumas gerina produkto kokybę, o adaptyvusis palaikomumas labai priklauso nuo PĮ aplinkos dinamiškumo. Plečiantis sistemai aktualiausias pakartotinis panaudojamumas (viena vertingiausių ir plačiai išnaudojamų AK savybių).

2.5.Atviro kodo produktų kokybės valdymas

2.5.1. Tenkinančio kokybės lygio pasiekimo numatymas

Atviro kodo ir komercinių produktų programinėje įrangoje aptinkamų klaidų skaičius stabilizuojasi pasiekus reikiamą patikimumo lygį. AK produktams, išskyrus tokius didelius projektus kaip *„Gnome“, „Linux“, „Mozilla“, „Alache“* ir pan., vertinti kokybę ir prognozuoti stabilumą bandoma naudojant įvairias statistines projektų metrikas (dydis, kūrėjų skaičius, veiklos aktyvumas, periodiškai randamų klaidų skaičius, puslapio peržiūros ir PĮ parsiuntimo dažnumas). Paprastai AK produktų svetainėse ar dideliuose portaluose (pvz. http://sourceforge.net/) pateikiama daugybė metrikų, pagal kurias vartotojui spręsti apie produkto kokybę sudėtinga.

PĮ produktų patikimumui prognozuoti plačiai naudojami eksponentiniai modeliai. AK projektams pritaikius aibę eksponentinių patikimumo augimo modelių, pagal tyrimo rezultatus, geriausiai AK projektų patikimumui (ir atitinkamai kokybei) prognozuoti tinkantis modelis – tai bendras Weillbullio pasiskirstymas [19]. Paveikslėlyje (**2 pav.**) pateiktas pavyzdinio projekto klaidų aptikimo dažnio ir pagal Weillbullio eksponentinį patikimumo modelį prognozuotų klaidų skaičiaus grafikas.



2 pav. Pavyzdinio AK projekto patikimumo augimo modelis (klaidų skaičius)

Remiantis tyrimais, AK projektų puslapių peržiūros ir programų parsiuntimo dažnio, bei klaidų radimo koreliacija silpna ir ši kokybės vertinimo metodika nepasiteisina – peržiūros ir parsiuntimo statistika indikuoja ne AK produkto kokybę, o priimtinumą vartotojams. Be to, daugumą pastebėtų klaidų aprašų pateikia ne vartotojai, o maža kūrėjų bendruomenė [20].

2.5.2. Programinių paketų kontrolė ir savanorių veiklos valdymas

Labai svarbi AK PĮ kokybės valdymo strategija – savanorių kontrolė. Problemų šaltinis-savanorių patikimumas, jų darbų koordinavimas, savanoriško darbo rizika ir pasekmės projekto

kokybei, skirtingas požiūris į savanorių pareigas. Iš projekto pasitraukus savanoriui, atsakingam už kritinės svarbos paketą, gali nukentėti projekto kokybė, arba projektas gali būti visai sustabdytas.

AK bendruomenę dalina dvi skirtingos savanorių pareigų vertinimo filosofijos. Pirmoji filosofija nurodo, kad savanoriškame darbe neturėtų būti jokių įsipareigojimų, o programuotojas gali daryti, ką panorėjęs. Pagal antrąją filosofiją, sutinkant dalyvauti projekte, savanoriui tenka atsakomybė už sako projektinę dalį. Nusprendęs palikti projektą, jis turėtų pasirūpinti, kad jo darbai būtų saugiai perduoti kitiems projekto dalyviams.

Paprastai savanorių aktyvumas nėra pastovus. Periodiškai savanoris gali apleisti savo pareigas (pvz., prie AK projekto dirbantis studentas turi skirti visą savo laiką mokslams sesijos metu). Savanorių veiklos stebėjimas ir valdymas gali smarkiai paveikti kuriamos PĮ kokybę. Ypač tai aktualu dideliuose projektuose, pvz., *„Debian“*, kuriame naudojama tokia savanorių darbo kontrolės metodika:

- stengiamasi, kad vieną paketą prižiūrėtų keletas programuotojų, o jei paketu nesidomima – šis paketas į eilinę distribuciją neįtraukiamas;
- paketų korekcijos kontrolė – pakeitimus gali atsiųsti tik tam tikri, identifikuoti projekto dalyviai (naudojami GPG – *„GNU Privacy Key*“ ar PGP – *„Pretty Good Privacy“* raktai);
- nauji projekto dalyviai kruopščiai atrenkami (socialinės integracijos, filosofijos ir techninių sugebėjimų, galimybės įsipareigoti tikrinimas, ilgas atrankos procesas);
- savanorių aktyvumo stebėjimas (jei pastebima, kad vartotojas pora savaičių neatnaujino savo programinio kodo, jam siunčiama užklausa; dar po poros savaičių, jei vartotojas neatsiliepia, jam siunčiamas perspėjimas apie paketo perdavimą; nesulaukus atsakymo, atsakomybė už paketą perduodama kitiems projekto dalyviams);
- nuolat kontroliuojama paketų būsena, pakeitimai ir kaupiamos paketų metrikos (kritinių netaisomų klaidų skaičius, paketo atitikimas produkto versijai);
- naudojami automatizuoti veiklos sekimo įrankiai [9].

2.5.3. AK projektų testavimas

Vidutiniams ir mažiems projektams paprastai nekuriama testavimo planų, kadangi AK aplinkoje PĮ kūrimo veikla nėra centralizuotai valdoma, o savanoriai tokių darbų kaip PĮ funkcionalumo specifikavimas ir dokumentavimas nemėgsta ir nedaro. Testavimo fazėje praleidžiama nedaug projekto laiko, pakartotino kodo peržiūros kartų skaičius proporcingai didesnis didesniuose ir populiaresniuose projektuose. Testavimo įrankiai naudojami mažesnei projektų

daliai, o didžiausias klaidų skaičius, remiantis tyrimų medžiaga, randamas AK projektų vartotojo sąsajoje [11].

AK projektuose testavimui paprastai skiriama mažiau dėmesio, nei kuriant komercinę programinę įrangą. Pagal tyrimų duomenimis:

- daugiau nei 80% AK projektų neturi jokių testavimo planų;
- vidutiniškai testavimo stadijoje praleidžiama 40% laiko;
- mažų projektų, kodas pakartotinai peržiūrimas tik 1.3 karto, didelių – 7.5;
- 39.6% projektų buvo naudojami testavimo įrankiai;
- daugiau nei 50% nenaudoja įrankių patikrinti kodo kelių padengimui;
- daugiausia klaidų randama projektų vartotojo sąsajoje – 28.6%.

2.6.Aktuali situacija AK produktų kokybės užtikrinime

Įvertinant visą aibę AK produktų privalumų, kokybė nėra svarbiausias iš jų. Kuriamos ir tobulinamos AK produktų kokybės valdymo strategijos yra individualios, o kokybės vertinimas naudojantis projektų metrikomis nėra tikslus.

3. Projektinė dalis

Projekto metu buvo atviro kodo priemonėmis realizuota individuali apskaitos sistema.

3.1.Sistemos paskirtis

Projekto tikslas - sukurti paprastos, suprantamos vartotojo sąsajos apskaitos programą, kontroliuoti smulkios, mažo biudžeto, valstybės finansuojamos socialinių paslaugų organizacijos buhalterijai ir kasdieninei veiklai. Naudojantis programine įranga, organizacijos darbuotojai galės optimizuoti savo veiklą, greičiau rasti ir koreguoti norimą informaciją (nereikės gaišti su klientų, rėmėjų popieriniais dokumentais), konfidencialiai atlikti finansines operacijos (automatizuoti periodines operacijas, naudotis ryšiu su banko sistema). Į elektroninį formatą, patogesniam naudojimui bus pervesti organizacijos duomenys - sukurtos inventoriaus, darbuotojų, klientų, finansinių operacijų duomenų bazės. Programine įranga galės naudotis visi organizacijos darbuotojai – ji tenkins ir vadovaujančio, ir buhalterijos personalo, ir kitų organizacijos darbuotojų (sociologų, psichologų) reikalavimus.

3.2.Egzistuojantys sprendimai (komercinės sistemos ir AK analogai)

Šiuo metu pasaulinėje rinkoje yra nemažai produktų, skirtų tam tikriems socialinių organizacijų poreikiams tenkinti. Dauguma jų – brangūs. Ne pelno siekiančių organizacijų biudžetas paprastai ribotas, nes jas finansuoja valstybė, ar kitos organizacijos, individualūs rėmėjai. Vakarų Europoje ar JAV, kur labdaringos veiklos rėmimas išplėtotas, o ekonominė padėtis geresnė, egzistuoja plati rinka firmoms, kuriančioms atitinkamą programinę įrangą.

3.1 lentelė. Komercinė programinė įranga nepelno organizacijoms

	Araize	Blackbaud	Fund E-Z	ForFund	Non-Profit Series	Nonprofit-Books
Pardavėjas	Araize, Inc.	Blackbaud, Inc.	Fund E-Z Development Corp	Mirasoft, Inc	Micro Information Products, Inc	B2PCommerce Corp
Tipinė kaina vienam vartotojui	2800 Lt	21000 Lt	5200 Lt	31500 Lt	Nuo 4500 Lt iki 9000 Lt	nuo 900 Lt iki 1800 Lt
Operacinė sistema	Windows	Windows	Windows	Windows	Windows	Windows
Tiklo PĮ	Windows, Novell and LANtastic	Windows and Novell	Windows and Novell	Windows and Novell	Windows and Novell	Windows and Novell

Didžiąją individualios apskaitos programinės įrangos rinkos dalį užima jau seniai ten įsigalėję komerciniai produktai. Populiarios ir keletas AK programų, skirtų buhalterijos tvarkymui. Dauguma jų, kartu su išeities tekstais platinamos nemokamai. Kai kurių AK programų papildomi moduliai platinami už simbolinę kainą. Parsisiųstus programų išeities tekstus galima keisti, tobulinti, pritaikyti savo specifiniams poreikiams.

Organizacijoms, nesiekiančioms pelno, užsiimančioms labdara ar teikiančioms socialines paslaugas, tvarkant buhalteriją, be įprastinių, prireikia tokių papildomų funkcijų, kaip paramos fondų tvarkymas ir savanorių veiklos administravimas. Yra nemažai komercinių programų, skirtų tik tokiems specifiniams uždaviniams.

Nepavyko rasti programinės įrangos su buhalterinėje apskaitos sistemoje integruotomis reikalingomis nepelno organizacijai funkcijomis.

3.2.1. „jGnash“

„*jGnash*“ programa gali būti naudojama sąskaitų balansui apskaičiuoti. Galima spausdinti čekius, gauti paprastas finansines ataskaitas, rašyti savo scenarijus. Programos duomenų apdorojimo sistema yra unikali, palyginus su kita finansine programine įranga.

„*jGnash*“ – dvigubo įvedimo (*angl.* „double entry“) sistema, bet galima nustatyti ir pavienio (*angl.* „single entry“) įvedimo rėžimą, palengvinant sąskaitų balanso koregavimą. Dvigubam įvedimui jGnash naudoja sąskaitas vietoj kategorijų. Dvigubas jGnash įvedimas veikia panašiai kaip kitose komercinėse finansinėse programose. Todėl nereikia mokytis naujo būdo sekti savo išlaidoms.

Programos kūrėjai tiki, kad dvigubo įvedimo finansinė programinė įranga neturėtų pamėgdžioti senas, popierines sąskaitų buhalterines knygas, kai tą patį rezultatą galima pasiekti naudojant paprastesnį apskaitos procesą. Kūrėjų tikslas – sukurti individualios finansinės apskaitos programinę įrangą, kuri būtų lengvai naudojama, tiksli, lanksti ir nemokama. Pilna individuali finansinė programa realizuoja:

- Sąskaitų tvarkyklę
- Biudžeto stebėjimo modulį
- Ataskaitų formavimo modulį
- Galimybę prijungti naujus modulius

Programos branduolys išlaikytas paprastu. Tai reiškia, kad vartotojui nereikia jaudintis, jog papildomos programos funkcijos(biudžeto sekimas ar ataskaitų formavimas) sunaudos didelę kompiuterio resursų dalį.

3.3.Reikalavimų specifikacija

3.3.1. Vartotojai

3.2 lentelė. Sąrašas asmenų, kurie betarpiškai naudosis sistema

Vartotojo kategorija	Vartotojų prioritetai			Vartotojo sprendžiami uždaviniai	Patirtis dalykinėje srityje			Patirtis informacinėse technologijose		
	Svarbiausi	Antraeiliai	Nesvarbūs		Naujokas	Įprastas darbuotojas	Srities specialistas	Naujokas	Patyręs	Informatikas
Organizacijos vadovas	+			Finansinių organizacijos operacijų kontrolė; Darbuotojų duomenių paieška, redagavimas ir papildymas; Klientų duomenų paieška, redagavimas ir papildymas; Pedagogų, psichologų, praktikantų, bei socialinių darbuotojų profesinės kvalifikacijos sekimas, kėlimas, koordinavimas; Ryšių su organizacijomis-partneriai palaikymas ir informacijos apie jas paieška			+	+		
Organizacijos darbuotojas (sociologas, psichologas)		+		Informacijos apie norimą klientą (asocialią šeimą ar vaiką) radimas; Informacijos apie klientą redagavimas, pildymas; Statistinių, specifinių ataskaitų apie organizacijos klientus formavimas ir spausdinimas;			+	+		
Buhalteris	+			Informacijos apie organizacijos sąskaitų būklę išgavimas ir kontrolė; Pinigų pervedimas tarp sąskaitų - atsiskaitant su prekių ar paslaugų tiekėjais, darbuotojais; Išlaidas ar įplaukas registravimas, sekimas ir kontrolė; Norimos specifikos finansinių ataskaitų sudarymas ir spausdinimas;			+		+	

3.3.2. Projekto apribojimai

Programinė įranga turi veikti organizacijos turimuose kompiuteriuose, neviršyti jų galimybių. Organizacija jau turi įsigijusi techninę įrangą

Sistema diegiama organizacijos kompiuteriu tinkle. Įvairiuose kompiuteriuose įdiegtos skirtingos operacinės sistemos (Windows XP, Windows 2000), planuojama greitai pereiti prie nemokamų Linux OS. Organizacija turi galimybę bet kada prisijungti prie interneto tinklo.

