VILNIAUS UNIVERSITETAS

Andrius
VALATAVIČIUS

Taikomųjų programų sąveikumo vertinimas taikant autonominio skaičiavimo technologijas

DAKTARO DISERTACIJA

Gamtos mokslai,
informatika N 009

VILNIUS 2019

Disertacija rengta 2014-2018 metais Vilniaus universitete.

Mokslinis vadovas:
prof. dr. Saulius Gudas (Vilniaus universitetas, gamtos mokslai, informatika – N 009).

Mokslinis konsultantas:
prof. dr. Audrius Lopata (Vilniaus universitetas, gamtos mokslai, informatika – N 009).

PADĖKA

Nuoširdžiai dėkoju savo šeimos nariams ir visai giminei už palaikymą, net ir sunkiausiomis akimirkomis motyvavusiems mane judėti į priekį ir nepasiduoti.

Labai dėkoju darbo vadovui prof. dr. Sauliui Gudui už puikų ir nuoširdų vadovavimą, palaikymą ir skatinimą siekti užsibrėžtų tikslų, kantrybę ir didžiulę pagalbą.

Ačiū UAB „Kvantas“ ir UAB „Intellerts“ komandoms už palaikymą rašant disertaciją ir suteiktą motyvaciją ir patarimus.

Dėkoju dr. Dariui Dilijonui už vertingas pastabas ir idėjas, kurios prisidėjo prie disertacijos įgyvendinimo.

Dėkoju recenzentams prof., dr. Leonidui Sakalauskui ir prof., dr. Rimantui Butleriui už vertingas ir konstruktyvias pastabas.

Andrius Valatavičius

Paveikslų sąrašas

Pav. 1. Publikacijų analizė Google Scholar paieškos sistemoje..... 15
Pav. 2. Sąveikumo barjerai..... 33
Pav. 3. Baziniai veiklos sąveikumo konceptai ir jų tarpusavio ryšiai, adaptuota (Chen, et al., 2008)..... 35
Pav. 4. Taikomųjų programų sąveikumo problemos (gamyklos veiklos srities pavyzdys)..... 39
Pav. 5. Veiklos procesų reinžinerija su sąveikumo komponentu ir be jo.40
Pav. 6. Skirtingi kibernetikos lygiai..... 43
Pav. 7. IBM Autonominio skaičiavimo komponento architektūra, adaptuota (Jacob, et al., 2004)..... 47
Pav. 8. Vidinio modelio naudojimas kuriant baltosios dėžės sprendimus. 48
Pav. 9. Modifikuotas MDA modelis, pagrindžiantis vidinio modelio naudojimą sistemų analizei. 49
Pav. 10. Lyginamoji EVC (viršuje) ir AK (apačioje) koncepcinė diagrama 50
Pav. 11. MEFF veikimo principinė diagrama 51
Pav. 12. Elementaraus valdymo ciklo (EVC) atitikmenys autonominio skaičiavimo valdymo ciklo (ASVC) funkcijoms 54
Pav. 13 Programinio kodo iškarpa – skirtingų duomenų struktūrų jungimo grafinis pavyzdys..... 63
Pav. 14. Užsakymų WSDL schemos susiejimas su veiklos procesų modeliu (CIM)..... 64
Pav. 15. Taikomųjų programų šaltinių analizės naudotojo sąsaja..... 68
Pav. 16. Duomenų bazės architektūra MSSQL serveryje. 69
Pav. 17. Nustatymų langas pridėti tinklo sąsają..... 70
Pav. 18. Analizės agento veiksmų diagrama..... 70
Pav. 19. Paslaugos metaduomenų nuskaitymas, atliekamas šia veiksmų sekos diagrama. 71
Pav. 20. Sąveikumo vertinimo sistemos su autonominiu komponentu koncepcinė diagrama..... 75
Pav. 21. Operacijų sąveikumo „šilumos žėmėlapis“ naudojant Levenshteino redagavimo nuotolio algoritmą..... 79
Pav. 23. Sąveikumo vertinimo matrica (M1 matricos dalis) tarp ExactOnline ir NMBRS..... 80
Pav. 24. Matrica M1 apribota operacijomis, kurių panašumas didesnis kaip 65 %..... 81

Pav. 25. Panašumo vertinimas naudojant redagavimo nuotolio algoritmus: a – Levenshteino, b – Jaro-Winklero, c – Jaccardo, d – Ilgiausios bendros sekos, e – visi metodai 83
Pav. 26. Tekstinės analizės lyginimas tarp veiklos programų. 84
Pav. 26. ExactOnline kartu su SuiteCRM struktūrinis panašumas naudojant LSA metodą 85
Pav. 27. Region 1 (Pav. 26), skirtingų sistemų panašumas naudojant LSA metodą. 87

Lentelių sąrašas

Lentelė 1. Integracijos ir sąveikumo problemų lentelė: 37
Lentelė 2. Žinomi veiklos programinės įrangos sąveikumo sprendimai. 42
Lentelė 3. Automoninio skaičiavimo technologijų galimybė pagerinti tinklo paslaugų galimybes 55
Lentelė 4. Verslo programų sąveikumo vertinimo metodų lyginimas. .. 58
Lentelė 5. Sutrumpintas tiriamų objektų sąrašas (tęsinys 1 priede) 72
Lentelė 6. Pasirinktų taikomųjų programų sąveikumo matavimai LISI metodu 74
Lentelė 7. Unikalios operacijos kiekvienos programos tinklo paslaugoje 77
Lentelė 8. Sąveikumo galimybių tyrimo rezultatai – operacijų skaičius pagal panašumo įverčio ribas 82
Lentelė 9. Dimensijų mažinimo naudojant latentinę semantinę analizę vektoriaus koordinatės 2D plokštumoje (V1, V2). 86

Žymėjimai

E	Veikla (angl. *Enterprise*)
S	Sistema
O	Objektas – sistemos dedamasis elementas, esinys (angl. *Object*)
h	Duomenų struktūra (angl. *Schema*) – objekto duomenų stuktūros aprašas, susidedantis iš: laukų pavadinimų, laukų tipų ir laukų ilgių rinkinių
d	Duomenys
f	Funkcija
p	Procesas
g	Tikslas (angl. *Goal*)
M	Modelis
k	Žinios, informaciniai elementai (angl. *Knowledge*)
r	Ryšys pvz.: $r<o_1,o_2>$ (ryšys tarp objektų 1 ir 2) arba $r<f_1, p_1>$ (ryšys tarp 1 funkcijos ir 1 proceso)
λ	Panašumo įvertis
λ_{red}	Redagavimo nuotolio įvertis
θ	Panašumo slenksčio konstanta (jei ji viršijama, laikoma, kad sistemos panašios)

Sąvokos

Agentas	Asmuo arba programa, savarankiškai ir autonomiškai atliekanti konkrečius veiksmus arba užduotis tam tikram tikslui įgyvendinti ir atstovaujanti kitą asmenį ar agentą.
Autonominės brandos indeksas	Tai indeksas, leidžiantis nustatyti įdiegto sprendimo autonomijos lygį, t. y. galimybę autonomiškai atlikti veiksmus (angl. *Autonomic Maturity Index, AMI*).
Autonominis komponentas	Tai autonominės skaičiavimo technologijos vienetas, gebantis pagal tam tikrą autonomijos brandos lygį atlikti užduotis. Sudėtingesniems uždaviniams galima sukurti daugiau kaip vieną autonominį komponentą, kurie sąveikauja vienas su kitu.
Autonominis valdiklis	Tai komponentas, kuris valdo kitą programinę arba techninę įrangą naudodamas grįžtamojo ryšio valdymo ciklą. Valdymo ciklas susideda iš stebėjimo, analizavimo, planavimo ir vykdymo funkcijų.
Autonominio skaičiavimo technologija	Tai yra programų sistemų technologija, leidžianti pagal tam tikrą autonomijos lygį vykdyti užduotis (angl. *Autonomic Computing*). Tikslas – sukurti save valdančią sistemą paslepiant valdymo sudėtingumą nuo naudotojų (Horn, 2001; Kephart & Chess, 2003). Paslėpus probleminės srities valdymo sudėtingumus naudotojai gali išsilaisvinti nuo sistemų priežiūros ir susitelkti į aukštesnio lygio (į verslo vykdymo orientuotas) problemas, adaptyvų programinį sprendimą, reaguojantį į aplinkos pokyčius.
Konceptas	Abstrakti idėja, atspindinti esminius (pagrindinius) bruožus, apie tai, ką ji reprezentuoja. Pvz.: medžio konceptas yra apibendrintos medžio savybės.
Modulis	Produktas (įrengimas, programinės įrangos paketas, integruojančioji infrastruktūra ir pan.), kurį tikslinga naudoti realizuojant konkretų procesą (Gudas, 2012).
Modelis	Tai abstrakti konstrukcija, kuria mėginama pakartoti kai kurias realios sistemos savybes.

Organizacijų architektūros karkasas	Organizacijos architektūros karkasas (angl. *Enterprise architecture framework*) apibrėžia principus ir struktūras, kurias naudojant aprašoma organizaciją sudarančių sistemų (strateginių veiklų, veiklos procesų, taikomųjų programų, techninės įrangos) architektūra. Žinomiausi EA karkasai: Zachman ISA, MODAF, ArchiMate ir kiti.
Organizacijų taikomųjų programų integracija	Organizacijų taikomųjų programų integravimas (angl. *Enterprise Application Integration, EAI*) – tai specializuotos technologijos ir paslaugų naudojimas verslo srityje siekiant integruoti taikomąsias programas arba technines įrangos sistemas.
Procesas	Darbo srautą organizacinėje sistemoje tarp išorinio tiekėjo ir išorinio naudotojo nurodantis didžiausias veiklos vienetas (Gudas, 2012).
Sąsaja	Tai prieiga prie valdomų duomenų šaltinio (angl. *Interface*), tiksliau – sistemos arba serverio. Sąsajos leidžia valdyti išteklius per daviklius (angl. *Sensors*) ir valdiklius (angl. *Effectors*).
Sąveikumas	Sąveikumas (angl. *Interoperability*) – tai veiklos taikomųjų programų ir jų posistemių galimybė efektyviai mainytis duomenis ir informacija, naudoti viena kitos funkcionalumus savo reikmėms.
Taikomoji programa	Tai veiklos programų sistemos, skirtos valdyti veiklos išteklius ir procesus (angl. *Enterprise Applications* ir *Enterprise Systems*).
Taikomųjų programų integracija	Sujungti skirtingi programiniai komponentai į bendrą visumą siekiant sukurti vientisą sistemą.
Tinklo paslauga	Tinklo paslauga (angl. *Web Service*) – tai veiklos sistemos dalis, kurioje apibrėžiama, kaip duomenys gali būti pasiekiami, ir leidžia pasiekti duomenis naudojant atvirus protokolus ir standartus, tokius kaip REST ir SOAP. Tinklo paslaugos leidžia jungti skirtingas programas.
Veikla	Tai verslo veikla, kitaip tariant, įmonė, organizacija arba kitokia verslo veikla, naudojanti programų sistemas savo procesuose (angl. *Enterprise*).

Žinios	Apskritai tai – mokėjimo, išsilavinimo turinys, informacija. Disertacijoje tai – duomenų rinkinys, padedantis atlikti veiksmus ir priimti sprendimus konkrečioje situacijoje.

Santrumpos

AMI – Autonominės brandos indeksas (angl. *Autonomic Maturity Index*)

ASVC – Autonominio skaičiavimo valdymo ciklas (angl. *Autonomic Computing Control Loop*)

BDAR – Bendrasis duomenų apsaugos reglamentas

BPM – Veiklos procesų modelis (angl. *Business Process Model*)

CIM – Nuo skaičiavimų nepriklausomi modeliai, tai yra MDA modelių lygmuo (angl. *Computation Independent Model*)

CRM – Santykių su klientais valdymo sistema (angl. *Customer Relationship Management*)

EAI – Organizacijų taikomųjų programų integravimas (angl. *Enterprise Application Integration*)

ERP – Išteklių planavimo sistema (angl. *Enterprise Resource Planning*)

EVC – Elementarus valdymo ciklas (angl. *EMC, Elementary Management Cycle*)

ICT – Informacijos ir komunikacijos technologijos (angl. *Information and Communications Technology*)

MDA – Modeliais grindžiama architektūra (angl. *Model Driven Architecture*)

PIM – Nuo platformos nepriklausomi modeliai, tai yra MDA modelių lygmuo (angl. *Platform Independent Model*)

PSM – Konkrečiai platformai pritaikyti modeliai, tai yra žemiausias modelių lygmuo iš MDA architektūros, gretimas PIM (angl. *Platform Specific Model*)

TVS – Turinio valdymo sistema (angl. *Content Management System, CMS*)

GUI – Grafinė vartotojo sąsaja (angl. *Graphical User Interface*)

TURINYS

PADĖKA 3
Paveikslų sąrašas 4
Lentelių sąrašas 5
Žymėjimai 6
Sąvokos 7
Santrumpos 10
TURINYS 11
ĮVADAS 13
1.1. Tyrimų sritis 15
1.2. Darbo aktualumas 16
1.3. Darbo tikslas ir uždaviniai 18
1.4. Mokslinis naujumas 18
1.5. Ginamieji teiginiai 19
1.6. Darbo rezultatų aprobavimas 20
1.7. Disertacijos struktūra 21
2. VERSLO VEIKLOS PROGRAMŲ SĄVEIKUMO SPRENDIMŲ APŽVALGA 23
2.1. Apibrėžimai 25
2.2. Taikomųjų programų integracija 27
2.2.1. Vertikalioji integracija 28
2.2.2. Horizontalioji integracija 28
2.2.3. Žvaigždinė integracija 29
2.3. Sąveikumas 29
2.3.1. Sąveikumo barjerai 30
2.3.2. Sąveikumo interesai 34
2.3.3. Sąveikumo sprendimai 34
2.4. Integracijos ir sąveikumo problemos 35
2.5. Programų sąveikumas – kibernetikos uždavinys 43

2.6. Autonominio skaičiavimo komponentai 45
2.7. Modeliais grindžiama architektūra sąveikumo automatizavimo problemai spręsti 48
2.8. Taikomųjų programų tinklo paslaugos 54
2.9. Taikomųjų programų sąveikumo vertinimas 56
2.10. Redagavimo nuotolio skaičiavimai 58
2.11. Skyriaus išvados 60
3. VEIKLOS PROGRAMŲ SĄVEIKUMO MATAVIMAI 61
3.1. Siūlomas autonominis integracijos sprendimas 63
3.2. Sąveikumas naudojant autonominio skaičiavimo technologijas 65
3.3. Skyriaus išvados 66
4. SĄVEIKUMO GALIMYBIŲ VERTINIMO EKSPERIMENTO APRAŠYMAS 68
5. SĄVEIKUMO GALIMYBIŲ VERTINIMO EKSPERIMENTO REZULTATAI 74
5.1. Vertinimas redagavimo nuotolio metodu 77
5.2. Vertinimas tekstinės analizės metodais 83
5.3. Vertinimas latentinės semantinės analizės metodu 84
5.4. Skyriaus išvados 87
6. IŠVADOS 89
6.1. Teorinės dalies išvados 89
6.2. Praktinės dalies išvados 90
Bibliografija 92
1. PRIEDAS – Tiriamų objektų sąrašas 98
2. PRIEDAS – Programų ir taikomųjų programų sudėtingumo lyginimas 100

ĮVADAS

Organizacijų veiklos programinė įranga nuolat plečiasi ir tobulėja, bet dažnai nepakanka vieno taikomosios programos paketo visiems organizacijos veiklos procesams. Organizacijų taikomoji programinė įranga – tai skaitmeninės programos, padedančios vykdyti ir valdyti organizacijų veiklą, pavyzdžiui, valdyti santykius su klientais (angl. *Customer Relationship Management, CRM*), valdyti elektronines parduotuves (angl. *E-Commerce*); sandėlio valdymo sistemos, apskaitos sistemos, veiklos išteklių planavimo sistemos (angl. *Enterprise Resource Planning, ERP*). Dažnai skirtingas veiklos funkcijas aptarnaujančios taikomosios programos ir jų paketai nėra integruoti, neretai net ir nėra galimybės jų integruoti dėl skirtingų šių programų gamintojų. Tačiau programos turi sąveikauti dėl verslo veiklos procesų palaikymo – duomenų mainai ir funkcijų vykdymas yra esminė sąveikaujančių programų paskirtis.

Įvairaus dydžio organizacijų veiklose naudojami sprendimai (programos) turi heterogeninių duomenų (panašių duomenų), tačiau duomenų mainus tarp skirtingų paketų užtikrina aptarnaujantis personalas – žmonės, integravimo specialistai. Šiuo metu nėra automatizuotų organizacijų taikomosios programinės įrangos integravimo ir sąveikumo sprendimų. Taip pat nėra būdų automatiškai atlikti sąveikumo sprendimų kūrimo galimybių įvertinimą. Tad taikomųjų programų sąveikumo problema yra aktuali moksliniu ir praktiniu požiūriais. Verslo taikomųjų programų sąveikumo plėtrai būtina sukurti sąveikumo vertinimo metodus, kurie leistų bent iš dalies automatizuoti analitiko, integracijos architekto ir programuotojo darbą.

Darbe pateikiamas sąveikumo galimybių vertinimas remiantis verslo veiklos programų tinklo paslaugų sąveikumo galimybių analize, naudojant verslo veiklos procesų modeliavimą BPM, modeliais grindžiamą sistemų architektūrą MDA ir autonominio skaičiavimo technologiją.

Verslo taikomųjų programų integravimas apibrėžiamas kaip programų jungimas, kai jų duomenys laikomi bendroje duomenų bazėje ir duomenų architektūra gali būti nepriklausoma nuo taikomųjų programųjų architektūros, arba tai – sistemos architektūra, leidžianti sistemos naudotojui pasiekti visas funkcijas per vieną naudotojo sąsają (Gartner, 2019).

Verslo taikomųjų programų sąveikumas apibrėžiamas kaip prietaisų ar programų, pagamintų skirtingų gamintojų, galimybė dirbti kartu ir mainytis duomenimis (Gartner, 2019).

Veiklos programos integravimas (angl. *Enterprise Application Integration, EAI*) – tai programos arba kompiuterizuotos sistemos architektūriniai principai, integruojantys kompiuterizuotas veiklos programas.

Veiklos programų integravimas dažnai atliekamas naudojantis technologijomis ir paslaugomis, kurios suformuoja tarpines programas (angl. *Middleware*), leidžiančias atlikti atskirtų (galimai nesusijusių, neturinčių bendros duomenų bazės) kompiuterizuotų sistemų ir programų integraciją. Veiklos programų integravimas – tai procesas, per kurį šios programos sujungiamos ir sudaroma viena struktūra siekiant supaprastinti ir galimai automatizuoti veiklos procesus, taip pat siekiant išvengti IT ir BP (angl. *Business Process*) reinžinerijos. Jungiamos programos gali būti siejamos per duomenų bazes, programos sąsajas API, tinklo paslaugas arba (rečiau) per naudotojo sąsają (GUI) (CEITON technologies, 2015).

Taikomųjų programų sąveikumo (angl. *Interoperability*) ir integralumo problemos kyla dėl organizacijų veiklos dinaminio pobūdžio (nuolat kinta veiklos procesai ir su jais siejama informacija / duomenys). Efektyvus šių problemų sprendimas gali būti pasiektas automatizavus taikomųjų programų integravimo ir sąveikumo sprendimų kūrimo procesą. Darbe nagrinėjama taikomųjų programų sąveikumo problema.

Kuriami metodai, sprendžiantys problemą, kaip pagerinti taikomųjų programų sąveikavimo sprendimų kūrimą, integravimo sprendimų kūrimą ir užtikrinti kokybiškus duomenų mainus ne tik tarp skirtingų taikomųjų programų, bet ir tarp skirtingų organizacijų (Rahm & Bernstein, 2001; Valatavičius & Dilijonas, 2014; McCann, et al., 2005; Halevy, et al., 2005). Dėl šios priežasties mokslinėje literatūroje nagrinėjamos organizacijų taikomųjų programų (ERP, CRM, TVS, E. komercijos, Apskaitos ir kt.) integracijos ir sąveikumo problemos. Analizuojami dinaminiai metodai siekia gerinti taikomųjų programų integracijos procesus ir juos automatizuoti arba bent sutrumpinti integracijos palaikymo ir aptarnavimo laiką bei sumažinti reikiamų žinių kiekį integracijos sistemoms palaikyti.

Disertacijoje pateikiama sąveikumo ir integracijos teorija. Aprašomi esminiai pasiekimai ir sukauptos žinios šioje srityje. Darbe nagrinėjami kitų autorių programų integravimo bei sąveikumo sprendimai nesiremia diskrečiais skaičiavimais ar deterministiniais metodais. Dažniausiai analizuojami metodai remiasi įžvalgomis, ekspertų žiniomis, klausimynais ir panašia nuo vertintojo žinių priklausančia informacija. Todėl siūlomas kiekybinis vertinimo metodas analizuoti šių programų tinklo paslaugų aprašus, pasitelkiant šiuos metodus: veiklos procesų modelių analizę, modeliais grindžiamą architektūrą, autonominių skaičiavimų technologiją, teksto redagavimo nuotolio ir semantinės analizės metodus. Jungiant šias skirtingas sritis ir technologijas į siūlomą metodą teigiama, kad įmanoma atlikti programų sąveikumo galimybių vertinimą ir nustatyti, ar skirtingos verslo programos gali sąveikauti, taip pat galima tokį sąveikumo sprendimo

kūrimą automatizuoti. Eksperimento aprašymo dalyje išdėstomi eksperimento principai, seka ir planas. Eksperimento dalyje taip pat nurodomi papildomi įrankiai, naudoti kuriant sprendimą: SQL Server, R, R studio, C# ir Microsoft visual studio. Eksperimento dalyje aprašomos analizuotos verslo programos, atliekama jų sąveikumo vertinimo analizė. Išanalizuota 13 programų.

Disertaciją sudaro 6 skyriai, 128 puslapiai, 68 šaltiniai, 28 paveikslai ir 8 lentelės, 143.895 spaudos ženklų su tarpais – 3,59 autoriniai lankai.

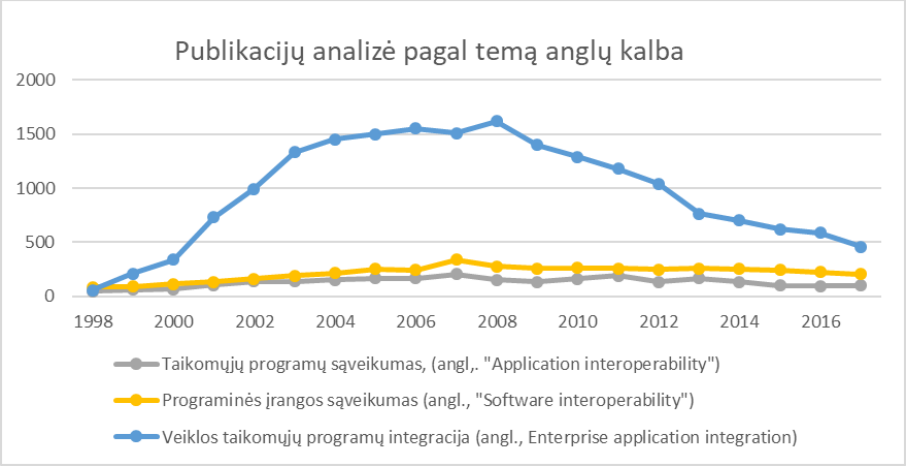
1.1. Tyrimų sritis

Šiuolaikinių organizacijų veiklos procesai greitai kinta, naudojamos skirtingų gamintojų taikomosios programos, tokios kaip e. prekybos platformos, sandėlio valdymo, finansų apskaitos, santykių su klientais valdymo ir daugybė kitų taikomųjų programų.

Pagrindinė šio darbo tyrimo sritis yra informacijos (veiklos duomenų) mainai tarp taikomųjų programų dinaminėje verslo veiklos aplinkoje. Ši taikomųjų programų tyrimo sritis yra vadinama taikomųjų programų sąveikumu (angl. *Application Interoperability*).

Užsienio literatūroje šioje tyrimų srityje dažnai naudojami tokie terminai: *Enterprise Application Integration*, *Application Interoperability***.**

Paveiksle (Pav. 1) parodytas terminų populiarumas – praeitame dešimtmetyje populiarėjo debesų technologijos ir verslo valdymo sistemos, o tai lėmė ir populiarėjančias integracijos bei sąveikumo sprendimo temas, tačiau nors verslo programų integracijos temos populiarumas sumažėjo, daugelis problemų liko neišspręstos, todėl tema ir toliau tebėra nagrinėtina.



Pav. 1. Publikacijų analizė Google Scholar paieškos sistemoje.

Dažnai pasitaiko tokių terminų: *Enterprise Application Integration, ERP Software Integration, Common ERP Integration, CRM Integration with SAP*.

Tyrimo mokslinė problema – verslo organizacijose naudojama daug taikomųjų programų, kurios yra sudėtingos – susideda iš posistemių programinės įrangos, todėl atsiranda daug heterogeninės informacijos. Problema yra ta, kad duomenų mainai tarp skirtingų gamintojų taikomųjų programų nėra praktiškai užtikrinami, nors ir yra pateikiamos rekomendacijos (Papazoglou, 2008; Walsh, 2002).

Sukurti duomenų mainų protokolai SOAP ir REST su rekomendacijomis, kaip juos aprašyti ir taikyti (įdiegti) sistemose, tačiau retai (ypač REST) laikomasi šių protokolų specifikacijos.

- SOAP – paprastas objektų prieigos protokolas (angl. *Simple Object Access Protocol).* Tai duomenų mainų protokolas, turintis paslaugomis grindžiamą metodologiją suprojektuotose taikomosiose programose. SOAP protokolas naudojamas struktūrizuoti ir keistis duomenimis tarp taikomųjų programų.
- REST – reprezentatyvus būsenų perkėlimas (angl. *Representational State Transfer*). Tai duomenų mainų architektūrinis stilius. Jis naudojamas struktūrizuoti ir keistis duomenimis tarp taikomųjų programų. REST pristatytas 2000 metais Fielding ir kt. (Fielding & Taylor, 2000).

Tyrimo objektas – duomenų ir funkcijų mainai tarp skirtingų gamintojų taikomųjų programų, kurios aptarnauja veiklos procesus.

1.2. Darbo aktualumas

Verslo taikomųjų programų integravimo tema jau gana seniai nagrinėjama. Chen ir kiti (Chen, et al., 2008) apibrėžė verslo sąveikumo ir integravimo architektūrą, taip pat papildė veiklos sąveikumo karkasą (Chen, 2006); Dijkmanas ir kiti apibrėžė veiklos procesų panašumo vertinimo metodus ir vertinimo procesą (Dijkman, et al., 2014); Dzemydienė ir kiti apibrėžė informacinių taikymo poreikį elektroninių viešųjų paslaugų sektoriuje (Dzemydienė & Naujikienė, 2009). El-Halwagi 2006 metais apibrėžė procesų integraciją ir jos poreikį (El-Halwagi, 2006).

Programų integravimas – tai skirtingų taikomųjų programų paketų jungimas į visumą siekiant sukurti vientisą veiklos valdymo sistemą (pavyzdžiui, ERP). Programų sąveikumas – tai taikomųjų programų paketų duomenų mainų procesai. Kuriant programų sąveikumo sprendimus siekiama ne sukurti integruotus sprendimus, o pritaikyti esamus ir adaptuotis. Programų

integravimas ir programų sąveikumas apima galimus veiksmus su skirtingų objektų duomenimis. Programos integruojamos kuriant sistemos architektūrą, o jų sąveikumas užtikrinamas, kai programos jau sukurtos ir jų struktūros ar veikimo principų negalima pakeisti. Sąveikumas ir integravimas – tai duomenų mainų sprendimai, kurie neretai painiojami – pavyzdžiui, EAI (angl. *Enterprise Application Integration*) ir EIF (angl. *European Interoperability Framework*) sprendžia labai panašias problemas: duomenų mainų tarp skirtingų objektų (Chen, et al., 2008; Linthicum, 2000).

Toliau nagrinėjamas tik sąveikumas, nes dinaminėje organizacijoje natūraliai keičiamos naudojamos taikomosios programos ir jų paketai, todėl norint užtikrinti efektyvų verslo procesų veikimą reikia įgyvendinti šių taikomųjų programų sąveikumo sprendimus. Dažniausiai integruotos programinės įrangos sprendimai, tokie kaip ERP, yra sukurti vieno gamintojo, o neintegruotos programos – kelių skirtingų gamintojų. Kadangi šios programos veikia vieną veiklą, reikia, kad jos sąveikautų ir užtikrintų sėkmingus duomenų mainus.

Sąveikumo vertinimo metodai šiuo metu yra subjektyvūs ir remiasi srities specialistų apklausomis bei analize. Pavyzdžiui, LISI metodas (Kasunic, 2001) pateikia aplinkos galimybių, leidžiančių programinei įrangai sąveikauti, vertinimą, tačiau neapibūdina vidinių programinės įrangos savybių ir galimybių keistis duomenimis.

Esamais sąveikumo vertinimo sprendimais (SPICE, LISI, OIM, LCIM, EIMM) (Guédria, et al., 2008) galima įvertinti verslo programų sąveikumo įgyvendinimo galimybes, pavyzdžiui:

- Fizines galimybes įgyvendinti integracijos ar sąveikumo sprendimus (Kasunic, 2001);
- Galimybes perduoti signalą per aparatinę įrangą (serverius, kompiuterius, tinklo plokštes);
- Galimybes perduoti signalą per programinę įrangą (tinklo plokščių tvarkykles, operacines sistemas, duomenų perdavimo ir kodavimo protokolus).

Šie sprendimai turi patikrinti taikomųjų programų savybes ir įvertinti, ar įmanomi duomenų mainai. Jie turi savybes apibrėžti aplinką, nustatyti integracijos ir sąveikumo sąlygas, įvardyti tam tikras problemas ir poreikį, tačiau išvardyti sprendimai neanalizuoja integracijos ir sąveikumo procesų iš taikomųjų programų perspektyvos, o pastarosios atlieka pagrindinį vaidmenį šioje srityje. Kiekvienam integracijos srities ekspertui svarbu turėti gilių žinių apie integruojamas sistemas ir galimybę keistis duomenimis. Sprendimai (Dzemydienė & Naujikienė, 2009; Kasunic, 2001) neįvertina sistemos

sandaros, duomenų perdavimo sąsajos architektūros. Dėl šios priežasties nekuriama efektyvių sprendimų integracijai ir sąveikumui užtikrinti. Verslo taikomųjų programų sąveikumo plėtrai būtina sukurti sąveikumo vertinimo metodus, kurie leistų bent iš dalies automatizuoti analitiko, integracijos architekto ir programuotojo darbo aplinką.

1.3. Darbo tikslas ir uždaviniai

Darbe sprendžiama organizacijų taikomųjų programų duomenų mainų proceso (sąveikumo) automatizavimo problema, siekiama sumažinti sąveikumo sprendimų įgyvendinimo sąnaudas (laiko, lėšų, žmogiškųjų išteklių, ekspertinių žinių).

