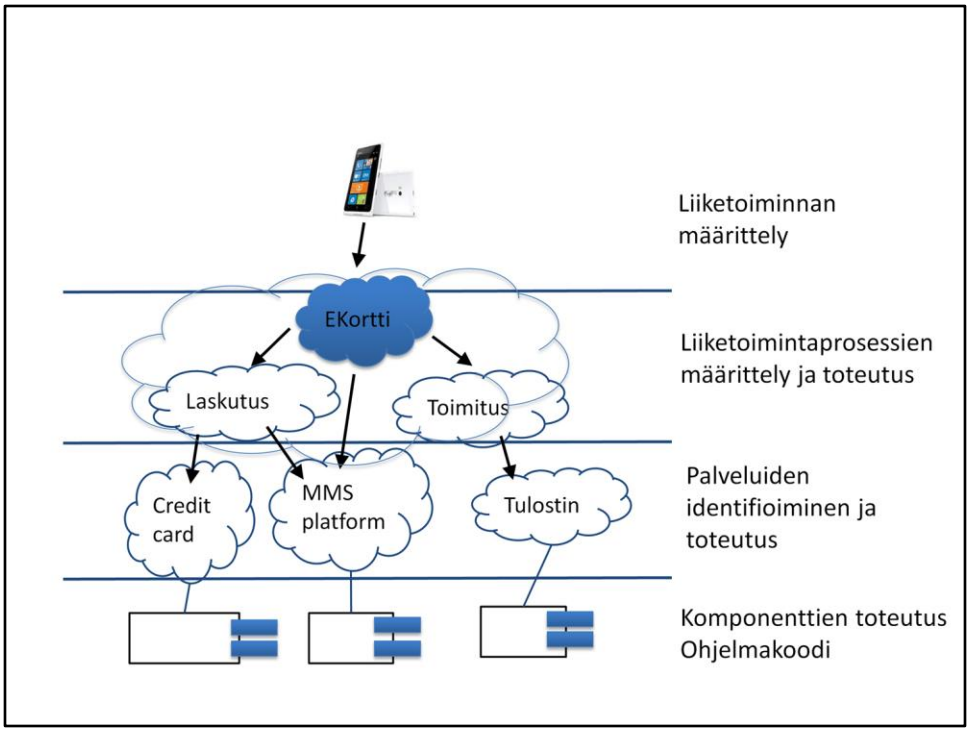
Designing and realizing SOA

TIE-23600 Palvelupohjaiset järjestelmät


EKortti
Laskutus
Toimitus
Credit card
MMS platform
Tulostin
Liiketoiminnan määrittely
Liiketoimintaprosessien määrittely ja toteutus
Palveluiden identifioiminen ja toteutus
Komponenttien toteutus Ohjelmakoodi

Top-down approach

- Design of services and processes
  - Ideally without considering the existing code base
  - i.e. what would be needed
- Implementation of the required services
  - Match with existing code base
    - Modify/use existing services and use wrapping techniques
    - Build new services
- A ”business driven approach”

Edellä esitetty tapa toteuttaa palvelupohjaisia järjestelmiä edustaa nk. ”top-down” –lähestymistapaa. Oleellisesti siinä siis edetään systemaattisesti abstrakteimmalta tasolla tarkentaen yhä yksityiskohtaisemmalle ja tarkemmalle tasolle päätyen lopulta itse toteutukseen. Ideaalisessa tapauksessa suunnittelu alkaa liiketoiminnan tasolta eikä liiketoimintaprosesseja eikä teknisiä realiteetteja vielä edes huomioida tuossa vaiheessa. Kun liiketoimintaprosessit on määritelty, tulee niiden toteuttaminen suunnitella teknisinä prosesseina. Nämä tekniset prosessit voidaan sitten toteuttaa osin hyödyntäen olemassa olevia järjestelmiä ja toisaalta toteuttamalla uusia palveluita tarvittaessa. Tätä lähestymistapaa kutsutaan myös ”liiketoimintaorientoituneeksi” lähestymistavaksi.

