Vilniaus universitetas
Matematikos ir informatikos institutas
L I E T U V A

INFORMATIKA (09 P)

PROGRAMŲ SISTEMŲ DINAMINĖS INTEGRACIJOS TAIKANT AUTONOMINIO SKAIČIAVIMO TECHNOLOGIJAS TYRIMASS

Andrius Valatavičius

2017 m. spalis

Mokslinė ataskaita MII-DS-09P-17-12

VU Matematikos ir informatikos institutas, Akademijos g. 4, Vilnius LT-08663

www.mii.lt

Santrauka

Mokslinėje ataskaitoje apžvelgiami reikšmingiausi programų sistemų integracijos, sąveikumo sprendimų automatizavimo tyrimų rezultatai. Įvardintos išanalizuotos pagrindinės programų sistemų integracijos problemos. Pateikiami keli programų sistemų integracijos plėtojimo pavyzdžiai, pagal kibernetikos lygius. Taip pat pateikiami duomenų struktūros bei agentais grindžiamos integracijos pavyzdžiai. Palygintos pagrindinės metodologijos integracijai kurti (programavimas, EAI sistemų naudojimas, agentinės technologijos bei kiti tyrinėjami metodai). Pateiktas siūlomas modelis duomenų integracijos automatizavimui naudojant verslo proceso žinias, kurį toliau vystant galima būtų sukurti žiniomis grindžiamą programų sistemų integracijos automatizavimo metodologiją, kuri teoriškai leistų sukurti algoritmus padėsiančius programų sistemų integracijos projektuotojui gauti reikiamas tiek verslo, tiek programų sistemų žinias. Patobulintas modelis apjungia tyrimo sritis iš verslo procesų valdymo modeliavimo ir programų sistemų architektūros modeliavimo sprendimus Apžvelgiamos sąsajos su kontrolės teorija, meta-duomenų modeliavimu, duomenų struktūrų apjungimo metodais, ir integravimo bei funkcinio suderinamumo sprendimų kūrimo įrankiais.

Reikšminiai žodžiai: programų sistemų integracija, autonominė kompiuterija, automatizavimas, duomenų integracija, funkcinis suderinamumas, programų sistemų sąveikumas

Turinys

1 Terminai

Sąveikumas – Skirtingų gamintojų sistemų, kompiuterių arba programų gebėjimas keistis ir naudotis nevienarūšiais duomenimis (angl. interoperability), (sinonimai: funkcinis suderinamumas, suderinamumas).
Sisteminė programa – (angl. System Software)
Taikomoji programinė įranga – (angl. Application software)

2 Santrumpos

AIA – Autonomic Interoperability Application
BPMN – Business process modeling notaion
CIM – Computation Independent Model
CRM – Customer Relationship management
DMN – Decision modeling notation
EAI – Veiklos programų sistemų integracija (angl. Enterprise Application Integration)
EEML – Extended Enterprise Modeling Language
EIF – European Interoperabiility Framework
ERP – Enterprise Resource Management
EMC – Elementarus valdymo ciklas (Elementary Management Cycle)
IMC – Internal model controler
KM – Žinių modelis (Knowledge Model)
MDA – Model Driven Architecture
MT – Management Transaction
MEFF – Modelių apsikeitimo failų formatas (angl. Model Exchange File Format)
UML – Unified modeling language
IBM - angl. International Business Machines Corporation
IM – vidinis modelis (angl. Internal Model)
OMG – Object management group
PIM – Platform independent model
PSM – Platform specific model
RW – Real World
SOA – Service Oriented Architecture
SDLC – Software Development Life Cycle

3 Įvadas

Mokslininkai ieško būdų kaip pagerinti programų sistemų tarpusavio susietumą ir užtikrinti kokybiškus duomenų mainus ne tik tarp skirtingų programų sistemų bet ir verslo subjektų. Dėl šios priežasties nemažai dėmesio skiriama programų sistemų integracijos problemoms spręsti. Analizuojami dinaminiai metodai siekia pagerinti programų sistemų integracijos procesus ir juos automatizuoti, arba bent jau sumažinti integracijos palaikymo ir aptarnavimo laiką bei žinių poreikį integracijos sistemoms palaikyti. Šio mokslinio darbo tikslas įvertinti esamus programų sistemų integracijos ir sąveikumo sprendimu. Išanalizuoti šių sprendimų automatizavimo galimybes ir pasiūlyti naują sprendimą kuris leistu pagerinti šių sprendimų kūrimo procesus.

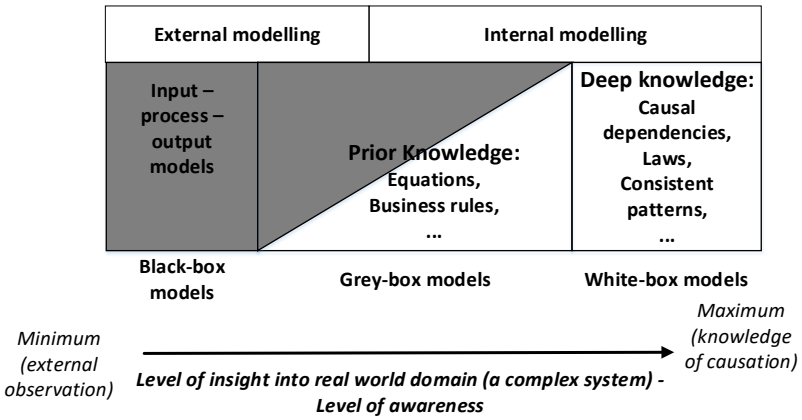
Ataskaitoje apžvelgiami reikšmingiausi programų sistemų integracijos ir jos automatizavimo doktorantūros studijų rezultatai. Įvardintos išanalizuotos pagrindinės programų sistemų integracijos problemos. Pateikiami keli programų sistemų integracijos plėtojimo pavyzdžiai, pagal kibernetikos lygius. Taip pat pateikiami duomenų struktūros bei agentais grindžiamos integracijos pavyzdžiai. Palygintos pagrindinės metodologijos integracijai kurti (programavimas, EAI sistemų naudojimas, agentinės technologijos bei kiti tyrinėjami metodai). Pateiktas siūlomas modelis duomenų integracijos automatizavimui naudojant verslo proceso žinias, kurį toliau vystant galima būtų sukurti žiniomis grindžiamą programų sistemų integracijos automatizavimo metodologiją, kuri teoriškai leistų sukurti algoritmus padėsiančius programų sistemų integracijos projektuotojui gauti reikiamas tiek verslo, tiek programų sistemų žinias. Tyrimai papildyti programų sistemų sąveikumo apžvalga ir analize. Ieškoma galimybė įvertinti sąveikumo galimybes tarp dviejų ar daugiau skirtingų programų sistemų. Apžvelgiamos sąsajos su kontrolės teorija, meta-duomenų modeliavimu, duomenų struktūrų (schemų) apjungimo metodais.

4 Programų sistemų integracijos ir sąveikumo problemos

Programų sistemų integracija yra vystoma įvairiomis formomis, tiek verslo tiek mokslininkų. Vystyti programų sistemų integracijos autonomiškumą naudinga keliomis prasmėmis:

- Gerinti, standartizuoti integracijos bei sąveikumo kūrimo procesus;
- Greičiau apdoroti programų sistemų informaciją;
- Atlikti sudėtingas manipuliacijas su dideliais duomenimis atpažįstant esminius duomenis, jų tarpusavio ryšius;
- Sumažinti nesėkmingų projektų skaičių [6];
- Sumažinti žinių poreikį reikalingą sėkmingiems programų sistemų integracijos projektams atlikti.

