Servisno orijentisana arhitektura i integrisanje poslovnih aplikacija

Složenost poslovnog okruženja proizvodnih preduzeća karakteriše izuzetno snažna uzročno posledična povezanost svih aktivnosti poslovanja, od najopštijih procesa dugoročnog planiranja, do tehnoloških postupaka u okviru kojih se dodaje direktna vrednost proizvodu. Pravovremenost i kontinuitet ovih aktivnosti i usklađenost njihovih veličina su osnovni uslovi uspešnog ukupnog poslovanja. Da bi oni bili ispunjeni, neophodno je izvršiti njihovu integraciju, i to u meri i obimu u kojima je moguće ostvariti nesmetanu komunikaciju između svih relevantnih poslovnih funkcija. Jedan od predstavljenih načina za to je obuhvaćen principima integrisanja poslovnih aplikacionih sistema (EAI). Koncepti, kao što su sinhrona i asinhrona interoperabilnost, kompenzacija transakcija i korišćenje kanoničnog modela podataka u interakciji dve aplikacije, odavno izlaze iz okvira ove pojedinačne discipline, i razrađuju se u različitim tehničkim okolnostima.

Objedinjene koncepte integrisanja poslovnih aplikacionih sistema (EAI), procesne orijentacije poslovanja (ERP) i eksponiranja poslovnih funkcija korišćenjem web servisa, danas obuhvata nova disciplina – razvoj servisno orijentisane arhitekture (SOA). Njen cilj je da ostvari tehničke uslove za integrisanje poslovnih funkcija jednog preduzeća, i to - korišćenjem poslovnih servisa i procesa kao subjekata integracije.

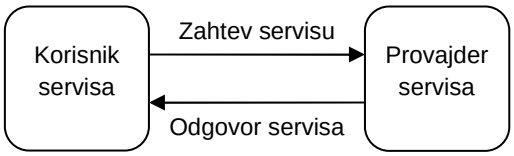
Dok se primenom tehnologije web servisa danas vrši realizacija koncepta poslovnih servisa, dominantna tehnologija njihove orkestracije je BPEL jezik. On se koristi za modeliranje strukture sinhronih ili asinhronih aktivnosti poslovnih procesa, definisanje njihove interoperabilnosti, odnosno kanala komunikacije sa partnerskim web servisima ili procesima i implementaciju mehanizama kompenzacije procesa, registracije poremećaja i aktiviranja postupaka za njihovu rezoluciju.

S obzirom na to da je broj industrijskih implementacija SOA arhitekture još uvek veoma mali, može se zaključiti da je tržište SOA aplikacija još uvek u stvaranju. Osnovni razlog za to je nedovoljna informisanost potencijalnih korisnika o mogućnostima organizacije poslovnog okruženja i velikom potencijalu mogućih ušteda kroz povećanje efikasnosti rada, sistematizovanu komunikaciju i izveštavanje i mikro-menadžment preduzeća kroz neposrednu kontrolu i praćenje njegovog poslovanja. Problem slabo dostupnog tržišta korisnika proističe i iz činjenice da su dostupni alati uobičajeno veoma teški za korišćenje, što utiče na to da su potrebna velika ulaganja u obuku korisnika, ali i određeni nivo reorganizacije preduzeća da bi se stekli uslovi za ostvarivanje navedenih ušteda.

Šta je servisno orijentisana arhitektura ?

Servisno orijentirana arhitektura predstavlja oblik organizacije integrisanog informacionog okruženja jednog poslovnog sistema, koji karakteriše ponuda i korišćenje njegovih distribuiranih funkcija, predstavljenih servisima. Ona obezbeđuje koncept uniformnih sredstava za eksponiranje, otkrivanje, interakciju i korišćenje pojedinačnih poslovnih funkcija, u cilju ostvarenja definisanih ciljeva. Njenu apstraktnu definiciju čini skup principa kojima se definiše novi koncept korporativnih informacionih sistema, od kojih su najznačajniji: granulacija poslovnih funkcija i obezbeđenje jedinstvenog pogleda na poslovanje kroz izgradnju arhitekture poslovnog sistema, čija je realizacija nezavisna od tehnologije

njene implementacije. Princip granulacije poslovnih funkcija se ostvaruje realizacijom servisa (Slika 1) - poslovne funkcija koja je potpuno definisana, sama sebi dovoljna i, ni na koji način ne zavisi od konteksta ili stanja ostalih servisa.



Slika 1. Servis

SOA može da se sagleda i kao jedna perspektiva ukupnog IT okruženja poslovnog sistema, u kontekstu razvoja, implementacije, ponude i korišćenja samostalnih, nezavisnih softverskih servisa koji podržavaju pojedinačne zahteve poslovnih procesa i korisnika. U SOA okruženju, servisi su dostupni korisnicima i poslovnim procesima, bez kontrole i koordinacije pristupa u okviru logičke arhitekture komponenata koji ih nude. SOA je neutralna sa stanovišta tehnoloških standarda i protokola, odnosno, ona ne podrazumeva korišćenje određene tehnologije i može se realizovati primenom bilo kog standarda interoperabilnosti, kao što su RPC[1], DCOM[2], ORB[3] ili SOAP[4].

Osnovni fokus SOA koncepta predstavlja modeliranje i implementacija poslovne, a ne tehničke infrastrukture. Svaki servis predstavlja ekspoziciju određene poslovne funkcije unutar jednog poslovnog okruženja. Tehnički servisi, kao što su obavljanje transakcije, perzistencija podataka, i sl., iako neophodni za tehničku implementaciju poslovnih servisa, nisu strateški relevantni za istraživanje i modeliranje jednog SOA okruženja. U tom smislu, tehnički detalji implementacije ne smeju imati nikakav uticaj na SOA strukturu visokog nivoa, naročito kada je u pitanju međuzavisnost različitih servisa ili njihovih komponenata.

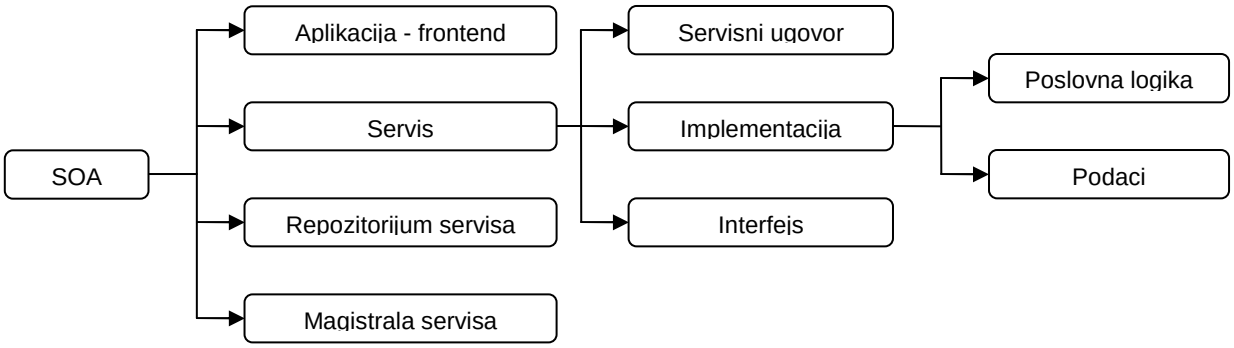
Termin „servisno orijentisana arhitektura" je uspostavljen od strane *Gartner, Inc.* korporacije i definisan kao "oblik višeslojne organizacije računarskih sistema koji obezbeđuje deljenje logike poslovanja i informacija od strane različitih softverskih sistema i načina njihovog korišćenja". Logika poslovanja je realizovana interoperabilnim servisima koji se zasnivaju na formalnoj definiciji, nezavisnoj od njihove realizacije. Danas, za formalnu definiciju interfejsa interoperabilnih servisa, uobičajeno se koristi WSDL specifikacija, dok se njihova realizacija može obezbediti primenom bilo koje programske paradigme (Java, .NET, itd.). Na taj način, obezbeđuje se sredstvo za integraciju tehnološki heterogenih sistema.

Osnovni elementi servisno orijentisane arhitekture

Servisno orijentisana arhitektura se zasniva na 4 ključne apstrakcije: aplikacija (*application frontend*), servis, repozitorijum servisa i magistralu servisa (*service bus*). Iako je vlasnik poslovnog procesa - aplikacija, servisi su ti koji obezbeđuju funkcionalnost koju ona, kao i ostali servisi mogu da koriste.

Šema osnovnih elemenata servisno orijentisane arhitekture je prikazana na slici 2.

[1] RPC - Remote Procedure Call
[2] DCOM - Distributed Component Object Model
[3] ORB - Object Request Broker
[4] SOAP - Simple Object Access Protocol



Slika 2. Osnovni elementi servisno orijentisane arhitekture

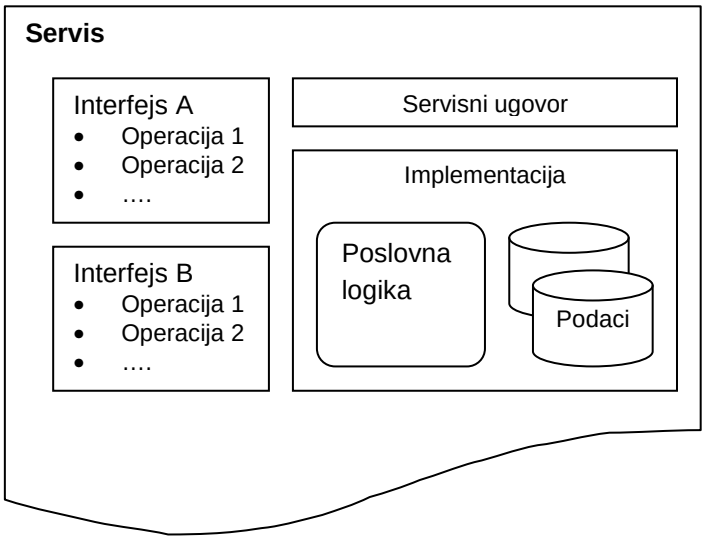
Opisani apstraktni koncept SOA sistema se nadgrađuje primenom jezika visokog nivoa, kao što je BPEL. Ovi jezici ili standardi se koriste kao metod za definisanje i podršku orkestraciji servisa u grupu servisa, objedinjenih u poslovnim procesima, implementiranih uz pomoć kompozitnih aplikacija.

Aplikacije predstavljaju aktivni element SOA arhitekture, posredstvom kojih se njene vrednosti realizuju kroz interakciju sa korisnicima. One iniciraju i upravljaju svim aktivnostima poslovnih sistema. Uopšteno, njihova uloga je iniciranje poslovnih procesa i prijem rezultata neke aktivnosti servisa. Životni vek aplikacije je odvojen od životnog veka servisa – one su češće predmet revizija od samih servisa.

Aplikacije mogu biti sredstvo za interakciju sa korisnicima putem korisničkog interfejsa, realizovanog web ili drugim klijentskim tehnologijama ili softverske komponente koje izvršavaju određenu funkciju nekog servisa stalno, periodično ili po izvršavanju nekog karakterističnog događaja u poslovnom sistemu.

Servis predstavlja primarni element SOA arhitekture - enkapsulaciju poslovnog koncepta niskog nivoa. Njega karakterišu implementacija kojom je realizovana poslovna logika i podaci, servisni ugovor (*Service Contract*) koji definiše funkcionalnost, uslove korišćenja i ograničenja za klijente servisa, i interfejs kojim se fizički eksponira funkcionalnost servisa.

Struktura servisa u SOA arhitekturi je prikazana na slici 3.



Slika 3. Struktura servisa SOA arhitekture

Servisni ugovor obezbeđuje neformalnu specifikaciju svrhe, funkcija, ograničenja i uslova korišćenja individualnog servisa. Oblik ove specifikacije može da varira u odnosu na svrhu servisa. Neobavezni deo servisnog ugovora je i definicija interfejsa, koja se može kreirati uz pomoć IDL[5] ili WSDL jezika. Iako

[5] IDL - Interface Description Language

neobavezna, ona je izuzetno važna, jer obezbeđuje dodatni nivo apstrakcije i nezavisnosti od tehnologije, odnosno, programskog jezika implementacije, mrežnog protokola, okruženja za izvršavanje, itd.

Funkcionalnost servisa se eksponira njegovim klijentima posredstvom servisnog interfejsa. Njegova implementacija se sadrži u servisnim *stub*-ovima (klijentski *proxy* interfejs), koji su implementirani u aplikacijama.

Implementacija servisa obuhvata tehničku realizaciju poslovne funkcije, kroz obezbeđivanje određene poslovne logike i podataka za njeno izvršavanje.

Jedan servisni ugovor čine sledeće komponente:

1. **Zaglavlje servisnog ugovora.** U okviru zaglavlja servisnog ugovora, definiše se naziv servisa, identifikuje njegov vlasnik i uloge - role, odgovorne za isporučene rezultate, donošenje odluka u vezi sa servisom, role koje je potrebno konsultovati i/ili informisati o servisu. U okviru zaglavlja se definiše i tip servisa, sa stanovišta podataka, procesa, funkcionalnosti i prezentacije.
2. **Funkcionalni opis servisa.** Ova sekcija servisnog ugovora predstavlja pregled funkcija servisa, odnosno metoda i akcija koje se koriste za ostvarivanje svake od njih. U okviru funkcionalnog opisa servisa, potrebno je definisati i način pozivanja servisa (SOAP, event trigger, REST, itd.).
3. **Nefunkcionalni opis servisa.** Veoma važne karakteristike servisa se predstavljaju u sekciji servisnog ugovora koja objedinjuje sve one karakteristike koje nisu neposredno vezane za funkciju koju servis obavlja. U okviru nje, definišu se prava pristupa servisu – ko može da mu pristupi i na koji način da ga pozove; tolerisani obim nepravilnog rada; nivo kvaliteta; njegova transakciona svojstva, itd.

Repozitorijum servisa se koristi za skladištenje servisnih ugovora individualnih servisa. On obezbeđuje alate za otkrivanje potrebnih servisa i pristup informacijama o uslovima njihovog korišćenja. Iako se veliki broj informacija o servisima skladišti u okviru servisnog ugovora, repozitorijumi servisa se mogu iskoristiti za skladištenje dodatnih korisnih informacija, kao što su fizička lokacija servisa, informacije o vlasniku, odnosno provajderu servisa, informacije o troškovima korišćenja, tehničkim ograničenjima, pitanjima bezbednosti, itd.

Projektovanje i razvoj servisno orijentisane arhitekture

Osnovne smernice za projektovanje, razvoj, održavanje i korišćenje SOA arhitekture su višestruko korišćenje, granularnost, modularnost i interoperabilnost servisa, kao i okruženje za njihovu identifikaciju, kategorizaciju, isporuku, monitoring i praćenje. Pritom, veoma je važno istaći i ulogu upravljanja životnim vekom servisa, efikasno korišćenje resursa i praćenje njegovih performansi i zrelosti.

Osnovni principi za projektovanje servisno orijentisane arhitekture su:

- **Enkapsulacija servisa.** Servis se predstavlja svojom definicijom, odnosno, interfejsom prema okruženju, koji otkriva njegovu funkciju i odgovarajućom, enkapsuliranom logikom za njeno izvršavanje.
- **Slaba međusobna povezanost.** Iako između servisa postoje relacije, oni ne smeju biti međusobno uslovljeni. Jedina vrsta veze koja se između njih uspostavlja je znanje o međusobnom postojanju.
- **Ekspozicija funkcionalnosti servisa putem servisnog ugovora** (*Service Contract*). Funkcija i opis servisa se predstavlja servisnim ugovorom (*Service Contract*).

- **Apstrakcija servisa.** Okruženju su dostupne samo informacije o servisu koje su predstavljene u njegovom ugovoru. Logika funkcionisanja je sakrivena.
- **Višestruko korišćenje servisa.** Logika se distribuira po servisima sa ciljem da se omogući njihovo višestruko korišćenje u različitim kontekstima.
- **Sposobnost kompozicije servisa.** Raznovrsne grupe servisa se mogu orkestrirati i koordinisati sa ciljem formiranja kompozitnih servisa.
- **Autonomija servisa.** Isključivu kontrolu nad logikom koja je enkapsulirana u okviru određenog servisa ima samo taj servis.
- **Neperzistentnost servisa.** Servisi se staraju o očuvanju minimalnog skupa informacija o aktivnostima koje se sprovode.
- **Eksponiranost servisa.** Servisi se projektuju tako da se mogu lako pronaći i analizirati od strane odgovarajućih mehanizama za njihovo otkrivanje.

Aktivnosti razvoja SOA arhitekture

Osnovne aktivnosti razvoja SOA arhitekture su:

- Dekompozicija SOA
- Definicija servisa i
- Implementacija servisa

Jednu od najvažnijih faza u implementaciji SOA arhitekture predstavlja identifikacija servisa i to na domenu postojećih aplikacija, nezavisnih ili integrisanih u jedinstveni informacioni sistem preduzeća. Identifikacija servisa se obavlja u okviru aktivnosti dekompozicije servisno orijentisane arhitekture.

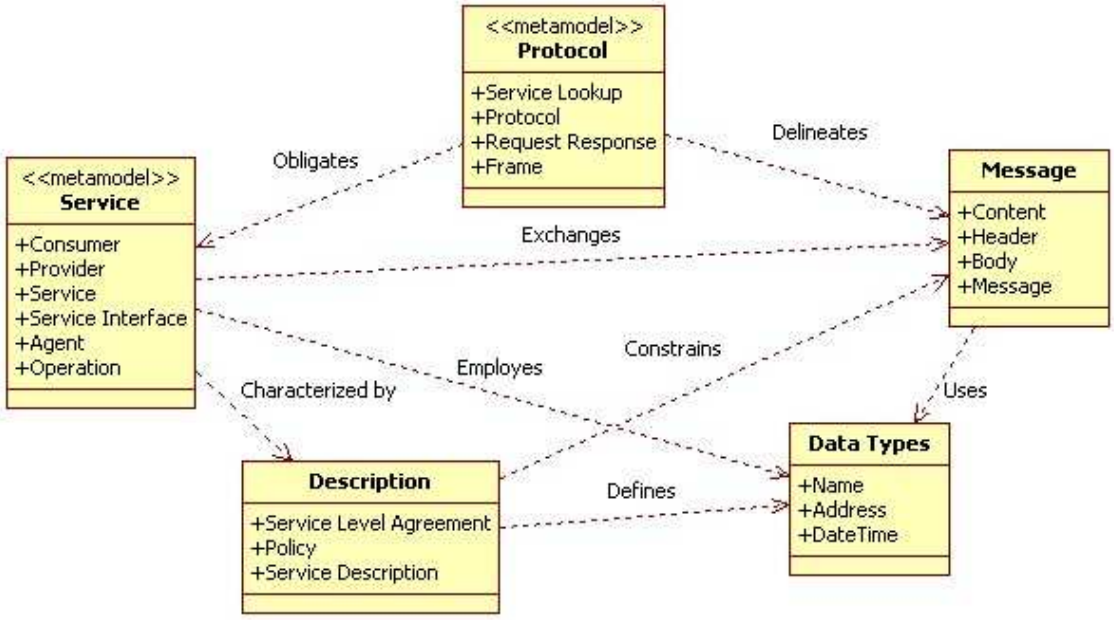
Dekompozicija servisa pretpostavlja:

- identifikaciju hijerarhije servisa, u kontekstu organizacije, procesa i funkcija poslovnog sistema,
- projektovanje semantičkog modela koji treba da ustanovi smernice za međusobnu komunikaciju servisa i obezbeđivanje njihove interoperabilnosti; i
- refaktorisanje identifikovanih servisa, u cilju obezbeđenja principa i standarda performansi, proširivosti, bezbednosti, itd.

Osnovna pretpostavka za servisno-orijentisanu dekompoziciju je postojanje korporativnog poslovnog modela – osnovne reprezentacije resursa i procesa uključenih u ostvarivanju operativnih, taktičkih i strateških poslovnih ciljeva. Svaki poslovni proces podrazumeva orkestraciju pojedinačnih resursa čijom se kontrolisanom interakcijom vrši implementacija poslovnih funkcija.

Definicija servisa obuhvata strukturu informacija o servisu koja je namenjena njegovom korisniku. Ona je predstavljena referentnim UML[6] meta-modelom, prikazanim na slici 4.

