Servisno orijentirana arhitektura za Internet stvari

Darko Andročec

UNIVERSITY OF ZAGREB
FACULTY OF ORGANIZATION AND INFORMATICS
VARAŽDIN

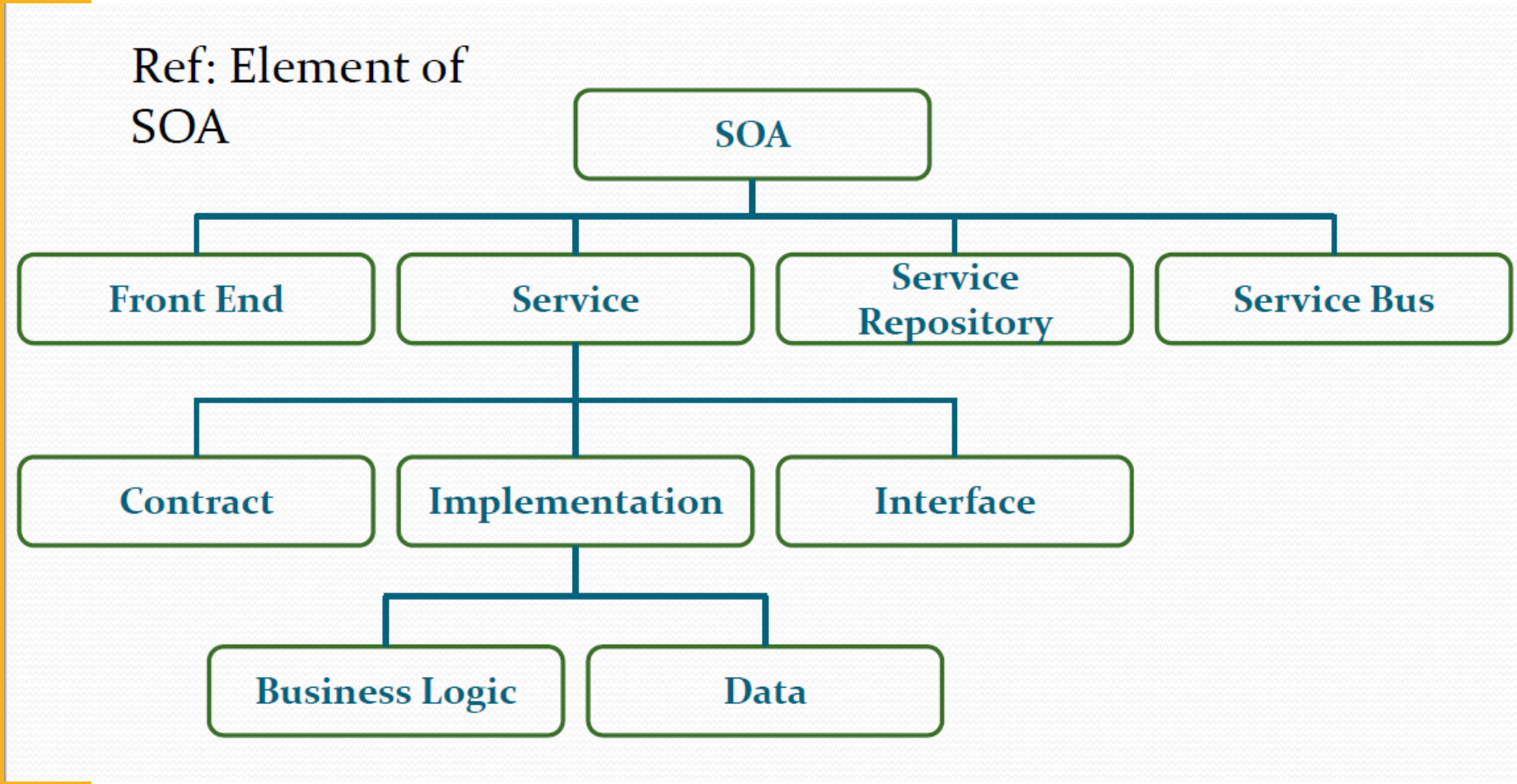
Definicija SOA

- Sommerville - Servisno orijentirane arhitekture (SOA) način su razvoja distribuiranih sustava gdje su komponente sustava samostalne usluge koje se izvode na geografski distribuiranim računalima.
- OASIS - Paradigma za organiziranje i korištenje distribuiranih mogućnosti koje mogu biti pod kontrolom različitih vlasničkih domena. Pruža jednoobrazno sredstvo za ponudu, otkrivanje, interakciju i korištenje mogućnosti za postizanje željenih učinaka u skladu s mjerljivim preduvjetima i očekivanjima.

