Izkušnje pri načrtovanju in razvoju storitvene arhitekture

Marko Tekavc, Matjaž B. Jurič
Fakulteta za elektrotehniko, računalništvo in informatiko, Smetanova 17, 2000 Maribor
marko.tekavc@uni-mb.si

Povzetek

Storitveno orientirana arhitektura (SOA) je trenutno zelo aktualna tema, saj veliko strokovnjakov s področja informacijskih tehnologij vidi prednosti takšne arhitekture v pospešitvi procesa razvoja aplikacij in prilagodljivosti takšnih sistemov ter s tem hitrejšemu odzivu na spreminjajoče se poslovne potrebe. Žal danes obstaja kaj malo izkušenj pri razvoju rešitev na SOA. Eden izmed razlogov za to je prav gotovo tudi v pomanjkljivi in omejeni podpori orodij za načrtovanje in razvoj takšnih rešitev. Vendar pa se z vedno bolj zrelimi standardi na tem področju izboljšuje tudi podpora orodij, s čemer pa načrtovanje in razvoj rešitev SOA postaja veliko hitrejše in predvsem enostavnejše. V prispevku si bomo zato pogledali ključne prednosti SOA, nato pa predstavili prednosti in slabosti uporabe orodij za razvoj SOA združljivih rešitev, temelječe na izkušnjah pri razvoju prototipa takšne rešitve.

Ključne besede: storitveno orientirana arhitektura, spletne storitve, povezljivost, WS-I, orkestracija, BPEL

Abstract

Design and development experience with service oriented arhitecture

Service oriented architecture (SOA) is currently a very popular topic. Information technology experts see its advantages in speeding up application's development process and in adaptability which leads us to faster response to changing business needs. Unfortunately, there are not many experiences in development of SOA solutions. One of the reasons for that lies surely in deficient and limited support of applications for planning and development of such solutions. Yet, with far more mature standards in this field, the tool's support is getting better, and with this, the SOA development solutions and their planning are becoming faster and easier. In the article, we will therefore have a look at the key advantages of SOA, and we will introduce the advantages and disadvantages of using the tools for development of SOA compliant solutions, based on experiences gained by the development of SOA prototype.

Keywords: service oriented architecture (SOA), web services, linkage, WS-I orchestration, BPEL

1 Uvod

Storitvene arhitekture so zadnji, najaktualnejši arhitekturni pristop, ki obljublja rešitve znanih težav, s katerimi se soočamo pri razvoju in vzdrževanju kompleksnih informacijskih sistemov. S tehnološkega vidika so storitvene arhitekture zgolj posebna vrsta porazdeljenih sistemov, pri katerih so komponente sistema storitve [2]. Vendar pa se razvoj SOA razlikuje od klasičnega razvoja objektnoorientiranih aplikacij, na kar moramo biti pozorni že pri načrtovanju storitvene arhitekture. Še do nedavnega ni bilo podpore orodij za implementacijo rešitev SOA ali pa je bila podpora zelo omejena. Tako je bilo za uspešno implementacijo potrebno podrobno poznavanje nizkonivojskih tehnologij, standardov, kompleksnih nastavitvenih datotek, namestitvenih procedur itd. Zaradi tega je bila izdelava storitev, še bolj pa zagotavljanje varnosti, povezljivosti in orkestracija storitev predvsem v domeni pilotskih projektov in akademskih krogov. Z dopolnitvijo in utrditvijo standardov na področju SOA pa se vse bolj uveljavljajo tudi orodja, ki danes omogočajo obsežnejšo in kakovostnejšo podporo razvoju na vseh nivojih SOA. Z uporabo orodij se sicer odpovemo delu fleksibilnosti, nemalokrat naletimo na nerazumne omejitve in nepredvidene težave, vendar pa hitrejši in predvsem enostavnejši razvoj odtehtata slabosti orodij in pripomoreta k širši uporabnosti novih tehnologij in pristopov.

V prvem delu prispevka na kratko podamo lastnosti in prednosti SOA, nato pa predstavimo naš pristop k načrtovanju in razvoju SOA in izkušnje, ki smo jih ob tem pridobili. Osredotočili se bomo predvsem na opis prednosti in slabosti uporabe razvojnih okolij ter na konkretne težave in omejitve, na katere smo naleteli pri razvoju in načrtovanju.

