

hyväksymispäivä arvosana

arvostelija

Palveluiden haku palveluorientoituneessa tietojenkäsittelyssä

Toni Könnilä

Helsinki 3.12.2015

HELSINGIN YLIOPISTO
Tietojenkäsittelytieteen laitos

Sisältö

1	Abstrakti	1
2	Johdanto	1
3	Palveluorientoitunut arkkitehtuuri	2
4	Palveluorientoituneiden arkkitehtuurien haittapuolet	4
5	Palveluiden haku	4
5.1	Yleisesti	4
5.2	Erilaisia rekistereitä	5
5.3	Palveluiden haun ongelmat	6
5.4	Semanttinen palveluiden haku	7
6	Yhteenveto	7

Liitteet

1 Malli ABC

1 Abstrakti

Service-oriented computing (SOC) on ajatusmalli, jossa yhden monoliittisen ohjelman sijaan keskitytään keskenään kommunikoiviin palveluihin. SOC nojaakin hyvin vahvasti palveluorientoituneeseen arkkitehtuurimalliin (myöhemmin SOA), joka ajaa tätä ideaa. SOA ei kuitenkaan tarjoa itsessään vastauksia siihen miten turvallisuus, palveluiden koordinointi ja ohjelman muu arkkitehtuuri on rakennettu.

Tässä aineessa keskitytään palveluorientoituneiden arkkitehtuurien yhteisiin piirteisiin ja erityisesti palveluidenhakuun (*service discovery*), ja siihen liittyen, palveluiden läpinäkyvyyteen (*location transparency*).

2 Johdanto

SOC:in on ajatusmalli, joka painottaa eri palveluiden käyttöä palvelun koostamiseksi. Yleensä lopputuloksena on siis palvelu, jonka ideana on kommunikoida muiden palveluiden kanssa, jotka ovat itsessään alustariippumattomia ja väljästi yhteydessä toisiinsa (*loosely coupled*). Business-taustaisena ideana SOA:issa (*Service-oriented architectures*) on tuottaa hyvin keskenään toimivia, mahdollisimman atomisia palveluita, jotka ovat uudelleenkäytettäviä, helposti korvattavissa ja ylläpidettävissä tuottaen nopeasti kustannustehokkaita hajautettuja palveluja lyhyellä kehitys-julkaisuikkunalla (*time-to-market*).

Yksittäisen palvelun tehtävä voi olla yksittäinen pyyntö, tai monimutkaisempi businessprosessi, ja tällaisten palveluiden tarjoajat (olipa ne sitten organisaatioita tai yksityishenkilöitä) mahdollistavat ohjelmallisen pääsyn palveluihinsa ja dataansa internetin yli käyttäen standardoituja välitystapoja, käyttäen yleensä XML-perustaisia tai REST-mallin mukaisia HTTP-protokollaan perustuvia rajapintoja käyttäen. Näiden palveluiden tarjoajien (*service providers*) vastuulla on antaa palvelukuvaus ja ylläpitää palvelua, ja antaa palveluun liittyvää teknistä sekä muuta tukea. Palvelua kutsuvat tahot voivat olla muita ohjelmia, tai yksittäisiä tahoja, esimerkiksi kotipalvelin joka pyytää resurssia REST-rajapinnan toteuttavalta palvelulta internetin yli. Tästä seuraa, että näillä näillä palveluilla on tiettyjä vaatimuksia ja edellytyksiä.

- *Teknologiariippumattomuus*: Palvelun ei kuulu olla liian teknologiariippuvainen, vaan sitä pitää pystyä kutsumaan alalla vallitsevilla työkaluilla ja alan standardien mukaisesti.

- *heikko linkittyneisyys (loosely coupled)*: palvelun tulee olla irti mahdollisesta kontekstista, ja toimia itsenäisesti (ainakin ulospäin) ilman tietoa muista palveluista tai komponenteista.
- *Palvelun haettavuus (service discovery) ja tuki sijaintiläpinäkyvyydelle (location transparency)*: Palveluiden määrittelyt ja varsinaiset sijaintitiedot tulee olla tallennettuna johonkin - esimerkiksi UDDI repositorioon, ja asiakkaila/tahoilla pitää olla pääsy palveluun tietämättä missä se varsinaisesti sijaitsee.