Glavne komponente

- Servis - labavo povezana softverska komponenta za višekratnu upotrebu koja sadrži diskretnu funkcionalnost, koja se može distribuirati i kojoj se može programski pristupiti.
- Konektori – poruke, metapodaci (opis servisa, sučelje servisa, semantički metapodaci)

Sve komponente SOA-e



Web servisi su

- Tehnologija
- Middleware
- Implementacija SOA-e

Arhitekturni principi (1)

- Labavo povezivanje - servisi održavaju odnos koji smanjuje ovisnost i samo održavaju svijest jedan o drugome
- Ugovor o usluzi - servisi se pridržavaju komunikacijskog sporazuma kako je zajednički definirano jednim ili više dokumenata s opisom usluge
- Apstrakcija usluge - Osim onoga što je opisano u ugovoru o usluzi, servisi skrivaju logiku od vanjskog svijeta
- Ponovna upotreba usluge - Logika je podijeljena na servise s namjerom promicanja ponovne upotrebe

Arhitekturni principi (2)

- Kompozitivnost servisa - Zbirke servisa mogu se koordinirati i sastaviti u složene servise
- Autonomija usluge – Servisi imaju kontrolu nad logikom koju enkapsuliraju
- Optimizacija servisa –visokokvalitetne usluge općenito se smatraju boljim od onih niske kvalitete
- Mogućnost otkrivanja usluge - usluge su osmišljene tako da budu opisane prema van kako bi se mogle pronaći i procijeniti putem dostupnih mehanizama otkrivanja

Poslovne dobrobiti kod korištenja SOA-e

- Smanjeni troškovi
- Povećanje produktivnosti zaposlenika
- Izrađene za integraciju i partnerstva
- Agilnost – izgrađena za promjene

Tehničke dobrobiti kod korištenja SOA-e

- Kreriranje skalabilnih sustava koji mogu evoluirati
- Bolje upravljanje kompleksnim sustavima
- Neovisnost o pojedinoj platformi
- Labavo povezivanje omogućuje fleksibilnost

Izazovi kod SOA-e

- Upravljanje servisima – premala pažnja upravljanju servisa može prouzročiti probleme s performansama i pouzdanošću
- Pružanje odgovarajuće sigurnosti za pojedine uloge
- Osiguranje interoperabilnosti servisa

Kada ne koristiti SOA pristup?

- U homogenim IT okruženjima
- Kada je kritična performansa u realnom vremenu
- Kada je poželjno čvrsto povezivanje (tight coupling)
- Kada se stvari ne mijenjaju

