ISSN 1407-7493 COMPUTER SCIENCE 2008-7493
DATORZINĀTNE

TECHNOLOGIES OF COMPUTER CONTROL

DATORVADĪBAS TEHNOLOĢIJAS

DESIGN OF WEB SERVICES USING GRAF TRANSFORMATION AND OPTIMIZATION

GRAFU TRANSFORMĀCIJU UN OPTIMIZĀCIJAS IZMANTOŠANA WEB SERVISU PROJEKTĒŠANĀ

Edzus Zeiris, Mg. 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

Atslēgas vārdi: WEB servisu projektēšana, datoru sistēmu arhitektūra, grafu segmentēšana, e-pakalpojums, e-pakalpojumu sistēmu arhitektūra, SOA, arhitektūras izvēle

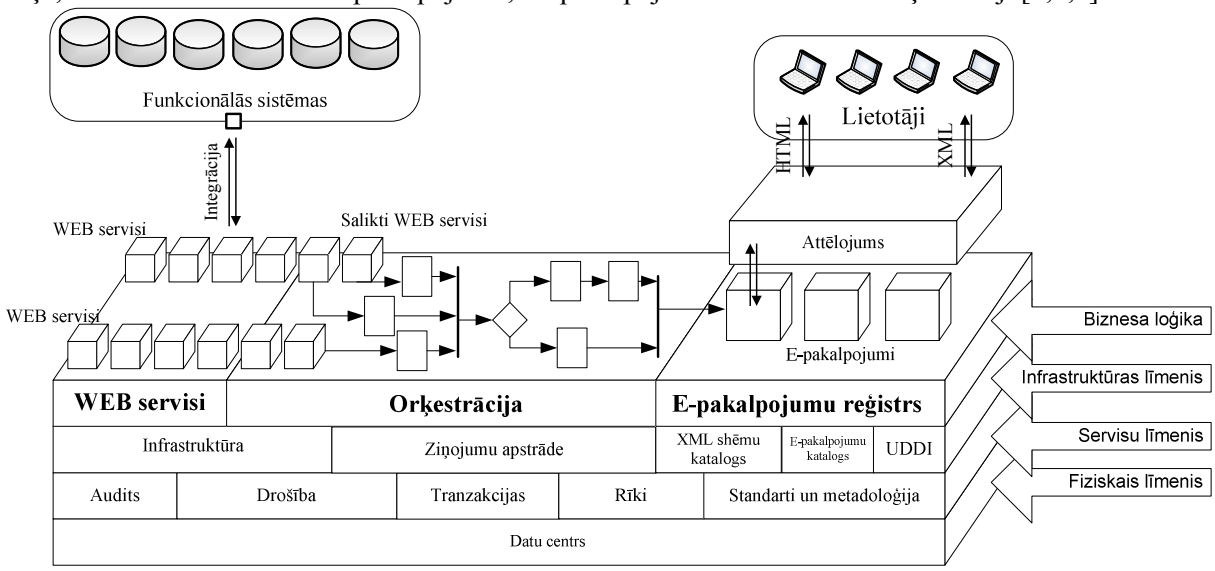
Ievads

Veidojot datoru sistēmu programmatūru, nepieciešams sākotnēji to noprojektēt. Datoru sistēmai ir nepieciešams definēt tās arhitektūru, kas apmierina visas datoru sistēmai izvirzītās prasības, kā arī projektēt sistēmas sastāvdaļas tā, lai tiktu izveidota visa nepieciešamā funkcionalitāte. Datoru sistēmu var izveidot dažādos veidos, kas veidos nepieciešamo funkcionalitāti un sasniegs izvirzītās prasības, bet dažādi risinājumu kvalitatīvais un kvantitatīvais novērtējums būs atšķirīgs, tāpēc nepieciešams pēc iespējas ātrāk noteikt veidojamās datoru sistēmas arhitektūras kvalitāti, lai izvairītos no iespējamās sistēmas pārstrādes. Tas nozīmē, ka ir nepieciešams izveidot veidojamās datoru sistēmas modeli un veikt novērtējumu pirms sistēmas izstrādes, lai pārliecinātos par veidojamās sistēmas kvalitatīvajiem rādītājiem. Sistēmas projektēšanas problēma ir ļoti būtiska e-pakalpojumu izstrādē, jo e-pakalpojumos bieži vien ir nepieciešams integrēt esošu datoru sistēmu komponentes, un šāda integrācija nedrīkst pazemināt jaunveidojamās sistēmas kvalitāti.
Šajā rakstā galveno uzmanību pievērsīsim WEB servisu projektēšanas problēmām, kas ir servisu orientētas arhitektūras (SOA) sistēmas komponenti. WEB servisi ir SOA būvētas arhitektūras pamatelementi, kuri nosaka sistēmas arhitektūru. Datoru sistēmas arhitektūra ir sistēmas struktūra vai struktūras, kas sevī ietver sistēmas komponentus, šo komponentu ārēji redzamās īpašības un to savstarpējās saites[1]. Kā praktiskie piemēri tiks izmantoti e-pakalpojumi, kuri tiek veidoti SOA un to komponenti ir WEB servisi. WEB serviss ir sistēmas komponente, kuru iespējams izsaukt un saņemt