2 Storitvena arhitektura

Storitveno orientirana arhitektura (SOA) je arhitekturna paradigma, ki predpisuje uporabo storitev in način komuniciranja med temi storitvami. Sama storitev predstavlja neko funkcionalnost, ki je neodvisna od drugih storitev, pri čemer opravlja natančno določeno

operacijo, ki jo lahko uporabimo tudi zunaj konteksta aplikacije [5]. Pri tem je pomembno, da ima storitev natančno definiran vmesnik, dejansko gre za natančen/podoben opis sporočil, ki si jih izmenjujejo storitve in do katerega lahko dostopimo prek omrežja z uporabo standardnih protokolov in podatkovnih tipov [6].

Storitvene arhitekture temeljijo na [2]:

- različnih tipih storitev: sporočilnih, podatkovnih, komponentnih in konverzacijskih,
- posrednikih storitev oz. storitvenem vodilu (ESB – Enterprise Service Bus),
- zmožnosti deklarativne kompozicije storitev z uporabo orkestracije ali koreografije,
- obravnavi izjem in kompenzaciji delnih učinkov procesov,
- prestrezanju zahtev in razširljivosti ter
- povezljivosti.

Realizacija storitvene arhitekture pa je nujno povezana z uporabo ustreznih tehnologij. V tem trenutku so za realizacijo storitvenih arhitektur najprimernejša in tudi najpogosteje uporabljana tehnologija spletne storitve, pri čemer pa za realizacijo zgoraj opisanih konceptov ne zadošča več zgolj uporaba treh osnovnih tehnologij (SOAP, WSDL in UDDI), pač pa je potrebno uporabiti sklad tehnologij WS-* [2].

2.1 Prednosti storitvene arhitekture

Seveda storitvene arhitekture ne uporabimo zgolj zato, ker je to aktualno, temveč zato, ker predvsem na procesnem nivoju prinaša določene prednosti:

- izboljšano odzivnost na poslovne spremembe,
- hitrejše povračilo vlaganj (ROI-Return Of Investment),
- preprostost definicije in sprememb poslovnih procesov,
- večjo konsistentnost in lažje vzdrževanje kode ter
- izboljšano povezljivost.

2.2 Načrtovanje storitvene arhitekture

Prvi korak razvoja storitvene arhitekture je načrtovanje, kjer se moramo oddaljiti od objektno orientiranega razmišljanja in začeti razmišljati o storitvah ter njihovem povezovanju. Načrtovanje informacijskih rešitev in njihova izvedba v skladu s paradigmami storitvene arhitekture velikokrat pomeni tudi delno opustitev lepih praks objektno orientiranega razvoja, predvsem uporabe vmesnikov, dedovanja in mnogoličnosti. Ker se med odjemalcem in storitvijo prenaša le stanje, ki ga je mogoče serializirati, takoj po prenosu pa se izgubi tudi povezava med izvornim objektom in razredom, lahko trdimo, da so za parametre metod, ki jih omogoča spletna storitev, uporabni le najpreprostejši vrednostni objekti, torej objekti, katerih edino poslanstvo je prenos podatkov. S tem pa se že precej oddaljimo od objektne paradigme. Pri izpostavljanju objektno zgrajenih komponent prek vmesnika spletne storitve se tako srečamo z določenimi težavami, še posebno, če komponente gradimo univerzalno, z upoštevanjem načel objektnega programiranja in s sistemi za distribuirane objekte v podzavesti. Delno se lahko nekaterim najenostavnejšim težavam izognemo z ročno implementacijo mehanizmov za serializacijo/deserializacijo sporočil SOAP ter z ročno izdelavo vmesnika WSDL. Vendar zaradi kompleksnosti in potencialnih napak takšnega pristopa skoraj ni zaslediti. Odvisni smo torej predvsem od orodij, ki nam pomagajo pri gradnji storitev. Žal smo pri večini orodij soočeni s tem, da ne znajo vmesnikov komponent, napisanih v nekem programskem jeziku, prevesti v ustrezen dokument WSDL. Prav tako nam za samodejno kreiranje dokumenta WSDL ni uspelo uporabiti parametrov, podanih v obliki vmesnikov. Rezultat je zato, objektno gledano, dokaj nekonsistentna koda. Zaradi tega smo v naši rešitvi v ločenih paketih izdelali posebne dele kode, namenjene zgolj izpostavitvi komponente v obliki spletne storitve.

