VILNIAUS UNIVERSITETAS
MATEMATIKOS IR INFORMATIKOS FAKULTETAS
PROGRAMŲ SISTEMŲ KATEDRA

Funkcinių taškų analizės metodų tyrimas

Function Point Analysis Methods Research

Magistro baigiamasis darbas

Atliko:	Šarūnas Gervė	(parašas)
Darbo vadovas:	Docentas dr. Sigitas Dapkūnas	(parašas)
Recenzentė:	Laura Savičienė	(parašas)

Vilnius – 2010

Funkcinių taškų analizės metodų tyrimas

Santrauka

Kad sugebėtume efektyviai valdyti programinės įrangos kūrimą, mes turime sugebėti matuoti programinę įrangą. Alan Albrecht 1979 metais IBM korporacijoje pasiūlė naudoti funkcinių taškų analizės metodą. Šis metodas matuoja programinės įrangos naudotojui suteikiamą funkcionalumą ir yra nepriklausomas nuo naudojamų programinės įrangos kūrimo technologijų. Šiuo metu yra jau 5 standartizuoti funkcinių taškų analizės metodai, kurie naudoja skirtingas taisykles programinės įrangos dydžio matavimui. Šiame darbe yra apibrėžiami vertinimo kriterijai ir pagal juos lyginami standartizuoti IFPUG, Mark II, COSMIC, NESMA ir FiSMA funkcinių taškų analizės metodai. Lyginant metodus yra nagrinėjami jų koncepciniai panašumai ir skirtumai, standartizuotumas, industrinių duomenų prieinamumas, konvertuojamumas, tinkamiausia taikymo fazė, matavimo taisyklių subjektyvumas, nefunkcinių naudotojų reikalavimų vertinimas, sertifikavimo ir skaičiavimo vadovų prieinamumas, metodus palaikantys programiniai įrankiai bei metodų taikymo sritis. Analizės rezultatai parodo, kad šiuo metu geriausia yra pasirinkti IFPUG arba COSMIC metodus.

Raktiniai žodžiai: programinės įrangos dydis, funkcinis programinės įrangos dydis, funkciniai taškai, funkcinių taškų analizė, funkcinių taškų analizės metodai, funkcinio dydžio matavimo metodų vertinimo kriterijai.

Function point analysis methods research

Summary

To effectively manage software development we must be able to measure software. Alan Albrecht in IBM Corporation in 1979 proposed the use of function point analysis method. This method measures functionality provided to the user by the software and is independent of the used software development technology. There are currently 5 standardized function point analysis methods, which use different rules for software size measurement. In this work estimation criteria are defined and used to compare standardized IFPUG, Mark II, COSMIC, NESMA and FiSMA function point analysis methods. During the comparison of methods their conceptual similarities and differences, standardization, availability of industry data, convertibility, the most suitable usage phase, subjectivity of measurement rules, estimation of non-functional user requirements, availability of certification and counting practices manuals, supporting software tools and the scope of methods applicability are examined. The analysis results show that the best choice is IFPUG or COSMIC methods.

Keywords: software size, functional software size, function points, function point analysis, function point analysis methods, functional size measurement methods estimation criteria.

TURINYS

Įvadas

Yra labai svarbu žinoti, ar projektas vyksta pagal planą. Yra sakoma, kad negalima kontroliuoti to, ko negalima pamatuoti. Kadangi viso programinės įrangos gyvavimo ciklo metu yra labai sunku numatyti projekto būseną, reikia organizacijoje įgyvendinti ir visiems projektams taikyti esminių matų matavimo programą. Šie esminiai matai leis palyginti projektų esamą būseną su planuotąja ir įgalins pirmalaikį jų būsenų neatitikimų nustatymą. Matai pašalina netikėtumus informuodami mus, kai projektas artėja prie tam tikrų problemų dar prieš joms įvykstant [EDB+05].

Norint organizacijoje vykdyti sėkmingą matavimo programą, paprastai yra minimi tokie kritiniai sėkmės faktoriai [IFPUG02]:

- **tinkamas nuolatinis marketingas**. Naujos matavimo programos pristatymas organizacijoje pakeis daugelio darbuotojų vykdomus darbo procesus. Tik vienintelis matavimų ir jų naudos pristatymas yra nepakankamas. Organizacijoje reikia pastoviai vykdyti marketingą;
- **prieigos prie informacijos užtikrinimas**. Surinkti matavimo duomenys turi būti išanalizuoti ir pateikiami visiems matavime dalyvaujantiems, ir suinteresuotiems asmenims;
- **išoriniai apmokymai ir konsultacijos**. Ekspertai gali padėti išvengti įvairių skaičiavimo nesusipratimų, todėl nereikia atmesti konsultantų pagalbos;
- **matų komplektas**. Kuriant sistemas reikia naudoti daugiau nei vieną matą;
- **duomenų tikslumas ir neprieštaringumas**. Visi surenkami matavimo duomenys turi būti tikslūs ir neprieštaringi;
- **matavimo programos integracija**. Sėkminga matavimų programa yra tokia programa, kuri tampa kasdienių organizacijos procesų dalimi. Matavimas neturi likti neįvertintu programinės įrangos kūrimo gyvavimo ciklo procesu.

Kad sugebėtume efektyviai numatyti ir valdyti programinės įrangos kūrimą, mes turime sugebėti matuoti programinę įrangą. Tokiu pat būdu, kaip centimetrai ir gramai suteikia mums informacijos apie fizinio, realaus pasaulio objektų dydį ir masę, programinės įrangos matai informuoja mus apie įvairias programinės įrangos charakteristikas [Vic03]. Yra daug įvairių programinės įrangos ir programinės įrangos kūrimo projektų charakteristikų, kurias galime išmatuoti, kaip, pavyzdžiui, programų sistemos dydis, sudėtingumas, patikimumas ir pelningumas [LB06].

Sistemos pastoviai didėja ir tampa sudėtingesnės. Jas vis sunkiau kurti ir analizuoti. Kodavimo įrankių tobulėjimas leidžia programinės įrangos kūrėjams lengviau pagaminti didesnius programinės

įrangos kiekius, kad patenkintų vis didėjančius vartotojų norus. Kadangi sistemos didėja, turi būti naudojamas kažkoks metodas, kuriuo būtų galima nustatyti sistemos dydį [Lon05]. Dydis yra vienas iš fundamentalių programinės įrangos matų.

Viena iš pagrindinių programinės įrangos kūrimo nesėkmės priežasčių yra nesugebėjimas tiksliai nustatyti programinės įrangos dydžio. Dydis yra kritinis faktorius kaštų, grafiko ir darbo sąnaudų nustatyme. Blogas dydžio apskaičiavimas paprastai būna pagrindinė priežastis kainos viršijime ir nesutilpime į darbų grafiką [AFS00].

Tradicinis programinės įrangos dydžio matas yra kodo eilučių skaičius. Kodo eilučių skaičius matuoja fizinį programinės įrangos dydį. Matuojant produktyvumą, kodo eilučių skaičiaus ar kokio kito fizinio dydžio matas nėra tinkamas. Fizinio dydžio matai turi mažai bendro su programinės įrangos teikiama nauda, o daugiau su pačia jos kūrimo technologija. Klientams rūpi funkcionalumas, o ne kiek kodo eilučių reikėjo jam sukoduoti ir derinti. Todėl papildomai reikia kažkokių kitų programinės įrangos dydžio skaičiavimo matų [LB06].

Alan Albrecht pasiūlė naudoti funkcinius taškus (angl. function points) IBM korporacijoje 1979 metais. Tai buvo pastangos įveikti sunkumus, susijusius su kodo eilučių skaičiaus, kaip programinės įrangos dydžio, matu, ir padėti kuriant mechanizmą, skirtą numatyti darbo sąnaudas, reikalingas kuriant programinę įrangą. Funkciniai taškai yra programinės įrangos funkcinio dydžio matavimo vienetai. Funkcinių taškų analizė (angl. function point analysis) buvo pirmas reikšmingas ir vis dar plačiausiai naudojamas funkcinio dydžio matavimas. Tai buvo pagrindas vėlesniems funkcinio dydžio matavimams [LB06].

Pagrindinė funkcinių taškų analizės nauda yra ta, kad ji izoliuoja sistemos dydžio matavimą nuo aplinkos faktorių [Vic03]. Kadangi funkcinių taškų analizė matuoja sistemas iš funkcinės perspektyvos, ji yra nepriklausoma nuo naudojamos technologijos. Nepriklausomai nuo programavimo kalbos, kūrimo metodo ar panaudotos techninės įrangos platformos, funkcinių taškų skaičius sistemai liks pastovus. Funkcinių taškų analizė gali būti naudojama nustatyti, ar įrankis, aplinka, programavimo kalba yra produktyvesnė palyginti su kitomis organizacijos viduje ar tarp kitų organizacijų. Tai yra viena iš didžiausių funkcinių taškų analizės vertybių [Lon05].

Funkciniai taškai ir juos naudojantys metodai yra pastoviai tobulinami, todėl buvo sukurta daug įvairių funkcinių taškų analizės metodų variantų, siekiant praplėsti originalaus metodo tinkamumą realaus laiko ir pilnoms algoritmų sistemoms [GDY05a]. Tarp tokių metodų yra 3D funkciniai taškai, savybių taškai (angl. feature points), objektiškai orientuoti funkciniai taškai ir t.t.

Šiuo metu egzistuoja jau 5 standartizuoti funkcinio dydžio matavimo metodai: IFPUG, Mark II, COSMIC, NESMA ir FiSMA.

Magistro darbo tikslas yra ištirti dabartinius standartizuotus funkcinių taškų analizės metodus, parodant jų skirtumus ir panašumus, bei nustatant, kokiais aspektais kiekvienas iš jų yra pranašesnis vienas nuo kito, kurį iš jų šiuo metu būtų geriausia naudoti programinės įrangos kūrimu užsiimančiai organizacijai. Kad galėtume tai nustatyti, tam mums reikalingi šiems metodams įvertinti ir palyginti tinkami kriterijai.

Pagal įvairius apžvelgtus šaltinius šiame darbe yra suformuluojami galimi vertinimo metodų pasirinkimo kriterijai. Atsižvelgiant į juos, yra apibrėžiami nagrinėjamų funkcinio dydžio matavimo metodų vertinimo kriterijai, nurodomas jų tinkamumas ir aktualumas tiriamiems standartizuotiems metodams.

Galiausiai pagal apibrėžtus kriterijus yra atliekamas kiekvieno standartizuoto IFPUG, Mark II, COSMIC, NESMA ir FiSMA funkcinio dydžio matavimo metodo vertinimas.

1. Funkcinio dydžio matavimas

1.1. Funkcinių taškų analizės metodai

Funkciniai taškai yra vieni populiariausių programinės įrangos matų pasaulyje. Nors jie pagrindinai skirti vertinti programinės įrangos kūrimo ir priežiūros projektų dydžiams, bet jie pritaikomi ir kitiems vertinimams. Jie taikomi kūrimo ir palaikymo kaštų vertinimui, proceso gerinimo analizei, kokybės vertinimui ir matavimui, ir pan. Pasaulyje vis daugiau organizacijų ir valstybinių institucijų pradeda vykdyti funkcinių taškų taikymo programas, todėl sertifikuotų funkcinių taškų vertintojų paklausa didėja žymiai sparčiau negu kitų programinės įrangos specialistų. Sertifikavimas nėra įmanomas be išbaigtų ir stabilių funkcinių taškų analizės skaičiavimo taisyklių [IFPUG00].

Šiuo metu yra išleisti 5 funkcinio dydžio matavimo metodų standartai. Kiekvienas iš šių metodų naudoja skirtingus programinės įrangos funkcinio dydžio matavimo būdus. Toliau yra detaliau apžvelgiamas kiekvienas iš šių penkių standartizuotų funkcinių taškų analizės metodų.

1.1.1. IFPUG funkcinių taškų analizės metodas

Tarptautinė funkcinių taškų naudotojų grupė (angl. International Function Point User Group) įsikūrė 1986 metais ir išleido nemažai funkcinių taškų skaičiavimo mokomųjų medžiagų. IFPUG ir toliau be paliovos plečiasi. Nors daugiausiai IFPUG metodas yra naudojamas Šiaurės Amerikoje, jis vis labiau plečiasi po visą pasaulį [Lon01].

Visas IFPUG funkcinių taškų analizės metodas susideda iš tokių žingsnių [GH01]:

1. nustatyti funkcinių taškų skaičiavimo tipą;
2. nustatyti skaičiavimo apimtį ir programinės įrangos ribas;
3. nustatyti visas duomenų funkcijas (vidiniai loginiai failai ir išoriniai interfeiso failai) ir jų sudėtingumus;
4. nustatyti visas transakcines funkcijas (išoriniai įvedimai, išoriniai išvedimai ir išorinės užklausos) ir jų sudėtingumus;
5. apskaičiuoti nekoreguotų funkcinių taškų reikšmę;
6. apskaičiuoti reikšmių koregavimo faktorių, paremtą 14 bendrųjų sistemos charakteristikų;
7. apskaičiuoti koreguotų funkcinių taškų reikšmę.

1.1.1.1. Funkcinių taškų skaičiavimo tipo nustatymas

Pirmas IFPUG funkcinių taškų skaičiavimo metodo žingsnis yra nustatyti taikomą funkcinių taškų skaičiavimo tipą. Trys galimi funkcinių taškų skaičiavimo tipai yra šie [GH01]:

- **naujo kuriamo projekto funkcinių taškų skaičiavimo tipas**. Juo matuojamas naudotojui suteikiamas funkcionalumas su pirmu programinės įrangos įdiegimu. Skaičiuojamas kuriamos programos funkcionalumas ir reikiamas funkcionalumas duomenų, esančių senuose duomenų failuose, konvertavimui į naujus duomenų failus;
- **papildančio projekto funkcinių taškų skaičiavimo tipas**. Juo matuojamos modifikacijos jau sukurtom programom ir skaičiuojamas naudotojui suteikiamas funkcionalumas, pridedant naujas, šalinant senas ir keičiant egzistuojančias funkcijas. Šiame funkcinių taškų skaičiavimo tipe galimas ir konvertavimo funkcionalumas;
- **programinės įrangos funkcinių taškų skaičiavimo tipas**. Juo matuojama jau įdiegta programinė įranga. Vertinamas esamos programinės įrangos suteikiamas funkcionalumas naudotojui.

1.1.1.2. Skaičiavimo apimties ir programinės įrangos ribų nustatymas

Skaičiavimo apimtis yra nustatoma pagal skaičiavimo tikslus. Ji apibūdina sistemas, programas ar programų dalis, kurios bus matuojamos. Tai gali būti perkamo programinio paketo teikiamos funkcijos, trečiųjų šalių gaminamos programos funkcijos, specifiniams tikslams atlikti skirtos programos funkcijos ir pan.

Programinės įrangos ribos apibrėžiamos, kaip riba tarp matuojamos programos ir tarp kitų išorinių programų [GH01]. Yra nustatoma, kokį funkcionalumą turi matuojama programa ir kokį kitos išorinės programos. Programinės įrangos ribos yra nustatomos iš naudotojo perspektyvos.

1.1.1.3. Duomenų funkcijų ir jų sudėtingumų nustatymas

Yra dviejų tipų duomenų funkcijos [GH01]:

- **vidinis loginis failas (VLF)**. Tai programos ribose palaikomų logiškai susijusių duomenų ar valdymo informacijos grupė. Pagrindinis VLF tikslas yra saugoti matuojamos programos duomenis, naudojamus vieno ar kelių elementarių procesų metu;
- **išorinis interfeiso failas (IIF)**. Tai logiškai susijusių duomenų ar valdymo informacijos grupė, į kurią kreipiasi programa, bet yra palaikoma kitos programos ribose. Pagrindinis IIF tikslas yra saugoti duomenis, į kuriuos kreipiamasi vieno ar kelių matuojamos programos

ribose vykdomų elementarių procesų metu. VLF apskaičiuotas vienoje programoje yra IIF kitos programos, naudojančios tą VLF, ribose.

Nustačius matuojamos programinės įrangos vidinių loginių failų ir išorinių interfeiso failų kiekį, reikia nustatyti kiekvieno jų sudėtingumą. VLF ir IIF sudėtingumas priklauso nuo dviejų faktorių [GH01]:

- **duomenų elementų tipai (DET)**. Tai unikalūs naudotojam suprantami, nesikartojantys laukai ar atributai;
- **įrašų elementų tipai (ĮET)**. Tai naudotojam suprantami duomenų elementų, esančių VLF ir IIF, pogrupiai.

Kiekvienam nustatytam VLF ir IIF turi būti priskirtas funkcinis sudėtingumas, priklausantis nuo duomenų elementų tipų ir įrašų elementų tipų, susijusių su VLF ar IIF, skaičiaus. Galimos sudėtingumo reikšmės: žemas, vidutinis arba aukštas. Jos priskiriamos pagal DET ir ĮET kiekį naudojant VLF ir IIF sudėtingumo matricos 1 lentelę.

1 lentelė. VLF ir IIF sudėtingumo matrica [BD08]

ĮET	**DET**		
	1-19	20-50	>50
1	Žemas	Žemas	Vidutinis
2-5	Žemas	Vidutinis	Aukštas
>5	Vidutinis	Aukštas	Aukštas

1.1.1.4. Transakcinių funkcijų ir jų sudėtingumų nustatymas

Transakcinės funkcijos yra skirstomos į tris grupes [BD08]:

- **išorinis įvedimas (IĮ)**. Tai elementarus procesas, kuris apdoroja duomenis ar valdymo informaciją, kuri atvyksta iš programos išorės. Pagrindinis IĮ tikslas yra palaikyti vieną ar daugiau VLF arba keisti sistemos elgseną;
- **išorinė užklausa (IU)**. Tai elementarus procesas, kuris siunčia duomenis ar valdymo informaciją į programos išorę. Pagrindinis IU tikslas yra pateikti informaciją naudotojui per duomenų išrinkimą ar valdymo informaciją iš VLF ar IIF. Vykdymo logika neturi jokių matematinių formulių ar skaičiavimų, nesukuria jokių išvedamų duomenų. Joks VLF nėra palaikomas bei nekeičiama sistemos elgsena;

- **išorinis išvedimas (II)**. Tai elementarus procesas, kuris siunčia duomenis ar valdymo informaciją į programos išorę. Pagrindinis II tikslas yra pateikti informaciją naudotojui per vykdymo logiką. Vykdymo logika turi turėti bent vieną matematinę formulę ar skaičiavimą, sukurti išvedamus duomenis, palaikyti vieną ar daugiau VLF arba keisti sistemos elgseną.

Nustačius matuojamos programinės įrangos išorinių įvedimų, išorinių užklausų ir išorinių išvedimų kiekį, reikia nustatyti kiekvieno jų sudėtingumą. IĮ, IU ir II sudėtingumas priklauso nuo programos ribas kertančių DET ir [GH01]:

- **kreipimosi failų tipai (KFT)**. Tai bendras skaitomų ar palaikomų VLF ir skaitomų IIF, vykdant IĮ ir II transakcijas, skaičius. Vykdant IU transakcijas skaičiuojamas tik skaitomų VLF ir IIF skaičius.

Kiekvienam nustatytam IĮ, IU ir II turi būti priskirtas funkcinis sudėtingumas, priklausantis nuo duomenų elementų tipų ir kreipimosi failų tipų, susijusių su IĮ, IU ir II, skaičiaus. Galimos sudėtingumo reikšmės: žemas, vidutinis arba aukštas. Jos priskiriamos pagal DET ir KFT kiekį naudojant IĮ sudėtingumo matricos 2 lentelę, bei IU ir II sudėtingumo matricos 3 lentelę.

2 lentelė. IĮ sudėtingumo matrica [BD08]

KFT	**DET**		
	1-4	5-15	>15
0-1	Žemas	Žemas	Vidutinis
2	Žemas	Vidutinis	Aukštas
>2	Vidutinis	Aukštas	Aukštas

3 lentelė. IU ir II sudėtingumo matrica [BD08]

KFT	**DET**		
	1-5	6-19	>19
0-1	Žemas	Žemas	Vidutinis
2-3	Žemas	Vidutinis	Aukštas
>3	Vidutinis	Aukštas	Aukštas

1.1.1.5. Nekoreguotų funkcinių taškų skaičiavimas

Nustačius duomenų ir transakcinių funkcijų kiekį ir jų sudėtingumus, skaičiuojami nekoreguoti funkciniai taškai (NFT). NFT skaičiuojami naudojant IFPUG nekoreguotų funkcinių taškų 4 lentelės koeficientais, susumuojant visų funkcijų sudėtingumų lygių įverčius:

NFT = Σ(VLF × S) + Σ(IIF × S) + Σ(IĮ × S) + Σ(IU × S) + Σ(II × S), kur

- NFT – nekoreguoti funkciniai taškai;
- VLF – vidinis loginis failas;
- IIF – išorinis interfeiso failas;
- IĮ – išorinis įvedimas;
- IU – išorinė užklausa;
- II – išorinis išvedimas;
- S – duomenų arba transakcinės funkcijos sudėtingumas iš atitinkamos sudėtingumo matricos;
- „Σ“ simbolis reiškia visų duomenų ir transakcinių funkcijų sumą.

4 lentelė. IFPUG nekoreguotų funkcinių taškų lentelė [GH01]

Funkcijos	**Funkcijų sudėtingumo lygiai**		
	Žemas	**Vidutinis**	**Aukštas**
Vidinis loginis failas	×7	×10	×15
Išorinis interfeiso failas	×5	×7	×10
Išorinis įvedimas	×3	×4	×6
Išorinė užklausa	×3	×4	×6
Išorinis išvedimas	×4	×5	×7

1.1.1.6. Reikšmių koregavimo faktoriaus skaičiavimas

Kad galėtume apskaičiuoti koreguotus funkcinius taškus, pirma reikia nustatyti reikšmių koregavimo faktorių (RKF). Šiuo žingsniu į programos vertinimą įtraukiami ir nefunkciniai (kokybiniai) reikalavimai. Skaičiuojant RKF yra atsižvelgiama į 14 bendrųjų sistemos charakteristikų (BSC). IFPUG metodas naudoja 14 pirmųjų BSC parodytų 1 priede. Kiekvienos BSC įtaka kuriamai sistemai yra įvertinama nuo 0 (jokios įtakos) iki 5 (didelė įtaka) balų. Bendras įtakos laipsnis (BĮL) apskaičiuojamas susumuojant visų 14 BSC įtakų vertes. Reikšmių koregavimo faktorius tada apskaičiuojamas taip:

RKF = (BĮL × 0,01) + 0,65, kur:

- RKF – reikšmių koregavimo faktorius;
- BĮL – bendras įtakos laipsnis.

RKF maksimali reikšmė gali pasiekti 1,35, o minimali 0,65, todėl ji gali koreguoti nekoreguotus funkcinius taškus +/- 35% [BD08].

1.1.1.7. Koreguotų funkcinių taškų skaičiavimas

Šiame etape skaičiuojami koreguoti funkciniai taškai (KFTa). Kaip jau minėta, yra trys funkcinių taškų skaičiavimo tipai ir kiekvienam jų koreguoti funkciniai taškai apskaičiuojami naudojant skirtingas formules.

Naujo kuriamo projekto funkciniai taškai apskaičiuojami pagal tokią formulę:

KPFT = (NFT + DKFT) × RKF, kur:

- KPFT - kuriamo projekto funkcinių taškų skaičius;
- NFT - nekoreguoti funkciniai taškai;
- DKFT - duomenų konvertavimo funkcinių taškų skaičius;
- RKF - reikšmių koregavimo faktorius.

Papildančio projekto funkciniai taškai apskaičiuojami taikant tokią formulę:

PPFT = [(NPF + MF + DKFT) × RKFPO] + (PF × RKFPR), kur:

- PPFT - papildančio projekto funkcinių taškų skaičius;
- NPF - nekoreguotų funkcinių taškų skaičius nuo naujai pridėtų papildančio projekto funkcijų;
- MF - nekoreguotų funkcinių taškų skaičius nuo modifikuotų papildančio projekto funkcijų;
- DKFT - duomenų konvertavimo funkcinių taškų skaičius;
- RKFPO - programos reikšmių koregavimo faktorius po papildančio projekto;
- PF - nekoreguotų funkcinių taškų skaičius nuo pašalintų papildančio projekto funkcijų;
- RKFPR - programos reikšmių koregavimo faktorius prieš papildantį projektą.

Programinės įrangos funkcinių taškų kiekis apskaičiuojamas pagal tokią formulę:

KFTa = NFT × RKF, kur:

- KFTa - programos koreguoti funkciniai taškai;
- NFT - programos nekoreguoti funkciniai taškai;
- RKF - reikšmių koregavimo faktorius.

1.1.2. Mark II funkcinių taškų analizės metodas

Mark II metodą sukūrė Charles R. Symons 1988 metais Šis metodas daugiausiai naudojamas Jungtinėje Karalystėje. Mark II metodą sudaro tokie žingsniai [UKSMA98]:

1. nustatyti matavimo perspektyvą ir tikslus;
2. nustatyti matavimo ribas;
3. nustatyti logines transakcijas;
4. nustatyti ir kategorizuoti duomenų esybių tipus;
5. suskaičiuoti įvesties duomenų elementų tipus, duomenų esybių tipus, į kuriuos yra kreipiamasi, ir išvesties duomenų elementų tipus;
6. apskaičiuoti funkcinį dydį;
7. apskaičiuoti įtakos laipsnius;
8. apskaičiuoti techninio sudėtingumo koregavimo reikšmę;
9. apskaičiuoti koreguotus funkcinius taškus.

1.1.2.1. Matavimo perspektyvos ir tikslų nustatymas

Pirmiausia, pradedant naudoti Mark II metodą, reikia nustatyti matavimo ribas. Kad būtų galima nustatyti šias ribas, reikia nustatyti matavimo perspektyvą ir tikslus. Todėl pirmas žingsnis naudojant Mark II metodą ir yra nustatyti, iš kokios perspektyvos (verslo įmonės, projekto, programos) bus matuojama programinė įranga ar projektas, kokie bus matavimo tikslai.

1.1.2.2. Matavimo ribų nustatymas

Matavimo ribų nustatymas parodo, koks funkcionalumas bus įtrauktas į skaičiavimus, ir koks nebus. Matavimo ribos vaizduoja abstrakčią ribą tarp programinės įrangos ir jos naudotojo.

1.1.2.3. Loginių transakcijų nustatymas

Nustačius matavimo ribas, reikia korektiškai nustatyti naudojamas logines transakcijas. Loginės transakcijos yra žemiausio lygio programinės įrangos palaikomi verslo procesai. Kiekviena loginė transakcija susideda iš trijų elementų: įvedimų, apdorojimų ir išvedimų.

1.1.2.4. Duomenų esybių tipų nustatymas ir kategorizavimas

Šiame žingsnyje yra patartina nusipiešti esybių ryšių diagramą, kad būtų galima nustatyti pagrindines sistemos esybes, kurios mus labiausiai ir domina. Esybė (duomenų esybių tipas) apibrėžiama, kaip esminis aktualus naudotojui objektas, apie kurį yra saugoma informacija. Ryšis tarp esybių, kuris turi atributus, taip pat yra esybė. Nepagrindinės esybės yra nekintančios ir turinčios tik kelis atributus. Visi kreipimaisi į nepagrindines esybes yra laikomi, kaip vienas bendras

kreipimasis į sistemos esybę. Taigi šiame žingsnyje yra nustatomos ir kategorizuojamos pagrindinės ir nepagrindinės esybės.

1.1.2.5. Įvesties duomenų elementų tipų, duomenų esybių tipų, į kuriuos yra kreipiamasi, ir išvesties duomenų elementų tipų apskaičiavimas

Šiame žingsnyje kiekvienai loginei transakcijai apskaičiuojami [UKSMA98]:

- **įvesties duomenų elementų tipai**. Šie tipai atkeliauja iš sistemos išorės ir keičia sistemos būseną. Jie siejami su duomenų saugojimu ir patvirtinimu (angl. validation);
- **duomenų esybių tipai, į kuriuos yra kreipiamasi**. Esybių tipai, į kuriuos kreipiasi loginė transakcija;
- **išvesties duomenų elementų tipai**. Šie tipai siunčiami į sistemos išorę, kad naudotojas galėtų juos pamatyti arba panaudoti. Jie siejami su duomenų apipavidalinimu ir vaizdavimu.

