hyväksymispäivä arvosana

arvostelija

Aine: Service discovery and location transparency in Service-oriented computing

Toni Könnilä

Helsinki 19.10.2015

HELSINGIN YLIOPISTO
Tietojenkäsittelytieteen laitos

Sisältö

1 Abstrakti **1**

2 Johdanto **1**

3 Palveluorientoitunut arkkitehtuuri **2**

4 Palveluorientoituneiden arkkitehtuurien haittapuolet **4**

5 Palveluiden haku ja läpinäkyvyys **4**

6 Yhteenveto **4**

Lähteet **4**

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 arkkitehtuureiden 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 mitä vain) mahdollistavat ohjelmallisen pääsyn palveluihinsa ja dataansa internetin yli käyttäen standardoituja välitystapoja, käyttäen yleensä XML-perustaisia tai REST-mallin mukaista 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 määritykset ja varsinaiset sijaintitiedot tulee olla tallennettuna johonkin - esimerkiksi UDDI[?] repositorioon, ja asiakkailla/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 yksittinen palvelu eroaa ajatusmaailmaltaan monoliittisen arkkitehtuuripuolen 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 rajotteisiin. Ohjelmat ovat tiiviisti linkattuina muihin kirjastoihin, ja yksittäisten luokkien ja palveluiden korvaaminen päätyy 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 haittapuolet

5 Palveluiden haku ja läpinäkyvyys

6 Yhteenveto

Lähteet

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.