Tässä aineessa keskitytään palveluiden haettavuuteen ja siihen miten se voidaan toteuttaa palveluorientoituneessa arkkitehtuurimallissa.

3 Palveluorientoitunut arkkitehtuuri

Perinteisesti SOC:in palvelumalli johon nojataan, on ollut SOA. SOA pyrkii käyttämään erillisiä palveluja koostaakseen monimutkaisiakin ohjelmistoja. Näiden erillisten palvelujen on tarkoitus olla uudelleenkäytettäviä ja joustavia, ja siten nopeuttaa sekä halventaa ohjelmistokehitystä.

Erillinen yksittäinen palvelu eroaa ajatusmaailmaltaan monoliittisen arkkitehtuuri-puolen mallista tietyissä osa-alueissa. Se on itsenäinen, väljästi linkitetty muualle, uudelleenkäytettävä, itsensä kuvaava ja helposti siirrettävä toiselle alustalle. Monoliittisessa arkkitehtuurissa ollaan yleensä tiukasti sidottu sovittuihin teknologioihin, ja samalla ollaan sidottu sen teknologian vaatimuksiin ja rajoitteisiin. Ohjelmat ovat tiiviisti linkattuina muihin kirjastoihin, ja yksittäisten luokkien ja palveluiden korvaaminen päättyy lopulta useiden moduulien muokkaamiseen tätä uutta korvaavaa moduulia tukevaksi.

Myös business-mielessä monoliittinen arkkitehtuuri kasvaa usein turhauttavaksi: yhä kasvavan ohjelman monimutkaisuus ja sisäinen linkittyneisyys muodostuu hidasteeksi uusille projektiin liittyville työntekijöille ja vaatii enemmän ja enemmän aikaa ohjelman kokonaisuuden ymmärtämiseksi. Ohjelman luokkien ja palveluiden linkittyneisyys muodostaa myös kokonaisuuden, jota on vaikeampi muokata, jolloin uudelta työntekijältä saattaa jäädä kooditasolla huomiotta tehdä muutokset kaikkiin paikkoihin joihin muutokseen liittyvä moduuli on kytköksissä. Tämä ongelma on usein itseään ruokkiva, ja ongelmat kasaantuvat entisestään, jos koodin laadusta ei pidetä

tasaisesti huolta. Tätä palveluorientoituneisuus sen sijaan yrittää helpottaa rajaamalla palvelut selkeiksi itsenäisiksi kokonaisuuksiksi.

Monoliittisessa arkkitehtuurissa koko ohjelma voidaan nähdä isona siilona, kun taas SOA on yksinkertaisimmillaan infrastruktuuri, joka tukee erillisten palveluiden kommunikointia keskenään standardien protokollien avulla. Myös näiden palveluiden tulee olla löydettävissä alalla vallitsevien standardirajapintojen avulla. Kun nämä osat toimivat, ohjelmat pääsevät tarvittaessa helposti käsiksi näihin palveluihin ilman tarvetta monimutkaisille räätälöidyille kommunikointiprotokollille.

SOA on looginen tapa suunnitella ohjelmistoja joko yksittäisten käyttäjien tarpeeseen, tai rajapinnaksi muille ohjelmistoille. Perinteinen SOA erittelee (kts. Kuva 1.1) osapuolet palvelun jakajaan, palvelua kutsuvaan osapuoleen (asiakas) sekä palvelurekisteriin jonka avulla pyydetty palvelu löydetään.

Palveluntarjoajan vastuulla on antaa palvelukuvaus ja ylläpitää palvelua, ja antaa palveluun liittyvää teknistä sekä muuta tukea. Palveluntarjoajan vastuulla on myös pitää huoli, että palvelu on löydettävissä palvelurekisterin avulla.

Palvelua kutsuvat tahot voivat olla muita ohjelmia, tai yksittäisiä tahoja, esimerkiksi kotipalvelin joka pyytää resurssia REST-rajapinnan (viite) toteuttavalta palvelulta internetin yli.

SOA:lle tyypillisessä skenaariossa palveluntarjoaja rakentaa toteutuksen palvelusta ja antaa sille palvelukuvauksen. Tämän jälkeen julkaisee sen palveluhakutaholle tai rekisterille, jota kautta palvelu on löydettävissä. Palvelua hakeva osapuoli pääsee käsiksi palveluun tällaisen tahon/rekisterin kautta (kuten esimerkiksi UDDI) ja linkittyy palvelukuvauksen avulla palveluntarjoajaan, jonka jälkeen kutsuva osapuoli pääsee kutsumaan palvelutoteutusta.

