VISOKA POSLOVNA ŠKOLA
STRUKOVNIH STUDIJA
ČAČAK

MAGISTARSKA TEZA

Modeliranje poslovnih procesa u razvoju aplikacija elektronskog poslovanja

Mentor:_____ Student: _____
Profesor: _____ Br.Indeksa: _____

Mesto, mesec, godina

Sadržaj:

1. Uvod

Razvoj aplikacija elektronskog poslovanja uvek zahteva angažovanje značajnih resursa organizacije i dosta vremena. Jedan od efikasnijih pristupa je da proces razvoja započne identifikacijom poslovnih resursa, a zatim da se nastavi analizom poslovnih procesa, od značaja za sve faze životnog ciklusa svakog od poslovnih resursa.

Poslednjih godina razvijene su nove generacije jezika i standarda za opis poslovnih procesa, kao i novi alati za njihovu implementaciju. Time je navedeni koncept dobio značajnu podršku u pogledu efikasnosti i preciznosti razvoja informacionog sistema u celini.

Polazeći od činjenice da poslovni proces u elektronskom poslovanju predstavlja skup aktivnosti, koje reprezentuju realnu poslovnu operaciju, svaki poslovni proces može se modelovati kao skup pojedinačnih poslovnih zadataka. U fazi implementacije, ove zadatke moguće je realizovati kreiranjem, ili korišćenjem postojećih Web servisa, u okviru, ili van organizacije.

Postojeći standardi vezani za Web servise, kao što su SOAP, WSDL i UDDI, nisu dovoljni da daju potpunu podršku izgradnji poslovnih procesa. Kao podrška modeliranju poslovnih procesa u elektronskom poslovanju, kreirano je nekoliko specifikacija standarda. Dobro rešenje za modeliranje poslovnih procesa postiže se izborom najboljih standarda. BPMN i BPEL su jezici, koji se koriste za opis izvršavanja poslovnog procesa u okviru sistema. U praksi se pokazalo da se njihovim prihvatanjem i korišćenjem kreiraju dobra rešenja.

Predmet istraživanja u okviru ove magistarske teze je analiza postojećih koncepata, dizajna, arhitekture i specifikacija standarda za modeliranje poslovnih procesa u B2B aplikacijama elektronskog poslovanja, sa posebnim naglaskom na analizu mogućnosti automatizacije pri opisivanju izvršavanja poslovnih procesa.

U okviru ovog rada prvo su definisani pojmovi poslovnog procesa, metodologije za modeliranje poslovnih procesa, modeliranje poslovnih procesa (BPM), arhitekture zasnovane na servisima (SOA), kao i međusoban odnos između BPM i SOA.

Detaljno su opisani svi koncepti i standardi vezani za razvoj Web servisa. Predstavljena je BPMN notacija, koja je razvijena je sa ciljem da obezbedi način za definisanje poslovnih procesa, lako razumljivu od strane svih poslovnih korisnika. Takođe je opisan BPEL jezik za izvršavanje poslovnih procesa.

Za opis ponašanja poslovnih procesa mogu se koristiti uzori za procese, koji se nazivaju P4 uzori, a osmišljeni su sa ciljem da pomognu dizajnerima procesa da odrede različite načine za povezivanje aktivnosti u okviru procesa.

Cilj rada na ovoj magistarskoj tezi je bio da se pokaže da prelazak iz faze u fazu životnog ciklusa modelovanja, nije uvek, ili do kraja, podržan standardima i tehnologijama, tako da je posebna pažnja posvećena mapiranju BPMN dijagrama u BPEL izvršni jezik i dat je osvrt na nekonzistentnost koja postoji u procesu mapiranja.

U toku rada na ovoj magistarskoj tezi, korišćena su dostupna saznanja, standardi, alati i metode vezane za modeliranje poslovnih procesa u aplikacijama elektronskog poslovanja.

U okviru rada dat je i konkretan primer primene modelovanja poslovnih procesa u razvoju aplikacija turističkog poslovanja.

2. Modeliranje poslovnih procesa u elektronskom poslovanju

2.1 Šta je poslovni proces?

Poslovni proces predstavlja posao, podeljen u više koraka ili aktivnosti, koji su neophodni da bi se obavila poslovna transakcija. Da bi se izvršile aktivnosti u okviru poslovnog procesa, može biti neophodna akcija od strane aplikacije ili čoveka. Tipično za poslovne procese je da su po svojoj prirodi dugotrajni, kao i da uključuju više strana i/ili aplikacija u okviru ili van organizacije.

Poslovni proces može biti veoma složen po svojoj strukturi. Može imati više učesnika, kao što su ljudi, organizacije i sistemi, koji izvršavaju više zadataka. Da bi ostvarili postavljeni zadatak, učesnici moraju koordinisano izvršavati definisane zadatke, najčešće grupisane u podprocese. U nekim situacijama, podprocesi se izvršavaju paralelno, u drugim su sekvencijalni. Neki procesi zahtevaju ponovno izvršavanje podprocesa. Većina procesa sadrži tačke odluke, koje dovode do grananja toka u zavisnosti od postavljenih, zadovoljenih ili ne zadovoljenih uslova. U okviru nekih procesa, učesnici moraju proslediti određene informacije. Prenos informacija može imati ulogu okidača, odnosno otpočinjanja novog zadatka. Neki procesi su ad-hoc procesi, što znači da njihovi podprocesi nemaju definisane okidače. Učesnici ne moraju da završe sve definisane zadatke pre nego što oni, ili neki drugi učesnik, započnu izvršavanje drugog zavisnog podprocesa.

2.2 Metodologije modeliranja poslovnih procesa

Metodologija je u Merriam–Webster rečniku definisana kao:

1. analiza principa metoda, pravila i postulata koje koriste discipline,
2. razvoj metoda, koje se primenjuju preko disciplina,
3. određena procedura, ili skup procedura.

Metodologija uključuje sledeće koncepte, kao što se odnosi na određene discipline ili polja istraživanja:

1. kolekcija teorija, koncepata, ili ideja;
2. komparativna studija različitih pristupa;
3. kritika pojedinačnih metoda.

Metodologija predstavlja više od prostog skupa metoda, jer se odnosi više na obrazloženja i filozofske pretpostavke, nego na određenu studiju.

Model predstavlja uzor, plan, predstavljanje, ili opis, dizajniran da prikaže strukturu objekta, sistema, ili koncepta.

Proces može izgledati drugačije kada se opisuje od strane različitih učesnika. Potrebno je da metodologija modelovanja poslovnih procesa bude sposobna da reprezentuje različite aspekte opisa procesa. Dobra metodologija treba da omogući predstavljanje procesa na način jednostavan za prebacivanje u oblik prepoznatljiv učesniku.

Modeliranje poslovnih procesa uobičajeno ima širi opseg u odnosu na modeliranje pojedinačnog softverskog sistema. Svrha modelovanja je da analitičar sistema može jasno da predstavi opseg sistema koji se modeluje i koji će biti na odabrani način implementiran.

Stefan Haberl je razvio skup od sedam kriterijuma za evaluaciju metodologija modelovanja poslovnih procesa [Haberl]:

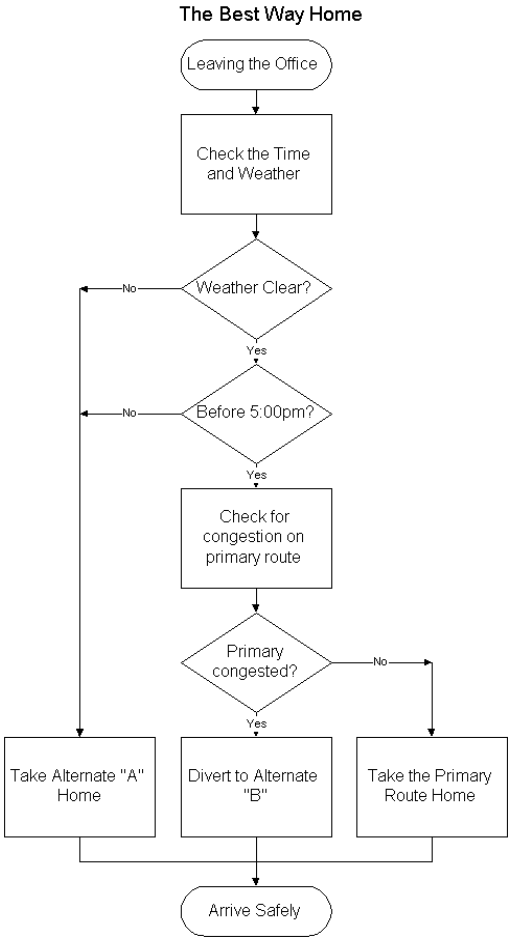
1. sposobnost da modeluje sve složene situacije vezane za poslovni proces:
    - sekvence
    - grananja
    - petlje
    - konkurentne elemente - paralelne grane i sinhronizaciju
    - kontrolu vremena izvršavanja
    - krajnji rok za izvršavanje
    - kontrolu grešaka
    - agregaciju
2. mogućnost za razlikovanje uloga, kao i mogućnost da se ulogama dodeljuju različiti zadaci
3. postojanje nedvosmislene grafičke prezentacije jezika
4. podržavanje transakcionog modela, koji omogućava opis situacije kada se proces mora vratiti u prethodno dobro stanje
5. definisanje kako će instance procesa biti pokrenute i identifikovane tokom izvršavanja
6. mogućnost za specifikaciju karakteristika poslovnog procesa, koji su važni za eksterne korisnike (što uključuje kvalitet usluga i cene)
7. jezik ne treba da sadrži detalje vezane za komunikacioni protokol.

Već dugi niz godina ljudi modeluju procese. Tehnike koju su koristili opisuju kako organizacije izvršavaju svoje poslove. Modeli su korišćeni u različite svrhe: za obuku novih zaposlenih, kao pomoć u procesu reinženjeringa, za kreiranje simulacionih modela za testiranje procesa, za razvoj sistema koji će automatizovati modelovani proces, kao vodič programerima za razvoj informacionih sistema, ili za direktno izvršavanje pomoću procesne mašine [Kerschberg].

Tokom godina razvijene su brojne metodologije modelovanja poslovnih procesa. Neke od njih, korišćene poslednjih trideset godina, su [Kerschberg]:

- Dijagrami toka (Flow Charts)

  Dijagrame toka koriste programeri da bi opisali logiku, ili putanju izvršavanja programa. Simboli i semantika dijagrama toka je ograničena na određeni broj kontrolnih struktura namenjenih programerima. Dijagrami toka su razvijeni da bi opisali put izvršavanja u okviru jednog procesa. Nemaju snagu da na pravi način modeluju grupu procesa koji među sobom sarađuju.



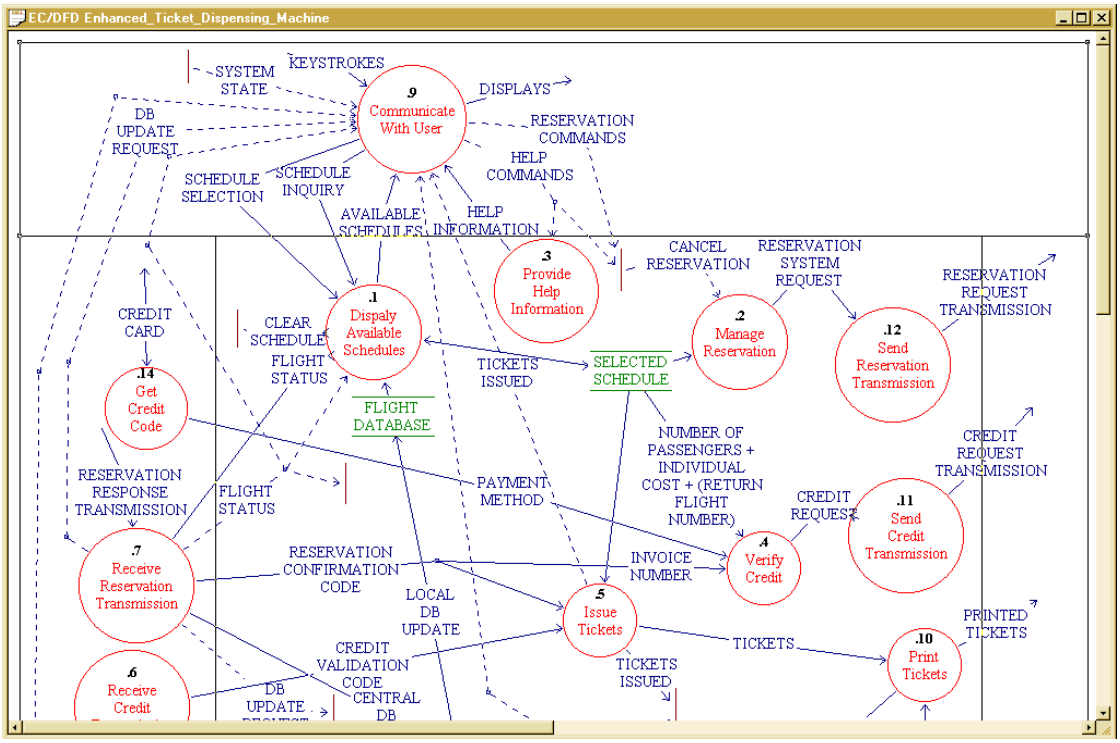
slika 1 – ANSI Flow Chart primer
[deming.eng.clemson.edu/pub/tutorials/qctools/flowm.htm]

- Dijagrami toka podataka (Data Flow Diagrams)

  Ovaj tip dijagrama se bavi podacima u okviru informacionog sistema. Počeo je da se koristi 1978. godine, posle izlaska knjige autora Tom DeMarco "Strukturna sistem analiza i sistemske specifikacije". Dijagrami toka podataka prikazuju niz koraka obrade podataka. Svaki korak dokumentuje jednu akciju, koja transformiše ili distribuira podatke. Nemaju mogućnost da opišu sekvence procesa ili mehanizme za kontrolu procesa.

- Dijagrami toka kontrole (Control Flow Diagrams)

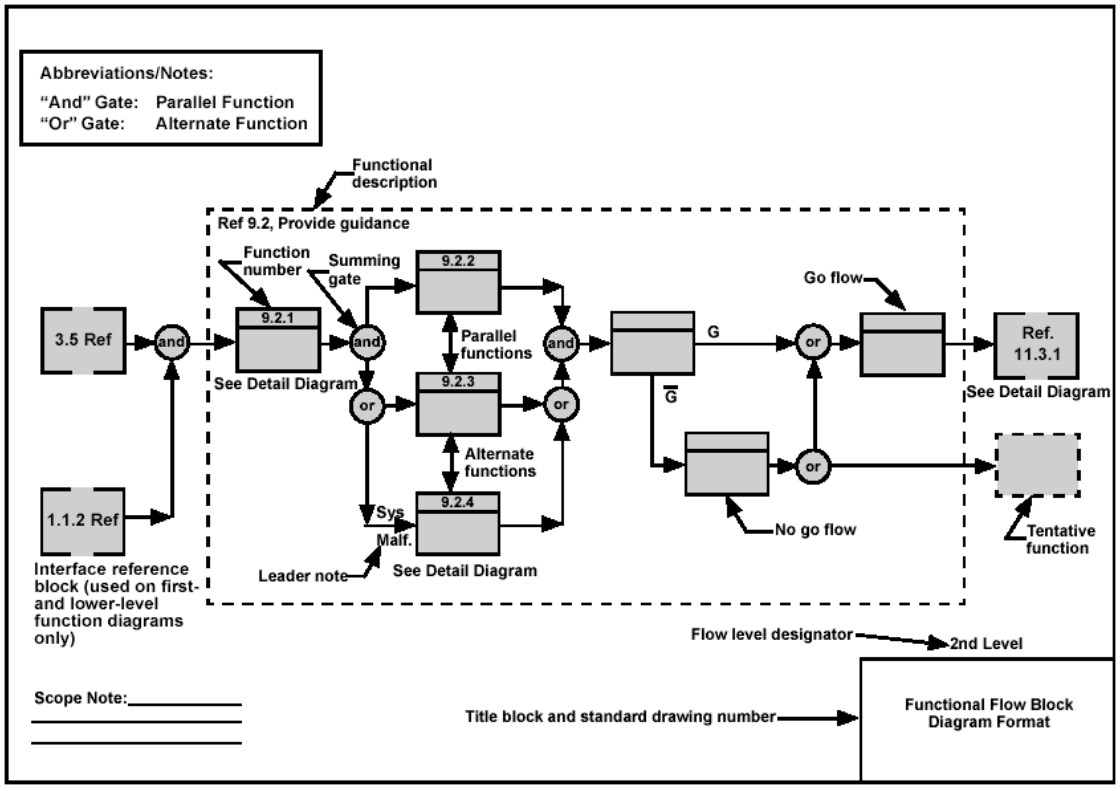
  Ova vrsta dijagrama je nastala kao proširenje dijagrama toka podataka. Koristi se da opiše aplikacije zasnovane na događajima, više nego za aplikacije zasnovane na podacima.



slika 2: Kombinovani Data/Control Flow Diagrami
[http://www.turbocase.com/features.html]

- Funkcionalni blok dijagrami toka (Functional Flow Block Diagrams)

  Ovaj tip dijagrama se uobičajeno koristi u klasičnim inženjerskim sistemima da prikaže redosled izvršavanja sistemskih funkcija. Dijagrami prikazuju pravilan redosled u izvršavanju funkcija, kao i konkurentnost prilikom izvršavanja. Konkurentne funkcije mogu se izvršavati paralelno (AND) ili alternativno (OR).



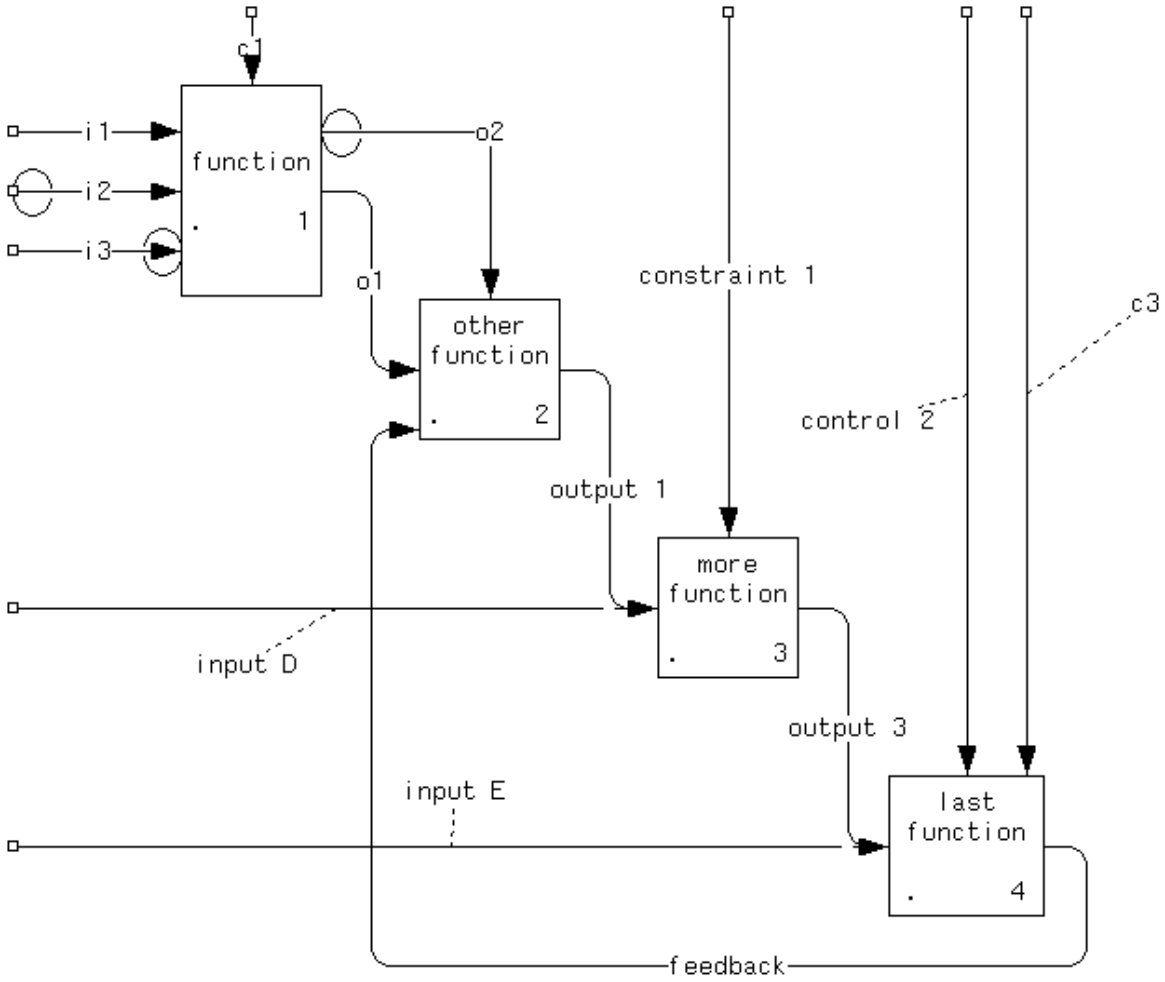
slika 3: Funkcionalni blok dijagram toka primer [Kerschberg]

- Gantt/PERT dijagrami

  I Gantt dijagrami (napravljeni 1917. godine kao metod koji omogućava vremensko planiranje poslova vezanih za konstrukciju brodova) i PERT dijagrami (napravljeni 1950. godine za američku mornaricu sa ciljem da omoguće prikaz složenih sekvenci u izvršavanju poslova) se koriste za upravljanje projektima. Veoma retko se koriste da modeliranje procesa, jer ne poseduju mogućnost opisa bilo koje druge informacije o procesu, izuzev redosleda izvršavanja zadataka. Ne podržavaju modeliranje ulaznih i izlaznih informacija, kao ni mehanizme za kontrolu procesa.

- IDEF

  IDEF je verovatno najkorišćenija tehnika za tradicionalno modeliranje poslovnih procesa. Postoji nekoliko tipova IDEF modela. Poznat je IDEF0 model aktivnosti, koji omogućava modeliranje zadataka koje izvršava jedna organizacija, i koji uključuju ulaze, izlaze, i kontrolu svakog zadatka. Zadaci, ili aktivnosti, mogu biti prikazani kao zadaci na visokom nivou, i mogu se razložiti u podaktivnosti.



slika 3: Primer IDEF0 diagrama
(http://www.threesl.com/pages/reference/diagrams/Idef0-diagram.php)

Poslednjih nekoliko godina razvijene su, ili su u razvoju, niz metoda, koje treba da omoguće modeliranje savremenih poslovnih procesa. Neke od tih metodologija su ebPML, BPMN, BPML, XLANG, WSFL, BPEL4WS, EDOC, XPDL, UML 2.0. U radu će biti detaljno opisane BPMN i BPEL metode za modeliranje poslovnih procesa.

2.3 Modeliranje poslovnih procesa

2.3.1 Business Process Modeling

Pojam Business Process Modeling (BPM), ili kako se jos naziva Business Process Management, odnosi se na dizajniranje, upravljanje i izvršavanje poslovnog procesa, a njegova snaga se nalazi u objedinjavanju i proširenju postojećih procesno orijentisanih tehnika i tehnologija.

Za poslovne analitičare, BPM znači sagledavanje organizacije kao skupa procesa koji se mogu definisati, kojim se može upravljati, i koji se mogu optimizovati. Umesto tradicionalne orijentacije, prema kojoj se poslovi dele po organizacionim celinama, BPM se orijentiše prema poslovima, bez obzira u kojoj se organizacionoj jedinici izvršavaju. Za tehničko osoblje, BPM predstavlja grupu tehnologija fokusiranih na definisanje, izvršavanje i nadgledanje procesne logike. Bez obzira na različite perspektive, obe grupe, i poslovni analitičari, i tehničko osoblje, imaju cilj da unaprede poslovne procese.

BPM treba koristiti samo za aplikacije koje su procesno orijentisane, odnosno koje su [Havey]:

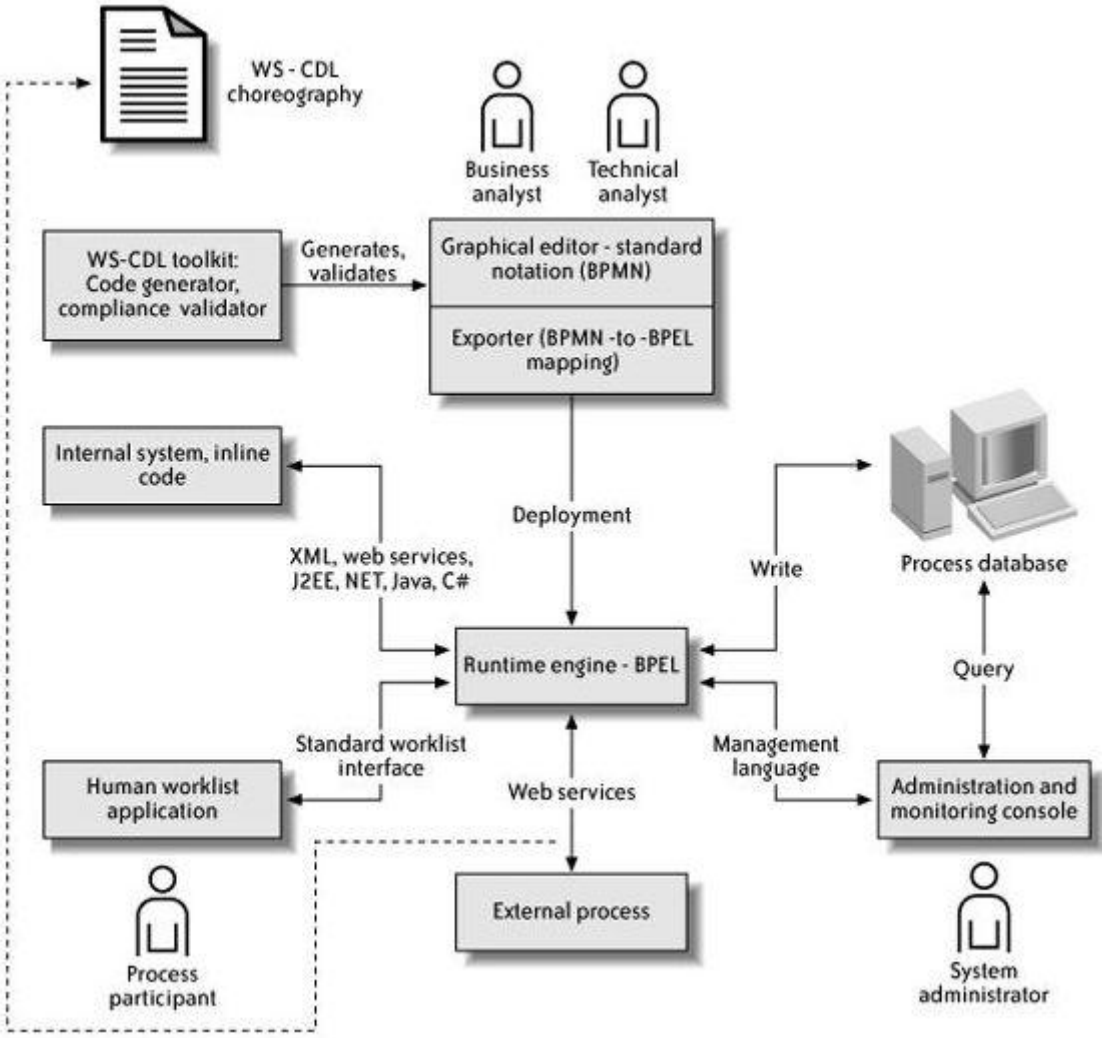
- dugotrajne, jer im je potrebno i više sati ili dana ili nedelja ili čak meseci da se izvrše,
- čuvaju stanje u bazama podataka,
- veći deo vremena čekaju na akciju koja će pokrenuti narednu aktivnost,
- čiji procesi su odgovorni za upravljanje i koordinaciju komunikacija između različitih uloga sistema i ljudi.

Procesne aplikacije treba da zadovolje barem deo navedenih karakteristika.

Razlozi za korišćenje BPM su sledeći:

- formalizacija postojećeg procesa,
- bolje razumevanje i na osnovu toga ugrađivanje poboljšanja poslovnog procesa,
- efikasnije izvršavanje poslovnih procesa zbog korišćenja BPM softvera,
- povećanje produktivnosti i smanjenje učešća ljudi u izvršavanju procesa,
- omogućavanje ljudima da reše komplikovane probleme,
- pojednostavljenje praćenje propisa.

2.3.2 BPM arhitektura



slika 4: BPM arhitektura [Havey]

Najbitniji deo, ili srce sistema, predstavlja procesna mašina, koja izvršava poslovni proces napisan u BPEL jeziku. Poslovni i tehnički analitičari dizajniraju procese koristeći grafički editor, koji podržava BPMN notaciju. Uobičajeno, editor omogućava kreiranje BPEL koda na osnovu BPMN dijagrama.