Sistemai kurti naudojami atviro kodo įrankiai ir atviro kodo produktai (*„jGnash“, „FireBird“, „FlameRobin“, „MySQL*“, kodas rašomas Java programavimo kalba).

Numatoma rami, stacionari darbo vieta, lengvai pasiekiamos bendradarbiaujančios sistemos, spausdinimo įrenginiai.

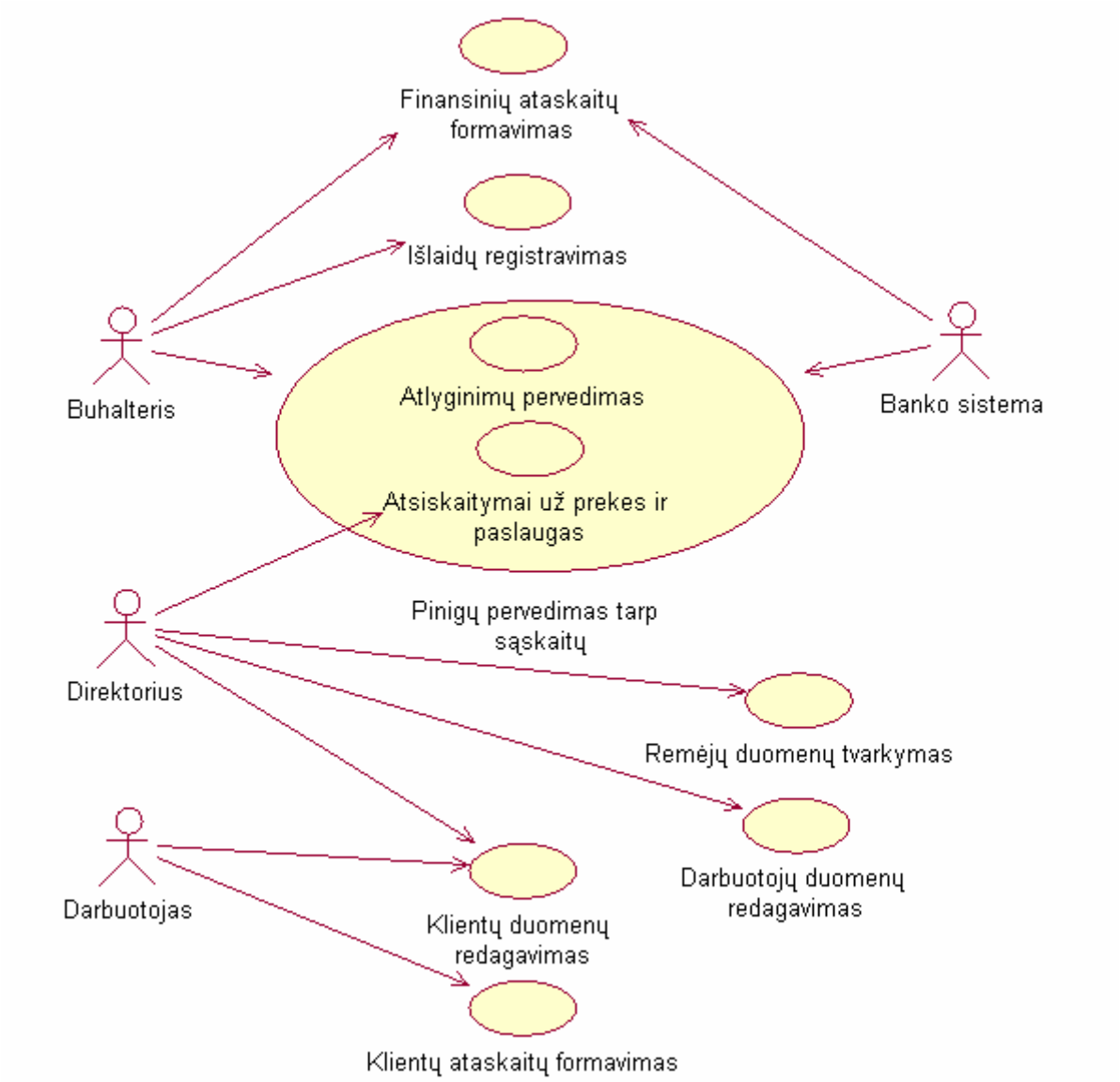
Programinę įrangą realizuos vienas programuotojas, nes projekto mastai nedideli – reikės tobulinti jau egzistuojančius sprendimus. Programuotojas dirba nemokamai, kaip savanoris, prisidedantis prie užsakovo organizacijos labdaringos veiklos.

Produktui sukurti bus reikalinga bent pora kompiuterių, sujungtų į tinklą. Taip reikalingas internetinis ryšys – sąsajai su banko sistema kurti. Reikalinga programinė įranga - įrankiai programuoti JAVA (pvz., „*CodeGuide Professional 7.0*“), duomenų bazė (pvz. *„MySQL Server“*), Windows aplinka.

Svarbūs faktai ir prielaidos:

- Kuriama programinė įranga bus pateikta kaip atviro kodo programinė įranga visiem norintiems ja naudotis.
- Bendravimui su internetinėmis sistemomis bus jungiamasi prie atitinkamų banko sąsajų, informacija perduodama numatyto formato įrašų failais.
- Duomenų bazę reikės įdiegti tik viename kompiuteryje. Klientinė sprendimo dalis veiks v visuose tinklo kompiuteriuose.

3.3.3. Funkciniai reikalavimai



3 pav. Programinės sistemos panaudojimo atvejai

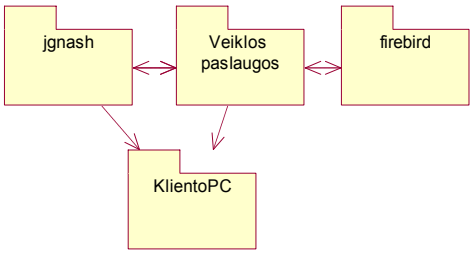
Projektas pagrįstas pakartotinu atviro kodo produktų panaudojimu, tobulinant ir specializuojant.

Kadangi sistema kuriama nemokamomis ,laisvai prieinamomis atviro kodo priemonėmis, jos išeities tekstai pateikiami visiems suinteresuotiems asmenims nemokamai. Programa dykai atiduota naudoji organizacijoje.

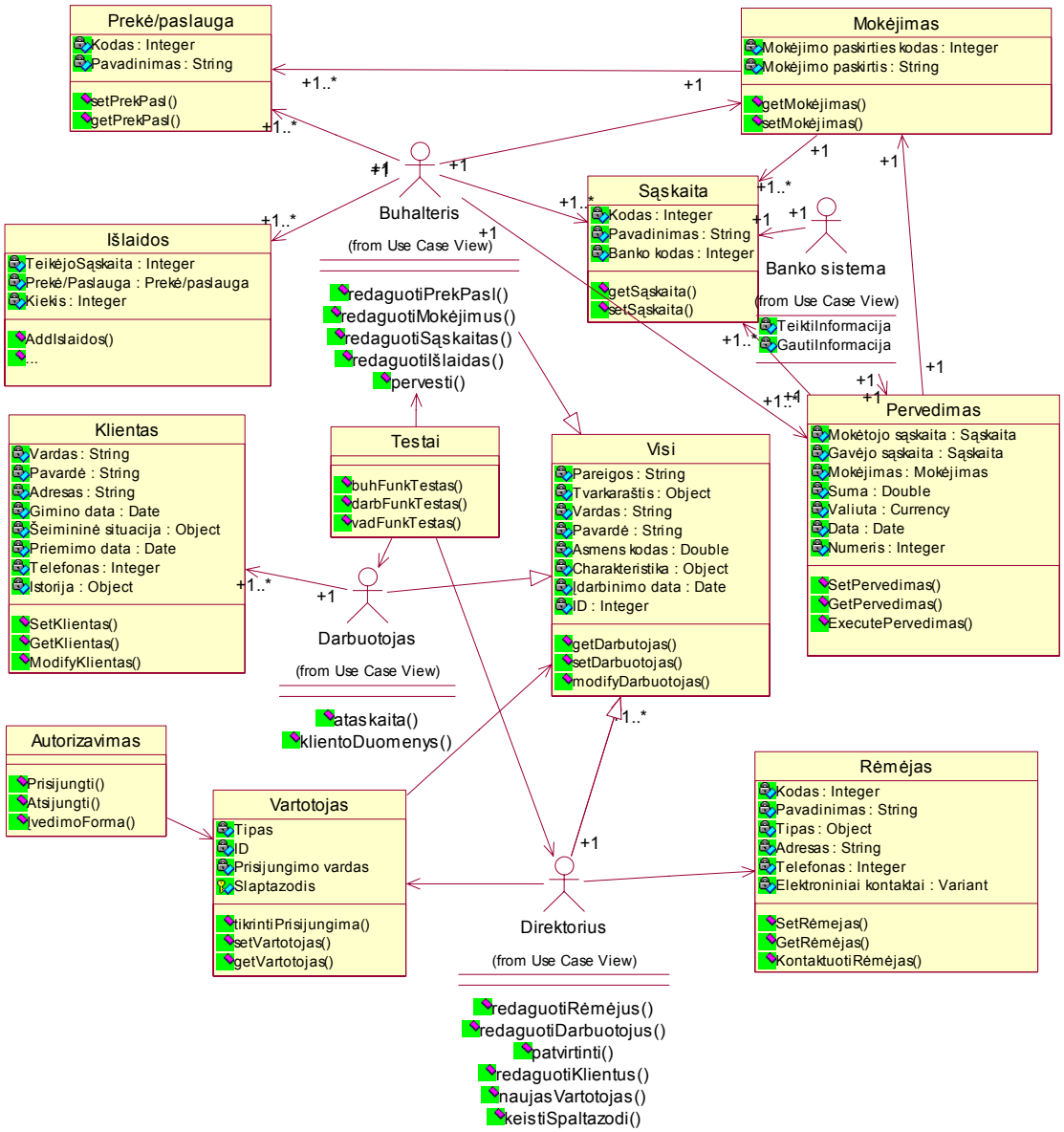
Pieš pradedant eksploataciją, vartotojas apmokomas, jam pateikiama tokia produkto dokumentacija

3.4.Architektūra

Sistemą sudaro AK programos „jGnash“ paketas ir keletas papildyto funkcionalumo paketų:



4 pav. Sistemos skaidymas į paketus

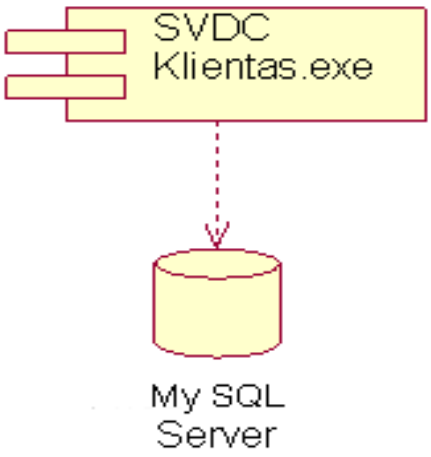


5 pav. Veiklos paslaugų komponento struktūra

6 pav. Sistemos išdėstymas

Vartotojo programinė įranga veiks Microsoft® Windows™ ir Linux operacinėse sistemose, jei ten bus įdiegta Java Virtual Machine (JVM) 1.4.0 ar naujesnė versija. Rekomenduojama 800*600 dpi ekrano raiška.

Komponentinė programos kūrimo strategija, leidžia ateityje sistemą papildytu naujais komponentais arba pakeiti jau egzistuojančius. „*jGnash*“ egzistuoja scenarijų rašymo galimybės, integruotas interpretatorius leidžia paprastai papildyti programą naujomis savybėmis ir jas testuoti.



7 pav. Sistemos skaidymas į komponentus

3.5.Realizacija

3.5.1. Vartotojo sąsaja

Trijų tipų vartotojams (vadovas, sociologas ir buhalteris) skirtos skirtingo funkcionalumo sąsajos.

a) Vartotojų sąsajos ir jų valdomos funkcijos (vartotojas-vadovas)

Atveriamas prisijungus vadovo tipo darbuotojo vartotoju.