Darbo tikslas – nustatyti taikomųjų programų sąveikumo įvertinimo principus ir sukurti sąveikumo įvertinimo automatizavimo metodą, kuris grindžiamas taikomųjų programų priežastinių ryšių analize.

Darbo objektas – organizacija, kurios veiklos procesai nuolat kinta ir kuri naudoja daugiau kaip vieno tiekėjo programas, galinčias turėti sąveikumo problemų, tokių kaip: duomenų kartojimasis, veiklos procesų dvigubinimas.

Uždaviniai

Darbo uždaviniai tikslui pasiekti:

1. Išanalizuoti aktualias organizacijų taikomųjų programų integravimo ir sąveikumo problemas, nustatyti taikytinus sprendimų principus.
2. Išanalizuoti organizacijų taikomųjų programų sąveikumo vertinimo metodų privalumus ir trūkumus, apibrėžti darbe siūlomo *sąveikumo vertinimo* sprendimo principus.
3. Sukurti taikomųjų programų sąveikumo kiekybinio įvertinimo metodą sujungiant organizacijos architektūros karkasus (angl. *Enterprise Architecture Frameworks*), modeliais grindžiamos architektūros (MDA) požiūrį (CIM lygmens modelius, PIM lygmens modelius) ir autonominio skaičiavimo technologiją.
4. Sukurti taikomųjų programų sąveikumo įvertinimo sistemos prototipą, grindžiamą teksto analizės metodais ir atlikti eksperimentinį tyrimą.

1.4. Mokslinis naujumas

1. Taikomųjų programų sąveikumo analizei naudoti veiklos procesų modeliai, identifikuojantys priežastinius veiklos procesų ryšius, ir

šios srities žinios leidžia atsekti priežastinius taikomųjų programų komponentų ryšius (taikant CIM, PIM lygmenų modelius).
2. Sukurtas taikomųjų programų sąveikumo galimybių kiekybinio įvertinimo metodas, kuris naudoja žinių struktūras, specifikuotas pagal CIM ir PIM modelius, autonominio skaičiavimo technologiją ir teksto analizės įrankius.
3. Pritaikius teksto redagavimo nuotolį naudojant Levenshteino, Jaro-Winklerio, Jaccardo ir ilgiausios bendros sekos metodus, atliktas programų sąveikumo galimybių pagal programų panašumą vertinimas.
4. Pritaikyta latentinė semantinė analizė programų sąveikumo galimybėms pagal programų struktūrą įvertinti.

1.5. Ginamieji teiginiai

1. Veiklos architektūros karkasų (angl. *EA frameworks*) ir MDA taikymas kartu sprendžiant taikomųjų programų sąveikumo problemas leidžia vizualizuoti ir identifikuoti atitikmenis tarp taikomųjų programų komponentų ir veiklos procesų priežastinių ryšių.
   Veiklos architektūros lygmenyje (veiklos architektūros karkase) sudaromi procesų modeliai atitinka MDA CIM lygmens modelius, taikomųjų programų architektūros lygmuo atitinka MDA PIM lygmens modelius. Šio atitikimo vizualizacija pateikiama 3 skyriuje, kuriame aprašytas ArchiMate karkaso naudojimas specifikuoti ryšiui tarp veiklos architektūros modelio ir taikomųjų programų architektūros lygmenų.
2. Sukurto verslo veiklos programų sąveikumo galimybių kiekybinio įvertinimo metodo pakanka kiekybiškai nustatyti sintaksinį ir semantinį programų panašumą.
3. Sąveikumo galimybėms įvertinti galima panaudoti CIM ir PIM modelius, kurie apima priežastinius ryšius tarp veiklos procesų ir jų atvaizdus (transformacijas) į taikomųjų programų komponentų sandarą.
4. Verslo veiklos programų sąveikumo sprendimas grįstas autonominių skaičiavimų technologijomis, kurios, kintant organizacijos veiklos procesams ar taikomosioms programoms, gali aptikti pokyčius, lemiančius taikomųjų programų sąveikumą.

1.6. Darbo rezultatų aprobavimas

Pagrindiniai tyrimo rezultatai spausdinti 4 mokslinėse publikacijose, rezultatai pristatyti 2 tarptautinėse mokslininkų konferencijose ir 3 respublikinėse konferencijose.

Publikuoti darbai:

- Valatavičius, A. G. S., 2017. Apie taikomųjų programų sąveikumo metodologiją, grindžiamą giluminėmis žiniomis. *Informacijos mokslai*, pp. 83–113.
- Gudas, S., Valatavicius, A., 2017. Normalization of Domain Modeling in Enterprise Software Development. *Baltic Journal of Modern Computing, 5*(4), pp. 329–350.

Konferencijų medžiaga:

- Valatavičius, A. & Gudas, S., 2015. *Enterprise Software System Integration Using Autonomic Computing.* Tartu, CEUR-WS, pp. 156–163.
- Valatavičius, A. & Gudas, S., 2015. *Towards Business Process Integration Using Autonomic Computing*. Kaunas, Technologija, pp. 81–81.
- Valatavičius, A. & Gudas, S., 2017. *Advanced Evaluation Methods of Multiple Application Software Interoperability*. Vilnius, Vilniaus Universitetas, p. 52.
- Valatavičius, A. G. S., 2017. Apie taikomųjų programų sąveikumo metodologiją, grindžiamą giluminėmis žiniomis. *Informacijos mokslai*, pp. 83–113.
- Valatavičius, A. & Gudas, S., 2018. *A Deep Knowledge Based Evaluation of Applications Interoperability*. Druskininkai, Vilniaus Univesitetas.
- Valatavičius, A. & Gudas, S., 2018. *Measuring Enterprise Application Software Interoperability Capability*. Estonia, CEUR-WS, pp. 104–113.

Pranešimai skaityti šiose konferencijose:

- Tarptautinė konferencija: Doktorantų konsorciumas BIR 2015 Estijoje: „Enterprise Software System Integration Using Autonomic Computing“;
- Tarptautinė konferencija: DB&IS 2016 Latvijoje tema: „Modelling Dynamic Enterprise Environment to Maintain Interoperability of Applications“;

- Tarptautinė konferencija: DAMSS 2016 „Data Analysis Methods for Software Systems“;
- Konferencija: XVIII tarptautinė kompiuterininkų konferencija LIKS 2017: „Towards Deep Knowledge Based Interoperability of Applications“. Straipsnis priimtas publikacijai žurnale „Informacijos Mokslai“;

1.7. Disertacijos struktūra

Įvade apžvelgiama tyrimų sritis, aptariamas mokslinio darbo aktualumas, pristatomi disertacijos darbo tikslai ir uždaviniai. Darbo eigoje paaiškintas darbo mokslinis naujumas, įvardyti ginamieji teiginiai ir pateiktas darbą aprobuojančių darbų sąrašas. Antroje dalyje (2 skyrius) aptariami verslo veiklos programų sąveikumo sprendimai, pateikiama šių sprendimų architektūra, metodai ir įrankiai. Aptariami apibrėžimai, paaiškinamos integracijos ir sąveikumo sąvokos. Apžvelgiama literatūra, aktuali kuriant sąveikumo sprendimus, ir sąveikumo įgyvendinimo galimybės (angl. *Capability*) vertinimo tarp skirtingų veiklos taikomųjų programų dinaminėje verslo aplinkoje. Trečioje dalyje (3 skyrius) aprašomi organizacijų taikomųjų programų sąveikumo metodai. Trečiame skyriuje aprašomi taikomųjų programų sąveikumo matavimai, sprendimai. Pabrėžiamas MDA ir gilių žinių naudojimo poreikis. Ketvirtoje dalyje (4 skyrius) aprašoma eksperimento aplinka. Pateikiamas eksperimentui atlikti naudojamų programinės įrangos sprendimų sąrašas. Aprašoma eksperimento veikla, pagrindimas ir numanomi rezultatai. Penktoje dalyje (5 skyrius) aptariami sąveikumo galimybių vertinimo eksperimento rezultatai. Aptariami skirtingi sąveikumo galimybių vertinimo būdai: redagavimo nuotolio analizės rezultatai, žodžių maišų analizės rezultatai ir latentinės semantinės analizės rezultatai. Eksperimento dalyje aprašomos analizuotos verslo programos, atliekama jų tarpusavio sąveikumo vertinimo analizė. Išanalizuota 13 programų. Šeštoje dalyje (6 skyrius) pateikiamos išvados ir rekomendacijos, apibendrinančios disertacijos darbą ir gautus rezultatus.

Disertaciją sudaro 6 skyriai, 104 puslapiai, pateikti 28 paveikslai ir 8 lentelės, 158.745 spaudos ženklų su tarpais – 3,97 autoriniai lankai. Disertacijoje remtasi 83 šaltiniais.

2. VERSLO VEIKLOS PROGRAMŲ SĄVEIKUMO SPRENDIMŲ APŽVALGA

Dinaminėje verslo aplinkoje, vykdant veiklos procesus ir apdorojant duomenis skirtingose sistemose, susiduriama su daug problemų. Dėl veiklos taikomųjų programų duomenų homonogeniškumo procesai dvigubinami, našumas mažėja, kyla duomenų valdymo ir saugumo problemų. Jau daugiau kaip 20 metų akademinėje literatūroje nagrinėjamos sąveikumo ir integracijos problemos, susijusios su veiklos taikomosiomis programomis ir sistemomis: veiklos resursų planavimo (ERP), santykių su klientais valdymo (CRM), apskaitos, sandėliavimo, žmogiškųjų išteklių (HR), e. verslo ir kitų panašaus pobūdžio (verslo veikloje naudojamų) taikomųjų programų. Žinoma, bet kuri veikla gali naudoti daugiau kaip vieną sistemą pagal savo poreikius, pavyzdžiui, Brinkerio Scotto iš „Chiefmartec“ analizė (Scott, 2017) atskleidžia, kad įprasta veikla naudoja vidutiniškai 43 CRM, 90 HR ir daugiau kitų Debesijos pagrindu veikiančių taikomųjų programų, tačiau nė neužsimenama, kiek dar ne Debesijos pagrindu veikiančių taikomųjų programų gali būti naudojama veikloje. Taip pat verta paminėti veiklos procesų analizę, nuo jos priklauso anksčiau išvardytų taikomųjų programų valdymo efektyvumas.

Apskritai taikomųjų programų integracija turi leisti sistemoms mainytis duomenimis arba fiziniais resursais nuosekliai siekiant bendro tikslo. Pagrindinis skirtumas tarp taikomųjų programų integracijos ir sąveikumo nėra tiksliai apibrėžtas, tačiau žinoma, kad integracija ir sąveikumas skiriasi. Integracijos atveju sprendžiamos visos srities problemos, o sąveikumo atveju – duomenų ir funkcijų mainų tarp kelių tos pačios verslo srities programų problema (Chen, et al., 2008). Taikomųjų programų integravimas gali būti: fizinis (pavyzdžiui, fizinis sujungimas kabeliais – duomenys tampa pasiekiami per tinklą), duomenų bazių integravimas į vieną, taikomųjų programų ir jų komponentų integravimas į vieną bendrą sistemą bei atskirų organizacijos veiklų jungimas į vieną bendrą. Sąveikumas, kita vertus, apibrėžia, kaip viena sistema gali gauti ir naudoti kitos sistemos duomenis ir funkcijas. Kitaip tariant, veiklos sistemos negali sąveikauti, jei nėra užtikrinta bent žemiausio (fizinio lygio) integracija. Apibendrinant, integruotos veiklos sistemos visada sąveikauja ir negali egzistuoti atskirai, tačiau sąveikaujančios sistemos nebūtinai turi būti integruotos ir gali egzistuoti nepriklausomai viena nuo kitos.

Šioje disertacijoje keliamas klausimas, kaip galima automatizuoti taikomųjų programų sąveikumą. Billas Gatesas yra pasakęs, kad norint kažką pagerinti svarbiausi yra matavimai (Gates, 2013). Siekiant sužinoti, ar galima

sukurti autonominį taikomųjų programų sąveikumo sprendimą, analizuoti reikia sąveikumo vertinimo matavimus, tiksliau, išmatuoti sąveikumo galimybes. Sąveikumą tarp veiklos procesų ir funkcijų taip pat vertino ir lietuvių autoriai. Gediminas Gricius 2014 metais (Gricius, 2015) aprašė ir suprojektavo daugiaagentinę sistemą nedidelio našumo įterptinėms sistemoms integruoti, kuri iš esmės rodo būtinybę kurti integracijos ir sąveikumo tarp taikomųjų programų metodus. Aurelijus Morkevičius 2013 metų disertacijoje (Morkevičius, 2013) nagrinėjo, kaip susijusios organizacijos veikla ir informacinės sistemos, ir kėlė klausimą, ar galima jas suderinti. Loreta Savulionienė 2014 metų disertacijoje (Savulionienė, 2014) aprašė ir pateikė duomenų susietumo vertinimo pagrindus naudojant dažnų posekių metodologijas. Dalia Dzemydienė ir Ramutė Naujikienė 2009 m. straipsnyje vertino elektroninių viešųjų paslaugų naudojimo ir informacinių taikomųjų programų sąveikumą (Dzemydienė & Naujikienė, 2009), aprašė sąveikumo problemas valstybiniame sektoriuje ir priežastis, dėl kurių šioje srityje atsiranda sąveikumo problemų. Visi išvardyti lietuvių autoriai iš dalies susiduria su sąveikumo problemomis tam tikrose srityse, bet nesprendžia sąveikumo tarp taikomųjų programų dinaminėje verslo aplinkoje problemų, taikomųjų programų architektūros įtakos sąveikumo procesams.

Apie veiklos procesų modelių panašumo identifikavimą rašė R. Dijkmanas ir kt. 2014 metais (Dijkman, et al., 2014), straipsnyje įvardijo daug galimų metodų, leidžiančių įvertinti verslo procesų panašumą skirtingais lygmenimis nuo sintaksinio iki semantinio. Jų darbe išaiškinti veiklos procesų pagrindai, kurie pritaikyti ir šioje disertacijoje. Kaip ir veiklos procesų modelius, taip ir taikomųjų programų architektūros panašumą galima vertinti pagal R. Dijkmano aprašytus žingsnius – sudėtingumo lygius vertinant (Dijkman, et al., 2014):

- Atributų tekstinį panašumą – naudojant teksto redagavimo nuotolio algoritmus (angl. *Edit Distance*).
- Struktūrinį panašumą – naudojant grafų redagavimo nuotolio skaičiavimus (angl. *Graph Edit Distance*).
- Elgsenos panašumą – naudojant iš funkcijų išvestus priežastinių ryšių pėdsakus (angl. *Causal Footprints*).

Šiame tyrime tirtas metaduomenų rinkimas ir analizė naudojant įvairiapusę informaciją, kurią galima gauti iš veiklos ir naudojamų taikomųjų programų. Literatūros apžvalgoje naudoti raktažodžiai: sąveikumas (angl. *Interoperability*); taikomųjų programų integracija (angl. *Enterprise Application Integration*); autonominiai skaičiavimai (angl. *Autonomic Computing*); kibersocialinės sistemos (angl. *Cyber-Social Systems*). Siekiant

parodyti temos aktualumą atsižvelgta į straipsnių šiomis temomis kiekį Google Scholar paieškos platformoje. Veiklos taikomųjų programų tinklo paslaugos užtikrina objektiškai orientuotą ir skaitomą plotmę perduoti informaciją žinomais informacijos formatais JSON, XML. Sąveikumo sprendimo būdai.

Tyrimo metu siekta apžvelgti kuo daugiau technologijų ir metodų, skirtų taikomosioms programoms automatizuoti ir veiklos duomenų analizei, todėl tyrimo struktūra atitinka šias temas:

- Integracija. Kaip taikoma? Kokie naujausi sprendimai?
- Sąveikumas. Kaip taikomas? Kokie naujausi sprendimai?
- Priežastinių ryšių analizė sistemose. Kas tai? Kodėl tai svarbu? Ką duos integracijos automatizavimui?
- Kibersocialinės sistemos. Kas tai? Kodėl tai susiję su taikomųjų programų integracija ir sąveikumu? Kaip galima panaudoti?
- Autonominės skaičiavimo sistemos. Kas tai? Kokie naujausi sprendimai? Kaip tai galima panaudoti?

Šioje disertacijoje nagrinėjama verslo veiklos programų integracija ir sąveikumas per SOAP arba REST protokolus (kiekvienai sistemai skirtingas). Šis apribojimas pasirinktas todėl, kad ne visos sistemos yra atvirojo kodo, turinčios galimą prieigą prie duomenų šaltinių.

2.1. Apibrėžimai

Nagrinėjama dinaminės organizacinės veiklos taikomųjų programų integracijos ir sąveikumo tema. Būtina apibrėžti, kad veikla – toliau E (angl. *Enterprise*). Pagal informacijos taikomųjų programų inžinerijos teorijos pagrindus kiekviena verslo veikla turi funkcijų (f), procesų (p) ir tikslų (g) rinkinį (Gudas, 2012). Be to, veikla turi funkcijų ir procesų sąveikų rinkinį (r) ir informacinius elementus (k), kurie reikalingi šioms sąveikoms realizuoti. Kitaip tariant, organizacinę veiklą galima išreikšti formalizuotai (teiginių logikos notacija) (Krajicek & Krajíček, 1995):

(1) $$E \ni \{g, f, p, r, k\}.$$

Verslo veiklos modelius kuria verslo analitikai. Verslo veiklos modelis (M) dažniausiai pateikia informaciją apie verslo tikslus, funkcijas, procesus, jų ryšius ir žinias, todėl nuo modelio M kokybės priklauso, kaip tiksliai atvaizduota realaus pasaulio (E) verslo veikla, t. y. diegiamas verslo veiklos modelis ir stebima, ar verslo veiklos procesai nestringa. Kitaip tariant, modelis yra veiklos dalis, kuris nurodo, kaip verslo procesai turėtų būti vykdomi, bet nebūtinai jie vykdomi pagal modelį. Nors standartiškai nėra įtraukiama į

aprašymą, šiame tyrime laikoma, kad veikla gali turėti modelį (M), taip pat veikla naudoja vieną arba kelias sistemas (S) veiklos valdymui užtikrinti. Tad veikla turi bent vieną M ir bent vieną S:

(2) $$E \ni \{M, S\}.$$

Todėl konkrečios veiklos modelis yra veiklos tikslų, funkcijų, procesų, ryšių ir informacinių elementų reprezentacija:

(3) $$M_E \ni \{g, f, p, r, k\}.$$

Šioje disertacijoje keliama prielaida, kad verslo veikla E yra dinamiška, jos f ir p gali keistis, E gali turėti vieną ir daugiau taikomųjų programų S, kurios taip pat gali kisti priklausomai nuo veiklos tikslų ir aplinkos veiksnių.

(4) $$S_E \ni \{S_1, S_2, S_3, \dots, S_n\}, n = \mathbb{N}.$$

Sistema gali būti sukurta Debesijos pagrindu arba diegiama vietoje kaip savarankiška sistema (angl. *Standalone on Site Application, Legacy Application*). Tyrimui neturi įtakos, kokiu pagrindu sukurta sistema, svarbu, kad ji būtų sukurta paslaugomis grindžiama architektūra ir visos arba dalis esinių būtų paslaugos kaip operacijos, galinčios keistis duomenimis ir pranešimais. Kiekviena sistema S_n gali kisti arba viena sistema gali pakeisti kitą (jeigu tik tenkina veiklos tikslus) veiklos gyvavimo periode. Taikomųjų programų integracijos ir sąveikumo galimybių analizei reikia žinoti, kokie yra sistemos objektai (arba esiniai) ir kokie ryšiai yra tarp šių objektų skirtingose sistemose:

(5) $$S_n \ni \{O_1, O_2, \dots, O_i\},$$

(6) $$r \ni \{r_1\langle O_{ni}, O_{n+1,i}\rangle, r_2\langle O_{ni}, O_{n+1,i}\rangle, \dots, r_j\langle O_{ni}, O_{n+1,i}\rangle\}.$$

Kiekvienam sistemos S objektui O galima rasti ryšį su kitos sistemos S objektu O. Kiekvienas objektas aprašomas duomenų struktūra (angl. *Schema*) – (h):

(7) $$O \ni h.$$

Kiekvienas objektas O turi duomenų struktūros aprašą h, kuris apibūdina objekto sandarą ir metaduomenis, šie susideda iš: duomenų laukų pavadinimų, duomenų laukų tipo, duomenų laukų ilgių ir apribojimų.

Šie apibrėžimai padės paaiškinti teoriją, susijusią su taikomųjų programų integracija ir sąveikumu.

Programų sudėtingumo lygiai išanalizuoti ir suklasifikuoti lentelėje, pateikiamoje 2 priede (2 priedas).

2.2. Taikomųjų programų integracija

Informatikos srityje taikomųjų programų integracijos tema yra gana sena, kilo iš poreikio atpažinti, perimti ir į bendrą sistemą sujungti skirtingas radijo ryšių technologijas. Amerikos kosmoso organizacijai NASA ši tema taip pat buvo aktuali, nes ryšių kontrolės centras turėjo palaikyti stabilų ryšį su dirbtiniais žemės palydovais esant dideliems trikdžiams ar trūkiams. Sistemos turėjo būti integruotos – jų komponentai dirbti kartu sklandžiai siekiant užtikrinti sklandų informacijos perdavimą. Šiuo metu taikomųjų programų integracija taip pat aktuali verslo srityje ir siejasi su verslo tikslais ir siekiais optimizuoti veiklos procesus. Google Scholar duomenimis, nuo 2014 m. publikuota 97 tūkst. straipsnių, susijusių su taikomųjų programų integracijos tema (angl. *Enterprise Application Integration*), nors dauguma iš jų skirti specifinėms sritims analizuoti. Sistemos darosi sudėtingos, jos ne visada kuriamos be standartų, o šių yra daug, ypač debesų pagrindu veikiančių taikomųjų programų. Todėl reikia skirti gana daug dėmesio integracijos architektūrai norint, kad sistemos sąveikautų.

Taikomųjų programų integracija turi keletą metodų:

- Vertikalioji integracija – viena sistema apima visus veiklos procesus;
- Horizontalioji integracija – integracija tarp taikomųjų programų, veiklų, skyrių, organizacijų, kurių tikslai skirtingi;
- Žvaigždinė integracija – decentralizuoti integracijos sprendimai vienam konkrečiam tikslui pasiekti.

Taikomųjų programų integracijos sprendimai yra vystomi įvairiomis kryptimis, tiek verslo, tiek mokslo srityse. Vystyti taikomųjų programų integracijos sprendimus naudinga keliomis prasmėmis: standartizuoti integracijos sprendimų kūrimo procesus (Chen, et al., 2008); greičiau apdoroti taikomųjų programų informaciją; atlikti sudėtingas manipuliacijas su dideliais duomenimis atpažįstant esminius duomenis, jų tarpusavio ryšius; sumažinti nesėkmingų projektų skaičių (Trotta, 2003); sumažinti žinių poreikį, reikalingą sėkmingiems taikomųjų programų integracijos projektams atlikti; didinti taikomųjų programų lankstumą; mažinti veiklos procesų kaštus (Panetto & Molina, 2008; Reijers & Mansar, 2005). Apibendrintai programų integraciją galima aprašyti šia išraiška:

(8) $$\{S_1, S_2, \dots, S_n\} \ni S' .$$

Sistemos S_1 ... S_n yra integruotos sistemos S' poaibiai – t. y. S' sistemos dalis. Atskiros sistemos S_1 ... S_n neturi autonomijos ir priklauso nuo S' funkcionalumo ir integracijos sprendimo kokybės.

2.2.1. Vertikalioji integracija

Vertikalioji integracija – integracijos metodas, užtikrinantis, kad viena sistema gali įvykdyti visus veiklos proceso žingsnius (Halevy, et al., 2005), nuo žaliavos gavimo iki galutinio produkto sukūrimo, išskirtinai tik savo resursais. Šiuo integracijos metodu integruojamos posistemės, turinčios atskirus tikslus, tačiau esančios bendros sistemos dalis. Geras veiklos sistemų su vertikalia integracija pavyzdys – veiklos išteklių planavimo sistemos (angl. *Enterprise Resource Planning, ERP*). ERP taikomųjų programų architektūra kuriama iš karto su vertikaliosios integracijos idėja: gaunami ir dokumentuojami ištekliai, žaliavos – tai atlieka sandėlio išteklių valdymo posistemis; gamybos valdymo posistemyje užtikrinamas laiko ir gamybos išteklių planavimas; apskaitos posistemės užtikrina skaidrią buhalterinę veiklą; klientų išteklių valdymo posistemės užtikrina klientų pirkėjų srautą. Tokia sistema vertikaliai integruota verslo veikloje, nes kuriamo produkto kelias yra visiškai apibūdinamas ERP sistemos. Kitaip tariant, veiklos valdymo grandinė ERP sistemos pagrindu dažniausiai beveik visai padengiama.

Vertikaliosios integracijos atveju egzistuoja tokia sistema arba sprendimas su objektais, kurie visiškai aprėpia visus veiklos E svarbius procesus p_n.

(9) $$\exists\, S_i \rightarrow \forall\{p_1, p_2, \dots, p_n\} \in E.$$

Šio integracijos metodo privalumai tie, kad sprendimą galima greitai sukurti, žinant konkrečią reikiamą architektūrą, t. y. turint visą veiklos E modelį M. Vertikaliosios integracijos trūkumas tas, kad tokiu integracijos metodu įgyvendintos posistemės retai gali būti pritaikomos atskirai. Jei jos gali būti pritaikomos atskirai, t. y. gali veikti atskirai nuo sistemos, jos yra horizontaliosios integracijos metodo dalis. Dinaminėje verslo aplinkoje dažnai atsiranda pokyčių, kurių sistemos laikui bėgant pradeda nepalaikyti. Reikia kurti, atnaujinti programos posistemių sprendimus, o naujus sprendimus vėl integruoti į visą sistemą.

2.2.2. Horizontalioji integracija

Horizontalioji integracija – taikomųjų programų integracijos metodas, kuriame sistemos posistemės turi savotišką autonomiją. Pavyzdys iš verslo srities – atskira organizacija, teikianti buhalterijos paslaugas. Kita organizacija perka šią buhalterijos paslaugą, sudariusi bendradarbiavimo sutartį keičiasi duomenimis naudodama tam tikras veiklos sistemos sąsajas ir buhalterinės sistemos sąsajas – verslo organizacijos integruojamos per paslaugų sistemas. Tokio tipo integracija dar vadinama verslas-verslui (B2B)

integracija. Horizontalioji integracija taip pat gali būti ir vidinėje verslo veikloje esant griežtesnei arba labiau padalytai verslo veiklos struktūrai. Integracija tarp padalinių naudojamų atskirų taikomųjų programų yra horizontali dėl to, kad padaliniai turi tam tikrą autonomiją ir iš dalies gali dirbti be kito padalinio įsikišimo. Dažnai tokio integracijos metodo taikymas turi būti palaikomas koordinuojančio posistemio, arba verslo prasme – skyriaus. Pavyzdžiui, verslo atžvilgiu IT skyrius siekia užtikrinti, kad kitos posistemės veiktų sklandžiai ir mainytųsi informacija. Laikant, kad veikla yra išskaidyta į autonominius padalinius, tokios veiklos apibrėžimas būtų toks:

(10) $$E \ni \{E_1, E_2, \dots E_n\}, n = \mathbb{N},$$

(11) $$\exists \{S_1, S_2, \dots, S_i\} \to \forall \{p_1, p_2, \dots, p_n\} \in \forall \{E_1, E_2, \dots E_n\}.$$

Egzistuoja tokių sistemų, kurių integracija apima visus veiklos procesus per visas veiklos pogrupius (skyrius ir poskyrius).

2.2.3. Žvaigždinė integracija

Žvaigždinė integracija – taikomųjų programų integracijos metodas, kuriame sistemos posistemių integracija yra decentralizuota. Šiuo atveju nėra IT skyriaus ar sistemos posistemės, prižiūrinčios ir valdančios integracijos procesus. Žvaigždinės integracijos metodu kiekviena posistemė integruojama tik su jos veiklos procesams naudinga posisteme. Pavyzdžiui, E-Komercijos posistemė jungtųsi su sandėlio valdymo posisteme, kad gautų prekių likučių duomenis. Tačiau sandėlio valdymo posistemei neaktualu, kokia informacija yra naudojama E-komercijos posistemėje.

(12) $$\exists S_n \to \neg \forall \{S_1, S_2, \dots, S_k\} \to \neg \forall \{p_1, p_2, \dots, p_i\} \in E, n \neq k.$$

Egzistuoja tokia sistema, kuri integruota su kai kuriomis kitomis sistemomis, aprėpiančiomis ne visus veiklos procesus.

Žvaigždinės integracijos privalumai – taupomi resursai posistemių architektūrai. Šios integracijos metodas leidžia atsiriboti ir kurti architektūrinius integracijos sprendimus tik toms posistemėms, kurios yra šiuo metu aktualios.

Žvaigždinės integracijos metodui reikalinga posistemė nebūtinai gali būti pasiekiama dėl tinklo paslaugų problemų, prieigos prie duomenų šaltinio trūkumų ar leidimų apribojimų. Tokią integraciją sunku valdyti ir stebėti.

2.3. Sąveikumas

Sąveikumas (angl. *Interoperability*) – veiklos taikomųjų programų ir jų posistemių galimybė efektyviai mainytis duomenis ir informacija, naudoti

viena kitos funkcionalumus savo reikmėms. Sąveikumas gali būti įgyvendinamas keliais lygiais: tarp vienos sistemos komponentų arba tarp kelių skirtingų taikomųjų programų komponentų. Projektuojant sudėtingas taikomųjų programų sistemas, būtina žinoti, ar sistemos gali sąveikauti, reikia žinoti, kaip nustatyti ar įvertinti sąveikumo galimybę.

Šiame darbe analizuojama verslo veiklos taikomųjų programų sąveikumo galimybės įvertinimo problema. Analizuojamas verslo veiklos tinklo paslaugų (angl. *Web Service*) struktūrinis panašumas, susidedantis iš: operacijų, objektų ir duomenų struktūrų panašumų. Šie panašumai leidžia įvertinti sąveikumo tarp verslo veiklos programų galimybę. Peržvelgus nagrinėjamą literatūrą identifikuojami sąveikumo srities tipai, barjerai, taikomųjų programų panašumų metrikos.

Sąveikumas yra skirstomas į tipus (Chen, et al., 2008):

- Sintaksinis (angl. *Syntactic*) – sąveikumas naudojant bendrus duomenų formatus ir protokolus.
- Semantinis (angl. *Semantic*) – gebėjimas interpretuoti informaciją, kuria keičiamasi.
- Tarp-sritinis (angl. *Sross Domain*) – keli sociopolitiniai esiniai, kitaip tariant, verslo partneriai, dirbantys kartu siekiant geresnio rezultato. Tai verslas-verslui (B2B) sąveikumo įgyvendinimo tipas.