rezultātu, izmantojot Simple Object Access Protocol (SOAP) protokolu un kuras saskarne tiek aprakstīta ar Web Service Description Language (WSDL) valodu.
Lai projektētu WEB servisus, kurus izmanto e-pakalpojumos, ir nepieciešams aplūkot un modelēt visus iespējamos e-pakalpojuma izveides variantus, lai pārliecinātos par izvēlētā risinājumu efektivitāti un izslēgtu iespēju, ka tiek izvēlēts risinājums, par kuru eksistē arī labāki risinājumi. Lai risinātu šo problēmu WEB servisu projektēšanu var novest uz grafu transformācijas uzdevumu, kas tālāk tiek aplūkots šajā rakstā[2,3].

E-pakalpojumu sistēmas arhitektūra

Aplūkosim e-pakalpojumu sistēmas kopējo arhitektūru. Arhitektūras loģiskajā shēmā (1. att.) ir parādīts, kā pakalpojumā iesaistītās sistēmas un moduļi tiek saistīti vienotā e-pakalpojumā. E-pakalpojuma sistēmas arhitektūrā tiek izdalīti vairāki loģiskie līmeņi, kas nodrošina kopējo sistēmas darbību. Fiziskajā līmenī atrodas datu centrs. Servisu līmenī atrodas rīki, kas nodrošina zemāka līmeņa servisus e-pakalpojumu darbības nodrošināšanai. Infrastruktūras līmenī atrodas rīki darbam ar e-pakalpojumiem. Pēdējā līmenī atrodas e-pakalpojuma biznesa loģika, kas ir katra e-pakalpojumu mainīgā daļa, un tieši šīs daļas izveide būtiski ietekmē katra konkrēta e-pakalpojuma darbību. Šajā rakstā ar e-pakalpojumu sistēmas arhitektūru sapratīsim kopējās e-pakalpojumu sistēmas mainīgo daļu, kas raksturo konkrētu e-pakalpojumu, t.i. pakalpojuma WEB servisu orķestrāciju[4,5,6].



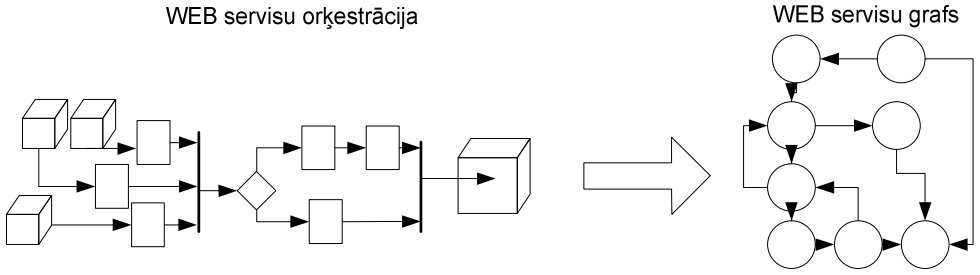
1.att. E-pakalpojumu sistēmas arhitektūra
Fig.1. E-services system architecture

Algoritma grafs

Lai varētu izveidot sistēmas iespējamo arhitektūras risinājumu, sākotnēji ir nepieciešams izveidot veidojamās sistēmas algoritmu. E-pakalpojumu algoritmu var definēt, kā izpildāmu darbību secību, ko ir iespējams ierakstīt blokshēmas veidā, vai arī kā orientētu grafu. Definēsim orientētu e-pakalpojuma algoritma grafu, kur grafa virsotnes pēc savas būtības ir e-pakalpojuma algoritma izpildāmas darbības, un loki grafā nozīmē to, ka e-pakalpojuma algoritma izpildē pēc darbības vienas virsotnes ietvertās darbības seko nākamā virsotnē ietvertā darbība. E-pakalpojuma algoritma grafs ir jāveido tā, lai tas atbilstu SOA principiem. Katra algoritma grafa virsotne ir veidota tā, lai virsotnē ietvertā darbība ir maksimāli neatkarīga no citu virsotņu darbībām, t.i. algoritma grafa virsotnē ietvertā darbība nav atkarīga no kopējā algoritma grafa stāvokļa. Algoritma grafs nesatur stāvokļa informāciju. Algoritma grafs satur vienu sākuma virsotni un vismaz vienu beigu virsotni[6,7].

