CRITERIONS FOR ARCHITECTURE ESTIMATION AND ARCHITECTURE SELECTION OF E-SERVICES SYSTEM

КРИТЕРИИ ДЛЯ ОЦЕНКИ АРХИТЕКТУРЫ СИСТЕМЫ ЭЛЕКТРОННЫХ УСЛУГ И ВЫБОР АРХИТЕКТУРЫ.

ELEKTRONISKO PAKALPOJUMU SISTĒMU ARHITEKTŪRAS NOVĒRTĒŠANAS KRITĒRIJI UN ARHITEKTŪRAS IZVĒLE

E. Žeiris, M. Ziema

Atslēgas vārdi: e-pakalpojums, algoritma grafs, web servisu grafs, pareto kopa, daudzkriteriāla optimizācija, e-pakalpojuma arhitektūra, web servisu grafa novērtēšanas kritēriji.

1. Elektronisks pakalpojums un tā realizācijas arhitektūra

Elektronisks pakalpojums (e-pakalpojums) tradicionāli tiek realizēts kā informācijas sistēmu darbību kopums. E-pakalpojums sevī satur vairāku sistēmu funkcionālās iespējas, kas rezultātā sniedz materiālu vai arī nemateriālu labumu sabiedrībai (fiziskām un juridiskām personām). Lai varētu runāt par e-pakalpojuma realizācijas arhitektūru, ir nepieciešams iziet cauri vairākiem tradicionālā pakalpojuma elektronizācijas posmiem (koncepcija, procesa specifikācija, elektronizācijas pakāpes, procesa formalizācija, funkcionālā specifikācija, projektēšana, implementēšana, ieviešana). Šajā rakstā tiks aplūkots tikai viens no pēdējiem posmiem – pakalpojuma soļu servisu izveide SOA (servisu orientēta arhitektūra) vidē, kas jāveic, e-pakalpojuma projektēšanas fāzē[1][2].

Ar e-pakalpojuma arhitektūru šajā rakstā mēs sapratīsim pakalpojumā iesaistītos Web servisus un to savstarpējās saites, kas ļauj sasniegt konkrēta pakalpojuma mērķi.

Pakalpojums pēc savas būtības sastāv no vairākām izpildāmām darbībām, kas izpildās sinhroni vai asinhroni. Katrai no darbībām ir jābūt pilnīgai, lai atbilstu SOA principiem – vienas darbības izpilde nav saistīta ar citu darbību stāvokli. Veidojot Web servisus SOA vidē, ir jāvadās pēc diviem pamatprincipiem[3]. Pirmkārt augsta kohēzija - tas nozīmē, ka vienā Web servisā ir jāapvieno vienveidīgas darbības un otrkārt minimāla savstarpējā atkarība – Web servisam jāstrādā autonomi, neatkarīgi no citiem servisiem, lai palielinātu tā atkārtotu izmantojamību.

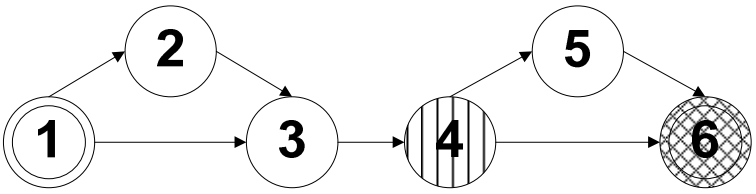
Ērtības labad šajā rakstā ar pakalpojuma vienu darbību sapratīsim Web servisu, kas to realizē. Projektējot e-pakalpojumu, ir jārisina problēma, kā sadalīt pakalpojuma algoritmu pa Web servisiem. Vai projektēt vienu Web servisu, vai maksimāli daudz, kur katrs Web serviss satur minimālu funkcionalitāti, lai tas atbilstu SOA? Šajā rakstā tiks sniegta metodika, kā veikt izvēli starp vairākiem e-pakalpojuma arhitektūras risinājumiem.