Sąveikumą galima vertinti panašumo rodikliu. Sintaksinis sąveikumas galimas, jei sintaksinis panašumas tarp sistemos objektų ir tipų viršija panašumo θ slenkstį (angl. *Threshold*), kuris yra konstanta. Sąveikumo vertinimo sprendimų peržiūrą atliko Rezaei ir kt. (Rezaei, et al., 2014), apžvelgti naujausi darbai iki 2014 metų. Šiame straipsnyje pabrėžiami sąveikumo sudėtingumo lygmenys ir lyginami karkasai, pateikiantys sąveikumo sprendimo kūrimo priemones bei rekomendacijas. Dauguma apžvelgtų darbų nevertina taikomųjų programų sąveikumo galimybės, išskyrus LISI metodą, reikalaujantį daug ekspertinių žinių (Chen, et al., 2008; Kasunic, 2001).

2.3.1. Sąveikumo barjerai

Sąveikumo problemų kyla dėl priežasčių, dar vadinamų sąveikumo barjerais (Chen, et al., 2008). Sąveikumo barjerai yra trijų kategorijų (Chen, 2006): koncepciniai, technologiniai, organizaciniai ir teisiniai.

Koncepciniai sąveikumo barjerai – tai sintaksiniai ir semantiniai informacijos skirtumai tarp taikomųjų programų. Šis barjeras apriboja įvairaus

lygmens sąveikumo architektūrą nuo aukšto – veiklos procesų modeliavimo, lygmens iki žemo duomenų struktūrų architektūros lygmens.

Technologiniai barjerai sprendžia fizikines technikos sąveikumo problemas, tokias kaip duomenų perdavimo signalai ir perduodanti platformų architektūra. Taip pat technologinis barjeras apriboja galimybes pateikti, talpinti, keistis duomenimis ir informacija tarp programų.

Organizaciniai barjerai sprendžia teisinius, reguliacinius sąveikumo aspektus. Jų tikslas užtikrinti, kad sąveikumas nepažeidžia įstatymų ir yra nepažeidžiamas piktavališkai ar kitaip nepaveikiamas žmogiškųjų klaidų.

Teisiniai – reikia užtikrinti, kad duomenys nebūtų naudojami blogiems tikslams arba nutekinti vykdant sąveikumo operacijas. Taip pat galima įtraukti naują BDAR (Bendrojo duomenų apsaugos reglamento) įstatymą, reguliuojantį, kaip turi būti tvarkomi asmeniniai duomenys.

Remiantis ISO/IEC 2382 standartu – sąveikumas apibrėžiamas kaip nutolusių komponentų duomenų apdorojimas, kitaip tariant, tai galimybė dviem ar daugiau funkciniams komponentams bendradarbiauti mainantis duomenimis, funkcijomis. Naudotojas šiame bendradarbiavimo procese turi mažai arba visai neturi žinių apie šių komponentų sandarą ir duomenų perdavimo būdus. Visos sąveikaujančios programos turi būti sukurtos pagal paslaugomis grindžiamos architektūros (SOA) sprendimus. Pagrindinis principas – pagal SOA sukurti tokią taikomąją programą, kad ji naudotojui būtų kaip juodoji dėžė, bet taip pat aprašyti įvesties ir išvesties parametrus, kad kitos sąveikaujančios taikomosios programos galėtų pasiekti pirmosios duomenis ir funkcijas (Krafzig, et al., 2005). Programos naudotojui nereikia žinoti programos paslaugos įvesties ir išvesties kintamųjų, kai sąveikumo problemos yra išspręstos sąveikumo sprendimo architekto, sukuriančio tarpinę programą (angl. *Middleware*) užtikrinti bendradarbiavimui (sąveikavimui) tarp dviejų ar daugiau verslo programų.

Naujame Europos Sąveikumo Karkase (ISA^2, 2017) sąveikumo sluoksnio rekomendacijos iš naujo apibrėžtos ir gali būti naudojamos kaupiant žinias apie veiklos programų sąveikumo galimybes. Europos Sąveikumo Karkase (EIF) apibrėžti sąveikumo sluoksniai: techninis, semantinis, organizacinis, teisinis, integruotų viešųjų paslaugų valdymo (angl. *Integrated Public Service Governance*). Šioje disertacijoje nagrinėjami semantiniai ir techniniai sąveikumo sluoksniai atsižvelgiant dar į tokius teisinius aspektus:

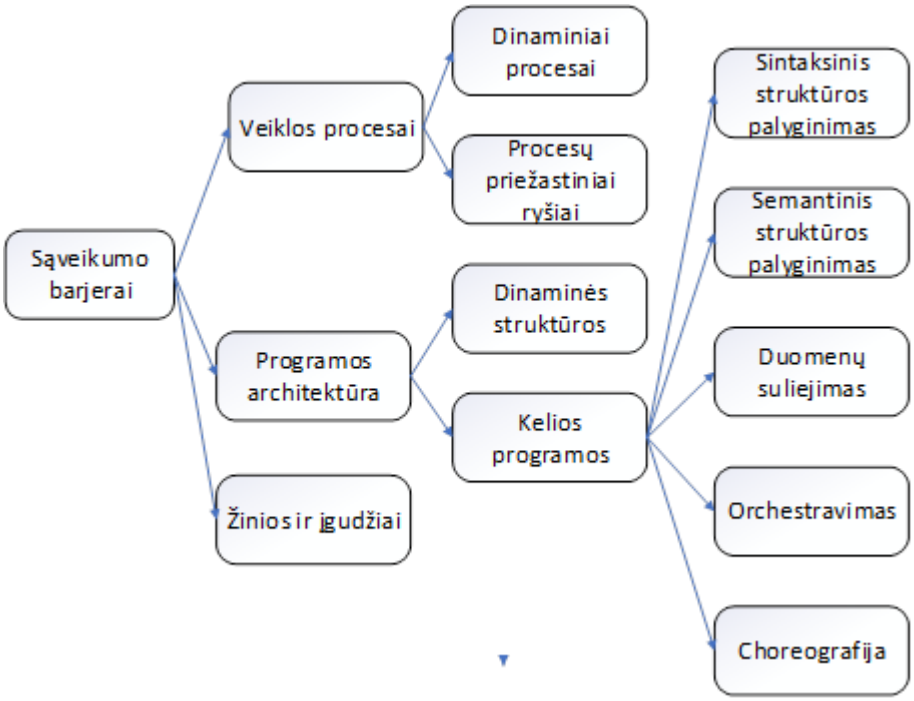
- Tinklo paslaugų taisyklių rinkiniai leidžia patikrinti pritaikymą fiziniame ir skaitmeniniame pasaulyje;
- Tinklo paslaugų taisyklės leidžia identifikuoti barjerus skaitmenizuotų duomenų mainų procesuose, pavyzdžiui:

duomenų apsikeitimas, apsikeitimo dažnis, duomenų mainų limitas per dieną, paslaugos komponento arba funkcijos aktyvavimas per dieną ir kiti.

Iš teisinės perspektyvos šioje disertacijoje nėra galimybės identifikuoti ir įvertinti informacijos komunikacijos technologijos (angl. *Information Communication Technology*, ICT) įtakos suinteresuotosioms šalims. Organizacinis sąveikumo sluoksnis iš EIF iš dalies apžvelgtas ir panaudotas veiklos procesų modelio analizei remiantis veiklos architektūra (angl. *Enterprise Architecture*) ir modeliais grindžiama architektūra (angl. *Model Driven Architecture, MDA*). Komponento sąveikumo žinių modelis gali būti formuojamas remiantis veiklos procesų diagrama ir padėtų sprendžiant technines bei semantines sąveikumo problemas. Remiantis naujuoju EIF karkasu – semantinis sąveikumas apima semantinį ir sintaksinį aspektus: semantiniui aspektui svarbi duomenų elementų ir jų ryšių reikšmė ir aprašymas, o sintaksinis aspektas aprašo duomenų elementų formatavimą ir apribojimus (ilgį, pasikartojimų skaičių). Techniniame EIF lygmenyje remiamasi prielaida, kad dauguma veiklos programų yra senos arba skirtos tik konkretiems veiklos procesams, bet nepadengia visų procesų organizacijoje. Tokiose atskirtose sistemose, nors ir turima bendrų duomenų, šios programos jomis nesidalija, tad negali būti vadinamos sąveikaujančiomis. Techniniam sąveikumo užtikrinimui reikalingi programos sąsajos aprašai (angl. *API reference*), apibrėžiantys įvesties ir išvesties kintamuosius. Iš čia – barjeras, jeigu programos neturi API aprašymų arba yra aprašytos nesilaikant standartų, tokių kaip, pavyzdžiui, SOAP, arba dažnai REST metodologijoje aprašymai yra laisvai struktūruoti ir jų nuskaitymo negalima automatizuoti. Pagrindiniai sąveikumo užtikrinimo barjerai yra:

- Veiklos procesai kinta, kai naujų programų įdiegiama versle.
- Programos yra dinamiškos ir jų struktūra gali kisti laikui bėgant.
- Kelios programos naudojamos toje pačioje srityje, pavyzdžiui, vietoje vienos ERP sistemos, padengiančios visus procesus, naudojama CRM programa klientams valdyti, atskira e. komercijos programa elektroninei parduotuvei valdyti ir atskira programa tvarkyti buhalterijai ir vesti apskaitai.
- Nėra bendrai paplitusių metodų aprašyti sąveikoms tarp skirtingų programų.
- Programos pokyčiai visada veikia veiklos procesus, todėl ankstesni veiklos procesų modeliai nustoja galioti ir negali būti naudojami žinioms apie priežastinius ryšius gauti.

- Siekiant užtikrinti programų sąveikumą, integracijos ekspertas turi atlikti šias užduotis:
    - Atlikti schemų lygiavimą (Ford, et al., 2008; Rahm & Bernstein, 2001; McCann, et al., 2005; Peukert & Eberius Julian, 2011; Silverston, 2011);
    - Užtikrinti įrašų susiejimą ir duomenų sujungimą (Kutsche & Milanovic, 2008; Dong & Divesh, 2013);
    - Užtikrinti procesų orchestravimą – laiką tarp kiekvieno duomenų mainų proceso įvykdymo;
    - Užtikrinti choreografiją – kiekvieno duomenų mainų proceso eilės tvarką.
- Giluminių žinių apie programas, veiklos procesus trūkumas, taip pat integravimo įgūdžių trūkumas.



Pav. 2. Sąveikumo barjerai.

Programų sąveikumo įgyvendinti dėl trūkstamų žinių ir įgūdžių barjero negalima renkant žinias apie pavienes programas, nes kitose programose jos gali nebegalioti – tai iteratyvus procesas, todėl ir apibrėžiamas kaip barjeras. Senesni sąveikumo barjerai EIF dokumentacijoje (IDABC, 2004) dabar laikytini sluoksniais (ISA2, 2017). Duomenys iš vienos programos negali sąveikauti su panašiais duomenimis kitoje programoje, jei visi barjerai nėra pašalinti (Chen, et al., 2008).

2.3.2. Sąveikumo interesai

Sąveikumo interesai keliami skirtingiems veiklos lygmenims, todėl gali būti traktuojami skirtingai ir jų architektūra dėl šių interesų taip pat gali skirtis (Chen, 2006).

- Duomenų sąveikumas – hierarchinių, reliacinių duomenų modelių sąveikumas. Mainymasis tapačiais – heterogeniniais, duomenimis, kurie gali būti skirtinguose kompiuteriuose, operacinėse sistemose, duomenų bazių valdymo sistemose ir skirtingose programinėse sistemose.
- Paslaugų sąveikumas – tai skirtingų taikomųjų programų veiklos, jų paslaugų sąveikumas (būtina sąlyga, kad jos sukurtos ir įgyvendintos nepriklausomai viena nuo kitos). Sprendžia sintaksines ir semantines problemas, taip pat ieško būdų gauti duomenų iš duomenų šaltinių – taikomųjų programų duomenų bazių ir paslaugų.
- Procesų sąveikumas – veiklos procesų sąveikumo užtikrinimas, kai šie padalyti skirtingiems skyriams ar veiklos grupėms. Veiklos procesai taip pat gali būti optimizuojami sąveikumo užtikrinimu, kai pašalinami panašūs pasikartojantys procesai, atsakingi už bendrą tikslą, bet veikiantys atskirai. Taip užtikrinama procesų optimizacija, kai bendrą tikslą atliekantys procesai skirtinguose padaliniuose sujungiami ir taip sutaupoma laiko ar kitų išteklių.
- Verslo sąveikumas – tai kompanijų dalijimasis ištekliais ir veikla taupant resursus. Toks sąveikumas apima sprendimų priėmimą, darbo metodus, įstatymų leidimus, kompanijos kultūrą, finansinius sprendimus. Galima būtų įvardyti, kad tai B2B (angl. *Business to Business*) sąveikumo tipas, kuriuo siekiama sukurti draugišką bendradarbiavimo tarp įmonių aplinką.

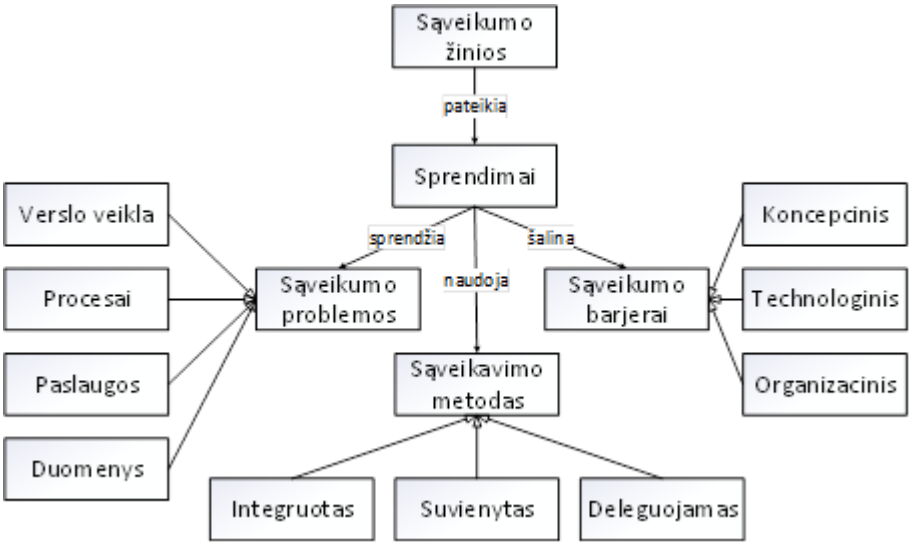
2.3.3. Sąveikumo sprendimai

Pagal ISO14258 integracijos sprendimai gali būti (ISO14258, Industrial automated systems, 1999):

- Integracijos – kai egzistuoja bendras formatas visiems modeliams.
- Unifikacijos – kai egzistuoja bendras duomenų metaformatas, toks duomenų metamodelis negali egzistuoti atskirai kaip esinys, tačiau gali pateikti naudingos semantinės reikšmės informacijos, leidžiančios rasti ryšius tarp skirtingų taikomųjų programų ir jų

modelių. Geras pavyzdys – paslaugos dokumentacija „WSDL“ ar kitu formatu, iš kurio tiksliai nėra žinoma, kokia konkreti duomenų struktūra slepiasi po paslaugomis, tačiau kuri leidžia nusakyti, kaip šią paslaugą galima išnaudoti sąveikumo tikslais.

- Federacijos – nėra bendro formato, todėl siekiant sukurti sąveikumą reikia surinkti ir išanalizuoti panašias savybes kiekvienos sąveikos metu. Šio sprendimo nėra aprašyta bendro modelio, kalbų ar darbo metodų.



Pav. 3. Baziniai veiklos sąveikumo konceptai ir jų tarpusavio ryšiai, adaptuota (Chen, et al., 2008).

Bazinių sąveikumo konceptų pavyzdyje matyti visų išvardytų sąveikumo barjerų, interesų ir sprendimų architektūra. Čia sąveikumo žinios leidžia sukurti sąveikumo projektą. Sąveikumo projektas veikia pagal interesus, šalina barjerus ir naudoja sprendimus.

2.4. Integracijos ir sąveikumo problemos

Literatūros apžvalgos metu, išanalizavus taikomųjų programų integracijos srityje atliktus tyrimus, įvardytos šios problemos, kylančios integruojant dvi arba daugiau skirtingų taikomųjų programų. Taip pat šių problemų randasi įvairiose organizacinėse veiklose (Panetto & Molina, 2008).

Skirtingos duomenų struktūros (angl. *Schema*) – gali skirtis pagal esinių ryšių diagramą duomenų bazėje, skirtingą SOA architektūrą (Rahm & Bernstein, 2001; Halevy, et al., 2005; Peukert & Eberius Julian, 2011).

Dažniausiai ši problema yra nagrinėjama integracijos sprendimus analizuojančiuose tyrimuose. Sunku atpažinti ir palyginti skirtingų programų duomenų pavadinimus, duomenų laukų arba stulpelių tipus ir ilgius. Ši problema yra viena iš sudėtingiausių siekiant užtikrinti sklandžius integracijos įgyvendinimo ir vėliau – sąveikumo procesus. Duomenų struktūros dinaminėje verslo aplinkoje gali ir kisti.

Skirtingi duomenų šaltiniai – skirtingos programų sistemos su skirtingomis duomenų bazėmis, skirtinga SOA architektūra, informacijos perdavimo šaltiniais, agentais ir jų tarpusavio komunikacijos protokolais. Išpopuliarėjus Debesijos technologijai daugelis sprendimų perkelti į nutolusius serverius, tačiau senesnių programinių paketų dalis galėjo likti įmonės serveriuose.

Dinaminės sistemos – nuolat kinta arba gali kisti programų struktūra, duomenų lentelių kiekis, esinių ryšių architektūra duomenų bazėje. Ši problema egzistuoja senesnėse įmonėse, kur norima seną programinę įrangą pakeisti nauja, migruoti duomenis į Debesijos serverius. Problema taip pat atsiranda, kai senoje programinėje įrangoje diegiami papildomi moduliai, atliekantys naują verslo veiklos funkciją.

Kelios sistemos – sprendžiami klausimai apie vienodų duomenų patikimumą, pirmumą integracijos procese. Jei yra kelios sistemos ir duomenys yra heterogeniniai, kyla priežastinių ryšių klausimų, pavyzdžiui: kuri sistema atlieka veiklos proceso funkcijas pirma, kuri antra? Ar veiklos proceso funkcijos persipina? Ar komponentai naudoja tuos pačius duomenis skirtinguose veiklos etapuose skirtingose sistemose? Dažnai šį klausimą reikia spręsti duomenų orchestravimo ir choreografijos metodais.

Skirtingi autentifikavimo procesai – susiję su skirtingais duomenų šaltiniais, nes kiekvienas duomenų šaltinis programos sistemoje gali turėti vartotojo prisijungimo ir autentifikavimo skirtumų, kuriuos reikia identifikuoti ir valdyti.

Regioniniai skirtumai – laiko juostos, valiutų skirtumai, kalbos koduočių skirtumai labai veikia atliekamų integracijos procesų kokybę, todėl reikalinga atsižvelgti į kiekvienos programų sistemos išankstines žinias, kuriant integraciją, o tai nėra įmanoma visada.

Semantinis esinių ir objektų atpažinimas – nagrinėjamas esinių ir objektų atpažinimas semantine prasme, integracijos procese bandoma susieti skirtingų taikomųjų programų esinius arba jų sudėtinius komponentus su kitos programos sistemos esiniais ir komponentais.

Orchestravimas ir choreografija – esinių ir objektų iš skirtingų taikomųjų programų integracijos eigos išdėstymas laike, laikantis numatytų apribojimų. Apribojimai išvedami pagal programų sistemos struktūrą arba

laikantis verslo eigos procesų, priklausomai nuo naudojamų taikomųjų programų ar verslo aplinkos.

Duomenų mėginiai – integracijos automatizavimo srities problema, kaip patikrinti duomenų teisingumą. Tikrinamas duomenų teisingumas, korektiškumas, formatavimas, susijęs su skirtingomis duomenų struktūromis, regionų skirtumais ir esiniais bei objektais.

Integracijos testavimas – nagrinėjama, kaip užtikrinti teisingą taikomųjų programų perdavimą iš vienos sistemos kitai, laiko trukmę, perduotų duomenų teisingą apdorojimą perduotajai programai (Villányi & Martinek, 2015).

CRUD (angl. *Create, Read, Update, Delete*) **taisyklių užtikrinimas**. Taikomųjų programų integracijos sritis, nagrinėjanti, kaip užtikrinti saugų ir kokybišką rašymą, skaitymą, naujinimą ir trynimą integruojamose programų sistemose laikantis tam tikrų nustatytų taisyklių.

Semantinė integracija – taikomųjų programų srityje stengiamasi išanalizuoti, kaip integracijos procesas galėtų suprasti savo aplinką, taip geriau reaguotų į trikdžius ir gebėtų kai kuriuos iš jų spręsti autonomiškai (Panetto & Molina, 2008; Halevy, et al., 2005).

Dokumentacijos trūkumas – taikomųjų programų integracijos srityje sukelia nemažai pirmiau išvardytų problemų, kurių negali išspręsti net ir patyręs integracijos projektuotojas, tyrėjai ieško automatizuotų būdų, kaip gauti ir pateikti daugiau informacijos apie integruojamas sistemas;

Integracija tarp verslo subjektų – tiriami saugumo sistemos, užtikrinančios saugų ir nešališką duomenų perdavimą, jų saugumą ir atsparumą vagystėms, protokolai. Ieškoma būdų automatizuoti identifikavimo, skaitmeninio parašo perdavimo technologijų, atsparumo duomenų vagystėms algoritmus.

Nestandartinių vienetų, kalbos skirtumų, perteklinių etikečių, formatavimo ir kitos (aut. past. daugiau neištirta) – paprastai integracijos projektuotojo sprendžiamos problemos, šioms ieškoma būdų, kaip surinkti integruotojo žinias naudojant integracijos sprendimus paieškai.

Netvarkingi duomenys – integracijos sistemos architekto sprendžiamos problemos, gilinamasi, kaip ištaisyti klaidas, kaip suprogramuoti automatinį jų aptikimą ir ištaisymą arba išvengimą. Sugalvoti, kokius metodus taikyti duomenims taisyti ir tvarkyti.

Duomenų perdavimo problemos – pavyzdžiui, naudojant SOAP protokolą duomenys įrašomi į XML formatą, kuris gerokai padidina perduodamų duomenų kiekį (Halevy, et al., 2005).

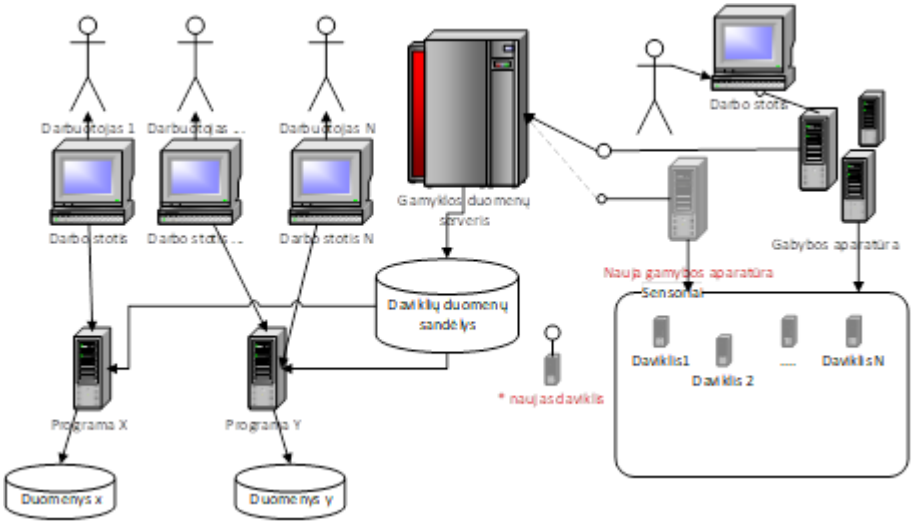
Integracijos ir sąveikumo problemų lyginimas pateikiamas tolesnėje lentelėje.

Lentelė 1. Integracijos ir sąveikumo problemų lentelė:

	Integracijos problemos	**Sąveikumo problemos**
1.	Gilių žinių ir vidinio modelio nebuvimas	Gilių žinių ir vidinio modelio nebuvimas
2	Skirtingos duomenų struktūros	-
3	Skirtingi duomenų šaltiniai	-
4	Dinaminės sistemos	Dinaminės sistemos
5	Kelios sistemos	Kelios sistemos
6	Skirtingi autentifikavimo procesai	-
7		Regioniniai skirtumai
8	Semantinis esinių ir objektų atpažinimas	-
9	Orkestravimas ir choreografija	Orkestravimas ir choreografija
10	Duomenų mėginiai	Duomenų mėginiai
11		Priežastiniai ryšiai
12	Integracijos testavimas	Integracijos testavimas
13		CRUD
14	Ontologijų integracija (Li, et al., 2005)	Ontologijų integracija
15	Dokumentacijos trūkumas	-
16	B2B integracija	-
17	Nestandartinių vienetų, kalbos skirtumų, perteklinių etikečių, formatavimo	-
18	Netvarkingi duomenys	Netvarkingi duomenys

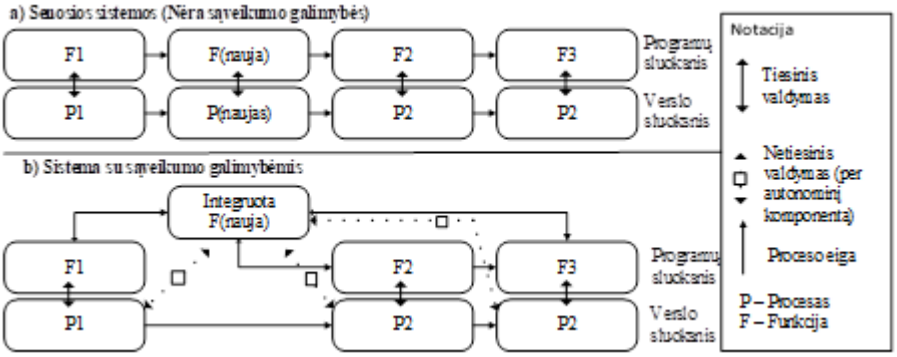
Sąveikumo problemą galima išgryninti pavyzdžiu. Imkime gamyklos veiklos scenarijų. Tarkime, kad gamykla naudoja sensorius savo veiklos procesuose siekdama užtikrinti kokybišką produkto gamybą, testavimą ir pakavimą. Kiekvienos gamyklos įrenginys turi vieną ar kelias sąsajas gauti duomenims. Šie duomenys gali būti surenkami sandėlio valdymo, gamybos planavimo, ryšių su klientais ir kitose veiklos sistemose. Tarkime, naujas prietaisas arba taikomoji programa diegiama gamykloje, kurios sąsajos ir galbūt duomenų struktūros pateikiamos kitu nei įprasta formatu. Čia kyla iššūkis naujas sistemas padaryti sąveikias su egzistuojančiomis sistemomis ir prietaisais, siekiant užtikrinti efektyvią veiklos procesų eigą. Papildomų darbų gamykloje bus atliekama išspręsti nesutapimams tarp senų ir naujų taikomųjų programų. Gamyklos darbuotojai ir programuotojai turės dirbti kartu siekdami taip pakoreguoti naujas ir esamas taikomųjų programų sąsajas, kad taikomosios programos veiktų optimaliai. Tai iteracinis procesas ir šių iteracijų ciklai panašūs į grįžtamojo ryšio kontūrų ciklus. Rasti problemą, suprasti problemą, pataisyti problemą – tai standartinis kiekvienos veiklos

tikslas – procesų optimizavimas. Tačiau ar yra būdų automatizuoti šiuos daug reikalavimų turinčius procesus?



Pav. 4. Taikomųjų programų sąveikumo problemos (gamyklos veiklos srities pavyzdys).

Paveiksle „Taikomųjų programų sąveikumo problemų pavyzdys“ (Pav. 4) vaizduojama daviklių, aparatinės įrangos ir programinės įrangos sąveikumo problema. Ir pavyzdyje pateiktoje informacijoje aprašoma situacija gali pakeisti veiklos procesus. Vidinės duomenų struktūros gali būti integruojamos su naujomis veiklos sistemomis, tačiau kyla klausimas, kaip įdiegti šiuos naujinimus nekeičiant veiklos procesų (VP) ir užtikrinant visišką sąveikumą tarp naujų ir senų taikomųjų programų. Iš čia atsiranda reikalavimų kurti autonominės integracijos ir sąveikumo sistemas, kuriose sąveikumas būtų automatizuojamas be jokių pertraukimų. Kuriant tokias autonomines sistemas būtų minimizuojamas darbuotojų aukštos kvalifikacijos poreikis, darbo susijusio su duomenų valdymu dubliavimas ir veiklos procesų reinžinerijos poreikis (Pav. 5). Integracijos sprendimų autonomiškumas šiame scenarijuje galimas tik naudojant metodologiją su grįžtamojo ryšio ciklais ir turint vidinį veiklos procesų modelį.



Pav. 5. Veiklos procesų reinžinerija su sąveikumo komponentu ir be jo.

Jei veiklos sistemos nėra pakankamai lanksčios būti pritaikytos prie esamų taikomųjų programų, reikės papildomų pastangų veiklos procesų reinžinerijai (Pav. 5). Tačiau su autonominiu sąveikumo sprendimu dinaminėje verslo veikloje veiklos sistemos gali būti jungiamos autonomiškai ir taip išsprendžiama veiklos proceso reinžinerijos problema.

Taikomųjų programų tinklo paslaugos padeda identifikuoti dvi problemas veiklos procesų sekoje:

- Naudotojai turi bendradarbiauti tarpusavyje (tarp skyrių, padalinių) apdorojant heterogeniškus duomenis.
- Vienam naudotojui, valdančiam du ar daugiau taikomųjų programų, kai šių taikomųjų programų duomenys heterogeniški, naudotojas atlieka tam tikrą procesą kelis kartus (pvz., registruoja naujus produktus į sistemas apskaitos ir e. parduotuvę) – atsiranda proceso dubliavimas.

Atlikus literatūros analizę įvardyti reikšmingi skirtumai tarp taikomųjų programų integracijos sprendimų. Lentelėje (Lentelė 2) pateikiamos žinomos taikomųjų programų integracijos metodologijos ir sprendimai. Lentelėje EAI sistemos – verslo taikomųjų programų integracijos sistemos, lengvinančios integracijos projektuotojų darbą, nes iš dalies automatizuojamas programavimo procesas, iš dalies palengvinamas integracijos projektavimo darbas. Lentelėje naudojamas žymėjimas R – rankiniu būdu atliekami pakeitimai ir reikalingas žmogaus įsikišimas priimant sprendimus bei keičiant integracijos sprendimą. A – visiškai automatizuotai priimami sprendimai – nebereikia žmogaus įsikišimo ir integracijos procesai kontroliuojami sprendimo. D – iš dalies automatizuoti sprendimai.