Palveluorientoituneet arkkitehtuurit toteuttavat SOC:in ideaa, mutteivat ota kantaa esimerkiksi turvallisuuteen, transaktioiden hallintaan tai koordinaatioon [viite papazoglouhun]. Tästä syystä onkin kehitelty xSOA (extended SOA) pyramidi [oma liite tähän].

4 Palveluorientoituneiden arkkitehtuurien haittapuole

5 Palveluiden haku

5.1 Yleisesti

Palveluntarjoajilla on hallussaan toteutus palvelusta, jota he ylläpitävät ja he myös määrittävät palvelukuvauksen tarjoamalleen palvelulle. Palvelukuvausten tarkoituksena on ilmoittaa palvelun tarkoituksesta, käyttäytymisestä, laadusta ja sen tulee muodostaa alusta löydettävyydelle, linkittymiselle ja palvelukompositiolle. Palvelut voivat tarvittaessa olla yhteydessä toisiinsa monimutkaisemmissa prosesseissa, kuten vaikkapa luottokortin tarkistuksessa tai tuotteen tilauksessa. Koska nämä palvelut ovat luonteeltaan alustariippumattomia, luo se puitteet kompositiopalveluiden tuottamiselle liittämällä yhteen toiminnaltaan yksinkertaisia tai monimutkaisempia palveluita.

Web service -malli koostuu kolmesta komponentista: asiakkaasta, palveluntarjoajasta sekä rekisteristä. Palveluiden haun kannalta nämä kaikki ovat oleellisia. Ilman rekisteriä kutsujaa ja tarjoajaa ei voida linkittää, koska yleensä asiakas ei tiedä missä palvelu sijaitsee. Palveluntarjoaja julkaisee palvelun rekisterissä luomalla WSDL (Web Service Description Language) määrittelyn, joka on XML-pohjainen rajapintakuvaus [viite: Subset WSDL to access Subset Service for Analysis Animesh Chaturvedi], joka kertoo kutsuvalle taholle miten pyydettyä palvelua voidaan kutsua, millaisia parametreja se odottaa, missä muodossa vastaus tulee ja millaisissa tietorakenteissa vastauksen sisältö on [viite: <http://www.w3.org/TR/wsdl20-primer>]. Perinteisesti nämä tahot ovat keskustelleet keskenään SOAP:n avulla. SOAP on protokolla, joka alkujaan oli akronyymi sanoista Simple Object Access Protocol. Myöhemmissä versioissa päätettiin ettei SOAP ole akronyymi ollenkaan.

SOAP luotiin toimimaan HTTP-protokollan yli, koska sitä tuetaan kaikissa internet selaimissa ja tarjoaa muodon, jolla XML viestit kääritään. Näin kaikki palvelut, jotka tukevat SOAP:ia voivat keskustella keskenään, eikä ole väliä miten palvelut ovat teknisesti toteutettu. Rekisterissä on siis WSDL-kuvaus palvelusta. Web service -mallissa asiakastaho tekee pyynnön rekisteriin, jonka jälkeen rekisteri vastaa kutsujalle lähettämällä palan WSDL-määrittelystä. Tämän jälkeen, mikäli palautettiin oikeanlaista tietoa, asiakkaalla on kaikki tarpeellinen tieto linkittyäkseen pal-

veluun. Saamansa WSDL:n avulla asiakas tekee pyynnön palveluntarjoajalle, joka antaa edelleen WSDL-määritellyssä muodossa vastauksen asiakkaalle. Kaikki nämä pyynnot palveluntarjoajan, rekisterin ja asiakkaan välillä ovat perinteisessä mallissa SOAP-muotoisia. Havainnollistus kuvassa 2.1.