Lai izvēlētos pēc iespējas labāku e-pakalpojuma arhitektūru, tiek piedāvāts izmantot grafu teoriju. Sākotnēji ir jādefinē e-pakalpojuma algoritma grafs, kas pēc savas būtības ir e-pakalpojuma izpildes algoritma apraksts orientēta grafa veidā, kur katra virsotne satur noteiktu skaitu e-pakalpojumā izpildāmu darbību, lai tā atbilstu SOA principiem un loki ir informatīvas saites starp darbībām. Algoritma grafu ir iespējams segmentēt dažādos veidos, tādējādi mainot algoritma darbības parametrus. Grafa segmentācija nozīmē to, ka tiek meklēti grafa virsotņu apvienojumi visos iespējamajos veidos. Segmentēto grafu var pārvērst par Web servisu grafu, kur katrs Web serviss realizē funkcionalitāti no algoritma grafa segmenta. Šādu Web servisa grafu, kas radies algoritma grafa segmentēšanas rezultātā, var uzskatīt par e-pakalpojuma arhitektūras raksturotāju.

Arhitektūras izvēli var novest uz tādu Web servisu grafu kopas atrašanu, kuri atbilst Pareto – optimumam pēc *N* uzdotiem kritērijiem.

2. Algoritma grafs

E-pakalpojuma būtību nosaka tā izpildes algoritms. Ja ir zināms e-pakalpojuma izpildes process, tad ir iespējams precīzi aprakstīt arī pakalpojuma izpildes algoritmu, piemērām, kādā nosacītā programmēšanas valodā. Lai varētu aprakstīt e-pakalpojumu kā Web servisu kopu ar savstarpējām saitēm, kas apraksta dotā pakalpojuma arhitektūru, definēsim e-pakalpojuma algoritma grafu, kā orientētu grafu $G = (S, L)$, kur $S = \{s_1, s_2, ...s_n\}$ ir galīga kopa - grafa virsotnes, kas pēc savas būtības ir e-pakalpojuma algoritma izpildāmas darbības, un $L \subset S \times S$. Loks $l_i = (s_j, s_k)$ grafā nozīmē to, ka e-pakalpojuma algoritma izpildē pēc darbības s_j izpildes seko darbības s_k izpilde. Loki e-pakalpojuma algoritma grafā norāda informācijas plūsmu. E-pakalpojuma algoritma grafa piemērs parādīts 1. attēlā. Virsotne, kas apzīmēta ar „1" ir algoritma sākums. Virsotnes „4" un „6" ir dažādu izstrādātāju veidotas, tāpēc tās tiek iekrāsotas dažādās krāsās.



1. attēls. E-pakalpojuma algoritma grafs.

Veidojot e-pakalpojuma algoritma grafu, lai to varētu izmantot par pamatu e-pakalpojumu arhitektūras izveidē, ir jāņem vērā daži ierobežojumi, kas saistīti ar SOA:

- Katrai algoritma grafa virsotnei s_i ir jārealizē sevī noslēgta izpildāma darbība. Tas nozīmē, ka virsotne darbojas atomārā transakcijā un nav saistīta ar citās virsotnēs realizētajiem algoritmiem. Šis nosacījums ir saistīts ar augstu kohēziju un minimālu atkarību.
- Katrai virsotnei ir jāsatur vismaz viena izpildāma darbība. Praksē tas ir saistīts ar algoritmu implementējošajām metodēm. Turpmākiem mērķiem apzīmēsim virsotnes darbības implementējošās metodes ar $M_{s_i} = \{m^0_{s_i}, m^1_{s_i}, m^2_{s_i}, ...m^g_{s_i}\}$ un katras metodes $m^j_{s_i}$ rindiņu skaitu ar $O(m^j_{s_i})$
- Darbības, kas atkārtojas algoritma izpildes laikā nevar tikt veidotas grafā kā dažādas virsotnes $\forall s_i, s_j, \quad s_i \neq s_j, \quad s_i \in S, \quad s_j \in S, \quad M_{s_i} \cap M_{s_j} = \varnothing$. Tas ir nepieciešams lai nodrošinātu augstu kohēziju un jau sākotnēji izslēgtu nevajadzīgu variantu apstrādi.

Lai varētu veikt grafa segmentēšanu un formāli to aprakstīt, aplūkosim grafu *G* kā attēlojumu Γ virsotņu kopā S, katrai virsotnei piekārtojot apakškopu $(s_i \subset S)$, kas pēc savas būtības ir no virsotnēs s_i sasniedzamās virsotnes. Γs_i ir no virsotnes izejošie loki un $\Gamma^{-1} s_i$ ir virsotnē ieejošie loki. Šādam e-pakalpojumu algoritma grafam eksistē vismaz viena virsotne $s_0 (s_0 \in S)$, kurā neieiet neviens loks $\Gamma^{-1} s_0 = \varnothing$, kuru sauksim par e-pakalpojuma algoritma sākumu un tāpat arī eksistē vismaz viena virsotne $s_r (s_r \in S)$ no kuras neiziet neviens loks $\Gamma s_r = \varnothing$, kuru sauksim par e-pakalpojuma algoritma rezultātu.[4]

Piemēram, 1. attēlā parādītajam grafam $s_0 = \{"1"\}$, $s_r = \{"6"\}$, $\Gamma\{"4"\} = \{"5","6"\}$, $\Gamma^{-1}\{"4"\} = \{"3"\}$.