8 pav. Vadovo vartotojo tipo sąsaja

[Klientai]	Mygtukas skirtas atverti visų organizacijos klientų sąrašo langą, kuriame vadovas gali valdyti jų duomenis. Klientų sąrašo langas atitinka pagrindinį sociologo/psichologo darbuotojo langą.
[Rėmėjai]	Meniu punktas skirtas atverti rėmėjų sąrašo langą, kuriame rodomas ir gali būti valdomas visų organizacijai paramą teikiančių asmenų ar organizacijų sąrašas.
[Darbuotojai]	Meniu punktas skirtas leisti vadovui peržiūrėti darbuotojų sąrašo lango turinį, iš kurio, su darbuotojų duomenimis galima atlikti įvairius veiksmus (ieškoti, koreguoti, ištrinti).
[Vartotojai]	Meniu punktas skirtas atverti vartotojų sąrašo langą. Tai sistemos vartotojų administravimo langas, kuriame galima nurodyti naujus programos vartotojus, koreguoti ar išmesti senus,

[Išlaidos]	Mygtukas skirtas kontroliuoti organizacijos išlaidas. Atveriamas buhalterinių duomenų kaupimo langas (planuojamų atsiskaitymų su rangovais ar paslaugų tiekėjais). Vadovas kiekvieną iš šių pervedimų gali patvirtinti arba ne.

b) Vartotojų sąsajos ir jų valdomos funkcijos (vartotojas-sociologas/psichologas)

Atveriamas prisijungus sociologo/psichologo tipo darbuotojo vartotoju.

9 pav. Sociologo/psichologo vartotojo tipo sąsaja

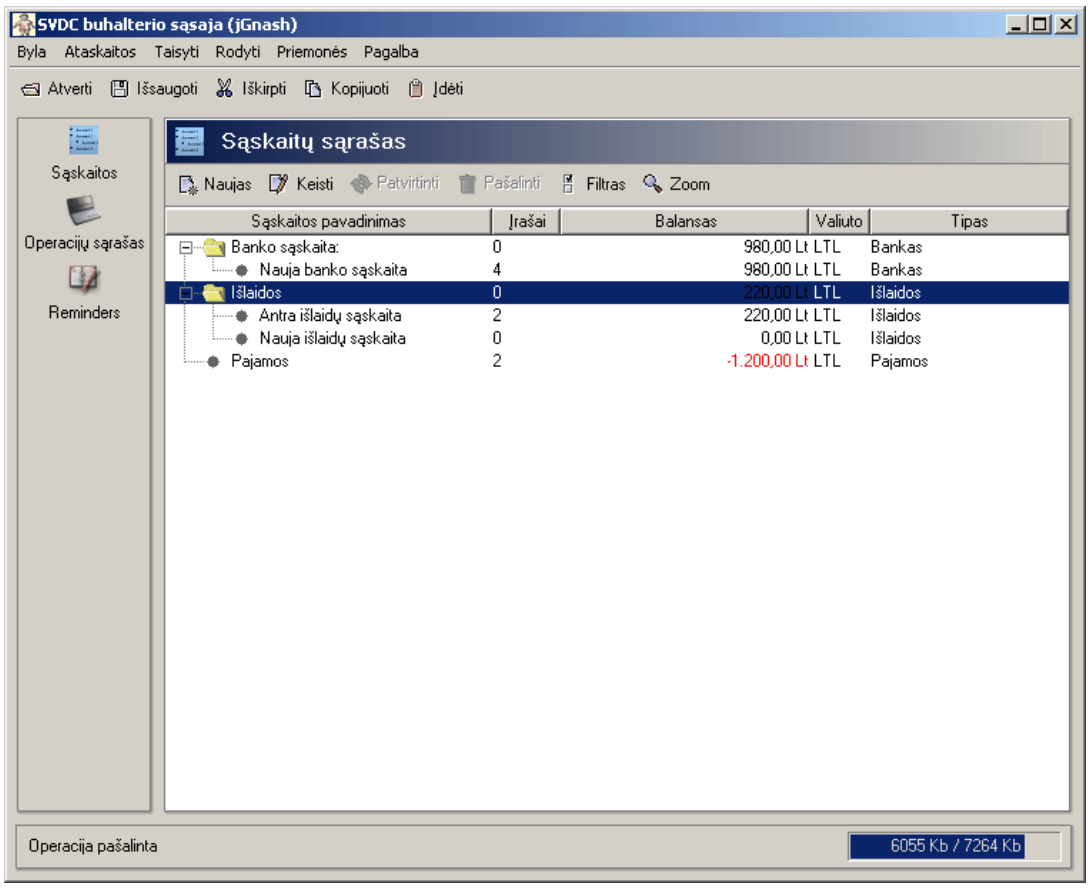
c) Sociologo/psichologo sąsaja

[Naujas]	Šis meniu punktas skirtas atverti naujo kliento įrašo sukūrimo langą, kuriame vartotojas gali įrašyti norimus naujo kliento atributus.
[Peržiūrėti]	Iš sąrašo pasirinkus eilutę ir paspaudus šį mygtuką, atveriamas kliento langas, užpildytas pasirinkto iš sąrašo kliento duomenimis. Juos galima koreguoti ir įrašą vėl išsaugoti. Analogiškas meniu punktas yra ir išskleidžiamame meniu, paspaudus dešinįjį pelytės mygtuką ant sąraše pasirinktos eilutės.
[Ištrinti]	Meniu punktas skirtas ištrinti pasirinktą iš sąrašo kliento įrašą. Analogiškas meniu punktas taip pat yra išskleidžiamame meniu.
[Ieškoti]	Viršutinėje lango dalyje yra grupė laukelių. Jie skirti įrašyti filtro, pagal kurį lange bus atrinkti ir rodomi klientų duomenys,

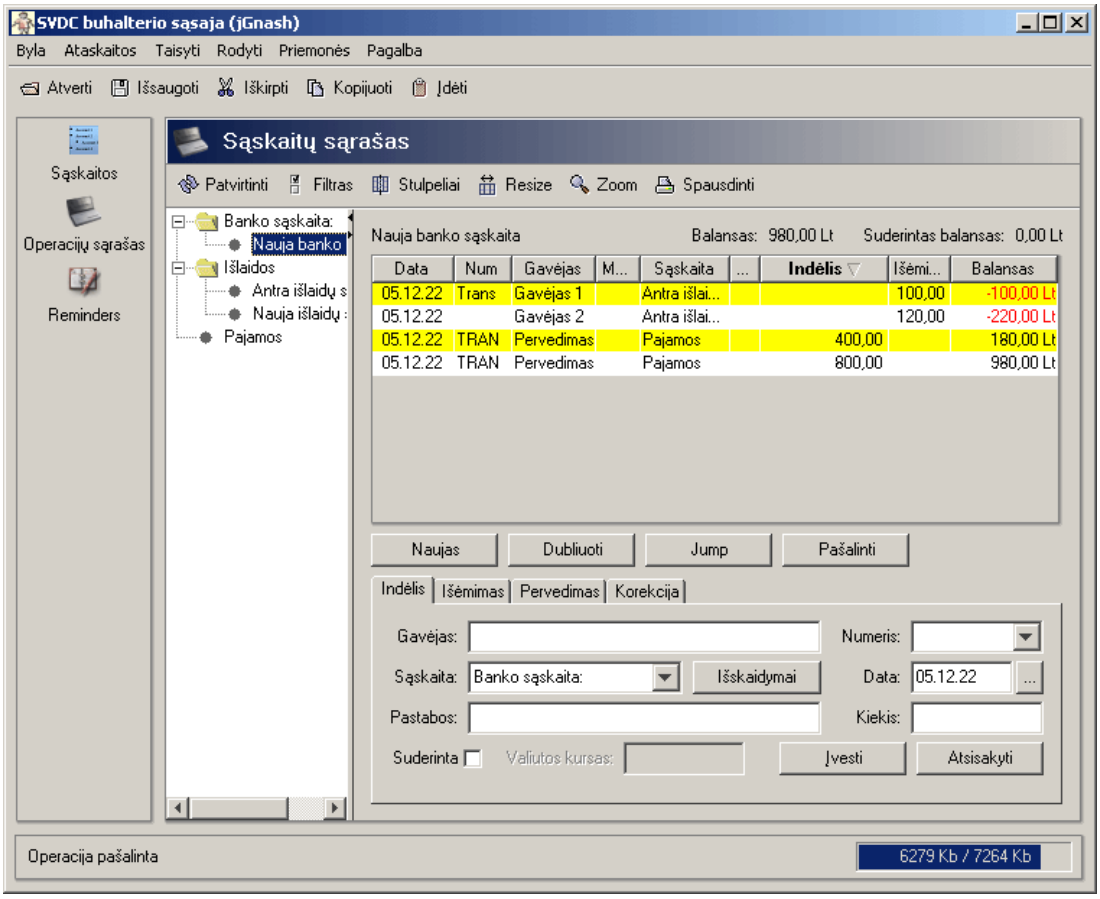
	parametrus. Paspaudus [ieškoti] mygtuką, Lange bus rodomi tik filtrą atitinkantys klientai.
[Spausdinti sąrašą]	Šis mygtukas skirtas suformuoti ir pateikti spausdinimui lango sąraše matomų klientų įrašų ataskaitą.
Ataskaitos	Meniu punktas turi daug submeniu punktų, kurių kiekvienas tam tikros formos, organizacijos naudojamą ataskaitos dokumentą.
Atnaujinti	Meniu punktas atnaujina rodomą sąrašą.
Išeiti	Meniu punktas skirtas uždaryti klientų langą.

d) Vartotojų sąsajos ir jų valdomos funkcijos (vartotojas-buhalteris)

Į SVDC apskaitos sistemą integruota kita apskaitos programa – „jGnash“. Buhalterio funkcionalumas veikia „jGnash“ programos pagrindu. jGnash sąsaja išversta į lietuvių kalbą, o šios programos vartotojo vadovą ir likusias specifikacijas galima rasti interneto svetainėje http://jgnash.sourceforge.net/wiki/index.php/Main_Page.



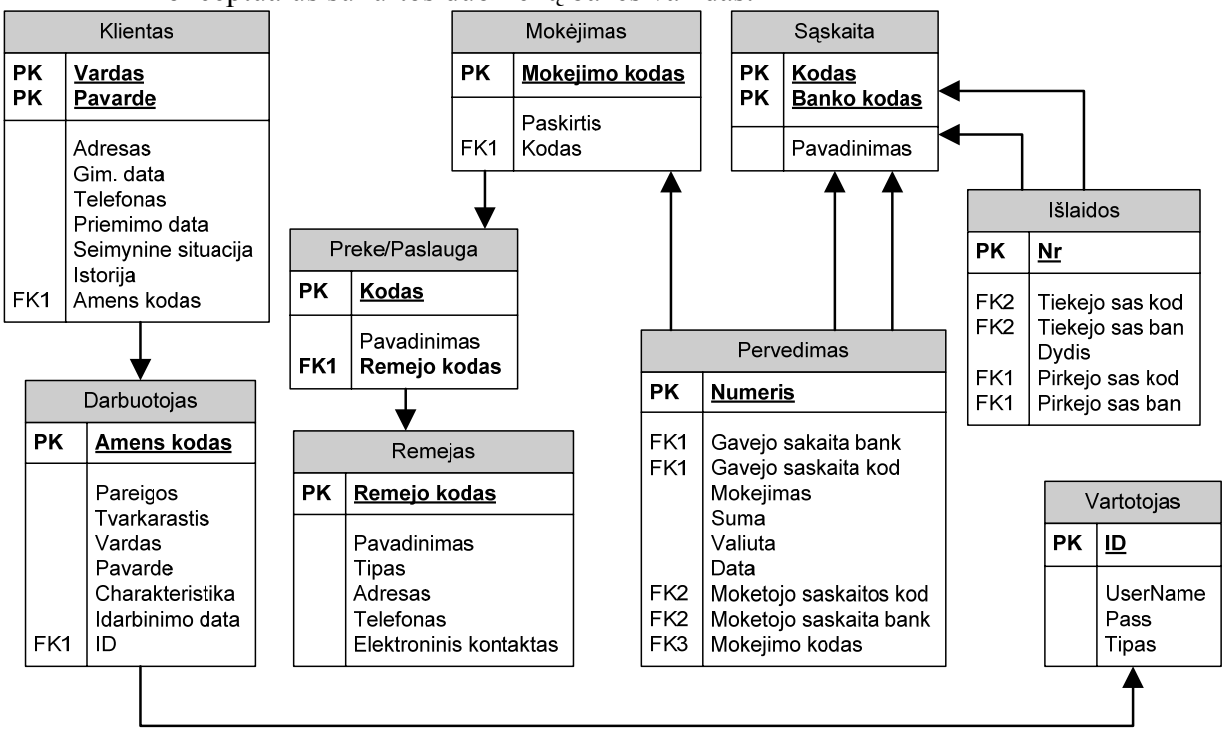
10 pav. „jGnash“ sąsaja (sąskaitos)



11 pav. „jGnash“ sąsaja (operacijos su sąskaitomis)

3.5.2. Duomenų bazė

Konceptualus sukurtos duomenų bazės vaizdas:



12 pav. SVDC duomenų bazės schema

3.6.Sistemos testavimas

Praktiškai neįmanoma pasiekti, kad programoje nebūtų jokių klaidų. Ypač, jei programa turi būti sukurta per griežtai ribotą laikotarpį. Siekiama aprašyti daugumą testavimo atvejų, kad atliekant testavimo darbus, būtų tikrinami visi reikalingi objektai, duomenų srautai, programinės įrangos apribojimai. Tikslas – taip patobulinti programinės įrangos kūrimą, kad būtų susidorojama su bet kokiomis klaidomis.

Užtikrinant programinės įrangos kokybę, atlikti skirtingų tipų testai.