1.1.2.6. Funkcinio dydžio apskaičiavimas

Funkcinis dydis (funkcinių taškų indeksas) yra visų loginių transakcijų įvesties duomenų elementų tipų ($T_į$), duomenų esybių tipų, į kuriuos yra kreipiamasi, (T_e) ir išvesties duomenų elementų tipų (T_i) suma. Taigi funkcinių taškų indeksas (FTI) programai apskaičiuojamas taip:

$FTI = S_į \times \Sigma T_į + S_e \times \Sigma T_e + S_i \times \Sigma T_i$, kur:

- FTI – funkcinių taškų indeksas;
- $T_į$ - įvesties duomenų elementų tipas;
- T_e - duomenų esybių tipas, į kurį yra kreipiamasi;
- T_i - išvesties duomenų elementų tipas;
- $S_į$, S_e, S_i – pramonės standartiniai vidurkiai;
- „Σ“ simbolis reiškia visų loginių transakcijų sumą.

Pramonės standartiniai įvesties duomenų elementų tipo, duomenų esybių tipo, į kurį yra kreipiamasi, ir išvesties duomenų elementų tipo vidurkių svoriai yra atitinkamai tokie (jie gali būti kalibruojami įmonės viduje):

- $S_į = 0,58$;
- $S_e = 1,66$;
- $S_i = 0,26$.

Programinės įrangos pakeitimų dydžiai skaičiuojami taikant tokias formules:

FTI pokyčio = Pridėtas + Modifikuotas + Pašalintas funkcionalumas;

FTI po pokyčio = FTI prieš pokytį + Pridėtas - Pašalintas funkcionalumas.

1.1.2.7. Įtakos laipsnių apskaičiavimas

Kaip ir IFPUG metode čia skaičiuojama bendrųjų sistemos charakteristikų įtaka kuriamai sistemai. Įtakos laipsnis (ĮL) skaičiuojamas kiekvienai iš 19 charakteristikų ir vertinamas skalėje nuo 0 (jokios įtakos) iki 5 (didelė įtaka). Mark II metode naudojamos tos pačios 14 IFPUG bendrųjų sistemos charakteristikų praplečiant jas penkiomis naujomis charakteristikomis. Jas visas galima pamatyti 1 priede. Esant reikalui, Mark II metode leidžiama pridėti savas naujas bendrąsias sistemos charakteristikas.

1.1.2.8. Techninio sudėtingumo koregavimo reikšmės apskaičiavimas

Susumavę visų bendrųjų sistemos charakteristikų įtakos laipsnius, gauname bendra įtakos laipsnį (BĮL). Tada techninio sudėtingumo koregavimo (TSK) reikšmė apskaičiuojama pagal tokią formulę:

TSK = (BĮL × K) + 0,65, kur:

- TSK – techninio sudėtingumo koregavimo reikšmė;
- BĮL – bendras įtakos laipsnis;
- K - dabartinis pramonės vidurkio koeficientas, kuris yra lygus 0,005 (reikšmė taip pat gali būti kalibruojama įmonės viduje).

TSK reikšmė varijuoja tarp 0,65 ir 1,125, todėl ji gali koreguoti galutinius funkcinius taškus nuo -35% iki +12,5%.

1.1.2.9. Koreguotų funkcinių taškų apskaičiavimas

Galiausiai koreguoti funkciniai taškai (KFTa) apskaičiuojami taip:

KFTa = FTI × TSK, kur:

- KFTa - koreguoti funkciniai taškai;
- FTI - funkcinių taškų indeksas;
- TSK – techninio sudėtingumo koregavimo reikšmė.

1.1.3. COSMIC funkcinių taškų analizės metodas

COSMIC konsorciumas buvo įkurtas 1998 metais, kaip savanoriška programinės įrangos matavimo ekspertų iš įvairių pasaulio šalių organizacija. 1999 metų pabaigoje jie išleido COSMIC pilnų funkcinių taškų metodo 2.0 versijos vadovą ir padarė jį viešai prieinamą pasauliniame tinkle. COSMIC konsorciumas 2007 metais pakeitė savo metodo pavadinimą į tiesiog COSMIC metodą ir išleido naują metodo 3.0 versijos vadovą [BD08]. COSMIC metodas sudarytas iš trijų pagrindinių

fazių: matavimo strategijos fazė, modelio kūrimo fazė ir matavimo fazė. Šios trys fazės sudaro tokius COSMIC metodo naudojimo žingsnius [CSMIC09]:

1. apibrėžti matavimo tikslus;
2. apibrėžti matavimo apimtį;
3. nustatyti funkcinius naudotojus ir matavimo ribas;
4. nustatyti grūdėtumo lygį;
5. taikyti bendrą programinės įrangos modelį;
6. nustatyti funkcinius procesus;
7. nustatyti dominančius objektus, duomenų grupes ir atributus;
8. nustatyti duomenų judėjimus;
9. įvertinti duomenų judėjimus ir agreguoti matavimo rezultatus;
10. paruošti matavimo rezultatų ataskaitą.

1.1.3.1. Matavimo tikslų apibrėžimas

Apibrėžiant matavimo tikslus, yra nustatoma, kam yra reikalingas matavimas, ir kur bus panaudoti matavimo metu gauti rezultatai.

1.1.3.2. Matavimo apimties apibrėžimas

Šiame žingsnyje yra nustatoma, kurie funkciniai naudotojų reikalavimai (FNR) bus įtraukti į matavimo procesą. Funkciniai naudotojų reikalavimai yra naudotojų reikalavimų poaibis, kuris apibūdina, ką programa turės daryti užduočių ir paslaugų terminais.

1.1.3.3. Funkcinių naudotojų ir matavimo ribų nustatymas

Pagal matuojamus FNR yra nustatomi funkciniai naudotojai. Funkcinis naudotojas, tai naudotojas, kuris yra programinės įrangos funkcinių naudotojo reikalavimų duomenų siuntėjas ir/arba numatytas gavėjas.

Nustačius funkcinius naudotojus, reikia nustatyti matavimo ribas. Riba yra apibrėžiama kaip abstraktus interfeisas tarp matuojamos programinės įrangos ir jos funkcinių naudotojų.

1.1.3.4. Grūdėtumo lygio nustatymas

Pradinėse programinės įrangos kūrimo projekto fazėse funkciniai naudotojų reikalavimai yra apibrėžiami aukštame abstrakcijos lygyje. Tęsiantis projektui FNR yra detalizuojami, mažinant abstrakcijos lygį ir atskleidžiant daugiau detalių. Šie skirtingi FNR detalumo lygiai yra žinomi kaip skirtingi grūdėtumo lygiai. Taigi reikia nustatyti matuojamų programinės įrangos artefaktų grūdėtumo lygį.

1.1.3.5. Bendro programinės įrangos modelio taikymas

COSMIC bendras programinės įrangos modelis yra taikomas kiekvienos atskiros programinės įrangos dalies, kuriai nustatytos skirtingos matavimo ribos, funkciniams naudotojų reikalavimams. COSMIC bendro programinės įrangos modelio taikymas reiškia, kiekvienam funkciniam naudotojui žinomų iššaukiamųjų įvykių (angl. triggering events) nustatymą, ir tada atitinkamų dominančių objektų, duomenų grupių ir duomenų judėjimų, kurie turi būti numatyti, kad reaguotų į tuos įvykius, nustatymą.

1.1.3.6. Funkcinių procesų nustatymas

Šiame žingsnyje iš FNR nustatomi matuojamos programinės įrangos funkciniai procesai. Funkcinis procesas tai elementarus FNR rinkinio komponentas, susidedantis iš unikalaus, darnaus ir nepriklausomai vykdomo duomenų judėjimų rinkinio. Funkcinis procesas yra iššaukiamas funkcinio naudotojo duomenų judėjimo, kuris praneša programai, kad funkcinis naudotojas aptiko iššaukiamąjį įvykį. Funkcinis procesas baigiasi, kai yra įvykdomi visi veiksmai, reikalinga reaguoti į iššaukiamąjį įvykį.

Iššaukiamasis įvykis, tai įvykis, kuris priverčia programinės įrangos naudotoją inicijuoti vieną arba daugiau funkcinių procesų.

1.1.3.7. Dominančių objektų, duomenų grupių ir atributų nustatymas

Šio žingsnio vykdymo metu yra nustatomos duomenų grupės, į kurias kreipiasi matuojama programinė įranga. Kad nustatytume duomenų grupes yra naudinga prieš tai nustatyti dominančius objektus ir jų atributus.

Dominantis objektas, tai bet koks funkciniuose naudotojų reikalavimuose nustatytas objektas, dominantis naudotoją. Tai gali būti bet koks fizinis objektas, taip pat bet koks konceptualus objektas arba konceptualaus objekto dalis funkcinio naudotojo pasaulyje, apie kurį programinė įranga turi apdoroti ir/arba saugoti duomenis.

Duomenų grupė yra aiškus, netuščias, nesurikiuotas ir nedubliuojamas duomenų atributų rinkinys, kuriame kiekvienas įtrauktas duomenų atributas apibūdina to paties dominančio objekto įvairius aspektus.

Duomenų atributas yra mažiausias informacijos paketas nustatytoje duomenų grupėje, talpinantis prasmingą informaciją iš programinės įrangos funkcinių naudotojų reikalavimų perspektyvos.

1.1.3.8. Duomenų judėjimų nustatymas

Šiame COSMIC metodo žingsnyje kiekvienam funkciniam procesui nustatomi duomenų judėjimai. Duomenų judėjimas, tai bazinis funkcinis komponentas, kuris perkelia vieną duomenų grupę. Yra keturi duomenų judėjimo tipai [CSMIC09]:

- **įėjimas (Į)**. Tai duomenų judėjimas, kuris perkelia duomenų grupę iš funkcinio naudotojo į duomenų grupės reikalaujantį funkcinį procesą;
- **išėjimas (I)**. Tai duomenų judėjimas, kuris perkelia duomenų grupę iš funkcinio proceso į duomenų grupės reikalaujantį funkcinį naudotoją;
- **skaitymas (S)**. Tai duomenų judėjimas, kuris perkelia duomenų grupę iš pastovios atminties į duomenų grupės reikalaujantį funkcinį procesą;
- **rašymas (R)**. Tai duomenų judėjimas, kuris perkelia duomenų grupę, esančią funkciniame procese, į pastoviąją atmintį.

Visas duomenų apdorojimas, vykstantis funkciniame procese, yra susiejamas su keturiais duomenų judėjimo tipais. Yra sutarta, kad funkcinio proceso duomenų judėjimais turi būti skaičiuojami ir funkcinio proceso duomenų apdorojimai.

1.1.3.9. Duomenų judėjimų įvertinimas ir matavimo rezultatų agregavimas

Kiekvienas duomenų judėjimas yra įvertinamas 1 CFT (cosmic funkciniu tašku). Šiame žingsnyje agreguojami visų nustatytų duomenų judėjimų rezultatai į vieną funkcinio dydžio reikšmę. Vieno funkcinio proceso dydis apskaičiuojamas sudedant visus jo duomenų judėjimus:

Dydis(funkcinis procesas) = Σ dydis(Įėjimai) + Σ dydis(Išėjimai) + Σ dydis(Skaitymai) + Σ dydis(Rašymai)

Bendrą programos funkcinį dydį tada galime apskaičiuoti susumuojant visus funkcinius procesus:

Bendras funkcinis dydis = Σ Dydis(funkciniai procesai)

Bet kokiam funkciniam procesui FNR pokyčio funkcinis dydis apskaičiuojamas sumuojant visus pridėtų, modifikuotų arba pašalintų duomenų judėjimų dydžius:

Pokyčio Dydis(funkcinis procesas) = Σ dydis(pridėti duomenų judėjimai) + Σ dydis(modifikuoti duomenų judėjimai) + Σ dydis(pašalinti duomenų judėjimai)

Bendrą programos pokyčių funkcinį dydį gausime susumuojant visus keičiamus funkcinius procesus:

Pokyčių bendras funkcinis dydis = Σ Pokyčio Dydis(funkciniai procesai)

1.1.3.10. Matavimo rezultatų ataskaitos paruošimas

Kartu su gautais matavimo rezultatais turi būti dokumentuojami tokie kiekvieno matavimo atributai [CSMIC09]:

- identifikaciniai matuojamos programinės įrangos duomenys (pavadinimas, versijos numeris ar konfigūracijos numeris);
- FNR nustatymui panaudoti informacijos šaltiniai;
- programinės įrangos sritis;
- matavimo tikslai;
- matavimo apimties aprašymas;
- programinės įrangos funkciniai naudotojai;
- FNR grūdėtumo lygis;
- projekto gyvavimo ciklo etapas, kuriame buvo atlikti matavimai;
- naujai sukurto funkcionalumo ar pakeisto funkcionalumo matavimo požymis;
- vertintojo identifikaciniai duomenys ir turimi COSMIC metodo naudojimo sertifikatai.

1.1.4. FiSMA funkcinio dydžio matavimo metodas

Pirmoji FiSMA funkcinio dydžio matavimo metodo, kuris dar tuo metu vadinosi Laturi metodu, versija buvo išleista 1991 metais. Šis metodas ir toliau buvo tobulinamas, o didžiausią įtaką jam turėjo Suomijos programinės įrangos matavimo asociacijos, kuri ir perėmė šį metodą į savo rankas, įkūrimas 1996 metais. Metodo pavadinimas buvo pakeistas į FiSMA funkcinio dydžio matavimo metodą [BD08]. FiSMA matavimo metodas susideda iš tokių žingsnių [FSMA07]:

1. surinkti dokumentaciją ir programinės įrangos kūrimo artefaktus;
2. nustatyti matavimo apimtį;
3. nustatyti funkcinius naudotojo reikalavimus;
4. nustatyti bazinius funkcinius komponentus;
5. klasifikuoti bazinius funkcinius komponentus;
6. priskisti reikšmes baziniams funkciniams komponentams;
7. apskaičiuoti funkcinį dydį;
8. dokumentuoti matavimo rezultatus.

1.1.4.1. Dokumentacijos ir programinės įrangos kūrimo artefaktų surinkimas

Šiame žingsnyje surenkama dokumentacija ir programinės įrangos kūrimo artefaktai, kad būtų galima apibūdinti funkcinius naudotojų reikalavimus kuriamai arba jau sukurtai programinei įrangai. Tokie artefaktai gali būti: preliminarūs naudotojų reikalavimai, naudojimo vadovai, esybių ryšių diagramos, įvairios ataskaitos, duomenų tėkmės diagramos ir pan. Jais galima užduočių ir paslaugų terminais apibūdinti, ką vykdys programa, nepriklausomai nuo bet kokių kokybinių ir techninių reikalavimų.

1.1.4.2. Matavimo apimties nustatymas

Šiame žingsnyje nustatomi matavimo tikslai ir pagal juos apibrėžiama matavimo apimtis. Matavimo apimtis nurodo, kurie funkciniai naudotojo reikalavimai yra įtraukiami į kuriamos ar tobulinamos programinės įrangos matavimą.

1.1.4.3. Funkcinių naudotojo reikalavimų nustatymas

Šiame žingsnyje išrenkami tik tie naudotojo reikalavimai, kurie nusako, ką programa darys užduočių ir paslaugų terminais.

1.1.4.4. Bazinių funkcinių komponentų nustatymas

Dabar nustatytuose funkciniuose naudotojo reikalavimuose yra identifikuojami baziniai funkciniai komponentai (BFK). FiSMA metode yra apibrėžiamos septynios skirtingos BFK klasės [FSMA07]:

- **interaktyvios galutinio naudotojo navigacijos ir užklausų paslaugos (u).** Šios paslaugos apibrėžia interaktyvaus naudotojo interfeiso dalis, kuriose nevyksta pastovių duomenų, saugomų sistemoje, palaikymas. Palaikymas reiškia paslaugą, kurios metu duomenys yra atnaujinami, kuriami arba šalinami. Kiekvienos navigacijos ir užklausų paslaugos funkcinis dydis priklauso nuo BFK duomenų elementų skaičiaus ir nuo unikalių esybių, į kurias reikia kreiptis, skaičiaus;
- **interaktyvios galutinio naudotojo įvedimo paslaugos (į).** Šios paslaugos apibrėžia interaktyvaus naudotojo interfeiso dalis, kuriose vyksta programinės įrangos duomenų saugojimų palaikymas. Iš naudotojo perspektyvos, interaktyvios galutinio naudotojo paslaugos vykdo tas verslo užduotis, kurios keičia programinės įrangos duomenų turinį. Iš informacinės sistemos perspektyvos, galutiniai naudotojai manipuliuoja sistemos duomenimis naudodami interaktyvias galutinio naudotojo paslaugas. Įvedimo funkcijų funkcinis dydis priklauso nuo matuojamų skirtingų BFK duomenų elementų skaičiaus ir nuo reikiamų skaitymo ir rašymo kreipimųsi į unikalias esybes skaičiaus;

- **neinteraktyvios galutinio naudotojo išvedimo paslaugos (i).** Šios paslaugos apibrėžia naudotojo interfeiso dalis, kurios yra neinteraktyvios ir nepalaiko programinės įrangos duomenų. Išvedimo funkcijų funkcinis dydis priklauso nuo skirtingų BFK duomenų elementų skaičiaus ir nuo reikiamų skaitymo kreipimųsi į esybes skaičiaus;
- **interfeiso paslaugos kitoms programoms (įv).** Šios paslaugos apibrėžia visus automatinius duomenų perkėlimus, kurie persiunčia duomenis iš matuojamos programinės įrangos į kitas programas ar prietaisus. Šių funkcijų funkcinis dydis priklauso nuo matuojamų skirtingų BFK duomenų elementų skaičiaus ir nuo reikiamų skaitymo kreipimųsi į esybes skaičiaus;
- **interfeiso paslaugos iš kitų programų (iš).** Šios paslaugos apibrėžia visus automatinius duomenų perkėlimus, kurie priima kitų programų ar prietaisų siunčiamas duomenų grupes. Šių funkcijų funkcinis dydis priklauso nuo matuojamų skirtingų BFK duomenų elementų skaičiaus ir nuo reikiamų skaitymo ir rašymo kreipimųsi į esybes skaičiaus;
- **duomenų saugojimo paslaugos (d).** Šios paslaugos apibrėžia grupę susijusių ir gerai apibrėžtų realaus pasaulio duomenų, kuriems naudotojas programinės įrangos reikalauja atlikti duomenų saugojimą. Susijusių ir gerai apibrėžtų duomenų grupės dažniausiai yra vadinamos esybėmis, duomenų grupėmis, duomenų klasėmis arba dominančiais objektais, priklausomai nuo kūrimo aplinkoje naudojamos terminologijos. Saugojimo paslaugų funkcinis dydis priklauso nuo skirtingų duomenų elementų grupėje skaičiaus;
- **algoritminės ir valdymo paslaugos (a).** Šios paslaugos yra naudotojų apibrėžtos, nepriklausomos duomenų valdymo funkcijos. Jos gali turėti aritmetinių ir loginių operacijų. Algoritminių ir valdymo paslaugų funkcinis dydis priklauso nuo skirtingų atliekamų operacijų skaičiaus ir nuo skirtingų kintamųjų, reikalingų atlikti šias paslaugas, skaičiaus.

1.1.4.5. Bazinių funkcinių komponentų klasifikavimas

Šiame žingsnyje klasifikuojami visi nustatyti BFK į atitinkamus BFK tipus. Iš viso yra 28 skirtingi BFK tipai. Jie pateikti 5 lentelėje.

5 lentelė. Bazinių funkcinių komponentų tipai

BFK tipai	**Trumpas aprašas**
Interaktyvios galutinio naudotojo navigacijos ir užklausų paslaugos (u)	
Funkcijos nurodymai (u1)	Paslaugos, kurios suteikia unikaliai identifikuojamą vizualų būdą naudotojui nurodyti vykdomą specifinę paslaugą
Prisijungimo ir atsijungimo funkcijos (u2)	Valdo naudotojų prieigą ir neleidžia neteisėto naudojimo

BFK tipai	**Trumpas aprašas**
Funkcijos sąrašai (u3)	Paslaugos, kurios suteikia aibę iš anksto apibrėžtų alternatyvų, įgalinančių naudotoją nurodyti vykdomą specifinę paslaugą
Pasirinkimo sąrašai (u4)	Parodo galutiniam naudotojui sąrašą priimtinų parametrų reikšmių
Duomenų užklausos (u5)	Parodo galutiniam naudotojui specifinį saugomų duomenų turinį
Generavimo indikatoriai (u6)	Padeda naudotojui parengti duomenis ir valdymo informaciją tolesnei paslaugai
Peržiūros sąrašai (u7)	Parodo panašių duomenų elementų sąrašą - pačias svarbiausias detales, padedančias filtruoti esybes tolesnėms operacijoms
Interaktyvios galutinio naudotojo įvedimo paslaugos (į)	
1-funkciniai įvedimo dialogai (į1)	Palaiko tiktai vieną iš trijų palaikymo tipų (kurti, atnaujinti arba šalinti)
2-funkciniai įvedimo dialogai (į2)	Palaiko du iš trijų palaikymo tipų (kurti, atnaujinti ir/arba šalinti)
3-funkciniai įvedimo dialogai (į3)	Palaiko visus tris palaikymo tipus (kurti, atnaujinti ir šalinti)
Neinteraktyvios galutinio naudotojo išvedimo paslaugos (i)	
Išvedimo formos (i1)	Paslaugos, sukuriančios vienodos struktūros spausdinamus arba ekrane vaizduojamus dokumentus
Ataskaitos (i2)	Paslaugos, sukuriančios nuo pateiktų duomenų priklausančios struktūros spausdinamus arba ekrane vaizduojamus dokumentus
Elektroninis paštas ir tekstinės žinutės (i3)	Paslaugos, sukuriančios standartizuotos struktūros elektroniniu būdu perduodamus išvedimo dokumentus
Monitoriaus ekrano išvedimas (i4)	Paslaugos, pastoviai vaizduojančios dokumentus, kurie yra reguliariai atnaujinami keičiantis duomenims
Interfeiso paslaugos kitoms programoms (įv)	
Pranešimai kitoms programoms (įv1)	Paslaugos, kuriose duomenų grupės yra siunčiamos tiesiogiai, dažniausiai realiu laiku, į bet kurią kitą programą
Neinteraktyvūs įrašai kitoms programoms (įv2)	Paslaugos, kuriose duomenų grupės yra įrašomos į laikiną failą siuntimui į kitą programą
Signalai į prietaisus ir kitas programas (įv3)	Paslaugos, kuriose duomenų eilutės arba pavienės informacijos dalys yra persiunčiamos į bet kuriuos kitus prietaisus ar programas
Interfeiso paslaugos iš kitų programų (iš)	
Pranešimai iš kitų programų (iš1)	Paslaugos, kuriose duomenys yra priimami tiesiogiai, dažniausiai realiu laiku, iš bet kurių kitų programų

BFK tipai	Trumpas aprašas
Neinteraktyvūs įrašai iš kitų programų (iš2)	Paslaugos, kuriose duomenys priimami neinteraktyviomis grupėmis iš bet kurių kitų programų
Signalai iš prietaisų ir kitų programų (iš3)	Paslaugos, kuriose duomenų eilutės arba pavienės informacijos dalys yra priimamos iš bet kurių kitų prietaisų ar programų
Duomenų saugojimo paslaugos (d)	
Esybės arba klasės (d1)	Aktualių naudotojui duomenų saugojimo paslaugos
Kiti įrašų tipai (d2)	Kitų duomenų be esybių ir klasių saugojimo paslaugos
Algoritminės ir valdymo paslaugos (a)	
Saugumo šablonai (a1)	Algoritminės saugumo paslaugos, pavyzdžiui, kriptografija
Skaičiavimų šablonai (a2)	Algoritminės aritmetinių arba loginių skaičiavimų paslaugos
Modeliavimų šablonai (a3)	Algoritminės modeliuojamų skaičiavimų paslaugos
Formato keitimo šablonai (a4)	Specialių formatų keitimo paslaugos
Duomenų bazių valymo šablonai (a5)	Duomenų saugojimą palaikančios paslaugos, tokios, kaip nereikalingų įrašų šalinimas
Kiti valdantieji šablonai (a6)	Apima visas kitas nepriklausomas, naudotojų apibrėžtas algoritmines ir valdančiąsias paslaugas

1.1.4.6. Reikšmių priskyrimas baziniams funkciniams komponentams

Kiekvienai iš septynių bazinių funkcinių komponentų klasių pagal atitinkamas formules yra apskaičiuojamas komponentų dydis.

1.1.4.6.1. Interaktyvios galutinio naudotojo navigacijos ir užklausų BFK formulė

$D_u = 0,2 + s/7 + k_s/2$, kur:

- D_u – užklausų dydis;
- s – duomenų elementų, laukų skaičius;
- k_s – skaitymo kreipimųsi į esybes skaičius.

1.1.4.6.2. Interaktyvios galutinio naudotojo įvedimo BFK formulė

$D_į = k \times (0,2 + s/5 + k_r/1,5 + k_s/2)$, kur:

- $D_į$ – įvedimų dydis;
- k – funkcionalumo koeficientas su reikšmėmis 1, 2 arba 3, priklausomai nuo turimų palaikymo tipų skaičiaus;
- s – duomenų elementų, laukų skaičius;

- k_r – rašymo kreipimųsi į esybes skaičius;
- k_s – skaitymo kreipimųsi į esybes skaičius.

1.1.4.6.3. Neinteraktyvios galutinio naudotojo išvedimo BFK formulė

$D_i = 1 + s/5 + k_s/2$, kur:

- D_i – išvedimų dydis;
- s – duomenų elementų, laukų skaičius;
- k_s – skaitymo kreipimųsi į esybes skaičius.

1.1.4.6.4. Interfeiso BFK kitoms programoms formulė

$D_{įv} = 0,5 + s/7 + k_s/2$, kur:

- $D_{įv}$ – interfeiso kitoms programoms dydis;
- s – duomenų elementų (atributų) skaičius;
- k_s – skaitymo kreipimųsi į esybes skaičius.

1.1.4.6.5. Interfeiso BFK iš kitų programų formulė

$D_{iš} = 0,2 + s/5 + k_r/1,5 + k_s/2$, kur:

- $D_{iš}$ – interfeiso iš kitų programų dydis;
- s – duomenų elementų, laukų skaičius;
- k_r – rašymo kreipimųsi į esybes skaičius;
- k_s – skaitymo kreipimųsi į esybes skaičius.

1.1.4.6.6. Duomenų saugojimo paslaugos formulė

$D_d = 1,5 + s/4$, kur:

- D_d – esybių ar įrašų dydis;
- s – duomenų elementų (atributų) skaičius.

1.1.4.6.7. Algoritminės ir valdymo paslaugos formulė

$D_a = 0,1 + s/5 + k_o/3$, kur:

- D_a – algoritmų dydis;
- s – duomenų elementų (kintamųjų, operandų) skaičius;
- k_o – operacijų skaičius.

1.1.4.7. Funkcinio dydžio apskaičiavimas

Visas matuojamos programinės įrangos dydis (D_v) yra apskaičiuojamas sudedant visų BFK klasėse apskaičiuotų dydžių reikšmes:

$D_v = D_u + D_į + D_i + D_{įv} + D_{iš} + D_d + D_a$

1.1.4.8. Matavimo rezultatų dokumentavimas

Galiausiai naudojant kokius nors programinius įrankius reikia dokumentuoti FiSMA metodo skaičiavimo rezultatus, kuriuose turėtų būti įtraukta:

- naudoto metodo identifikaciniai duomenys (naudota versija);
- BFK klasės ir tipai;
- kiekvienam BFK tipui skaičiuoti specifiniai komponentai (skaitymo ir rašymo kreipimaisi);
- BFK klasių skaičiavimo rezultatai;
- visas matuojamos programinės įrangos dydis.

1.1.5. NESMA funkcinių taškų analizės metodas

Olandų programinės įrangos matų asociacija 1989 metais išleido pirmąjį funkcinių taškų analizės metodą, paremtą Albrecht funkcinių taškų analizės metodo principais. Naudojant pirmąsias NESMA metodo versijas, buvo gaunami žymiai mažesni skaičiavimų rezultatai nei naudojant IFPUG metodą, dėl skirtingų naudojamų taisyklių ir jų interpretacijų. Vis dėlto glaudžiai bendradarbiaujant šiom abiem asociacijom, naujausios jų metodų versijos yra labai panašios. Anot NESMA organizacijos narių, kurie taip pat priklauso ir IFPUG skaičiavimo praktikų komitetui, metodai yra 95-98% panašūs [BD08].