3. Web servisu grafs un tā raksturlielumi

Web servisu grafs tiek iegūts segmentējot algoritmu grafu. Par cik ir iespējamas vairākas segmentācijas, definēsim kopu $X = \{G'\}$, kas satur visus iespējamos grafus, kas ir rekursīvi atvasināti no sākotnējā Web servisu grafa *G*. Grafa pārveidojumi tiek veikti, apvienojot virsotnes. Divu virsotņu s_i un s_j apvienojums s' ir kā abu virsotņu izejošo un ieejošo loku apvienojums. $\Gamma s' = \Gamma s_i \cup \Gamma s_j$ un $\Gamma^{-1} s' = \Gamma^{-1} s_i \cup \Gamma^{-1} s_j$. Metožu kopa jaunizveidotajā virsotnē veidojas sekojoši $M_{s'} = M_{s_i} \cup M_{s_j}$ Visi mērījumi, apvienojot virsotnes, summējas.

E-pakalpojuma izpildē bieži vien ir iesaistītas dažādas funkcionālās sistēmas, kuras ir veidojuši dažādi izstrādātāji, līdz ar to principā nav iespējama šo sistēmu darbību realizācija vienā Web servisā. Šim nolūkam ir nepieciešams katrai algoritma grafa virsotnei jau sākotnēji noteikt ar ko tā nav apvienojama. Citiem vārdiem sakot katrai algoritma grafa virsotnei piekārtosim pazīmi $I(s_i)$. Šādā veidā virsotņu apvienojums s' ir iespējams tikai tad, ja $I(s_i) = I(s_j)$.

Definēsim Web servisu grafa raksturlielumus. Web servisu grafu raksturo :

- Izstrādes izmaksas *C;*
- Ātrdarbība *T*;
- Uzturēšanas sarežģītība (izmaksas) *E*;
- Web servisu atkārtota izmantojamība *R*;
- Integritāte *I*.

Turpmākās e-pakalpojumu Web servisu grafa metrikas un to formulas ir uzdotas tādā veidā, ka tās visas ir minimizējamas.

Izstrādes izmaksas

Lai izrēķinātu katras Web servisu grafa virsotnes izstrādes izmaksas C_{s_i}, pielietosim sekojošu algoritmu:

$$C_{s_i} = \left(\sum_{m_{s_i} \in M_{s_i}} \mathrm{O}(m_{s_i}) + W_C(s_i) \right)^p \qquad (1)$$

$W_C(s_i)$ ir Web servisu grafa virsotnes s_i Web servisa implementācijas koda rindiņu skaits, kas ir konstante katrai no Web servisu realizācijas vidēm. Pakāpe p raksturo attiecīgas realizācijas programmēšanas valodas implementācijas sarežģītību (*p* pēc savas būtības ir daļskaitlis). E-pakalpojuma izstrādes izmaksas attiecīgi tiek aprēķinātas sekojoši:

$$C = \sum_{s_i \in S} C_{s_i} \qquad (2)$$

Ātrdarbība

E-pakalpojuma ātrdarbības aprēķiniem katram Web servisu grafa stāvoklim piekārtosim vidējo tā izpildes laiku milisekundēs pie vidēja apjoma ieejas datiem t_{s_i}. Lai katru no stāvokļiem izpildītu e-pakalpojumu realizācijas vidē, ir nepieciešams papildus laiks uz katru no stāvokļiem $t_{ws} = \log(t_{s_i})$ E-pakalpojuma kopējās ātrdarbības aprēķins

$$T = \sum_{s_i \in S} (t_{s_i} + t_{ws}) + t_{epak} + t_{data} \qquad (3)$$

Izpildes laikam jāpieskaita papildus laiks $t_{epak} = \log(n)$, kas nepieciešams, lai darbinātu e-pakalpojumu kādā no e-pakalpojumu izpildes vidēm, kur $n = |S|$ (Web servisu skaits grafā), un laiks, kas nepieciešams, lai pārsūtītu datus no viena e-pakalpojuma stāvokļa uz nākamo

$t_{data} = \sum_{l_i \in L} t_{l_i}$. Katram no Web servisu grafa lokiem $l_i \in L$ ir jāpiekārto vidējais laiks milisekundēs t_{l_i} , kas nepieciešams, lai pārsūtītu datus pa šo loku.

