ISSN 1407-7493 COMPUTER SCIENCE 2008-7493
DATORZINĀTNE

TECHNOLOGIES OF COMPUTER CONTROL

DATORVADĪBAS TEHNOLOĢIJAS

SELECTION OF WEB SERVICES USING INDUSTRY STANDARDS UML AND XMI

INDUSTRIJAS STANDARTU UML UN XMI IZMANTOŠANA WEB SERVISU IZVĒLĒ

Peteris Stipravietis, Phd student
Riga Technical University
Faculty of Computer Science and Information Technology,
Institute of Computer Control, Automation and Computer Engineering
Meza 1/3, 3rd floor. LV-1048, Riga, Latvia
E-Mail: peteris@zzdats.lv

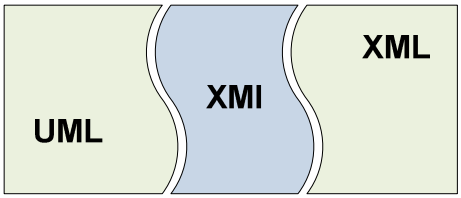
Maris Ziema, ass. prof, dr. sc. ing.
Riga Technical University
Faculty of Computer Science and0 Information Technology,
Institute of Computer Control, Automation and Computer Engineering
Meza 1/3, 3rd floor. LV-1048, Riga, Latvia
E-Mail: maris@zzdats.lv

Atslēgvārdi: XMI, UML, Web servisi, Web servisu identificēšana, metadati, VISS

Ievads

SOA (servisorientēta arhitektūra) ir programmatūras izstrādes arhitektūra, kas izmanto tīklā pieejamos komponentus. Tā ļauj izmantot atkalizmantojamus servisus, kuros ir implementēta atbilstoša biznesa funkcionalitāte. Šādus servisus pēc tam var izmantot dažādi klienti no citām sistēmām un biznesa procesiem. Pārsvarā gadījumu šie servisi ir uz XML bāzētie Web servisi, un daudzu sistēmu vai procesu izstrādē pienāk brīdis, kad jāizlemj, kādus Web servisus izmantot, lai sistēma vai process korekti veiktu savas funkcijas. Parasti sistēmas vai procesa izstrādātājs veic meklēšanu Web servisu reģistrā, uzdodot dažādus meklēšanas parametrus – standarta UDDI (nosaukums, kategorija, publicētājs) [1] vai paplašinātos metadatus – reitingu, ieejas/izejas parametrus, kas piekārtoti hierarhiskām datu struktūrām [2] u.c. Analizējot atrasto Web servisu kopu, ļoti noderīgi ir zināt visu iespējamo par izmantojamo Web servisu – šajā gadījumā informāciju par to, kā tieši serviss veic savu uzdevumu, kādus autentifikācijas un autorizācijas mehānismus tas izmanto, kā tiek veikta kļūdu apstrāde u.c. Ja Web servisa izstrādātājs šādu informāciju ir ievietojis tā metadatos, tad servisa izmantotājam tā tikai jāspēj nolasīt un jāveic novērtējums – vai šis serviss ir pietiekami derīgs/atbilstošs dotā uzdevuma veikšanai. Principā šis ir gadījums, kad Web serviss savos metadatos piedāvā savu projektējumu. Katrs izstrādātājs, aprakstot sava servisa darbību metadatos, var izveidot savas shēmas, taču šāda pieeja apgrūtinās servisa pieejamību – servisa izmantotājam jāsaprot šāds metadatu standarts. Ja biznesa procesa nodrošināšanai jāizmanto piecu atšķirīgu izstrādātāju piedāvātie servisi un tie katrs ir izmantojuši atšķirīgas pieejas darbības aprakstam, ievērojami palielinās laiks, kas nepieciešams, lai novērtētu izvēlētā servisa atbilstību procesa vajadzībām. Līdz ar to droši var