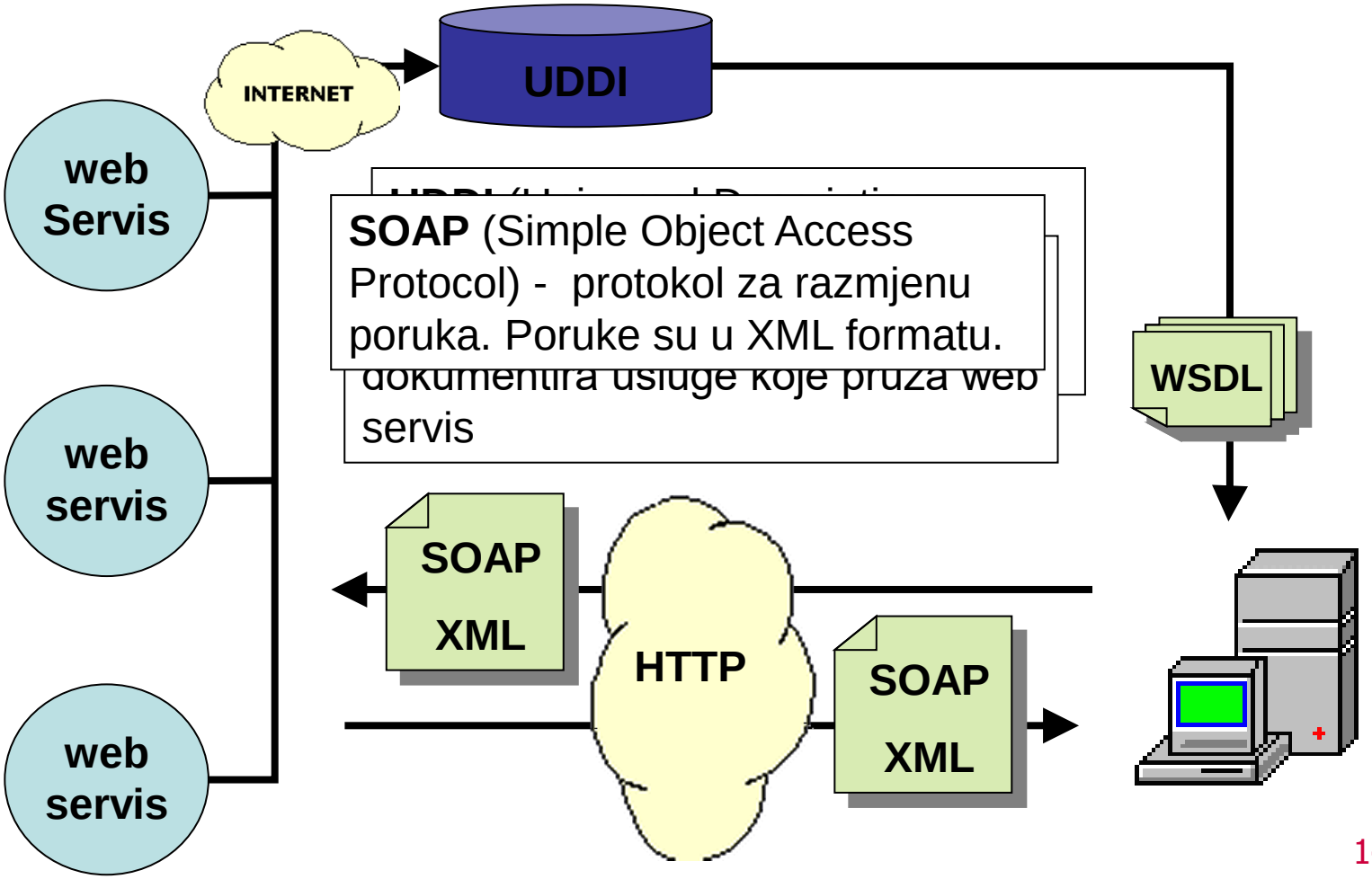
Internet u budućnosti

- omogućuje razmjenu podataka između aplikacija i web portala
- podaci se ne prepisuju, nego se dohvaćaju direktno s mjesta nastanka
- umjesto zamornog traženja, korisnik sve informacije o traženom sadržaju dobiva na jednom mjestu

Web servisi (1)

- softverska aplikacija koja nije vezana za operacijski sustav ili programski jezik
- distribuirana okruženja - Internet ili intranet
- standardizirani sustav za razmjenu poruka
- razmjena podataka
- obavljanje poslovnih transakcija

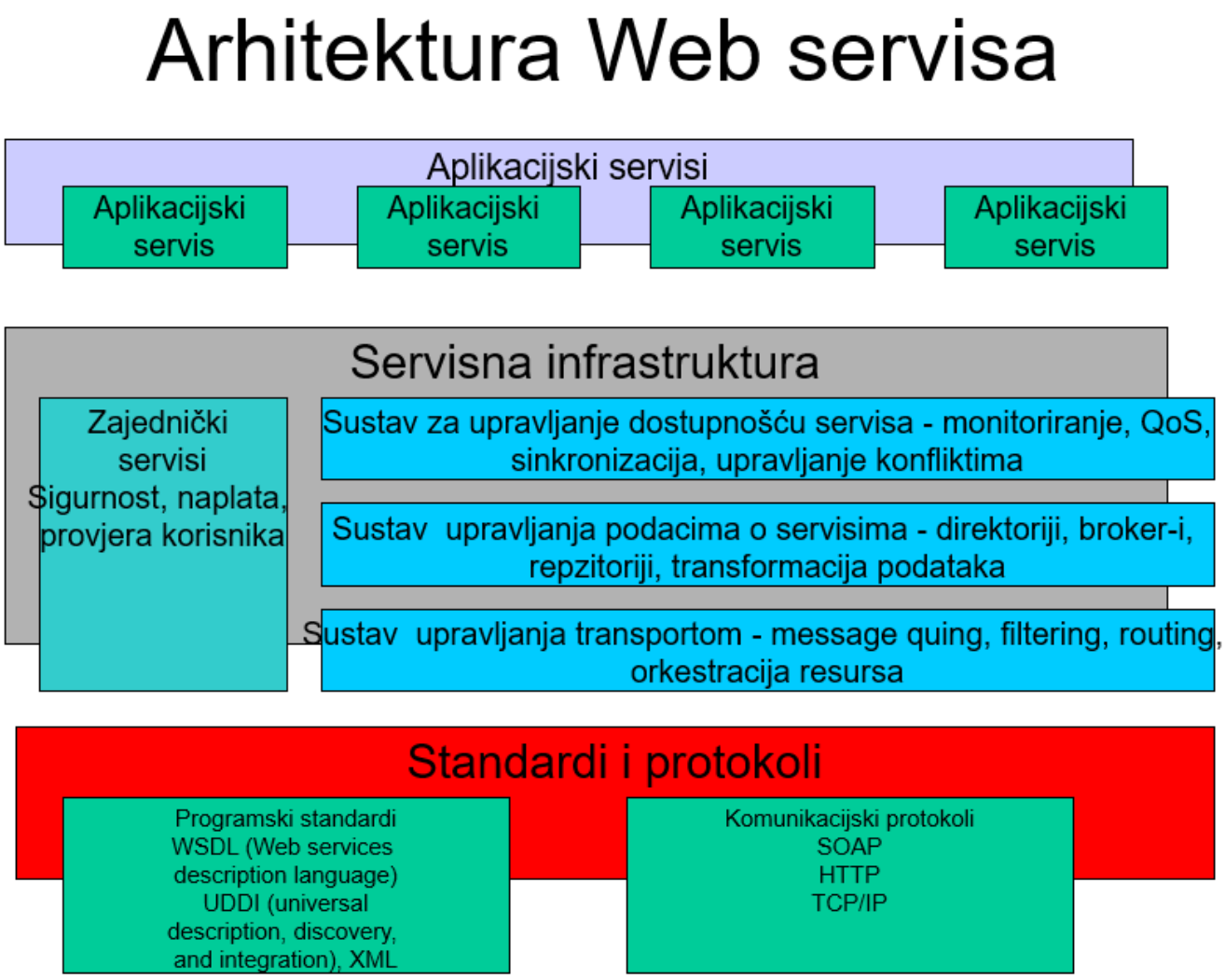
Web servisi (2)