[6] UML - Unified Modelling Language



Slika 4. Referentni UML model SOA arhitekture

Definicija servisa obuhvata sledeće elemente:

1. **Informacije na osnovu kojih korisnik treba da utvrdi koji servis mu je potreban.** One obuhvataju opis namene i ciljeva servisa, ograničenja u korišćenju i nivo kvaliteta, uslovi koje korisnik treba da ispuni da bi mogao da ga koristi. Pored toga, korisnik treba da zna kako se servis koristi. U tom kontekstu, definicija servisa treba da sadrži strukturu zahteva i isporučenog rezultata servisa, uslove pod kojima dolazi do određenih rezultata, i sl.
2. **Tehničke informacije o pozivanju servisa.** Komunikacioni protokoli, formati poruka, uključujući i tehnike serijalizacije, lokacija servisa, zahtevi bezbednosti, opis SOAP zaglavlja, kvantitativno izražene parametre kvaliteta servisa, kao što su vreme dostupnosti, vreme odgovora, obradna moć servisa, itd.

Osnovni princip implementacije servisa je izgradnja posebnog sloja servisa kojim se racionalizuju funkcije postojećih aplikacija, sa stanovišta realizacije poslovnih procesa, čija je tehnička implementacija sakrivena od očiju korisnika. Ovaj poseban sloj predstavlja virtuelnu platformu za upravljanje korporativnim poslovnim procesima.

Orkestracija servisa

U servisno orijentisanoj arhitekturi, servisi predstavljaju izolovane, pojedinačne poslovne funkcije korporativnog okruženja. Izvršavanje svake pojedinačne funkcije rezultuje ostvarenjem ograničenog cilja pojedinačne poslovne aktivnosti, kroz izvršenje upita, transakcije, autentifikacije i sl. Ispunjenjem ograničenog cilja se ne postiže domet značaja jednog poslovnog procesa i uticaja njegovog izvršavanja na ukupne performanse preduzeća. Međutim, ovaj uticaj se može ostvariti izvršavanjem niza poslovnih aktivnosti u određenom redosledu, sa tačno definisanom strukturom u kontekstu jednog poslovnog procesa, pri čemu izvršenje svake poslovne aktivnosti podrazumeva pozivanje jednog ili više servisa.

Očigledno je da poslovni proces predstavlja osnovno sredstvo visokog nivoa za orkestriranje servisa u servisno orijentisanoj arhitekturi. Osnovne karakteristike implementacije poslovnih procesa kao osnovnog sredstva za orkestraciju servisa su:

- Servise karakteriše neperzistentnost, odnosno izuzeće funkcije beleženja trenutnog stanja servisa. S obzirom na to, implementacija poslovnih procesa podrazumeva upravljanje kontekstom izvršenja servisa, odnosno njihovog pozivanja u određenom redosledu, upravljanje tokovima podataka i eventualno skladištenje podataka o stanju procesa.
- Skoro sve poslovne procese karakteriše određeni nivo manuelnih aktivnosti. Da bi one mogle da se sprovode na odgovarajući način, potrebno je da implementacija poslovnih procesa obuhvati i podršku radnim listama, pristup sistemu zasnivan na pravima (*role-based*), "zaključavanje" aktivnosti koje se trenutno izvršavaju, itd.
- Implementacija poslovnih procesa kao sredstva za orkestraciju servisa predstavlja autonoman element servisno orijentisane arhitekture, potpuno nezavisan od tehnologije razvoja servisa, odnosno infrastrukture korišćene za njihovu koordinaciju.
- Poslovni procesi predstavljaju sloj realizacije servisno orijentisane arhitekture koji je, za razliku od implementacije samih servisa, podložan kratkoročnim promenama. U tom smislu, neophodno je da oni budu realizovan na fleksibilan, modularni, proširivi način, primenom odgovarajućih široko usvojenih standarda.
- Osnovni elementi orkestracije servisu su *engine* za orkestraciju i jezik za orkestraciju. *Engine* predstavlja aplikacioni softver (server) koji omogućava izvršavanje jednog ili više jezika za orkestraciju.
- Jezik za orkestraciju servisa predstavlja skup semantičkih i sintaksnih pravila za formulisanje i kodiranje strukture koordinacije servisa.

BPEL jezik za modeliranje procesa

BPEL4WS model procesa predstavlja objedinjenu specifikaciju web servisa (WSDL – *Web Services Definition Language*) i sintakse za njihovu orkestraciju u poslovni proces. Ona reprezentuje prirodnu evoluciju tehnologija za realizaciju web servisa, ka SOA paradigmi. Osnovni cilj razvoja BPEL modela procesa je kompozicija, orkestracija i koordinacija web servisa, odnosno, njihova integracija u kompozitne servise, logički predstavljene *poslovnim procesom*.

BPEL predstavlja specifikaciju sa najširom podrškom velikih korporacija (IBM, Microsoft, Oracle, BEA), nastalu kao fuzija dva standarda: XLANG (Microsoft) i WSFL[7] (IBM). Prva verzija je nastala avgusta 2002. godine. U to vreme, inicijativa IBM i Microsoft korporacije je bila kritikovana kao udarac već rasprostranjenom otvorenom standardu – BPML. Najrasprostranjenija verzija standarda - 1.1 usvojena je marta 2003. godine.

Osnovna razlika BPEL jezika i ostalih standarda za modeliranje poslovnih procesa se sastoji u direktnoj pokrivenosti njegovih manuelnih aktivnosti. Naime, ljudska interakcija sa procesom nije obuhvaćena BPEL specifikacijom. Ona je, pre svega, namenjena podršci izvršenju automatskih poslovnih procesa, zasnovanih na orkestriranju postojećih web servisa – poslovnih funkcija servisno orijentisane arhitekture. Ljudske interakcije sa procesom, odnosno uključivanje manuelnih aktivnosti u poslovni proces primenom BPEL specifikacije se mogu obezbediti na dva načina.

[7] WSFL - Web Services Flow Language

Prvi način obuhvata korišćenje standardnih BPEL elemenata u definiciji procesa, sa jedne i, razvoj generičkog WSDL klijenta koji sadrži funkciju automatskog generisanja odgovarajuće HTML[8] forme, na osnovu definicije ulaznih parametara metode web servisa, sa druge strane.

Drugi način predstavlja korišćenje ekstenzije BPEL specifikacije za definisanje manuelnih aktivnosti - BPEL4People, koju su razvili istraživači IBM i SAP korporacije.

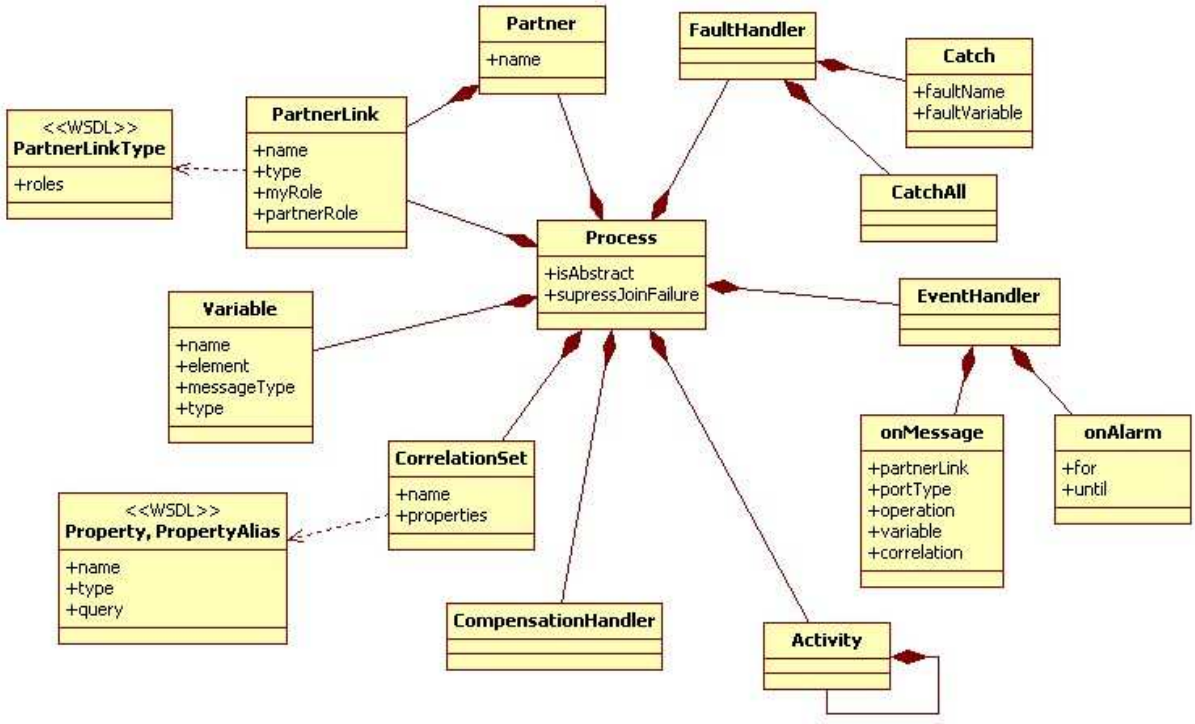
Modeliranje poslovnih procesa primenom BPEL specifikacije

BPEL definicija procesa se sastoji od dva tipa datoteka XML strukture: WSDL i BPEL. WSDL datoteke sadrže specifikaciju web servisa, odnosno podatke o tipovima partnerskih linkova (*partner links*), karakteristikama (*properties*), tipovima portova (*port types*) i operacijama (*operations*).

BPEL datoteke sadrže XML definiciju procesa, uključujući njegove aktivnosti, partnerske linkove, skupove korelacija (*correlation sets*), promenljive (*variables*) i hendlere događaja (*event handlers*), grešaka (*fault handlers*) i kompenzacije (*compensation handlers*).

Model procesa se sastoji od njegove osnovne definicije, koju čini niz aktivnosti povezanih relacijama, i odgovarajućih interfejsa, koje je potrebno definisati da bi proces mogao da komunicira sa eksternim izvorima podataka. Ovaj interfejs je predstavljen WSDL definicijama web servisa, koje su upotpunjene odgovarajućim BPEL ekstenzijama. Osnovne aktivnosti koje definišu tok procesa predstavljen BPEL modelom su aktivnosti prijema (*Receive*), preuzimanja (*Pick*), poziva (*Invoke*) i odgovora (*Reply*).

Na slici 5, prikazan je UML model ukupnog BPEL modela poslovnog procesa i struktura procesa.



Slika 5. UML model ukupnog BPEL modela poslovnog procesa

Definicija svih objekata koji se koriste u BPEL specifikaciji i prikazani su na UML dijagramu gore, data je u tabeli 1.

[8] HTML - Hypertext Markup Language

Naziv objekta	Mesto definicije	Opis
Process	BPEL	Poslovni proces koji sadrži jedan ili više drugih objekata.
Variable	BPEL	Promenljiva koju koristi proces – tip zasnovan na WSDL tipu poruke, XSD element ili osnovni XSD tip. Proces može imati nijednu ili više promenljivih. Sve promenljive čije se vrednosti razmenjuju u okviru interakcije jednog procesa sa partnerskim servisom se moraju deklarisati u okviru procesa. Dodatno, njihove vrednosti se mogu koristiti za jednostavnu obradu u okviru procesa.
Property, Property Alias	WSDL	Osobina predstavlja jedan token podataka WSDL poruke. Alias osobine je *XPath* izraz koji se koristi za pronalaženje vrednosti osobine. Osobine se koriste za definisanje korelacije BPEL promenljivih.
CorrelationSet	BPEL	Skup korelacija predstavlja skup jedne ili više osobina koje se koriste za uzajamno referenciranje podataka iz jedne poruke i stanja procesa.
Partner Link Type	WSDL	Tip partnerskog linka predstavlja definiciju referenciranja tipova portova na partnerske role.
Partner Link	BPEL	Partnerski link predstavlja deklaraciju procesa o tome koje veze sa partnerskim web servisima su podržane.
Partner	BPEL	Skup partnerskih linkova
Compensation Handler	BPEL	Aktivnost koja sadrži logiku otkazivanja ili povraćaja procesa na prethodno stanje. Ona se izvršava u slučaju da je proces koji je već izvršen potrebno vratiti u inicijalno stanje.
Fault Handler, Catch, CatchAll	BPEL	Skup hendlera za obradu izuzetaka, na osnovu tipa greške, unutar jednog procesa. Obrada izuzetaka, ili grešaka, se veoma često vrši prilikom sinhronog poziva operacije partnerskog web servisa. Prihvat greške (*catch*) se može vršiti u okviru aktivnosti procesa, prilikom čega ona poziva drugu, namensku aktivnost koja vrši obradu greške. Ukoliko aktivnost procesa ne predviđa prihvat greške, on se vrši u okviru odgovarajućeg okvira (*scope*) - autonomnog dela procesa sa svojim skupom hendlera, promenljivih i skupova korelacija.
EventHandler, onMessage, onAlarm	BPEL	Skup hendlera za obradu događaja unutar jednog procesa.
Activity	BPEL	Osnovni tip BPEL aktivnosti. BPEL specifikacija definiše da se proces sastoji od samo jedne aktivnosti koja je struktuirana u posebne delove.

Tabela 1. Specifikacija BPEL objekata

Sa stanovišta realizacije jednog izvršnog BPEL procesa, može se reći da postoje dva tipa WSDL interfejsa. Jedan predstavlja interfejs ka samom procesu i ne sadrži implementaciju u nekom programskom jeziku. On sadrži operacije koje izvršava proces, definisane u okviru *MyRole* atributa partnerskih linkova iz BPEL definicije. Obično se koristi za prihvat podataka koji se koriste ili obrađuju u toku procesa, bez obzira na to da li se njihova obrada vrši u okviru definicije procesa, ili se prihvaćeni podaci prosleđuju partnerskom web servisu.

Ostali WSDL interfejsi predstavljaju sredstvo za komunikaciju procesa sa partnerskim web servisima, odnosno njihovu orkestraciju. Ovi servisi su uobičajeno implementirani nezavisno od razvoja procesa i

izvršavaju se kao web aplikacije na nekom aplikacionom serveru. Njihove operacije se koriste posredstvom definicije *PartnerRole* atributa partnerskih linkova iz BPEL definicije.

Modeliranje relacija između procesa i njegovih partnera

Osnovni uslov za uspostavljanje konverzacionih relacija između procesa i njihovih partnera je definisanje tipova partnerskih linkova (*partner link types*) i rola (*roles*) u okviru WSDL interfejsa. Tip partnerskog linka predstavlja definiciju interfejsa koji se koristi za interakciju dva servisa, odnosno servisa i procesa. Ova definicija obuhvata specifikaciju poruka i tipova portova koje koristi svaki od servisa u komunikaciji. Svaki tip partnerskog linka obuhvata definiciju jedne ili dve role, koje se koriste u definiciji partnerskog linka u BPEL modelu procesa, kao atributi *MyRole* i *PartnerRole*. Ove role se odnose na uloge procesa i partnerskog web servisa u komunikaciji.

Generalno, jedna rola se koristi u sinhronim interakcijama, jer se jedna operacija web servisa koristi za upućivanje zahteva i isporuku rezultata (*request/response*). Sa druge strane, kod asinhrone interakcije jedan partner može imati različite role u njenom trajanju. Naime, u prvom koraku, partnerski servis prima zahtev od procesa za nekom obradom. U tom slučaju, proces ima ulogu klijenta servisa, dok partnerski servis ima ulogu njegovog provajdera. U drugom koraku, posle konačnog vremena, potrebnog za obradu, partnerski servis igra ulogu klijenta, jer on inicira interakciju sa ciljem da isporuči rezultat obrade procesu koji ga je prethodno zahtevao. U ovom slučaju, proces ima ulogu provajdera servisa, koji se sastoji samo od prijema poslatog rezultata. Iz ovih razloga, prilikom asinhronih interakcija, tip partnerskog linka mora imati definisane obe role, dok su operacije WSDL interfejsa u dva koraka interakcije – različite, kao i tipovi portova koji ih nude. Prilikom definisanja tipova portova i operacija za isporuku prethodno zahtevane informacije u asinhronoj interakciji, koristi se termin – *callback*.

Definicija tipova partnerskih linkova se može skladištiti u posebnim artefaktima, ali je zbog referenciranja rola na tipove portova poželjno da se ona nalazi u istom prostoru imena, zbog čega se uobičajeno pridružuju WSDL definiciji interfejsa.

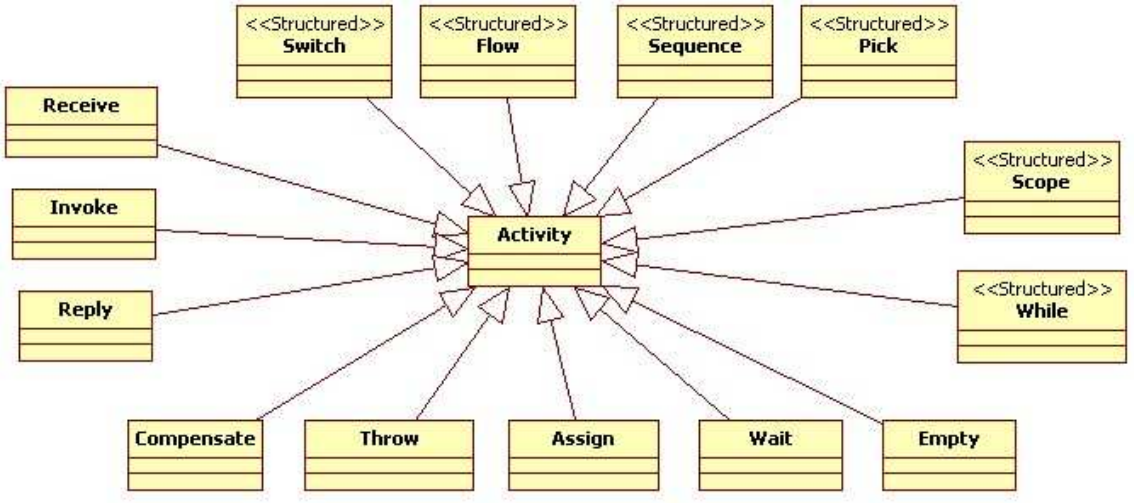
U velikom broju slučajeva, izuzetno je važno obezbediti dinamičnu prirodu BPEL procesa. Naime, veoma često, svi mogući tokovi procesa, ali i karakteristike njegovih aktivnosti, nisu poznati u trenutku njihovog modeliranja. Aktuelna BPEL specifikacija podržava određeni nivo dinamičnosti procesa kroz primenu tzv. dinamičkih partnerskih linkova, odnosno dinamičkog referenciranja na partnerske linkove. Ovaj koncept omogućava dodavanje novih servisa nakon instalacije procesa, odnosno u toku njegove konfiguracije i, čak – izvršavanja. Njegovom primenom se stvaraju uslovi za kreiranje adaptirajućih procesa, čija se struktura menja u zavisnosti od uslova njihovog izvršavanja.

Modeliranje aktivnosti poslovnih procesa primenom BPEL specifikacije

Početna aktivnost svakog procesa mora biti određenog tipa, s obzirom na to da inicijacija jedne njegove instance podrazumeva prihvat određenog skupa ulaznih informacija. One mogu biti:

- priložene od strane korisnika BPM sistema,
- informacije na koje početna aktivnost čeka, osluškujući neki web servis (*Receive*), ili
- one koje početna aktivnost preuzima od njega u tačno određenom trenutku (*Pick*).

Na slici 6, prikazan je UML model aktivnosti poslovnih procesa podržanih BPEL specifikacijom.



Slika 6. UML model aktivnosti poslovnih procesa

Svi mogući tipovi aktivnosti predviđeni BPEL specifikacijom su prikazani u tabeli 2.