- Vienetų testavimas (atliekamas naudojant juodos dėžės testavimo metodus, naudojant „*JavaUnit*“ (http://www.junit.org) įrankį)
    - Servisai
    - Klientų programos (naudojamas „*Abbot Java GUI Test Framework*“ http://abbot.sourceforge.net/
        - Vadovo
        - Buhalterio
        - Sociologo
    - Duomenų bazė (naudojamas „*Dbunit Database Testing Framework*“ - http://dbunit.sourceforge.net/)
- Integravimo testavimas
- Regresinis testavimas (naudojamas „*JavaUnit*“ (http://www.junit.org) įrankis)
    - Servisai
    - Klientų programos
        - Vadovo
        - Buhalterio
        - Sociologo
    - Duomenų bazė
- Vartotojo sąsajos testavimas
    - Vadovo
    - Buhalterio
    - Sociologo
- Priėmimo testavimas
    - Klientų programos
        - Vadovo

- Buhalterio
- Sociologo

Techniniai ir resursų apribojimai galimybėms atlikti visus reikalingus programinės įrangos testavimus.

- Kliento įmonėje programos veikimą galima testuoti tik ribotą laiką (darbo valandomis)
- Nėra pakankamai didelės vartotojų grupės, kad tuo pačiu metu dirbtų programa, atliekant stresinį testavimą.

3.7.Vartotojo dokumentacija

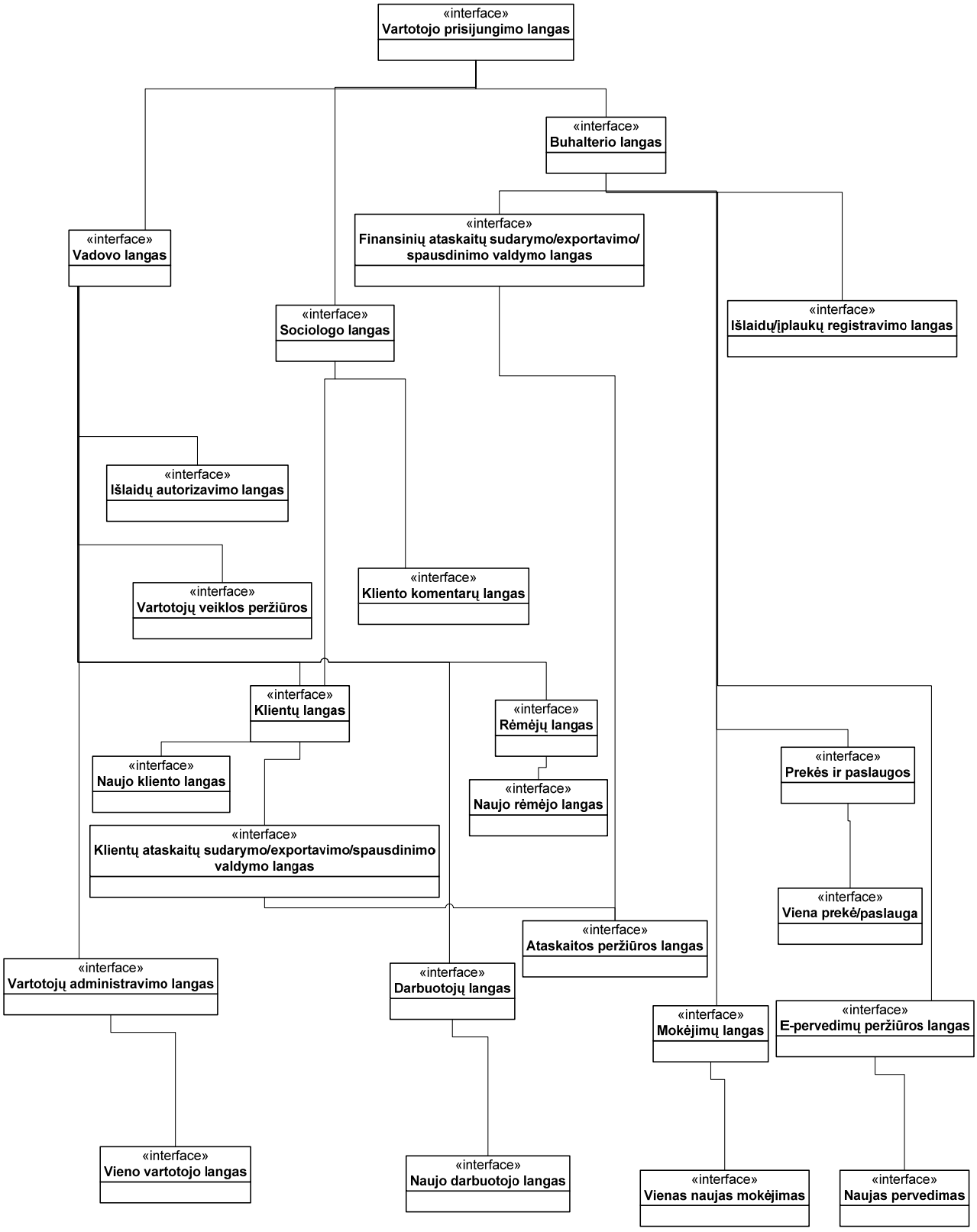
Produktui galioja bendroji viešoji licencija (GNU – *„General Public License“*).

Atviro kodo produktas skirtas automatizuoti nepelno siekiančios, smulkios socialinių paslaugų organizacijos, veiklos procesus ir juos valdyti vieningoje integruotoje sistemoje. Programa tenkina skirtingas roles turinčių vartotojų poreikius, jiems pateikiant vartotojo veiklą atitinkančią sąsają.

Sistemos galimybės:

- Skirtingą naudotojų tipų funkcionalumą atitinkančios sąsajos;
- Buhalterinės informacijos sekimas ir valdymas;
- Automatizuota sąveika su e-bankininkystės sistemomis;
- Standartinių finansinių ataskaitų formavimas ir spausdinimas;
- Klientų, darbuotojų, sistemos vartotojų ir kitos informacijos duomenų bazėje sekimas ir valdymas;
- Organizacijos formų ir įvairių vartotojo suformuotų ataskaitų spausdinimas, bei eksportavimas į *.pdf formatą;

Sistema skirta trijų tipų vartotojams – **buhalteriui, sociologui/psichologui ir organizacijos vadovui**. Numatyta, kad organizacijoje yra ir kitokių darbuotojų kategorijų, tačiau jie SVDC apskaitos sistema nesinaudos (pvz. savanoriai ar praktikantai). Startavus programą prašoma prisijungti nurodant vartotojo vardą ir slaptažodį. Kiekvienas iš pagrindinių vartotojo sąsajos langų turi savo meniu ir aibę mygtukų.



13 pav. Klientų sąsajų navigacijos žemėlapis

3.7.1. Įdiegimo vadovas

Reikalavimai sistemai:

- Operacinė sistema – Windows 2000 arba vėlesnė arba Linux;
- Operatyvinės atminties dydis – rekomenduojama 256 Mb ir daugiau;
- Laisvos vietos kietajame diske (programai diegti) – 100Mb (papildoma vieta reikalinga diegti kitus programos veikimui reikalingus produktus);

Reikalavimai įdiegtai programinei įrangai:

- J2SE Runtime Environment 5.0
- MySQL Server 5.0
- Norint koreguoti programinę įrangą (platinamas ir šio AK produkto programos kodas) reikalinga Java 2 SDK, SE v1.4.2_04 arba vėlesnė versija, bet koks MySQL duomenų bazės AK administravimo įrankis (pvz. „*SQLyog*“) ir bet koks Java IDE (pvz. „*X-developer*“)

Prieš diegiant sistemą reikia patikrinti:

- Ar išpildyti minimalūs techniniais reikalavimai;
- Ar įdiegta reikalinga minimali programinė įranga;
- Ar surinkti pradiniai duomenys (darbinės stoties, kurioje yra organizacijos MySQL duomenų bazė, IP adresas).

Kai tinke yra keletas kompiuterių, SVDC organizacijos MySQL duomenų bazė veikia tik viename iš jų. Kituose kompiuteriuose, kliento programa jungiasi prie šios duomenų bazės, naudojant IP adresą, nurodomą kaip parametrą, diegiant programą.

Programos įdiegimas:

Paleisti “SVDC.bat” (atveriamas langas su adreso įrašymo laukeliu, patvirtinimo/atšaukimo mygtukų pora, bei pasiūlytų keliu iki programos katalogo). Šalia kelio iki programos katalogo laukelio yra mygtukas, skirtas atverti standartinį katalogo ir kelio iki jo pasirinkimo langą. Pagal nutylėjimą, sukuriamas programos “SVDC” katalogas yra “C” disko “Program Files” kataloge.

Įdiegus sistemą, automatiškai startuojama SVDC apskaitos programa, rodoma vadovo vartotojo sąsaja, kurioje vartotojas gali konfigūruoti kitų sistemos vartotojų duomenis (smulkesnė informacija apie tai pateikiama programos administravimo ir vadovo sąsajos skyreliuose).

Programa atitinkamai šalima iš sistemos, paleidus programa “SVDCuninstall.bat”.

3.7.2. Administravimo vadovas

Sistemoje integruotas „jGnash“ buhalterijos įrankis, kuriame galima nustatyti norimą vartotojo sąsajos kalbą.

Sistemos vartotojai administruojami prisijungus prie pačios sistemos. Šia teisę turi valdančiojo personalo rolę turintys vartotojai (vadovo vartotojo sąsajoje yra atitinkamas meniu punktas ir vartotojų duomenų langai). Pagal nutylėjimą vadovo prisijungimo prie sistemos vardas „SVDC“, slaptažodis „a1!“.

Valdyti MySQL duomenų bazę tinka bet koks MySQL administratorius (DB buvo sukurta naudojant SQLyog programėle). Sistemos diegimo metu buvo įdiegta duomenų bazė, pavadinimu SVDCDB.

3.8.Sistemos diegimas ir palaikymas

Sistema diegiama viešoje įstaigoje „Senamiesčio vaikų dienos centras“. Vyksta „SVDC individualios apskaitos sistemos“ programos bandomojo eksploatacija ir ruošiamasi projekto plėtrai. Vartotojų pastabos nuolat registruojamos – renkami plėtros reikalavimai ir klaidų raportai.

4. Tyrimo dalis

Nėra vieningos metodikos skirtos įvertinti AK kodo produktų kokybę. Kokybė vertinama specifiniais kiekvienam projektui būdais, dažniausiai bandant vertinimą pagrįsti įvairiomis metrikomis. AK aplinkoje naudojami eksperimentiniai ir individualūs kokybės užtikinimo metodai. Pats kokybės valdymo procesas smarkiai skiriasi, priklausomai nuo projekto dydžio ir aktualumo:

- didesniuose AK projektuose siekiama kurti ir formalizuoti naujus kokybės užtikrinimo metodus;
- smulkesniuose projektuose, kuriuose retai taikoma kokia nors formali kontrolė, kokybės reguliavimo sistema tampa pati AK aplinka, o ypač – AK ideologija.

4.1. Atviro kodo produktų klasifikavimas (kokybės vertinimui)

Kaip buvo paminėta, kokybės užtikrinimo (paremto kokybės vertinimu) metodai įvairiuose projektuose smarkiai skiriasi. Šiame tyrime AK produktai skiriami į porą grupių – tai dideli projektai ir maži bei vidutiniai AK projektai. Klasifikuojama atsižvelgiant į programinių modulių skaičių ir prie projekto dirbančių kūrėjų (savanorių) skaičių.

Charakteringi klasių atstovų pavyzdžiai:

- *„Mozilla“, „Alache“, „Linux“, “FireBird“* – tai stambios sistemos;
- įvairūs individualūs projektai, mažų programuotojų grupių projektai – tai smulkios arba vidutinės sistemos, pvz. *„WebCalender“, „Semagic“, „Kaffeine Player“, „JGnash“, „FlameRobin“*.

Tiriant šias grupes nagrinėjami magistriniame darbe realizuotai sistemai kurti panaudoti AK produktai – *„FireBird“*, *„JGnash“, „FlameRobin“*.

4.2. Atviro kodo produktų grupių atstovų informacija

Siekiant iliustruoti AK projektų kokybės vertinimą, buvo surinkta statistinė informacija apie charakteringus nagrinėjamų produktų klasių atstovus.