WEB servisu grafs

Kā jau iepriekš tika minēts, tad par e-pakalpojumu sistēmas arhitektūru var uzskatīt savstarpēji saistītus WEB servisus starp kuriem tiek nodoti dati. Šī iemesla dēļ ir iespējams definēt WEB servisu grafu, kā orientētu grafu, kas apraksta e-pakalpojumu SOA sistēmas arhitektūru. Grafa virsotnes ir WEB servisi un loks grafā nozīmē to, ka WEB servisi ir savstarpēji saistīti, pie kam loka virziens norāda, ka pēc grafa virsotnē ievietotā WEB servisa izpildes seko nākamā virsotnē ievietotā WEB servisa izpilde. Loki WEB servisu grafā norāda informācijas plūsmu. SOA veidotas datoru sistēmas arhitektūras pieraksts WEB servisu grafa veidā parādīts 2.att.

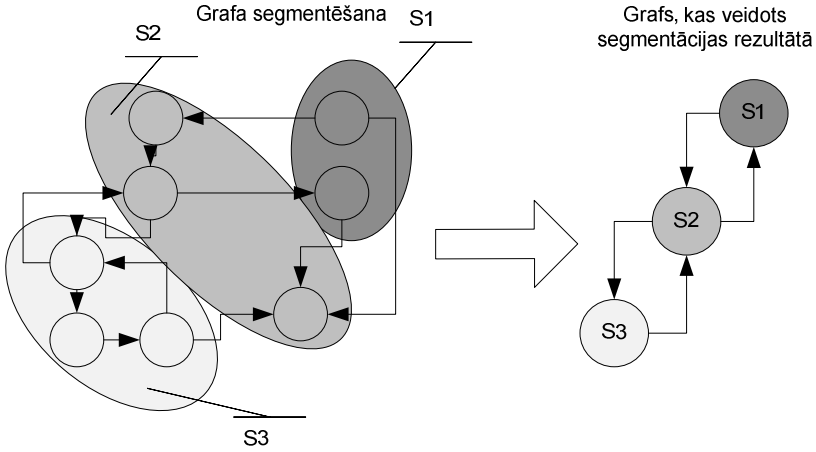


2.att. SOA datoru sistēmas pieraksts WEB servisu grafā
Fig.2. SOA software display as graph of WEB services

Ņemot vērā, ka e-pakalpojuma algoritms apraksta e-pakalpojuma darbību, tad uz WEB servisu grafu balstītā sistēmas projektēšanā par sākotnējo e-pakalpojuma arhitektūras WEB servisu grafu var pieņemt algoritma grafu, kuru tālāk ir iespējams transformēt, lai varētu modelēt iespējamo e-pakalpojuma sistēmas arhitektūras variantus[6,7].

WEB servisu grafa transformācijas

Lai iegūtu maksimāli pieņemamu e-pakalpojuma sistēmas arhitektūru, ir nepieciešams aplūkot visus iespējamos variantus. Par cik e-pakalpojumu sistēmas arhitektūru ir iespējams pierakstīt kā WEB servisu grafu, tad aplūkosim grafu segmentēšanu, kā projektēšanas metodi. Ar grafa segmentēšanu turpmāk sapratīsim grafa virsotņu apvienojošu kopu vai kopas. Lai varētu atrast visus iespējamos WEB servisu grafa segmentus, tādā veidā mainot WEB servisu grafa parametrus un ietekmējot sistēmas arhitektūras kvalitātes rādītājus, ir nepieciešams atrast visas iespējamās grafa virsotņu kopu kombinācijas. Ja tiek aplūkoti visi iespējamie WEB servisu grafa segmenti, tad arī ir iespējams izvēlēties maksimāli pieņemamu arhitektūru starp atrastajiem segmentiem. Grafa segmentācija trīs segmentos S1, S2 un S3 parādīta 3.att.



3.att. Grafa segmentācijas piemērs
Fig.3. Example of graph segmentation

Segmentējot WEB servisu grafa virsotnes, ja vienā segmentā tiek iekļauta vairāk kā viena virsotne, tad tas nozīmē, ka WEB servisu funkcionalitāte tiek apvienota. Lai veiktu visu iespējamo kombināciju WEB servisu grafa segmentāciju un iegūtu WEB servisu grafus, kas veidoti segmentācijas rezultātā, par pirmo WEB servisu grafu tiek pieņemts sākotnējais algoritma grafs. Segmentācijai tiek izmantots rekursīvs algoritms. Algoritms ņem katras divas padotā grafa virsotnes un izveido jaunu WEB servisu grafu, kur izvēlētās virsotnes ir apvienotas. Visi izveidotie WEB servisu grafi tiek glabāti binārā kokā, lai varētu maksimāli ātri atrast vai izveidotais grafs jau neeksistē. Tiek pārbaudīts vai jaunizveidotais grafs jau neeksistē binārajā kokā. Ja tāds grafs ir radīts pirmo reizi, tad tas tiek pievienots binārajam kokam un tiek rekursīvi izsaukta segmentācijas procedūra, padodot jaunizveidoto grafu. WEB servisu grafa segmentācijas algoritms, kas pierakstīts pseido programmēšanas valodā, parādīts 4.att.

