

VILNIAUS UNIVERSITETAS
MATEMATIKOS IR INFORMATIKOS FAKULTETAS
INFORMATIKOS KATEDRA

Baigiamasis bakalauro darbas

Funkcinių taškų analizė
(Function Point Analysis)

Atliko: IV kurso (kompiuterių mokslo) I grupės studentas

Artūras Jasevičius

(parašas)

Darbo vadovas:

doc. dr. Sigitas Dapkūnas

(parašas)

Recenzentas:

doc. dr. Antanas Mitašiūnas

(parašas)

Vilnius
2016

Turinys

Terminų sutrumpinimų sąrašas	3
Įvadas	4
1. IFPUG metodas	5
1.1. Istorija	5
1.2. FPA Privalumai ir trūkumai	5
2. Funkcinių taškų skaičiavimas	7
2.1. Funkcinių taškų skaičiavimo tipo, ribų ir tikslo nustatymas	7
2.2. Duomenų elementų tipų atpažinimas	8
2.2.1. Duomenų perdavimo DET	9
2.2.2. RET	10
2.2.3. DET grafinėje vartotojo aplinkoje	10
2.3. Duomenų funkcijų ir jų įtakos UFP skaičiavimas	11
2.3.1. Vidinės loginės bylos	11
2.3.2. Išorinės sąsajos bylos	11
2.3.3. Funkcinių taškų skaičiavimas byloms	12
2.4. Transakcinių funkcijų ir jų įtakos UFP skaičiavimas	12
2.4.1. Išoriniai įėjimai	12
2.4.2. Išoriniai išėjimai	13
2.4.3. Išorinės užklausos	13
2.5. Projekto vystymo funkciniai taškai	14
2.6. Reikšmės koregavimo faktorius	14
2.6.1. Pagrindinės sistemos charakteristikos	15
2.6.2. VAF skaičiavimas	15
2.6.3. VAF kritika	15
2.7. Koreguotų funkcinių taškų skaičiavimas	16
2.7.1. Nekoreguoti funkciniai taškai	17
2.7.2. Funkcinių taškų skaičius	17
2.7.3. Projekto vystymo funkcinių taškų skaičiavimas	17
2.7.4. Programinės įrangos funkcinių taškų skaičiavimas	17
2.7.5. Projekto keitimo funkcinių taškų skaičiavimas	18
3. Automatinis funkcinių taškų skaičiavimas	19
4. Praktinis pavyzdys: SMDB funkcinių taškų skaičiavimas	20
4.1. SMDB funkcinių taškų skaičiavimo tipas	20
4.2. SMDB ribos ir funkcinių taškų skaičiavimo tikslas	20
4.3. SMDB nekoreguotų funkcinių taškų skaičiavimas	20
4.3.1. SMDB duomenų funkcijos	20
4.3.1.1. SMDB vidinės loginės bylos	20
4.3.1.2. SMDB išorinės sąsajos bylos	21
4.3.2. SMDB transakcinės funkcijos	21
4.3.2.1. SMDB išoriniai įėjimai	22
4.3.2.2. SMDB išoriniai išėjimai	23
4.3.2.3. SMDB išorinės užklausos	23
4.4. SMDB nekoreguoti funkciniai taškai	24
4.5. SMDB reikšmės koregavimo faktorius	24
4.6. SMDB funkcinių taškų skaičius	26
4.7. SMDB funkcinių taškų skaičiavimo analizė	26

Išvados	28
Conclusions	29
Literatūra	30
Priedas Nr.1	
Priedas Nr.2	

Terminų sutrumpinimų sąrašas

DET	<i>Data Element Type</i>	Duomenų elemento tipas
EI	<i>External Input</i>	Išorinis įėjimas
EIF	<i>External Interface File</i>	Išorinė sąsajos byla
EO	<i>External Output</i>	Išorinis išėjimas
EQ	<i>External Inquiry</i>	Išorinė užklausa
FP	<i>Function Points</i>	Funkciniai taškai
FPA	<i>Function Point Analysis</i>	Funkcinių taškų analizė
FPC	<i>Function Point Count</i>	Funkcinių taškų skaičius
FTR	<i>File Types Referenced</i>	Kreipimosi bylų tipai
GSC	<i>General System Characteristics</i>	Pagrindinės sistemos charakteristikos
GUI	<i>Graphical User Interface</i>	Grafinė vartotojo sąsaja
IFPUG	<i>International Function Point Users Group</i>	Tarptautinė funkcinių taškų naudotojų grupė
ILF	<i>Internal Logical File</i>	Vidinė loginė byla
LOC	<i>Lines Of Code</i>	Kodo eilučių skaičius
RET	<i>Record Element Type</i>	Įrašo elemento tipas
SMDB		Santuokos metrikų duomenų bazė
SNAP	<i>Software Non-functional Assessment Process</i>	Nefunkcinis programinės įrangos vertinimo procesas
UFP	<i>Unadjusted Function Points</i>	Nekoreguoti funkciniai taškai
UML	<i>Unified Modelling Language</i>	Vieninga modeliavimo kalba
VAF	<i>Value Adjustment Factor</i>	Reikšmės koregavimo faktorius
XML	<i>Extensible Markup Language</i>	Išplėstinė dokumentų aprašų kalba

Įvadas

Kasdienybėje žmonės vis dažniau susiduria su naujosiomis technologijomis. Daugelis kasdien naudojami kompiuteriais, išmaniaisiais telefonais, bankomatais, kasos aparatais, automobilių stovėjimo terminalais ar mobiliosiomis programėlėmis. Kad šie įrenginiai veiktų, jiems reikia programinės įrangos, kuri laikui bėgant tampa vis sudėtingesnė ir jos kūrimui sugaištama vis daugiau laiko.

Programinę įrangą kuriančios kompanijos stengiasi tiksliai nustatyti kuriamos programinės įrangos dydį ir jos kūrimui reikalingus laiko ir žmogiškuosius išteklius. Nežinant kiek laiko užtruks programavimo procesas, neįmanoma efektyviai organizuoti darbo ir planuoti darbuotojų laiko. Praktikoje vis dar nėra naudojamas joks vieningas programinės įrangos dydžio nustatymo metodas. Įmonės bando sukurti savo vidinius metodus, pagal kuriuos nustato programos kūrimui reikalingas sąnaudas.

Jau Romos imperijos laikais sudėtingoms problemoms spręsti buvo remiamasi „skaldyk ir valdyk“¹ principu, puikiai tinkančiu ir kuriant sudėtingą programinę įrangą. Pačiame programos kūrimo pradžioje suskaidžius ją į nedideles ir lengvai suprantamas dalis, tampa daug lengviau įvertinti programos sudėtingumą, nustatyti reikalingą funkcionalumą ir programavimui reikalingą laiką. Būtent šiuo principu remiasi vis populiarijanti programinės įrangos dydį funkciniais taškais skaičiuojanti metodika — funkcinių taškų analizė (FPA).

FPA yra sistemos komponentų klasifikavimo technika, kai sistema yra suskaidoma į mažesnes dalis, kad jas būtų galima geriau suprasti, analizuoti ir įvertinti. Naudojant šią techniką sistema yra skaidoma į penkis komponentus ir pagrindines sistemos charakteristikas. Pirmosios trys klasės analizuoja programinės įrangos darbą su bylomis, o kitos dvi — su įvairiais duomenimis. Skaičiavimo tikslumui užtikrinti naudojamos pagrindinės sistemos charakteristikos, apibūdinančios pagrindinius sistemos parametrus ir jų įtaką programinei įrangai [Jon13; Lon02].

Šio darbo tikslas yra ištirti Tarptautinės funkcinių taškų naudotojų grupės (IFPUG) vis tobulinamą funkcinių taškų analizės metodą. FPA stipriųjų ir silpnųjų savybių analizė bei aptarti kai kurių autorių pasiūlymai padės geriau suprasti skaičiavimo techniką ir subtilybes, o darbe atliktas funkcinių taškų skaičiavimas leis praktiškai pamatyti visą skaičiavimo procesą. IFPUG metodas buvo pasirinktas dėl jo dažno naudojimo funkcinių taškų skaičiavime. Norint pažvelgti į FPA iš kuo įvairesnių perspektyvų, buvo remtasi ne tik IFPUG leidžiamais leidiniais, bet ir kitų mokslininkų bei programinės įrangos kūrėjų publikacijomis.

Darbe trumpai apžvelgta FPA istorija, detaliau išnagrinėti pagrindiniai penki FPA elementai, pagrindinės sistemos charakteristikos ir IFPUG funkcinių taškų skaičiavimo metodas. Taip pat išanalizuota funkcinių taškų skaičiavimo automatizavimo problema. Antroje darbo pusėje praktiškai skaičiuojant veikiančios ir naudojamos programos funkcinius taškus ištirtas visas funkcinių taškų skaičiavimo procesas, sistemos charakteristikų vertinimas bei išanalizuoti gauti rezultatai.

¹lot. *Divide et impera!*

1. IFPUG metodas

1.1. Istorija

Vienas pirmųjų programinės įrangos dydžio nustatymo metodų buvo kodo eilučių skaičiavimas (LOC). Naudojantis šiuo metodu programa yra vertinama skaičiuojant jos kodo eilučių skaičių. Yra kelios šio metodo variacijos, tačiau jos visos turi trūkumų. Pirmiausia, nėra aiškaus apibrėžimo, kas yra programos kodo eilutė (fizinė eilutė, vykdymo eilutė, loginė eilutė ir pan.). Taip pat yra sudėtinga numatyti programos kodo eilučių skaičių prieš pradedant programuoti [BTP12].

Kita LOC metodo problema yra vertinimo priklausomybė nuo programavimo kalbos: neįmanoma tiesiogiai palyginti dviejų skirtinga programavimo kalba įgyvendintų projektų. LOC metodas nevertindavo ir programinės įrangos projektavimo bei testavimo sąnaudų, buvo priklausomas nuo programuotojo patirties [LJ90].

1970—ųjų pabaigoje Allan J. Albrecht pasiūlė naują FPA metodą, kuris turėjo išspręsti problemas, kylančias vertinant programinę įrangą LOC metodu. Tai buvo pirmasis bandymas įvertinti programinę įrangą atsižvelgiant į jos funkcionalumą, o ne į patį programavimą. FPA tikslas yra apskaičiuoti funkcinę programinės įrangos apimtį. Šis metodas yra nepriklausomas nuo naudojamos technologijos ar programavimo kalbos ir yra lengvai pritaikomas įvairaus tipo programinės įrangos vertinimui.

1986 m. buvo suformuota tarptautinė funkcinų taškų naudotojų grupė (IFPUG), kuri tobulina ir populiarina Albrecht pasiūlytą FPA metodą: du kartus metuose leidžia žurnalą *MetricViews*, nuolat atnauja IFPUG skaičiavimo vadovą (dabartinė versija: 4.3.1), plėtoja SNAP² metodą.