Na izvršavanje procesa može uticati i komunikacija sa ljudima i sa aplikacijama. Ljudi, koji učestvuju u procesu, koriste aplikacije dijagrama toka, da bi se povezali sa procesnom mašinom, i na taj način pregledali i izvršavali aktivnosti, koji čekaju da budu izvršene. Postoje dva tipa interakcije sa aplikacijama: interna i eksterna. Internim aplikacijama, koje se nalaze u okviru kompanijske mreže, ali van adresnog prostora od procesne mašine, se pristupa pomoću neke od integracionih tehnologija, kao što su Web servisi, J2EE, COM, uobičajeno koristeći XML kao format poruka. Eksterne aplikacije su uobičajeno Web servisi, sastavni delovi poslovnih procesa drugih kompanija.

BPM sistemski administratori koriste grafičke konzole da bi administrirali i nadgledali procese koji se izvršavaju. Konzole sa procesnim mašinama komuniciraju preko jezika za upravljanje. Procesne mašine čuvanje stanje procesa u bazi podataka. Da bi izvršili neki upit, sistemski administratori mogu da pristupaju bazi procesa.

2.3.3 Service Oriented Architecture

Service Oriented Architecture (SOA) opisuje koncepte, arhitekturu, i procesni okvir, da bi obezbedila jeftin razvoj, integraciju i održavanje informacionih sistema kroz smanjenje kompleksnosti i stimulaciju integracije i njihovog ponovnog korišćenja.

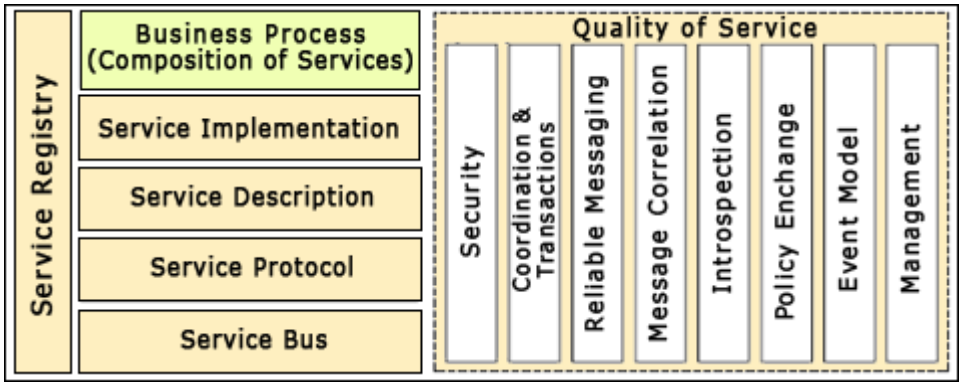
SOA ne predstavlja radikalno novu arhitekturu, već više evoluciju poznatih distribuiranih arhitektura i integracionih metoda. SOA poboljšava i proširuje fleksibilnost ranijih integracionih metoda (EAI) i distribuiranih arhitektura, i fokusira se na ponovno korišćenje postojećih aplikacija i sistema, efikasnu interoperabilnost i integraciju aplikacija, kao i kompoziciju poslovnih procesa preko servisa obezbeđenih aplikacijama. Važna osobina SOA je sposobnost da primeni promene, koje će nastati u budućnosti, na relativno jednostavan i lak način.

SOA predstavlja više od skupa tehnologija. SOA nije direktno povezana sa bilo kojom tehnologijom, mada se često implementira korišćenjem Web servisa. Web servisi predstavljaju najpodesniju tehnologiju za realizaciju SOA.

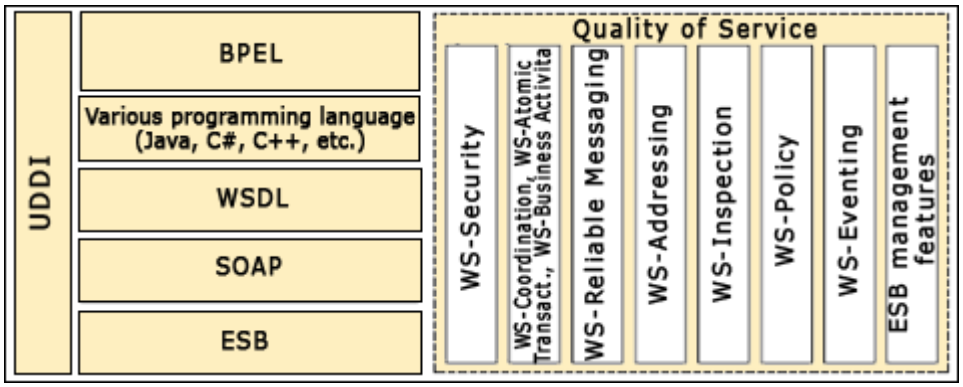
Najvažniji SOA koncepti su [Juric]:

- servisi (Servisi obezbeđuju poslovnu funkcionalnost, kao što je na primer aplikacija za prodaju. Servisi u SOA obezbeđuju poslovnu vrednost, sakrivaju implementacione detalje i autonomni su.);
- interfejsi (Korisnici servisu pristupaju preko interfejsa. Interfejs servisa definiše skup javnih potpisa operacija. Interfejs predstavlja ugovor između servis provajdera i servis korisnika. Interfejs je odvojen od njegove implementacije, samoopisujući je i nezavistan od platforme. Opis interfejsa obezbeđuje osnovu za implementaciju servisa od strane servis provajdera, kao i osnovu za implementaciju servisa od strane servis korisnika. Svaki interfejs definiše skup operacija.);
- poruke (Operacije su definisane kao skup poruka. Poruke specificiraju podatke koji će se razmenjivati, i opisuju ih na način nezavistan od platforme i jezika korišćenjem šema. Servisi razmenjuju samo podatke, što se razlikuje od objektno-orijentisanog pristupa, gde se ponašanje, odnosno implementacioni kod takođe može razmeniti. WSDL predstavlja jezik koji se koristi za opis poruka.);
- sinhrona i asinhrona komunikacija (Korisnici servisa mogu koristiti sinhroni i asinhroni komunikacioni mod da bi pozivali operacije servisa. U sinhronom modu, operacija servisa vraća odgovor servis korisniku po završetku obrade. Servis korisnik mora da čeka sa se operacija završi. U asinhronom modu, operacija servisa ne vraća odgovor servis korisniku, mada može obezbediti informaciju da je operacija izvršena uspešno.);
- slabo povezivanje (Slabo povezani servisi su servisi koje prikazuju samo neophodne zavisnosti i smanjuju sve vrste veštačkih zavisnosti. Minimalna zavisnost osigurava da će u slučaju promene jednog servisa biti potrebne minimalne promene kod drugih povezanih servisa. Posledica je unapređenje robustnosti, što čini sisteme elastičnim u odnosu na promene i preporučuje ih za ponovno korišćenje.);
- registri (Servis provajderi publikuju servise u registre, dok servis korisnici servisa pretražuju registre da bi pronašli potreban servis. Primer servis registra predstavlja UDDI.);
- kvalitet servisa (Atributi kao što su sigurnost, pouzdano razmenjivanje poruka, transakcije, korelacija, adresiranje obezbeđuju kvalitet servisa, što kod Web servisa obezbeđuju WS-* specifikacije.);
- kompozicija servisa u poslovni proces (Verovatno najvažniji SOA koncept. Kompozicija servisa dozvoljava nam da obezbedimo podršku za poslovne procese

na fleksibilan i relativno jednostavan način. Takođe omogućava brze promene poslovnih procesa i sa manje napora. BPEL predstavlja jezik za kompoziciju poslovnih procesa.).



slika 5: Grafički prikaz najvažnijih SOA koncepata



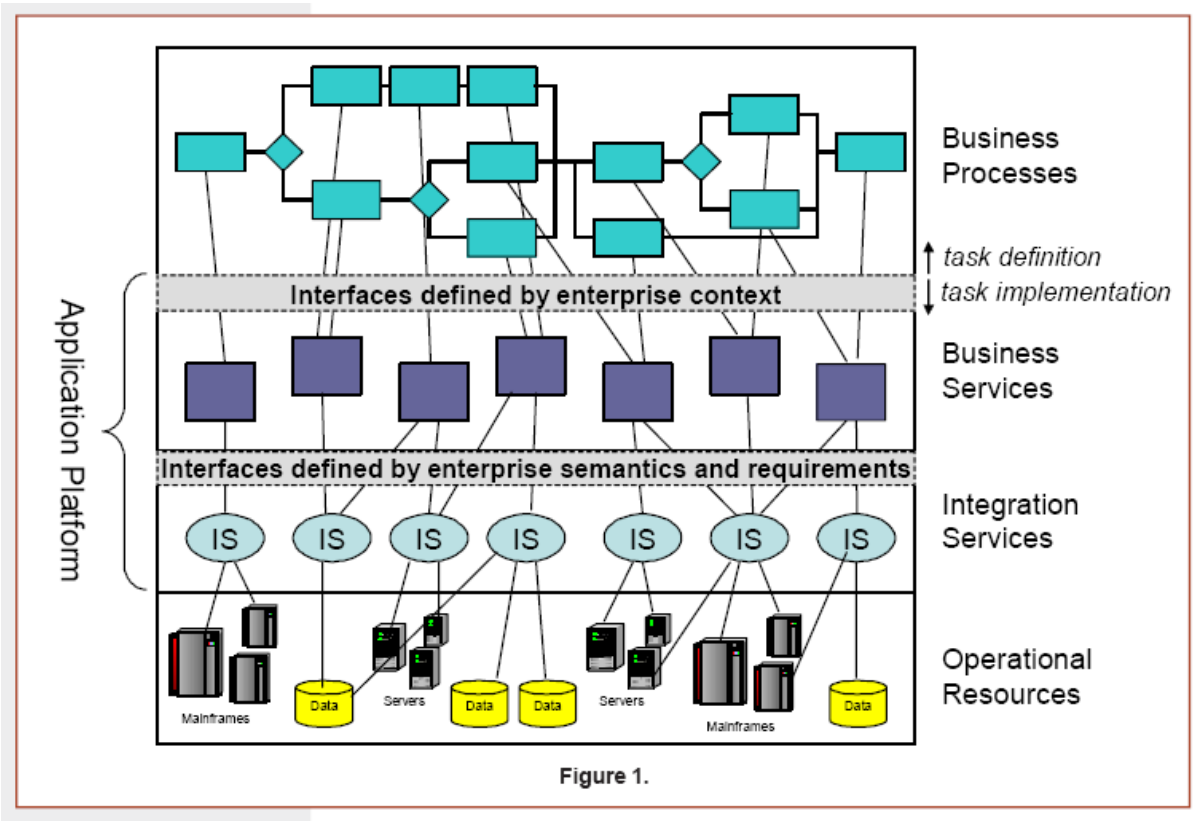
slika 6: Grafički prikaz tehnologija koje obezbeđuju realizaciju SOA koncepata

2.3.4 SOA i BPM

Gde BPM prestaje, a SOA počinje, i obrnuto, ili koja su preklapanja među njima?

BPM osposobljava poslovne analitičare da usaglase IT sisteme sa strateškim poslovnim ciljevima, kreirajući dobro poznate poslovne procese preduzeća, nadledajući njihove performanse i optimizujući ih da bi bili efikasniji. Svaki poslovni proces je modelovan kao skup individualnih zadataka obrade. Ovi zadaci su uobičajeno implementirani kao servisi u okviru preduzeća. BPM sistem obezbeđuje skup alata koji dozvoljava poslovnom analitičaru da kreira modele procesa, koristi notacije kao što je BPMN, i obavlja automatizaciju poslovnih procesa, ili izvršava model, pozivajući servise. Dodatno, BPM sistem može obezbediti nadgledanje i upravljanje.

SOA obezbeđuje aplikacionu platformu koja povezuje poslovne procese i operativne resurse, kao što je prikazano na slici 7. Na nivou poslovnog procesa, SOA obezbeđuje interfejse koji direktno podržavaju izvršavanje poslovnih zadataka, ali definiše te interfejse na način da podrži konzistentnost i mogućnost ponovnog korišćenja. Na nivou operacionih resursa, SOA podržava postojeće mogućnosti kao integracione servise. Ali to ne čini tako što direktno mapira postojeće aplikacije kao servise. U stvari, SOA obezbeđuje interfejse za novi servis, koji se zasniva na semantici preduzeća i funkcionalnim zahtevima, i mapira ih u postojeće sisteme. Na kraju, to povezuje najviše i najniže nivoe zajedno, preko kompozicije servisa da bi se kreirao aplikacioni nivo.

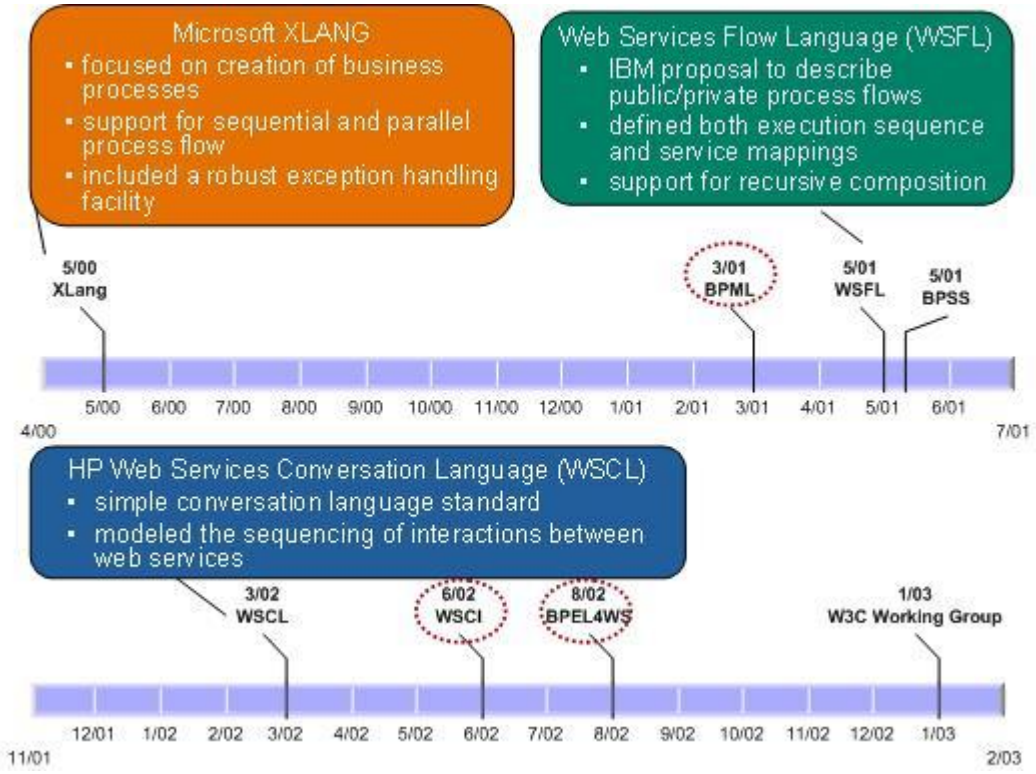


slika 7: BPM i SOA [Rosen]

Zajedno, BPM i SOA obezbeđuju prilično dobru kombinaciju za automatizaciju poslovanja preduzeća. BPM obezbeđuje na visokom nivou apstrakciju za definisanje poslovnih procesa, kao i mogućnost nadgledanja i upravljanja procesima. Servisi obezbeđuju funkcije koje podržavaju te procese. SOA obezbeđuje mogućnost kombinovanja servisa i

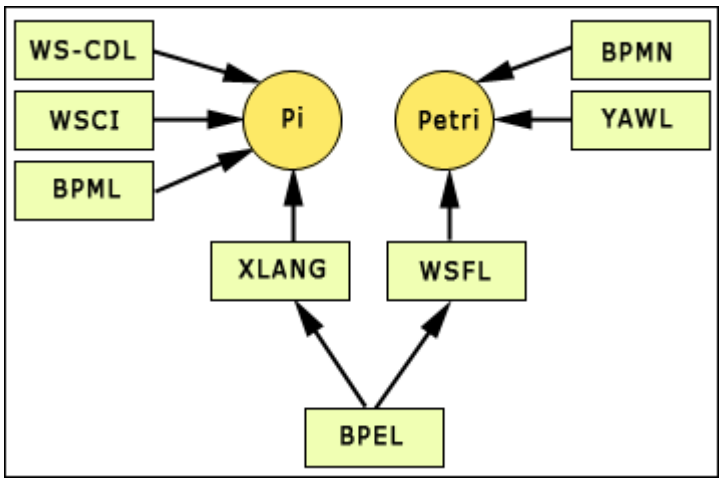
podršku da se kreira agilno, fleksibilno preduzeće. BPM bez SOA je korisno za kreiranje aplikacija, ali teško proširivo. SOA bez BPM je korisna za kreiranje višestruko korišćenih i konzistentnih servisa, ali bez sposobnosti da se ti servisi uključe u agilno, konkurentno preduzeće.

2.3.5 Teoretske osnove BPM standarda



slika 8: Prikaz razvoja standarda za poslovne procese

Većina BPM standarda svoju teoretsku podlogu ima u Pi-Calculus algebarskom sistemu i Petri mrežama.



slika 9: Prikaz stabla teorija za BPM standarde (pravougaonici predstavljaju standarde, krugovi teorije, a strelice zavisnosti. Nivo zavisnosti je ekstremno promenljiv.)

Pi (Pi-Calculus) je 1990-tih godina razvio škotski matematičar Robin Milner i predstavlja algebarski sistem za definisanje konkurentnih procesa, koji komuniciraju među sobom dinamički. Svaki proces se sastoji od jedne ili više akcija, koje mogu biti uređene sekvencijalno, paralelno ili uslovno, kao i rekurzivno. Akcija šalje i prima informacije preko kanala. Prema Pi-Calculus konvenciji, kada jedan proces šalje informacije drugom, on uključuje ime kanala, koji će koristiti drugi proces za odgovor. To je varijabla i ona može menjati svoju vrednost u zavisnosti od promena uslova prilikom davanja odgovora.

Petri (Petri network ili Petri net) je 1962. godine razvio matematičar Carl Adam Petri i predstavlja formalni jezik za grafičko modeliranje procesa. Petri net može da se koristi za implementaciju semantike kontrole toka procesa, uključujući osnovna grananja i pravila za spajanja, kao i za komplikovane sinhronizacione scenarije.

2.3.6 Organizacije za kreiranje standarda

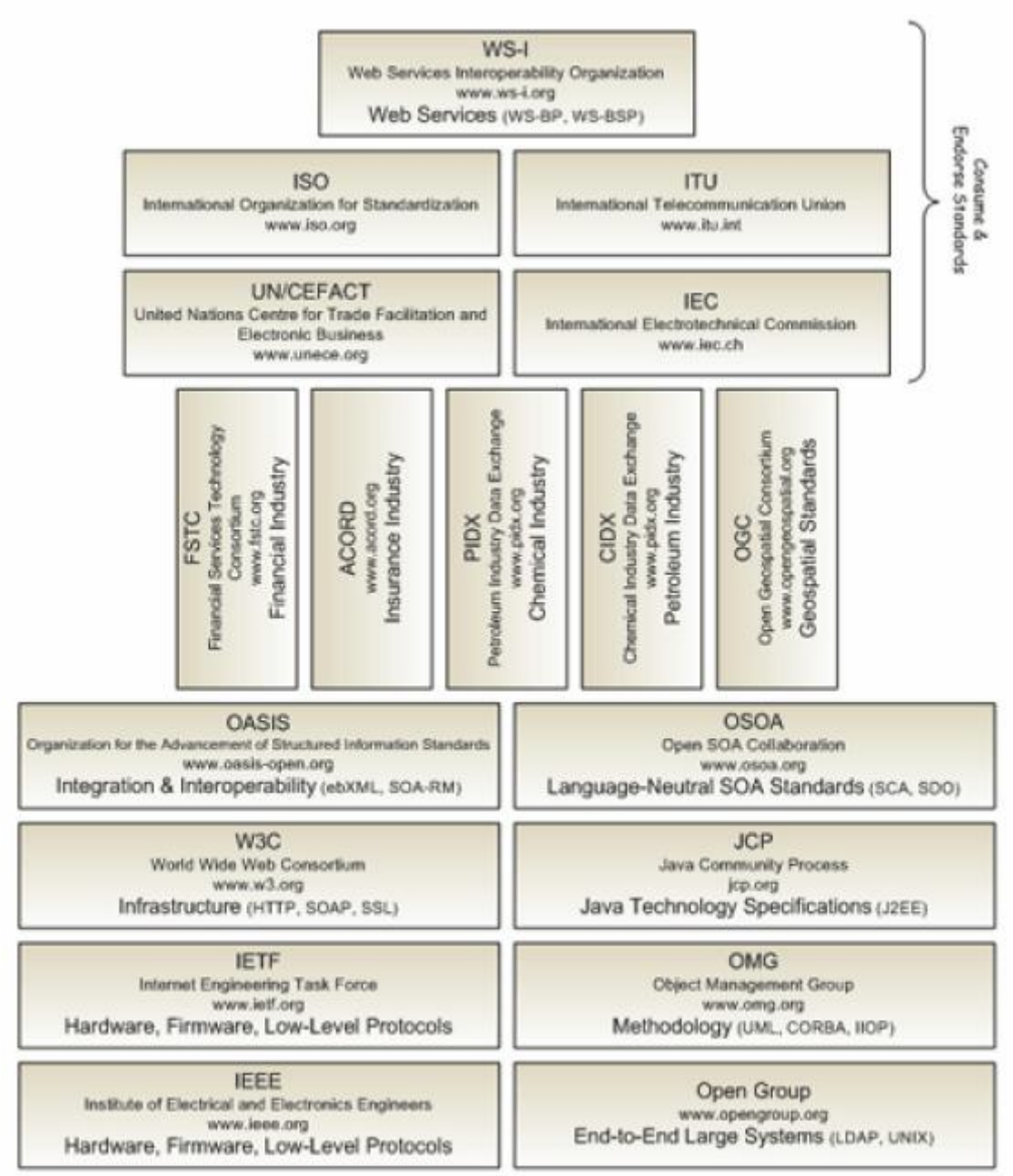
Organizacije za kreiranje standarda su entiteti, čije osnovne aktivnosti predstavljaju razvoj, koordinaciju, objavljivanje, dopunjavanje, ponovno objavljivanje, tumačenje, ili održavanje standarda, koji su važni velikom broju korisnika van organizacije koja kreira standarde.

Pre nego što pregledamo i opišemo različite standarde vezane za poslovne procese, prikazaćemo neke od glavnih organizacija za njihovo kreiranje. Pojedini ljudi koji učestvuju u radu tih organizaciju su odgovorni za kreiranje nekih od najbitnijih tehnologija današnjice.

Najaktivnije organizacije, koje se bave standardima za kreiranje i upravljanje poslovnim procesima su:

- W3C - World Wide Web Consortium
- OASIS – Organization for the Advancement of Structured Information Standards
- WS-I – Web services Interoperability Organization
- JCP – Java Community Process
- OSOA – Open SOA Collaboration
- BPMI - Business Process Management Initiative

Dijagram prikazuje neke od glavnih organizacija za kreiranje metoda, standarda, specifikacija, APIja i slično.



slika 10: Najaktivnije organizacije za kreiranje standarda

3. Modeliranje poslovnih procesa i Web servisi

3.1 Definicija Web servisa

Web servisi predstavljaju najnoviju distribuiranu tehnologiju, kao i najviše odgovarajuću tehnologiju za realizaciju SOA. Web servisi se postali tehnologija koja se uobičajeno koristi za interoperabilnost i integraciju aplikacija i informacionih sistema. Web servisi obezbeđuju tehnološku osnovu za postizanje interoperabilnosti između aplikacija koje koriste različite softverske platforme, operativne sisteme i programske jezike. Izgrađeni su na XML jeziku. Dok je XML u osnovi standard za integraciju na nivou podataka, Web servisi su postali standard za integraciju na nivou servisa. [Juric]

Web servisi su kao distribuirana tehnologija podržani od strane svih glavnih proizvođača softvera. Oni su prva tehnologija koja ispunjava obećanje o univerzalnoj interoperabilnosti između aplikacija koje rade na različitim platformama. Osnovne specifikacije na kojima su zasnovani Web servisi su SOAP, WSDL i UDDI. SOAP, WSDL i UDDI su zasnovani na XML jeziku, što čini da poruke koje stižu do Web servisa, odgovori koji se vraćaju, kao i opis Web servisa, budu ljudski čitljive. [Juric]

Web servis predstavlja softverski sistem dizajniran da podrži interoperabilnu mašina-mašina interakciju preko mreže. On poseduje interfejs opisan u mašinski čitljivom formatu (WSDL). Drugi sistemi komuniciraju sa Web servisom korišćenjem SOAP poruka, koje se obično prenose preko HTTP, XML serializovanim u saradnji sa drugim Web standardima. [Web Services Arch]

Jednostavna definicija bi glasila: Web servis je aplikacija koja obezbeđuje Web API. Za API se može reći da podržava komunikaciju aplikacija-aplikacija. Web API predstavlja API koji dozvoljava aplikacijama da komuniciraju koristeći XML i Web. [Manes]

Web servisi pojednostavljuju proces komunikacije između aplikacija. Pojednostavljenje utiče na smanjenje troškova i vremena razvoja, lako održavanje, kao i smanjenje ukupnih troškova.

Dva sistema koriste Web servise da komuniciraju međusobno kada aplikacija kreira XML dokument u obliku poruke i pošalje ga preko mreže do Web servisa. Opciono, Web servis šalje odgovor na zahtev u formi XML dokumenta. Web servis standardi definišu interfejs poruke, format, mapiranje sadržaja poruke u implementaciju servisa, opciona zaglavlja, kao i način za publikovanje servisa i otkrivanje drugih Web servisa.

Neke od ključnih osobina Web servisa su [IBM SOA]:

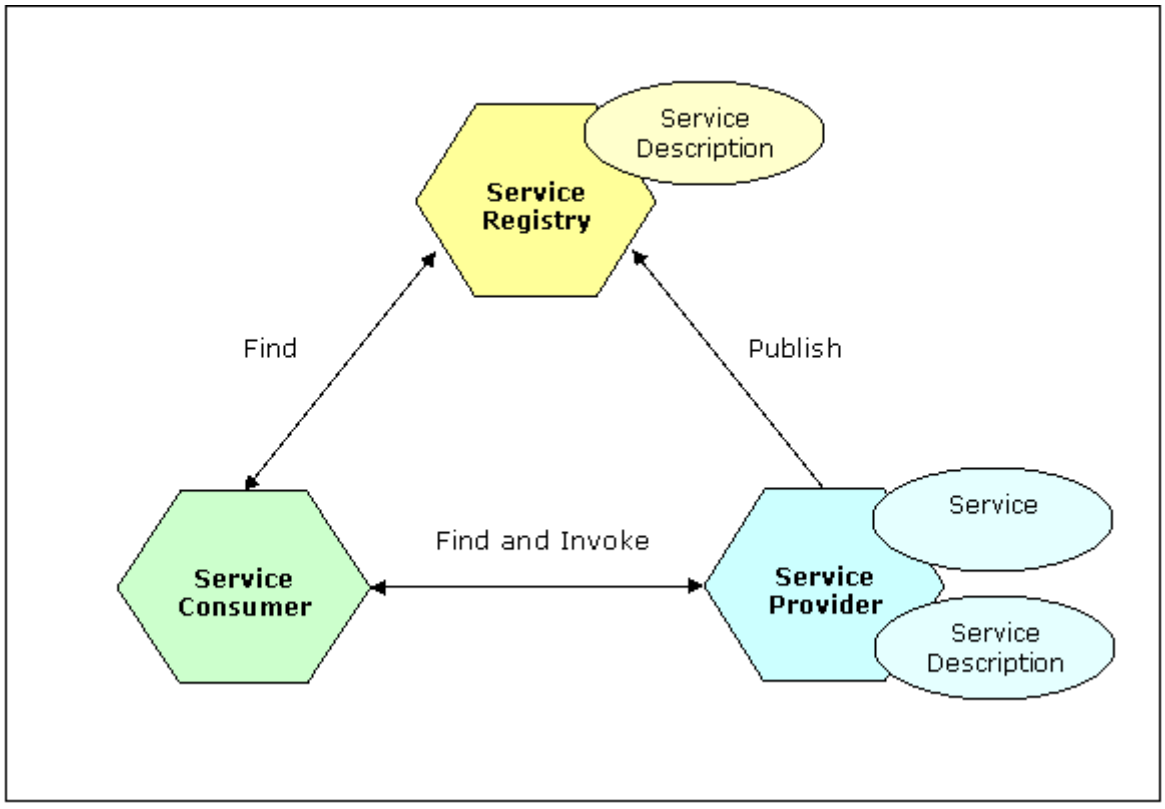
- **Web servisi su samostalni.**
  Na klijentskoj strani nije potreban dodatni softver. Dovoljan je samo programski jezik sa XML i HTTP podrškom. Na serverskoj strani, potreban je Web server.
- **Web servisi su samoopisujući.**
  Definicija formata poruke se nalazi u samoj poruci. Nisu potrebna nikakvi dodatna skladišta za meta podatke, niti alati za njihovo kreiranje.
- **Web servisi su modularni.**
  Web servisi predstavljaju tehnologiju za isporučivanje i obezbeđivanje pristupa poslovnim funkcijama preko Web-a; J2EE, CORBA, kao i drugi standarda koji predstavljaju tehnologije za primenu Web servisa.
- **Web servisi mogu biti publikovani, locirani i pokrenuti preko Weba.** Za to su neophodni sledeći standardi:
  - Simple Object Access Protocol (SOAP)
  - Web Service Description Language (WSDL)

    - Universal Description, Discovery, and Integration (UDDI)
- **Web servisi su nezavisni od jezika i interoperabilni.**
  Interakcija Web servisa i njegovog klijenta je dizajnirana tako da bude nezavisna od platforme i jezika. Njihova interakcija zahteva WSDL dokument za definiciju interfejsa i opis servisa, zajedno sa mrežnim protokolom. Pošto ni Web servis, ni njegov klijent ne znaju međusobno na kojim se platformama nalaze i pomoću kojeg jezika su kreirani, interoperabilnost je obezbeđena.
- **Web servisi su bazirani na otvorenim standardima.**
  XML i HTTP predstavljaju tehničku osnovu Web servisa. Veliki deo Web servis tehnologije je kreiran koristeći Open Source projekte.
- **Web servisi su dinamički.**
  Opis i pronalaženje Web servisa se mogu automatizovati preko UDDI i WSDL.