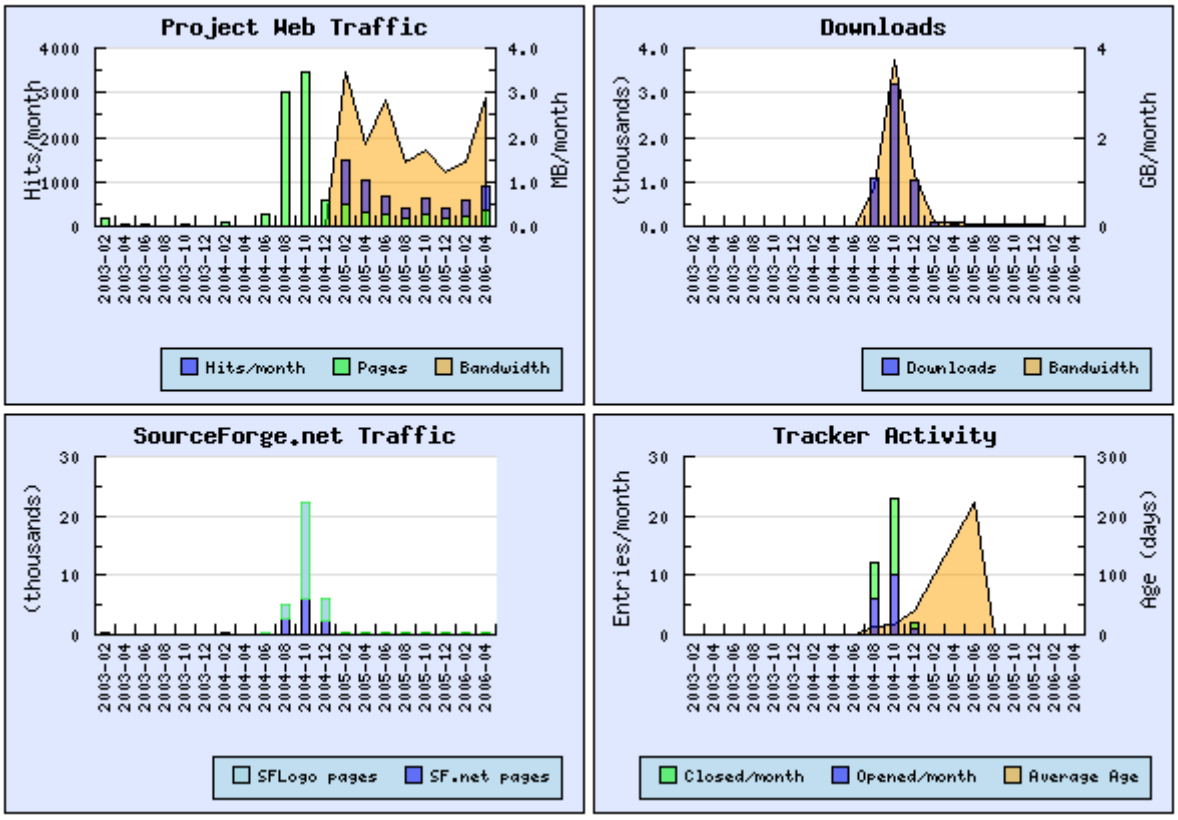
4.2.1. Smulkūs projektai

a) *„FlameRobin“*

Projekto statistika:

- Projekto administratoriai : mbabuskov, nandod

- Kūrėjai : 6
- Duomenų bazės aplinka : Firebird/InterBase
- Kūrimo statusas : 3 - Alpha
- Skirta : Developers, End Users/Desktop, System Administrators
- Licenzija : Mozilla Public License 1.0 (MPL)
- Operacinė sistema: All 32-bit MS Windows (95/98/NT/2000/XP), Linux, OS X
- Operacinė sistema: C++
- Tema : Front-Ends
- Išversta į kalbas : English
- Naudotojo sąsaja : Win32 (MS Windows), X Window System (X11)
- Užregistruota : 2003-02-08 17:36
- Veiklos aktyvumas (paskutinę savaitę) : 90.76
- Klaidos (**6 atviros / 74 viso**)
- Naujų savybių pageidavimai (**35 atviros / 72 viso**)
- Apytikslis prenumeratorių skaičius: 49 [14]

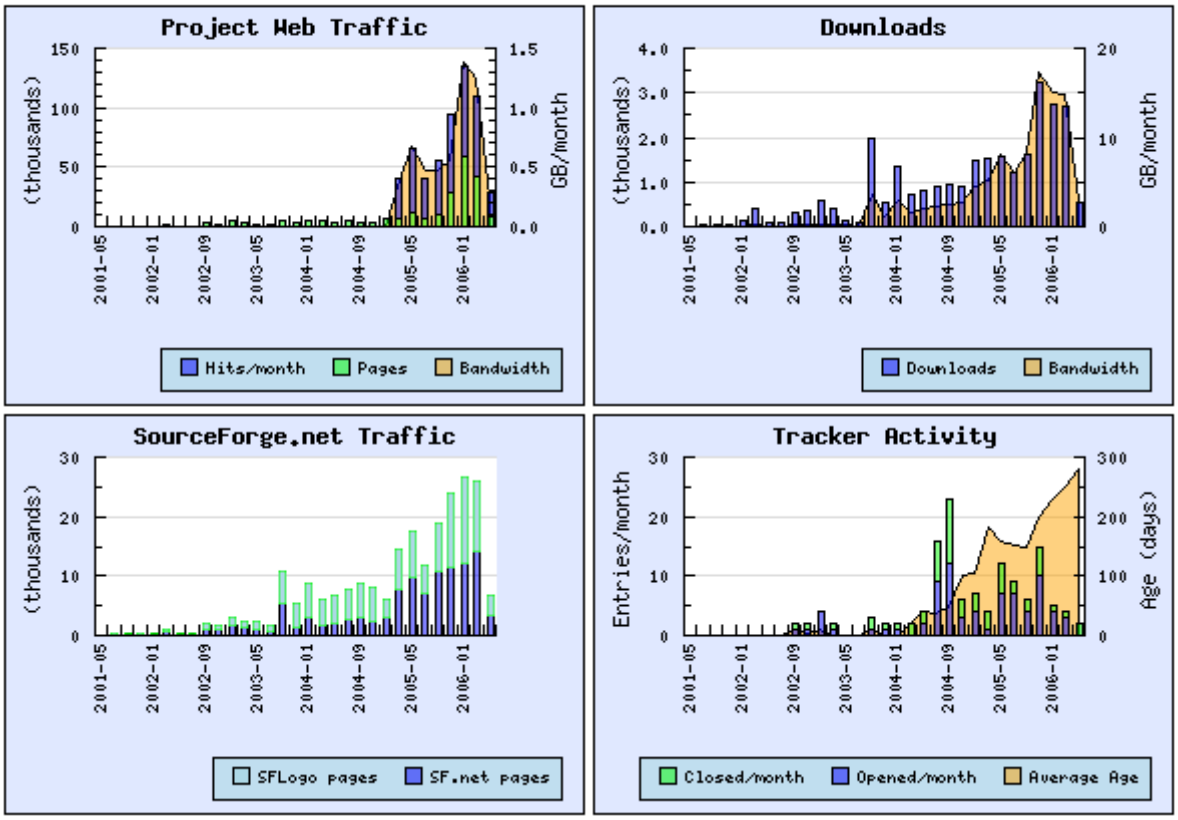


14 pav. „*FlameRobin*“ projekto statistika

b) *„jGnash“*

Projekto statistika:

- Projekto administratoriai: ccavanaugh
- Kūrėjai: 4
- Duomenų bazės aplinka: Proprietary file format
- Kūrimo statusas : 4 - Beta, 6 - Mature
- Skirta : Advanced End Users, Advanced End Users, Developers, End Users/Desktop, End Users/Desktop, Education
- Licenzija : GNU General Public License (GPL), GNU Library or Lesser General Public License (LGPL)
- Operacinė sistema: All 32-bit MS Windows (95/98/NT/2000/XP), All 32-bit MS Windows (95/98/NT/2000/XP), All POSIX (Linux/BSD/UNIX-like OSes), Linux, Linux, OS X, Solaris
- Operacinė sistema: C++, Delphi/Kylix, Java
- Tema : Computer Aided Instruction (CAI), Testing, Accounting, Investment
- Kalbos : English, French, German, Italian, Lithuanian, Portuguese, Russian
- Naudotojo sąsaja: Cocoa (MacOS X), Java Swing, Win32 (MS Windows), X Window System (X11)
- Projekto UNIX pavadinimas: jgnash
- Užregistruota : 2001-05-14 05:08
- Veiklos aktyvumas (paskutinę savaitę): 99.35
- Klaidų (**6 atviros / 75 viso**)
- Palaikymo prašymai (**0 atviri / 7 viso**)
- “Patches” (**4 atvirų / 21 viso**)
- Naujų savybių pageidavimai (**31 atvirų / 50 viso**)



15 pav. „*jGnash*“ projekto statistika

Visi statistikos duomenys gauti iš AK kodo projektų portalo „SourceForge“ [14].

4.2.2. Stambūs projektai

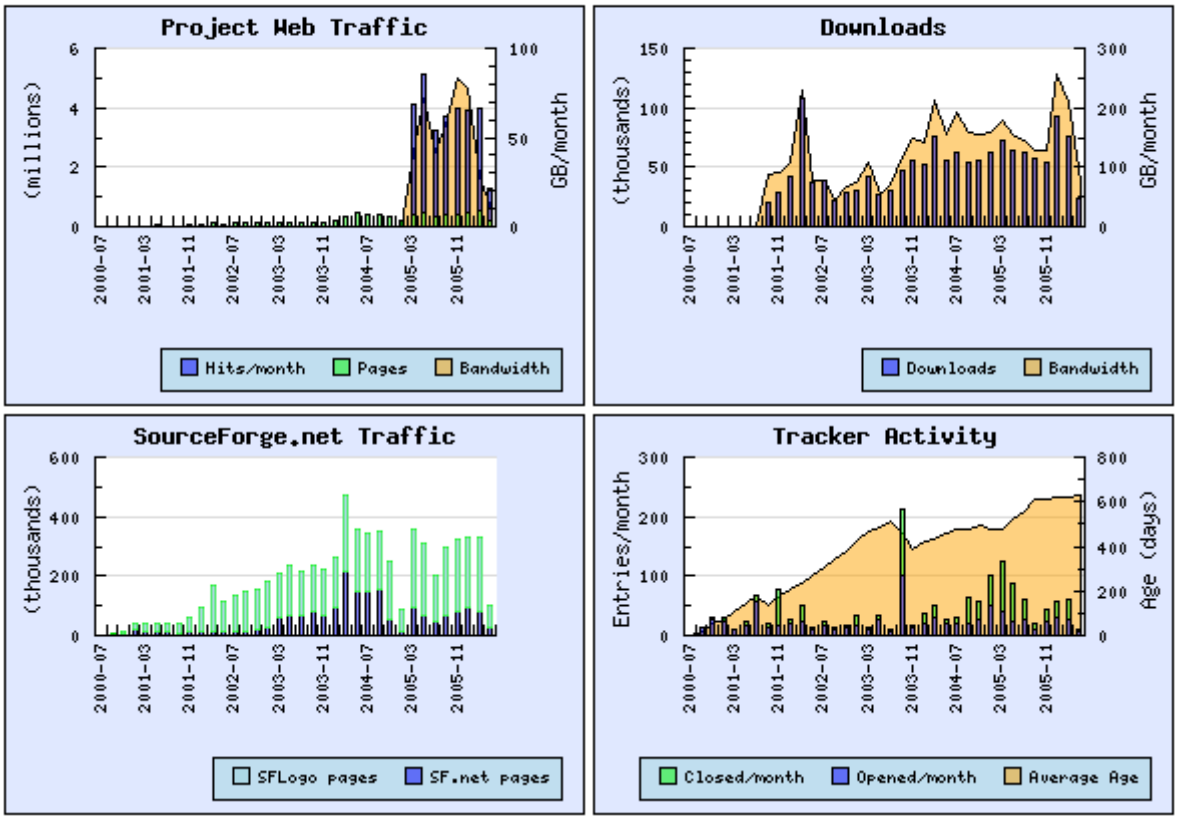
Anksčiau aprašyti AK produktai yra smulkūs, prie jų dirba mažiau nei dešimties asmenų grupelės. „*FireBird*“ projektas daug stambesnis. Jo kokybės valdymui neužtenka vien skatinančios kokybišką darbą AK ideologijos, kuri veikia savikontrolę kūrėjų grupelėje (pagarba, mokymas ir mokymasis, programuotojo ego).

Populiariems atviro kodo projektams būtinos papildomos kokybės užtikrinimo metodikos, garantuojančios vartotojus tenkinančią produkto kokybę.

a) **FireBird** 

Projekto statistika:

- Projekto administratoriai: awharrison, bellardo, dchri, dimitr, helebor, pcisar, seanleyne, skywalker
- Kūrėjai: 94
- Kūrimo statusas: 5 - Production/Stable
- Skirta: Developers, End Users/Desktop, System Administrators
- Licenzija : Mozilla Public License 1.0 (MPL)
- Operacinė sistema: All 32-bit MS Windows (95/98/NT/2000/XP), All POSIX (Linux/BSD/UNIX-like OSes)
- Programavimo kalbos: C, C#, C++, Java, Object Pascal
- Tema : Database Engines/Servers
- Naudotojo sąsaja: Non-interactive (Daemon)
- Projekto UNIX pavadinimas: firebird
- Užregistruota : 2000-07-30 19:52
- Veiklos aktyvumas (paskutinę savaitę): 99.96
- Klaidų (110 **atviros / 692 viso**)
- Palaikymo prašymai (**0 atviri / 7 viso**)
- "Patches" (2 **atvirų / 8 viso**)
- Naujų savybių pageidavimai (14 **atvirų / 203 viso**)
- Testai (**27 atvirų / 149 viso**)



16 pav. „*FireBird*“ projekto statistika

„*Firebird*“ projekte vykdoma speciali kokybės užtikrinimo veikla.

„*FireBird*“ kokybės užtikrinimas

Yra daug metodų užtikrinti programinės įrangos kokybei, bet nei vienas iš jų negali išspręsti visų kokybės problemų, nes kokybė nėra lengvai pasiekimas tikslas. „*Firebird*“ projekte vienu metu naudojama keletas metodų. Kai kurie iš metodų tiesiogiai susiję su kūrimo procesu, kiti metodai iškelti į atskirą, „FireBird“ kokybės užtikrinimo projektą, Trečia metodų grupė jungia pirmas dvi metodų grupes. Nors visi šie metodai suformuoja sudėtingą kokybės užtikrinimo procesą, kiekvienas metodas atskirtai yra lengvai suprantamas ir naudojamas.