Albrecht pasiūlytą FPA metodą kritikavo ir tobulino kiti autoriai. Didžiojoje Britanijoje išsivystė Mk II metodas. Kanadoje naudojamas COSMIC metodas, skirtas realaus laiko sistemoms vertinti. Suomiškas FiSMA ir olandiškas NESMA metodai beveik niekuo nesiskiria nuo dabartinio IFPUG [DAC06; Fet99; YS15; Lon02].

Apie funkcinų taškų analizės svarbą programinės įrangos kūrimo procese galime spręsti ir iš to, jog yra sukurta vienuolika ISO standartų, susijusių su programinės įrangos matavimu funkciniais taškais. Nuo 2002 metų pradėti sertifikuoti įvairūs funkcinų taškų skaičiavimo metodai (COSMIC, Mk II, NESMA, FiSMA), o 2009 metais patvirtintas ISO/IEC 20926:2009 standartas IFPUG skaičiavimo metodui [ISO09].

1.2. FPA Privalumai ir trūkumai

Kaip ir kiekvienas programinės įrangos vertinimo metodas, FPA turi privalumų ir trūkumų. Pirmiausia, FPA yra metodas, leidžiantis ganėtinai anksti įvertinti programinės įrangos kainą ir išteklius, reikalingus jos kūrimui. Kuo anksčiau nustatoma programinės įrangos apimtis, tuo tiksliau

²*Software Non-functional Assessment Process* – IFPUG sukurtas nefunkcinio programinės įrangos dydžio matavimo procesas, papildantis funkcinų taškų analizės metodą [IFP15].

galima planuoti darbą. Funkciniai taškai taip pat gali būti naudojami vertinant komandos produktyvumą ir darbo kokybę.

Žinant programinės įrangos dydį kiekvienu projekto gyvavimo momentu yra daug lengviau suvaldyti projekto apimties pokyčius. Pasikeitus projekto reikalavimų specifikacijai, galima ganėtinai tiksliai įvertinti tų pokyčių įgyvendinimui reikalingus kaštus.

Kadangi FPA yra nepriklausoma nuo programavimo kalbos, kiekviena įmonė gali nustatyti vieno funkcinio taško įkainį palygindama jau įvykusių projektų kainas ir sąnaudas. Taip funkciniai taškai gali būti naudojami pagrindžiant vienokią ar kitokią programinės įrangos kainą pateikiant projekto apimtį įvertintą funkciniais taškais.

Norint kuo tiksliau įvertinti kiekvieną projektą, IFPUG metode yra naudojamas reikšmės koregavimo faktorius, kuris priklauso nuo įvairių kuriamos programinės įrangos ypatybių ir jos veikimo aplinkos.

Dar vienas šio metodo privalumas yra tai, jog funkcinis taškus skaičiuoti gali žmogus, neturintis programavimo patirties. Tačiau norint užtikrinti korektišką ir tikslų vertinimą, skaičiavimu turėtų užsiimti IFPUG mokymus baigęs darbuotojas, turintis atitinkamą sertifikatą ir besinaudojantis naujausia IFPUG vartotojo vadovo versija [Sub14; SV15].

FPA metodas turi ir trūkumų. Silpnoji šio metodo pusė yra jo subjektyvumas ir galimi netikslumai. Funkcinius taškus skaičiuojantis žmogus skaičiavimo procese turi daugybę kartų subjektyviai vertinti funkcinis taškus. Būtent dėl šios priežasties yra svarbu, jog tai atliktų sertifikuotas darbuotojas [Jon13].

Kaip jau minėta anksčiau, funkcinį taškų skaičiavimas gali būti atliekamas ganėtinai anksti, tačiau to ne visada pakanka. Norint kuo tiksliau įvertinti projektą, reikia žinoti sistemos reikalavimus ir vartotojo poreikius, o šie dokumentai dažniausiai sugeneruojami jau po sutarties pasirašymo. Taigi iki tol sunku tiksliai nustatyti programinės įrangos dydį funkciniais taškais.

Funkcinių taškų skaičiavimas yra palyginti lėtas ir brangus procesas, todėl beveik niekada neatliekamas sistemoms, kuriose funkcinį taškų skaičius gali būti didesnis nei 10000 funkcinį taškų. Funkciniai taškai taip pat priklauso nuo bendrų savokų supratimo kompanijos viduje, todėl vienos kompanijos įvertinimas funkciniais taškais gali skirtis nuo kitos kompanijos įvertinimo [Jon13; Kit97].

Taigi, nors FPA yra nemažai laiko reikalaujantis, greitai įsisavinamas ir subjektyvus procesas, jo dėka galima ganėtinai tiksliai įvertinti programinę įrangos dydį, suvaldyti jos pokyčius kūrimo procese ir nustatyti bei pagrįsti galutinį įvertinimą pinigais.

2. Funkcinių taškų skaičiavimas

Kaip jau buvo minėta anksčiau, norint gauti tikslius rezultatus, FPA turi būti atliekama sertifikuoto specialisto, besinaudojančio naujausia funkcinių taškų skaičiavimo vadovo versija. Kitas svarbus faktorius, nuo kurio priklauso skaičiavimo tikslumas yra programinės įrangos dokumentacijos išmanymas. Žmogaus, puikiai išmanančio programą konsultacija taip pat gali labai pagerinti skaičiavimo kokybę.

Pagrindinis funkcinių taškų skaičiavimo tikslas yra nustatyti sistemos koreguotų funkcinių taškų skaičių, pagal kurį galima įvertinti jos kūrimui ar tobulinimui reikalingus išteklius. Funkcinių taškų skaičiavimas turėtų būti kiekvieno projekto sudedamoji dalis.

Funcinių taškų skaičiavimo procesą sudaro šie žingsniai [Lon04]:

1. Funkcinių taškų skaičiavimo tipo nustatymas.
2. FPA ribų ir tikslo nustatymas.
3. Duomenų funkcijų ir jų įtakos nekoreguotiems funkciniais taškams (UFP) skaičiavimas.
4. Transakcinių funkcijų ir jų įtakos UFP skaičiavimas.
5. Reikšmės koregavimo faktoriaus nustatymas.
6. Koreguotų funkcinių taškų skaičiavimas.

2.1. Funkcinių taškų skaičiavimo tipo, ribų ir tikslo nustatymas

Funkciniai taškai gali būti skaičiuojami tiek programinei įrangai, tiek IT projektui. funkcinių taškų skaičiavimas skirstomas į tris tipus:

- naujo projekto FPA: skaičiuojami naujos programinės įrangos funkciniai taškai, įskaitant jos diegimą klientui;
- projekto vystymo FPA: skaičiuojami jau egzistuojančios programinės įrangos pakeitimo funkciniai taškai;
- programinės įrangos FPA: skaičiuojami jau veikiančios programinės įrangos funkciniai taškai.

Kadangi kompiuterių sistemos bendrauja su kitomis kompiuterių sistemomis ir/arba žmonėmis, reikia apibrėžti kiekvienos sistemos, kurioje bus atliekami matavimai, ribas. Nustatinėjant FPA ribas reikėtų remtis ne techniniu aprašymu, bet žiūrėti į sistemą iš vartotojo pusės.

Sistemos ribas galima nustatyti pačioje programinės įrangos gyvavimo pradžioje. Jei programa yra pakaitinė (pakeičia jau esamą programą), jos ribos turėtų būti tokios pat, kaip ir prieš tai buvusios programos.

Internetinių sistemų ribų nustatymas yra panašus į tradicinį, taigi ribos nustatomos ne tik vartotojo aplinkai ar langų grupei, bet visai sistemai. Dažniausiai internetinės sistemos yra jau egzistuojančios programinės įrangos galimybių praplėtimas.

Klientinių programų ribos turi būti nustatomos ir klientui, ir serveriui, nes nei klientas, nei serveris negali pilnai funkcionuoti atskirai.

Norint nustatyti tikslias sistemos ribas, pirmiausia reikia tiksliai išsiaiškinti FPA tikslą, išanalizuoti, kaip ir kokios programos keičiasi duomenimis ir identifikuoti verslo taisykles, pagal kurias veikia programa. Vėliau skaičiavimo eigoje, labiau įsigilinus į programinę įrangą ir jos funkcijas, FPA ribos turi būti peržiūrimos iš naujo.

Prieš skaičiuojant funkcinis taškus ir nustatinėjant funkcinių taškų skaičiavimo ribas, vertėtų susipažinti su tam tikra projekto dokumentacija (specifikacijų dokumentais, UML diagramomis, vartotojų dokumentacija, duomenų perdavimo diagramomis) ir net pabendrauti su pačiais vartotojais [Lon04; ZZW⁺09].

2.2. Duomenų elementų tipų atpažinimas

Funkcinių taškų analizėje visi komponentai yra priskiriami vienam iš trijų tipų (žr. 1 lentelėje):

- duomenų elemento tipas (DET);
- kreipimosi bylos tipas (FTR);
- įrašo elemento tipas (RET).

Duomenų elemento tipas yra unikalus vartotojo atpažįstamas, nerekursyvus, nepasikartojantis, dinamiškas elementas, nuskaitomas iš bylos arba sukuriamas iš duomenų elementų, esančių kreipimosi bylose. Jis pat gali sužadinti darbinės funkcijas arba būti joms perduodamas kaip papildoma informacija. Jei DET naudojamas rekursyviai, kreipiamas dėmesys tik į pirmą panaudojimą.

Kreipimosi bylos tipas yra darbinės funkcijos nurodytas bylos tipas (vidinė loginė byla arba išorinės sąsajos byla).

Įrašo elemento tipas naudojamas duomenų elementų pogrupio (vidinė loginė byla ar išorinė sąsajos byla) atpažinimui. Norint lengviau atpažinti bylas, reikia atkreipti dėmesį į loginį duomenų grupavimą.

1 lentelė. Komponentų tipai [Lon04]

Komponentas	DET	FTR	RET
Išoriniai įėjimai (EI)		✓	✓
Išoriniai išėjimai (EO)		✓	✓
Išorinės užklausos (EQ)		✓	✓
Išorinės sąsajos bylos (EIF)	✓		✓
Vidinės loginės bylos (ILF)	✓		✓

Tolesniame skaičiavimo procese pagal tam tikrus kriterijus bus įvertintas kiekvieno komponento sudėtingumas (*žemas, vidutinis, aukštas*). Pačioje programinės įrangos kūrimo pradžioje gali būti sudėtinga nustatyti tikslias RET ir DET reikšmes. Kadangi jie yra esminiai programos

vertinimo komponentai, juos galima vertinti keliais būdais: įvertinti visas darbinės ir duomenų funkcijas kaip *vidutines* arba remtis panašių programų komponentų vertinimais [Lon02; Lon04].