3.2 Arhitektura Web servisa

Popularna interpretacija Web servisa se zasniva na IBM-ovoj arhitekturi, koja se sastoji od tri elementa:

- **Service Consumer**: Predstavlja aplikaciju, softverski modul ili drugi Web servis, koji inicira traženje Web servisa u registru, povezuje se sa Web servisom i izvršava Web servis operaciju.
- **Service Provider**: Predstavlja mrežno adresni entitet, koji prihvata i izvršava zahteve od Service Consumer. Takođe publikuje opis svojih servisa u Service Registry, tako da ga potencijalni korisnici mogu pronaći.
- **UDDI (Service Registry)**: Obezbeđuje otkrivanje i pronalaženje Web servisa. Sadrži skladište dostupnih Web servisa i dozvoljava pretraživanje informacija koje sadrži od strane zainteresovanih potencijalnih korisnika.



Slika 11: Arhitektura Web servisa

Operacije kod arhitekture Web servisa:

- **Publish**: Da bi bio dostupan, Web servis mora biti objavljen, tako da ga zainteresovani korisnici mogu otkriti i pozvati na izvršavanje.
- **Find**: Zainteresovani potencijalni korisnik pronalazi Web servis ispitivanjem registra servisa po nekom kriterijumu.
- **Bind and Invoke**: Posle pronalaženja opisa Web servisa, korisnik se povezuje sa Web servisom i poziva ga na izvršavanje u skladu sa informacijama koje se nalaze u opisu Web servisa.

Artifakti kod arhitekture Web servisa:

- **Service**: Servis stavljen na raspolaganje za korišćenje preko objavljenog interfejsa.
- **Service Description**: Predstavlja opis Web servisa, koji specificira način interakcije sa Servis provajderom. Opisuje se format zahteva i odgovor od strane servisa.

3.3 Osnovne tehnologije vezane za Web servise

Web servisi predstavljaju tehnologiju koja je relativno brzo široko prihvaćena kao način za integraciju aplikacija. Razlog leži u činjenici da Web servisi obezbeđuju distribuirani pristup za integraciju veoma različitih aplikacija putem Interneta. Specifikacija Web servisa je potpuno nezavisna od programskog jezika, operativnog sistema, kao i hardvera, i obezbeđuje slabu povezanost između Web servisa i aplikacije koja ga konzumira. Web servisi su bazirani na otvorenim tehnologijama kao što su:

- eXtensible Markup Language (XML),
- SOAP,
- Web Services Description Language (WSDL),
- Universal Description, Discovery and Integration (UDDI).

3.3.1 XML

Extensible Markup Language (XML) opisan je od strane radne grupe World Wide Web Consortium (W3C), koja ga je kreirala, na sledeći način: "XML je podkategorija SGML (Standard Generalized Markup Language). Njegov cilj je da izvornom SGML-u omogući postavljanje, preuzimanje i obradu na Web-u na način kako je to danas moguće na HTML (Hypertext Markup Language). XML je napravljen sa ciljem da bude jednostavan za primenu i da omogući kombinovanu upotrebu sa SGML-om i HTML-om." [XMLSpec]

XML definiše skup standarda za označavanje strukture i sadržaja dokumenta, koji omogućavaju njihovu čitljivost na različitim platformama i u okviru različitih aplikacija.

XML - sintaksa

XML predstavlja meta jezik za označavanje, što znači da može da se koristi da definiše druge jezike za označavanje. Struktura XML dokumenta se definiše korišćenjem tagova, koji sadrže labelu i vrednost za svaki element. Jedan tag predstavlja reč ili skup karaktera u okviru znakova <>.

XML predstavlja jezik za kreiranje elektronskih dokumenata. Jedan XML dokument predstavlja hijerarhiju XML elemenata. Svaki element reprezentuje deo informacija koje su sadržane u dokumentu.

Sadržaj XML dokumenta uključuje:

- procesing instrukciju,
- elemente,
- atribute i
- komentare.

XML ne sadrži ograničeni skup unapred definisanih elemenata, i zbog toga se elementi od kojih se sastoji jedan XML dokument moraju pisati tako da poštuju sledeća pravila:

- preporučeni početak XML dokumenta je
  <?xml version="1.0" encoding="UTF-8" ?>
- element može sadržati podatke i druge elemente
- element se sastoji od početnog taga i završnog taga
- imena tagova nisu unapred definisana
- svaki dokument mora imati najmanje jedan element, koji se naziva koreni tag
- svi ostali tagovi su hijerarhijski ispod korenog taga
- imena elemenata ne smeju sadržati razmak, niti mogu počinjati sa brojem ili interpunkcijskim znakom
- imena elemenata su case-sensitive
- svaki početni tag mora imati odgovarajući završni tag (<ime> i </ime>)
- ukoliko ne postoji sadržaj, element se definiše na sledeći način:
  <ime></ime> ili <ime/>
- u okviru početnog taga se mogu definisati osobine elemenata tj. dodati atributi

Poštovanje pravila definisanih u prethodnom tekstu čini XML dokument dobro formiranim (well-formed). Svi XML dokumenti moraju biti dobro formirani.

Osim toga, XML dokumentu se može proveravati ispravnost. XML dokument je validan ukoliko njegov sadržaj odgovara skupu zahteva koji su definisani u odgovarajućoj šemi. U XML-u, šema predstavlja formalni opis korišćenih XML elemenata. Za opis šeme jednog XML dokumenta može se koristiti DTD (Document Type Definition), XSD (XML Schema Definition) ili XDR (XML-Data Reduced).

Zašto XML?

XML može strukturno i hijerarhijski opisati podatke. Takođe, može definisati relacije među podacima, i sve to korišćenjem tekstualnog sadržaja baziranog na Unicodu.

Prednosti korišćenja XML dokumenata:

- jednostavnost korišćenja (čitljivost od strane čoveka i mašine, postojanje malog broja pravila za kreiranje),
- proširivost (mogućnost kreiranja sopstvenih elemenata),
- čitljivost na različitim platformama i u okviru različitih aplikacija,
- otvorenost (kompletno otvoren standard, slobodno dostupan na Web-u).

Ukoliko se XML koristi kao format za podatke u okviru jedne aplikacije, upotreba XML-a je svrsishodna, ako je zadovoljeno bar četiri od sledećih navedenih kriterijuma:

- postoji potreba za razmenjivanjem podataka između različitih softverskih platformi,
- postoje odgovarajući alati za kreiranje ili korišćenje XML podataka,
- zahtev za brzinom u radu sa podacima nije veoma važan,
- sadržaj nije binarni (slike, muzički fajlovi),
- XML dokument ne sadrži nedozvoljene kontrolne karaktere.

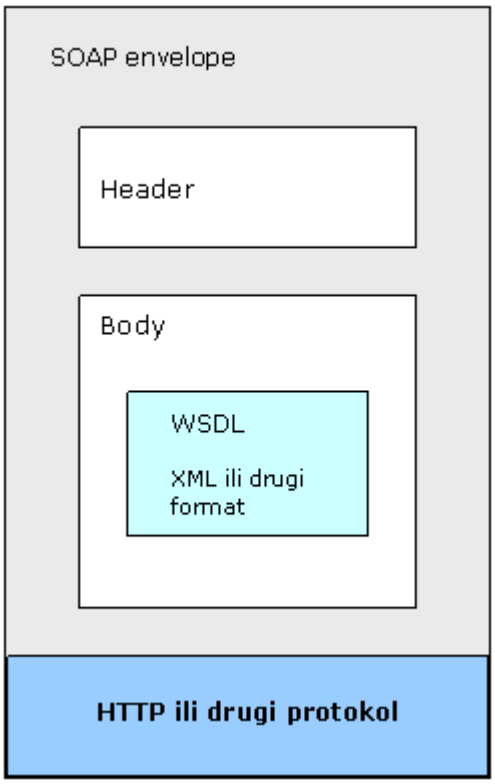
3.3.2 SOAP

SOAP je protokol kreiran za transfer XML poruka. SOAP predstavlja komunikacioni mehanizam za povezivanje Web servisa. Kao i XML, SOAP održava W3C konzorcijum, čiji je cilj razvoj interoperabilnih specifikacija i vodiča. U vreme pisanja ovog rada, aktuelna je verzija SOAP 1.2. Do pojave poslednje verzije, slova u nazivu SOAP su najčešće objašnjavana kao skraćenica od Simple Object Access Protocol. Od pojave poslednje SOAP verzije, slova u nazivu nemaju neko posebno značenje.

SOAP se sastoji od tri dela:

1. Prvo deo obezbeđuje omotač za slanje Web servis poruke preko Internata ili intraneta. Naziva se SOAP Envelope i on:
    - opisuje šta se nalazi u poruci,
    - sadrži pravila kako procesirati poruku,
    - uključuje informacije na osnovu kojih Web servis treba da odgovori na poruku,
    - sadrži informacije koje ukazuju da li je odgovor na poruku opcioni ili obavezan,
    - sastoji se od opcionog dela koji se naziva Header, koji predstavlja zaglavlje poruke, preko kojeg se mogu obezbediti informacije o proveri identiteta, enkodiranju podataka, ili kako bi primalac SAOP poruke trebao da je obradi,
    - i ima obavezan deo Body, koji predstavlja telo poruke, jer sadrži samu poruku, koja može biti definisana korišćenjem WSDL specifikacije.
2. SOAP takođe ima i skup pravila za definisanje specijalnih tipova podataka za Web servis. Ova osobina je korisna, jer dozvoljava definiciju jedinstvenih tipova podataka za jedan servis. Bilo koji podatak koji može biti opisan kao XML struktura, može biti ugrađen i korišćen u okviru SOAP poruke.
3. SOAP uključuje pravila za opis metoda Web servisa, pri čemu se uključuje struktura za opis poziva Web servisa i odgovora Web servisa.

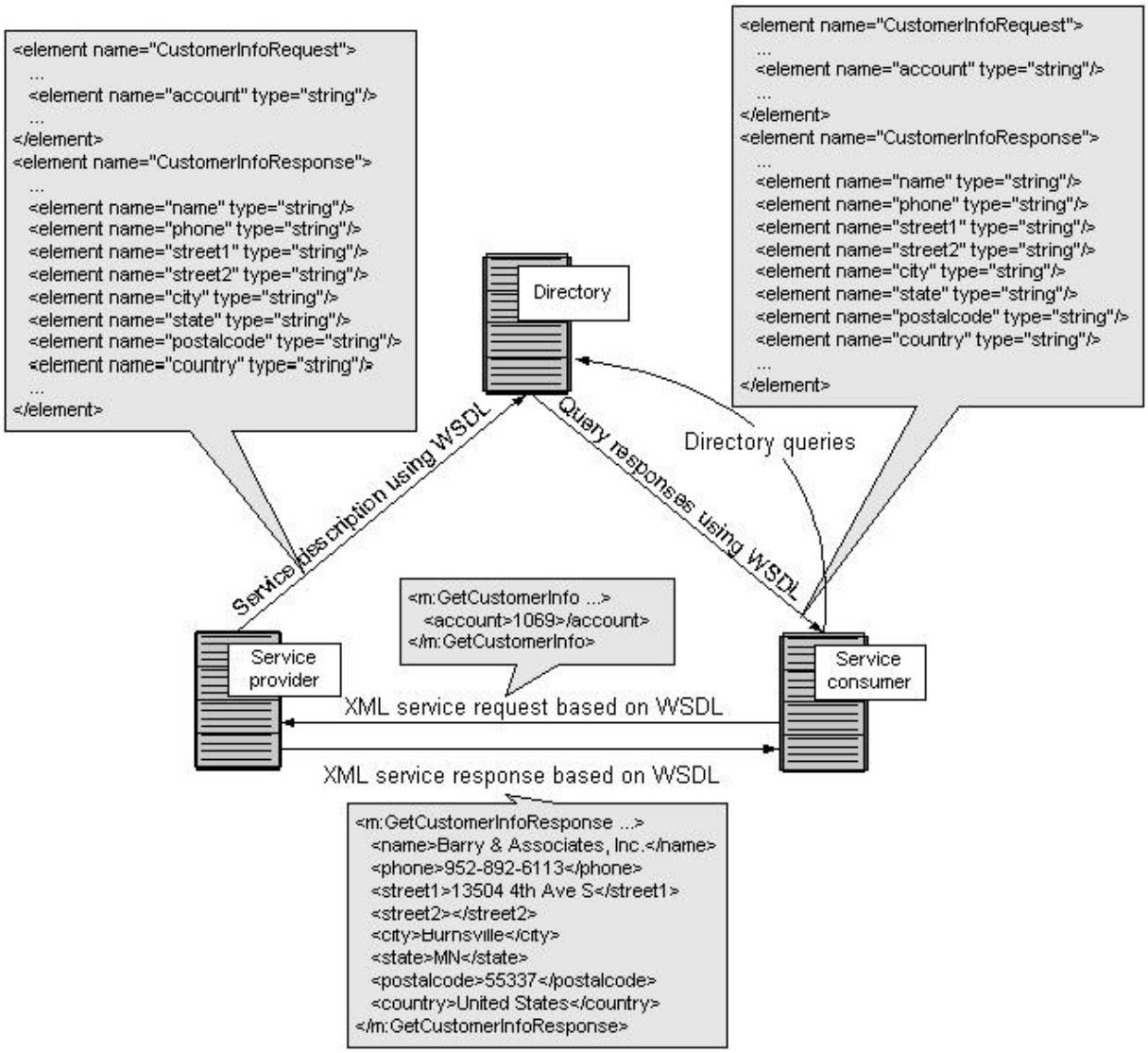
SOAP uobičajeno koristi HTTP protokol, ali i drugi protokoli kao što je SMTP (Simple Mail Transfer protokol) mogu biti korišćeni.



Slika 12: Grafički prikaz delova SOAP poruke

```
<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance" xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">

<soap:Header>
   <MyHeader xmlns="http://tempuri.org/">
      <KorisnickoIme>string</KorisnickoIme>
      <Lozinka>string</Lozinka>
   </MyHeader>
 </soap:Header>

<soap:Body>
   <PreuzmiCenovnik xmlns="http://tempuri.org/ ">
      <Datum>string</Datum>
   </PreuzmiCenovnik>
  </soap:Body>

</soap:Envelope>
```

Slika 13: Primer SOAP poruke



Slika 14: Slika detaljnije prikazuje Web servis poruke

3.3.3 WSDL

Web Services Description Language (WSDL) predstavlja na XML-u zasnovan jezik za opis Web servisa. WSDL dokument opisuje šta Web servis radi, kako se sa njim može komunicirati i gde se može pronaći. Delovi opisa Web servisa se mogu nalaziti u više dokumenata, što obezbeđuje više fleksibilnosti i povećava mogućnost ponovnog korišćenja.

Prilikom pretraživanja registra Web servisa, može se dobiti njegov opis sadržan u WSDL dokumentu.

WSDL ima sličnu ulogu kao IDL (Interface Definition Language) kod konvencionalnih middleware platformi:

- opisuje servis,
- može biti korišćen da automatski generiše kod za poziv servisa.

Kao i IDL jezici, WSDL ne sadrži informacije o:

- semantici,
- poslovnim protokolima i komunikaciji.

Opis delova WSDL dokumenta

<portType>
- definiše interfejs od Web servisa,
- u specifikaciji WSDL 2.0, <portType> će promeniti ime u <interface>
- sastoji se od niza deklaracija za operacije
- WSDL dokument sadrži nula ili više <portType> elemenata. Obično samo jedan.
- <portType> ima atribut name, čije ime mora biti jedinstveno, i preko kojeg se element <binding> referiše na portType

<operation>
- definiše potpis metode: ime, input, output i fault
- input i output elementi su vezani za message
- različite kombinacije input/output definišu različite tipove operacija

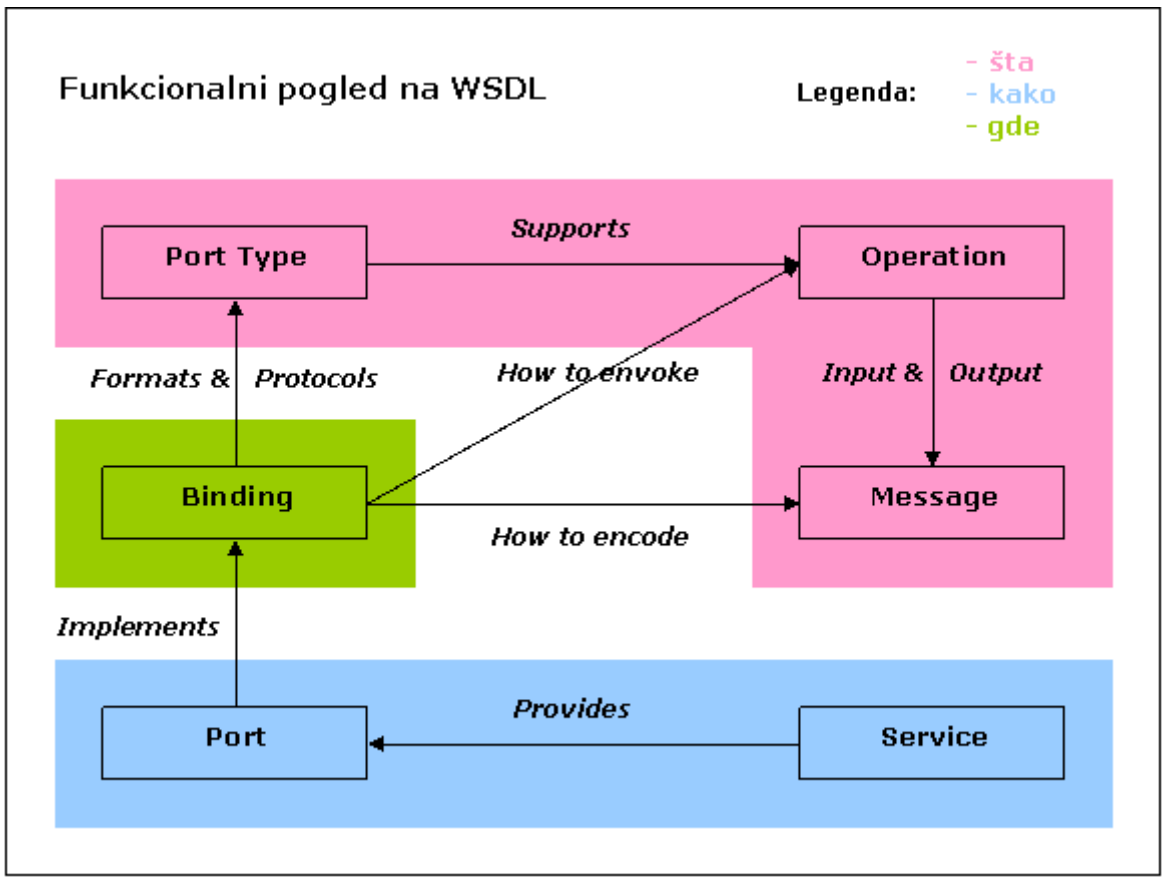
<message>
- opisuje apstraktni oblik za input, output i fault
- WSDL može imati nula ili više <message> elemenata
- svaki **<message>** element ima ime, koje je jedinstveno u okviru dokumenta
- svaki **<message>** sadrži kolekciju <part> elemenata

<part>
- nalazi se u okviru <message> elementa
- može se porediti sa parametrima u metodi
- ima dva svojstva: <name> i <kind>
- <kind> može biti ili <type> ili <element>
- <element> se odnosi na element definisan u XML šemi
- <type> se odnosi na simpleType ili complexType u XSD dokumentu

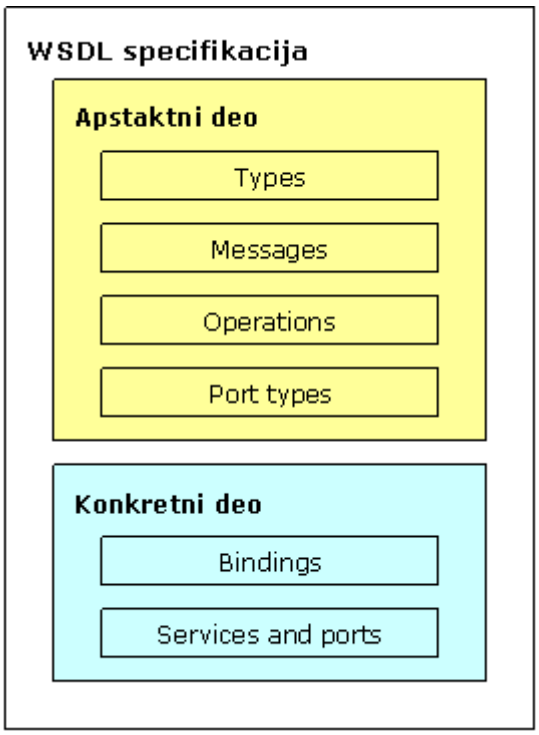
<types>
- podrazumeva se da je type sistem zasnovan na XML Schema

<binding>
- link na <portType> se postiže preko <portType> name atributa
- uobičajeno je da postoj isamo jedan <binding> element
- definiše SOAPaction, kao i pojavljivanje input i output poruka

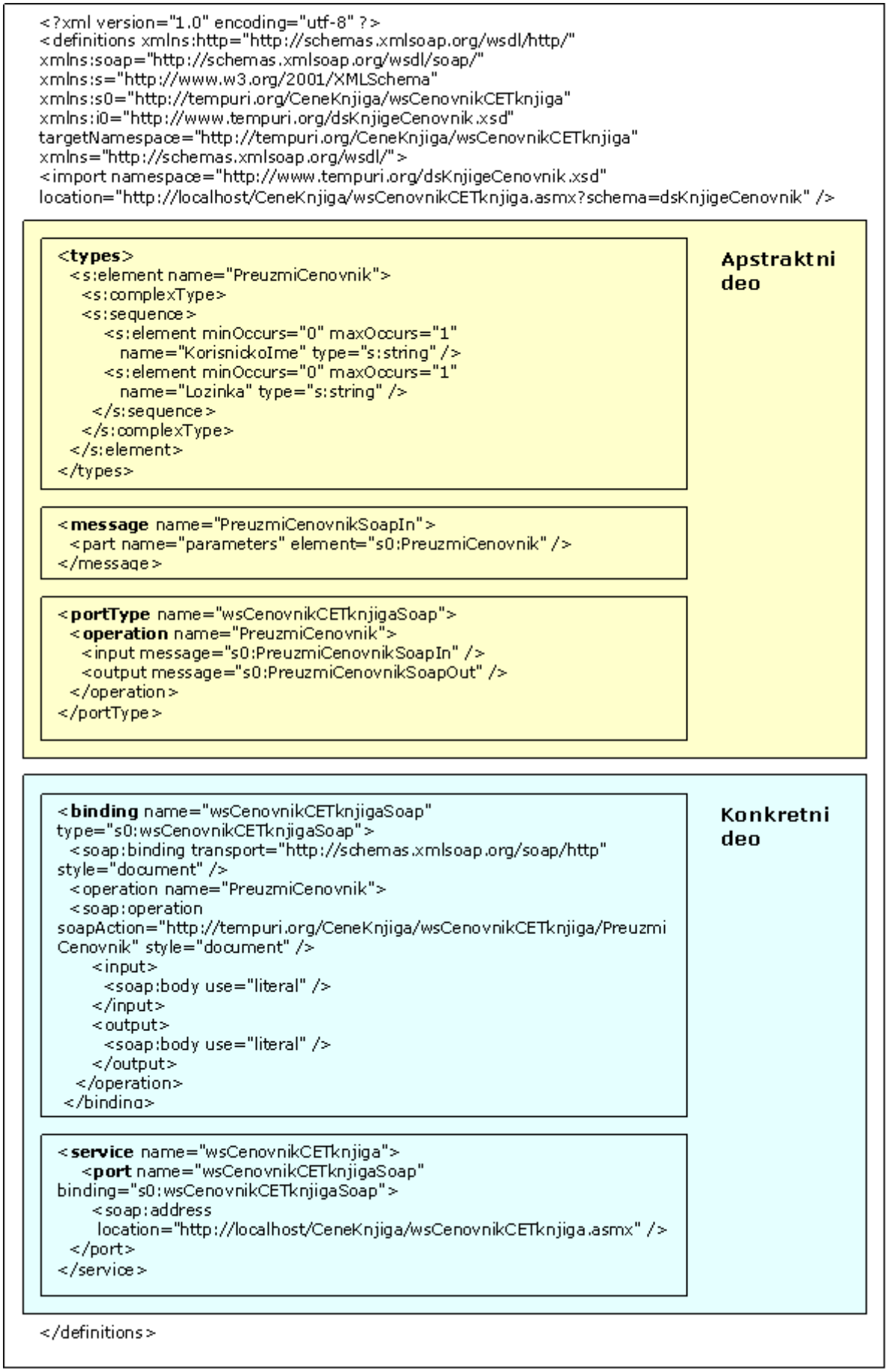
<service>
- sadrži skup <port> elemenata
- definiše interfejs za povezivanje sa Web servisom preko mrežnog protokola koristeći URI



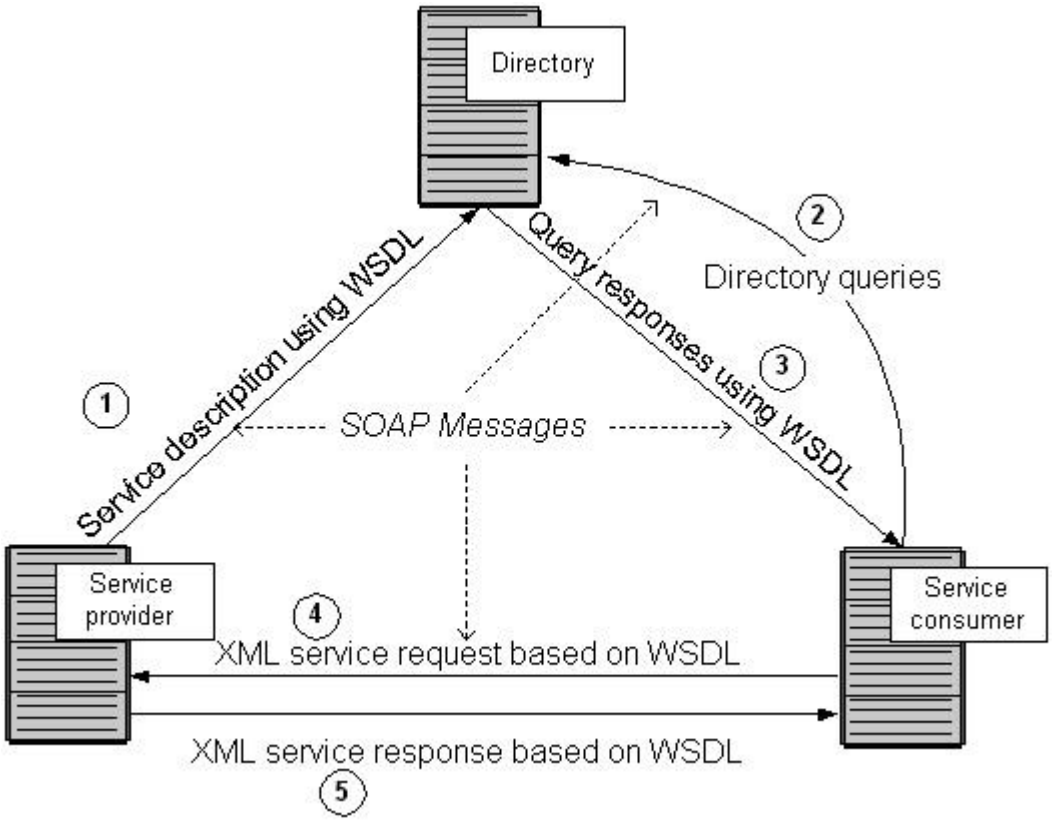
Slika 15: Funkcionalni pogled na WSDL



Slika 16: Strukturni pogled na WSDL dokument



Slika 17: Primer WSDL dokumenta



Slika 18: Ilustracija korišćenja WSDL

WSDL elementi se uobičajeno automatski generišu prilikom kreiranja Web servisa.