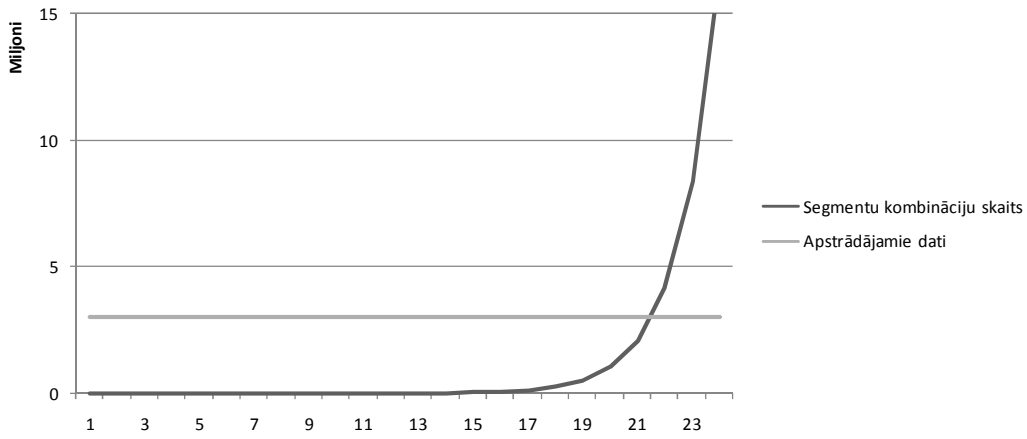
```
G = Graph to segment
Create empty binary tree B
Call procedure Segment (G)

Procedure Segment (Graph G)
Begin
   For each vertex V1 in G vertexes loop
          For each vertex V2 in G vertexes and V2 index > V1 index loop
                 If V1 and V2 can merge then
                        Create new graph G1 where V1 and V2 are merged
                        If B not contains G1 then
                               Add G1 to B
                               Call procedure Segment (G1)
                        End If
                 End If
          End Loop
   End Loop
End
```

4.att. WEB servisu grafa segmentēšanas algoritms
Fig.4. WEB services graph segmentation algorithm

Lai iegūtu visas iespējamās WEB servisu grafa segmentu kombinācijas, WEB servisu grafs ir jāsegmentē vairākos līmeņos. Tas nozīmē, ka vispirms ir jāizrēķina dotā WEB servisu grafa visas iespējamās segmentu kombinācijas, kas veido jaunus WEB servisu grafus, un pēc tam iegūtos WEB servisu grafus var segmentēt atkārtoti, līdz brīdim kamēr vairs nav atrodamas jaunas kombinācijas. Šādā veidā var iegūt visas iespējamās grafa segmenta kombinācijas. Vairāk par vienu līmeni ir iespējams segmentēt algoritma grafam, kura virsotņu skaits pārsniedz 3 virsotnes. Grafa pirmā līmeņa segmentu skaitu var aprēķināt pēc formulas

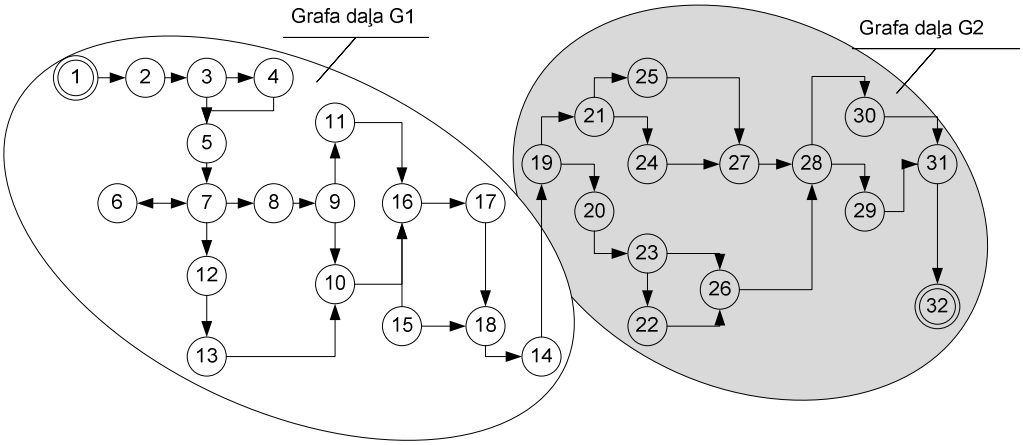
$$r = 2^n - C_n^1 \quad (1)$$

,kur *n* ir grafa virsotņu skaits.
Pirmā līmeņa algoritma grafa segmentu skaits atkarībā no virsotņu skaita parādīts 5. att.



