VILNIAUS UNIVERSITETAS

ANDRIUS VALATAVIČIUS

TAIKOMŲJŲ PROGRAMŲ SĄVEIKUMO VERTINIMAS TAIKANT AUTONOMINIO SKAIČIAVIMO TECHNOLOGIJAS

Daktaro disertacija,

Fiziniai mokslai, Informatika (09P)

Vilnius, 2018

Disertacija rengta 2014-2018 metais Vilniaus universiteto Matematikos ir Informatikos institute.

Mokslinis vadovas:

Prof. dr. Saulius Gudas (Vilniaus universitetas, fiziniai mokslai, informatika – 09 P)

PADĖKA

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

Nuoširdžiai dėkoju savo šeimos nariams už palaikymą net ir sunkiausiomis akimirkomis motyvavusius mane judėti į priekį ir nepasiduoti.

Labai dėkoju 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.

Andrius Valatavičius

SANTRAUKA

Organizacijų veiklos programinė įranga nuolat plečiasi ir tobulėja, bet dažnai nepakanka vieno taikomosios programos paketo visiems organizacijos veiklos procesams padengti. Organizacijų taikomoji programinė įranga, tai skaitmeninės programos padedančios vykdyti ir valdyti organizacijų veiklą, pavyzdžiui valdyti ryšius su klientais (angl. Customer Relationship Management, CRM), valdyti elektroninę parduotuvę (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 ir dažnai nėra galimybės jų integruoti dėl skirtingų šių programų gamintojų. Todėl programos turi sąveikauti tarpusavyje dėl verslo veiklos procesų palaikymo – duomenų mainai ir funkcijų vykdymas yra esminė sąveikaujančių programų paskirtis.

Paveikslų sąrašas

Lentelių sąrašas

Žymėjimai

E	Veikla (angl. Enterprise)
S	Sistema
O	Objektas - sistemos dedamasis elementas, esybė (angl. Object)
h	Schema (objekto duomenų stuktūros aprašas)
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 (viršijus – laikoma kad sistemos panašios)

Sąvokos

Modulis	Produktas (įrengimas, programinės įrangos paketas, integruojančiuoji infrastruktūra ir pan.), kurį tikslinga naudoti konkretaus proceso realizavime [32].
Modelis	Tai abstrakti konstrukcija, kuria mėginama atkartoti kai kurias realios sistemos savybes.
Procesas	Darbo srautą organizacinėje sistemoje tarp išorinio tiekėjo bei išorinio vartotojo nurodantis didžiausias veiklos vienetas [32].
Konceptas	Abstrakti idėja atspindinti bazines (pagrindines) charakteristikas apie tai ką ji reprezentuoja. Pvz.: Medžio konceptas yra medžio savybių generalizavimas.
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.
Interfeisas (sąsaja)	Tai prieiga prie valdomo išteklio, tiksliau sistemos arba serverio. Sąsajos leidžia valdyti išteklius per sensorius ir efektorius (angl. effectors).
Tinklo servisas	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 servisai leidžia apjungti skirtingas aplikacijas (programines sistemas) nepaisant techninės aplinkos apribojimų.
Veikla	Tai įmonė, organizacija arba stambi veikla naudojanti programų sistemas savo procesuose (angl. Enterprise).
Taikomoji programa	Tai veiklos programų sistemos skirtos valdyti veiklos išteklius ir procesus (angl. Enterprise applications) ir (angl. Enterprise systems).
Taikomųjų programų integracija	Skirtingų programinių komponentų apjungimas į bendrą visumą siekiant sukurti vieningą sistemą.
Sąveikumas	Skirtingų taikomųjų programų komponentų bendradarbiavimas: mainymasis duomenimis, funkcijomis.
Organizacijų architektūros karkasai	(angl. Enterprise architecture framework)
Organizacijų taikomųjų programų integracija	Organizacijų taikomųjų programų integravimas (angl. enterprise application integration - EAI)

Santrumpos

ICT – informacijos ir komunikacijos technologijos (angl. Information and communications technology)

OAuth – atviro standarto prieigos delegavimas.

TVS – Turinio valdymo sistema (angl. content management system)

EVC – Elementarus valdymo ciklas (angl. EMC – elementary management cycle)

ASVC – Autonominio skaičiavimo valdymo ciklas (angl. autonomic computing control loop)

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

TURINYS

ĮVADAS

Organizacijų veiklos programinė įranga nuolat plečiasi ir tobulėja, bet dažnai nepakanka vieno taikomosios programos paketo visiems organizacijos veiklos procesams padengti. Organizacijų taikomoji programinė įranga, tai skaitmeninės programos padedančios vykdyti ir valdyti organizacijų veiklą, pavyzdžiui valdyti ryšius su klientais (angl. Customer Relationship Management, CRM), valdyti elektroninę parduotuvę (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 ir dažnai nėra galimybės jų integruoti dėl skirtingų šių programų gamintojų. Programos turi sąveikauti tarpusavyje dėl verslo veiklos procesų palaikymo – duomenų mainai ir funkcijų vykdymas yra esminė sąveikaujančių programų paskirtis.

Įvairaus dydžio organizacijų veikloje 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 saveikumo sprendimų kūrimo galimybių įvertinimą. Taikomųjų programų sąveikumo problema yra aktuali problema moksliniu ir praktiniu požiūriais. Verslo taikomųjų programų sąveikumo plėtrai būtina sukurti sąveikumo vertinimo metodus, kurie įgalintų bent dalinai automatizuoti analitiko, integracijos architekto ir programuotojo darbo aplinką.

Darbe pateikiamas sąveikumo galimybių vertinimo sprendimas panaudojant programų tinklo servisus ir veiklos procesų modelius.

Sąvokų apibrėžimai : integravimas, dinaminis integravimas, sąveikumas.

Verslo taikomųjų programų integravimas apibrėžiamas kaip programų sujungimas laikant jų duomenis bendroje duomenų bazėje ir duomnų architektūra gali būti nepriklausoma nuo aplikacijų architektūros arba tai sistemos architektūra leidžianti sistemos vartotojui pasiekti visas funkcijas per vieną vartotojo sąsają (Gartner žodynas) [43].

Verslo taikomųjų programų saveikumas apibrėžiamas kaip prietaiso ar programos pagamintų skirtingų gamintojų galimybė dirbti kartu ir mainytis duomenimis [43].

Veiklos programos integravimas (EAI, angl., enterprise application integration) tai proramos arba kompiuterizuotos sistemos architektūriniai principai integruojantys kompiuterizuotas veiklos programas. Veiklos programų integravimas dažnai atliekamas naudojantis technologijomis ir servisais kurie suformuoja tarpines-applikacijas (angl., middleware) leidžiančias atlikti atskirtų (galimai nutolusių, nesinaudojančių bendra duomenų baze) kompiuterizuotų sistemų ir programų integraciją. Veiklos programų integravimas, tai procesas kurio metu šios programos apjungiamos sukuriant vieną struktūrą siekiant supaprastinti ir galimai automatizuoti veiklos procesus, taip pat siekiant išvengti IT ir BP (angl. business process) reinžinerijos. Apjungiamos programos gali būti susiejamos per duomenų bazes, programos interfeisus API, tinklo servisus, arba (bet rečiau) per vartotojo sąsają (GUI) [49].

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, analizuojantys 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 [21], [27], [36, 37]. Dėl šios priežasties mokslinėje bendruomenėje nagrinėjamos organizacijų taikomųjų programų (ERP, CRM, TVS, E-Komercijos, Apskaitos ir kt.) integracijos ir sąveikumo problemos. Analizuojami dinaminiai metodai siekia pagerinti taikomųjų programų integracijos procesus ir juos automatizuoti, arba bent jau sumažinti integracijos palaikymo ir aptarnavimo laiką bei žinių poreikį integracijos sistemoms palaikyti.

Šiame darbe nagrinėjami organizacijų veiklos taikomųjų programų integracijos ir sąveikumo sprendimai.

1.1. Tyrimų sritis

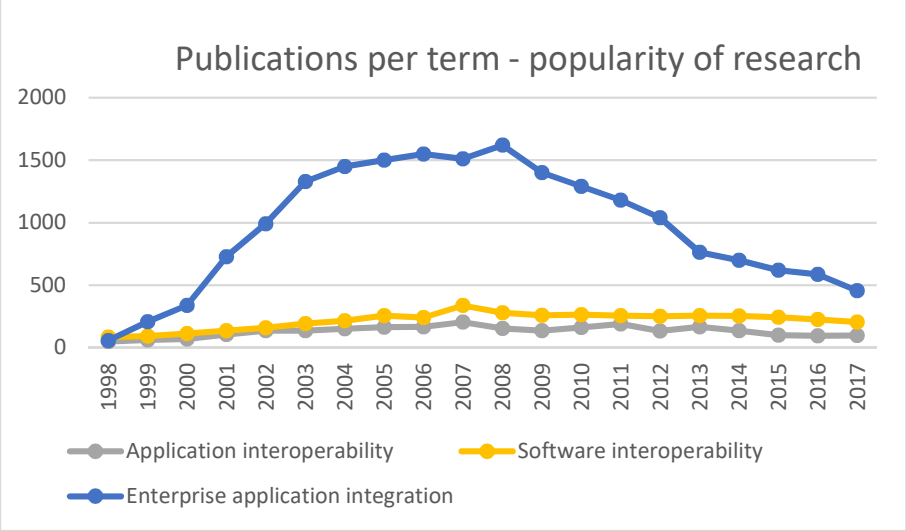
Šiuolaikinėse organizacijose veiklos procesas yra greitai kintantis, naudojama daug SKIRTINGŲ GAMINTOJŲ taikomųjų programų ~~sprendimų~~ tokių kaip e-prekybos platformos, sandėlio valdymo, finansų apskaitos, ryšių su klientais valdymo ir daugybĖ kitų taikomųjų programų.

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“, „Aplication interoperability“**.**

Paveiksle 1 pavaizduotas terminų populiarumas, praeitam dešimtmetyje buvo populiarėjančios debesų technologijos ir verslo valdymo sistemos, tačiau nors temos populiarumas sunyko daugelis problemų teliko neišspręstos.



Paveikslas 1. Literatūros analizė Google mokslinčiaus 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 tai, kad duomenų mainai tarp skirtingų gamintojų taikomųjų programų nėra praktiškai užtikrinami, nors ir yra pateikiamos rekomendacijos [19, 29].

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

- SOAP – paprastas objektų prieigos protokolas (angl. Simple Object Access Protocol). Tai duomenų mainų protokolas servisais grindžiama metodologija suprojektuotose taikomosiose programose. Jis naudojamas struktūrizuoti duomenis 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., 2000 [6].

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 ganėtinai senai nagrinėjama. Chen ir kiti [1] apibrėžė verslo sąveikumo ir integravimo architektūrą, taip pat papildė veiklos sąveikumo karkasą [2]; Dijkman ir kiti apibrėžė veiklos procesų panašumo vertinimo metodus ir vertinimo procesą [3]; Dzemydienė ir kiti apibrėžė informacinių taikymo poreikį elektroninių viešųjų paslaugų sektoriuje [4]. El-Halwagi 2006 metais apibrėžė procesų integraciją ir jos poreikį [5].

Programų integravimas tai skirtingų taikomųjų programų paketų apjungimas į bendrą visumą siekiant sukurti vieningą veiklos valdymo sistemą (pavyzdžiui ERP). Programų sąveikumas, tai taikomųjų programų paketų duomenų mainų procesai. Kuriant programų sąveikumo sprendimus nesiekiama sukurti integruotų sprendimų tačiau pritaikyti esamus ir adaptuotis. Programų integravimas ir programų sąveikumas apima galimus veiksmus su duomenimis iš skirtingų objektų. Programos integruojamos kuriant sistemos architektūra, o jų sąveikumas yra užtikrinamas, kai programos jau sukurtos ir jų struktūros ar veikimo principų nėra galimybės pakeisti. Sąveikumas ir integravimas – tai duomenų mainų sprendimai kurie dažnai maišomi ir pavyzdžiui EAI (angl. enterprise application integration) ir EIF (angl. european interoperability framework) sprendžia labai panašias problemas: duomenų mainų tarp skirtingų objektų [1,44].

Toliau nagrinėjamas tik sąveikumas, dėl to kad esant dinaminėi organizacijai, natūraliai kinta 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 sukurtos kelių skirtingų gamintojų. Kadangi šios programos paveikia vieną veiklą, reikalinga kad jos būtų sąveikaujančios siekiant užtikrinti sėkmingus duomenų mainus.

Sąveikumo vertinimo metodai šiuo metu yra subjektyvūs ir remiasi, apklausomis, bei analize pasitelkiant srities specialistus. Pavyzdžiui LISI metodas [12] pateikia vertinimą apie aplinkos galimybes leidžiančias programinei įrangai sąveikauti, tačiau neapibūdina vidinių programinės įrangos savybių ir jų galimybės keistis duomenimis.

Esamais sąveikumo vertinimo sprendimais (SPICE, LISI, OIM, LCIM, EIMM) [50] galima įvertinti verslo programų sąveikumo įgyvendinimo galimybes pavyzdžiui:

- Fizines galimybes įgyventinti integracijos ar sąveikumo sprendimus[12];
- Įvertina galimybes perduoti signalą per aparatinę įrangą (serveriai, kompiuteriai, tinklo plokštes);
- Įvertina galimybes perduoti signalą per programinę įrangą (tinklo plokščių draiveriai, operacinės sistemos, duomenų perdavimo ir kodavimo protokolai).

Šie sprendimai turi patikrinti taikomųjų programų savybes ir įvertinti, ar įmanomi duomenų mainai. Jie turi savybes apibrėžti aplinką, nustatyti integracijos ir sąveikumo sąlygas, įvardinti tam tikras, problemas ir poreikį, tačiau išvardinti sprendimai neanalizuoja integracijos ir sąveikumo procesų iš taikomųjų programų perspektyvos, kurios atlieka pagrindinį vaidmenį šioje srityje. Kiekvienam integracijos srities ekspertui svarbu turėti gilumines žinias apie integruojamas sistemas ir ~~jų~~ galimybę apsikeisti duomenimis. Sprendimai [4, 12] neįvertina sistemos sandaros, duomenų perdavimo sąsajos architektūros. Dėl šios priežasties, neatsiranda efektyvių sprendimų integracijai ir sąveikumui užtikrinti. Verslo taikomųjų programų sąveikumo plėtrai būtina sukurti sąveikumo vertinimo metodus, kurie įgalintų bent dalinai 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 stygiaus).

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

Darbo objektas – organizacija, kurios veiklos procesai nuolatos kinta ir kuri naudoja programas iš daugiau nei vieno tiekėjo kurios gali turėti sąveikumo problemų, tokių kaip: duomenų pasikartojimas, veiklos procesų dubliavimas.

Uždaviniai

Darbo uždaviniai tikslui realizuoti:

1. Išanalizuoti organizacijų taikomųjų programų integravimo ir sąveikumo aktualias problemas, nustatyti taikomus jų 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 panaudojant veiklos procesų architektūrą (CIM lygmens modelius) ir taikomųjų programų architektūrą (PIM lygmens modelius) ir jų tarpusavio atvaizdus.
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 panaudoti veiklos procesų modeliai, kurie identifikuoja priežastinius ryšius tarp veiklos procesų ir šios domeno žinios įgalina atsekti šiuos priežastinius ryšius tarp taikomųjų programų komponentų (panaudojant CIM, PIM lygmenų modelius).

2. Sukurtas taikomųjų programų sąveikumo galimybių kiekybinio įvertinimo metodas, kuris panaudoja žinių struktūras specifikuotas CIM ir PIM modeliais, autonominio skaičiavimo technologiją ir teksto analizės įrankius.
3. Pritaikius teksto redagavimo nuotolio analizė naudojant Levenshtein, Jaro-Winkler, Jaccard ir ilgiausios bendros sekos metodus atliktas programų sąveikumo galimybių pagal programų panašumą vertinimas.
4. Pritaikyta latentinė semantinė analizė programų sąveikumo galimybių pagal programų panašumą vertinimui.

1.5. Ginamieji teiginiai

1. Veiklos architektūros karkasų (angl. EA frameworks) ir MDA taikymas **kartu sprendžiant taikomųjų programų** sąveikumo problemas įgalina 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 panaudojimas specifikuoti sąryšiui** 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 galimybių vertinimui 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 technologijų pagrindu, kuris kintant organizacijos veiklos procesams ar taikomosioms programoms, gali aptikti pokyčius įtakojančius taikomųjų programų sąveikumą.

1.6. Darbo rezultatų aprobavimas

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

Publikuoti darbai:

- Valatavičius, A., Gudas, S., Apie taikomųjų programų sąveikumo metodologiją, grindžiamą giluminėmis žiniomis. Informacijos mokslai, 79(79), pp.83-113.
- Publikuotas straipsnis: 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, Andrius & Gudas, Saulius, 2015. Towards business process integration using autonomic computing. Informacinės

technologijos 2015: Konferencijos pranešimų medžiaga, pp.81–84.

- Valatavičius, A. and Gudas, S., Apie taikomųjų programų sąveikumo metodologiją, grindžiamą giluminėmis žiniomis. Informacijos mokslai, 79(79), pp.83-113.
- Valatavičius, A. and Gudas, S., 2018. Measuring Enterprise Application Software Interoperability Capability.
- Valatavičius, A. and Gudas, S., 2015. Enterprise software system integration using autonomic computing. *CEUR-WS. org*, *1420*, pp.156-163.
- Valatavičius, Andrius & Gudas, Saulius, 2016. Modeling environment to maintain interoperability of enterprise applications. Data analysis methods for software systems : 8th international workshop on data analysis methods for software systems, Druskininkai, December 1-3, 2016, pp.63–64.
- Valatavičius, Andrius & Gudas, Saulius, 2017. Advanced evaluation methods of multiple application software interoperability. 9th International workshop on Data Analysis Methods for Software Systems (DAMSS), Druskininkai, Lithuania, November 30 - December 2, 2017, p.52.
- Valatavičius, Andrius & Gudas, Saulius, 2018. Advanced evaluation methods of multiple application software interoperability. 10th International workshop on Data Analysis Methods for Software Systems (DAMSS), Druskininkai, Lithuania, November 29 - December 1, 2018, p.87.

Pranešimai skaityti šiose konferencijose:

- Tarptautinė konferencija: Dalyvauta Doktorantų konsorciume BIR 2015 Estijoje tema: “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 „Data Analysis Methods for Software Systems“ 2016;
- Konferencija: XVIII tarptautinėje kompiuterininkų konferencijoje LIKS 2017, tema: Towards deep knowledge based interoperability of applications. Straipsnis priimtas publikacijai žurnale „Informacijos Mokslai“;

1.7. Disertacijos struktūra

Pirmoje dalyje apžvelgiama tyrimų sritis, mokslinio darbo aktualumas, disertacijos darbo tikslai ir uždaviniai. Darbo eigoje išgrynintas darbo mokslinis naujumas, įvartinti ginamieji teiginiai ir pateiktas darbą aprobuojančių darbų sąrašas. Antroje dalyje apžvelgiama literatūra aktuali kuriant sąveikumo sprendimus ir sąveikumo įgyvendinimo galimybės (angl. Capability) vertinimo tarp skirtingų veiklos taikomųjų programų dinaminėje verslo aplinkoje.

Antrame skyriuje aprašoma organizacijų taikomųjų programų sąveikumo metodų apžvalga. Aptariami apibrėžimai, paaiškinamos integdacijos ir sąveikumo sąvokos. Trečiame skyriuje aprašomi taikomųjų programų sąveikumo matavimai, sprendimai. Pabrėžiamas MDA ir giluminių žinių naudojimo poreikis.

2. Verslo veiklos programų sąveikumo sprendimų apžvalga

Dinaminėje verslo aplinkoje, vykdant veiklos procesus ir apdorojant duomenis skirtingose sistemose, susiduriama su daug problemų. Esant veiklos taikomųjų programų duomenų homonogeniškumui susiduriama su procesų dubliavimusi, našumo mažėjimu, bei duomenų valdymo ir saugumo problemomis. Jau daugiau nei 20 metų akademinėje visuomenėje nagrinėjamos sąveikumo ir integracijos problemos susijusios su veiklos aplikacijomis ir sistemomis: veiklos resursų planavimo (ERP), ryšių su klientais valdymo (CRM), apskaitos, sandeliavimo, žmogiškųjų išteklių (HR), e – verslo ir kitų panašaus pobūdžio (verslo veikloje naudojamų) taikomųjų programų. Žinoma, kad, bet kokia veikla gali naudoti daugiau nei vieną sistemą savo poreikiams įgyvendinti pavyzžiui Scott Brinker iš Chiefmartec analizė [25] atskleidžia, kad vidutinė veikla naudoja vidutiniškai 43 CRM, 90 HR ir daugiau kitų Debesijos pagrindu veikiančių aplikacijų, tačiau net neužsimenama kiek ne Debesijos pagrindu veikiančių taikomųjų programų gali būti veikloje. Taip pat verta paminėti veiklos procesų analizę, nuo jos priklauso anksčiau išvardintų taikomųjų programųvaldymo efektyvumas.

Bendrąja prasme taikomųjų programų integracija turi leisti sistemoms mainytis duomenimis arba fiziniais resursais vieningai 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 sprendžiamos duomenų ir funkcijų mainų tarp keliu tos pačios verslo srities programų [1]. Integravimas tarp taikomųjų programų gali būti: fizinis (pavyzdžiui

sujungimas fiziškai kabeliais – duomenys tampa pasiekiami per tinklą), gali būti duomenų bazės integravimas į vieną, taikomųjų programų ir jų komponentų integravimas į vieną bendrai naudojamą sistemą, bei atskirų organizacijos veiklų apjungimas į vieną bendrą. Sąveikumas, kita vertus, apibrėžia, kaip viena sistema gali gauti ir panaudoti kitos sistemos duomenis ir funkcijas. Kitaip tariant veiklos sistemos negali sąveikauti tarpusavyje jei nėra užtikrinta bent žemiausio (fizinio lygio) integracija. Apibendrinant, integruotos veiklos sistemos yra visuomet sąveikaujančios ir negali egzistuoti atskirai, tačiau sąveikaujančios sistemos nebūtinai turi būti integruotos ir gali egzistuoti nepriklausomai nuo viena kitos.