Bottom-up approach

- Identification of services and processes from existing code
- Using e.g. wrapping techniques to implement proper service interfaces
- Goals
  - to extend the functionality of a legacy system
  - to migrate a legacy system for a new environment
  - For software integration and/or interoperability reasons
  - Etc.
- An ”IT driven approach”

Edelliselle ”top-down” –lähestymistavalle vastakkainen vaihtoehto on nk. ”bottom-up” –lähestymistapa. Siinä lähdetään nimenomaan olemassa olevista teknisistä valmiuksista. Aluksi pyritään identifioimaan tarvittavat palvelut ja prosessit koodista. Hyödyntämällä erilaisia käärimistekniikoita ja takaisinmallinnustekniikoita (reverse engineering techniques), olemassa oleva koodi muunnetaan palvelupohjaiseksi.

Tätä toiselta nimeltään ”IT-lähtöistä” lähestymistapaa käytetään paljon. Sitä hyödynnetään mm. kun halutaan laajentaa legacy-järjestelmien toiminnallisuutta, integroida ne muiden järjestelmien kanssa, käyttää niitä uudessa ympäristössä jne. Toisaalta tässä lähestymistavassa on myös selkeät heikkoutensa. Suurin niistä lienee se, että muodostettu palvelujoukko on harvoin ideaalinen ja aidosti ”SOA-hengen mukainen”.

Meet-in-the-middle approach

- In practice, approach is rarely neither strictly top-down nor strictly bottom-up
- Strict top-down: may lead to services that are single-purpose, nonreusable and difficult to implement
- Strict bottom-up: may lead to services that are not needed or don’t fulfill business requirements

G. Coticchia, Seven Steps to a Successful SOA Implementation. *Business Integration Journal,*10, 5, 2006, http://www.bijonline.com/, pp. 10-13. :

> *"…Business services cannot be developed bottom-up, ad hoc, in other words driven by immediate project needs and implementation details instead of long-time business needs. But one-step top-down approach is not either the only way to go, as it tends to ignore the details and make false assumptions on the cost and scheduling of implementations. Instead, collaborative, iterative top-down mechanism should be used."*

Käytäntö on osoittanut, että ehkä paras edetä olisi yhdistää top-down ja bottom-up –lähestymistapoja. Bottom-up –lähestymistapa saattaa johtaa joustamattomaan palvelujoukkoon eikä huomioi liiketoimintatarpeita. Top-down –lähestymistapa puolestaan saattaa johtaa siihen, ettei teknisiä realiteetteja oteta oikealla tavalla huomioon. Esimerkkeinä voivat olla väärät oletukset toteutuksen kompleksisuudesta, toteutukseen tarvittavasta ajasta jne. Coticchia (2006) on todennut tutkittuaan useaa käytännön projektia, että nk. "meet-in-the-middle" olisi tässäkin tapauksessa paras vaihtoehto. Hän toteaa, että suunnittelun tulisi alkaa aidosti liiketoiminnan näkökulmasta, mutta tekniset realiteetit tulisi huomioida jo melko aikaisessa vaiheessa. Tämä lisäisi myös kommunikaatiota liiketoiminta- ja IT-vastuullisten kesken.

Service granularity

- Granularity layers
  - Typically, SOA systems form a hierarchy, such that higher level services make use of lower level services
  - Services at the lowest level are fine grained providing resources and infrastructure services
  - High level services are coarse grained business services used by the business processes and end users
- Service granularity
  - The service size and the scope of functionality provided by a service within one interaction
  - The services should have the right granularity to accomplish a proper unit of work and to enable service reusability and composability

Service Layers

- Utility services
  - Non-business-centric functionality
- Entity services
  - ”Functional context derived from business entities”
    - Eg. Customer, Order, Invoice, …
- Task services
  - ”Functional context based on a specific business process”
  - Not very reusable

Yksi tapa hahmottaa paremmin sitä, millaisia palvelut ovat on jakaa niitä kerroksiin. Tässä yksi jakotapa.

”Apupalvelu” (utility service) tarjoaa toimintoja, joilla ei ole suoraa linkkiä liiketoimintavaatimuksiin. Muut palvelut ovat apupalveluiden asiakkaita.