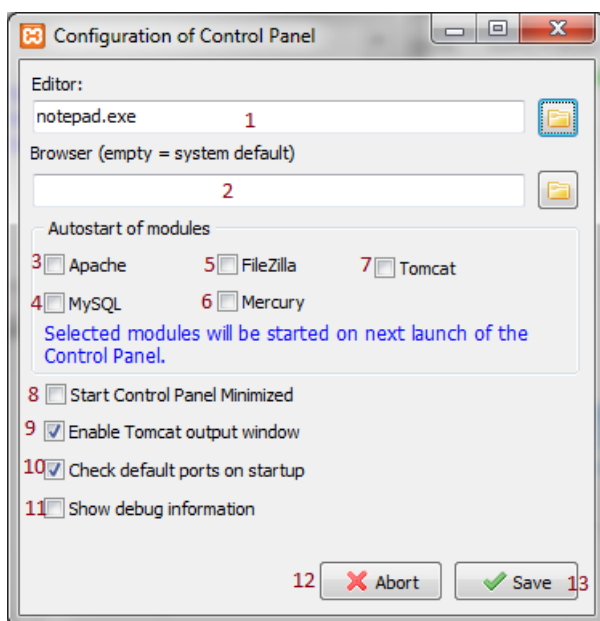
2.2.1. Duomenų perdavimo DET

Duomenys sistemoje arba į/iš jos gali būti perduodami įvairiais būdais [Lon02; Lon04]:

- Išoriniai įėjimai (EI): duomenų įvedimo laukai, klaidų pranešimai, suskaičiuotos reikšmės, mygtukai.
- Išoriniai išėjimai (EO): ataskaitos duomenų laukai, programos veikimo metu nustatytos reikšmės, klaidų pranešimai ir stulpelių antraštės, kurios yra skaitomos iš vidinių loginių bylų.
- Išorinių užklausų (EQ) įėjimai: laukeliai, pagal kuriuos ieškoma duomenų, pelės paspaudimai, ekrane vaizduojami laukeliai su duomenimis ar informacija.

1 paveiksle pateiktas programos langas su 23 DET. Kai vienas elementas atlieka kelias funkcijas, jis yra skaičiuojamas kaip keli DET:

- Laukeliai pažymėti numeriais nuo 1 iki 11 vienu metu yra ir EI ir EO, nes juose galima ne tik įvesti reikšmes, bet ir matyti sistemoje jau anksčiau nustatytas reikšmes.
- Mygtukas *Save* (13 laukelis) yra EI.
- Mygtukas *Abort* (12 laukelis) neskaičiuojamas kaip DET, nes jis neatlieka jokio veiksmo su duomenimis.

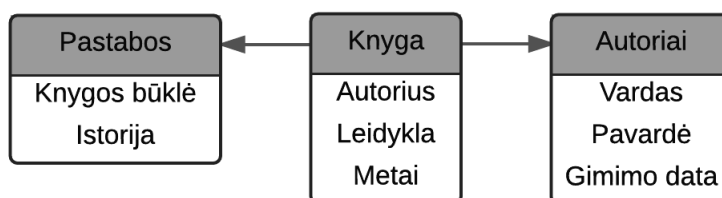


1 pav. XAMPP programos nustatymų langas

2.2.2. RET

Įrašo elemento tipas yra viena iš sudėtingiausių sąvokų funkcinių taškų analizėje. Dauguma įrašo elemento tipų yra priklausomi nuo tėvo-vaiko (*parent-child*) ryšių. Taigi, vaiko informacija yra tėvo informacijos poaibis, o tėvo-vaiko ryšys yra vienas su daugeliu ryšys. Kiekviena vaiko informacijos dalis yra tiesiogiai susieta su lauku tėvo byloje [Lon02; Lon04].

2 paveiksle nubraižyta labai supaprastinta bibliotekos knygų saugyklos diagrama. Joje matomi 8 DET elementai (*knygos būklė, istorija, autorius, leidykla, metai, vardas, pavardė, gimimo metai*) ir tuo pat metu 3 RET elementai, sudaryti iš DET elementų grupių (*pastabos, knyga, autoriai*). Jei šie duomenys yra saugomi programos viduje, tai jie tuo pat metu yra vidinės loginės bylos (ILF), o jei už programos ribų — išorinės sąsajos bylos (EIF).



2 pav. Bibliotekos knygų saugykla

2.2.3. DET grafinėje vartotojo aplinkoje (GUI)

Pagal griežtą IFPUG praktinio skaičiavimo vadovą duomenų elementas yra vartotojo atpažįstamas, nerekursyvus laukelis. Deja, tai nepadeda vertinant programas su GUI. Programose, turinčiose GUI, duomenų elementas yra informacija, laikoma vidinėje loginėje byloje arba naudojama duomenų perdavimui iššaukti:

- radijo mygtukai;
- piktogramos;
- pasirinkimo laukeliai;
- garsai;
- komandiniai mygtukai;
- nuotraukos;
- grafiniai vaizdai;
- pranešimai.

Kadangi IFPUG skaičiavimo vadovas nenurodo, kaip reikia skaičiuoti funkcinius taškus realaus laiko ir integruotose sistemose, tai jose taikomi kai kurie tradiciniai funkcinių taškų skaičiavimo būdai. Pavyzdžiui automobilio kompiuterinėje sistemoje duomenų elementais gali būti laikomi diagnostinės bylos laukai, tokie kaip diagnostikos laikas, techninės įrangos būklė diagnostikos metu, temperatūra, įtampa ir pan. Realios laiko sistemos gali neturėti jokios „tradicinės vartotojo sąsajos“. Tai yra, signalas realios laiko sistemai gali būti jos pačios išėjimas ar būseną. Realios laiko ar integruotos sistemos gali nustatyti techninės įrangos būseną (arba vietą) ir pagal gautus duomenis keisti sistemos parametrus [Lon02; Lon04].

2.3. Duomenų funkcijų ir jų įtakos UFP skaičiavimas

2.3.1. Vidinės loginės bylos

Vidinės loginės bylos (ILF) yra grupė vartotojo atpažįstamų logiškai susietų duomenų, kurie egzistuoja visos programos viduje ir yra pasiekiami per išorinius įėjimus. Šie duomenys turi tam tikrą loginę struktūrą ir yra saugomi bylose.

Paprastai kiekviena vidinė loginė byla turi turėti bent viena su ja susietą išorinį išėjimą ir/arba išorinę užklausą. Informacija yra saugoma ILF tam, kad vėliau ją būtų galima panaudoti. Vidinės loginės bylos gali būti naudojamos kaip kitų programų EO arba EQ.

ILF vertinimas pateiktas 2 lentelėje.

2 lentelė. Vidinių loginių bylų vertinimas [Lon04]

RET	DET		
	1–19	20–50	>50
<2	Žemas (7)	Žemas (7)	Vidutinis (10)
2–5	Žemas (7)	Vidutinis (10)	Aukštas (15)
>5	Vidutinis (10)	Aukštas (15)	Aukštas (15)

Vidinės loginės bylos gali saugoti verslo duomenis, valdymo duomenis arba taisyklėmis besiremiančius duomenis. ILF duomenų tipas yra toks pat, kaip ir EI, kuris gali prieiti prie tos bylos.

Pavyzdžiui, valdymo duomenys gali būti parametų ar būsenų nustatymai. Dalis automobilio sistemos saugo informaciją apie tepalo slėgį, variklio temperatūrą ir pan. Ši sistemos dalis nesi rūpina duomenų perdavimu vairuotojui. Kai pasikeičia situacija, byloje esanti informacija yra atnaujinama. Duomenų vaizdavimas vairuotojui yra visiškai atskiras procesas. Jis nepadedą mašinai važiuoti, o tik informuoja vairuotoją apie jos būklę.

Realaus laiko ir integruotose sistemose (pvz. telefone), yra visų trijų tipų duomenys: verslo duomenys (pats skambutis), taisyklėmis besiremiantys duomenys (skambučio kelias telefonų tinkle) ir valdymo duomenys (taisyklės, pagal kurias telefonai bendrauja vienas su kitu) [Lon02; Lon04].

2.3.2. Išorinės sąsajos bylos

Išorinės sąsajos bylos (EIF) yra vartotojo atpažįstama logiškai susietų duomenų grupė, kuri naudojama tik kaip nuoroda. Šie duomenys yra už programos ribų ir yra tvarkomi kitos programos išoriniais įėjimais. EIF yra kitos programos ILF. Programoje byla gali būti laikoma arba EIF, arba ILF, bet ne abiem vienu metu.

Kiekviena išorinės sąsajos byla turi turėti bent viena su ja susietą išorinį išėjimą arba išorinę užklausą. Kiekviena programa, naudojanti EIF, turi įtraukti ją į savo funkcinių taškų skaičių.

EIF vertinimas pateiktas 3 lentelėje.

3 lentelė. Išorinių sąsajos bylų vertinimas [Lon04]

RET	DET		
	1–19	20–50	>50
<2	Žemas (5)	Žemas (5)	Vidutinis (7)
2–5	Žemas (5)	Vidutinis (7)	Aukštas (10)
>5	Vidutinis (7)	Aukštas (10)	Aukštas (10)

2.3.3. Funkcinių taškų skaičiavimas byloms

Bylų ir jų tipų identifikacijai paprastai naudojama programos dokumentacija ir duomenų perdavimo taisyklės. Bylos tipui (ILF ar EIF) nustatyti naudojama programos dokumentacija ir duomenų perdavimo taisyklės. DET ir RET tipui nustatyti naudojama programos dokumentacija (duomenų modeliais) ir bylų taisyklės. Kiekvienai bylai priskiriama reikšmė (*žemas*, *vidutinis*, *aukštas*), kuri priklauso nuo jos tipo, duomenų elementų ir kreipimosi bylų tipų. Bylos nekoreguotų funkcinių taškų įvertis gaunamas susumavus visų bylų reikšmes [Jon13; Lon04].

2.4. Transakcinių funkcijų ir jų įtakos UFP skaičiavimas

2.4.1. Išoriniai įėjimai

Išoriniai įėjimai (EI) yra elementarūs procesai, kuriuose duomenys kerta nustatytas programos ribas iš išorės į vidų. Šie duomenys patenka į programą iš išorės (duomenų įvedimo ekrano ar kitos programos) ir gali būti naudojami vidinėse loginėse bylose arba dalyvauti informacijos valdyje.

Jei išorinis įėjimas prideda, pakeičia arba ištrina informacija vidinėje loginėje byloje, tai jis skaičiuojamas kaip trys skirtingi EI. Išoriniai įėjimai (ypač keitimas ir trynimasis) gali būti atliekami išorinėmis užklausomis.

EI vertinimas pateiktas 4 lentelėje.

4 lentelė. Išorinių įėjimų vertinimas [Lon04]

FTR	DET		
	1–4	5–15	>15
<2	Žemas (3)	Žemas (3)	Vidutinis (4)
2	Žemas (3)	Vidutinis (4)	Aukštas (5)
>2	Vidutinis (4)	Aukštas (5)	Aukštas (5)

Išoriniais įėjimais gali būti verslo duomenys, valdymo duomenys arba duomenys, besiremiantys tam tikromis taisyklėmis. Verslo duomenimis gali būti laikomi klientų vardai, adresai, telefonai ir pan. Programoje duomenų elementai yra tie elementai, kurie sukelia tam tikrą veiksmą arba pakeičia programos elgesį. Valdymo informacija nurodo kaip, kada ir kokie duomenys bus perduodami [Jon13; Lon04].