Tip aktivnosti	Opis
Receive	Aktivnost koja prima SOAP poruku od strane nekog web servisa. Poslovni proces obezbeđuje pristup web servisima svojim partnerima kroz aktivnosti prijema poruka (*receive*) i odgovarajućih odgovora (*reply* aktivnosti). U okviru *receive* aktivnosti, definiše se link partnera od kojeg se očekuje slanje poruke procesu, odnosno očekivani tip porta i operacija web servisa. Ovaj tip aktivnosti predstavlja inicijalnu aktivnost svakog procesa, pri čemu, više uporednih receive aktivnosti mogu predstavljati ulaz u proces – jezgro njegovog instanciranja, pod uslovom da je njihov parametar *createInstance*, podešen na *yes*. S obzirom na to da *receive* aktivnost predstavlja obradu prijema neke poruke, očigledno je da ona "blokira" proces – proces se ne može instancirati, odnosno, nastaviti, sve dok se ne izvrši prijem poruke.
Reply	Aktivnost koja vraća sinhroni odgovor web servisu, nakon prijema odgovarajuće poruke (*Receive* aktivnost).
Invoke	Aktivnost koja poziva partnerski web servis, na sinhroni ili asinhroni način. Specifikacija asinhronog poziva obuhvata definisanje ulazne promenljive – parametra zahteva, dok se odziv ne definiše. Kod sinhronog poziva, definišu se parametri - promenljive zahteva i odziva. Tom prilikom je potrebno, ali ne i neophodno, obezbediti prihvat i obradu greške (*fault handling*), koji može da generiše WSDL interfejs.
Compensate	Aktivnost koja aktivira kompenzaciju procesa.
Throw	Aktivnost koja registruje grešku u odvijanju procesa i aktivira izvršavanje hendlera grešaka u procesu.
Assign	Aktivnost koja kopira vrednost jedne promenljive u drugu, odnosno vrši manipulaciju jednom promenljivom, tako što je primenom *XPath*, *XQuery* ili XSLT upita, izraza ili funkcija transformiše, a rezultat transformacije dodeljuje drugoj promenljivoj.

Wait	Aktivnost koja zaustavlja izvršavanje procesa u definisanom trajanju, ili do zadatog vremena.
Empty	Prazna aktivnost. Ne izvršava ništa.
Switch	Ekskluzivna OR struktura. Izvršava aktivnost samo u slučaju da je vrednost određenog izraza - *true*.
Flow	Struktura sa različitim paralelnim aktivnostima. Sve aktivnosti koje su definisane unutar *Flow* bloka se izvršavaju uporedno.
Sequence	Sekvencijalno izvršavanje skupa aktivnosti. Sve aktivnosti koje su definisane unutar *Sequence* bloka se izvršavaju postavljenim redosledom.
Pick	Struktura koja čeka na događaje. Izvršava aktivnost u njenom domenu, za koju je definisano da se izvršava prilikom određenog događaja (*onMessage*, *onEvent*, *onAlarm*).
While	Izvršavanje aktivnosti u petlji za svo vreme za koje je vrednost zadatog *XPath* izraza – true.
Scope	Poseban autonomni deo procesa sa svojim skupom hendlera, promenljivih i skupova korelacija.

Tabela 2. Tipovi aktivnosti predviđeni BPEL specifikacijom

Manipulacija vrednostima promenljivih procesa

Sve poruke koje se razmenjuju između partnera servisno-orijentisane arhitekture, odnosno procesa i partnerskih servisa, kao i promenljive deklarisane u okviru samih procesa su opisane XML tipovima. Ovi XML tipovi su definisani u okviru WSDL interfejsa ili eksternih šema koje su u njih uvežene. Manipulacija XML dokumentima se vrši primenom osnovnih elemenata *XPath* standarda – upita, izraza i funkcija, pri čemu se za složeniju manipulaciju koriste i XQuery i XSLT standardi.

Osnovna aktivnost koja se koristi za manipulaciju vrednostima promenljivih procesa je <*assign*> BPEL aktivnost, odnosno njen blok <*copy*> i odgovarajući elementi <*from*> i <*to*>, u okviru kojih se definiše ulazna promenljiva, odnosno oblik njene manipulacije i destinacija kojoj se dodeljuje njen rezultat. Manipulacija promenljivim procesa se može vršiti na sledeće načine:

- Izborom elemenata složenih promenljivih – XML dokumenata, primenom *XPath* upita;
- Primenom *XPath* izraza;
- Primenom osnovnih *XPath* funkcija, kao što su manipulacija stringovima, numeričke funkcije, itd.
- Primenom BPEL *XPath* funkcija, koje omogućavaju da *XPath* izrazi u manipulaciji koriste informacije iz procesa;

Pored ovih načina, mogu se koristiti i *XPath* funkcije koje su ugrađene u alat, koji se koristi za projektovanje servisno-orijentisane arhitekture (npr. *JDeveloper BPEL Designer*), kao i *XPath* funkcije koje su implementirane samostalno, od strane razvojnog tima, odgovornog za projektovanje i implementaciju BPEL procesa.

Na kraju, veoma je važno istaći da vrednost, dodeljena izlaznoj promenljivoj, mora odgovarati tipu koji je korišćen za njeno deklarisanje.

Korelacija poruka

Korelacija poruka predstavlja jedan od mehanizama BPEL specifikacije koja obezbeđuje mogućnost učešća procesa u *statefull* konverzacijama – interakcijama procesa dugog trajanja sa partnerskim servisima, npr. u asinhronoj komunikaciji.

Kada jednom web servisu, koji predstavlja orkestrirani deo određenog procesa, pristigne poruka, potrebno je odrediti kojoj tačno instanci procesa je potrebno tu poruku isporučiti. Lociranje ove instance je zadatak mehanizma za korelaciju poruka. Uzajamno referenciranje podataka u dugo trajućim transakcijama se uobičajeno vrši posredstvom određenog identifikatora – jedinstvene oznake skupa podatka. Ovaj princip nije iskorišćen u implementaciji BPEL procesa, već se od dizajnera procesa očekuje da izabere podatak koji će se koristiti za uzajamno referenciranje pristiglih poruka u web servis i njihovih kopija koje su isporučeni procesu. Ovaj podatak mora biti jedinstven, odnosno, jednoznačno ukazivati na odgovarajuću poruku. Na primer, proces obrade narudžbina započinje unosom same narudžbine u proces, ili odgovarajući sistem koji se nalazi u domenu integracije. S obzirom na to da se, u okviru aktivnosti unosa vrši i skladištenje relevantnih informacija o narudžbini u bazu podataka, sastavni deo strukture narudžbine je i jedinstveni identifikator, dodeljen od strane baze podataka. Ovaj identifikator se može koristiti kao korelacioni identifikator. Korelacioni identifikatori se definišu korišćenjem `property` elementa u okviru WSDL interfejsa. Za uspostavljanje reference između vrednosti `property` elemenata i odgovarajućih tipova u WSDL interfejsu, koriste se `propertyAlias` elementi.

Ukoliko u okviru procesa ne postoji jedinstvena promenljiva koja bi ukazivala na njegovu određenu instancu, onda se koristi jedinstven skup podataka. Ovaj skup tipova se naziva korelacioni skup (*correlation set*). Njega karakterišu naziv i osobine (*properties*) WSDL interfejsa, čije su vrednosti ekstrahovane iz instanci WSDL poruka, primenom *XPath* izraza. Na ove osobine se referenciraju promenljive `receive`, `invoke`, `replay` i `pick` aktivnosti, a BPEL *engine* ih koristi za inicijalizaciju ili upoređenje promenljive u toku izvršenja jedne instance procesa.

Upravljanje izuzecima i greškama

U toku izvršenja jedne instance procesa, određene greške se mogu pojaviti u njemu samom, ili u partnerskim web servisima sa kojima on komunicira. BPEL specifikacija omogućava prihvat (*catch*) ovih grešaka i njihovu obradu u okviru izvršenja posebnih delova procesa, definisanih *fault handler* elementima. Pored toga, učinak već izvršenih aktivnosti koji predstavljaju deo dugo trajuće transakcije, koja je prekinuta iz bilo kog razloga, neophodno je zanemariti, a podatke koje su one izmenile – restaurirati na prethodne vrednosti. *Compensation handler* elementi obuhvataju opis akcija koje je potrebno preduzeti u ovom slučaju, u okviru BPEL definicije procesa. S obzirom na to da obrada grešaka i izuzetaka podrazumeva izvršenje jednog broja aktivnosti, a veoma retko – pojedinačne aktivnosti, ovaj skup akcija se smešta u poseban okvir BPEL procesa, struktuiranu aktivnost definisanu `scope` elementom.

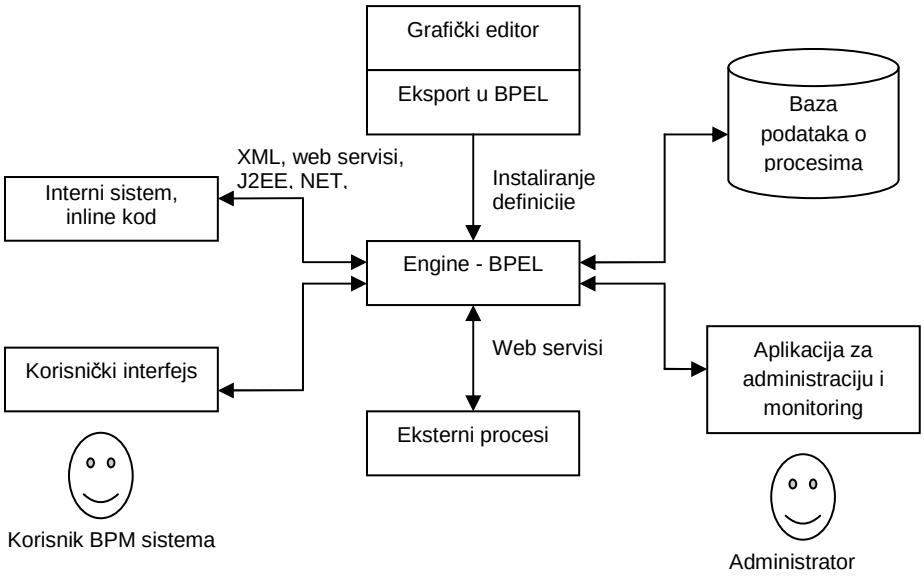
Do greške u izvršenju jedne aktivnosti procesa dolazi jer je poziv određenog servisa rezultovao prihvatom greške, jer je došlo do BPEL generičke greške ili je pomenuta aktivnost `throw` tipa. Kada do neke od ovih grešaka dođe, aktivnost obaveštava okvir (`scope`) u kome se nalazi o tome, koji zaustavlja izvršenje svih narednih aktivnosti. Takođe, o greški se obaveštava i *fault handler* – okvir koji je pomenuta aktivnost predvidela za obradu grešaka, čije se aktivnosti izvršavaju.

Sistemi za modeliranje i upravljanje procesima (BPM sistemi) zasnovani na servisno-orijentisanoj arhitekturi

Upravljanje poslovnim procesima (BPM) predstavlja oblast primene računara i odgovarajućih mrežnih tehnologija u definisanju, simulaciji, izvršavanju, optimizaciji, merenju i kontroli poslovnih procesa. Sve ove aktivnosti su obuhvaćene odgovarajućim standardima i softverskim rešenjima za njihovu realizaciju, objedinjenim u jedinstveni BPM sistem, čija je konvencionalna arhitektura prikazana na slici 7.

Osnovni zahtevi koji jedan BPM sistem treba da ispuni su sposobnost projektovanja, izvršenja, praćenja i administracije poslovnih procesa koji obuhvataju interakcije sa različitim sistemima u okruženju i interakcije korisnika BPM sistema sa sistemom i drugim korisnicima. Objedinjeni BPM sistem treba da u potpunosti podrži zahteve implementacije, skladištenja i izvršavanja web servisa, kao i njihove orkestracije korišćenjem BPEL standarda.

Projektovanje poslovnog procesa predstavlja postupak modeliranja toka aktivnosti koje je potrebno izvršiti u cilju rešavanja nekog poslovnog problema. Alat za modeliranje treba da obezbedi mogućnost korišćenja standardnih grafičkih elemenata za vizuelizaciju procesa, kao i generisanje koda za izvršenje procesa (BPEL), kojim se, prema izabranom standardu opisuje modelirani proces i koji se koristi kao ulaz u *engine* za izvršenje procesa.



Slika 7. Ukupna arhitektura integrisanog BPM sistema

BPM sistem treba da ima sposobnost praćenja procesa koji se trenutno izvršavaju, jer je potrebno obezbediti uvid u aktivnosti koje zaustavljaju ili usporavaju proces i razne mogućnosti izveštavanja. Administracija BPM sistema obuhvata akcije instaliranja novih definicija procesa, privremenog i definitivnog zaustavljanja procesa, itd.

Akteri jednog BPM sistema su njegovi korisnici i web servisi koji se koriste kao osnovni subjekti integracije poslovnog procesa sa internim funkcijama informacionog sistema preduzeća i njegovim partnerima. Sa stanovišta interakcije korisnika BPM sistema i njega samog, arhitektura BPM modela treba da obezbedi *role-based security* model koji korisnicima omogućuje samo uvid u aktivnosti nad kojima korisnici imaju definisane odgovornosti, zatim samu listu aktivnosti i konačno odgovarajući interfejs za unos određenih veličina i parametara aktivnosti.

Automatske aktivnosti BPM sistema se odnose na poziv web servisa koje nisu deo samog sistema, ali mogu to da budu. Takođe, arhitektura BPM sistema treba da omogući da komponente eksternih sistema pozivaju aktivnosti BPM sistema. Ove komponente mogu da budu eksterne sa stanovišta preduzeća (aplikacije dobavljača i klijenata), sa stanovišta sistema (email server, računovodstveni sistem) ili sa stanovišta *engine*-a (jednostavni delovi koda za proračune, parsiranje, itd.).

Karakteristike BPM sistema

Osnovne karakteristike savremenog, integrisanog BPM sistema su:

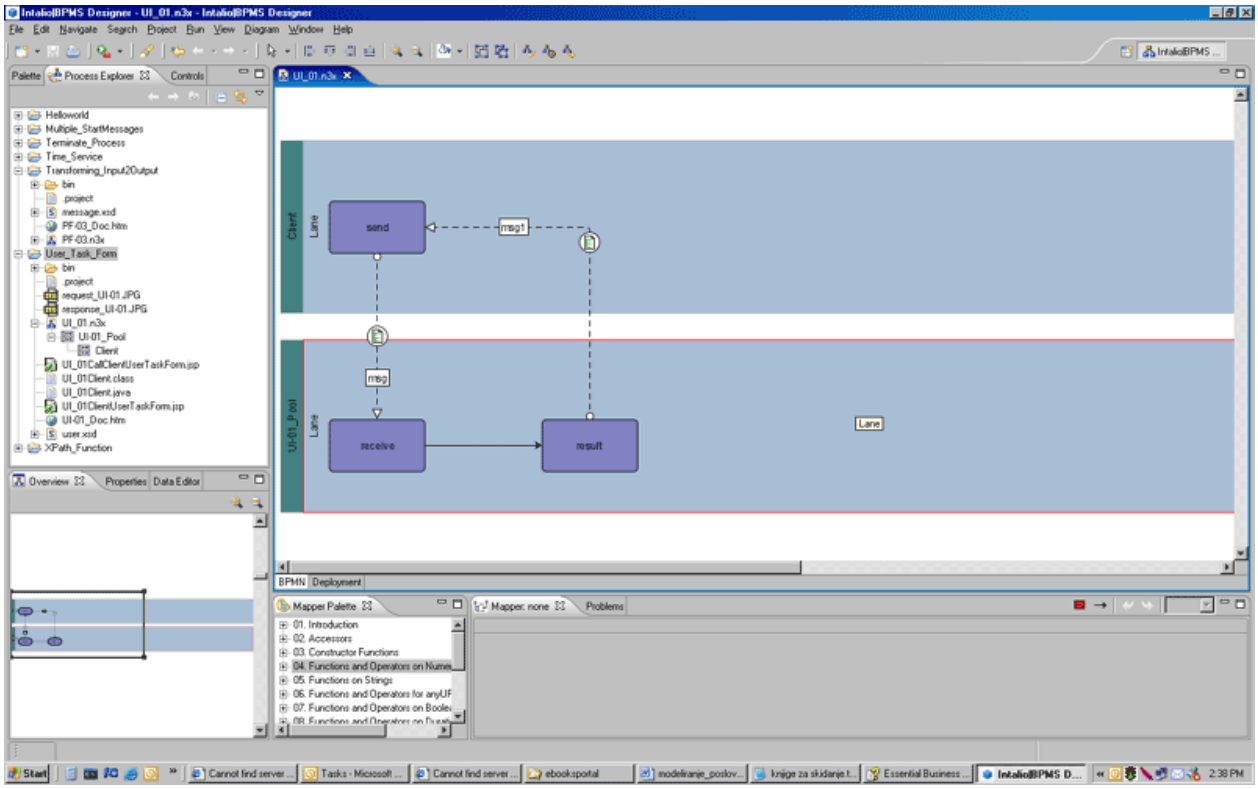
- grafičko modeliranje procesa,

- *engine* za izvršavanje definisanih poslovnih procesa,
- alati i interfejsi za interakciju korisnika i drugih podsistema sa BPM sistemom,
- administracija i monitoring poslovnih procesa,
- jedinstveni model podataka o procesima i
- upravljanje verzijama procesa.

Grafičko kreiranje procesa

Za grafičko kreiranje procesa, čije se izvršavanje podržava savremenim BPM sistemima, koriste se dva standarda za notaciju: BPMN i UML dijagrami aktivnosti. S obzirom na to da je BPMN ekspresivniji i podržava preslikavanje u BPEL, on se najčešće i koristi. Alati koji se uobičajeno koriste za modeliranje procesa primenom BPEL specifikacije, koriste notaciju koja je veoma slična BPMN notaciji, ali ne i u potpunosti kompatibilna. Naime, specifikacija BPMN notacije se zasniva na BPML jeziku za modeliranje procesa. BPML jezik je, za razliku od BPEL standarda, pre svega namenjen realizaciji workflow sistema, pa je i njegova sintaksa, kao i raspoloživi okvir funkcija, različita.

Na slici 8, prikazana je radna površina alata za modeliranje poslovnih procesa *Intalio BPMS Designer*. Alat je otvorenog koda (*open-source*) i izgrađen je na *Eclipse SDK* platformi. Softver za modeliranje poslovnih procesa je integrisan i sa odgovarajućim serverskim rešenjem.

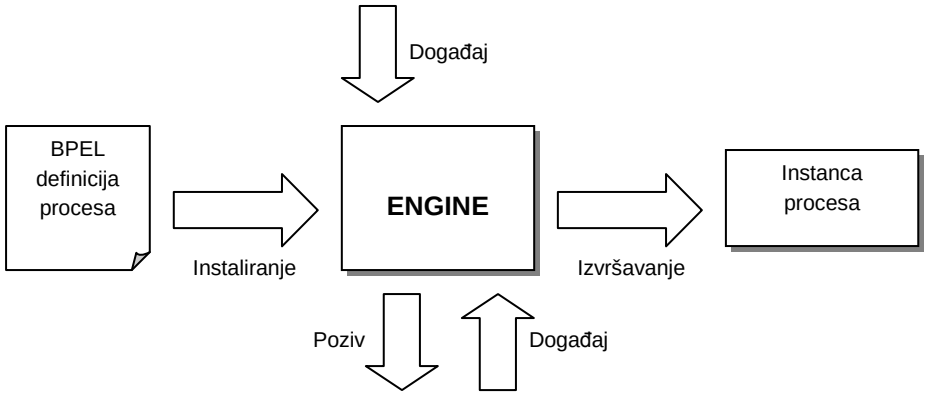


Slika 8. Radna površina alata za modeliranje poslovnih procesa Intalio BPMS Designer

Engine za podršku izvršavanju poslovnih procesa (BPM runtime engine)

Uloga *engine*-a je da prihvata definicije procesa (modele, izražene BPEL, ili nekim drugim jezikom) i izvršava njihove instance (procese). BPEL kodom se opisuje niz koraka, a *engine* ima ulogu da obezbedi okruženje za njihovo izvršavanje, u skladu sa opisanim pravilima.

Engine je baziran na događajima (*event-driven*) i uobičajeno, najveće vreme provodi u čekanju stimulansa koji "okida" izvesnu aktivnost, nakon čega ponovo prelazi u stanje čekanja. On je, takođe, zadužen i za komunikaciju sa eksternim procesima (internim sistemima i web servisima).