apgalvot, ka standartizētu risinājumu izmantošana atvieglotu novērtēšanu, tādējādi samazinot procesa izstrādes izmaksas un laiku. Standarti, kurus iespējams izmantot dotās problēmas risināšanā, ir UML (Unified Modeling Language) un XMI (XML Metadata Interchange). UML ir sistēmu un procesu modelēšanas valoda, kas nodrošina augstu abstrakcijas līmeni. UML diagrammas ļauj aprakstīt sistēmas vai procesa struktūru, uzvedību un mijiedarbību ar citām sistēmām vai procesiem [3]. XMI ir OMG (Object Management Group) standarts, kas paredzēts metadatu apmaiņai starp sistēmām un ir bāzēts uz XML, pašreizējā XMI versija ir 2.1 [4]. Liela daļa no UML modelēšanas rīkiem nodrošina iespēju UML diagrammas eksportēt uz XMI formātu, kas ir kā tilts starp UML un XML [5] – UML apraksta objektus, bet XML – datus (1. attēls).



1.att. XMI – tilts starp UML un XML
Fig.1. XMI bridges the gap between UML and XML

Fakts, ka XMI ir bāzēts uz XML, padara to par sevišķi pievilcīgu servisa izpildes aprakstīšanai metadatos, kuri arī ir bāzēti uz XML.

Web servisu metadatu papildināšana

Ar UML palīdzību iespējams aprakstīt Web servisa projektējumu, izmantojot aktivitāšu vai sekvenču diagrammas Web servisa struktūras un mijiedarbības attēlošanai. Kad atbilstošā diagramma ir izstrādāta, to iespējams transformēt uz XMI. Šo transformāciju nodrošina dažādi UML modelēšanas rīki – Altova UModel, Visual Paradigm, Rational Rose, ArgoUML un citi. No šāda rīka eksportēto XMI dokumentu Web servisa metadatos var iekļaut gan atsevišķā elementā, gan izmantot VISS (Valsts Informācijas Sistēmu Savietotājs) esošās metadatu struktūras. VISS paredzēts e-pakalpojumu izstrādātājiem un informācijas sistēmu integrētājiem. VISS sastāv no pakalpojumu, Web servisu un XML shēmu katalogiem – tos pārlūkojot un veicot tajos meklēšanu, izstrādātāji var izvēlēties sev nepieciešamos servisus vai shēmas. Autori ir piedalījušies vairāku VISS reģistrā publicēto Web servisu un shēmu izstrādē un iesaka izmantot esošo metadatu modeli – nav jāmaina esošā modeļa specifikācija, turklāt nav izšķirošas nozīmes, kur tiek glabāts XMI apraksts, jo tā saturs nemainās. Izmantojot elementu RELATION [6], kas paredzēts kā norāde uz saistītiem resursiem, iespējams uzturēt vairākas XMI apraksta versijas, tās uzglabājot kopējā shēmu katalogā, piemēram [7]. Papildus tam RELATION elementa izmantošana arī nodrošina vairāku norāžu uz XMI dokumentiem iekļaušanu metadatos – gan dažāda griezuma, gan atšķirīgas detalizācijas pakāpes diagrammas.
1. tabulā redzams piemērs, kā metadatu elementā RELATION iekļauts Web servisa metodes GetPersonDocuments apraksts XMI formātā, kas atrodams uzdotajā vietnē.

1.tabula
Table 1.

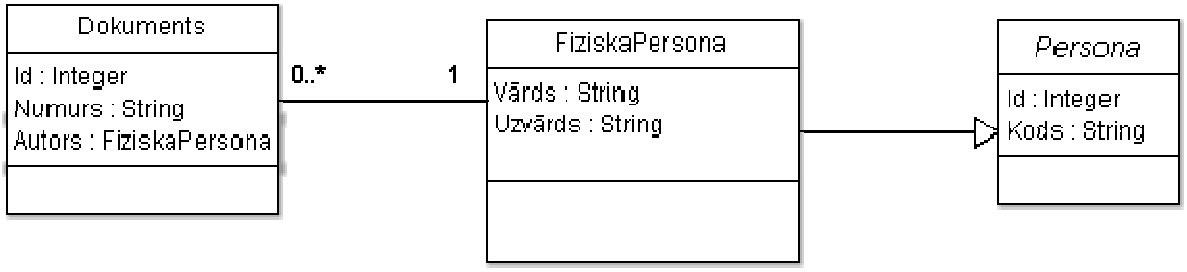
RELATION elementa izmantošanas piemērs
An example of the usage of RELATION element