NESMA ir IFPUG metodai išskiria penkis tuos pačius naudotojų funkcijų tipus: vidinius loginius failus, išorinius interfeiso failus, išorinius įvedimus, išorinius išvedimus ir išorines užklausas. Funkcijų tipų ir sudėtingumų nustatymo taisyklės yra tokios pačios, tik su keliomis išimtimis [NSMA08]:

- **išorinė užklausa prieš išorinį išvedimą**. IFPUG metode išorinės užklausos funkcija apibrėžiama, kaip funkcija, kuri pateikia duomenis naudotojui iš loginių failų nevykdant jokių papildomų duomenų apdorojimų (skaičiavimai, vidinių loginių failų atnaujinimai ir pan.). Visais kitais atvejais funkcija laikoma, kaip išorinis išvedimas. NESMA metode taikomos tos pačios taisyklės, tik papildomai turi būti įvestas unikalus pasirinkimo raktas ir išvedimai turi būti nustatyti apimtyse. Todėl kai kuriais atvejais IFPUG metodas skaičiuos išorinę užklausą, o NESMA metodas tą pačią funkciją skaičiuos, kaip išorinį išvedimą;
- **išorinės užklausos sudėtingumas**. NESMA metode išorinės užklausos įvedimo dalies funkcinis sudėtingumas yra paremtas išorinio įvedimo funkcijos sudėtingumo taisyklėmis, o išvedimo dalies funkcinis sudėtingumas yra paremtas išorinio išvedimo funkcijos

taisyklėmis. Didesnio sudėtingumo reikšmė iš šių dviejų bus panaudojama, kaip išorinės užklausos sudėtingumas. IFPUG metode funkcinis sudėtingumas yra nustatomas, kaip ir visose kitose transakcijose, suskaičiuojant duomenų elementų tipų, kertančių programos ribas, skaičių;

- **neišreikštinė užklausa**. Keičiant arba šalinant duomenis, duomenys yra dažniausiai pirma pateikiami naudotojui peržiūrėti. Tai yra žinoma kaip neišreikštinė užklausa. NESMA metode neišreikštinė užklausa nėra laikoma atskira transakcine funkcija, bet kaip keitimo ir šalinimo funkcijų dalis. Neišreikštinės užklausos naudotojui pateikiami duomenų elementų tipai yra pridedami prie keitimo ir šalinimo funkcijose apskaičiuotų duomenų elementų tipų. IFPUG metode nėra apibrėžtos šio atvejo specifinės taisyklės;
- **kodo duomenys (kodo lentelės)**. Paprastai, esybės yra sudarytos iš pagrindinių duomenų (verslo objektų) arba iš antrinių duomenų (pagalbiniai duomenys). Pagrindiniams duomenims NESMA ir IFPUG metodai laikosi tų pačių skaičiavimo taisyklių. Antriniai duomenys dažniausiai susideda iš kodo lentelių. Skaičiuojant duomenų funkcijas NESMA metodas klasifikuos visas kodo lenteles, kaip vieną vidinį loginį failą arba išorinį interfeiso failą. Įrašų elementų tipų skaičius bus prilyginamas nustatytų kodo lentelių skaičiui. IFPUG metode kodo lentelės ir su jomis susijusios transakcinės funkcijos nėra skaičiuojamos funkciniais taškais;
- **fizinė laikmena**. Jeigu duomenų elementų tipų ir loginių apdorojimų skaičius yra vienodas, įvedimai iš skirtingų laikmenų bus skaičiuojami, kaip vienas išorinis įvedimas. Tas pats galioja ir išoriniams išvedimams. Ataskaitos, kurios gali būti vaizduojamos skirtingose laikmenose, yra skaičiuojamos, kaip viena išorinio išvedimo funkcija. IFPUG metode netaikomos jokios specifinės skaičiavimo taisyklės šiai situacijai;
- **užklausos su keliais pasirinkimais (ir/arba situacijos)**. NESMA skaičiavimo taisyklėse numatyta skaičiuoti tiktai nesuderinamus (angl. mutually exclusive) pasirinkimus. IFPUG netaiko jokių skaičiavimo taisyklių šiai situacijai.

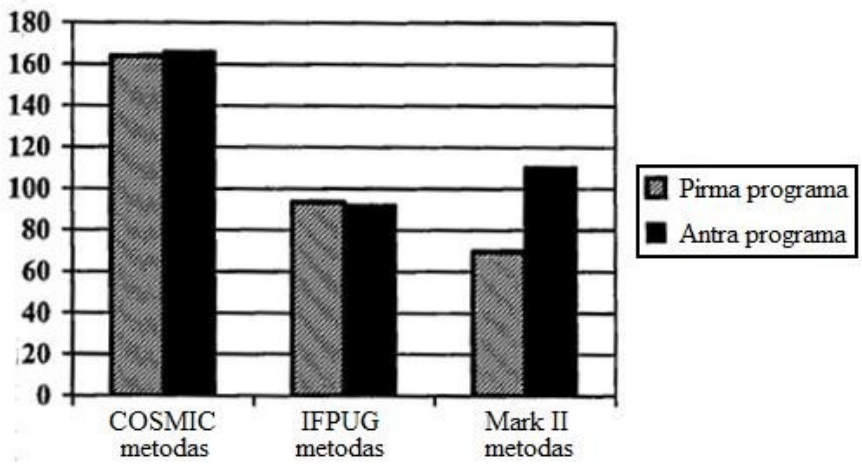
1.2. Ankstesni metodų palyginimai

Buvo atlikti ir ankstesni funkcinių taškų analizės metodų palyginimai. Šiame skyriuje apžvelgiami straipsniai, kuriuose buvo lyginami ir vertinami mūsų nagrinėjami standartizuoti funkcinio dydžio matavimo metodai.

COSMIC organizacija pateikia pirmos kartos funkcinio dydžio matavimo metodo IFPUG ir antros kartos funkcinio dydžio matavimo metodo COSMIC palyginimą. Šių metodų vertinimas atliktas įvairiais aspektais: taikymo sritis, matavimo perspektyva, metodo koncepcijos tinkamumas moderniem programinės įrangos funkcinių naudotojų reikalavimų dokumentavimo metodams, koncepcijos vertinimo subjektyvumas, dydžio skalės pagrįstumas, matavimo greitis, rezultatų tarpusavio konvertuojamumas ir palyginamųjų duomenų prieinamumas. Yra teigiama, kad COSMIC metodas yra gerokai pranašesnis už IFPUG ir kitus pirmos kartos funkcinio dydžio matavimo metodus. Funkcinio dydžio matavimas COSMIC metodu yra patikimesnis, lengviau atliekamas ir pakartojamas [CSMIC08].

Paul Vickers išnagrinėjo IFPUG ir Mark II metodus, bei pateikia jų taikymo kritiką. IFPUG metodas kritikuojamas dėl komponentų sudėtingumo įvertinimo skalės nepakankamumo dabartiniams vertinimo poreikiams, nekoreguotų funkcinių taškų lentelės reikšmių tinkamumo, vidinių loginių failų nustatymo taisyklių tikslumo, bendrųjų sistemos charakteristikų suprantamumo ir jų vertinimo dydžio vienodumo, nepakankamo atsižvelgimo į vidinį transakcijų vykdymo sudėtingumą, netinkamo pavienių ir kompleksinių sistemų vertinimo skirtumo, bei blogesnio didelių sistemų vertinimo. Mark II metodas kritikuojamas, kaip ir IFPUG metodas, dėl bendrųjų sistemos charakteristikų taikymo. Dalis IFPUG metodą remiančių idėjų turi trūkumų, todėl Mark II funkcinių taškų analizės metodas suteikia tikslesnį verslo sistemų funkcinio dydžio vertinimą [Vic03].

Taip pat buvo atlikta trumpa IFPUG, COSMIC ir Mark II metodų kokybinė ir kiekybinė lyginamoji analizė vertinant dvi skirtingas programas. Programų vertinimo rezultatus galime pamatyti 1 pav.



1 pav. Dviejų programų vertinimo COSMIC, IFPUG ir Mark II metodais rezultatai [CCS+04]

Atlikus metodų analizę yra teigiama, kad [CCS+04]:

- dėl praktiškesnės ir realistiškesnės vertinimo perspektyvos COSMIC metodas suteikia geresnį vertinimą nei IFPUG ir Mark II metodai. Tuo labiau COSMIC metodas nenaudoja koregavimo faktorių, kaip IFPUG ir Mark II metodai;
- lyginant šiuos tris metodus, tampa aišku, kad IFPUG metodas turi mažiausią sąryšį su programų charakteristikomis, bandant nustatyti programų dydį.

Onur Deminors ir Cigdem Gencel aptaria koncepcinius IFPUG, COSMIC ir Mark II funkcinio dydžio matavimo metodų panašumus ir skirtumus, bei pasiūlo metodų unifikuotą modelį, kuriuo galima [DG09]:

- **sinchroninis matavimas**. Modelis įgalina sinchroninį vieno, dviejų arba visų trijų metodų taikymą. Organizacijoms, kurioms reikia naudoti skirtingus funkcinio dydžio matavimo metodus, gali naudoti unifikuotą modelį, taip sumažinant laiko ir darbo sąnaudas;
- **automatizuotas matavimas**. Kadangi modelis aiškiai apibrėžia funkcinio dydžio matavimo metodų sąryšius, visų metodų bendras automatizavimas yra daug paprastesnis;
- **skirtingų dydžių konvertavimas.** Unifikuotas modelis padės programinę įrangą kuriančiai organizacijai pagal istorinius duomenis, surinktus naudojant ankstesnį metodą, pakeisti funkcinio dydžio matavimo metodą. Tačiau tam reikia surinktų detalių matavimo duomenų, ne tik matavimo rezultatų;
- **matavimo rezultatų platinimas**. Unifikuotas modelis taip pat įgalina rinkti tų pačių produktų funkcinio dydžio matavimo duomenis skirtingais metodais. Šiuos rezultatus nesunku platinti ir padaryti viešai prieinamais.

Cigdem Gencel, Onur Deminors ir Erhen Yuceer įvertino dviejų realaus laiko sistemų kūrimo projektus Mark II ir COSMIC metodais. Vertinimo rezultatai pateikti 6 lentelėje.

6 lentelė. Realaus laiko sistemų vertinimo Mark II ir COSMIC metodais rezultatai [GDY05b]

Projekto pavadinimas	**Mark II dydis (Mark II FT)**	**COSMIC dydis (CFT)**	**Kodo eilučių skaičius**	**Kodo eilučių skaičius /Mark II FT**	**Kodo eilučių skaičius /CFT**
Projektas-1	4513,2	3408	20076	4,4	5,9
Projektas-2	1120,54	879	3896	3,5	4,4

Tyrimas buvo atliktas su tikslu ne spręsti, kuris metodas tinkamesnis vertinti realaus laiko sistemų funkcinį dydį, o nustatyti metodų gerinimo galimybes. Kadangi Mark II ir COSMIC metodai naudoja skirtingą matavimo koncepciją, jų matavimo rezultatų negalima tiesiogiai palyginti. Palyginimui reikalingas kurio nors metodo rezultatų konvertavimas kito metodo rezultatų matais.

Yra teigiama, kad abu metodai yra taikytini matuojant realaus laiko sistemų funkcinį dydį, bet su kai kuriais apribojimais, esant algoritminiams komponentams. Programinės įrangos kūrimo metu COSMIC metodą galima taikyti anksčiau nei Mark II metodą, nes COSMIC metodui nereikalingas duomenų elementų tipų įvedimų ir išvedimų komponentuose nustatymas. Numatomos šių metodų gerinimo galimybės: konvertavimo funkcijų formulavimas ir algoritminių manipuliacijų matavimo apibrėžimas [GDY05b].

Total Metrics organizacija pateikia funkcinio dydžio matavimo metodų panaudojimo svarbą ir ribojimus, o taip pat mini tokius pagrindinius metodo pasirinkimo kriterijus [TM07]:

- **pritaikomumas kuriamos programinės įrangos sričiai**. Daugelis organizacijų, kuriančių verslo sistemas, renkasi IFPUG metodą, o, kuriančių realaus laiko sistemas, renkasi COSMIC metodą;
- **industrinių duomenų prieinamumas**. Jeigu reikalingi industriniai duomenys produktyvumo palyginimui, tai šis kriterijus yra gana svarbus. Daugelyje iš užbaigtų projektų, kuriuose buvo panaudoti funkcinio dydžio vertinimai, informacijos saugyklų yra panaudotas IFPUG metodas, bet ir COSMIC metodo panaudojimas tampa vis populiaresnis ir labiau taikomas;
- **apmokymų ir įrankių prieinamumas**. Daugelis populiariausių įrankių yra sukurti vertinti funkcinį dydį IFPUG metodu, bet augant COSMIC metodo populiarumui, netrukus atsiras daugiau ir COSMIC metodą palaikančių įrankių;
- **apmokytų, patyrusių ir sertifikuotų ekspertų prieinamumas**. Daugelis ekspertų yra apmokyti ir sertifikuoti IFPUG metodo naudojimui, kas yra siūloma daugelyje šalių. COSMIC metodo sertifikavimu užsiima įmonės, esančios Europoje, Šiaurės Amerikoje ir Australijoje.

Pam Morris ir Jean Marc Desharnais pateikia verslo ir ne verslo sistemų vertinimo tyrimą IFPUG ir COSMIC metodais. Yra pabrėžiama, kad [MD98]:

- COSMIC metodas yra efektyvesnis matuojant ne verslo sistemų teikiamą funkcionalumą;

- COSMIC metodas lengviau pritaikomas;
- COSMIC metodo naudojimui reikia detalesnių kiekvieno proceso vidinio veikimo reikalavimų.

Gu Xunmei, Song Guoxin ir Zheng Hong įvairiais aspektais palygina IFPUG ir COSMIC funkcinio dydžio matavimo metodus, bei pateikia jų skirtumus ir panašumus. Pagrindiniai nustatyti panašumai: matavimo žingsniai, funkcinių reikalavimų išreiškimas, matavimo pritaikymo fazė ir konvertuojamumas. Pagrindiniai nustatyti skirtumai: skaičiavimo taisyklės, skirtingi gaunami rezultatai, matavimo perspektyva, elgesys su techniniais ir kokybiniais reikalavimais, bei taikymo sritis [XGH06].

NESMA organizacija pateikia pagrindinius IFPUG ir NESMA metodų skirtumus. Pagrindiniai šių metodų skirtumai apima išorinių užklausų ir išorinių išvedimų, išorinių užklausų sudėtingumų, neišreikštinių užklausų, kodo lentelių, fizinių laikmenų ir užklausų su keliais pasirinkimais skirtingus traktavimus [NSMA08].

2. Metodų vertinimo kriterijai

2.1. Galimi vertinimo metodų pasirinkimo kriterijai

Visiems programinės įrangos kūrėjams ir vertintojams reikalingi tokie vertinimo metodai, kurie pateiktų patikimus vertinimus, apimančius visą programinės įrangos gyvavimo ciklą. Tam daugelis autorių pateikia begales įvairių reikalavimų ir kriterijų, kuriais remiantis galima įvertinti įvairius vertinimo metodus. Šiame poskyryje apibendrinsime ir suformuluosime įvairių autorių siūlomus vertinimo metodų pasirinkimo kriterijus. Tam bus panaudoti [Zus98] šaltinyje aprašyti Watts ir Jones, [BD08] nurodyti Noth ir Kretzschmar, bei [MS99], [MAO+98] ir [TM07] šaltinių autorių siūlomi vertinimo metodų svarbiausieji pasirinkimo kriterijai.

Apžvelgus paminėtus šaltinius, galima suformuluoti tokius galimus vertinimo metodų pasirinkimo kriterijus:

1) **palyginamumas (angl. comparability)**. Matavimo rezultatai turi būti palyginami su kitais tos pačios klasės metodų matavimo rezultatais;
2) **objektyvumas (angl. objectivity)**. Matavimo rezultatai turi nepriklausyti nuo subjektyvios vertintojų nuomonės. Nėra svarbu kas atlieka matavimus;
3) **standartizuotumas (angl. standartization)**. Metodas turi būti standartiškai apibrėžtas ir nedviprasmiškas. Kaip blogą pavyzdį galima paminėti kodo eilučių skaičių, kurį galima interpretuoti skirtingai. Galima skaičiuoti, pavyzdžiui, visas kodo eilutes arba tik kodo eilutes be komentarų. Standartizuotumas parodo, ar visi metodą naudoją tuo pačiu būdu;
4) **naudingumas (angl. usefulness)**. Matavimo rezultatai turi suteikti naudingą informaciją. Jais remiantis galima priimti reikiamus sprendimus;
5) **oficialios naudotojų grupės (angl. formal user groups)**. Metodas turi turėti oficialią naudotojų grupę ir atitinkamą išleistą metodo standartinį apibrėžimą. Metodo panaudojimas turi būti aiškiai specifikuotas ir apibrėžtas;
6) **įrankių palaikymas (angl. tool support)**. Metodas turi būti palaikomas programiniais įrankiais. Vertinimo metodas yra geresnis, jeigu jis turi automatinį palaikymą. Toks vertinimo metodas yra efektyvesnis;
7) **konvertuojamumas (angl. convertibility)**. Matavimo rezultatai gali būti konvertuojami į kitus to paties tipo matus. Kaip, pavyzdžiui, kodo eilučių skaičius į funkcinius taškus;
8) **naudojimo lengvumas (angl. ease of use)**. Metodas turi būti panaudojamas be ilgų pasiruošimo darbų, tokių kaip, pavyzdžiui, užbaigtų projektų istorinės informacijos rinkimas;

9) **lengvas išmokstamumas (angl. ease to learn)**. Metodą turi būti nesunku išmokti naudoti po kelių jo pritaikymų;
10) **apskaičiavimo pastangos (angl. effort to arrive at an estimate)**. Metodo naudojimo trukmė neturi viršyti paties programinės įrangos kūrimo proceso trukmės. Šis kriterijus taip pat apibrėžia metodo naudojimo efektyvumą ir pelningumą;
11) **skaidrumas (angl. transparency)**. Metodo įvedamų reikšmių transformavimas į gaunamus rezultatus turi būti matomas, kad juo būtų galima pasitikėti;
12) **ankstyvas pritaikymas (angl. early applicability)**. Metodą turi būti galima taikyti ankstyvose programinės įrangos gyvavimo ciklo fazėse. Svarbu naudoti tokį metodą, kuris gali būti taikomas norimos programinės įrangos kūrimo fazės metu;
13) **pakartotinis panaudojimas geresniam įvertinimui (angl. ability to iterate the results for better estimates)**. Šis kriterijus apibūdina pakartotinį metodo naudojimą skirtingu projekto vykdymo metu. Matavimai tampa vis tikslesni, kai projekto metu gaunama vis naujesnė informacija;
14) **tikslumas (angl. exactness)**. Tikslumas apibrėžia skirtumą tarp apskaičiuoto ir tikro rezultato. Šis kriterijus parodo, ar gauti rezultatai yra artimi tikriesiems rezultatams;
15) **atsekamumas (angl. traceability)**. Vertinimas yra atsekamas, jeigu trečioji šalis gali suprasti ir pakartoti atliktus skaičiavimus. Turi būti aišku, kodėl buvo daromos tam tikros prielaidos ir atlikti tam tikri vertinimai;
16) **įvertinamumas (angl. evaluation)**. Įvertinamumo kriterijus reikalauja, kad esamoje projekto fazėje vertinimui įtakos turintys faktoriai turi būti objektyviai apskaičiuojami;
17) **parametrų skaičius (angl. number of parameters)**. Šis kriterijus nurodo, kad visi skaičiavimuose naudojami faktoriai turi tiesiogiai paveikti vertinimo rezultatus. Metodas neturi reikalauti nereikalingos arba perteklinės informacijos;
18) **stabilumas (angl. stability)**. Vertinimui naudojant tuos pačius įvesties duomenis, metodas turi išvesti tuos pačius rezultatus;
19) **defektų aptikimas (angl. defect localization)**. Šis kriterijus apibūdina vertinimo metodo galimybę atpažinti netinkamus įvedimo duomenis arba prieštaringus duomenis;
20) **lankstumas (angl. flexibility)**. Metodas be žymių pakeitimų gali būti naudojamas skirtingose programinės įrangos kūrimo aplinkose ir visiems skirtingiems programinės įrangos kūrimo projektų tipams;

21) **prisitaikomumas (angl. adaptability)**. Prisitaikomumo kriterijus apibrėžia metodo gebėjimą reaguoti ir atsižvelgti į pasikeitusias sąlygas. Parodo, ar lengva metodą modifikuoti, atnaujinti ir kalibruoti;

22) **apibrėžtumas (angl. definition)**. Nurodo, ar metodas yra tinkamai apibrėžtas techninėje dokumentacijoje, ar metodas tiksliai apibrėžia, kuriuos rezultatus jis gali ir kurių negali pateikti. Taip pat šis kriterijus nurodo, ar įvedami kintamieji tiksliai apibrėžti, ar yra apibrėžta, į ką reikia atkreipti dėmesį ir ką ignoruoti matuojant kintamuosius;

23) **suderinamumas (angl. compatibility)**. Parodo metodo koncepcijos suderinamumą su kitais to tipo metodais;

24) **pajėgumas (angl. power)**. Parodo įvedamų kintamųjų reikiamą detalumo ir išbaigtumo lygį, norint gauti prasmingus rezultatus;

25) **pigumas (angl. cheapness)**. Parodo, ar metodas sunaudoja mažai piniginių ir laiko resursų;

26) **šykštumas (angl. parsimony)**. Parodo, ar metodas vengia naudoti faktorius, kurie neturi pastebimos įtakos gaunamiems rezultatams;

27) **patikimumas (angl. reliability)**. Parodo, ar įvedimo duomenys ir rezultatai yra patikimi, ar vertinimo procesas yra pakartojamas, ar jis statistiškai dėsningas;

28) **pagrįstumas (angl. validity)**. Parodo, ar metodas buvo panaudotas atitinkamuose techniniuose kontekstuose ir patvirtintas nepriklausomų tyrėjų;

29) **nepriklausomumas (angl. independence)**. Parodo, ar įvedami kintamieji yra nepriklausomi tarpusavy, ar išvengiamas dubliavimas;

30) **išreiškiamumas (angl. articulation)**. Parodo, ar metodu nesunkiai galima vertinti atskirus programinės įrangos komponentus, o ne tik gauti bendrą rezultatą;

31) **pasitikėjimas (angl. confidence)**. Parodo, ar įmanoma numatyti tikėtiną vertinimo paklaidą. Dydžio matas turėtų būti išreiškiamas rėžiu su minimalia ir maksimalia reikšmėmis, nei vienu skaičiumi;

32) **suprantamumas (angl. ease of understanding)**. Parodo, ar metodo apibrėžimas, skaičiavimo procedūros ir taisyklės yra aiškios ir pakankamai detalios, kad skirtingi vertintojai gautų panašius skaičiavimų rezultatus;

33) **skaičiavimo pastangos (angl. counting effort)**. Nurodo pastangas, reikalingas atlikti skaičiavimus;

34) **pritaikomumas kuriamos programinės įrangos sričiai (angl. applicability to domain of software that you need to measure)**. Nurodo tinkamumą kuriamiems programinės įrangos tipams, kaip, pavyzdžiui, verslo sistemoms ar realaus laiko sistemoms;

35) **industrinių duomenų prieinamumas (angl. availability of equivalent industry data)**. Jeigu reikalingi industriniai duomenys produktyvumo palyginimui, tai šis kriterijus yra gana svarbus. Tam tikslui galima naudoti ISBSG (angl. International Software Benchmarking Standards Group) saugykloje saugomų projektų, kuriuose buvo panaudoti funkcinio dydžio matavimai, duomenis;

36) **apmokymų prieinamumas (angl. availability of training)**. Parodo prieinamus metodo taikymo apmokymus;

37) **apmokytų, patyrusių ir sertifikuotų ekspertų prieinamumas (angl. availability of trained experienced certified metrics experts)**. Nurodo sertifikuotų ekspertų ir sertifikatų gavimo prieinamumą.

Čia paminėti vertinimo metodų pasirinkimo kriterijai yra tinkami palyginti ir įvertinti įvairius vertinimo metodus.

2.2. Funkcinio dydžio matavimo metodų vertinimo kriterijai

Įvairiuose šaltiniuose yra apibrėžta labai daug įvairių vertinimo metodų pasirinkimo kriterijų, bet magistro darbo apimtis yra per maža jais visais įvertinti nagrinėjamus metodus. Todėl suformuluokime magistro darbo autoriaus nuomone svarbiausius iš jų, pagal kuriuos ir atliksime tiriamų standartizuotų IFPUG, Mark II, COSMIC, NESMA ir FiSMA funkcinio dydžio matavimo metodų vertinimus:

1) **koncepciniai panašumai ir skirtumai**. Kad geriau suvoktume tiriamus metodus ir galėtume juos vertinti pagal kitus kriterijus, pirmiausia panagrinėsime jų koncepcinius panašumus ir skirtumus. Pažiūrėsime kaip tiriamuose funkcinio dydžio matavimo metoduose apibrėžtos duomenų tipo ir transakcijos koncepcijos;
2) **standartizuotumas**. Labai svarbus kiekvieno vertinimo metodo aspektas yra jo standartizuotumas. Standartizuoti metodai yra efektyvesni, patikimesni ir kokybiškesni. Standartuose yra aprašomos geriausios taikomos praktikos, jie skatina novatoriškumą. Funkcinio dydžio matavimo metodas turi būti standartiškai apibrėžtas ir nedviprasmiškas, kad kiekvienas jį galėtų naudoti tuo pačiu būdu ir gautu tuos pačius palyginamus rezultatus. Todėl šis kriterijus tiriamiems metodams yra labai svarbus. Vertindami metodus pagal šį kriterijų išnagrinėsime jų ir kitus, su funkcinio dydžio matavimu susijusius, standartus;

3) **industrinių duomenų prieinamumas**. Kiekvienai programinę įrangą kuriančiai organizacijai norinčiai konkuruoti su kitomis organizacijomis ir gerinti jos vykdomus kūrimo procesus yra labai svarbu vykdyti tiek vidinę, tiek išorinę lyginamąją analizę (angl. benchmarking). Išorinei lyginamajai analizei atlikti reikia pakankamo kiekio industrinių duomenų, todėl labai svarbus yra jų prieinamumas. Kad galėtume sėkmingai naudoti mūsų tiriamus funkcinio dydžio matavimo metodus, reikia turėti daug projektų, kuriuose panaudoti šie metodai, informacijos. Vertindami metodus panagrinėsime, keliuose projektuose yra panaudoti tiriami metodai populiariausioje, funkcinių taškų matą naudojančioje, duomenų apie užbaigtus projektus saugykloje;
4) **konvertuojamumas**. Funkcinio dydžio matavimo metodais gautų rezultatų konvertuojamumas labai praverčia neturint pakankamai daug industrinių duomenų. Konvertavimas įgalina ir skirtingais funkcinio dydžio matavimo metodais gautų rezultatų palyginimą. Ankstyvam ir greitam tiriamų metodų pritaikymui galima naudoti kodo eilučių skaičiaus konvertavimą į funkcinius taškus. Vertinant tiriamus funkcinio dydžio matavimo metodus pagal šį kriterijų išnagrinėsime jais gautų funkcinių taškų tarpusavio konvertuojamumo ir kodo eilučių skaičiaus konvertuojamumo į funkcinius taškus galimybes;
5) **metodų taikymo fazė**. Kiekvienas vertinimo metodas yra tuo geresnis, kuo ankstyvesnėje programinės įrangos gyvavimo ciklo fazėje jį galima sėkmingai pritaikyti. Šiuo atžvilgiu programinės įrangos dydžio matavimo metodą turėtų būti galima taikyti jau ankstyvu programinės įrangos gyvavimo ciklo metu, kad būtų galima nustatyti reikalingas darbo sąnaudas. Sukūrus programinę įrangą, ją reikia palaikyti. Todėl taip pat yra svarbi galimybė tiriamus metodus taikyti palaikymo fazės metu, taip nustatant reikiamas darbo sąnaudas palaikyti programinę įrangą. Taigi vertinant tiriamus funkcinio dydžio matavimo metodus pagal šį kriterijų išnagrinėsime, kada yra tinkamiausia taikyti šiuos metodus, ar galimas ankstyvesnis jų taikymas bei jų taikymas palaikymo fazės metu;
6) **subjektyvumas metoduose**. Įvairūs funkcinio dydžio matavimo metodų subjektyvumai jų naudojamose matavimo taisyklėse gali privesti prie skirtingų programinės įrangos vertintojų gaunamų labai skirtingų rezultatų. Funkcinio dydžio matavimo metodų taikymas turi būti kuo objektyvesnis, kad jų taikymo metu gaunami rezultatai būtų nepriklausomi nuo subjektyvios vertintojų nuomonės. Vertinant tiriamus funkcinio dydžio matavimo metodus pagal šį kriterijų panagrinėsime įvairius šiuose metoduose subjektyvumą įtakojančius faktorius;

7) **nefunkciniai naudotojų reikalavimai**. Nors kaip yra apibrėžta funkcinio dydžio matavimo metodų koncepcijų standarte, kad standartizuotais funkcinio dydžio matavimo metodais turi būti vertinami tik funkciniai naudotojų reikalavimai, kai kuriais iš jų galima įvertinti ir nefunkcinius naudotojų reikalavimus. Todėl vertinant tiriamus metodus pagal šį kriterijų panagrinėsime, kaip šie metodai atsižvelgia ir į nefunkcinius naudotojų reikalavimus;
8) **sertifikavimo ir skaičiavimo vadovų prieinamumas**. Kad metodai būtų labiau globaliai pripažįstami, juos sukūrusios organizacijos turėtų pasiūlyti galimybę tapti sertifikuotais šių metodų taikytojais. Sertifikuoti funkcinio dydžio matavimo metodų taikytojai gali atlikti tikslesnius matavimus ir tinkamai įvertinti subjektyvias šiuose metoduose naudojamas taisykles. Be suteikiamos sertifikavimo galimybės taip pat yra svarbu, kad metodus sukūrusios organizacijos ne tik pateiktų metodų skaičiavimo taisyklių vadovus, bet ir kitus metodo naudojimą palengvinančius dokumentus. Todėl vertinant nagrinėjamus metodus pagal šį kriterijų bus ištiriama sertifikavimo galimybių ir įvairių šių metodų taikymą palengvinančių dokumentų prieinamumas;
9) **metodus palaikantys programiniai įrankiai**. Įvairūs programiniai įrankiai gerokai palengvina ir paspartina funkcinių taškų vertintojų darbą, atliekami kokybiškesni ir labiau standartizuoti matavimai. Funkcinio dydžio matavimo metodo taikymas tampa gerokai efektyvesnis, jeigu metodo naudojimą palaiko programiniai įrankiai. Vertinant tiriamus funkcinio dydžio matavimo metodus pagal šį kriterijų išnagrinėsime, kiek ir kokie populiariausi programiniai įrankiai palaiko šiuos metodus ar jais gautus rezultatus;
10) **metodų taikymo sritis**. Šiuo metu skirtingos organizacijos kuria įvairių tipų programinę įrangą. Funkcinio dydžio matavimo metodas yra tuo geresnis, kuo įvairesniems programinės įrangos tipams matuoti jis yra tinkamas. Todėl metodų vertinimas pagal šį kriterijų yra ganėtinai svarbus. Vertinant tiriamus funkcinio dydžio matavimo metodus pagal šį kriterijų bus išnagrinėta, kokių tipų programinei įrangai matuoti jie yra tinkamesni.