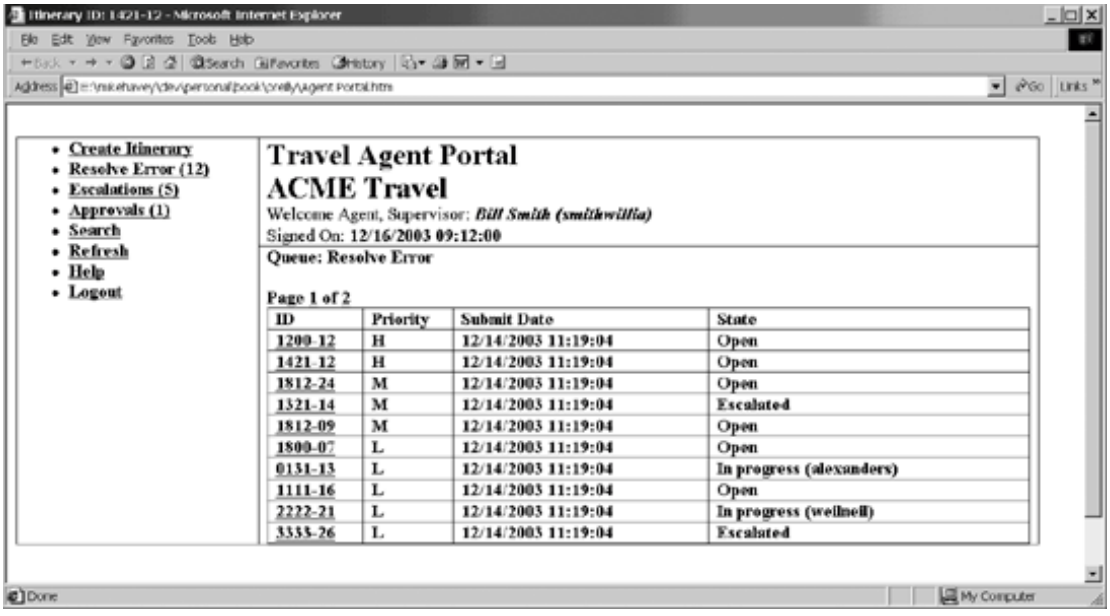
Slika 9. Funkcije BPM engine-a

Kao što je prikazano na slici 9, uloge BPM *engine*-a su: detekcija događaja kojima se stimuliše izvršavanje određene aktivnosti; konkretno izvršavanje; i upravljanje pozivima eksternih procedura, internih sistema i web servisa.

Interakcija korisnika sa BPM sistemom

Koncepti interakcije korisnika sa BPM sistemom, odnosno svi elementi manuelnih aktivnosti jednog poslovnog procesa, su preuzeti od *workflow* alata. Ovi koncepti su predstavljeni u ovom poglavlju.

Na slici 10, prikazan je primer koji demonstrira osnovne elemente korisničkog interfejsa, namenjenog interakciji korisnika sa BPM sistemom, u slučaju obavljanja manuelnih aktivnosti. Osnovna komponenta korisničkog interfejsa BPM sistema je radna lista (*queue*), prikazana u centralnom delu ekrana na slici 10.



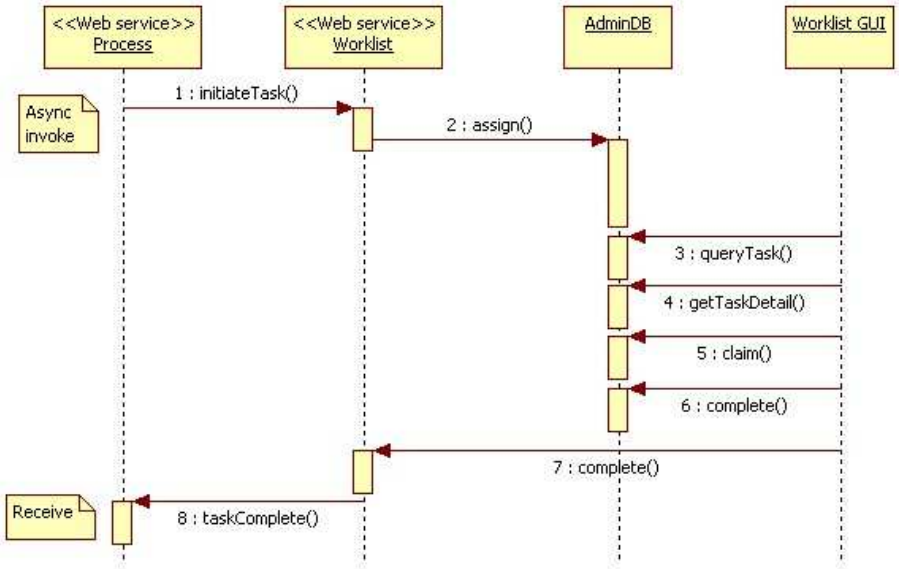
Slika 10. Osnovni elementi korisničkog interfejsa namenjenog korisniku BPM sistema

Radna lista se može prikazati za tri vrste aktivnosti: rešavanje problema (*Resolve Error*), eskalacije (*escalations*) i potvrde (*approvals*). Izborom neke od aktivnosti iz radne liste, prikazuje se novi prozor sa statusom i ostalim informacijama o akciji, kao i web formom za izvršavanje aktivnosti i prebacivanje u stanje čekanja.

Sa stanovišta realizacije BPM sistema radna lista može biti predstavljena generičkim web servisom `Worklist`, čije metode za inicijaciju jednog koraka (`initiateTask`) i njegov završetak

(`taskComplete`) pozivaju drugi web servis – proces (`Process`), tako što u prvom slučaju prosleđuju svoj identitet, a u drugom podatke unete u pomenutu web formu. Stanje jednog koraka se čuva perzistentnim tako što se sve informacije o njemu zapisuju u bazu podataka.

Implementaciona šema ove procedure je prikazana na slici 11.



Slika 11. Implementaciona šema slučaja korišćenja BPM sistema

Sa stanovišta procesa, identitet korisnika koji izvršava određeni korak je irelevantan. Ono što je uobičajeno važno je grupa kojoj on pripada, ili njegova uloga (*role*) ili odgovornost. Osnovne karakteristike sistema koji podržava elemente bezbednosti sa stanovišta odgovornosti na njim su:

- sposobnost da izgled korisničkog sistema i dostupnost funkcija prilagođava postojećim rolama aktivnog korisnika; i
- zaštita od prikaza ili izvršenja funkcija sistema koje nisu definisane kao dostupne rolama koje su dodeljene aktivnom korisniku.

Tipične akcije nad koracima u poslovnom procesu karakteristične za korisnike, odnosno, role, su:

- **Dodeljivanje (*assign*) –** Odgovornost nad izvršenjem određenog koraka u poslovnom procesu se dodeljuje korisniku ili grupi korisnika sa istom rolom;
- **Prihvatanje (*claim*) –** Korisnik koji pripada grupi korisnika sa određenom rolom, kojoj je dodeljena odgovornost nad izvršavanjem koraka u poslovnom procesu prihvata ličnu odgovornost za to;
- **Preuzimanje (*yank*)** - Korisnik koji pripada grupi korisnika sa određenom rolom, kojoj je dodeljena odgovornost nad izvršavanjem koraka u poslovnom procesu, preuzima odgovornost od drugog korisnika koji ju je već preuzeo;
- **Vraćanje (*balk*)** - Korisnik koji pripada grupi korisnika sa određenom rolom, kojoj je dodeljena odgovornost nad izvršavanjem koraka u poslovnom procesu vraća preuzetu odgovornost korisniku od kojeg ju je oduzeo.

Interakcija BPM sistema sa drugim sistemima

Veoma mali broj procesa se izvršava u izolovanom okruženju. Pored interakcije sa korisnicima, BPM sistem ostvaruje i određeni nivo interakcije sa komponentama drugih sistema, internih i eksternih, sa stanovišta preduzeća. Štaviše, ova funkcija BPM sistema, sa stanovišta koncepta servisno orijentisane arhitekture je i njegova primarna funkcija.

Eksterna interakcija se vrši putem web servisa, a interna – uz pomoć delova koda, ili klijent-server veza ostvarenih sa drugim sistemima. U tom smislu, čvorovi definicije procesa mogu biti:

- **Receive** – proces prima poruku od drugog sistema;
- **Receive-Reply** – proces prima poruku od drugog sistema i šalje nazad odgovor;
- **Send** – proces šalje poruku drugom sistemu;
- **Send-Receive** – proces šalje poruku drugom sistemu i čeka na odgovor.

Sa stanovišta tehnologije interne komunikacije, BPM sistem treba da podržava neposredni unos i administraciju delova koda, ali i fleksibilan, proširivi sistem za definisanje interfejsa prema drugim sistemima. Uobičajeno, u praksi se koristi *plug-in* model adaptera, koji se u Javi realizuje uz pomoć EJB *remote* interfejsa ili JCA[9] arhitekture. BPM Engine mora da implementira i JNDI[10], da bi registracija adaptera bila moguća.

Pored toga što savremeni BPM sistemi imaju mogućnosti komunikacije sa eksternim web servisima, oni su sami dostupni drugim sistemima primenom iste tehnologije. Engine treba da obuhvati i *listener* web servisa koji zna kako da prihvati SOAP poruku, ubaci je u proces, prihvati odgovor procesa i pošalje ga nazad kao SOAP poruku.

Administracija i monitoring

Administracija i monitoring BPM sistema obuhvata funkcije praćenja procesa čije je izvršavanje u toku, instaliranje novih definicija procesa, poništavanje procesa čije je izvršavanje zastalo, uklanjanje procesa koji više nisu potrebni poslovnom okruženju, itd.

U tabeli 3, prikazani su svi objekti BPM sistema koje je potrebno administrirati i aktivnosti administracije i monitoringa.

Upravljani objekti	Aktivnosti
Definicije procesa	1. Pretraga definicija procesa instaliranih u *engine*, sa filtrima; 2. *Deploy* – instaliranje nove definicije procesa u *engine*; 3. Uklanjanje definicije procesa; 4. Aktiviranje – Omogućavanje korisnicima da izvrše instanciranje postojeće definicije procesa, odnosno kreiranje novog procesa; 5. Deaktiviranje - Onemogućavanje korisnika da izvrše instanciranje postojeće definicije procesa, odnosno kreiranje novog procesa.
Procesi	1. Pretraga procesa, sa filtrima za izbor definicije, stanja procesa, datuma kreiranja i datuma završetka i promenljivih procesa; 2. *Suspend* – privremeno obustavljanje instance procesa; 3. *Resume* – ponovno pokretanje instance procesa nakon privremenog obustavljanja; 4. *Terminate* – definitivno obustavljanje instance procesa.
Aktivnosti i zadaci iz radne liste	1. Pretraga aktivnosti i zadataka iz radne liste, sa filtrima za izbor tipa role, procesa kojem aktivnost pripada, datuma početka i završetka; 2. *Suspend* – privremeno obustavljanje aktivnosti; 3. *Resume* – ponovno pokretanje aktivnosti nakon privremenog obustavljanja;

[9] JCA - Java Connector Architecture
[10] JNDI - Java Naming and Directory Interface

	4. *Terminate* – definitivno obustavljanje aktivnosti.
Korisnici i role	1. Dodavanje, izmena i uklanjanje korisnika; 2. Dodavanje ili uklanjanje rola; 3. Dodeljivanje ili uklanjanje rola korisnicima.
Aplikacije	1. Podešavanje konekcija aplikacija (web servisi, baza podataka,..) preko kojih se vrši komunikacija sa internim i eksternim sistemima.

Tabela 3. Aktivnosti administracije i monitoringa BPM sistema

Upravljanje verzijama procesa

Tokom svog životnog veka, definicije procesa se menjaju u postupcima uklanjanja bagova i drugih problema, usavršavanja, optimizacije i izrade novih interfejsa za korisnike i interne i eksterne sisteme. U sistemima u kojima instance definicije procesa – sami procesi, imaju dugo trajanje, svaka izmena definicije se mora izvršiti veoma pažljivo, da ne bi došlo do narušavanja integriteta postojećih instanci.

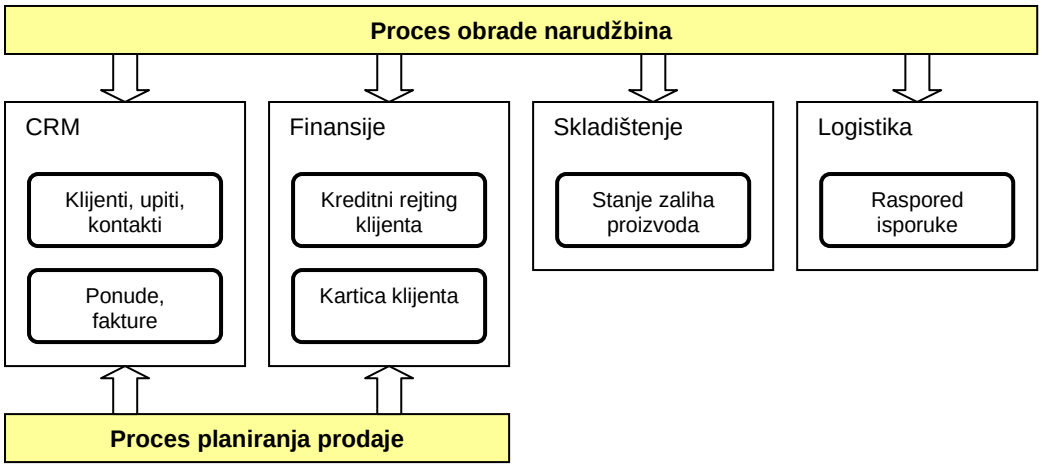
Ukoliko je u pitanju manja izmena, vezana za uređivanje ili usavršavanje aktivnosti, zamena koda se jednostavno izvršava, pri čemu se u toku zamene, uz pomoć alata za administraciju sve tekuće aktivnosti privremeno suspenduju, u vremenu izvršavanja zamene koda.

Ukoliko se vrši veća izmena definicije procesa, mora se sačuvati njegova prethodna verzija, a sistem treba da na nove okolnosti reaguje, tako što će sve ranije započete procese izvršavati prema starijoj definiciji. Svi procesi koji su započeti nakon izvršene izmene treba da se izvršavaju prema novoj definiciji.

Primer primene integrisanog SOA okruženja za realizaciju poslovnih procesa

Osnovna karakteristika ERP sistema je *integrisano* okruženje za distribuiranu manipulaciju istim skupovima podataka. Jedan od principa kojim se to postiže je primena centralnog, jedinstvenog repozitorijuma poslovnih informacija, realizovanog korišćenjem neke relacione baze podataka. Pored toga, jedna od karakteristika integrisanog ERP okruženja, u kontekstu razvoja servisno orijentisane arhitekture je i korišćenje centralnog, jedinstvenog registra poslovnih servisa. Njihova implementacija se može oslanjati na postojeći sloj informacionog sistema preduzeća, ili razvijene elemente poslovne logike koji koriste informacije iz centralnog repozitorijuma. U svakom slučaju, poslovni servisi predstavljaju osnovni sloj integrisanog okruženja. Njihovom implementacijom se ostvaruju principi jedinstvenog sloja informacionog sistema – centralne tačke pristupa u komunikaciji poslovnih aplikacija sa repozitorijumom podataka, odnosno elementima poslovne logike.

Na slici 12, između ostalog, kao primer je prikazana međuzavisnost procesa obrade narudžbina i različitih podataka, čije se upravljanje vrši u okviru ERP repozitorijuma. Proces obrade narudžbina, kao i ostali poslovni procesi podržani ERP sistemom, podrazumevaju jedinstvenu manipulaciju nad različitim repozitorijumima podataka i funkcija, za čije je održavanje odgovornost distribuirana. Na osnovu informacija koje radnik dobija iz *finansijskog modula* ERP sistema, on može da dobije uvid u to da li je klijent platio poslednju ili sve dosadašnje narudžbine, odnosno, ukoliko nije, kakvo je njegovo kreditno stanje. Na osnovu funkcija i podataka iz modula za magacinsko poslovanje, radnik može da dobije informaciju o tome da li na zalihama postoji proizvod; ukoliko ne postoji, kada će postojati, itd. Iz logističkog modula, radnik može da dobije informaciju o terminima isporuka u region iz kojeg je klijent koji je poslao narudžbinu, i na taj način, obezbedi informaciju o mogućem vremenu isporuke.



Slika 12. Primer integrisanog poslovnog okruženja

Očigledno je da proces obrade narudžbina zavisi od potpunosti i tačnosti informacija kojima su snabdeveni ERP repozitorijumi. Njihova tačnost i potpunost zavisi od ažurnosti i odgovornosti radnika iz određenih sektora preduzeća koje ih održavaju. Jedan od snažnih argumenata za ovo je i procena da upotrebljivost jednog MRP II sistema dostiže nivo potreban za ostvarivanje koristi samo ukoliko je najmanje 98% podataka o zalihama tačno, potpuno i ažurno.

Realizacija jednog poslovnog procesa uz pomoć servisno orijentisane arhitekture se sastoji od sledećih faza:

- Generički opis procesa
- Dekompozicija procesa
- Implementacija servisa i
- Orkestracija servisa

Realizacija procesa predstavlja iterativni postupak – faze realizacije se ponavljaju u određenom broju iteracija – revizija, u cilju ispunjenja osnovnog cilja. Prve dve faze realizacije se odvijaju sekvencijalno, dok se implementacija servisa i njihova orkestracija mogu vršiti uporedno. Faze realizacije su detaljnije opisane u daljem tekstu.

1. Generički opis procesa. Generički opis procesa obuhvata izradu dijagrama aktivnosti sa pregledom svih koraka koje je potrebno sprovesti u cilju uspešnog sprovođenja poslovnog procesa. U okviru ove faze razvoja, potrebno je identifikovati:

- Sve aktivnosti poslovnog procesa i njihov redosled;
- Automatske i manuelne aktivnosti procesa;
- Rizike od uspešnog izvršavanja procesa; i
- Sinhrone i asinhrone aktivnosti procesa

Ovu klasifikaciju je potrebno izvršiti u fazi generičkog opisivanja procesa, jer se tokom njegove dekompozicije vrši implementacija interfejsa koji se koristi za opis servisa.

2. Dekompozicija procesa. Osnovna aktivnost implementacije jednog poslovnog procesa u servisno orijentisanoj arhitekturi je identifikacija servisa. Ona se sprovodi *top-down* dekompozicijom procesa. U okviru dekompozicije procesa, vrši se:

- Identifikacija servisa primenom metode iterativne dekompozicije procesa i njihova klasifikacija prema kriterijumu vlasništva nad servisima (servisi procesa i servisi partnera, različitih organizacionih celina) i njihovoj strukturi;
- Faktorisanje servisa - modeliranje servisa iz perspektive njihovog vlasništva, njihova klasifikacija prema kriterijumima odgovornosti nad podacima i dostupnosti;

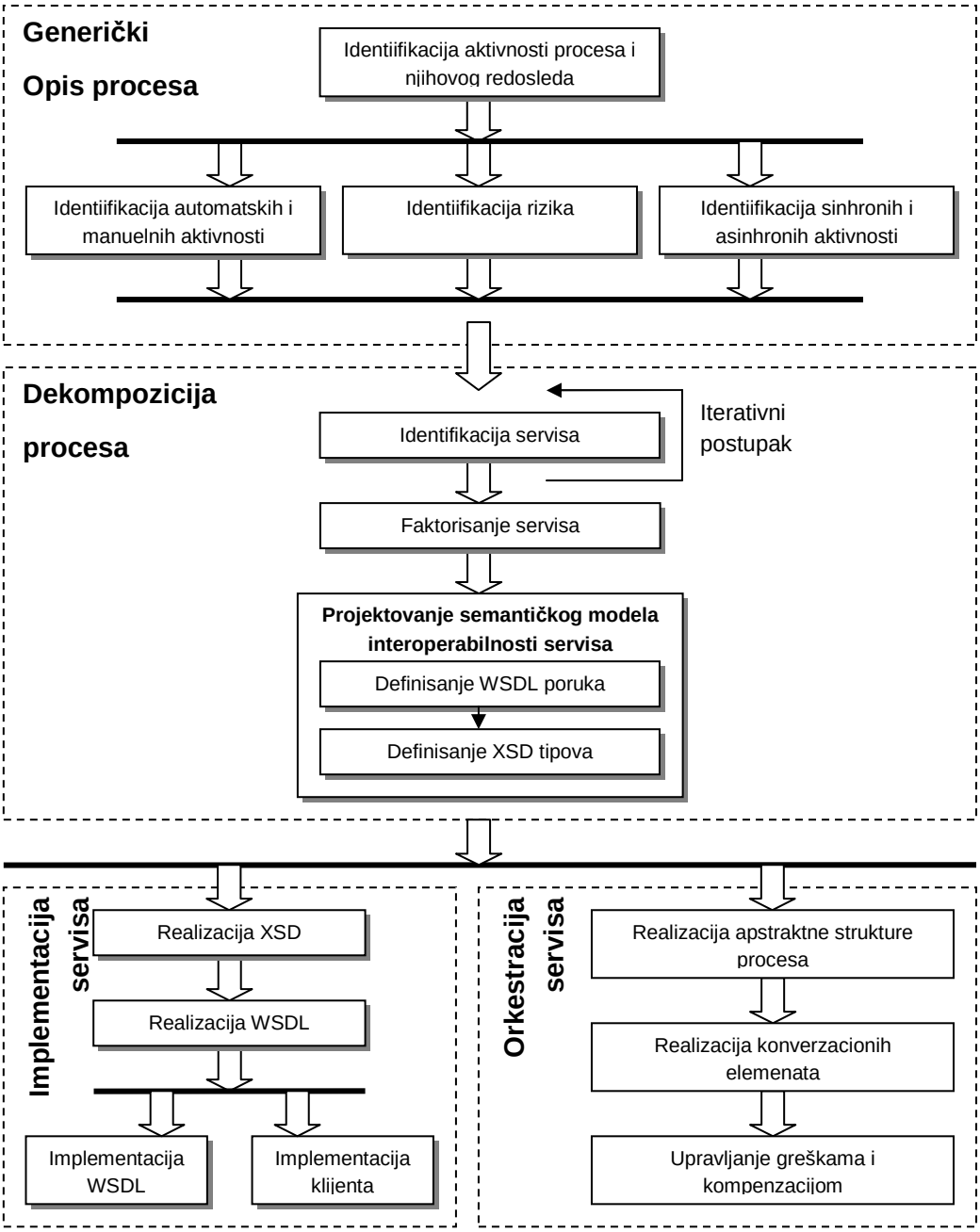
- Projektovanje semantičkog modela interoperabilnosti servisa - identifikacija i definicija poruka koje se razmenjuju između korisnika i provajdera svih identifikovanih servisa;

3. Implementacija servisa. Dekompozicijom procesa se stvaraju uslovi za realizaciju funkcionalnog modela servisa kao i semantičkog modela njihove interoperabilnosti. Na osnovu izrađenih simboličkih modela, vrši se realizacija XSD rečnika, kao i samih WSDL interfejsa, a potom i njihova implementacija. Implementacija servisa obuhvata sledeće aktivnosti:

- Realizacija XSD rečnika
- Realizacija WSDL interfejsa
- Implementacija WSDL interfejsa i
- Implementacija klijenta.