Automatizavimo aspektu didėle problema yra gilių žinių apie integruojamas sistemas nebuvimas. Dažnai įgyvendinti integracijos sprendimai yra įgyvendinti juodosios – dėžes 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 1. Baltosios dėžės / Pilkosios dėžės modeliai

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

Išanalizavus programų sistemų integracijos srityje atliekamus tyrimus, literatūros apžvalgos metu įvardintos šios problemos kylančios integruojant dvi arba daugiau skirtingų programų sistemų:

- **Skirtingos duomenų struktūros** – gali skirtis pagal esybių ryšių diagramą duomenų bazėje, skirtinga SOA architektūra;
- **Skirtingi duomenų šaltiniai** – skirtingos programų sistemos su skirtingomis duomenų bazėmis, skirtinga SOA architektūra, informacijos perdavimo šaltiniais, agentais, jų tarpusavio komunikavimo protokolais;
- **Dinaminės sistemos** - nuolatos keičiasi arba gali keistis programų sistemos struktūra, duomenų lentelių kiekis, esybių ryšiai duomenų bazėje;
- **Kelios sistemos** – sprendžiami klausimai apie vienodų duomenų patikimumą, pirmumą integracijos procese;
- **Skirtingi autentifikavimo procesai** – susiję su skirtingais duomenų šaltiniais, nes kiekvienas duomenų šaltinis programos sistemoje gali turėti vartotojo prisijungimo ir autentifikavimo skirtumu kuriuos reik 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 programų sistemų esybes arba jų sudėtinius komponentus su kitos programos sistemos esybėmis ir komponentais;
- **Orkestravimas ir choreografija** – Esybių ir objektų iš skirtingų sistemų integracijos eigos išdėstymas laike, laikantis numatytų apribojimų. Apribojimai išvedami pagal programų sistemos struktūrą arba laikantis verslo eigos procesų, priklausomai nuo naudojamų programų sistemų 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 programų sistemų perdavimą iš vienos sistemos kitai, laiko trukmę, perduotų duomenų teisingą apdorojimą perduotajai programai.
- **CRUD** (angl. Create Read / Update / Delete) **taisyklių užtikrinimas**. Programų sistemų integracijos sritis nagrinėjanti kaip užtikrinti saugų ir kokybišką rašymą, skaitymą, atnaujinimą ir trynimą integruojamose programų sistemose laikantis tam tikrų nustatytų taisyklių.
- **Ontologijų integracija -** programų sistemų srityje stengiamasi išanalizuoti kaip integracijos procesas galėtų suprasti savo aplinka, taip geriau reaguotų į trikdžius ir gebėtų kai kuriuos iš jų spręsti autonomiškai;
- **Dokumentacijos trūkumas** – programų sistemų 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;
- **B2B 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**- paprastai integracijos projektuotojo sprendžiamos problemos, šioms ieškoma būdų kaip surinkti integruotojo žinias panaudojant integracijos sprendimų paieškai.
- **Netvarkingi duomenys** – integracijos sistemos architekto sprendžiamos problemos, gilinimąsi kaip ištaisyti klaidas, kaip suprogramuoti automatinį jų aptikimą ir ištaisymą arba išvengimą. Sugalvoti kokius metodus taikyti duomenų taisymui ir tvarkymui.

Lentelė 1 Integracijos ir sąveikumo problemų lentelė:

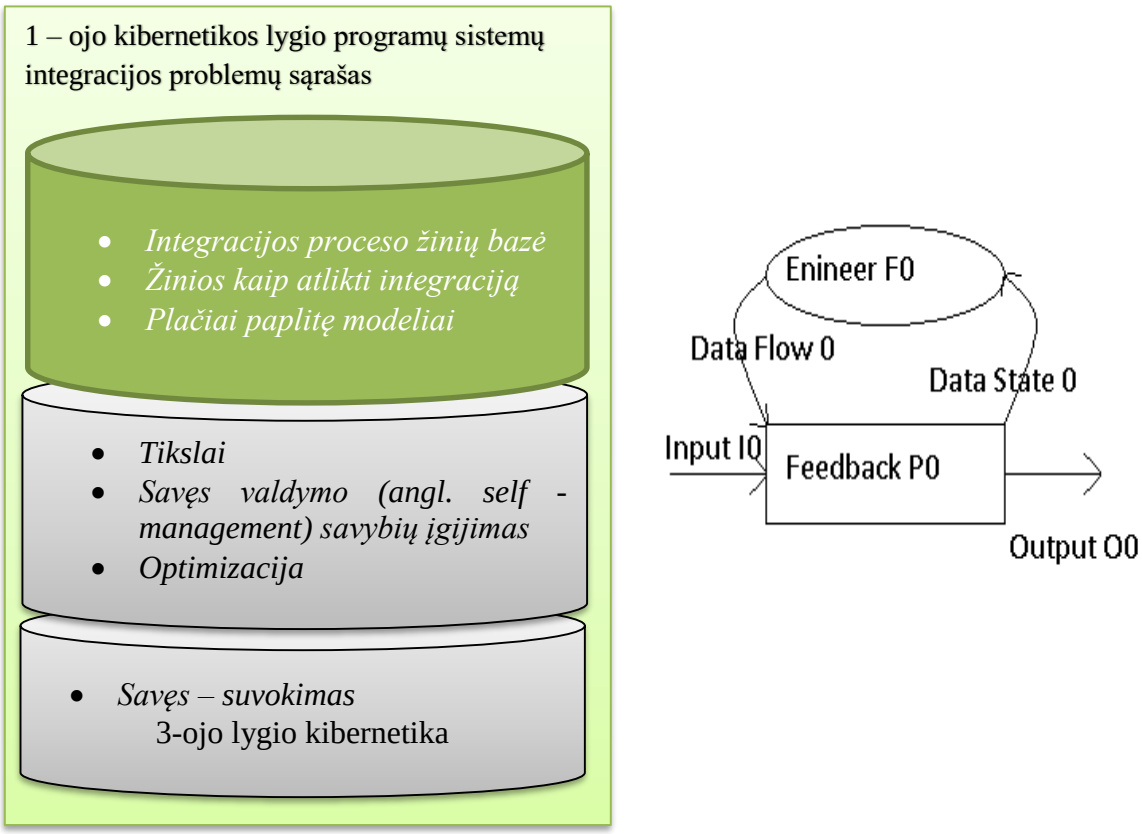
	Integracijos problemos	**Sąveikumo problemos**
1.	Gilių žinių ir vidinio modelio nebuvimas	Gilių žinių ir vidinio modelio nebuvimas
2	Skirtingos duomenų struktūros	
3	Skirtingi duomenų šaltiniai	
4	Dinaminės sistemos	Dinaminės sistemos
5	Kelios sistemos	Kelios sistemos
6	Skirtingi autentifikavimo procesai	
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	Ontologijų integracija
15	Dokumentacijos trūkumas	

16	B2B integracija	
17	Nestandartinių vienetų, kalbos skirtumų, perteklinių etikečių, formatavimo	
18	Netvarkingi duomenys	Netvarkingi duomenys