W3C konzorcijum je prihvatio WSDL kao standard za opis Web servisa. U vreme pisanja ovog rada, aktuelna je verzija WSDL 1.2. U izradi je specifikacija WSDL 2.0. U sledećoj tabeli je dat uporedni prikaz aktuelne i predložene, nove specifikacije.

Tabela 1: WSDL 1.1 vs WSDL 2.0

WSDL 2.0	**WSDL 1.1**
• Endpoints • Interfaces – Podrška za nasleđivanje interfejsa • Uklonjena je podrška za preklapanje operacija • Messages su definisane preko Types • Operations su ugnježđeni u Interfaces • Endpoints su ugnježđeni u Bindings • 9 Message Exchange Patterns • Novo: Features i Properties	• Ports • PortTypes • Podržava preklapanje (oveloading) operacija • Messages se sastoje od Parts • 6 Top level elemenata: Messages, Operations, PortTypes, Bindings, Ports i Services. • 4 Transmission Primitives (One-way, Request-Response, Solicit-Response, Notification)

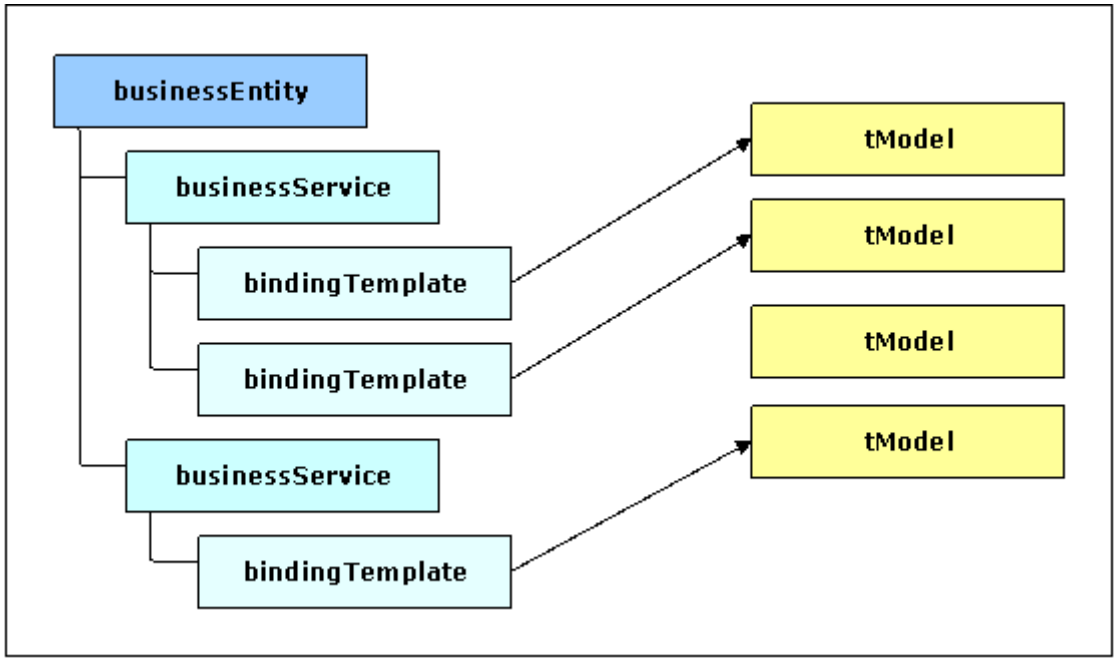
3.3.4 UDDI

Universal Description, Discovery, and Integration (UDDI) obezbeđuje mehanizam za oglašavanje i otkrivanje Web servisa. UDDI predstavlja registar za Web servise, s tim što je i sam UDDI Web servis. Osnovna razlika između UDDI registra i drugih registara i direktorijuma je što UDDI obezbeđuje mehanizam za kategorizaciju poslova i servisa korišćenjem više klasifikacionih šema (taksonomija). Ove taksonomije pomažu korisnicima Web servisa da pronađu odgovarajući servis, koji odgovara njihovim zahtevima.

UDDI registru se može pristupiti preko Web interfejsa ili koristeći automatizovane programske metode.

UDDI uključuje četiri primarna tipa podataka:

- businessEntity (opisuje Service Provider),
- businessService (sadrži ne-tehničke podatke o Web servisu),
- bindingTemplate (sadrži tehničke informacije za pristup Web servisu, na primer URL),
- tModel (tehnički Model).

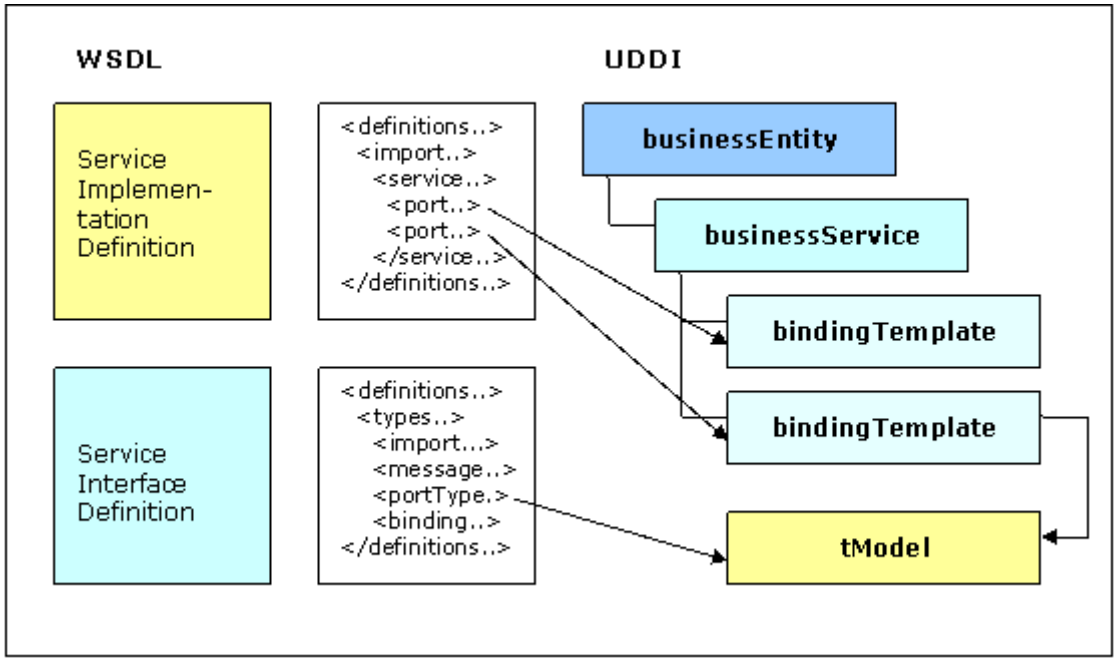


Slika 19: UDDI tipovi podataka

Zajedno, ovi elementi se koriste da bi obezbedili sledeće vrste informacija:

- bele strane - podatke o Service Provider (ime, adresa, osoba za kontakt),
- žute strane - koji tipovi servisa se nude,
- zelene strane - tehničke informacije kako koristiti ponuđene servise, uključujući pokazivače na WSDL opise servisa, koji se ne nalaze u UDDI registru.

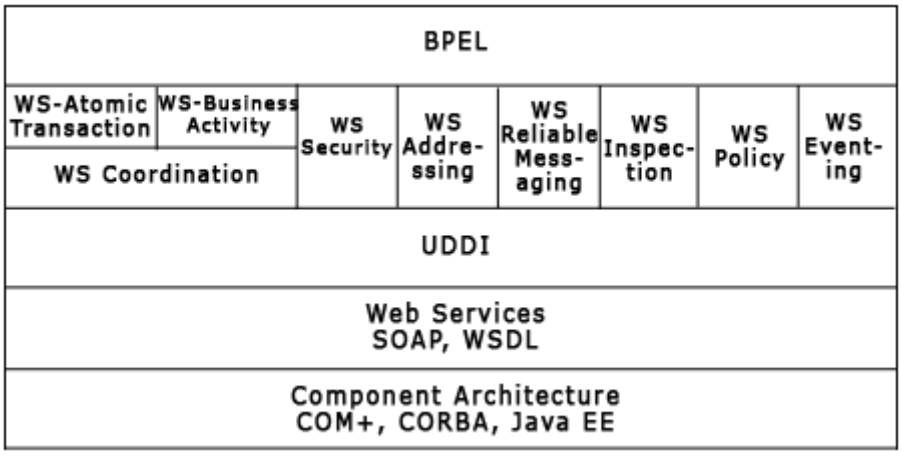
Sledeća slika prikazuje vezu između UDDI registra i WSDL dokumenta.



Slika 20: Mapiranje WSDL u UDDI

3.4 Dodatne tehnologije vezane za Web servise

SOAP, WSDL i UDDI obezbeđuju osnovnu infrastrukturu za Web servise. Međutim, u stvarnim aplikacijama, neophodna je dodatna funkcionalnost, pre svega za obezbeđenje pouzdanosti, sigurnosti, transakcija, razmenu složenih poruka. Postoji više predloga i specifikacija, koji pokrivaju skoro sve aspekte Web servisa.



slika 21: Grafički prikaz Web servis tehnologija

Primeri nekih od postojećih predloga i specifikacija su:

- **WS-Coordination**
  Predstavlja proširivo okruženje za obezbeđenje protokola koji koordiniraju akcije distribuiranih transakcija. Definisano okruženje obezbeđuje Web servisu da kreira kontekst neophodan za proširenje aktivnosti na druge servise i da funkcioniše u heterogenim okruženjima.
- **WS-Transaction**
  Uključuje podršku za dva tipa transakcija i odgovarajućih specifikacija: Atomic Transaction (**WS-AtomicTransaction**), koje rade po principu sve-ili-ništa za

kratkotrajne aktivnosti i moraju podržavati ACID svojstva, i Business Activity (**WS-BusinessActivity**), koje se koriste za koordinaciju dugotrajnih aktivnosti koje se odnose na poslovnu logiku.

- **WS-Security**
  Opisuje kako postići integritet, koji dozvoljava primaocu da bude siguran da podaci koje je primio preko poruke nisu modifikovani u transportu, i poverljivost, koja obezbeđuje da podaci ne mogu biti čitani u transportu. Takođe opisuje kako slati sigurnosne tokene, kao što su kombinacije korisnik/lozinka, Kerberos tiket, ili neku vrstu sertifikata.
- **WS-ReliableMessaging**
  Pouzdana isporuka poruka je veoma važna zbog obezbeđenja integriteta sistema. Neisporučivanje poruka se može desiti zbog mrežnih grešaka, grešaka u radu komponenti, kao i zbog grešaka u radu sistema. WS-ReliableMessaging specificira standarde za pouzdanu isporuku poruka, tako što se: identifikuje svaka poruka, prati njihova isporuka, i garantuje isporuka poruka od izvora do cilja, u ispravnom redosledu odašiljanja.
- **WS-Addressing**
  Specifikacija proširuje mogućnosti koje obezbeđuje WSDL jezik vezano za definisanje adrese poruka, tako što omogućava sledeće:
  - opis adrese se može dinamički generisati i prilagoditi,
  - pomaže u kreiranju kreira novih instanci servisa,
  - adresne informacije mogu biti deljene između komunikacionih partnera u čvrsto povezanim okruženjima.
- **WS-Inspection**
  Definiše specifikacije koje se koriste za dinamičko otkrivanje dokumenata koji opisuju Web servis. WS-Inspection dokument ne sadrži opis servisa, on samo pomaže u pronalaženju opisa željenog dokumenta.
- **WS-Policy**
  Specifikacija je dizajnirana za kreiranje dokumenata vezanih za polise i definiše Web servis polise neophodne za komunikaciju sa drugim servisima. Specifikacija ne definiše kako transportovati ili pronaći polisu.
- **WS-Eventing**
  U poslovnom procesu, servis može biti zainteresovan da primi obaveštenje kad god se desi određeni tip događaja kod drugih učestvujućih servisa. Web servis može da se pretplati kod drugih Web servisa koji generišu događaje koji ga interesuju. Specifikacija dozvoljava kreiranje i brisanje pretplate na događaje. Za svaku pretplatu se može definisati vreme isteka, posle čeka pretplatnik može obnoviti svoju pretplatu, ili je prekinuti.

3.5 Sigurnost kod Web servisa

Pod sigurnošću softverskog sistema se podrazumeva zaštita od neovlašćenog čitanja, promene ili uništavanja informacija. Nivo sigurnosti koji treba da se postigne zavisi od potreba konkretnog softverskog sistema.

Komunikacija sa Web servisima se obavlja pomoću poruka preko Interneta, koji je poznat po svojoj nesigurnosti. Otvorena priroda Interneta omogućava da paketi mogu biti lako preuzeti, pročitani ili promenjeni od strane neovlašćenih osoba.

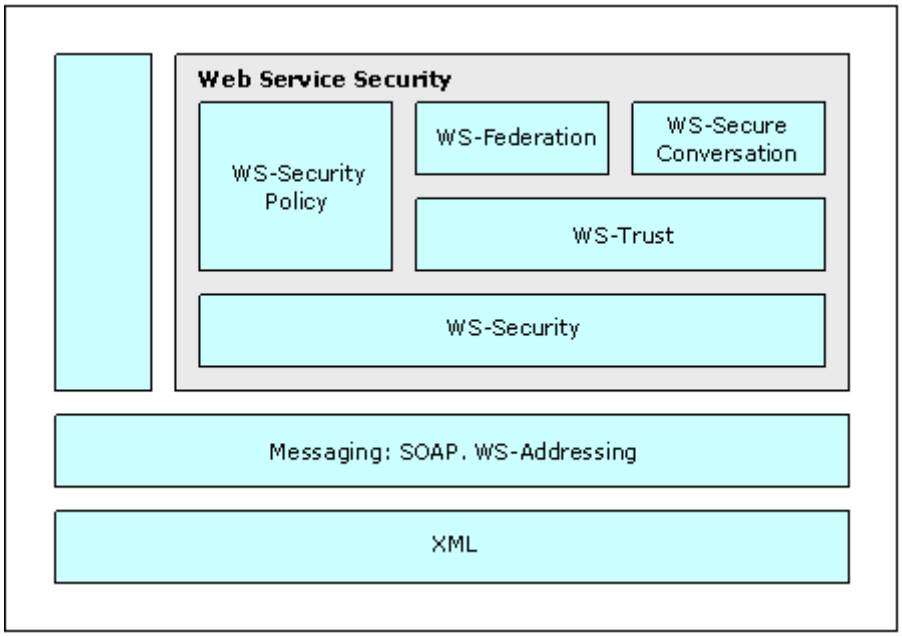
Bez primenjene sigurnosti, Web servisi nisu zaista korisni, i nikada ne bi mogli da dostignu sav svoj potencijal i značaj u povezivanju aplikacija.

3.5.1 Sigurnosni standardi

Verovatno najaktivniji delovi XML Web servis zajednice su grupe koje razvijaju standarde za sigurnost.

U ovom trenutku, postoji više sigurnosnih standarda, koji se nalaze u različitim fazama razvoja od strane više različitih organizacija, kao i više raznih implementacija u sličnim fazama dovršenosti.

Sa jedne strane, postojanje velikog broja sigurnosnih specifikacija je dobro, jer je njihovo postojanje neophodno za široko prihvatanje i korišćenje Web servisa. Sa druge strane, veliki broj specifikacija vezanih za sigurnost Web servisa dovodi do konfuzije kod odabira konkretnih specifikacija za korišćenje, odnosno zahteva veći broj posvećenih divelopera.



Slika 22: Odnosi između Web servis specifikacija

WS-Security

WS-Security standard definiše proširenja za SOAP, tako što obezbeđuje mahanizam za pridruživanje sigurnosnih tokena SOAP poruci. Sigurnosni token može biti binarni, X.509 sertifikat, Kerberos tiket. Standard je potpuno proširiv i može podržati mnogo tipova tokena. WS-Security definiše nekoliko tagova koji mogu uključiti sigurnosne informacije u XML dokument. Takve informacije se ugrađuju u SOAP zaglavlje.

Primer SOAP zaglavlja sa uključenim Security elementom za definisanje sigurnosti:

```
<S:Header>
        <wsse:Security
                xmlns:wsse="http://schemas.xmlsoap.org/ws/2002/04/secext">
                ...
        </wsse:Security>
</s:Header>
```

Sigurnosne informacije se mogu sastojati od imena korinika i njegove lozinke, mogu sadržati digitalni sertifikat ili informacije o algoritmima korišćenim za kriptovanje tela poruke.

Specifikacije vezane za sigurnost Web servisa su dizajnirane da rade zajedno kako bi obezbedile potreban nivo sigurnosti. Međutim, svaka pojedinačna specifikacija poseduje svoju sopstvenu funkcionalnost:

- WS-SecurityPolicy. Opisuje mogućnosti i ograničenja sigurnosnih polisa (na primer, zahtevani sigurnosni tokeni, podržani algoritmi za kriptovanje, ili zaštićeni delovi poruka).
- WS-Trust. Opisuje trust modele koji omogućavaju Web servisima da sigurno sarađuju.
- WS-SecureConversation. Definiše proširenja u odnosu na WS-Security standard za obezbeđenje sigurne komunikacije, odnosno definiše nekoliko tipova tokena i obezbeđuje mehanizam za uspostavljanje sigurne sesije između Web servisa koji komuniciraju međusobno u dužim intervalima.
- WS-Federation. Obezbeđuje skupu organizacija da uspostavi jedan virtualni sigurnosni domen, odnosno definiše mehanizam za obezbeđenje informacija o identitetu, atributima, autentifikaciji i autorizaciji kod servisa koji se nalaze u različitim sigurnosnim domenima.

3.5.2 Primena sigurnosti kod Web servisa

Najjednostavniji način za obezbeđenje sigurnosti Web servisa je kriptovanje mrežnog transporta korišćenjem HTTPS i SSL protokola. Međutim, ovaj tip sigurnosti je prilično grub, jer predstavlja sve-ili-ništa pristup i ne omogućava mnogo kontrole nad njegovim sprovođenjem. Da bi se obezbedila finija kontrola, potrebno je da se sigurnost implementira na aplikacionom nivou, što zahteva primenu sigurnosnih standarda.

Sigurnost kod Web servisa se može primeniti pomoću:
- poverljivosti i integriteta,
- autentifikacije i autorizacije,
- dostupnosti.

Poverljivost i integritet

Upravljanje poverljivošću i integritetom predstavlja proces koji obezbeđuje da će sadržaj poruke biti zaštićen od neovlašćenog pristupa, kao i da informacije prilikom transfera neće biti promenjene. Ovaj proces obično zahteva neki oblik kriptovanja. Korišćenje SSL protokola obezbeđuje zaštitu poruke tokom transporta, ali da bi se kriptovao deo ili cela poruka, potrebno je koristiti standarde za XML enkripciju.

Enkripcija koristi niz karaktera - ključ, da bi uz pomoć matematičkih algoritama šifrovala i dešifrovala podatke. Jedan oblik enkripcije se naziva simetrična enkripcija, kod kojeg se poruka i šifruje i dešifruje korišćenjem istog ključa. Drugi oblik enkripcije se naziva asimetrična enkripcija. Kod ovog oblika se poruka šifruje i dešifruje korišćenjem dva ključa, jednim koji je javno dostupan, i drugim koji je privatan. Ako je poruka šifrovana korišćenjem javnog ključa, može biti dešifrovana samo korišćenjem privatnog ključa. I obrnuto, ako je poruka šifrovana korišćenjem privatnog ključa, može biti dešifrovana samo korišćenjem javnog ključa.

Autentifikacija i autorizacija

Autentifikacija predstavlja proces provere identiteta korisnika ili aplikacije. Autorizacija određuje da li već provereni korisnik ili aplikacija ima dozvolu da izvrši određenu akciju. Autentifikacija zahteva nekakav dokaz identiteta, koji se naziva sigurnosni token, koji može biti kombinacija korisnik/lozinka, Kerberos tiket, ili neka vrsta sertifikata.

Dostupnost

Dostupnost obezbeđuje da informacije budu raspoložive tokom vremena onima kojima su potrebne. Upravljanje dostupnošću obično znači upravljanje situacijama kada Web servis nije raspoloživ, što uobičajeno obuhvata upravljanje greškama i stavljanje poziva u redove čekanja za kasnije procesiranje.

3.6 Kada treba koristiti Web servise?

Web servisi su korisni i prilagodljivi, ali svakako imaju svoja ograničenja. Preporuka je da se koriste u sledećim situacijama:

- Web servisi su odlični za povezivanje različitih sistema. Na primer, Web servisi se mogu koristiti da povežu više platformi, kao što su Windows, Macintosh, Linux, UNIX, AS/400, kao i mainframe sistemi. Web servisi se takođe mogu koristiti za povezivanje različitih programskih jezika, kao što su Visual Basic, Java, C++, RPG, COBOL, Perl.
- Web servisi se dobro ponašaju u nekontrolisanim uslovima. Na primer, ako se komunicira sa različitim korisnicima, i naročito ako ne postoji kontrola nad korisničkim sistemima, Web servisi se mogu koristiti kao fleksibilno i prilagodljivo rešenje.
- Web servisi su nenadmašni u podršci višekanalnim klijentima. Na primer, Web servis se može koristiti za isporučivanje istih podataka bogatim klijentima, Web čitačima, mobilnim uređajima.
- Web servisi se dobro ponašaju u dinamičkim uslovima. Ukoliko sistem treba da se prilagodi promenljivim uslovima, da odgovori na potrebu da se doda ili promeni deo sistema, Web servisi predstavljaju odličan izbor.
- Web servisi su dobri u prikupljanju informacija. Web servisi se mogu koristiti da povuku informacije zajedno sa različitih izvora za portal, aplikacije vezane za kupce, i druge aplikacije za različite vrste integracije.
- Web servisi smanjuju ponavljanje podataka. Ako postoji više aplikacija koje u osnovi obavljaju isti posao, Web servisi se mogu koristiti za centralizaciju te funkcionalnosti u jedan skup deljivih servisa.

3.7 Kada ne treba koristiti Web servise?

Bez obzira što su Web servisi korisni i prilagodljivi, ne treba da se koriste kao rešenje za sve situacije.

- XML je veoma prilagodljiv, ali ne predstavlja najkompaktniji i najefikasniji mehanizam za transfer podataka. SOAP poruke su mnogo veće u odnosu na binarne poruke kreirane od strane RPC, RMI, COBRA ili DCOM. Takođe, obrada XML poruke traje duže u odnosu na obradu binarne poruke. Razlika u performansama postaje uočljivija sa povećanjem veličine i složenosti poruke.

- SOAP ne treba da bude potpuna zamena za tradicionalne tehnologije, kao što su RPC, RMI, COBRA ili DCOM. Ove tehnologije još uvek imaju veoma važnu ulogu u razvoju aplikacija. One su dizajnirane da obezbede mehanizam sa veoma visokim performansama za komunikaciju sa različitim komponentama u okviru homogenog aplikacionog sistema ili servisa.

- Web servis ne obezbeđuje razvojni komponentni model, tako da se aplikacije ne mogu praviti korišćenjem Web servisa. Aplikacije se prave korišćenjem komponent tehnologije zasnovane na određenom programskom jeziku, kao što je Java ili Visual Basic.

3.8 Infrastruktura za razvoj Web servisa

Posle donošenja odluke o kreiranju Web servisa, potrebno je doneti odluku oko izbora softverskih proizvoda, koji će obezbediti implementaciju XML, SOAP, WSDL, UDDI i drugih Web servis tehnologija.

Većina prodavaca softvera ističe mogućnost da njihov proizvod kreira Web servis.

Osnovna infrastruktura za podršku Web servisima se sastoji od okruženja za komunikaciju između aplikacija preko SOAP poruka, kao i alata potrebnih za razvoj, isporuku i upravljanje Web servisima. [Manes]

Ova osnovna infrastruktura se naziva i Web servis platforma. Trenutno najpoznatije platforme su Microsoft .NET Framework i Sun Microsystems Java 2 Enterprise Edition (J2EE). I .NET i Java predstavljaju odlične platforme za razvoj Web servisa. .NET je jednostavniji za korišćenje, ali Java pruža više opcija i mogućnosti. Kao i u bilo kojoj situaciji, izbor platforme se bazira na zahtevima projekta. U narednoj tabeli su date ključne razlike između .NET i Jave.

Tabela 2. Ključne razlike između .NET i Java

	.NET	**Java**
Kompajliranje u	IL	bytecode
Izvršavanje	CLR	JVE
Jezici	Više jezika	Java
Proizvođač	Microsoft	Više proizvođača
Platforma	Windows	Bilo koja
XML/SOAP podrška	Integrisana	Omogućena preko ekstenzija
Standardizacija	C# predstavlja ISO standard; .NET predstavlja više od C#	JCP je otvoren
Licenciranje	Neograničeno za C#	Ograničeno

Najveći deo kompanija koristi više programskih jezika na različitim platformama, i takođe, većina kompanija koristi više platformi za Web servise. To je ono što predstavlja najveću vrednost kod Web servisa - ne postoji zavisnost od jednog proizvoda ili od jednog proizvođača softvera.

4. BPMN

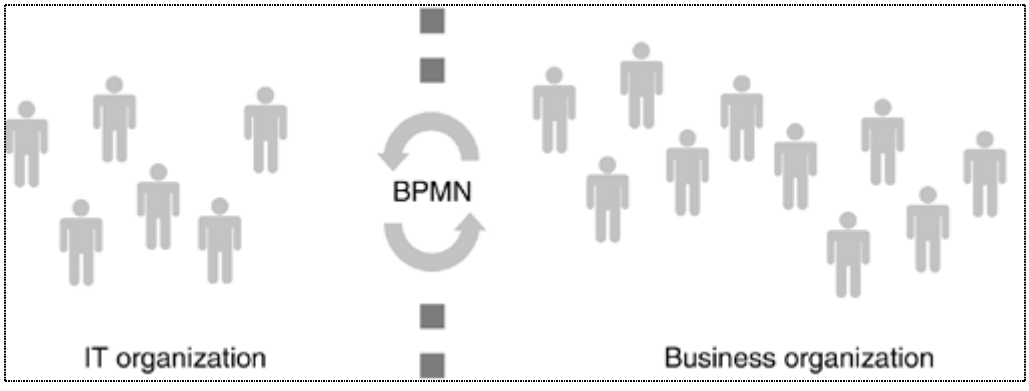
4.1 BPMI standardi

BPMI je neprofitna organizacija stvorena 2000. godine od strane kompanije Intalio, sa ciljem da kreira standarde za BPM. Članice ove organizacije su i BEA, Fujitsu, IBM, IDS Scheer, Pegasystems, PeopleSoft, SAP, SeeBeyond, Tibco, Virtria, WebMethods. I BPMI je članica nekoliko organizacija: W3C, OASIS, OMG, and the WfMC. Kroz članstva u ovim organizacijama, BPMI može da učestvuje u diskusijama vezano za skoro sve standarde koje se odnose na BPM.

BPMI je odgovorna za sledeće funkcionalne specifikacije:

- **Business Process Modeling Notation (BPMN)** - Predstavlja jezik napravljen sa ciljem da omogući grafičko kreiranje dijagrama toka poslovnog procesa od strane poslovnih analitičara i divelopera.
- **Business Process Modeling Language (BPML)** - Predstavlja XML jezik čija je svrha da kodira tok izvršavanja poslovnog procesa u formu odgovarajuću za interpretaciju od strane mašine za izvršavanje procesa. Verziju 1.0 je napisao Assaf Arkin iz kompanije Intalio novembra 2002. godine. Prema priznanju BPMI organizacije, BPML je izgubio bitku u odnosu na BPEL jezik, koji se uglavnom koristi kao XML jezik za izvršavanje procesa.
- **Business Process Query Language (BPQL)** - Cilj kreiranja jezika je da bude osnova za praćenje i nadgledanje poslovnih aktivnosti (Business Activity Monitoring - BAM). Nije još publikovan.
- **Business Process Semantic Model (BPSM)** - Treba da predstavlja zajednički model za sve modele poslovnih procesa. Nije još publikovan.
- **Business Process Extension Layers (BPXL)** - Predstavlja standardni skup proširenja za transakcije, poslovna pravila, upravljanje zadacima i ljudsku interakciju. Nije još publikovan.

4.2 Osnove BPMN



slika 23: BPMN je pozicioniran kao interfejs između poslovne organizacije i IT organizacije

Stephen White iz IBM kompanije je napisao verziju 1.0 BPMN specifikacije 2004. godine.

BPMN je razvijen je sa ciljem da obezbedi notaciju za definisanje poslovnih procesa, lako razumljivu od strane svih poslovnih korisnika, od poslovnih analitičara koji treba da skiciraju procese, do tehničkih korisnika koji su odgovorni za implementaciju tehnologije pomoću koje se izvršavaju procesi, i na kraju do poslovnih ljudi, koji će upravljati i nadgledati poslovne procese. BPMN omogućava da se prevaziđe praznina između dizajniranja poslovnog procesa i njegove implementacije. BPMN predstavlja intuitivnu notaciju, koja se lako usvaja i koristi.