„*Firebird*“ 2.0 testuotojų rinkimas

Testavimą atlieka savanoriai, todėl „*Firebird*“ kūrėjai nori kaip galima sutrumpinti su kokybės užtikrinimu susijusios veiklos trukmę, tuo nepakenkiant griežtiems galutinės versijos kokybės reikalavimams. Kokybės užtikrinimo veiklos rezultatai smarkiai priklauso nuo grįžtamojo ryšio su naudotojais.

Iki „*Firebird*“ 2.0 versijos, atgalinis ryšys buvo atsitiktinis ir visiškai priklausantis nuo vartotojų. Paprastai buvo išleidžiama versija (*angl.* „built“), kurį laiką laukiama atsiliepimų apie ją ir tada sprendžiamos registruotos problemos (kartu su kūrėjų bendruomenės rastomis klaidomis).

Jeigu po paskutinės versijos išleidimo nebūdavo randama didelių PĮ trūkumų, ši PĮ versija buvo išleidžiama kaip galutinė (*angl.* „release“). Tokio testavimo trūkumas – nežinoma kaip kruopščiai programa buvo išbandyta, funkcionalumo padengimo ir apkrovimo atžvilgiu. Apie kokybę galima tik spėlioti, kreipiant dėmesį į projekto parsisiuntimo, atsiliepimų skaičiaus metrikas, gandus ir kūrimo brandumo lygį (įvertinamas „kokybės indeksas“). Siekiant gerinti kokybę, galima tik ilginti testavimo laiką, t.y. padidinti laiko periodus tarp versijų išleidimo.

Situacijai pagerinti, pradedant 2.0 versija, buvo inicijuotas valdomo testavimo projektas (*angl.* „field-test program“), skirtas papildyti kokybės užtikrinimo veiklą. Valdomo testavimo tikslas – surinkti tikslią informaciją apie atliktus testus (kaip versija buvo testuota ir kokie rezultatai buvo gauti). Tokiu būdu išaiškinami kokybės trūkumai ir galima pakoreguoti kokybės užtikrinimui skirtų resursų paskirstymą taip, kad būtų greičiau pasiekta tenkinanti vartotojo poreikius kokybė. Savanoriui užtenka prisiregistruoti ir aprašyti kokiu būdu jis norėtų testuoti sistemą (vertinami visi testavimo būdai, artimi realiam sistemos naudojimui). Savanoriui siunčiami pranešimai apie testuotinas versijas ir laukiama testų rezultatų.

Savanorių įsipareigojimas negarantuotas (savanoris gali nenorėti atlikti aktualios versijos testavimo arba visai praranda susidomėjimą projektu), todėl savanoriškas darbas apdovanojamas. Aktyviausiems ir daliai atsitiktinai parenkamų savanorių paruošiamos smulkios dovanėlės, pvz., marškinėliai su projekto simbolika.

Valdomas testavimo projektas atviras visiems, norintiems prisijungti bet kuriuo versijos kūrimo ciklo metu.

„*Firebird*“ testavimo metodologija ir įrankiai

- **Peržiūra (*angl.* „peer review“)**

Kodo peržiūra – tai dažniausiai minima atviro kodo kūrimo modelio savybė. Informacija apie „*Firebird*“ projekto kodo pakeitimus teikiama kūrėjams (specialios elektroninio pašto grupės, naujienų grupės), visos rastos problemos aptariamos.

„*Firebird*“ fondas paskyrė specialią dotaciją šį darbą kuruojančiam žmogui. Šiuo metu peržiūrimi tik branduolio kodo pakeitimai.

- **Kodo tikrinimas (*angl.* „code audit“)**

Kodo tikrinimas – tai kita peržiūros forma. Peržiūra susijusi tik su kodo pasikeitimais ir mažomis, kuriamomis kodo dalimis. Kodo tikrinimas apima visą programinį projekto kodą

Šis kokybės užtikrinimo metodas nėra naudojamas reguliariai (išskyrus kritinės svarbos sistemose), kadangi reikalauja daug darbo valandų ir specifinių žinių. „*Firebird*“ projekte yra labai daug paveldėto kodo (35 MB), su kuriuo kūrėjai nėra susipažinę – kodas nagrinėjamas

bandant suvokti jo funkcionalumą, o ne tikrinti jo veikimo korektiškumą. Tokiu būdu buvo rasta tik keletas klaidų.

Viso kodo peržiūra – didžiulis darbas. Ieškoma savanorių, kurie norėtų imtis šio darbo. Kiekvienos pilnai peržiūrėtos išeities kodo laikmenos antraštėje rašoma peržiūros data ir šį darbą atlikusio asmens informacija.

- **Testavimo atvejų kūrimas**

Kūrėjai dažnai naudoja savo sukurtas aplikacijas testuoti bazinį produkto funkcionalumą. Šios aplikacijos ne visada kuriamos. Jau sukurtos testavimo programos gali būti pakartotinai panaudotos kokybės užtikrinimo grupės darbams.

Šuo metu testavimo atvejai naudojami tik „*JDBC type 4 driver*" ir ".*NET Data Provider*" kūrėjų grupėse (testai pagrįsti „*UnitTest*" karkasu).

Dėl vidinės „*Firebird"* struktūros, vienetų testavimas „*Firebird*" kodui yra nepraktiškas. Vienetų testai turi verifikuoti teisingą posistemių veikimą, pvz., atminties valdymo, paslaugų programų funkcijas. Ieškoma savanorių, norinčių kurti vienetų testus.

- **Automatizuotas juodos dėžės testavimas**

Šio testavimo paskirtis –verifikuoti galutinio produkto veikimo teisingumą:

1. Teisingi įėjimo duomenys duoda teisingus rezultatus;
2. Neteisingi įėjimo duomenys duoda teisingus klaidos aptikimo rezultatus;
3. Produktas teisingai reaguoja į ribų sąlygas;
4. Produktas teisingai reaguoja į nenumatytas sąlygas;
5. Nuoseklumas (rezultatai neprieštaringi, vykdant veiksmus bet kokia tvarka);
6. Naudojamumas.

4 ir 6 punktų testai neautomatizuojami ir paprastai priklauso nuo rankinio darbo.

Visos rastos klaidos sekamos specialiomis priemonėmis. Sekimo sistema yra svarbi kokybės užtikrinimo ciklo dalis – be jos negali sėkmingai veikti jokia kokybės užtikrinimo procedūra.

„*Firebird*" paveldėjo automatizuotą testavimo sistemą „*TCS*". Šia sistema sudėtinga naudotis ir ją išplėsti, todėl ji buvo pakeista „*QMTest*".

- **Klaidų sekimo sistema (*angl.* „tracker")**

Naudojama keletas klaidų sekimo įrankių (atskiriant įvairias klaidų rūšis). Savanorių primygtinai prašoma naudoti tinkamą sekimo sistemą, siekiant išvengti neteisingų/perteklinių įrašų apie klaidas:

  - „Bug tracker" sistema naudojama sekti stabilios sistemos versijos klaidas;

- „Field-test issue tracker“ sistema naudojama sekti kuriamos sistemos versijos klaidas (alfa/beta);
- „Feature request tracker“ sistema naudojama sekti pageidaujamas naujas sistemos savybes.

Šiuo metu visos klaidų sekimo sistemos patalpintos „SourceForge“ svetainėje, bet tiriama galimybė pereiti prie JIRA įrankio [7].

4.3.Siūlomas AK produktų kokybės vertinimo modelis

Analitinėje dalyje buvo aptartos AK projektų metrikos ir metrikų įverčių naudojimas produkto kokybei nustatyti. Vidutiniam vartotojui yra aktualus mažų ir vidutinių projektų kokybės vertinimas - naudojamus savo darbams AK produktus vartotojams dažniausiai tenka pasirinkti savarankiškai. Didelių AK projektų kokybė, palyginus, nėra tokiu probleminiu klausimu, kadangi stambiuose projektuose organizuotas kokybės valdymo procesas, užtikrinantis pakankamą galutinio produkto kokybę.

Remiantis atliktos kokybės vertinimo metodų analizės rezultatais, pavieniai metrikų įverčiai negali būti naudojami kaip patikimas kokybės matas. Nors svetainėse pateikiamų PĮ metrikų panaudojimas kokybei įvertinti būtų labai greitas ir paprastas būdas, tačiau, remiantis analizės metu surinkta informacija, jis nėra patikimas. Kad statistiniai metrikų duomenys taptų naudingi, reikia įvertinti visą aibę metrikų ir kitų faktorių, pvz., statistikos rinkimo laikotarpį, eksponentinės patikimumo augimo kreivės formą konkrečiam projektui, savanorių aktyvumą bei susidomėjimą projektu ir t.t.. Siūlomas teorinis neformalus preliminarios AK produktų kokybės vertinimo modelis pagrįstas produktų statistikos metrikų vertinimu - integruojama keletas egzistuojančių metodų. Visiškai įsitikinti pakankama AK produkto kokybe galima tik praktiškai jį naudojant. Tačiau empirinis kokybės tikrinimas užtrunka per ilgai.

Siūloma, renkantis kokybišką AK produktą, kreipti dėmesį į tokius įverčius:

- Projekto gyvavimo laikas;

  Vertinant projektą pagal statistinius duomenis, labai svarbu šių duomenų pilnavertiškumas. Nustatant projekto vystymo tendencijas, turi būti surinkti vertinamojo projekto viso gyvavimo ciklo duomenys, pageidautina, kad šis periodas nebūtų trumpesnis nei vieneri metai. Gyvavimo ciklo pradžioje, naujo projekto kokybė visada bus nepakankamai gera.

- AK bendruomenės susidomėjimas projektu

  Labai svarbu atkeipti dėmesį, ar vertinamas projektas populiarus tarp bendruomenės narių – kūrėjų ir savanorių, ar pakankamai asmenų juo domisi. Šis įvertis gali būti nustatomas stebint projekto internetinės svetainės duomenų srautus (fiksuojama kiek kartų puslapis buvo naršomas, siunčiamų ir gautų duomenų kiekis). Jei susidomėjimas projektu jo gyvavimo metu mažas, arba šiuo laikotarpiu yra sumažėjęs, tai dažniausiai indikuoja, kad projektas neaktualus ir jo kokybė prastesnė.

- Projekto parsiuntimo (*angl*. „download“) dažnumas;

  Šis įvertis skirtas nustatyti projekto priimtinamumą vartotojams ir nustatomas stebint projekto parsisiuntimo dažnį. Paprastai, dažnas projekto parsiuntimas indikuoja kokybišką projekto vartotojo sąsają ir gerą naudojamumą. Kuo didesnė produkto naudotojų aibė, tuo nuodugniau bus peržiūrėtas projekto funkcionalumas ir rasta daugiau klaidų.

- Klaidų sekimo sistemos veikla;

  Įvertis skirtas stebėti ar efektyviai vyksta klaidų taisymas ir programos tobulinimas. Jis nustatomas stebint klaidų aptikimo ir taisymo dažnumą, vidutinį klaidų pataisymo greitį. Projektas bus kokybiškesnis, kai į aptiktas klaidas reaguojama greitai. Augant projekto patikimumui, stabilizuojasi klaidų aptikimo dažnis (aptiktoms klaidoms galioja eksponentinis „Weibull“ pasiskirstymas). Kai klaidų aptikimo dažnis sumažėja – projektas yra pakankamai aukšto patikimumo. Šis įvertis glaudžiai susijęs su projekto veikos aktyvumu. Ryšys tarp patikimumo ir klaidų aptikimo dažnumo galioja tik aktyviai vystomiems produktams. Kitu atveju, kai projekto kūrėjai dirba vangiai, retas klaidų aptikimas gali indikuoti tik tai, kad projektas tapo nepopuliariu.

- Pataisytų ir aptiktų klaidų santykis

  Šis santykis leidžia įvertinti aktualią projekto būseną ir, priklausomai nuo klaidų sekimo sistemos veiklos, prognozuoti projekto kokybę.

- Pageidautinų ir realizuotų naujų savybių santykis

  Šis įvertis taip pat leidžia vertinti aktualią sistemos būseną. Kai projektas jau brandžioje kūrimo stadijoje, t.y., klaidų aptikimo dažnis stabilizavęsis, artėjantis prie nulio santykis tarp pageidautinų ir realizuotų projekto

savybių nurodo aktyvią projekto palaikymo veiklą, santykis artėjantis prie vieneto – funkcionalumo trūkumus.

Naudojant siūlomą integruotą metodą, mažų ir vidutinio dydžio projektų kokybės įvertinimas ir palyginimas užtrunka daugiau laiko, nei paprastas kokybės numatymas pagal pavienes AKPĮ produkto statistikos metrikas, kadangi atsižvelgiama į daugiau kokybę įtakojančių faktorių ir AKPĮ kokybė įvertinama tiksliau. Modelio panaudojimas įmanomas tik tais atvejais, kai pasiekiama statistinė projekto informacija pakankamai ilgame laiko intervale.

5. Eksperimentinė dalis

Naudojant atviro kodo įrankius kurti programinę įrangą, arba AK produktus modifikuojant pagal savus poreikius, svarbu iš aibės egzistuojančių tinkamų AK programų pasirinkti kokybiškiausią. Tai tiesiogiai įtakoja kuriamos PĮ kokybę.