”Entiteettipalvelu” (entity service) tarjoaa toimintoja, jonkin ”entiteetin” käsittelyyn. Tämä entiteetti on sellainen, että se on käsitteenä olemassa ohjelmistomaailman ulkopuolellakin. Esim. Asiakas, Tilaus, Lasku, …

”Tehtäväpalvelu” (task service) kuvaa jonkin liiketoimintaprosessin tai sen osan. Käyttää hyväkseen apu- ja entiteettipalveluita. Esim. palvelu joka luo laskutusraportin.

http://soapatterns.org/design_patterns/service_layers

http://serviceorientation.com/soaglossary/entity_service
http://serviceorientation.com/soaglossary/utility_service
http://serviceorientation.com/soaglossary/task_service

Another way to classify services

- Basic services
  - Basic data services
    - Often have ACID properties
  - Basic logic services
- Composed services
  - Use basic services and other composed services
- Process services
  - Present long-term workflows or business processes

Nicolai M. Josuttis: SOA in Practice

Service-Oriented Modeling and Architecture (SOMA)

- A method to model and design SOA, proposed by IBM
- Implements IBM's other method, service-oriented analysis and design (SOAD), through
  - identification, specification and realization of the three main elements of SOA:
    - services,
    - components that realize those services (a.k.a. "service components"), and
    - flows that can be used to compose services
- More information
  - A. Arsanjani, Service-Oriented Modeling and Architecture: How to Identify, Specify and Realize Services for your SOA, IBM developerWorks, 2004.
  - N. Bierberstein, S. Bose, M. Fiammante, K. Jones, and R. Shah, Service-Oriented Architecture (SOA) Compass: Business Value, Planning, and Enterprise Roadmap, IBM Press, ISBN-13: 978-0-13-187002-4, Oct 2005.
  - A. Asanajani, L.-J. Zhang, M Ellis, A. Allam, and K. Channabasavaiah, S3: A Serivce-Oriented Reference Architecture, IT Professional, IEEE Computer Society, May/June 2007.

IBM:n SOMA on menetelmä, jonka tarkoituksena on tukea SOA-pohjaisten järjestelmien suunnittelua, mallintamista ja toteuttamista. Koska se on yksi ensimmäisistä ja ehkä myös laajimmin tunnetuista suunnittelumenetelmistä ja koska tällaisten menetelmien tarve (ja niiden puute) on laajalti havaittu, käsitellään SOMAa seuraavaksi esimerkinomaisesti. On kuitenkin syytä korostaa, että SOMAa ei menetelmänä voida missään nimessä kutsua (edes de facto) standardiksi.

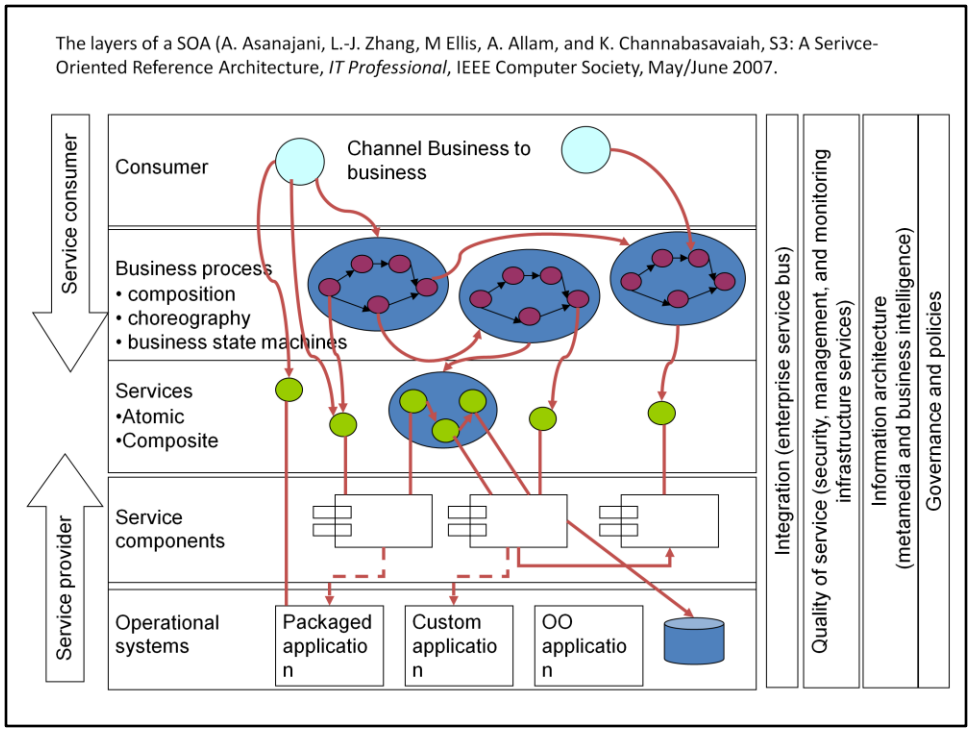
SOMA toteuttaa IBM:n toisen menetelmän SOAD, joka on kehitetty palveluorientoituneiden järjestelmien analyysi- ja suunnittelumenetelmäksi. Tämä menetelmä pohjautuu perinteisiin olio- ja komponenttipohjaisten järjestelmien analyysi- ja suunnittelumenetelmiin ja laajentaa niitä SOAn kannalta oleellisilla näkökulmilla. SOMA koostuu kolmesta päävaiheesta: SOAn peruselementtien *tunnistaminen*, *spesifiointi* ja *realisointi*. Nämä peruselementit ovat *palvelut*, palvelut realisoivat *palvelukomponentit* (service components) sekä palveluiden yhdistämiseen käytettävät *vuot*.