Lentelės stulpeliai nuo 1 iki 5 aprašo sprendimų įgyvendinimo platformas (taip pat ir sprendimų sudėtingumo lygį):

1. **Baziniai metodai** – specialiai suprogramuoti algoritmai konkrečiam integracijos ir sąveikumo sprendimui įgyvendinti. Visuomet kuriami sprendimai per tam tikras programavimo kalbas (tiesiogiai), pavyzdžiui, C#, Java, c++, python.
2. **EAI sistemos** – tai modeliais kuriami integracijos ir sąveikumo sprendimai. Naudotojas dažniausiai naudoja grafinę sąsają sprendimams kurti. Daugelis sprendimo aspektų jau yra įprogramuoti į sąsają, tačiau retkarčiais dar tenka spręsti problemas atliekant programavimo užduotis, pavyzdžiui, darbui su „Talend“ – reikia mokėti Java programavimo kalbą, kitaip sudėtingesnių integracijos sprendimų negalima įgyvendinti.
3. **Integracijos agentai** – specializuotų integracijos agentų naudojimas. Dažniausiai agentai sukurti specifinei užduočiai ar problemai spręsti, sprendimų sėkmė priklauso nuo agento funkcionalumų išpildymo ir galimybių susidoroti su problemomis.
4. **Dirbtinis intelektas** – integracijos sprendimuose pasireiškia aktyvus dirbtinių neuroninių tinklų algoritmų naudojimas.
5. **Kiti sprendimai** – įvairūs mišrūs sprendimai integracijos ir sąveikumo problemoms spręsti.

Žinomų sąveikumo sprendimų lentelėje (Lentelė 2) analizuojami sprendimai neturi nei automatizavimo, nei savęs valdymo funkcijų, nes nebuvo tokių rasta nagrinėjamuose šaltiniuose. Dauguma analizuotų autorių šaltinių nepateikia konkrečių duomenų, skirtų atkartoti jų pasiektus rezultatus. Negalime įvertinti nagrinėjamų autorių sukurtų algoritmų, nei jų metodų taikomųjų programų integracijos automatizuojant. Žinomų sąveikumo sprendimų lentelėje (Lentelė 2) pateikiamos savybės, kurios pasiekiamos konkrečia metodologija .

Lentelė 2. Žinomi veiklos programinės įrangos sąveikumo sprendimai.

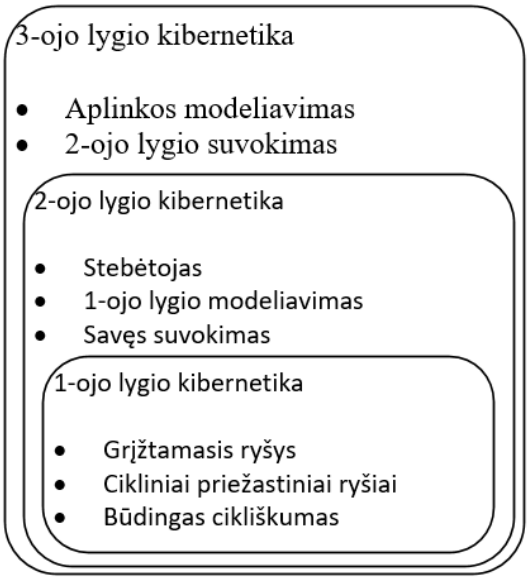
			Sprendimo metodologija (sudėtingumo lygmenys)				
	Problema		**1**	**2**	**3**	**4**	**5**
Žinomos integracijos problemos	Duomenų šaltinių pokyčių aptikimas		R	R	R	R	R
	Duomenų struktūros pokyčių aptikimas		R	R	R	D	D
	Dinaminės sistemos (keičiasi duomenų struktūros)		R	R	R	R	R
	Savybės / Objekto atpažinimas		R	D	A	A	A
	Domenų mėginiai		R				
	Duomenų heterogeniškumas		R	D	A	A	A
	Integracijos testavimas		R	A	D	D	D
	Duomenų įvedimas apribotas		R	D	A	A	A
	CRUD (Kurti / Skaityti / Naujinti / Trinti) funkcionalumas		R	D	D	D	D
	Tikslais grindžiama architektūra		R	R	D	D	D
Žinomos technologinės galimybės	Autonominis brandos indeksas		R	R	D	D	D
	Sprendimo aptarnavimas, nepriklausomas nuo darbuotojų		-	-	A	A	D
	Autonominės galimybės	Savęs redagavimas	-	-	D	D	D
		Sveikimas / Pasitaisymas	-	-	D	D	D
		Optimizacijos galimybė	-	-	D	D	D
		Savisauga	-	-	D	D	D

Automatizavimo aspektu didelė problema yra gilių žinių apie integruojamas sistemas nebuvimas. Lentelėje (Lentelė 2) pateikti sudėtingumo lygmenys: R – Rankinis, D – Dinaminis, A – Automatinis, jeigu „-“ – sudėtingumo lygmuo nežinomas. Dažnai integracijos sprendimai įgyvendinti pagal juodosios dėžės modelį: matomi įvesties ir išvesties kintamieji, neaišku, kokie priežastiniai ryšiai vyksta pačiose sistemose, ekspertas supranta šiuos ryšius, nes gali rezultatus pasitikrinti pačioje sistemoje, tačiau automatizuojant to padaryti be gilių žinių nepavyks.

Tyrinėjant taikomųjų programų integracijos sritį pastebėta, kad integracijos problemų yra daug, todėl programų integracijos ir sąveikumo galimybių vertinimas yra sudėtingesnis, kai reikia įvertinti, ar sistemos gali būti integruojamos arba sąveikauti. Išnagrinėti literatūros šaltiniai ir gauta informacija leido sudaryti taikomųjų programų integracijos srities pagrindinių problemų sąrašą.

2.5. Programų sąveikumas – kibernetikos uždavinys

Taikomųjų programų integracijos problemos yra sprendžiamos skirtingais kibernetikos lygmenimis. Kibernetikos lygmenys veikia sprendžiamų taikomųjų programų duomenų mainų problemas. Skirtingų kibernetikos lygmenų paveiksle (Pav. 6) apibrėžiami kibernetikos lygmenys. 1-ajame kibernetikos lygmenyje turime ciklinį priežastinių ir grįžtamųjų ryšių sprendimą, kuriame nėra automatizuotų metodų įvardytoms taikomųjų programų integracijos problemoms spręsti tačiau yra galim . Šiame kibernetikos lygmenyje taip pat naudojamos integracijos procese sukauptos žinios, kaip gauti informaciją, kokios sistemos yra integruojamos, kokios taikomųjų programų savybės, kokios duomenų struktūros ir panaši informacija, reikalinga integravimo projektavimo etape siekiant, kad integracijos projektas būtų sėkmingas.



Pav. 6. Skirtingi kibernetikos lygiai.

Kaip minėta, kai kurios iš įvardytų problemų sprendžiamos naudojant 1-ojo lygmens kibernetiką:

Skirtingos duomenų struktūros – integravimo specialisto, programuotojo rankiniu būdu atliekamas dviejų programų duomenų struktūros lyginimas, užprogramuojamas integracijos sprendime. Reikia programiškai keisti struktūrą esant programų architektūriniams pakitimams, integracijos procesas dažnai nustoja veikti; Jei duomenys d skirtingų programų (1, 2) yra ontologiškai ekvivalentiški, jų struktūros h sistemose 1 ir 2 gali skirtis (formulė 13).

(13) $$d_2 \equiv d_1 \rightarrow h_2 \neq h_1 .$$

1-asis kibernetikos lygmuo leidžia apibrėžti duomenų struktūrą, aprašyti, kokie gali būti priežastiniai ryšiai ir ciklišk ai vykdyti integravimo ar sąveikumo sprendimą. Programų sąveikumo sprendimas negali pats pasitaisyti ar koreguotis be stebėtojo (programuotojo) įsikišimo (Novikov, 2016).

Skirtingi duomenų šaltiniai – ištiriami, kokie duomenų šaltiniai yra programų sistemose, kokios autentifikavimo sistemos ir kokie parametrai reikalingi, tada rankiniu būdu užprogramuojama reikalinga informacija apie tai, kur rasti duomenų šaltinį. Esant duomenų šaltinių pokyčiams reikalingas perprogramavimas.

(14) $$d \in S_i \notin S_j .$$

Dinaminės sistemos – rankiniu būdu programuojami pakeitimai ir pritaikymas prie pakitusios sistemos siekiant palaikyti integracijos procesą, kitu atveju procesas nustoja veikti ir informacija neperduodama.

(15) $$S_{1,t} \neq S_{2,t} .$$

Kelios sistemos – ištiriami ir nustatomi prioritetai, perdavimo politika integruojamiems duomenims.

(16) $$S_1 \ni \{d_1, d_2, d_3, \dots\} \cup S_2 \ni \{d_2, d_3, d_5, \dots\} \rightarrow S' \ni \{d_2, d_3, \dots\}.$$

Integracijos projektavimo ir kūrimo procesas nėra automatizuotas ir pirmojo kibernetikos lygmens nebeužtenka šios srities automatizavimui atlikti. Antrajame kibernetikos lygmenyje atsiranda stebėtojas – integravimo ir sąveikumo specialistas, žinantis, kaip turi veikti sąveikumo sprendimas. Specialistas turi žinių, kitaip tariant, modelį, kaip veikia integravimo sprendimas ir kokia yra integruojamų programų architektūra (Novikov, 2016; Mancilla, 2011). Taip pat atsiranda savęs suvokimas, išreiškiamas tikslais, kuriuos turi siekti įgyvendinti integruojanti programa (kuri yra tarpininkas

tarp dviejų skirtingų integruojamų taikomųjų programų). Sistema turėtų turėti apgalvotas būsenas ir tikslus, tam tikrą savęs koregavimo laipsnį, kuriuo galėtų įgyvendinti integracijos proceso optimizavimo uždavinį. Šiuo metu vystoma nemažai tyrimų siekiant įgyvendinti save valdančias sistemas, praktikoje tokių integracijos platformų nepasitaiko.

Trečiojo lygmens kibernetika apibūdina tokią integracinę sistemą, kuri suvokia savo tikslus ir pati gali spręsti autonomiškai visas anksčiau išvardytas problemas, pati reaguoja į aplinkos pokyčius ir prisitaiko prie verslo pokyčių, kad verslo procesai nenustotų funkcionuoti. Trečiojo lygmens kibernetika turi savo aplinkos modelį (Mancilla, 2011).

Visi kibernetikos lygmenys teoriškai sujungia sisteminį poreikį programoms mainytis duomenimis ir stebėjimo bei modeliavimo procesus kiekvienam 1-ojo lygmens kibernetikos uždaviniui.

2.6. Autonominio skaičiavimo komponentai

Autonominio skaičiavimo technologija sukurta įmonės IBM iniciatyva (Horn, 2001; Kephart & Chess, 2003). Ši technologija skirta kelti automatizavimo ir autonominio brandumo lygmenį (AMI) siekiant automatizuoti sudėtingus ir daug priežiūros reikalaujančius procesus. Autonominio skaičiavimo pagrindai aprašomi IBM straipsniuose, pateikiamos pradinės idėjos apie autonominių skaičiavimų technologijos savybes ir funkcijas. Jeffrey Kephartas atstovauja IBM tyrimų centrui, jo specializacija – autonominiai skaičiavimai, netiesinė dinamika ir mašininis mokymasis. Pagrindinis straipsnis, parašytas Kepharto kartu su Chessu – 2003 metais apie autonominių skaičiavimų viziją: „The vision of autonomic computing“ (Kephart & Chess, 2003). Autonominio skaičiavimo metodologija susideda iš šių komponentų:

- Sąsajų (angl. *Touchpoints*),
- Žinių šaltinių (angl. *Knowledge*),
- Autonominių valdiklių (angl. *Autonomic Managers*),
- Valdomų išteklių (angl. *Managed Resource*).

Yra ir kitokių autonominio skaičiavimo komponentų struktūrų (Lin, et al., 2005), tačiau Kepharto ir Chesso komponentų sandara pritaikyta kuriant eksperimentą. 5 skyriuje aprašytame eksperimente autonominio skaičiavimo technologijų pagrindu kuriamas sprendimas turi sąsajas – tai taikomųjų programų sąsajų sąrašas, aprašomas žmogaus, bet jį toliau naudoja sukurtas sprendimas. Sąsajos – tai komponentai, naudojami duomenims mainyti tarp taikomųjų programų. Sąsaja nebūtinai priklauso autonominiam komponentui. Autonominis valdiklis (angl. *Autonomic Manager*) pasiekia valdomuosius

išteklius per sąsajos komponentus, pavyzdžiui, gali būti sukurta sąsaja, skirta valdyti duomenų bazių serverį, serverio duomenų bazes ir duomenų bazių lenteles.

Žinių šaltinis kuriamas analizės metu, o siūlomame sprendime kaupiami du žinių tipai:

- Sąsajų struktūra, jų įvesties ir išvesties kintamieji, panašumų analizės duomenys.
- Autonominio sprendimo analizės metu vykstančių procesų aprašymai.

Valdomieji ištekliai yra sąsajų lentelėje aprašytos programos (sąsajų lentele galima laikyti eksperimente aprašytą duomenų bazių lentelę „Endpoints“, Pav. 16). Teigiama, kad autonominių skaičiavimų technologija turi 4 pagrindines ir kelias šalutines savybes. Pagrindinės savybės susideda iš:

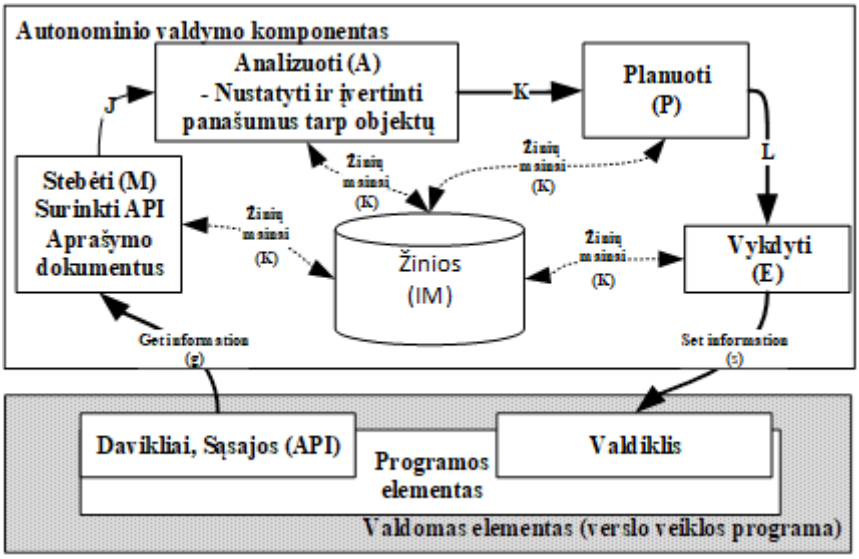
- savistabos (angl. *Self-Monitoring*),
- susikonfigūravimo (angl. *Self-Configuring*),
- pasitaisymo (angl. *Self-Repairing*),
- savisaugos (angl. *Self-Healing*).

Realybėje visas šias keturias savybes sunku įgyvendinti. Mūsų sprendimas kol kas turi tik savistabos galimybę. Pagrindinės priežastys, dėl kurių nepavyksta įgyvendinti visų savybių, – darbo apimtis. Toks autonominio sprendimo metodas būtų labai sudėtingas ir reikalautų daugiau kaip vieno autonominio komponento. Susikonfigūravimo savybei įgyvendinti reikia apibrėžti parametrų erdvę ir konfigūracines galimybes. Pasitaisymo savybei įgyvendinti reikia apibrėžti galimų gedimų erdvę, o savisaugos galimybėms įgyvendinti reikia apibrėžti galimas saugumo spragas. Dėl šios priežasties šiuo metu kuriami autonominiai sprendimai vis dar nepasiekia maksimalaus autonominio brandumo indekso (AMI) (Kephart & Chess, 2003).

Bendros paskirties autonominio skaičiavimo technologija remiasi nuo analizuojamos domeno srities nepriklausomais modeliais. Sukurti metamodeliai naudojami taisyklių varikliui (angl. *Policy Engine*) konfigūruoti. Nuo domeno srities nepriklausoma architektūra išskaidoma į dalis, kurios apibrėžia bendrosios paskirties autonominę architektūrą su keičiamomis taisyklėmis ir į ICT sistemos metamodelį su sąsaja į šios ICT architektūrą ir taikomųjų programų modelių kūrimo įrankius. Kitame etape apibrėžiama specifinei veiklos sričiai būdinga ICT ontologija, taip pat išteklių ir taisyklių rinkiniai. Taikymui būdingi sprendimai apibrėžiami uždarojo ciklo principu ir kiekvienam programinės įrangos sprendimui (sistemai) turėtų būti apibrėžtas jos modelis. Taip pat apibrėžiami valdymo adapteriai, pateikiamas pritaikytas taisyklių rinkinys ir visiškai nustatytos autonominės taisyklės,

apimančios valdomos sistemos modeliavimą ir pirmiau išvardytus apibrėžimus (grįžtamasis ryšys).

IBM autonominio skaičiavimo technologijos iliustracijoje (Pav. 7) pateikiami 4 pagrindiniai žingsniai, reikalingi autonominiam komponentui: stebėjimas, analizavimas, planavimas ir vykdymas. Šiems komponentams ir jų veikimui žinių komponentas suteikia reikalavimų ir informaciją. Autonominis valdiklis valdo resursus per daviklius ir valdiklius. Šio tyrimo atžvilgiu tai yra tinklo paslaugų valdikliai ir davikliai.



Pav. 7. IBM Autonominio skaičiavimo komponento architektūra, adaptuota (Jacob, et al., 2004).

Esminiai skirtumai tarp autonominio skaičiavimo komponento ir paprastos programinės įrangos pateikiami 2 priede (2 priedas). Programų ir taikomųjų programų sudėtingumo lyginimo pavyzdyje pateikiamas programų klasifikacijos grafikas, sudarytas identifikuoti sprendimų skirtumus tarp paprastos programos, agentų ir autonominio skaičiavimo komponentų.

Autonominis valdiklis pats kuria žinias. Dažniausiai šios žinios kuriamos stebėjimo funkcijoje surenkant duomenis iš jutiklio sąsajų. Vykdymo funkcijos dalis taip pat gali atnaujinti žinias aprašant įvykdytus veiksmus ir jų rezultatus. Žinių rinkinys yra kiekvieno autonominio valdiklio dalis. Jei žinios dalijamos tarp autonominių valdiklių, jos talpinamos į bendros paskirties žinių šaltinį. IBM autonominio skaičiavimo principų aprašyme išskiriami keli žinių tipai:

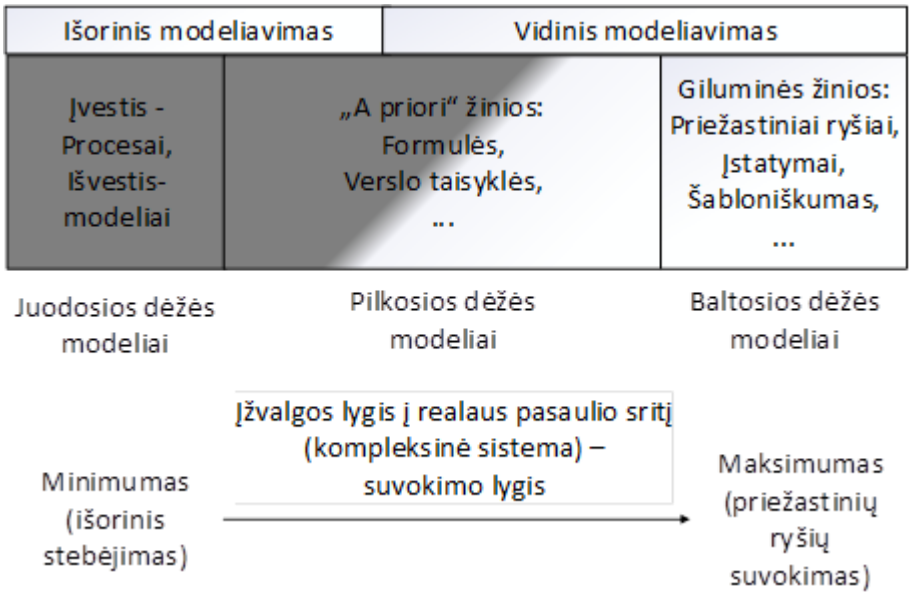
- Sprendimo topologijos – aprašo komponentus ir jų konstrukciją ir sprendimo arba verslo sistemą.

- Taisyklių – aprašo, ar sistemai reikalingi pakeitimai, kokie pakeitimai galimi.
- Problemų identifikavimo – šios žinios apima iš stebėjimo surinktus duomenis, simptomus ir sprendimų medžius. Problemų identifikavimo procesas pats gali kurti žinias. Sistema kuria atsakomuosius veiksmus tam tikroms problemoms spręsti.

Žinios, sukauptos stebėjimo funkcijos, pagal IBM rekomendacijas turėtų būti aprašomos bendrojo bazinio įvykio (angl. *Common Base Event*) formatu. Bendrasis bazinis įvykis – standartinis formatas ir turinio specifikacija aprašant įvykių struktūrą.

2.7. Modeliais grindžiama architektūra sąveikumo automatizavimo problemai spręsti

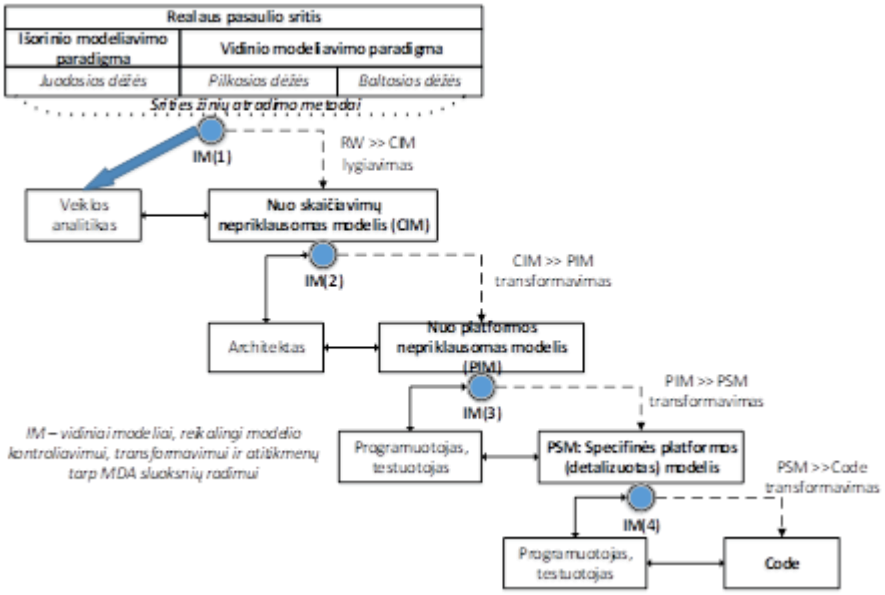
Siekiant pagerinti žiniomis grindžiamų sprendimų kūrimą, pristatyta vidinio modelio paradigma, naudojama modeliais grindžiamoje taikomųjų programų inžinerijoje (Pav. 8).



Pav. 8. Vidinio modelio naudojimas kuriant baltosios dėžės sprendimus.

Analizuojant autonominės kompiuterijos galimybes ir struktūrą, pastebėti panašumai tarp veiklos procesų valdymo modelio struktūros elementaraus valdymo ciklo (EVC). Panašumai leido išskirti bendras savybes, kuriomis remiantis galima išgauti reikiamą informaciją iš veiklos procesų ir panaudoti ją kuriant integraciją.

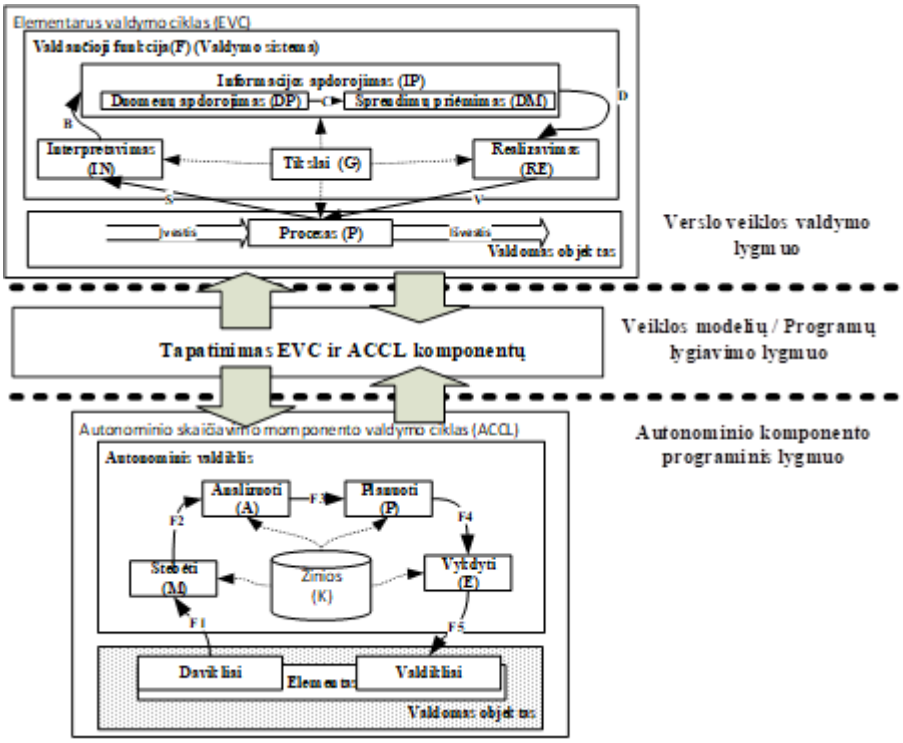
Modifikuotas MDA modelis (Pav. 9) – pagrindžia „domeno sąveikų gilumines žinias“ ir veiklos procesų atvaizdavimą į programų architektūrą.



Pav. 9. Modifikuotas MDA modelis, pagrindžiantis vidinio modelio naudojimą sistemų analizei.

Remiantis modifikuotu MDA modeliu (Pav. 9) argumentuojama, kad reikalingos žinios, kitaip vidinis modelis (IM), norint atlikti lygmenų transformaciją iš veiklos modelio į veiklos programinę įrangą. Tai aktualu, nes pabrėžia, kad programa kuriama siekiant pritaikyti ją verslui ir atitikti tam tikrus verslo veiklos procesus. Analizuotas autonomines kompiuterijos komponento (Jacob, et al., 2004) ir elementaraus valdymo ciklo EVC (Gudas,

2012) panašumas, dėl kurio iškelta teorinė galimybė taikomųjų programų integracijos automatizavimui panaudoti autonominės kompiuterijos principus.



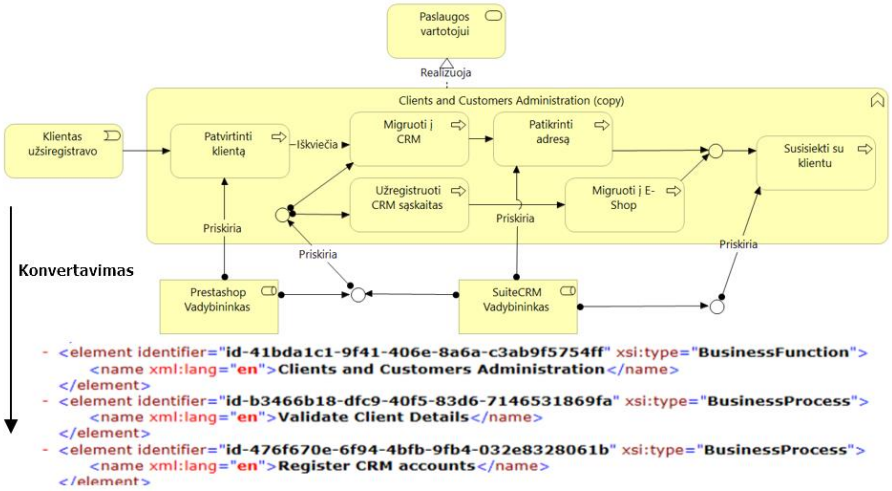
Pav. 10. Lyginamoji EVC (viršuje) ir AK (apačioje) koncepcinė diagrama.

Elementaraus valdymo ciklas pristatytas prof. dr. Sauliaus Gudo (Gudas, 2012) kaip bazinis elementas verslo veiklos valdymo modeliavime. Yra susijęs su autonominio skaičiavimo komponentu (Pav. 13). Valdymo funkcija (F) yra sudėtinė struktūra, kuri remiasi tikslais grindžiamais informacijos transformavimo žingsniais ir susideda iš: IN – interpretavimo, DP – duomenų apdorojimo, DM – sprendimo priėmimo, RE – sprendimų realizacijos ir duomenų bei informacijos srautų (S, B, C, D, V). Valdomas objektas yra materialaus srauto transformacijos procesas (P) su įeigos komponentais (sudėtinėmis dalimis ir energija) ir išeigos komponentais (produktais ir paslaugomis). Valdomąjį objektą valdo valdančioji funkcija F. Elementarus valdymo ciklas ir autonominės kompiuterijos valdymo ciklas (Jacob, et al., 2004) yra panašūs, nes buvo kuriami realaus pasaulio veikimo principais. Tikslais grindžiami informacijos apdorojimo ciklai tarp EVC ir ACCL yra panašūs, panašumai atvaizduojami Paveiksle 5. Išskirti šie panašūs elementaraus valdymo ciklai ir autonominės kompiuterijos valdymo ciklai: interpretavimas (I) susijęs su stebėjimu (M) ir analizės komponentu (A) – abu

analizuoja informaciją iš susijusių objektų, gautų iš ryšių F1, S. Informacijos valdymo procesas (IP) susijęs su planavimo ir vykdymo elementais, nes abu elementai atlieka manipuliacijas su informacija. Realizacijos komponentas (RE) susijęs su vykdymo komponentu (E), bet taip pat susijęs su planavimo komponentu (P). Tikslai susieti su žiniomis, nes abu turi gauti informaciją ir instrukcijas. Abu komponentai EVC ir ACCL atsakingi už žemesnių lygmenų valdomus elementus. Abu komponentai turi grįžtamojo ryšio ciklą.

Panašumai tarp elementaraus valdymo ciklo EVC ir autonominės kompiuterijos rodo, kad šiuos verslo srities modelius galima interpretuoti ir suprasti autonomine kompiuterija, o tai padėtų sukurti naują modelį taikomųjų programų integracijos problemoms spręsti.

AC ir EVC lyginimo paveiksle pavaizduota programos architektūra sujungta dirbtiniu būdu, tačiau modeliuojant veiklos architektūrą dažniausiai naudojamasi įvairiomis priemonėmis. Todėl antrai daliai naudota ArchiMate veiklos modeliavimo priemonė, leidžianti eksportuoti standartizuotą modelio failų formato dokumentą (angl. *Model Exchange File Format*, *MEFF*), sukurtą ArchiMate platformoje ir standartizuotą OMG, konvertavimo pavyzdys pateikiamas MEFF veikimo principinėje diagramoje (Pav. 10). Šiuo metodu modelį lengva formatuoti XML formatu ir analizuoti su kitomis sisteminėmis priemonėmis.



Pav. 11. MEFF veikimo principinė diagrama.