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;*
- ***Skirtingi duomenų šaltiniai*** *– ištiriami kokie duomenų šaltiniai yra programų sistemose, kokios autentifikavimo sistemos ir kokie parametrai reikalingi, tuomet rankiniu būdu suvedama (įprogramuojama) reikalinga informacija. Esant duomenų šaltinių pokyčiams reikalingas perprogramavimas.*
- ***Dinaminės sistemos*** *– rankiniu būdu programuojami pakeitimai ir pritaikymas pakitusiai sistemai siekiant palaikyti integracijos procesą, kitu atveju procesas nustoja veikti ir informacija neperduodama.*
- ***Kelios sistemos*** *– ištiriami ir nustatomi prioritetai, perdavimo politika integruojamiems duomenims;*

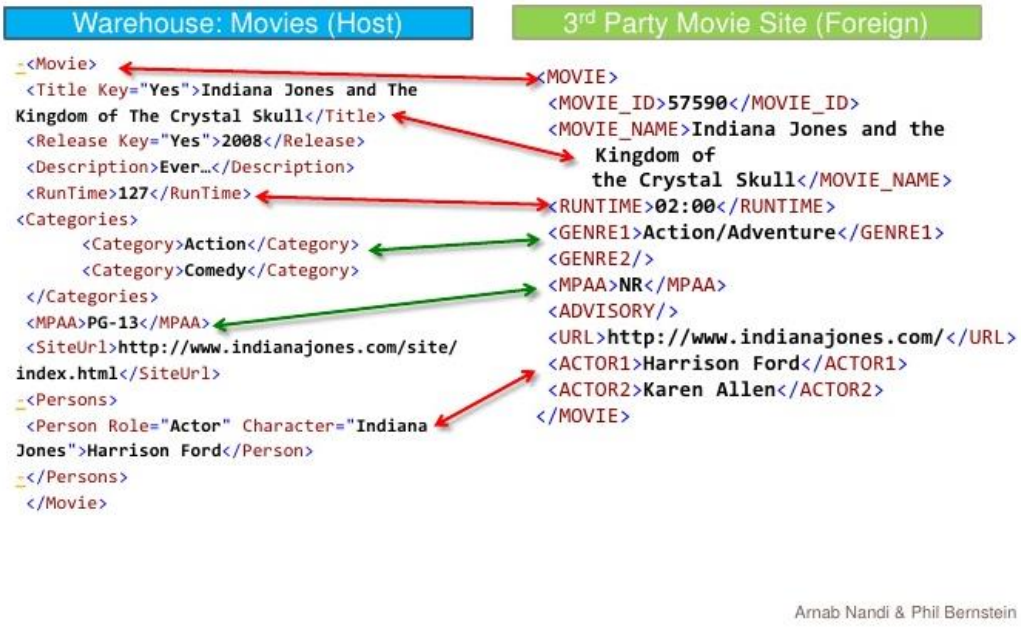
Analizuota kaip skirtingi kibernetikos lygiai įtakoja/įtakotų programų sistemų integracijos problemas:



Paveikslas 2. Autoriaus diagrama reprezentuojanti skirtingus kibernetikos lygius

Paveikslas 2 apibrėžiami skirtingi kibernetikos lygiai. 1-ajame kibernetikos lygyje nėra automatizuotų metodų įvardintoms programų sistemų integracijos problemoms spręsti. Šiame kibernetikos lygyje taip pat naudojamos integracijos procese sukauptos žinios, kaip atlikti informaciją, kokios sistemos yra integruojamos, kokios tų sistemų savybės kokios jų duomenų struktūros ir panaši informacija reikalinga integravimo projektavimo stadijoje kad integracijos projektas būtų sėkmingas. 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 programų sistemų). 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.

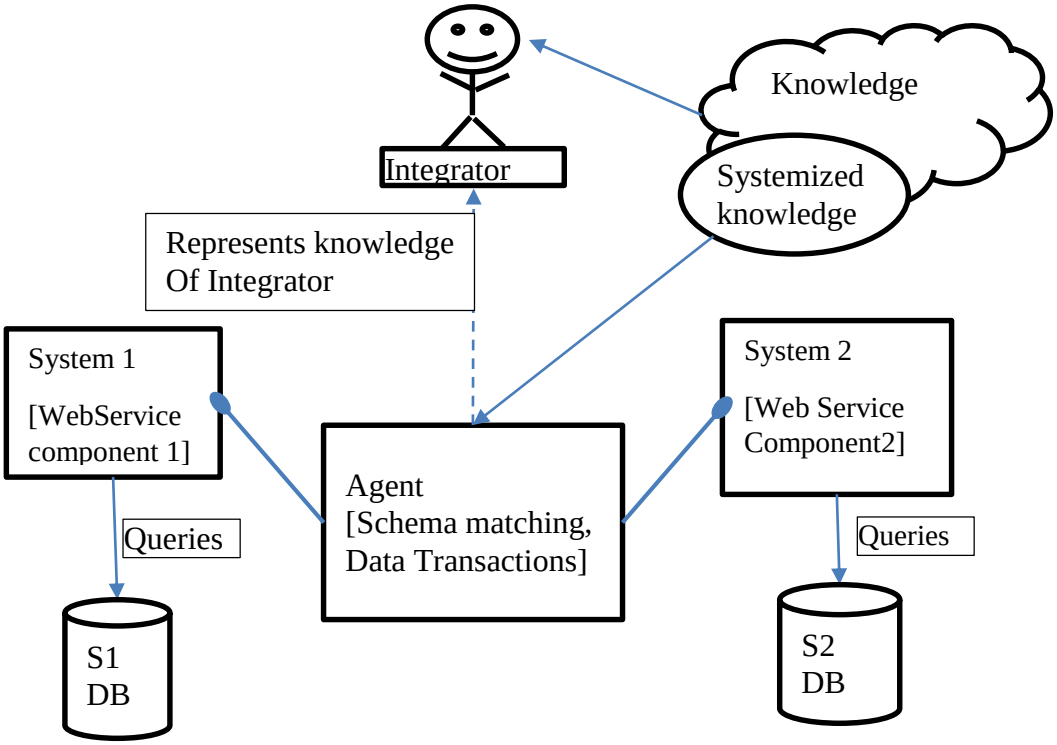
Toliau pateikiamas skirtingos duomenų struktūros apjungimo grafinis pavyzdys (paveikslas ę):



Paveikslas 3 Autoriaus skirtingų duomenų struktūrų apjungimo grafinis pavyzdys

Paveiksle pavaizduota dviejų skirtingų nutolusių programų sistemų duomenų struktūros kurias reikia integruoti ir užtikrinti jų sąveikumą abejose sistemose tiek Sandėlio (Warehouse), tiek Filmų puslapio (Party Movie Site). Esant griežtai statinėms struktūroms, struktūrų apjungimas vykdomas programiškai arba EAI (angl. Enterprise application integration) sistemų pagalba. Norint automatizuoti duomenų struktūros procesus tyrėjai dirba prie algoritmų kurie galėtų atpažinti naudojamus raktažodžius, duomenų tipus, spręstų semantinius esybių ir objektų atpažinimo uždavinius [1, 7, 8, 19].