BPMN se definiše preko Business Process Diagram (BPD), koji je baziran na tehnici blok-dijagrama i koji je prilagođen za kreiranje grafičkih modela operacija poslovnih procesa. Business Process Model (BPM) predstavlja mrežu grafičkih objekata, odnosno aktivnosti i kontrola kojima se definiše njihov redosled izvršavanja.

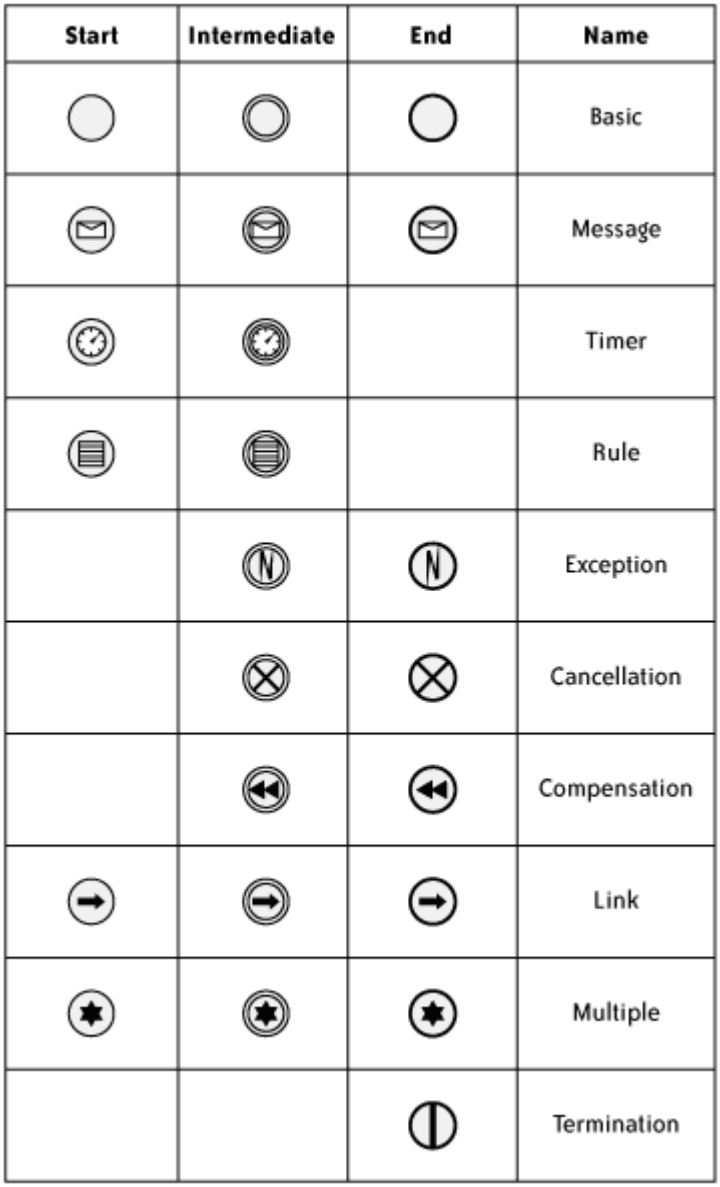
BPMN dijagram je sastavljen od skupa grafičkih elemenata. Elementi omogućavaju lak razvoj jednostavnih dijagrama, različiti su jedan u odnosu na drugi i izgledaju blisko većini poslovnih analitičara. Na primer, aktivnosti su predstavljene kao pravougaonici, a odluke kao rombovi.

Četiri osnovne kategorije BPD elemenata su:
- objekti toka,
- objekti veza,
- swimlines,
- artifakti.

4.2.1 Objekti toka

Objekti toka predstavljaju osnovne grafičke elemente za definisanje ponašanja poslovnih procesa. Objekti toka su Event, Activity i Gateway elementi.

Event element se prikazuje pomoću kruga i predstavlja nešto što se dešava tokom izvršavanja poslovnog procesa. Event element utiče na izvršavanje toka procesa i obično ima uzrok (okidač) i posledicu (rezultat). Event elementi su kategorizovani po fazi u kojoj se dešavaju u procesu (Start, Intermediate i End) i po tipu (Basic, Message, Timer, Rule, Exception, Cancellation, Compensation, Link, Multiple, Termination).

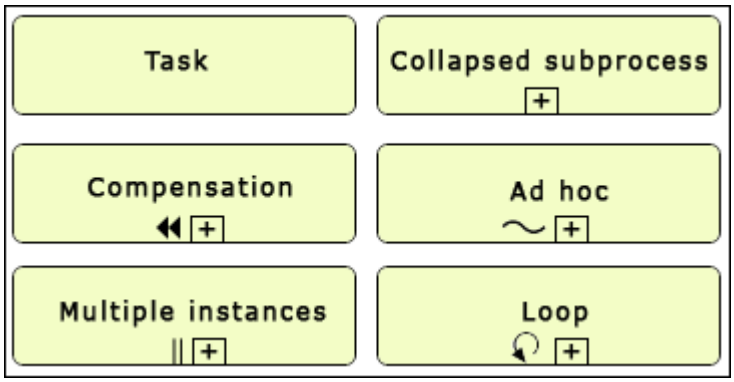


slika 24: Prikaz kompletnog skupa Event elemenata

Tabela: Opis BPMN Event elemenata			
Type	**Start**	**Intermediate**	**End**
Basic	Nosilac položaja ili početak podprocesa.	Nosilac položaja	Nosilac položaja ili kraj podprocesa.
Message	Proces se startuje po prijemu poruke. (na primer, poziv Web servis metode implementirane u procesu).	Proces čeka poruku (na primer, čeka odgovor na poslati zahtev).	Poruka koja se šalje određenom procesu (na primer, za poziv Web servisa).
Timer	Startni događaj definiše vreme kada će biti okinut (na primer, svake nedelje	Tačka kada je došlo definisano vreme.	

Tabela: Opis BPMN Event elemenata			
Type	**Start**	**Intermediate**	**End**
	u podne).		
Rule	Uslov, definisan u okviru procesa, je ispunjen (na primer, proces se startuje kada cena dostigne određeni nivo).	Uslov je ispunjen. Koristi se samo za upravljanje izuzecima.	
Exception		Podiže ili hvata grešku.	Generiše grešku.
Cancellation		Zaustavlja izvršavanje aktivnosti.	Prekida transakciju.
Compensation		Okida i izvršava akciju kompenzacije.	Izvršava akciju kompenzacije.
Link	Link startni događaj povezuje se na link završni događaj od procesa na istom nivou.	Veza od ili do druge aktivnosti.	Povezuje se na link start događaj od procesa na istom nivou.
Multiple	Dva ili više okidača mogu pokrenutu proces, ako se desi bilo koji od njih, proces se startuje. Okidači mogu biti poruke, timer, pravilo, ili tip.	Dva ili više okidača mogu da nastave da čekaju proces, ako se bilo koji od njih izvrši, proces se nastavlja.	Kada se proces završi, očekuje se nekoliko rezultata (na primer, potrebno je poslati nekoliko poruka).
Termination			Prekida sve aktivnosti u procesu. Izvršava kompenzaciju ili upravlja izuzecima.

Activity element se predstavlja pomoću pravougaonika sa zaobljenim uglovima i on je opšti pojam za posao koji se obavlja. Activity element može biti prost ili složen. Složena aktivnost, koja se naziva i proces, takođe može imati svoj skup prostih ili složenih aktivnosti, kao i druge BPMN elemente. Tipovi ovog elementa su: Task i Sub-Process (predstavljen je kao pravougaonik sa zaobljenim uglovima sa znakom plus u centru donjeg dela). Znak plus predstavlja prikaz složenog procesa, detaljni o njemu se prikazuju na odvojenom dijagramu. Proces može biti označen sa simbolima koji predstavljaju kompenzaciju, višestruko pojavljivanje, petlje i ad hoc obradu. Kompenzacija predstavlja aktivnost koja poseduje logiku za vraćanje u prethodno stanje. Ad hoc proces sadrži skup aktivnosti koje se mogu izvršiti u bilo kom redosledu. Petlja aktivnost može biti konfigurisana kao standardna petlja (while ili until petlja), ili da podržava višestuke instance (foreach petlja).



slika 25: BPMN Activity elementi

Tipovi Task elementa:

Service	Poziv Web servisa
Receive	Čekanje poruke (alternativa za izgradnju događaja)
Send	Slanje poruke
User, Manual	Posao se izvršava od strane čoveka-učesnika
Script	Logika sadržana u programskom ili skripting jeziku (na primer, izvršavanje dela Java koda)
Reference	Koristi definiciju drugog posla u okviru procesa; ne duplira definiciju, već je deli

Gateway element predstavljaju glavni način za kontrolu tokova u okviru BPMN dijagrama. Predstavlja se kao romb i koristi se za označavanje uslovnih grananja ili spajanja puteva u okviru procesa.

Symbol	Name
	Exclusive OR
	Exclusive OR
	Exclusive OR (Event-based)
	Exclusive OR
	Complex
	Parallel

slika 26: BPMN Gateway elementi

Gateway Exclusive OR element, koji je baziran na podacima, koristi kontrolne strukture *if-then-else* i *switch* sa međusobno isključivim slučajevima. U situaciji kada se vrši grananje u okviru procesa, tačno jedan uslov mora biti ispunjen.

Drugi tip Gateway Exclusive OR elementa, koji je baziran na događaju, koristi *pick* kontrolnu strukturu. U situaciji kada se vrši grananje u okviru procesa, svaka grana vodi do jednog čvora sa događajem. Kontrola prolazi kroz granu gde se okida prvi događaj, a sve ostale grane se ignorišu.

Gateway Inclusive OR element je sličan gateway Exclusive OR elementu, koji je baziran na podacima, izuzev što dozvoljava prolazak kroz svaku granu *switch* kontrolne strukture, čiji uslov je ispunjen.

Gateway Complex element se uglavnom koristi kod spajanja puteva u okviru procesa, kada se proverava definisani izraz da bi se odredila putanja kojom će se nastaviti izvršavanje procesa.

Gateway Parallel element, koristi *flow* kontrolnu strukturu, i kod grananja u okviru procesa, dozvoljava prolazak kroz svaku od definisanih putanja. Kod spajanja, blokira nastavak izvršavanja procesa, sve dok se ne izvrše sve definisane grane.

4.2.2 Objekti veza

Objekti toka se povezuju međusobno korišćenjem objekata veza. Postoje tri osnovna tipa elemenata koji obezbeđuju ovu funkciju.

Sequence Flow		Element je predstavljen punom linijom sa punom strelicom i koristi se da prikaže redosled izvršavanja aktivnosti u okviru procesa.
Message Flow		Element je predstavljen isprekidanom linijom sa nepopunjenom strelicom i koristi se da prikaže tok poruka između dva odvojena učesnika u procesu, koji primaju i šalju poruke.
Association		Element je predstavljen linijom od tačaka sa otvorenom strelicom i koristi se da poveže podatke, tekst i druge artifakte sa objektima toka, odnosno da prikaže ulaze i izlaze iz aktivnosti.

4.2.3 Swimlanes

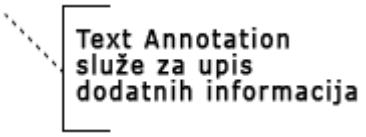
Swimlines čine Pool elementi i, u okviru Pool elementa, Lane elementi. Pool prikazuje aktivnosti jednog učesnika, najčešće preduzeća. Lane element u okviru Pool elementa prikazuje sastavne delove jednog učesnika, najčešće odeljenja preduzeća.

Swimlines		
Pool		Pool element predstavlja učesnika u procesu. Može biti određeni poslovni entitet (kompanija) ili se može nalaziti u nekoj poslovnoj ulozi (kupac, proizvođač). Takođe, ima ulogu grafičkog kontejnera za odvajanje skupa aktivnosti od drugih Pool

		elemenata, obično u kontekstu B2B situacija.
Lane		Element omogućava podelu unutar Pool elementa i proširuje ga ili vertikalno ili horizontalno, sa ciljem da organizuje i kategorizuje aktivnosti.

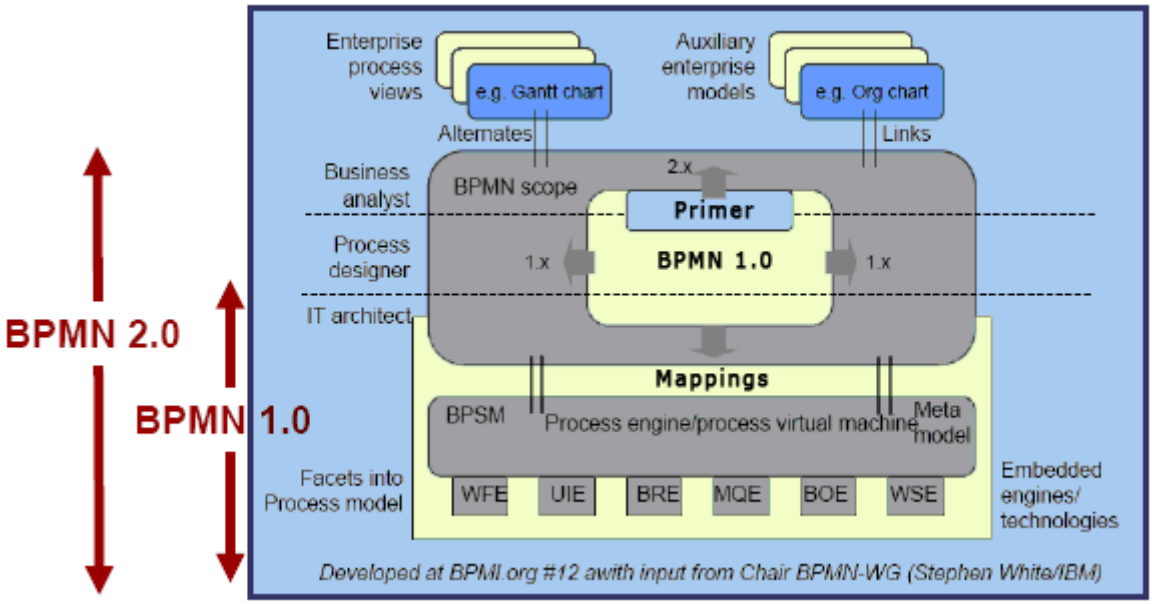
4.2.4 Artifakti

BPMN je dizajniran da dozvoli onima koji ga koriste (modelari, alati za modeliranje) određenu fleksibilnost u proširenju osnovne notacije, kao i u obezbeđenju mogućnosti da se doda kontekst, odgovarajući za specifičnu situaciju koja se modeluje. Bilo koji broj artifakta može se dodati u dijagram.

Data Object		Predstavljaju mehanizam za prikaz podataka, koji su ulaz ili izlaz iz aktivnosti. Povezani su sa aktivnostima preko Associations elemenata.
Group		Koristi se za dokumentaciju ili u svrhu analize. Ne utiče na redosled toka aktivnosti.
Annotation		Obezbeđuje dodatne tekstualne informacije o BPMN dijagramu.

BPMN verzija 2.0

BPMI organizacija, odgovorna za kreiranje BPMN i drugih standarda vezanih za modeliranje poslovnih procesa, priznaje da je BPMN 1.0 samo prvi korak u razvoju ove notacije. Trenutno se razvija nova verzija BPMN 2.0, koja će sadržati poboljšanja i zadovoljiti potrebe organizacija i analiza lanaca vrednosti. Nova verzija treba da sadrži nove definicije vezane za meta model i formate za razmenu poruka.



slika 27: Funkcionalni obim BPMN 1.0 i BPMN 2.0

5. BPEL

Kompozicija servisa u poslovne procese zahteva definisanje aktivnosti i poruka između uključenih servisa. WSDL obezbeđuje osnovni tehnički opis i specifikacije za poruke koje se razmenjuju. Međutim, opis koji obezbeđuje WSDL ne ide dalje od opisa jednostavne interakcije između klijenta i servisa. Ove interakcije mogu biti sinhrone ili asinhrone, kao i sa ili bez stanja. Ove relacije su neodgovarajuće za izražavanje, modeliranje i opis složenih kompozicija više servisa. Zbog postojećih ograničenja WSDL jezika, neophodan je mehanizam za opis kompozicije servisa u kompleksnije procese. [Juric]

Kompozicija servisa u poslovne procese može biti realizovana korišćenjem nekog od poznatih programskih jezika (Java, C#, ...). Međutim, korišćenje programskih jezika za kompoziciju servisa najčešće ima za posledicu kreiranje nefleksibilnih rešenja, najviše zbog nejasne granice između toka procesa i poslovne logike, koji ne smeju biti čvrsto povezani. [Juric]

Opšta potreba za kreiranjem rešenja za automatizaciju izvršavanja poslovnih procesa zahteva i standarde i specijalizovani jezik za kompoziciju servisa u poslovne procese, koji će obezbediti mogućnost izražavanja poslovnih procesa na standardizovan način korišćenjem opšte prihvaćenog jezika. BPEL predstavlja takav jezik, koji je veoma brzo postao dominantan standard. Glavni cilj BPEL jezika je da standardizuje proces automatizacije između servisa. [Juric]

Business Process Execution Language (BPEL) predstavlja na XML zasnovan jezik za definiciju i opis poslovnih procesa. BPEL jezik je prvobitno kreiran od strane IBM, Microsoft i BEA. Sada je u nadležnosti neprofitnog konzorcijuma OASIS (Organization for the Advancement of Structured Information Standards), koji razvija, održava i promoviše standarde za elektronsko poslovanje, uključujući ebXML, SGML, UDDI, PKI i BPEL.

BPEL je nastao od WSFL i XLANG jezika sa ciljem da omogući "programiranje na veliko".

Korišćenjem BPEL jezika se mogu definisati jednostavni i kompleksni poslovni procesi. Uz neka proširenja, BPEL je sličan tradicionalnim programskim jezicima, jer ima elemente tipa petlje, grananja, variable, dodeljivanja, koji dozvoljavaju definisanje poslovnih procesa na algoritamski način. BPEL je specijalizovani jezik fokusiran na definisanje poslovnih procesa.

Najvažniji elementi BPEL jezika se odnose na pozivanje Web servisa. BPEL omogućava da se poziv Web servisa obavi na jednostavan način, kako sinhrono, tako i asinhrono. Operacije se mogu pozivati i sekvencijalno i paralelno. Takođe omogućava opis povratnih funkcija, upravljanje greškama, obezbeđuje podršku za procese koji se dugo izvršavaju, kao i kompenzaciju, što omogućava da se opovrgne posao koji je odradio proces, koji se nije završio uspešno. BPEL jezik omogućava [Juric]:

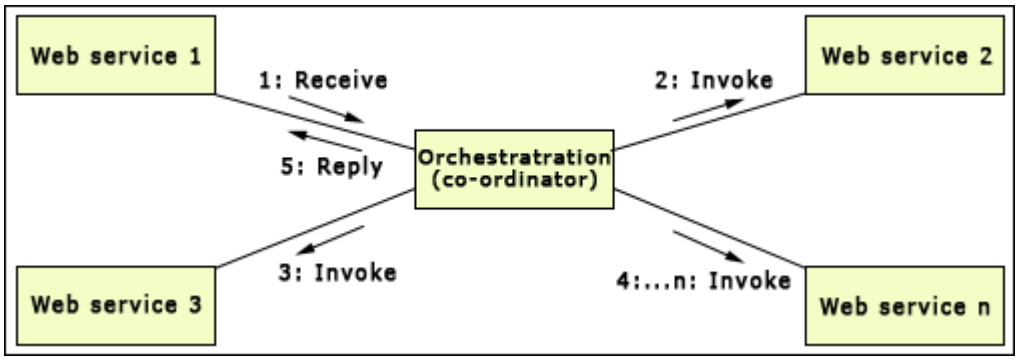
- opis logike poslovnog procesa kroz kompoziciju servisa;
- sastavljanje većih poslovnih procesa od manjih procesa i servisa;
- upravljanje sinhronim i asinhronim pozivanjem operacija servisa, kao i upravljanje povratnim funkcijama koje se izvršavaju kasnije;
- poziv operacija servisa sekvencijalno ili paralelno;
- selektivno kompenzira neuspešno završene aktivnosti;
- održavanje višestruke dugotrajne transakcione aktivnosti;
- nastavak prekinutih ili neuspešno završenih aktivnosti, što minimizuje potrebu za njihovim ponovnim izvršavanjem;

- rutiranje dolazećih poruka u odgovarajuće procese i aktivnosti;
- uzajamno povezivanje zahteva u okviru ili van poslovnog procesa;
- planiranje izvršavanja aktivnosti zasnovano na vremenu izvršavanja i definiše njihov redosled izvršavanja;
- izvršavanje aktivnosti paralelno i definisanje kako će se paralelni tokovi sinhronizovati u odnosu na uslove sinhronizacije;
- struktuiranje poslovnih procesa u nekoliko oblasti;
- upravljanje događajima zasnovanim na porukama i vremenu.

5.1 Orkestracija i koreografija

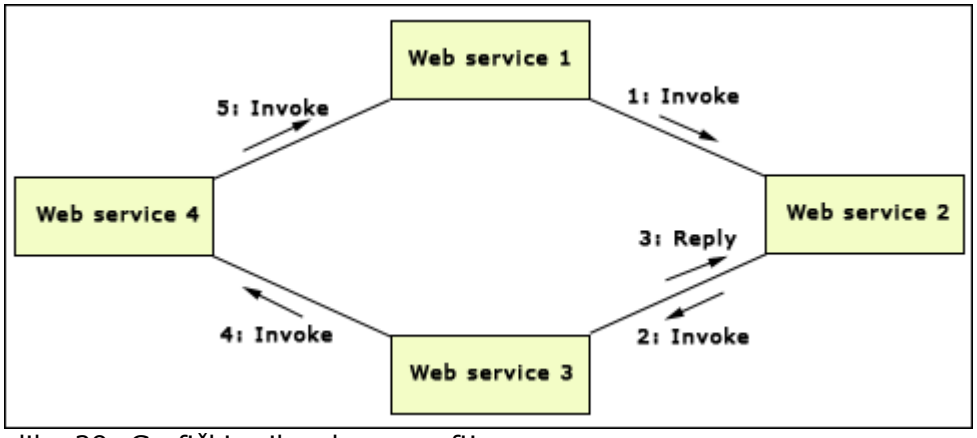
U zavisnosti od zahteva, kompozicija servisa može upućivati na privatne ili javne procese, zašta se koriste termini orkestracija i koreografija.

U orkestraciji, centralni proces ima kontrolu nad uključenim Web servisima i koordiniše izvršavanje različitih operacija Web servisa. Ovo se izvršava prema zahtevima orkestracije. Uključeni Web servisi ne znaju da su uključeni u kompoziciju i da su deo višeg poslovnog procesa. Jedino centralni koordinator orkestracije zna ovo, tako da je orkestracija centralizovana sa eksplicitnim definicijama operacija i redosledom pozivanja Web servisa. Orkestracija se uobičajeno koristi u privatnim poslovnim procesima.



slika 28: Grafički prikaz orkestracije

Koreografija ne zavisi od centralnog koordinatora. Svaki Web servis uključen u koreografiju tačno zna kada treba da izvrši svoje operacije i sa kim treba da sarađuje. Koreografija predstavlja zajednički napor usmeren na razmenu poruka u javnim poslovnim procesima.



slika 29: Grafički prikaz koreografije

Posmatrajući iz perspektive kompozicije Web servisa za izvršavanje poslovnog procesa, orkestracija ima prednost u odnosu na koreografiju. Orkestracija je fleksibilnija, mada granica između orkestracije i koreografije nije uvek uočljiva. Orkestracija ima sledeće prednosti [Juric]:

- tačno se zna ko je odgovoran za izvršenje celog poslovnog procesa;
- mogućnost dodavanja Web servisa, bez njihovog znanja o tome da su deo poslovnog procesa;
- obezbeđivanje alternativnih scenarija u slučaju pojave greške.

5.2 Izvršni i apstraktni procesi

BPEL obezbeđuje podršku za orkestraciju i koreografiju kroz izvršne i apstraktne poslovne procese.

Izvršni poslovni procesi su procesi koji su sastavljeni od skupa postojećih servisa i specificiraju tačan algoritam aktivnosti i ulaznih i izlaznih poruka. Ovi procesi se izvršavaju od strane BPEL mašina. Izvršni procesi su veoma važni, jer su oni direktan odgovor na problem automatizacije poslovnih procesa. Sa BPEL izvršnim procesima se može specificirati tačan algoritam kompozicije servisa na relativno jednostavan i otvoren način, koji se može izvršiti na odgovarajućoj BPEL mašini. Izvršni procesi popunjavaju prazninu između specifikacija poslovnih procesa i koda odgovornog za njihovo izvršavanje.

Kada se definiše BPEL izvršni proces, u stvari se definiše novi Web servis, koji je kompozicija postojećih servisa. Interfejs novog BPEL kompozitnog servisa koristi skup "port types", preko kojih obezbeđuje izvršavanje operacija kao kod bilo kog drugog servisa.

Apstraktni poslovni procesi nisu izvršni. Uglavnom se koriste da bi:

- opisali ponašanje servisa bez tačnog saznanja u kom će poslovnom procesu učestvovati;
- definisali protokole saradnje između više učesnika i precizno opisali eksterno ponašanje svakog učesnika.

Apstraktni procesi se retko koriste. Najčešći scenario je da se koriste kao šablon za definisanje izvršnih procesa. Mogu se koristiti da zamene skup pravila uobičajeno izraženih govornim jezikom, koji je često dvosmislen.

Iz BPEL ugla, poslovni proces predstavlja kolekciju koordinisanih poziva Web servisa, kao i povezanih aktivnosti koje proizvode rezultat, u okviru jedne ili više organizacija.

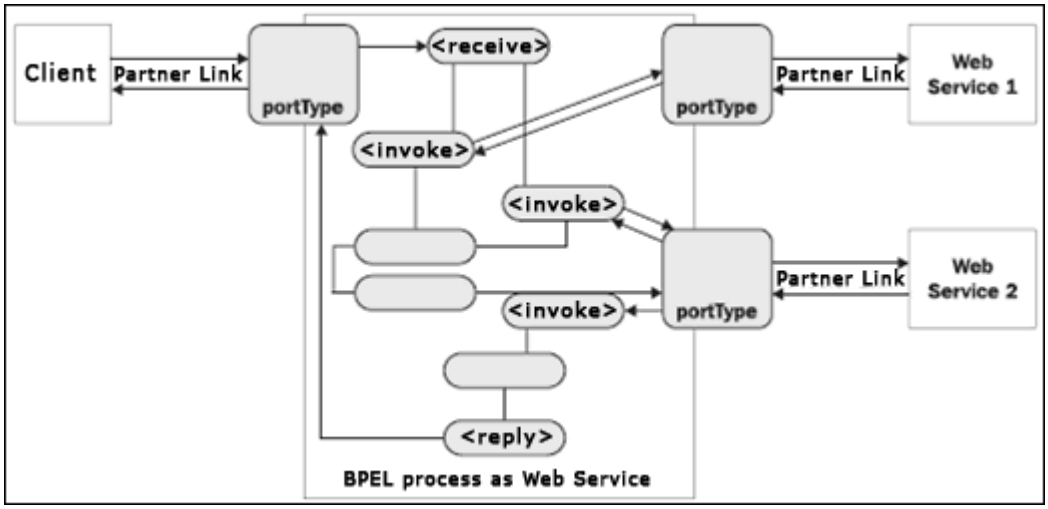
BPEL proces se sastoji od više koraka. Svaki korak se naziva aktivnost. BPEL podržava primitivne i strukturne aktivnosti.