Tyrimo metu ištyrinėtos prielaidos, kad naudojant ir taikant organizacijų veiklos modelius visada remiamasi vidiniu modeliu su būtinai esančia hierarchine struktūra (pvz., modeliuojant realųjį pasaulį, kuriant veiklos procesų modelį, kuriant taikomųjų programų procesų modelį ir kuriant

programos architektūrą visada taikomas vidinis modelis su ankstesniuose modeliuose įgytomis žiniomis).

Pagrindiniai metodai, kurie dažniausiai skleidžiami (programiškai sukurti integravimo sprendimai), neturi tikslų, dažniausiai yra greitesni, lengviau prižiūrimi (taisyti, pritaikyti) ekspertams, kurie sukūrė sprendimą, brangesni klientams, sunku išlaikyti naujus darbuotojus, sunku išlaikyti žinias apie sprendimą. Ne visi projektai yra sėkmingi ir reikalauja skirtingos trukmės, kad būtų sukurti sveiki ir tvirti sprendimai.

Įmonių programų integravimo (EAI) sistemos teikia grafinius dizainerius. EAI sistemoms nereikia patyrusių programuotojų, kad sukurtų integraciją. Leidžia proceso orchestravimą ir choreografiją vystymosi etape. Neveikia dinamiškai, kai įdiegiami sprendimai, paprastai jie turi būti išlaikyti, o integravimo procesai kartais turi būti sustabdyti. Tačiau vienas iš privalumų – dizaineris gali grafiškai matyti ir stebėti procesus. EAI projektai taip pat apsiriboja vieno projekto apimtimi. Klaidos nustatomos ir užregistruojamos, jei įgyvendinamos pagal dizainą. Gali būti surinkti žurnalai apie integracijos procesą, apsiribojantys duomenų šaltinių rezultatais, tik maža suma gaunama iš EAI sistemos. Manoma, kad 70 proc. EAI sprendimų nepavyksta (Trotta, 2003). Dėl kritinės klaidos EAI sprendimai nutrūksta ir nustoja veikti, kol jų neištaiso integratoriai.

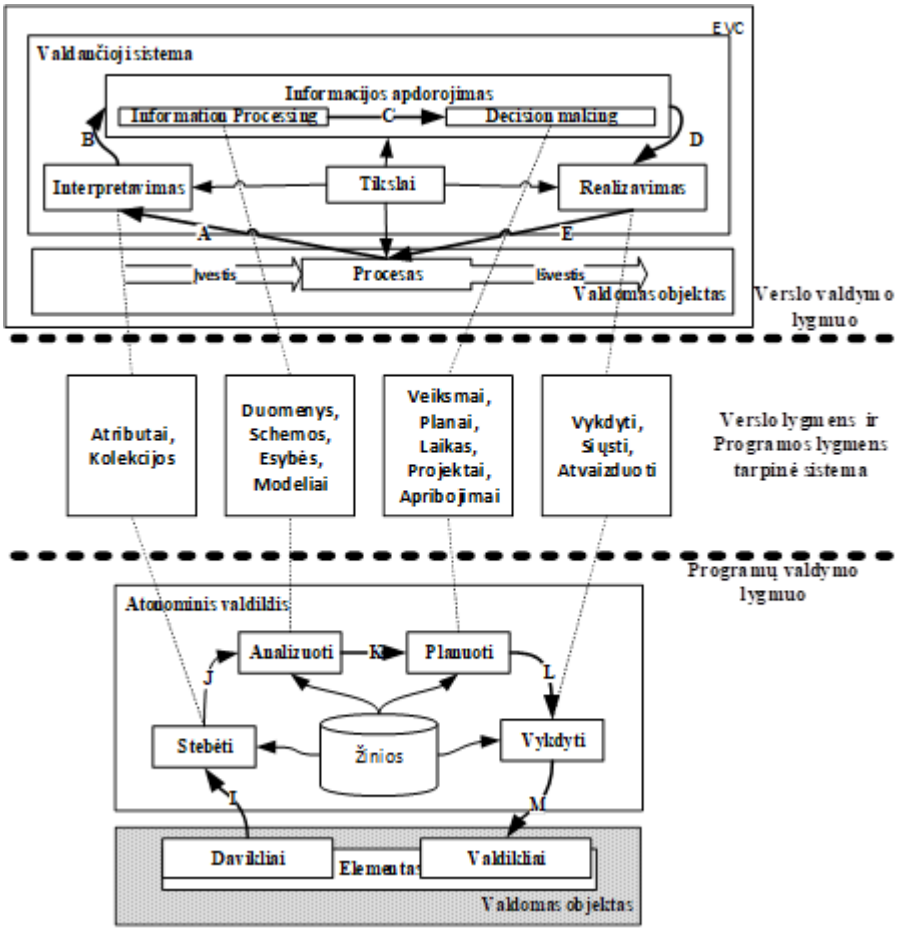
Integracijos agentai yra struktūros, sukurtos ar sukurtos asmeniu, kurį galima pritaikyti ir konfigūravimo priemonę, kad atitiktų integracijos poreikius. Norint įdiegti ir konfigūruoti tam tiktą integracijos agentą, reikia specialių techninių žinių. Kai kuriems agentams įdiegti reikia tinkamai nustatyti daug žinių (daugiausia programavimo) (Zinnikus, et al., 2008). Kiti agentai gali būti lengvai paruošiami, bet nepritaikyti konkretiems verslo poreikiams. Konfigūravimo agentai reikalauja pritaikymo, jei programos duomenų architektūra pakeista iš bet kurio duomenų šaltinio. Kai konfigūravimo agentai yra galingi įrankiai, jie gali bendradarbiauti su kitais agentais, dalydami informacijos analizės duomenis. Agento funkcionalumas gali pagerėti, jei prijungiami kiti agentai. Kai kurie agentai remia planavimą ir vykdymą integruojant procesus, tačiau neaišku, kokių veiksmų reikia šiai agentų technologijai įgyvendinti (Zinnikus, et al., 2008; Li, et al., 2005; Peng, et al., 1998). Integracijos agentai turi savo tikslus. Paprastai aprašomas žinių elementas. Reikia kvalifikuotų ekspertų projektuoti, konfigūruoti ir (arba) konfigūruoti. Paprastai naudojama konkretiems scenarijams arba ribotam procesų kiekiui (tik projekto dydžiui). Nėra informacijos apie verslo procesą ar jo elgesį. Programuojamus integracijos sprendimus turi skirti techniniai ir patyrę darbuotojai. Programuojami integracijos sprendimai paprastai orientuojasi į mažesnes integracijos formas: duomenų architektūra

suderinama (Rahm & Bernstein, 2001). Ne visi integracijos agentai gali prisitaikyti prie pokyčių.

Kiti dinaminių procesų integravimo būdai, kartais susiję su bendradarbiaujančių organizacijų veiklos procesų susiejimu – kitaip tariant, verslas-verslui (B2B) procesų integracija (El-Halwagi, 2006; Pavlin, et al., 2009; Kutsche & Milanovic, 2008; Valatavičius & Dilijonas, 2014; Euzenat, et al., 2010; Tan, et al., 2004). Šie metodai sprendžia pagrindines verslo procesų integravimo problemas, dauguma iš jų yra išbandyti ribotoje ir sukomplektuotoje aplinkoje, o tokių metodų rezultatai – perspektyvūs, tačiau dažniausiai teoriniai. Nežinoma, kokia yra pateiktų sprendimų struktūra, kaip juos įgyvendinti, išskyrus semiotinį organizacinio modeliavimo metodą, naudojant normų analizę (Tan, et al., 2004).

Teoriškai informacijos taikomųjų programų pagrindai apibrėžia elementarius valdymo ciklus (EVC) (Gudas, 2012). Elementarus valdymo ciklas susideda iš valdymo sistemos (angl. *Management System*) ir valdomo objekto (angl. *Managed Object*). Valdymo sistema apibrėžia tikslus, interpretavimą, informacijos analizę ir talpinimą bei sprendimų priėmimą ir įgyvendinimą. Valdomas objektas apibrėžiamas kaip procesas su įeigos ir išeigos ištekliais. Elementaraus valdymo ciklo dizainas padengia beveik visą natūralų veiklos proceso valdymo veikimą. Elementarus valdymo ciklas, kaip ir autonominio skaičiavimo valdymo ciklas ASVC, yra panašūs, nes buvo projektuojami remiantis realiojo pasaulio modeliavimo principais. Esminis skirtumas tas, kad EVC labiau atsižvelgia į materialius proceso srautus, o šioje disertacijoje daugiau dėmesio skiriama informacinių procesų srautams. Panašumai tarp EVC ir ASVC pateikiami „Elementaraus valdymo ciklo (EVC) atitikmenys autonominio skaičiavimo valdymo ciklo (ASVC) funkcijoms“ lentelėje.

EVC ir ASVC lyginime (Pav. 12) komponentai EVC ir ASVC yra panašūs. Interpretacija yra susijusi su susijusių objektų stebėjimu ir analize iš duomenų, gautų iš informacijos šaltinių. Informacijos valdymo procesas yra susijęs su planavimo ir vykdymo komponentais. Realizacija yra labiau vykdymo dalis, tačiau taip pat susijusi su planavimu autonominiame valdymo cikle. Tikslai – labai artimai susiję su žiniomis, nes apibrėžia informaciją, reikalingą operacijoms vykdyti. Tikslai ir žinios susiję su visų kitų komponentų valdymu ir kontroliavimu. Abu (EVC ir ASVC) yra cikliniai ir rodo, kad metodologijos dinamiškos ir gali turėti pagrindinių savybių.



Pav. 12. Elementaraus valdymo ciklo (EVC) atitikmenys autonominio skaičiavimo valdymo ciklo (ASVC) funkcijoms.

Išvardyti EVC ir ASVC panašumai leidžia manyti, kad autonominių skaičiavimų komponentai tikrai galėtų padėti pagerinti taikomųjų programų sąveikavimo metodus.

Apibendrinant, MDA architektūra panaši į autonominio skaičiavimo komponentų architektūrą. MDA modeliai ir veiklos procesų modeliai gali būti susieti siekiant gauti daugiau informacijos apie integruojamas sistemas.

2.8. Taikomųjų programų tinklo paslaugos

Veiklos sistemos sukurtos naudojantis paslaugomis grindžiama architektūra (angl. *Service Oriented Architecture*, SOA) lemia, kad sistemos esiniai ir operacijos pateikiamos kaip paslaugos, kurios gali būti prieinamos tinkle, kitaip tariant, tinklo paslaugos. Šį aspektą svarbu paminėti disertacijoje, nes

ne visos veiklos sistemos gali būti atvirojo kodo, jų integravimui ar sąveikumui užtikrinti galima jungtis tik prie tinklo paslaugų sąsajų. Šiuo atveju negalima pasiekti duomenų bazių norint sukurti sujungtus duomenų atvaizdus (angl. *Views*). Tinklo paslaugų ir automoninio skaičiavimo technologijos galimybės (Lentelė 3) atspindi tinklo paslaugų savybes ir galimybes pagerinti šias savybes autonominio skaičiavimo technologija.

Lentelė 3. Automoninio skaičiavimo technologijų galimybė pagerinti tinklo paslaugų galimybes.

Tinklo paslaugos	**Autoniminių skaičiavimų metodas**
Valdymą atlieka patyrę darbuotojai. Pasikeitus architektūrai daromos modifikacijos integracijos ir sąveikumo sprendimuose	Save valdanti sistema naudojanti žinias ir nusitaikanti į tikslus: Politika (*Policies*), Taisyklės (*Rules*), Apribojimai (*Restrictions*), kita.
Prisitaikymo prie aplinkos greitis priklauso nuo valdančios komandos greičio	Greitas prisitaikymas, jei žinomas problemos sprendimas.
Dideli modifikavimo kaštai	Savaime modifikuojasi
Jungiasi prie iš anksto numatytų paslaugų tuo pačiu metu	Gali prisijungti prie duomenų šaltinių pagal poreikius, taisykles ir apribojimus.
Jungimosi prie programos paslaugų orchestravimas, atliekamas programuojant, naudojant EAI įrankius.	Orchestravimas gali būti atliekamas išanalizavus veiklos procesų (BP) žinias
Turi galimybę aktyvuoti ir išjungti paslaugas, jei tai įgyvendinta integracinėje sistemoje.	Galimybė aptikti jungimosi sutrikimus analizuojant vykdymo laikus ir istorinius integracijos proceso duomenis.
Sprendimas plačiai paplitęs ir naudojamas.	Sprendimas reikalauja eksperimentų ir testavimo, dar kol kas nėra žinoma, kokių gali iškilti problemų

Lentelė aprašo pagrindinius skirtumus tarp programuojamo sąveikumo sprendimo projekto naudojant tinklo paslaugas ir siūlomo metodo naudojant pritaikytą autonominio skaičiavimo metodą, skirtą sąveikumo sprendimui kurti ir valdyti.

El-Halwagi ir kt. (El-Halwagi, 2006) daugiausia dėmesio skyrė procesų integracijai, pabrėžiant integracijos problemų pagrindus. Y. Pengas ir kt. (Peng, et al., 1998) daugiausia dėmesio skyrė daugiafunkcei įmonių integravimo sistemai, siekiant išspręsti su verslo suderinamumu susijusias problemas, bet nepateikė informacijos apie integracijos valdymą. R. McCannas ir kt. (McCann, et al., 2005) pabrėžė duomenų integravimo taikomųjų programų priežiūros žemėlapių nustatymo problemas. X. Luna Dong, et al. (Dong & Divesh, 2013) atliko duomenų integravimo tyrimą, pabrėžė dažniausiai pasitaikančias problemas, susijusias su integracijos tema, 2006 metais. E. Rahmo ir kt. (Rahm & Bernstein, 2001) pasikartojanti apklausa parodė integracijos dalyko patobulinimus ir tai, ko dar trūksta šioje srityje. Autoriai analizavo dabartines įmonių programinės įrangos taikomųjų programų integravimo problemų tendencijas .

Sistemos integracija inžinerijos ir informacinių technologijų srityje yra skirtinga. Inžinerinės sistemos integracija apima posistemio komponentus, integruotus į vieną, kuri paskui veiktų kartu kaip sistema. Informacinių technologijų sistemos integravimas sieja skirtingus atskirų programinės įrangos taikomųjų programų, kurios yra valdomos ar koordinuojamos, duomenų šaltinius arba tiesiog sumažina prieigos taškus, reikalingus tam tikrai informacijai pasiekti.

Žemiausias programinės įrangos sistemos integravimo lygmuo yra duomenų integravimas ir duomenų architektūros lyginimas. Duomenų architektūros suderinimo metodai padeda kurti tokius algoritmus, kurie gali susieti skirtingų programinės įrangos duomenų šaltinius. Sąveikumo vertinimas gali turėti kelis tipus: sąveikumo galimybių, suderinamumo, sąveikumo efektyvumo (angl. *Interoperability Potentiality, Compatibility, Performance*) (Chen, 2006).

Esami programų sąveikumo metodai (

Lentelė 4):

1. aplinkos aspektų įvertinimas naudojant vertinimo korteles (angl. *Scorecard*): LISI (Kasunic, 2001),
2. komunikacijos funkcijų sąveikumo galimybės naudojant panašumų matricą, I-Score metodologiją (Ford, et al., 2008),
3. lyginimas pagal funkcionalumą (Dzemydienė & Naujikienė, 2009).

LISI metodas gali būti patogus būdas įvertinti sąveikumo galimybių rezultatus, tačiau nebūtinai pateikia tikslią ir objektyvią informaciją apie tai, kokie yra bendri duomenys (Kasunic, 2001). LISI vertinimas taip pat gali užtrukti ilgai, nes jo metu reikia įvardyti galimybes. Kiekvienai programai, kuriai norima atlikti sąveikumo galimybių vertinimą, reikia atsakyti į klausimyną ir gaunamas rezultatas kaip sąveikumo galimybių įvertis konkrečiai sistemai. Šio metodo trūkumas tas, kad užpildžius klausimyną neatsižvelgiama į programos vidaus architektūros panašumus su kitos programos panašumais. Taip pat neperteikia programos struktūros, pavyzdžiui, neatsako į klausimus: Kokios gali būti bendros funkcijos? Kokie bendri objektai tarp sąveikaujančių sistemų? LISI metodas nėra deterministinis, tad rezultatai atsitiktinai gali būti skirtingi dėl analitiko žinių. Taip pat šio metodo naudojimas remiasi subjektyvia samprata apie sistemas ir priklauso nuo analitiko žinių apie kiekvieną iš taikomųjų programų. Jei žinių kiekis apie sistemas mažas – šis matavimo būdas netinka objektyviai įvertinti taikomųjų programų sąveikumo galimybių.

I-Score sąveikumo galimybių vertinimo metodas (Ford, et al., 2008) taikomas identifikuojant ir klasifikuojant atributus bei savybes. Kiekvienos poros atstumas išmatuojamas ir rezultatai talpinami į panašumų matricą. Šio metodo esmė – įvertinti atributus ir funkcijas, paversti juos į panašumų matricą. Metodo esmė labai panaši į disertacijoje siūlomą; pagrindinis skirtumas tas, kad autorius naudoja skirtingus panašumo vertinimo metodus ir matuoja panašumą ne pagal programų architektūros atributus, o pagal jų galimybes keistis informacija, kitaip tariant, komunikacijos būdus.

Vertinimas pagal funkcionalumą atliekamas stebint programų veiklą ir vizualiai vertinant jos architektūrą. Toks lyginimas dažnai siejasi su sąveikumo sprendimo siūlytojo žiniomis ir įžvalgomis apie tyrinėjamas sistemas. Įžvalgos paliečia duomenų bazių fizinio, semantinio ir socialinio sąveikumo lygmenis (Dzemydienė & Naujikienė, 2009).

Lentelė 4. Verslo programų sąveikumo vertinimo metodų lyginimas

	LISI	I-Score	Pagal funkcionalumą
Kaip atliekama reikalavimų analizė?	Klausimynas	Identifikuojant atributus ir funkcijas	Analizuojant, stebint programos veikimo principus
Kaip vertinama Sąveikumo galimybė?	Klausimyno atsakymai ranguojami LISI lentelėje	Kuriama panašumų matrica	UML sekų diagrama atspindi sąveikumo galimybes
Kas vykdo duomenų surinkimą?	Žmogus	Žmogus	Žmogus
Kas vykdo panašumų vertinimą?	Žmogus	Automatizuotas	Žmogus

Integracijos ir sąveikumo procesai turi savus standartus ir taisykles. Ne visi taikomųjų programų kūrėjai, architektai, platintojai laikosi standartų, todėl turi būti kuriami ir plėtojami integracijos ir sąveikumo standartai.

2.10. Redagavimo nuotolio skaičiavimai

Redagavimo nuotolio (angl. *Edit Distance*) skaičiavimai leidžia apskaičiuoti skirtumus tarp žodžių tekste. Redagavimo nuotolio skaičiavimas gali būti panaudotas eksperimente skaičiuojant programų architektūrinius panašumus.

Redagavimo nuotolis skaičiuojamas pagal tai, kiek reikia atlikti viename žodyje pakeitimų, kad jis sutaptų su kitu žodžiu. Laikoma, kad nagrinėtose programose žodžiai sintaksiškai ir iš dalies semantiškai panašūs, jei nėra sinonimai ir redagavimo nuotolis yra 0. Žodžiai semantiškai ir sintaksiškai skirtingi, kai redagavimo nuotolis didesnis už 0. Siekiant įvertinti skirtingų veiklos taikomųjų programų S schemų h tarpusavio panašumą, pasitelkiami skaičiavimai, leidžiantys įvertinti redagavimo nuotolius λ_{red}. Taigi pirmas sistemų panašumo vertinimas remsis tokia formuluote:

(17) $S_1 \rightarrow S_2 \; kai \; \sum \lambda_{red}\{h_1, h_2\} \geq \theta$.

, kai λ_{red} – redagavimo nuotolio įvertis, θ – panašumo slenksčio konstanta (jei viršijama, laikoma, kad sistemos panašios).

Redagavimo nuotolio skaičiavimas naudojamas analizuojant tokiems atributų skirtumams kaip, pavyzdžiui:

- Product – Products
- Product – ProductList
- Product – GetProducts
- Product – RetrieveProducts

Yra daug būdų skaičiuoti redagavimo nuotolio įverčius. Šioje disertacijoje naudoti 4 redagavimo nuotolio būdai (Navarro, 2001):

- Levenshteino
- Jaro – Winklerio
- Ilgiausios bendros sekos (angl. *Longest Common Subsequence*)
- Jaccardo

Levenshteino metodu skaičiuojamas redagavimo nuotolis pagal mažiausią skaičių teksto simbolių pakeitimų, reikiamų pakeisti vieną žodį kitu. Levenshteino algoritmas yra seniausiai pasiūlytas žinomas algoritmas, aprašytas 1965 metais. Dviem žodžiams a ir b randamas mažiausias simbolių pakeitimų skaičius, kad a taptų b.

(18) $d_{i0} = \sum_{k=1}^{i} \omega_{del}(b_k)$, $for \; 1 \leq i \leq m.$

(19) $d_{0j} = \sum_{k=1}^{i} \omega_{ins}(a_k)$, $for \; 1 \leq j \leq n.$

(20) $$d_{ij} = \begin{cases} d_{i-1,j-1} & for \; a_j = b_i \\ min \begin{cases} d_{i-1,j} + \omega_{del}(b_i) \\ d_{i,j-1} + \omega_{ins}(a_j) \\ d_{i-1,j-1} + \omega_{sub}(a_j, b_i) \end{cases} & for \; a_j \neq b_i \end{cases} \quad for \; 1 \leq i \leq m, 1 \leq j \leq n.$$

Jaro-Winklerio metodu skaičiuojama, kiek panašių atributų (simbolių junginių) turi lyginami žodžiai. Skaičiuojama, kiek simbolių junginių reikia transponuoti siekiant sutapatinti pateiktus žodžius.

$$(21)\ sim_j = \begin{cases} 0 & if\ m = 0 \\ \frac{1}{3}\left(\frac{m}{|s1|} + \frac{m}{|s_2|} + \frac{m-t}{m}\right) & if\ m \neq 0 \end{cases}.$$

$$(22)\ \left\lfloor \frac{\max(|s_1|,|s_2|)}{2} \right\rfloor - 1.$$

$$(23)\ sim_w = sim_j + (lp(1 - sim_j)).$$

Ilgiausios bendros sekos metodu apskaičiuojama, kiek simbolių junginių sutampa pateiktuose žodžiuose.

$$(24) O\left(2^{n1}\sum_{i>1} n_i \times\ 2^{m1}\sum_{j>1} m_j\right).$$

Jaccardo metodu apskaičiuojama, kiek panašių atributų yra abiejuose lyginamuose žodžiuose, suskaičiuojamas bendras atitinkamų simbolių skaičius.

$$(25) J = \frac{M_{11}}{M_{01}+M_{10}+M_{11}}.$$

$$(26) d_j = \frac{M_{01}+M_{10}}{M_{01}+M_{10}+M_{11}} = 1 - J.$$

Šiomis teksto lyginimo formulėmis galima atlikti teksto analizę ir su tam tikru automatizavimu spręsti apie taikomųjų programų panašumą.

2.11. Skyriaus išvados

Apžvelgti literatūros šaltiniai, kurių autoriai nagrinėjo taikomųjų programų integracijos ir sąveikumo temą. Tema yra vis dar aktuali ir sukelia daug diskusijų, nes ne visos įvardytos integracijos ir sąveikumo problemos yra išspręstos. Dalis iš išprestų problemų vis dar nėra plačiai įsisavinamos versle. Įvardyti taikomųjų programų integracijos tipai ir paaiškinti sąveikumo principai. Įvardytos integracijos ir sąveikumo problemos. Aprašyti autonominiai skaičiavimų agentai. Skyriuje apžvelgti esami programų sąveikumo vertinimo metodai.

Literatūros apžvalga leido įvertinti verslo programų pagrindines integracijos ir sąveikumo problemas:

1. Skirtingos duomenų struktūros,
2. Skirtingi duomenų šaltiniai,
3. Gilių žinių ir vidinio modelio nebuvimas,
4. Dinaminės sistemos,
5. Kelios sistemos,
6. Skirtingi autentifikavimo procesai,
7. Semantinis esinių ir objektų atpažinimas,
8. Orchestravimas ir choreografija,
9. Duomenų mėginių ėmimas,
10. Integracijos testavimo nebuvimas,
11. Ontologijų integracija,
12. Dokumentacijos trūkumas,
13. B2B integracija,
14. Nestandartinių vienetų, kalbos skirtumų, perteklinių etikečių, formatavimo,
15. Netvarkingi duomenys.

Literatūros analizės metu suklasifikuota programinė įranga pagal sudėtingumo lygius, kurie pateikiami 2 priede (2 priedas).

Literatūros analizės metu pasirinktas redagavimo nuotolio skaičiavimo metodas programų sąveikumo galimybėms įvertinti.

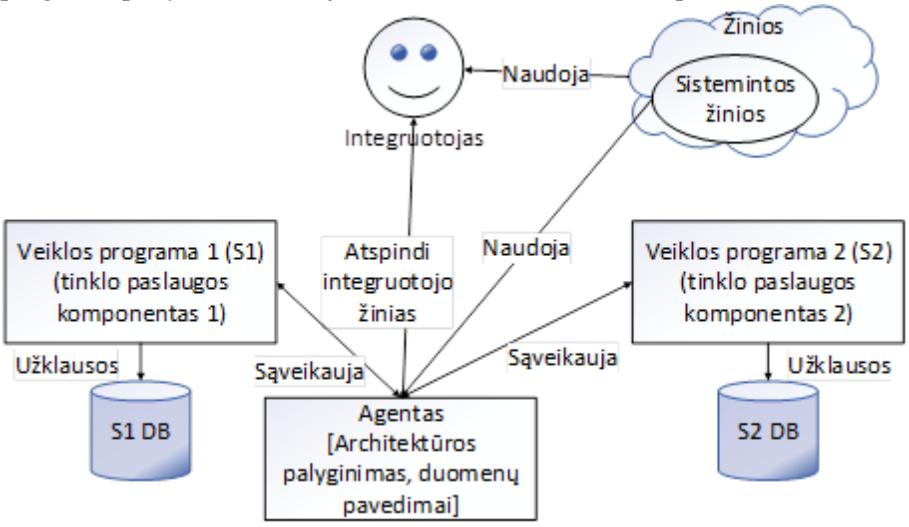
Literatūros analizėje apžvelgti metodai leido atrasti sąsajų tarp sąveikumo problemų ir autonominio skaičiavimo technologijų pagrindžiant tyrimo tikslą tuo, kad įmanoma sukurti taikomųjų programų sąveikumo įvertinimo principus ir sąveikumo įvertinimo automatizavimo metodą, grindžiamą taikomųjų programų priežastinių ryšių analize.

3. VEIKLOS PROGRAMŲ SĄVEIKUMO MATAVIMAI

Iš literatūros apie verslo veiklos programas žinoma, kad jos sudėtingėja ir reikia automatizuotų sprendimų programų sąveikumui spręsti. Šiame skyriuje pateikiamas siūlomas veiklos programų sąveikumo vertinimo metodas. Šio metodo esmė – sujungiamos skirtingos technologijos:

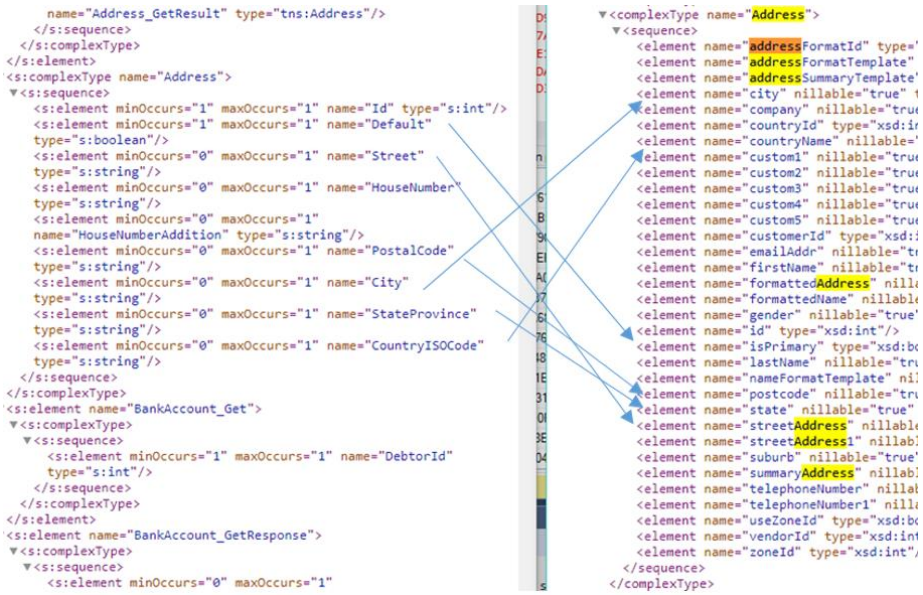
1. Verslo veiklos procesų modeliavimas (BPM)
2. Modeliais grindžiama architektūra (MDA)
3. Autonominė skaičiavimo technologija
4. Verslo veiklos programų sąveikumo vertinimo sprendimai

Taikomųjų programų sąveikumo sprendimo pavyzdys naudojant agentines technologijas apibūdina agento ryšį tarp sistemų atliekant duomenų transakcijas (Pav. 12). Integracijos projektuotojas, sąveikumo sprendimus projektuojantis ir įgyvendinantis asmuo, dar vadinamas integruotoju (angl. *Integrator*). Integratorius sukuria agentą su iš anksto aprašytomis integracijos taisyklėmis, sisteminėmis žiniomis apie integruojamas programų sistemas ir jų savybes. Integracijos projektuotojas pats naudoja žinias apie verslo procesus, integruojamas sistemas ir jų struktūrą bei kuria tokį agentą, kuris naudodamas šias žinias turės tam tikrą judėjimo laisvę esant taikomųjų programų pokyčiams, vėluojantiems duomenis ar kitoms problemoms.



Pav. 12. Taikomųjų programų integravimo pavyzdys naudojant agentines technologijas.