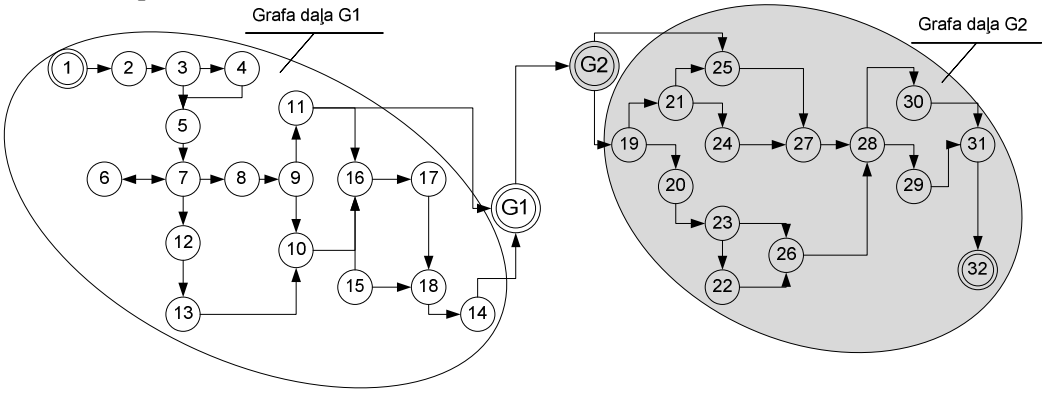
5.att. Pirmā līmeņa grafa segmentāciju skaits atkarībā no virsotņu skaita
Fig.5. First level graph segmentation count dependent of vertex count

Kā redzams 5. att., tad virsotņu skaitam pārsniedzot 20, apstrādājamo segmentu skaits pieaug ļoti strauji un sasniedz skaitļus, kurus reālā laikā apstrādāt ir neiespējami. Visu segmentu kombināciju atrašana ir NP-pilna problēma. Tas nozīmē, ka ir nepieciešams veikt pilnu pārlasi. Ņemot vērā, ka šis ir tikai pirmā līmeņa grafu segmentu skaits, e-pakalpojumu sistēmas arhitektūras alternatīvo risinājumu meklēšanu, kas balstīta uz visu alternatīvo risinājumu aplūkošanu, efektīvi var darboties tikai tad, ja algoritma grafa virsotņu skaits nepārsniedz 20 vienas sistēmas virsotnes.
Visu grafa segmentu skaitu var iegūt tikai rekursīvi segmentējot algoritma grafu, katrā rekursijas solī apvienojot ik pa katrām divām algoritma grafa virsotnēm un pārbaudot vai iegūtais grafs jau nav atrasts. Ja iegūtais grafs jau ir atrasts kāda citā rekursijas solī, vai ir segmentētas visas virsotnes , tad rekursiju beidz.
Kā jau iepriekš tika minēts, tad segmentēšanas algoritmā ir iespējams izmantoto tikai pilnu pārlasi. Lai paātrinātu pilnās pārlases algoritmu un pēc iespējas paātrinātu tā darbību, atrasto segmentēto grafu glabāšanai tiek izmantots binārais koks, kurā meklēšanas algoritma ātrums, lai noteiktu vai grafs jau ir atrasts, ir logaritmisks. Apzīmēsim algoritmiskās sarežģītības funkciju ar *O*, tādā gadījumā algoritmiskā sarežģītība atrasto risinājumu pārbaudei ir *O(log n)*, kur *n* ir atrasto WEB servisu skaits, bet ņemot vērā to, ka kopējais algoritms ir rekursīvs, un ir nepieciešams aplūkot visus iespējamos variantus, tad kopējā algoritma sarežģītība ir $O(2^n)$, kur *n* ir sākotnēji segmentējamā WEB servisu grafa virsotņu skaits. Tāpēc ir nepieciešams risinājums liela apjoma WEB servisu grafu projektēšanai, izmantojot grafu transformācijas.
Praksē e-pakalpojums, kura algoritma grafs saturētu vairāk par 20 virsotnēm bez ierobežojumiem ir reti sastopams, jo parasti vienā e-pakalpojumā tiek iesaistītas dažādas sistēmas, kas rada ierobežojumus, tādā veidā samazinot iespējamo segmentu skaitu. Tas nozīmē, ka WEB servisu grafam, kas tiek balstīts uz algoritmu grafu arī tiek uzlikti ierobežojumi, kas neļauj segmentēt kopā vienā segmentā virsotnes, kam ir uzlikti šādi ierobežojumi. Atkarībā no uzlikto ierobežojumu skaita ir iespējams apstrādāt arī lielāka apjoma WEB servisu grafus, un aprēķināt tiem visas iespējamās segmentu kombinācijas.
Ja ir nepieciešams apstrādāt lielāku e-pakalpojuma algoritma grafu nekā 22 virsotnes, un izveidot visus iespējamos WEB servisu grafus to segmentējot, tad šo uzdevumu ir nepieciešams sadalīt vairākos mazākos uzdevumos. Algoritma grafā ir jāatrod tādas algoritma grafa daļas G1 un G2, kas ir loģiski atdalāmas un doto algoritma grafu ir jāsadala divos vai vairākos mazākos algoritma grafos, kurus jau var pārveidot katru atsevišķi par WEB servisu grafu, un pielietot pilnu segmentāciju. Grafu sadalīšana ir attiecināma uz skaitļošanas uzdevumiem, kur nepieciešams grafu sadalīt divās (vai vairākās) lielās daļās, minimizējot šķautņu skaitu, kas šķērso šķēlumu. Lai atrisinātu šo problēmu iespējams izmantot standarta algoritmus no grafu teorijas, piemēram uz plūsmām balstītie algoritmi[8]
Liela algoritma grafa sadalīšana mazākos parādīta 6. att., kur katru no norādītajiem segmentiem ir iespējams risināt kā atsevišķu e-pakalpojumu sistēmas arhitektūras izstrādes uzdevumu.