Šioje disertacijoje keliamas klausimas kaip galima automatizuoti taikomųjų programų sąveikumą. Bilas Geitsas yra pasakęs, kad svarbiausia norint kažką pagerinti – yra matavimai [8]. 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 [9], aprašė ir suprojektavo daugia-agentinę sistemą nedidelio našumo įterptinėms sistemoms integruoti kuri iš principo parodo būtinybę kurti integracijos ir sąveikumo metodus tarp taikomųjų programų. Aurelijus Morkevičius 2014 metų disertacijoje [16] nagrinėjo kaip susijusios organizacijos veikla ir informacinių sistemų ir kėlė klausimą ar galima jas suderinti. Loreta Savulionienė 2014 metų disertacijoje [24] aprašė ir pateikė pagrindus duomenų susietumo vertinimui naudojant dažnų posekių metodologijas. Dalia Dzemydienė ir Ramutė Naujikienė straipsnyje vertino elektroninių viešųjų paslaugų

naudojimo ir informacinių taikomųjų programų sąveikumą 2009 m. [4] jos aprašė sąveikumo problemas valstybinių sektorių lygyje ir aprašė įtakojančius faktorius šioje srityje sąveikumo problemoms atsirasti. Visi išvardinti lietuvių autoriai iš dalies susiję su sąveikumo problemomis tam tikrose srityse, bet nesprendžia sąveikumo problemų tarp taikomųjų programų dinaminėje verslo aplinkoje, taikomųjų programų architektūros įtakos sąveikumo procesams.

Apie veiklos procesų modelių panašumo identifikavimą rašė R. Dijkman ir kt., 2011 metais [3], straipsnyje įvardijo daug galimų metodų leidžiančių įvertinti verslo procesų panašumą skirtingais lygiais nuo sintaksinio iki semantinio. Jų darbe išaiškinti veiklos procesų pagrindai kurie buvo pritaikyti ir šiame straipsnyje. Kaip ir veiklos procesų modelius taip ir taikomųjų programų architektūros panašumą galima vertinti naudojantis R. Dijkman aprašytais žingsniais – sudėtingumo lygiais vertinant [3]:

- 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 analizuotas meta-duomenų rinkimas ir analizė naudojant įvairiapusišką informaciją kurią būtų galima išgauti iš veiklos ir taikomųjų programų naudojamų joje. 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 įvertinta šių temų straipsnių kiekis Google Scholar paieškos platformoje. Veiklos taikomųjų programų tinklo servisai 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 taikomųjų programų automatizavimui, veiklos duomenų analizei todėl tyrimo struktūra apima 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 sistemos, kurių galima prieiga prie duomenų šaltinių.

2.1. Apibrėžimai

Nagrinėjama dinaminės organizacinės veiklos taikomųjų programų, aplikacijų integracijos ir sąveikumo tema. Būtina apibrėžti jog veikla - toliau E (angl. enterprise). Pagal informacijos taikomųjų programų

inžinerijos teorijos pagrindus [32] kiekviena verslo veikla turi funkcijų (f), procesų (p) ir tikslų (g) rinkinį [32]. Be kita ko veikla turi funkcijų ir procesų sąveikų rinkinį (r), bei informacinius elementus (k) kurie reikalingi pastarųjų sąveikų realizacijai. Kitaip tariant organizacinę veiklą galima išreikšti formalizuotai (teiginių logikos notacija) [14]:

$$E \ni \{g, f, p, r, k\}$$

Apie tai kaip tikslai, funkcijos, procesai, jų ryšiai ir žinios tiksliai atitinka realaus pasaulio E veiklą galima žinoti tik iš modelio (M) kokybės. Kitaip tariant modelis yra veiklos dalis. Nors standartiškai nėra įtraukiama į aprašymą šiame tyrime traktuojama jog veikla gali turėti modelį (M) taip pat veikla naudoja vieną arba kelias sistemas (S) veiklos valdymui užtikrinti. Tuomet veikla be kita ko turi bent vieną M ir bent vieną S:

$$E \ni \{M, S\}$$

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

$$M_E \ni \{g, f, p, r, k\}$$

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

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

Sistema gali būti sukurta Debesijos pagrindu arba diegiama vietoje kaip savarankiška sistema (angl. standalone on site application, legacy application). Tyrimo įtakai neturi įtakos kokiu pagrindu sukurta sistema, svarbu, kad sistema būtų sukurta naudojant paslaugomis grindžiama architektūra ir visos arba dalis esybių yra paslaugos kaip

operacijos kurios gali 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 esybes) ir kokie ryšiai yra tarp šių objektų skirtingose sistemose:

$$S_n \ni \{O_1, O_2, \dots, O_i\},$$

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

Kiekvienam sistemos S objektui O galima rasti ryšį su kitos sistemos S objektu O. Kiekvienas objektas aprašomas schema (h):

$$O \ni h$$

Kiekvienas objektas O turi schemą h, kuri apibūdina šio objekto sandarą ir meta-duomenis kurie 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.

2.2. Taikomųjų programų integracija

Informatikos srityje taikomųjų programų integracijos tema yra ganėtinai sena kilusi iš poreikio skirtingos radijo ryšių technologijos būtų atpažintos, perimtos ir apjungtos į vieningą sistemą. Amerikos kosmoso organizacijai NASA ši tema taip pat buvo aktuali, nes ryšių kontrolės centras turėjo stabiliai kontaktuoti su dirbtiniais žemės palydovais esant dideliems trikdžiams ar trūkiams. Sistemos turėjo būti integruotos – jų komponentai dirbti vienas su kitu sklandžiai siekiant užtikrinti gerą informacijos perdavimą. Šiuo metu taikomųjų programų integracija taip pat aktuali verslo stityje ir siejasi su verslo tikslais ir

siekiais optimizuoti veiklos procesus. Google scholar duomenimis nuo 2014 m buvo 97 tūkst. straipsnių publikuota susiję su taikomųjų programųintegracijos tematika (angl. enterprise application integration) nors dauguma iš jų yra skirti specifinėms sritims analizuoti. Sistemos darosi sudėtingos, jos ne visada kuriamos nenaudojant standartų ir jų yra daug, ypač debesų pagrindu veikiančių taikomųjų programų. Todėl reikia pakankamai daug skirti dėmesio integracijos architektūrai norint, kad sistemos komunikuotų tarpusavyje.

Taikomųjų programų integracija turi keletą metodų:

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

Taikomųjų programų integracijos sprendimai yra vystomi įvairiomis kryptimis, tiek verslo tiek mokslininkų. Vystyti taikomųjų programų integracijos sprendimus naudinga keliomis prasmėmis: standartizuoti integracijos sprendimų kūrimo procesus [1]; Greičiau apdoroti taikomųjų programų informaciją; atlikti sudėtingas manipuliacijas su dideliais duomenimis atpažįstant esminius duomenis, jų tarpusavio ryšius; sumažinti nesėkmingų projektų skaičių [26]; sumažinti žinių poreikį reikalingą sėkmingiems taikomųjų programų integracijos projektams atlikti; didinti taikomųjų programų lankstumą; mažinti veiklos procesų kaštus [18, 23]. Apibendrintai, programų integraciją galima būtų aprašyti šia išraiška:

$$\{S_1, S_2, \dots, S_n\} \ni S'$$

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 tai toks integracijos metodas užtikrinantis, kad viena sistema galėtų įvykdyti visus veiklos proceso žingsnius [37], nuo žaliavos gavimo iki galutinio produkto sukūrimo, išskirtinai tik savo resursais. Šiuo integracijos metodu integruojamos posistemės turinčios atskirus tikslus, tačiau esantys dalis bendros sistemos. Geras pavyzdys būtų veiklos sistemos su vertikalia integracija būtų veiklos išteklių planavimo sistemos (ERP – angl., enterprise resource planning). ERP taikomųjų programų architektūra kuriama iš karto su vertikalios 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 pilnai apibūdinamas ERP sistemos. Kitaip tariant veiklos valdymo grandinė ERP sistemos pagrindu dažniausiai beveik visai padengiama.

Vertikalios integracijos atveju egzistuoja tokia sistema arba sprendimas su objektais kurie pilnai aprėpia veiklos E visus svarbius procesus p_n .

$$\exists\, S_i \rightarrow \forall\{p_1, p_2, \dots, p_n\} \in E$$

Šio integracijos metodo privalumai tokie, kad sprendimą galima greitai sukurti, žinant konkrečią reikiamą architektūrą t.y. turint pilną veiklos E modelį M. Vertikalios 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, tai yra horizontaliosios integracijos metodo dalis. Dinaminėje verslo aplinkoje dažnai atsiranda pokyčių kurių sistemos laikui bėgant pradeda nepalaikyti. Reikia kurti, atnaujinti programos posistemių sprendimus, o nauji sprendimai reikia vėl integruoti į visą sistemą.

2.2.2. Horizontalioji integracija

Horizontalioji integracija tai toks taikomųjų programų integracijos metodas, kuriame sistemos posistemės turi savotišką autonomiškumo lygį. Imant pavyzdį iš verslo srities, tai būtų atskira organizacija teikianti buhalterijos paslaugas. Kita organizacija samdo šią buhalterijos paslaugą sudarius bendradarbiavimo sutartį keičiasi duomenimis panaudojant tam tikras veiklos sistemos sąsajas ir buhalterinės sistemos sąsajas – verslo organizacijos yra 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 susiskaldžiusiai verslo veiklos struktūrai. Tokia integracija tarp padalinių naudojamų atskirų taikomųjų programų būna horizontali dėl to, kad padaliniai turi tam tikrą autonomiškumo laipsnį ir dalinai gali dirbti be kito padalinio įsikišimo. Dažnai tokio integracijos metodo taikymas turi būti palaikomas koordinuojančio posistemio, arba verslo prasme – skyriaus. Pavyzdžiui, verslo atžvilgiu tai yra IT skyrius siekiantis užtikrinti, kad

kitos posistemės veiktų sklandžiai ir mainytųsi informacija. Traktuojant, kad veikla yra išskaidyta į autonominius padalinius, tokios veiklos apibrėžimas būtų toks:

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

$$\exists \{S_1, S_2, \dots, S_i\} \rightarrow \forall\{p_1, p_2, \dots, p_n\} \in \forall\{E_1, E_2, \dots E_n\}$$

Egzistuoja tokios sistemos kurių integracija apima visus veiklos procesus per visas veiklos pogrupius (skyrius ir poskyrius).

2.2.3. Žvaigždinė integracija

Žvaigždinė integracija tai toks 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. Tai pavyzdžiui, E-Komercijos posistemė jungtųsi su sandėlio valdymo posisteme, siekiant gauti prekių likučių duomenis. Tačiau sandėlio valdymo posistemei neaktualu kokia informacija yra naudojama E-komercijos posistemėje.

$$\exists S_n \rightarrow \neg\forall\{S_1, S_2, \dots, S_k\} \rightarrow \neg\forall\{p_1, p_2, \dots, p_i\} \in E, n \neq k$$

Egzistuoja tokia sistema, kuri integruota su kai kuriom kitom sistemomis, kurios aprėpia ne visus veiklos procesus.

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

Žvaigždinės integracijos metodu reikalinga posistemė nebūtinai gali būti pasiekiama dėl tinklo servisų, prieigos prie duomenų šaltinio

trūkumų ar leidimų apribojimų. Tokią integraciją sunku valdyti ir stebėti.

2.3. Sąveikumas

Sąveikumas (angl., interoperability) tai veiklos programinės įrangos taikomųjų programų ir jų posistemių galimybė efektyviai mainytis duomenimis ir informacija išnaudoti viena kitos funkcionalumus savo reikmėms. Sąveikumas gali būti įgyvendinamas keliais lygiais: tarp vienos sistemos komponentų arba tarp kelių skirtingų taikomųjų programų komponentų. Norint sužinoti ar sistemos gali būti tarpusavyje sąveikaujančios, reikia turėti būdų šį sąveikumą išmatuoti. Šiame darbe analizuojamas taikomųjų programų tinklo serviso struktūrinis panašumas, operacijų panašumas, objektų panašumas bei duomenų struktūrų panašumai, kurie leistų įvertinti sąveikumo tarp taikomųjų programų galimybę. Peržvelgus nagrinėjamą literatūrą identifikuojami sąveikumo srities tipai, barjerai, taikomųjų programų panašumų metrikos.

Sąveikumas yra skirstomas į tipus [1]:

- Sintaksinis (angl. syntactic) – sąveikumas naudojant bendrus duomenų formatus ir protokolus.
- Semantinis – gebėjimas interpretuoti informaciją, kuria keičiamasi.
- Tarp-sritinis (angl. cross domain) – kelios socio-politinės esybės, kitaip, verslo partneriai, dirbantys kartu siekiant geresnio rezultato. Tai verslas-verslui (B2B) sąveikumo įgyvendinimo tipas.

Sąveikumą galima vertinti panašumo rodikliu. Sintaksinis sąveikumas būtų 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. [55] apžvelgti naujausi darbai iki 2014 metų. Pastarajame straipsnyje pabrėžiami sąveikumo sudėtingumo lygiai ir palyginami karkasai kurie pateikia sąveikumo sprendimo kūrimo priemones bei rekomendacijas. Dauguma apžvelgtų darbų nevertina taikomųjų programų sąveikumo galimybės, išskyrus LISI metodą kuris reikalauja daug ekspertinių žinių [1, 12].

2.3.1. Sąveikumo barjerai

Sąveikumo problema kyla dėl priežasčių dar vadinamų sąveikumo barjerais [1]. Sąveikumo barjerai yra trijų kategorijų (D. Chen, Enterprise Interoperability Framework 2006): koncepciniai, technologiniai, organizaciniai ir teisiniai.

Koncepciniai saveikumo barjerai – tai sintaksiniai ir semantiniai informacijos skirtumai tarp taikomųjų programų. Šis barjeras apriboja įvairaus lygio sąveikumo architektūrą nuo aukšto – veiklos procesų modeliavimo lygio iki žemo duomenų struktūrų architektūros lygio.

Technologiniai – šio tipo barjerai sprendžia fizikines technikos sąveikumo problemas, tokias kaip duomenų perdavimo signalai, platformų architektūra, kuri juos perduoda. Taip pat technologinis barjeras apriboja galimybes pateikti, talpinti keistis ir komunikuoti duomenis ir informaciją per programines sistemas.

Organizaciniai – šie barjerai sprendžia teisinius, reguliacinius sąveikumo aspektus. Jų tikslas užtikrinti, kad sąveikumas nepažeidžia

įstatymų ir yra nepažeidžiamas piktavalių ar kitaip nepaveikiamas žmogaus klaidų.

Teisiniai – reikia užtikrinti, kad duomenys nebūtų naudojami blogiems tikslams arba nutekinti vykdant sąveikumo operacijas. Taip pat galima įtraukti naują BDAR (bendras duomenų apsaugos reglamentas) 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 funkciniames komponentams bendradarbiauti mainantis duomenimis, funkcijomis. Vartotojas šiame bendradarbiavimo procese turi mažai arba visai neturi žinių apie šių komponentų sandarą ir duomenų perdavimo būdus. Su šiuo aprašymu visos sąveikaujančios programos, turi būti sukurtos pagal servisais grindžiamos architektūros (SOA) sprendimus. Pagrindinis principas SOA sukurti tokia aplikaciją, kad tai vartotojui būtų kaip juodoji dežė bet taip pat aprašant įvesties išvesties parametrus kad kitos sąveikaujančios aplikacijos galėtų pasiekti pastarosios duomenis ir funkcijas [45 – 330 p]. Applikacijos vartotojui nereikia žinoti applikacijos serviso įvesties ir išvesties kintamųjų, kai sąveikumo problemos yra išspręstos sąveikumo sprendimo architekto, kuris sukuria tarpinę aplikaciją (angl. middleware) užtikrinti bendradarbiavimui (sąveikavimui) tarp dviejų ar daugiau verslo applikacijų.

Naujame Europos Sąveikumo Karkase (ISA^2,2017) sąveikumo sluoksnio rekomendacijos iš naujo apibrėžtos ir gali būti naudojamos, kaupiant žinias apie veiklos applikacijų 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 papildomai atsižvelgiant į tokius teisinius aspektus:

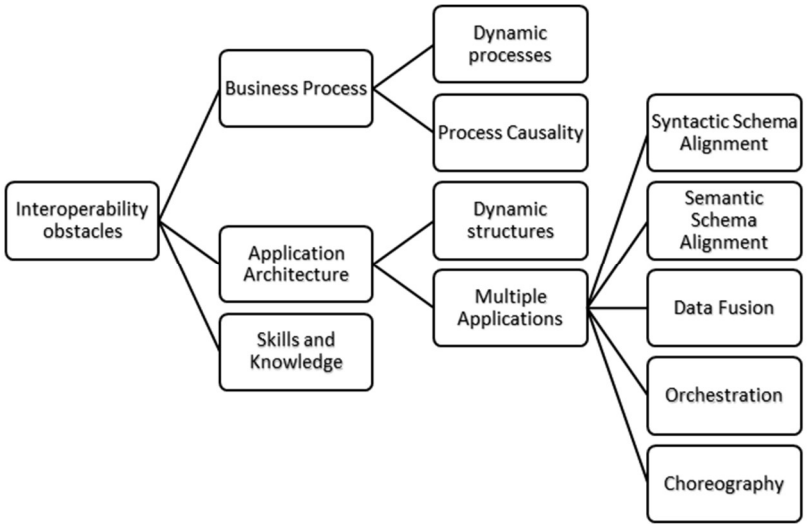
- Tinklo servisų taisyklių rinkiniai leidžia patikrinti pritaikymą fiziniame ir skaitmeniniame pausaulyje
- Tinklo servisų taisyklės leidžia identifikuoti barjerus kylančius skaitmenizuotų duomenų mainų procesuose, pavyzdžiui: duomenų apsikeitimas, apsikeitimo dažnis, duomenų mainų limitas per dieną, serviso komponento arba funkcijos aktyvavimas per dieną ir kiti.

Iš teisinės perspektyvos šioje disertacijoje nėra galimybės identifikuoti ir įvertinti informacijos komunikavimo technologijos (ICT – angl., Information Communication Technology) įtakos suinteresuotoms šalims. Organizacinis sąveikumo sluoksnis iš EIF dalinai apžvelgtas ir panaudotas veiklos procesų modelio analizei rementis veiklos architektūra (EA – angl., enterprise architecture) ir modeliais grindžiama architektūra (MDA - angl., model driven architecture). 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 abu semantinį ir sintaksinį aspektus, kur semantinis aspektas fokusuojamas į reikšmę ir aprašymą duomenų elementų ir jų ryšių o sintaksinis aspektas aprašo duomenų elementų formatavimą ir apribojimus (ilgį, pasikartojimų skaičių). Techniniame EIF lygmenyje remiamasi prielaida, jog dauguma veiklos

programų yra senos arba apribotos tik konkretiems veiklos procesams, bet nepadengia visų procesų organizacijoje. Tokiose atskirtose sistemose nors ir turima bendrų duomenų tačiau šios programos jomis nesidalina todėl negali būti vadinamos sąveikaujančiomis. Techniniam sąveikumo užrtikrinimui reikalingi programos interfeiso aprašai (angl., API reference) apibrėžiantys įvesties ir išvesties kintamuosius. Iš čia – barjeras, jeigu programos API neturi aprašymų arba aprašyti nesilaikant standartų tokių kaip pavyzdžiui SOAP, dažnas pavizdys REST metodologijoje, kur aprašymai dažnai yra laisvai struktūruoti ir jų nuskaitymo negalima automatizuoti. Pagrindiniai sąveikumo uržikrinimo barjerai yra:

- Veiklos procesai kinta kai naujos programos įdiegiamos versle
- Programos yra dinamiškos ir jų struktūra gali keistis laike
- Kelios programos naudojamos toje pačioje srityje, pavyzdžiui vietoj vienos ERP sistemso padengiančios visus procesus naudojama CRM programa klientų valdymui, atskirai e-komercijos programa elektroniniai parduotuvei valdyti, ir atskira programa buhalterijos tvarkymui ir apskaitos vedimui.
- Nėra bendrai paplitusių metodų, aprašyti sąveikoms tarp skirtingų programų
- Programos pokyčiai visada įtakoja veiklos procesus, todel, ankstesni veiklos procesų modeliai tampa nebegaliojantys ir negali būti naudojami žinioms apie priežastinius ryšius ištraukti

- Siekiant užtikrinti programų sąveikumą, integracijos ekpertas turi atlikti šias užduotis
    - Atlikti schemų sulygiavimą [7,21,36,38,46]
    - Užtikrinti įrašų susiejimą ir duomenų apjungimą [22,39]
    - Užtikrinti procesų orchestavimą – laiką tarp kiekvieno duomenų mainų proceso įvykdymo
    - Užtikrinti choreografiją – eilės tvarką kiekvieno duomenų mainų proceso
- Giluminių žinių apie programas, veiklos procesus trūkumas, taip pat integravimo įgudžių trukumas.



Paveikslas 2. Sąveikumo barjerai.

Programų sąveikumo įgyvendinimui dėl trūkstamų žinių ir įgudžių barjero įveikti negalima surenkant žinias apie pavienes programas, nes kitose programose jos gali nebegalioti – tai iteratyvus procesas todėl ir apbrėžiamas kaip barjeras. Anksčiau sąveikumo barjerai senoje EIF dokumentacijoje [47] dabar laikytini sluoksniais [48]. Duomenys iš

vienos programos negali sąveikauti su panašiais duomenimis kitoje programoje jei visi barjerai nėra išspręsti, pereiti [1].