4. Orkestracija servisa. Orkestracija servisa u servisno-orijentisanoj arhitekturi se vrši realizacijom BPEL strukture za izvršenje poslovnih procesa i njenom instalacijom u odgovarajući engine. Aktivnosti na razvoju okruženja za orkestraciju servisa jednog poslovnog procesa su:

- Realizacija apstraktne definicije procesa;
- Realizacija konverzacionih elemenata servisno orijentisane arhitekture; i
- Realizacija mehanizama za upravljanje događajima, alarmima, greškama i kompenzacijom procesa.

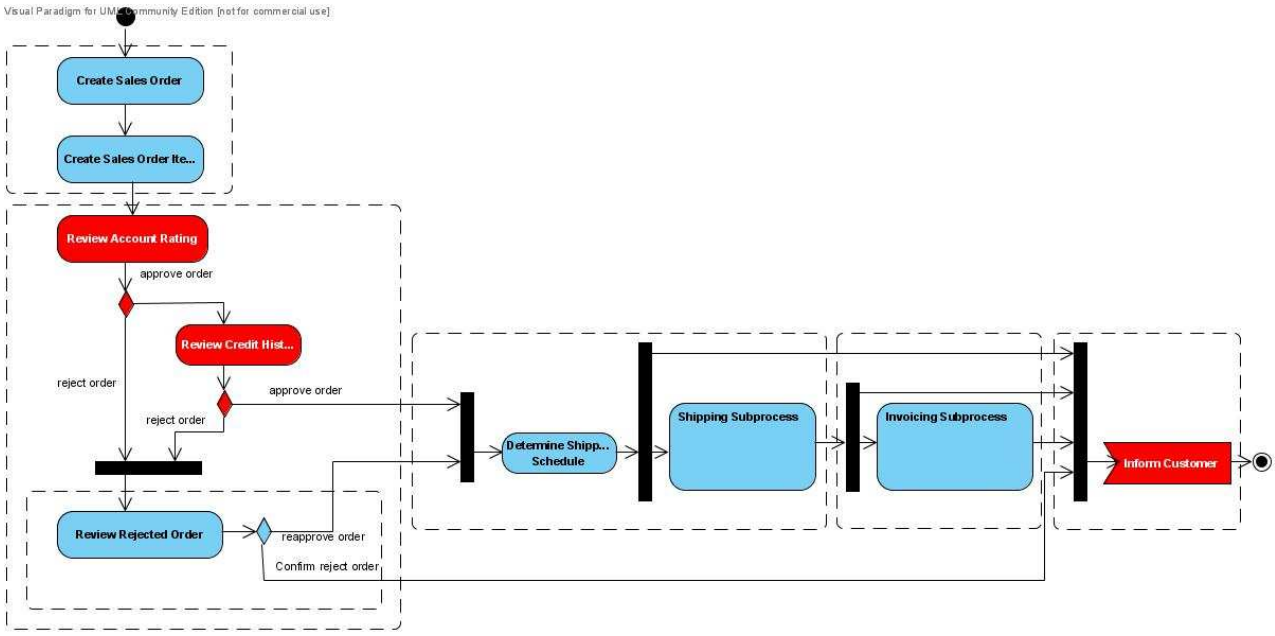


Slika 13. Šema realizacije poslovnog procesa u SOA okruženju

Generički opis poslovnog procesa obrade narudžbina

Jedan od najreferentnijih primera primene ERP sistema u poslovnom okruženju je unos i obrada narudžbina. Proces obrade narudžbina (*order fullfillment*) predstavlja transformaciju narudžbine klijenta u fakturu, odnosno prihod (proces ne obuhvata postupak prodaje, koji podržavaju odgovarajući CRM sistemi), pri čemu su, u različitim fazama obrade, za donošenje relevantnih odluka, potrebni različiti tipovi informacija. Proces obrade narudžbine obuhvata veći broj aktivnosti za koje su odgovorni različiti akteri u okviru različitih sektora preduzeća. Primenom ERP sistema, jedan broj odluka se može donositi automatizovano, pri čemu se vrši i preraspodela odgovornosti, u odnosu na konvencionalno poslovanje, koje se vrši u okviru nezavisnih, nepovezanih grupa aktivnosti. ERP sistem obezbeđuje i tačan uvid u to na kakvom se nivou transformacije nalazi narudžbina, odnosno u čijem se domenu odgovornosti trenutno nalaze aktivnosti vezane za njenu transformaciju.

Sveobuhvatan proces obrade narudžbina je predstavljen UML dijagramom aktivnosti na slici 14.



Slika 14. UML dijagram aktivnosti primera procesa obrade marudžbina

Na slici 14, aktivnosti sa pozadinom crvene boje se izvršavaju automatski, dok su aktivnosti sa plavom pozadinom – manuelne. Isprekidanom linijom su uokvirene aktivnosti koje čine posebne podprocese:

- kreiranje narudžbine,
- analizu rizika realizacije narudžbine,
- isporuku,
- kreiranje računa i
- obaveštavanje klijenta.

Jedna od osnovnih pretpostavki za identifikaciju struktura interfejsa pojedinačnih servisa, odgovornih za sprovođenje navedenih aktivnosti je identifikacija sinhronih i asinhronih aktivnosti. U ovom slučaju, sa stanovišta procesa obrade narudžbina u *make-to-stock* modelu, asinhrone aktivnosti se sprovode u okviru:

- određivanja rasporeda isporuke,
- podprocesa isporuke (*shipping subprocess*) i
- obrade fakture (invoicing subprocess).

Važno je napomenuti da je proces obrade narudžbina asinhrone prirode – ukoliko je najmanje jedna aktivnost procesa asinhrona, proces je, u celini, asinhrone prirode.

Pojedinačne aktivnosti procesa obrade narudžbine su opisane u daljem tekstu.

Proces započinje kreiranjem narudžbine od strane predstavnika službe za odnose sa klijentima (*Customer Service Representative*). U okviru ovog postupka, on je u obavezi da uspostavi referencu sa prethodno uspostavljenim odgovarajućim odnosom sa klijentom, koji može biti običan upit, ponuda, dugoročni ugovor o saradnji, itd. Referenca se uspostavlja radi automatskog izbora postojećeg ili kreiranja novog konta klijenta koji će se koristiti za obradu narudžbine. Pored toga, uspostavljanjem ove reference, održava se međuzavisnost svih poslovnih procesa preduzeća i stvara osnova za sticanje uvida u njegovu ukupnu efikasnost i produktivnost.

Važan parametar narudžbine je i način isporuke. *Pojedinačna isporuka* podrazumeva da će klijentu biti pakovani i dostavljani proizvodi iz pojedinačnih stavki narudžbine, u kratkom roku od kojeg su oni

dostupni u magacinima, u okviru ATP[11] zaliha. *Grupna isporuka* znači da će svi proizvodi u jedinstvenoj isporuci biti dostavljeni klijentu, u kratkom roku nakon najkasnije dostupnosti bilo kog od njih. Dalje, isporuka se može izvršiti u okviru redovnih termina isporuke u region u kojem se nalazi klijent, ili posebnim transportom, čime se smanjuje vreme isporuke ali i povećavaju njeni troškovi.

Unos stavki narudžbine (Create Sales Order Items). U sledećem koraku, predstavnik službe za odnose sa klijentima, u ponavljajućem postupku unosi stavke narudžbine. Stavke narudžbine karakteriše proizvod, količina i najraniji datum kada definisana količina određenog proizvoda postoji u nekom od magacina preduzeća, u okviru ATP zaliha.

Nakon kreiranja narudžbine, odgovornost za naredne postupke prelazi na sistem za automatsku evaluaciju rizika i predstavnika prodajne službe (*Sales Representative*), koji je odgovoran za reviziju automatskog odbijanja narudžbine, preporučenog na osnovu izvršene analize rizika.

Analiza rizika realizacije određene narudžbine se vrši procenom konta klijenta u okviru kojeg se narudžbina vrši, odnosno analizom njegove kreditne sposobnosti, i procenom ažurnosti klijenta, na osnovu istorije zaduživanja. Svaka od ovih analiza se vrši automatski, pri čemu proces autonomno odlučuje o prihvatanju ili preporučuje odbijanje narudžbine. Negativna preporuka procesa može da se revidira od strane predstavnika prodajne službe, dok prihvatanje narudžbine, sa stanovišta analize rizika koju sistem vrši automatski, inicira dalje odvijanje procesa obrade narudžbine.

Procena kreditne sposobnosti klijenta (Review Account Rating). Procena kreditne sposobnosti klijenta se vrši na osnovu ukupnog ostvarenog prihoda i nivoa trenutnog zaduženja, koje karakterišu fakture koje još uvek nisu plaćene.

Poslovnom politikom preduzeća definišu se egzaktni kriterijumi na osnovu kojih se vrši faktorisanje i evaluacija zaduženja konta klijenta, odnosno definiše preporuka za prihvatanje ili odbijanje narudžbine. U ovoj fazi procene rizika, na osnovu pomenutih parametara, sistem može da preporuči da se narudžbina odbija zbog visokog nivoa zaduženja, u odnosu na ukupan prihod konta, ili da se evaluacija rizika njene realizacije nastavi.

Procena ažurnosti klijenta (Review Credit History). Ukupan obim zaduženja, odnosno kreditna sposobnost klijenta nije jedini relevantan parametar za procenu rizika realizacije narudžbine. Pored toga što se sagledava relativno u odnosu na ukupan prihod na konto, potrebno je razmotriti i faktore istorije zaduženja.

Njih može da karakteriše prosečno vreme potrebno za plaćanje dostavljenih faktura, ukupno ili u poslednjem periodu; poslednji datum priliva na konto, itd. Na osnovu ovih podataka, sistem prihvata narudžbinu ili preporučuje odbijanje narudžbine.

Odbijanje narudžbine predstavlja svojevrsni izuzetak sa stanovišta ukupnog poslovanja, tako da je potrebno da predstavnik prodajne službe izvrši analizu svake odbijene narudžbine i, na osnovu uvida u sve podatke, odnosno, razloge zbog kojih je sistem preporučio odbijanje narudžbinu, odbijanje autorizuje, ili narudžbinu ipak dozvoli i na taj način - vrati u proces obrade.

Procena negativne preporuke se vrši na osnovu rezultata analize rizika i svih informacija o klijentu – njegovog potpunog profila, odnosno njegovim kontima.

Definisanje termina isporuke se vrši na osnovu podataka o načinu isporuke, unetih prilikom kreiranja narudžbine, terminima transporta u region u kojem se nalazi klijent, odnosno njihovim popunjenostima i dostupnosti odgovarajućih resursa u izlaznim magacinima i transportnih sredstava.

Termin ili termini isporuke se određuju automatski, pri čemu je potrebno generisati naloge za isporuku koji se dostavljaju izlaznim magacinima.

[11] ATP - Available To Promise

Iako sama isporuka ima određeni uticaj na proces transformacije narudžbine u *make-to-stock* modelu poslovanja, on je zanemarljiv, tako da ovaj podproces neće biti posebno obrađen u primeru.

Proces generisanja fakture predstavlja odgovornost službenika računovodstva. S obzirom na jednostavnost podprocesa i zanemarljiv uticaj na sveobuhvatni proces obrade narudžbine, on neće biti posebno obrađen u primeru.

U određenim fazama obrade narudžbine, obavezno se vrši informisanje klijenta o stanju njene transformacije. Informisanje klijenta se vrši:

- U slučaju odbijanja narudžbine. Klijent se obaveštava o odbijanju narudžbine i razlozima za to;
- U trenutku definisanja preciznog ili preciznih datuma isporuke. Klijent se obaveštava o terminu ili terminima isporuke, u zavisnosti od izabranog načina isporuke;
- U trenutku isporuke. Klijent se šalje potvrda o izvršenoj isporuci;
- U trenutku generisanja odgovarajuće fakture. Klijentu se šalje faktura.

Dekompozicija poslovnog procesa

Identifikacija servisa se obavlja u okviru aktivnosti dekompozicije servisno orijentisane arhitekture, odnosno poslovnog procesa. Dekompozicija jednog korporativnog procesa pretpostavlja:

- identifikaciju hijerarhije servisa, u kontekstu organizacije, procesa i funkcija poslovnog sistema,
- faktorisanje identifikovanih servisa, u cilju obezbeđenja principa i standarda performansi, proširivosti, bezbednosti, itd.
- projektovanje semantičkog modela koji treba da ustanovi smernice za međusobnu komunikaciju servisa i obezbeđivanje njihove interoperabilnosti;

Za dekompoziciju procesa, odnosno identifikaciju hijerarhijskog modela servisa, koristi se *top-down* princip. *Top-down* princip dekompozicije jednog korporativnog procesa podrazumeva izvršenje iterativnog postupka identifikacije servisa koje on orkestrira i drugih procesa koji su odgovorni za njihovu implementaciju. Iteracijom ovog postupka po dubini hijerarhije, može se izgraditi potpuni hijerarhijski model SOA arhitekture, odnosno svih njenih zavisnih elemenata neophodnih za sprovođenje osnovnog korporativnog procesa.

Identifikacija servisa procesa obrade narudžbina

Identifikacija servisa procesa obrade narudžbina se vrši na osnovu rezultata dekompozicije ovog procesa, koja je prikazana na slici 15. Prema prikazanom primeru, dekompozicija procesa obrade narudžbina se obavlja u dve iteracije koju obuhvataju sledeći koraci:

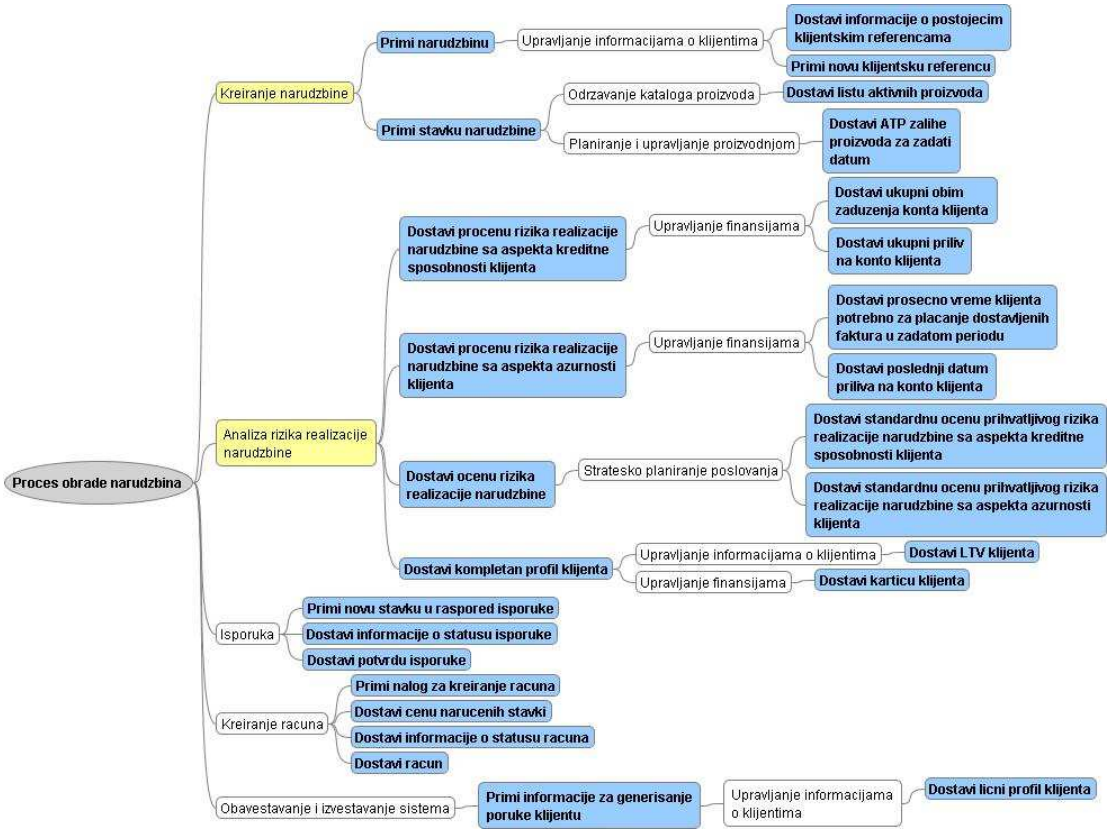
Iteracija 1:

- Identifikacija aktivnosti ili podprocesa procesa obrade narudžbina,
- Identifikacija servisa koje direktno orkestrira proces obrade narudžbina,

Iteracija 2:

- Identifikacija poslovnih procesa koji su odgovorni za implementaciju servisa koje direktno orkestrira proces obrade narudžbina,
- Identifikacija servisa koje orkestriraju odgovorni identifikovani poslovni procesi

Na šemi dekompozicije procesa obrade narudžbina, prikazanoj na slici 15, identifikovane aktivnosti su prikazane elementima sa pozadinom žute boje, procesi ili podprocesi – elementima sa pozadinom bele boje, dok su identifikovani servisi predstavljeni elementima sa plavom pozadinom i tekstom ispisanim kurzivom.



Slika 15. Dekompozicija procesa obrade narudžbina

5.2.2.2. Faktorisanje servisa procesa obrade narudžbina

Analiza servisa koje koriste uočeni procesi je ograničena na servise od interesa za izvršenje procesa obrade narudžbina. Na osnovu izvršene dekompozicije, uočeni su sledeći složeni poslovni procesi od kojih zavisi izvršenje procesa obrade narudžbina:

- Upravljanje informacija o klijentima
- Održavanje kataloga proizvoda
- Planiranje i upravljanje proizvodnjom
- Upravljanje finansijama
- Strateško planiranje poslovanja
- Isporuka

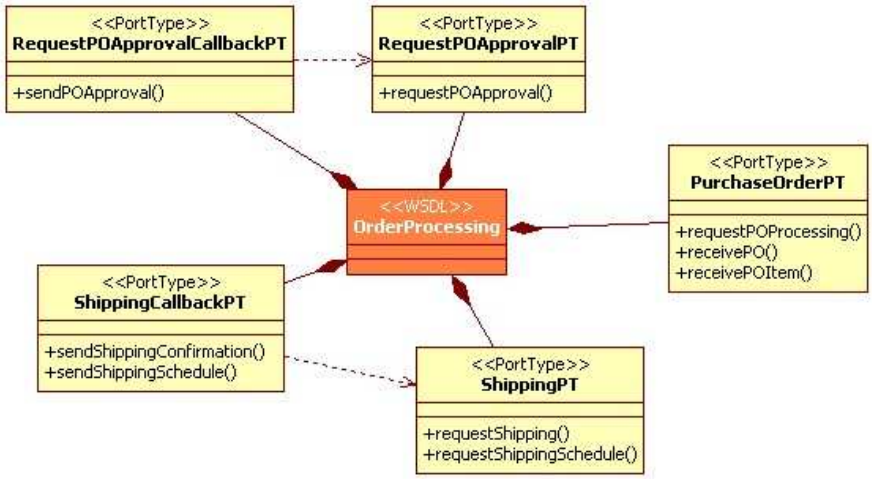
Navedeni složeni poslovni procesi su identifikovani sa stanovišta ukupne organizacije poslovanja u jednom preduzeću i u najvećem broju slučajeva su implementirani u okviru jedne njegove organizacione celine. Svaka organizaciona celina predstavlja jednog *partnera* u orkestraciji procesa koji se protežu horizontalno duž cele njegove poslovne infrastrukture. Primer ovakvog horizontalnog procesa je upravo proces obrade narudžbina.