Pieraksta veids	Piemērs
OO sintakse	Metadata.relation.providesDefinitonOf.Content=" http://www.foo.bar/GetDocumentCount.xmi";
XML sintakse	<meta name="relation.providesDefinitionOf" content="http://www.foo.bar/GetDocumentCount.xmi">

Protams, rodas jautājums, kāpēc vispār ir nepieciešams XMI – kāpēc metadatos nevar glabāt atsauces uz UML diagrammām? Kā jau iepriekš tika minēts, katrs izstrādātājs savus servisus var aprakstīt sev tīkamā veidā, šajā gadījumā katrs ar savu UML modelēšanas rīku. Lai aplūkotu diagrammas, uz darbstacijas jābūt atbilstošai programmatūrai, kas to nodrošinātu. Lai arī UML ir standarts, tas apraksta, kādām ir jāizskatās diagrammām, nevis kādā formātā tās jāsaglabā. XMI izmantošana atvieglo Web servisu izmantotāju darbu – jāizmanto tikai viena programmatūra, nevis visi iespējamie UML rīki. XMI izmantošana arī atvieglo dažādo izstrādātāju izveidoto diagrammu apmaiņu (kam XMI arī ir paredzēts). UML diagrammas metadatos var arī ietvert kā attēlus, taču papildus standartizācijai XMI dokumentā iespējama arī meklēšana – piemēram, pārbaudīt, vai Web servisa aprakstā ir iekļauta autentifikācija uzņēmuma aktīvajā direktorijā. No attēla vai UML diagrammas šādus datus ar automatizētiem rīkiem iegūt nav iespējams.

Web servisu izvēle

Lai sāktu lietot Web servisus savos risinājumos, vispirms no reģistra jāatlasa atbilstošie servisi. Izmantojot paplašinātos Web servisu metadatus, iespējama meklēšana pēc ievaddatu un izvaddatu parametru datu struktūrām, tādējādi samazinot atrodamo Web servisu kopu [2]. Lai veiktu meklēšanu, uzdodot šādas hierarhijas, izstrādātājam vispirms jāuzzina pieejamās struktūras – pieslēdzoties reģistram, jānolasa UML klašu diagramma. Arī šo diagrammu ieteicams publicēt kā XMI dokumentu – tad tajā ir iespējams meklēt, vai dotais reģistrs vispār piedāvā Web servisus, kuri darbojas, izmantojot uzdoto datu struktūru.



2.att. vienkāršas datu hierarhijas UML klašu diagramma
Fig.2. UML class diagram of simple data hierarchy

Piemēram, 2. attēlā ir redzama kādā Web servisu reģistrā izmantotā datu hierarhija kā UML klašu diagramma. Atbilstošā XMI apraksta fragments ir redzams 3. attēlā, saspiests tā, lai būtu redzamas klases.

```
<?xml version = '1.0' encoding = 'UTF-8' ?>
<XMI xmi.version = '1.2' xmlns:UML = 'org.omg.xmi.namespace.UML' timestamp = 'Mon Oct 06 22:18:20
  <XMI.header>    <XMI.documentation>
      <XMI.exporter>ArgoUML (using Netbeans XMI Writer version 1.0)</XMI.exporter>
      <XMI.exporterVersion>0.26(6) revised on $Date: 2007-05-12 08:08:08 +0200 (Sat, 12 May 2007)
    </XMI.documentation>
    <XMI.metamodel xmi.name="UML" xmi.version="1.4"/></XMI.header>
  <XMI.content>
    <UML:Model xmi.id = '-64--88-1-102-54f048e3:11cd38349ae:-8000:0000000000000EA5'
      name = 'untitledModel' isSpecification = 'false' isRoot = 'false' isLeaf = 'false'
      isAbstract = 'false'>
      <UML:Namespace.ownedElement>
        <UML:Class xmi.id = '-64--88-1-102-54f048e3:11cd38349ae:-8000:0000000000000EA6'
          name = 'Dokuments' visibility = 'public' isSpecification = 'false' isRoot = 'false'
          isLeaf = 'false' isAbstract = 'false' isActive = 'false'>...
        <UML:Class xmi.id = '-64--88-1-102-54f048e3:11cd38349ae:-8000:0000000000000EAF'
          name = 'FiziskaPersona' visibility = 'public' isSpecification = 'false'
          isRoot = 'false' isLeaf = 'false' isAbstract = 'false' isActive = 'false'>...
        <UML:Class xmi.id = '-64--88-1-102-54f048e3:11cd38349ae:-8000:0000000000000EBC'
          name = 'Persona' visibility = 'public' isSpecification = 'false' isRoot = 'false'
          isLeaf = 'false' isAbstract = 'true' isActive = 'false'>...
```