SOAP (Simple Object Access Protocol) - protokol za razmjenu poruka. Poruke su u XML formatu.

dokumentira usluge koje pruža web servis

Arhitektura web servisa



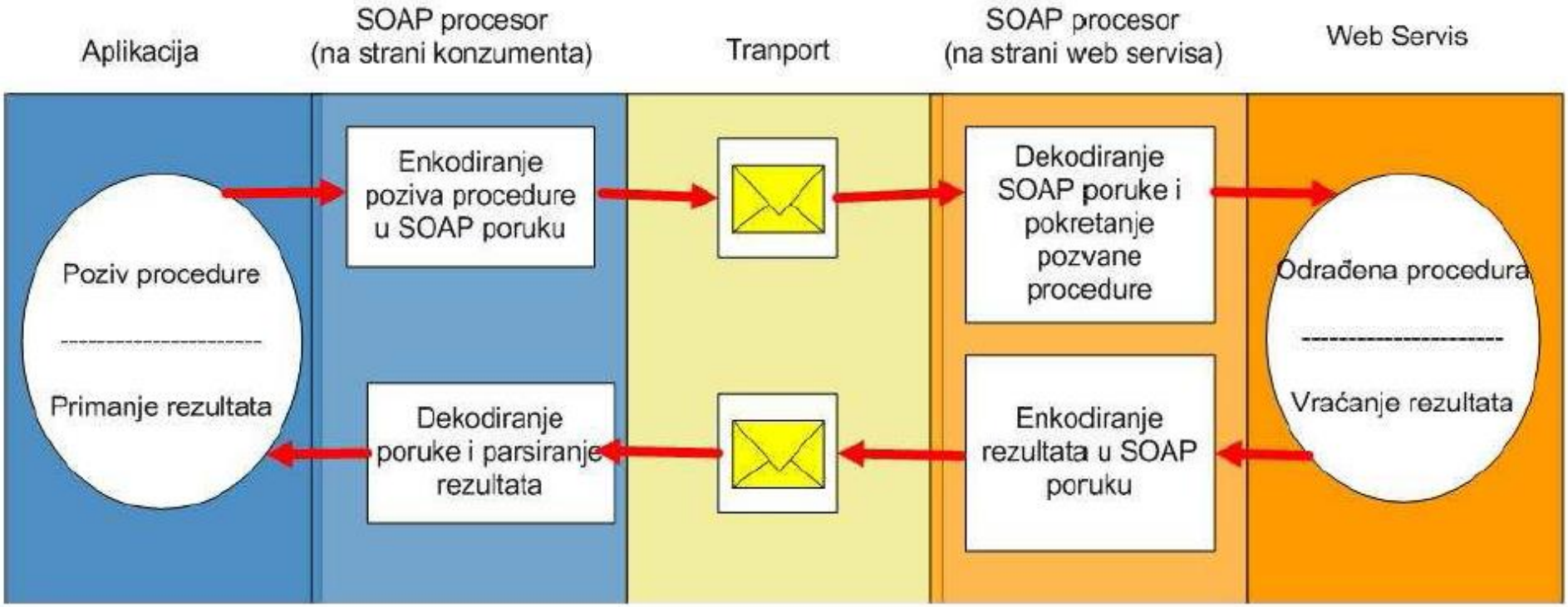
Grupa protokola

- XML (eXtensible Markup Language)
- SOAP (Simple Object Access Protocol)
- WSDL (Web Service Description Languages)
- UDDI (Universal Description, Discovery and Integration)