Tällainen malli on hyvin perinteinen, oleellisina osina ovat WSDL, rekisterit ja SOAP viestit. On kehitetty myös muita tapoja keskustella palvelun ja asiakkaan välillä. Nykyään monet web-palvelut toteutetaan REST-tyyppisesti (citation needed). Ongelmaksi palvelun haun kannalta muodostuu se, ettei REST:ä ole standardoitu, eikä siis ole olemassa standardoitua tapaa miten palveluiden haku REST-tyyppisille rajapinnoilla toteutetaan. Normaalisti REST-tyyppisen palvelun haku perustuu sen URI:n. Esimerkiksi osoitteessa <http://yhtio.com/api/kayttajat/> voisi listata kyseisen palvelun käyttäjät. On täysin palveluntarjoajan vastuulla, että asiakkaat löytävät kyseiset resurssit. Hyvässä tapauksessa resurssien metadatatista löytyy tietoa siitä mitä voidaan tehdä seuraavaksi tai mitä muita resursseja voidaan hakea. REST ei siis varsinaisesti ota kantaa miten palvelun haku on toteutettu, eikä se ole lähtökohtaisesti osa REST:iä.

Erinäisiä lähestymistapoja yhdistää REST:iä traditionaaliseen malliin kuitenkin löytyy. Eräs ohjelmointi- ja palvelunkoostamistapa, jossa kaikki yksittäiset palvelut ovat täysin hajautettuja ja jotka keskustelevat keskenään RESTmäisesti, ilman keskinäisiä palvelumäärittelyksiä ja sopimuksia, on mikropalveluarkkitehtuuri. Mikropalveluarkkitehtuuri on nimitys, jonka puolestapuhuja esimerkiksi Martin Fowler on. Käytännössä mikropalvelut ovat SOA arkkitehtuurin mukaisia, mutta eräät kokivat SOA:n terminä liian laveaksi ja monimerkitykselliseksi. Mikropalveluihin ei oletuksena kuulu rekisteriä, joten nämä tarvitsevat oman tapansa linkittää palvelun kutsuja sekä palvelun tarjoaja. Tällaisia ratkaisuja voivat olla vaikkapa selainpuolen tai palvelinpuolen palvelun haku. Joka tapauksessa nämäkin ratkaisut päätyvät usein rekisterin käyttöön (esimerkiksi Netflix ja EUREKA viite). Ratkaisut loppuviimein ovat hyvin samanlaisia, erona on ettei REST palvelunhakua ole standardoitu ainakaan vielä samalla tavalla kuin Web Service maailmassa.

5.2 Erilaisia rekistereitä

Perinteisen mallin palvelut nojaavat yleensä erilaisiin rekistereihin. OASIS kehitti UDDI:n vuonna 2000 siltä ideapohjalta, että palvelua käyttävät tahot linkitettäisiin palvelun tarjoajiin julkisen tai yksityisen rekisterin avulla. Tällaisessa visiossa kuka tahansa esimerkiksi luottokorttitunnistusta tarvitseva asiakas tekisi pyynnön palve-

lurekisteriä vastaan ja valitsisi oikean SOAP:ia tukevan palvelun. Tällaisessa maailmassa julkisesti ylläpidettävä UDDI-rekisteri olisi kriittisen tärkeä kaikille. UDDI ei kuitenkaan päätenyt niin yleiseen käyttöön kuin sen julkaisivat olisivat toivoneet ja sen kehittäminen loppui vuonna 2007 [<https://lists.oasis-open.org/archives/uddi-spec/200807/msg00000.html>]. Se on kuitenkin edelleen osa Web Services Interoperability standardia ja tärkeä osa perinteistä Web Service palvelunhakumallia.

UDDI-rekisteri koostuu kolmesta osasta:

- *Valkoiset sivut* sisältävät tietoa palveluntarjoajasta. Tähän sisältyy esimerkiksi julkaisijan/yrityksen nimi ja kuvaus, mahdollisesti monella kielellä, ja palvelu voi olla löydettävissä pelkästään tämän tiedon avulla. Valkoiset sivut saattavat sisältää myös palveluntarjoajan yhteystiedot kuten puhelinnumero ja osoite.
- *Keltaiset sivut* luokittelevat palvelut standardoitujen taksonomioiden mukaan.¹ Yhteen valkoiseen sivuun voi liittyä useampi keltainen sivu, eli kuvaukset useammalle palvelulle.
- *Vihreät sivut* sisältävät teknistä tietoa miten palveluun pääsee käsiksi. Sisältävät siis palvelun osoitteen, parametrit ja viitteet vaadittuihin rajapintoihin tai protokoliin joilla palvelu toimii.