2.4.2. Išoriniai išėjimai

Išoriniai išėjimai (EO) yra elementarūs procesai, kuriuose gauti duomenys kerta nustatytas programos ribas iš vidaus į išorę. Išoriniai išėjimai taip pat gali atnaujinti ILF. Duomenys pranešimų arba bylų (sukurtų pagal vienos ar kelių vidinių loginių bylų ir išorinių sąsajos bylų informaciją) pavidalu siunčiami kitoms programoms. Gauti duomenys paprastai yra tam tikrų algoritmų ar skaičiavimų rezultatai. Gaunami duomenys atsiranda, kai vienas ar keli duomenų elementai sujungiami su formulėmis ir taip sugeneruojami arba gaunami papildomi duomenų elementai. Šie duomenys neatsiranda jokiose FTR.

Duomenų perdavimai tarp kelių programų laikomi sąsajomis. Programos išorėje esančius duomenis galima pasiekti tik naudojantis išoriniais įėjimais arba išorinėmis užklausomis.

EO vertinimas pateiktas 5 lentelėje.

5 lentelė. Išorinių įėjimų vertinimas [Lon04]

FTR	DET		
	1–5	6–19	>19
<2	Žemas (4)	Žemas (4)	Vidutinis (5)
2–3	Žemas (4)	Vidutinis (5)	Aukštas (7)
>3	Vidutinis (5)	Aukštas (7)	Aukštas (7)

Kitaip nei kiti komponentai, išoriniai išėjimai dažniausiai būna verslo duomenys. Perspėjimo pranešimai laikomi EO, nes jie yra kokių nors loginių verslo procedūrų rezultatas. Pavyzdžiui, pardavimų programa gali perspėti pardavėją, jog pirkėjas, norintis pirkti prekę, neturi reikiamo kiekio pinigų. Perspėjimo pranešimai skiriasi nuo klaidų pranešimų, nes jie yra elementarūs procesai, o klaidų pranešimai – tik elementarių procesų dalys.

Duomenys, gaunami tekstiniu pavidalu (eilutėmis ir stulpeliais) ir grafiniu formatu yra dviejų išorinių išėjimų pavyzdys [Jon13; Lon04].

2.4.3. Išorinės užklausos

Išorinės užklausos (EQ) yra elementarūs procesai su įėjimo ir išėjimo komponentais, kurie gauna duomenis iš vienos ar kelių ILF arba EIF. Įėjimo procesas neatnaujina ir nekeičia jokių FTR, o išėjimo procesas neturi jokių išvedamų duomenų.

EQ vertinimas pateiktas 6 lentelėje.

6 lentelė. Išorinių užklausų vertinimas [Lon04]

FTR	DET		
	1–5	6–19	>19
<2	Žemas (3)	Žemas (3)	Vidutinis (4)
2–3	Žemas (3)	Vidutinis (4)	Aukštas (6)
>3	Vidutinis (4)	Aukštas (6)	Aukštas (6)

EQ gali būti laikomi verslo duomenys (klientų vardai, adresai, telefonai ir pan.), valdymo duomenys, duomenys besiremiantys tam tikromis taisyklėmis (lentelės įrašas, kuris nurodo, kiek dienų klientas gali vėluoti sumokėti už prekes), valdymo duomenys, pasirinkimo sąrašai, langai su informacija apie klientą, duomenų atstatymo (*Reset/Restore*) funkcijos (visi pakeisti laukai užpildomi anksčiau išsaugotomis reikšmėmis)[Jon13; Lon04].

2.5. Projekto vystymo funkciniai taškai

Projekto vystymo funkcinis taškus tenka skaičiuoti, kai jau esame projekte yra daromi kokie nors pakeitimai. Jie beveik visada įtakoja funkcinį taškų skaičių, nes keičiantis programinės įrangos apimčiai, funkcionalumui, atsirandant ar pranykstant kokioms nors funkcijoms keičiasi ir projekto UFP įvertis.

Projekto vystymo funkcinio tašku vadinamas toks funkcinis taškas, kuris gaunamas atlikus projekto pakeitimą, pakeičiantį, pridedantį ar panaikinantį esamą projekto funkcionalumą. Taip įvyksta, jei:

- prie išorinio įėjimo pridedami DET,
- pakeičiami išorinio įėjimo DET,
- atsiranda nauji FTR,
- pakeičiami EI esantys skaičiavimai.

2.6. Reikšmės koregavimo faktorius

Bendrasis reikšmės koregavimo faktorius (VAF) yra skirtas UFP reikšmės koregavimui atsižvelgiant į pagrindines sistemos, kurioje veiks programinė įranga, charakteristikas (GSC). Kiekviena charakteristika turi konkretų aprašymą, pagal kurį galima nustatyti jos reikšmę sistemai skalėje nuo 0 iki 5.

2.6.1. Pagrindinės sistemos charakteristikos

Pagal IFPUG vadovą, pagrindinių sistemos charakteristikų yra 14 [Lon04]:

1. Ryšio priemonės, naudojamos duomenų keitimuisi.
2. Išskirstytas duomenų apdorojimo naudojimas.
3. Programinės įrangos greičio ar duomenų pralaidumo reikalavimai.
4. Techninės įrangos apkrovimas.
5. Duomenų perdavimo dažnumas.
6. *Online* duomenų įvedimo naudojimas.
7. Programinės įrangos patogumas vartotojui.
8. *Online* atnaujinimo galimybė.
9. Sudėtingų matematinių ar loginių duomenų apdorojimas.
10. Pakartotinis programinės įrangos panaudojimas.
11. Programinės įrangos diegimo paprastumas.
12. Naudojimo, paleidimo, atsarginių kopijų saugojimo paprastumas.
13. Programinės įrangos pritaikomumas keliems tinklams/organizacijoms.
14. Programinės įrangos keitimo sudėtingumas.

Kiekvienos charakteristikos įtaka sistemai vertinama pagal 7 lentelę.

7 lentelė. Įtakos sistemai įvertinimas [Lon04]

Įvertinimas	Įtaka sistemai
0	Nėra arba nedaro jokios įtakos
1	Labai maža įtaka
2	Nedidelė įtaka
3	Vidutinė įtaka
4	Žymi įtaka
5	Didelė įtaka

2.6.2. VAF skaičiavimas

Įvertinus kiekvieną iš 14 GSC, reikšmės koregavimo faktorius skaičiuojamas pagal šią formulę [Lon04]:

$$VAF = 0,65 + \sum_{i=1}^{14} C_i / 100 ,$$

čia C_i — kiekvienos iš 14 GSC įvertinimas.

2.6.3. VAF kritika

C. J. Lokan savo studijoje [Lok00] kritikuoja VAF reikalingumą. Pirmiausia jis pastebi, jog egzistuoja dvigubas sudėtingumo skaičiavimas UFP ir VAF vertinime, o kai kurios sistemos charakteristikos siejasi viena su kita. Toks skaičiavimas prieštarauja skaičiavimo teorijai.

Jis taip pat pastebėjo, jog nors VAF reikšmė gali būti įvairi, dažniausiai ji tik labai nežymiai įtakoja FPA rezultatą. Tai tik patvirtina teiginį, jog reikšmės koregavimas FPA procese yra neefektyvus ir nereikalingas.

Nors IFPUG vis dar naudoja VAF savo vartotojo vadove, tačiau Mark II funkcinių taškų skaičiavimo metodikoje ir ISO standarte šis faktorius jau nebefigūruoja.

C. J. Lokan taipogi kritikuoja GSC teigdamas, jog IFPUG siūloma vertinimo skalė yra per maža, o charakteristikų sąrašą reikia koreguoti, nes jis yra pasenęs. GSC taip pat turi būti naudojamos ne kaip VAF dalis, bet kaip atskiras sistemos įvertinimas norint geriau suprasti jos veikimą ir palyginti su kitomis sistemomis. Jis siūlo apjungti kai kurias charakteristikas ir pateikia tokį 5 pagrindinių sistemos charakteristikų sąrašą [Lok00]:

- nepriklausomumas nuo kitų sistemų,
- sudėtingumas,
- interaktyvumas,
- veiksmingumas,
- vartotojo patogumas.

N. Choursiya ir R. Yadav savo darbe [CY15] teigia, jog dabar IFPUG naudojamos sistemos charakteristikos yra neefektyvios, nes kompiuterinės sistemos laikui bėgant labai pakito. Jie pasiūlė naują pagrindinių sistemos charakteristikų, kurios tiesiogiai ar netiesiogiai įtakoja programinės įrangos apimtį ir jos vertinimą, sąrašą:

- Programavimas pagal pavyzdžius.
- Trumpai egzistuojančio kodo rašymas (*throw away codes*).
- Lengvai skaitomo kodo rašymas.
- Tinkamumo tikrinimas.
- Darbuotojų sauga.
- Patikimumas.
- Analizės įrankiai.
- Klaidų aptikimo įrankiai.
- Atnaujinimų perspėjimai.
- Darbas su daugeliu langų.
- Lygiagretieji skaičiavimai.
- Patikrinamas kodas.
- *Online* pagalbos įrankiai.
- Lengvas aplinkos keitimas vartotojui.
- Lankstus kodas.
- Galingas klaidų aptikimo mechanizmas.
- Keitimo galimybės.
- Priežiūros paprastumas.

2.7. Koreguotų funkcinių taškų skaičiavimas

Kuriant naują programinę įrangą reikia žinoti viso projekto apimtį. Tai reiškia, jog reikia atsižvelgti į būsimos programos funkcinius taškus ir į visus papildomus funkcinius taškus, kurie taip pat bus naudojami programos kūrime. Pavyzdžiui, gali reikėti sukurti laikiną nedidelę programą, kuri turimus duomenis pakoreguoja pagal reikiamą formatą. Taigi, dalis funkcinių taškų bus programoje, kuri bus diegiama sistemoje, o dalis — laikinose programose, kurių užsakovas nematys.

Jei reikia pakeisti jau parašytą programą, svarbu žinoti programos apimtį, funkcinių taškų skaičių, projekto pakeitimus, pridėtas ir pašalintas funkcijas. Paprastai vystomo projekto dydis

apskaičiuojamas pakeistą ar pridėtą funkcionalumą dauginant iš reikšmės koregavimo faktoriaus. Projekto vystymo metu VAF paprastai nekinta.

Taip pat svarbu žinoti, kiek pasikeičia pati programa projekto vystymo eigoje. Ar ji tampa didesnė nei buvo anksčiau? Jei didesnė, tai kokie funkcionalumai pridėti? Kokie funkcionalumai praplėsti? Praktikoje programos dydis dažniausiai priklauso nuo pridėto funkcionalumo [Lon02; Lon04].