6.att. Liela e-pakalpojuma algoritma grafa sākotnējās daļas
Fig.6. Initial partitions of large e-service algorithm graph

Ja grafa sadalīšanas uzdevuma rezultātā nav iespējams atrast tādas grafa daļas, kur tās savā starpā saista tikai viena šķautne, tad algoritma grafā atrod grafa daļas ar minimālu šķautņu sasaisti, un algoritma grafu, pārveidojot uz WEB servisu grafu, ievieš papildus divas virsotnes. Vienu virsotni grafa daļai G1 kā beigu stāvokli un vienu virsotni grafa daļai G2 kā sākuma virsotni. WEB servisu grafā realizācijā tā ir kā maršrutēšana e-pakalpojumu sistēmas orķestrācijā (skat. 1.att.). Papildus virsotņu ieviešana parādīta 7. att.[9]



7.att. Papildus virsotņu ieviešana WEB servisu grafa daļās
Fig.7. Additional vertexes in WEB service graph partitions

Daudzkriteriālas optimizācijas pielietošana

Lai izvēlētos e-pakalpojumu sistēmas arhitektūru, kas apmierina visas tai izvirzītās prasības, ir iespējams pielietot daudzkriteriālu optimizāciju, jo parasti izvirzītās prasības ir savstarpēji pretrunīgas. Projektējot datoru sistēmu, bieži vien tiek izvēlēta arhitektūra, kas ir pieņemama, bet nav maksimāli labākā (tiek izvēlēt lokāls maksimums). Tāpēc ir nepieciešams aplūkot visus iespējamos arhitektūras izveides variantus un izvēlēties veidojamo sistēmas arhitektūru starp tiem. Lai izvēlētos e-pakalpojumu sistēmas arhitektūru, vispirms e-pakalpojuma algoritms tiek aprakstīts orientēta grafa veidā. Iegūtais algoritma grafs tiek pieņemts par WEB servisu grafu, kuru segmentējot visās iespējamajās virsotņu kombinācijās, iegūst WEB servisu grafus no kuriem izvēlēties iespējamo e-pakalpojuma sistēmas arhitektūru. Lai to izdarītu, ir nepieciešams novērtēt visiem iegūtajiem WEB servisu grafiem izvirzītos e-pakalpojuma sistēmas kritērijus (ātrdarbība, atkārtota izmantojamība, uzturamība, izstrādes izmaksas, mērogojamība, drošība, u.c.). Izmantojot iegūto WEB servisu grafu kopu un to novērtējumus, iespējams pielietot daudzkriteriālas optimizācijas metodes, lai izvēlētos no visu aprēķināto WEB servisu kopas konkrētu risinājumu[10,11,12].

Kopsavilkums

Datoru sistēmas arhitektūras projektēšana ir sarežģīts uzdevums. Projektējot SOA veidotas sistēmas, ir jārisina WEB servisu projektēšanas uzdevums. Datorsistēmas uzbūvē izmantojamie WEB servisi būtiski ietekmē izveidotās sistēmas kvalitātes rādītājus, tāpēc ir ļoti svarīgi izvēlēties tādus WEB servisus, kas rezultātā veido maksimāli pieņemamu sistēmas arhitektūru. Lai varētu izvēlēties pieņemamu datoru sistēmas arhitektūru, ir nepieciešams jau pirms sistēmas izveides modelēt tās darbību un to novērtēt. Šim nolūkam ir iespējams pielietot grafus un to segmentēšanu. Segmentējot WEB servisu grafu , kas apraksta veidojamās sistēmas arhitektūru, iegūt visus iespējamos veidojamās sistēmas arhitektūras variantus. Pēc visu WEB servisu grafa segmentu atrašanas ir iespējams izvēlēties pieņemamo arhitektūru, izmantojot optimizācijas metodes. Ņemot vērā to, ka parasti datorsistēmai izvirzītās kvalitātes prasības ir vairākas un pretrunīgs, tad arhitektūras izvēli no atrastajiem WEB servisu grafiem var balstīt uz daudzkriteriālu optimizāciju.