2.3.2. Sąveikumo interesai

Sąveikumo interesai keliami skirtingiems veiklos lygiams todėl gali būti traktuojami skirtingai ir jų architektūra dėl šių interesų taip pat gali skirtis (D. Chen, Enterprise Interoperability Framework 2006 [2]).

- Duomenų sąveikumas – hierarchinių, reliacinių duomenų modelių sąveikumas. Mainymasis tapačiais – heterogeniniais duomenimis kurie gali būti skirtinguose kompiuteriuose, operacinėse sistemose, duomenų bazių valdymo sistemose bei **skirtingose programinėse sistemose**.
- **Servisų sąveikumas** – tai skirtingų veiklos taikomųjų programų, jų servisų sąveikumas (būtina sąlyga kad jie sukurti ir įgyvendinti nepriklausomai vienas nuo kito). Sprendžia sintaksines ir semantines problemas taip pat būdų gauti duomenis iš duomenų šaltinių iš taikomųjų programų duomenų bazių ir servisų.
- **Procesų sąveikumas** – veiklos procesų sąveikumo užtikrinimas esant jų išsiskaidymui per skirtingus skyrius ar veiklos grupes. 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 apjungiami ir taip sutaupoma laiko ar kitų išteklių sąskaita.

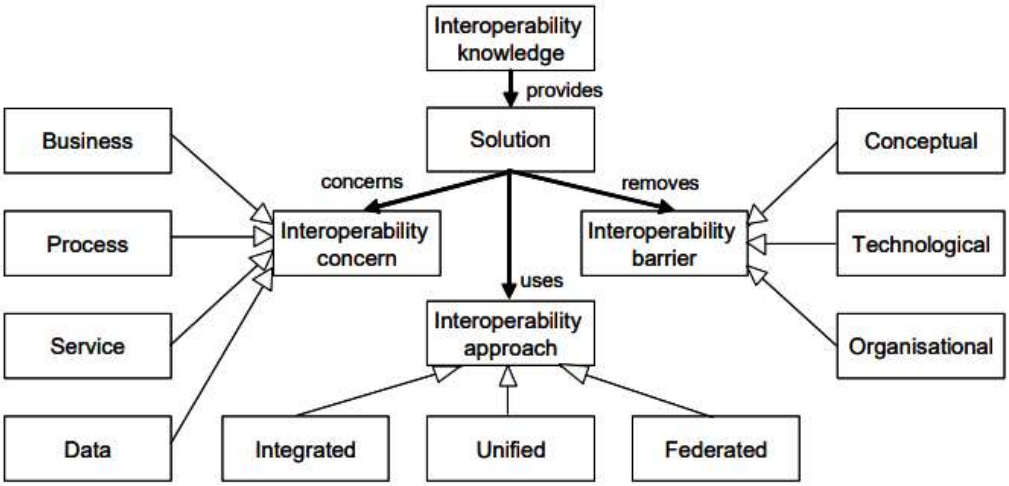
- Verslo sąveikumas – tai kompanijų dalinimasis ištekliais ir veikla taupant resursus. Toks sąveikumas apima sprendimų priėmimą, darbo metodus, įstatymų leidimus, kompanijos kultūrą. Finansinius sprendimus. Galima būtų įvardinti, kad tai B2B (angl. business to business) sąveikumo tipas siekiant sukurti 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 meta-formatas, toks duomenų meta modelis negali egzistuoti atskirai kaip esybė, tačiau gali pateikti naudingos – semantinės reikšmės informacijos, leidžiančios rasti ryšius tarp skirtingų taikomųjų programų ir jų modelių. Geras pavyzdys būtų serviso dokumentacija „WSDL“ ar kitu formatu, kuriuo tiksliai nėra žinoma, kokia konkrečiai duomenų struktūra slepiasi po servisais, tačiau leidžia nusakyti kaip šį servisą galima išnaudoti sąveikumo tikslais.
- Federacijos – nėra bendro formato, todėl, siekiant sukurti sąveikumą reikia surinkti ir išanalizuoti panašias savybes

kiekvienos sąveikos metu. Šio sprendimo būdu nėra aprašyta bendro modelio, kalbų ar darbo metodų.



Paveikslas 3. *Baziniai veiklos sąveikumo konceptai ir jų tarpusavio ryšiai pagal D. Chen 2008 [1].*

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

2.4. Integracijos ir sąveikumo problemos

Išanalizavus taikomųjų programų integracijos srityje atliekamus tyrimus, literatūros apžvalgos metu įvardintos šios problemos kylančios integruojant dvi arba daugiau skirtingų taikomųjų programų. Taip pat šios problemos pasireiškia įvairiose organizacinėse veiklose [18].

Skirtingos duomenų struktūros (angl. schema) – gali skirtis pagal esybių ryšių diagramą duomenų bazėje, skirtinga SOA architektūra [21, 37, 38]. Dažniausiai ši problema yra nagrinėjama integracijos

sprendimus analizuojančiuose tyrimuose. Sunku atpažinti ir išlygiagretinti skirtingų programų duomenų pavadinimus, duomenų laukų arba stulpelių tipus ir ilgius. Ši problema yra viena iš pagrindinių 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, jų tarpusavio komunikavimo protokolais. Išpopuliarėjus Debesijos technologijai daugelis sprendimų perkelta į nutolusius serverius, tačiau dalis senesnių programinių paketų galėjo likti įmonės serveriuose.

Dinaminės sistemos - nuolatos keičiasi arba gali keistis programų struktūra, duomenų lentelių kiekis, esybių 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 atliekantis 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 ryšiu klausimai 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 įtakoja integracijos procesų atlikimo kokybę todėl reikalinga atsižvelgti į kiekvienos programų sistemos išankstines žinias, kuriant integraciją, kas nebūna prieinama visą laiką;

Semantinis Esybių ir Objektų atpažinimas – nagrinėjamas esybių ir objektų atpažinimas semantine prasme, integracijos procese bandoma susieti skirtingų taikomųjų programų esybes arba jų sudėtinius komponentus su kitos programos sistemos esybėmis ir komponentais;

Orkestravimas ir choreografija – Esybių 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 esybėmis bei objektais;

Integracijos testavimas – nagrinėjama kaip užtikrinti teisingą taikomųjų programų perdavimą iš vienos sistemos kitai, laiko trukmę, perduotų duomenų teisingą apdorojimą perduotajai programai [28].

CRUD (angl. Create Read / Update / Delete) **taisyklių užtikrinimas**. Taikomųjų programų integracijos sritis nagrinėjanti kaip užtikrinti saugų ir kokybišką rašymą, skaitymą, atnaujinimą ir trynimą

integruojamose programų sistemose laikantis tam tikrų nustatytų taisyklių.

Semantinė integracija - taikomųjų programų srityje stengiamasi išanalizuoti kaip integracijos procesas galėtų suprasti savo aplinka, taip geriau reaguotų į trikdžius ir gebėtų kai kuriuos iš jų spręsti autonomiškai [18], [37];

Dokumentacijos trūkumas – taikomųjų programų integracijos srityje sukeliantis nemažai aukščiau išvardintų problemų, kurių negali išspręsti net ir patyręs integracijos projektuotojas, tyrėjai ieško automatizuotų būdų kaip išgauti ir pateikti daugiau informacijos apie integruojamas sistemas;

Tarp verslo subjektų integracija – tiriami protokolai saugumo sistemos užtikrinančios saugų ir nešališką duomenų perdavimą jų saugumą ir atsparumą nulaužimams. Ieškoma būdų automatizuoti identifikavimo, skaitmeninio parašo perdavimo technologijų, atsparumo nulaužimams ir duomenų vogimui algoritmai.

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 panaudojant integracijos sprendimų paieškai.

Netvarkingi duomenys – integracijos sistemos architekto sprendžiamos problemos, gilinimasi kaip ištaisyti klaidas, kaip suprogramuoti automatinį jų aptikimą ir ištaisyma arba išvengimą. Sugalvoti kokius metodus taikyti duomenų taisymui ir tvarkymui.

Duomenų perdavimo problemos – pavyzdžiui naudojant SOAP protokolą duomenys įvelkami į XML formatą kuris gerokai padidina perduodamų duomenų kiekį [37].

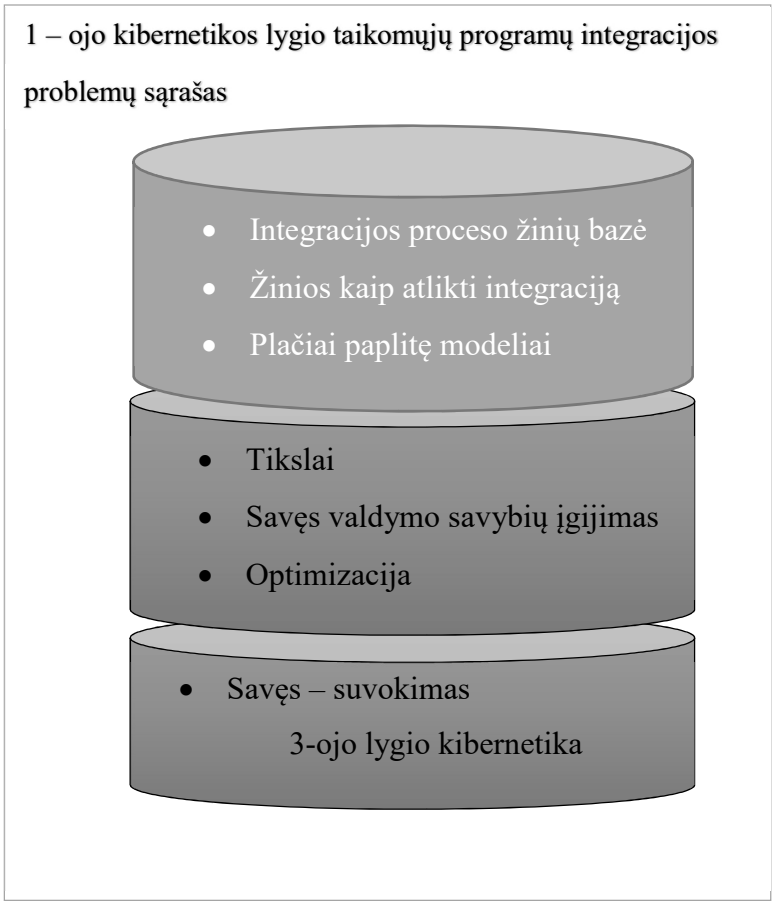
Integracijos ir sąveikumo problemų palyginimas pateikiamas sekančioje lentelėje.

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 Esybių 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 [15]	Ontologijų integracija
15	Dokumentacijos trūkumas	-
16	B2B integracija	-
17	Nestandartinių vienetų, kalbos skirtumų, perteklinių etikečių, formatavimo	-
18	Netvarkingi duomenys	Netvarkingi duomenys

Taikomųjų programų integracijos problemų analizė panaši į skirtingų kibernetikos lygių aprašymus. Kibernetikos lygiai įtakoja sprendžiamų taikomųjų programųintegracijos problemas. Paveiksle 4 apibrėžiami skirtingi kibernetikos lygiai. 1-ajame kibernetikos lygyje

nėra automatizuotų metodų įvardintoms taikomųjų programų integracijos problemoms spręsti. Šiame kibernetikos lygyje taip pat naudojamos integracijos procese sukauptos žinios, kaip atlikti informaciją, kokios sistemos yra integruojamos, kokios tų taikomųjų programų savybės kokios jų duomenų struktūros ir panaši informacija reikalinga integravimo projektavimo stadijoje siekiant, kad integracijos projektas būtų sėkmingas.



Paveikslas 4. Skirtingus kibernetikos lygiai saveikumo srityje

Kaip minėta kai kurias iš įvardintų problemų sprendžiamos 1-ojo lygio kibernetikos pagalba:

- **Skirtingos duomenų struktūros** – rankiniu būdu atliekamas duomenų struktūros apjungimas, reikalauja programiškai keisti struktūrą esant jų pakitimams, integracijos procesas dažnai nustoja veikti;

$$d_2 \equiv d_1 \rightarrow h_2 \neq h_1$$

- **Skirtingi duomenų šaltiniai** – ištiriami kokie duomenų šaltiniai yra programų sistemose, kokios autentifikavimo sistemos ir kokie parametrai reikalingi, tuomet rankiniu būdu užprogramuojama reikalinga informacija. Esant duomenų šaltinių pokyčiams reikalingas perprogramavimas.

$$d \in S_i \notin S_j$$

- **Dinaminės sistemos** – rankiniu būdu programuojami pakeitimai ir pritaikymas pakitusiai sistemai siekiant palaikyti integracijos procesą, kitu atveju procesas nustoja veikti ir informacija neperduodama.

$$S_{1,t} \neq S_{2,t}$$

- **Kelios sistemos** – ištiriami ir nustatomi prioritetai, perdavimo politika integruojamiems duomenims;

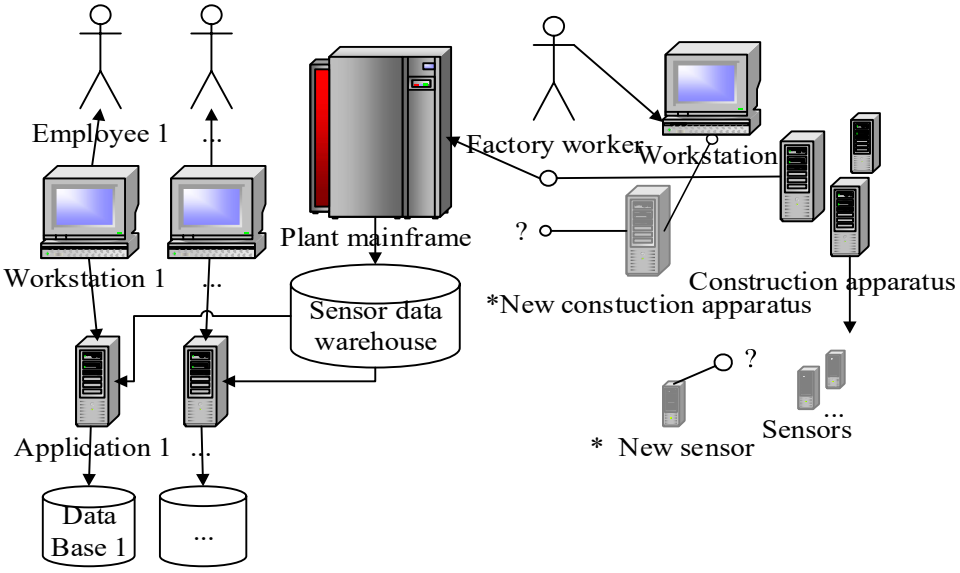
$$S_1 \ni \{d_1, d_2, d_3, \dots\} \cup S_2 \ni \{d_2, d_3, d_5, \dots\} \rightarrow S' \ni \{d_2, d_3, \dots\}$$

Tačiau integracijos projektavimo ir kūrimo procesas nėra automatizuotas. Antrajame kibernetikos lygyje atsiranda tikslai, kuriuos turi siekti įgyvendinti integruojanti aplikacija (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 lygio kibernetika apibūdintų tokią integracinę sistemą, kuri suvoktų savo tikslus ir pati galėtų spręsti autonomiškai visas anksčiau išvardintas problemas, pati reaguotų į pokyčius ir prisitaikytų tam, kad verslo procesai nesustotų funkcionuoti. Trečiojo lygio kibernetika yra daugiau teorinė nei praktinė ir iki jos įgyvendinimo reiktų išpildyti antrojo lygio kibernetikos punktus.

Sąveikumo problemą galima išgryninti pavyzdžiu. Imkime gamyklos veiklos scenarijų. Tarkime, kad gamykla naudoja sensorius savo veiklos procesuose siekiant 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 iššūkis naujas sistemas padaryti sąveikias su egzistuojančiomis sistemomis ir prietaisais, siekiant užtikrinti efektyvią veiklos procesų eigą. Papildomi darbai gamykloje bus atliekami siekiant išspręsti nesutapimus tarp senų ir naujų taikomųjų programų. Gamyklos darbuotojai ir programuotojai turės dirbti kartu siekiant taip pakoreguoti naujas ir esamas taikomųjų programų sąsajas siekiant optimalaus taikomųjų programų veikimo.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.

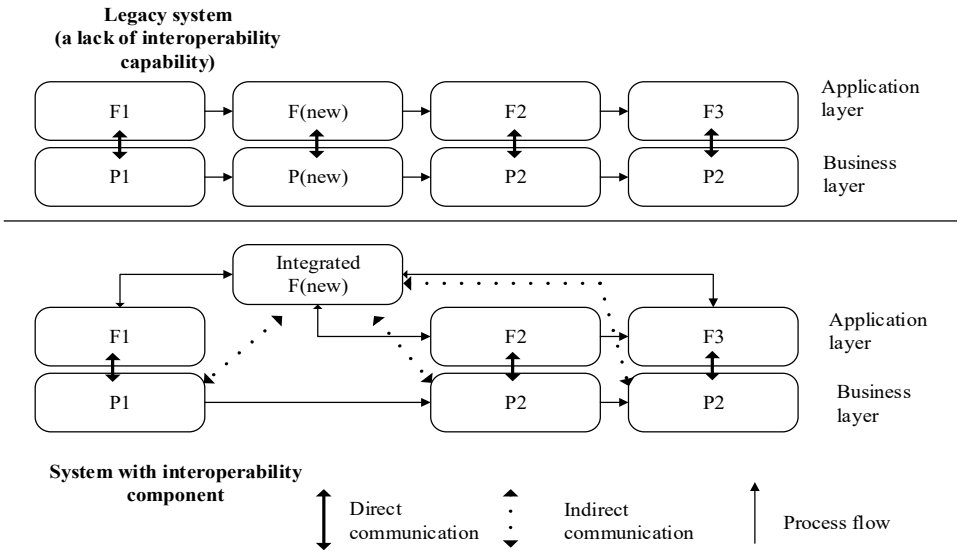
Tačiau ar yra būdų automatizuoti šiuos daug reikalavimų turinčius procesus?



Paveikslas 5. Taikomųjų programų sąveikumo problemos (gamyklos veiklos srities pavyzdys)

Paveiksle 5 pavaizduojamą 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 atnaujinimus nekeičiant veiklos procesų (VP) ir užtikrinant pilną sąveikumą tarp naujų ir senų taikomųjų programų. Iš čia kyla reikalavimai kurti autonominės integracijos ir sąveikumo sistemas kuriose sąveikumas būtų automatizuojamas be jokių pertraukimų. Kuriant tokias autonomines sistemas būtų minimizuojamas darbuotojų aukštos kvalifikacijos poreikis, minimizuojamas darbo susijusio su duomenų valdymu dubliavimas, ir poreikis veiklos procesų re-inžinerijai

(Paveikslas 6). Autonomiškumas integracijos sprendimų šiame scenarijuje galimas tik naudojant metodologiją su grįžtamojo ryšio ciklais ir turint vidinį veiklos procesų modelį.



Paveikslas 6. Veiklos procesų reinžinerija su sąveikumo komponentu ir be jo.

Jei veiklos sistemos nėra pakankamai lanksčios būti pritaikytos prie esamų taikomųjų programų, tuomet bus skiriamos papildomos pastangos veiklos procesų re-inžinerijai (Paveikslas 6 – Legacy system). Tačiau su autonominiu sąveikumo sprendimu dinaminėje verslo veikloje, veiklos sistemos gali būti apjungiamos autonomiškai ir taip išsprendžiama veiklos proceso re-inžinerija.

Taigi, taikomųjų programų tinkle servisai galėtų padėti identifikuoti dvi problemas veiklos procesų sekoje:

- Vartotojai turi bendradarbiauti tarpusavyje (tarp skyrių, padalinių) apdorojant heterogenimius duomenis.

- Vienam vartotojui valdančiam du ar daugiau taikomųjų programų, kai šių taikomųjų programų duomenys heterogeniniai, vartotojas atlieką tam tikrą procesą kelis kartus – atsiranda proceso dubliavimasis.

Atlikus literatūros analizę įvardinti reikšmingi skirtumai tarp taikomųjų programų integracijos sprendimų. Lentelėje (Lentelė 2) pateikiamos žinomos taikomųjų programų integracijos metodologijos ir sprendimai. Lentelėje EAI sistemos tai verslo taikomųjų programų integracijos sistemos palengvinančios integracijos projektuotojų darbą, iš dalies automatizuojant programavimo procesą, dalinai palengvina integracijos projektavimo darbą. Lentelėje naudojimas žymėjimas R – rankiniu būdu atliekami pakeitimai ir reikalingas žmogaus įsikišimas priimant sprendimus bei keičiant integracijos sprendimą. A – pilnai automatizuotai priimami sprendimai – nebereikia žmogaus įsikišimo ir integracijos procesai pilnai kontroliuojami sprendimo. D- dalinai automatizuoti sprendimai.

Lentelės stulpeliai nuo 1 iki 5 aprašo sprendimų įgyvendinimo platformas (taip pat ir sprendimo 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. Vartotojas dažniausiai naudoja grafinę sąsają sprendimams kurti. Daugelis sprendimo aspektų jau yra įprogramuoti į sąsają, tačiau retkarčiais dar

tenka išspręsti problemas atliekant programavimo užduotis, pavyzdžiui darbui su Talend – reikia mokėti „Java“ programavimo kalbą, kitaip sudėtingesnių integracijos sprendimų negalima įgyvendinti.