3.att. XMI dokumenta piemērs
Fig.3. An example of XMI document

Apstaigājot XMI aprakstu kā XML dokumentu, iespējams atlasīt visas tajā izmantotās datu struktūras, kuras pēc tam var izmantot meklēšanā, piemēram, lai noskaidrotu, vai dotajā reģistrā ir Web servisi, kas kā ieejas parametru saņem objektu persona un atgriež objektu Dokuments kolekciju – tas ir, uzdotai personai atgriež tās radītos dokumentus. Meklēšanas mehānisms sīkāk aprakstīts darbā [2]. Izmantojot XPath meklēšanu, iespējams arī automatizēti apstrādāt visu pieejamo Web servisu reģistru izmantoto datu hierarhiju XMI aprakstus, lai noskaidrotu, vai tā satur izstrādātājam nepieciešamo struktūru – ja ne, tad skaidrs, ka uzdevumam atbilstošu Web servisu tajā nav. Diemžēl šāda risinājuma sekmīgai izmantošanai visiem reģistriem būtu jāizmanto kopēja 'superhierarhija', kurā vienā hierarhijā būtu pārstāvētas visu reģistru visas izmantotās hierarhijas. Uzticēties tam, ka visi izstrādātāji savas hierarhijas izstrādās puslīdz līdzīgā stilā, nevar – lai šāds risinājums nedarbotos, pilnīgi pietiek ar to, lai identiskās hierarhijās identiskas datu struktūras būtu nosauktas atšķirīgās valodās.
Kad no reģistra ir atlasīta atbilstošo Web servisu kopa, izstrādātājam tie katrs jāpārbauda un jānovērtē, vai tie atbilst izvirzītajām prasībām – drošības ierobežojumi, funkcionālās prasības u.c. Zinot par Web servisu tikai tā izstrādātāju, kategoriju, ievaddatu un izvaddatu kopas, šādu informāciju nav iespējams iegūt. Savukārt, ja Web servisa metadatos ir tā projektējums vai norādes uz to, informācijas daudzums, kas pieejams lēmuma veikšanai, krasi palielinās. Protams, jebkurš uz XML bāzēts standarts nav īpaši viegli lasāms – tas attiecas arī uz XMI. Daudz vieglāk ir uztvert vizuālu informāciju diagrammu veidā – XMI aprakstu pārveidojot par attēlu, kurš satur sākotnējo UML diagrammu. Novērtējot attēlu, pieņemt lēmumu par Web servisa izmantošanu ir vieglāk. Attēla ģenerēšana iespējama, apstaigājot XMI aprakstu kā XML dokumentu un nolasītos objektus iezīmējot attēlā, izmantojot .NET System.Drawing vārdtelpā esošās klases, piemēram, Bitmap.