Svaki od partnera jednog horizontalnog procesa je *vlasnik* određenog broja servisa, potrebnih za njegovo sprovođenje. Izuzetak predstavljaju servisi koji su direktno odgovorni za sprovođenje horizontalnog procesa (prvi nivo iteracije), čiji vlasnik nije partner procesa, već sam proces. Ovaj oblik dekompozicije procesa se koristi zbog identifikacije partnera servisno orijentisane arhitekture, čiji servisi će biti predstavljeni jednim interfejsom, odnosno jednom WSDL datotekom. Jedan interfejs obuhvata servise čiji je vlasnik partner servisno orijentisane arhitekture, odnosno one servise čija se dostupnost može obezbediti unutar jednog segmenta komunikacione infrastrukture preduzeća.

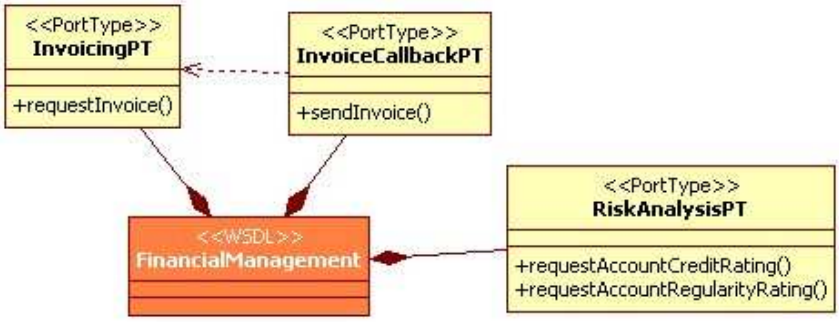
Osnovni principi za modeliranje interfejsa jednog partnera su:

- Servisi jednog partnera će biti grupisani unutar jednog interfejsa na strukture (*port types*) koje će implementirati posebni mrežni završeci (portovi).
- Strukture obuhvataju servise koji pripadaju jednom funkcionalnom segmentu organizacione celine – vlasnika servisa.
- Posebna struktura se predviđa za svaku *callback* operaciju asinhrone aktivnosti – operaciju "vraćanja" rezultata, na osnovu prethodno poslatog zahteva.

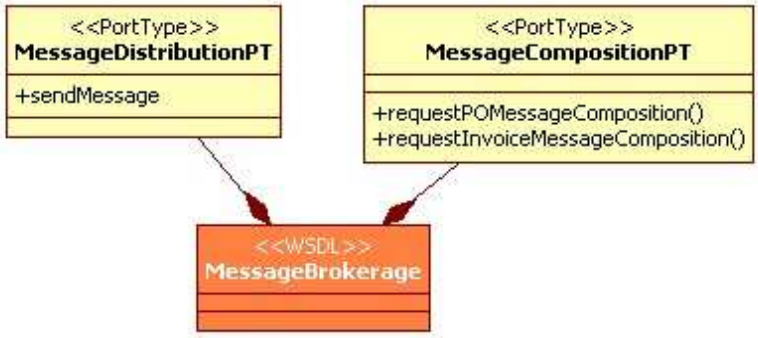
Na slikama 16-19, prikazani su modeli interfejsa svih partnera procesa obrade narudžbine. Modeli interfejsa su predstavljeni UML notacijom i obuhvataju sledeće nivoe: WSDL datoteka, tipovi portova (*port types*) i operacije (*operation*).



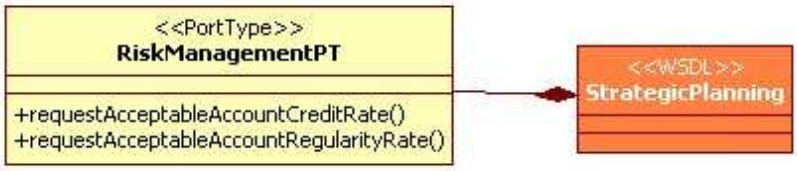
Slika 16. Model interfejsa partnera procesa obrade narudžbina



Slika 17. Model interfejsa partnera - finansijske službe



Slika 18. Model interfejsa partnera - agenta za distribuciju poruka



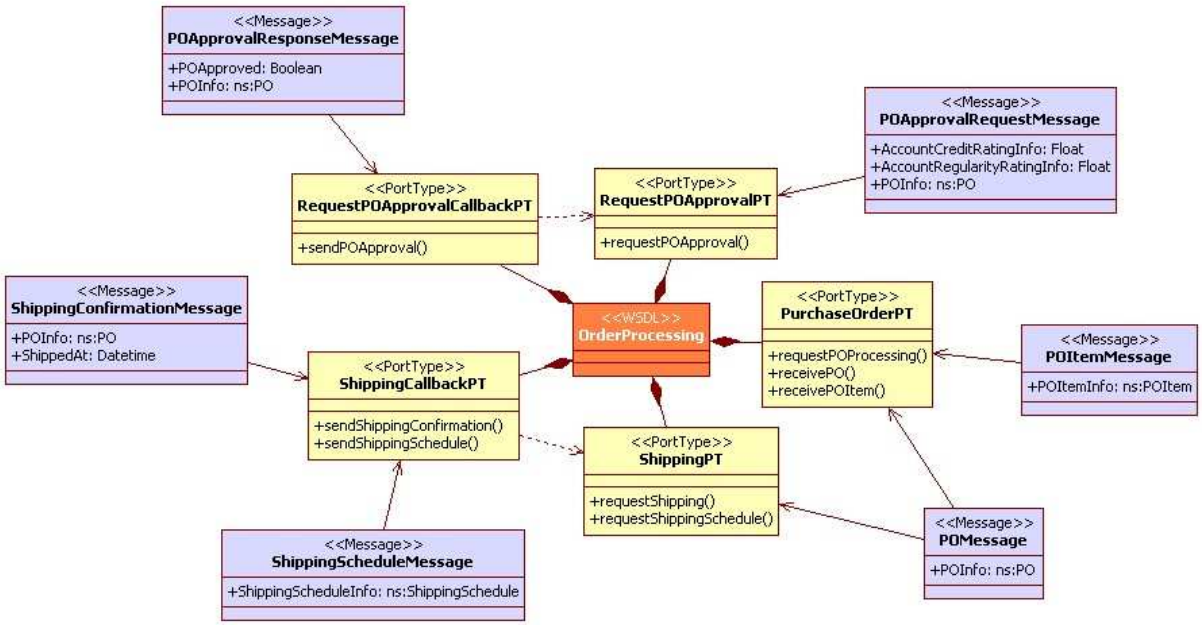
Slika 19. Model interfejsa partnera - službe strateškog planiranja

Projektovanje semantičkog modela interoperabilnosti servisa poslovnog procesa

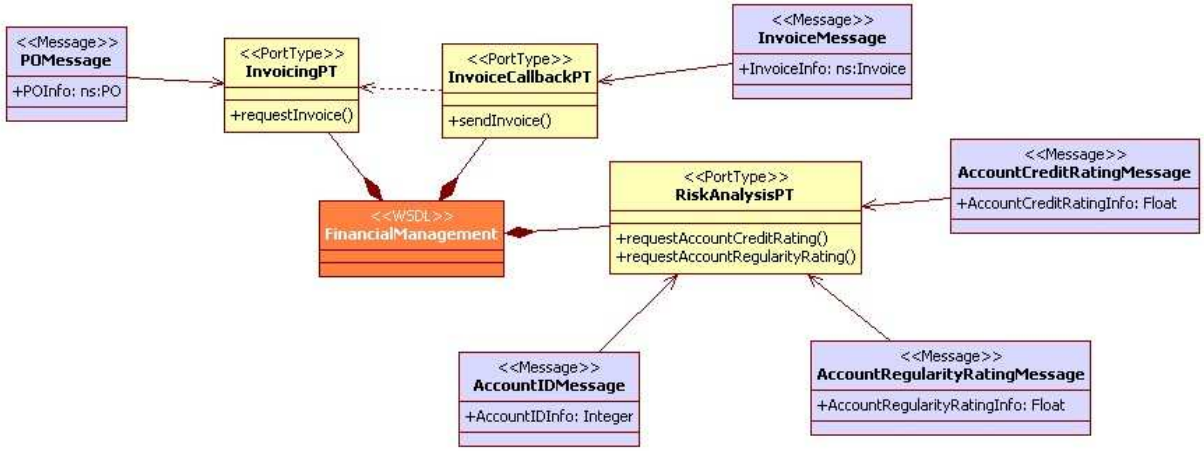
Poslednju fazu dekompozicije servisno orijentisane arhitekture karakteriše detaljno definisanje interfejsa svakog od uočenih servisa, odnosno, specifikacija poruka za pozivanje servisa i isporuku odgovarajućeg rezultata. U praksi, to podrazumeva specifikaciju modela podataka koji se koriste u razmeni poruka između operacija WSDL datoteke i korisnika odgovarajućeg web servisa.

Iako WSDL standard omogućuje definisanje tipova podataka poruka koje se razmenjuju u okviru komunikacije korisnika i provajdera servisa, detaljnu specifikaciju poruka je potrebno podržati definisanjem XSD tipova.

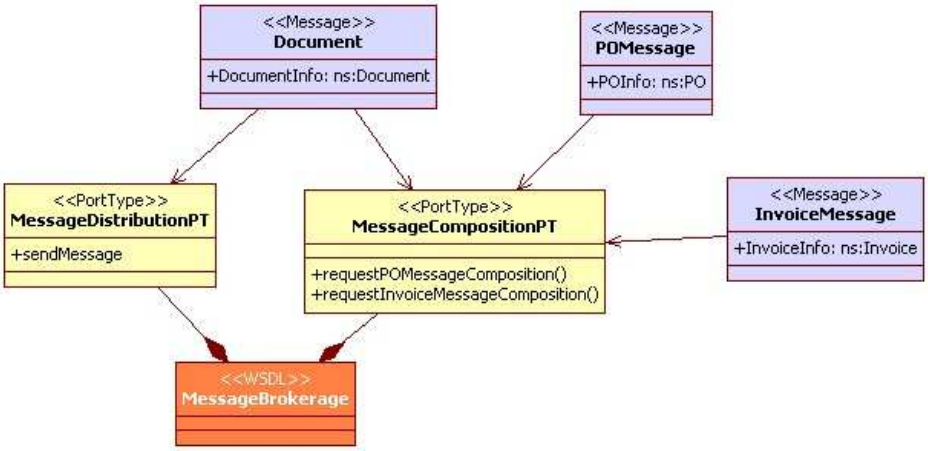
Prvi korak faze detaljne definicije tipova podataka relevantnih za model interfejsa web servisa predstavlja identifikacija poruka koje web servisi razmenjuju. Ona se vrši na osnovu modela interfejsa, a njen rezultat, relevantan za proces obrade narudžbina, prikazan je na slikama 20-23.



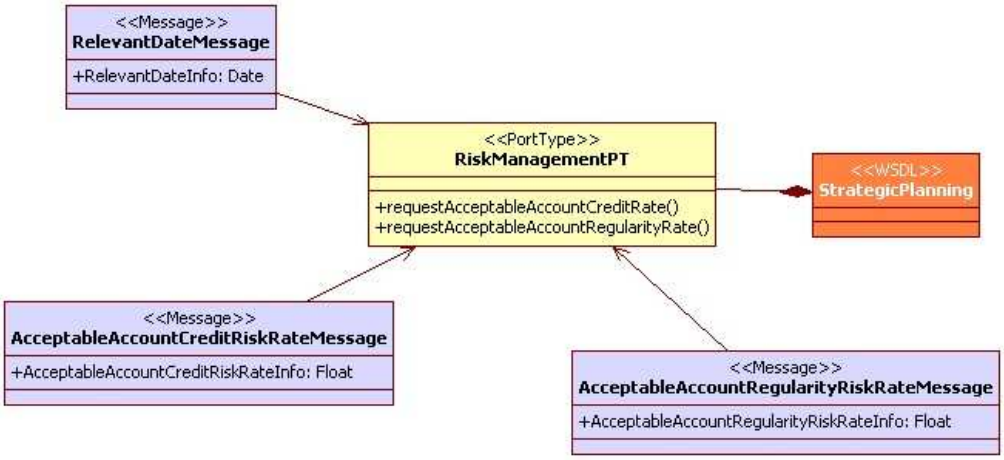
Slika 20. Semantički model interoperabilnosti partnera procesa obrade narudžbina



Slika 21. Semantički model interoperabilnosti partnera - finansijske službe



Slika 22. Semantički model interoperabilnosti partnera - agenta za distribuciju poruka



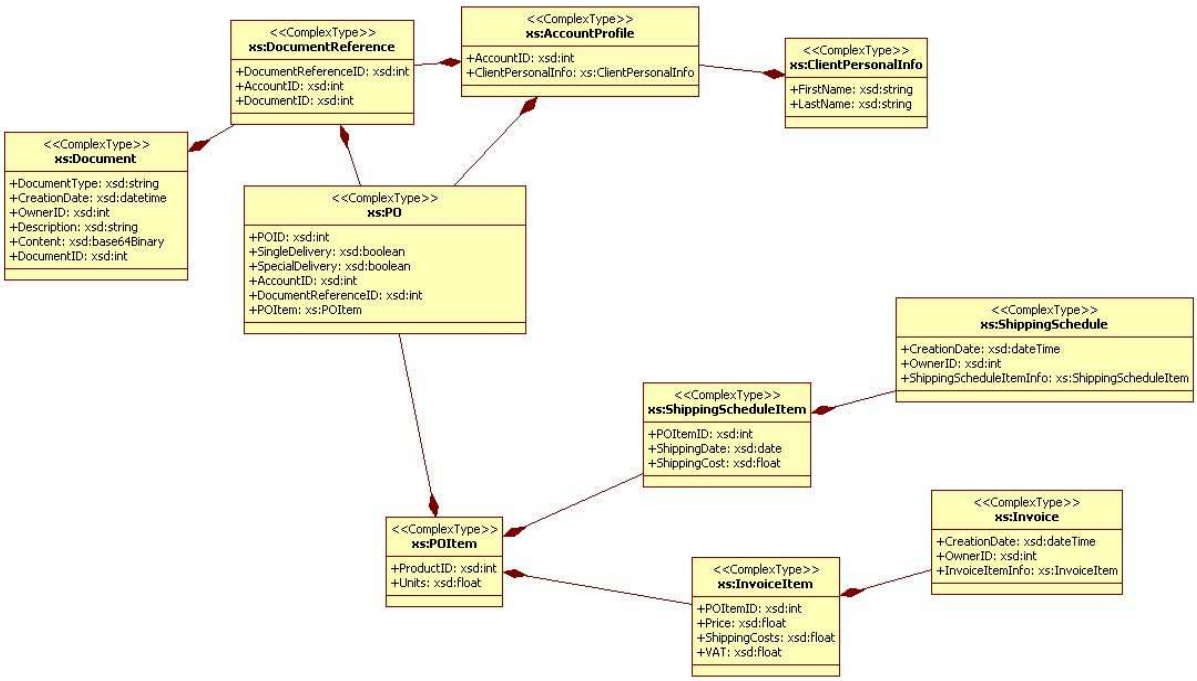
Slika 23. Semantički model interoperabilnosti partnera - službe strateškog planiranja

Drugi korak u projektovanju semantičkog modela interoperabilnosti servisa procesa narudžbina predstavlja definisanje tipova – detaljna specifikacija modela podataka koji se koriste u razmeni poruka između operacija WSDL datoteke i korisnika odgovarajućeg web servisa.

Detaljnu specifikaciju modela podataka podrazumeva refaktorisanje tipova podataka, identifikovanih tokom aktivnosti definisanja poruka koje razmenjuju WSDL interfejsi i korisnici web servisa. Refaktorisanje tipova podataka obuhvata:

- *Identifikaciju tipova koje koriste više web servisa – WSDL interfejsa.* Tipovi podataka koje karakteriše višestruko korišćenje u različitim WSDL interfejsima se definišu u XSD datoteci – zajedničkom rečniku podataka za proces obrade narudžbina.
- *Identifikaciju uzajamnih relacija između tipova podataka.* Ukoliko se u WSDL interfejsu koriste poruke čiji je jedan ili više delova (*part*) predstavljen u drugom WSDL interfejsu, onda se taj deo ili ti delovi definišu u XSD datoteci.
- *Analizu porekla podataka određenog tipa.* Ukoliko se utvrdi da se podaci određenog tipa dobijaju na osnovu određene (složene) kalkulacije, taj tip se definiše u WSDL datoteci. U suprotnom, tip podatka (pod uslovom da je u prethodnim koracima refaktorisanja, utvrđeno da je tip potrebno definisati u XSD rečniku) se definiše u XSD datoteci.
- *Identifikaciju strukture tipova podataka.*

S obzirom na to da se u fazi identifikacije strukture tipova podataka definišu novi tipovi – delovi postojećih, očigledno je da refaktorisanje treba da ima iterativni karakter. Postupak refaktorisanja se vrši do trenutka kada je ustanovljeno da je struktura svih tipova podataka potpuno i jednoznačno definisana. Struktura XSD rečnika je prikazana na slici 24 i, zajedno sa tipovima definisanim u okviru WSDL interfejsa, čini ukupan semantički model interoperabilnosti servisa poslovnog procesa obrade narudžbina.



Slika 24. Struktura XSD rečnika

Implementacija servisa

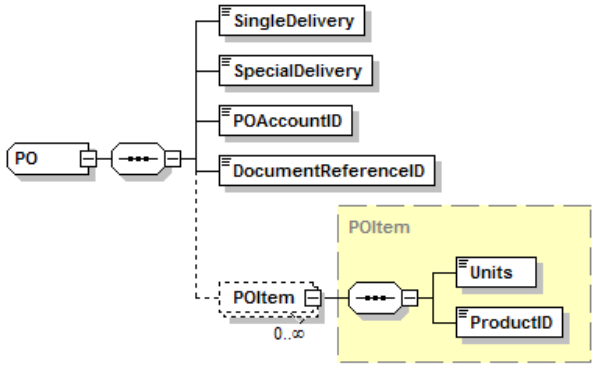
Implementacija servisa se vrši na osnovu rezultata faze dekompozicije poslovnog procesa, u kojoj je simbolički definisana osnova za nju. Aktivnosti implementacije servisa su:

- Realizacija XSD rečnika
- Realizacija WSDL interfejsa, i
- Implementacija WSDL interfejsa

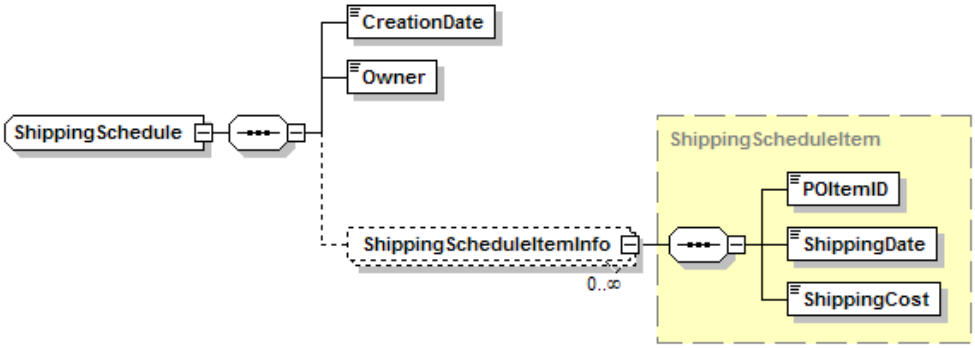
5.2.3.1. Realizacija XSD rečnika

Na osnovu projekta semantičkog modela interoperabilnosti servisa poslovnog procesa, odnosno rezultata aktivnosti refaktorisanja tipova podataka, vrši se realizacija XSD rečnika. Implementacija XSD rečnika se uobičajeno vrši uz pomoć nekog od alata za podršku kreiranju i održavanju XML struktura.