3. Metodų palyginimas ir vertinimas

3.1. Koncepciniai panašumai ir skirtumai

Kad galėtume ištirti panašumus ir skirtumus tarp skirtingų funkcinio dydžio matavimo metodų, apibrėžkime bendrą funkcinio dydžio matavimo procesą. Kaip pagrindą šiam procesui panagrinėkime tiriamų penkių funkcinio dydžio matavimo metodų bendras savybes ir skirtumus tarp jų matavimo funkcijų.

Funkcinio dydžio matavimo metodai apibrėžia dvi programinės įrangos matavimo fazes, kurios padeda nustatyti elementus, reikalingus funkcinio dydžio matavimui [DG09].

Pirma funkcinio dydžio matavimo fazė yra funkcinių naudotojų reikalavimų išgavimas iš turimų artefaktų ir jų išreiškimas tinkama funkcinio dydžio matavimui forma. Funkciniai naudotojų reikalavimai apibrėžia programą kaip transakcijų ir duomenų tipų rinkinį.

Transakcijos naudoja duomenų tipus kaip įvedimus, apdoroja juos ir naudoja kaip išvedamus apdorojimo rezultatus. Funkcinio dydžio matavimo metodai naudoja transakcijas ir duomenų tipus bazinių funkcinių komponentų nustatymui. Tada šie metodai kiekvieną BFK suskirsto į BFK tipus ir nustato tipų atributus, aktualius bazinių skaičių gavimui.

Kitos fazės metu yra apskaičiuojamas kiekvieno BFK funkcinis dydis, pritaikant matavimo funkciją BFK tipams ir jų susijusiems atributams. Vertintojas tada agreguoja rezultatus, kad apskaičiuotų visą matuojamos sistemos dydį.

Toliau panagrinėkime duomenų tipo ir transakcijos koncepcijų skirtumus tiriamuose funkcinio dydžio matavimo metoduose.

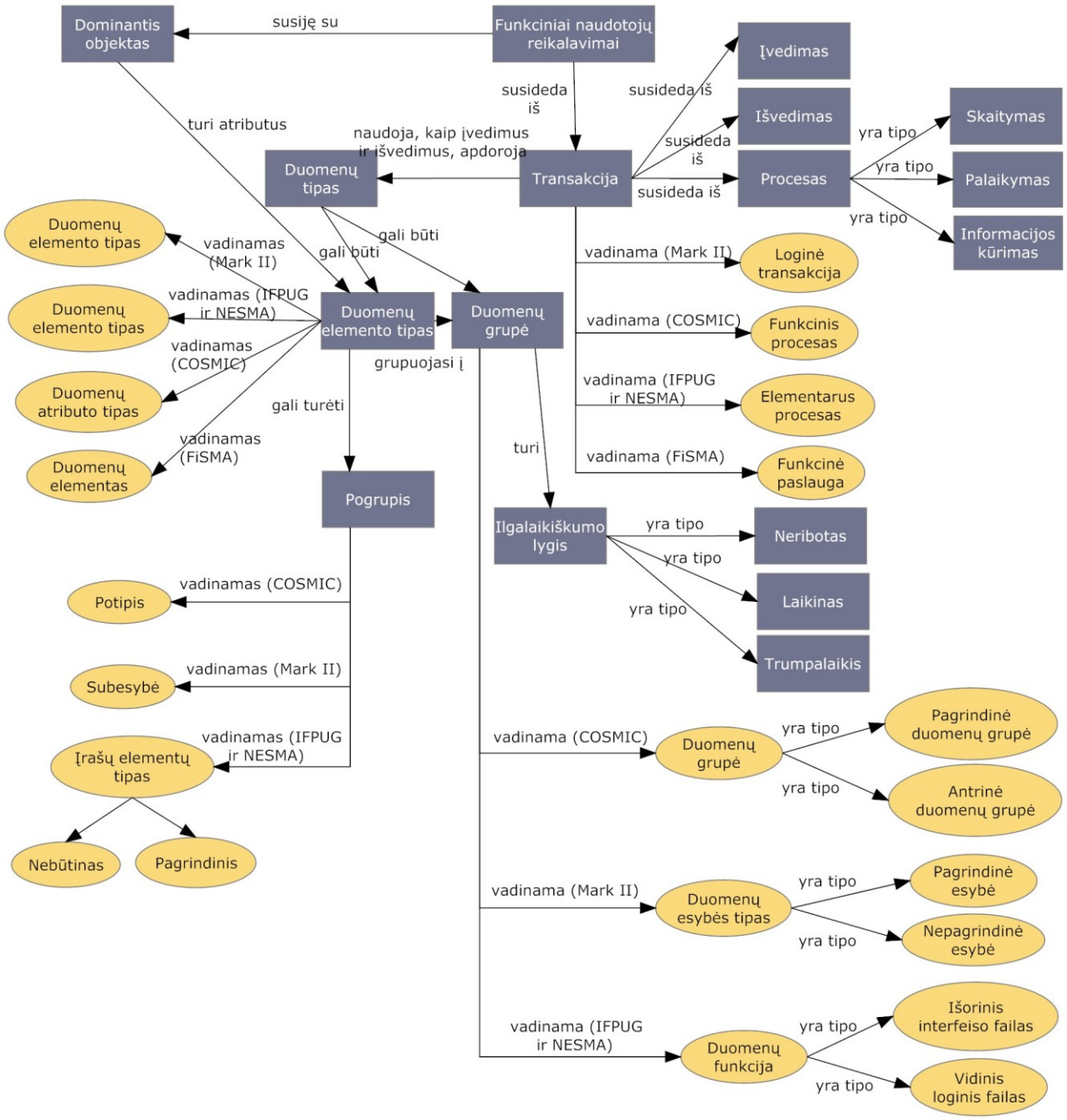
3.1.1. Duomenų tipo koncepcija

2 pav. parodytos bendrosios funkcinio dydžio matavimo metodų esminės koncepcijos ir sąsajos tarp jų. Taip pat pavaizduotos skirtingos metodų naudojamos terminologijos ir skirtingos jų apibrėžiamos specifinės koncepcijos.

Svarbi bendra koncepcija tarp tiriamų penkių metodų yra duomenų tipas, kurį vertintojas atpažįsta kaip DET arba duomenų grupę. DET parodo smulkiausius duomenų elementus, prasmingus naudotojui, ir yra dominančio objekto atributas, susijęs su transakcija. Duomenų elementų tipai sudaro logiškai susijusias grupes – duomenų grupes.

IFPUG ir NESMA metoduose DET apibrėžiamas kaip unikalus, naudotojo atpažįstamas, nesikartojantis laukas. Mark II metode DET apibrėžiamas kaip unikalus, naudotojo atpažįstamas informacijos apie esybės tipus elementas. COSMIC metode DET laikomas duomenų atributu ir apibrėžiamas kaip mažiausia informacijos dalelė nustatytoje duomenų grupėje, turinti prasmę,

žiūrint iš FNR perspektyvos. FiSMA metode DET vadinamas duomenų elementu ir apibrėžiamas kaip unikalus, naudotojo atpažįstamas, nesikartojantis BFK laukas.



2 pav. Funkcinio dydžio matavimo metodų koncepcinė sąsaja

IFPUG, NESMA, Mark II ir COSMIC funkcinio dydžio matavimo metodai taip pat sutaria dėl duomenų grupės koncepcijos, nors ir naudoja skirtingą terminologiją. IFPUG metodas laiko duomenų grupę kaip duomenų funkciją ir apibrėžia ją kaip logiškai susijusių duomenų ar valdymo informacijos grupę, į kurią kreipiasi programa. Mark II metodas vadina duomenų grupę duomenų esybės tipu ir apibrėžia ją kaip kažkokį realaus pasaulio objektą, apie kurį naudotojas nori laikyti

informaciją. COSMIC metodas duomenų grupę vadina duomenų grupe ir apibrėžia ją kaip skirtingą, netuščią, nesurikiuotą ir nedubliuojamą duomenų atributų rinkinį, kuriame kiekvienas įtrauktas duomenų atributas apibūdina to paties dominančio objekto įvairius aspektus. FiSMA metode duomenų grupė neapibrėžiama.

Kiekvienas metodas naudoja savitą duomenų grupių klasifikavimą. IFPUG metodas turi dviejų tipų duomenų funkcijas: VLF ir IIF. VLF yra tie failai, kuriuos naudotojas palaiko matuojamos programos ribose, o IIF, tai failai, kurie yra palaikomi kitos programos ribose. Mark II metodas turi du duomenų esybės tipus: pagrindinė ir nepagrindinė, priklausomai nuo to, ar duomenų esybės tipas yra pagrindinis matuojamai programai. COSMIC metode duomenų grupė klasifikuojama kaip pagrindinė duomenų grupė arba antrinė duomenų grupė, priklausomai nuo to, ar tai yra dominantis objektas tam tikrų tipų naudotojams, jų naudojamuose funkciniuose procesuose.

COSMIC metodas skiria duomenų grupes, priklausomai nuo jų ilgalaikiškumo lygio ir apibrėžia trumpalaikį, laikiną ir pastovų lygius. IFPUG, NESMA ir Mark II metodai kita vertus tiesiogiai nenustatinėja ilgalaikiškumo charakteristikos. Ilgalaikiškumo lygis priklauso nuo duomenų grupės buvimo vietos, kuri gali būti įvedimo/išvedimo įrenginys arba kintama arba pastovi saugykla.

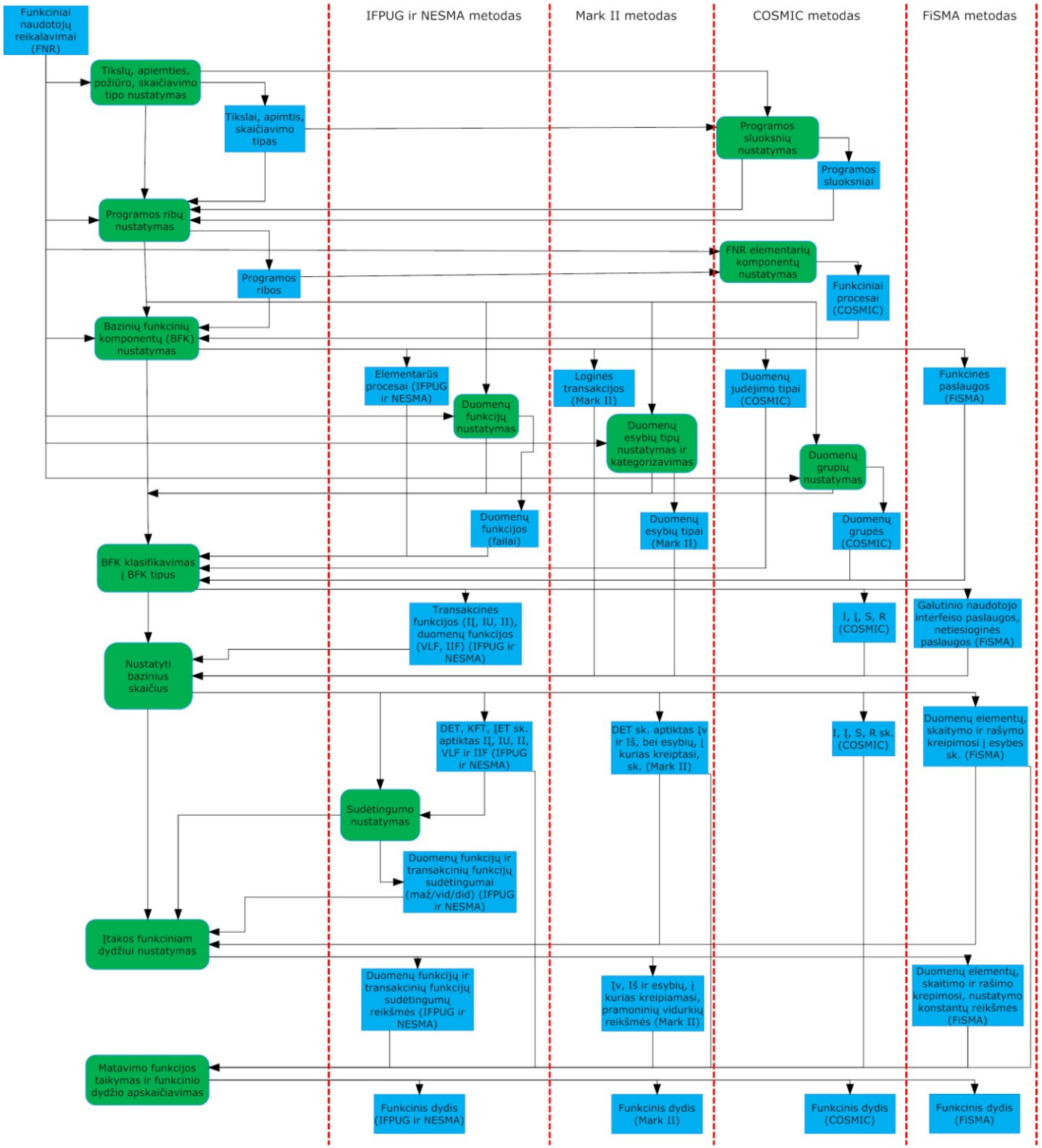
Pogrupis yra kita svarbi koncepcija. Pogrupis yra naudotojo atpažįstama duomenų elementų tipų grupė, esanti duomenų grupėje. IFPUG ir NESMA metoduose pogrupis vadinamas įrašų elementų tipu, kas yra naudotojo atpažįstamas duomenų elementų pogrupis, esantis VLF arba IIF. ĮET galima suskirstyti į du tipus: nebūtinas ir pagrindinis. Mark II metode taip pat naudojami duomenų pogrupiai, kurie vadinami subesybėmis. COSMIC metode skirtingi dominantys objektai gali būti laikomi kaip tam tikrų objektų potipiai.

3.1.2. Transakcijos koncepcija

3 pav. parodyti nagrinėjamų funkcinio dydžio matavimo metodų panašumai ir skirtumai tarp jų vykdomų procesų.

Visi tiriami funkcinio dydžio matavimo metodai daugmaž sutaria dėl transakcijos koncepcijos. IFPUG ir NESMA metodai vadina transakciją kaip transakcinę funkciją ir apibrėžia ją kaip smulkiausią veiklos vienetą, reikšmingą naudotojui. Ji turi būti savarankiška ir palikti programą pastovioje būsenoje. Mark II metodas transakciją vadina logine transakcija ir apibrėžia ją kaip žemiausio lygio verslo procesą, iššauktą unikaliu atitinkamu išorinio pasaulio įvykiu arba informacijos prašymu. Loginė transakcija, pabaigusi darbą, palieka programą pastovioje būsenoje. COSMIC metode transakcija vadinama funkciniu procesu ir apibrėžiama kaip elementarus funkcinių naudotojo reikalavimų rinkinio komponentas, sudarantis unikalų, darnų ir nepriklausomai įvykdomą

duomenų judėjimų rinkinį. Jis iššaukiamas vienu arba daugiau iššaukiamųjų įvykių ir yra užbaigiamas, kai yra įvykdoma tai, ko yra reikalaujama iššaukiamojo įvykio tipu. FiSMA metodas transakciją vadina funkcine paslauga ir apibrėžia ją kaip atskirus FiSMA BFK.



3 pav. Funkcinio dydžio matavimo metodų vykdomas procesas

Nors ir transakcijos koncepcija yra panaši visuose metoduose, bet yra skirtumai, kaip elgiamasi su esybėmis ir jų atributais, kai yra matuojamas transakcijos funkcionalumas. IFPUG ir NESMA metodai apibrėžia tris transakcinių funkcijų tipus: išorinis įvedimas, išorinis išvedimas ir išorinė užklausa. Tačiau Mark II ir COSMIC metodai neskirsto transakcijų į tipus. Kita vertus Mark II metode laikoma, kad loginė transakcija gali būti sudaryta iš trijų komponentų: įvedimo, apdorojimo ir išvedimo. COSMIC metode funkcinis procesas yra sudarytas iš keturių žemesnių procesų tipų, apibrėžtų kaip duomenų judėjimo tipai, įvykstantys funkcinio proceso vykdymo metu. Šie keturi duomenų judėjimo tipai yra: įėjimas, išėjimas, skaitymas ir rašymas. FiSMA metodas skirsto funkcines paslaugas į dvi grupes: galutinio naudotojo interfeiso paslaugos ir netiesioginės paslaugos. Šios dvi paslaugų grupės toliau skaidomos į BFK klases ir BFK tipus.

IFPUG ir NESMA metodai matuoja transakcinės funkcijos dydį priklausomai nuo jos tipo, duomenų elementų tipų ir kreipimosi failų tipų skaičiaus. Mark II metode loginės transakcijos dydis yra įvedimo, apdorojimo ir išvedimo komponentų dydžių suma. Įvedimo ir išvedimo komponentų dydis yra proporcingas unikaliai apdorotų duomenų elementų tipų, kertančių programos ribas, skaičiui. Apdorojimo komponento dydis yra proporcingas pagrindinių duomenų esybių tipų, į kuriuos kreiptasi vykdant loginę transakciją, skaičiui. COSMIC metode funkcinio proceso dydis yra proporcingas įėjimo, išėjimo, skaitymo ir rašymo duomenų judėjimų skaičiui. COSMIC metodas vykdo matavimus ne duomenų elementų tipų judėjimų lygyje, bet duomenų grupių judėjimų lygyje. FiSMA metode funkcinės paslaugos apskaičiuojamos kiekvienai BFK klasei atskirai. Priklausomai nuo BFK klasės, funkcinės paslaugos dydis yra proporcingas duomenų elementų, skaitymo ir rašymo kreipimosi į esybes skaičiui.

Atsižvelgiant į visų tiriamų funkcinio dydžio matavimo metodų matavimo skales, visi metodai naudoja absoliučią skalę gauti bazinius skaičius. IFPUG ir NESMA metodai, apskaičiavus bazinius skaičius, naudoja nominalią skalę, kad priskirtų funkcinio dydžio reikšmes kiekvienam BFK tipui. Mark II metodas priskiria svorius baziniams skaičiams, o COSMIC metodas priskiria vieną matavimo vienetą kiekvienam BFK tipo baziniam skaičiui ir naudoja santykinę skalę.

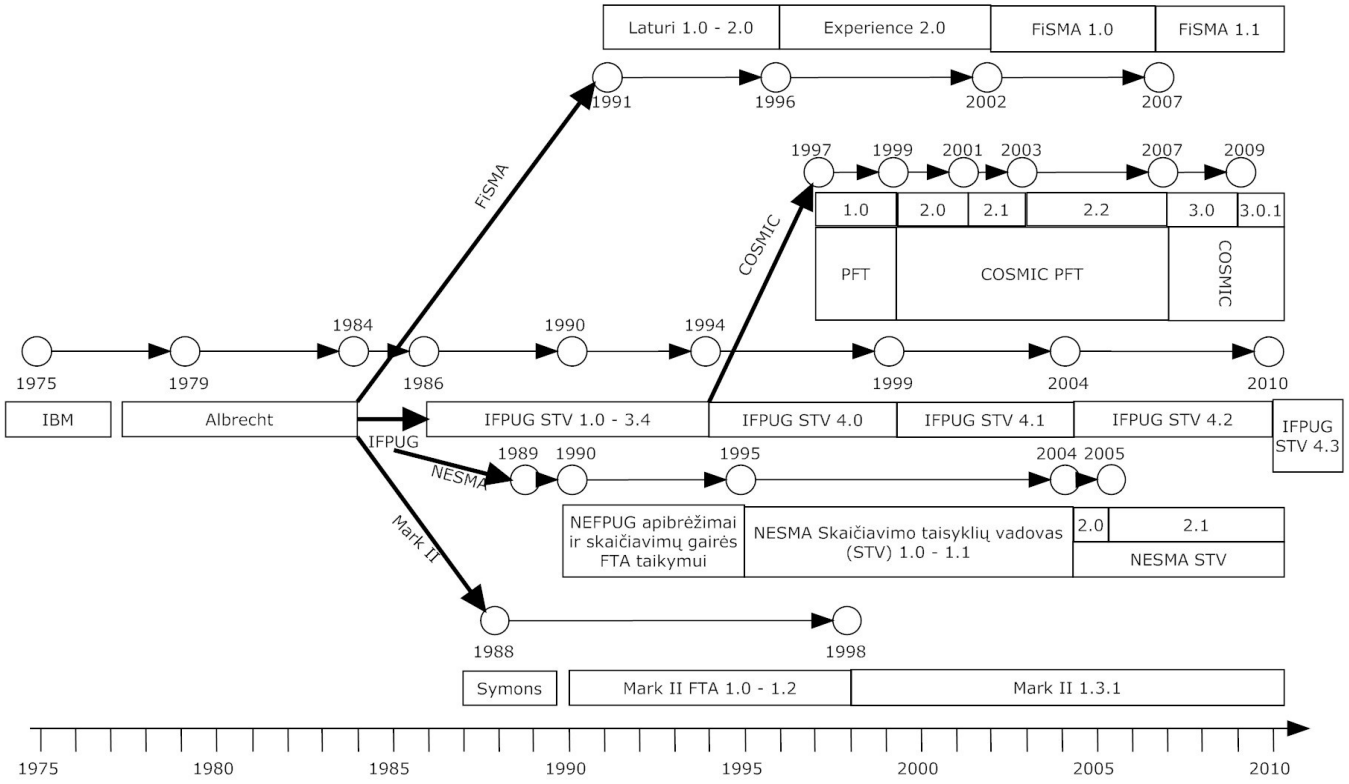
Mark II metodas funkcinio dydžio apskaičiavimui naudoja pramonės standartinių vidurkių reikšmes. Skaičiuojant BFK IFPUG ir NESMA metodais, jiems galimos priskiriamos reikšmės turi apatinę ir viršutinę ribas. Mark II, COSMIC ir FiSMA metodai tokių ribojimų neturi.

3.2. Standartizuotumas

Funkcinio dydžio matavimo metodas kaip ir bet kuris kitas matavimo metodas turi būti standartiškai apibrėžtas ir nedviprasmiškas, kad visi metodo naudotojai galėtų jį naudoti tokiu pačiu

būdu ir gautų nedviprasmiškus matavimo rezultatus. Standartų pliusas yra toks, kad jie įtvirtina geriausių praktikų žinias iš matų specialistų visame pasaulyje.

1979 metais Allan Albrecht IBM korporacijoje sukūrė funkcinių taškų analizės metodą, skirtą matuoti programinės įrangos dydį, kaip alternatyvą kodo eilučių skaičiui. 4 pav. parodytą IFPUG, NESMA, Mark II, COSMIC ir FiSMA funkcinio dydžio matavimo metodų evoliucija per pastaruosius 35 metus.



4 pav. Funkcinio dydžio matavimo metodų evoliucija

Metodų evoliucijos metu tarp jų atsirado koncepcinių nesuderinamų. 1996 metais ISO įkūrė darbo grupę, kuri turėjo išskaidrinti koncepcinį pagrindą ir apibrėžti bendrus funkcinio dydžio matavimo metodų principus. 1998 metais buvo išleistas ISO/IEC 14143-1 standartas. Šiame standarte aprašytos fundamentalios funkcinio dydžio matavimo koncepcijos, tokios kaip funkciniai naudotojų reikalavimai, baziniai funkciniai komponentai, bazinių funkcinių komponentų tipai, reikalavimai, kuriuos turi įgyvendinti metodas, norėdamas būti funkcinio dydžio matavimo metodu [GD08]. Šio ISO standarto turinio analizė buvo atlikta remiantis matavimo proceso modeliu, apibrėžtu [JA97] šaltinyje. Šis ISO dokumentas buvo analizuojamas iš tokių perspektyvų: reikalingi žingsniai matavimo metodų projektavimui, matavimo metodų taikymas ir matavimo rezultatų pateikimas. Atlikta analizė parodė, kad su keliomis išimtimis į didžiąją dalį matavimo metodo

koncepcijų buvo atsižvelgta šiame ISO dokumente [AJ99]. Šiuo metu iš viso yra išleisti šeši ISO/IEC 14143 serijos standartai. Šie standartai pateikti 7 lentelėje.

7 lentelė. ISO/IEC 14143 serijos standartai

Standarto pavadinimas	**ISO/IEC standarto numeris**
Koncepcijų apibrėžimai	ISO/IEC 14143-1:2007
Programinės įrangos dydžio matavimo metodų suderinamumo su ISO/IEC 14143-1 standartu vertinimas	ISO/IEC 14143-2:2002
Funkcinio dydžio matavimo metodų verifikavimas	ISO/IEC TR 14143-3:2003
Nuorodų modelis	ISO/IEC TR 14143-4:2002
Funkcinių sričių nustatymas funkcinio dydžio matavimui	ISO/IEC TR 14143-5:2004
ISO/IEC 14143 serijos ir susijusių tarptautinių standartų panaudojimo vadovas	ISO/IEC 14143-6:2006

ISO/IEC 14143-2 standarte aprašytos kandidatinio funkcinio dydžio matavimo metodo suderinamumo su ISO/IEC 14143-1 standartu vertinimo taisyklės. Kandidatinis metodas turi tenkinti ISO/IEC 14143-1 būtinuosius reikalavimus, kad būtų laikomas funkcinio dydžio matavimo metodu.