V načrtu implementacije storitvene arhitekture smo le-to razdelili v tri korake: implementacijo storitev, zagotavljanje povezljivosti ter povezovanje oziroma orkestracijo storitev. V naslednjih poglavjih bomo opisali posamezen korak in podali nekaj izkušenj, ki smo jih ob implementaciji posameznega koraka pridobili.

3 Implementacija storitev s pomočjo razvojnih orodij

Čeprav storitvena arhitektura ne temelji zgolj na spletnih storitvah, so prav spletne storitve zaželen in na standardih osnovan način za realizacijo storitvene arhitekture [3]. Razvoj in kompozicija storitev, pa je lahko zelo zahtevno delo, pri katerem moramo do potankosti poznati predpisane standarde, natančno uglasiti razvojno okolje in pri tem pripraviti veliko nastavitvenih datotek, ki nimajo neposredne poslovne vrednosti. Ugotovili smo že, da si moramo za hitrejši in učinkovitejši razvoj storitev pomagati z razvojnimi orodji. Takšen razvoj pomaga razvijalcem in

načrtovalcem avtomatizirati številne dolgotrajne naloge, ki so pogosto tudi vzrok napak. Danes nam orodja na področju razvoja za SOA ponujajo veliko pomoči v različnih fazah in področjih razvoja. Orodja nam avtomatizirajo in transparentno pripravijo razvojno okolje ter praviloma nudijo grafične vmesnike z naborom številnih komponent, ki jih lahko preprosto in intuitivno uporabimo. Ena izmed ključnih prednosti uporabe orodij je podpora dostopu do podatkovnih baz, kjer najdemo zelo različne nivoje podpore, vse od popolne podpore objektno relacijskega preslikovanja do zgolj avtomatizacije dela nalog za dostop do podatkovne baze. Za razvoj SOA je posebej zanimiva podpora avtomatskemu ustvarjanju spletnih storitev, ki razvijalcem omogoča, da se osredotočijo zgolj na problem z vidika programskega jezika. Tipično nam orodja omogočajo izdelavo ogrodja spletnih storitev iz datotek WSDL, razredov in tudi shem XML. Prav tako preprosto pa lahko s pomočjo orodij izdelamo tudi odjemalce spletnih storitev. Nazadnje lahko s pomočjo orodij aplikacijo tudi zapakiramo in neposredno namestimo na aplikacijski strežnik.

3.1 Vidiki uporabe orodij za razvoj SOA

Nedvomno uporaba orodij z avtomatizacijo številnih dolgotrajnih in utrudljivih nalog razvoja storitev in kasneje poslovnih procesov poveča produktivnost razvijalcev. Ker je večina kode in nastavitev, namenjenih zgolj infrastrukturi abstrahirane ali pa jo samodejno ustvari orodje, se lahko razvijalci osredotočijo na implementacijo poslovne logike. Ob tem pa s prikritjem in avtomatizacijo številnih kompleksnih nalog orodja močno olajšajo učenje in omogočijo tudi neizkušenim programerjem hiter pristop k razvoju SOA [5]. Seveda pa ima uporaba orodij tudi nekatere pomanjkljivosti in omejitve. Če si želimo poenostaviti razvoj in izkoristiti prednosti, ki nam jih ponujajo orodja, se moramo sprijazniti z omejitvami teh orodij. Sploh kadar gre za prvo podporo orodja določeni tehnologiji, pogosto naletimo na težavo, ki je ne moremo rešiti na način kot so si ga zamislili snovalci orodja. Še ena izmed ključnih slabosti uporabe orodij je, da le-ta velikokrat uporabljajo svoje verzije rešitev in ne sledijo standardom, ki jih predpisujejo neodvisne organizacije. Kljub vsemu v večini primerov prednosti, ki jih prinaša uporaba orodij odtehtajo slabosti.