Dalis integracijos automatizacijos algoritmų realizuoti agentų pagalba. Programų sistemų integracijos pavyzdys naudojant agentų technologijas:



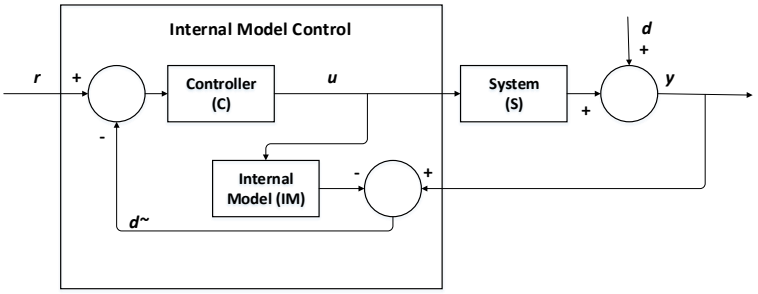
Paveikslas 4 Autoriaus programų sistemų integracijos pavyzdys naudojant agentines technologijas.

Programų sistemų 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 sistemų, duomenų vėlavimui ar kitoms problemoms. Pats agentas nesupranta nesuprojektuotų probleminių situacijų nes jis neturi konkretaus verslo procesų modelio ir nesupranta integruojamų sistemų pavyzdžiui System 1 ir System 2 (Paveikslas 3) struktūros. 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.

Sistemizuotų žinių išgavimas yra sudėtingiausia dalis, ir dalis ekspertų gerai atlieka šią užduotį tačiau naujiems integracijų specialistams tai padaryti yra sunku dėl juodosios dėžės principų. Nepatyręs specialistas neturi pakankamai išankstinių žinių (angl. prior knowledge; žr. Paveikslas 1). Priežastinių ryšių žinių, verslo taisyklių trūkumas lemia kad integracijos projektai tampa labai sudėtingi ir dažnai nepasiseka [6] jų įgyvendinti ypač dinaminėje verslo aplinkoje.

5 Autonominiai skaičiavimai ir kontrolės teorija dinaminėse sistemose

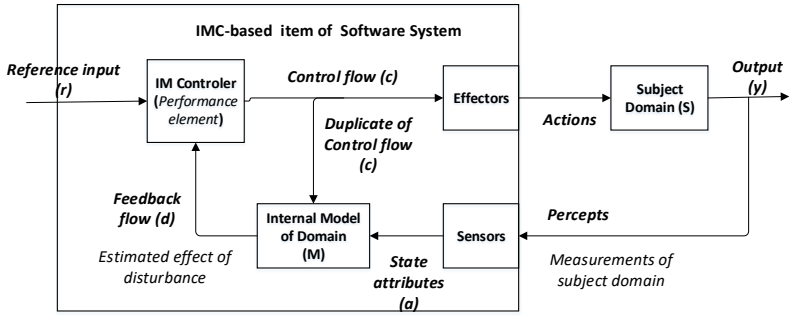
Disertacijos tema aprėpia autonominių skaičiavimų, kontrolės teorijos, verslo procesų ir duomenų integracijos paradigmas. Stengiamasi išnaudoti šių paradigmų metodologiją. Vidinis modelis padeda apibrėžti būtinus priežastinius ryšius vykstančius veiklos procese kuriuos būtų galima automatizuot kuriant integracijos ir sąveikumo sprendimą.



Paveikslas 5. Vidiniu modeliu paremta valdoma Sistema.

Vidinis modelis sukuria aplinką galimybei turėti daugiau nei vieną grįžtamojo ryšio kontūrą valdomoje sistemoje. Grįžtamasis ciklas perkelia informacinius srautus u, atsižvelgiant į aplinkos trikdžius d ir sistemos būsenos atributus y į vidinį modelį. Vidinio modelio pagalba galima geriau įvertinti sistemos būseną ir pateikti geresnius sprendimus nes yra žinomos funkcinės priklausomybės. Tokiu atveju d~ yra vidinis grįžtamasis ryšys, kuris padeda efektyviau valdyti procesą. Vidinio modelio taikymo pavyzdžiai pateikti lentelėje Lentelė 2. Vidinio modelio naudojimo pavyzdžiai.

Detalizuoto vidinio modelio pritaikymas laikantis IBM autonomic computing valdymo tranzakcijų principu kad valdomas objektas, tiksliau autonominis objektas turintis savyje žinių komponentą t.y. vidinį modelį, turėtų turėti sensorius ir efektorius. IM controller dalis atlieka kitas funkcijas: Stebėjima, analizę, planavimą, vykdymą (monitoring, analyze, plan, execute) (JACOB, Bart, et al. 2010).



Paveikslas 6. Taikomosios programos architektūros komponentas su vidiniu modeliu

6 Programų sistemų integracijos metodų ir EAI galimybės

Atlikus literatūros analizę įvardinti reikšmingi skirtumai tarp programų sistemų integracijos sprendimų. Lentelėje (Lentelė 1) pateikiamos žinomos programų sistemų integracijos metodologijos ir sprendimai. Lentelėje EAI sistemos tai verslo programų sistemų integracijos sistemos palengvinančios integracijos projektuotojų darbą, iš dalies automatizuojant programavimo procesą, dalinai palengvina integracijos projektavimo darbą.

Lentelė 1 [33] Žinomos programų sistemų integracijos metodologijos ir sprendimai.

		Algorithms (programming)	EAI:	Integration agents	Other methods
Short List of known integration problems	Different data sources	Assigned manually	Talend SAP PI [22, 23]	[9]	-
	Schema/Ontology mapping	Mapped manually		[9]	MAVERIC [12]
	Dynamic systems (schema changes)	Adapted manually		[9]	-
	Entity/ Object recognition	Do not use		No information	
	Data - sampling	Manual	[22, 23]	No information	
	Data heterogeneity	Solved manually		No information	
	Integration testing	Manual testing	[22, 23]	No information	
	Independent from staff	Staff required for maintenance and supervision			Unknown
	CRUD logic	Done manually	Talend, SAP PI [22, 23]	Use Frameworks	
	Goal oriented architecture	Usually None		[9]	
List of maximally reached capabilities	Solution maintenance **dependent** from staff	All	All	[9]	MAVERIC [12]
	Self - abilities	None	None	Able to change parameters, react to pre-determined changes	
	Autonomic Maturity Index [4]	1	2	3	4

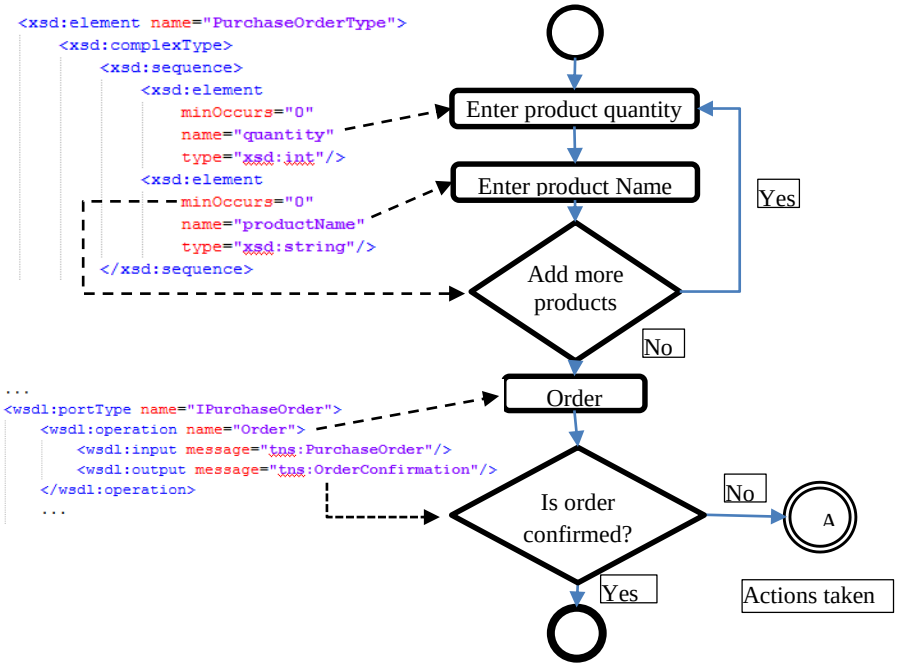
Lentelėje 1 analizuojami sprendimai neturi nei automatizavimo ir savęs valdymo galimybių. Dauguma analizuotų autorių šaltinių nepateikia konkrečių duomenų skirtų atkartoti jų pasiektus rezultatus ir įvertinti jų sukurtus algoritmus ir metodus programų

sistemų integracijos automatizavime. Lentelėje pateikiamos kai kurios tikrintos savybės kurios pasiekiamos konkrečia metodologija.