ISO/IEC 14143-3 standarte aprašyti procesai, skirti objektyviai įvertinti įvairius funkcinio dydžio matavimo metodo aspektus, tokius kaip pakartojamumas (angl. repeatability), atkuriamumas (angl. reproducibility), tikslumas (angl. accuracy), konvertuojamumas (angl. convertibility) ir tinkamumas funkcinėms sritims (angl. applicability to functional domains).

ISO/IEC 14143-4 standarte yra aprašytos nuorodos, skirtos funkcinio dydžio matavimo metodų verifikavimui. Šiuos nuorodos suskirstytos į verslo, realaus laiko ir mokslines sistemas. Jos yra naudingos lyginant matavimo rezultatus, gautus naudojant skirtingus funkcinio dydžio matavimo metodus.

ISO/IEC 14143-5 standarte yra aprašytos programinės įrangos klasės (funkcinės sritys), kurioms galima naudoti funkcinio dydžio matavimo metodus, o taip pat ir funkcinių naudotojų reikalavimų charakteristikos, kurios parodo, kuriai funkciniai sričiai jie priklauso.

ISO/IEC 14143-6 standartas nurodo taisykles, kuriomis reiktų naudotis renkantis standartizuotą funkcinio dydžio matavimo metodą, kuris būtų tinkamas norimai funkcinei sričiai. Funkcinio dydžio matavimo metodo taikymo metu gautas funkcinis dydis gali būti panaudojamas įvairiais tikslais viso programinės įrangos gyvavimo ciklo metu. Šiame standarte pateikiami patarimai, kaip reikia naudoti funkcinį dydį. ISO/IEC 14143-6 standartas nerekomenduoja konkretaus funkcinio dydžio matavimo metodo pasirinkimo [TK05].

Visi nagrinėjami funkcinio dydžio matavimo metodai jau yra standartizuoti. Jų standartai pateikti 8 lentelėje. Nors šie metodai naudoja skirtingas taisykles funkciniam programinės įrangos dydžiui apskaičiuoti, bet jie atitinka būtinuosius ISO/IEC 14143-1 standarto reikalavimus.

8 lentelė. Funkcinio dydžio matavimo metodų ISO/IEC standartai

Funkcinio dydžio matavimo metodas	**ISO/IEC standarto numeris**
IFPUG funkcinio dydžio matavimo metodas 2009	ISO/IEC 20926:2009
COSMIC-FFP funkcinio dydžio matavimo metodas	ISO/IEC 19761:2003
FiSMA 1.1 funkcinio dydžio matavimo metodas	ISO/IEC 29881:2008
Mark II funkcinių taškų analizė – skaičiavimų panaudojimo vadovas	ISO/IEC 20968:2002
NESMA funkcinio dydžio matavimo metodas versija 2.1 – apibrėžimai ir skaičiavimų gairės funkcinių taškų analizės taikymui	ISO/IEC 24570:2005

Kadangi ISO/IEC 14143-1 standarte apibrėžta, kad skaičiuoti funkciniam programinės įrangos dydžiui turi būti naudojami tik funkciniai naudotojų reikalavimai, bet ne nefunkciniai (techniniai ir programinės įrangos kokybės) reikalavimai, todėl IFPUG ir NESMA metodų standartuose reikšmių koregavimo faktoriaus ir Mark II standarte techninio sudėtingumo koregavimo reikšmės, bei koreguotų funkcinių taškų apskaičiavimo žingsnis laikomas neprivalomu.

Galima pastebėti, kad funkcinio dydžio matavimo vienetas yra vienintelis programinės įrangos kūrimo ir palaikymo procesų matavimo vienetas, kuris buvo formalizuotas ISO standarto lygyje. Visi kiti panašūs matai, kaip kūrimo darbo sąnaudos, trukmė, defektai ar pagaminimo greitis neturi tarptautiniu mastu priimto ir standartizuoto šių matų rinkimo, tikrinimo ir palyginimo metodo [TM07].

3.3. Industrinių duomenų prieinamumas

Kiekviena organizacija, norėdama konkuruoti su kitomis organizacijomis ir tobulėti, būtinai turi vykdyti lyginamąją analizę. Lyginamoji analizė, tai sistemingas organizacijoje vykdomų procesų matavimas ir palyginimas su geriausių organizacijų vykdomais procesais, taip nustatant pačias geriausias praktikas, ir jų pritaikymas savoje organizacijoje siekiant patobulinti esamus procesus.

Gali būti vykdoma vidinė arba išorinė lyginamoji analizė. Vidinė lyginamoji analizė vykdoma vienos organizacijos viduje, siekiant nustatyti, kurie jos padaliniai dirba blogiau arba geriau palyginus su kitais. Pagrindinė vidinės lyginamosios analizės idėja yra nustatyti geriausiai dirbančius padalinius ir pritaikyti juose vykdomas praktikas kitiems organizacijos padaliniams

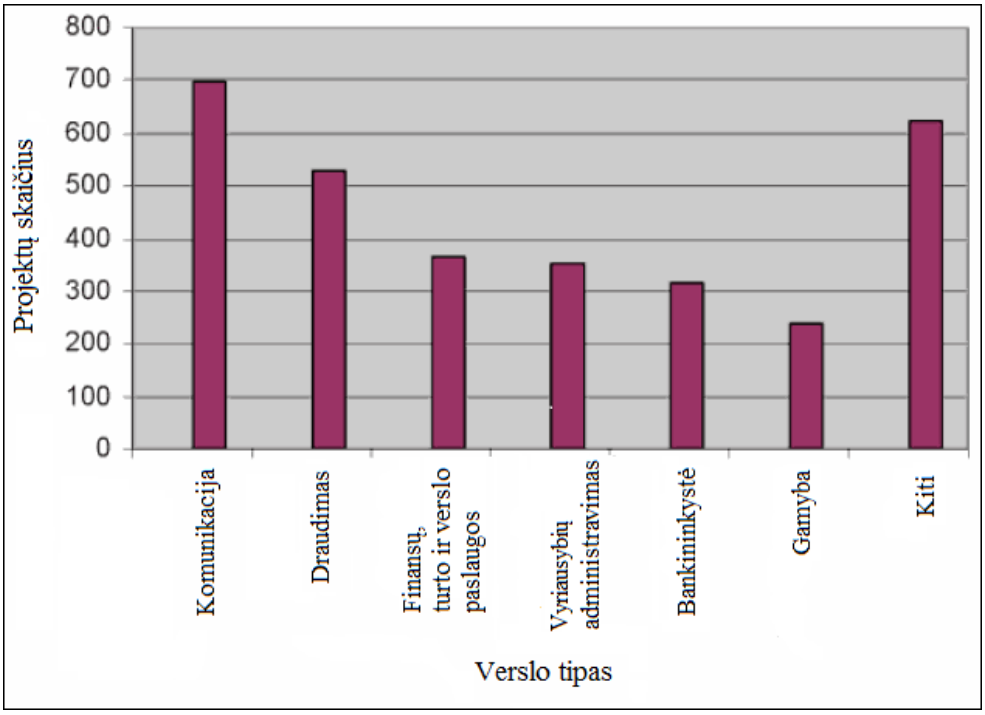
[IFPUG02]. Išorinės lyginamosios analizės metu yra lyginami organizacijos projektų duomenis su kitų, geriausių tos srities, organizacijų įvykdytų projektų duomenimis.

Vienas iš fundamentalių ir pagrindinių organizacijoje renkamų matų turėtų būti programinės įrangos dydis. Kadangi kodo eilučių skaičius nėra visuotinai standartizuotas matas, todėl organizacija, norėdama vykdyti išorinę lyginamąją analizę, turėtų kaupti atitinkamu funkcinio dydžio matavimo metodu gautą programinės įrangos funkcinio dydžio informaciją. Daugelis tarptautinių industrinių duomenų saugyklų tiekėjų kaupia įvairią projektų informaciją, kurioje figūruoja funkcinio dydžio matas. Kad kuo geriau galėtume atlikti išorinę lyginamąją analizę, reikia turėti kuo daugiau prieinamų industrinių duomenų, tokių, kurie naudojami ir tinkami savoje organizacijoje. Yra svarbu naudoti tokį funkcinio dydžio matavimo metodą, kuriuo gautų rezultatų lyginamųjų analizių saugyklose yra sukaupta daugiausia.

ISBSG yra plačiausiai naudojama lyginamosios analizės saugykla, apimanti IT projektus. Šioje saugykloje pagrindiniu programinės įrangos dydžio vienetu yra laikomi funkciniai taškai. Šiuo metu ISBSG saugykloje talpinama informacija apie 4106 užbaigtus projektus. Pagal projekto tipą ISBSG saugykloje yra sukaupta 58,5% papildymo, 39,3% naujo kūrimo, 2% perkūrimo ir 0,2% kito tipo projektų [BD08]. 5 pav. parodytas šioje saugykloje projektų skaičius pagal verslo tipą. Tarp ISBSG saugykloje sukauptų projektų buvo panaudoti visi nagrinėjami funkcinio dydžio matavimo metodai. 9 lentelėje galime pamatyti keliuose projektuose buvo panaudoti tiriami funkcinių taškų analizės metodai.

Kaip matome daugiausiai yra sukaupta informacijos apie projektus, kuriuose panaudotas IFPUG funkcinio dydžio matavimo metodas, o mažiausiai, kuriuose panaudotas Mark II metodas.

Yra tokia aktuali problema, kad būsimi arba nauji metodų naudotojai nori lyginamosios analizės duomenų, bet tik keli esami naudotojai skiria pastangų pateikti šiuos duomenis. Tam COSMIC organizacija įkūrė lyginamosios analizės komitetą. Jo tikslas yra padidinti projektų, kuriuose panaudotas COSMIC metodas, duomenų, skirtų lyginamajai analizei, kiekį ir kokybę. Duomenų kokybę siekiama pagerinti gerinant ISBSG duomenų rinkimo klausimyną. Duomenų kiekį siekiama padidinti skatinant COSMIC metodo naudotojus pateikti projektų duomenis į ISBSG saugyklą. Taip pat šis komitetas tiria įvairias konvertavimo formules, kurių pagalba iš ISBSG saugykloje esamų kitų metodų funkcinių taškų galima konvertuoti juos į COSMIC funkcinius taškus [HS08].



5 pav. ISBSG saugyklos projektų skaičius pagal verslo tipą [BD08]

9 lentelė. Funkcinio dydžio matavimo metodų panaudojimas ISBSG saugykloje [BD08]

	IFPUG	NESMA	COSMIC	Mark II	FiSMA
Projektų skaičius ISBSG saugykloje	3108	152	117	35	340

Prieš pradedant palyginimus nereikia pamiršti atidžiai pažvelgti į lyginamus rezultatus. Kad būtų galima tinkamai atlikti palyginimus, turi sutapti projektų tipai, matavimo perspektyvos ir kiti svarbūs projektų atributai. Kitu atveju gausime neadekvačius rezultatus, pagal kuriuos nebus galima priimti reikiamų sprendimų.

3.4. Konvertuojamumas

3.4.1. Metodų rezultatų tarpusavio konvertuojamumas

Neturint pakankamai duomenų atlikti lyginamąją analizę, galima vienu funkcinio dydžio matavimo metodu gautą funkcinį dydį paversti kito metodo funkciniu dydžiu. Tam reikia turėti atitinkamas konvertavimo tarp šių metodų funkcinių dydžių formules.

Reiktų pabrėžti, kad tiksli matematinė konvertavimo tarp pirmos kartos funkcinio dydžio metodo (IFPUG, NESMA, Mark II, FiSMA) ir COSMIC metodo formulė yra neįmanoma, nes pirmos kartos metodų BFK neturi tikslaus susiejimo su COSMIC metodo BFK. Taip pat skiriasi ir šių metodų matavimo taisyklės [CSMIC07].

Kad statistiškai išvesta konvertavimo formulė būtų pakankamai tiksli, reikia turėti pakankamai (nemažiau 10) apskaičiuotų funkcinių dydžių norimais funkcinio dydžio matavimo metodais. Taigi

geriausia organizacijai yra statistiškai išvesti funkcinių dydžių konvertavimo formulę iš sukauptų pakankamo kiekio funkcinio dydžio matavimo duomenų, naudojant regresinę analizę.

Panagrinėkime įvairias statistiškai išvestas nagrinėjamų funkcinio dydžio matavimo metodų konvertavimo formules. Toliau šiame skyrelyje konvertavimo formulėse IFPUG ir NESMA funkciniai taškai yra nekoreguoti funkciniai taškai, R yra koreliacinis koeficientas, o R^2 – nustatomasis koeficientas (angl. coefficient of determination).

Pirmąją funkcinių taškų konvertavimo tyrimą 1999 metais aprašė Fetke [Fet99]. Jis IFPUG ir COSMIC metodais išmatavo penkias duomenų saugojimo sistemas, bet konvertavimo formulės neišvedė. Pagal Fetke gautus duomenis Vogelezang ir Lesterhuis išvedė, o vėliau Abran, Desharnais ir Azziz pakoregavo ir gavo tokią COSMIC ir IFPUG metodų rezultatų konvertavimo formulę:

$CFT = 1{,}1 \times FT\ (IFPUG) - 7{,}6$, kur $R^2 = 0{,}97$.

Antrąjį eksperimentą 1999 metais atliko Ho [HAF99], kuris pataisė Fetke atliktus matavimus. Nors Ho neišvedė naujos konvertavimo funkcijos, pagal jo gautus rezultatus ją išvedė [CBD+10] autoriai:

$CFT = -6{,}6 + 1 \times FT\ (IFPUG)$, kur $R^2 = 0{,}98$.

Praėjus šiek tiek laiko, 2003 metais, Vogelezang ir Lesterhuis [VL03] pateikė pirmąjį NESMA ir COSMIC metodų funkcinių taškų konvertavimo tyrimą pagal išmatuotas 11 programų. Šio tyrimo autoriai išvedė tokią konvertavimo formulę:

$CFT = 1{,}2 \times FT\ (NESMA) - 86{,}8$, $R^2 = 0{,}99$.

Pagal Vogelezang ir Lesterhuis matavimo rezultatus, juose panaudoti du didžiausi projektai turi labai didelę įtaką regresiniam modeliui. Jų išvesta konvertavimo formulė nelabai tinkama mažiems projektams, nes taikant ją net galima neigiama reikšmė. Todėl mažiems projektams toje aplinkoje reikia išvesti kitokį regresinį modelį naudojant tiktai panašaus diapazono duomenis. Vogelezang ir Lesterhuis formulę [DA07] autoriai išskaidė į:

$CFT = 0{,}75 \times FT\ (NESMA) - 2{,}6$, kur $R^2 = 0{,}85$, kai $NESMA\ FT < 200$;

$CFT = 1{,}2 \times FT\ (NESMA) - 108$, kur $R^2 = 0{,}99$, kai $NESMA\ FT > 200$.

Išmatavę šešias programas Abran, Desharnais ir Aziz [ADA05] 2005 metais pristatė naują IFPUG ir COSMIC metodų rezultatų konvertavimo tyrimą. Tyrimo metu gauta formulė:

$CFT = 0{,}84 \times FT\ (IFPUG) + 18$, kur $R^2 = 0{,}91$.

Dar vieno tyrimo, atlikto 2006 metais, metu Desharnais ir Abran [DA07] ištyrę 14 valdymo informacinių sistemų (angl. management information system) tipo programų ir gavo tokią COSMIC ir IFPUG rezultatų konvertavimo formulę:

CFT = 1,0 × FT (IFPUG) – 3,22, kur R^2 = 0,93.

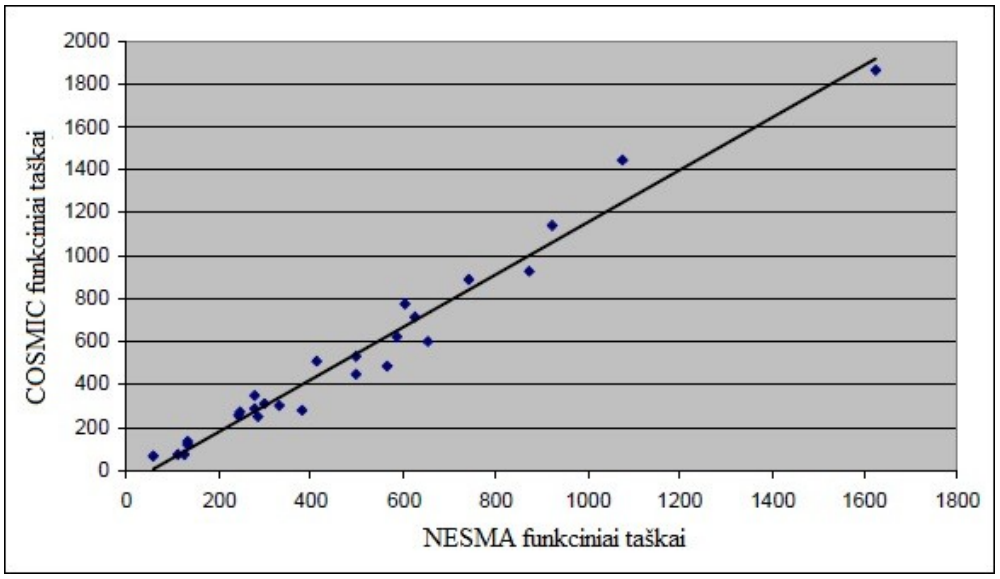
Van Heeringen [Hee07] 2007 metais ištyrė 26 naujo kūrimo tipo projektus, atliktus įvairiuose organizacijose. 6 pav. pavaizduotas šių projektų matavimo rezultatų grafikas. Van Heeringen išvedė tokia konvertavimo formulę:

CFT = 1,22 × FT (NESMA) – 64, kur R^2 = 0,97.

[CSMIC07] šaltinyje Van Heeringen statistiškai išvesta konvertavimo tarp NESMA ir COSMIC metodų funkcinių dydžių formulė yra toliau suskaidoma į:

CFT = 0,68 × FT (NESMA) + 16, kur R^2 = 0,45, kai NESMA FT < 200;

CFT = 1,24 × FT (NESMA) - 80, kur R^2 = 0,96, kai NESMA FT > 200.



6 pav. NESMA ir COSMIC funkciniai dydžiai 26 projektuose [CSMIC07]

7 pav. parodyta 39 projektų funkciniai dydžiai apskaičiuoti Mark II ir IFPUG funkcinio dydžio matavimo metodais [Sym99]. Symons statistiškai išvestos Mark II ir IFPUG rezultatų konvertavimo formulės, galiojančios iki 1500 IFPUG arba 2500 Mark II dydžio funkcinių taškų, atrodo taip (R^2 nepateiktas):

FT (Mark II) = 0,9 × FT (IFPUG) + 0,0005 × FT $(IFPUG)^2$;

FT (IFPUG) = 1000 × (SQRT(0,81 + 0,002 × FT (Mark II)) – 0,9.

[BD08] šaltinyje taip pat pateiktos Mark II ir IFPUG metodų funkcinio dydžio pagal ISBSG saugykloje saugomus duomenis konvertavimo formulės (R^2 nepateiktas):

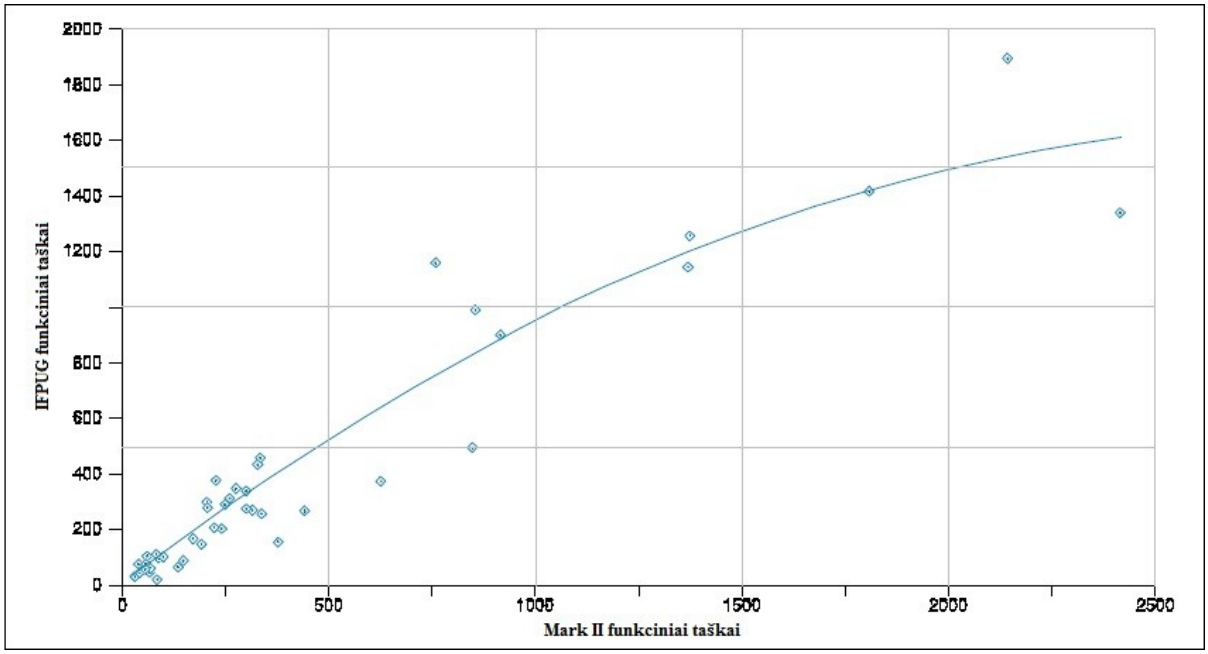
FT (IFPUG) = 41,4 + 0,77 × FT (Mark II);

FT (Mark II) = 20,3 + 1,25 × FT (IFPUG).

Reikia paminėti, kad NESMA metodas nors ir šiek tiek skiriasi nuo IFPUG metodo, bet, kaip aiškiai pažymėta NESMA skaičiavimo taisyklių vadove, jų skaičiavimo rezultatai gali būti laikomi

ekvivalenčiais [CBD+10]. Kaip minima [BD08] šaltinyje, NESMA ir IFPUG metodai tampa vis panašesni, ir šiuo metu jie jau 95-99% panašūs. Iš to galima spręsti, kad konvertavimo formulė tarp NESMA ir IFPUG metodų rezultatų yra:

FT (NESMA) = FT (IFPUG).



7 pav. Mark II ir IFPUG nekoreguoti funkciniai taškai [Sym99]

Šiuo metu kol kas nėra žinoma jokių konvertavimo formulių tarp FiSMA ir kitų funkcinio dydžio matavimo metodų funkcinių dydžių [FSMA07].

Kaip matome, daugiausia statistiškai išvestų funkcinio dydžio konvertavimo formulių turi IFPUG, o tuo pačiu ir NESMA funkcinio dydžio matavimo metodai. Jais gautus rezultatus galima konvertuoti į COSMIC ir Mark II metodų funkcinius dydžius.

Taip pat pateikta daug įvairių COSMIC funkcinio dydžio konvertavimo formulių iš IFPUG ir NESMA funkcinio dydžio. Mark II metodo rezultatų konvertavimas pateiktas tiktai į ir iš IFPUG metodo, o FiSMA metodo funkcinio dydžio konvertavimui kol kas nepateiktos išvis jokios konvertavimo formulės.

3.4.2. Kodo eilučių skaičiaus konvertavimas į funkcinius taškus

Jeigu yra žinomas programinės įrangos kodo eilučių skaičius ir naudota programavimo kalba, galima šį skaičių išreikšti funkciniais taškais. Tam yra naudojamos programavimo kalbų specifinės konstantos. Šis kodo eilučių skaičiaus pavertimas funkciniais taškais angliškai yra vadinamas „backfiring“. Toks konvertavimas dažniausiai yra naudojamas kai yra žinomas tik kodo eilučių skaičius, o funkcinių taškų analizė yra negalima arba ją taikyti yra nepraktiška. Kadangi funkcinių taškų matas yra nepriklausomas nuo naudojamos programavimo kalbos kuriant programinę įrangą,

funkciniai taškai įgalina palyginimus tarp skirtingų sistemų. Kodo eilučių skaičiaus konvertavimas į funkcinius taškus suteikia greitą ir nebrangų būdą apskaičiuoti funkcinį dydį. Taip pat gali būti naudojamas ir atvirkščias šio konvertavimo būdas, kai norima gauti kodo eilučių skaičių iš funkcinių taškų. To gali prireikti, kai pačioje projekto pradžioje prireikia kodo eilučių skaičiaus įvairiems kaštų vertinimo modeliams [DG00].

Caper Jones 1991 metais pirmasis pateikė programavimo kalbų konvertavimo koeficientus. Jie parodyti 10 lentelėje. Vėliau Caper Jones kartu su SPR (angl. Software Productivity Research) organizacija papildė šį sąrašą iki 700 programavimo kalbų. Panašius konvertavimo koeficientus pateikia DCG (angl. David Consulting Group) [DCG06] ir QSM (angl. Quantitative Software Management) [QSM09] organizacijos. Reikia pabrėžti, kad yra naudojamos loginių, o ne fizinių kodo eilučių skaičius. Gaunami funkciniai taškai yra IFPUG funkciniai taškai.

10 lentelė. Programavimo kalbų konvertavimo koeficientai [BD08]

Programavimo kalba	**Programavimo kalbų konvertavimo koeficientas (kodo eilučių skaičius per funkcinį tašką)**	**Programavimo kalba**	**Programavimo kalbų konvertavimo koeficientas (kodo eilučių skaičius per funkcinį tašką)**
Basic Assembler	320	IBM CICS/VS	40
Macro assembly	213	ORACLE	40
C	128	Visual Basic	35
ANSI COBOL 74	107	Visual C++	34
Fortran	107	SAS	32
Interpreted Basic	107	Symantec C++	29
ANSI COBOL 85	107	EIFEL	21
Fortran 90	80	SmallTalk/V	21
Microfocus Cobol	80	ABAP/4	16
LISP	64	Programų generatoriai	16
C++	55	ANSI SQL	13
CICS	46	Excel 3-4	6

Žinoma, organizacijai yra geriau naudoti pagal savo duomenis išvestus programavimo kalbų konvertavimo koeficientus. Taip bus užtikrinamas šių koeficientų tinkamumas esamai organizacijai. Taip pat naudojant savus projektų duomenis galima patikrinti viešai skelbiamų programavimo kalbų konvertavimo koeficientų tinkamumą savai organizacijai.

Kodo eilučių skaičiaus konvertavimą į funkcinius taškus patariama naudoti tik tada, kai nebėra jokių kitų funkcinių taškų nustatymo būdų. Šiuo būdu gauti rezultatai nuo tikrųjų gali skirtis net iki 400%. Bet žinoma kartais netikslūs matavimai yra geresni negu neatlikti jokie matavimai.

Matome, kad pateiktuose konvertavimo koeficientų lentelėse yra naudojami IFPUG funkciniai taškai. Šios lentelės tinkamos ir NESMA metodo naudotojams, bet trūksta nustatytų konvertavimo koeficientų tinkamų kitiems nagrinėjamiems funkcinio dydžio matavimo metodams.

3.5. Metodų taikymo fazė

Panagrinėkime, kada yra tinkamiausia taikyti nagrinėjamus funkcinio dydžio matavimo metodus programinės įrangos kūrimo gyvavimo ciklo metu. Taip pat panagrinėkime, ar galima pritaikyti šiuos metodus ir anksčiau, bei jų taikymo tinkamumas palaikymo (angl. maintenance) fazės metu.

3.5.1. Tinkamiausia metodų taikymo fazė

Optimalus pirmasis visų nagrinėjamų funkcinio dydžio matavimo metodų pritaikymas yra po atliktos reikalavimų analizės. Reikalavimų analizės metu yra gaunamas naudotojų reikalavimų aprašas ir duomenų struktūrų aprašas. To visiškai pakanka funkcinių taškų skaičiavimui visais nagrinėjamais metodais.

Reikalavimai įtakoja visas programinės įrangos gyvavimo ciklo fazes. Todėl dviprasmiškai, nepilnai ir neteisingai apibrėžti reikalavimai turi neigiamą poveikį visoms po reikalavimų analizės einančioms fazėms. Blogų reikalavimų neaptikimas dažniausiai baigiasi papildomu darbu ištaisyti ankstesnėse gyvavimo ciklo fazėse atliktą darbą. Kad minimizuotume darbo sąnaudas, skirtas šiam papildomam darbui ir defektų taisymo vėlesnėse programinės įrangos kūrimo gyvavimo ciklo fazėse išlaidas, daugelis organizacijų naudoja įvairius reikalavimų dokumentų peržiūros metodus [TA08]. Kaip vieną iš šių technikų gali būti naudojama ir funkcinių taškų analizė. Todėl kaip vieną iš teigiamų nepagrindinių funkcinio dydžio matavimo metodų taikymo pasekmių galima paminėti defektų radimą reikalavimų specifikacijose.