3.2 Predstavitev izkušenj pri uporabi orodij za izdelavo spletnih storitev

Med orodji, ki so bila v danem trenutku na voljo, smo izbrali Oracle JDeveloper, ki se je v kombinaciji z okoljem BPEL Process Manager v našem primeru izkazal za najprimernejše orodje. Uporabili smo večino najpomembnejših prednosti, ki nam jih pri izdelavi storitev ponujajo orodja. Zelo enostavna in prilagodljiva je možnost izdelave deskriptorjev za pakiranje spletne storitve ter neposredna namestitev storitev na spletni ali aplikacijski strežnik. Ker orodje podpira izdelavo spletnih storitev iz javanskih razredov, smo se razvoja spletnih storitev lotili po izdelanem načrtu in pri tem večinoma sledili pravilom objektno orientiranega razvoja. Tak pristop se ni izkazal za najboljšega, zato smo kasneje, kot je to opisano v poglavju o načrtovanju, rešitev nekoliko prilagodili. Pri tem smo bili v razredih, ki smo jih nato izpostavili kot spletne storitve, še posebej pozorni na uporabo tipov, ki se lahko serializirajo v od jezika neodvisne XML tipe. Verzija orodja, ki smo jo uporabljali, je omogočala zgolj minimalno prilagoditev raznih parametrov pri avtomatski izdelavi spletnih storitev iz razredov. Najbolj izrazita slabost je ta, da spletne storitve, ki jih izdela orodje, niso popolnoma skladne s standardi in da ni moč določiti tipa kodiranja SOAP – izdelati je mogoče le rpc/encoded spletne storitve. Ta omejitev je bila zaradi težav pri orkestraciji tudi vzrok za izbiro najnovejše verzije razvojnega orodja, ki pri izdelavi spletnih storitev ponuja večjo fleksibilnost in podpira tudi izdelavo standardnih spletnih storitev JAX-RPC. Ker je šlo zgolj za pilotski projekt, smo takšno odločitev lahko sprejeli, seveda bi bilo precej drugače, če bi morali našo rešitev namestiti v produkcijsko okolje na predpisan spletni strežnik. Kajti če smo želeli izrabiti prednosti, ki jih prinaša nova verzija orodja, smo morali uporabiti tudi novo verzijo aplikacijskega strežnika s podporo za J2EE 1.4. Vse skupaj kaže, da orodja za podporo izdelave storitev še niso popolnoma dozorela, vendar velik napredek med verzijama orodij pomeni, da se orodja na tem področju intenzivno izboljšujejo.

4 Zagotavljanje povezljivosti

Prav tako kot razvoj storitev je pomembno tudi zagotavljanje povezljivosti le-teh. Večina rešitev SOA temelji na spletnih storitvah in ker le-te predpisuje

množica standardov (XML, SOAP, WSDL, UDDI), naj bi bila povezljivost zagotovljena že sama po sebi. Seveda pa stvari v praksi vedno ne uspejo slediti teoriji. Kot smo ugotovili že v prejšnjem razdelku, lahko posebno težavo zaradi specifičnih rešitev posameznih proizvajalcev predstavlja prav povezljivost storitev, izdelanih s pomočjo orodij. Za standarde na tem področju skrbi organizacija WS-I, ki je v ta namen izdelala dokument specifikacije osnovnega profila povezljivosti spletnih storitev. Osnovni profil sestoji iz niza omejitev in priporočil, z upoštevanjem katerih razvijalci implementirajo povezljive spletne storitve. Dokument, ki opisuje osnovni profil WS-I, določa zahteve, ki jih morajo izpolnjevati spletne storitve in sporočila med spletnimi storitvami, da so v skladu s tem profilom. Te zahteve so podane v obliki oštevilčenih trditev, ki so zapisane tako za končne točke spletnih storitev, kot tudi artefakte (sporočila, WSDL opise ...) le-teh. Del profila pa so tudi testna orodja, s katerimi lahko preizkusimo skladnost naše storitve s profilom. Kljub velikemu napredku, ki ga je prinesel osnovni profil k povezljivosti spletnih storitev, pa problema povezljivosti ne rešuje popolnoma. Za to obstajata vsaj dva razloga: prvič, veliko število obstoječih spletnih storitev, je bilo nameščeno na platforme, ki niso skladne z osnovnim profilom in drugič, razvijajo se tudi nove spletne storitve, ki uporabljajo nove izboljšave spletnih storitev, kot npr. WS-Security ali lastne razširitve, ki pokrivajo lastnosti, ki jih ni v osnovnem profilu.