SOAP (1)

- W3C XML bazirani komunikacijski protokol za razmjenu poruka između aplikacija
- ima sve prednosti XML-a
- komunikacija između različitih aplikacija dostupnih putem Internet protokola

SOAP (2)


Aplikacija
SOAP procesor
(na strani konzumenta)
Tranport
SOAP procesor
(na strani web servisa)
Web Servis
Poziv procedure

Primanje rezultata
Enkodiranje
poziva procedure
u SOAP poruku
Dekodiranje
poruke i parsiranje
rezultata
Dekodiranje
SOAP poruke i
pokretanje
pozvane
procedure
Enkodiranje
rezultata u SOAP
poruku
Odrađena procedura

Vraćanje rezultata

Sintaksa SOAP-a

- XML dokumenti strukturirani po dodatnom skupu pravila
- SOAP Envelope element - root element
- SOAP Body element - nositelj poruke
- SOAP Header element - dodatni podaci
- SOAP Fault element - poruke o greškama

WSDL

- standard za dokumentiranje usluga koje pruža web servis
- notacija je bazirana na standardnoj XML shemi
- u jednoj XML datoteci definira sve što je potrebno za pisanje korisničkog softvera koji koristi usluge Web servisa

UDDI

- implementacija UDDI specifikacije je UDDI Business Registry
- bijele stranice - opisuju pružatelja usluge
- žute stranice - industrijske kategorije
- zelene stranice - opisuju sučelje servisa (WSDL)

Interoperabilnost

- **WS-I** (Web Services Interoperability Organization) - http://www.ws-i.org/ - sada dio OASIS-a
- promicanje novih standarda potrebnih da koncept Web servisa zaživi u potpunosti
- **WS-* specifikacije** (WS-Security, WS-Policy, WS-SecurityPolicy, WS-Trust, WS-SecureConversation, WS-Privacy, itd.)

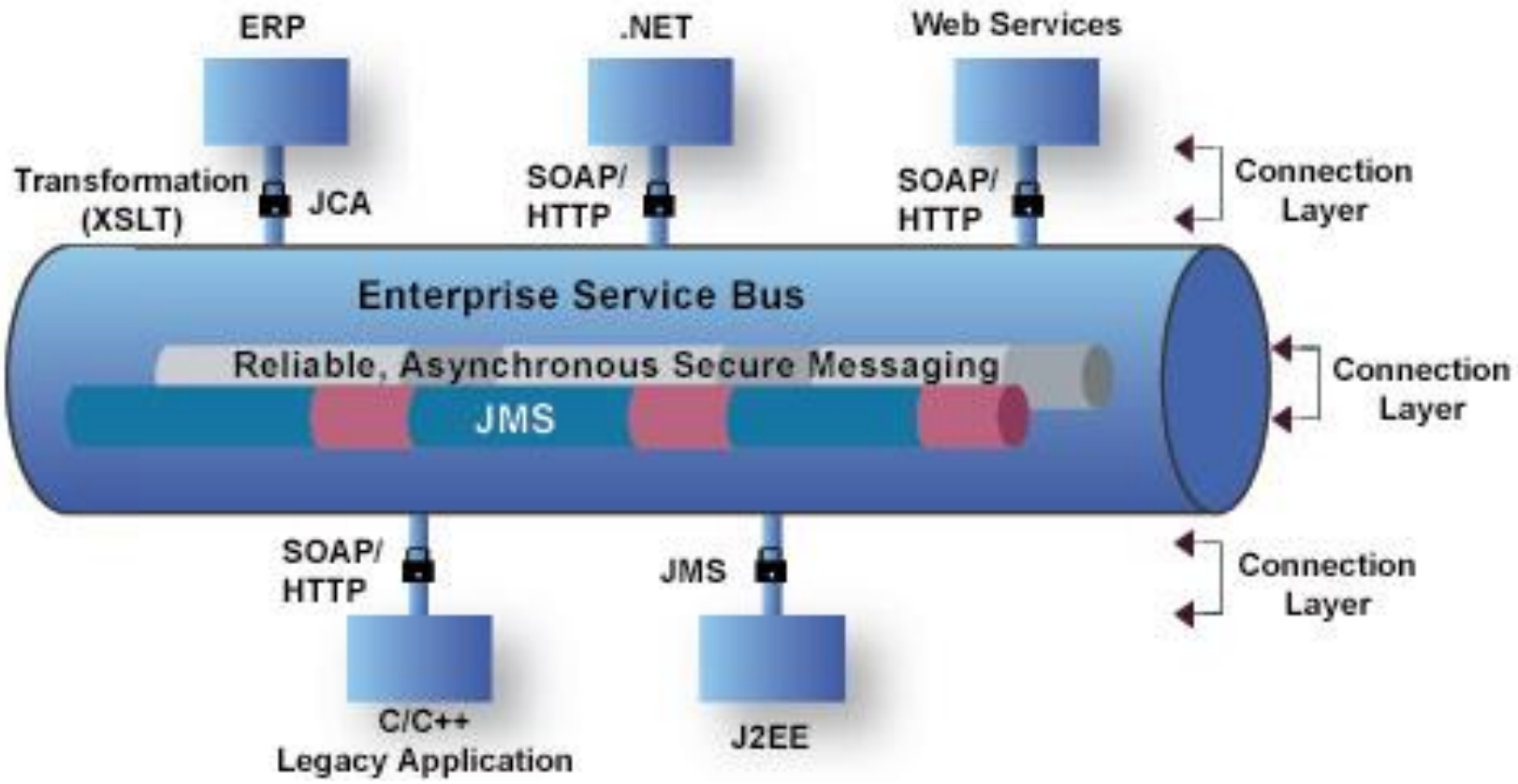
ESB za integraciju (1)