Vienkartinis funkcinių taškų apskaičiavimas viso projekto metu neduos didelės naudos, nes vykdant projektą gali atsirasti naujos informacijos, kaip apimties pokyčiai ar reikalavimų patikslinimai. Todėl yra rekomenduojama atlikinėti funkcinių taškų analizę po kiekvienos toliau einančios programinės įrangos kūrimo projekto fazės. Funkcinio dydžio perskaičiavimas skirtingu projekto vykdymo metu patikslina tikrąsias projekto darbo sąnaudas [BD08].

Pateikime keletą galimų pagrindinių matų, kuriuose naudojami funkciniai taškai. Šie matai suskirstyti į keturias kategorijas: produktyvumo, kokybės, finansinius ir palaikymo. Jie pavaizduoti

11 lentelėje. Šie matai gali būti naudojami įvairiuose sistemų kūrimo gyvavimo ciklo fazėse. Matų ir programinės įrangos kūrimo gyvavimo ciklo fazių sąryšis parodytas 12 lentelėje.

11 lentelė. Keletas efektyviai panaudojamų funkcinių taškų matų

Matas	**Aprašymas**	**Formulė**
Produktyvumo		
Valandos funkciniam taškui	Skaičiuoja valandų kiekį, reikalingą sukurti vieną funkcinį tašką	Bendras projekto valandų skaičius / Sukurtų funkcinių taškų skaičius
Pagaminimo greitis (angl. rate of delivery)	Laikas, reikalingas pagaminti programinę įrangą galutiniam naudotojui	Funkcinių taškų skaičius / Praėjęs kalendorinis laikotarpis
Sukurtas funkcionalumas	Parodo sukurto funkcionalumo pagaminimo produktyvumą	Bendros kūrimo darbo sąnaudų valandos / Bendras sukurtas funkcionalumas
Kokybės		
Funkcinių reikalavimų dydis	Parodo bendrą funkcijų, reikalaujamų galutinio naudotojo, skaičių, išreikštą funkciniais taškais	Reikalaujamų funkcinių reikalavimų skaičius funkciniais taškais
Išbaigtumas	Pagaminto palyginti su reikalaujamo funkcionalumo matas	Pagamintas funkcionalumas / Reikalaujamas funkcionalumas
Pokyčio greitis (angl. rate of change)	Matuoja funkcinės specifikacijos pokytį, atsiradusį kūrimo proceso metu	Pakitę reikalavimai / Sutarti reikalavimai
Defektų tankumas (angl. defect density)	Parodo rastų defektų skaičių tam tikroje programinės įrangos kūrimo projekto gyvavimo ciklo fazėje palyginti su bendru kuriamos programos dydžiu	Defektų skaičius fazėje / Bendras funkcinių taškų skaičius
Testavimo atvejų apimtis (angl. test case coverage)	Parodo testavimo atvejų skaičių, reikalingą atlikti	Testavimo atvejų skaičius / Bendras funkcinių taškų

Matas	Aprašymas	Formulė
	nuodugnų kūrimo projekto testavimą	skaičius
Dokumentacijos apimtis (angl. volume of documentation)	Dokumentacijos, sudarytos kūrimo projekto gyvavimo ciklo metu, puslapių skaičius nuo kūrimo darbo sąnaudų	Puslapių skaičius per dokumentą / Bendras funkcinių taškų skaičius
Finansiniai		
Kaštai funkciniam taškui	Pagaminto funkcinio taško kaštai	Bendri kaštai / Bendras funkcinių taškų skaičius
Pataisymo kaštų koeficientas (angl. repair cost ratio)	Naudojamas sekti kaštus, reikalingus pataisyti veikiančias programas	(Bendras valandų skaičius taisymui × kaštai per valandą) / Programos funkcinių taškų skaičius
Palaikymo		
Palaikymas	Palaikyti programinę įrangą reikiamų kaštų matas	Palaikymo kaštai / Programos funkcinių taškų skaičius
Patikimumas	Patirtų gedimų skaičius palyginti su bendru programos dydžiu	Gedimų skaičius / Bendras programos funkcinių taškų skaičius
Paskyrimo apimtis (angl. assignment scope)	Viso darbo laiko (angl. full-time) resursų, reikalingų programos palaikymui, skaičius	Bendras programos funkcinių taškų skaičius / Viso darbo laiko resursų, reikalingų palaikyti programą, skaičius
Augimo greitis (angl. rate of growth)	Parodo programos funkcionalumo augimą per apibrėžtą laiko tarpą	Esamų funkcinių taškų skaičius / Pirminis funkcinių taškų skaičius
Stabilumo koeficientas (angl. stability ratio)	Skirtas parodyti, kaip efektyviai programa arba atlikti pakeitimai atitinka naudotojų lūkesčius	Pokyčių skaičius / Programos funkcinių taškų skaičius

12 lentelė. Funkcinių taškų matų panaudojimas sistemų kūrimo gyvavimo cikle [GH01]

Matas	**Programinės įrangos kūrimo fazė**					
	Reikalavimų	**Projektavimo**	**Kūrimo**	**Testavimo**	**Įgyvendinimo**	**Palaikymo**
Produktyvumo						
Valandos funkciniam taškui	**X**	**X**	**X**	**X**	**X**	
Pagaminimo greitis	**X**	**X**	**X**	**X**	**X**	
Sukurtas funkcionalumas					**X**	
Kokybės						
Funkcinių reikalavimų dydis	**X**					
Išbaigtumas					**X**	
Pokyčio greitis	**X**	**X**	**X**	**X**	**X**	
Defektų tankumas	**X**	**X**	**X**	**X**	**X**	
Testavimo atvejų apimtis				**X**		
Dokumentacijos apimtis					**X**	
Finansiniai						
Kaštai funkciniam taškui					**X**	
Pataisymo kaštų koeficientas						**X**
Palaikymo						
Palaikymas						**X**
Patikimumas					**X**	**X**
Paskyrimo apimtis						**X**
Augimo greitis						**X**
Stabilumo koeficientas						**X**

Kai yra reikalinga apskaičiuoti projektui atlikti reikiamas darbo sąnaudas ankstesnės nei reikalavimų analizės fazės pabaigos metu, yra rekomenduojama sukurti funkcinių taškų prognozės metodą (angl. function point prognosis method). Tam reikia įvertinti jau atliktų programinės įrangos kūrimo projektų istorinę informaciją ir atlikti regresinę analizę. Paprastai tam gali užtekti ir 16-20 užbaigtų projektų informacijos [BD08].

Taip pat yra labai svarbu kaupti projekto informaciją ir įgautą patirtį po projekto užbaigimo. Taip bus galima pagerinti dabartinius naudojamus matavimo metodus tolimesniems jų taikymams vėlesniuose projektuose. Dažnai užbaigus projektą nėra fiksuojami tokie darbo sąnaudų mato komponentai, kaip neapmokami viršvalandžiai, projektų vadybos darbo sąnaudos, kokybės užtikrinimo darbo sąnaudos, administravimo darbo sąnaudos, galutinių naudotojų ir techninių specialistų, nepriklausančių pagrindinei kūrimo komandai, darbo sąnaudos. Kad būtų galima tobulėti ir mokytis iš ankstesnių projektų, yra labai svarbu fiksuoti visas projektuose panaudotas darbo sąnaudas.

Svarbu yra atsiminti, kad joks vienas matas nesuteikia visos reikiamos informacijos, todėl programinės įrangos matavimas yra efektyvus tik tada, kai tarpusavyje naudojami ir analizuojami įvairūs matai.

3.5.2. Ankstyvas metodų taikymas

Pasitaiko taip, kad reikia apskaičiuoti būsimas darbo sąnaudas dar prieš įvykdant reikalavimų analizę, o tam reikia apskaičiuoti funkcinį programinės įrangos dydį. Čia praverčia funkcinių taškų prognozės arba įvairus ankstyvo taikymo metodai. Panagrinėkime juos ir jų sąsajas su nagrinėjamais penkiais funkcinio dydžio matavimo metodais.

Olandų programinės įrangos matų asociacija pateikia tris skirtingus funkcinių taškų skaičiavimo NESMA metodu tipus [NSMA10a]:

- **detalusis funkcinių taškų skaičiavimas**. Tai standartinis funkcinių taškų skaičiavimas, kurio metu nustatomos visos transakcinės ir duomenų funkcijos, joms priskiriami sudėtingumai ir apskaičiuojamas bendras funkcinis dydis;
- **apytikris funkcinių taškų skaičiavimas**. Šio tipo skaičiavimo metu taip pat nustatomos visos transakcinės ir duomenų funkcijos. Kiekvienai duomenų funkcijai (VLF, IIF) priskiriamas žemas sudėtingumas, o transakciniai funkcijai (IĮ, II, IU) vidutinis sudėtingumas. Tada apskaičiuojamas bendras funkcinis dydis;
- **indikatyvus funkcinių taškų skaičiavimas**. Šio tipo skaičiavimo metu yra nustatomos tik duomenų funkcijos (VLF, IIF), o bendras funkcinis dydis apskaičiuojamas pagal tokią formulę:
    - indikatyvus funkcinis dydis = 35 × VLF kiekis + 15 × IIF kiekis.

NESMA apytikris ir indikatyvus funkcinių taškų skaičiavimai buvo sukurti norint juos pritaikyti ankstyvesniu metu negu įprastą detalųjį skaičiavimą. Apytikriame funkcinių taškų skaičiavime naudojami iš anksto numatyti funkcijų sudėtingumai. Indikatyvus funkcinio dydžio

skaičiavimas remiasi prielaida, kad skaičiavime vidutiniškai bus apie tris IĮ (pridėti, pakeisti ir pašalinti informaciją, esančią VLF), du II, vienas IU kiekvienam VLF ir apie vieną II ir vieną IU kiekvienam IIF.

Dar vienas iš ankstyvo taikymo būdų yra esamiems metodams apibrėžti taisykles ir BFK tipus, kuriuos bus galima pritaikyti skirtingo detalumo lygiuose. Yra pasiūlyta įvairių tokių metodų kaip, pavyzdžiui, ankstyva ir greita funkcinių taškų analizė paremta IFPUG ir COSMIC metodais, bei supaprastinti funkciniai taškai paremti IFPUG metodu [GD08], o taip pat ir įvairūs funkcinio dydžio apytikrio skaičiavimo COSMIC metodu būdai kaip, pavyzdžiui, vidutinio funkcinio proceso, fiksuoto dydžio klasifikavimo būdai [CSMIC07] ir pan.

Dar vienas supaprastintas IFPUG metodo variantas yra SPR funkciniai taškai. Šis metodas supaprastino funkcinių taškų skaičiavimą panaikindamas funkcijų sudėtingumų klasifikavimą. Visos duomenų ir transakcijų funkcijos yra įvertinamos vidutiniu sudėtingumu. Papildomai 14 BSC yra sutrauktos tiktai iki dviejų: problemos sudėtingumas ir duomenų sudėtingumas [BD08]. Naudojant įvairius ankstyvo taikymo metodus, žinoma, yra geriausia naudoti ne metodu siūlomus vidurkius, o pagal organizacijos istorinę informaciją susidarytus nuosavus vidurkius.

Ankstyvam ir greitam metodų pritaikymui taip pat galima taikyti ir proporcijų metodą. Tam, pavyzdžiui, galima pasitelkti ISBSG saugyklos duomenis. Atrinkus panašaus tipo projektus, įvertintus tuo pačiu funkcinio dydžio matavimo metodu, pagal saugyklos duomenis galima nustatyti, kiek kokių komponentų proporcionaliai sudaro kuriamą programinę įrangą. Naudojant IFPUG ir NESMA metodus galima, pavyzdžiui, nustatyti VLF ir IIF, o IĮ, II ir IU apskaičiuoti pasitelkus proporcijos formulę pagal saugykloje esamus duomenis. Bet vėlgi, nereikia pamiršti, kad tam projektų saugykloje reikia turėti pakankamai daug patikimų duomenų lyginamajai analizei.

Nors ankstyvieji funkcinių taškų skaičiavimo metodai gali būti taikomi ankščiau ir greičiau nei standartiniai, bet jais gaunami ne tokie tikslūs rezultatai. Tačiau praktinis ankstyvos ir greitos funkcinių taškų analizės panaudojimas rodo, kad gaunami rezultatai nuo tikrųjų dažniausiai skiriasi tik apie ± 10%, o sugaištas matavimo laikas būna 50-90% trumpesnis [SCM05].

Kaip matome yra įvairių ankstyvo ir greito funkcinių taškų vertinimo metodų, paremtų IFPUG, NESMA ir COSMIC metodais, bet trūksta tokių metodų pagal Mark II ir FiSMA metodus.

3.5.3. Metodų taikymas palaikymo fazės metu

Kai programinė įranga yra sukurta ir įdiegta, tada ją reikia palaikyti (modifikuoti), kad programinė įranga ir toliau galėtų tenkinti amžinai besikeičiančius verslo ir techninės aplinkos poreikius. Palaikymas apima daug įvairių veiklų, kurios yra atliekamos šios programinės įrangos

gyvavimo ciklo fazės metu. Kai kurios iš jų apima funkcinius pokyčius, kuriems galima pritaikyti funkcinių taškų analizę.

IEEE apibrėžia tris palaikymo tipų kategorijas [IFPUG05]:

- **pritaikomasis palaikymas (angl. adaptive maintenance)**. Programinės įrangos palaikymas, siekiant kompiuterinę programą pritaikyti pasikeitusiai aplinkai;
- **korekcinis palaikymas (angl. corrective maintenance)**. Programinės įrangos palaikymas, siekiant ištaisyti techninės arba programinės įrangos defektus;
- **tobulinamasis palaikymas (angl. perfective maintenance)**. Programinės įrangos palaikymas, siekiant patobulinti kompiuterinę programą.

Panagrinėkime kaip tiriami funkcinių taškų analizės metodai gali būti naudojami palaikymo fazės metu, kokias rekomendacijas ar metodikas šiuos metodus sukūrusios organizacijos siūlo naudoti matuojant funkcinį dydį palaikymo fazės metu.

3.5.3.1. IFPUG palaikymo fazės rekomendacijos

IFPUG pabrėžia, kad programinės įrangos palaikymas nepakeičia jos funkcionalumo. Jeigu projekto metu yra sukuriamas naujas, papildomas arba pašalinamas senas funkcionalumas, toks projektas yra klasifikuojamas kaip papildantysis (angl. enchancement) projektas. IFPUG skaičiavimo taisyklių vadove [IFPUG05] yra pateiktos kiekvieno palaikymo tipo skaičiavimo rekomendacijos.

Pritaikomasis palaikymas apima pakeitimus, skirtus patenkinti naujus arba pasikeitusius verslo reikalavimus arba pridėti naują funkcionalumą, kuris nėra pateiktas ankstesnėje sistemos laidoje. Taip pat gali apimti ir pakeitimus, reikalingus patenkinti besikeičiančius techninius reikalavimus. Pritaikomasis palaikymas yra inicijuojamas verslo poreikio pridėti, pakeisti arba pašalinti verslo funkcionalumą, kas yra analogiška IFPUG metodo papildančio projekto koncepcijai.

Korekcinis palaikymas apima pakeitimus, skirtus pataisyti defektus. Jis neapima verslo funkcionalumo pakeitimų, bet užtikrina, kad anksčiau sukurtas funkcionalumas būtų taisyklingai vykdomas. Darbo sąnaudos atlikti korekcinį palaikymą turi būti priskirtos originaliam kūrimo arba papildymo projektui, kurio metu ir atsirado šie defektai.

Tobulinamasis palaikymas gali apimti pakeitimus, skirtus palaikyti platformos arba sistemos programinės įrangos pagerinimus (angl. upgrades), vykdymo optimizavimui ir pan. Šio palaikymo metu nėra keičiamas sistemų funkcionalumas. Nors IFPUG metodas nėra tinkamas vertinti tobulinamąjį palaikymą, bet šis palaikymas gali turėti įtakos bendrosioms sistemos charakteristikoms, kurios turėtų būti peržiūrimos.

Funkcinio dydžio matavimas nustato verslo reikalavimų dydį. Pakitusioje aplinkoje jis matuoja pokyčio padarinius tiems verslo reikalavimams. Todėl funkcinio dydžio matavimas gali būti taikomas pritaikomajam palaikymui, kurio metu yra pridedamas, pakeičiamas arba pašalinamas programinės įrangos funkcionalumas, o taip pat yra vykdomas ir duomenų konvertavimas.

3.5.3.2. NESMA papildančio projekto metodika

NESMA organizacija išleido atskiras rekomendacijas kaip matuoti papildančius projektus [NSMA09]. Papildymas reiškia informacinės sistemos funkcionalumo pokytį, kuris įgyvendinamas pritaikomojo palaikymo metu. Kito tipo palaikymas kaip, pavyzdžiui, korekcinis arba tobulinamasis, nėra įtraukti į rekomendacijas, nes jų metu informacinės sistemos funkcionalumas nesikeičia. Šiose rekomendacijose yra apibrėžiamas naujas metodas, neįtrauktas nei į NESMA, nei į IFPUG skaičiavimo taisyklių vadovus.

Pirmiausia apskaičiuojamas bendras naujai pridėtų duomenų ir transakcinių funkcijų dydis ($BFT_{pridėtas}$), kuris yra lygus visų naujai pridėtų funkcijų dydžiui ($FT_{pridėtas}$):

$BFT_{pridėtas} = FT_{pridėtas}$

Skaičiuojant pašalintų duomenų ir transakcinių funkcijų bendrą dydį ($BFT_{pašalintas}$), naudojama formulė:

$BFT_{pašalintas} = FT_{pašalintas} \times 0,4$

Duomenų funkcijų (DF) procentinis pokytis apskaičiuojamas:

DF Procentinis pokytis = Pridėtų, pakeistų, pašalintų duomenų elementų tipų skaičius × 100 / Originalus duomenų funkcijų duomenų elementų tipų skaičius

Panaudojant duomenų funkcijų poveikio faktorius, pateiktus 13 lentelėje, nustatome poveikio faktorių ($P_{DF\ pakeistas}$).

13 lentelė. Duomenų funkcijų poveikio faktorius [NSMA09]

DF Procentinis pokytis	≤ 33%	≤ 67%	≤ 100%	> 100%
Poveikio faktorius	0,25	0,5	0,75	1

Pakitusių duomenų funkcijų bendras dydis ($BFT_{pakeistas}$) apskaičiuojamas:

$BFT_{DF\ pakeistas} = FT_{DF\ pakeistas} \times P_{DF\ pakeistas}$

Transakcinėms funkcijoms (TF) atitinkamai skaičiuojamas duomenų elementų tipų ir kreipimosi failų tipų procentinis pokytis:

TF DET Procentinis pokytis = Pridėtų, pakeistų, pašalintų duomenų elementų tipų skaičius × 100 / Originalus transakcinių funkcijų duomenų elementų tipų skaičius

TF KFT Procentinis pokytis = Pridėtų, pakeistų, pašalintų kreipimosi failų tipų skaičius × 100 / Originalus transakcinių funkcijų kreipimosi failų tipų skaičius

Panaudojant transakcinių funkcijų poveikio faktorius, pateiktus 14 lentelėje, nustatome poveikio faktorių ($P_{TF\ pakeistas}$).

14 lentelė. Transakcinių funkcijų poveikio faktorius [NSMA09]

TF KFT Procentinis pokytis	**TF DET Procentinis pokytis**		
	≤ 67%	≤ 100%	> 100%
≤ 33%	0,25	0,5	0,75
≤ 67%	0,5	0,75	1
≤ 100%	0,75	1	1,25
> 100%	1	1,25	1,5

Pakitusių transakcinių funkcijų bendras dydis ($BFT_{TF\ pakeistas}$) apskaičiuojamas:

$BFT_{TF\ pakeistas} = FT_{TF\ pakeistas} \times P_{TF\ pakeistas}$

Bendras pokyčių dydis ($BFT_{pakeistas}$) gaunamas:

$BFT_{pakeistas} = BFT_{DF\ pakeistas} + BFT_{TF\ pakeistas}$

Nekoreguotas papildančio projekto ($NPFT_{projektas}$) funkcinis dydis yra suma visų papildytų transakcinių ir duomenų funkcijų funkciniam dydžiui:

$NPFT_{projektas} = BFT_{pridėtas} + BFT_{pakeistas} + BFT_{pašalintas}$

Galiausiai nustačius reikšmių koregavimo faktorių prieš projektą ($RKF_{prieš}$) ir po projekto (RKF_{po}), apskaičiuojamas papildančio projekto dydis ($PFT_{projektas}$) koreguotais funkciniais taškais:

$PFT_{projektas} = BFT_{pridėtas} \times RKF_{po} + BFT_{pakeistas} \times RKF_{po} + BFT_{pašalintas} \times RKF_{prieš}$

3.5.3.3. FiSMA standartiniai produktyvumo faktoriai

FiSMA organizacija, remdamasi savo patirtimi, išleido specialų, atskirą nuo funkcinio dydžio matavimo, dokumentą [FSMA01a], kuriame aprašė 22 standartinius produktyvumo faktorius, kurie įtakoja darbo sąnaudas, reikalingas palaikyti programinę įrangą. Juos galima pamatyti 2 priede. Šie faktoriai klasifikuoti į keturias skirtingas kategorijas. Kiekvienas faktorius įvertinamas eiliškumo skalėje pagal situacijos gerumą, kurios penkios reikšmės yra: labai gera, gera, vidutinė, bloga, labai bloga. Pagrindinė įvertinimo idėja yra ta, kad kuo geresnės yra palaikymo sąlygos, tuo geresne reikšme yra įvertinama faktoriaus situacija.

3.5.3.4. UKSMA programinės įrangos palaikymo matavimo standartas

UKSMA kartu su ISBSG organizacija 2001 metais, kaip kokybės matavimo standarto dalį, išleido programinės įrangos palaikymo ir priežiūros matavimo standartą. Šiame standarte

atskiriamas palaikymo, priežiūros ir eksploatavimo darbas (paskirstymas, išleidimas, duomenų bazės valdymas, informacijos paieškos palaikymas) nuo sistemų kūrimo ir tobulinimo. Šio standarto tikslas yra apibrėžti įvairius su programinės įrangos palaikymu ir priežiūra susijusius matus, tarp kurių yra ir funkciniai taškai [BD08].

Kaip matome tik COSMIC funkcinio dydžio matavimo metodui nėra pateiktos jokios taisyklės ar rekomendacijos, kaip naudoti šį metodą programinės įrangos palaikymo fazės metu.

3.6. Subjektyvumas metoduose

Panagrinėkime tiriamų penkių standartizuotų funkcinio dydžio matavimo metodų panaudojimo subjektyvumą.

Vienas iš pagrindinių IFPUG metodo trūkumų yra tai, kad funkcinių taškų skaičiavimo proceso metu yra atliekama daug subjektyvių vertinimų, skirtingų vertintojų apskaičiuotas galutinis funkcinis dydis gali labai skirtis. Viena iš šio metodo panaudojimo subjektyvumo vietų yra galimas skirtingas funkcinių reikalavimų vertinimas. Pavyzdžiui, kartais gali būti neaišku, ar funkciją vertinti kaip IĮ ar kaip IIF tipo. Taip pat yra labai subjektyvus ir nuo vertintojų nuomonės priklausantis funkcijų sudėtingumų vertinimas. Skirtingas funkcijų sudėtingumų vertinimas gali labai paveikti galutinį funkcinį dydį, nes sudėtingumų reikšmės ganėtinai labai skiriasi. Žinoma, nereikia pamiršti ir BSC vertinimo subjektyvumo [Jal05]. Vertintojas turi nuspręsti, kokią įtaką projektui turi kiekviena bendroji sistemos charakteristika. Dažniausiai skirtingi vertintojai skirtingose organizacijose turės skirtingus šių charakteristikų vertinimo būdus [Vic03]. Šie visi paminėti elementai padaro IFPUG metodo skaičiavimus ganėtinai subjektyvius. Dėl didelio metodų panašumo, visa tai galioja ir NESMA metodui.

Nors su didesne patirtimi vertintojai gauna panašesnius rezultatus, tačiau, norint labiau sumažinti subjektyvumus, IFPUG ir NESMA organizacijos turėtų apibrėžti tikslesnes funkcinių taškų skaičiavimo taisykles.

Mark II metodas sumažina subjektyvumą, skaičiuodamas esybes vietoje duomenų rinkinių. Bet nereikia pamiršti, kad Mark II metodas taip pat naudoja BSC, ir net 5 daugiau negu IFPUG ir NESMA metodai. Tuo tarpu šiuo požiūriu COSMIC ir FiSMA metodų taikymas yra objektyvesnis.

3.7. Nefunkciniai naudotojų reikalavimai

Funkciniai naudotojų reikalavimai, tai poaibis naudotojo reikalavimų, kurie apibūdina, ką turės daryti programinė įranga užduočių ir paslaugų terminais [ISO07]. Nors standartizuotais funkcinio dydžio matavimo metodais turi būti vertinami tik funkciniai naudotojų reikalavimai, tačiau panagrinėkime kaip šiais metodais galima vertinti ir nefunkcinius naudotojų reikalavimus.

Nefunkciniai naudotojų reikalavimai gali būti [Mor06]:

- **kokybės apribojimai**. Pavyzdžiui, panaudojamumas, patikimumas, efektyvumas, pernešamumas;
- **organizacijos ribojimai**. Pavyzdžiui, sistemos naudojimo vieta, naudojama techninė įranga, atitikimas standartams;
- **aplinkos ribojimai**. Pavyzdžiui, veikimo suderinamumas (angl. interoperability), saugumas, privatumas;
- **įgyvendinimo ribojimai**. Pavyzdžiui, programavimo kalba, pagaminimo laikas.

IFPUG, NESMA ir Mark II metodai vertinti nefunkcinius naudotojų reikalavimus naudoja bendrąsias sistemos charakteristikas. IFPUG ir NESMA metoduose yra naudojamos 14 BSC, o Mark II metode 19 BSC. BSC naudojimas yra labai dažnai kritikuojamas. Pirmiausia yra neaišku, ar šių charakteristikų užtenka apibrėžti sistemos kūrimą. Galbūt reikia papildomų charakteristikų. Taip pat kiekvienos charakteristikos įtaka kuriamai sistemai yra vienoda (vertinama nuo 0 iki 5 balų). Nors vienos charakteristikos gali gerokai labiau įtakoti reikalingas kūrimo darbo sąnaudas negu kitos. Dar viena BSC problema yra ta, kad pagal pateikiamus apibrėžimus kai kurias BSC yra sunku atskirti viena nuo kitos [Vic03].

Nors tiesiogiai FiSMA metode nėra matuojami nefunkciniai naudotojų reikalavimai, tačiau FiSMA organizacija išleido atskirą dokumentą [FSMA01b], kuriame apibrėžė 21 standartinį produktyvumo faktorių, kurie padeda vertinti naujo kūrimo arba papildymo projektus. Juos galima pamatyti 3 priede. Šie faktoriai klasifikuoti į keturias skirtingas kategorijas. Kiekvienas faktorius vertinamas eiliškumo skalėje pagal situacijos gerumą, kurios penkios reikšmės yra: labai gera, gera, vidutinė, bloga, labai bloga.

COSMIC funkcinio dydžio matavimo metode neapibrėžtos jokios taisyklės, kaip skaičiuoti nefunkcinius naudotojų reikalavimus.

3.8. Sertifikavimo ir skaičiavimo vadovų prieinamumas

Panagrinėkime, kokios yra tiriamų metodų naudotojų sertifikavimo galimybės bei, kokius skaičiavimo vadovus ir papildomus susijusius dokumentus siūlo šiuos funkcinio dydžio matavimo metodus išleidusios organizacijos.