Na slikama 25 i 26, kao primer su prikazani dijagrami kompleksnih XSD tipova PO (*PurchaseOrder*) i *ShippingSchedule*, generisani od strane ovog alata.



Slika 25. Dijagram kompleksnog XSD tipa – PO (*PurchaseOrder*)



Slika 26. Dijagram kompleksnog XSD tipa – ShippingSchedule

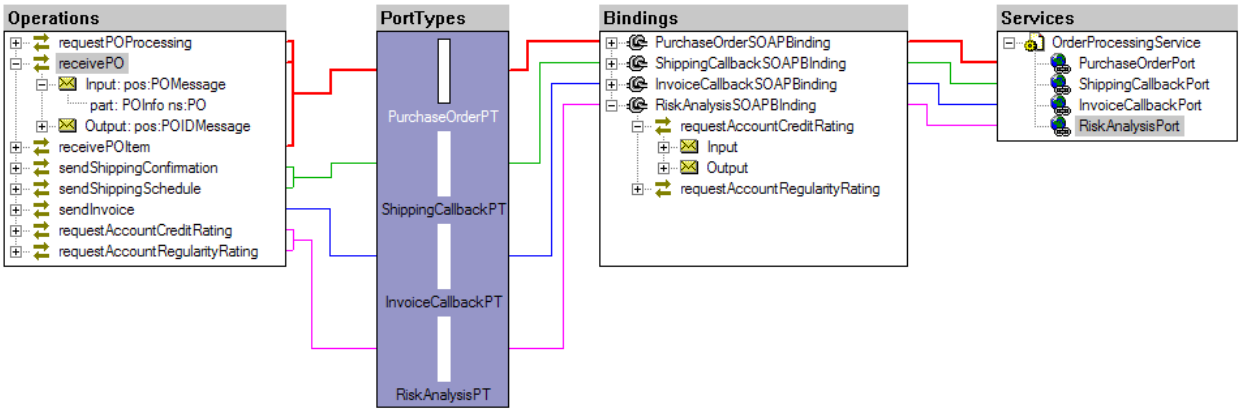
Realizacija WSDL interfejsa

WSDL standard igra ključnu ulogu u ostvarenju i praktičnoj realizaciji mnogih prednosti servisno orijentisane arhitekture u odnosu na tradicionalne tehnologije integrisanja koporativnog informacionog sistema. U servisno orijentisanoj arhitekturi, WSDL interfejsi predstavljaju direktnu realizaciju servisnih ugovora, odnosno, obezbeđuju jednostavno definisanje strukturnih atributa servisa – njihovih meta podataka.

Realizacija WSDL interfejsa predstavlja jednu od ključnih aktivnosti razvoja servisno orijentisane arhitekture. Razvijen WSDL interfejs obuhvata kompletnu implementaciju servisnog ugovora i služi kao osnova za uporedni nastavak njegovog razvoja – u smeru implementacije klijenta servisa, predstavljenog WSDL interfejsom i u smeru implementacije samog servisa – programski razvoj logičke arhitekture za predstavljanje podataka i funkcionisanje poslovnih pravila.

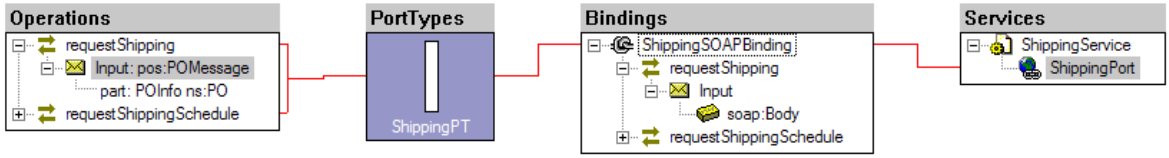
Kao i XSD rečnik, i WSDL interfejsi, odnosno njihova struktura potrebna za podršku izvršenju zadatog poslovnog procesa, definišu se na osnovu projekta semantičkog modela interoperabilnosti servisa poslovnog procesa, odnosno rezultata aktivnosti refaktorisanja tipova podataka.

Dijagram WSDL interfejsa *OrderProcessing.wsdl*, generisan uz pomoć ovog alata, prikazan je na slici 27.



Slika 27. Dijagram WSDL interfejsa OrderProcessing.wsdl

Dijagram WSDL interfejsa *Shipping.wsdl*, prikazan je na slici 28.



Slika 28. Dijagram WSDL interfejsa Shipping.wsdl

Orkestracija servisa

Izvršavanje svake pojedinačne funkcije rezultuje ostvarenjem ograničenog cilja pojedinačne poslovne aktivnosti, kroz izvršenje upita, transakcije, autentifikacije i sl. Ispunjenjem ograničenog cilja se ne postiže domet značaja jednog poslovnog procesa i uticaja njegovog izvršavanja na ukupne performanse preduzeća. Međutim, ovaj uticaj se može ostvariti izvršavanjem niza poslovnih aktivnosti u određenom redosledu, sa tačno definisanom strukturom u kontekstu jednog poslovnog procesa, pri čemu izvršenje svake poslovne aktivnosti podrazumeva pozivanje jednog ili više servisa.

Orkestracija servisa u servisno-orijentisanoj arhitekturi se vrši realizacijom BPEL strukture za izvršenje poslovnih procesa i njenom instalacijom u odgovarajući engine.

Aktivnosti na razvoju okruženja za orkestraciju servisa jednog poslovnog procesa su:

- Realizacija apstraktne definicije procesa podrazumeva definisanje njegovih osnovnih i strukturnih elemenata i njegovih promenljivih u okviru kojih se čuvaju informacije o trenutnom stanju procesa i prosleđuju narednim aktivnostima;
- Realizacija konverzacionih elemenata servisno orijentisane arhitekture obuhvata proširivanje WSDL interfejsa definicijom tipova partner linkova i definisanja konverzacionih elemenata same definicije procesa – partner linkova i, opciono, partnera;
- Realizacija mehanizama za upravljanje događajima, alarmima, greškama i kompenzacijom procesa.
- BPEL implementacija procesa

Realizacija apstraktne definicije procesa

Na osnovu definisanog generičkog opisa poslovnog procesa obrade narudžbina, odnosno njegovog UML dijagrama aktivnosti, definisane su osnovne aktivnosti procesa:

Kreiranje narudžbine (*Create Sales Order* i *Create Sales Order Items* aktivnosti). Zahvaljujući funkcijama generičkog WSDL klijenta, koji omogućava definisanje kompleksnih SOAP poruka, kreiranje narudžbine i unos stavki narudžbine se mogu izvršiti u okviru jedinstvene aktivnosti.

Korišćene promenljive: `POMessage` (tip očekivanog odziva)

Preuzimanje veličine prihvatljivog rizika za preduzeće, sa stanovišta kreditne sposobnosti klijenta se vrši sinhronim pozivom odgovarajuće metode WSDL interfejsa koji obezbeđuje pristup informacijama kojima se upravlja strateškim poslovanjem.

Korišćene promenljive: `AcceptableAccountCreditRiskRateMessage` (tip očekivanog odziva servisa), `RelevantDateMessage` (parametar poziva).

Prethodna podešavanja promenljivih: Pre poziva metoda, potrebno je pozivom *XPath* funkcije `current_date()`, postaviti aktuelni datum, na osnovu kojeg se zatražuje procena prihvatljivog rizika.

Preuzimanje kreditne sposobnosti klijenta predstavlja aktivnost koja se izvršava automatski, u maniru poziva udaljene metode (*Remote Procedure Call*) – sinhronom komunikacijom sa web servisom finansijskog sektora.

Korišćene promenljive: `AccountCreditRatingMessage` (tip očekivanog odziva servisa), `AccountIDMessage` (parametar poziva).

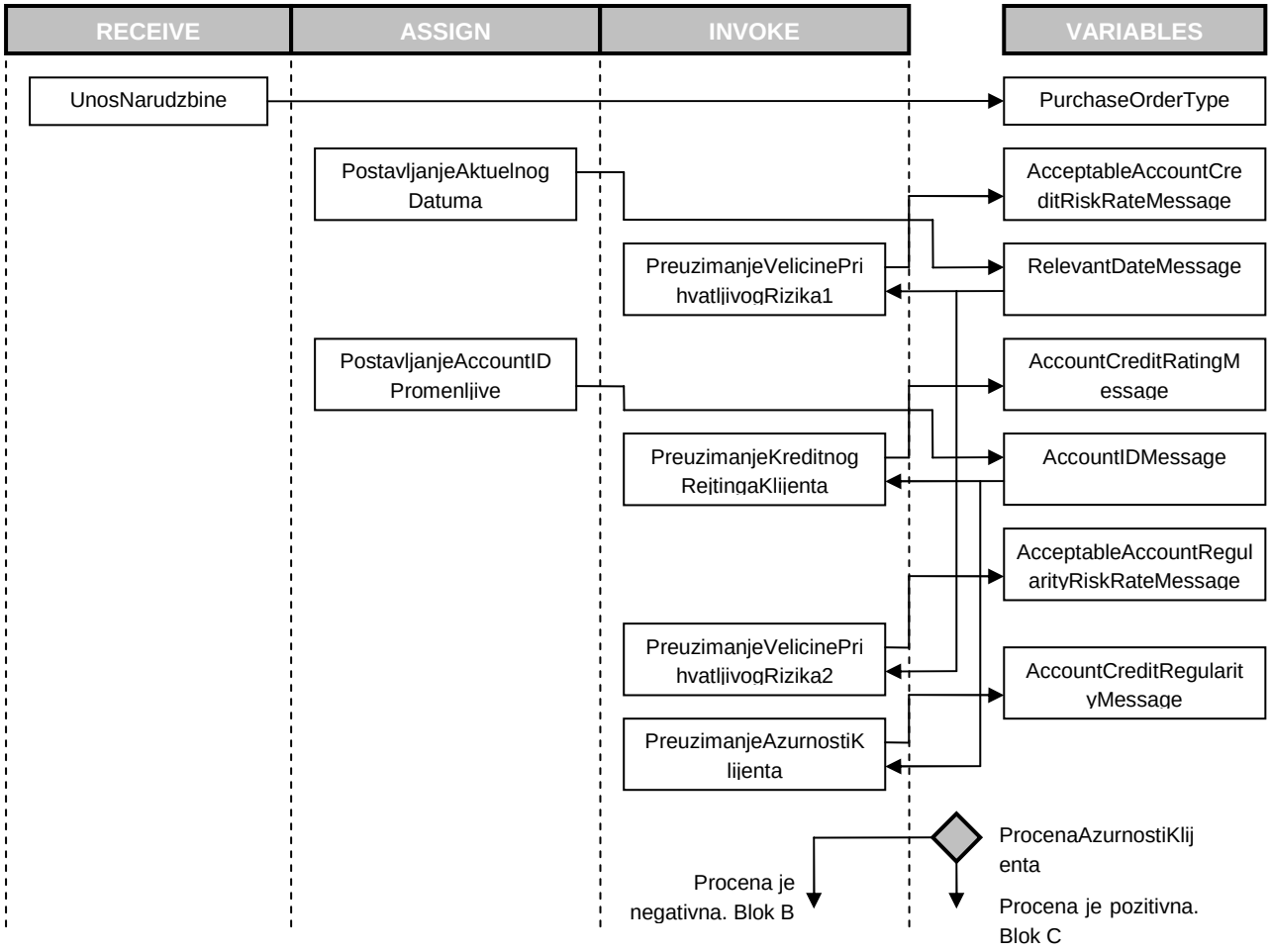
Prethodna podešavanja promenljivih: Pre poziva metoda, potrebno je korišćenjem *XPath* upita nad `POMessage` kompleksnom promenljivom, postaviti vrednost parametra poziva – `AccountIDMessage`.

Preuzimanje veličine prihvatljivog rizika za preduzeće, sa stanovišta ažurnosti klijenta se vrši sinhronim pozivom odgovarajuće metode WSDL interfejsa koji obezbeđuje pristup informacijama kojima se upravlja strateškim poslovanjem.

Korišćene promenljive: `AcceptableAccountRegularityRiskRateMessage` (tip očekivanog odziva servisa), `RelevantDateMessage` (parametar poziva).

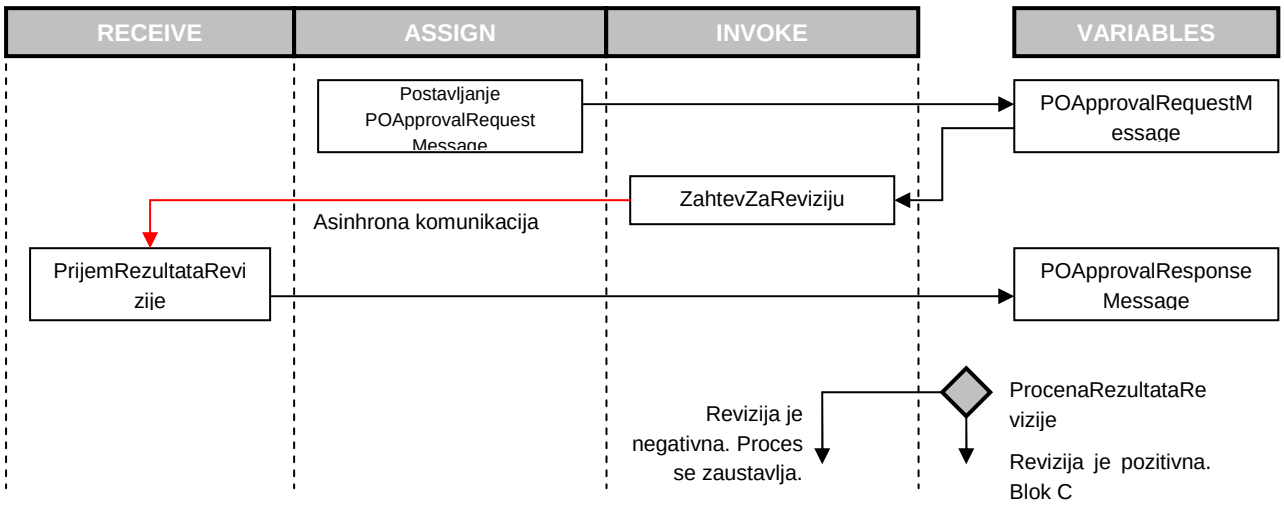
Preuzimanje ažurnosti klijenta predstavlja aktivnost koja se izvršava automatski, u maniru poziva udaljene metode (*Remote Procedure Call*) – sinhronom komunikacijom sa web servisom finansijskog sektora.

Korišćene promenljive: `AccountRegularityRatingMessage` (tip očekivanog odziva servisa), `AccountIDMessage` (parametar poziva).



Slika 29. Blok A apstraktne definicije procesa obrade narudžbina

Procena rizika. Na osnovu prostog upoređivanja preuzetih kreditne sposobnosti klijenta i ocene njegovih ažurnosti, sa preuzetim veličinama prihvatljivog rizika za preduzeće, sa stanovišta kreditne sposobnosti klijenta i njegove ažurnosti, vrši se procena. Pozitivna ocena implicira dalje odvijanje procesa. Ukoliko je ona negativna, prelazi se na aktivnost 7 (vidi blok B, Slika 30). Ukoliko je pozitivna, prelazi se na aktivnost 8 (vidi blok C, Slika 31).



Slika 30. Blok B apstraktne definicije procesa obrade narudžbina

Revizija negativne procene rizika usled neprihvatljivog kreditnog rejtinga klijenta ili neažurnosti. Ukoliko je automatska procena rizika daje negativan rezultat, neophodno je da se stekne lični uvid u razloge usled kojih je do nje došlo, a ona prihvati, čime bi se proces obrade narudžbine uslovno rečeno - zaustavio, ili odbije, na osnovu čega bi se stekli uslovi da se proces obrade aktuelne narudžbine nastavi. S obzirom na to da je u ovom koraku neophodna ljudska intervencija, ovaj blok procesa se vrši asinhrono, izvršavanjem sledećih BPEL aktivnosti.

Upućivanje zahteva za revizijom. Predstavlja aktivnost poziva udaljene metode, pa se koristi `invoke` deklaracija. S obzirom na to da upućivanje zahteva za reviziju predstavlja iniciranje asinhrone interakcije sa partnerskim servisom, od pozvate metode se ne očekuje odgovor.

Potrebne promenljive: `POApprovalRequestMessage` (parametar poziva)

Prethodna podešavanja promenljivih: Pre upućivanja zahteva, potrebno je konstruisati parametar poziva, tako što će se njegovim atributima dodeliti vrednosti veličina ažurnosti klijenta (`AccountRegularityRatingMessage`) i njegovog kreditnog rejtinga (`AccountCreditRatingMessage`), kao i sama narudžbina (`POMessage`).

Prijem odgovora sa rezultatom revizije. Na osnovu odgovora, primljenog od strane partnerskog web servisa, na osnovu rezultata receive aktivnosti, donosi se odluka o nastavku procesa. Ukoliko je revizija pozitivna, proces se nastavlja i prelazi se na aktivnost 8 (vidi blok C, Slika 76). Ukoliko nije, proces se zaustavlja.

Potrebne promenljive: `POApprovalResponseMessage` (tip očekivanog odziva od strane procesa - za partnerski servis, ovo je ulazna promenljiva)

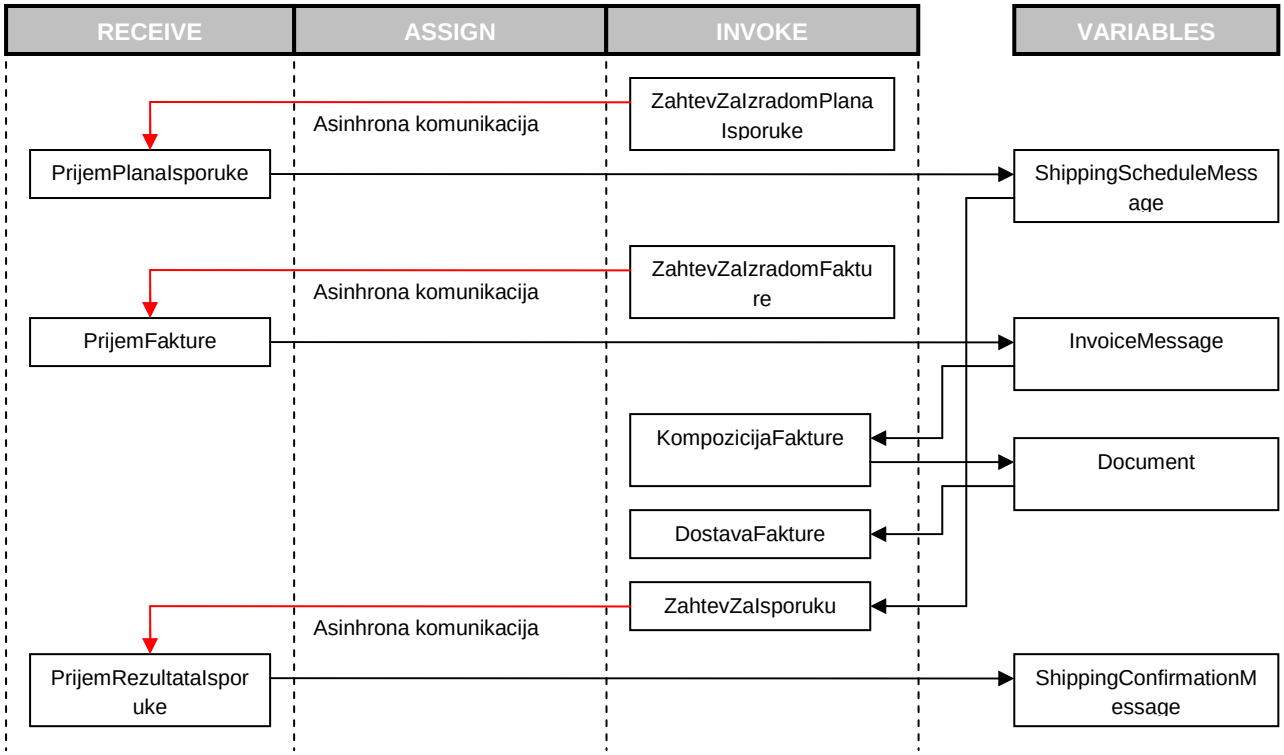
Definisanje plana isporuke i izrada fakture. Na osnovu dosadašnjih aktivnosti, utvrđena je specifikacija narudžbine i izvršena provera rizika za njenu realizaciju. U narednom delu procesa, potrebno je definisati plan isporuke i kreirati fakturu. Iako to nije predviđeno generičkim opisom procesa, a s obzirom na to da su aktivnosti definisanja plana isporuke i izrade fakture asinhrone, radi poboljšanja performansi procesa, ove dve aktivnosti će se vršiti uporedno.