7 Nagrinėjamas autonominės integracijos sprendimas

7.1 WSDL integracijos analize

Verslo programų sistemų integracijos metodas gali būti papildytas autonominės kompiuterijos technologija kuri analizuotų WEB servisus ir tyrinėtų jų veiklą, tyrinėtų jų duomenų struktūrų dokumentus (WSDL, paveikslas 4).



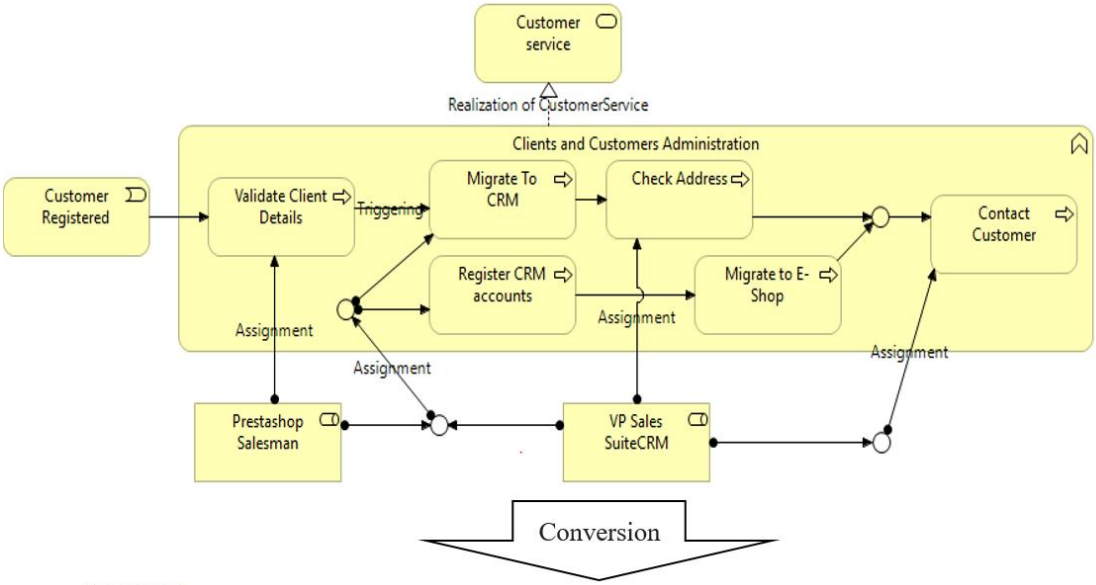
Paveikslas 7. [33] WSDL schema of Order to business process alignment

Pavyzdyje (paveiksle 4) pateikiama dalis **užsakymo sukūrimo proceso** iš vieno verslo subjekto. Šis verslo procesas pavaizduotas veiksmų sekos diagramos iškarpa (paveikslas 4) . 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. 4 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.

7.2 Duomenų iš modelio ištraukimas ir analizė

Ankstesniame skyriuje pavaizduota schema sujungta gauta dirbtiniu būdu, tačiau modeliuojant veiklos architektūrą dažniausiai naudojamasi įvairiomis priemonėmis.

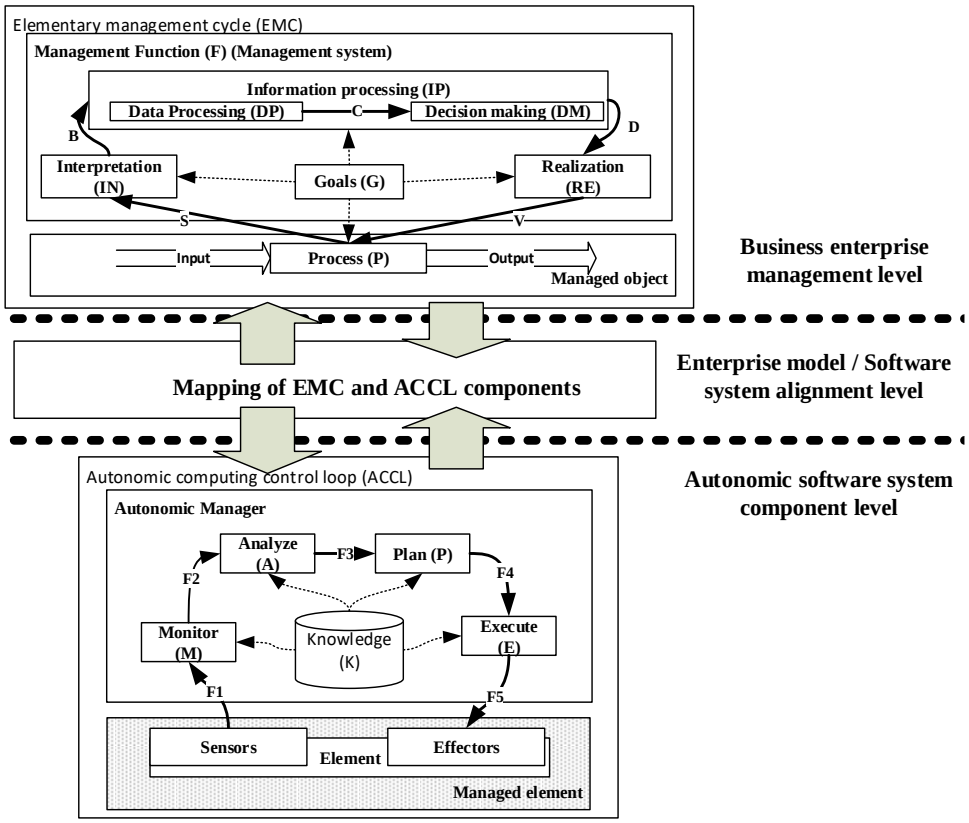
Todėl antrai daliai naudota ArchiMate veiklos modeliavimo priemonė leidžianti išsieksportuoti standartizuotą modelio failų formato dokumentą (angl. MEFF – Model Exchange File Format) sukurtą ArchiMate platformoje ir standartizuotą OMG. Šiuo metodu modelį lengva formatuoti XML formatu ir analizuoti su kitomis sisteminėmis priemonėmis.




```
- <element identifier="id-41bda1c1-9f41-406e-8a6a-c3ab9f5754ff" xsi:type="BusinessFunction">
    <name xml:lang="en">Clients and Customers Administration</name>
  </element>
- <element identifier="id-b3466b18-dfc9-40f5-83d6-7146531869fa" xsi:type="BusinessProcess">
    <name xml:lang="en">Validate Client Details</name>
  </element>
- <element identifier="id-476f670e-6f94-4bfb-9fb4-032e8328061b" xsi:type="BusinessProcess">
    <name xml:lang="en">Register CRM accounts</name>
  </element>
```

Šiuo metu MEFF yra patogi priemonė kuri vystoma ir palaikoma OMG organizacijos. Ja galima pasinaudoti programuojant integracijos sprendimus.