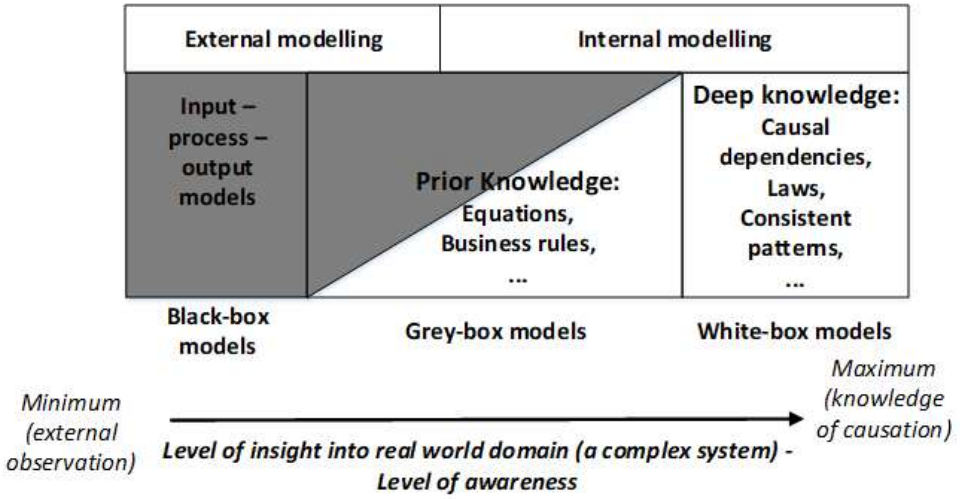
3. **Integracijos agentai** – specializuotų integracijos agentų naudojimas. Dažniausiai agentai sukurti specifinei užduočiai ar problemai spręsti, sprendimų sėkmė priklauso nuo agento funkcionalumų išpildymo ir galimybių susidoroti su problemomis.
4. **Dirbtinis intelektas** – integracijos sprendimuose pasireiškia aktyvus dirbtinių neuroninių tinklų algoritmų naudojimas.
5. **Kiti sprendimai** – įvairūs mišrūs sprendimai integracijos ir sąveikumo problemoms spręsti.

Lentelė 2. Žinomi veiklos programinės įrangos sąveikumo sprendimai.

			Sprendimo metodologija (sudėtingumo lygiai)				
	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/ Atnaujinti/Trinti) funkcionalumas		R	D	D	D	D
	Tikslais grindžiama architektūra		R	R	D	D	D
Žinomos technologinės galimybės	**Autonominis subrendimo indeksai**		R	R	D	D	D
	Sprendimo priežiūra nepriklausoma nuo darbuotojų		-	-	A	A	D
	Autonominės galimybės	**Savęs redagavimo galimybė**	-	-	D	D	D
		Sveikimo / Pasitaisymo galimybė	-	-	D	D	D
		Optimizacijos galimybė	-	-	D	D	D
		Savisaugos galimybė	-	-	D	D	D

Lentelėje (žr., lentelė 2) analizuojami sprendimai neturi nei automatizavimo ir savęs valdymo galimybių. Dauguma analizuotų autorių šaltinių nepateikia konkrečių duomenų skirtų atkartoti jų pasiektus rezultatus ir įvertinti jų sukurtus algoritmus ir metodus taikomųjų programų integracijos automatizavime. Lentelėje pateikiamos kai kurios tikrintos savybės kurios pasiekiamos konkrečia metodologija.

Automatizavimo aspektu didelė problema yra gilių žinių apie integruojamas sistemas nebuvimas. Dažnai įgyvendinti integracijos sprendimai yra įgyvendinti juodosios – dėžės modeliu: matomi įvedimo ir išvedimo kintamieji, neaišku kokie priežastiniai ryšiai vyksta pačiose sistemose, ekspertas supranta šiuos ryšius, nes gali rezultatus pasitikrinti pačioje sistemoje, tačiau automatizuojant to padaryti nepavyks neturint gilių žinių.

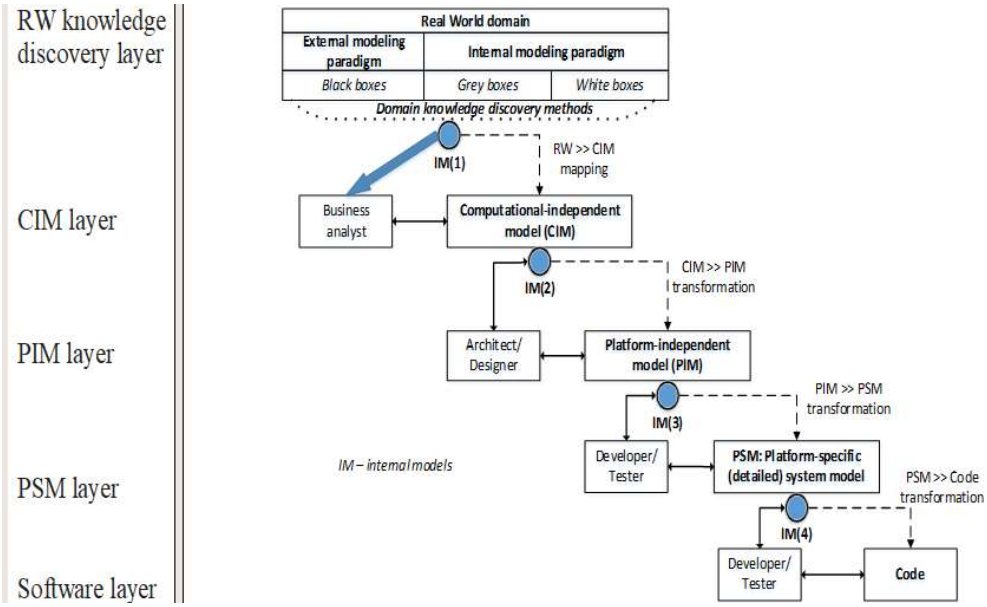


Paveikslas 7. Vidinio modelio naudojimas kuriant baltosios-dėžės sprendimus.

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

Tyrinėjant taikomųjų programų integracijos sritį pastebėta, kad integracijos problemų yra didelis sąrašas, todėl nuspręsta siaurinti temą analizuojant ir gilinantis į vieną iš problemų - programų integracijos ir sąveikumo galimybių vertinimas. Kuomet reikia įvertinti ar sistemos gali būti integruojamos arba sąveikauti tarpusavyje. Išnagrinėti literatūros šaltiniai kurių gauta informacija leido identifikuoti pagrindinių problemų taikomųjų programų integracijos srities sąrašą. Analizuojant autonominės kompiuterijos galimybes ir struktūra pastebėti panašumai tarp veiklos procesų valdymo modelio struktūros elementaraus valdymo ciklo (EVC). Panašumai leido išskirti bendras savybes kuriomis remiantis galima būtų išgauti reikiamą informaciją iš veiklos procesų panaudojant ją integracijos kūrime.

Modifikuota MDA schema (žr., pavekslą 8) – ji pagrindžia „domeno sąveikų gilumines žinias“ ir veiklos procesų atvaizdavimą į programų architektūrą.



Paveikslas 8. Modifikuota MDA schema pagrindžianti vidinio modelio naudojimą sistemų analizei.

2.5. Taikomųjų programųtinklo paslaugos

Veiklos sistemos sukurtos naudojantis paslaugomis grindžiama architektūra (angl., service oriented architecture, SOA) lemia kad sistemos esybės ir operacijos pateikiamos kaip paslaugos kurios gali būti prieinamos tinkle, kitaip, 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 apjungtus duomenų atvaizdus (angl., views). Žemiau pateikiama lentelė atspindinti tinklo paslaugų panaudojimą naudojant autonominių skaičiavimų metodus.

Lentelė 3. Tinklo paslaugų ir automoninio skaičiavimo metodo – privalumai, trūkumai

Tinklo paslaugos	**Autoniminių skaičiavimų metodas**
Valdymą atlieka patyrusi komanda, reikiant pakeitimams daromos modifikacijos integracijos sistemose, kad patenkintų poreikius.	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	Maži modifikavimo kaštai
Jungiasi prie iš anksto numatytų servisų tuo pačiu metu	Gali prisijungti prie duomenų šaltinių pagal poreikius taisykles ir apribojimus.
Jungimosi prie servisų orchestravimas atliekamas programuojant, naudojant EAI įrankius.	Orchestravimas gali būti atliekamas autonomiškai išanalizavus veiklos procesų (BP) žinias
Turi galimybe aktyvuoti ir išjungti servisus jei tai yra įgyvendinta integracinėje sistemoje.	Galimybė autonomiškai 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 kokios gali iškilti problemos

2.6. Darbai vertinantys taikomųjų programų sąveikumą

El-Halwagi ir kt. [5] daugiausia dėmesio buvo skiriama procesų integracijai, pabrėžiant integracijos problemų pagrindus. Y. Peng ir kt. [30] daugiausia dėmesio buvo skirta daugiafunkcinei įmonių integravimo sistemai, siekiant išspręsti su verslo suderinamumu susijusias problemas, nepateikė informacijos apie integracijos valdymą. R. McCann ir kt. [36] buvo pabrėžiamos duomenų integravimo taikomųjų programų priežiūros žemėlapių nustatymo problemos. X. Luna Dong, et al. [39] atliko duomenų integravimo tyrimą, pabrėžė dažniausiai pasitaikančias problemas, susijusias su integracijos tema 2006 metais. E. Rahm ir kt. [21] pasikartojanti apklausa parodė integracijos dalyko patobulinimus ir tai, ko dar trūksta toje srityje. Autoriai analizavo dabartines tendencijas įmonių programinės įrangos taikomųjų programų integravimo problemose.

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

Mažiausias programinės įrangos sistemos integravimo lygis yra duomenų integravimas ir schema. Schemos suderinimo metodai padeda kurti tokius algoritmus, kurie gali susieti skirtingų programinės įrangos

duomenų šaltinius. Mes palyginame įprastus "IBM autonominio brandumo indekso (angl, autonomic maturity index - AMI) " programinės įrangos taikomųjų programų integravimo metodus ir tokių integravimo taikomųjų programų kūrimo sudėtingumo lygį.

2.7. Sąveikumo laipsnių tipai

Sąveikumas vertinimas gali turėti kelis tipus: sąveikumo galimybių, suderinamumo, sąveikumo efektyvumo (angl., interoperability potentiality, compatibility, performance) N. Daclin, D. Chen (2006) [2].

2.8. Integracijos ir sąveikumo standartai

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.

2.9. Autonominio skaičiavimų komponentai

Autonominio skaičiavimo technologija sukurta įmonės IBM. Šios technologijos viziją aprašė Jeff Kephart (2003 m.) [13]. Ši technologija skirta kelti automatizavimo ir autonominių skaičiavimų lygį. Autonominio skaičiavimo pagrindai aprašomi IBM straipsniuose juose pateiktos pradinės idėjos apie autonominių skaičiavimų savybes ir funkcijas. Jeff Kephart atstovauja IBM tyrimų centrui jo specializacija autonominiai skaičiavimai, netiesinė dinamika ir mašininis mokymasis. Pagrindinis straipsnis, kurį rašė kartu su DM Chess – autonominių skaičiavimų vizija: „The vision of autonomic computing“ 2003 metais.

Autonominio skaičiavimo metodologija susideda iš šių komponentų:

- Sąsajų (angl. Touchpoints)
- Žinių šaltinių
- Autonominių valdiklių (angl. autonomic managers)
- Valdomų išteklių (angl. managed resource)

Tokia komponentų sandara pritaikyta kuriant eksperimentą. 5 Skyriuje aprašytame eksperimente autonominio skaičiavimo pagrindu kuriama technologija turi sąsajas – tai, taikomųjų programų sąsajų sąrašas aprašomas žmogaus, bet jį toliau nodoja sukurtas sprendimas. Žinių šaltinis kuriamas analizės eigoje, siūlomame sprendime kaupiami du žinių tipai:

- Sąsajų struktūra, jų įvesties, išvesties kintamieji, panašumų analizės duomenys.
- Autonominio sprendimo analizės metu vykstančių procesų aprašymai.
- Valdomieji ištekliai yra sąsajų lentelėje aprašytos programos.

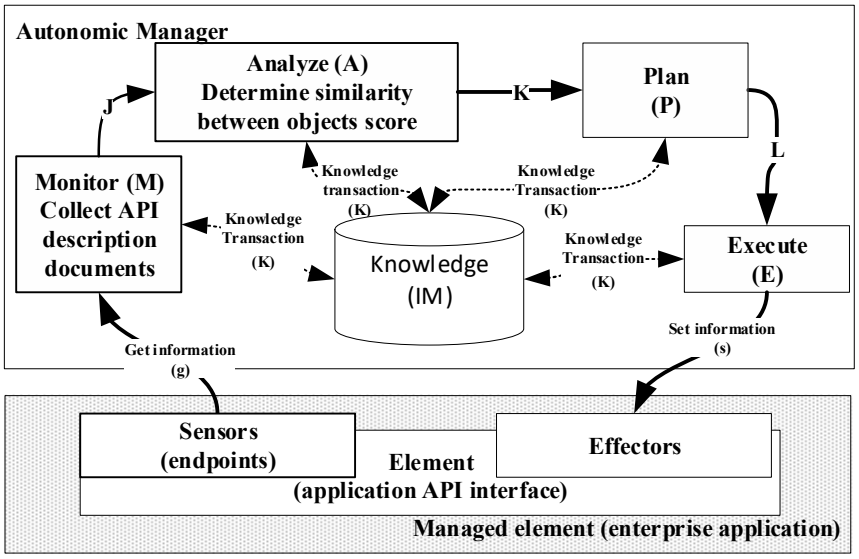
Teigiama jog autonominių skaičiavimų technologija turi 4 pagrindines ir kelias šalutines savybes. Pagrindinės savybės susideda iš:

- 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 remiasi tik savistabos galimybe.

Bendros paskirties autonominio skaičiavimo technologija remiasi nuo analizuojamos domeno srities nepriklausomais modeliais. Sukurti meta-modeliai naudojami taisyklių variklio (angl., policy engine) konfigūravimui. Nuo domeno srities nepriklausoma architektūra išskaidoma į dalis kurios apibrėžia bendrosios paskirties autonominę architektūrą su keičiamomis taisyklėmis ir į ICT sistemos meta-modelį su sąsaja į šios ICT architektūrą ir taikomųjų programų modelių kūrimo įrankius. Sekančiame etape apibrėžiami specifinei veiklos sričiai būdinga ICT ontologija ir taip pat apibrėžiami 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 pilnai apibrėžtos autonominės taisyklės apimančios valdomos sistemos modeliavimą ir aukščiau išvardintus apibrėžimus (grįžtamasis ryšys).

IBM autonominio skaičiavimo technologijos iliustracijoje (Paveikslas 9) pateikiami 4 pagrindiniai žingsniai reikalingi autonominiam komponentui: stebėjimas, analizavimas, planavimas, vykdymas. Šiems komponentams ir jų veikimui žinių komponentas suteikia reikalavimus ir informaciją. Autonominis valdiklis valdo resursus per sensorius ir efektorius. Šio tyrimo atžvilgiu tai yra tinklo serviso sensoriai ir efektoriai.



Paveikslas 9. IBM Autonominio skaičiavimo komponento architektūra [11]

Priede 2 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 dalis kiekvieno autonominio valdiklio. Jei žinios dalinamos tarp autonominių valdiklių tuomet jos talpinamos bendros paskirties žinių šaltinyje. IBM autonominio skaičiavimo principų aprašyme išskiriami keli žinių tipai:

- Sprendimo topologijos – aprašo komponentus ir jų konstrukciją ir sprendimo arba verslo sistemą.
- Taisyklių - aprašo ar reikalingi pakeitimai sistemai, 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 tikrų problemų sprendimui.

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

2.9.1. Programų sąsajos

Sąsajos (angl., Touchpoints) tai komponentai, kurie 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, duomenų bazes tame serveryje ir lenteles tose duomenų bazėse.

2.10. Skyriaus išvados

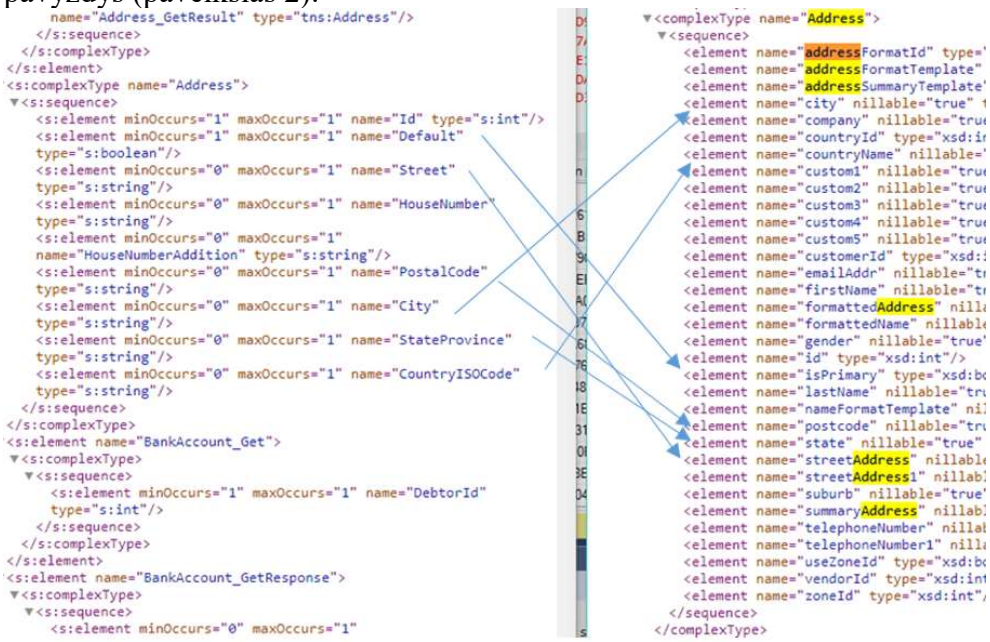
Apžvelgta daug literatūros šaltinių nagrinėjusių taikomųjų programų integraciją ir sąveikumą. Tema yra vis dar aktuali ir sukelia daug diskusijų, nes ne visos įvardintos integracijos ir sąveikumo problemos yra išspręstos. Dalis iš išpręstų problemų vis dar nėra plačiai įsisavinamos versle. Įvardinti taikomųjų programų integracijos tipai, ir paaiškinti sąveikumo principai. Įvardintos integracijos ir sąveikumo problemos. Aprašyti autonominiai skaičiavimų agentai.

- Esami metodai:
    - aplinkos aspektų įvertinimas naudojant vertinimo korteles (angl. scorecard): LISI [12],
    - naudojant I-Score metodologiją [7],
    - palyginimas pagal funkcionalumą [4].
- Sintaksinis įvertinimas atliktas atliekant teksto analizę naudojant redagavimo nuotolio skaičiavimus, kurie leidžia įvertinti duomenų struktūrų panašumą tarp skirtingų objektų.

Semantinis įvertinimas atliktas palyginant operacijas, objektus ir duomenų laukų pavadinimus naudojant „*www.schema.org*„ tinklapyje aprašytas ontologijas.

3. Veiklos programų sąveikumo matavimai

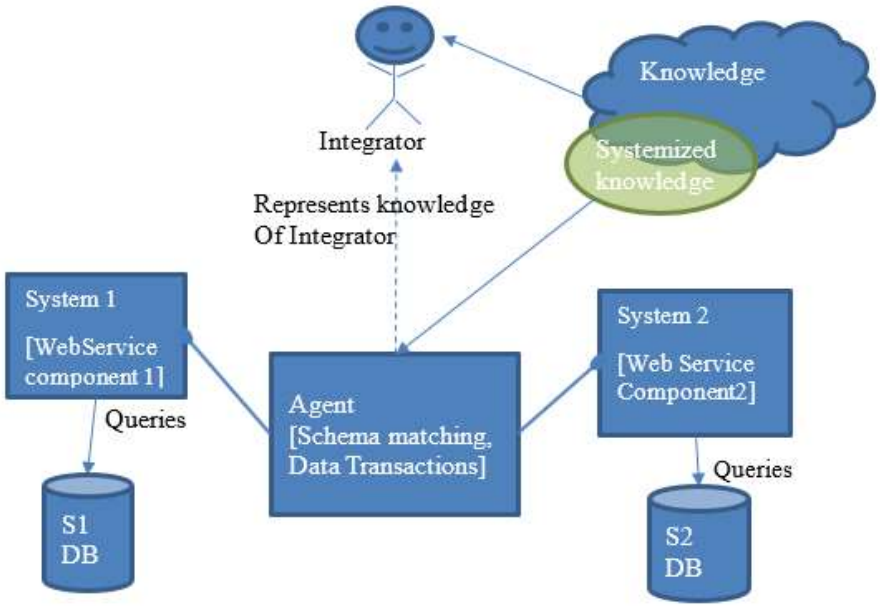
Toliau pateikiamas skirtingos duomenų struktūros apjungimo grafinis pavyzdys (paveikslas 2):



Paveikslas 10 Skirtingų duomenų struktūrų apjungimo grafinis pavyzdys

Paveiksle pavaizduota dviejų skirtingų nutolusių taikomųjų programų duomenų struktūros kurias reikia apjungti. Esant griežtai statinėms struktūroms, struktūrų apjungimas vykdomas programiškai arba EAI (angl., enterprise application integration) taikomųjų programų pagalba. Norint automatizuoti duomenų struktūros procesus tyrėjai dirba prie algoritmų kurie galėtų atpažinti naudojamus raktažodžius, duomenų tipus, spręstų semantinius esybių ir objektų atpažinimo

uždavinius [5, 35]. Dalis integracijos automatizacijos algoritmų realizuoti agentų pagalba.



Paveikslas 11. Autoriaus taikomųjų programų integracijos pavyzdys naudojant agentines technologijas.