5.3 Palveluiden haun ongelmat

Haasteeksi perinteisessä WSDL:n, rekisteriin ja SOAP:aan keskittyvässä mallissa on tullut semanttisuuden puute [viite Bringing Semantics to Web Services with OWL-S + A Method for Semantically Enhancing the Service Discovery Capabilities of UDDI], sillä WSDL kertoo kyllä miten palvelua käytetään, mutta ei tarjoa ratkaisuja siihen mitä tapahtuu kun palvelua käytetään. Palvelua käyttävä ihminen lukee sen luonnollisella kielellä palvelukuvauksesta. Jos halutaan automatisoituja palveluja, jotka osaavat luotettavasti ratkaista ongelmia, tarvitaan tapa jättää ihminen välistä. Tarvitaan siis jokin ratkaisu tuottamaan semanttisia palvelukuvauksia. Palveluiden haku on luonteeltaan semanttista, mutta perinteinen UDDI, SOAP, WSDL perustuvan mallin rajoitteena on kertoa mitä kaikkea palvelu pystyy tekemään. Malli on luonteeltaan syntaksipohjainen, joka ei pysty itsessään ilmaisemaan koko kontekstia, jossa palvelut toimivat ja mitä ne pystyvät tekemään.

¹Standardoituja taksonomioita ovat esimerkiksi SIC (Standard Industrial Classification) ja NAICS (North American Industry Classification System)

UDDI:n avulla kyselyt palauttavat usein väärää palveluita, varsinkin kun rekisterien määrä kasvaa

Erilaisia lähestymistapoja asian korjaamiseksi on esitetty, ja ratkaisujen esittely löytyy kohdata "Semanttinen palveluiden haku".

5.4 Semanttinen palveluiden haku

Palveluiden haku semanttisen webin saralla on perustavanlaatuinen tutkimusongelma. Semanttiset palvelut ovat visio arkkitehtuurista, jossa käyttäjällä on mahdollisuus automaattisesti yhdistää tarvittavia palveluita toteuttaakseen jokin tietty tehtävä tai tehtäväkokonaisuus. Olkoon käyttäjä "Simo". Simo haluaa löytää halvan lennon määränpäähänsä josta haluaa jatkaa hotellille joka tyydyttää Simon tarpeet hinnaltaan ja sijainniltaan. Semanttisten web palveluiden visiona on yhdellä pyynnöllä yhdistellä palveluita, jotka antavat Simolle vastauksen haluttuun ongelmaan. Simo voisi hyvinkin olla myös muu e-Business palvelu, joka yhdistelee palveluita saadakseen ratkaisuja ongelmiinsa.

WSDL, SOAP ja UDDI standardit kärsivät semanttisen esityksen puutteesta, joten toive alan standardeilla koostettujen palveluiden automaattisesta integroimisesta jää täyttymättä. Jotta palveluista saataisiin kehitettyä semanttisia, on haasteita joihin täytyy löytää ratkaisu. Ensinnäkin kun palveluiden määrä lisääntyy merkittävästi, ja oikeiden palveluiden löytäminen on tärkeää, palvelunhaun tehokkuus ja tarkkuus tulee olla merkittävästi parempi kuin nyt. Toisekseen, kun palvelu saadaan tehokkaasti löydettyä, pitäisi pystyä automaattisesti linkittämään se kyseistä palvelua kutsuvaan palveluun. Molemmat näistä haasteista ovat riippuvaisia palveluntarjoajan kykyyn kuvata tarjoamansa palveluiden toiminnot sekä palveluita kutsuvien osapuolten kykyyn määritellä yksikäsitteisesti ja koneluettavasti tarpeensa palvelulta.

Eräs tapa lähestyä näitä haasteita, eli perinteisen mallin semanttisuuden lisäämistä on lisätä UDDI:in laajennus

6 Yhteenveto

Liite 1. Malli ABC

Liitteet ovat tässä vain sisällysluettelon ja esitystavan mallina. Jokainen liite aloitetaan yleensä uudelta sivulta, jonka alkuun tulee liitteen numero ja nimi. Kunkin liitteen sivut numeroidaan erikseen.

Liite on paitsi dokumenttia täydentävä osuus myös itsenäinen kokonaisuus. Liite ei siten voi olla pelkästään kuva tai ohjelmanpätkä, vaan liitteessä on ilmaistava sen sisällön laatu ja tarkoitus.