Literatūra

1. Bass L., Clements P., Kazman R. Software Architecture in Practice – Addison-Wesley, 2003 – 528lpp.
2. Bosch J. Design and Use of Software Architectures – Addison-Wesley, 2000 – 354lpp.
3. Hong Zhu Software Design Methodology – Elsevier Butterworth-Heinemann, 2005 – 340lpp.
4. Īpašo uzdevumu ministrija e-pārvaldes lietās sekretariāts. Integrētas valsts informācijas sistēmas koncepcija – Rīga, 2005 – 34lpp.
5. Microsoft Corporation. Connected Government Framework. Architecture and Design Blueprint – Microsoft Corporation, 2003 – 178lpp.
6. Zeiris E., Ziema M. E-Service Architecture Selection Based on Multi-criteria Optimization // 8th International Conference, PROFES 2007, Riga, Latvia, July 2007 Proceedings – Springer-Verlag Berlin Heidelberg, 2007 – 345-357lpp.
7. Žeiris E., Ziema M. Elektronisko pakalpojumu sistēmu arhitektūras izvēle // Rīgas Tehniskās universitātes zinātniskie raksti. 5. sēr., Datorzinātne. Informācijas tehnoloģija un vadības zinātne. – 32. sēj. (2007), 99 – 107 lpp.
8. Arora S., Rao S., Vazirani U. Geometry, Flows, and Graph-Partitioning Algorithms // Communications of The ACM – Nr. 51 (2008), 96-105lpp.
9. Gross J. L., Yellen J. Hanbook of Graph Theory – CRC Press, 2004 – 1167lpp.
10. Žeiris E., Ziema M. Elektronisko pakalpojumu sistēmu arhitektūras novērtēšanas kritēriji un arhitektūrās izvēle // Rīgas Tehniskās universitātes zinātniskie raksti. 5. sēr., Datorzinātne. Informācijas tehnoloģija un vadības zinātne. – 27. sēj.(2006), 91 – 98 lpp.
11. Miettinen K. M. Nonlinear Multiobjective Optimization – Kluwer Academic Publishers, 1998 – 298lpp.
12. Parlos P. M. Multi-Criteria Decision Making Methods: A Comparative Study – Kluwer Academic Publishers, 2000 – 288lpp.

Žeiris E., Ziema M. Grafu transformāciju un optimizācijas izmantošana WEB servisu projektēšanā

Sistēmu projektēšana ir svarīgs uzdevums, kas jārisina visu datoru sistēmu izveides procesā. Datoru sistēmas veiksmīga darbināšana ir atkarīga no izvēlētās arhitektūras. Parasti datoru sistēmas arhitektūrai tiek izvirzīti vairāki kritēriji, kas jāizpilda, lai sasniegtu vēlamo rezultātu. Lai varētu vēl pirms pašas sistēmas izveides pieņemt lēmumu par arhitektūras izvēli, ir nepieciešams modelēt to. E-pakalpojumu sistēmu arhitektūra sastāv no vairākām komponentēm. Ļoti būtiska e-pakalpojumu sistēmas sastāvdaļa ir katra konkrēta pakalpojuma orķestrācija, kas sastāv no WEB servisiem un to savstarpējām saitēm. Lai varētu veikt šādas e-pakalpojumu sistēmas projektēšanu, vispirms ir nepieciešams aprakstīt e-pakalpojuma algoritmu orientēta grafa veidā, kur grafa virsotnes pēc savas būtības ir e-pakalpojuma algoritma izpildāmas darbības, un loki grafā nozīmē to, ka e-pakalpojuma algoritma izpildē pēc darbības vienas virsotnes ietvertās darbības seko nākamā virsotnē ietvertā darbība. Sākotnēji pieņemot e-pakalpojuma algoritma grafu par WEB servisu grafu var veikt projektēšanas tālākās darbības. WEB servisu grafa virsotnes ir WEB servisi un loks grafā nozīmē to, ka WEB servisi ir savstarpēji saistīti, pie kam loka virziens norāda, ka pēc grafa virsotnē ievietotā WEB servisa izpildes seko nākamā virsotnē ievietotā WEB servisa izpilde. Segmentējot WEB servisu grafa virsotnes savā starpā visās iespējamajās kombinācijās, tiek iegūti visi iespējamie WEB servisu grafi, kas apraksta e-pakalpojumu sistēmas arhitektūru. Visu iespējamo WEB servisu grafa segmentu atrašana notiek ar rekursīva algoritma palīdzību. Ja ir