Kadangi agentas neturi žinių apie galimus veiksmus ir pasekmes (priežastinius ryšius), agentas nesupranta (t. y. negali analizuoti ir atlikti jam priskirtų veiksmų) probleminių situacijų. Neturėdamas konkretaus verslo procesų modelio agentas nesupranta integruojamų taikomųjų programų struktūros, pavyzdžiui, „System 1“ ir „System 2“ (žr. Pav. 12). Esant dramatiškiems pokyčiams, neatitinkantiems agento suvokimo (agente suprogramuotų sisteminių žinių) modelio, agentas nustos veikti ir verslo procesai nebevyks arba bus sutrikdyti, kol agentas nebus atitinkamai pakoreguotas. Toliau pateikiamas skirtingos duomenų struktūros jungimo grafinis pavyzdys (Pav. 13):



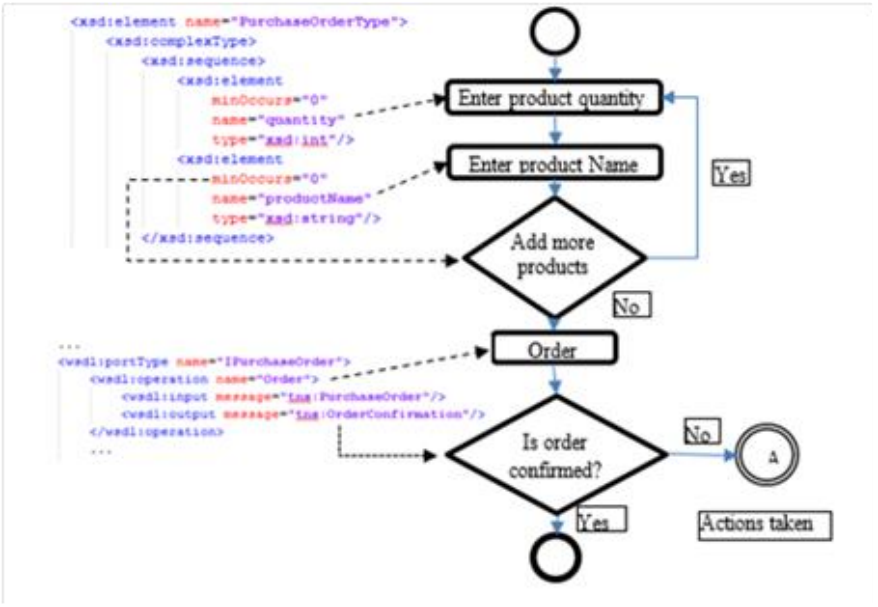
Pav. 13. Programinio kodo iškarpa – skirtingų duomenų struktūrų jungimo grafinis pavyzdys.

Programinio kodo iškarpoje (Pav. 13) pavaizduotas dviejų skirtingų nutolusių taikomųjų programų duomenų struktūros, kurias reikia sujungti.Griežtai statinių struktūrų jungimas vykdomas programiškai arba EAI (angl. *Enterprise Application Integration*) taikomosiomis programomis. Norėdami automatizuoti duomenų struktūros procesus tyrėjai kuria algoritmus, kurie galėtų atpažinti naudojamus raktažodžius, duomenų tipus, spręstų semantinius esinių ir objektų atpažinimo uždavinius (El-Halwagi, 2006; McCann, et al., 2005). Dalis iš integracijos automatizavimo algoritmų realizuoti naudojant agentus.

Norint sužinoti, ar įmanoma automatizuoti ar pagerinti programų integraciją ir sąveikumą, reikia nustatyti kiekvienos tiriamos sistemos panašumą su kita sistema.

3.1. Siūlomas autonominis integracijos sprendimas

Verslo taikomųjų programų integracijos metodas gali būti papildytas autonominės kompiuterijos technologija, kuri analizuotų tinklo paslaugas ir tyrinėtų jų veiklą, duomenų struktūrų dokumentus (WSDL, Pav. 14).



Pav. 14. Užsakymų WSDL schemos susiejimas su veiklos procesų modeliu (CIM).

Užsakymų WSDL schemos lygiavimo pavyzdyje (Pav. 14) pateikiama **užsakymo kūrimo proceso** dalis iš vieno verslo subjekto. Šis verslo procesas pavaizduotas veiksmų sekos diagramos iškarpa (Pav. 14). Paveiksle vaizduojama, kaip konkrečios programos tinklo paslaugų WSDL duomenų struktūros dokumentą galima susieti su konkrečios veiklos procesu. Taigi jei autonominės kompiuterijos žinių komponentas turėtų verslo veiklos procesų žinias (tokioje pat modelio formoje arba formalaus aprašymo formoje kaip „https://schema.org/Order“ pateiktame pavyzdyje), teoriškai galima būtų sukurti integracijos metodą, kuris galėtų suprasti savo kontekstą ir vidinę sandarą. Paveiksle 14 raktažodžiai „Order“, „Confirmation“ gaunami iš verslo veiklos proceso modelio. Algoritmas, paremtas autonominės kompiuterijos principais, išanalizuotų elementus, žinutes ir operacijas WSDL dokumente. Naudojant savaime apsimokantį dirbtinį intelektą simuliuotoje erdvėje galima išmokyti autonominį integracijos algoritmą naudotis šia informacija, kad galėtų keisti savo parametrus prisitaikydamas prie konkrečios struktūros.

Taikomųjų programų sąveikumo metodas su autonominio skaičiavimo technologija analizuoja tinklo paslaugų šaltinius stebėdamas jų veiklą ir analizuodamas jų duomenų struktūros pakeitimus sistemose. Pavyzdžiui, duotai tinklo paslaugai procesas, kaip manipuliuoti duomenimis, gali būti gaunamas iš WSDL failų arba aprašomųjų REST protokolo failų. Paveiksle 13 vaizduojama, kaip duomenų architektūra yra sutapatinama su procesu. Jei

autonominio skaičiavimo žinių elementas turėtų veiklos procesų žinias (modelių, formalių aprašymų pavidalu panašiai kaip „*https://schema.org/order*“), pirmiausia toks sąveikumo metodas galėtų turėti savęs pažinimo galimybę (angl. *Self-Aware*) ir suprasti duomenų kontekstą iš veiklos procesų perspektyvos. Antra, gali būti atpažįstama veiklos procesų grandinė, kuri leidžia valdyti orchestravimo ir choreografijos veiksmus sąveikaujančiuose komponentuose. Galiausiai kiekvienos programos duomenų architektūra galėtų būti lyginama su modeliais, egzistuojančiais žinių bazėje, ir galima būtų apibrėžti naujus arba trūkstamus elementus.

Paveikslas 14 vaizduoja įsivaizduojamą modelių jungimą tarp veiklos procesų sekų ir WSDL schemos. Kaupiant ir saugant žinias autonominiam skaičiavimo komponentui šis sujungimas gali būti pasiekiamas programiškai. Iliustruotame pavyzdyje atrandami raktažodžiai „Order“ ir „Confirmation“ (liet. užsakymas ir parvirtinimas). Dėl šių raktažodžių elementai žinutės ir operacijos gali būti analizuojamos iš tokių dokumentų. Pavyzdyje reikalavimai registruojant pirkimo užsakymą duoti ir įvardyti elemento „PurchaseOrderType“ lauke, o užsakymą patvirtinanti žinutė grąžinama elemento „OrderConfirmationType“ lauke. Naudojant vien tik bandymo ir klaidų metodą, simuliuotoje aplinkoje galima apmokyti išmanų autonominio skaičiavimo komponentą – šis apmokymas ir būtų žinios apie sąveikaujančių taikomųjų programų struktūrą. Taikomoji programa dažniausiai neturėtų leisti kurti užsakymo su atsitiktiniais produktais neegzistuojančiais sistemoje ir todėl klaidos žinutė turėtų būti grąžinama. Klaidos žinutės gali reikšti ir kad perduodami duomenys yra neteisingi arba kad nėra tokio elemento. Jei šis produktas galėtų būti užregistruotas sistemoje užsakymo vykdymo metu – tai reikštų sėkmingą integraciją tik jeigu tai buvo apibrėžta veiklos proceso žinių modelyje.

3.2. Sąveikumas naudojant autonominio skaičiavimo technologijas

Autonominio skaičiavimo idėja pristatyta Jeffrey O. Kepharto ir Davido M. Chesso 2003 metais kaip būdas susidoroti su vis didėjančiu verslo programų kompleksiškumu (Kephart & Chess, 2003). Aukščiausio lygio autonomija (t. y. visiškai automatizuotas sprendimas, veikiantis be jokio žmogaus įsikišimo) pasiekiamas nustatant tikslus ir apribojimus (angl. *Goal and Policies*). Idealiu atveju neturėtų būti reikalaujama techninių įgūdžių turinčių žmonių įsikišimo. Gali būti, kad autonominiai sprendimai reikalautų daugiau valdymo lygmens žinių turinčių darbuotojų, kai keičiami nustatymai, tikslai ir apribojimai. Šiuo metu net naudojant autonominę kompiuteriją reikia nurodyti prieigos prie kiekvienos sistemos taškus ir kai kurias sistemas paruošti sąveikavimui,

pavyzdžiui, įjungti tam tikrus tinklo paslaugų modulius. Autonominių skaičiavimų elementai gerai tinka didelių ir sunkių užduočių sprendimams kurti, kaip ir agentai, autonominio skaičiavimo komponentai gali dirbti grupėmis; pagrindinis skirtumas yra tas, kad autonominiai skaičiavimo komponentai turi tikslus produktyvumui, apsaugai ir atsistatymui po trikdžių ar atakų.

Pagrindinis motyvas naudoti autonominio skaičiavimo komponentus yra tas, kad jie turi keturias galimybes: savistabos, susikonfigūravimo, pasitaisymo, savisaugos. Šios galimybės yra autonominių komponentų dedamieji tikslai. Autonominiai komponentai padeda išskaidyti sudėtingas problemas į lengvesnes leidžiant atskirti valdomą komponentą ir autonominį valdiklį. Autonominis valdiklis pagal apibrėžimą turi turėti procesus, leidžiančius stebėti, analizuoti, planuoti ir vykdyti užduotis. Šie procesai susieti su žinių komponentu, kuris saugo reikšmingą informaciją, surinktą kiekvieno proceso metu ir leidžia šią informaciją naudoti kitiems komponentams. Visas autonominio skaičiavimo komponento veikimas yra ciklinis ir vadinamas autonominio skaičiavimo valdymo ciklu – ASVC. Jungiant skirtingus ASVC galima kurti stiprias sistemas, kurios sprendžia kompleksines užduotis su tam tikra autonomija. Autonomijos lygmenys skirstomi į penkias pakopas:

1. Bazinis
2. Valdomas
3. Prognozuojamas
4. Prisitaikantis
5. Autonomiškas

Norint spręsti sudėtingas sąveikumo problemas šioje disertacijoje pasirinkta autonominio skaičiavimo technologija dėl visų pirmiau išvardytų savybių ir dėl sąsajų su veiklos procesų sekų modeliais. Elementarūs valdymo ciklai struktūriškai panašūs į ASVC, todėl tikėtina, kad šie panašumai gali padėti išspręsti sąveikumo problemas. Detalesnė analizė pateikiama eksperimentinėje dalyje.

3.3. Skyriaus išvados

Šiame skyriuje pasiūlytas ir apibrėžtas verslo programų sąveikumo vertinimo sprendimų metodas. Sąveikumo vertinimo sprendimų metodas susideda iš siūlomo autonominio sąveikumo sprendimo, redagavimo nuotolio skaičiavimų ir modeliais grindžiamos architektūros modelių analizės. Siūlomo metodo esmė – skirtingų technologijų jungimas:

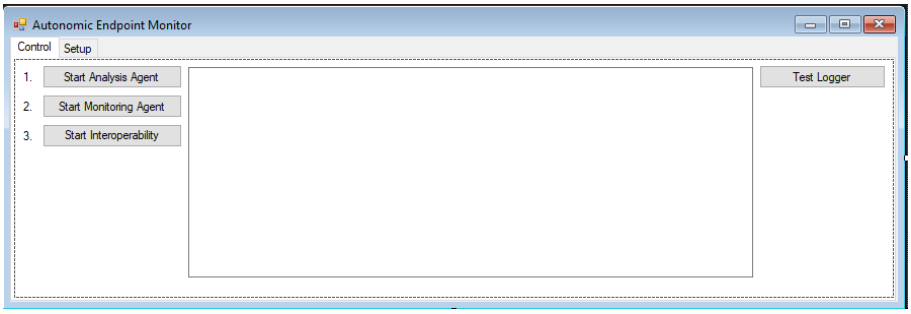
1. Verslo veiklos procesų modeliavimas (BPM)

2. Modeliais grindžiama architektūra (MDA)
3. Autonominiai skaičiavimo komponentai
4. Verslo veiklos programų sąveikumo vertinimo sprendimai

Šis technologinis jungimas leidžia gauti giluminių žinių apie verslo veiklos sritį ir procesus. Modeliuojant verslo procesus naudojamas BPM metodas ir ArchiMate paketas. ArchiMate modeliai yra lengvai interpretuojami naudojant programinį kodą, todėl tinka autonominio skaičiavimo komponento žinių elementui papildyti reikalinga informacija. Autonominis skaičiavimo komponentas naudoja sukauptas žinias ir atlieka programų sąveikumo vertinimo analizę, planuoja veiksmus su duomenimis ir įvykdo suplanuotus veiksmus.

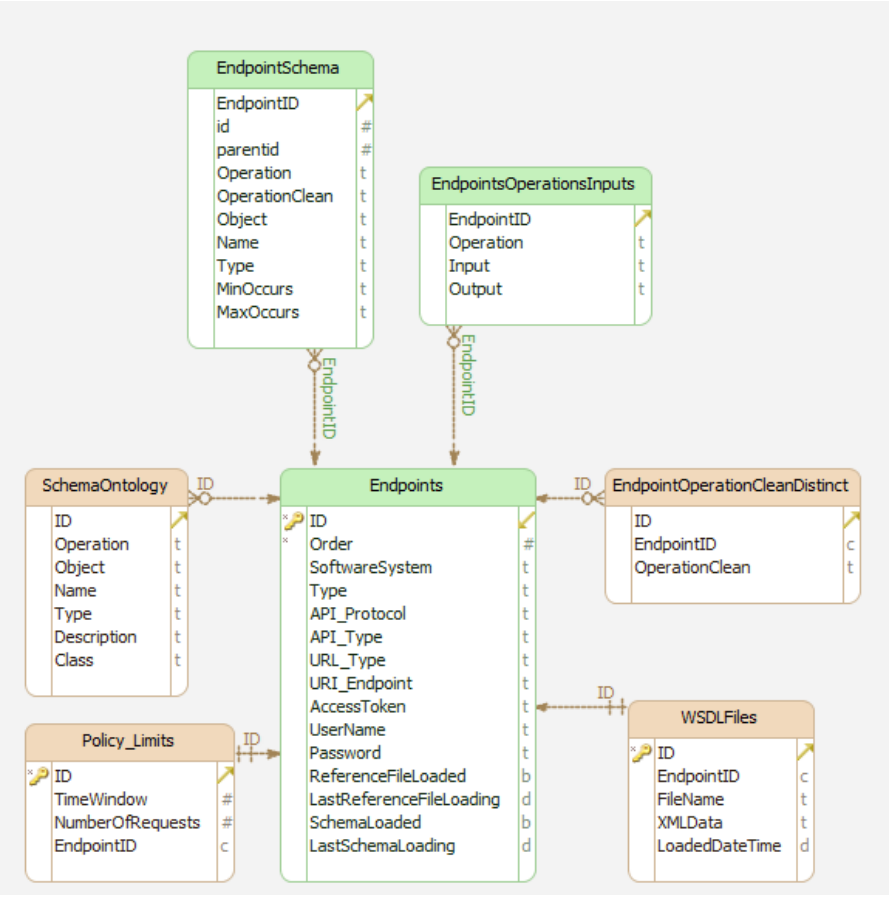
4. SĄVEIKUMO GALIMYBIŲ VERTINIMO EKSPERIMENTO APRAŠYMAS

Šiame skyriuje nagrinėjamas šiame darbe aprašyto eksperimento procesas. Tyrimui C# programavimo kalba sukurtas įrankis, leidžiantis išgauti, stebėti ir surinkti duomenis iš pateiktų duomenų šaltinių. Šie duomenų bazių šaltiniai pateikti lentelėje „Endpoints“ (Paveikslas 17). Lentelė „Endpoints“ turi laukus, kuriuose nurodomas programos pavadinimas, tinklo paslaugos protokolo tipas ir nuoroda į tinklo paslaugos aprašymo dokumentą. Tai pagrindinė informacija, reikalinga pradėti rinkti duomenis iš tinklo paslaugos sąsajos. Įrankis pavadinimu „Autonomic Endpoint Monitor“ turi tris agentus (Paveikslas 16): Analizės, Stebėjimo ir Sąveikumo vykdymo. Kol kas tik analizės agentas yra įgyvendintas ir toliau nagrinėjamas šiame eksperimente. Paleidus analizės agentą inicijuojamas procesas, analizuojantis duomenų šaltinių nuorodas į tinklo paslaugos dokumentus.



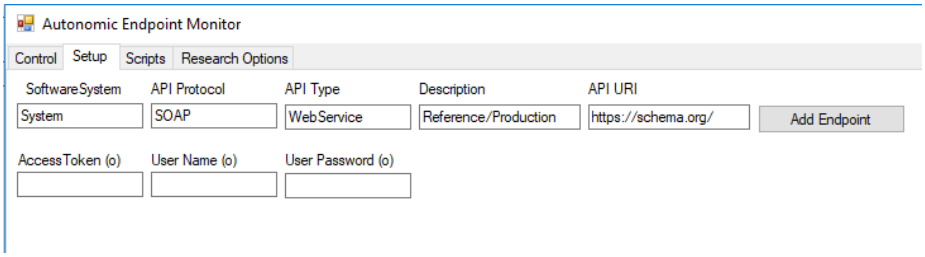
Pav. 15. Taikomųjų programų šaltinių analizės naudotojo sąsaja

Sukurta reliacinė duomenų bazė, aprašanti duomenis, skirtus užduotims įgyvendinti. Reliacinės duomenų bazės tikslas – gauti ir išsaugoti analizės metu gautus duomenis kaip žinias – tolesnių etapų ir agentų veikimui užtikrinti. Naudojamos kelios reliacinės duomenų bazės: PostgreSQL ir Microsoft SQL Server. Sistema pradėta projektuoti naudojantis nemokama Microsoft licencija, tačiau joje būta vietos apribojimų, todėl taip pat naudotasi PostgreSQL reliacine duomenų baze.



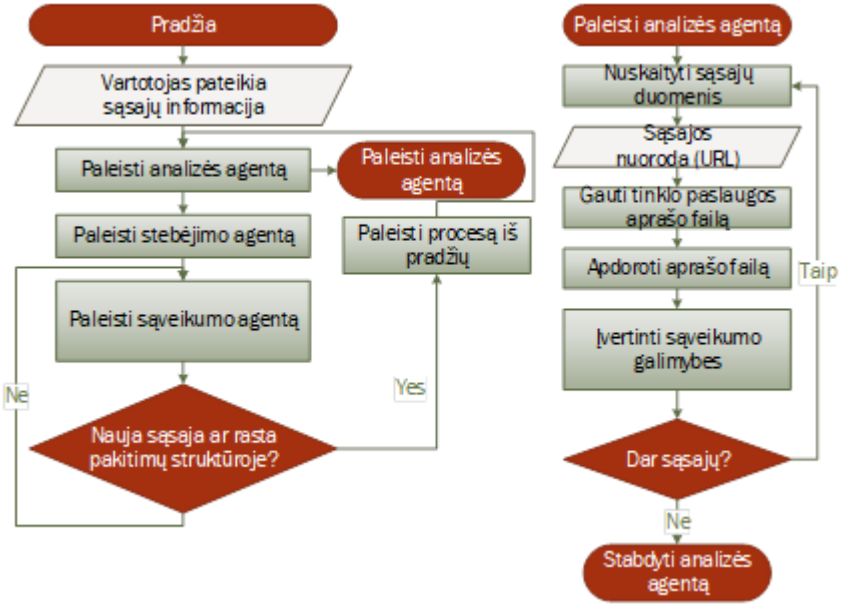
Pav. 16. Duomenų bazės architektūra Microsoft SQL serveryje.

Duomenys gaunami eilės tvarka skaitant sąsajų duomenis, aprašytus lentelėje „Endpoints“, susisiekiama su tinklo paslauga ir bandoma gauti schemos dokumentą WSDL arba tiesiogiai išanalizuoti programinės sąsajos šaltinį. Dėl programinės įrangos verslo veiklos atstovas gali įvesti tinklo paslaugų adresus, kurie išsaugomi lentelėje „Endpoints“. Nustatymų lange „Setup“ galima įvesti reikiamus duomenis sistemos sąsajos aprašymui analizuoti (Pav. 17). Suvedama reikalinga informacija (sistemos pavadinimas, tinklo sąsajos protokolas, apibūdinimas) ir pateikiama nuoroda į sąsajos aprašą bei prisijungimo duomenys, jei tokie yra. Paspaudus mygtuką „Add Endpoint“ duomenys atsiranda „Endpoints“ lentelėje (Pav. 16). Iš čia agentas, atliekantis analizę, nuskaito sąsajos adrese (API URI) laikomą aprašymą ir pradeda analizę.



Pav. 17. Nustatymų langas pridėti tinklo sąsają.

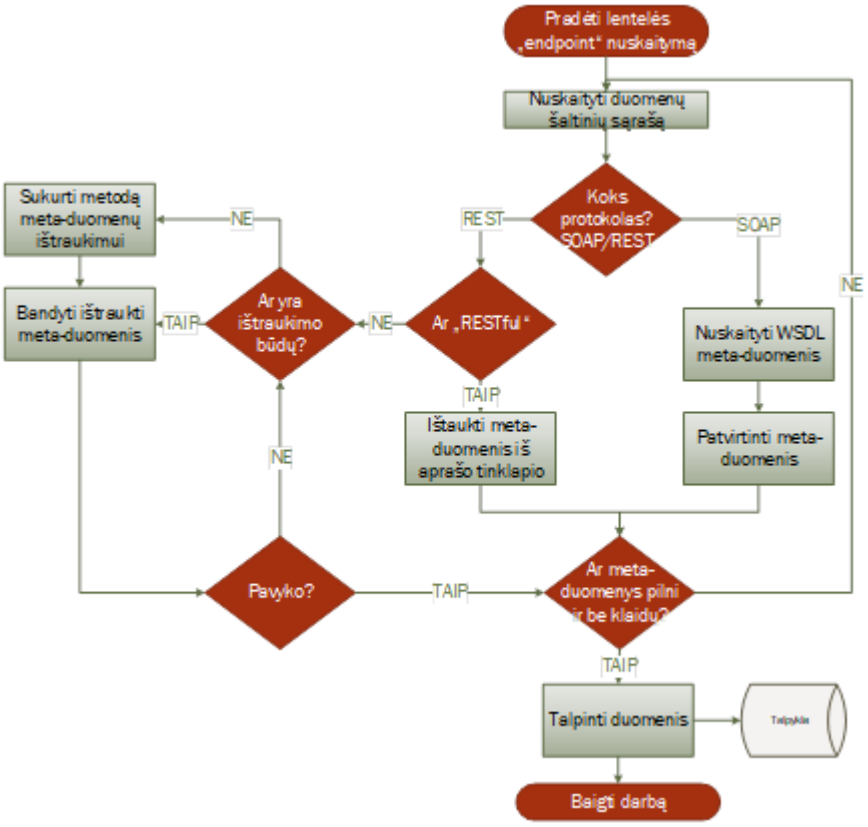
Analizės agentas skaito įrašus lentelėje „Endpoints“, juos analizuoja, analizuoja tinklo paslaugų turinį ir pateikia jį „Endpoint Schema“ lentelėje.



Pav. 18. Analizės agento veiksmų diagrama

Analizės agentas veikia skaitydamas visus aprašytus prieigos taškus ir analizuodamas jų struktūrą pagal pavaizduotą veiksmų seką (Pav. 18).

Duomenų bazėje aprašome lyginamų programų prieigos taškus, tinklo sąsajas. Autonominis komponentas kreipiasi į URI_Endpoint lauke nurodytą adresą, nuskaito struktūrą (SOAP – WSDL arba REST HTML). Toliau autonominis agentas atlieka tokią veiksmų seką metaduomenims gauti:



Pav. 19. Paslaugos metaduomenų nuskaitymas atliekamas šia veiksmų sekos diagrama

Veiksmų sekos diagrama vaizduoja, kaip atliekamas tinklo paslaugos metaduomenų gavimas ir nuskaitymas (Pav. 19). Analizės agentas surenka įmanomus duomenis, iš kurių po to gali planuoti tolimesnį savo darbą. Analizės metu gauti duomenys talpinami į duomenų talpyklą. Kai analizės darbai baigti, komponentas baigia darbą ir nustato, kada pradės vykdyti kitą iteraciją. Šiame tyrimo etape analizės ir planavimo darbai nėra automatizuoti, jie atliekami atskirai – rankiniu būdu.

Testavimui „Endpoints“ duomenų bazė surinkta iš (Rezaei, et al., 2014) taikomųjų programų lyginimų lentelių, taip pat iš kitų šaltinių, atitinkančių paieškos kriterijus: CRM API, ERP API, E-Commerce API, Accounting API. Surinktų duomenų lyginimų lentelė (tęsinys 1 priede).

Lentelė 5. Sutrumpintas tiriamų objektų sąrašas (tęsinys 1 priede).

Programa	**Tipas**	**API**	**Licencija**
SuiteCRM	CRM	SOAP	
NMBRS	HR-Payrol	SOAP	
ExactOnline	ERP	REST	
PrestaShop	E-Commerce		
KonaKart	E-Commerce	SOAP	GNU LGPL
LemonStand	E-Commerce	REST	Proprietary
Magento	E-Commerce	SOAP, REST	OSL 3.0
NopCommerce	E-Commerce	REST	GPL
...	...	...	...

Lyginimų lentelė skirta autonominiam analizavimo agentui. Analizavimo agentas tikrina, ar visos sistemos turi duomenų architektūrą aprašantį dokumentą, t. y. ar galima ištraukti duomenų struktūras iš kiekvieno tikrinamo objekto. API (angl. *Application Programing Interface*) – tai taikomųjų programų programavimo sąsaja, kitaip tariant, sąsaja, per kurią pasiekiamos funkcijos ir duomenys. Šiame tyrime tirti tik du plačiau paplitę API protokolai: SOAP ir REST (

Lentelė 5). Surinktų duomenų sudėtį galima aprašyti šia forma:

$\forall\{\exists h_1 \rightarrow S_1, \exists h_2 \rightarrow S_2, \ldots, \exists h_n \rightarrow S_n\}$.

Kitaip tariant, visų sistemų schemos priklauso tik toms sistemoms. Negali egzistuoti duomenų architektūra (angl. *Schema*), nepriklausanti sistemai. Ištrauktą duomenų architektūros aprašą agentas pateikia duomenų bazės lentelėje „EndpointSchemas“ po schemos struktūros apdorojimo, jeigu šis sėkmingas.

Šiam tyrimui pasirinktos PrestaShop, ExactOnline, NMBRS ir SuiteCRM platformos. Kiekviena platforma turi skirtingų vaidmenų ir aspektų verslo veikloje.

ExactOnline (S1) – veiklos išteklių planavimo (ERP) sistema, ji turi daugybė modulių – daugiausia skirta apskaitai, bet turi ir produktų valdymo ir CRM galimybių.

PrestaShop (S2) – e. komercijos sistema, skirta elektroninei parduotuvei valdyti. Prestashop yra lanksti ir joje gali būti diegiama daugybė modelių, tačiau daugiausia tai – produktų pardavimo ir užsakymų valdymo sistema.

SuiteCRM (S3) – santykių su klientais valdymo (CRM) sistema, jos pagrindinis tikslas – aktyviai bendrauti su klientais ir tiekėjais, planuoti įvykius ir susitikimus, planuoti pirkinius ir registruoti užsakymus.

NMBRS (S4) – algalapių ir darbo laiko apskaitos sistema, jos pagrindinis tikslas – apskaičiuoti darbo užmokestį, įsiskolinimus darbuotojams ir planuoti darbuotojų darbo laiką.

Specialiai šiam tyrimui pasirinktos tokios skirtingos sistemos siekiant parodyti jų sąveikumo galimybes, išmatuoti sąveikumo potencialą ir pavaizduoti, kurios tinklo paslaugų operacijos galėtų sąveikauti.

Šiame skyriuje aprašyta eksperimento eiga, naudojami įrankiai, programiniai paketai ir technologijos. Aprašomi duomenų gavimo metodai ir šių metodų įgyvendinimas. Aprašomi surinkti duomenys ir jų pobūdis. Iš viso surinkta 30 programų šaltinių, iš jų analizuoti 13 programų paketų, nes apsiribota tik SOAP ir REST sąsajas turinčiais paketais. Pagrindinė analizuotų programų sistemų kombinacija susideda iš 4 programų: PrestaShop, ExactOnline, NMBRS ir SuiteCRM.

5. SĄVEIKUMO GALIMYBIŲ VERTINIMO EKSPERIMENTO REZULTATAI

Sąveikumo galimybių vertinimo eksperimentas remiasi prielaida, kad galima vertinti, ar sistemos sąveikios pagal struktūrų panašumus. Renkantis, kokias programas reiktų analizuoti, remtasi verslo veiklos programomis, sukurtomis remiantis paslaugomis grindžiama architektūra SOA (angl. *Service Oriented Architecture*). Pagrindinis dėmesys skiriamas išskirtinai SOAP ir REST protokolams.Iš pradžių tinklo paslaugų operacijos yra lyginamos tarpusavyje.

Pirmiausia, tyrimo metu bandyta naudoti egzistuojantį LISI metodą programų sąveikumui vertinti. Jis sukurtas ir aprašytas M. Kasunico (Kasunic, 2001) 2001 metais bendradarbiaujant su Amerikos saugumo skyriumi DoD (angl. *Department of Defence*). Iki šiol šis metodas dažnai cituojamas ir papildomas kituose sąveikumo ir integravimo sprendimų straipsniuose (Basson, et al., 2016; Sitton & Reich, 2016).

Lentelė 6. Pasirinktų taikomųjų programų sąveikumo matavimai LISI metodu.

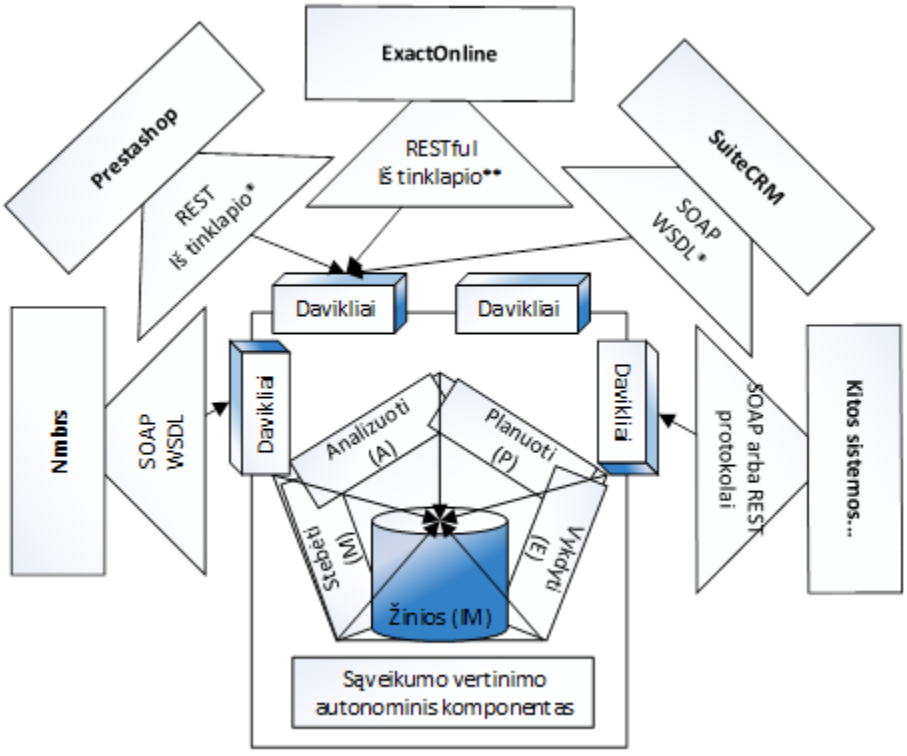
a) Techninis požiūris, techninis sąveikumo vertinimas.		b) Sisteminis požiūris, programų sąveikumo lentelė			
Šaltinis, programa	Atitikimas standartams	S1	S2	S3	S4
S1 ExactOnline	Y		Y	Y	G
S2 PrestaShop	Y	Y		Y	Y
S3 SuiteCRM	G	Y	G		Y
S4 NMBRS	G	Y	R	R	

Iš šios lentelės (Lentelė 6) matyti taikomųjų programų: a – atitikimas standartams ir b – sąveikumo įverčių lentelė, kurioje G – atitinka standartus, tai būdinga SOAP protokolą naudojančioms sistemoms, nes SOAP gerai aprašomas ir lengvai analizuojamas. Y – ne visai atitinka standartus, o aprašymai dažniausiai skiriasi.