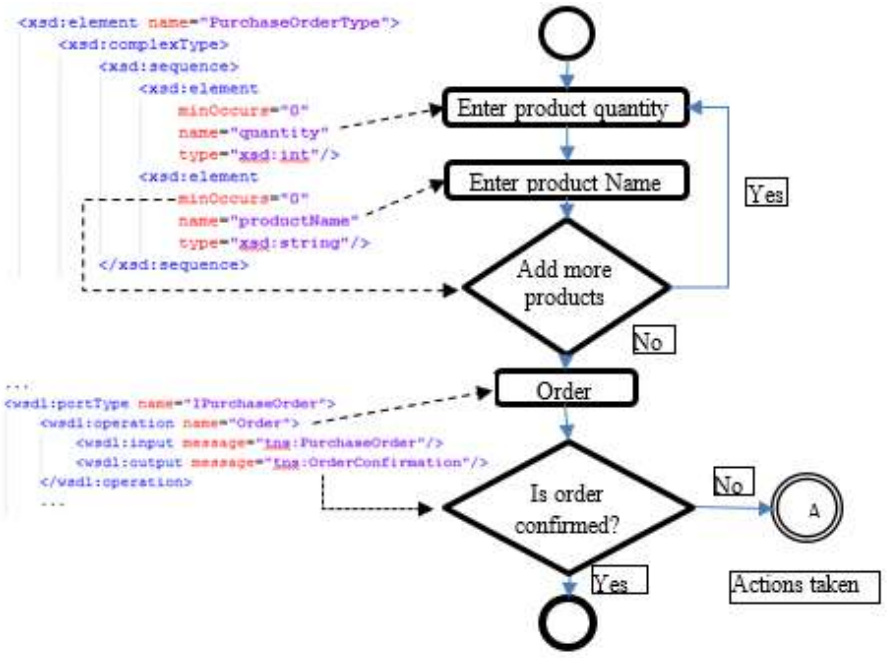
Taikomųjų programų integracijos pavyzdys naudojant agentines technologijas. Integracijos projektuotojas „Integrator“ sukuria agentą su iš anksto aprašytomis integracijos taisyklėmis, sisteminėmis žiniomis apie integruojamas programų sistemas ir jų savybes. Integracijos projektuotojas pats naudoja žinias apie verslo procesus, integruojamas sistemas jų struktūra, ir kuria tokį agentą kuris išnaudodamas šias žinias turės tam tikrą judėjimo laisvę esant pokyčiams tarp taikomųjų programų, duomenų vėlavimui ar kitoms problemoms. Pats agentas nesupranta nesuprojektuotų probleminių situacijų, nes jis neturi konkretaus verslo procesų modelio ir nesupranta integruojamų taikomųjų programų pavyzdžiui „System 1“ ir „System

2“ struktūros (žr. paveikslą 11). Esant dramatiškiems pokyčiams už agento suvokimo (agente suprogramuotu sisteminių žinių) modelio, agentas nustos veikti ir verslo procesai nebevyks arba bus sutrikdyti kol agentas nebus atitinkamai pakoreguotas.

Norint sužinoti ar įmanoma automatizuoti ar pagerinti programų integraciją ir sąveikumą reikia išmatuoti kiekvienos tiriamos sistemos panašumą kitai sistemai.

3.1. Siūlomas autonominis integracijos sprendimas

Verslo taikomųjų programų integracijos metodas gali būti papildytas autonominės kompiuterijos technologija kuri analizuotų tinklo servisus ir tyrinėtų jų veiklą, tyrinėtų jų duomenų struktūrų dokumentus (WSDL, paveikslas 12).



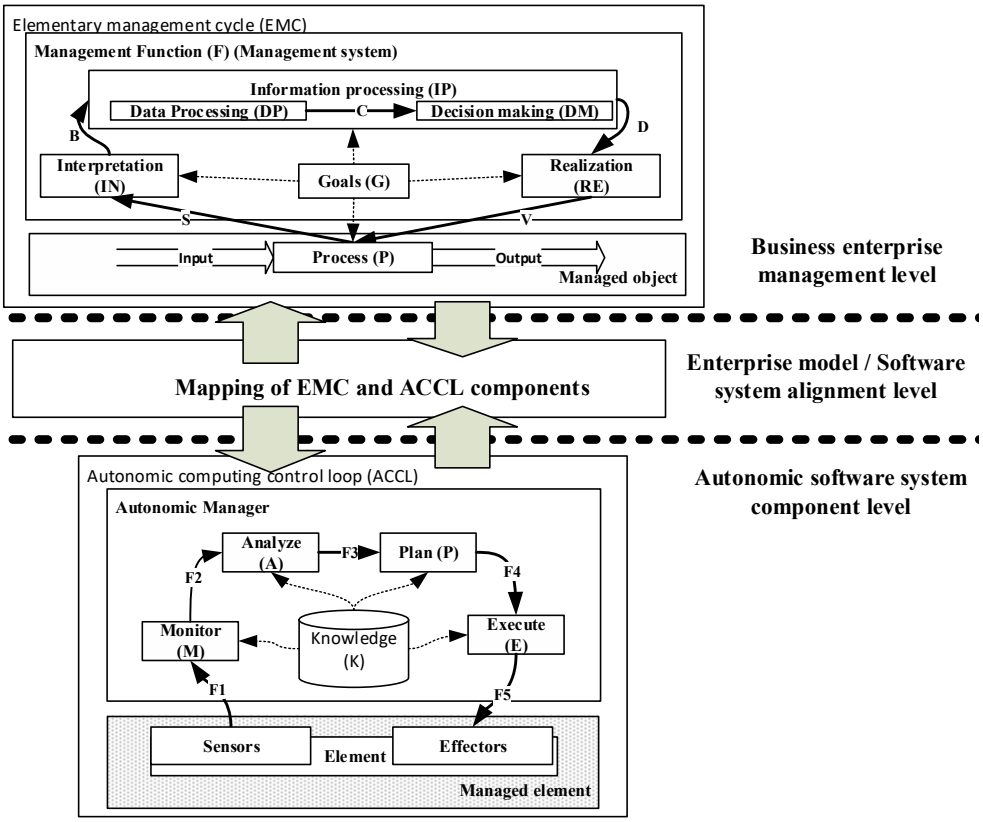
Paveikslas 12. Užsakymų WSDL schemos lygiavimas su veiklos proceso modeliu

Pavyzdyje (paveiksle 12) pateikiama dalis **užsakymo sukūrimo proceso** iš vieno verslo subjekto. Šis verslo procesas pavaizduotas veiksmų sekos diagramos iškarpa (paveikslas 12). Paveiksle vaizduojama kaip konkrečios programos sistemos serviso 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), tuomet teoriškai galima butu sukurti toki integracijos metodą kuris galėtų suprasti savo kontekstą ir vidinę sandarą. 12 Paveiksle, raktažodžiai “Order”, “Confirmation” išgaunami iš verslo veiklos proceso modelio. Algoritmas paremtas autonominės

kompiuterijos principais išanalizuotų elementus, žinutes ir operacijas iš WSDL dokumento. Naudojant savaime apsimokanti dirbtinį intelektą simuliuotoje erdvėje galima išmokinti autonominį integracijos algoritmą naudotis šia informacija, kad algoritmas galėtų keisti savo parametrus prisitaikydamas prie konkrečios struktūros.

Toliau analizuojamas autonomines kompiuterijos komponento [11] ir elementaraus valdymo ciklo EVC [32] panašumai dėl kurių buvo

iškelta teorinė galimybė taikomųjų programų integracijos automatizavimui panaudoti autonominės kompiuterijos principus.



Paveikslas 13. Autonominės kompiuterijos komponento (AC) [11] palyginimas (apačioje) su elementariu valdymo ciklu (EVC) [32] (viršuje)

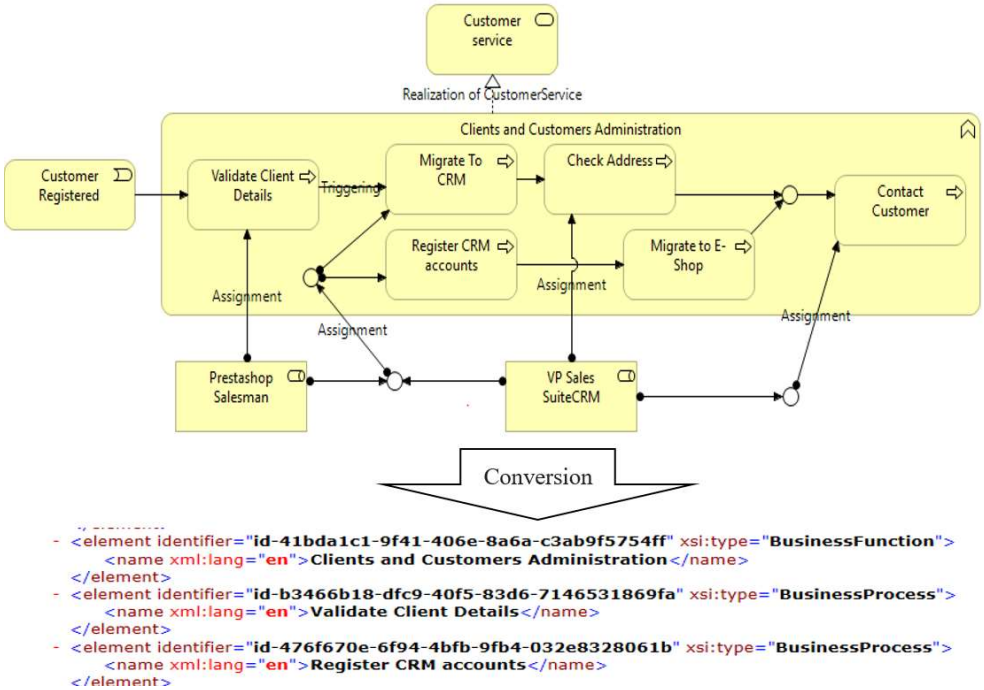
Elementaraus valdymo ciklas (Paveikslas 5 EVC) buvo pristatytas prof. dr. Sauliaus Gudo [3] kaip bazinis elementas verslo veiklos valdymo modeliavime. Valdymo funkcija (F) yra sudėtinė struktūra kuri remiasi tikslais grindžiamais informacijos transformavimo žingsniais kuri susideda iš: IN – interpretavimo, 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, energija) ir išeigos komponentais (produktais, paslaugomis). Valdomąjį objektą valdo valdančioji funkcija F. Elementarus valdymo ciklas ir autonominės kompiuterijos valdymo ciklas [11] yra panašūs, nes buvo kuriami realaus pasaulio veikimo principais. Tikslais grindžiami informacijos apdorojimo ciklai tarp EMC 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 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 EMC ir ACCL atsakingi už žemesnių lygių 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 autonominės kompiuterijos pagalba, kas padėtų sukurti naują modelį taikomųjų programų integracijos problemoms spręsti.

AC ir EVC palyginimo paveiksle pavaizduota schema sujungta dirbtiniu būdu, tačiau modeliuojant veiklos architektūrą dažniausiai naudojamasi įvairiomis priemonėmis. Todėl antrai daliai naudota

„ArchiMate“ veiklos modeliavimo priemonė leidžianti išeksportuoti standartizuotą modelio failų formato dokumentą (angl., MEFF – model exchange file format) sukurtą ArchiMate platformoje ir standartizuotą OMG (principinė diagrama pavaizduota paveiksle 13). Šiuo metodu modelį lengva formatuoti XML formatu ir analizuoti su kitomis sisteminėmis priemonėmis.



Paveikslas 14. MEFF veikimo principinė diagrama

Tyrimo metu ištyrinėtos prielaidos, kad panaudojant ir taikant organizacijų veiklos modelius visada remiamasi vidiniu modeliu su būtinai esančia hierarchine struktūra. (pvz. modeliuojant realų pasaulį, kuriant veiklos procesų modelį, kuriant taikomųjų programų procesų modelį ir kuriant programos architektūrą visuomet yra taikomas vidinis modelis su ankstesniuose modeliuose įgytomis žiniomis.)

Pagrindiniai metodai, kurie dažniausiai skleidžiami (programiškai sukurti integravimo sprendimai), neturi tikslų, dažniausiai jie 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 skirtingo laiko, 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 stadijoje. Neveikia dinamiškai, kai įdiegiami sprendimai, paprastai jie turi būti išlaikyti, o integravimo procesai kartais turi būti sustabdyti. Vienas iš privalumų: dizaineris gali grafiškai matyti ir stebėti procesus. EAI projektai taip pat apsiriboja vienos projekto apimtimi. Klaidos nustatomos ir užregistruojamos, jei jas įgyvendina pagal dizainą. Gali būti arba surinkti žurnalus apie integracijos procesą, žurnalai apsiriboja duomenų šaltinių rezultatais, tik maža suma gaunama iš EAI sistemos. Manoma, kad EAI sprendimai nepavyksta 70 proc. [26]. Dėl kritinės klaidos EAI sprendimai nutrūksta ir nustoja veikti, kol jų neištaisys integratoriai.

Integracijos agentai yra struktūros, sukurtos ar sukurtos asmeniu, kurį galima pritaikyti ir konfigūravimo priemonę, kad atitiktų integracijos poreikius. Norėdami įdiegti ir konfigūruoti integracijos agentą, reikia specialių techninių žinių apie integracijos agentus. Kai kuriems agentams reikia tinkamai nustatyti daug žinių (daugiausia programavimą) [10]. Kitas agentas gali būti lengvai paruoštas, bet nėra pritaikytas konkretiems verslo poreikiams. Konfigūruoti agentai

reikalauja pritaikymo, jei schema pakeista iš bet kurio duomenų šaltinio. Po to, kai konfigūravimo agentai gali būti galingi įrankiai, jie gali bendradarbiauti su kitais agentais, dalydami informacijos analizės duomenis. Funkcionalumas gali pagerėti prijungus kitus agentus. Kai kurie agentai remia planavimą ir vykdymą integruojant procesus, tačiau neaišku, kokių veiksmų reikia šiam agentų technologijos įgyvendinimui [10], [15], [30]. 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 jos elgesį. Skiriant programuojamus integracijos sprendimus, reikia atlikti techninius ir patyrusius darbuotojus. Programuojami integracijos sprendimai paprastai orientuojasi į mažesnes integracijos formas: schema suderinama [21]. Ne visi integracijos agentai gali prisitaikyti prie pokyčių.

Kitame išplėstiniame sprendime pateikiami dinaminių procesų integravimo būdai, kartais susiję su bendradarbiaujančių organizacijų veiklos procesų susiejimu – kitaip vadinama B2B procesų integracija [5], [20], [22], [27], [33, 34]. Tokie metodai sprendžia pagrindines verslo procesų integravimo problemas, dauguma jų yra išbandytos ribotoje ir sukomplektuotoje aplinkoje, tokių metodų rezultatai buvo perspektyvūs, tačiau jie dažniausiai yra teoriniai. Nežinomas kokia yra

pateiktų sprendimų struktūra, išskyrus semiotinį organizacinio modeliavimo metodą, naudojant normų analizę [34].

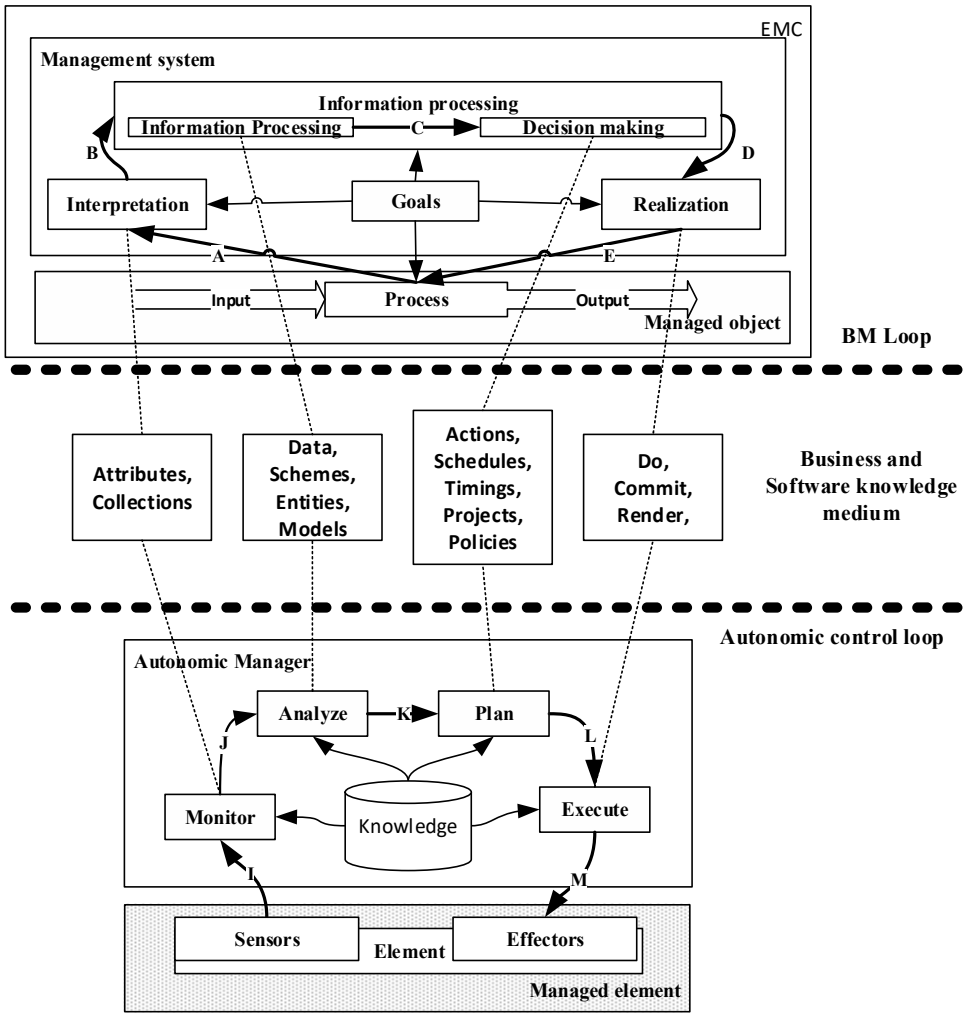
3.2. Veiklos aplikacijų integracijų principai

Taikomųjų programų sąveikumo metodas su autonominio skaičiavimo technologija analizuoja tinklo servisų šaltinius stebint jų veiklą ir analizuojant jų duomenų struktūros pakeitimus sistemose. Pavyzdžiui duotam tinklo servisui, procesai kaip manipuliuoti duomenimis gali būti išgaunamas iš WSDL failų arba aprašomųjų REST protokolo failų. Paveiksle 13 atvaizduojama kaip schema yra sutapatinama su procesu. Jei autonominio skaičiavimo žinių elementas turėtų veiklos procesų žinias (modelių, formalių aprašymų pavidalu panašiai kaip „*https://schema.org/order*“) tuomet pirmiausiai toks sąveikumo metodas galėtų turėti savęs pažinimo galimybę (angl., self aware) ir suprasti duomenų kontekstą iš veiklos procesų perspektyvos. Antra, veiklos procesų grandinė gali būti atpažįstama kuri leidžia valdyti orchestravimo ir choreografijos veiksmus sąveikaujančiuose komponentuose. Galiausiai kiekvienos sistemos schema galėtų būti palyginama su modeliais egzistuojančiais žinių bazėje ir galima būtų apibrėžti naujus arba trūkstamus elementus.

Paveikslas 13 vaizduoja įsivaizduojamą modelių apjungimą 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 įvardinti elemento „PurchaseOrderType“ lauke ir užsakymą patvirtinanti žinutė gražinama elemento „OrderConfirmationType“ lauke. Naudojant vien tik bandymo ir klaidų metodu, simuliuotoje aplinkoje galima ištreniruoti išmanų autonominio skaičiavimo elementą, šis apmokymas ir būtų žinios apie struktūrą sąveikaujančių taikomųjų programų. Taikomoji programa dažniausiai neturėtų leisti kurti užsakymo su atsitiktiniais produktais neegzistuojančiais sistemoje ir todėl klaidos žinutė turėtų būti gražinama. Klaidos žinutės gali būti analizuojamos ir sekančiame scenarijuje reiškia, 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.

Teoriškai informacijos taikomųjų programų pagrindai apibrėžia elementarius valdymo ciklus (EVC) [32]. Elementarus valdymo ciklas susideda iš valdančiosios sistemos (angl., management system) ir valdomo objekto (angl., managed object). Valdančioji sistema apibrėžia tikslus, interpretavimą, informacijos analizę ir talpinimą bei sprendimų priėmimą ir realizavimą. 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 jie buvo projektuojami remiantis realaus pasaulio modeliavimo principais. Esminis skirtumas, kad EVC labiau atsižvelgia į materialius proceso srautus, o šioje disertacijoje daugiau dėmesio skiriama informaciniių procesų srautams. Panašumai tarp EVC ir ASVC pateikiami lentelėje žemiau:



Paveikslas 15. EVC (paveiksle EMC) ir ASVC palyginimai (Autonomic control loop)

Paveiksle 15 komponentai EVC ir ASVC kurie 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; ir tikslai ir žinios susiję su valdymu ir kontroliavimu visų kitų komponentų; abu yra cikliniai ir rodo, kad metodologijos dinamiškos ir gali turėti pagrindines savybes.

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

3.3. Sąveikumas naudojant autonominio skaičiavimo technologijas

Autonominio skaičiavimo idėja pristatyta Jeffrey O. Kephart ir David M. Chess, 2003 metais, kaip būdas susidoroti suvis didėjančiu verslo programų kompleksiškumu [13]. Aukščiausio lygio autonomiškumas pasiekiamas nustatant tikslus ir apribojimus. Idealiu atveju neturėtų būti reikalaujama techninių įgūdžių turinčių žmonių įsikišimo. Gali būti, kad autonominiai sprendimai reikalautų daugiau valdymo lygio žinių turinčių darbuotojų keičiant nustatymus, tikslus ir apribojimus. Šiuo metu net su autonominės kompiuterijos pagalba reikia nurodyti prieigos taškus prie kiek vienos sistemos ir kai kurias sistemas paruošti sąveikavimui, pavyzdžiui įjungti tam tikrus tinklo servisų 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, apsisaugojimui ir atsistatymui esant trikdžiams ar atakoms.