Toliau analizuojamas autonomines kompiuterijos komponento [29] ir elementaraus valdymo ciklo EVC [2] panašumai dėl kurių buvo iškelta teorinė galimybė programų sistemų integracijos automatizavimui panaudoti autonominės kompiuterijos principus.



Paveikslas 8. [33] Autonominės kompiuterijos komponento palyginimas [3] (apačioje) su elementariu valdymo ciklu [2] (viršuje)

Elementaraus valdymo ciklas (Paveikslas 5 EMC) buvo pristatytas prof. dr. Sauliaus Gudo [2] 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 [4] yra panašūs nes buvo kuriami realaus pasaulio veikimo principais. Tikslais grindžiami informacijos apdorojimo ciklai tarp EMC ir ACCL labai panašūs. Panašumai atvaizduojami Paveiksle 5. Išskirti šie panašūs elementaraus valdymo ciklai ir autonominės kompiuterijos valdymo ciklai: a) Interpretavimas (I) susijęs su stebėjimu (M) ir analizės komponentu (A) – abu analizuoja informaciją iš susijusių objektų gautų iš gautų iš ryšių F1, S. b) Informacijos valdymo procesas (IP) susijęs su planavimo ir vykdymo elementais nes abu elementai atlieka manipuliacijas su informacija. c) Realizacijos komponentas (RE) susijęs su vykdymo komponentu (E) bet taip pat susijęs su planavimo komponentu (P). d) Tikslai susieti su žiniomis nes

abu turi gauti informaciją ir instrukcijas. e) Abu komponentai EMC ir ACCL atsakingi už žemesnių lygių valdomus elementus. f) Abu komponentai turi grįžtamojo ryšio ciklą.
Panašumai tarp elementaraus valdymo ciklo EMC, ir autonominės kompiuterijos rodo kad šiuos verslo srities modelius galima interpretuoti ir suprasti autonominės kompiuterijos pagalba, kas padėtų sukurti naują modelį programų sistemų integracijos problemoms spręsti.

8 Vidiniu modeliu grindžiamos sistemos

Atlikta literatūros analizė parodė kad pagal iškeltus poreikius ir tikslus labiausiai tyrimo pritaikymui tiktų vidinio modelio konceptas. Vidinis modelis apibrėžtas 1970 kaip geras kontroliavimo teorijos pavyzdys (Conant et al, 1970). Mes išnagrinėjome kontroliavimo teorijos aspektus ir bandėme pritaikyti programinės įrangos integracijai ir sąveikoms dinaminėje verslo aplinkoje užtikrinti. Kadangi verslo aplinka dinaminė ir sprendimai turi būti pritaikomi pagal besikeičiančią verslo aplinką vidinio modelio valdymo schemą, mes ja tyrime bandėme pasinaudoti kurdami teorinę autonominę integracinę sistemą. Pagal išnagrinėtus autorius ir vidinio modelio panaudojimą galime išskirti šią reikšmingą literatūrą

Lentelė 2. Vidinio modelio naudojimo pavyzdžiai

Autoriai	Tema	Naudojimas
Moen et al., 2006		quality management, risk management, business process management
Brache, 2002		quality management, risk management, business process management
Fayol, 2016		Fayol's business function model
(Porter et al., 1985);		Porter's Value Chain Model
		Deming's PDCA cycle
		Rummler-Brach's enterprise performance management model
Kephart et al., 2003		intelligent agents, autonomic computing
Winter et al., 1998		reactive software applications
Mareels et al., 1996		adaptive systems
Abdelzaher et al., 2008		computing systems

Iš šio tyrimo iškelta prielaida, kad programų sistemų inžinerijoje figūruoja dvi programų architektūros modeliavimo paradigmos: juodosios dėžės ir baltosios dėžės.

9 Išvados

- Tyrinėjant programų sistemų integracijos sritį pastebėta, kad problemų yra didelis sąrašas ir kad jų sprendimo apimtis gali būti per didelė disertacijos darbui, todėl nuspręsta siaurinti temą analizuojant ir gilinantis į vieną iš problemų: integruojamų komponentų orkestravimo ir choreografijos automatizavimą taikant verslo žinių modelius.
- Išnagrinėta 33 literatūros šaltiniai kurių gauta informacija leido identifikuoti pagrindinių problemų programų sistemų integracijos srities sąrašą.
- Analizuojant autonominės kompiuterijos galimybes ir struktūra pastebėti panašumai tarp veiklos procesų valdymo modelio struktūros elementaraus valdymo ciklo. Panašumai leido išskirti bendras savybes kuriomis remiantis galima būtų išgauti reikiamą informaciją iš veiklos procesų panaudojant ją integracijos kūrime.
- Tęsiamas eksperimentinis tyrimas turėtu leisti įvertinti sąveikavimo galimybes tarp programų sistemų, tai būtų naujas vertinimo būdas sukurtas tokių sistemų sąveikumo galimybių vertinimui
- Tyrimo metu ištyrinėtos prielaidos kad panaudojant ir taikant organizacijų veiklos modelius visada remiamasi vidiniu modeliu su būtinai esančia hierarchine struktūra. (pvz. Modeliuojant Realu pasaulį, kuriant veiklos procesų modelį, kuriant programų sistemų procesų modelį ir kuriant programos architektūrą visuomet yra taikomas vidinis modelis su ankstesniuose modeliuose įgytomis žiniomis.)

Literatūra

1. **A. Valatavicius, D. Dilijonas.** Dynamic B2B process integration (in Lithuanian). In: Proceedings of "Informacinės Technologijos", pp. 34-39. Kaunas (2014)
2. **S. Gudas.** Foundations of the information systems' engineering theory (in Lithuanian). Vilnius University, Vilnius (2012)
3. **J. O. Kephart, D. M Chess.** The Vision of Autonomic computing. IBM Thomas J. Watson research center (2003)
4. **B. Jacob, R. Lanyon-Hogg, D. K Nadgir, A. F. Yassin.** A Practical Guide to the IBM Autonomic Computing Toolkit (2004)
5. **E. Rahm, P. A. Bernstein.** A survey of approaches to automatic schema matching. In VLDB. Volume 10, pp. 334-350 (2001)
6. **G. Trotta.** Dancing Around EAI 'Bear Traps'. In Business Process Management (BPM) Best Practices (2003)
7. **El-Halwagi, M.M.** Process Integration. Elsevier. ISBN-9780123705327 (2006)
8. **S. Tan, K. Liu, Z. Xie**. A semiotic approach to organizational modeling using norm analysis. In: Universityof Reading (2006)
9. **Y. Peng, T. Finin, Y. Labrou, B. Chu, J Long, W.J. Tolone, A. Boughannam.** A multi-agent system for enterprise integration. In: Proceedings of PAAM (1998)
10. **A. Halevy, A. Rajaraman, J. Ordille.** Data Integration: The teenage years (2006)
11. **X. Luna Dong, F. Naumann**. Data Fusion – Resolving data conflicts for integration. In: VLDB Endowment, pp 1654-1655 (2009)
12. **R. McCann, B AlShelbi, W. Le, Hoa Nguyen, L. Vu, A. Doan.** Mapping Maintenance for Data Integration Systems. In: VLDB (2005)