2.7.1. Nekoreguoti funkciniai taškai

Nekoreguoti funkciniai taškai (UFP) skaičiuojami pagal 8 lentelę [Lon04]:

8 lentelė. Nekoreguotų funkcinų taškų skaičiavimas

Komponentas	Komponentų sudėtingumas			
	Žemas	Vidutinis	Aukštas	Iš viso
Išoriniai įėjimai	$EI \times 3$	$EI \times 4$	$EI \times 6$	UFP_{EI}
Išoriniai išėjimai	$EO \times 4$	$EO \times 5$	$EO \times 7$	UFP_{EO}
Išorinės užklausos	$EQ \times 3$	$EQ \times 4$	$EQ \times 6$	UFP_{EQ}
Vidinės loginės bylos	$ILF \times 7$	$ILF \times 10$	$ILF \times 15$	UFP_{ILF}
Išorinės sąsajos bylos	$EIF \times 5$	$EIF \times 7$	$EIF \times 10$	UFP_{EIF}
Bendras nekoreguotų funkcinų taškų skaičius				UFP

2.7.2. Funkcinių taškų skaičius

Galutinis funkcinų taškų skaičius (FPC) gaunamas sudauginus reikšmės koregavimo faktorių ir nekoreguotų funkcinų taškų skaičių. Standartinė lygtis atrodo taip [Lon04]:

$$FPC = UAF \times VAF.$$

2.7.3. Projekto vystymo funkcinų taškų skaičiavimas

Projekto vystymo funkcinų taškų (DFP — *Development project Function Points*) skaičiavimui naudojama tokia formulė:

$$DFP = (UFP + CFP) \times VAF.$$

Formulėje įtrauktas papildomas įvertis, vadinamas keitimo funkcinų taškų įverčiu (CFP — *Conversion Function Points*). Kai sena programa keičiama nauja, senojoje programoje esančius duomenis reikia perkelti į naująją. Šiai užduočiai atlikti kartais reikia parašyti nedidelę pagalbinę programą. Kai naujoji programa veikia, pagalbinė programa nebenaudojama [Lon02; Lon04].

2.7.4. Programinės įrangos funkcinų taškų skaičiavimas

Bet kuriuo metu programinės įrangos funkcinus taškus (AFP — *Application Function Points*) galima apskaičiuoti pagal tokią formulę:

$$AFP = ADD \times VAF .$$

ADD — prie programos pridėtų funkcijų nekoreguotų funkcinių taškų įvertis. Kadangi programoje buvo daroma daug pakeitimų, kurie nebuvo vertinami, šiuo atveju ADD nurodo visą programos funkcionalumą tą akimirka, kai skaičiuojami funkciniai taškai [Lon04].

2.7.5. Projekto keitimo funkcinių taškų skaičiavimas

Skaičiuojant projekto keitimo funkcinių taškų (EFP — *Enhancement project Function Points*) įvertį praktikoje naudojama tokia formulė:

$$EFP = ((ADD + CHGA + DEL) \times VAF) .$$

CHGA — funkcinių taškų, kurie buvo pakeisti, įvertis po jų pakeitimo, DEL — funkcinių taškų, kurie buvo pašalinti, absoliutinis įvertis [Lon04].

Projekto funkcinių taškų skaičius po jo keitimo praktikoje paprastai skaičiuojamas pagal formulę:

$$AFP = (UFPB + ADD + CHGA - CHGB - DEL) \times VAF .$$

UFPB — nekoreguotų funkcinių taškų įvertis prieš tobulinimą, CHGB — funkcinių taškų, kurie buvo pakeisti tobulinant programą įvertis iki jų pakeitimo.

Gautas funkcinių taškų įvertis padeda nustatyti programos kainą, jos sudėtingumą ir programavimui reikalingą laiką, dar prieš pradedant ją rašyti. Šis metodas taip pat leidžia įvertinti jau naudojamą programą ir palyginti ją su alternatyviomis programomis.

Norint pakeisti tam tikras programos dalis, to pakeitimo kainą daug lengviau nustatyti remiantis konkrečiais matavimo vienetais — funkciniais taškais, kurie bus pridėti prie esamos programinės įrangos [Lon04].

3. Automatinis funkcinių taškų skaičiavimas

Siekiant sumažinti funkcinių taškų skaičiavimo kainą per pastaruosius 15 metų buvo sukurta nemažai tam skirtų įrankių. Išanalizavęs automatinio ir rankinio funkcinių taškų skaičiavimo rezultatus R. Heller teigia [Hel13], jog automatiniai funkcinių taškų skaičiavimo įrankiai yra labai netikslūs. Automatinis įvertis dažnai skiriasi nuo rankinio skaičiavimo įverčio net iki 40%. Jis taip pat pamini, jog laikas, sugaištas ruošiant duomenis netiksliam automatiniam skaičiavimui ir pats skaičiavimo procesas trunka beveik tiek pat, kiek tikslus rankinis funkcinių taškų skaičiavimas.

Nors FPA yra naudojama jau seniai, vis dar nėra sukurtas visiškai patikimas ir tikslus įrankis automatiniam funkcinių taškų skaičiavimui. UML kalba standartizuotu būdu atvaizduoja sistemą, taigi naudodamiesi jos statine (klasių) ir dinamine (sekų) diagramomis galime išmatuoti tos sistemos UFP. Nuo 1999 m. buvo pasiūlyti keli metodai, skirti funkcinių taškų skaičiavimui naudojant UML diagramas [Lee15].

Vienas iš metodų pirmiausia siūlo apibrėžti, kokio tipo funkciniai taškai bus skaičiuojami ir nustatyti funkcinių taškų skaičiavimo ribas. Kitame žingsnyje remiantis klasių ir sekų diagramomis galima automatiškai suskaičiuoti duomenų funkcijų tipus (ILF ir EIF) ir įvertinti kiekvieno jų sudėtingumą. Toliau skaičiuojami perdavimo funkcijų tipai (DET ir FTR) ir jų sudėtingumai. Tokio metodo įgyvendinimui užtenka trijų skaičiavimo blokų ir dviejų duomenų bazių, o jo skaičiavimai lyginant su FPA specialisto skaičiavimais, yra ganėtinai tikslūs [UKI99].

Vartotojo sąsajai XML kalbą naudojančių programų dydį funkciniais taškais taip pat galima suskaičiuoti automatiškai nuskaitant visas XML bylas. Šis metodas yra ganėtinai tikslus ($\pm 6,65\%$) ir labai greitas [LQ15].

Įvairūs autoriai labai įvairiai vertina automatinio funkcinių taškų skaičiavimo galimybes, o jų tyrimų rezultatai labai skiriasi. Šiuo metu automatinis funkcinių taškų skaičiavimas vis dar nėra nei greitesnis, nei pigesnis, nei tikslesnis už įprastinį rankinį skaičiavimą, tačiau ateityje jis labai palengvins įmonių darbą ir leis tiksliau ir greičiau įvertinti programinės įrangos kūrimui reikalingas sąnaudas.

4. Praktinis pavyzdys: SMDB funkcinių taškų skaičiavimas

Šiame skyriuje aprašytas realiai veikiančios darbo autoriaus sukurtos programinės įrangos, santuokos metrikų duomenų bazės (SMDB), funkcinių taškų skaičiavimas. SMDB yra intraneto puslapis, kuriuo naudodamasis vartotojas gali peržiūrėti santuokų metrikas, įrašyti naujų santuokų duomenis ir spausdinti įvairius dokumentus.

4.1. SMDB funkcinių taškų skaičiavimo tipas

Kadangi SMDB yra jau egzistuojanti programa, funkcinių taškų skaičiavimo tipas yra „programinės įrangos FPA“. Nors skaičiavimo tipo nustatymas yra svarbus tik skaičiavimo proceso pabaigoje, tačiau žinant jį skaičiavimo pradžioje galima praleisti kai kurias smulkmenas, kurios neturi įtakos skaičiavimo rezultatui.

4.2. SMDB ribos ir funkcinių taškų skaičiavimo tikslas

Antrasis FPA žingsnis yra skaičiavimo ribų ir tikslo nustatymas. Atliekamo skaičiavimo tikslas yra SMDB dydžio funkciniais taškais nustatymas ir paties skaičiavimo proceso analizė. Funkcinių taškų skaičiavimas atliekamas tik SMDB. *Apache HTTP* ir *MySQL* serveriai, reikalingi SMDB veikimui, yra paliekami už funkcinių taškų skaičiavimo ribų.

4.3. SMDB nekoreguotų funkcinių taškų skaičiavimas

Pirmieji du FPA žingsniai nereikalauja daug laiko sąnaudų, tačiau yra svarbūs, nes padeda nustatyti programos apimtį ir funkcinių taškų skaičiavimo tikslą. Trečiasis etapas yra ilgiausias ir sudėtingiausias — būtent čia vyksta tikrasis skaičiavimas. Suskaičiavus ir susumavus visas duomenų bei transakcines funkcijas, ir įvertinus jų sudėtingumą, gaunamas programos nekoreguotų funkcinių taškų įvertis.

4.3.1. SMDB duomenų funkcijos

4.3.1.1. SMDB vidinės loginės bylos

Didžioji dalis SMDB duomenų yra saugomi duomenų bazės lentelėse ir tik nedidelė dalis — tekstinėje byloje. SMDB naudojamos duomenų bazės lentelės yra šios:

- Santuokos
- Kunigai
- JiCivBukle
- Parapijos
- JisCivBukle

Visos lentelės yra vienoje plokštumoje ir viena su kita nesusietos — jose saugomi skirtingi duomenys. Dalis nustatymams skirtų duomenų saugoma tekstinėje byloje *Nustatymai*. Šiose lentelėse

ir byloje *Nustatymai* esantys duomenys yra ILF, nes jie yra saugomi ir tvarkomi SMDB programos ribose. Detali duomenų bazės lentelių ir tekstinės bylos *Nustatymai* analizė pateikta priede Nr. 1.

Žinant kiekvienos loginės bylos RET ir DET skaičius pagal 2 lentelę buvo apskaičiuotas ILF įvertinimas funkciniais taškais (9 lentelė).

9 lentelė. SMDB ILF įvertinimas funkciniais taškais

ILF	RET	DET	Sudėtingumas	FP
Santuokos	2	55	Aukštas	15
Parapijos	1	1	Žemas	7
Kunigai	1	1	Žemas	7
JisCivBukle	1	1	Žemas	7
JiCivBukle	1	1	Žemas	7
Nustatymai	2	1	Žemas	7
Prisijungimas	1	1	Žemas	7
Suma:				57

4.3.1.2. SMDB išorinės sąsajos bylos

SMDB nenaudoja jokių už programos ribų esančių duomenų, taigi EIF reikšmė yra lygi 0 ir nedaro jokios įtakos funkcinį taškų skaičiui.