Primitivne aktivnosti predstavljaju osnovne elemente i koriste se za zajedničke poslove, kao što su:

- pozivanje Web servisa, korišćenjem <invoke>
- čekanje na zahtev, korišćenjem <receive>
- generisanje odgovora za sinhrone operacije, korišćenjem <reply>
- manipulisanje sa varijablama podataka, korišćenjem <assign>
- ukazivanje na greške i izuzetke, korišćenjem <throw>

Primitivne aktivnosti možemo kombinovati u mnogo kompleksnije algoritme, koje definišu korake poslovnog procesa. Da bi kombinovao primitivne aktivnosti, BPEL podržava nekoliko strukturnih aktivnosti. Najvažnije su:

- <sequence> za definisanje skupa aktivnosti, koje će biti pozivane u određenom redosledu;
- <flow> za definisanje skupa aktivnosti, koje će biti pozivane paralelno;
- <switch> za implementaciju grananja;
- <while> za definisanje petlji.



slika 30: BPEL proces kao Web servis [Juric]

5.3 Struktura BPEL jezika

BPEL definiše model i gramatiku za opis ponašanja poslovnog procesa, zasnovanog na međusobnom uticaju procesa i njegovih partnera. Interakcija sa svakim partnerom se izvršava korišćenjem interfejsa Web servisa. Struktura relacija na nivou interfejsa je prezentovana preko elementa *partner link*. BPEL proces definiše i opisuje koordinaciju servisa sa partnerima radi postizanja poslovnog cilja, kao i stanje i logiku neophodnu za koordinaciju. Takođe, BPEL uspostavlja sistematski mehanizam za ponašanje u slučaju dešavanja poslovnih izuzetaka kao i obrade grešaka. BPEL predstavlja mehanizam za definisanje kako pojedinačne ili kompozitne aktivnosti u okviru procesa mogu biti kompenzovane u slučaju da se dese izuzeci ili promena zahteva partnera. [BPEL4WS 1.1]

BPEL je na XML-u baziran jezik, sagrađen na osnovu specifikacija za Web servise. WSDL poruke i XSD Schema definicije tipova obezbeđuju model podataka korišćen od strane BPEL procesa. XPath obezbeđuje podršku za manipulaciju podacima. Svi eksterni resursi i

partneri su prezentovani kao WSDL servisi. BPEL obezbeđuje proširivost za prihvatanje budućih verzija pomenutih standarda. [BPEL4WS 1.1]

```
<process name="ncname" targetNamespace="uri"
queryLanguage="anyURI"
expressionLanguage="anyURI"
suppressJoinFailure="yes|no"
enableInstanceCompensation="yes|no"
abstractProcess="yes|no"
xmlns="http://schemas.xmlsoap.org/ws/2003/03/business-process/">

<partnerLinks>
<!-- Note: At least one role must be specified. -->
<partnerLink name="ncname" partnerLinkType="qname"
myRole="ncname"? partnerRole="ncname">
</partnerLink>
</partnerLinks>

<partners>
<partner name="ncname">
<partnerLink name="ncname"/>
</partner>
</partners>

<variables>
<variable name="ncname" messageType="qname"
type="qname" element="qname" />
</variables>

<correlationSets>
<correlationSet name="ncname" properties="qname-list" />
</correlationSets>

<faultHandlers>
<!-- Note: There must be at least one fault handler or default. -->
<catch faultName="qname" faultVariable="ncname">
    activity
</catch>
<catchAll>
    activity
</catchAll>
</faultHandlers>

<compensationHandler>
    activity
</compensationHandler>

<eventHandlers>
<!-- Note: There must be at least one onMessage or onAlarm handler. -->
<onMessage partnerLink="ncname" portType="qname"
operation="ncname" variable="ncname">
<correlations>
<correlation set="ncname" initiate="yes|no" >
<correlations>
activity
</onMessage>
```

```
<onAlarm for="duration-expr"  until="deadline-expr">
activity
</onAlarm>
</eventHandlers>

activity

</process>
```

Opis atributa u okviru korenog taga:

- **queryLanguage**. Ovaj atribut specificira XML jezik za upite korišćen za selekciju čvorova za dodeljivanje, definiciju osobina, kao i za druge namene. Podrazumevana vrednost za ovaj atribut je XPath 1.0.
- **expressionLanguage**. Atrubut predstavlja jezik za kreiranje izraza korišćenih u okviru procesa. Podrazumevana vrednost za ovaj atribut je XPath 1.0.
- **suppressJoinFailure**. Atribut određuje da li će joinFailure greška biti zamenjena za sve aktivnosti u procesu. Podrazumevana vrednost za ovaj atribut je "no".
- **enableInstanceCompensation**. Atribut određuje da li instanca procesa u celini može biti kompenzirana sa značenjem specifičnim za određenu platformu. Podrazumevana vrednost za ovaj atribut je "no".
- **abstractProcess**. Atribut određuje da li će proces biti definisan kao apstraktni, a ne izvršni. Podrazumevana vrednost za ovaj atribut je "no".

Aktivnosti

Aktivnost može imati neku od sledećih vrednosti:

- <receive>
- <reply>
- <invoke>
- <assign>
- <throw>
- <terminate>
- <wait>
- <empty>
- <sequence>
- <switch>
- <while>
- <pick>
- <flow>
- <scope>
- <compensate>

<receive> tag obezbeđuje da poslovni proces sačeka dolazak odgovarajuće poruke.

```
<receive partnerLink="ncname" portType="qname" operation="ncname"
variable="ncname" createInstance="yes|no" standard-attributes>
standard-elements
<correlations>
<correlation set="ncname" initiate="yes|no" />
</correlations>
</receive>
```

<reply> tag dozvoljava poslovnom procesu da pošalje poruku kao odgovor na poruku dobijenu kroz <receive>. Kombinacija <receive> i <reply> kreira request-response WSDL portType operaciju za proces.

```
<reply partnerLink="ncname" portType="qname" operation="ncname"
variable="ncname" faultName="qname" standard-attributes>
standard-elements
<correlations>
<correlation set="ncname" initiate="yes|no" />
</correlations>
</reply>
```

<invoke> tag dozvoljava poslovnom procesu da pozove u jednom ili oba pravca (request-response) operaciju na portType.

```
<invoke partnerLink="ncname" portType="qname" operation="ncname"
inputVariable="ncname" outputVariable="ncname" standard-attributes>
standard-elements
<correlations>
<correlation set="ncname" initiate="yes|no" pattern="in|out|out-in" />
</correlations>
<catch faultName="qname" faultVariable="ncname">
activity
</catch>
<catchAll>
activity
</catchAll>
<compensationHandler>
activity
</compensationHandler>
</invoke>
```

<assign> tag može biti korišćen za promenu vrednosti varijabli. Može sadržati bilo koji broj osnovnih dodeljivanja. Sintaksa za aktivnost dodeljivanja je:

```
<assign standard-attributes>
standard-elements
<copy>
from-spec
to-spec
</copy>
</assign>
```

<throw> element generiše grešku u okviru poslovnog procesa.

```
<throw faultName="qname" faultVariable="ncname" standard-attributes>
standard-elements
</throw>
```

<wait> element dozvoljava da se čeka određeno vreme. Tačno jedan kriterijum za prestanak čekanja mora biti specificiran.

```
<wait (for="duration-expr" | until="deadline-expr") standard-attributes>
standard-elements
</wait>
```

<empty> element dozvoljava da se umetne "no-op" instrukcija u poslovni proces, što može biti korisno za sinhronizaciju konkurentnih aktivnosti.

```
<empty standard-attributes>
standard-elements
</empty>
```

<sequence> element obezbeđuje da se definiše kolekcija aktivnosti, koja će biti izvršavana sekvencijalno.

```
<sequence standard-attributes>
standard-elements
activity
</sequence>
```

<switch> element obezbeđuje odabir tačno jedne aktivnosti iz skupa ponuđenih.

```
<switch standard-attributes>
standard-elements
<case condition="bool-expr">
activity
</case>
<otherwise>
activity
</otherwise>
</switch>
```

<while> element dozvoljava ponavljanje aktivnosti, sve dok je zadovoljen neki kriterijum.

```
<while condition="bool-expr" standard-attributes>
standard-elements
activity
</while>
```

<pick> element dozvoljava blokiranje i čeka da stigne odgovarajuća poruka ili da istekne alarm.

```
<pick createInstance="yes|no" standard-attributes>
standard-elements
<onMessage partnerLink="ncname" portType="qname"
operation="ncname" variable="ncname" />
<correlations>
<correlation set="ncname" initiate="yes|no" >
</correlations>
activity
</onMessage>
<onAlarm (for="duration-expr" | until="deadline-expr")>
activity
</onAlarm>
</pick>
```

<flow> element omogućava da se specificira jedna ili više aktivnosti, koje će se izvršavati konkurentno.

```
<flow standard-attributes>
standard-elements
<links>
<link name="ncname">
</links>
activity
</flow>
```

<scope> element omogućava definisanje ugnježdene aktivnosti sa svojim sopstvenim varijablama, obradama grešaka.

```
<scope variableAccessSerializable="yes|no" standard-attributes>
standard-elements
<variables>
... see above under <process> for syntax ...
</variables>
<correlationSets>
... see above under <process> for syntax ...
</correlationSets>
<faultHandlers>
... see above under <process> for syntax ...
</faultHandlers>
<compensationHandler>
... see above under <process> for syntax ...
</compensationHandler>
<eventHandlers>
...
</eventHandlers>
activity
</scope>
```

<compensate> element se koristi da bi pozvao kompenzaciju. Ovaj element može biti pozvan samo u okviru obrade grešaka ili u okviru obrade druge kompenzacije.

```
<compensate scope="ncname" standard-attributes>
standard-elements
</compensate>
```

<partnerLinkTypes>

<partnerLinkType> element opisuje interakciju između BPEL procesa i uključenih strana, koje čine Web servisi, koje BPEL proces poziva, i klijenta koji poziva BPEL proces.
Svaki <partnerLinkType> može imati jedan ili dva <role> elementa. Za sinhrone operacije, definiše se samo jedna uloga, odnosno jedan <role> element. Svaki <role> element specificira tačno jedan WSDL <portType> element.

```
<plnk:partnerLinkType name="ncname">
<plnk:role name="ncname">
<plnk:portType name="qname" />
</plnk:role>
<plnk:role name="ncname">
<plnk:portType name="qname" />
</plnk:role>
</plnk:partnerLinkType>
```

<partnerLinks>

Servisi sa kojima poslovni proces ima interakciju su u BPEL modelovani kao <partnerLinks> elementi. Svaki partner link sadrži atribut koji ukazuje na partnerLinkType element. Svaki partner link ima svoje ime, koje se koristi za sve interakcije servisa preko tog partner linka.

```
<partnerLinks>
<partnerLink name="ncname" partnerLinkType="qname"
myRole="ncname" partnerRole="ncname">
</partnerLink>
</partnerLinks>
```

<partner>

Partner element se koristi da bi reprezentovao sposobnosti koje se zahtevaju od biznis partnera. Partner je definisan kao podskup od partner linkova vezanih za proces. Definicije partnera su opcione i nije neophodno da pokrivaju sve partner linkove u okviru procesa.

```
<partner name="SellerShipper"
xmlns="http://schemas.xmlsoap.org/ws/2003/05/partner-link/">
<partnerLink name="Seller"/>
```

```
<partnerLink name="Shipper" />
</partner>
```

5.4 Životni ciklus poslovnog procesa

BPEL poslovni proces reprezentuje dugotrajnu interakciju, koja nadgleda stanje i koja ima početak, trajanje i kraj. Kreiranje instance BPEL procesa je uvek implicitno. Aktivnosti koje primaju poruke (<receive> i <pick> aktivnosti), mogu biti označene tako da izvršavanje neke od tih aktivnosti uzrokuje da se kreira novi poslovni proces. To se izvršava tako što se createInstance atribut od aktivnosti stavlja na vrednost "yes".

Da bi mogao da bude instanciran, svaki poslovni proces mora da ima barem jednu startnu aktivnost. To mora biti inicijalna aktivnost, u smislu da ne postoji osnovna aktivnost koja logički prethodi.

Poslovni proces može da čuva stanje i najčešće se veoma dugo izvršava. Za svaki poziv poslovnog procesa kreira se njegova instanca. Možemo razmišljati o poslovnom procesu kao o šablonu za kreiranje poslovnih instanci. To je veoma slično odnosu klasa-objekat, gde klasa predstavlja šablon za kreiranje objekta u vreme izvršavanja.

Poslovni proces se završava na jedan od sledećih načina:

- Kada se aktivnost koja definiše ponašanje procesa u potpunosti izvrši. U ovom slučaju, kraj poslovnog procesa se dešava normalno.
- Kada se desi greška tokom izvršavanja procesa, koja može biti obrađena ili ne. U ovom slučaju, kraj poslovnog procesa se dešava nenormalno.
- Kada se proces eksplicitno prekine korišćenjem terminate aktivnosti. U ovom slučaju, kraj poslovnog procesa se dešava nenormalno.
- Ako je compensation handler definisan za ceo poslovni proces, instanca poslovnog procesa može biti kompenzovana posle normalnog završetka, što se omogućava setovanjem enableInstanceCompensation atributa na "yes".

5.5 BPEL For People

Interakcija sa ljudima nije trenutno podržana BPEL jezikom, koji je primarno dizajniran da podrži automatizaciju poslovnih procesa zasnovanih na Web servisima. Međutim, u praksi, izvršavanje većine poslovnih procesa zahteva uzajamno dejstvo sa ljudima. Interakcija može biti veoma jednostavna, kao što je davanje nekog odobrenja, i može se protezati do veoma složenih scenarija, kada se, na primer, očekuje unos podataka od strane korisnika. [BPEL4People]

Da bi se podržao veliki broj situacija koje uključuje ljude u izvršavanje poslovnog procesa, neophodno je proširenje BPEL jezika.

BPEL4People je kreiran od strane kompanija IBM i SAP, i predstavlja nadogradnju na BPEL jezik, tako da se njegove karakteristike mogu kombinovati sa osnovnim karakteristikama BPEL jezika, kada god je to neophodno.

Odnos između ljudi i poslovnog procesa je složen i može se definisati kao:

- "opšte uloge ljudi" - predstavlja različite načine na koje ljudi imaju uzajamno dejstvo sa procesima;

- "veze sa ljudima" - predstavlja način na koji proces identifikuje ljude, koji sa njim komuniciraju.

Primeri opštih uloga predstavljaju:

- inicijatori procesa (osoba koja kreira instancu procesa);
- stakeholder (osoba koja može imati uticaj na napredovanje izvršavanja procesa, na primer, neko ko će zakačiti neki dokument, proslediti dokument, ili jednostavno posmatrati izvršavanje procesa);
- administrator poslovnog procesa (može da izvršava administrativne aktivnosti).

Ljudi mogu biti uključeni u poslovni proces preko specijalne vrste implementacije aktivnosti. Da bi se to omogućilo, potrebna je nova vrsta BPEL aktivnosti. Sa stanovišta BPEL poslovnog procesa, ljudska aktivnost je osnovna aktivnost, koja se ne implementira softverski, već se realizuje akcijom koju izvršava ljudsko biće. Izvršilac aktivnosti je definisan pomoću linka (people link). People link se koristi da reprezentuje različite grupe ljudi koji učestvuju u izvršavanju procesa, a za njihovu implementaciju se koriste poslovi. Posao predstavlja nedeljivu jedinicu, koju izvršava ljudsko biće. Posao definiše akciju koju korisnik mora za izvrši.

Ljudi učestvuju u izvršavanju poslovnih procesa tako što koriste korisnički interfejs. Kada se koriste ljudske aktivnosti, ulazni i izlazni podaci moraju biti predstavljeni na takav način da ih korisnik može interpretirati. Primeri predstavljanja podataka su: HTML obrazac, Excel radna sveska, GUI dijalog, SMS, govorna poruka.

Da bi se podržao veliki broj situacija koje uključuje ljude u izvršavanje poslovnog procesa, neophodno je proširenje BPEL jezika. Možda je najjednostavnije elemente već kreirane specifikacije BPEL4People, dodati u postojeću specifikaciju za BPEL jezik, čime bi neophodna interakcija sa ljudima u toku izvršavanja poslovnog procesa bila formalizovana.

5.6 Budućnost BPEL jezika

Opšte je poznata činjenica da, bez obzira na postojanje različitih nedostataka, neki standard ili specifikacija bude prihvaćena, ukoliko postoji dobar softver koji ga primenjuje, i ukoliko postoje knjige i članci koji ga detaljno opisuju i pojašnjavaju.

Softveri koji kreiraju i izvršavaju BPEL poslovne procese, moraju da zadovolje više zahteva:

- jednostavno korišćenje;
- jednostavno kreiranje tokova, kao i njihova promena i brisanje;
- jednostavno testiranje;
- mogućnost nadgledanja izvršavanja;
- sposobnost za simultano upravljanje velikim brojem tokova;
- jednostavna integracija sa Web servisima.

S obzirom da na tržištu softvera postoji više proizvoda, kreiranih od strane najvećih svetskih softverskih kompanija, koji zadovoljavaju napred navedene zahteve, budućnost BPEL izvršnog jezika je obezbeđena.

6. "Process Four" ili P4 uzori za opis ponašanja poslovnih procesa

Kao rešenja za zajedničke probleme se koriste dizajn uzori. Poznati su uzori za objektno-orijentisani dizajn, koji se nazivaju Gang of Four, ili skraćeno, GoF.

Za opis ponašanja poslovnih procesa, grupa koju su činili Wil van der Aalst, Arthur ter Hofstede, Bartek Kiepuszewski i Alistair Barros, je publikovala uzore za procese [P4]. Uzori su nazvani "Process Four" ili skraćeno, P4. P4 lista opisuje 20 uzora specifičnih za ponašanje procesa. Uzori se nalaze u rangu od veoma jednostavnih do veoma kompleksnih, i pokrivaju ponašanja koja mogu biti prepoznata kod većine modela poslovnih procesa. Korišćenje uzora za kontrolu toka procesa pomaže dizajnerima procesa da odrede različite načine za povezivanje aktivnosti. P4 uzor je skup aktivnosti procesa uređenih na način da rešavaju određeni problem. Svi uzori su podeljeni u 6 grupa:

1. osnovni uzori;
2. napredni uzori za grananje i spajanje;
3. strukturni uzori;
4. uzori za višestruke instance;
5. uzori za stanja;
6. uzori za prekide.

6.1 - Osnovni uzori

Ova grupa uzora obuhvata: izvršavanje aktivnosti u sekvenci, grananje, spajanje, paralelno izvršavanje. U osnovne uzore spadaju: Sequence, Parallel Split, Synchronization, Exclusive Choice i Simple Merge.

Sequence uzor

Svrha ovog uzora je sekvencijalno izvršavanje aktivnosti, gde se jedna aktivnost startuje tek pošto se prethodna aktivnost završi. Potreba za ovim uzorom je očigledna, jer skoro svaki proces ima najmanje jedan segment sa dva ili više koraka koje se odvijaju sekvencijalno, jedan za drugim.

slika 31: Primer Sequence uzora

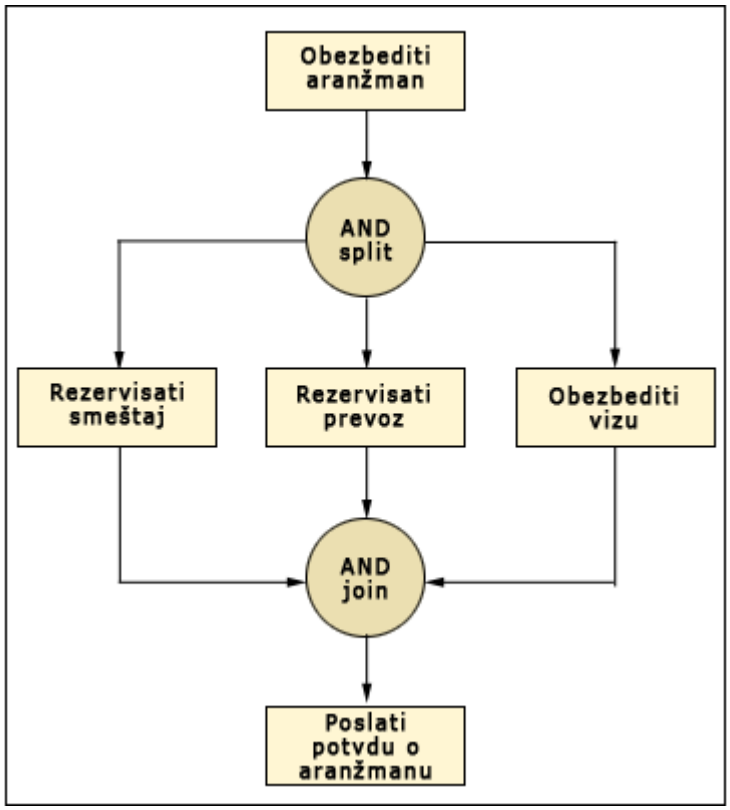
Parallel Split (AND-split)uzor

Svrha ovog uzora je grananje jedne aktivnosti u više paralelnih puteva izvršavanja. Koristi se kada je potrebno da se više tokova aktivnosti izvršava skoro u isto vreme.

Synchronization (AND-join) uzor

Svrha ovog uzora je da nekoliko paralelnih aktivnosti konvergira u jednu aktivnosti, koja čeka da se završe sve prethodne aktivnosti da bi počela da se izvršava. Sinhronizacija, ili spajanje paralelnih puteva izvršavanja razgranatih pomoću Parallel Split uzora, je čest

zahtev. Primer prikazan na slici 32, prikazuje da aktivnost Poslati potvrdu o aranžmanu ne može biti startovana dok se sve tri prethodne aktivnosti (Rezervisati smeštaj, Rezervisati prevoz, i Obezbediti vizu) ne završe.



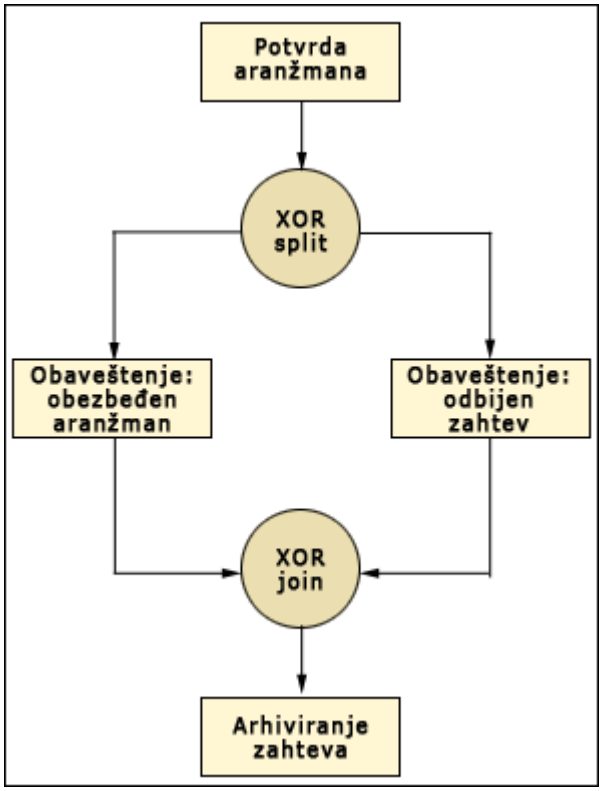
slika 32: Parallel Split i Synchronization

Exclusive Choice (XOR-split) uzor

Uzor je definisan kao lokacija u procesu gde se tok deli na dva ili više ekskluzivnih alternativnih puteva. Uzor je ekskluzivan, jer samo jedan od alternativnih puteva moze biti izabran za nastavak izvršavanja procesa.

Simple Merge (XOR-join) uzor

Svrha ovog uzora je da nekoliko ekskluzivnih uslovnih puteva konvergira u jednu aktivnost, koja počinje svoje izvršavanje kada se izvrši aktivnost na izabranom putu.



slika 33: Exclusive Choice and Simple Merge

6.2 - Napredni uzori za grananje i spajanje

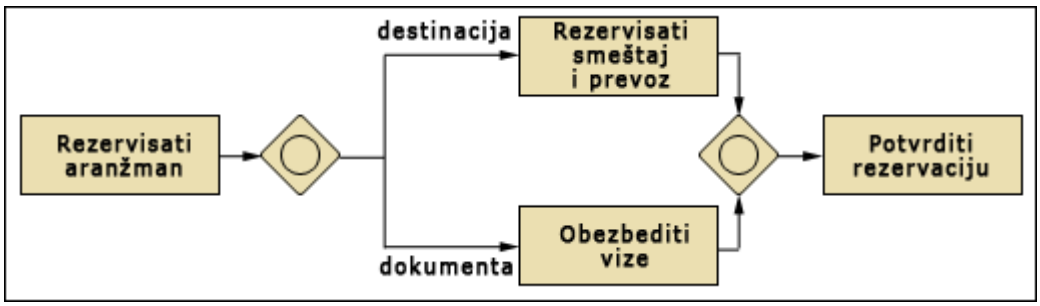
Ova grupa uzora obuhvata: Multi-Choice, Synchronizing Merge, Multi-Merge, kao i Discriminator i N-out-of-M-Join.

Multi-Choice (Inclusive OR-split) uzor

Svrha ovog uzora je da omogući izbor jedne ili više paralelnih grana, gde se svaka grana izvršava ukoliko zadovoljava postavljeni uslov. Uzor opisuje račvanje procesa na više grana, gde je svaki krak zasnovan na uslovu koji je poznat jedino tokom izvršavanja. U odnosu na Exclusive Choice, uzor se razlikuje po tome što dozvoljava da se aktivnosti koje se nalaze na više od jednog putu izvrše. Exclusive Choice dozvoljava izvršavanje aktivnosti na tačno jednoj grani.

Synchronizing Merge (Inclusive OR-join) uzor

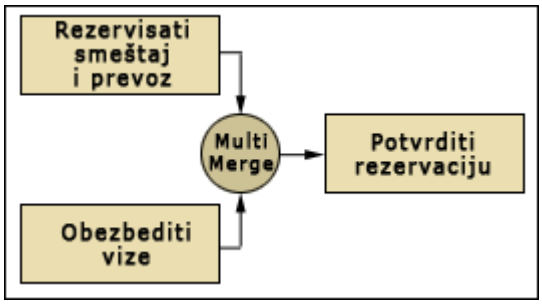
Svrha ovog uzora je da spoji grane dobijene korišćenjem Multi-Choice uzora, odnosno, ovaj uzor čeka da se sve aktivne putanje u paralelnoj strukturi završe.



slika 34: Multi-Choice i Synchronizing Merge uzori

Multi-Merge (Uncontrolled Join) uzor

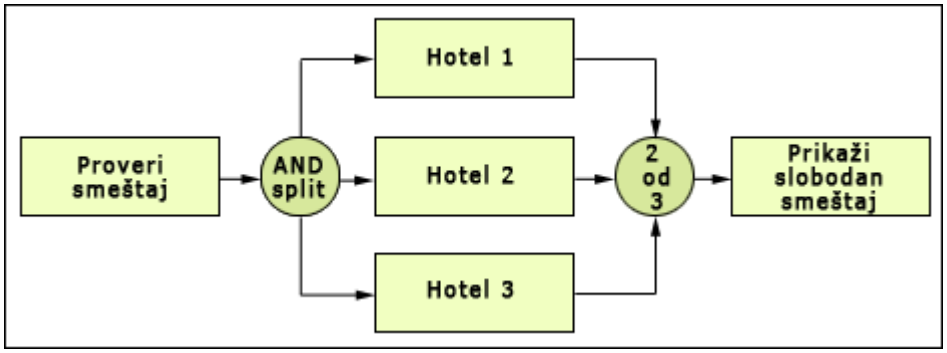
Ovaj uzor dozvoljava svakoj grani da se izvršava nezavisno od drugih, i na taj način omogućava više niti izvršavanja do kompletiranja procesa. Koristan je u slučajevima gde dve ili više paralelnih grana imaju zajedničku kraj.



slika 35: Multi-Merge uzor

Discriminator i N-out-of-M-Join (Complex Join) uzor

Kod Discriminator uzora, kada više paralelnih grana konvergira u tačku spajanja, tačno jednoj grani je, na osnovu uslova u vreme izvršavanja, dozvoljeno da nastavi proces. Ostale grane ostaju blokirane. Discriminator uzor je specijalan slučaj N-out-of-M-Join uzora, gde M paralelnih grana dolazi do tačke konvergencije, ali samo prvih N može da nastavi proces.



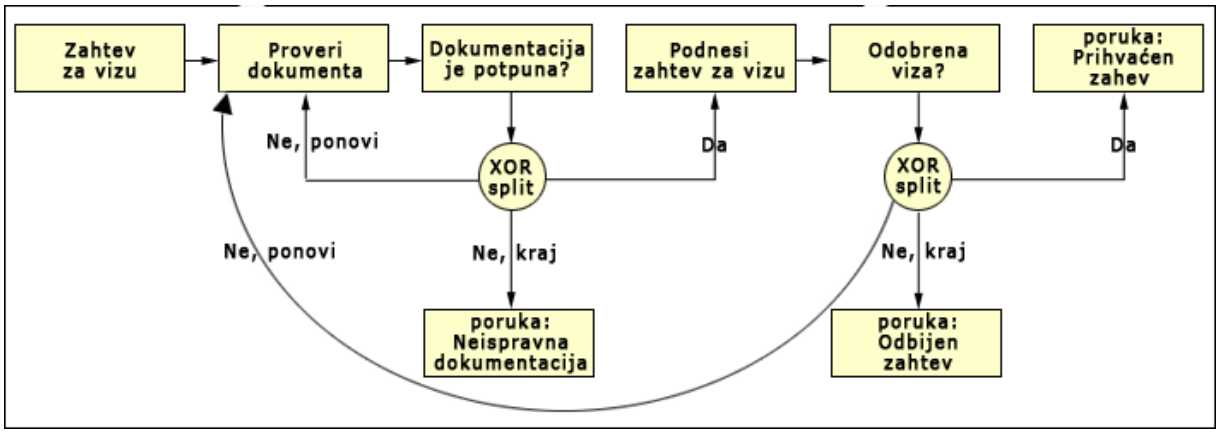
slika 36: 2-of-3 join

6.3 - Strukturni uzori

Ova grupa uzora obuhvata Arbitrary Cycles i Implicit Termination uzore.