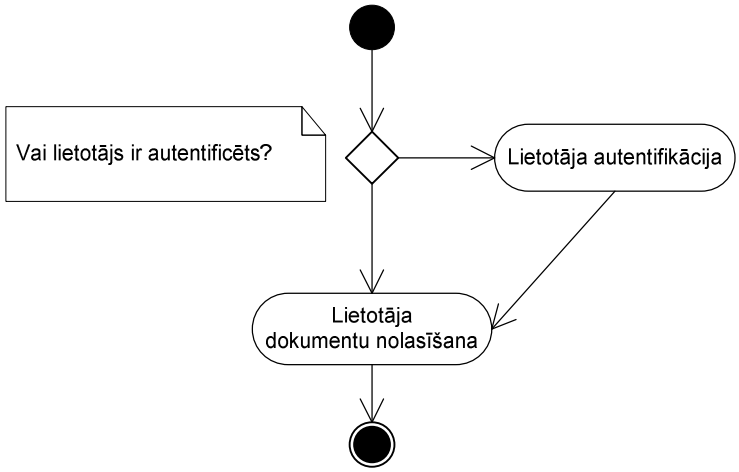
Risinājuma praktisks pielietojums

Lai pārbaudītu risinājuma izmantojamību reālos apstākļos, tika izveidoti vairāki Web servisi ar metodēm GetDocumentCount un publicēti Web servisu reģistrā. Divu Web servisu uzdevums bija atgriezt autentificētā lietotāja sagatavoto dokumentu skaitu, savukārt vēl viens Web serviss atgrieza visu pieejamo dokumentu skaitu, neskatoties uz lietotāju. Web servisu uzskaitījums dots 2. tabulā.

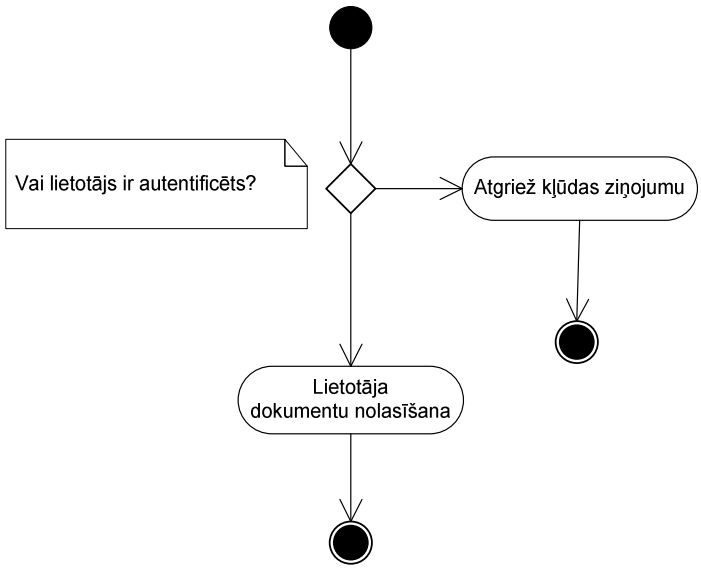
2.tabula
Table 2.

Testēšanas nolūkos zveidotie Web servisi
Web services created for testing purposes

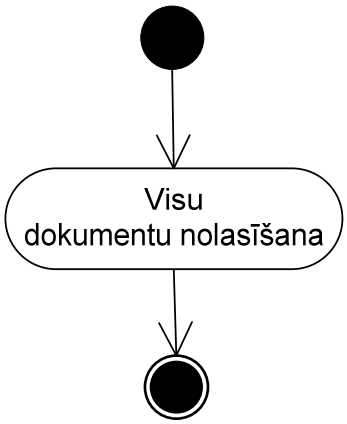
Nosaukums	Darbība
Service	Atgriež lietotāja dokumentu skaitu. Ja lietotājs nav autentificēts, tas tiek izdarīts. Metodes GetDocumentCount izpildes algoritms parādīts 4. attēlā.
Service_2	Atgriež lietotāja dokumentu skaitu. Ja lietotājs nav autentificēts, Web servisa darbība beidzas ar kļūdu. Metodes GetDocumentCount izpildes algoritms parādīts 5. attēlā.
Service_3	Atgriež visu lietotāju kopējo dokumentu skaitu. Netiek pārbaudīts, vai lietotājs ir autentificēts. Metodes GetDocumentCount izpildes algoritms parādīts 6. attēlā.