13. **P.A. Bernstein, J. Madhavan, E. Rahm.** Generic schema matching 10 years later. In: VLDB. Volume 4, (2011) 11.
14. **G Pavlin, M Kamermans, M. Scafeş.** Dynamic Process Integration Framework: Toward Efficient Information Processing in Complex Distributed Systems. In: Proceedings of the 3rd International Symposium on Intelligent Distributed Computing (2009)
15. **R-D. Kutsche, N. Milanovic.** Model-Based Software and Data Integration. In: MBSDI (2008)
16. **B.J. Overeinder, P.D. Verkaik and F.M.T. Brazier**. Web Service Access Management for Integration with Agent Systems.In: SAC'08 (2008).
17. **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)
18. **L. Li, B. Wu, Y. Yang.** Agent-based Ontology Integration for Ontology-based Applications. In: CRPIT (2005)
19. **P. Shvaiko, J. Euzenat.** Ontology matching: state of the art and future challenges. In: IEEE Transactions on Knowledge and Data Engineering. Volume 25 (2011) 158 – 176.
20. **X. Luna Dong, F. Naumann**. Big Data Integration. In: VLDB Endowment, pp. 1188-1189 (2009)
21. **G. Hohpe and B. Woolf.** Enterprise Integration Patterns. ISBN-13: 978-0321200686 (2011)
22. Talend Data Integration solution, https://www.talend.com/products/data-integration
23. Process Integration (PI) & SOA Middleware, http://scn.sap.com/community/pi-and-soa-middleware
24. **Dirk Krafzig, Karl Banke, Dirk Slama**. Enterprise SOA. Service-Oriented Architecture: Best Practices, 2004.
25. **OpenGroup**. Service Oriented Architecture: What Is SOA?, 2013
26. **L. Silverston**. The Data Model Resource book. Volume 1 (2001).
27. **E. Pemhert. J. Eberius, E. Rahm**. A self-configuring Schema Matching System. In Data Engineering (ICDE), 2012 IEEE 28th International Conference. (2012) 306 – 317.
28. Bart Jacob, Richard Lanyon-Hogg, Devaprasad K Nadgir, Amr F Yassin, A Practical Guide to the
29. IBM Autonomic Computing Toolkit, 2004
30. **Ingo Feinerer**, A Formal Treatment of UML Class Diagrams as an Efficient Method for Configuration Management, Institute of Computer Languages Vienna University of Technology, 2007;
31. **Mr Kenneth J. Sherry**, Business Process Modelling with BPMN, 2012;
32. **Anton Michlmayr, Florian Rosenberg, Christian Platzer, Martin Treiber, Scharam Dustdar.** Towards Recovering the Broken SOA Triangle – A Software Engineering Perspective. In IW-SOSWE'07, 2007;
33. **A. Valatavičius, S. Gudas**, Enterprise software system integration using autonomic computing. In CEUR-WS.org/Vol-1420, 156-163;

ABDELZAHER, Tarek et al. Introduction to control theory and its application to computing systems. In: Performance Modeling and Engineering. Springer US, 2008. p. 185-215.
ArchiMate® 3.0 Specification [interactive], The Open Group, 2016. Document Number: C162 [reviewed 2017 y. June 15 d.]. Internet access: <http://pubs.opengroup.org/architecture/archimate3-doc/>. ISBN: 1-937218-74-4.

ASHBY, W. Ross .An introduction to cybernetics. S.l.: London Chapman & Hall Ltd, 1956 [reviewed 2017 y. June 9 d.]. Internet access: <http://dspace.utalca.cl/bitstream/1950/6344/2/IntroCyb.pdf>.
BARTON, Rick. *Talend Open Studio Cookbook*. Packt Publishing Ltd., 2013. ISBN:1782167277.
BENATALLAH, Boualem et al. *Developing adapters for web services integration*. In International Conference on Advanced Information Systems Engineering. Springer Berlin Heidelberg, 2005. p. 415-429.
BERNSTEIN, Philip A., et al. *Generic schema matching, ten years later*. In Proceedings of the VLDB Endowment, 2011. 4, (11) p. 695-701.
BRACHE, Alan P. How organizations work: Taking a holistic approach to enterprise health. John Wiley & Sons, 2002.
CONANT, Roger C.; ASHBY, W. Ross. *Every good regulator of a system must be a model of that system*. In: International journal of systems science, 1970. 1, (2) p.89-97.
CZARNECKI, Krzysztof; HELSEN Simon. Classification of model transformation approaches. In: Proceedings of the 2nd OOPSLA Workshop on Generative Techniques in the Context of the Model Driven Architecture, 2003. 45, (3).
DIETZ, Jan LG. *The deep structure of business processes*. Communications of the ACM, 2006. 49(5) p. 58-64.
DONG, Xin Luna; NAUMANN Felix. *Data fusion: resolving data conflicts for integration.* In: Proceedings of the VLDB Endowment, 2009. 2, (2) p.1654-1655.
DONG, Xin Luna; SRIVASTAVA, Divesh. *Big data integration*. In: Data Engineering (ICDE), IEEE 29th International Conference, 2013. p. 1245-1248.
European interoperability framework for pan-european egovernment services. European Communities, 2004 [reviewed 2017 y. June 3 d.]. Internet access: <http://ec.europa.eu/idabc/servlets/Docd552.pdf>. ISBN 92-894-8389-X.
EL-HALWAGI, Mahmoud M. Process integration. Academic Press, 2016. 7. ISBN 0-12-370532-0.
FAYOL, Henri. *General and industrial management*. Ravenio Books, 2016.
FEINERER, Ingo. A formal treatment of UML class diagrams as an efficient method for configuration management. na, 2007.
FOWLER, Martin. (UML distilled: a brief guide to the standard object modeling language. Addison-Wesley Professional, 2004.
FRANCIS, Bruce A.; WONHAM, W. Murray. The internal model principle of control theory. In: Automatica, 1976. 12, (5) p. 457-465.
GALASSO, François, et al. *A method to select a successful interoperability solution through a simulation approach*. Journal of Intelligent Manufacturing, 2016. 27, (1) p. 217-229.
GAUDIN, Benoit, et al. Nixon A Control Theory-Based Approach for Self-Healing of Un-handled Runtime Exceptions. In: Proceeding of the 8th ACM international conference on Autonomic computing, ACM, 2011. p. 217-220.
GEORGAKARAKOU, Chrysanthi E.; ECONOMIDES Anastasios A. Software Agent Technology: an Overview Application to Virtual Enterprises. Agent and Web Service Technologies in Virtual Enterprises, N.
Protogeros (ed.), Idea Group Publ.
GEORGAKOPOULOS, Diimitrios; HORNICK, Mark; SHETH, Amit. An Overview of Workflow Management: From Process Modeling to Workflow Automation Infrastructure. In: Distributed and Parallel Databases, 1995. 3, (2) p. 119-153.
GLANVILLE, Ranulph. Second order cybernetics. In: Systems Science and Cybernetics, 2002. 3 p. 59-85.
GROSSMANN, Georg; SCHREFL Michael; STUMPTNER, Markus. *Exploiting semantics of inter-process dependencies to instantiate predefined integration patterns*. In: Australian Computer Society, 2007. .p 155-16.
GUDAS, Saulius. Information Systems Engineering and Knowledge-Based Enterprise Modelling: Towards Foundations of Theory. In: Springer Proceedings in Business and Economics, 2016. p. 481- 497. ISBN 978-3-319-33865-1.
GUDAS, Saulius; LOPATA, Audrius. Towards internal modelling of the information systems application domain. In: *Informatica*, 2016. 27, (1) p. 1 – 29. ISSN 0868-4952.
GUDAS, Saulius; LOPATA, Audrius. Meta-model based development of use case model for a business function. In: Information Technology and Control, 2015. 36, (3).
GUDAS, Saulius. Foundations of the information systems' engineering theory. In: Vilnius University Press, 2012. p. 384.
GUDAS, Saulius. Knowledge-Based Enterprise Framework: A Management Control View. In: New Research on Knowledge Management Models and Methods. InTech, 2012.
HALEVY, Alon; RAJARAMAN, Anand; ORDILLE, Joann. Data integration: the teenage years. In: Proceedings of the 32nd international conference on Very large data bases. VLDB Endowment, 2006. p. 9-16.
HEYLIGHEN, Francis; JOSLYN, Cliff . Cybernetics, and Second-Order Cybernetics. In: Encyclopedia of physical science & technology, 2001. 4, p. 155-170.
HOHPE, Gregor, WOOLF Bobby . *Enterprise integration patterns*. In: 9th Conference on Pattern Language of Programs, 2002. p. 1-9
HUEBSCHER, Markus C.; MCCANN, Julie A. A survey of autonomic computing-degrees, models, and applications. ACM Computing Surveys (CSUR), 2008. 40, (3) p. 7.
JACOB, Bart, et al. *A practical guide to the IBM autonomic computing toolkit. In: IBM Redbooks, 2004*. 4, p. 10.
KARDOŠ, Martin; DROZDOVÁ, Matilda. *Analytical method of CIM to PIM transformation in Model Driven Architecture (MDA)*. Journal of Information and Organizational Sciences, 2010. 34, (1) p. 89-99.