Uzturēšanas izmaksas

E-pakalpojuma uzturēšanas izmaksas ir atkarīgas no tā cik daudz atkarību ir e-pakalpojuma Web servisu grafā un cik izmaksā izmaiņas. Jo vairāk ir atkarību un vairāk ir koda vienā Web servisā, jo grūtāk ir veikt izmaiņas, kā arī ir potenciāli vairāk iespēju rasties kļūdām, kas palielina uzturēšanas izmaksas Ekspluatācijas izmaksas aprēķināsim visam Web servisu grafam.

$$E = k \times \frac{\sum_{m_{s_i} \in M_{s_i}} \mathrm{O}(m_{s_i})}{n}, \quad (4)$$

kur $k = |L|$ loku skaits grafā.

Atkārtota izmantojamība

Atkārtota izmantojamība tik iegūta maksimāli tad, ja ir nodalītas metodes, kā individuāli Web servisi, ko var pielietot kā atsevišķus vienumus. Tādā veidā palielinās iespēja, ka pie nepieciešamības būs pieejams jau gatavs Web serviss, kas implementē nepieciešamo funkcionalitāti. Atkārtotu izmantojamību aprēķināsim sekojoši.

$$R = \frac{\sum_{s_i \in S} |M_{s_i}|}{n}, \quad (5)$$

kur $|M_{s_i}|$ metožu skaits Web servisu grafa stāvoklī S_i.

Integritāte

Tā kā Web servisu grafs no praktiskās implementācijas viedokļa var tikt sadalīts tīklā, kur katrs pakalpojuma solis ir jāizsauc attālināti, tad ir iespējams ietekmēt pakalpojuma integritāti, ja nav sasniedzams vai neizpildās kāds no Web servisiem. No šāda viedokļa integritāte būs vislabākā, ja visu funkcionalitāti realizēs viens serviss un nenotiks datu pārsūtīšana starp vairākiem (minimāls loku skaits)[5][6][7].

$$I = k \quad (6)$$

4. Arhitektūras izvēle kā daudzkriteriāls optimizācijas uzdevums

Nepieciešams atrast visus iespējamos Web servisu grafus no kopas *X*, kas ir vislabākie (vai arī ne sliktāki) pēc visiem izvirzītajiem kritērijiem (izstrādes izmaksas, ātrdarbība, uzturēšanas sarežģītība, atkārtota izmantojamība, integritāte). Būtiski ir aplūkot visus kritērijus vienlaicīgi, jo nevar pateikt kurš no tiem ir svarīgāks, kā arī tie savstarpēji nav salīdzināmi.

Šo uzdevumu risināsim kā daudzkriteriālu uzdevumu, kas reducējas uz pareto kompromisu kopas P atrašanu[8]:

$$Q(X) \to \min_{X \in \Omega} \to P, \quad (7)$$

kur $Q(X) = \{q_1(X), q_2(X) \ldots q_N(X)\}$ - minimizējamie kritēriji; Ω - X definīcijas apgabals.

Risinājumu $X_{i^*} \in \Omega$ sauksim par pareto optimālu $X_{i^*} \in P$ tad un tikai tad, ja neeksistē $X_j \in \Omega$ tāds, ka

$$q_i(X_j) \le q_i(X_{i^*}) \quad (8)$$

visiem $i = \{1,2,...N\}$, kur vismaz viena ir stingra nevienādība. Citiem vārdiem sakot, jebkurai vērtībai $X_{i^*} \in P$ kopā Ω nav atrodama labāka vērtība par izvēlēto.