Pagrindinė mintis naudoti autonominio skaičiavimo komponentus yra tai kad jie turi keturias galimybes: nusistatymo, optimizacijos, pasitaisymo, apsaugos. Šios galimybės yra autonominių komponentų dedamieji tikslai. Autonominiai komponentai padeda išskaidyti sudėtingas problemas į mažesnes leidžiant atskirti valdomą komponentą ir autonominį valdiklį. Autonominis valdiklis pagal apibrėžimą turi turėti procesus leidžiančius stebėti, analizuoti, planuoti ir vykdyti užduotis. Šie procesai tarpusavyje susieti su žinių komponentu kuris saugo reikšmingą informaciją surinktą kiekvieno proceso metu ir leidžią šia informaciją panaudoti kitiems komponentams. Visas autonominio skaičiavimo komponento veikimas yra ciklinis ir vadinamas autonominio skaičiavimo valdymo ciklu – ASVC. Apjungiant skirtingus ASVC leidžia kurti stiprias sistemas kurios sprendžia kompleksines užduotis su tam tikru autonomiškumo lygiu. Autonomijos lygiai 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 anksčiau išvardintų 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.

3.4. Redagavimo nuotolio skaičiavimai

Redagavimo nuotolio (angl., edit distance) skaičiavimai leidžia apskaičiuoti skirtumus tarp žodžių tekste. Mes skaičiuojame redagavimo nuotolį pagal tai kiek reikia atlikti vienam žodyje pakeitimų, kad jis sutaptu su kitu žodžiu. Traktuojama, kad 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 sisemų panašumo vertinimas remsis tokia formuluote:

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

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

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

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

Yra daug būdų skaičiuoti redagavimo nuotolio įverčius. Šioje disertacijoje pargrinde naudoti 4 redagavimo nuotolio būdai [17]:

- Levenshtein
- Jaro – Winkler
- Ilgiausios bendros sekos (anlg. longest common subsequence)

- Jaccard

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

(1) $d_{i0} = \sum_{k=1}^{i} \omega_{del}(b_k)\ , \qquad for\ 1 \le i \le m$

(2) $d_{0j} = \sum_{k=1}^{i} \omega_{ins}(a_k)\ , \qquad for\ 1 \le j \le n$

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

Jaro-Winkler metodu skaičiuojama kiek panašių atributų (simbolių junginių) yra lyginamuose žodžiuose. Skaičiuojama kiek simbolių junginių reikia transponuoti siekiant sutapatinti pateiktus žodžius.

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

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

(6) $sim_w = sim_j + (lp(1 - sim_j))$

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

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

Jaccard metodu apskaičiuojama kiek panašių atributų yra abejuose lyginamuose žodžiuose, suskaičiuojamas bendras atitinkamų simbolių skaičius.

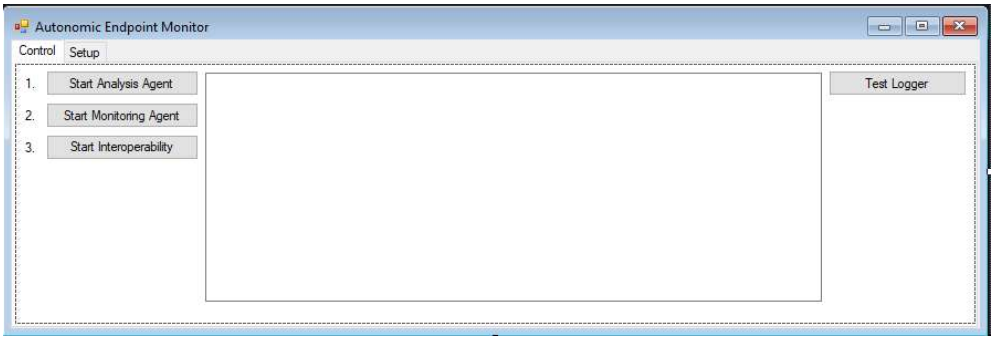
(8) $J = \frac{M_{11}}{M_{01}+M_{10}+M_{11}}$

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

Naudojant šias teksto palyginimo formules galime atlikti teksto analizę ir su tam tikrų automatizavimo lygiu spręsti apie taikomųjų programų panašumą.

4. Sąveikumo galimybių vertinimo eksperimento aprašymas

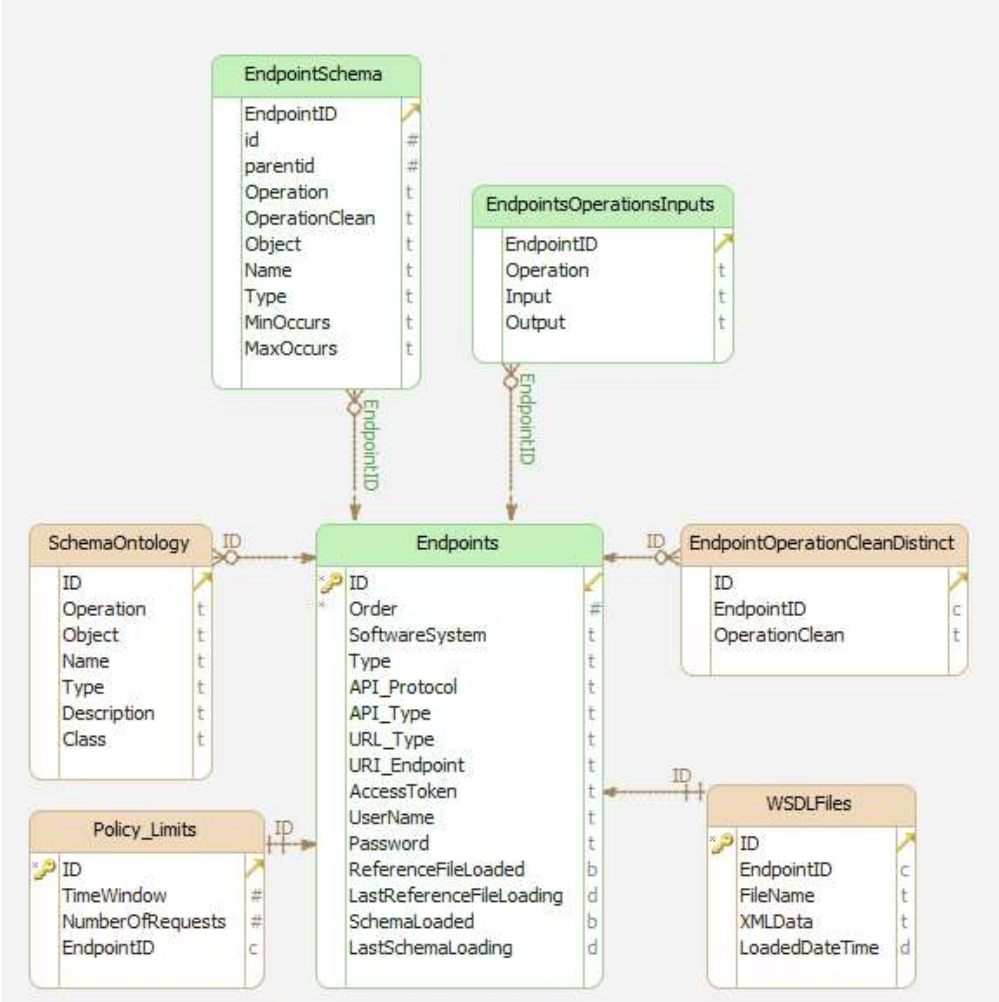
Šiame skyriuje nagrinėjamas eksperimento aprašyto šiame darbe procesas. Tyrimui, C# programavimo kalba, sukurtas įrankis leidžiantis išgauti, stebėti ir surinkti duomenis iš pateiktų duomenų šaltinių. Šie šaltiniai duomenų bazėje talpinami lentelėje „Endpoints“ (Paveikslas 17). Lentelė „Endpoints“ turi laukus kuriuose talpinamas programos pavasdinimas, 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, kuris analizuoja duomenų šaltinių nuorodas į tinklo paslaugos dokumentus.



Paveikslas 16. Taikomųjų programųšaltinių analizės vartotojo sąsaja.

Sukurta reliacinė duomenų bazė aprašanti duomenis skirtus autonominių užduočių įgyvendinimui. Reliacinės duomenų bazės tikslas išgauti ir išsaugoti analizės metu gautus duomenis kaip žinias – tolesnių etapų ir agentų veikimui užtikrinti. Naudojamos kelios reliacinės duomenų bazės: PostgreSQL ir Microsoft SQL server.

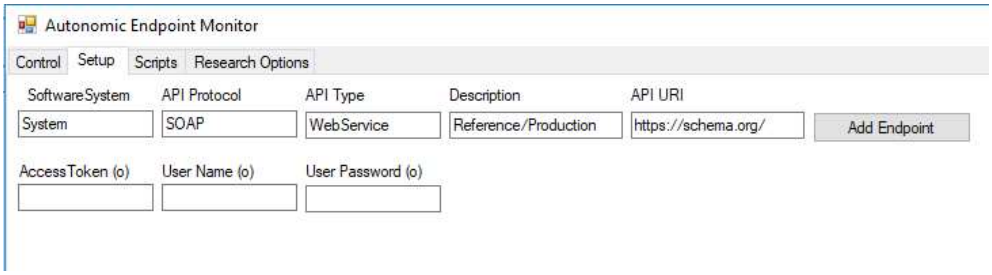
Sistema buvo pradėta projektuoti naudojantis nemokama Microsoft licencija, tačiau joje teko susidurti su vietos apribojimais todėl taip pat pasinaudota PostgreSQL reliacine duomenų baze.



Paveikslas 17. Duomenų bazės architektūra MSSQL serveryje.

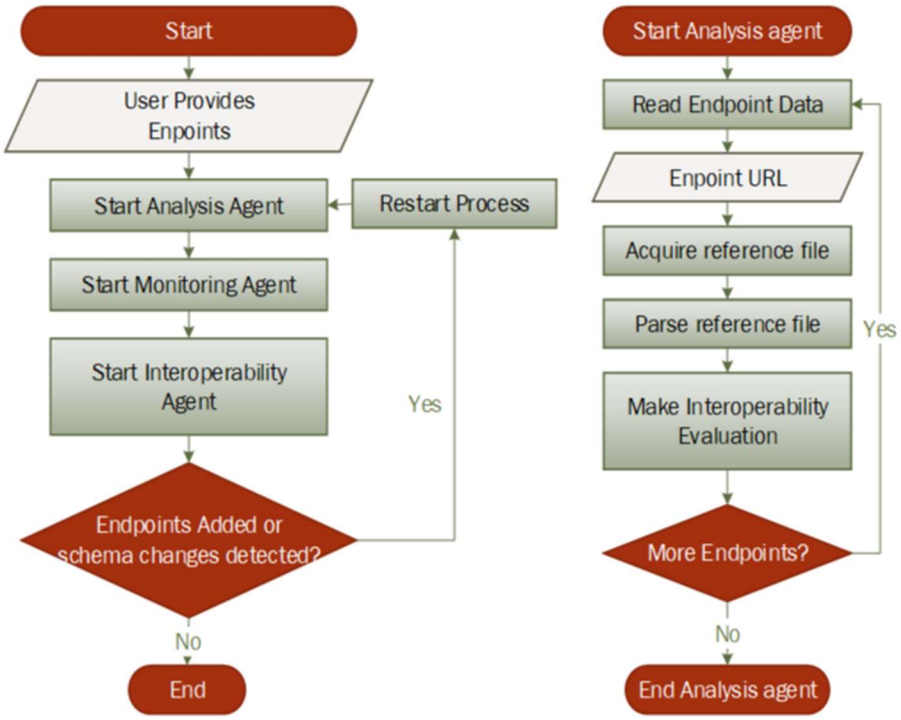
Duomenys išgaunami eilės tvarka skaitant sąsajų duomenis aprašytus lentelėje „Endpoints“ susisiekiama su tinklo servisu ir bandoma išgauti schemos dokumentą WSDL arba tiesiogiai išanalizuoti

programinės sąsajos šaltinį. Programinės įrangos pagalba verslo veiklos atstovas gali įvesti tinklo servisų adresus kurie išsaugomi lentelėje „Endpoints“. Nustatymų lannge „Setup“ galima įvesti reikiamus duomenis sistemos interfeiso aprašymui analizuoti (Paveikslas 18). Suvedus reikalingą informaciją (sistemos pavadinimą, tinklo sąsajos protokolą, apibūdinimą) ir pateikus nuorodą į sąsajos aprašą, bei prisijungimo duomenis jei tokie yra. Paspaudus mygtuką „Add Endpoint“ duomenys atsiranda „Endpoints“ lentelėje (Paveikslas 17). Iš čia agentas atliekantis analizę nusiskaito sąsajos adrese (API URI) laikomą aprašymą ir pradeda analizę.



Paveikslas 18. Nustatymų langas, pridėti tinklo interfeiso sąsajai.

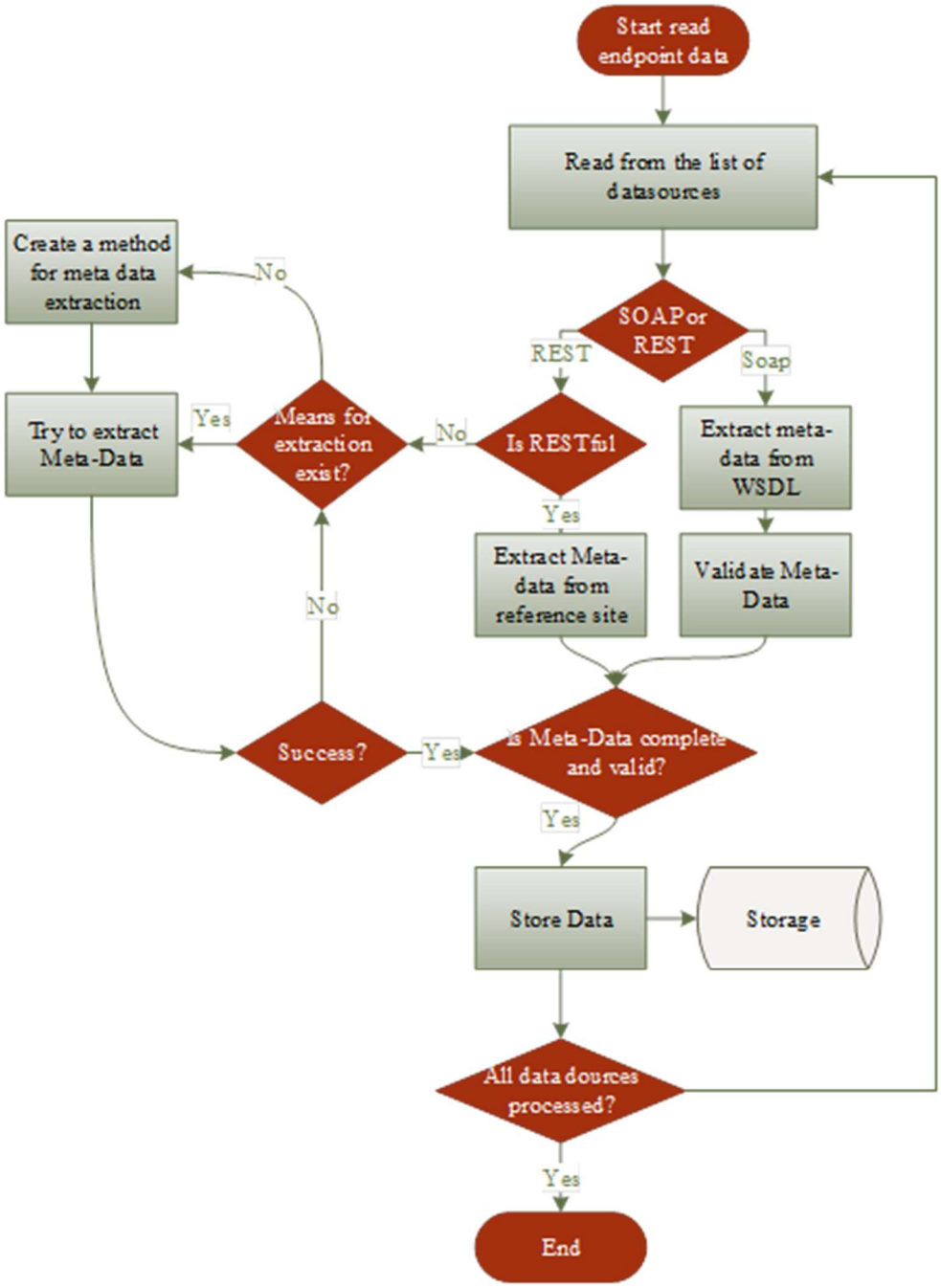
Analizės agentas skaito įrašus lentelėje Endpoints juos išanalizuoja, ir išanalizuoja tinklo servisų turinį ir patalpina jį „Endpoint Schema“ lentelėje.



Paveikslas 19. Analizės agento veiksmų diagrama.

Analizės agentas veikia skaitydamas visus aprašytus prieigos taškus ir analizuodamas jų struktūrą pagal pavaizduotą veiksmų seką (Paveikslas 19).

Duomenų bazėje aprašome lyginamų programų prieigos taškus, jų tinklo sąsajas. Autonominis komponentas kreipiasi į URI_Endpoint lauke nurodytą adresą, vykdo struktūros nuskaitymą (SOAP – WSDL arba REST HTML). Toliau autonominis agentas atlieka tokią veiksmų seką meta duomenų išgavimui:



Paveikslas 20. Serviso meta-duomenų nuskaitymas atliekamas šia veiksmų sekos diagrama.

Veiksmų sekos diagrama vaizduoja kaip atliekamas tinklo serviso meta-duomenų išgavimas ir nuskaitymas (Paveikslas 20). Analizės agentas surenka įmanomus duomenis, iš kurių vėliau gali planuoti tolesnį savo darbą. Analizės metu gauti duomenys talpinami duomenų talpykloje. 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 vikipedijos [35] taikomųjų programų palyginimų lentelių taip pat iš kitų šaltinių atitinkančių paieškos kriterijus: CRM API, ERP API, E-Commerce API, Accounting API. Surinktų duomenų palyginimų lentelė (tęsinys 1 Priede).

Lentelė 4. Sutrumpintas tiriamų objektų sąrašas (tęsinys 1 Priede)

Taikomoji programa	**Tipas**	**API**	**Licenzija**
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
...	...	...	...

Palyginimų lentelė skirta autonominiam analizavimo agentui. Analizavimo agentas tikrina ar visos sistemos turi schemas, t.y. galima ištraukti duomenų struktūras iš kiekvieno tikrinamo objekto. API (angl. application programing interface) – tai aplikacijų programavimo sąsąja kitaip, sąsaja per kurią pasiekiamos funkcijos ir duomenys. Šiame tyrime tirtos tik du plačiau paplitę API protokolai: SOAP ir REST (Lentelė 4). Surinktų duomenų sudėtį galima aprašyti šia forma:

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

Kitaip, visų sistemų schemos priklauso tik toms sistemoms. Negali egzistuoti schema kuri nepriklauso sistemai. Ištrauktas schemas agentas talpina į duomenų bazės lentelę „EndpointSchemas“ po schemos struktūros apdorojimo jeigu jis sėkmingas.

5. Sąveikumo galimybių vertinimo eksperimento rezultatai

Sąveikumo galimybių vertinimo eksperimentas remiasi prielaida, kad galima vertinti ar sistemos sąveikios pagal jų struktūros panašumus. Renkantis kokias programas reiktų analizuoti, remtasi verslo veiklos programomis sukurtomis remiantis servisais grindžiama architektūra SOA (angl. service oriented architecture). Pagrindinė koncentracija skiriama išskirtinai SOAP ir REST protokolams. Pradžioje, tinklo servisų operacijos yra palyginamos tarpusavyje. Šiam tyrimui buvo pasirinktos PrestaShop, ExactOnline, NMBRS ir SuiteCRM platformos. Kiekviena platforma turi skirtingas roles ir aspektus verslo veikloje.

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

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

SuiteCRM (S3) – Tai ryšių 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) – Tai 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 įrodyti jų galimas sąveikumo galimybes, išmatuoti sąveikumo potencialą ir pavaizduoti kurios tinklo servisų operacijos galėtų būti sąveikaujančios.

Pirmiausia, tyrimo metu bandyta panaudoti egzistuojantį LISI metodą programų sąveikumui vertinti. Jis sukurtas ir aprašytas M. Kasunic [12] 2001 metais bendradarbiaujant su amerikos saugumo skyriumi DoD (angl., department of defence). Ligi šiol šis metodas dažnai cituojamas ir papildomas kituose sąveikumo ir integracijų straipsniuose [51, 52].

Lentelė 5. Pasirinktų taikomųjų programų sąveikumo galimybių 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 5) matyti taikomųjų programų: a - atitikimą standartams ir b - sąveikumo įverčių lentelę kurioje G – atitinka standartus, tai būdinga SOAP protokolu naudojančiomis sistemoms, nes SOAP gerai aprašomas ir lengva išanalizuoti. Y – ne visai atitinka standartams, ir aprašymai dažniausiai skiriasi.

REST protokolas neturi griežtos aprašymų metodikos (tik rekomendacinę) arba rekomendacinė metodika nenaudojama. Iš sąveikumo palyginimų 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 funkcijų yra panašios.
- G – Green; Sistemos sąveikaujančios – įmanoma kad sistemos galėtų laisvai dalintis duomenimis ir funkcijomis.