ZahtevZaIzradomPlanaIsporuke. Predstavlja aktivnost poziva udaljene metode, pa se koristi `invoke` deklaracija. S obzirom na to da upućivanje zahteva za reviziju predstavlja iniciranje asinhrone interakcije sa partnerskim servisom, od pozvane metode se ne očekuje odgovor.

Potrebne promenljive: `POMessage` (parametar poziva)

PrijemPlanaIsporuke. Dostavljanje plana isporuke od strane partnerskog web servisa se vrši pozivom odgovarajuće *callback* metode, pri čemu `receive` aktivnost osluškuje njeno izvršavanje.

Potrebne promenljive: `ShippingScheduleMessage` (tip očekivanog odziva od strane procesa - za partnerski servis, ovo je ulazna promenljiva)



Slika 31. Blok C apstraktne definicije procesa obrade narudžbina

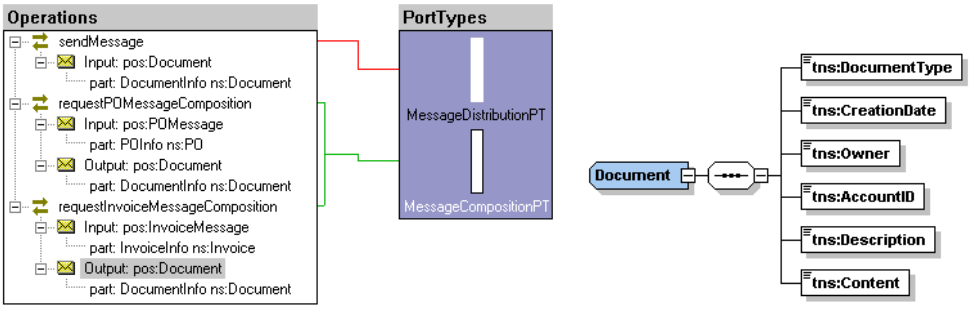
ZahtevZaIzradomFakture. Predstavlja aktivnost poziva udaljene metode, pa se koristi `invoke` deklaracija. S obzirom na to da upućivanje zahteva za reviziju predstavlja iniciranje asinhrone interakcije sa partnerskim servisom, od pozvane metode se ne očekuje odgovor.

Potrebne promenljive: `POMessage` (parametar poziva)

PrijemFakture. Dostavljanje fakture od strane partnerskog web servisa se vrši pozivom odgovarajuće *callback* metode, pri čemu `receive` aktivnost osluškuje njeno izvršavanje.

Potrebne promenljive: `InvoiceMessage` (tip očekivanog odziva od strane procesa - za partnerski servis, ovo je ulazna promenljiva)

Slanje fakture klijentu. Na osnovu poznatih podataka, odnosno, pre svega, generisane fakture, moguće je i potrebno obavestiti klijenta o stanju procesa obrade narudžbine i poslati mu fakturu. Slanje fakture se vrši na osnovu ustanovljenih interfejsa za eksternu komunikaciju dokumentima u okviru poslovnog informacionog sistema. Njegova uprošćena arhitektura je prikazana na slici 32. i realizovana WSDL interfejsom *MessageBrokerage.wsdl*.



Slika 32. Uprošćena struktura interfejsa za razmenu dokumenata i struktura XML tipa Document

Distribucija dokumenata u okviru pretpostavljenog poslovnog informacionog sistema se vrši u dva koraka. Najpre se izvršava kompozicija dokumenta, na osnovu njegovog tipa (račun,

narudžbenica, itd.), a potom i njegovo slanje na definisano odredište. Odredište dokumenta predstavlja deo njegove definicije. U ovom slučaju, ono se referencira na odredišnu adresu i način dostave posredstvom atributa `AccountID`, objekta `Document`.

Očigledno je da se faktura klijentu, nakon njenog kreiranja, dostavlja izvršenjem dve sinhrone aktivnosti, koje vrše poziv `requestInvoiceMessageComposition` i `sendMessage` metoda

KompozicijaFakture. Predstavlja aktivnost poziva udaljene metode, pa se koristi `invoke` deklaracija.

Potrebne promenljive: `InvoiceMessage` (parametar poziva), `Document` (tip očekivanog odziva)

DostavaFakture. Predstavlja aktivnost poziva udaljene metode, pa se koristi `invoke` deklaracija.

Potrebne promenljive: `Document` (parametar poziva)

Isporuka. Poslednje aktivnost procesa obrade narudžbine je isporuka proizvoda obuhvaćenih narudžbinom. Sa stanovišta modeliranja procesa, isporuka predstavlja podproces – složenu celinu sa svojim okvirom promenljivih, aktivnosti i servisa. U ovom primeru, ovaj podproces nije razrađen, već se sagledava kao jedna – asinhrona aktivnost. S obzirom na to, ona se sastoji od BPEL aktivnosti poziva servisa kojim se inicira podproces isporuke i očekivanja uspešnog odziva odgovarajuće *callback* metode istog servisa.

ZahtevZaIsporuku. Predstavlja aktivnost poziva udaljene metode, pa se koristi `invoke` deklaracija. S obzirom na to da upućivanje zahteva za reviziju predstavlja iniciranje asinhrone interakcije sa partnerskim servisom, od pozvane metode se ne očekuje odgovor.

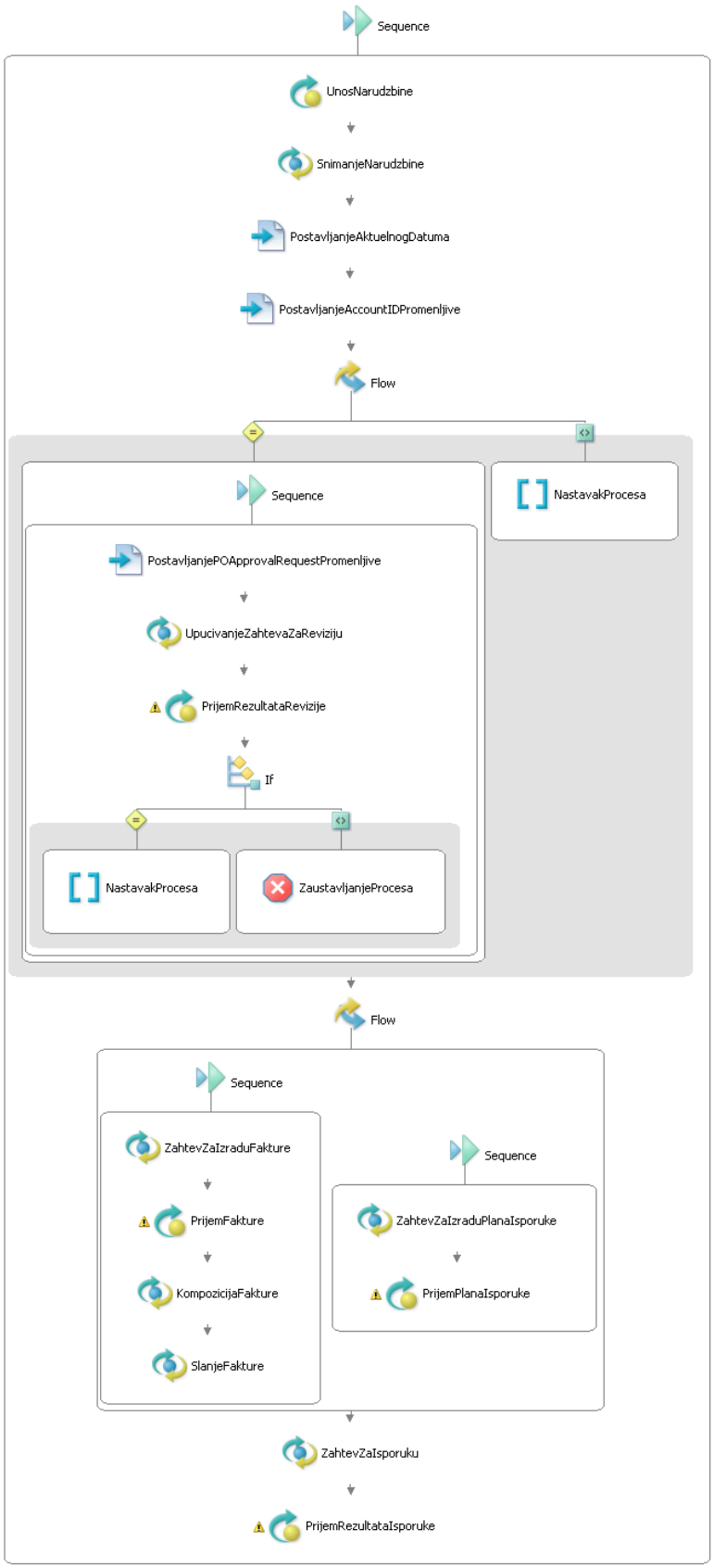
Potrebne promenljive: `ShippingScheduleMessage` (parametar poziva)

PrijemRezultataIsporuke. Dostavljanje potvrde o isporuci od strane partnerskog web servisa se vrši pozivom odgovarajuće *callback* metode, pri čemu `receive` aktivnost osluškuje njeno izvršavanje.

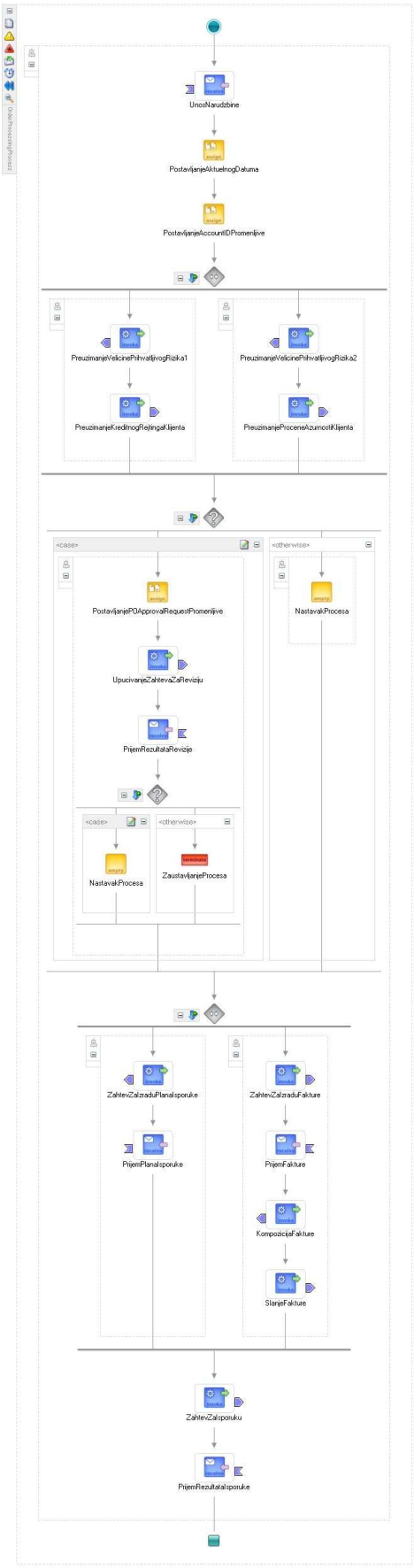
Potrebne promenljive: `ShippingConfirmationMessage` (tip očekivanog odziva od strane procesa - za partnerski servis, ovo je ulazna promenljiva)

Izvršavanjem poslednje aktivnosti, instanca procesa obrade narudžbine se završava. Na osnovu prikazane, apstraktne definicije procesa, moguće je izvršiti implementaciju BPEL procesa. Implementacija apstraktne definicije procesa podrazumeva definisanje njegovih osnovnih i strukturnih BPEL elemenata i odgovarajućih, predviđenih promenljivih u okviru kojih se čuvaju informacije o trenutnom stanju procesa i prosleđuju narednim aktivnostima.

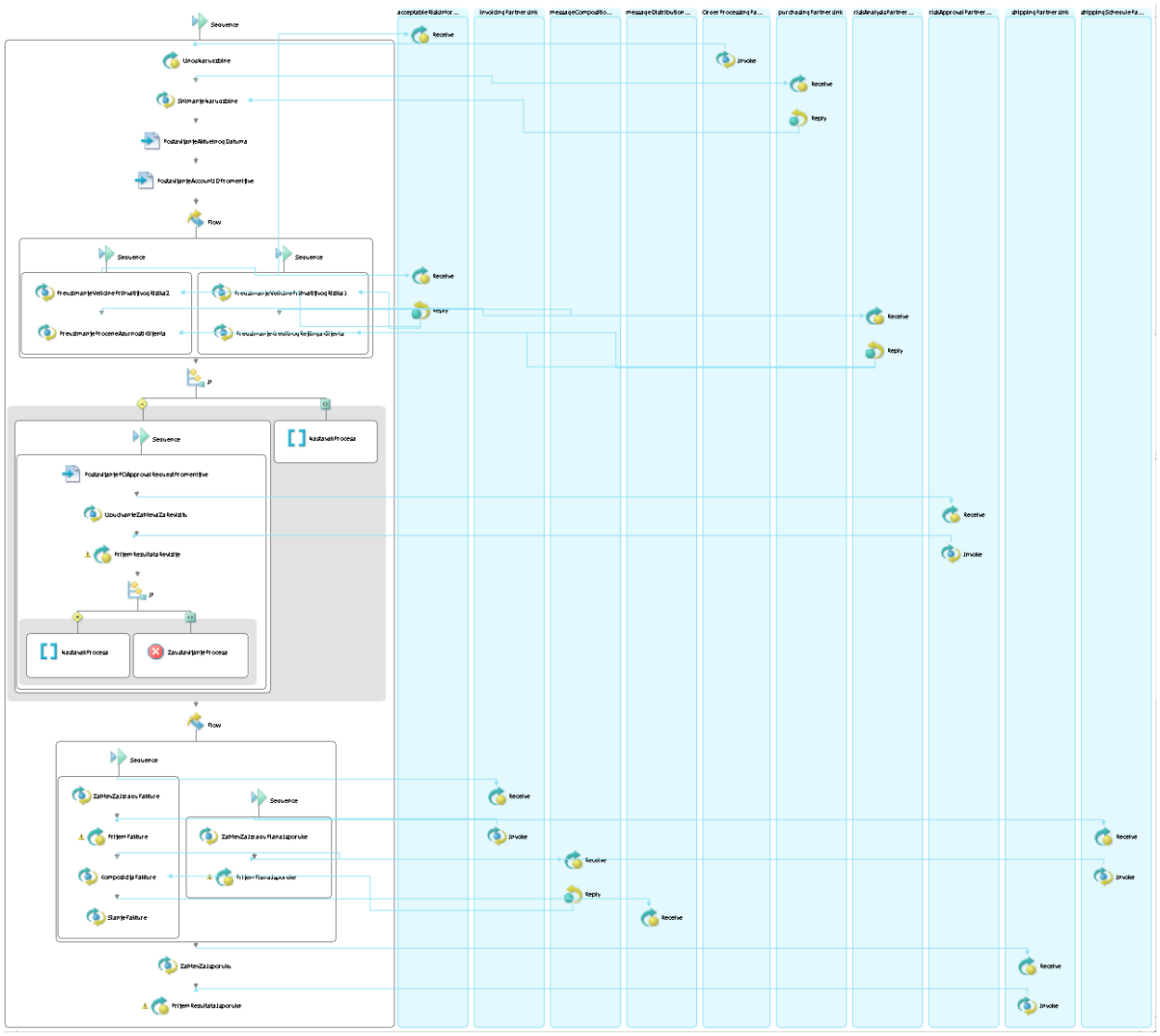
BPEL implementacija prikazane apstraktne definicije procesa je izvršena primenom alata ActiveBPEL Designer i Oracle JDeveloper BPEL Designer 10g. Šema BPEL implementacije, urađena u ActiveBPEL Designer alatu je prikazana na slici 33. Šema BPEL implementacije urađena u Oracle JDeveloper alatu je predstavljena na slici 34.



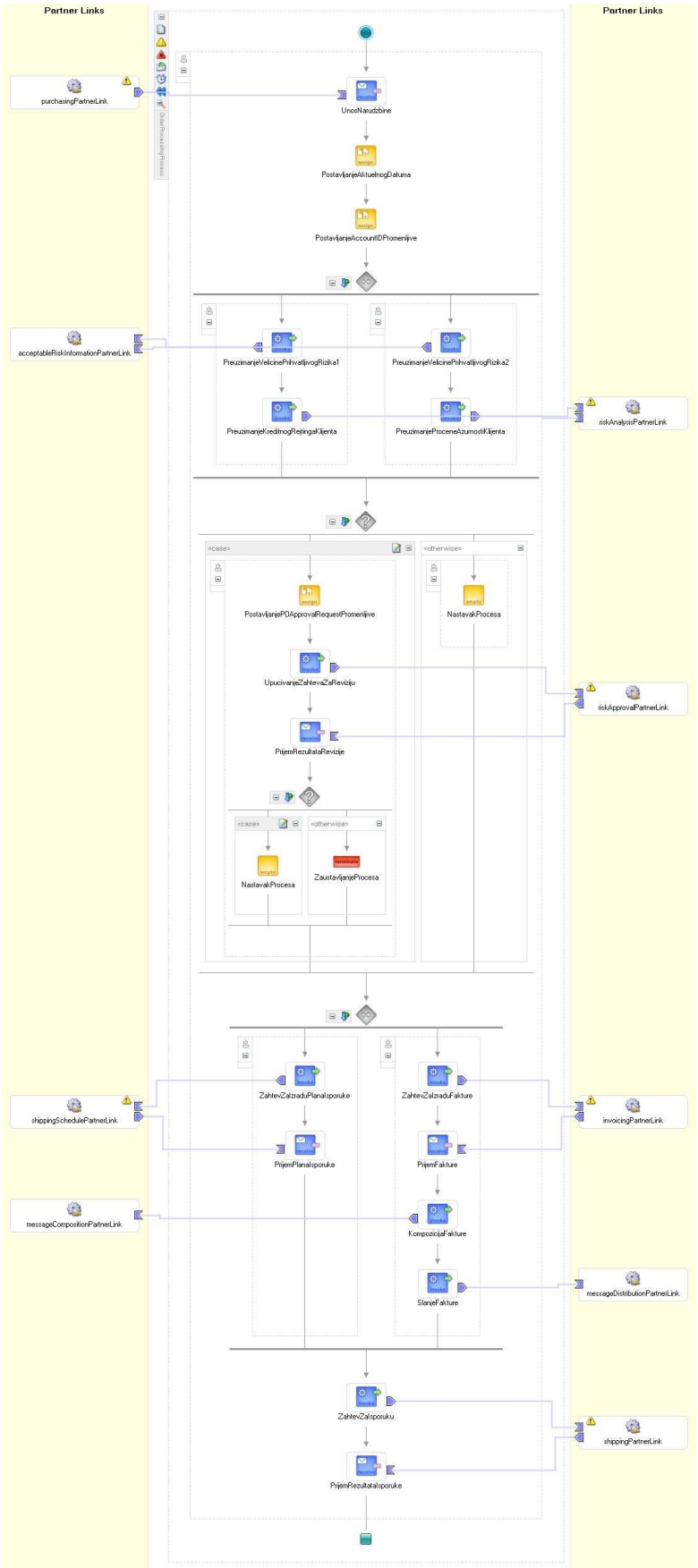
Slika 33. BPEL struktura procesa obrade narudžbina (ActiveBPEL Designer)



Slika 34. BPEL struktura procesa obrade narudžbina (Oracle JDeveloper BPEL Designer)



Slika 35. Strukturni i konverzacioni elementi BPEL definicije procesa obrade narudžbina (ActiveBPEL Designer)



Slika 36. Strukturni i konverzacioni elementi BPEL definicije procesa obrade narudžbina (Oracle JDeveloper BPEL Designer)