jāapstrādā e-pakalpojums, kur algoritma grafa virsotņu skaits pārsniedz divdesmit virsotnes, vai nu ir nepieciešams veikt sākotnējo analīzi un izvirzīt ierobežojumus WEB servisu grafa virsotņu segmentēšanai, kas neļauj apvienot vienā segmentā noteiktas virsotnes, vai nu arī veikt algoritma grafa sadalīšanu daļās, kur katru algoritma grafa daļu pārveidot atsevišķi par WEB servisu grafu un veikt segmentu meklēšanu katrai daļai atsevišķi, risinot uzdevumu kā vairākus atsevišķus. Pēc visu iespējamo WEB servisu grafu segmentu atrašanas ir iespējams veikt arhitektūras izvēli starp atrastajiem WEB servisu grafiem, pielietojot daudzkriteriālu optimizāciju. Visu iespējamo variantu aplūkošana, izslēdz iespēju, ka tiek izvēlēts sistēmas arhitektūras lokālais maksimums.

Zeiris E., Ziema M. Design of WEB services using graf transformations and optimization

Design of system architecture is important problem that must be solved in each software development process. Success of software exploitation depends on selected architecture. Usually there is more than one requirement for software architecture that must be solved to reach the expected result. To make architecture selection before the software is developed, architecture must be modelled. E-service system architecture stands up from many components. Very important element of e-service system architecture is orchestration of particular e-service that contains WEB services and mutual relations. To design architecture such of e-service system, first full a e-service algorithm must be described as oriented graph where vertexes of the graph in point of fact are executable activities of e-service algorithm and the edges in graph means that in the execution of the algorithm graph after activity in the one of the vertexes follows activity in the next vertex. Initially assume that e-service algorithm graph is WEB service graph future design of e-service can be made by transformation of the WEB service graph. The vertexes in WEB service graph are WEB services and edges in WEB services graph means that WEB services are mutually related and direction of the edge shows that after WEB service execution in one vertex follows next WEB service execution in next vertex. By mutual segmentation of WEB service graph vertexes in all possible combinations all possible WEB service graphs are calculated that describes the architecture of e-service system. The all WEB service graph segments are calculated by recursive algorithm. If a large e-service algorithm graph must be processed with more than twenty vertexes the initial design and analysis must be performed to put restrictions on WEB service graph segmentation that restricts to join defined vertexes in one segment or e-service algorithm graph must be partitioned in several equal partitions with minimal cross partition edges and each algorithm graph partition can be transformed to WEB service graph and solved as a separated task of design. After detecting all possible WEB service graph vertexes combination segments, architecture selection can be performed by selecting e-service system architecture from set of segmented WEB service graphs. Selection of e-service architecture instance can be performed as multi-criteria optimisation task in evaluated set of WEB service graphs. Examination of all possible solutions of system architecture exclude probability that local maximum can be selected.

Жейрис Э., Зиема М. Проектирование WEB сервисов с применением трансформации и оптимизации графа

Проектирование системы – важная задача, решать которую необходимо в процессе создания любой компьютерной системы. Успешная работа системы зависит от выбранной архитектуры. Обычно к архитектуре выдвигается ряд требований, которые необходимо выполнить для достижения желаемого результата. Для того, чтобы принять решение о выборе архитектуры до начала разработки, систему необходимо смоделировать. Архитектура системы е-услуг состоит из нескольких компонентов. Существенной частью системы является оркестрация каждой услуги, состоящая из WEB сервисов и связей между ними. Для проектирования подобных услуг необходимо описать алгоритм работы е-услуги в виде ориентированного графа, где вершины графа являются выполняемыми действиями е-услуги, а дуги указывают последовательность выполнения действий алгоритма е-услуги. Приняв граф алгоритма е-услуги за граф WEB сервисов, можно переходить к следующим шагам проектирования. Вершины графа WEB сервисов являются WEB сервисами, а дуги графа определяют, каким образом WEB сервисы связаны между собой, причем направление дуги показывает, что за выполнением WEB сервиса в текущей вершине следует выполнение WEB сервиса в вершине, на которую указывает дуга. Сегментируя вершины графа WEB сервисов между собой во всех возможных комбинациях, получаем все возможные графы WEB сервисов, которые описывают архитектуру системы е-услуг. Все возможные сегменты определяются при помощи рекурсии. Если необходимо обработать граф алгоритма е-услуги, количество вершин которого превышает двадцать, возможны два решения. Первым решением может стать начальный анализ е-услуги и выдвижение ограничений на объединение вершин в графе WEB сервисов в один сегмент. Другим решением является разделение алгоритма на несколько частей, причем каждая часть будет рассматриваться как отдельный граф WEB сервисов. После определения всех возможных сегментов графа WEB сервисов появляется возможность выбрать архитектуру из всех найденных графов, используя многокритериальную оптимизацию. Перебор всех возможных вариантов исключает возможность выбора локального максимума для архитектуры системы