Ir dots e-pakalpojuma algoritma grafs. Sākotnēji tas tiek pieņemts par Web servisu grafu *G*. Uzdevums ir atrast grafu kopu $P = \{G'^*\}$, kas atbilst Ω ierobežojumiem un minimizē kritērijus Q (mūsu gadījumā q_1 ir izstrādes izmaksas, ko aprēķina, izmantojot formulu (2), q_2 ir ātrdarbība, ko aprēķina, izmantojot formulu (3), q_3 ir ekspluatācijas sarežģītība, ko aprēķina, izmantojot formulu (4), q_4 ir atkārtota izmantojamība, ko aprēķina, izmantojot formulu (5) un q_5 ir integritāte, ko aprēķina, izmantojot formulu (6)).

Atrastās kopas *P* Web servisu grafi ir iespējamie arhitektūras risinājumi.

5. Web servisu grafu optimizācijas piemērs

Piedāvāto metodi demonstrēsim ar vienkāršu piemēru. Veiksim Web servisu grafu izvēli kā daudzkriteriālu optimizāciju, kas balstīta uz 1. attēla parādītā e-pakalpojuma algoritma grafa.

Lai uzskatāmi parādītu daudzkriteriālu optimizāciju un Pareto optimumu kopu, izvēlēsimies divus kritērijus pēc kuriem optimizēt:

- ātrdarbība;
- atkārtota izmantojamība.

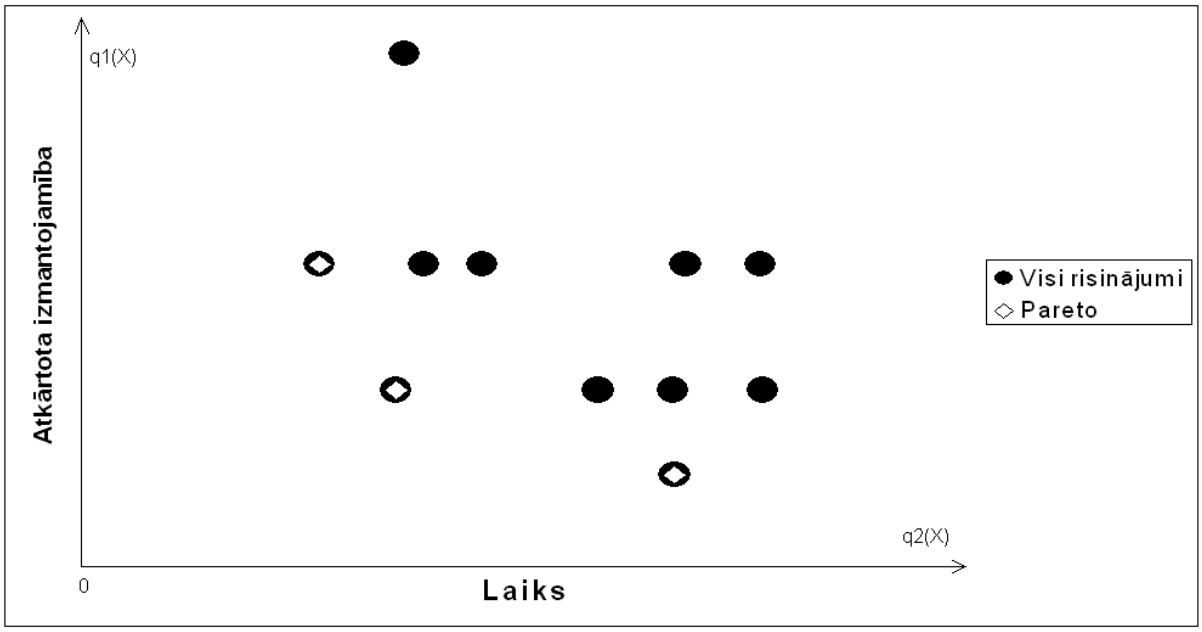
Ieviesīsim sekojošas algoritma grafa virsotņu pazīmes *I*. *I*(1)=*I*(2)=*I*(3)=*I*(5)=1, *I*(4)=2, *I*(6)=3.

Izrēķināsim visas iespējamās kombinācijas, izmantojot rekursīvu algoritmu. Šādā gadījumā kopējais kopas $X = \{G'\}$ apjoms ir 15 atvasinātie grafi, kas attēloti 1. tabulā, no kuriem atrast grafus, kuri ietilpst pareto kopā $P = \{G'^*\}$.

Tabula 1. Visi iespējamie Web servisu grafu risinājumi.