4.3.2. SMDB transakcinės funkcijos

Transakcinių funkcijų vertinimui turi būti išanalizuota vartotojo sąsaja. Kiekvienas vartotojo sąsajos langas turi elementus, kurie gali būti išoriniai įėjimai, išėjimai ar užklausos. Skaičiuojant FTR ir DET transakcinėms funkcijoms reikia gerai suprasti programos veikimą ir duomenų perdavimus joje. Visos vartotojo sąsajos langų vaizdai pateikti priede Nr. 2.

10 lentelėje pateiktas visų programoje vykstančių procesų aprašymas.

10 lentelė. SMDB procesai

Procesas	Proceso aprašymas
Prisijungimas	Vartotojui įvedus teisingą slaptažodį sukuriamą nauja sesija. Prisijungus nukreipiama į tuščią duomenų įvedimo langą (P1 pav.).
Duomenų įvedimas	Vartotojas gali įvesti naujos arba redaguoti jau esamos metrikos duomenis ir spausdinti dokumentus (P2 pav.).
Nustatymų keitimas	Vartotojas gali keisti programos nustatymus (P3 pav.).
Paieška	Vartotojas pagal įvairius kriterijus gali vykdyti santuokos anketos paiešką, spausdinti reikalingus dokumentus bei adresus ant vokų (P4 pav.).
Lipdukų spausdinimas	Vartotojas gali atspausdinti lipdukus su adresais (P5 ir P6 pav.).
Metrikos spausdinimas	Vartotojas gali atspausdinti pasirinktos santuokos metriką (P7 pav.).
Pranešimo CMS spausdinimas	Vartotojas gali atspausdinti pasirinktos santuokos pranešimą CMS skyriui (P8 pav.).
Pranešimų parapijoms spausdinimas	Vartotojas gali atspausdinti pasirinktos santuokos pranešimus parapijoms (P9 pav.).

Žinant programoje vykstančius procesus galima pradėti funkcinių taškų skaičiavimą.

4.3.2.1. SMDB išoriniai įėjimai

Išoriniai įėjimai analizuojamoje programoje yra vartotojo įvedami duomenys, kurie yra įrašomi į vieną ar kelias vidines logines bylas. Išanalizavus vartotojo sąsajos langus ir suskaičiavus išorinių įėjimų DET ir FTR, pagal 4 lentelę buvo apskaičiuotas EI įvertinimas funkciniais taškais (11 lentelė).

11 lentelė. SMDB EI įvertinimas funkciniais taškais

Procesas	DET	FTR	Sudėtingumas	FP
Prisijungimas	2	1	Žemas	3
Duomenų įvedimas	65	3	Aukštas	5
Nustatymų keitimas	5	3	Aukštas	5
Paieška	5	1	Žemas	3
Lipdukų spausdinimas	1	1	Žemas	3
			Suma:	19

4.3.2.2. SMDB išoriniai išėjimai

Išoriniais išėjimais duomenys išeina už programos ribų. Tam, kad duomenų išvedimas būtų laikomas išoriniu išėjimu, jame turi būti atliktas bent vienas matematinis ar loginis veiksmas, o duomenys turi būti įrašyti į logines bylas arba koku nors būdu įtakoti programos veikimą. Išanalizavus vartotojo sąsajos langus ir suskaičiavus išorinių išėjimų DET ir FTR, pagal 5 lentelę buvo apskaičiuotas EO įvertinimas funkciniais taškais (12 lentelė).

12 lentelė. SMDB EO įvertinimas funkciniais taškais

Procesas	DET	FTR	Sudėtingumas	FP
Lipdukų spausdinimas	4	1	Žemas	4
Suma				4

4.3.2.3. SMDB išorinės užklausos

Išorines užklausas gali būti sunku atskirti nuo išorinių išėjimų. Jos yra paprastesnės, nes jose negali būti atliekami matematiniai ar loginiai skaičiavimai ar duomenų įrašymas į logines bylas. EQ negali įtakoti programos veikimo: dažniausiai tai yra įvairios ataskaitos ar nesudėtingi užklausų rezultatai. Išanalizavus vartotojo sąsajos langus ir suskaičiavus išorinių išėjimų DET ir FTR, pagal 6 lentelę buvo apskaičiuotas EQ įvertinimas funkciniais taškais (13 lentelė).

13 lentelė. SMDB EQ įvertinimas funkciniais taškais

Procesas	DET	FTR	Sudėtingumas	FP
Duomenų įvedimas	8	5	Aukštas	6
Nustatymų keitimas	5	2	Žemas	3
Paieška	12	1	Žemas	3
Lipdukų spausdinimas	4	1	Žemas	3
Metrikos spausdinimas	42	1	Vidutinis	4
Pranešimo CMS spausdinimas	21	2	Aukštas	6
Pranešimo parpajoms spausdinimas	19	1	Žemas	3
Suma:				28

4.4. SMDB nekoreguoti funkciniai taškai

Susumavus visus duomenų ir transakcinių funkcijų įvertinimus buvo apskaičiuotas programos nekoreguotų funkcinio taškų skaičius (14 lentelė).

14 lentelė. SMDB nekoreguotų funkcinio taškų skaičius

Funkcijos tipas	Sudėtingumas	Daugiklis	Funkciniai taškai	Funkcinių taškų suma
ILF	6 × žemas	× 7 =	42	57
	0 × vidutinis	× 10 =	0	
	1 × aukštas	× 15 =	15	
EIF	0 × žemas	× 5 =	0	0
	0 × vidutinis	× 7 =	0	
	0 × aukštas	× 10 =	0	
EI	3 × žemas	× 3 =	9	19
	0 × vidutinis	× 4 =	0	
	2 × aukštas	× 5 =	10	
EO	1 × žemas	× 4 =	4	4
	0 × vidutinis	× 5 =	0	
	0 × aukštas	× 7 =	0	
EQ	4 × žemas	× 3 =	12	28
	1 × vidutinis	× 4 =	4	
	2 × aukštas	× 6 =	12	
Nekoreguoti funkciniai taškai:				108

4.5. SMDB reikšmės koregavimo faktorius

Žinant nekoreguotų funkcinio taškų skaičių galima pereiti prie kito žingsnio — VAF nustatymo. Tyrimo tikslais VAF skaičiavimas buvo atliktas naudojantis ne tik IFPUG, bet ir kitų autorių siūlomais sistemos charakteristikų vertinimais.

15 lentelėje pateiktas analizuojamos programos pagrindinių sistemos charakteristikų įvertinimas pagal IFPUG, o 16 lentelėje – sistemos įvertinimas naudojantis N. Choursiya ir R. Yadav siūlomomis charakteristikomis.

15 lentelė. SMDB GSC įvertinimas pagal IFPUG

Charakteristika	Įvertinimas
Duomenų perdavimai	3
Išskirstytas duomenų apdorojimas	1
Galios reikalavimai	0
Techninės įrangos apkrovimas	2
Duomenų perdavimo dažnumas	0
Online duomenų įvedimas	5
Patogumas vartotojui	4
Online atnaujinimas	3
Sudėtingų duomenų apdorojimas	2
Pakartotinis naudojimas	2
Pritaikymas keliems tinklams/organizacijoms	1
Nesudėtingas diegimas	1
Nesudėtingas programinės įrangos keitimas	4
Naudojimo paprastumas	2
Suma:	30

16 lentelė. SMDB GSC pagal N. Choursiya ir R. Yadav [CY15]

Charakteristika	Įvertinimas
Programavimas pagal pavyzdžius	0
Trumpai egzistuojančio kodo rašymas (<i>throw away codes</i>)	2
Lengvai skaitomo kodo rašymas	3
Tinkamumo tikrinimas	5
Darbuotojų sauga	3
Patikimumas	5
Analizės įrankiai	2
Klaidų aptikimo įrankiai	1
Atnaujinimų perspėjimai	0
Darbas su daugeliu langų	1
Lygiagretieji skaičiavimai	2
Patikrinamas kodas	2
Online pagalbos įrankiai	0
Lengvas aplinkos keitimas vartotojui	0
Lankstus kodas	3
Galingas klaidų aptikimo mechanizmas	2
Keitimo galimybės	3
Priežiūros paprastumas	3
Suma:	37

Žinant sistemos charakteristikų įvertinimą pagal 2.6.2 skyriuje pateiktą formulę galima apskaičiuoti funkcinių taškų reikšmės koregavimo faktorių. Šio faktoriaus reikšmė gali svyruoti nuo 0,65 iki 1,35, taigi ji šiek tiek daugiau nei trečdaliu gali padidinti ar sumažinti sistemos funkcinių taškų skaičių. Naudojant IFPUG sistemos charakteristikas, reikšmės koregavimo faktorius yra:

$$VAF_{IFPUG} = 0,65 + 30/100 = 0,95 .$$

Naudojant N. Choursiya ir R. Yadav sistemos charakteristikas buvo gauta šiek tiek didesnė VAF reikšmė:

$$VAF_{CY} = 0,65 + 37/100 = 1,02 .$$

4.6. SMDB funkcinių taškų skaičius

Paskutinis funkcinių taškų skaičiavimo žingsnis yra koreguotų funkcinių taškų (FPC) apskaičiavimas. FPC skaičiuojami pagal 2.7.2 skyriuje patiektą formulę. Skaičiavimui naudojama UFP reikšmė iš 14 lentelės. Kadangi buvo apskaičiuotos dvi dvi VAF reikšmės, FPC reikšmės taip pat gautos dvi:

$$FPC_{IFPUG} = 108 \times 0,95 = 102,6 ,$$

$$FPC_{CY} = 108 \times 1,02 = 110,16 .$$

4.7. SMDB funkcinių taškų skaičiavimo analizė

Funkcinių taškų skaičiavimas buvo įdomi užduotis. Nors teorija buvo jau išnagrinėta, atliktas praktinis darbas leido geriau suprasti skaičiavimo metodiką.

Pirmieji du skaičiavimo žingsniai (FPA tipo, ribų ir tikslo nustatymas) nebuvo sudėtingi. Duomenų funkcijų (ILF ir EIF) skaičiavimas taip pat nesukėlė didelių sunkumų. Šiek tiek daugiau laiko užėmė visų ILF suskaičiavimas ir jų analizė skaičiuojant RET ir DET.

Daugiausia laiko buvo praleista skaičiuojant transakcines funkcijas. Šiame etape buvo padarytos visų programos langų momentinės kopijos ir ištirti visi jų elementai priskiriant juos EI, EO arba EQ. Dvejonių sukėlė būtent EO ir EQ skaičiavimas, nes šie procesai yra labai panašūs ir ne visuomet lengvai skiriami.

Visus rezultatus surašius į vieną lentelę reikėjo suskaičiuoti reikšmės koregavimo faktorių įvertinant pagrindines sistemos charakteristikas. Tyrimo tikslais VAF skaičiavimui buvo naudojamas tiek IFPUG, tiek N. Choursiya ir R. Yadav siūlomos sistemos charakteristikomis. Antruoju atveju VAF reikšmė buvo šiek tiek didesnė.

Paskutiniame etape pagal formulę buvo apskaičiuoti koreguoti funkciniai taškai. Kadangi buvo apskaičiuotos dvi skirtingos VAF reikšmės, FPC reikšmės taip pat buvo skirtingos. Skirtumas tarp reikšmių buvo ganėtinai nedidelis — 7%.