4.att. Web servisa Service metode GetDocumentCount – aktivitāšu diagramma
Fig.4. The activity diagram of Web service Service method GetDocumentCount



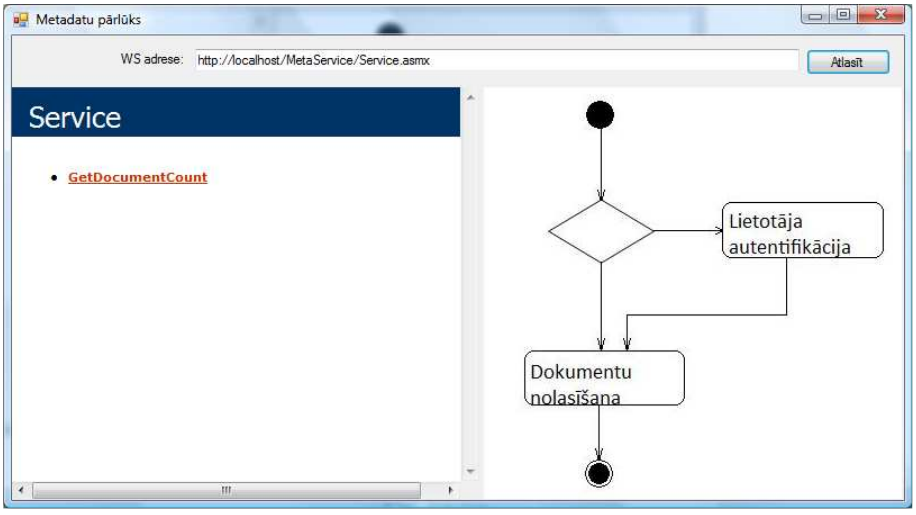
5.att. Web servisa Service_2 metode GetDocumentCount – aktivitāšu diagramma
Fig.5. The activity diagram of Web service Service_2 method GetDocumentCount



6.att. Web servisa Service_3 metode GetDocumentCount – aktivitāšu diagramma
Fig.6. The activity diagram of Web service Service_3 method GetDocumentCount

Pēc Web servisu publicēšanas tika izveidotas to aktivitāšu diagrammas (4. – 6. attēli), no kurām tika uzģenerēti XMI apraksti, izmantojot rīku ArgoUML. Adreses uz šiem XMI dokumentiem tika iekļautas izveidoto Web servisu metadatos, izmantojot VISS metadatu standarta elementu RELATION. Ja Web servisā ir vairākas metodes, autori iesaka katram metadatu RELATION elementam CONTENT atribūtā XMI apraksta faila nosaukumu uzdot vienādu ar metodes nosaukumu, lai vieglāk varētu saprast, kurai metodei atbislt kurš XMI apraksts – piemēram, ja Web servisā ir metode GetDocumentCount, tad tā arī vajadzētu saukt atbilstošo XMI aprakstu.

Pēc tam tika veikta meklēšana reģistrā, izmantojot meklēšanu ar datu hierarhijām „Persona" un „Dokuments" (detalizēta informācija par šādas meklēšanas principiem atrodama darbā [2]). Uzdodot šādus parametrus, no reģistra tika atgriezti visi 2. tabulā minētie Web servisi. Aplūkojot rezultātus, redzams, ka visi trīs Web servisi satur metodi GetDocumentCount – tātad atgriež dokumentu skaitu, taču nekur nav pateikts, kādi dokumenti tiek skaitīti, vai ņemts vērā, vai lietotājs ir autentificēts u.t.t.
Redzams, ka īstā Web servisa izvēlei nepieciešams zināt metodes izpildes algoritmu. Lai to izdarītu, .NET vidē tika izveidots rīks, kas pārlasa Web servisa metadatus un atgriež visus XMI dokumentus, kas minēti Web servisu metadatu elementā RELATION. Izveidotajā rīkā redzamas metodes, kuras nodrošina Web serviss – uzklikšķinot uz metodes nosaukuma, rīks no XMI apraksta ģenerē metodes izpildes algoritmu. 7. attēlā redzams piemērs ģenerētajam attēlam.



7.att. Web servisa Service metodes GetDocumentCount ģenerētais izpildes algoritms
Fig.7. The generated image of Web service Service method GetDocumentCount execution algorithm

Aplūkojot visu Web servisu piedāvāto metožu izpildes algoritmus, tika secināts, ka uzdevuma veikšanai piemērotākais ir Service_2, kas neautentificēta lietotāja pieprasījumam atgriež atbilstošu kļūdas paziņojumu.