Arbitrary Cycles (GOTO / loop) uzor

Uzor omogućava ponavljanje jedne ili skupa aktivnosti ciklično.



slika 37: Arbitrary Cycles (GOTO / loop) uzor

Implicit Termination uzor

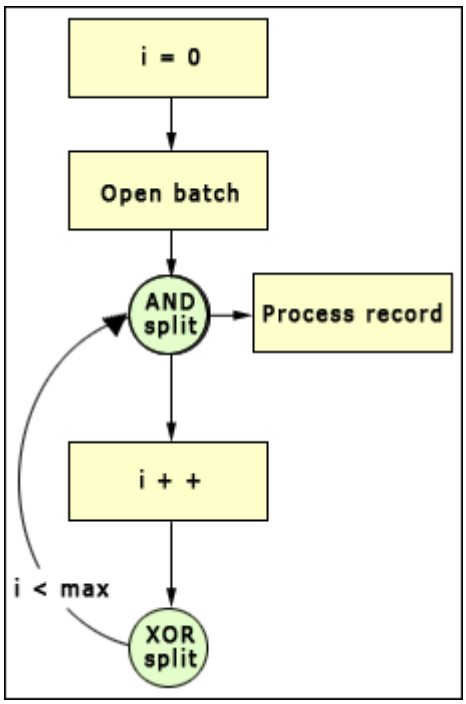
Ovaj uzor dozvoljava da se proces završi samo kada su sve aktivnosti u svim granama završene.

6.4 - Uzori za višestruke instance (MI)

Višestruke instance jedne aktivnosti se izvršavaju konkurentno, što uslovljava podršku i mogućnost sinhronizacije spajanja grana. Ova grupa uzora obuhvata: Without Synchronization, With Design-Time Knowledge, With Runtime Knowledge i Without Runtime Knowledge.

Without Synchronization uzor

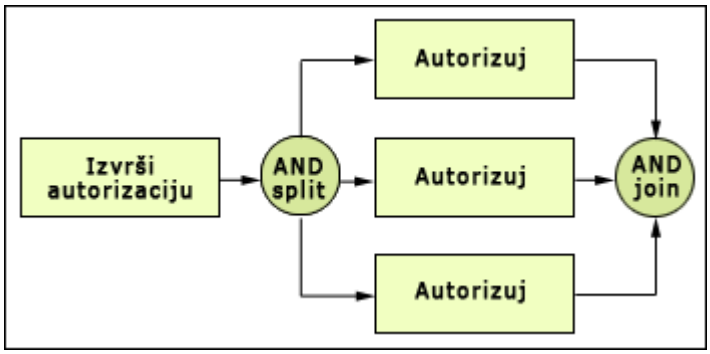
Svrha ovog uzora je da izvrši više instanci jedne aktivnosti, ali bez sinhronizacije.



slika 38: Without Synchronization uzor

With Design-Time Knowledge uzor

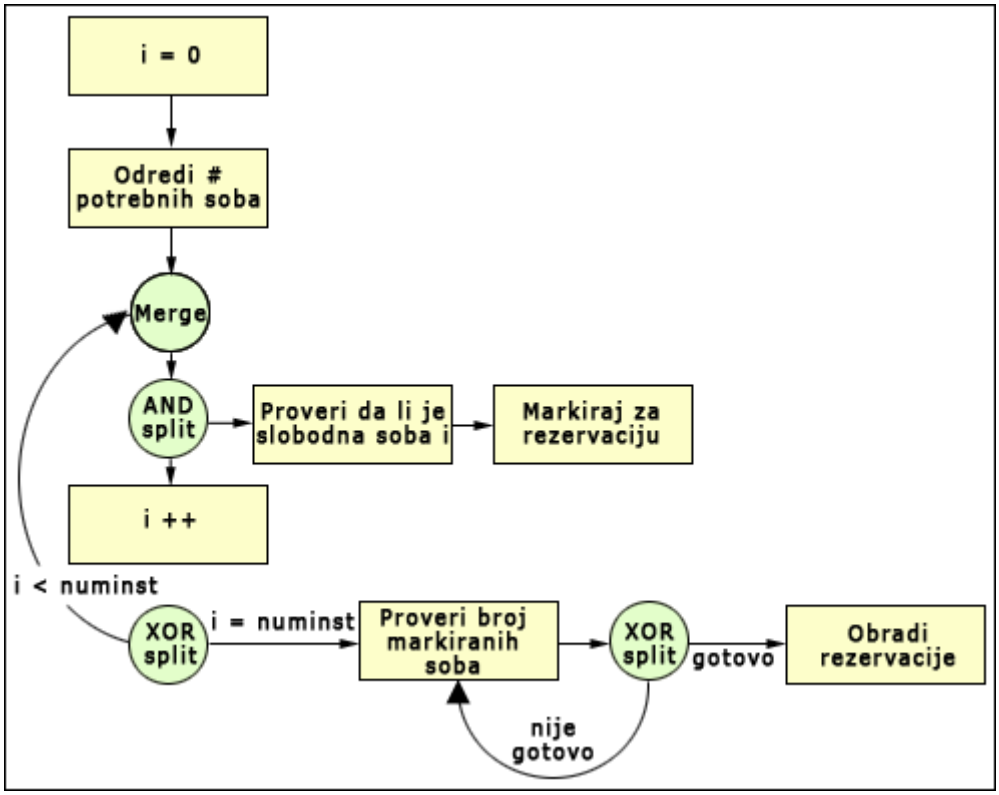
Svrha ovog uzora je da izvrši N konkurentnih instanci jedne aktivnosti, gde je N konstanta za vreme izvršavanja. Instance se spajaju pre nego što se nastavi ostatak izvršavanja procesa.



slika 39With Design-Time Knowledge uzor

With Runtime Knowledge uzor

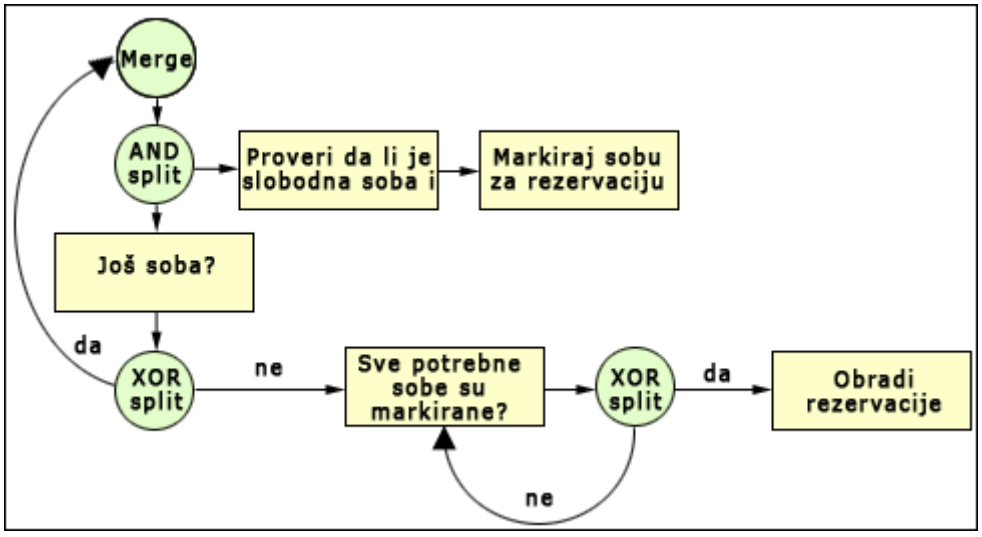
Svrha ovog uzora je da izvrši N konkurentnih instanci jedne aktivnosti, gde je N poznata za vreme izvršavanja, pre nego što se startuje Multiple Instances petlja. Instance se spajaju pre nego što se nastavi ostatak izvršenja posla.



slika 40: With Runtime Knowledge uzor

Without Runtime Knowledge uzor

Svrha ovog uzora je da izvrši više konkurentnih instanci jedne aktivnosti, gde je broj instanci nije poznat, sve dok se ne dođe do određene tačke pri obradi instanci. Instance se spajaju pre nego što se nastavi ostatak izvršenja posla.



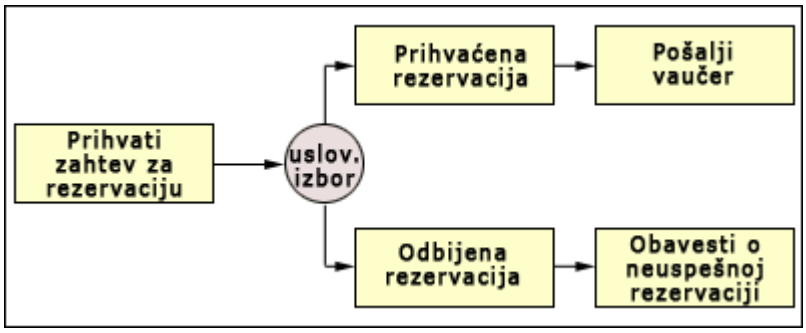
slika 41: Without Runtime Knowledge uzor

6.5 - Uzori za stanja

Uzori se primenjuju na procese koji su zasnovani na događajima i koji provode dosta vremena čekajući da se desi događaj, koji će pokrenuti narednu aktivnost. Ova grupa uzora obuhvata: Deferred Choice, Interleaved Parallel Routing i Milestone.

Deferred Choice (Event Choice and Pick) uzor

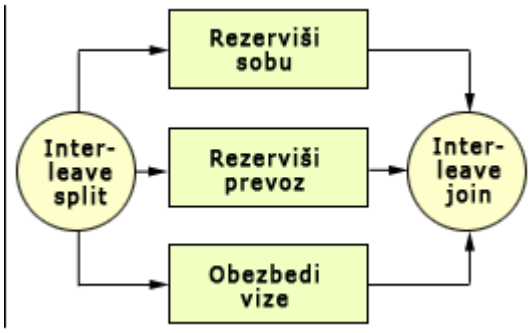
Svrha ovog uzora je slična svrsi Exclusive Choice uzora, i sastoji se u tome da se kreira odluka o praćenju jednog od više puteva izvršavanja, s tim što se odluka o odabranom putu ne donosi trenutno, već na osnovu podizanja nekog događaja. Procesi koji su zasnovani na događajima veoma često koriste ovaj uzor. Situacija u kojoj se često koristi je čekanje na pozitivan ili negativan odgovor na zahtev (request).



slika 42: Deferred Choice uzor

Interleaved Parallel Routing (Ad Hoc Process) uzor

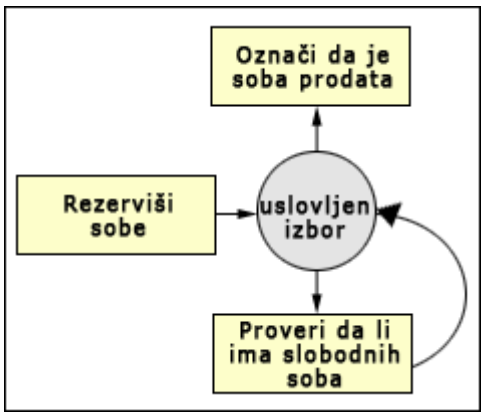
Svrha ovog uzora je da nekoliko aktivnosti budu izvršavane redno, a ne paralelno kako to ime uzora sugeriše, pri čemu je redosled izvršavanja slučajan, jer se ne može unapred znati.



slika 43: Interleaved Parallel Routing uzor

Milstone uzor

Svrha ovog uzora je da opiše scenario u kojem aktivnost može biti izvršena samo posle dešavanja, koji omogućava određeni događaj, ali pre nego što događaj bude onemogućen. Aktivnost može biti izvršavana više puta. Primer za ovaj uzor može da bude aukcija, u toku koje mogu biti podnešene ponude za licitiranje samo između događaja kojim se otvara i zatvara aukcija.



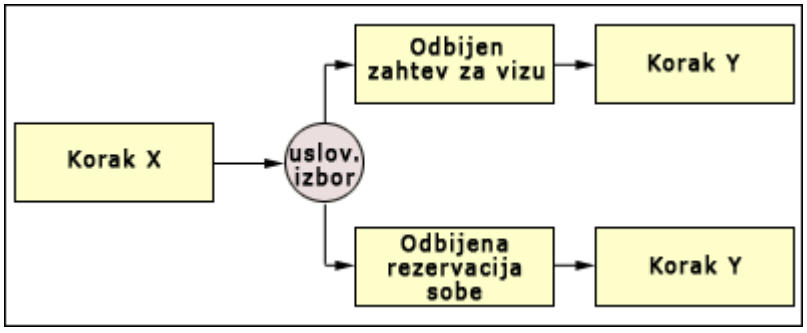
slika 44: Milstone uzor

6.6 - Uzori za prekide

Ova grupa obuhvata Cancel Activity i Cancel Case uzore.

Cancel Activity (Kill Activity) uzor

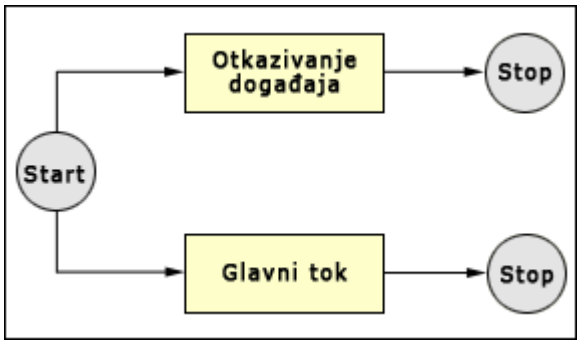
Svrha ovog uzora je da zaustavi izvršavanje određene aktivnosti procesa kao posledicu podizanja događaja otkazivanja. Koristan je kod prekidanja aktivnosti, koje dugo traju, ili koje su obustavljene.



slika 45: Cancel Activity uzor

Cancel Case (Kill Process) uzor

Svrha ovog uzora je da zaustavi izvršavanje procesa kao posledicu podizanja događaja otkazivanja.



slika 46: Cancel Case uzor

6.7 BPEL i uzori

BPEL ima ugrađenu podršku za 14 (13 direktno i 1 indirektno) od 20 P4 uzora.

P4 uzor	Podrška	BPEL pristup
Sequence	+	sequence aktivnost
Parallel Split	+	flow aktivnost
Synchronization	+	flow aktivnost nastavljena drugom aktivnošću, koja se neće izvršiti dok se svi paralelni putevi u toku ne završe
Exclusive Choice	+	switch aktivnost
Simple Merge	+	switch aktivnost nastavljena drugom aktivnošću, koja se neće izvršiti dok se selektovana aktivnost u switch ne završi
Multi-Choice	+	flow aktivnost sa uslovnim vezama na aktivnosti koje će biti izabrane
Synchronizing Merge	+	koristi eliminaciju nekorišćenih puteva da bi spojili rezultati višestukog izbora
Multi-Merge	-	ne
Discriminator	-	ne
Arbitrary Cycles	-	ne
Implicit Termination	+	flow aktivnost koja se prekida
MI Without Synchronization	+	invoke u while petlji
MI With Design-Time Knowledge	+	izvršavanje svake instance kao odvojene aktivnosti u okviru flow aktivnosti
MI With Runtime Knowledge	-	ne
MI Without Runtime Knowledge	-	ne
Deferred Choice	+	pick aktivnost
Interleaved Parallel Routing	+-	izvršavanje aktivnosti serijski, u bilo kom redosledu (promenljiva AccessSerializable ima vrednost yes)
Milestone	-	u okviru while petlje, ispitivanje uslova
Cancel Activity	+	prekid aktivnosti
Cancel Case	+	terminate aktivnost

6.8 BPMN i uzori

BPMN ima podršku za većinu od 20 P4 uzora.

P4 uzor	Podrška	BPMN pristup
Sequence	+	Normal sequence flow
Parallel Split	+	Parallel gateway za grananje
Synchronization	+	Parallel gateway za spajanje
Exclusive Choice	+	Exclusive gateway za grananje
Simple Merge	+	Exclusive gateway za spajanje
Multi-Choice	+	Inclusive gateway za grananje
Synchronizing Merge	+	Inclusive gateway za spajanje
Multi-Merge	+	Nekonstrolisano grananje i spajanje
Discriminator	+	Complex gateway element kao mesto spajanja

		za prihvatanje jednog od dolazećih puteva
Arbitrary Cycles	+	Sequence flow koji dozvoljava petlje
Implicit Termination	+	dozvoljeni višestruki završni događaji - prvi događaj koji se okine, prekida proces
MI Without Synchronization	+	marker za višestruke instance
MI With Design-Time Knowledge	+	koriste se višestruke instance
MI With Runtime Knowledge	+-	naprednije korišćenje višestrukih instanci
MI Without Runtime Knowledge	-	relativno teško kodiranje
Deferred Choice	+	Exclusive data-based gateway za grananje
Interleaved Parallel Routing	+	ad hoc proces
Milestone	+	
Cancel Activity	+	upravljanje izuzecima
Cancel Case	+	upravljanje izuzecima ili implicitni prekid

7. Mapiranje BPMN dijagrama u BPEL izvršni jezik

Svrha kreiranja BPMN dijagrama je dvostruka, sa jedne strane treba da omogući poslovni pogled na poslovni proces, a sa druge strane da omogući generisanje izvršnog koda pomoću BPEL jezika.

Bez obzira na svoje ime - Business Process Execution Language – BPEL se realno ne bavi poslovnim procesima, bar ne na onaj način kako se njima bavi BPM zajednica. Na primer, sa BPM stanovišta, poslovni proces je sastavljen od različitih podprocesa koji mogu biti ugnježdeni ili ulančani međusobno. Svaki podproces može biti vlasništvo ili može biti izvršavan od strane različitih poslovnih jedinica. BPEL ne poznaje koncept podprocesa, koji je osnovna jedinica za mogućnost ponovnog korišćenja u BPM. To ne znači da se korišćenjem BPEL ne može kreirati proces od ugnježdenih i ulančanih komponenti. Takođe, BPEL ne podržava koncepte ljudi učesnika identifikovanih preko uloga, grupa ili korisničkih imena. U BPEL, aktivnosti procesa se dodeljuju servis endpointima, koji se razrešavaju preko URL. Ne postoji odvojeni endpoint servisa za svaku ulogu, grupu ili imenovanog korisnika. Umesto toga, postoji jedan endpoint, koji rutira zadatak prema odgovarajućem korisniku ili redu čekanja. [Silver]

BPMN specifikacija sadrži više od 50 strana koje prikazuju mapiranje BPMN u BPEL jezik. Cilj je da se premosti jaz između grafičkog dizajna i izvršavanja procesa. Sam proces mapiranja nije jednostavan, pre svega zbog postojanja različitog broja elemenata.

Pažljivije posmatranje, pokazuje da je mapiranje samo delimično, ostavljajući po strani modele sa nestruktuiranim topologijama, kao i delove kao što su OR-join ili kompleksni Gateway. Osim toga, pošto je mapiranje samo opisano, ono predstavlja predmet interpretacija. Uopšteno govoreći, mnogo dvosmislenosti se može naći u BPMN specifikaciji vezano za mapiranje u BPEL jezik. [P4 BPMN]

7.1 Mapiranje BPMN dijagrama

BPMN dijagram može biti sastavljen od skupa (polu-) nezavisnih komponenata, koje se prikazuju u različitim Pool elementima. Za Pool elemente ne postoji specifično mapiranje. U slučaju da sadrži Process elemente ("white box"), može se mapirati u BPEL proces. Međutim, u slučaju mapiranja sadržaja Process elementa, može postojati jedan ili više izvedenih procesa, neophodnih za upravljanjem složenim ponašanjem, kao što su petlje. Ako je Pool element tipa "black box", njegovi atributi mogu se koristiti za definisanje PartnerLink BPEL elementa.

[Havey]

BPMN	**BPEL**
Start event: Basic, Message, Timer, Rule, Link	Receive element sa atributom createInstance = yes
Start event: Multiple	Pick element sa atributom createInstance = yes
End event: Message	Invoke element ili Reply element
End event: Error	Throw element
End event: Compensation	Compensate element
End event: Terminate	Terminate element

End event: Link	Invoke element
Intermediate event: Error	Upravljanje greškom ili podizanje greške.
Activities: MI	Nije moguće direktno mapiranje. Mogu se koristiti različiti elementi.
Activities: Subprocesses	Koristiti Invoke element za pokretanje.
Gateway: Exclusive data-based	Switch element
Gateway: Exclusive event-based	Pick element
Gateway: Inclusive	Više Switch elemenata u okviru Flow elementa.
Gateway: Complex	Nije moguće direktno mapiranje.
Gateway: Parallel	Flow element
Sequence Flow	Modeliranje toka kontrole od jednog elementa do drugog ili eksplicitno, korišćenjem Flow link elemenata, ili implicitno, korišćenjem kontrolnih struktura kao što su Sequence, While ili Switch.
Message Flow	Nije moguće direktno mapiranje.
Pool	Nije moguće direktno mapiranje.
Lane	Nije moguće direktno mapiranje.
Artifact	Nije moguće direktno mapiranje.
Assotiation	Nije moguće direktno mapiranje.
Exception Flow	Sve aktivnosti koje slede Intermediate Event, a sve dok se Exception Flow ne vrati u Normal Flow, mogu da se mapiraju u *faultHandlers* BPEL element.
Compensation Association	Moguće je mapiranje u *compensationHandler* element.

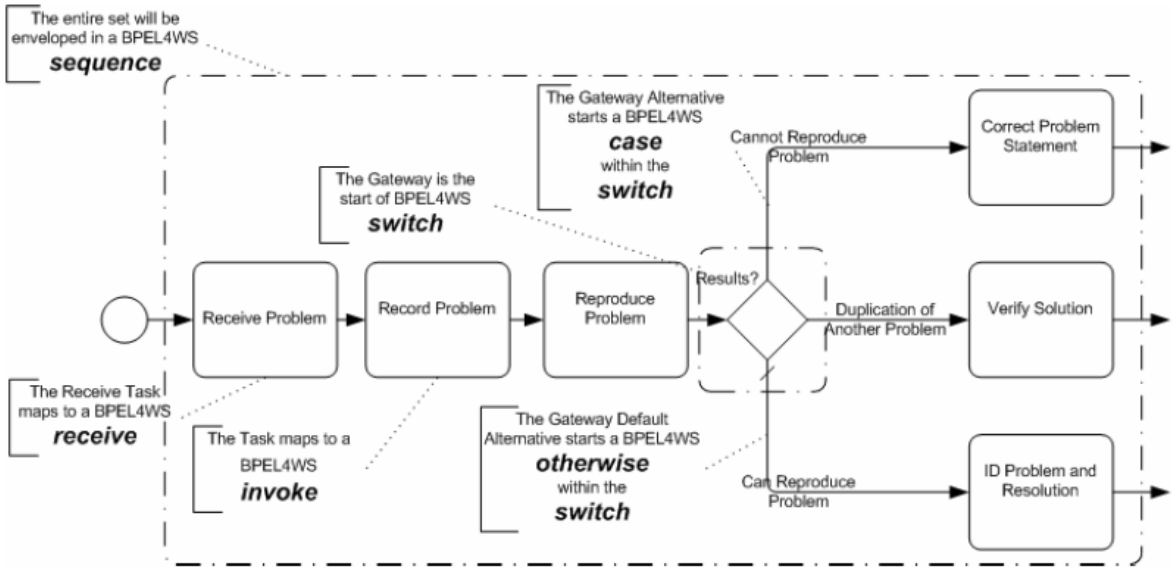


Figure 9 : A BPD with Annotations to Show the Mapping to BPEL4WS

slika 47: Grafički prikaz mapiranja BPMN dijagrama u BPEL kod [White]

Problem nemogućnosti jednoznačnog mapiranja BPMN notacije i BPEL izvršni jezik, proizvođači softvera rešavaju neki od sledećih načina:

- Grafički predstave BPEL proces i nazivaju ga BPMN.
- Koriste jedan alat za kreiranje BPMN dijagrama, a drugi za formiranje BPEL procesa.

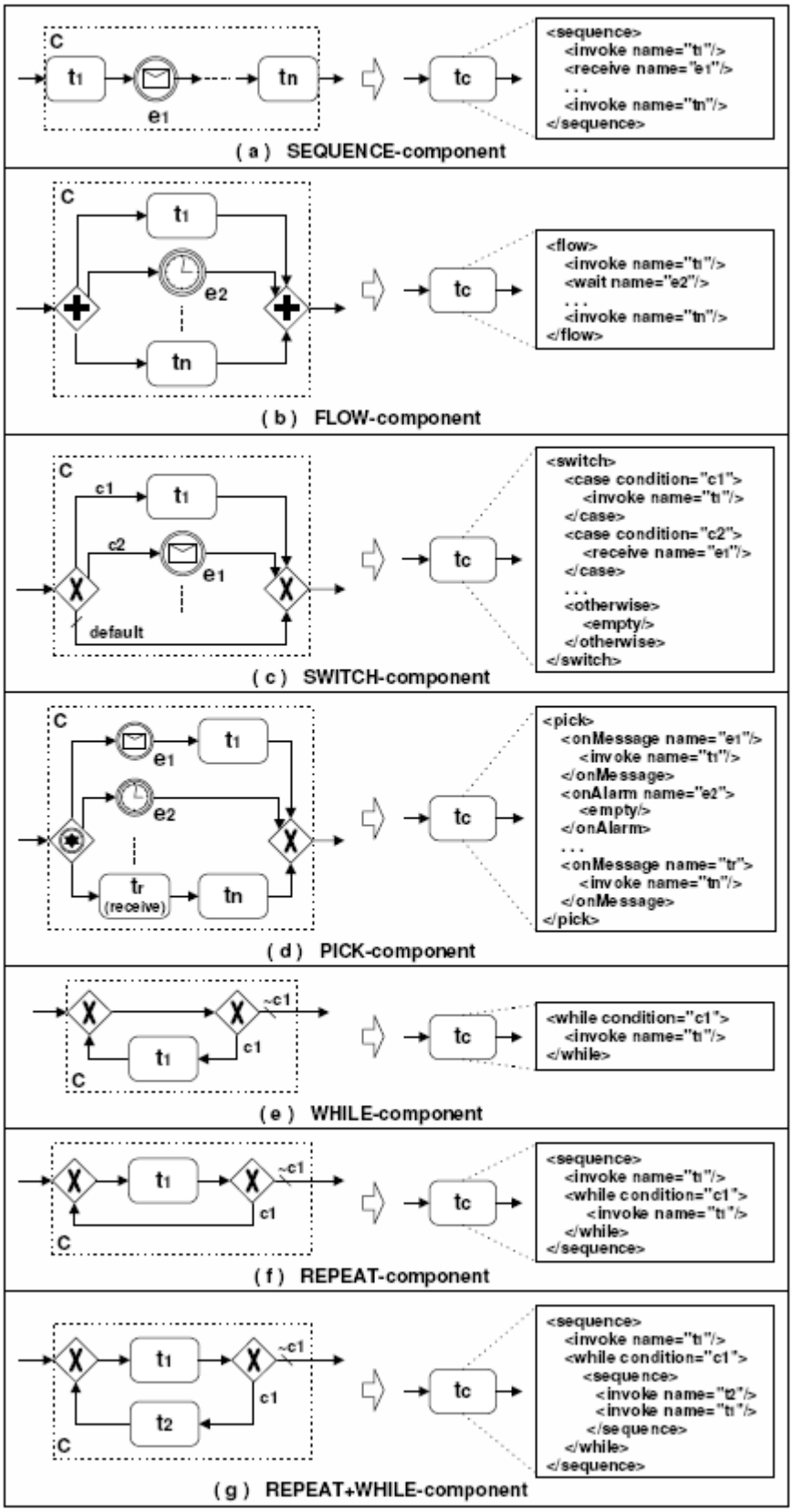
- Od kreiranog BPMN dijagrama kreira se BPEL izvršni kod, prilagođen određenoj procesnoj mašini, pri čemu se uobičajeno zahteva da BPMN ne sadrži elemente, koji se ne mogu direktno mapirati.

Trenutna situacija je takva, da bi se pomoću bilo kod dostupnog alata generisao BPEL od BPMN, potrebno je da se reše dva problema:

- prvi, da se dodaju svi detalji vezani za implementaciju pre nego što se BPMN izveze u BPEL, što često dovodi do uobičajenog pitanja "Da li očekujete da će poslovni analitičar postati programer, ili obrnuto, da li očekujete da će programer postati poslovni analitičar?";
- i drugi, zbog loše struktuiranog BPMN grafikona, ponekad izvoz nije moguć, ili za posledicu ima nevažeći BPEL kod.

Problemi koji se javljaju kod generisanja BPEL koda na osnovu BPMN dijagrama su karakteristični kod svih modela na visokom nivou. Modeli na visokom nivou moraju da se na odgovarajući način sinhronizuju sa modelima na nižem nivou, što svakako nije jednostavno uraditi, osim u retkim situacijama kada je moguće obaviti mapiranje jedan-na-jedan.