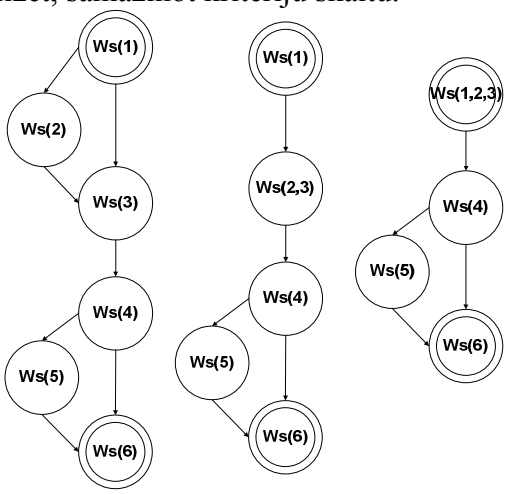
Nr.	*Grafs*	*Ātrdarbība*	*Atkārtota izmantojamība*	*Pareto*
1.	G'_1	1.6681	0.83	*
2.	G'_2	1.4545	1.6	
3.	G'_3	0.6686	2.75	*
4.	G'_4	0.9093	4.6	
5.	G'_5	1.6983	2.75	
6.	G'_6	1.6986	2.75	
7.	G'_7	1.6653	1.6	
8.	G'_8	1.9091	2.75	
9.	G'_9	1.9092	2.75	
10.	G'_{10}	1.9138	1.6	
11.	G'_{11}	0.9633	2.75	
12.	G'_{12}	0.8854	1.6	*
13.	G'_{13}	1.1292	2.75	
14.	G'_{14}	1.9139	1.6	
15.	G'_{15}	1.9140	1.6	

Dotajam piemēram Pareto kopa sastāv no trīs risinājumiem.. Pareto kopa uzskatāmi ir parādīta 2. attēlā. Grafiski pareto risinājumi atrodas vistuvāk koordinātu sākumpunktam.



2. attēls. Pareto kopas izkārtojums attiecībā pret visiem risinājumiem.

Pareto kopa atbilstoši šiem kritērijiem sastāv no 3. attēlā parādītiem risinājumiem. Ja izvēlēto risinājumu skaits ir mazs, kā dotajā piemērā, tad konkrēta risinājuma izvēli var veikt e-pakalpojuma projektētāji, balstoties uz savu pieredzi, vai arī var uzstādīt prioritātes, ko var realizēt, samazinot kritēriju skaitu.



3. attēls. Optimizēti Web servisu grafi.

6. Kopsavilkums

E-pakalpojuma izveide ir pietiekoši sarežģīts uzdevums. Viena no problēmām ir e-pakalpojuma projektēšana – pakalpojuma sadalījums pa Web servisiem. Konkrētā risinājuma izvēle ietekmē vairākus būtiskus e-pakalpojuma izpildes novērtējumus (izstrādes izmaksas, ātrdarbība, ekspluatācijas sarežģītība, atkārtota izmantojamība un integritāte). Lai izvēlētos kādu konkrētu risinājumu tiek piedāvāts sākotnēji izveidot e-pakalpojuma algoritma grafu, kuru pārveidot par Web servisu grafu. Web servisu grafu novērtēšanai ir piedāvātas formulas (2), (3), (4), (5), (6). Lai varētu veikt izvēli starp visiem iespējamajiem variantiem, tiek piedāvāts izmantot daudzkriteriālu optimizāciju – Pareto optimuma meklēšana, kā rezultātā tiek iegūta Pareto kopa. Grafi, kas ietilpst Pareto kopā, ir iespējamie Web servisu grafa

risinājumi. Web servisu grafus, kas ietilpst atrastajā Pareto kopā, var detalizēti projektēt, implementēt un darbināt e-pakalpojumu izpildes vidē.

Rakstā minēto metodiku var pielietot visa veida e-pakalpojumiem, kā arī līdzīgās situācijās.

7. Literatūra

1. Mike P. Papazoglou. Service – Oriented Computing: Concepts, Characteristics and Directions. Keynote for the 4th International Conference on Web Information Systems Engineering, December 10-12, 2003. – p.3-12.

2. Mike P. Papazoglou, Jian Yang. Design Methodology for Web Services and Business Processes. Proceedings of the Third International Workshop on Technologies for E-Services, 2002 – p. 54-64.

3. E. Žeiris, M. Ziema. E-pakalpojumu izveides problēmas. Rīgas Tehniskās universitātes zinātniskie raksti. Datorzinātne. 5. sērija 19. sējums. Rīga 2004.g. – lpp. 48-53.