SOMA - Designing SOA systems

- ”The design strategy for a SOA does not start from the “bottom-up" as is often the case with a Web services-based approach. You must remember that SOA is more strategic and business-aligned.” (Ali Arsanjani, Service-oriented modeling and architecture, IBM developerWorks, 2004).
- Activities of a service consumer (i.e. a user of a service, which can be a client or a service):
  - Service identification, service categorization, choreography or composition, quality of service (QoS)
- Activities of a service provider:
  - Component identification, component specification, service realization, service management, standards implementation, service allocations to components, layering the SOA, technical prototyping, product selection, architectural decisions (state, flow, dependencies)

A. Arsanjani korostaa artikkelissaan ”Service-oriented modeling and architecture”, IBM developerWorks, 2004, että suunniteltaessa SOA-pohjaisia järjestelmiä tulisi välttää ”bottom-up” –tyyppistä lähestymistapaa, jossa siis lähdetään yksityiskohdista (esim. olemassa olevasta koodista) ja valitaan/identifioidaan palvelut näiden yksityiskohtien perusteella tai johdattamana. Tällainen lähestymistapa harvoin johtaa parhaaseen mahdolliseen lopputulokseen. Arsanjani tähdentää, että sen sijaan tulisi muistaa, että SOAlla on strategista merkitystä ja sen tulisi huomioida myös liiketoimintaorientoitunut näkökulma. Tällöin siis ”top-down” tai mahdollisesti näiden kahden lähestymistavan (top-down ja bottom-up) yhdistelmä olisi parempi. Arsanjani sanoo edelleen, että ”bottom-up” –lähestymistapaa käytetään paljon Web-palvelujärjestelmiä toteutettaessa. Kuten Arsanjani edelleen toteaa, ”bottom-up” –lähestymistapaa käytetään paljon Web-palvelujärjestelmiä toteutettaessa (service mining, software migration). Se ei silti liene se oikea tai paras lähestymistapa myöskään Web-palvelujärjestelmiä – jotka myös ovat SOA-pohjaisia – toteutettaessa.

Kalvolla on listattu SOA-pohjaisen järjestelmän suunnittelussa huomioitavat palvelun käyttäjän (consumer) ja palvelun tarjoajan (provider) tehtävät ja roolit. Tässä palvelun käyttäjä (consumer) on erotettu termistä ”asiakas” (client). Käyttäjä on palvelua käyttävä taho ja se voi olla joko asiakas tai palvelu. Huomaa, että palvelun käyttäjän aktiviteetit muodostavat alijoukon palvelun tarjoajan aktiviteeteista. Palvelun tarjoaja esimerkiksi myös tunnistaa, kategorisoi jne. palveluita. Palvelun käyttäjä spesifioi ensin ne palvelut, joita se tarvitsee (tyypillisesti etsimällä niitä tietyin kriteerein) ja varmistuttuaan sen jälkeen siitä, että etsittyjen palvelujen (kriteerit) ja toisaalta tarjotun/löydetyn palvelun spesifikaatiot vastaavat toisiaan (vaaditulla tavalla), se ottaa yhteyttä ko. palveluun. Palvelun tarjoajan puolestaan tulee julkaista tarjoamansa palvelut, sekä toiminnallisuuden että laatuattribuuttien (QoS) osalta. Tämä implisiittinen sopimus palvelun tarjoajan ja käyttäjän välillä voi mahdollisesti kypsyä eksplisiittiseksi SLA-sopimukseksi, joka on neuvoteltavissa joko elektronisesti (esim. ebXML, jota käsitellään myöhemmin) tai muutoin.