- Enterprise Service Bus
- Skup kontejnera servisa koji objedinjavaju različite vrste IT resursa
- Kontejneri su međusobno povezani sustavom razmjene poruka
- Kontejneri prilagođavaju IT resurse prema standardiziranom servisnom modelu (komunikacija XML porukama)

ESB za integraciju (2)

- Središnja administracija i nadzor distribuiranog sustava upravljanjem i razmjenom poruka (message transformation services, message routing services)
- Omogućuje ad-hoc promjene u sustavu, bez potrebe za narušavanjem funkcionalnosti sustava
- Middleware
- Povezivanje podsustava i aplikacija
- Posredovanje – usmjeravanje, modeli interakcije, upravljanje verzijama
- Upravljanje i nadzor – upravljanje porukama, transakcijma, sigurnošću, QoS-om…

ESB


ERP
.NET
Web Services
Transformation
(XSLT)
JCA
SOAP/
HTTP
SOAP/
HTTP
Connection
Layer
Enterprise Service Bus
Reliable, Asynchronous Secure Messaging
JMS
Connection
Layer
SOAP/
HTTP
JMS
Connection
Layer
C/C++
Legacy Application
J2EE

WS-BPEL (1)

- Specifikacija OASIS-a
- WS-BPEL (engl. Web Services – Business Process Execution Language)
- Definira jezik za izgradnju poslovnih procesa zasnovanih na web servisima
- WS-BPEL (tj. njezina prva verzija BPEL4WS) nastala je 2002. godine kao rezultat zajedničkog rada tvrtki IBM, Microsoft i BEA
- U razvoj jezika priključile su se i neke druge softverske tvrtke kao što su SAP i Siebel Systems.
- Specifikacija je u 2007. predana organizaciji OASIS kako bi postala službena, otvorena norma