Tokio vertinimo principu matyti, kad S1 (ExactOnline) turėtų gerai atitikti S4 (NMBRS), taip pat S3 (SuiteCRM) turėtų gerai sąveikauti su S2 (PrestaShop), (Lentelė 5).

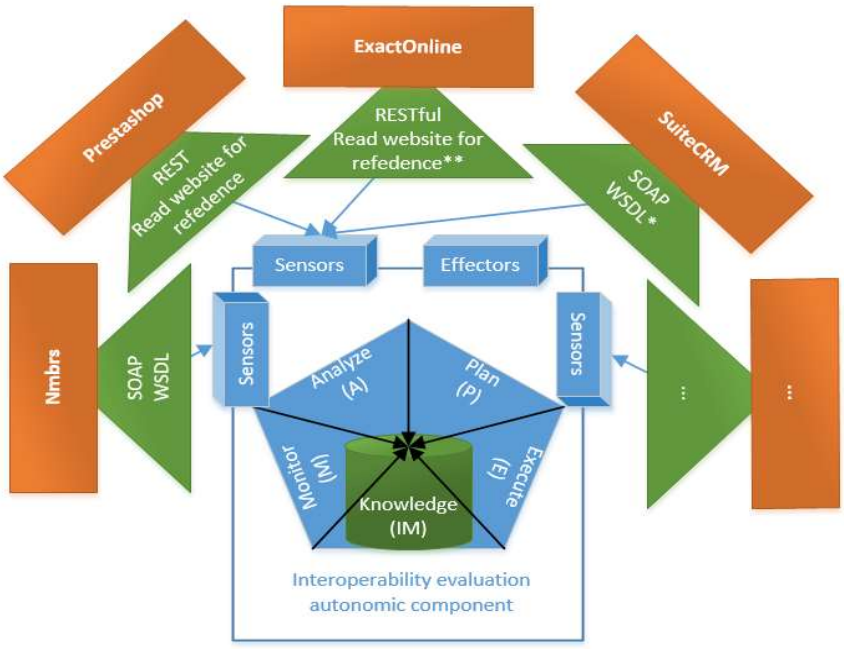
Nors LISI metodas gali būti patogus būdas įvertinti sąveikumo galimybių rezultatus, tačiau jis nebūtinai pateikia tikslią ir objektyvią informaciją apie tai kokie yra bendri duomenys. LISI vertinimas taip pat gali užtrukti daug laiko, jo metu reikia įvardinti galimybes. Kiekvienai programai, kuriai norima atlikti sąveikumo galimybių vertinimą reikia atsakyti klausimyną ir jo rezultatas gaunamas, kaip sąveikumo įvertis konkrečiai sistemai. Šio metodo trūkumas, kad, užpidžius klausimyną neatsižvelgiama į programos vidaus architektūrors panašumus su kitos programos panašumais. Taip pat neperteikia programos sruktūros, pavyzdžiui neatsako į klausimus: Kokios gali būti bendros funkcijos? Kokie bendri objektai tarp sąveikaujančių sistemų? Taip pat šio metodo naudojimas remiasi subjektyvia samprata apie sistemas ir priklauso nuo analitiko žinių apie kiekvieną iš taikomųjų programų. Jei mažas žinių kiekis apie sistemas - šis matavimo būdas netinka objektyviai įvertinti taikomųjų programų sąveikumo galimybes.

Eksperimento metu atlikti žingsniai siekiant apskaičiuoti taikomųjų programų sąveikumo galimybę:

1. Aptikti tinklo serviso dokumentaciją
2. Ištraukti ir išanalizuoti meta-duomenis iš dokumentacijos failo
3. Kategorizuoti išanalizuotus meta duomenis 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 įvairus tačiau būdingi tiktai konkrečiam objektui, pavyzdžiui objektui klientas budingi 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 meta-duomenis:
    a. Gauti šaltinio pavadinimą
    b. Gauti serviso pavadinimą
    c. Išgauti aktyvavimo metodus (GET, POST, PUT, DELETE, PATCH, HEAD...)
    d. Išgauti operacijų pavadinimo įvesties ir išvesties duomenis
        i. Panaikinti operacijose pasikartojančius, nereikšmingus pavadinimus tokius kaip: list; all; get; retrieve.
5. Išsaugoti operacijų meta-duomenis duomenų bazėje
6. Apdoroti operacijas naudojant redagavimo nuotolio algoritmus
7. Išsaugoti rezultatus duomenų bazėje
8. Duomenų vizualizacija

Šioms sistemoms (NMBRS, PrestaShop, ExactOnline, SuiteCRM) taikytas dalinai automatizuotas duomenų šaltinių analizės metodas įgyvendintas C# kalboje. Pateikiama koncepcinė diagrama vaizduojanti analizuojamas taikomųjų programų ir jų protokolus siekiant įvertinti jų architektūra ir galimybę sąveikauti tarpusavyje.



Paveikslas 21 sąveikumo vertinimo sistemos su autonominiu komponentu koncepcinė diagrama.

Koncepcinėje diagramoje paveiksle 19 pateikiamos kelios eksperimente analizuotos sistemos. Sukurtas autonominis komponentas šiam eksperimentui, tačiau jo įgyvendinimas dar neišbaigtas. Autonomins komponentas gali stebėti schemų pokyčius sistemoje ir juos analizuoti, bei išsaugoti rezultatus duomenų bazėje. Pagal SOAP ir REST protokolų apibrėžimus SOA taikomųjų programų kūrimas turėtų remtis tam tikromis dizaino rekomendacijomis. Tačiau, ypač REST protokolą naudojančioms sistemos, dažniausiai nepaiso rekomendacijų ir nepateikia vieningų taikomųjų programų tinklo servisus aprašančių failų. Jie dažnai skiriasi ir sukurti tokį autonominį komponentą tinkantį daugeliui taikomųjų programų yra labai sudėtinga. Iš paveikslo 12 žvaigždute „*“ pažymėtiems protokolams reikėjo atlikti papildomus

veiksmus siekiant išgauti operacijas ir objektus iš šių taikomųjų programų. Kai kuriais atvejais (SuiteCRM) neužtenka remtis vien tik tinklo serviso apraše pateikta informacija, kad būtų galima sukurti objektų aprašus ir atlikti meta-duomenų analizę. Tokiu atveju šiam sprendimui buvo sukurta papildoma galimybė kreiptis į tinklo servisą, kad šis gražintų objektų sąrašą.

Autonominis agentas gali atlikti duomenų stebėjimą ir išsaugoti pokyčius stebėjimo etape M (Paveikslas 19). Išsaugoti duomenys buvo apdorojami analizavimo etape A, jie išvalyti ir formatuoti taip kad būtų galima naudoti kartu su redagavimo nuotolio algoritmais.

Redagavimo nuotolių analizė atlikta R kalboje naudojant „stringdist“ paketą. Naudojantis meta-duomenų informacija išanalizavus tinklo servisus galima suskaičiuoti kiek unikalių operacijų (nesikartojančios per objektus, laukus ir tipus) galima atlikti su sistemomis. Unikalių operacijų skaičius tarp lyginamųjų taikomųjų programų pateikiamas lentelėje:

Lentelė 6. Unikalios operacijos kiekvienos veiklos sistemos tinklo servise

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 tačiau servisai NMBRS_Employees ir MIVA turi daug daugiau operacijų. Iš viso eksperimente išnagrinėtos 6861 unikalios operacijos rišys kitaip, panašumas su kitos programos operacijomis. Visų programų operacijos buvo išvalomos, paliekant tik reikšminius žodžius galimai nurodančius objektą. Iš viso liko 6228 tokių unikalių išvalytų operacijų. Apibendrinant, galima teigti jog vidutiniškai vienai sistemai gali turėti 426 unikalių išvalytų operacijų, tačiau matyti iš lentelės, kad sistemos kurios turi per daug operacijų lengvai gali iškraipyti šį vertinimą, todėl iš skaičiavimo pašalinus MIVA ir SchemaOrg operacijas 132 unikalias išvalytas operacijas vienai sistemai.

Operacijų panašumo įvertinimas kiekvienai pasirinktai taikomąjai programai ir jos porininkei skaičiuojamas operacijų pavadinimo redagavimo nuotolis įvertintas procentine išraiška nuo 0 iki 100. Jeigu redagavimo nuotolis kiekvienai operacijai yra pakankamai žemas, o procentinė panašumo išraiška aukšta, tai indikatorius, kad programų pora panaši. Rezultatai paveiksluose (Paveikslas 22, Paveikslas 23,

Paveikslas 24, Paveikslas 25) apibendrina operacijų pavadinimų redagavimo nuotolio rezultatus. Matricos aprašytos M1 – M6 pateikia visų galimų operacijų panašumus (Paveikslas 22). Žaliai išreikštos operacijos yra skirtingos, geltonai ir raudonai nuspalvintos operacijos yra panašios. Matricos M1 – M6 (Paveikslas 22) kartojasi, nes taip pateikiamos įmanomos kombinacijos tarp taikomųjų programų, kaip ir LISI pavyzdyje (Lentelė 5). Kiekviena matrica atspindi dviejų taikomųjų programų panašumą pavyzdžiui M1 yra ExactOnline ir NMBRS operacijų panašumo šilumos žemėlapis. Kur reikšmingi regionai atvaizduojami „šiltesnėm“ spalvomis.

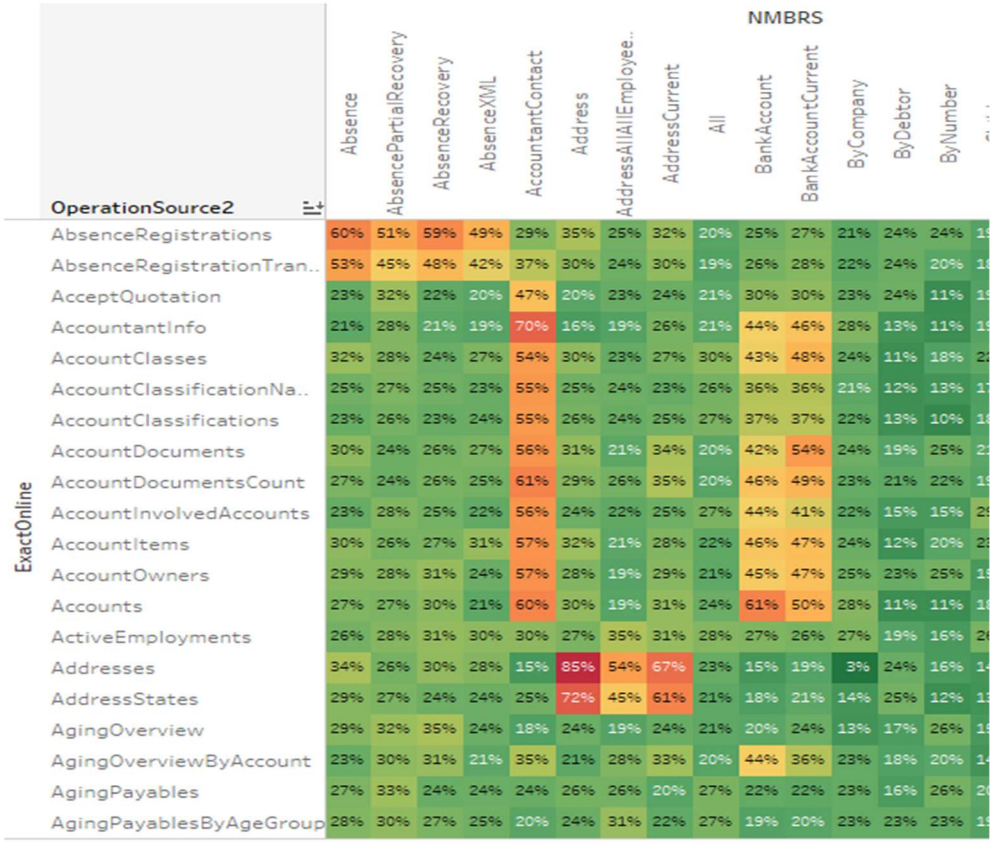
Paveikslas 22 Operacijų sąveikumo “šilumos žėmėlapis” naudojant Levenshtein redagavimo nuotolio algoritmą.

Imkime matricą M1, k ur atvaizduojamas panašumas tarp ExactOnline ir NMBRS sąveikumo vertinimas. Raudona spalva žymimos operacijos kurių panašumas > 85% lyginant su kitomis operacijomis (žalia). Raudonos dėmės taip pat indikuoja didesnę tikimybę, kad operacijos yra labiau semantiškai panašios. Pavyzdžiui ExactOnline operacija „AbsenceRegistrations“ sutampa su NMBRS operacija „Absence“ su 60 % panašumo procentu. Skaičiavimai, yra atvaizduojami tik Levenshtein metodu (Paveikslas 23), tačiau kiti algoritmai taip pat buvo panaudoti (Jaccard, Jaro-Winkler ir ilgiausios bendros sekos). Padidinus vaizdą M1 matricoje matyti pagrindiniai įverčiai ir operacijos (Paveikslas 14). Iš galimų 6 taikomųjų programų panašumo vertinimo kombinacijų (M1-M6) naudojant apibendrintą-ansamblio vertinimo metodą visi redagavimo nuotolio skaičiavimai buvo apibendrinti. Iš redagavimo nuotolio algoritmų veikimo galima pateikti tokias išvadas:

1. Jaro – Winlker ir ilgiausios bendros sekos algoritmai buvo linkę labiau vertinti operacijas vidutiniškai su 50%, šie labiau pateikia vidutiniškus rezultatus.
2. Levenshtein algoritmas labiau atskiria kraštutinumus, tačiau taip pat neišskyrė daug aukštu įvertinimu pateikiamų rezultatų.
3. Jaccard algoritmas išskyrė labai unikalias operacijas nuo labai artimai panašių operacijų. Tačiau algoritmas nebuvo linkęs suteikti panašiom operacijoms geresnių rezultatų, nei kiti algoritmai.

Paveiksle 14 galima išskirti įdomius atvejus kaip redagavimo nuotolio algoritmas rado panašumų tarp operacijų: Absence x

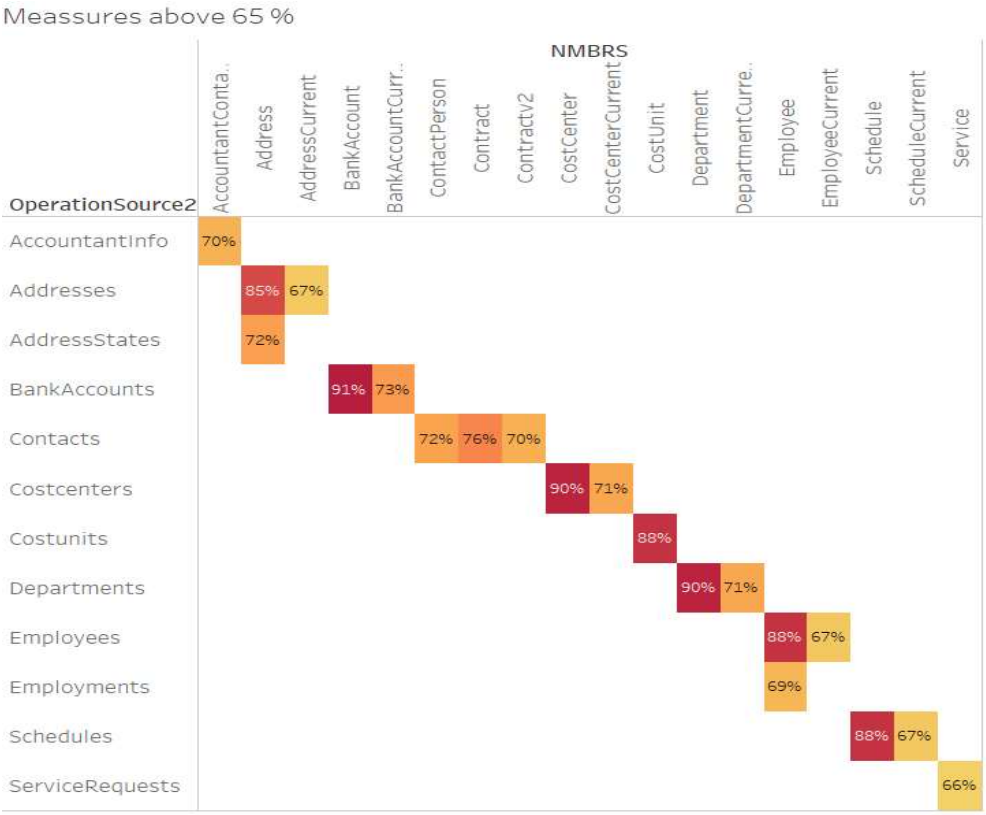
AbsenceRegistrations (60%), AccountantInfo x AccountantContact (70%), Addresses x Address (85%), Accounts x BankAccount (61%).



Paveikslas 23. Sąveikumo vertinimo matrica (dalis M1 matricos) tarp ExactOnline ir NMBRS

Norint įvertinti tik geriausius atvejus įdomu išanalizuoti atvejus kurie yra didesni nei 65%. Vertinant visų redagavimo nuotolio algoritmų apibendrintus rezultatus, galima pastebėti, kad tarp ExactOnline ir NMBRS taikomųjų programų operacijos „Accounts“, „absences“ ir „addresses“ labiausiai panašios. Tarp ExactOnline (E) ir NMBRS (N) egzistuojačios sintaksiškai panašios operacijos: E Addresses – N Address (85%); E BankAccounts – N BankAccount

(91%); E Costcenters – N CostCenter (90%); E Costunits – N CostUnit (88%); E Departments – N Department (90%); E Employees – N Employee (88%); E Schedules – N Schedule (88 %).



Paveikslas 24. Matrica M1 apribota operacijoms kurių panašumas didesnis nei 65%

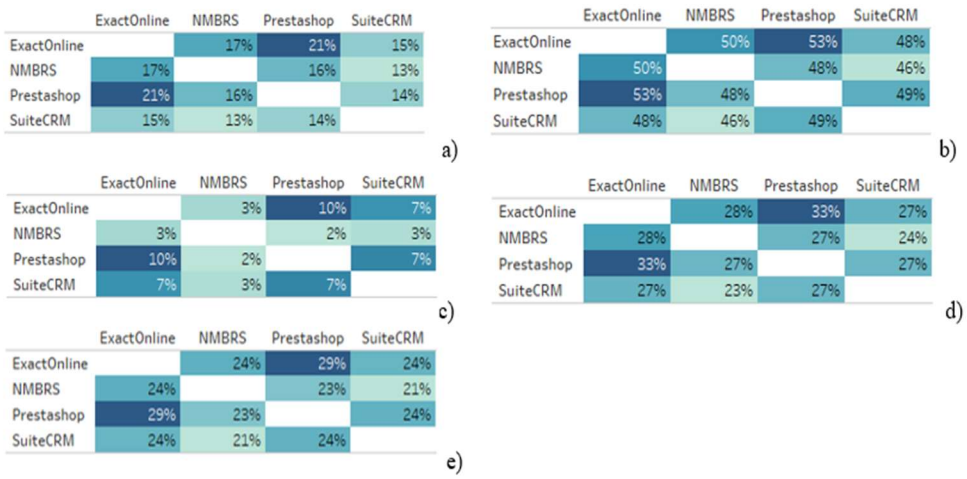
Iš rezultatų taip pat matyti (Paveikslas 24), kad tarp taikomųjų programų kai kurios operacijos taip pat gali būti sumaišytos pavyzdžiui: Contacts x Contracts (76%) ir Contacts x ContractsV2 (70%) nors semantiškai yra labai skirtingi. Lyginant rezultatus ExactOnline ir Prestashop rasta 20 operacijų su rezultatu didesniu nei 65%. Galima analizuoti ir nustatyti ribas, pagal semantines reikšmes siekiant atmesti

neteisingus palyginimus. Pastaba, kad ExactOnline 293 operacijos ir PrestaShop 49 operacijos turi tik 20 galimų sąveikumo taškų su 65% apjungimo galimybės įverčiu. Taip pat pastebėtina, jog ExactOnline ir Prestashop taikomųjų programų rezultatų analizė pateikė operacijų kurios 100% sintaksiškai identiškos (Lentelė 5). Laukai: Addresses; Contacts; Currencies; Employees; Werehouses – identiški su 100% įverčiu, tačiau rasta ir klaidingų rezultatų Projects- Products (74%).

Lentelė 7 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 ties 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 (Paveikslas 25).



Paveikslas 25. Panašumo vertinimas naudojant redagavimo nuotolio algoritmus: a - Levenshtein, b - Jaro-Winkler, c - Jaccard, d - Ilgiausios bendros sekos, e - visų metodų ansamblis.

Vertinimo amplitudes (procentiniai įverčiai) tarp redagavimo nuotolio sprendimų skiriasi dėl to skirtingi metodai skirtingai vertina panašias ir nepanašias operacijas, tačiau įdomu kad nors ir procentiniai vertinimo įverčiai skiriasi tačiau labiausiai panašios sistemos išlieka panašios (ExactOnline – Prestashop).

5.1. Tekstinė analizė

Tekstinė šaltinių analizė atliekama siekiant iškryninti ir išvalyti tekstą nuo nepageidaujamų priešdelių. Duomenų atvaizdavimas 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 integracijų specialistas gali aproksimuotai įvertinti vartojamų raktažodžių populiarimą ir palyginti su kitomis sistemomis. Pavyzdyje patektos trijų programų „KonaKart“, „Zen Cart“ ir „Suite CRM“ žodžių maišų diagramos atitinkamai A,B ir C (paveikslas 24).

Paveikslas 26. Tekstinės analizės palyginimas tarp veiklos programų.

Vykdant sąveikumo vertinimui skirtą tekstinę analizę pavyko apžvelgti 14 sistemų sandarą. Taip pat lyginant kiekvienos programos sandarą su duomenimis iš Schema.org galime teigti jog įmanoma atlikti sistemų semantinį palyginimą.