Seveda organizacija WS-I razvija tudi nove profile za povezljivost, vendar pa dokler ne bodo standardi spletnih storitev popolnoma končani in stabilni bomo vedno znova naleteli na težave pri povezljivosti. Če bomo uporabljali funkcije, ki so zunaj osnovnega profila (npr. alternativne transportne protokole ali varnostne mehanizme), je dobro, da najprej naredimo spletno storitev brez teh funkcionalnosti in jo testiramo za zmožnost povezovanja. Ko imamo ustrezno specifikacijo WSDL in povezljivo spletno storitev, pa uvedemo še dodatne funkcionalnosti. Poleg predpisanega v osnovnem profilu WS-I je priporočeno upoštevati še naslednje smernice:

- Čeprav nas osnovni profil WS-I pri izbiri kodiranja omeji na kodiranje literal v povezavi z RPC ali document, je priporočena uporaba kombinacije document/literal.
- Pri načrtovanju sheme XML, namenjene uporabi v spletni storitvi, je dobro upoštevati naslednja priporočila: shemo vedno izdelajmo z orodji, neodvisnimi od platforme; še posebej pa se izogibajmo uporabi za platformo specifičnih tipov. Prepričajmo se, da shema vsebuje vse informacije, ki jih spletna storitev potrebuje za delovanje. Pomembno je tudi, da se zavedamo težav, ki jih lahko povzroči uporaba funkcionalnosti, ki jih ne podpirajo večje platforme.
- Če želimo izdelati WSDL, ki bo kompatibilen z osnovnim profilom WS-I, ga lahko napišemo na roko in s pomočjo WS-I orodij preverimo, ali je dokument skladen z zahtevami profila. Ker je takšna izdelava WSDL navadno časovno dolgotrajna, ga praviloma izdelamo s pomočjo orodji, kasneje pa ga po potrebi popravimo tako, da odpravimo vse posebnosti platforme.

5 Orkestracija storitev

Za storitveno orientirano arhitekturo je zelo pomembna zmožnost sestavljanja in integracije spletnih storitev v poslovni proces. Zaželeno je, da tak proces predstavimo na neki standardizirani način z uporabo splošno priznanega jezika. Takšen standardni jezik je danes BPEL4WS, katerega poglaviten namen je standardizacija avtomatizacije procesov med spletnimi storitvami. Za uresničitev koncepta deklarativne kompozicije storitev lahko uporabimo jezik BPEL in orkestracijo storitev. Pri orkestraciji centralni proces, ki je lahko tudi sam spletna storitev, prevzame nadzor nad drugimi spletnimi storitvami in koordinira izvajanje operacij spletnih storitev, pri čemer vpletenim spletnim storitvam ni treba vedeti, da so del večjega poslovnega procesa [1].

Uporaba orodja za povezovanje storitev v poslovni proces je nepogrešljiva, še posebej zato, ker pri tem praviloma sodelujejo poslovni uporabniki, ki jih ne zanimajo standardi in niže ležeče tehnologije. Orodja za podporo izdelavi procesov BPEL so praviloma skladna s standardom BPEL4WS in nam dajejo vizualno okolje za izgradnjo procesov BPEL skupaj s standardnimi gradniki (npr. invoke, switch, assign). Najpomembnejše funkcionalnosti orodij za razvoj procesov BPEL so povleci in spusti okolje za načrtovanje procesov, vizualni urejevalnik XPath, neposredna namestitev procesa na strežnik, UDDI in WSIL raziskovalca in grafični vmesnik za izdelavo transformacij XSLT.