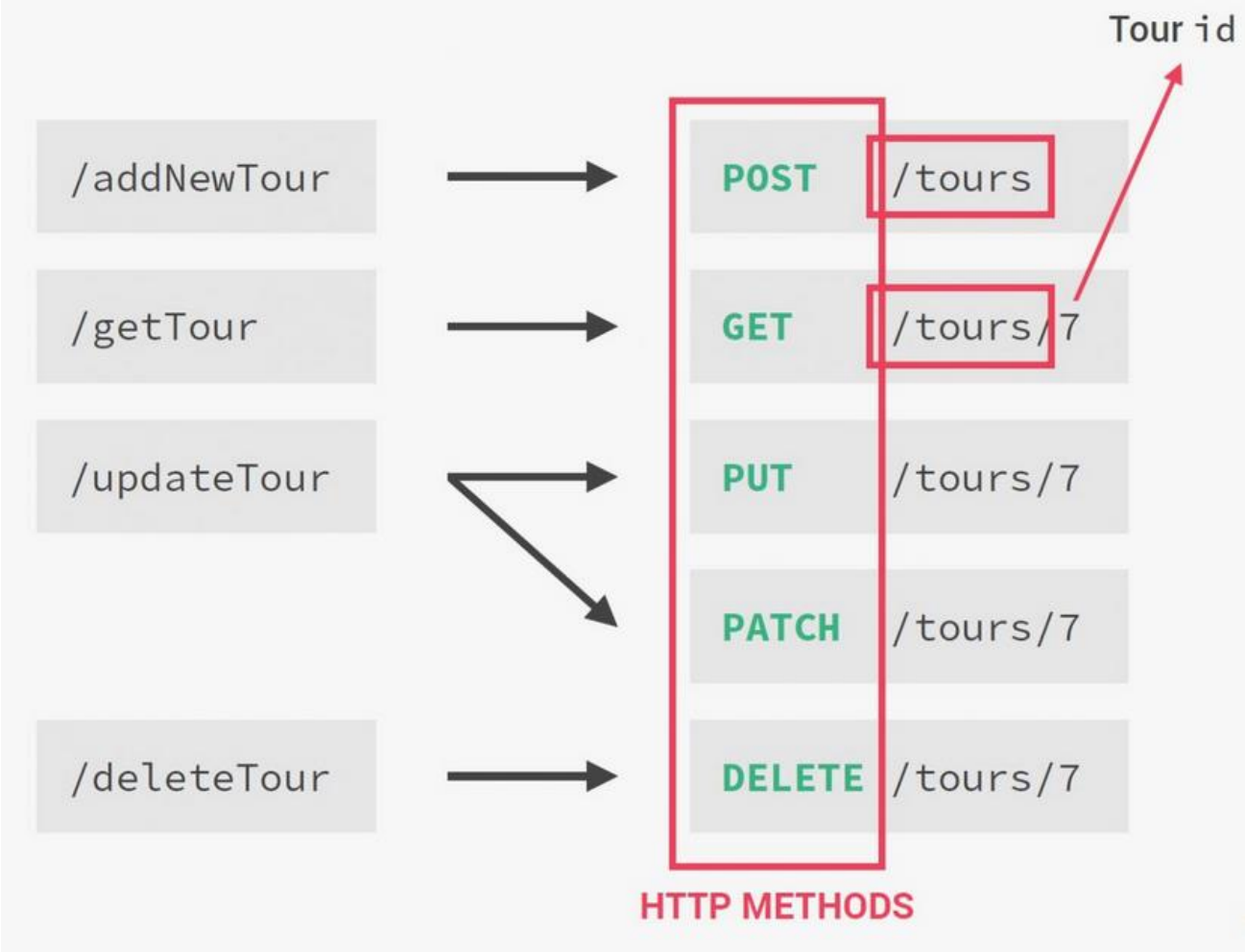
WS-BPEL (2)

- WS-BPEL koristi već postojeće mehanizme specifikacije web servisa– poglavito normu XML te normu za opis Usluga weba WSDL.
- Glavna ideja jest ta da se stvori jedna upravljačka usluga – tzv. poslužitelj BPEL procesa – koja na osnovi danog XML-ovskog predloška ima odgovornost provođenja poslovnog procesa.
- Upravljačka usluga naziva se BPEL-ovskim procesorom te ima svoj vlastiti WSDL-ovski opis te je i sama po svim karakteristikama jedan klasični web servis.

REST servisi

- REST = Representational State Trasfer
- Jednostavniji, više integrirani sa HTTP-om
- Ne zahtijevaju XML poruke niti WSDL opise servisa
- Danas se češće koristi nego SOAP
- Podaci se vraćaju najčešće u JSON formatu – rjeđe se koristi XML i YAML
- Resursi imaju URL ili URI kojima se identificiraju
- Svaki poziv čini jednu akciju (kreiranje, čitanje, mijenjanje ili brisanje podataka)
- Isti URL se koristi za sve operacije, ali se mijenja HTTP metoda koja definira vrstu operacije

Primjer REST servisa



Osobine REST servisa

- Mogućnost privremenog spremanja (Cacheable)
- Jedinstveno sučelje (Uniform interface URI)
- Izričito korištenje HTTP metoda
- Prijenos XML-om ili JSON-om
- Nepostojanje stanja („stateless”) – poslužitelj ne pamti nikakve podatke o klijentu, klijent sa svakim zahtjevom mora slati sve potrebne informacije za razumijevanje i obradu tog zahtjeva

Prednosti RESTfull servisa

- Jednostavnost
- Fleksibilnost formata vraćenih podataka
- Korištenje postojeće mrežne infrastruktura
- Brzo savladavanje tehnike

Resurs

- Ključna apstrakcija informacija u REST-u je resurs
- Bilo koja informacija koja se može imenovati može biti resurs – dokument ili slika, privremeni servis, skup drugih resursa ili stvarni objekt (npr. osoba)
- Stanje resursa u određeno vrijeme se naziva reprezentacijom resursa
- Reprezentacija resursa sastoji se od podataka, metapodataka koji opisuju podatke i hipermedijskih linkova koji pomažu klijentu pri prijelazu na sljedeće željeno stanje

REST API

- REST API sastoji se od skupa međusobno povezanih resursa
- Resursima se pristupa putem URI-ja (Uniform Resource Identifiers)
- API (engl. application programming interface) je skup pravila koja definiraju kako se aplikacije ili uređaji mogu međusobno povezivati i komunicirati.
- REST API-ji su uobičajena metoda za povezivanje komponenti i aplikacija u arhitekturi mikroservisa.