3.8.1. Sertifikavimas

IFPUG organizacija siūlo keturis skirtingus sertifikavimo tipus [IFPUG10]:

- **funkcinių taškų specialisto sertifikavimas**. Sertifikuoti funkcinių taškų specialistai yra pripažįstami kaip turintys reikiamus įgūdžius ir žinias, reikalingas atlikti suderinamus ir

tikslius funkcinių taškų skaičiavimus, bei turinčius gerą supratimą apie naujausias skaičiavimo praktikas;

- **programinės įrangos matavimo specialisto sertifikavimas**. Sertifikuoti programinės įrangos matavimo specialistai sugeba net tik matuoti programinę įrangą, bet ir pritaikyti įvairius matus, padedančius priimti svarbius projekto valdymo sprendimus;
- **mokymo medžiagos sertifikavimas**. Mokymo medžiaga, kuri aprašo tikslų ir darnų funkcinių taškų skaičiavimą, gali būti sertifikuojama. Mokymo medžiaga turi atitikti pačią naujausią skaičiavimo taisyklių vadovo versiją ir kitus IFPUG organizacijos pateikiamus mokymo medžiagos sertifikavimo kriterijus;
- **programinės įrangos sertifikavimas**. Programinė įranga, palaikanti funkcinių taškų skaičiavimą, taip pat gali būti sertifikuojama. Yra išskiriami trys programinės įrangos tipai:
    - **1 tipas**. Programinė įranga, kuri suteikia funkcinių taškų rinkimo ir skaičiavimo funkcionalumą, kur naudotojas funkcinius taškus skaičiuoja rankiniu būdu, o programinė įranga veikia kaip duomenų saugykla ir atlieka atitinkamus funkcinių taškų apskaičiavimus;
    - **2 tipas**. Programinė įranga, kuri suteikia funkcinių taškų rinkimo ir skaičiavimo funkcionalumą, kur naudotojas ir programinė įranga kartu nustato funkcinius taškus. Naudotojas atsakinėja į programinės įrangos pateikiamus klausimus ir programinė įranga priima sprendimus, užfiksuoja juos ir atlieka atitinkamus apskaičiavimus;
    - **3 tipas**. Programinė įranga automatiškai atlieka programos funkcinių taškų skaičiavimus pasinaudojant keliais informacijos šaltiniais, tokiais kaip matuojama programinė įranga, duomenų bazės valdymo sistema, programinės įrangos projektavimo ir kūrimo įrankių aprašai. Programinė įrangą užfiksuoja ir atlieka atitinkamus skaičiavimus. Naudotojas gali interaktyviai įvesti kai kuriuos duomenis, bet jo dalyvavimas skaičiavimų metu yra minimalus.

Sertifikuoto funkcinių taškų specialisto sertifikatas galioja tris metus. Norint jį pratęsti, reikia sertifikavimo egzaminą laikyti iš naujo. Tačiau IFPUG organizacija siūlo tam alternatyvą. Sertifikuoti funkcinių taškų specialistai gali pratęsti savo sertifikatus 2-3 metams atlikdami tam tikras IFPUG organizacijos apibrėžtas veiklas. Pavyzdžiui, paruošti atitinkamas su funkciniais taškais susijusias rekomendacijas, publikuoti straipsnius, dalyvauti atitinkamose IFPUG konferencijose, dalyvauti IFPUG skaičiavimo standartų komitete ir pan.

Dėl didėjančio funkcinių taškų specialistų sertifikavimo egzaminų kiekio IFPUG organizacija nusprendė automatizuoti egzaminų laikymą. Automatizuoto egzamino turinys niekuo nesiskiria nuo atliekamo ant popieriaus egzamino, tačiau šis egzaminas atliekamas elektroniniu būdu. Tai įgalina greitesnį egzamino rezultatų paskelbimą, sumažina sertifikavimo komiteto darbą ir suteikia didesnį egzamino vietų pasirinkimą.

COSMIC organizacija suteikia galimybę tapti sertifikuotu funkcinių taškų analitiku. Pateikto egzamino metu yra tikrinamos pagrindinės funkcinio dydžio matavimo, COSMIC metodo ir matavimo proceso žinios. Sėkmingai išlaikius egzaminą, yra suteikiamas sertifikatas ir asmuo yra įtraukiamas į sertifikatų turėtojų sąrašą, kuris yra viešai pateikiamas COSMIC internetiniame puslapyje. COSMIC sertifikato galiojimo trukmė yra neribota išlaikytai COSMIC metodo versijai [CSMIC10].

NESMA kartu su EXIN (angl. Examination Institute for Information Science) organizacija irgi suteikia sertifikuoto funkcinių taškų analitiko sertifikavimo paslaugas. NESMA pateikiamas egzaminas yra labai panašus į IFPUG egzaminą. Jame reikia tiek teorinių funkcinių taškų analizės žinių, tiek praktinės NESMA metodo taikymo patirties. NESMA suteikto sertifikato galiojimas yra neribotas [NSMA10b].

Savo ruoštu UKSMA organizacija nebeteikia Mark II metodo sertifikavimo galimybės, o FiSMA organizacija niekada ir neteikė FiSMA metodo sertifikavimo paslaugos.

Apibendrinant galima pasakyti, kad IFPUG organizacija suteikia daugiausia sertifikavimo galimybių. Tačiau IFPUG sertifikatų galiojimo laikas yra ribotas, tuo tarpu COSMIC ir NESMA nėra. Taip pat galima pastebėti ir sertifikavimo kainų skirtumus. IFPUG sertifikuoto funkcinių taškų specialisto sertifikavimas IFPUG nariams kainuoja 250, o nariams studentams 50 JAV dolerių. Nereikia pamiršti, kad narystė irgi yra mokama. Tuo tarpu ne nariams kaina siekia 500 JAV dolerių. COSMIC metodo sertifikavimo egzaminas kainuoja 100, o studentams 50 JAV dolerių. NESMA metodo sertifikavimo egzamino laikymas kainuoja 340 JAV dolerių.

3.8.2. Skaičiavimo taisyklių vadovai ir kiti susiję dokumentai

Kiekvieno metodo standartą galima nusipirkti iš ISO organizacijos. Yra svarbus tiriamų funkcinio dydžio matavimo metodų skaičiavimo taisyklių vadovų ir kitų susijusių dokumentų, palengvinančių šių metodų taikymą, prieinamumas.

IFPUG metodo skaičiavimo taisyklių 4.2.1 versijos vadovas susideda iš keturių dalių. Pirmoje dalyje yra pateikiamos funkcinių taškų skaičiavimo procesas ir taisyklės. Antroje dalyje pateiktos įvairios skaičiavimo praktikos, kaip funkcinių taškų taikymas programinės įrangos palaikymo

veiklose. Trečiojoje dalyje pateikiami įvairūs funkcinių taškų skaičiavimo pavyzdžiai. Ketvirtoje dalyje yra pateikiamos apibendrintos skaičiavimo lentelės ir specialių terminų žodynas.

IFPUG organizacija be IFPUG metodo skaičiavimo taisyklių vadovo pateikia ir keturis problemų nagrinėjimų dokumentus (angl. case studies). Pirmuose trijuose dokumentuose aprašomi funkcinių taškų skaičiavimai skirtingose tos pačios žmogiškųjų resursų sistemos realizacijose. Ketvirtame dokumente aprašytas funkcinių taškų skaičiavimas eismo valdymo sistemos realaus laiko komponentuose.

COSMIC organizacija be skaičiavimo taisyklių vadovo atskiruose dokumentuose pateikia COSMIC metodo apžvalgą, susijusias papildomas temas (ankstyvas metodo taikymas, rezultatų konvertuojamumas) ir specialių terminų žodyną. Taip pat pateikia metodo taikymo trijose verslo ir keturiose realaus laiko sistemose dokumentus. Papildomai pateikia ir pagrindinių COSMIC metodo naudotojų sąrašą.

NESMA be metodo skaičiavimo taisyklių vadovo dar pateikia funkcinių taškų analizės taikymo programinės įrangos papildymui, funkcinio dydžio matavimo nuorodos modelio, NESMA metodo ankstyvo taikymo, NESMA ir IFPUG metodų skirtumų ir kelis problemų nagrinėjimo dokumentus.

FiSMA organizacija papildomai prie FiSMA metodo skaičiavimo taisyklių vadovo pateikia ir naujo kūrimo projektų, palaikymo ir modifikavimo projektų, pakartotino naudojimo matavimo analizės dokumentus.

UKSMA organizacija savo ruoštu pateikia tik Mark II funkcinio dydžio matavimo metodo skaičiavimo taisyklių vadovą.

IFPUG organizacijos visi pateikiami dokumentai yra mokami. NESMA metodo skaičiavimo taisyklių vadovas yra mokamas, o kai kurie kiti dokumentai pateikiami nemokamai. Tuo tarpu visi COSMIC, FiSMA ir Mark II metodo dokumentai pateikiami nemokamai.

Apibendrinant galima pasakyti, kad IFPUG ir COSMIC organizacijos pateikia daugiausia programinės įrangos funkcinio dydžio matavimą palengvinančių dokumentų.

3.9. Metodus palaikantys programiniai įrankiai

Dar vienas labai svarbus aspektas nagrinėjamiems metodams yra juos palaikantys programiniai įrankiai. Programinės įrangos matavimo proceso efektyvumas priklauso nuo jo automatizacijos, pasiektos programiniais įrankiais, lygio [ED07]. Programiniai įrankiai įgalina atlikti efektyvesnius, kokybiškesnius ir labiau standartizuotus matavimus [BD08].

Pirmasis žingsnis vertinant programinę įrangą yra nustatyti gaminamo produkto dydį. Ankstesni kaštų vertinimo įrankiai, tokie kaip COCOMO neturėjo dydžio skaičiavimo funkcionalumo, todėl informacija apie programinės įrangos dydį turėdavo būti įvedama rankomis. Tačiau funkcinių taškų mato atsiradimas padarė funkcinio dydžio skaičiavimo logiką standartine savybe įvairiuose vertinimo įrankiuose, pradedant SPQR/20, kuris atsirado 1985 metais. Nuo 1998 metų dydis yra standartinė savybė komerciniuose projektų vertinimo įrankiuose ir į juos įtraukiami įvairūs dydžio matavimo metodai, paremti programos kodo eilučių skaičiavimu arba funkcinių taškų skaičiavimu [Jon05].

Komerciniai programinės įrangos vertinimo įrankiai yra tikslesni, efektyvesni ir atlieka matavimus greičiau, negu tai padaro žmonės. Tačiau nei vienas vertinimo metodas nėra visiškai atsparus klaidoms. Geriausia, vertinant projektus ir programinę įrangą, yra naudoti programinės įrangos vertinimo įrankius, susietus su projekto valdymo įrankiais, atidžiai prižiūrint patyrusių projektų vadovų ir vertinimo specialistų [Jon05].

Panagrinėkime populiariausius įrankius, kuriuose naudojami nagrinėjami funkcinio dydžio matavimo metodai ar jų rezultatai, plačiau.

Nagrinėjamus metodus ir funkcinius taškus palaiko įvairūs programinės įrangos funkcinio dydžio matavimo (SCOPE [TM10], fsmIT [PSCD10], SIESTA [SNB10], MeterIT-Cosmic [Tel10], NH‘s Function Point Analyzer [NHS10], Function Point Workbench [CSM10]), projekto apimties valdymo (northernSCOPE [FSMA08], Experience Pro [SP10]), projekto gyvavimo ciklo valdymo (PQMPlus [QPMG10], Function Point Modeler [Ozt10]), projektų vertinimo ir analizės, kuriais galima vertinti dydį, darbo sąnaudas, kaštus, grafiką ir defektus (EstimatorPal [CC10a], FPAPal [CC10b], SPR KnowledgePLAN [SPR10], COSMOS [CDT10]) programiniai įrankiai.

15 lentelėje parodyta, kuriuos iš nagrinėjamų funkcinio dydžio matavimo metodų ar jais gautus rezultatus palaiko paminėti programiniai įrankiai.

15 lentelė. Programinių įrankių naudojami funkcinio dydžio matavimo metodai

Programinis įrankis	**IFPUG metodas**	**COSMIC metodas**	**NESMA metodas**	**FiSMA metodas**	**Mark II metodas**
SCOPE	**X**				
fsmIT		**X**			
SIESTA	**X**	**X**	**X**		
MeterIT-Cosmic		**X**			
NH‘s Function Point Analyzer	**X**				

Programinis įrankis	IFPUG metodas	COSMIC metodas	NESMA metodas	FiSMA metodas	Mark II metodas
Function Point Workbench	**X**		**X**		
northernSCOPE	**X**	**X**	**X**	**X**	**X**
Experience Pro	**X**	**X**		**X**	**X**
PQMPlus	**X**				
Function Point Modeler	**X**				
EstimatorPal	**X**				**X**
FPAPal	**X**				
SPR KnowledgePLAN	**X**				
COSMOS	**X**				

Taip pat kai kuriais įrankiais galima atlikti ir ankstyvus nagrinėjamų metodų taikymus. SIESTA ir MeterIT-Cosmic įrankiai palaiko ankstyvo taikymo COSMIC metodą, Function Point Workbench įrankis palaiko IFPUG ir NESMA metodų ankstyvą taikymą.

Vienas iš IFPUG metodą palaikančių įrankių pasirinkimo kriterijų galėtų būti jų sertifikuotumas. Function Point Workbench įrankis yra sertifikuotas IFPUG organizacijos pirmu programinės įrangos tipu, o PQMPlus sertifikuotas pirmu ir antru programinės įrangos tipu. Trečiu tipu kol kas dar joks įrankis nėra sertifikuotas.

Su kai kuriais įrankiais (pavyzdžiui, su knowledgePLAN) yra galima atlikti įvykių modeliavimą (angl. simulation). Taip galima nustatyti kiekvieno atskiro parametro (produktyvumo matų, kokybės matų, trukmės, darbo sąnaudų) įtaką atliekamiems matavimams [BD08].

Taip pat dalis funkcinius taškus matuojančių ir juos naudojančių įrankių turi galimybę eksportuoti duomenis į kitus įrankius, pavyzdžiui, į projektų valdymo įrankius. Taip pat įrankiai gali būti kalibruojami, kad labiau atitiktų organizacijos reikalavimus.

Ne visi programiniai matavimo ir vertinimo įrankiai yra tinkami visoms situacijoms, todėl kartais yra pravartu naudoti ir kelis įrankius. Žinoma reikia atsižvelgti į įrankių kainas ir jų atsiperkamumą.

Atlikti gerus projektų ir programų vertinimus ir matavimus vien įrankių neužtenka, reikia tam ir žinių, nes tik nuo gerų įvedamų duomenų priklauso ir geri gaunami vertinimo rezultatai.

Kaip matome daugiausia programinių matavimo ir vertinimo įrankių palaiko IFPUG funkcinio dydžio matavimo metodą ir jo taikymo metu gautus funkcinius taškus. Taip pat nemažai įrankių palaiko ir COSMIC metodą, o mažiau Mark II, NESMA ir FiSMA metodus.

3.10. Metodų taikymo sritis

Dar vienas iš nagrinėjamų penkių standartizuotų funkcinių taškų matavimo metodų vertinimo kriterijų yra jų taikymo sritis. Panagrinėkime, kokių tipų programinei įrangai matuoti yra tinkamesni tiriami metodai.

Kaip teigia nagrinėjamų metodų kūrėjai, COSMIC metodas yra tinkamas matuoti valdymo informacines sistemas, realaus laiko sistemas ir šių sistemų hibridus, o IFPUG, NESMA, Mark II ir FiSMA metodai yra tinkami matuoti bet kokio tipo programinę įrangą. Panagrinėkime kaip šiais metodais galime matuoti realaus laiko sistemas.

Realaus laiko sistemų procesai dažniausiai pasižymi tokia savybe, kad labai skiriasi jų paprocesių skaičius. Realaus laiko sistemų funkcinio dydžio matavimo metodas turi atsižvelgti į tai, kad kai kurie procesai turi tik kelis paprocesius, o kai kurie labai daug paprocesių [PMA+97]. COSMIC metodas matuoja programinės įrangos funkcinį dydį detalesniame lygyje, nei IFPUG, NESMA, Mark II ir FiSMA metodai, nes jis vienintelis apibrėžia paprocesius: įėjimo, išėjimo, skaitymo ir rašymo duomenų judėjimai.

Realaus laiko sistemose valdymo duomenys yra naudojami valdyti sistemą arba mechaninį prietaisą. Yra dviejų rūšių valdymo duomenų struktūros: kelių egzempliorių (angl. multiple occurrence) duomenų grupės ir vieno egzemplioriaus (angl. single occurrence) duomenų grupės. Kelių egzempliorių duomenų grupės gali turėti daugiau nei vieną to paties įrašo tipo egzempliorių. Vieno egzemplioriaus duomenų grupė turi tiktai vieną įrašo egzempliorių [PMA+97]. Tipiškas valdymo informacinės sistemos loginis failas turi kelių egzempliorių duomenų struktūrą, kur kiekvienas įrašas turi vieną arba daugiau laukų. Remiantis IFPUG ir NESMA metodų taisyklėmis, kelių egzempliorių duomenų grupės yra matuojamos kaip VLF arba IIF [ADM+98]. Realaus laiko sistemos savo ruoštu dažniausiai turi didelį vieno egzemplioriaus duomenų grupių, kurias sunku sugrupuoti į VLF ir IIF, kiekį [PMA+97].

Pirmos kartos funkcinio dydžio matavimo metodai, tokie kaip IFPUG, NESMA, Mark II ir FiSMA yra tinkamesni matuoti valdymo informacines sistemas. Antros kartos funkcinio dydžio matavimo metodas COSMIC buvo sukurtas tiek valdymo informacinių sistemų, tiek realaus laiko ar išskirstytų sistemų matavimui. Matuojant tokią programinę įrangą, jos reikalavimai gali apimti kelis programinės įrangos sluoksnius, išreikštus iš skirtingų perspektyvų ir skirtinguose detalumo lygiuose [Sym07]. Į visą tai galima atsižvelgti naudojant COSMIC metodą.

Taip pat galima paminėti, kad visi nagrinėjami funkcinio dydžio matavimo metodai nėra tinkami matuoti tokias sistemos, kuriose yra naudojama daug sudėtingų matematinių algoritmų.

3.11. Metodų vertinimo pagal kriterijus apibendrinimas

IFPUG, COSMIC, NESMA, FiSMA ir Mark II funkcinių taškų analizės metodų vertinimo rezultatai pagal apibrėžtus kriterijus pateikti 16 lentelėje.

16 lentelė. Metodų vertinimo pagal apibrėžtus kriterijus rezultatai

Vertinimo kriterijus	IFPUG metodas	COSMIC metodas	NESMA metodas	FiSMA metodas	Mark II metodas
Standartizuotumas	Standartizuotas, koreguotų funkcinių taškų apskaičiavimas yra nebūtinas žingsnis	Standartizuotas, koreguotų funkcinių taškų apskaičiavimas yra nebūtinas žingsnis	Standartizuotas, koreguotų funkcinių taškų apskaičiavimas yra nebūtinas žingsnis	Standartizuotas	Standartizuotas
Industrinių duomenų prieinamumas	3108 projektų ISBSG saugykloje	117 projektų ISBSG saugykloje	152 projektų ISBSG saugykloje	340 projektų ISBSG saugykloje	35 projektų ISBSG saugykloje
Metodų rezultatų tarpusavio konvertuojamumas	Galima konvertuoti į NESMA, COSMIC, Mark II, bei iš NESMA ir Mark II funkcinių taškų	Galima konvertuoti iš IFPUG ir NESMA funkcinių taškų	Galima konvertuoti į IFPUG, COSMIC, bei iš IFPUG funkcinių taškų	Nėra konvertavimo formulių	Galima konvertuoti į ir iš IFPUG funkcinių taškų
Kodo eilučių skaičiaus konvertavimas į funkcinius taškus	Yra konvertavimo koeficientų lentelės	Nėra konvertavimo koeficientų lentelių	Nėra konvertavimo koeficientų lentelių	Nėra konvertavimo koeficientų lentelių	Nėra konvertavimo koeficientų lentelių
Tinkamiausia metodų taikymo fazė	Po reikalavimų analizės	Po reikalavimų analizės	Po reikalavimų analizės	Po reikalavimų analizės	Po reikalavimų analizės
Ankstyvas metodų taikymas	Yra ankstyvo ir greito taikymo metodų	Yra ankstyvo ir greito taikymo metodų	Yra ankstyvo ir greito taikymo metodų	Nėra ankstyvo ir greito taikymo metodų	Nėra ankstyvo ir greito taikymo metodų

Vertinimo kriterijus	IFPUG metodas	COSMIC metodas	NESMA metodas	FiSMA metodas	Mark II metodas
Metodų taikymas palaikymo fazės metu	Rekomendacijos kaip skaičiuoti funkcinius taškus palaikymo fazės metu	Nėra pateikta jokių taisyklių ar rekomendacijų	Atskiros papildančiųjų projektų palaikymo metu matavimo taisyklės	Atskirai apibrėžti 22 standartiniai produktyvumo faktoriai, kurie įtakoja darbo sąnaudas, reikalingas palaikyti programinę įrangą	Kartu su ISBSG organizacija išleistas programinės įrangos palaikymo ir priežiūros matavimo standartas
Subjektyvumas metoduose	Funkcijų sudėtingumų, BSC vertinimas		Funkcijų sudėtingumų, BSC vertinimas		BSC vertinimas
Nefunkciniai naudotojų reikalavimai	Naudoja 14 BSC	Netaiko jokių taisyklių	Naudoja 14 BSC	Atskirai apibrėžti 21 standartinis produktyvumo faktorius	Naudoja 19 BSC
Sertifikavimas	Funkcinių taškų specialisto, programinės įrangos matavimo specialisto, mokymo medžiagos, programinės įrangos sertifikavimas	Funkcinių taškų analitiko sertifikavimas	Funkcinių taškų analitiko sertifikavimas	Neteikia sertifikavimo galimybės	Neteikia sertifikavimo galimybės

Vertinimo kriterijus	IFPUG metodas	COSMIC metodas	NESMA metodas	FiSMA metodas	Mark II metodas
Skaičiavimo taisyklių vadovai ir kiti susiję dokumentai	Daug papildomų dokumentų, visi mokami	Daug papildomų dokumentų, visi nemokami	Mažai papildomų dokumentų, dalis jų nemokami	Mažai papildomų dokumentų, visi nemokami	Nėra papildomų dokumentų
Metodus palaikantys programiniai įrankiai	Labai daug įrankių	Nemažai įrankių	Mažai įrankių	Mažai įrankių	Mažai įrankių
Metodų taikymo sritis	Valdymo informacinės sistemos	Valdymo informacinės sistemos, realaus laiko sistemos, išskirstytos sistemos	Valdymo informacinės sistemos	Valdymo informacinės sistemos	Valdymo informacinės sistemos

Mark II funkcinio dydžio matavimo metodas neturi daug industrinių duomenų, neturi išvestų ankstyvo ir greito taikymo metodų, neteikia sertifikavimo galimybių, nepateikia jokių papildomų matavimus palengvinančių dokumentų bei neturi daug metodą palaikančių programinių įrankių. Šis metodas plačiau už Jungtinės Karalystės ribų ir neišplito. Paskutinis skaičiavimo taisyklių vadovas išleistas 1998 metais ir metodas daugiau nebetobulinamas, nebeatnaujinami jo pramonės standartiniai vidurkiai. Mark II metodo kūrėjas Charles Symons šiuo metu aktyviai dalyvauja COSMIC funkcinio dydžio matavimo metodo tobulinime.

FiSMA funkcinio dydžio matavimo metodas buvo standartizuotas tik 2008 metais, todėl kol kas jis nėra plačiai paplitęs. Nėra išvestų jokių šio metodo konvertavimo formulių, ankstyvo ir greito taikymo metodų, nėra pateiktos jokios šio metodo sertifikavimo galimybės, nėra daug papildomų metodo taikymą palengvinančių dokumentų ir metodą palaikančiu programinių įrankių.

IFPUG ir NESMA funkcinio dydžio matavimo metodai kuo toliau, tuo labiau tampa panašūs. Po kurio laiko tarp jų gali nebelikti ir jokių skirtumų. Todėl patartina rinktis IFPUG metodą, nes šio metodo yra daug industrinių duomenų, konvertavimo formulių, konvertavimo koeficientų lentelių, suteikiamos didesnės sertifikavimo galimybės, pateikiama daug matavimus palengvinančių papildomų dokumentų bei daug šį metodą palaikančių įrankių.

COSMIC funkcinio dydžio matavimo metodas turi pateiktų įvairių konvertavimo formulių, ankstyvo ir greito taikymo metodų, yra suteikiama funkcinių taškų analitiko sertifikavimo galimybė, daug matavimus palengvinančių papildomų dokumentų, nemažai metodą palaikančių programinių įrankių. Nors nėra labai daug šio metodo industrinių duomenų, bet COSMIC organizacijoje yra įkurtas lyginamosios analizės komitetas, kurio tikslas yra didinti industrinių duomenų kiekį ir skatinti metodo naudojimą. Todėl artimoje ateityje tiek COSMIC metodo industrinių duomenų, tiek jį palaikančių programinių įrankių turėtų gerokai padaugėti. Taip pat COSMIC metodas yra labiau pritaikytas šiuolaikinės programinės įrangos, tokios kaip realaus laiko ar išskirstytų sistemų, matavimui.

Rezultatai ir išvados

Dabartiniame informacinių technologijų pasaulyje nemaža dalis programinės įrangos kūrimo projektų baigiasi nesėkmėmis. Norėdamos pagerinti šią situaciją, programinę įrangą kuriančios organizacijos turėtų įgyvendinti įvairių projekto ir programinės įrangos charakteristikų matavimo programą. Nuo to labai priklauso kiekvieno projekto sėkmė.

Yra daug įvairių programinės įrangos ir programinės įrangos kūrimo projektų charakteristikų, kurias galime išmatuoti, kaip, pavyzdžiui, programinės įrangos dydis, sudėtingumas, patikimumas ir pelningumas.

Vienas fundamentaliausių programinės įrangos matų yra jos dydis. Dydis yra kritinis faktorius kaštų, grafiko ir darbo sąnaudų nustatyme. Kadangi programos kodo eilučių skaičius yra tik fizinis programinės įrangos dydžio matas, kuris yra priklausomas nuo kūrimo technologijos ir kuris yra apskaičiuojamas tik vėlyvu programinės įrangos gyvavimo ciklo metu, todėl programinės įrangos matavimuose reikia naudoti ir funkcinius taškus, kurie yra funkcinis programinės įrangos dydžio matas.

Šiuo metu yra labai daug įvairių funkcinių taškų analizės principą įgyvendinančių metodų, bet standartizuoti tiktai 5 iš jų. Jie buvo pasirinkti vertinimui šio magistro darbo tyrimo metu. Šie metodai yra standartizuoti IFPUG, Mark II, COSMIC, NESMA ir FiSMA funkcinio dydžio matavimo metodai. Kad kiekviena organizacija galėtų efektyviai vykdyti programinės įrangos matavimo programą, jai yra labai svarbu naudoti jos poreikius labiausiai atitinkantį standartizuotą funkcinio dydžio matavimo metodą.

Šiame darbe buvo apibendrinti ir suformuluoti įvairiuose šaltiniuose minimi galimi vertinimo metodų pasirinkimo kriterijai. Pagal šiuos kriterijus buvo sudaryti magistro darbo autoriaus nuomone 10 aktualiausių funkcinio dydžio matavimo metodų vertinimo kriterijų. Taip pat apibrėžtas kiekvieno iš šių kriterijų svarbumas ir tinkamumas vertinamiems 5 standartizuotiems funkcinio dydžio matavimo metodams.

Toliau pagal šiuos apibrėžtus kriterijus buvo įvertinti visi 5 standartizuoti funkcinio dydžio matavimo metodai: IFPUG, Mark II, COSMIC, NESMA ir FiSMA.

Pagal atlikto šių 5 standartizuotų funkcinio dydžio matavimo metodų vertinimo visais apibrėžtais kriterijais rezultatus parodomi šiuo metu labiausiai tinkami naudoti funkcinio dydžio matavimo metodai.

Atlikto tyrimo rezultatai parodo, kad iš dabartinių standartizuotų funkcinių taškų analizės metodų geriausia yra pasirinkti IFPUG arba COSMIC metodus. Jie turi daug išvestų konvertavimo formulių, turi ankstyvo ir greito taikymo metodų, yra suteikiamos galimybės tapti sertifikuotais šių metodų taikytojais, pateikia daug matavimus palengvinančių papildomų dokumentų, nemažai metodą palaikančių programinių įrankių. IFPUG funkcinio dydžio matavimo metodas tinkamesnis toms organizacijos kurios kuria valdymo informacines sistemas, tuo tarpu COSMIC metodą patartina rinktis organizacijoms, kuriančioms realaus laiko ar išskirstytas sistemas.