Pri razvoju naše SOA smo uporabili orodje Oracle BPEL Designer, ki je skoraj v celoti izpolnilo naša

pričakovanja. Orkestracijo storitev smo tako v celoti izvedli z uporabo grafičnega okolja pri čemer smo uspešno uporabili večino najpomembnejših funkcionalnosti, ki jih nudi orodje. Na težave smo naleteli pri sklicevanju na spletne storitve, ki so uporabljale rpc/encoded kodiranje sporočil SOAP. Orodje ni znalo razčleniti sheme, ki je vsebovala polje elementov in zato nismo mogli uporabiti nekaterih storitev. Z zamenjavo verzije razvojnega okolja za razvoj storitev smo lahko izdelali standardne spletne storitve, ki so uporabljale document/literal kodiranje sporočil SOAP, ki smo jih nato lahko preprosto vključili in jih med seboj povezali v razne procese. Proces BPEL, izdelan s tem orodjem, lahko namestimo na katerikoli strežnik BPEL, pri čemer pa moramo biti pozorni na to, da ne uporabljamo nekaterih naprednih funkcij, ki so specifične za Oraclove produkte.

6 Sklep

Storitveno orientirana arhitektura postaja vse pomembnejša in bo v prihodnosti, tako skupina Gartner, prevladujoča arhitektura [4]. Že danes je mogoče izdelati rešitve, ki temeljijo na šibko sklopljenih storitvah, vendar pa je za to potrebno veliko znanja, truda in časa. Z orodji, ki so na tržišču ali na tržišče šele prihajajo, lahko vloženi trud in čas za načrtovanje, razvoj in orkestracijo storitev močno zmanjšamo. Pri načrtovanju in razvoju našega pilotskega projekta smo potrdili tezo o hitrejšem in predvsem preprostejšem razvoju s pomočjo orodij. Kljub temu da smo naleteli na nekaj težav, ki so bile posledica nedovršenosti razvojnih orodij, se je takšen razvoj storitveno orientirane rešitve izkazal za primernega, če smo pripravljeni za hitrost in preprostost razvoja žrtvovati del fleksibilnosti in sprejeti nekaj kompromisov.

7 Viri in literatura

[1] JURIČ, Matjaž B.: Business Process Execution Language for Web Services, Packt Publishing, Oktober 2004.

[2] JURIČ, Matjaž B.: Storitvena arhitektura – zgolj kompozicija spletnih storitev?, COTL – časopis centra za objektno tehnologijo, Letnik 11, številka 1, pomlad 2005.

[3] MAHMOUD, Qusay H.: Service-Oriented Architecture (SOA) and Web Services: The Road to Enterprise Application Integration (EAI), Sun Microsystems, April 2005.

[4] ORT, Ed: Service-Oriented Architecture and Web Services: Concepts, Technologies, and Tools, Sun Microsystems, April 2005.

[5] The Middleware Company: Assisted SOA Development, The Middleware Company, November 2004.

[6] World Wide Web Consortium: Service Oriented Architecture, Web Services Architecture Group, 2003.

Marko Tekavc je podiplomski študent na Inštitutu za informatiko Fakultete za elektrotehniko, računalništvo in informatiko v Mariboru, kjer je tudi zaposlen kot asistent za področje informatike. Sodeluje na različnih projektih, tako aplikativnih kot znanstveno-raziskovalnih. Raziskovalna področja, s katerimi se ukvarja, pokrivajo objektno orientirane tehnologije, spletne storitve in XML, kompozicijo poslovnih procesov, agilne metode in procese ter preoblikovanje programske kode.

Dr. Matjaž B. Jurič je izredni profesor na Inštitutu za informatiko Fakultete za elektrotehniko, računalništvo in informatiko v Mariboru. Ukvarja se s storitvenimi arhitekturami, kompozicijo poslovnih procesov, z integracijo in elektronskim poslovanjem ter s spletnimi storitvami in z optimizacijo zmogljivosti. Je avtor oziroma soavtor knjig Business Process Execution Language for Web Services (Packt Publishing), .NET Serialization Handbook, J2EE Design Patterns Applied, Professional J2EE EAI in Professional EJB (Wrox Press) ter poglavij v knjigah More Java Gems (Cambridge University Press) in Technology Supporting Business Solutions (Nova Science Publishers). Članke je objavljal tudi v revijah SOA-Web Services Journal, eAI Journal, Java Report, Java Developers Journal in sodeloval na konferencah, kot so OOPSLA, Oracle Open World, Java Development, BEA Forum, Wrox Conferences itd. Sodeloval je pri številnih projektih doma in v tujini, med drugim tudi pri razvoju RMI-IIOP, sestavnega dela platforme Java 2. Je tudi član odbora član BPEL Advisory Board.