Nobeigums

Metadatu papildināšana ar Web servisa izpildes algoritma shēmu XMI formātā ir vienkārša, jo pārsvarā visi izstrādes un projektēšanas rīki piedāvā diagrammu eksportu uz XMI formātu. Diemžēl katrs rīks XMI ģenerē ar dažādām niansēm – piem., ArgoUML UML aktivitāšu diagrammā esošos objektus eksportē uz XMI formātu to izveidošanas secībā, līdz ar to jāizstrādā algoritmi, kas izveido diagrammas attēlu pietiekami saprotamu, nevis haotisku un ar līnijām, kuras krustojas. Dažādiem UML izstrādes rīkiem līdz ar to vēl joprojām ir problēmas saprast un importēt XMI dokumentus, kuri nākuši no citas vides.
Standartu izmantošana stipri atvieglo Web servisu izvēles risinājuma izstrādi –jāizstrādā tikai attēlu ģenerēšana no XMI, lai gan iepriekšminēto iemeslu dēļ šis uzdevums nav triviāls. Izstrādāto risinājumu iespējams izmantot jebkurā Web servisu reģistrā, ne tikai tādos, kuri uzlaboti, lai varētu izmantot meklēšanu pēc datu hierarhijām – galvenais, lai Web servisi saturētu norādes uz XMI dokumentiem. Nākotnē paredzēts izstrādāt meklēšanu XMI dokumentos – vai tie satur noteiktu funkcionalitāti, piemēram, lai uzreiz varētu atlasīt Web servisus, kuros tiek veikta autentifikācija aktīvajā direktorijā.

Literatūra

1. UDDI specifikācija [Elektroniskais resurss] – http://www.uddi.org/pubs/uddi_v3.htm
2. Stipravietis P., Ziema M. Paplašinātu metadatu izmantošana WEB servisu identifikācijā // Rīgas Tehniskās universitātes zinātniskie raksti. 5.sēr., Datorzinātne. Datorvadības tehnoloģijas. – 32.sēj. (2007), 108.-115.lpp.
3. Object Management Group – UML [Elektroniskais resurss] – http://www.uml.org/
4. XML Metadata Interchange [Elektroniskais resurss] – http://www.omg.org/technology/documents/formal/xmi.htm
5. Timothy J. Grose, Gary C. Doney, Stephen A. Brodsky, PhD Mastering XMI – Wiley Publishing, 2002.
6. Valsts informācijas sistēmu savietotājs, metadatu un e-pakalpojumu identifikācijas standarts v1.0
7. VISS IS servisu, XML ziņojumu shēmu un e-pakalpojumu katalogs [Elektroniskais resurss] – https://ivis.eps.gov.lv/ivisportal/

Stipravietis P., Ziema M. Industrijas standartu UML un XMI izmantošana WEB servisu izvēlē
Servisorientētā arhitektūra (SOA) mūsdienās ir ieņēmusi vienu no vadošajām lomām programmatūras izstrādē, izmantojot savā starpā nesaistītas, loģiski noslēgtas tīkla komponentes jeb servisus, kuras var atkārtoti izmantot citu risinājumu izstrādē. Visbiežāk šie servisi tiek publicēti kā XML Web servisi. Lai atvieglotu risinājumu izstrādi, ir izveidoti Web servisu reģistri, kuros pēc dažādiem parametriem var veikt meklēšanu, lai atrastu nepieciešamos Web servisus. Diemžēl esošie reģistru risinājumi ir vai nu ierobežoti meklēšanas iespējās (piemēram, standarta UDDI reģistri) vai arī nedod informāciju par to, kā Web serviss veic savu uzdevumu. Šādos gadījumos ir zināmi tikai Web servisa ieejas un izejas dati, bet nav zināms izpildes algoritms. Piemēram, ir pieejami 3 Web servisi, kas katrs atgriež dokumentu skaitu, taču katrs veic citādas darbības, ja servisa izsaucējs nav autentificēts. Lai izvēlētos piemērotāko Web servisu, ir vai nu jāzina to projektējums vai arī jāizmēģina visi trīs un tad jāizvēlas piemērotākais.
Izmantojot standartus UML (Unified Modeling Language) un XMI (XML Metadata Interchange), ir iespējams Web servisa metadatos publicēt tā izpildes algoritmu. Izstrādājot Web servisu, tā izpildes algoritms tiek aprakstīts ar UML aktivitāšu diagrammas palīdzību. Daudzi UML projektēšanas rīki piedāvā iespēju šīs diagrammas eksportēt uz XMI formātu. Kad Web serviss tiek publicēts reģistrā, tā metadati tiek papildināti ar