5.2. Latentinė semantinė analizė

Keliama prielaida, kad žodžiai naudojami sistemose yra reikšmiškai panašūs jei pasikartoja panašiose teksto vietose – dar vadinama distribucine semantika . Remiantis tokia prielaida galime naudoti latentinį semantinį indeksavimą (angl. latent semantic indexing) tokių reikšmiškų panašumo aptikimo tarm skirtingų programų.

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

- “RODBC” – duomenų bazių prisijungimui
- “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“ [68] . Latentinės semantinės analizės paskirtis pagal aprašymą yra rasti semantinius panašumus tarp dokumentų.

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.

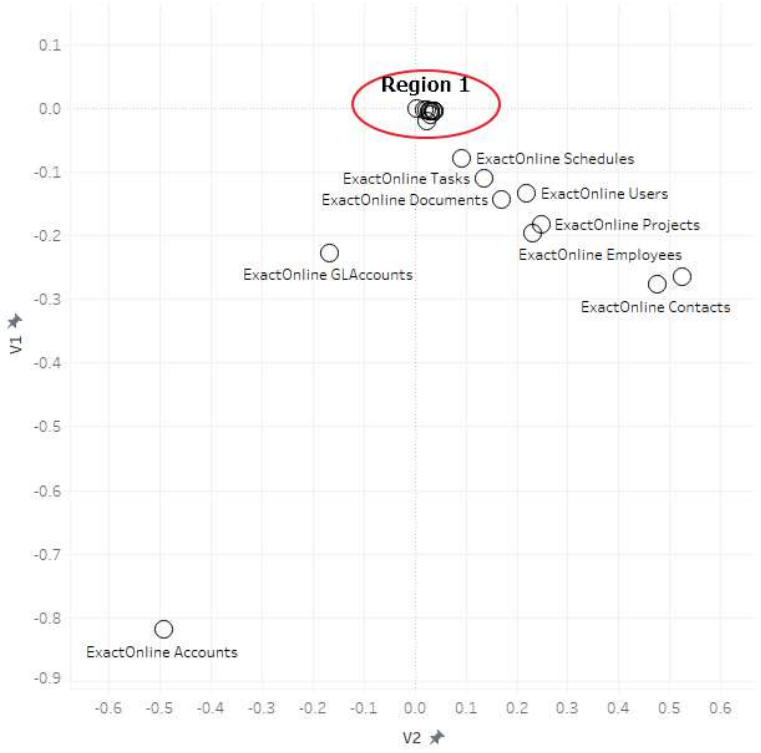
Testas: Operacijos visų sistemų kurios redagavimo nuotolio rezultate buvo 100% panašios.

Testo pavadinimas: Results_LSA_OperationsOnbjects100List

Testo aprašymas: Teste lyginame ExactOnline ir SuiteCRM programas. Šio testo duomenis imame tik tuos 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 kadangi dokumentai tai duomenų struktūrų rinkiniai, dėl šios priežasties nors savybių daug tačiau jos retos.

Semantinio panašumo tyrimo etape matyti, kad ExactOnline programos objektai išsibarstę labiau nei SuiteCRM paketo objektai (Paveikslas 27).



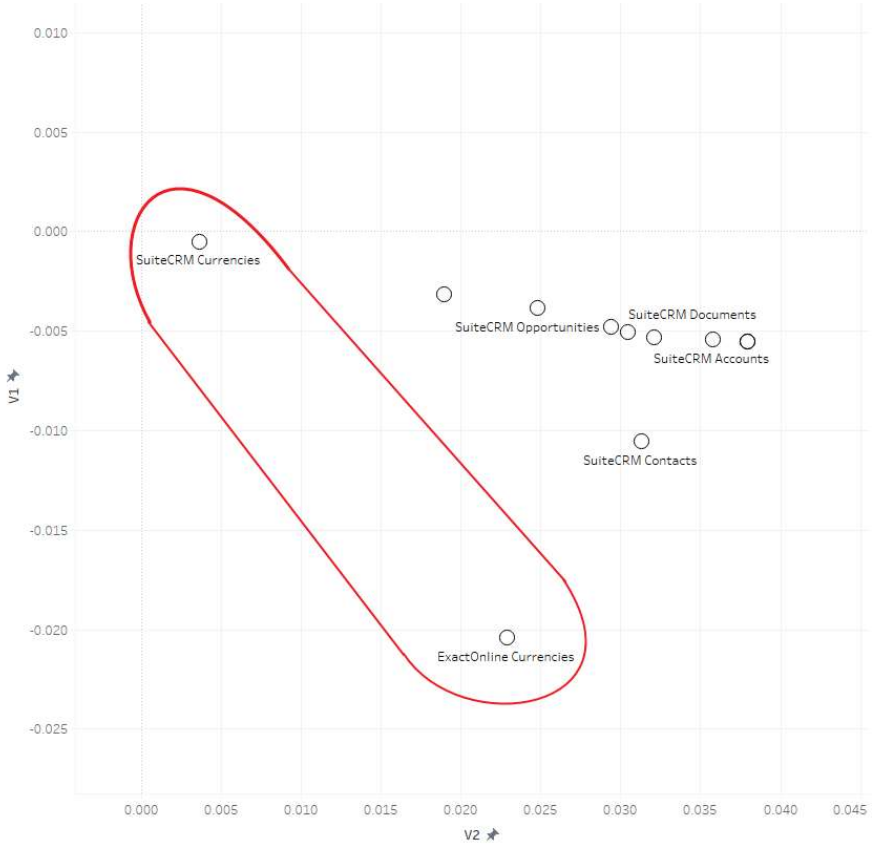
Paveikslas 27. ExactOnline kartu su SuiteCRM struktūrinis panašumas naudojant LSA metodą.

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 nei ExactOnline objektai kurie turi mažiau bendrų bruožų per objektą.

Lentelė 8, 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 (Region 1 Paveikslas 27) matyti kad į šį regioną pateko ExactOnline objektas “Currencies”. Tai rodo, kad semantiškai ExactOnline Currencies – turi semantiškai panašių bruožų su šia grupe.



Paveikslas 28, Region 1 (Paveikslas 27), skirtingų sistemų panašumas. naudojant LSA metodą. SuiteCRM ir ExactOnline moduliai “currencies” turi komponentus kurie turi labai panašius pavadinimus todėl plokštumoje atvaizduojami glaudžiau.

Apibendrinant, tyrimas leidžia nustatyti semantinį panašumą tarp objektų ir jų sudėtinių dalių. Šio tyrimo privalumas – gaunamas atstumo vektorius leidžiantis patikrinti objektų artumą viena kitai, tačiau reikia

geriau pasiruošti duomenis. Šiame etape reiktų gero rezultatų interpretacijos mechanizmo kurį būtų galima automatizuoti.

6. Išvados ir rekomendacijos

Naujas sprendimas siūlo naudotis autonominio valdymo elementus ir architektūrą taikomųjų programų sąveikumui. Skirtingai nei kiti ištirti metodai, šis sprendimas leidžia kurti savęs valdymo galimybes turinčius sprendimus. Mes galime pamatuoti daugybės taikomųjų programų schemų panašumus, o turint matavimus galime atlikti tikslesnes sąveikumo operacijas.

1. Disertacijos metu pavyko ištraukti kontekstinę tinklo serviso aprašymo informaciją ir ją išanalizuoti išskiriant: operacijas, objektus, laukus, laukų tipus, laukų kardinalumą. Todėl galima teigti jog įmanoma bent dalinai automatizuoti tinklo servisų aprašų analizę. Taip pat šis pasiekimas paremtų autonominio skaičiavimo principo stebėjimo ir analizės (angl: monitor (M), analyze (A) – paveiksluose 9 ir 19) žingsnių išpildymą.
2. Palyginti panašumai tarp elementariųjų valdymo ciklų ir autonominio valdymo komponento. Atrasti panašumai tarp šių dviejų skirtingų sričių. Tikėtina, kad dėl jų panašumų autonominio skaičiavimo metodai gali būti naudojami analizuojant ir integruojant veiklos procesus ir juos valdančius objektus (taikomoji programas).
3. Eksperimentinis tyrimas atskleidžia analitinį požiūrį į taikomųjų programų sąveikumą. Ištirti pavyzdžiai atvaizduoja skirtumus veiklos procesuose ir parodoma kaip šie skirtumai gali būti išmatuojami šioje disertacijoje pasiūlytų metodu.
4. Pabrėžtini tolimesni tikslai: sukurti autonominio skaičiavimo pagrindu veikiantį sąveikumo sprendimą.

5. Taikomųjų programų modelių susiejimas su veiklos procesų sluoksnio modeliais leidžia išgauti tikslią informaciją reikalingą automatizuojant taikomųjų programųs ąveikumo sprendimo kūrimą.

Pasiūlytas sprendimas remiasi redagavimo nuotolio skaičiavimo metodu siekiant įvertinti progamų sąveikumo galimybę ir galimai ją automatizuoti. Tai taip pat leistų pagerinti programų integracijos ir sąveikumo sprendimo kūrimo procesus. Šiai sričiai reikalingi kompetentingi ir gerai skirtingas taikomoji programas pažįstantys ekspertai, tačiau šioje disertacijoje paliečiama galimybė šias ekspertines žinias leisti generuoti ir kaupti pačiam autonominio skaičiavimo metodui. Nors šioje disertacijoje eksperimente nebuvo pilnai įgyvendinti bent vieno autonominio skaičiavimo principu valdomu ciklų tačiau pradžia šioje srityje padaryta. Išmokus automatizuoti integracijos metodus, tai būtų labai didelė pažanga informacijos apdorojimo ir sklaidos grupėje.

Bibliografija

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

[2] Chen, D., 2006, June. Enterprise Interoperability Framework. In EMOI-INTEROP.

[3] Dijkman, R., Dumas, M., Van Dongen, B., Käärik, R. and Mendling, J., 2011. Similarity of business process models: Metrics and evaluation. Information Systems, 36(2), pp.498-516.

[4] Dzemydienė, D., Naujikienė, R., 2009. Elektroninių viešųjų paslaugų naudojimo ir informacinių taikomųjų programų sąveikumo vertinimas. Informacijos mokslai, 2009, 50.

[5] El-Halwagi, M.M. Process Integration. Elsevier. ISBN-9780123705327 (2006)

[6] Fielding, R.T. and Taylor, R.N., 2000. Architectural styles and the design of network-based software architectures (Vol. 7). Doctoral dissertation: University of California, Irvine.

[7] Ford, T., Colombi, J., Graham, S. and Jacques, D., 2008. Measuring system interoperability. Proceeding Cser.

[8] Gates, B., 2013. Measuring progress. Annual Letter, Gates Foundation. <https://www.gatesfoundation.org/Who-We-Are/Resources-and-Media/Annual-Letters-List/Annual-Letter-2013>. Peržiūrėta 2018-09-25.

[9] Gricius, G. Daugiaagentinių taikomųjų programų kūrimo metodų išvystymas nedidelio našumo įterptinių taikomųjų programų integravimui. Disertacija, 2015.

[10] I. Zinnikus, C. Hahn, and K. Fischer. A Model-driven, Agent-based Approach for the Integration of Services into a Collaborative Business Process. In: AAMAS 2008, pp. 241-248 (2008)

[11] Jacob, B., Lanyon-Hogg, R., Nadgir, D.K. and Yassin, A.F., 2004. A practical guide to the IBM autonomic computing toolkit. IBM Redbooks, 4.

[12] Kasunic, Mark. Measuring systems interoperability: challenges and opportunities. Carnegie-Mellon Univ Pittsburgh Pa Software Engineering Inst, 2001.

[13] Kephart, J.O. and Chess, D.M., 2003. The vision of autonomic computing. Computer, (1), pp.41-50.

[14] Krajicek, J. and Krajíček, J., 1995. Bounded arithmetic, propositional logic and complexity theory. Cambridge University Press.

[15] Li, B. Wu, Y. Yang. Agent-based Ontology Integration for Ontology-based Applications. In: CRPIT (2005)

[16] Morkevičius A, Business and information systems alignment method based on enterprise architecture models, Doctoral Dissertation 2014, Kaunas

[17] Navarro, Gonzalo. A guided tour to approximate string matching. ACM computing surveys (CSUR), 2001, 33.1: 31-88.

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

[19] Papazoglou, M., 2008. Web services: principles and technology. Pearson Education.

[20] Pavlin G., Kamermans, M. Scafeş. Dynamic Process Integration Framework: Toward Efficient In-formation Processing in Complex Distributed Systems. In: Proceedings of the 3rd International Sym-posium on Intelligent Distributed Computing (2009)

[21] Rahm, E. and Bernstein, P.A., 2001. A survey of approaches to automatic schema matching. the VLDB Journal, 10(4), pp.334-350.

[22] R-D. Kutsche, N. Milanovic. Model-Based Software and Data Integration. In: MBSDI (2008).

[23] Reijers, H.A. and 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.

[24] Savulionienė L. 2014, Susietumo taisyklių paieška didelėse duomenų bazėse. Disertacija

[25] Scott Brinker, 2017, The average enterprise uses 91 marketing cloud services, <https://chiefmartec.com/2017/06/average-enterprise-uses-91-marketing-cloud-services/> Paskutinį kartą peržiūrėta: 2018-09-27.

[26] Trotta, G., 2003. Dancing around EAI'bear traps'. Business Process Management (BPM) Best Practices. < http://www.ebizq.net/topics/int_sbp/features/3463.html?page=1 > Paskutinį kartą peržiūrėta: 2018-09-27.

[27] Valatavicius A., Dilijonas D. Dynamic B2B process integration (in Lithuanian). In: Proceedings of "Informacinės Technologijos", pp. 34-39. Kaunas (2014)

[28] Villányi, B. and Martinek, P., 2015. Improved Accuracy Evaluation of Schema Matching Algorithms. Acta Polytechnica Hungarica, 12(6).

[29] Walsh, A.E., 2002. Uddi, Soap, and WSDL: the web services specification reference book. Prentice Hall Professional Technical Reference.

[30] Y. Peng, T. Finin, Y. Labrou, B. Chu, J Long, W.J. Tolone, A. Boughannam. A multi-agent sys-tem for enterprise integration. In: Proceedings of PAAM (1998).

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

[32] Gudas, S., 2012. Knowledge-based enterprise framework: a management control view. In *New Research on Knowledge Management Models and Methods*. InTech.

[33] Euzenat, J., Meilicke, C., Stuckenschmidt, H., Shvaiko, P. and Trojahn, C., 2011. Ontology alignment evaluation initiative: six years of experience. In *Journal on data semantics XV* (pp. 158-192). Springer, Berlin, Heidelberg.

[34] Tan, S., Liu, K. and Xie, Z., 2004. A Semiotic approach to organisational modelling using norm analysis. In *6th Int. Conf. Enterp. Inf* (pp. 1-15).

[35] Wikipedia, Comparison of shopping cart software < https://en.wikipedia.org/wiki/Comparison_of_shopping_cart_software>, Paskutinį kartą paržiūrėta 2018-07-18

[36] McCann, R., AlShebli, B., Le, Q., Nguyen, H., Vu, L. and Doan, A., 2005, August. Mapping maintenance for data integration

systems. In *Proceedings of the 31st international conference on Very large data bases* (pp. 1018-1029). VLDB Endowment.

[37] Halevy, A.Y., Ashish, N., Bitton, D., Carey, M., Draper, D., Pollock, J., Rosenthal, A. and Sikka, V., 2005, June. Enterprise information integration: successes, challenges and controversies. In *Proceedings of the 2005 ACM SIGMOD international conference on Management of data* (pp. 778-787). ACM.

[38] Peukert, E., Eberius, J. and Rahm, E., 2011. AMC-A framework for modelling and comparing matching systems as matching processes.

[39] Dong, X.L. and Srivastava, D., 2013, April. Big data integration. In *Data Engineering (ICDE), 2013 IEEE 29th International Conference on* (pp. 1245-1248). IEEE.

[40] Valatavičius, A., Gudas, S., Apie taikomųjų programų sąveikumo metodologiją, grindžiamą giluminėmis žiniomis. Informacijos mokslai, 79(79), pp.83-113.

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

[42] IBM Corporation, "Autonomic Computing Concepts," http://www.ibm.com/autonomic/pdfs/AC_Concepts.pdf, 2001

[43] Gartner Glossary, Nuoroda <https://www.gartner.com/it-glossary>, peržiūrėta 2019-04-14.

[44] Linthicum, D.S., 2000. Enterprise application integration. Addison-Wesley Professional.

[45] KRAFZIG, Dirk; BANKE Karl; SLAMA, Dirk. Enterprise SOA: service-oriented architec-ture best practices. Prentice Hall Professional, 2005.

[46] SILVERSTON, Len; INMON, William H.; GRAZIANO Kent. The data model resource book: a library of logical data models and data warehouse designs. John Wiley & Sons, Inc, 1997. ISBN:0471153672.

[47] IDABC, Echipa; INDUSTRY, D. G. European interoperability framework for pan-European e-government services. European Communities, 2004. http://ec.europa.eu/idabc/servlets/Docd552.pdf. Peržiūrėta Birželio 3 d. 2017. ISBN 92-894-8389-X.

[48] Interoperability solutions for public administrations, businesses and citizens (ISA²), New European Interoperability Framework. https://ec.europa.eu/isa2/sites/isa/files/eif_brochure_final.pdf. Accessed 9 March 2019.

[49] CEITON technologies "Front-end and back-end EAI" CEITON technologies Peržiūrėta Liepos 28 d. 2015

[50] Guédria, W., Naudet, Y. and Chen, D., 2008, November. Interoperability maturity models–survey and comparison–. In *OTM Confederated International Conferences" On the Move to Meaningful Internet Systems"* (pp. 273-282). Springer, Berlin, Heidelberg.

[51] Basson, H., Bouneffa, M., Matsuda, M., Ahmad, A., Chung, D. and Arai, E., 2016. Qualitative Evaluation of Manufacturing Software Units Interoperability Using ISO 25000 Quality Model. In *Enterprise Interoperability VII* (pp. 199-209). Springer, Cham.

[52] Sitton, M. and Reich, Y., 2016, July. Enterprise Systems Engineering for Improving Cross-enterprise Effectiveness. In *INCOSE International Symposium* (Vol. 26, No. 1, pp. 2085-2100).

[53] Sahlgren, Magnus (2008). "The Distributional Hypothesis" (PDF). Rivista di Linguistica. 20 (1): 33–53.

[54] Grossman, D.A. and Frieder, O., 2012. Information retrieval: Algorithms and heuristics (Vol. 15). Springer Science & Business Media.

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

[56] Tolk, A. and Muguira, J.A., 2003, September. The levels of conceptual interoperability model. In Proceedings of the 2003 fall simulation interoperability workshop (Vol. 7, pp. 1-11). Citeseer.

[57] CINTUGLU, Mehmet H.; YOUSSEF, Tarek; MOHAMMED, Osama A. Development and application of a real-time testbed for multiagent system interoperability: A case study on hierarchical microgrid control. IEEE Transactions on Smart Grid, 2018, 9.3: 1759-1768.

[58] SHVAIKO, Pavel; EUZENAT Jérôme. Ontology matching: state of the art and future challenges. In: IEEE Transactions on knowledge and data engineering, 2013. 25, (1) p.158-176.

[59] PEUKERT, Eric; EBERIUS Julian; RAHM Erhard. A self-configuring schema matching system. In: 2012 IEEE 28th International Conference on Data Engineering. IEEE, 2012. p. 306-317.

[60] PAVLIN, Gregor; KAMERMANS, Michiel; SCAFES, Mihnea. Dynamic process integration framework: Toward efficient information processing in complex distributed systems. In: Informatica, 2010. 34, (4).

[61] OVEREINDER, Benno J.; VERKAIK, P. D.; BRAZIER, Frances MT. Web service access management for integration with agent systems. In: Proceedings of the 2008 ACM symposium on Applied computing. ACM, 2008. p. 1854-1860.

[62] ЛЕВЕНШТЕЙН, Владимир Иосифович. Двоичные коды с исправлением выпадений, вставок и замещений символов. In: Доклады Академии наук. Российская академия наук, 1965. p. 845-848.

[63] KRAFZIG, Dirk; BANKE Karl; SLAMA, Dirk. Enterprise SOA: service-oriented architecture best practices. Prentice Hall Professional, 2005.

[64] International Organisation for Standartisation, ISO/IEC 2382:2015 Information technology - Vocabulary, 2015. https://www.iso.org/obp/ui/#iso:std:iso-iec:2382:ed-1:v1:en. Accessed 9 March 2019

[65] HOHPE, Gregor, WOOLF Bobby . Enterprise integration patterns. In: 9th Conference on Pattern Language of Programs, 2002. p. 1-9.

[66] HEYLIGHEN, Francis; JOSLYN, Cliff . Cybernetics, and Second-Order Cybernetics. In: Encyclopedia of physical science & technology, 2001. 4, p. 155-170.

[67] DZEMYDIENĖ, Dalė; NAUJIKIENĖ, Ramutė. Elektroninių viešųjų paslaugų naudojimo ir informacinių sistemų sąveikumo vertinimas. Informacijos mokslai, 2009, 50.

[68] Grossman, D.A. and Frieder, O., 2012. Information retrieval: Algorithms and heuristics (Vol. 15). Springer Science & Business Media.

1. PRIEDAS – Tiriamų objektų sąrašas

Verslo veiklos programų tipai:

1. Buhalterijos
2. Elektroninių mokėjimų ir sąskaitų išrašymo
3. Verslo analitikos
4.

Taikomoji programa	**Tipas EN**	**PROTOCOL**	**License**
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

5. PRIEDAS – Programų ir taikomųjų programų sudėtingumo palyginimas