4. Jānis Dambīts. Modernā grafu teorija. Datorzinību centrs, 2002. g. – lpp. 155.

5. Radu Sion, Junichi Tatemura. Dynamic Stochastic Models for Workflow Response Optimization, 2005 IEEE International Conference on Web Services (Industry Track) IEEE ICWS 2005 – p. 8.

6. Cardoso, J., A. Sheth, and J. Miller. Workflow Quality of Service. in International Conference on Enterprise Integration and Modeling Technology and International Enterprise Modeling Conference (ICEIMT/IEMC-02). 2002. Valencia, Spain – p. 13.

7. Tao Yu, Kwei-Jay Lin. Service Selection Algorithms for Composing Complex Services with Multiple QoS Constraints. International Conference on Service-Oriented Computing 2005 – p. 130-143.

8. М. Г. Зиема, Л. А. Растригин. Парето-зависимые множества при решении многокритериальных задач на графе и их применение для синтеза специализированных вычислительных систем. Архитектура и проектирование вычислительных систем. Проблемы адаптации и моделирования. Рига 1986 – с. 128-137.

Edzus Zeiris, M. Sc. Comp.
Riga Technical University
Faculty of Computer Science and Information Technology,
Institute of Computer Control, Automation and Computer Engineering
Address: Meza 1/3, 3rd floor. LV-1048, Riga, Latvia
E-Mail: edzus@zzdats.lv

Maris Ziema, Dr. Sc. Ing.
Riga Technical University
Faculty of Computer Science and Information Technology,
Institute of Computer Control, Automation and Computer Engineering
Address: Meza 1/3, 3rd floor. LV-1048, Riga, Latvia
E-Mail: maris@zzdats.lv

Rakstā ir aprakstīta metodika e-pakalpojumu arhitektūras izvēlei, kā daudzkriteriāls optimizācijas uzdevums. Sākotnēji ir aprakstīta problēmas nostādne – e-pakalpojuma arhitektūra kā Web servisu grafa izvēle no vairākiem iespējamiem risinājumiem. Lai risinātu problēmu, tiek definēts algoritma grafs, kā matemātiskais modelis. Ir aprakstīta pāreja no algoritma grafa uz Web servisu grafu. Definēti pieci Web servisu grafa novērtēšanas kritēriji (izstrādes izmaksas, ātrdarbība, ekspluatācijas sarežģītība, atkārtota izmantojamība un integritāte). Konkrētās arhitektūras risinājums ir kā daudzkriteriāls optimizācijas uzdevums, kas reducējas uz Pareto kopas atrašanu. Izvēlētā e-pakalpojuma arhitektūra ir Web servisu grafs, kas ietilpst atrastajā pareto kopā. Dotā pieeja ir parādīta arī uz piemēra pamata un rezultāti ir atspoguļoti rakstā.

Zeiris E., Ziema M. Criterions for Architecture Estimation and Architecture Selection of E-Services System
Article gives methodical material for e-service architecture selection as multicriterial optimization task. Initially problem position is described – e-service architecture as Web service graph selection of multiple available solutions. To solve this problem, algorithm graph is defined as a mathematical model for transformations. Reduction from the algorithm graph to the Web service graph is described. Five Web service graph estimation criterions are defined (Costs of Production, Time of Execution, Exploitation Costs, Reusability and Integrity). Specific architecture solution is multicriterial optimization task that is reduced to detecting the Pareto set. Selected E-service architecture is a Web service graph that belongs to detected Pareto set. The given approach is demonstrated on example base and results are displayed in article.

Жейрис Э., Зиема М. Критерии для оценки архитектуры системы электронных услуг и выбор архитектуры.
В статье описана методика выбора архитектуры е-услуг как задача многокритериальной оптимизации.Первоначально рассмотрена суть проблемы – выбор графа веб-сервисов для архитектуры из множества возможных решений. Для решения проблемы задается граф алгоритма как математическая модель. Далее описан переход от графа алгоритма к графу веб-сервисов. Определяются пять оценочных критериев для графа веб-сервисов (расходы по разработке, быстродействие, сложность эксплуатации, повторное использование и целостность). Выбор конкретной архитектуры – это задача многокритериальной оптимизации, сводящаяся к нахождению множества Парето. Выбранная архитектура е-услуг – это граф веб-сервисов, принадлежащий к найденному множеству. Данный подход также отображен в примере.