Kalvon SOAn kerroksia havainnollista kuva on piirretty mukaellen kuvaa, joka on esitetty artikkelissa A. Asanajani, L.-J. Zhang, M Ellis, A. Allam, and K. Channabasavaiah, S3: A Serivce-Oriented Reference Architecture, IT Professional, IEEE Computer Society, May/June 2007. Kirjoittajat toteavat, että kuvan yhdeksän kerrosta ovat suhteellisen itsenäisiä, minkä vuoksi organisaatio voi valita palvelun tarjoajan ja käyttäjän integraation asteen. Esimerkiksi liiketointaprosessikerros ei välttämättä esiinny SOA-ratkaisussa. Tällöin palvelun käyttäjä ja tarjoaja voivat kommunikoida suoraan.

Operatiivisten järjestelmien kerros (Operational systems) sisältää kaikki ne käytettävät sovellukset ja/tai niiden osat, jotka tukevat liiketoiminta-aktiviteetteja. Näihin sovelluksiin kuuluvat esimerkiksi legacy-järjestelmät, tietokannat ja vaikkapa pakatut sovellukset ja ratkaisut kuten ERP- tai CRM-ratkaisut. Palvelukerros (Services) puolestaan sisältää kaikki SOAan kuuluvat palvelut: sekä sellaiset palvelut, jotka palvelun tarjoaja tarjoaa, että käytettävät palvelut. Jokaisesta palvelusta tulee olla tieto tarjottavista operaatioista, kontaktipisteestä, kutsuprotokollan yksityiskohdista sekä palvelun semantiikasta (esim. liiketoimintakonteksti). Web-palvelujen tapauksessa kolme ensimmäistä saadaan WSDL-dokumentista. Seuraavassa liiketoimintaprosessikerroksessa (Business process) organisaatio koostaa palvelukerroksen palveluista prosesseja. Lopuksi kuluttajakerroksessa (Consumer) hoidetaan interaktiot käyttäjän tai muiden SOA-ekosysteemiin kuuluvien ohjelmien kanssa. Tämän kerroksen kautta organisaatio voi toisaalta välittää dataa sovelluksille ja käyttäjille tiettyjen preferenssien mukaan ja toisaalta se voi nopeasti luoda liiketoimintaprosesseille ja sovelluksille front-end-liitännän, jonka avulla se voi mukautua erilaisiin ulkopuolisiin muutoksiin.

Edellä mainittujen kerrosten lisäksi Asanajari *et al. m*äärittävät neljä muuta edellisiä kerroksia hallinnoivia ja niihin vaikuttavia kerroksia. Integraatiokerros (Integration) integroi käytännössä kerrokset 2-4 (Service components, Services, Business processes). Integrointi voidaan toteuttaa vaikkapa ESB-ratkaisuna. Tämän kerroksen avulla organisaatio voi reitittää, välittää ja kuljettaa palvelukutsuja palvelun pyytäjältä konkreettiselle palvelulle. Laatuattribuuttikerros (Quality of service, QoS) huolehtii ei-funktionaalisista vaatimuksista liittyen luotettavuuteen, saavutettavuuteen, hallittavuuteen, skaalautuvuuteen ja turvallisuuteen. Informaatioarkkitehtuurikerros (Information architecture) voi esimerkiksi sisältää tietoa ja/tai referenssejä koskien teollisuuden alalle spesifejä tai teollisuusalojen välisiä tietorakakenteita ja XML-pohjaisia metadata-arkkitehtuureja (XML Schema). Tämä kerros sisältää SOA-ekosysteemissä tarvittavan metadatan. Viimeisenä kerroksena Asanajani *et al.* esittävät hallintakerroksen (Governance and policies). Tämä kerros on vastuussa SOA-ekosysteemin hallinnoinnista koko liiketoimintaoperaatioiden elinkaaren ajan. Tämä kerros määrittää myös periaatteet ja säännöt SLA-sopimuksille (esim. kapasiteetti, tehokkuus, turvallisuus). Hallinnoinnin merkitys SOA-pohjaisille ratkaisuille onkin viime aikoina saanut paljon huomiota. Kuten kaikille järjestelmille, se on tärkeää myös SOA-pohjaisille ratkaisuille.