KEPHART, Jeffrey O.; CHESS, David M. *The vision of autonomic computing. In: Computer 2003*. 36, (1) p. 41-50.
KUMAR, Shrawan. *Kac-Moody groups, their flag varieties, and representation theory*. In: Springer Science & Business Media, 2012.
KUTSCHE, Ralf-Detlef; MILANOVIC Nikola, eds Model-Based Software and Data Integration: First International Workshop. Proceedings. Vol. 8. Springer Science & Business Media. MBSDI, 2008.
KRAFZIG, Dirk; BANKE Karl; SLAMA, Dirk. *Enterprise SOA: service-oriented architecture best practices*. Prentice Hall Professional, 2005.
KROGSTIE, John. EEML2005: extended enterprise modeling language. Norvegian University of Science and Technology, 2005.
LABROU, Y. Peng1 et al. *A multi-agent system for enterprise integration, 1998*.
LANKHORST, Marc. Communication of Enterprise Architectures. In: Enterprise Architecture at Work, 2013. p. 61-74.
LI, Li; WU, Baolin; YANG, Yun. *Agent-based ontology integration for ontology-based applications*. In: Proceedings of the 2005 Australasian Ontology Workshop-Volume 58. Australian Computer Society, Inc., 2005. 58 p. 53-59.
MAREELS, Iven; POLDERMAN, Jan Willem. Adaptive systems: an introduction. In: Springer Science & Business Media, 2012. ISBN 978-1-4612-6414-9.
MCCANN, Robert, et al. *Mapping maintenance for data integration systems*. In: Proceedings of the 31st international conference on Very large data bases. VLDB Endowment, 2005. p. 1018-1029.
MEDINA-MORA, Raul, et al. The action workflow approach to workflow management technology. In: Proceedings of the 1992 ACM conference on Computer-supported cooperative work. ACM, 1992.
MICHLMAYR, Anton, et al. *Towards recovering the broken SOA triangle: a software engineering perspective*. In: 2nd international workshop on Service oriented software engineering: in conjunction with the 6th ESEC/FSE joint meeting. ACM, 2007. p. 22-28.
MOEN, Ronald; NORMAN Clifford. *Evolution of the PDCA cycle*, 2006.
OSIS, Janis. Software development with topological model in the framework of MDA. In: CAiSE Workshops, 2004. 1, p. 211 – 220.
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.
PAPAZOGLOU, Michael P., et al. *Service-oriented computing: a research roadmap*. In: International Journal of Cooperative Information Systems 2008. 17, (2) p. 223-255.
PARASHAR, Manish; HARIRI Salim. Autonomic computing: An overview. In: Unconventional Programming Paradigms, 2005. p. 97-97.
PAVLIN, Gregor; KAMERMANS, Michiel; SCAFES, Mihnea. Dynamic process integration framework: Toward efficient information processing in complex distributed systems. In: Informatica, 2010. 34, (4).
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.
PORTER, Michael E.; MILLAR Victor E. *How information gives you a competitive advantage*, 1985.
RAHM, Erhard; BERNSTEIN, Philip A.. *A survey of approaches to automatic schema matching*. In: The VLDB Journal, 2001. p. 334-350.
SENDALL, Shane; KOZACZYNSKI Wojtek. Model transformation: The heart and soul of model-driven software development. IEEE software, 2003. 20, (5) p. 42-45.
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.
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.
TIN, Chung; POON, Chi-Sang. *Internal models in sensorimotor integration: perspectives from adaptive control theory*. Journal of Neural Engineering, 2005. 2, (3), S147.
TROTTA, Gian. *Dancing Around EAI'Bear Traps'*. Business Process Management (BPM) Best Practices, 2003.
VALATAVICIUS, Andrius; DILIJONAS, Darius. *Dynamic B2B process integration*. In: Proceedings of "Informacinės Technologijos, 2014. P. 34-39.
VALATVICIUS, Andrius; GUDAS, Saulius. *Enterprise Software System Integration Using Autonomic Computing*. CEUR-WS. org, 1420, 2015. p. 156-163.
VAN DEN BOSCH, Marcel APM, et al. *A selection-method for Enterprise Application Integration solutions*. In International Conference on Business Informatics Research, 2010. p. 176-187.
VERNADAT, François. UEML: towards a unified enterprise modelling language. In: International Journal of Production Research, 2002. 40, (17) p. 4309-4321.
VERNADAT, François B. Interoperable enterprise systems: Principles, concepts, and methods. In: Annual reviews in Control, 2007. 31, (1) p. 137-145.
WINOGRAD, Terry; FLORES Fernando. Understanding computers and cognition: A new foundation for design. Intellect Books, 1986.
WHITE, Stephen A. et al. *BPMN 2.0 handbook second edition: methods, concepts, case studies and standards in business process modeling notation*. Future strategies, inc., 2011.

WINTER, Kirsten; SANTEN Thomas; HEISEL Maritta. An agenda for specifying software components with complex data models. In: Computer Safety, Reliability and Security, 1998. p. 16-31.

ZACHMAN, John A. A framework for information systems architecture. IBM systems journal, 1987. 26, (3) p. 276-292.

ZINNIKUS, Ingo; HAHN Christian; FISCHER Klaus. A model-driven, agent-based approach for the integration of services into a collaborative business process. In: Proceedings of the 7th international joint conference on Autonomous agents and multiagent systems-Volume 1. In: International Foundation for Autonomous Agents and Multiagent Systems, 2008. p. 241-248.