Capers Jones atliktais tyrimais, standartinis funkcinių taškų skaičiavimo greitis yra maždaug 500 FP per dieną (62,5 FP/val.) [Jon13]. SMDB funkcinių taškų skaičiavimas užtruko maždaug penkias valandas. Skaičiavimo greitis buvo 3 kartus mažesnis nei profesionalių funkcinių taškų skaičiuotojų. Tai nenustebino, nes paprastai įmonėse funkcinis taškus skaičiuoja sertifikuoti ir

specialius mokymus turėję darbuotojai, o darbo autoriui praktinis funkcinų taškų skaičiavimas buvo naujovė.

Vienas pagrindinių funkcinų taškų analizės tikslų yra laiko, skirto programinės įrangos kūrimui, įvertinimas. Yra žinoma, jog SMDB kūrimui darbo autorius sugaišo maždaug 130 val. Žinant programos kūrimo laiką (t) ir jos įvertinimą funkciniais taškais (FPC) galima apskaičiuoti produktyvumą (vieno funkcinio taško kūrimo laiką valandomis):

$$\begin{aligned} \text{produktyvumas} &= \frac{t}{FPC}, \\ \text{produktyvumas}_{IFPUG} &= \frac{150}{102,6} = 1,27, \\ \text{produktyvumas}_{CY} &= \frac{130}{110,16} = 1,18. \end{aligned}$$

Išanalizavusi savo programinės įrangos kūrimo istoriją (programų apimtį funkciniais taškais ir jų kūrimo laiką) kiekviena įmonė gali nustatyti savo produktyvumą valandomis vienam funkciniam taškui. FPA versle dažniausiai naudojama vertinant vidutinio dydžio (100 – 10000 funkcinų taškų) programinę įrangą. Didesnės apimties programų vertinimas yra neefektyvus ir per brangus, o mažesnių – gali būti netikslus.

C. Jones ir kiti autoriai atliko tyrimus ir nustatė, jog vidutinio dydžio programose (apie 1000 funkcinų taškų) vienam funkciniam taškui įgyvendinti reikia maždaug 1 – 4 valandų. Kuo mažesnė yra programos apimtis, tuo didesnis yra programavimo produktyvumas [Jon13; Jon14; RK13]. Apskaičiuotas SMDB produktyvumas (pagal IFPUG) yra 1,27 valandos funkciniam taškui. Toks aukšto produktyvumo įvertis buvo tikėtasi, nes SMDB nėra sudėtinga programa, joje nėra sudėtingų skaičiavimų ar didelių duomenų perdavimų. Suskaičiavus dar kelių autoriaus sukurtų programų funkcinis taškus ir jų įgyvendinimo trukmę būtų galima nustatyti darbo produktyvumą ir šį įvertį naudoti numatant būsimų programų kūrimo laiką.

Išvados

Kasdien naudojamos programos laikui bėgant tampa vis sudėtingesnės, o jų kūrimui reikia vis daugiau išteklių. Daugelis programinę įrangą kuriančių kompanijų susiduria su sunkumais, kai reikia tiksliai nustatyti programos kūrimo kaštus ir darbui reikalingą laiką. Nors dar nėra vieningos tam tikslui naudojamos metodikos, vis daugiau įmonių programų dydį skaičiuoja funkciniais taškais. Vieningų ir konkrečių matavimo vienetų naudojimas sudaro galimybę projektus lyginti tarpusavyje, o nustatant jų kainas — atsižvelgti į anksčiau buvusių projektų sudėtingumą bei sąnaudas.

Šiame darbe remiantis IFPUG bei įvairių mokslininkų publikacijomis išanalizuotas programinės įrangos įvertinimui funkciniais taškais naudojamas IFPUG funkcinį taškų skaičiavimo metodas. Jame taip pat buvo trumpai aprašytos XX a. antroje pusėje atsiradusios FPA ištakos bei aptartą šio metodo privalumai ir trūkumai.

Didelė darbo dalis buvo skirta funkcinį taškų skaičiavimo proceso analizei ir įvairių autorių pasiūlymams, kaip ją galima patobulinti. Teorinė darbo dalis buvo pritaikyta praktiškai skaičiuojant santuokos metrikų duomenų bazės funkcinį taškų, analizuojant patį skaičiavimo procesą ir įvertinant programos kūrėjo produktyvumą valandomis vienam funkciniam taškui.

Funkcinį taškų skaičiavimo populiarinimui visame pasaulyje leidžiami įvairūs leidiniai, žurnalai ir mokslinės publikacijos. Ateityje šis programinės įrangos matavimo metodas galėtų ne tik pagreitinti ir palengvinti sudėtingų programų kūrimą, bet ir leisti įmonėms efektyviau naudoti savo išteklius. Šie pokyčiai dar aukščiau pakeltų kokybės kartelę naujosioms technologijoms, o dėl tikslesnio programavimui reikalingų sąnaudų numatymo vis daugiau projektų įvyktų laiku ir būtų kokybiškesnis.

Conclusions

For more than 40 years programmers are looking for methods to evaluate the software size. Function point analysis is often presented as most convenient way to accomplish this task. It becomes possible to evaluate productivity and quality of programming process only knowing how to count software size.

IFPUG function point counting method is normally used in practice as it is an accurate and language-independent but it may be slow and expensive as well. After determining function point counting type, boundaries and scope, every elementary process of a software is evaluated. At the end of a counting process general system characteristics are used to adjust the value of function point count.

The author of this work has counted function points of a small program, analysed the counting process and results. The programmers' performance was measured in hours per function point.

In the future function point analysis should become automatic. This would make software creation process quicker and easier, and should be widely used to plan software programming resources and carrying out the work planned.

Keywords: Allan Albrecht, function point analysis, function point count, function points, software measuring, software evaluation, software size, software quality, IFPUG.

Literatūra

- [BTP12] K. Bhatt, V. Tarey, and P. Patel. Analysis of source lines of code (sloc) metric. *International journal of emerging technology and advanced engineering*, 2(5):150–154, 2012.
- [CY15] N. Choursiya and R. Yadav. An enhanced function point analysis (fpa) method for software size estimation. *International journal of computer science and information technologies*, 6(3):2797–2799, 2015.
- [DAC06] J. M. Desharnais, A. Abran, and J. Cuadrado. Convertibility of function points to cosmic-ffp: identification and analysis of functional outliers. *Ensura*:190, 2006.
- [Fet99] T. Fetcke. A generalized structure for function point analysis. *Proc. of the 11th iwsm, lac superieur, canada*:143–153, 1999.
- [Hel13] R. Heller. Automated function point counting—a fact based analysis. *Q/p management group, inc. www.qpmg.com/library/confirm_download.php*, 2013.
- [IFP15] IFPUG. Snap assessment practices manual:157, 2015-05.
- [YS15] F. A. Yusra and M. Sadiq. A fuzzy based approach for function point analysis. *Advances in computer science and information technology*, 2(8):89–91, 2015.
- [ISO09] ISO/IEC. IFPUG functional size measurement method. http://www.iso.org/iso/home/store/catalogue_tc/catalogue_detail.htm?csnumber=51717. 2009.
- [Jon13] C. Jones. Function points as a universal software metric. *Sigsoft softw. eng. notes*, 38(4):1–89, 2013-07.
- [Jon14] C. Jones. Evaluating software metrics and software measurement practices. 2014.
- [Kit97] Barbara Kitchenham. Counterpoint: the problem with function points. *Ieee software*, 14(2):29, 1997.
- [Lee15] H. K. Lee. Automatic measurement of function points from java applications. In *The international conference on software engineering, mobile computing and media informatics (semcmi2015)*, 2015, p. 59.
- [LJ90] G. C. Low and D. R. Jeffery. Function points in the estimation and evaluation of the software process. *Software engineering, ieee transactions on*, 16(1):64–71, 1990.
- [Lok00] C. J. Lokan. An empirical analysis of function point adjustment factors. *Information and software technology*, 42(9):649–659, 2000.
- [Lon02] D. Longstreet. Fundamentals of function point analysis. *Blue springs: longstreet consulting inc*, 2002.
- [Lon04] D. Longstreet. Function points analysis training course. *Software metrics*:111, 2004-10.

- [LQ15] J. S. Lent and Y. Qu. On xml based automated function point analysis: an effective method to assess developer productivity. *Lecture notes on software engineering*, 3(4):245, 2015.
- [RK13] H.K. Raju and Y.T. Krishnegowda. Software sizing and productivity with function points. *Lecture notes on software engineering*, 1(2):204, 2013.
- [Sub14] K. Subbaraman. Demystifying function point estimation. *Ifpug metric views*, 8(2):12–13, 2014-07.
- [SV15] G. S. Simões and C. E. Vazquez. Ten fundamental questions about function point analysis. *Ifpug metric views*, 9(2):4–7, 2015-08.
- [UKI99] T. Uemura, S. Kusumoto, and K. Inoue. Function point measurement tool for uml design specification. In *Software metrics symposium, 1999. proceedings. sixth international*. IEEE, 1999, pp. 62–69.
- [ZZW⁺09] Yinhuan Zheng, Yilong Zheng, Beizhan Wang, and Liang Shi. Estimation of software projects effort based on function point. In *4th international conference on computer science & education*, 2009, pp. 941–943.