REST protokolas neturi griežtos aprašymų metodikos (tik rekomendacinę) arba rekomendacinė metodika nenaudojama. Iš sąveikumo lyginimų lentelės 3b žymėjimas spalvomis anglų kalba:

- R – Red; turi mažai arba visai neturi bendrų duomenų, funkcijos skirtingos, sąveikumas negalimas.
- Y – Yellow; turi bendrų duomenų, dalis iš funkcijų yra panašios.
- G – Green; sistemos sąveikaujančios – įmanoma, kad sistemos galėtų laisvai dalytis duomenimis ir funkcijomis.

Iš šio vertinimo matyti, kad S1 (ExactOnline) turėtų labai atitikti S4 (NMBRS), taip pat S3 (SuiteCRM) turėtų gerai sąveikauti su S2 (PrestaShop) (Lentelė 6). Sąveikumo vertinimo sistemos su autonominiu komponentu koncepcinėje diagramoje (Pav. 20) pateikiamos kelios eksperimente analizuotos sistemos. Sukurtas autonominis komponentas šiam eksperimentui, tačiau jo įgyvendinimas dar neužbaigtas. Autonominis komponentas gali stebėti schemų pokyčius sistemoje ir juos analizuoti bei saugoti rezultatus duomenų bazėje. Pagal SOAP ir REST protokolų apibrėžimus SOA taikomųjų programų kūrimas turėtų remtis tam tikromis dizaino rekomendacijomis. Tačiau ypač REST protokolą naudojančios sistemos dažniausiai nepaiso rekomendacijų ir nepateikia bendrų taikomųjų programų tinklo paslaugas aprašančių failų. Šie dažnai skiriasi ir sukurti autonominį komponentą, tinkantį daugeliui taikomųjų programų, yra labai sudėtinga. Sąveikumo vertinimo sistemos su autonominiu komponentu koncepcinėje diagramoje (Pav. 20) žvaigždute „*“ pažymėtiems protokolams reikėjo atlikti papildomų veiksmų siekiant gauti operacijas ir objektus iš šių taikomųjų programų.



Pav. 20. Sąveikumo vertinimo sistemos su autonominiu komponentu koncepcinė diagrama.

Kai kuriais atvejais (SuiteCRM) neužtenka remtis vien tik tinklo paslaugos apraše pateikta informacija, kad būtų galima sukurti objektų aprašus ir atlikti metaduomenų analizę. Tokiu atveju šiam sprendimui sukurta papildoma galimybė kreiptis į tinklo paslaugą, kad ši grąžintų objektų sąrašą.

Eksperimento metu atlikti žingsniai siekiant apskaičiuoti taikomųjų programų sąveikumo galimybę:

1. Aptikti tinklo paslaugos dokumentaciją
2. Ištraukti ir išanalizuoti metaduomenis iš dokumentacijos failo
3. Kategorizuoti išanalizuotus metaduomenis į:
    a. Operacijas – tai funkcijos, procedūros, operacijos, kurios dažniausiai yra veiksmai, susiję su duomenų apdorojimu, pavyzdžiui: gauti klientų sąrašą, pakeisti užsakymo duomenis, įvesti naują prekę.
    b. Metodus – tai veiksmas, naudojamas atlikti konkrečiai operacijai, pavyzdžiui: gauti, įrašyti, trinti, atnaujinti, nustatyti (angl. *GET, SET, INSERT, DELETE, SELECT* ir kt.).
    c. Objektus – tai pagrindiniai programos komponentai, bendrai apibūdinantys konkrečius duomenis, pavyzdžiui: klientai, kontaktai, produktai, užsakymai, inventorius, transakcijos ir kt.
    d. Laukų pavadinimus – kiekvienas objektas turi laukus, laukų pavadinimai gali būti įvairūs, tačiau būdingi tiktai konkrečiam objektui, pavyzdžiui, objektui „klientas“ būdingi laukai yra šie: vardas, pavardė, adresas, telefono numeris, amžius, el. pašto adresas ir kt.
    e. Laukų tipus – tai būdas apibūdinti kiekvieno lauko tipą, pavyzdžiui, telefonas visada skaitmuo, vardas ir pavardė visada tekstas (angl. *String*).
4. Išskirtinai operacijoms sukurti metaduomenys:
    a. Gauti šaltinio pavadinimą
    b. Gauti tinklo paslaugos pavadinimą
    c. Gauti aktyvavimo metodus (GET, POST, PUT, DELETE, PATCH, HEAD...)
    d. Gauti operacijų pavadinimo įvesties ir išvesties duomenis
        i. Panaikinti operacijose pasikartojančius, nereikšmingus pavadinimus, tokius kaip: „list“, „all“, „get“, „retrieve“.
5. Saugoti operacijų metaduomenis duomenų bazėje

6. Apdoroti operacijas naudojant redagavimo nuotolio algoritmus
7. Saugoti rezultatus duomenų bazėje
8. Duomenų vizualizacija

Šioms sistemoms (NMBRS, PrestaShop, ExactOnline, SuiteCRM) taikytas iš dalies automatizuotas duomenų šaltinių analizės metodas, įgyvendintas C# kalba. Pateikiama koncepcinė diagrama, vaizduojanti analizuojamas taikomąsias programas ir jų protokolus siekiant įvertinti jų architektūrą ir galimybę sąveikauti.

Autonominis agentas gali atlikti duomenų stebėjimą ir saugoti pokyčius stebėjimo etape M (Pav. 20). Išsaugoti duomenys apdorojami analizavimo etape A, išvalomi ir formatuojami taip, kad būtų galima naudoti kartu su redagavimo nuotolio algoritmais.

5.1. Vertinimas redagavimo nuotolio metodu

Redagavimo nuotolių analizė atlikta su R statistinio programavimo kalba naudojant „stringdist“ paketą. Iš metaduomenų informacijos ir išanalizavus tinklo paslaugas galima suskaičiuoti, kiek unikalių operacijų (nesikartojančių per objektus, laukus ir tipus) galima atlikti su sistemomis. Lyginamųjų taikomųjų programų unikalių operacijų skaičius pateikiamas lentelėje:

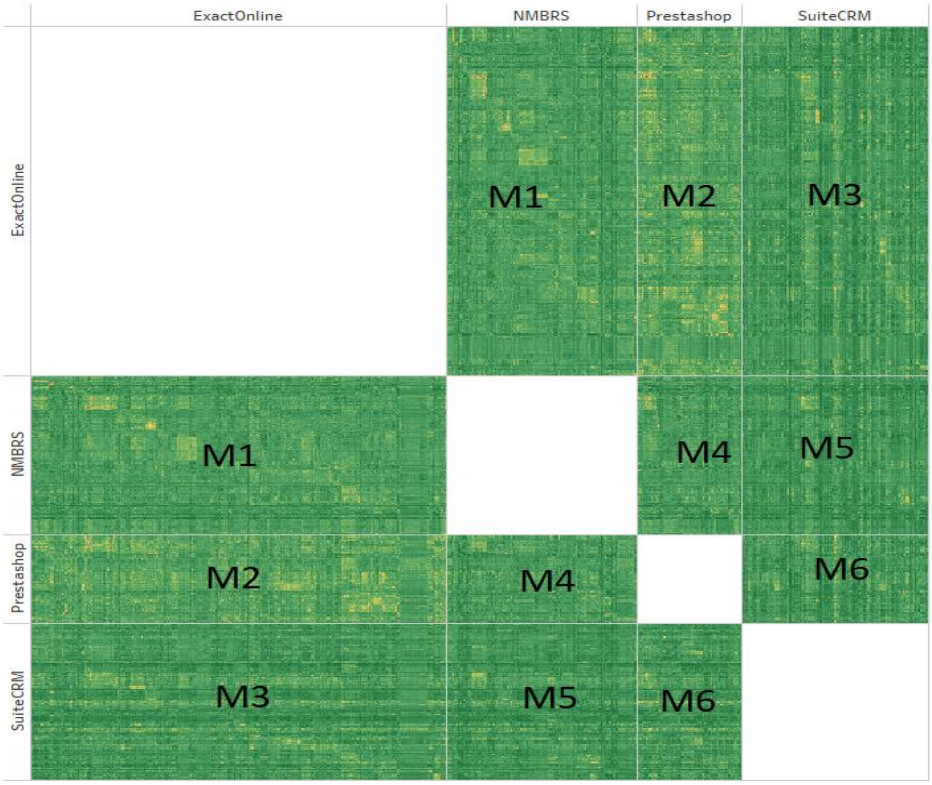
Lentelė 7. Unikalios operacijos kiekvienos programos tinklo paslaugoje

Verslo programa	**API**	**Operacijos**	**Aprašymas**
NMBRS_SingleSignOn	REST	4	On-site payroll and HR
PrestaShop	REST	49	On-site e-commerce
NMBRS_ReportService	SOAP	60	On-site payroll and HR
NMBRS_DebtorService	SOAP	60	On-site payroll and HR
LemonStand	REST	74	On-site e-commerce
SuiteCRM	SOAP	98	On-site CRM
KonaKart_StoreFront	SOAP	128	On-site e-commerce
OpenCart	REST	160	Cloud e-commerce
Zen Cart	REST	175	On-site e-commerce
NMBRS_CompanyService	SOAP	200	On-site payroll and HR
KonaKart_Administration	SOAP	226	On-site e-commerce
ExactOnline	REST	293	Cloud accounting
NMBRS_Employees	SOAP	402	On-site payroll and HR
MIVA	REST	4322	Cloud e-commerce

Daugiausiai unikalių išvalytų operacijų turi NMBRS_Employees (402) ir MIVA (4322).

Nors didžiausias paketas yra ExactOnline, paslaugos NMBRS_Employees ir MIVA turi daug daugiau operacijų. Iš viso eksperimento metu išnagrinėtos 6861 unikalios operacijos ir jų ryšiai, kitaip tariant, panašumai su kitos programos operacijomis. Visų programų operacijos išvalomos, paliekami tik reikšminiai žodžiai, galimai nurodantys objektą. Iš viso liko 6228 tokių unikalių išvalytų operacijų. Apibendrinant galima teigti, kad viena sistema gali turėti vidutiniškai 426 unikalių išvalytų operacijų, tačiau matyti iš lentelės, kad sistemos, turinčios per daug operacijų, gali iškraipyti vertinimą, todėl iš skaičiavimo pašalinus MIVA ir www.schema.Org operacijas lieka 132 unikalių išvalytų operacijų vienai sistemai.

Operacijų panašumo vertinimas kiekvienai pasirinktai taikomajai programai ir jos porininkei skaičiuojamas kaip operacijų pavadinimo redagavimo nuotolis, įvertintas procentine išraiška nuo 0 iki 100. Jeigu redagavimo nuotolis kiekvienai operacijai yra pakankamai mažas, o procentinė panašumo išraiška didelė, tai parodo, kad programų pora panaši. Rezultatai paveiksluose (Pav. 21, Pav. 22, Pav. 23, Pav. 24) apibendrina operacijų pavadinimų redagavimo nuotolio rezultatus. Matricos, aprašytos M1–M6, pateikia visų galimų operacijų panašumus (Pav. 21). Žaliai išreikštos operacijos yra skirtingos, geltonai ir raudonai nuspalvintos operacijos yra panašios. Matricos M1–M6 (Pav. 21) kartojasi, nes taip pateikiamos įmanomos kombinacijos tarp taikomųjų programų, kaip ir LISI pavyzdyje (Lentelė 6). Kiekviena matrica atspindi dviejų taikomųjų programų panašumą, pavyzdžiui, M1 yra ExactOnline ir NMBRS operacijų panašumo žemėlapis. Reikšmingi regionai vaizduojami „šiltesnėmis“ spalvomis.



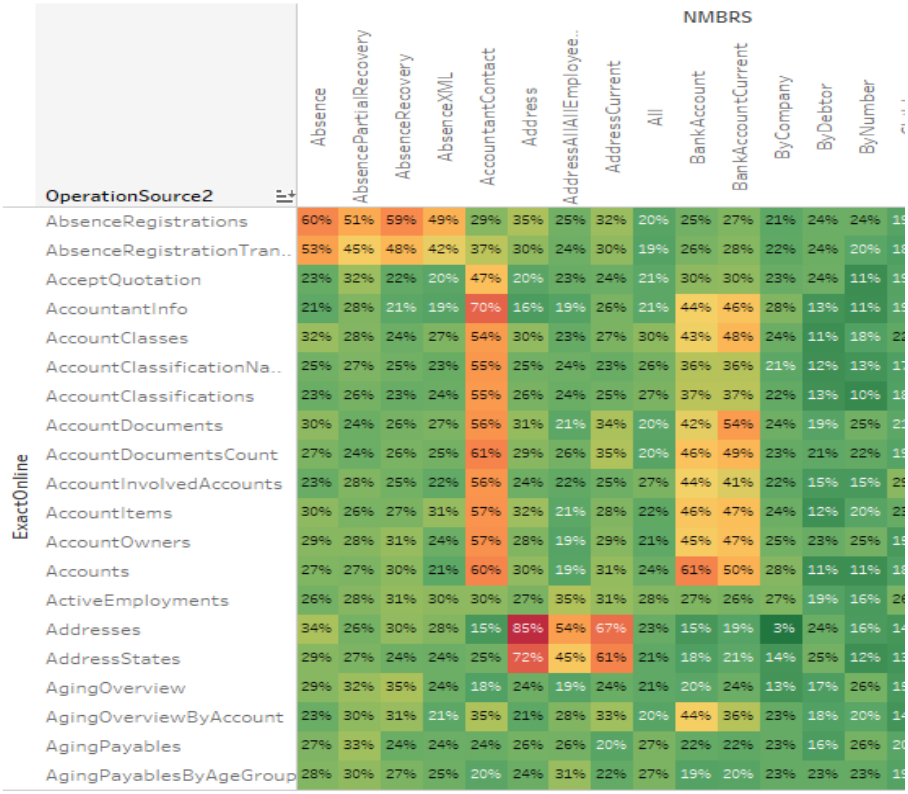
Pav. 21. Operacijų sąveikumo „šilumos žėmėlapis“ naudojant Levenshteino redagavimo nuotolio algoritmą.

Imkime matricą M1, kur vaizduojamas panašumas tarp ExactOnline ir NMBRS sąveikumo vertinimas. Raudona spalva žymimos operacijos, kurių panašumas > 85 % lyginant su kitomis operacijomis (žalia). Raudonos dėmės taip pat parodo didesnę tikimybę, kad operacijos semantiškai panašesnės. Pavyzdžiui, ExactOnline operacija „AbsenceRegistrations“ 60 % panaši į NMBRS operaciją „Absence“. Skaičiavimai yra vaizduojami tik Levenshteino metodu (Pav. 22), tačiau kiti algoritmai taip pat naudoti (Jaccardo, Jaro-Winklerio ir ilgiausios bendros sekos). Padidinus vaizdą M1 matricoje matyti pagrindiniai įverčiai ir operacijos (Pav. 14). Iš galimų 6 taikomųjų programų panašumo vertinimo kombinacijų (M1–M6) naudojant apibendrintą-ansamblio vertinimo metodą visi redagavimo nuotolio skaičiavimai apibendrinti. Iš redagavimo nuotolio algoritmų veikimo galima pateikti tokias išvadas:

1. Jaro-Winklerio ir ilgiausios bendros sekos algoritmai linkę vertinti operacijas vidutiniškai su 50 %, tad pateikia daugiau vidutiniškų rezultatų.

2. Levenshteino algoritmas labiau atskiria kraštutinumus, tačiau taip pat nepateikė daug aukšto vertinimo rezultatų.
3. Jaccardo algoritmas išskyrė labai unikalias operacijas nuo labai artimų operacijų. Algoritmas nebuvo linkęs suteikti panašioms operacijoms geresnių rezultatų už kitus algoritmus.

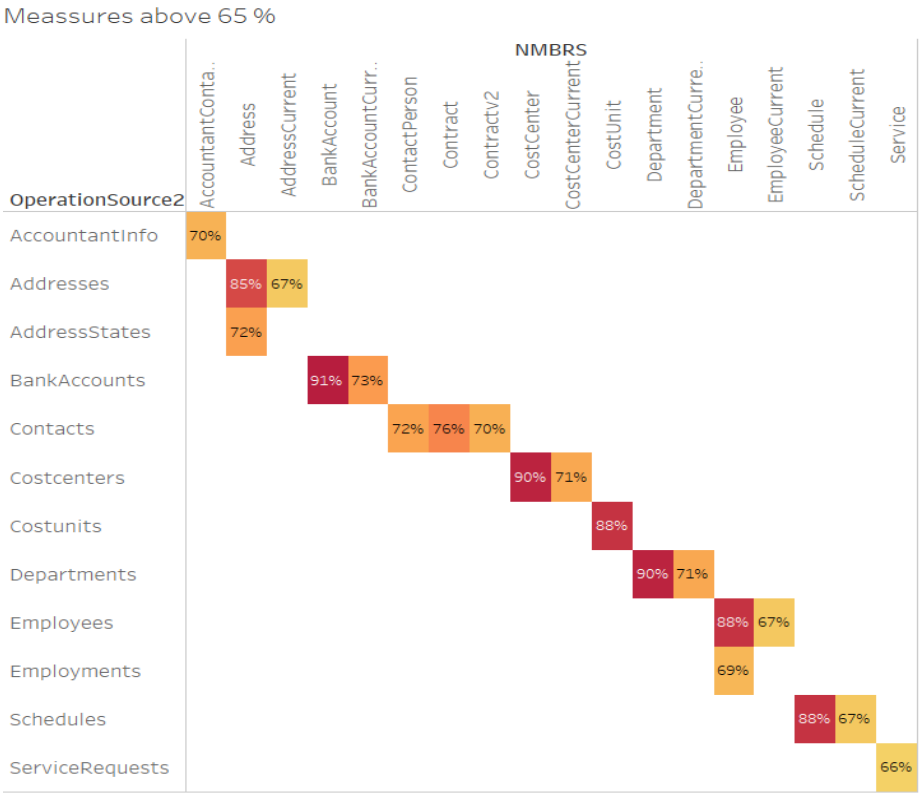
Paveiksle 14 galima išskirti įdomius atvejus – kaip redagavimo nuotolio algoritmas rado panašumų tarp operacijų: Absence ir AbsenceRegistrations (60 %), AccountantInfo ir AccountantContact (70 %), Addresses ir Address (85 %), Accounts ir BankAccount (61 %).



Pav. 22. Sąveikumo vertinimo matrica (M1 matricos dalis) tarp ExactOnline ir NMBRS.

Norint įvertinti tik geriausius atvejus įdomu išanalizuoti vertinimus, didesnius už 65 %. Vertinant visų redagavimo nuotolio algoritmų apibendrintus rezultatus, galima pastebėti, kad ExactOnline ir NMBRS taikomųjų programų operacijos „Accounts“, „absences“ ir „addresses“ panašiausios. ExactOnline (E) ir NMBRS (N) sintaksiškai panašios operacijos: E Addresses ir N Address (85 %); E BankAccounts ir N BankAccount (91 %); E Costcenters ir N CostCenter (90 %); E Costunits ir N

CostUnit (88 %); E Departments ir N Department (90 %); E Employees ir N Employee (88 %); E Schedules ir N Schedule (88 %).



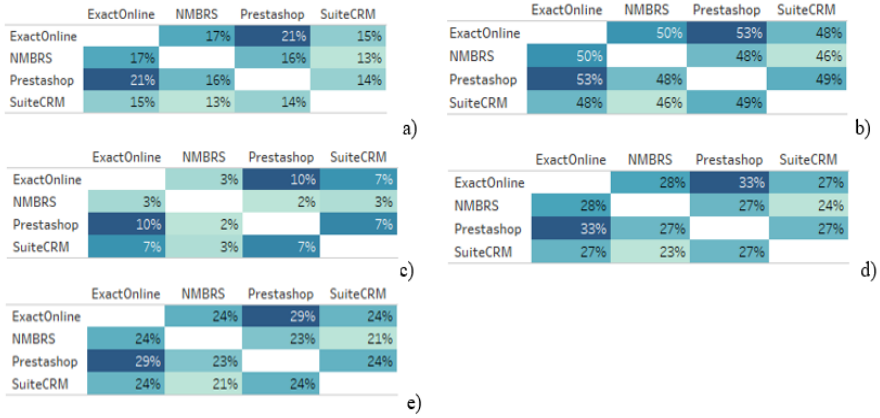
Pav. 23. Matrica M1 apribota operacijoms, kurių panašumas didesnis už 65 %.

Iš rezultatų taip pat matyti (Pav. 23), kad tam tikros taikomųjų programų operacijos taip pat gali būti sumaišytos, pavyzdžiui: „Contacts“ ir „Contracts“ (76 %) ir „Contacts“ ir „ContractsV2“ (70 %), nors semantiškai yra labai skirtingos. Lyginant rezultatus ExactOnline ir Prestashop rasta 20 operacijų su rezultatu, didesniu už 65 %. Galima analizuoti ir nustatyti ribas, pagal semantines reikšmes siekiant atmesti neteisingus lyginimus. Pastaba, kad ExactOnline 293 operacijos ir PrestaShop 49 operacijos turi tik 20 galimų sąveikumo taškų su 65 % jungimo galimybės įverčiu. Taip pat pastebėtina, kad ExactOnline ir Prestashop taikomųjų programų rezultatų analizė pateikė keletą operacijų, kurios 100 % sintaksiškai identiškos (Lentelė 6). Laukai: „Addresses“; „Contacts“; „Currencies“; „Employees“; „Werehouses“ – identiški su 100 % įverčiu. Vertinant panašumą rasta ir klaidingų rezultatų: „Projects“ ir „Products“ (74 %), jie turėtų būti sprendžiami taikant semantinę analizę.

Lentelė 8 Sąveikumo galimybių tyrimo rezultatai – operacijų skaičius pagal panašumo įverčio ribas.

	Panašumas >=		100 %				
	60 %	70 %	Enseble	Levenshtein	Jaro-Winkler	Jaccard	Longest Common Subsequence
ExactOnline X NMBRS (EN)	40	619	-	1	1	-	-
ExactOnline X Prestashop (EP)	54	101	-	-	-	-	-
ExactOnline X SuiteCRM (ES)	48	225	-	8	8	8	8
ExactOnline X KonaKart		356	-	1	1	3	1
ExactOnline X LemonStand		239	-	2	2	2	2
ExactOnline X OpenCart		213	-	3	3	3	3
ExactOnline X MIVA		920	-	-	-	-	-
ExactOnline X Zen Cart		168					
NMBRS X Prestashop (NP)	11	6	1	1	1	1	1
MMBRS X SuiteCRM (NS)	7	-	-	-	-	-	-
SuiteCRM X Prestashop (SP)	13	6	1	1	1	5	1

Lentelėje EP su 70 % panašumo įverčiu rasta 18 operacijų ir 5 operacijos su 100 % redagavimo nuotolio vertinimu. Lyginant visų taikomųjų programų bendrą įvertį pagal visų operacijų panašumų vidurkį galima įvertinti bendrą taikomųjų programų panašumą (Pav. 24).



Pav. 24. Panašumo vertinimas naudojant redagavimo nuotolio algoritmus: a – Levenshteino, b – Jaro-Winklerio, c – Jaccardo, d – Ilgiausios bendros sekos, e – visi metodai

Vertinimo amplitudės (procentiniai įverčiai) tarp redagavimo nuotolio sprendimų skiriasi, todėl skirtingi metodai skirtingai vertina panašias ir nepanašias operacijas, tačiau įdomu, kad nors ir procentiniai vertinimo įverčiai skiriasi, panašiausios sistemos lieka panašios (ExactOnline ir Prestashop).

5.2. Vertinimas tekstinės analizės metodais

Tekstinė šaltinių analizė atliekama siekiant išgryninti ir išvalyti tekstą nuo nepageidaujamų priešdėlių. Duomenų vaizdavimas dažnai padeda priimti kitus sprendimus. Šiame eksperimento etape panaudota žodžių maišų (angl. *Bag of Words*) metodologija siekiant atvaizduoti, kokie gali būti objektų laukų panašumai ir skirtumai. Kartais tenka atskirti objektą nuo operacijos, pavyzdžiui: „sendInvoice“ turi būti atskiriamas į operaciją „send“ ir objektą „invoice“. Iš išskirtų objektų gaunamos vizualizacijos integravimo specialistas gali aproksimuotai įvertinti vartojamų raktažodžių populiarumą ir palyginti su kitomis sistemomis. Pavyzdyje patektos trijų programų KonaKart, Zen Cart ir Suite CRM žodžių maišų diagramos atitinkamai A, B ir C (Pav. 24).

Pav. 25. Tekstinės analizės lyginimas tarp veiklos programų.

Vykdant sąveikumui vertinti skirtą tekstinę analizę pavyko apžvelgti 13 sistemų sandarą. Taip pat lyginant kiekvienos programos sandarą su duomenimis iš „www.schema.org“ galima teigti, kad įmanoma atlikti sistemų semantinį lyginimą.

5.3. Vertinimas latentinės semantinės analizės metodu

Keliama prielaida, kad žodžiai, naudojami sistemose, yra reikšmiškai panašūs, jei pasikartoja panašiose teksto vietose – tai dar vadinama distribucine semantika. Remiantis tokia prielaida galima naudoti latentinį semantinį indeksavimą (angl. *Latent Semantic Indexing*) aptikti reikšmingą panašumą skirtingose programose.

Eksperimentuose naudotas R statistikos paketas, versija 3.5.1. Instaliuotos bibliotekos:

- „RODBC“ – duomenų bazėms prisijungti
- „tm“ – teksto analizės įrankis (angl. *Text Mining*)
- „quanteda“ – teksto analizės įrankis

Testai atliekami naudojant Latent Semantic Analysis“ įrankį iš „quanteda“ bibliotekos. Latentinė semantinė analizė aprašyta Grossman ir Frieder knygoje *Information Retrieval, Algorithms and Heuristics* (Grossman & Frieder, 2012). Latentinės semantinės analizės paskirtis pagal aprašymą – rasti semantinių dokumentų panašumų.

Latentinė semantinė analizė atliekama skaičiuojant termų dažnį kaip santykį tarp termų svorių ir užklausų svorių. Iš termų dažnių matricos randamos vektorių koordinatės sumažintos iki 2 dimensijų. Šios vektorių koordinatės ir yra semantinis atstumas tarp lyginamų duomenų šaltinių.

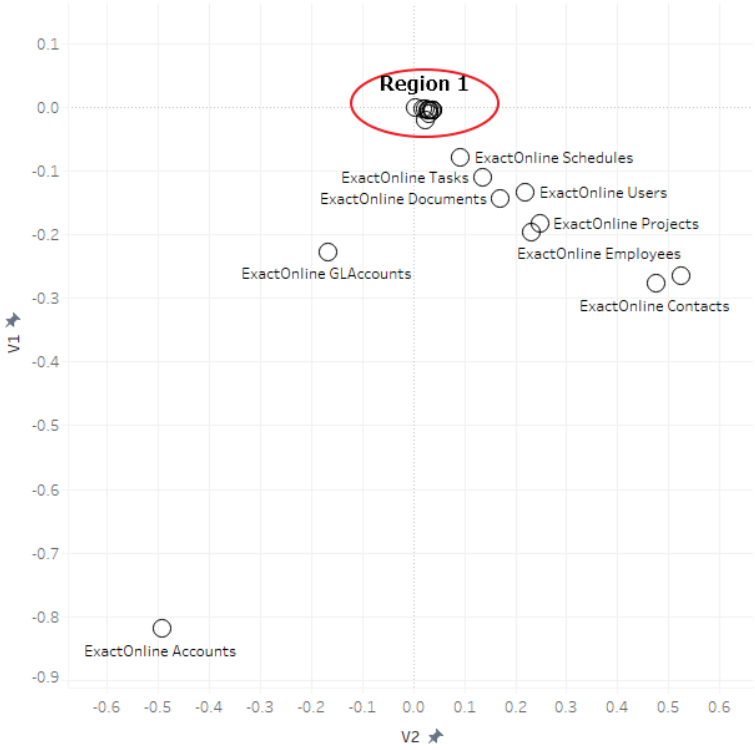
Testas: Visų sistemų, kurių redagavimo nuotolio rezultatas buvo 100 % panašumas, operacijos.

Testo pavadinimas: „Results_LSA_OperationsObjects100List“.

Testo aprašymas: Teste lyginamos ExactOnline ir SuiteCRM programos. Imami tik tie testo duomenys, kuriuose operacijos atitiko 100 % redagavimo nuotolio matavimo rezultatą.

Dokumentų savybių matrica sudaryta iš 6 dokumentų ir 520 savybių, kurios 87,1 % retos. Didelis retumas rodo didelį tų pačių žodžių pasikartojimą ir jis būdingas tik tam tikro tipo dokumentams, šiuo atveju dokumentai – duomenų struktūrų rinkiniai, tad nors savybių daug, jos retos.

Semantinio panašumo tyrimo etape matyti, kad ExactOnline programos objektai išsibarstę labiau už SuiteCRM paketo objektus (Pav. 26).



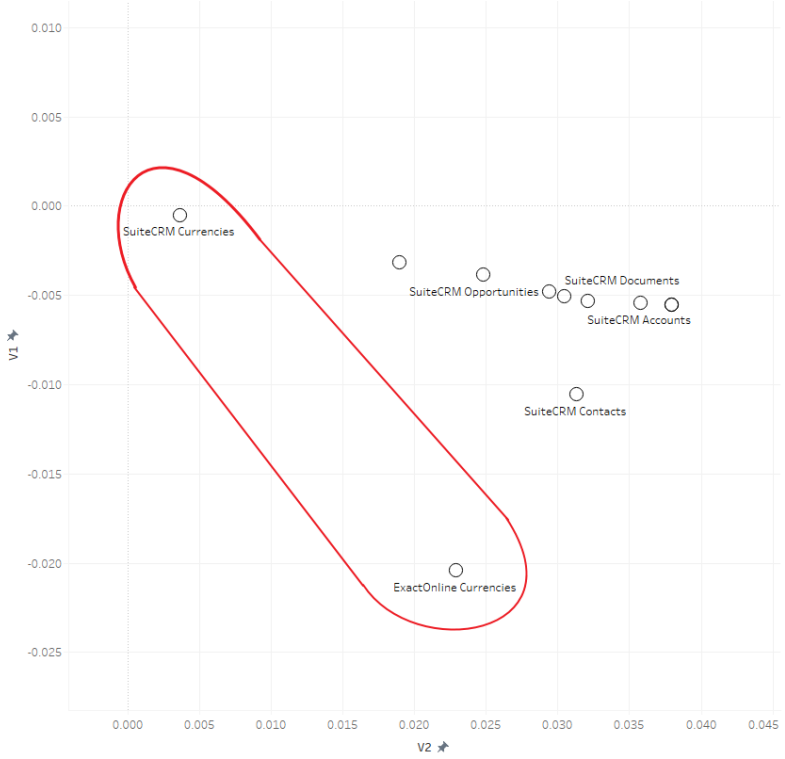
Pav. 26. ExactOnline kartu su SuiteCRM struktūrinis panašumas naudojant LSA metodą.