Tyrimo dalyje pateiktą metodiką derėtų taikyti tik AK produktams, kurių statistiniai duomenys (AK kūrimo aplinkoje) buvo renkami ilgą laikotarpį (pageidautina ne trumpiau, nei metus). Kitu atveju kokybės vertinimas bus nepatikimas. Dėl šios priežasties magistrinio darbo programinės įrangos kokybei vertinti pasiūlytas modelis netaikomas.

Atliekant eksperimentus, pasiūlytas preliminarios AK produktų kokybės vertinimo modelis pritaikytas porai mažos apimties AK įrankių, naudotų realizuojant magistrinio darbo projekto programinę įrangą – „*jGnash*“ ir „*Flamerobin*“ (pilki laukeliai skirti pažymėti pranašesnį, įverčio atžvilgiu, produktą).

5.1 lentelė. Įrankių kokybės palyginimas

Įverčio pavadinimas	„*jGnash*“	„*Flamerobin*“
Projekto gyvavimo laikas	Projekto gyvavimo ciklas ilgesnis, „jGnash“ buvo įregistruotas „SourceForge“ svetainėje nuo 2001 metų vidurio.	Projekto gyvavimo ciklas trumpesnis – šis projektas gyvuoja apie tris metus. „*Flamebird*“ buvo įregistruotas „SourceForge“ svetainėje 2003 metų pradžioje.
AK bendruomenės susidomėjimas projektu	Susidomėjimas šiuo projektu vidutiniškai 30 kartų didesnis nei susidomėjimas „*Flamerobin*“ įrankiu. Dabar bendruomenės susidomėjimas šiuo projektu jau slopsta – išleista stabili versija.	Palyginus nepopuliarus projektas. Po 2004 pabaigos susidomėjimas juo vis mažėja.
Projekto parsiuntimo dažnumas	Programos kopijų parsisiuntimo dažnis augo iki 2006 metų pradžios. Projektas priimtinas vartotojui.	Projekto parsisiuntimo pikas buvo 2004 metų pradžioje. Nuo to laiko parsiunčiamų programos kopijų skaičius

		smarkiai sumažėjo. Projektas tapo nevartotinu.
Klaidų sekimo sistemos veikla	Klaidų sekimo veikla daug aktyvesnė nei „*Flamerobin*“ projekte, klaidos dažniau aptinkamos ir pataisomos. Klaidų skaičius stabilizavęsis, todėl galima daryti išvadą, kad produktas yra pakankamai patikimas.	Klaidų sekimo sistema buvo aktyvi tik 2004 metų viduryje, vėliau susidomėjiams šiuo projektu išblėso, o klaidų taisymo laikas išaugo. Projektas prastai palaikomas.
Pataisytų ir aptiktų klaidų santykis	0,1	0,1
Pageidautinų ir realizuotų naujų savybių santykis	0 Projekto palaikymo darbai aktyvūs.	0,5

Remiantis eksperimento rezultatais, „*Flamebobin*“ DB tvarkyklės kokybė prastesnė, nei „*jGnash*“ apskaitos įrankio. Praktiškai pritaikius abi programas, preliminarus kokybės vertinimas pasiteisino, o „*Flamerobin*“ įrankis buvo pakeistas kitu, kokybiškesniu („*SQLyog*“).

Siekiant patobulinti šį kokybės vertinimo modelį ir papildyti jį detalesniais punktais, reiktų atlikti išsamesnį tarpusavyje panašių atviro kodo produktų (pagal mastą ir funkcionalumą) palyginimą.

6. Išvados

1. Atlikta literatūros šaltinių analizė rodo, kad atviro kodo aplinkoje sukurti projektai yra populiarūs, jie dažnai naudojami ir individualioms reikmėms, ir įmonėse. AK produktų populiarumas ir sėkminga konkurencija su komercine PĮ yra vienas akivaizdžiausių pakankamos AK produktų kokybės požymių.
2. Išanalizuota aktuali bendroji kokybės situacija atviro kodo aplinkoje sukurtiems produktams. Ištirti naudojami atviro kodo produktų kokybės vertinimo būdai, bei kokybės užtikrinimo metodikos. Aptarti įvairių vertinimo metrikų ir kokybės valdymo metodikų privalumai ir trūkumai, galimybes pritaikyti įvairių tipų projektams.
3. Realizavus individualios apskaitos sistemą, naudojant atviro kodo projektavimo, kodavimo įrankius, modifikuojant kitus atviro kodo produktus, buvo įvykdyti ir dokumentuoti būtini komercinės programinės įrangos kūrimo etapai, sukuriant reikalavimų, architektūros, detalios architektūros specifikacijas ir vartotojo vadovą.
4. Dideliuose atviro PĮ projektuose paprastai veikia individuali kokybės užtikrinimo sistema, garantuojanti pakankamai gerą galutinio produkto kokybę. Kokybės užtikrinimo sistemos, kuriamos projekto bendruomenės narių, labai skiriasi ir priklauso nuo projekto specifikos. Mažuose ir vidutiniuose atviro kodo projektuose, atskiros kokybės užtikrinimo valdymo veiklos nevykdoma.
5. Vartotojui yra aktualus mažų ir vidutinių projektų kokybės vertinimas, nes naudojamus savo darbams AK produktus jis dažniausiai renkasi savarankiškai.
6. Pasiūlytas neformalus mažo ir vidutinio dydžio, atviro kodo produktų kokybės vertinimo modelis. Išskirtos AKPĮ savybės, į kurias reiktų kreipti dėmesį, pasirenkant AK produktą. Dėmesys kreipiamas į šešis aspektus - projekto gyvavimo laiką, AK bendruomenės susidomėjimą projektu, projekto parsiuntimo dažnumą, klaidų sekimo sistemos veiklą, pataisytų ir aptiktų klaidų santykį, pageidautinų ir realizuotų naujų savybių santykį.
7. Pasiūlytas atviro kodo projektų kokybės vertinimo modelis pritaikytas magistrinio darbo programinę įrangą realizuoti naudotiems AK įrankiams. Empiriškai patikrinus šių įrankių kokybę, preliminarus jų kokybės vertinimas, taikant teorinį modelį, pasiteisino.

8. Norint modelį patobulinti, reiktų atlikti praktinius eksperimentus su didesne AK produktų aibe, stebint kiekvieno iš įverčių poveikio kokybei svarbą.

7. LITERATŪRA

[1] **Atviras kodas Lietuvai** [interaktyvus]. 2002, [žiūrėta 2006.04.24]. Prieiga per internetą www.akl.lt

[2] **Bezroukov, N**. Open Source Software Development as a Special Type of Academic Research (Critique of Vulgar Raymondism). Iš *First Monday* [interaktyvus], Nr.10. 1999, spalis [žiūrėta 2006.05.17]. Prieiga per internetą http://www.firstmonday.org/issues/issue4_10/bezroukov/index.html

[3] **Bezroukov, N**. A Second Look at the Cathedral and the Bazaar. Iš *First Monday* [interaktyvus], Nr.12. 1999, gruodis. Prieiga per internetą http://www.firstmonday.org/issues/issue4_12/bezroukov/index.html

[4] **Crowston, K., Scozzi, B**. Exploring the Strengths and Limits if Open Source Software Engineering Processes: A Research Agenda. 2002, [žiūrėta 2006.04.23]. Prieiga per internetą http://opensource.ucc.ie/icse2002/CrowstonScozzi.pdf

[5] **Dravis Group LLC, The**. Open Source Software: Case Sudies Examining its Use. 2003, kovas [žiūrėta 2006.04.23]. Prieiga per internetą http://dravisgroup.com/HTML/reports/OSSCaseStudies.pdf

[6] **Evers, S**. An Introduction To Open Source Software Development. *Magistro darbas.* Berlyno technologijos universitetas. 2000, rugpjūtis [žiūrėta 2006.05.16]. Prieiga per internetą http://user.cs.tu-berlin.de/~tron/opensource/opensource.pdf

[7] **„Firebird".** QA Sub-project. 2006, [žiūrėta 2006.05.20] . Prieiga per internetą http://www.firebirdsql.org/index.php?op=devel&sub=qa

[8] **McLaughlin, D. S.** Opening the code: software excellence as a function of its development environment. *Magistro darbas.* Džordžtauno universiteto menų ir tikslųjų mokslų fakultetas. 2001 [žiūrėta 2005.11.16] Prieiga per internetą http://opensource.mit.edu/papers/mclaughlin.pdf

[9] **Michlmayr, M.** Managing volunteer activity in open source projects, *UNENIX Annual Technical Conference, 2004* [žiūrėta 2005.11.20]. Prieiga per internetą http://opensource.mit.edu/papers/michlmayr-mia.pdf

[10] **Paulavičiūtė I.** Atviro kodo produktų kokybės vertinimas ir kokybės užtikrinimo strategijos. *Informacinės technologijos 2006: konferencijos pranešimų medžiaga.* Kaunas, 2006, p. 70-74.

[11] **Raja, U., Barry, E.** Investigating Quality in Large-Scale Open Source Software, iš *The ACM Digital library* [interaktyvus]. *2003* [žiūrėta 2005.12.15]. Prieiga per internetą http://portal.acm.org/dl.cfm

[12] **Rasch, C. M**. A Brief History of Free/Open Source Software Movement, iš *Open Knowledge* [interaktyvus]. 2000, gruodis [žiūrėta 2006.05.16]. Prieiga per internetą http://www.openknowledge.org/writing/open-source/scb/brief-open-source-history.html

[13] **Samoladas, I., Stamelos, I.** Assessing Free/Open Source Software Quality, iš *Free/Open Source Research Community* [interaktyvus]. 2004, vasaris [žiūrėta 2005.11.15]. Prieiga per internetą http://opensource.mit.edu/papers/samoladasstamelos.pdf

[14] **SoureForge** (atviro kodo projektų portalas) [interaktyvus]. 2006 [žiūrėta 2006.04.21]. Prieiga per internetą http://sourceforge.net/

[15] **Stewart, K. J., Gosain, S.** The impact of ideology on effectiveness in open source software development teams, iš *Free/Open Source Research Community* [interaktyvus]. 2004, vasaris [žiūrėta 2005.11.15]. Prieiga per internetą http://opensource.mit.edu/papers/stewartgosain2.pdf

[16] **Stoltz, M**. The Case for Government Promotion of Open Source Software. Recommendations for Government Action. 1999, [žiūrėta 2006.04.23]. Prieiga per internetą http://www.netaction.org/opensrc/oss-report.pdf

[17] **Van Wendel de Joode, R**. Understanding open source communities: An organizational perspective. Daktaro disertacijos darbas. Delfto technologijos universitetas, 2005. Prieiga per internetą http://repository.tudelft.nl/consumption/idcplg?IdcService=GET_FILE&RevisionSelectionMethod=latestReleased&dDocName=312582

[18] **Weber, K.** Social Aspects of Non-Proprietary Software. Iš *IJIE International Journal of Information Ethics,* Nr. 2 .2004, lapkritis [žiūrėta 2006.05.16]. Prieiga per internetą http://container.zkm.de/ijie/ijie/no002/ijie_002_27_weber.pdf

[19] **Zhao, L., Elbaum, S.** A Survey On Quality Related Activities in Open Source**,** *ACM SIGSOFT, Software Engineering Notes*, 2000, Nr. 25, p. 54-57.

[20] Zhou, Y., Davis, J. Open source software reliability model: an empirical approach, iš *The ACM Digital library* [interaktyvus]. 2004 [žiūrėta 2005.12.01]. Prieiga per internetą http://portal.acm.org/dl.cfm

8. Terminų ir santrumpų žodynas

8.1 lentelė. Terminai ir santrumpos

Sutrumpinimas/Terminas	Paaiškinimas
GUI	Grafinė vartotojo sąsaja („Graphical User Interface“)
BVL (GLP)	Bendroji viešoji licencija („General Public License“)
MPL	„Mozilla Public License“
AK (OS)	Atviras kodas („Open Source“)
SQL	„Structured Query Languages“
QA	Kokybės užtikrinimas („Quality Assurance“)

9. Priedai

9.1. 1 PRIEDAS Konferencijos „Konferencijos „Informacinės technologijos 2006“ pranešimo medžiaga

ATVIRO KODO PRODUKTŲ KOKYBĖS VERTINIMAS IR KOKYBĖS UŽTIKRINIMO STRATEGIJOS

Indrė Paulavičiūtė

Kauno technologijos universitetas, Informatikos fakultetas, Programų inžinerijos katedra

Studentų g.50, Kaunas

Straipsnyje nagrinėjami atviro kodo (AK) produktams taikytini kokybės vertinimo kriterijai ir kokybės užtikrinimo problemos. AK produktų naudojimas personalinėje ir verslo aplinkoje įrodo pakankamą jų kokybės lygį, nors jiems negalioja tradiciniai kokybės užtikrinimo metodai. AK produktams nėra sukurto brandaus kokybės vertinimo modelio – šiems produktams bandoma taikyti eksperimentinius vertinimo modelius (pvz. projektų statistikos kriterijai). AK produktų kokybė taip pat valdoma netradiciniais metodais (pvz. savanorių kontrolė).