norādi uz XMI dokumentu, kas satur tā izpildes algoritmu. Pēc tam, kad sistēmas izstrādātājs ir veicis meklēšanu reģistrā, lai atrastu piemērotus Web servisus, katram atrastajam servisam tiek apstrādāts atbilstošais XMI dokuments un ģenerēts izpildes algoritma attēls. Aplūkojot šos attēlus, sistēmas izstrādātājs uzreiz var izvēlēties piemērotāko Web servisu uzdotā uzdevuma veikšanai.

Stipravietis P., Ziema M. Selection of Web services using industry standards UML and XMI

Nowadays service-oriented architecture (SOA) has emerged as one of the main approaches to be used in software development, mainly in business process automation and systems integration. It uses loosely coupled autonomous network components called services, which can be later be reused in other solutions. These services often are exposed as XML Web services. To facilitate the development of solutions which use Web services and to increase availability of such services special registries are used, i.e. UDDI (Universal Description Discovery and Integration). Although the developer can query registries for appropriate services, the registries often offers only limited query parameters to use or do not provide information regarding to how does the web service perform its task. In these cases only input and output parameters are known. For example, there are three Web services into registry and they all return the document count, but each of them acts differently if the caller is not authenticated. To choose the most appropriate Web service, one either must know the design of all three services or try them all and then make the choice.

It is possible to publish the execution algorithm of the Web service in its metadata using industry standards UML (Unified Modeling Language) and XMI (XML Metadata Interchange). The Web service developer describes it using UML activity diagram and exports it to XMI – many of UML design tools provide such functionality. When the Web service is being published in the registry, the URI to generated XMI document is appended to its metadata. After the querying the registry for appropriate web services, each XMI document of every service is then downloaded and parsed, generating an image of the execution algorithm in result. After the evaluation of these images, the developer can easily choose the most appropriate Web service to perform given task.

Стиправиетис П., Зиема М. Выбор WEB сервисов с применением стандартов индустрии UML и XMI

В настоящее время Сервисно-ориентированная архитектура (SOA) заняла свое место как один из основных подходов, которые используеться в разработке программного обеспечения, главным образом в автоматизации бизнес-процессов и интеграции систем. SOA использует автономные сетевые компоненты, названные службами, которые могут позже быть многократно использованны в других решениях. Эти службы часто разработанны как XML Веб-службы. Чтобы облегчить разработку решений, которые используют Веб-службы и увеличить доступность таких служб, используются специальные реестри, например, UDDI. Хотя разработчик может сделать запрос реестру для соответствующих служб, реестри часто предлагает только ограниченные параметры запроса, или не предоставляют информацию о том, как веб-служба выполняет свою задачу. В этих случаях известны только параметры ввода и вывода. Например, в реестр находятся три Веб-службы, и все они возвращают число документов, но каждый из них по-другому обрабатывает ситуацию, если вызывающая программа не заверена. Выбрать самую соответствующую Веб-службу, разработчик либо должен иметь проектирование всех трех служб или проверить их всех и затем сделать выбор.

Алгоритм выполнения Веб-службы возможно хранить в ее метаданных, используя стандарты UML (Унифицированный язык моделирования) и XMI (Обмен Метаданных XML). Разработчик Веб-службы описывает алгоритм выполнения Веб-службы с помощью диаграмм активностней UML и экспортирует на формат XMI – многие из средств проектирования UML обеспечивают такие функциональные возможности. Когда Веб-служба записывается в реестре, к метаданным прилагается URI, указывающий на сгенерированный документ XMI. После запроса реестру на соответствующих веб-служб каждый документ XMI каждого сервиса загружается и анализируется программой,, которая генерирует изображение алгоритма выполнения. После оценки этих изображений разработчик может легко выбрать самую соответствующую Веб-службу, чтобы выполнить данную задачу.