Mikroservisi

- Mikroservisna se arhitektura sastoji od malih servisa, zvanih mikroservisi (dalje u tekstu: servisi), autonomnih programskih komponenti koje pružaju usluge drugim servisima i s kojima surađuju.
- Cilj razvoja servisa u mikroservisnoj arhitekturi je da u konačnici servis bude što manji, optimiziran i fokusiran isključivo na svoju domenu.
- Servisi moraju biti međusobno neovisni
- Servisi koriste aplikacijsko programsko sučelje (eng. application programming interface - API) za komuniciranje i surađivanje s drugim servisima putem mreže.
- Svaki servis pruža svoje sučelje drugim servisima pa odabir tehnologije sučelja nije trivijalan zadatak.

Razlika između SOA-e i mikroservisa

SOA	Mikroservisna arhitektura
Maksimiziranje ponovnog korištenje servisa	Fokusira se razdvajanje
Sustavna promjena zahtjeva izmjenu monolita	Sustavna promjena znači dodavanje novog servisa
DevOps i Continuous Delivery postaju popularni ali ne nužni	Veliki fokus na DevOps i Continuous Delivery
Fokus na ponovnom korištenju poslovne funkcionalnosti	Veća važnost na koncept ograničenog konteksta
Za komunikaciju se koristi Enterprise Service Bus (ESB)	Za komunikaciju se koriste šturi i jednostavni sustavi poruka
Podržava nekoliko protokola poruka	Koristi jednostavne protokole poput HTTP-a, REST-a ili RPC-a
Koristi zajedničku platformu za razmještanje svih servisa	Aplikacijski poslužitelji se ne koriste, normalno je da se koriste platforme u oblaku
Korištenje kontejnera poput Docker-a nije toliko popularno	Kontejneri rade vrlo dobro s servisima
SOA servisi dijele spremište podatka	Svaki servis može imati svoje spremište podataka
Zajedničko upravljanje i standardi	Opušteno upravljanje, s većim naglaskom na suradnju timova i slobodu izbora

Modeliranje mikroservisa

- Slabo povezani servisi ne znaju puno jedni o drugima, čak i kada neposredno surađuju.
- Cilj je visoke kohezije (eng. high cohesion) grupiranje logike programa u povezane cjeline.
- Svaka cjelina sadrži logiku i ponašanje koje je usko vezano za njezin kontekst, dok je sve ostalo, što nije direktno vezano za taj kontekst, u drugoj cjelini.
- Uzevši u obzir tehnologije koje se koriste u mikroservisnoj arhitekturi, tehnologije integracije imaju najveću važnost.
- Ispravnim odabirom tehnologije integracije ona se neće morati mijenjati za vrijeme razvoja aplikacije, čime se izbjegava teška izmjena svih servisa zbog takve promjene.

WebSocket protokol (1)

- WebSocket protokol je po mnogima najbolja tehnologija i komunikacijska tehnika dostupna za korištenje u svrhu izgradnje Web aplikacija u realnom vremenu.
- Pruža perzistentnu, asinkronu, dvosmjernu komunikaciju (eng. full duplex) preko jedne TCP veze kojom je moguće slati neograničen broj zahtjeva tako dugo dok je veza otvorena i to sa vrlo malim troškom u resursima aplikacije.
- Standardiziran je od strane W3C konzorcija 2011. godine sa pripadnim WebSocket okvirom za razvoj (eng. API), a danas ga podržava velika većina modernih web preglednika

WebSocket protokol (2)

- Veza koja se otvara WebSocket protokolom je takve prirode da dopušta i klijentu i poslužitelju da, neovisno jedan o drugome, šalju poruke jedan drugome, čak i u isto vrijeme.
- Također, zbog malog trošenja resursa poput radne memorije, poslužitelj može u isto vrijeme imati otvoren velik broj veza prema više klijenata koristeći WebSocket protokol što omogućuje izradu skalabilnih Web aplikacija u realnom vremenu, pogotovo ako očekujemo da aplikaciju koristi velik broj korisnika u isto vrijeme.

Web servisi za IoT

- Web servisi ključni su za razvoj IoT-a
- Pogotovo se koriste u WoT, čiji su zapravo tehnološki temelj

Pitanja i rasprava