Priedas Nr. 1

SMDB vidinės loginės bylos

P1 lentelė. SMDB ILF: Santuokos (MySQL lentelė)

Laukelis	Apibūdinimas	DET
san_data	Santuokos data	+
reg_data	Santuokos registracijos data	+
pran_nr	Pranešimo CMS numeris	
indeksas	Unikalus santuokos indeksas	
met_nr	Santuokos metrikos Nr.	
civ_vieta	Civilinės santuokos vieta	+
civ_data	Civilinės santuokos data	+
jis_vardas	Sutuoktinio duomenys	+
jis_pavarde		+
jis_tevo_vardas		+
jis_tevo_pavarde		+
jis_tel		+
jis_sutv_vieta		+
jis_sutv_data		+
jis_religija		+
jis_parapija		+
jis_motinos_vardas		+
jis_motinos_pavarde		+
jis_motinos_merg_pavarde		+
jis_krik_vieta		+
jis_krik_data		+
jis_gim_vieta		+
jis_gim_data		+
jis_civ_bukle		+
jis_AT_tipas		+
jis_AT_nr		+
jis_ak		+
jis_adresas		+
ji_vardas		+
ji_pavarde		+
ji_tevo_vardas		+
ji_tevo_pavarde		+
ji_tel		+
ji_sutv_vieta		+
ji_sutv_data		+
ji_religija		+
ji_parapija		+
ji_motinos_vardas		+
ji_motinos_pavarde		+
ji_motinos_merg_pavarde		+

ji_krik_vieta		+
ji_krik_data		+
ji_gim_vieta		+
ji_gim_data		+
ji_civ_bukle	Sutuoktinės duomenys	+
ji_AT_tipas		+
ji_AT_nr		+
ji_ak		+
ji_adresas		+
adresas_po_sant	Sutuoktinių adresas po santuokos	+
liud1_vardas		+
liud1_pavarde		+
liud1_adresas	Liudininkų duomenys	+
liud2_vardas		+
liud2_pavarde		+
liud2_adresas		+
pastabos	Pastabos	+
kunigas	Santuoką palaiminęs kunigas	+
Suma:		55

P2 lentelė. SMDB ILF: Parapijos (MySQL lentelė)

Laukelis	DET
id	
Parapija	+
Suma:	1

P3 lentelė. SMDB ILF: Kunigai (MySQL lentelė)

Laukelis	DET
id	
Kunigas	+
Suma:	1

P4 lentelė. SMDB ILF: JisCivBukle (MySQL lentelė)

Laukelis	DET
id	
Bukle	+
Suma:	1

P5 lentelė. SMDB ILF: JiCivBukle (MySQL lentelė)

Laukelis	DET
id	
Bukle	+
Suma:	1

P6 lentelė. SMDB ILF: Nustatymai (tekstinė byla)

Laukelis	DET
Klebonas	+
Metai	+
Suma:	2

Priedas Nr. 2

SMDB vartotojo sąsajos langų vaizdai

Vilniaus Bernardinų bažnyčia
Santuokos metrikų duomenų bazė

Nustatymai Paieška Nauja metrika

Slaptažodis: *****

Prisijungti

P1 pav. Prisijungimo lango vaizdas

Vilniaus Bernardinų bažnyčia
Santuokos metrikų duomenų bazė

Nustatymai Paieška Nauja metrika

Duomenys įrašyti

Užregistravimo santuokai data: 2015-12-15

Bažnytinės santuokos data: 2015-12-15

Metikos Nr.: 13. 0000

Civilinės santuokos data:

Civilinės santuokos vieta:

Pranešimo Nr.: 13. 0000

Spausdinti metriką

Spausdinti CMS pranešimą

Spausdinti pranešimą parapijoms

	Sužadėtinis	Sužadėtinė
Pavardė:		
Vardas(-ai):	Vladyslav	Aušra
Asmens kodas:		
Gimimo data:		
Gimimo vieta:	Svenčionių raj. St. Jurgio par.	Kretinga
Religija:	Katalikas	Katalikė
Krikšto data:	2015-12-15	
Krikšto vieta:	Vilniaus Bernardinų bažnyčia	Vilniaus Bernardinų bažnyčia
Sutvirtinimo data:	YYYY-MM-DD	YYYY-MM-DD
Sutvirtinimo vieta:		
Civilinė būklė:	Nevedęs	Netekėjusi
Gyvenamoji vieta adresas:	Al. Švenčionių g. 12 Vilnius	Vilnius
Gyvenamoji vieta telefonas:		
Gyvenamoji vieta parapija:	Vilniaus Bernardinų bažnyčia	Vilniaus Bernardinų bažnyčia
Tėvo vardas:	Genadij	Česlovas
Tėvo pavardė:		
Motinos vardas:		
Motinos pavardė:		
Motinos mergautinė pavardė:		
Santuokos liudytojai vardas:	Karina	Ramūnas
Santuokos liudytojai pavardė:		
Santuokos liudytojai adresas:		
Numatomas adresas po santuokos:	Vilnius	
AT dokumentas tipas:	Pasas	Pasas
AT dokumentas Nr.:		
Santuoką palaimino:		
Pastabos:		

Įrašyti

P2 pav. Duomenų įvedimo lango vaizdas

Atsijungti

Vilniaus Bernardinų bažnyčia Santuokos metrikų duomenų bazė

Nustatymai
Paieška
Nauja metrika

Klebonas:

Metai:

Įrašų skaičius: 223

Paskutinės metrikos Nr. 171010

Paskutinio pranešimo Nr. 112010

Nauja parapija/bažnyčia:

Naujas kunigas:

P3 pav. Nustatymų keitimo lango vaizdas

Atsijungti

Vilniaus Bernardinų bažnyčia Santuokos metrikų duomenų bazė

Nustatymai
Paieška
Nauja metrika

Metrikos Nr.	Pranešimo Nr.	Santuokos data	Jaunasis	Jaunoji	
					Ieškoti
171016	112016	2016-01-15	Tomas Skučinskas	Lina Rodauskaitė	M C P
171016	112016	2016-01-15	Rokas Bernatavičius	Rita Navickaitė	M C P
171016	112016	2016-01-15	Deividas Vaitiekaitis	Kristina Vaitiekaitė	M C P
171016	112016	2016-01-15	Jonas Kirilovas	Lina Rodauskaitė	M C P
171016	112016	2016-01-15	Romas Krivickas	Donata Vaitiekaitė	M C P
171016	112016	2016-01-15	Arturas Sedzickas	Donata Vaitiekaitė	M C P
171016	112016	2016-01-15	Vladas Zekas	Donata Vaitiekaitė	M C P
171016	-	2016-01-15	Sebastien Luc Greber	Donata Vaitiekaitė	M C P
171016	112016	2016-01-15	Alfredas Paciuskas	Donata Vaitiekaitė	M C P
171016	-	2016-01-15	Jonas Gata Viden	Donata Vaitiekaitė	M C P

P4 pav. Paieškos lango vaizdas

Atsijungti

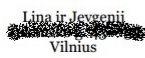
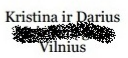
Vilniaus Bernardinų bažnyčia Santuokos metrikų duomenų bazė

Nustatymai
Paieška
Nauja metrika

Lipdukų skaičius lape:

Metrikų numeriai: - iki

P5 pav. Lipdukų spausdinimo lango vaizdas

Irašų skaičius: 17 Spausdinami adresai metrikoms: 1-2016 - dabar Lipdukų skaičius lape: 16 Lapų skaičius: 2 OK	
Rutavičiūtė ir Sebastien Luc  Gintarija ir Arūnas  Lina ir Jevgenij 	Milda ir Vladas  Emilija ir Renatas  Kristina ir Darius 

P6 pav. Lipdukų spausdinimo lango vaizdas

SANTUOKOS METRIKA NR. 18		
	Jaunasis	Jaunoji
Pavardė		
Vardas	Vladyslav	Aušra
Asmens kodas		
Tėvo vardas, pavardė	Genadijus	Česlovas
Motinos vardas, pavardė		
Gimimo data, vieta	Švenčionių raj.,	Kretinga
Kur ir kada pakrikštyti	Kimiliškių bažnyčia, ~1972 m.	Kretingos Viešpaties Apreikimo Švč. Mergelei Marijai parapija,
Kur ir kada sutvirtinti		
Būklė	Nevedęs	Netekėjusi
Jaunųjų adresai	Vilnius	Vilnius
Kada, kur ir kas palaimino santuoka	2016 m. Vilniaus Šv. Pranciškaus Asyžiečio (Bernardinų) bažnyčia,	
Santuokos liudytojai ir jų adresai	Karina	Klaipėda
	Ramūnas	Klaipėda
Pastabos		
Jaunavedžių parašai	Liudytojų parašai	Klebonas
A.V.		

P7 pav. Metrikos spausdinimo lango vaizdas

VILNIAUS ŠV. PRANCIŠKAUS ASYŽIEČIO (BERNARDINŲ) PARAPIJA
Šventaragio g. 4, LT-01122 Vilnius

Vilniaus Šv. Pranciškaus Asyžiečio (Bernardinų) parapija
(religijinė bendruomenė ar bendrijos pavadinimas)

Maironio g. 10, LT-01124 Vilnius, tel. 8-616-01159, info@bernardinai.lt, jm.k. 192051599
(religijinė bendruomenė ar bendrijos duomenys: adresas, telefonas, el. paštas, pranešimo kodas)

Vilniaus miesto CM skyriui
(adresatas, t.y. CM įstaiga)

PRANEŠIMAS APIE BAŽNYČIOS (KONFESIJŲ) NUSTATYTĄ TVARKĄ
SUDARYTĄ SANTUOKĄ
2016 m. gegužės mėn. 5 d., Nr. [redacted]

Santuokos sudarymo vieta: Vilniaus Šv. Pranciškaus Asyžiečio (Bernardinų) parapija
(religijinė bendruomenė ar bendrijos pavadinimas)

Santuokos sudarymo data ir įrašo numeris: [redacted]

DUOMENYS APIE SUTUOKTINIUS		
	JIS	JI
1. Asmens kodas	[redacted]	[redacted]
2. Vardas	Vladyslav	Aušra
3. Pavardė iki santuokos	[redacted]	[redacted]
4. Gimimo data	[redacted]	[redacted]
5. Gimimo vieta	[redacted]	[redacted]
6. Pilietybė	Lietuvos	Lietuvos
7. Šeimyninė padėtis	Nevedęs	Netekėjusi
8. AT dokumentas (pasas ar tapatybės kortelė, kiti dokumentai)	Pasas	Pasas

Parapijos klebonas _____ (parašas) br. kun. Evaldas Darulis OFM _____ (vardas ir pavardė)

A.V. _____

P8 pav. Pranešimo CMS spausdinimo lango vaizdas

Vilniaus Šv. Pranciškaus Asyžiečio (Bernardinų) parapija, Vilniaus arkivyskupija
PRANEŠIMAS APIE SANTUOKOS PALAIMINIMĄ

Prašome padaryti įrašą parapijos krikštų registravimo knygoje.

Krikšto vieta: [redacted]

Krikšto data: [redacted]

Santuoką sudariusiojo vardas, pavardė: Vladyslav [redacted]

Tėvo vardas, pavardė: Genadijus [redacted]

Motinos vardas, pavardė: Valentina [redacted]

Santuoką sudarė su: Aušra [redacted]

Santuoka buvo palaiminta Vilniaus Šv. Pranciškaus Asyžiečio (Bernardinų) bažnyčioje.

Santuokos palaiminimo data: 2016 [redacted]

Santuoką palaimino: [redacted]

A.V. _____ Data _____ Parašas _____

Vilniaus Šv. Pranciškaus Asyžiečio (Bernardinų) parapija, Vilniaus arkivyskupija
PRANEŠIMAS APIE SANTUOKOS PALAIMINIMĄ

Prašome padaryti įrašą parapijos krikštų registravimo knygoje.

Krikšto vieta: [redacted]

Krikšto data: [redacted]

Santuoką sudariusiojo vardas, pavardė: Aušra [redacted]

Tėvo vardas, pavardė: Česlovas [redacted]

Motinos vardas, pavardė: [redacted]

Santuoką sudarė su: [redacted]

Santuoka buvo palaiminta Vilniaus Šv. Pranciškaus Asyžiečio (Bernardinų) bažnyčioje.

Santuokos palaiminimo data: [redacted]

Santuoką palaimino: [redacted]

A.V. _____ Data _____ Parašas _____

P9 pav. Pranešimų parapijoms spausdinimo lango vaizdas