Literatūros sąrašas

[ADA05] A. Abran, J. M. Desharnais, F. Aziz. Measurement Convertibility - From Function Points to COSMIC-FFP. 2005, 8 pages.

[ADM+98] A. Abran, J. M. Desharnais, M. Maya, D. S. Pierre, P. Bourque. Design of a Functional Size Measurement for Real-Time Software. Proceedings of the 4th IEEE International Symposium and Forum on Software Engineering Standards, 1998, 29 pages.

[AFS00] Air Force Software Technology Support Center. Guidelines for Successful Acquisition and Management of Software-Intensive Systems. 2000, 945 pages.

[AJ99] A. Abran, J.P. Jacquet. A Structured Analysis of the New ISO Standard on Functional Size Measurement - Definition of Concepts. Proceedings of the 4th IEEE International Symposium and Forum on Software Engineering Standards, 1999, 12 pages.

[BD08] M. Bundschuh, C. Dekkers. The IT Measurement Compendium: Estimating and Benchmarking Success with Functional Size Measurement. Springer-Verlag Berlin Heidelberg, 2008, 643 pages.

[CBD+10] J. J. Cuadrado-Gallego, L. Buglione, M. J. Dominguez-Alda, M. F. D. Seville, J. A. G. D. Mesa, O. Deminors. An experimental study on the conversion between IFPUG and COSMIC functional size measurement units. Information and Software Technology 52, 2010, 11 pages.

[CC10a] Chemuturi Consultants. EstimatorPal.
[žiūrėta 2010-04-13]. Prieiga per internetą:
<http://www.effortestimator.com/epal.html>

[CC10b] Chemuturi Consultants. FPAPal.
[žiūrėta 2010-04-13]. Prieiga per internetą:
<http://www.effortestimator.com/fpal.html>

[CCS+04] J. J. Cuadrado, J. Crespo, M. A. Sicilia, A. Amescua, L. Garcia. Comparative analysis of different methods of function points measurement. Software Measurement European Forum 2004, 2004, 8 pages.

[CDT10] COSMOS Development Team. COSMOS.
[žiūrėta 2010-04-13]. Prieiga per internetą:
<http://csciwww.etsu.edu/cosmos/>

[CSM10] CHARISMATEK Software Metrics. Function Point WORKBENCH. 2010, 18 pages.

[CSMIC07] Common Software Measurement International Consortium. The COSMIC Functional Size Measurement Method Version 3.0 Advanced and related Topics. 2007, 31 pages.

[CSMIC08] Common Software Measurement International Consortium. Advantages of the COSMIC Method.
[žiūrėta 2010-05-05]. Prieiga per internetą:
<http://www.yarenty.com/cosmicffp/cosmic-about/36-cosmic-about/20-advantages-of-the-cosmic-method.html>

[CSMIC09] Common Software Measurement International Consortium. The COSMIC Functional Size Measurement Method Measurement Manual. 2009, 80 pages.

[CSMIC10] Common Software Measurement International Consortium. The COSMIC Certification Scheme.
[žiūrėta 2010-04-09]. Prieiga per internetą:
<http://www.cosmicon.com/certificationV3.asp>

[DA07] J. M. Desharnais, A. Albran, J. Cuadrado. Convertibility of Function Points to COSMIC-FFP: Identification and Analysis of Functional Outliers. Proceedings of the 17th international workshop on software measurement, 2007, 16 pages.

[DCG06] David Consulting Group. Software Sizing with Function Points. 2006, 5 pages.

[DG00] C. Dekkers, I. Gunter. Using "Backfiring" to Accurately Size Software: More Wishful Thinking Than Science? IT Metrics Strategies, VI tomas, 11 Nr., 2000, 8 pages.

[DG09] O. Deminors, C. Gencel. Conceptual Association of Functional Size Measurement Methods. Software, IEEE, 2009, 8 pages.

[ED07] C. Ebert, R. Dumke. Software Measurement: Establish, Extract, Evaluate, Execute. Springer-Verlag Berlin Heidelberg, 2007, 561 pages.

[EDB+05] C. Ebert, R. Dumke, M. Bundschuh, A. Schmietindorf. Best Practices in Software Measurement: How to use metrics to improve project and process performance. Springer-Verlag Berlin Heidelberg, 2005, 295 pages.

[Fet99] T. Fetcke. The Warehouse Software Portfolio - A Case Study in Functional Size Measurement. 1999, 111 pages.

[FSMA01a] Finnish Software Measurement Association. Experience MT22 Situation Analysis Model. 2001, 9 pages.

[FSMA01b] Finnish Software Measurement Association. Experience ND21 Situation Analysis Model. 2001, 7 pages.

[FSMA07] Finnish Software Measurement Association. FiSMA 1.1 Functional Size Measurement Method. 2007, 20 pages.

[FSMA08] Finnish Software Measurement Association. northernSCOPE: customer-driven scope control for ICT projects. 2008, 4 pages.

[GD07] C. Gencel, O. Deminors. Conceptual Differences Among Functional Size Measurement Methods. Software, IEEE, 2007, 9 pages.

[GD08] C. Gencel, O. Deminors. Functional Size Measurement Revisited. ACM Transactions on Software Engineering and Methodology, 2008, 37 pages.

[GDY05a] C. Gencel, O. Deminors, E. Yuceer. A Case Study on Using Functional Size Measurement Methods for Real Time Systems. 15th International Workshop on Software Measurement, 2005, 11 pages.

[GDY05b] C. Gencel, O. Demirors, E. Yuceer. Utilizing Functional Size Measurement Methods for Real Time Software Systems. 11th IEEE International Software Metrics Symposium, 2005, 4 pages.

[GH01] D. Garmus, D. Herron. Function Point Analysis: Measurement Practices for Successful Software Projects. Addison-Wesley, Inc., 2001, 363 pages.

[HAF99] V. T. Ho, A. Albrain, T. Fetcke. A Comparative Study Case of COSMIC-FFP, Full Function Point and IFPUG methods. 1999, 10 pages.

[Hee07] H.S. van Heeringen. Changing from FPA to COSMIC - A transition framework. 2007, 13 pages.

[HS08] H. S. van Heeringen, L. Santillo. Proposals for Increasing Benchmarking Data Quantity and Quality of Projects Measured in COSMIC. International Conferences IWSM 08, 2008, 9 pages.

[IFPUG00] International Function Point Users Group. Function Point Counting Practices Manual. 2000, 370 pages.

[IFPUG02] International Function Point Users Group. IT Measurement: Practical Advice from the Experts. Pearson Education, Inc., 2002, 759 pages.

[IFPUG05] International Function Point Users Group. Function Point Counting Practices Manual: Part 2 – Counting Practices. Release 4.2.1, 2005, 117 pages.

[IFPUG10] International Function Point Users Group. IFPUG Certifications. [žiūrėta 2010-04-09]. Prieiga per internetą: <http://www.ifpug.org/certification/>

[ISO07] The International Organization for Standartization. Information technology – Software measurement – Functional size measurement – Part 1: Definition of concepts. 2007, 6 pages.

[JA97] J.P. Jacquet, A. Abran. From Software Metrics to Software Measurement Methods - A Process Model. 3rd International Software Engineering Standards Symposium, 1997, 12 pages.

[Jal05] P. Jalote. An Integrated Approach to Software Engineering. Springer Science+Business Media, Inc., 2005, 566 pages.

[Jon05] C. Jones. How Software Estimation Tools Work. Software Productivity Research, Inc., 2005, 28 pages.

[LB06] L. M. Laird, M. C. Brennan. Software Measurement and Estimation: A Practical Approach. A John Wiley & Sons, Inc., 2006, 258 pages.

[Lon01] D. Longstreet. Function Point Analysis Training Course. Longstreet Consulting, Inc., 2001, 111 pages.

[Lon05] D. Longstreet. Fundamentals of Function Point Analysis. Longstreet Consulting Inc., 2005, 9 pages.

[MAO+98] M. Maya, A. Abran, S. Oligny, D. St-Pierre, J. Desharnais. Measuring the functional size of real-time software. ESCOM-ENCRESS 98, 1998, 10 pages.

[MD98] P. Morris, J. M. Desharnais. Measuring ALL the Software not just what the Business Uses. Proceedings of the IFPUG Conference, 1998, 17 pages.

[Mor06] P. Morris. Counting “Non-Functional“ Requirements when they are implemented as Software. 2006, 4 pages.

[MS99] R. Meli, L. Santillo. Function Point Estimation Methods: A Comperative Overview. FESMA 98 - The European Software Measurement Conference, 1999, 14 pages.

[NHS10] Norbert Heidbuchel‘s Software. NH‘s Function Point Analyzer. [žiūrėta 2010-04-13]. Prieiga per internetą: <http://www.nh-software.com/#go3>

[NSMA08] Netherlands Software Metrics Association. FPA according to NESMA and IFPUG: the pesent situation. 2008, 3 pages.

[NSMA09] Netherlands Software Metrics Association. Function Point Analysis for Software Enhancement: Guidelines. Version 2.2.1, 2009, 25 pages.

[NSMA10a] Netherlands Software Metrics Association. Early Function Point Counting. [žiūrėta 2010-04-01]. Prieiga per internetą: <http://www.nesma.nl/section/fpa/earlyfpa.htm>

[NSMA10b] Netherlands Software Metrics Association. Certification FPA (CFPA). [žiūrėta 2010-04-09]. Prieiga per internetą: <http://www.nesma.nl/section/home/certification.asp>

[Ozt10] M. Ozturk. Function Point Modeler: Take a Look Inside Your Software. 2010, 31 pages.

[PMA+97] D. St. Pierre, M. Maya, A. Abran, J. M. Desharnais. Adapting Function Points to Real-Time Software. 1997 IFPUG Fall Conference, 1997, 14 pages.

[PSCD10] Pentad Software Consultancy and Development. fsmIT. [žiūrėta 2010-04-13]. Prieiga per internetą: <http://www.cosmicon.com/vendorsV3.asp>

[QPMG10] Q/P Management Group. PQMPlus – Is your software project destined to succeed or doomed to fail. [žiūrėta 2010-04-13]. Prieiga per internetą: <http://www.qpmg.com/pqmplus.htm>

[QSM09] Quantitative Software Management. Function Point Language Table. [žiūrėta 2010-04-06]. Prieiga per internetą: <http://www.qsm.com/?q=resources/function-point-languages-table/index.html>

[Sym07] C. Symons. Advancing Functional Size Measurement – which size should we measure. SMEF 07, 2007, 12 pages.

[Sym99] C. Symons. Conversion between IFPUG 4.0 and MkII Function Points. Version 3.0, 1999, 10 pages.

[SCM05] L. Santillo, M. Conte, R. Meli. Early & Quick Function Point: Sizing More with Less. 11th IEEE International Software Metrics Symposium, 2005, 6 pages.

[SNB10] Sogeti Nederland BV. SIESTA. [žiūrėta 2010-04-13]. Prieiga per internetą: <http://www.cffp.nl/Home/Siesta/index.jsp>

[SP10] 4SUM Partners. Experience® Pro version 3.1. [žiūrėta 2010-04-03]. Prieiga per internetą: <http://www.4sumpartners.com/prod01.htm>

[SPR10] Software Productivity Research. SPR KnowledgePLAN. [žiūrėta 2010-04-13]. Prieiga per internetą: <http://www.spr.com/spr-knowledgeplanr.html>

[TA08] S. Trudel, A. Abran. Improving quality of functional requirements by measuring their functional size. IWSM 2008, 2008, 18 pages.

[Tel10] Telmaco. MeterIT-Cosmic. [žiūrėta 2010-04-13]. Prieiga per internetą: <http://www.telmaco.co.uk/productlist.html>

[TK05] E. Tkacz, A. Kapczynski. Internet – Technical Development and Applications. Springer-Verlag Berlin Heideberg, 2009, 279 pages.

[TM07] Total Metrics. How to Decide which Method to Use. Total Metrics June 2007 Version 1.0, 2007, 7 pages.

[TM10] Total Metrics. SCOPE 3.0 Function Point Software – When organizations are serious about Function Point Counting. [žiūrėta 2010-04-13]. Prieiga per internetą: <http://www.totalmetrics.com/function-point-software/scope-project-sizing-software/overview>

[UKSMA98] United Kingdom Software Metrics Association. MK II Function Point Analysis Counting Practices Manual. 1998, 92 pages.

[Vic03] P. Vickers. An Introduction to Function Point Analysis. Northumbria University, 2003, 28 pages.

[VL03] F. Vogelezang, A. Lesterhuis. Applicability of COSMIC Full Function Points in an administrative environment - Experiences of an early adopter. 2003, 8 pages.

[Zus98] H. Zuse. A Framework of Software Measurement. Walter de Gruyter & Co., 1998, 755 pages.

[XGH06] G. Xunmei, G. Guoxin, Z. Hong. The Comparison between FPA and COSMIC-FFP. Software Measurement European Forum 2006, 2006, 9 pages.

Sąvokų apibrėžimai ir santrumpos

BFK (baziniai funkciniai komponentai) – funkcinių naudotojų reikalavimų elementarus vienetas, apibrėžtas funkcinio dydžio matavimo metoduose ir naudojamas matavimo tikslais.

BSC (bendrosios sistemos charakteristikos) **–** IFPUG, NESMA ir Mark II metoduose apibrėžiamos kaip nefunkcinės sistemos charakteristikos.

CFT (cosmic funkcinis taškas) – COSMIC metode naudojamas funkcinio dydžio vienetas.

DET (duomenų elementų tipai) – IFPUG ir NESMA metoduose apibrėžiami kaip unikalūs naudotojam suprantami, nesikartojantys laukai ar atributai. Mark II metode DET apibrėžiamas kaip unikalus, naudotojo atpažįstamas informacijos apie esybės tipus elementas.

FNR (funkciniai naudotojų reikalavimai) - poaibis naudotojų reikalavimų, kurie apibūdina, ką programa turės daryti užduočių ir paslaugų terminais.

FT (funkcinis taškas) - programinės įrangos funkcinio dydžio matavimo vienetas.

FTA (funkcinių taškų analizė) - programinės įrangos funkcinio dydžio matavimo procesas.

I (išėjimas) - COSMIC metode apibrėžiamas kaip duomenų judėjimas, kuris perkelia duomenų grupę iš funkcinio proceso į duomenų grupės reikalaujantį funkcinį naudotoją.

II (išorinis išvedimas) - IFPUG ir NESMA metoduose apibrėžiamas kaip elementarus procesas, kuris siunčia duomenis ar valdymo informaciją į programos išorę, o vykdymo logika turi matematines formules ar skaičiavimus.

IIF (išorinis interfeiso failas) – IFPUG ir NESMA metoduose apibrėžiamas kaip logiškai susijusių duomenų ar valdymo informacijos grupė, į kurią kreipiasi programa, bet yra palaikoma kitos programos ribose.

IĮ (išorinis įvedimas) - IFPUG ir NESMA metoduose apibrėžiamas kaip elementarus procesas, kuris apdoroja duomenis ar valdymo informaciją, kuri atvyksta iš programos išorės.

IU (išorinė užklausa) - IFPUG ir NESMA metoduose apibrėžiama kaip elementarus procesas, kuris siunčia duomenis ar valdymo informaciją į programos išorę, o vykdymo logika neturi jokių matematinių formulių ar skaičiavimų.

Į (įėjimas) – COSMIC metode apibrėžiamas kaip duomenų judėjimas, kuris perkelia duomenų grupę iš funkcinio naudotojo į duomenų grupės reikalaujantį funkcinį procesą.

ĮET (įrašų elementų tipai) - IFPUG ir NESMA metoduose apibrėžiami kaip naudotojam suprantami duomenų elementų, esančių vidiniuose loginiuose failuose ir išorinių interfeisų failuose, pogrupiai.

KFT (kreipimosi failų tipai) - IFPUG ir NESMA metoduose apibrėžiami kaip bendras skaitomų ar palaikomų vidinių loginių failų ir skaitomų išorinių interfeisų failų, vykdant išorinių įvedimų ir išorinių išvedimų transakcijas, skaičius.

KFTa (koreguoti funkciniai taškai) - IFPUG ir NESMA metodais apskaičiuotas funkcinis dydis pritaikius reikšmių koregavimo faktorių arba Mark II metodu apskaičiuotas funkcinis dydis pritaikius techninio sudėtingumo koregavimo reikšmę.

NFT (nekoreguoti funkciniai taškai) – IFPUG ir NESMA metodais apskaičiuotas funkcinis dydis nepritaikius reikšmių koregavimo faktoriaus.

R (rašymas) - COSMIC metode apibrėžiamas kaip duomenų judėjimas, kuris perkelia duomenų grupę, esančią funkciniame procese, į pastoviąją atmintį.

RKF (reikšmių koregavimo faktorius) - IFPUG ir NESMA metoduose apibrėžiamas kaip nefunkcinę sistemos elgseną atspindintis koeficientas.

S (skaitymas) - COSMIC metode apibrėžiamas kaip duomenų judėjimas, kuris perkelia duomenų grupę iš pastovios atminties į duomenų grupės reikalaujantį funkcinį procesą.

STV (skaičiavimo taisyklių vadovas) – funkcinio dydžio matavimo metodų dokumentas, kuriame aprašytos taisyklės, kaip reikia matuoti programinės įrangos funkcinį dydį.

TSK (techninio sudėtingumo koregavimo reikšmė) - Mark II metode apibrėžiama kaip nefunkcinę sistemos elgseną atspindintis koeficientas.

VLF (vidinis loginis failas) – IFPUG ir NESMA metoduose apibrėžiamas kaip programos ribose palaikomų logiškai susijusių duomenų ar valdymo informacijos grupė.

Priedai

1 priedas. Bendrosios sistemos charakteristikos

Bendrosios sistemos charakteristikos	Trumpas aprašas
1. Duomenų perdavimas	Programos tiesioginio bendravimo su procesoriumi laipsnis
2. Išskirstytas duomenų apdorojimas	Programos duomenų persiuntimas tarp savo komponentų laipsnis
3. Našumas	Programos atsako laiko ir našumo laipsnis
4. Dažnai naudojama konfigūracija	Sistemos naudojamų kompiuterinių resursų laipsnis
5. Transakcijų greitis	Verslo transakcijų greičio laipsnis
6. Tiesioginis duomenų įvedimas	Duomenų įvedimo per interaktyvias transakcijas laipsnis
7. Galutinio naudotojo efektyvumas	Programos naudojimo paprastumo ir suprantamumo naudotojui laipsnis
8. Tiesioginiai pakeitimai	Programos vidinių loginių failų tiesioginio atnaujinimo laipsnis
9. Sudėtingas apdorojimas	Programos vykdymo logikos sudėtingumo laipsnis
10. Pakartotinis panaudojamumas	Programos ir jos kodo pakartotinio panaudojimo kitose sistemose laipsnis
11. Įdiegimo paprastumas	Perėjimo iš ankstesnių aplinkų ir įdiegimo paprastumo laipsnis
12. Veikimo paprastumas	Programos rūpinimosi paleidimo, dubliavimo ir atstatymo procesais laipsnis
13. Panaudojimas keliose vietose	Programos tinkamumas skirtingoms techninėms ir programinėms aplinkoms laipsnis
14. Pakeitimų lengvumas	Programos vykdymo logikos ar duomenų struktūrų lengvo pakeitimo laipsnis
15. Kitų sistemų reikalavimai	Bendravimo su kitomis sistemomis laipsnis
16. Saugumas, privatumas ir	Sistemos saugumo ir duomenų

Bendrosios sistemos charakteristikos	Trumpas aprašas
audituojamumas	konfidencialumo užtikrinimo laipsnis
17. Naudotojo apmokymai	Naudotojo apmokymų reikalingumo laipsnis
18. Tiesioginis sistemos panaudojimas trečiųjų šalių	Leidimo naudotis sistema trečiosioms šalims laipsnis
19. Dokumentacija	Projekto metu sukuriamų dokumentų tipų skaičius

2 priedas. Programinės įrangos palaikymo standartiniai produktyvumo faktoriai

Kategorija	Produktyvumo faktoriaus pavadinimas	Produktyvumo faktoriaus aprašymas
Organizacija	Laidų ir versijų politika	Ateities laidų ir versijų aiškumas, formalumas, vidinis vientisumas ir ilgalaikis planavimas
Organizacija	Resursų prieinamumas poreikiams ateityje	Resursų pakankamumas ir sistemingas personalo, techninės įrangos, programinės įrangos, darbo vietų ir reikiamų suplanuotam palaikymui įgūdžių paskirstymas
Organizacija	Sutarčių sudarymo procedūra	Palaikymo sutarties, apibrėžiančios kiekvienos paslaugos tipą ir/arba transakciją, abipuses atsakomybes, paslaugų lygį, sukuriamų produktų priėmimo kriterijus ir kitas reikalingas kontrakto sąlygas, darnumas, išbaigtumas ir detalumas
Organizacija	Suinteresuotų šalių skaičius	Žmonių ir/arba organizacijų, dalyvaujančių palaikymo paslaugų valdyme ir sprendimų priėmime, skaičius
Organizacija	Prioritetų nustatymas ir pokyčių kontrolė	Pakeitimų prašymų klasifikavimas ir analizė pagal apibrėžtus kriterijus (pavyzdžiui, kritiškumas, skubumas, kaina), kad nustatyti prioritetus pakeitimų prašymams ir nuspręsti dėl reikiamų abipusių veiksmų
Organizacija	Organizacinė kultūra	Bendras personalo požiūris į palaikymo procesą ir jo vertinimas kompanijos lygyje, atitinkamas premijų mechanizmas ir kiti kultūriniai faktoriai

Kategorija	Produktyvumo faktoriaus pavadinimas	Produktyvumo faktoriaus aprašymas
Procesas	Programos kodo leidimo metodai ir įrankiai	Kodo redaktorių, transliavimo įrankių, kodo bibliotekų, kodo vientisumo įrankių lygis ir įtaka
Procesas	Testavimo metodai ir įrankiai	Testavimo atvejų ir medžiagos, testavimo veiklų, regresinio testavimo ir testavimo rezultatų tvarkymo įrankių ir procedūrų lygis ir įtaka
Procesas	Dokumentavimo metodai ir įrankiai	Palaikymo personalui ir galutiniams naudotojams reikalingų dokumentų kūrimo, tvarkymo ir platinimo įrankių ir procedūrų lygis ir įtaka
Procesas	Komunikavimo mechanizmai	Pakeitimų prašymų, klaidų ir gedimų registravimo, perdavimo ir tvarkymo metodų, įrankių ir procedūrų lygis ir įtaka
Procesas	Laidų metodai ir įrankiai	Modifikuotų programų ir susijusių veikimo aplinkos duomenų laidų įrankių ir procedūrų lygis ir įtaka
Produktas	Funkcionalumo reikalavimai	Reikalavimų ir verslo taisyklių, interfeisų lygio įvairovė ir sudėtingumas
Produktas	Patikimumo reikalavimai	Gedimų sunkumas ir gedimų įtaka naudotojams ir programos veikimui
Produktas	Panaudojamumo reikalavimai	Naudotojų skaičius, skirtingų naudotojų įgūdžių palaikymas, nenutrūkstamas veikimas, specialūs naudotojų pritraukimo reikalavimai
Produktas	Efektyvumo reikalavimai	Atsako ir operacijų veikimo laiko, skirtumų tarp operacinės ir kompiuterio apkrovos, transakcijų ir duomenų apimties reikalavimai
Produktas	Eksploatavimo patogumo reikalavimai	Aplinkos stabilumas, standartizuotas kodas ir komponentų struktūros, architektūros aiškumas, pakeitimų poreikis
Produktas	Pernešamumo reikalavimai	Pritaikomumas ir įdiegiamumas skirtingose aplinkose, architektūros ir struktūrinių komponentų atvirumas, platformų ir aplinkų kintamumas
Žmonės	Personalo kūrimo aplinkų	Palaikymo personalo patirtis ir žinios apie kūrimo

Kategorija	Produktyvumo faktoriaus pavadinimas	Produktyvumo faktoriaus aprašymas
	įgūdžiai	aplinkas, įrankius ir platformas
Žmonės	Personalo žinios apie programą	Palaikymo personalo žinios apie programas ir sąsajų sistemas (tiek tiekėjo, tiek ir kliento)
Žmonės	Personalo bendravimo įgūdžiai	Palaikymo personalo komandos formavimo ir bendravimo lygis, sugebėjimas bendradarbiauti su verslo partneriais
Žmonės	Personalo motyvacija ir atsakomybės	Personalo motyvacija sukurti programą ir gerinti kliento verslą
Žmonės	Atmosfera komandoje	Įtaka darbo sąlygoms, savimoka, profesionalios karjeros galimybės

3 priedas. Naujo kūrimo arba papildymo projektų standartiniai produktyvumo faktoriai

Kategorija	Produktyvumo faktoriaus pavadinimas	Produktyvumo faktoriaus aprašymas
Projektas	Kliento dalyvavimas	Kaip aktyviai klientas dalyvauja kūrimo procese
Projektas	Kūrimo aplinkos našumas ir prieinamumas	Įrankių prieinamumas ir įrangos (įrankiai, techninė įranga, fizinė aplinka, tinklas ir pan.) našumo lygis viso projekto metu
Projektas	IT personalo prieinamumas	IT personalo prieinamumas viso projekto metu
Projektas	Skirtingų suinteresuotų šalių skaičius	Bendras dalyvaujančių organizacijų ir priklausomų lygiagrečių projektų skaičius
Projektas	Grafiko spaudimas	Grafiko spaudimas, numatyto grafiko palyginimas su panašaus vidutinio projekto grafiku
Procesas	Standartų įtaka	Projekte taikomų standartų ir procedūrų kokybė
Procesas	Metodų įtaka	Projekto metu naudojamų metodų panaudojimas ir kokybė
Procesas	Įrankių įtaka	Projektui prieinamų įvairių (programavimo, testavimo, dokumentavimo, projektų valdymo ir pan.) įrankių panaudojimas ir kokybė
Procesas	Pokyčių valdymo lygis	Projekto funkcinių reikalavimų stabilumas ir nuspėjamumas, pokyčių valdymo proceso branda

Kategorija	Produktyvumo faktoriaus pavadinimas	Produktyvumo faktoriaus aprašymas
Procesas	Programinės įrangos kūrimo proceso branda	Programinės įrangos kūrimo procesų ir su gyvavimo ciklu susijusių projekto veiklų stabilumas
Produktas	Programinės įrangos funkcionalumo reikalavimai	Galutinių naudotojų poreikių suderinamumas, reikalavimų sudėtingumas, integracijos lygis
Produktas	Programinės įrangos patikimumo reikalavimai	Atsparumas gedimams ir atkuriamumas skirtinguose panaudojimo atvejuose
Produktas	Programinės įrangos panaudojamumo reikalavimai	Naudotojų interfeisų ir darbo eigos logikos suprantamumas ir lengvas išmokimas
Produktas	Programinės įrangos efektyvumo reikalavimai	Efektyvus resursų panaudojimas ir atitinkamas našumas kiekvienu panaudojimo atveju, prie bet kokios reikiamos darbo apkrovos
Produktas	Programinės įrangos eksploatavimo patogumo reikalavimai	Aplinkos stabilumas, planuojamas sistemos gyvavimo laikas, defektų diagnozavimo kritiškumas ir testų našumas
Produktas	Programinės įrangos pernešamumo reikalavimai	Pritaikomumas ir įdiegiamumas skirtingose aplinkose, architektūros ir struktūrinių komponentų atvirumas
Žmonės	Personalo analizės įgūdžiai	Projekto personalo analizės įgūdžiai projekto pradžioje
Žmonės	Personalo žinios apie programą	Projekto komandos (tiek tiekėjo, tiek ir kliento) žinios apie programos sritį projekto pradžioje
Žmonės	Personalo įgūdžiai su įrankiais	Projekto komandos (tiek teikėjo, tiek kliento) vidutinė patirtis su kūrimo ir dokumentavimo įrankiais projekto pradžioje
Žmonės	Projekto vadovybės patirtis	Projekto vadovo ir kito esminio personalo patirtis
Žmonės	Projekto komandos komandiniai įgūdžiai	Projekto komandos narių sugebėjimas efektyviai dirbti kartu, laikantis geriausių projekto praktikų