1. Įvadas

Atviro kodo aplinka, kurioje negalioja komercinės programinės įrangos kūrimo principai, taip pat yra potencialiai tinkama kokybiškų produktų kūrimui. Formalizuotą kūrimo procesą ir centralizuotus valdymo sprendimus (formalūs vartotojo reikalavimai ir projektavimo specifikacijos), kurių pagalba tradiciškai valdoma projekto kokybė, pakeičia kitos produkto kokybiškumą skatinančios priemonės. Kokybę pozityviai įtakoja atviro kodo (AK) ideologija bei kūrimo specifika - naudojama modulinė programinės įrangos (PĮ) struktūra, adaptyvus sistemos modelis ir dažni atnaujinimai, PĮ mutacija, aktyvus AK bendruomenės indėlis (resursų ir atsakomosios reakcijos atžvilgiu). Dėl griežtų formalių taisyklių nebuvimo, tradiciniai kokybės užtikrinimo, valdymo ir įvertinimo metodai atviro kodo projektams negali būti taikomi, o pats kokybės užtikrinimo procesas yra probleminis. Neegzistuoja bendro brandaus AK kokybės vertinimo modelio ar bendrų kokybės valdymo metodų – bandoma kurti ir taikyti eksperimentinius vertinimo modelius.

2. Atviro kodo PĮ kūrimo aplinka

AK aplinka skatina programinės įrangos kūrėjų konkuravimą ir inovacijas, palaiko individualų kūrybiškumą. Kokybiškesnė PĮ kuriama dėl:

- masiško iteratyvaus programinio kodo peržiūrėjimo (paprastai tą atlieka patyrę programuotojai);
- modulinio dizaino (kiekvienas modulis gali būti modifikuojamas atskirai, taip sumažinant riziką sukurti naujų klaidų);
- Adaptyvaus sistemos modelio (dažnas naujų versijų pristatymas, PĮ evoliucija, mutacija, aktyvios kūrėjų/naudotojų bendruomenės indėlis, vartotojai kartu yra ir kūrėjai);

AK aplinkoje paprastai nėra formalių vartotojo reikalavimų ir projektavimo specifikacijų, bet pasiekiama pakankamai gera kokybė, kurią ypač skatina atviro kodo ideologija. Ji pozityviai veikia ne tik AK kūrėjų komandų darbo efektyvumą, bet ir kognityvų pasitikėjimą, bei bendravimo kokybę grupėje

2.1 Normos

Atviro kodo ideologijos normos įpareigoja kūrėjus neskelti projekto į keletą („no forking“), kodo pakeitimus platinti tik tinkamais kanalais, gerbti kodo autorystę ir nenutrinti autorystės žymių.

2.2 Įsitikinimai

Atviro kodo bendruomenės nariai laikosi įsitikimų, kad AK produktai yra geresnės kokybės, nei komerciniai jų analogai, nemokama programinė įranga ir laisvai pasiekiama informacija naudingesnė, praktinis darbas svarbesnis už teoriją, statusas įgyjamas pripažinus AK bendruomenei

2.3 Vertybės

Atviro kodo bendruomenės vertybės - dalinimasis informacija ir pagalba kolegoms, techninės žinios, savišvieta, bendradarbiavimas ir reputacijos bendruomenėje susikūrimas, dalyvaujant projektuose.

Dalis AK bendruomenės įsitikinimų mažina dalyvių indėlį į komandinį darbą ir atitinkamai neigiamai paveikia produkto užbaigimą.

3. AK produktų kokybės vertinimas

AK programinės įrangos kūrimo modelio atsiradimas gali pakeisti požiūrį į kokybės užtikrinimą. Nėra sukurto brandaus atviro kodo produktų kokybės vertinimo modelio ir kokybės užtikrinimo procedūrų (dauguma komercinės programinės įrangos kokybės įverčių negali būti taikomi).

Suklestėjus AK judėjimui, aktualiausi tapo tokie klausimai:

- kokiais būdais galima tikrinti AK produktų kokybę;
- kokie faktoriai įtakoja AK produktų kokybę;

AK projektus galima įvertinti, taikant ISO/IEC 9126 standartą, pagal šešis pagrindinius atributus:

- Funkcionalumas

labai naudingi atviri AK produktų standartai ir produktų sąveika;

produktai tinkami ir plačiai taikomi, pvz. AK operacinės sistemos ar programavimo kalbos;

AK projektai retai detaliai specifikuojami, todėl mažai specifinio funkcionalumo AK produktų;

- Patikimumas

  klaidos operatyviai randamos ir taisomos (didelė bendruomenė), pvz. 50% „Apache“ tarnybinės stoties klaidų pataisoma per mažiau nei 24 valandas nuo klaidos pastebėjimo datos;

  organizuotas komunikavimas tarp suinteresuotų projekto dalyvių;

  programinio kodo pasiekiamumas daro AK produktus patikimesniais negu komerciniais analogai;

- Naudojamumas

  AK produktai skirti kūrėjams, o ne galutiniam vartotojui;

  naudojamumo klaidos sunkiai aprašomos ir taisomos;

  dažnai AK įrangą sudėtinga įdiegti;

- Palaikomumas

  programinės įrangos kokybė prastėja, kai nėra jos palaikymo, o AK projektuose negarantuojamas nei palaikymas, nei pageidaujamų funkcijų realizavimas;

  rizikuojama, kad sumažėjus savanorių, prisidedančių prie projekto, palaikymas bus nutrauktas;

  nedaug dokumentacijos, todėl palaikyti produktą sudėtinga;

- Perkeliamumas

  ši savybė - prioritetas, kuriant atviro kodo PĮ;

- Efektyvumas

  nėra patikimų šio atributo vertinimo strategijų ir priklauso nuo vartotojo (pvz. Windows ir Linux);

AK produkto kokybės reikalavimai, vystant produktus, keičiasi. Pirmiausia turi būti tenkinami funkciniai reikalavimai – vertinant, produktas lyginamas su komercinių produktų funkcionalumu arba vartotojų poreikiais, jei analogiško komercinio produkto nėra. Vėliau svarbiu tampa patikimumas. Vartotojas gali rinkis senesnę, patikimesnę versiją, turinčią mažiau funkcionalumo. Ėmus produktą vartoti, svarbus palaikomumas (korekcinis, adaptyvus, prevencinis), o plečiantis sistemai aktualiausias pakartotinis panaudojamumas (viena vertingiausių AK savybių).

4. AK produktų kokybės valdymas

4.1 Tenkinančio kokybės lygio pasiekimo numatymas

Atviro kodo ir komercinių produktų programinėje įrangoje aptinkamų klaidų skaičius stabilizuojasi, pasiekus reikiamą patikimumo lygį. AK produktams, išskyrus tokius didelius projektus kaip „Gnome“, „Linux“, „Mozilla“, „Alache“ ir pan., vertinti kokybę ir prognozuoti stabilumą bandoma naudojant įvairias statistines projektų metrikas (dydis, kūrėjų skaičius, veiklos aktyvumas, periodiškai randamų klaidų skaičius, puslapio peržiūros ir PĮ parsiuntimo dažnumas). Paprastai AK produktų svetainėse ar dideliuose portaluose (pvz. http://sourceforge.net/) pateikiama daugybė metrikų, pagal kurias vartotojui spręsti apie produkto kokybę sudėtinga. Remiantis tyrimais, AK projektų puslapių peržiūros ir programų parsiuntimo dažnio, bei klaidų radimo koreliacija silpna ir ši kokybės vertinimo metodika nepasiteisina – peržiūros ir parsiuntimo statistika indikuoja ne AK produkto kokybę, o priimtinumą vartotojams. Be to, daugumą pastabėtų klaidų aprašų pateikia ne vartotojai, o maža kūrėjų bendruomenė.

4.2 Programinių paketų kontrolė ir savanorių veiklos valdymas

Labai svarbi AK PĮ kokybės valdymo strategija – savanorių kontrolė. Problemas sukelia savanorių patikimumas, jų darbų koordinavimas, savanoriško darbo rizika ir pasekmės projekto kokybei, skirtingas požiūris į savanorių pareigas. Iš projekto pasitraukus savanoriui, atsakingam už kritinės svarbos paketą, gali nukentėti projekto kokybė, arba projektas gali būti visai sustabdytas.

AK bendruomenę dalina dvi skirtingos filosofijos apie savanorių pareigas. Pirmoji filosofija nurodo, kad savanoriškame darbe neturėtų būti jokių įsipareigojimų, o programuotojas gali daryti ką panorėjęs. Pagal antrąją filosofiją, sutikdamas dalyvauti projekte, savanoris įgauna atsakomybę už sako projektinę dalį, o nusprendęs palikti projektą, jis turėtų pasirūpinti, kad jo darbai būtų saugiai perduoti.

Paprasti savanorių aktyvumas nėra pastovus. Periodiškai savanoris gali apleisti savo pareigas (pvz. prie AK projekto dirbantis studentas turi skirti visą savo laiką mokslams sesijos metu). Savanorių veiklos stebėjimas ir valdymas gali smarkiai įtakoti kuriamos PĮ kokybę. Ypač tai aktualu dideliuose projektuose, pvz. „Debian“, kuriame naudojamas tokia savanorių darbo kontrolės metodika:

- stengiamasi, kad vieną paketą prižiūrėtų keletas programuotojų, o jei paketu nesidomima – šis paketas į eilinę distribuciją neįtraukiamas;
- paketų korekcijų kontrolė – pakeitimus gali atsiųsti tik tam tiki, identifikuoti projekto dalyviai (naudojami GPG – „GNU Privacy Key“ ar PGP- „Pretty Good Privacy“ raktai);
- nauji projekto dalyviai kruopščiai atrenkami (socialinės integracijos, filosofijos ir techninių sugebėjimų, įsipareigojimų tikrinimas, ilgas atrankos procesas);
- savanorių aktyvumo stebėjimas (jei pastebima, kad vartotojas pora savaičių neatnaujino savo programinio kodo, jam siunčiama užklausa, po dar poros savaičių, jei vartotojas neatsiliepia, jam siunčiamas perspėjimas apie paketo perdavimą, nesulaukus atsakymo, atsakomybė už paketą perduodama kitiems projekto dalyviams);
- nuolat kontroliuojama paketų būsena, pakeitimai ir kaupiamos paketų metrikos (kritinių netaisomų klaidų skaičius, atitikimas versijai);

- naudojami automatizuoti veiklos sekimo įrankiai.

4.3 AK projektų testavimas

Vidutiniams ir mažiems projektams paprastai nekuriama testavimo planų, testavimo fazėje praleidžiama nedaug projekto laiko, pakartotino kodo peržiūros kartų skaičius proporcingai didesnis didesniuose ir populiaresniuose projektuose, testavimo įrankiai naudojami mažesnei projektų daliai, o didžiausias klaidų skaičius randamas AK projektų vartotojo sąsajoje.

AK projektuose testavimui paprastai skiriama mažiau dėmesio, nei kuriant komercinę programinę įrangą. Pagal tyrimų duomenimis:

- daugiau nei 80% AK projektų neturi jokių testavimo planų;
- vidutiniškai testavimo stadijoje praleidžiama 40% laiko;
- mažiems projektams, kodas pakartotinai peržiūrimas tik 1.3 karto, didesniems – 7.5;
- 39.6% projektų naudojam testavimo įrankius;
- daugiau nei 50% nenaudoja įrankių patikrinti padengimui;
- daugiausia klaidų randama projektų vartotojo sąsajoje – 28.6%.

5. Išvados

Įvertinant visus AK produktų privalumus, kokybė nėra svarbiausias iš jų. Kuriamos ir tobulinamos AK produktų kokybės valdymo strategijos yra individualios, o kokybės vertinimas, naudojantis projektų metrikomis – netikslus.

Literatūros sąrašas

[1] **Katherine J. Stewart, Sanjay Gosain.** The impact of ideology on effectiveness in open source software development teams, *Free/Open Source Research Community*, 2005, prieiga internete http://opensource.mit.edu/papers/stewartgosain2.pdf, 2005

[2] **Martin Michlmayr**. Managing volunteer activity in open source projects, 2004 UNENIX Annual Technical Conference, prieiga internete http://opensource.mit.edu/papers/michlmayr-mia.pdf

[3] **Ioannis Samoladas, Ioannis Stamelos**. Assessing Free/Open Source Software Quality, *Free/Open Source Research Community*, preiga internete http://opensource.mit.edu/papers/samoladasstamelos.pdf

[4] **David S. McLaughlin**. Opening the code: software excellence as a function of its development environment, Thesis, 2001, prieiga internete http://opensource.mit.edu/papers/mclaughlin.pdf

[5] **Uzma Raja, Evelyn Barry**. Investigating Quality in Large-Scale Open Source Software, *The ACM Digital librayr* http://portal.acm.org/dl.cfm, Texas, 2003

[6] **Ying ZHOU, Joseph DAVIS**. Open source software reliability model: an empirical approach, *The ACM Digital librayr* http://portal.acm.org/dl.cfm, *Sydney, Australia,*, 2004

[7] **Luyin Zhao, Sebastian Elbaum.** A Survey On Quality Related Activities in *Open **Source,*** ACM SIGSOFT, Software Engineering Notes, vol 25 no 3, 2000 pp 54-57.

Quality Assessment and Quality Assurance for Open Source Products

In this paper, the criteria for quality assessment and problems of open source (OS) product quality assurance are discussed. Broad use of open source products both in personal and business environment prove sufficient quality, despite of absence of traditional quality assurance methods for these products. There is no mature model for OS product quality evaluation - only experimental models are applied (for example, OS project statistics criteria evaluation). OS product quality is also managed in unconventional methods, related to OS specific software development procedures (for example, volunteer control).