Šio paveikslo duomenų lentelės, latentinės semantinės analizės vektoriai V1 ir V2 atspindi elementų poziciją plokštumoje. Iš to galima spręsti, kad SuiteCRM objektai yra glaudžiau susiję už ExactOnline objektus, kurie turi mažiau bendrų bruožų per objektą.

Lentelė 9. Dimensijų mažinimo naudojant latentinę semantinę analizę vektoriaus koordinatės 2D plokštumoje (V1, V2).

Programa	V1	V2
ExactOnline Opportunities	-0,1964	0,231243
ExactOnline Accounts	-0,81883	-0,49184
ExactOnline GLAccounts	-0,22777	-0,16639
ExactOnline Users	-0,13468	0,219865
ExactOnline Schedules	-0,0795	0,092412
ExactOnline Employees	-0,26499	0,52448
ExactOnline Projects	-0,18315	0,248085
SuiteCRM Accounts	-0,00545	0,035791
SuiteCRM Contacts	-0,01055	0,031326
SuiteCRM Documents	-0,00508	0,030471
SuiteCRM Tasks	-0,00384	0,02481
SuiteCRM Project	-0,00533	0,032129
SuiteCRM Opportunities	-0,00482	0,029431
ExactOnline Contacts	-0,27745	0,474998
ExactOnline Currencies	-0,02043	0,022884
ExactOnline Tasks	-0,11047	0,137231
ExactOnline Documents	-0,14473	0,17171
SuiteCRM Users	-0,00555	0,037981
SuiteCRM Schedulers	-0,00319	0,019005
SuiteCRM Currencies	-0,00053	0,003633
SuiteCRM Employees	-0,00555	0,037981

Padidinus regioną 1 (Pav. 27) matyti, kad į šį regioną pateko ExactOnline objektas „Currencies“. Tai rodo, kad ExactOnline Currencies turi semantiškai panašių bruožų su šia grupe.



Pav. 27. Region 1 (Pav. 26), skirtingų sistemų panašumas naudojant LSA metodą.

SuiteCRM ir ExactOnline moduliai „currencies“ turi komponentus, kurie turi labai panašius pavadinimus, todėl plokštumoje vaizduojami glaudžiau.

Apibendrinant, tyrimas leidžia nustatyti semantinį panašumą tarp objektų ir jų sudėtinių dalių. Šio tyrimo privalumas – gaunamas atstumo vektorius, leidžiantis patikrinti objektų artumą vieno kitam, tačiau reikia geriau pasiruošti duomenis. Šiame etape reiktų gero rezultatų interpretacijos mechanizmo, kurį būtų galima automatizuoti.

5.4. Skyriaus išvados

Šiame skyriuje atlikti eksperimentai leido patikrinti galimybę skaitiškai ir iš dalies automatizuotu būdu vertinti programų sąveikumo galimybes. Apžvelgtos eksperimente pasirinktos programos sąveikumui vertinti. Nubrėžta koncepcinė diagrama (Pav. 20) vaizduoja pagrindinius tyrimo aspektus.

Atlikta LISI analizė nepateikia gerų įžvalgų tolimesnei ar automatizuotai analizei vykdyti. I – Score metodo nebuvo galima pritaikyti, nes metodas nekreipia dėmesio į programų architektūrą. Lyginamosios sistemų analizės n

nebuvo galima pritaikyti, nes taip surinkta informacija neleidžia atlikti sąveikumo galimybių vertinimo automatizuotu būdu.

Redagavimo nuotolio analizė tinka pavienėms operacijoms, objektams ir jų laukams lyginti. Redagavimo nuotolis daugiausiai parodo sintaksinius panašumus, bet tam tikrais atvejais, kai objektas ir jo laukai iš skirtingų sistemų atitinka vieni kitus, parodo semantinius panašumus ir skirtumus.

Žodžių maišų analizė leidžia iškirti kontekstui reikšmingiausius žodžius ir identifikuoti sistemos tipą pagal raktinius žodžius. Bet vargu ar duomenis galima pritaikyti sprendimui automatizuoti.

Latentinės semantinės analizės metodas leidžia patikslinti redagavimo nuotolio analizėje gautas reikšmes ir paryškinti, atrasti semantinius panašumus tarp programos objektų ir jų laukų.

6. IŠVADOS

Šioje disertacijoje nagrinėtos verslo veiklos programų integracijos ir sąveikumo problemos. Teorinėje dalyje apžvelgti ir išanalizuoti esami verslo veiklos programų integracijos ir sąveikumo sprendimai, technologijos ir metodika. Pasiūlytas kompleksinis programų sąveikumo matavimo sprendimas, apimantis kelias skirtingas metodologijas: verslo veiklos procesų modeliavimą (BPM), modeliais grindžiamą architektūrą (MDA), autonominių skaičiavimų technologiją ir verslo veiklos programų sąveikumo galimybių vertinimą. Aprašomo metodo dalis (galimybė įvertinti programų sąveikumo galimybes) ginama eksperimentu atliekant redagavimo nuotolio ir latentinę semantinę analizę. Praktinėje dalyje suformuotas ir aprašytas eksperimentas leidžia patvirtinti galimybę automatizuotai įvertinti verslo programų sąveikumo galimybes.

6.1. Teorinės dalies išvados

1. Išsikeltas darbo tikslas „nustatyti taikomųjų programų sąveikumo įvertinimo principus ir sukurti sąveikumo įvertinimo automatizavimo metodą, kuris grindžiamas skirtingų technologijų (MDA, BPM, autonominio skaičiavimo technologija ir redagavimo nuotolio bei latentinės semantinės analizės metodų) sujungimu“ įgyvendintas ir detaliai aprašytas 3 skyriuje.
2. Sukurtas ir aprašytas programų sąveikumo vertinimo sprendimas siūlo naudotis autonominio skaičiavimo technologija ir programų architektūra taikomųjų programų sąveikumo analizei.
3. Ne taip, kaip kiti ištirti metodai, šis sprendimas leidžia analizuoti: savęs valdymo, redagavimo, optimizavimo ir pasitaisymo galimybes turinčius autonominius sprendimus programų sąveikumo sprendimų kūrimo srityje. Šio sprendimo rezultatas leidžia matuoti daugelio taikomųjų programų architektūrinius panašumus.
4. Pateiktas sprendimas jungia organizacijos architektūros karkasus (angl. *Enterprise Architecture Frameworks*), modeliais grindžiamos architektūros (MDA) požiūrį (CIM lygmens modelius, PIM lygmens modelius) ir autonominio skaičiavimo technologiją sąveikumo kiekybiniam vertinimui.
5. Lyginti panašumai tarp elementaraus valdymo ciklo modelio ir autonominio valdymo komponento. Atrasti panašumai tarp šių

dviejų skirtingų sričių: veiklos modelių ir programinės įrangos modelių. Dėl jų panašumų autonominio skaičiavimo metodai gali būti naudojami analizuojant ir integruojant veiklos procesus ir juos valdančius objektus (taikomoji programa).

6. Taikomųjų programų modelių siejimas su veiklos procesų sluoksnio modeliais leidžia gauti tikslią informaciją, reikalingą automatizuojant taikomųjų programų sąveikumo sprendimo kūrimą.

Siūlomu metodu galime pamatuoti (įvertinti) daugelio taikomųjų programų struktūrinius panašumus, o turint matavimus galime planuoti ir atlikti tikslesnes sąveikumo operacijas.

6.2. Praktinės dalies išvados

1. Gauta kontekstinė programos tinklo paslaugų aprašymo informacija ir išanalizuota išskiriant: operacijas, objektus, laukus, laukų tipus ir laukų kardinalumą. Ši galimybė automatizuotai gauti sistemos architektūros aprašus leidžia plačiau pažvelgti į galimybes šiuos duomenis analizuoti įvairiais metodais.
2. Gavus programų sąveikumo įverčius nustatyta, kad metodas atitinka autonominio skaičiavimo principo stebėjimo ir analizės (angl: *Monitor* (M), *Analyze* (A) – Pav. 9) žingsnius.
3. Eksperimentinis tyrimas patvirtina taikomųjų programų sąveikumo įvertinimo metodo veiksnumą, nes leidžia skaitiškai įvertinti skirtingų sistemų panašumą. Ištirti pavyzdžiai nustato taikomųjų programų struktūrų panašumus ir sąsajas. Pasiūlytu metodu įvertinami panašumai ir skirtumai.
4. Dėl plačios tyrimo srities nepavyko sukurti visiškai autonominio skaičiavimo pagrindu veikiančio sąveikumo sprendimo. Tačiau užsibrėžti uždaviniai įgyvendinti ir tikslas pasiektas. Šiuo metu sprendimas yra dalinis ir apima tik stebėjimo ir analizės žingsnius autonominėje skaičiavimo technologijoje (Pav. 9).

Pasiūlytas sprendimas remiasi redagavimo nuotolio skaičiavimo metodu siekiant įvertinti programų sąveikumo galimybę ir galbūt ją automatizuoti. Tai taip pat leistų pagerinti programų integracijos ir sąveikumo sprendimo kūrimo procesus. Šiai sričiai reikalingi kompetentingi ir gerai skirtingas taikomąsias programas išmanantys ekspertai, tačiau šioje

disertacijoje užsimenama ir apie galimybę šias ekspertines žinias generuoti ir kaupti pačiam autonominio skaičiavimo metodui. Nors šios disertacijos eksperimente nebuvo visiškai įgyvendinti bent vieno iš autonominio skaičiavimo principu valdomų ciklų, tačiau pradžia padaryta. Išmokus automatizuoti integracijos metodus, tai būtų labai didelė pažanga informacijos apdorojimo ir sklaidos srityje.

Bibliografija

Basson, H. et al., 2016. Qualitative evaluation of manufacturing software units interoperability using ISO 25000 quality model. *Enterprise interoperability*, VII tomas, pp. 199-209.

Boxer, P. & Kenny, V., 1990. The economy of discourses: a third order cybernetics?. *Human Systems Management*, 4(9), pp. 205-224.

CEITON technologies, 2015. *Front-end and back-end EAI.* [Tinkle] [Kreiptasi 28 07 2015].

Chen, D., 2006. Enterprise Interoperability Framework. *EMOI-INTEROP*.

Chen, D., Doumeingts, G. & Vernadat, F., 2008. Architectures for enterprise integration and interoperability: Past, present and future. *Computers in industry*, 59(7), pp. 647-659.

Cintuglu, M. H., Youssef, T. & Mohammed, O. A., 2016. Development and application of a real-time testbed for multiagent system interoperability: A case study on hierarchical microgrid control. *IEEE Transactions on Smart Grid*, 9(3), pp. 1759-1768.

David, C., 2006. *Enterprise Interoperability Framework.* s.l., EMOI-INTEROP.

Dijkman, R. et al., 2014. Similarity of business process models: Metrics and evaluation.. *Information Systems*, 36(2), pp. 498-516.

Di, L. & Kobler, B., 2000. NASA standards for earth remote sensing data. *International Archives of Photogrammetry and Remote Sensing, 33(B2; PART 2)*, pp. 147-155.

Dong, X. L. & Divesh, S., 2013. *Big data integration.* s.l., IEEE, pp. 1245-1248.

Dzemydienė, D. & Naujikienė, R., 2009. Elektroninių viešųjų paslaugų naudojimo ir informacinių sistemų sąveikumo vertinimas. *Informacijos mokslai*, Issue 50, pp. 267-273.

El-Halwagi, M. M., 2006. *Process integration.* s.l.:s.n.

Euzenat, J. et al., 2010. Results of the ontology alignment evaluation initiative 2009. *University of Trento*.

Fielding, R. T. & Taylor, R. N., 2000. *Architectural styles and the design of network-based software architectures.* Irvine: University of California.

Ford, T., Colombi, J., Graham, S. & Jacques, D., 2008. *Measuring system interoperability.* s.l., Proceeding Cser.

Gartner, 2019. *IT Glossary*. [Tinkle] Available at: https://www.gartner.com/it-glossary/ [Kreiptasi 16 02 2019].

Gates, B., 2013. *Measuring progress. Annual Letter, Gates Foundation.* [Tinkle]
Available at: https://www.gatesfoundation.org/Who-We-Are/Resources-and-Media/Annual-Letters-List/Annual-Letter-2013
[Kreiptasi 25 September 2018].

Gonzalo, N., 2001. A guided tour to approximate string matching.. *ACM computing surveys (CSUR),* 1(33), pp. 31-88.

Gricius, G., 2015. *Daugiaagentinių sistemų kūrimo metodų išvystymas nedidelio našumo įterptinių sistemų integravimui,* Vilnius: Vilnius University.

Grossman, D. A. & Frieder, O., 2012. Information retrieval: Algorithms and heuristics. *Springer Science & Business Media,* 11.15 tomas.

Gudas, S., 2012. Knowledge-Based Enterprise Framework: A Management Control View. *New Research on Knowledge Management Models and Methods,* pp. 179-217.

Gudas, S. & Hou, H., 2012. Knowledge-based enterprise framework: a management control view. *New Research on Knowledge Management Models and Methods,* pp. 179-218.

Gudas, S. & Valatavicius, A., 2017. Normalization of Domain Modeling in Enterprise Software Development. *Baltic Journal of Modern Computing,* 5(4), pp. 329-350.

Guédria, W., Naudet, Y. & Chen, D., 2008. Interoperability maturity models–survey and comparison–. *OTM Confederated International Conferences" On the Move to Meaningful Internet Systems,* 11.pp. 273-282.

Halevy, A. et al., 2005. *Enterprise information integration: successes, challenges and controversies.* s.l., ACM, pp. 778-787.

Heylighen, F. & Joslyn, C., 2001. Cybernetics and second-order cybernetics. *Encyclopedia of physical science & technology,* Issue 4, pp. 155-170.

Hohpe, G. & Woolf, B., 2004. *Enterprise integration patterns: Designing, building, and deploying messaging solutions.* s.l.:Addison-Wesley Professional.

Horn, P., 2001. *Autonomic computing: IBM's perspective on the state of information technology.* New York, IBM T.J. Watson Labs.

IBM, 2006. *An architectural blueprint for autonomic computing.* [Tinkle] Available at: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.150.1011&rep=rep1&type=pdf
[Kreiptasi 12 02 2019].

IDABC, 2004. *European interoperability framework for pan-european eGovernment services.* [Tinkle]

Available at: http://ec.europa.eu/idabc/servlets/Docd552.pdf [Kreiptasi 16 02 2019].

International Organisation for Standartisation, 2015. *ISO/IEC 2382:2015 Information technology - Vocabulary.* [Tinkle] Available at: https://www.iso.org/obp/ui/#iso:std:iso-iec:2382:ed-1:v1:en [Kreiptasi 04 02 2019].

ISA2, 2017. *Interoperability solutions for public administrations, businesses and citizens.* [Tinkle] Available at: https://ec.europa.eu/isa2/sites/isa/files/eif_brochure_final.pdf [Kreiptasi 29 04 2019].

Jacob, B., Lanyon-Hogg, R., Nadgir, D. & Yassin, A., 2004. *A practical guide to the IBM autonomic computing toolkit. IBM*, Durham: International Technical Support Organization.

Kasunic, M., 2001. *Measuring systems interoperability: Challenges and opportunities.* s.l.:Carnegie-Mellon Univ Pittsburgh Pa Software Engineering Inst.

Kephart, J. O. & Chess, D. M., 2003. The vision of autonomic computing. *Computer*, 1 tomas, pp. 41-50.

Krafzig, D., Banke, K. & Slama, D., 2005. *Enterprise SOA: service-oriented architecture best practices.* s.l.:Prentice Hall Professional.

Krajicek, J. & Krajíček, J., 1995. *Bounded arithmetic, propositional logic and complexity theory*, Cambridge: Cambridge University Press.

Kutsche, R.-D. & Milanovic, N., 2008. *Model-Based Software and Data Integration: First International Workshop.* Berlin, Springer Science & Business Media.

Li, L., Wu, B. & Yang, Y., 2005. *Agent-based Ontology Integration for Ontology-based Applications.* Proceedings of the 2005 Australasian Ontology Workshop-Volume 58, Australian Computer Society, Inc..

Lin, P., MacArthur, A. & Leaney, J., 2005. Defining autonomic computing: A software engineering perspective. *Australian Software Engineering Conference*, pp. 88-97.

Linthicum, D. S., 2000. *Enterprise application integration.* s.l.:Addison-Wesley Professional.

Liu, S., Li, W. & Liu, K., 2014. Assessing pragmatic interoperability of information systems from a semiotic perspective. *International Conference on Informatics and Semiotics in Organisations*, pp. 32-41.

Mancilla, R., 2011. Introduction to sociocybernetics (Part 1): Third order cybernetics and a basic framework for society.. *Journal of Sociocybernetics*, 9(42), pp. 35-56.

McCann, R. et al., 2005. *Mapping maintenance for data integration systems.* s.l., VLDB Endowment, pp. 1018-1029.

Morkevičius, A., 2013. *Business and information systems alignment method based on enterprise architecture models,* Kaunas: KAUNAS UNIVERSITY OF TECHNOLOGY.

Navarro, G., 2001. *A guided tour to approximate string matching.* s.l., ACM computing surveys (CSUR), pp. 31-88.

Novikov, D. A., 2016. CYBERNETICS 2.0.. *Advances in Systems Science and Applications*, 16(1), pp. 1-18.

Overeinder, B. J. & Verkaik, P. D. B. F. M., 2008. Web service access management for integration with agent systems. *Proceedings of the 2008 ACM symposium on Applied computing*, pp. 1854-1860.

Panetto, H. & Molina, A., 2008. Enterprise integration and interoperability in manufacturing systems: Trends and issues.. 7(59), pp. 641-646.

Papazoglou, M., 2008. *eb services: principles and technology. Pearson Education.* s.l.:Pearson Education.

Pavlin, G., Kamermans, M. & Scafeş, M., 2009. Dynamic process integration framework: Toward efficient information processing in complex distributed systems. *Intelligent Distributed Computing III*, pp. 161-174.

Peng, Y. et al., 1998. A multi-agent system for enterprise integration. *International Journal of Agile Manufacturing*, 2(1), pp. 213-229.

Peukert, E. & Eberius Julian, R. E., 2011. *AMC-A framework for modelling and comparing matching systems as matching processes.* s.l., IEEE, pp. 1304-1307.

Peukert, E., Eberius, J. & Rahm, E., 2012. *A self-configuring schema matching system.* s.l., In 2012 IEEE 28th International Conference on Data Engineering, pp. 306-317.

Rahm, E. & Bernstein, P. A., 2001. A survey of approaches to automatic schema matching. *The VLDB Journal*, 4(10), pp. 334-350.

Reijers, H. A. & Mansar, S. L., 2005. Best practices in business process redesign: an overview and qualitative evaluation of successful redesign heuristics. *Omega*, 33(4), pp. 283-306.

Rezaei, R., Chiew, T. K. & Lee, S. P., 2014. A review on E-business Interoperability Frameworks. *Journal of Systems and Software*, 93 tomas, pp. 199-216.

Sahlgren, M., 2008. The distributional hypothesis. *Italian Journal of Disability Studies*, Issue 20, pp. 33-53.

Savulionienė, L., 2014. *Susietumo taisyklių paieška didelėse duomenų bazėse,* Vilnius: Vilnius University.

Scott, B., 2017. *Chiefmartec.* [Tinkle] Available at: https://chiefmartec.com/2017/06/average-enterprise-uses-91-marketing-cloud-services/

Shvaiko, P. & Euzenat, J., 2011. Ontology matching: state of the art and future challenges. *IEEE Transactions on knowledge and data engineering,* 1(25), pp. 158-176.

Silverston, L., 2011. *The data model resource book, Volume 1: A library of universal data models for all enterprises.*. s.l.:John Wiley & Sons.

Silverston, L. & Inmon, W. H. ,. G. K., 1997. *The data model resource book: a library of logical data models and data warehouse designs.* s.l.:John Wiley & Sons, Inc..

Silverston, L., Inmon, W. H. & Graziano, K., 1997. *he data model resource book: a library of logical data models and data warehouse designs.* s.l.:John Wiley & Sons, Inc.

Sitton, M. & Reich, Y., 2016. Enterprise Systems Engineering for Improving Cross-enterprise Effectiveness. *INCOSE International Symposium,* 26(1), pp. 2085-2100.

Tan, S., Liu, K. & Xie, Z., 2004. *A Semiotic approach to organisational modelling using norm analysis.* s.l., s.n., pp. 1-15.

Tolk, A., 2003. Beyond technical interoperability-introducing a reference model for measures of merit for coalition interoperability. *Old Dominion Univ Norfolk VA.*

Tolk, A. & Muguira, J. A., 2003. *The levels of conceptual interoperability model.* s.l., Citeseer, pp. 1-11.

Trotta, G., 2003. Dancing around EAI 'bear traps'. *Business Process Management (BPM) Best Practices.*

Valatavičius, A. & Dilijonas, D., 2014. *Dinaminė „B2B“ procesų integracija.* s.l., Technologija, pp. 34-39.

Valatavičius, A. G. S., 2017. Apie taikomųjų programų sąveikumo metodologiją, grindžiamą giluminėmis žiniomis. *Informacijos mokslai,* pp. 83-113.

Valatavičius, A. & Gudas, S., 2015. *Enterprise software system integration using autonomic computing.* Tartu, CEUR-WS, pp. 156-163.

Valatavičius, A. & Gudas, S., 2015. *Towards business process integration using autonomic computing.* Kaunas, Technologija, pp. 81-81.

Valatavičius, A. & Gudas, S., 2017. *Advanced evaluation methods of multiple application software interoperability*. Vilnius, Vilniaus Universitetas, p. 52.

Valatavičius, A. & Gudas, S., 2018. *A deep knowledge based evaluation of applications interoperability*. Druskininkai, Vilniaus Univesitetas.

Valatavičius, A. & Gudas, S., 2018. *Measuring Enterprise Application Software Interoperability Capability.* Estonia, CEUR-WS, pp. 104-113.

Van der Bosch, M. A., van Steenbergen, M. E., Lamaitre, M. & Bos, R., 2010. A selection-method for Enterprise Application Integration solutions. *International Conference on Business Informatics Research,* pp. 176-187.

Villányi, B. & Martinek, P., 2015. Improved Accuracy Evaluation of Schema Matching Algorithms. *Acta Polytechnica Hungarica,* 6(12).

Walsh, A. E., 2002. *Uddi, Soap, and WSDL: the web services specification reference book.* s.l.:Prentice Hall Professional Technical Reference.

Wikipedia, 2019. *Comparison of shopping cart software.* [Tinkle] Available at: https://en.wikipedia.org/wiki/Comparison_of_shopping_cart_software

Zinnikus, I., Hahn, C. & Fischer, K., 2008. A model-driven, agent-based approach for the integration of services into a collaborative business process.. *International Foundation for Autonomous Agents and Multiagent Systems,* 1 tomas, pp. 241-248.

Левенштейн, В. И., 1965. Двоичные коды с исправлением выпадений, вставок и замещений символов. *Доклады Академии наук,* 163(4), pp. 845-848.

1. PRIEDAS – Tiriamų objektų sąrašas

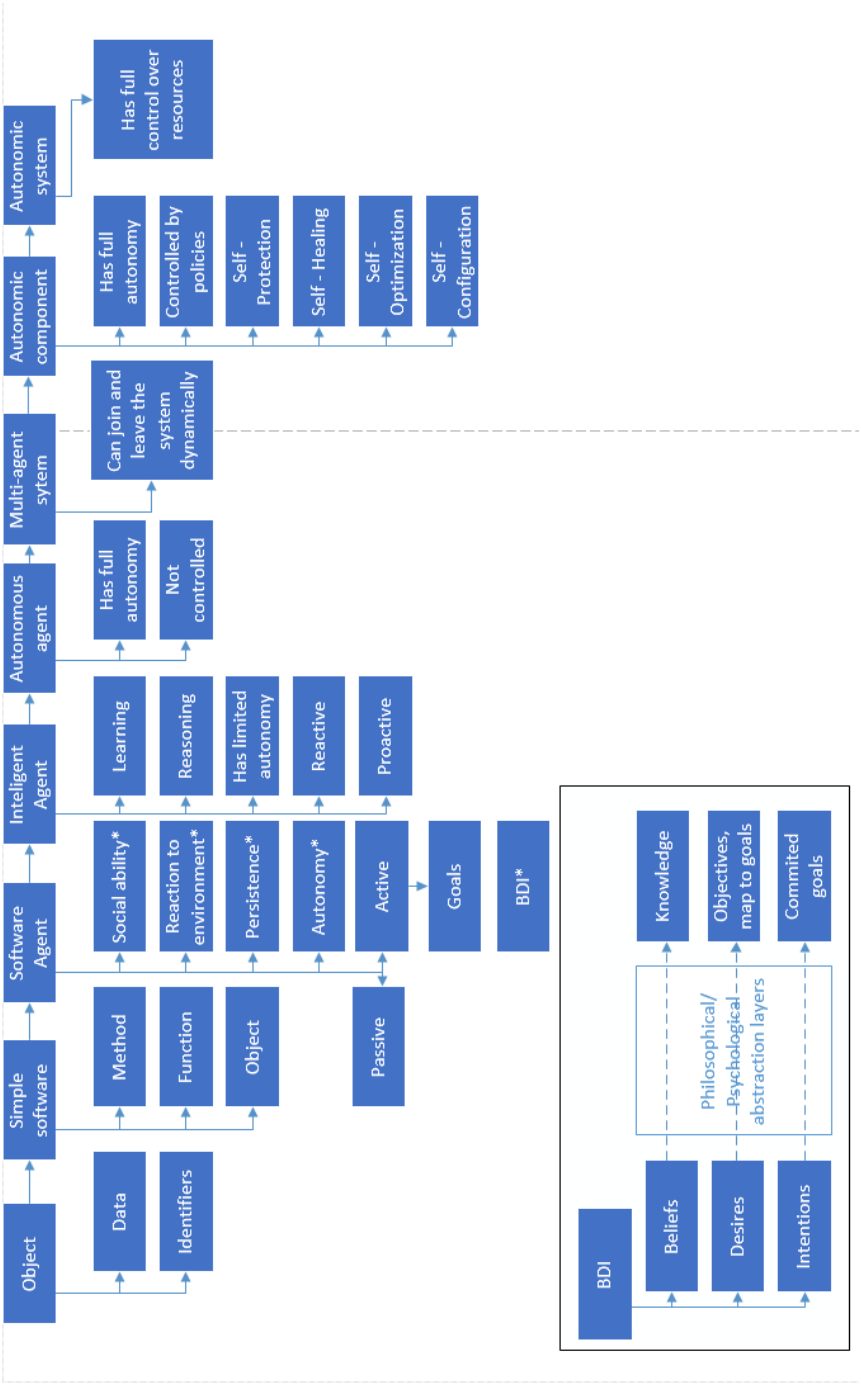
Verslo veiklos programų tipai:

1. Buhalterijos
2. Elektroninių mokėjimų ir sąskaitų išrašymo
3. Verslo analitikos

Taikomoji programa	**Tipas EN**	**PROTOCOL**	**Licencija**
SuiteCRM	CRM	SOAP	
NMBRS	HR-Payrol	SOAP	
ExactOnline	ERP	REST	
PrestaShop	E-Commerce		
KonaKart	E-Commerce	SOAP	GNU LGPL
LemonStand	E-Commerce	REST	Proprietary
Magento	E-Commerce	SOAP, REST	OSL 3.0
Miva	E-Commerce	REST	Proprietary
NopCommerce	E-Commerce	REST	GPL
OpenCart	E-Commerce	REST	GPL
OsCommerce	E-Commerce	-	GNU GPL
Pimcore	E-Commerce	REST	BSD
Sana	E-Commerce	REST	Proprietary
Shopify	E-Commerce	REST	Proprietary
Spree Commerce	E-Commerce	REST	BSD
Storehippo	E-Commerce	REST	Proprietary
SupaDupa	E-Commerce	NA	Proprietary
uCoz	E-Commerce	NA	Proprietary
VirtoCommerce	E-Commerce	REST	OSL 3.0
VirtueMart	E-Commerce	REST	GPL
WooCommerce	E-Commerce	REST	GPL
Zen Cart	E-Commerce	REST	GPL
Apache OFBiz	E-Commerce	REST	Apache
Batavi	E-Commerce	NA	GPL
BigCommerce	E-Commerce	REST	Proprietary
Drupal	TVS	REST	GPL
Act!	CRM	REST	Proprietary
Adempiere	CRM		GPL
Base CRM	CRM		SaaS
CiviCRM	CRM		AGPL
Dolibarr	CRM		GPL, SaaS
Dynamics CRM	CRM		Proprietary

Epesi CRM	CRM		MIT
GNU Enterprise	CRM		GPL
Group-Office	CRM		AGPL
HubSpot CRM Free	CRM		SaaS
Neolane	CRM		Proprietary
Nutshell CRM	CRM		SaaS
Pega CRM	CRM		Proprietary
Pipedrive	CRM		SaaS
Pivotal CRM	CRM		Proprietary
Really Simple Systems	CRM		SaaS
SageCRM	CRM		Proprietary / SaaS
Salesbox CRM	CRM		SaaS
Salesforce.com	CRM		Proprietary
Streak	CRM		Proprietary
SugarCRM	CRM		AGPL
SuperOffice CRM	CRM		Proprietary
TeamLab	CRM		GPL
TeamWox	CRM		Proprietary
Tryton	CRM		GPL
WORKetc	CRM		SaaS
Zoho CRM	CRM		SaaS

2. PRIEDAS – Programų ir taikomųjų programų sudėtingumo lyginimas



UŽRAŠAMS

UŽRAŠAMS

UŽRAŠAMS

Andrius Valatavičius

TAIKOMŲJŲ PROGRAMŲ SĄVEIKUMO VERTINIMAS TAIKANT AUTONOMINIO SKAIČIAVIMO TECHNOLOGIJAS

Daktaro disertacijos santrauka
Gamtos mokslai
Informatika (N 009)
Redaktorė Jorūnė Rimeisytė - Nekrašienė

Vilniaus universiteto leidykla
Saulėtekio al. 9, LT-10222 Vilnius
El. p. info@leidykla.vu.lt,
www.leidykla.vu.lt
Tiražas 20 egz.