Na slici je prikazano nekoliko primera BPMN dijagrama i njima odgovarajući BPEL kod.



slika 48: Primeri elemenata BPMN dijagrama sa mapiranjem u BPEL kod [BPMN2BPEL]

7.2 BPMN/BPEL vs. UML

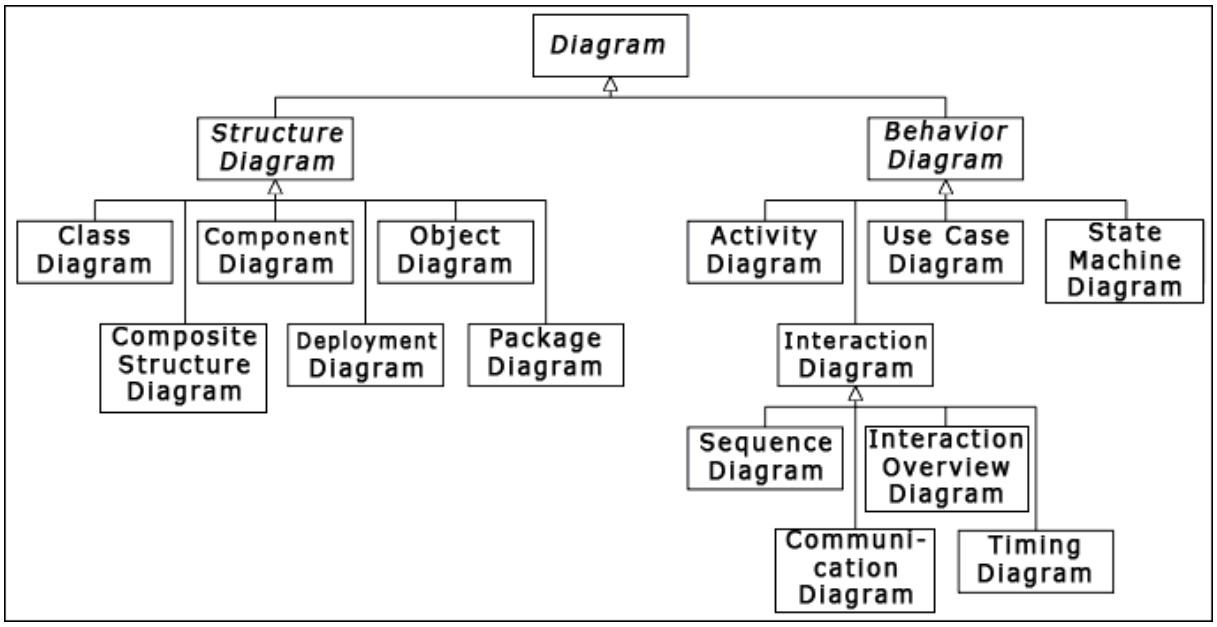
Veoma često se postavlja pitanje o svrsishodnosti uvođenja novih jezika i notacija za modeliranje poslovnih procesa, s obzirom da već određeni niz godina postoji standardizovan i veoma moćan jezik za modeliranje Unified Modeling Language (UML).

UML predstavlja jezik za specifikaciju, vizualizaciju, konstrukciju i dokumentovanje proizvoda softverskog sistema, kao i za modeliranje poslovnih procesa i drugih ne – softverskih sistema. UML predstavlja skup najboljih praktično primenjenih inženjerskih modela koji se koriste u razvoju velikih i složenih sistema.

Razlog što je UML postao standardni jezik za modeliranje je sadržan u činjenici da je nezavistan od programskih jezika.

UML podržava bogat skup grafičkih elemenata, što omogućava da na razumljiv način opiše klase, komponente, čvorove, aktivnosti, dijagrame toka, slučajeve korišćenja, objekte, stanje, kao i da modeluje relacije između navedenih elemenata. UML podržava i mogućnost proširenja preko stereotipnih elemenata.

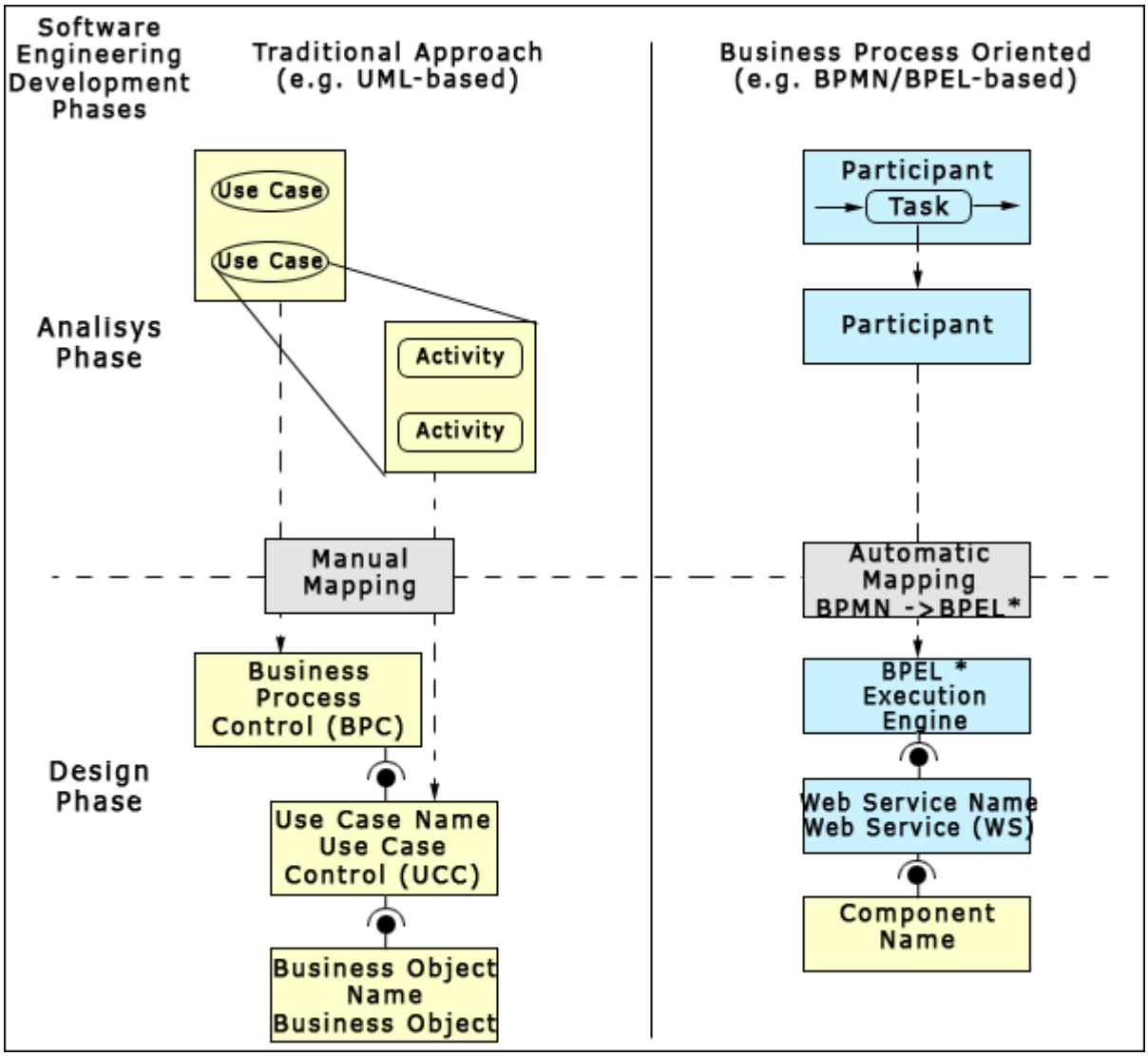
UML podržava 13 tipova dijagrama, razvrstanih u 3 kategorije: strukturni dijagrami, dijagrami ponašanja i dijagrami interakcije, koji predstavljaju podskup dijagrama ponašanja.



slika 49: UML 2.0 tipovi dijagrama

UML jezik podržava faze analize i dizajna u okviru procesa razvoja aplikacija. Faza analize obuhvata prikupljanje zahteva korisnika, i formalizuje se pomoću UML slučaja korišćenja i dijagrama aktivnosti, koji zajedno predstavljaju poslovnu logiku softverske aplikacije. U okviru dizajn faze, poslovna logika se mapira u komponente, koje čine arhitekturu aplikacije. Ovo mapiranje se obavlja manuelno, što dovodi do neefikasnosti, nekonzistentnosti, kao i nefleksibilnosti. Sve do sada, UML nije obezbedio mogućnost da na odgovarajući način mapira slučajeve korišćenja i dijagrame aktivnosti u jezik za izvršavanje procesa. Zbog toga, neophodno je da se izabere drugi jezik za programiranje

poslovnih procesa. BPMN dijagrami i BPEL jezik su rešenja za navedeni problem. [Dev SOA]



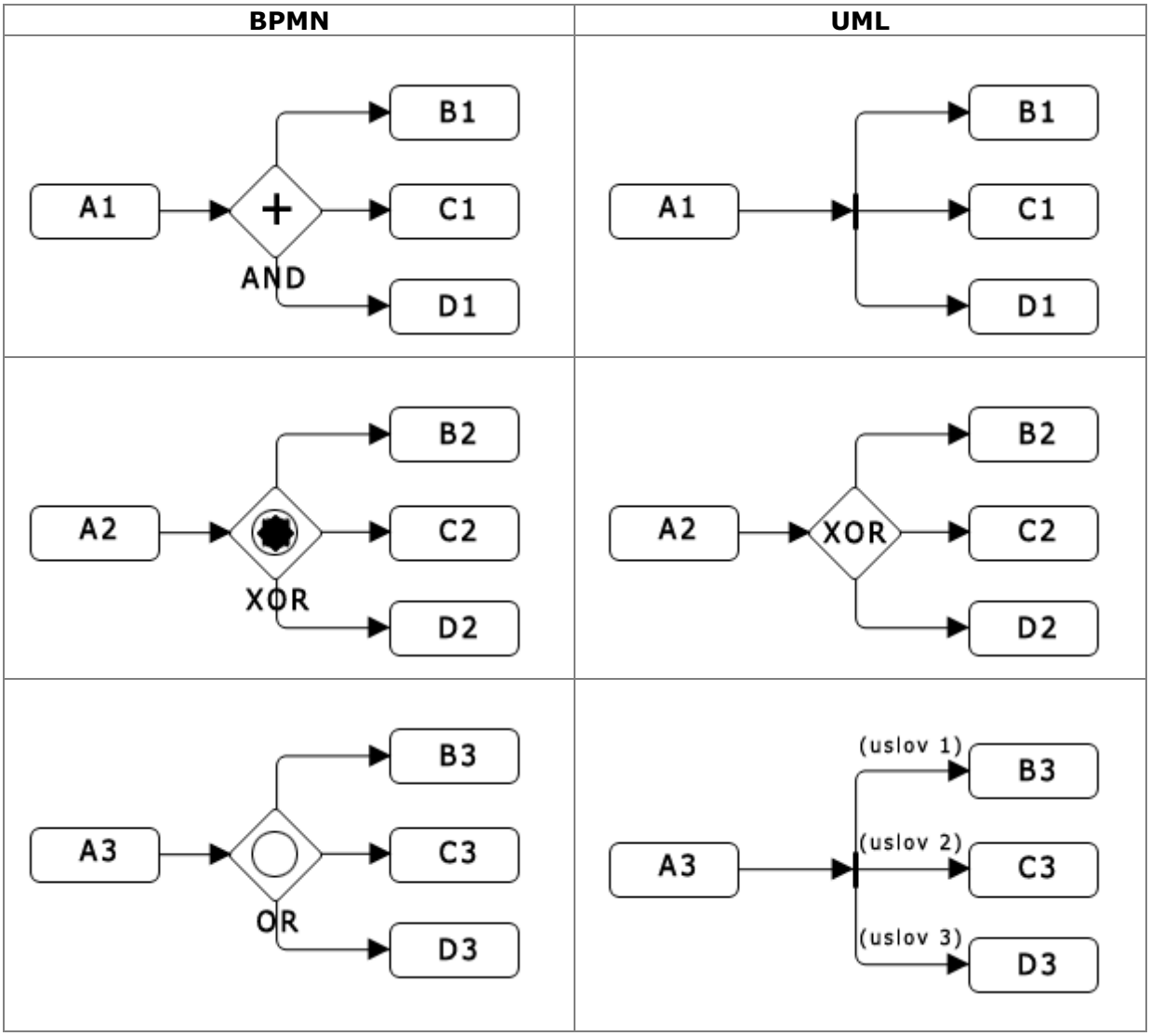
slika 50: Pristupi modeliranju u fazi analize i fazi dizajna

Ukratko, razlozi za postojanje BPMN notacije i BPEL jezika, pored UML jezika, su sledeći:

- namenjeni su različitim ciljnim grupama:
  - UML je namenjen softverskim analitičarima, dizajnerima i diveloperima, jer nudi specifikaciju, vizuelizaciju i mogućnost dokumentovanja artifakta za kompjuterski zasnovane sisteme,
  - BPMN i BPEL su namenjeni poslovnim analitičarima, arhitektama sistema i softverskim inženjerima, jer nude specifikaciju, vizuelizaciju i mogućnost dokumentovanja artifakta vezanih za poslovni domen,
- zasnivaju se na različitim paradigmama:
  - UML je zasnovan na objektno orijentisanom pristupu,
  - BPMN i BPEL su zasnovani na procesnom pristupu,
- različitost u nivou implementacije:
  - UML jeziku nedostaje mogućnost implementacije poslovnih modela, koji su visoko konceptualni.

Bez obzira na postojanje navedenih različitosti, ipak postoji mogućnost da se za predstavljanje poslovnih procesa koriste UML dijagrami aktivnosti. Na sledećoj slici je

prikazano nekoliko uzora za opis ponašanja poslovnih procesa pomoću BPMN notacije i preko UML dijagrama aktivnosti.



slika 51: Nekoliko uzora predstavljenih i pomoću BPMN i UML

Bitno je naglasiti, da je programiranje zasnovano na procesnom pristupu, sledeći korak u razvoju softverskog inženjeringa. Prema tome, procesno orijentisano programiranje ne treba da zameni, već da dopuni postojeće pristupe programiranju, kao što su strukturno, objektno i komponentno orijentisano programiranje. [Dev SOA]

8. Alati za podršku izgradnji poslovnih procesa

8.1 Alati za dizajn i razvoj poslovnih procesa

Postoji nekoliko alata, koji obezbeđuju grafičko okruženje za dizajn i razvoj poslovnih procesa. Neki alati se uglavnom, mada ne uvek, isporučuju zajedno sa BPEL serverima. Najvažniji alati za dizajn i razvoj poslovnih procesa su:

- Sybase PowerDesigner 12 (http://www.sybase.com/powerdesigner/index.html)
- Oracle JDeveloper 10g (http://www.oracle.com/technology/products/jdev/index.html)
- Oracle BPEL Designer for Eclipse (http://www.oracle.com/technology/products/ias/bpel/index.html)
- IBM WebSphere Studio Application Developer Integration Edition (http://www.ibm.com/software/integration/wsadie/)
- iGrafx BPEL (http://www.igrafx.com/products/bpel/index.html)
- itp Process Modeler for Microsoft Visio (http://www.itp-commerce.com/)

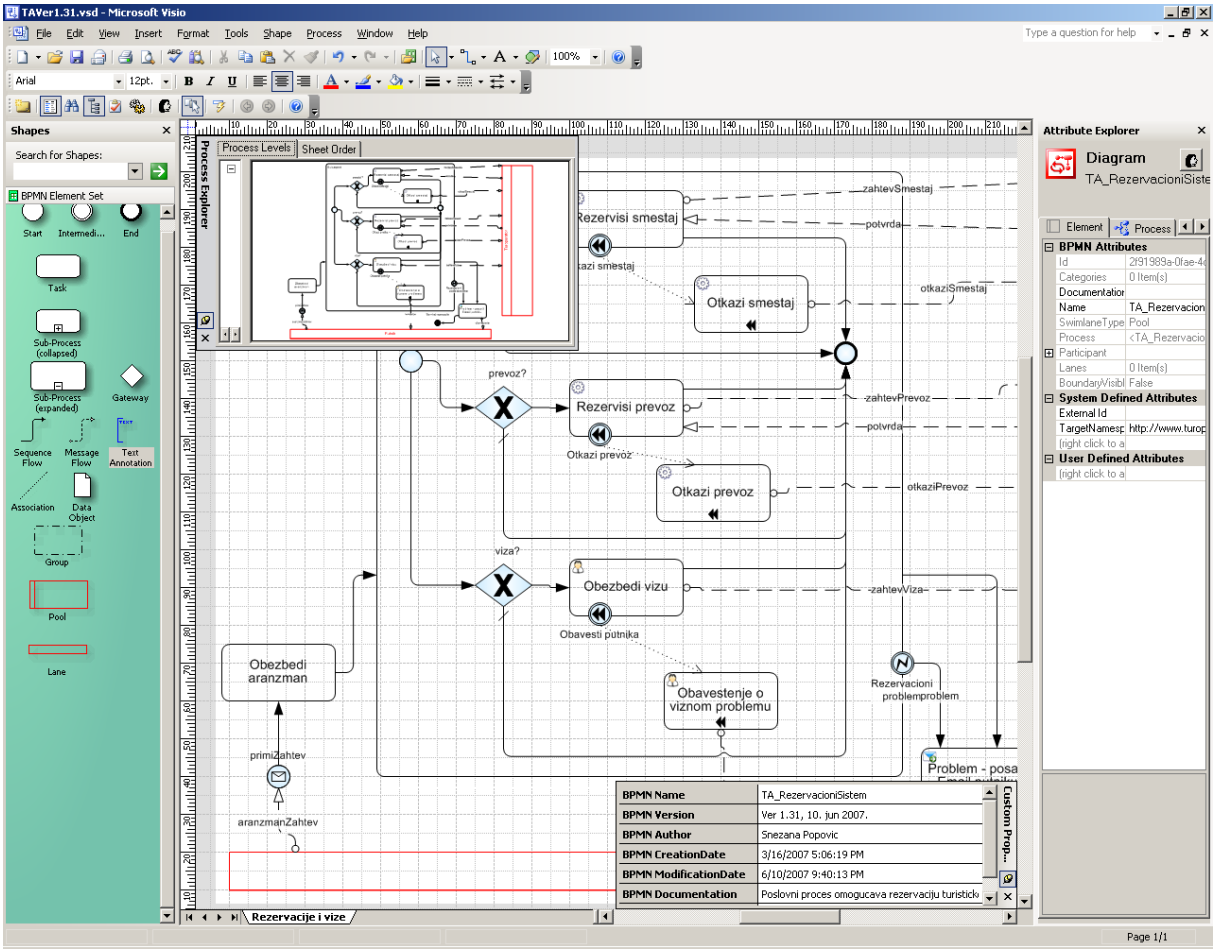
8.1.1 ITP Process Modeler for Microsoft Visio

Za kreiranje BPMN dijagrama za implementaciju poslovanja turističke agencije prikazan u radu, korišćen je ITP Process Modeler. Process Modeler za Microsoft Visio omogućava kreiranje BPMN dijagrama po BPMN 1.0 specifikaciji, uključujući sve elemente i sve atribute definisane u navedenoj specifikaciji.

ITP alat sadrži skup šablona, koji se importuju u Microsoft Visio. BPMN dijagram se crta tako što se šabloni prevlače na platno za crtanje. Iako Process Modeler sadrži ukupno 15 osnovnih šablona, oni pokrivaju 100% BPMN mogućnosti, što je ostvarivo zahvaljujući prozoru za podešavanje svojstava za svaki pojedinačni element, kao i mogućnosti da se koristi source editor za dalja podešavanja.

Alat je veoma komforan za korišćenje, i osim mogućnosti za kreiranje BPMN dijagrama, nudi i sledeće mogućnosti:

- sintaksnu validaciju modela;
- mogućnost prebacivanja kreiranog BPMN dijagrama u BPEL izvršni jezik, prilagođen određenoj procesnoj mašini;
- brojčano označavanje elemenata, izveštavanje i dokumentovanje dijagrama;
- simulaciju pomoću analize modela, koja se zasniva na vremenu i kreiranju instanci procesa;
- trejsing i dibaging modela, što omogućava da se obavi navigacija kroz model korak po korak.



slika 52: ITP BPMN alat za crtanje BPMN dijagrama

8.2 BPEL serveri

BPEL serveri obezbeđuju izvršno okruženje za izvođenje BPEL poslovnih procesa. BPEL je čvrsto povezan sa Web servisima i modernim softverskim platformama koje podržavaju razvoj Web servisa, posebno sa Java Enterprise Edition i Microsoft .NET okruženjem. Povezanost omogućava BPEL serverima da koriste servise koje obezbeđuju ove platforme, kao što su sigurnost, upravljanje transakcijama, skalabilnost, povezivanje sa bazama podataka, korišćenje komponenti kao što su EJB (Enterprise Java Beans) i COM+ (Component Object Model), messaging sisteme kao što su JMS (Java Message Service) ili MSMQ (Microsoft Message Queue), i drugo.

Najvažniji komercijalni BPEL serveri su:

- Microsoft BizTalk
  (http://www.microsoft.com/biztalk/)
- Oracle BPEL Process Manager
  (http://www.oracle.com/technology/products/ias/bpel/index.html)
- IBM WebSphere Business Integration Server Foundation
  (http://www.ibm.com/software/integration/wbisf)
- IBM alphaWorks BPWS4J
  (http://www.alphaworks.ibm.com/tech/bpws4j)

- BEA WebLogic Integration (http://www.bea.com/framework.jsp?CNT=index.htm&FP=/content/products/weblogic/integrate/)
- OpenStorm Service Orchestrator (http://www.openstorm.com/)
- Active Endpoints ActiveWebflow (http://www.active-endpoints.com/products/index.html)
- Sun Java Integration Suite for Java Enterprise Suite (http://www.sun.com/software/javaenterprisesystem/), fnekada poznat kao SeeBeyond eInsight Business Process Manager (http://www.seebeyond.com/software/einsight.asp)
- Cape Clear Orchestration Studio (http://www.capeclear.com/products/)
- OpenLink Virtuoso Universal Server (http://virtuoso.openlinksw.com/)
- Parasoft BPEL Maestro (http://www.parasoft.com/jsp/products/home.jsp?product=BPEL)
- Fiorano Business Integration Suite (http://www.fiorano.com/products/fesb/fioranobis.htm)
- PolarLake Integration Suite (http://www.polarlake.com/en/html/products/integration/index.shtml)
- Fuego BPM (http://www.fuegotech.com/fuego_software.html)
- Digité Enterprise Business Process Management (http://www.digite.com/4.0/products/digite_ent_business-process.htm)

Postoji i nekoliko open-source implementacija BPEL servera:

- ActiveBPEL Engine (http://www.activebpel.org/)
- FiveSight Process eXecution Engine PXE (http://www.fivesight.com/pxe.shtml)
- bexee BPEL Execution Engine (http://sourceforge.net/projects/bexee)
- Apache Agila (http://wiki.apache.org/agila/), ranije poznat kao Twister (http://www.smartcomps.org/twister/)

8.2.1 BizTalk Server 2006

Kao izvršno okruženje za izvođenje BPEL poslovnog procesa turističke agencije, korišćen je BizTalk Server 2006, koji je kreirala softverska kuća Microsoft.

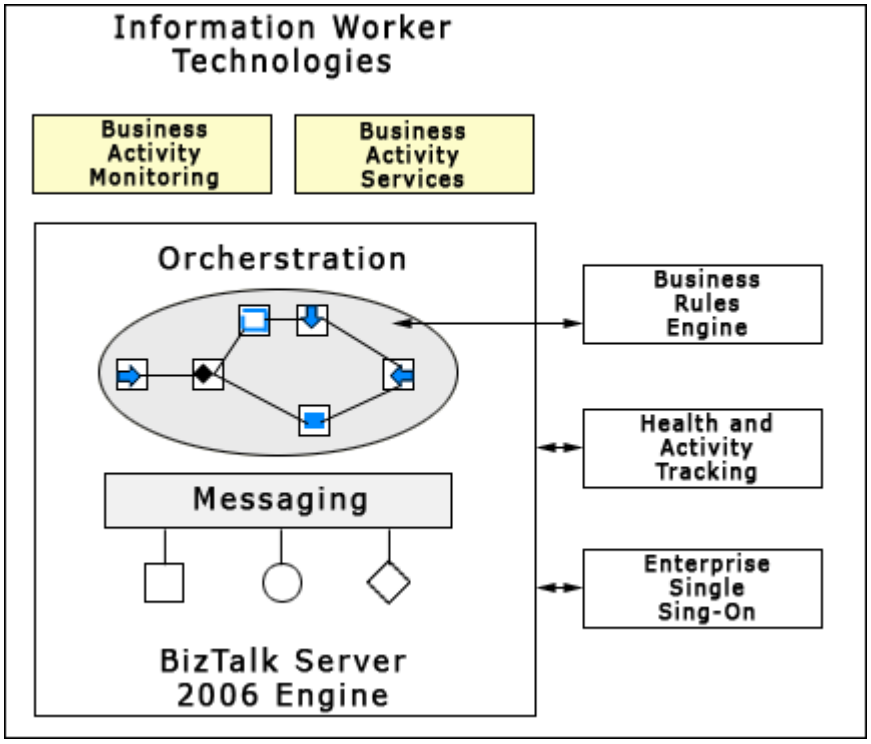
BizTalk Server 2006 je moćan server za upravljanje poslovnim procesima. BizTalk Server 2006 pomaže organizacijama da povežu više različitih sistema i obezbeđuje komunikaciju sa poslovnim partnerima, odnosno omogućava rešavanje problema koji se javljaju kod automatizacije poslovnih procesa:

- postojanje različitih aplikacija;
- nedostatak vremena za razvoj integrisanih rešenja korišćenjem postojećih razvojnih alata;
- suviše duga isporuka integrisanih rešenja, koja vrlo često traje isto kao njihov razvoj;
- potreba za blagovremenim kreiranjem izveštaja na osnovu podataka koji se nalaze na više mesta;
- teškoće pri promeni integrisanih internih aplikacija, zbog napora i novca koji treba uložiti;
- nedostatak konzistentnih metoda za implementaciju poslovnih procesa;
- ograničeno praćenje izvršavanja poslovnog procesa;
- teškoće pri promeni poslovnih partnera, zbog potrebe za promenom IT infrastrukture, koja omogućava komunikaciju.

Kreirani poslovni proces mogu nadgledati poslovni korisnici, a ne samo tehnički obrazovano osoblje.

Cilj BizTalk Server 2006 softvera je da pomogne organizacijama da kreiraju automatizovane poslovne procese, koji spajaju različite sisteme u povezanu celinu. [Chappell BTS]

BizTalk Server 2006 komponente



slika 53: Glavne komponente BizTalk Server 2006

Svakako najvažniju komponentu BizTalk Server 2006 predstavlja BizTalk Server 2006 Engine. Sastavni delovi ove mašine su:

- Messaging komponenta - Obezbeđuje mogućnost komunikacije sa drugim softverima pomoću adaptera, koji podržavaju različite protokole i formate podataka, uključujući Web servise i mnoge druge.
- Orchestration - Obezbeđuje podršku za kreiranje i izvršavanje grafički definisanih procesa. Omogućava implementaciju logike koja povezuje sve delove poslovnog procesa.

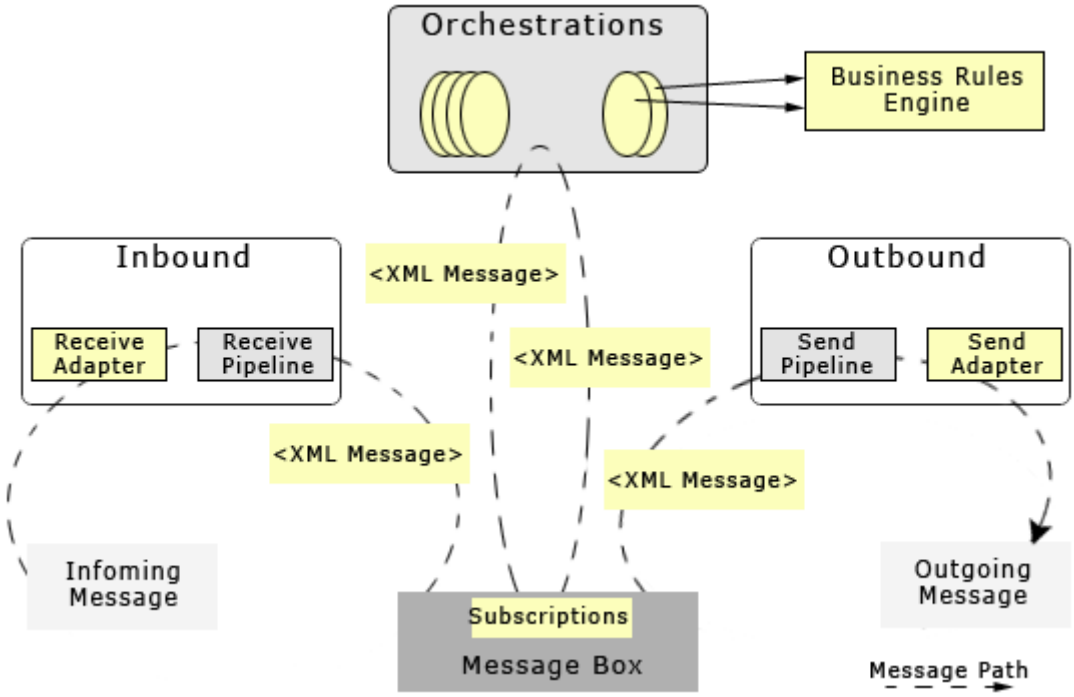
Tehnologije koje mogu biti korišćene:

- Business Rules Engine - omogućava upravljanje složenim skupovima poslovnih pravila.
- Health and Activity Tracking (HAT) - alat namenjen diveloperima i administratorima za nadgledanje i upravljanje BizTalk 2006 Engine i orkestracije koja se izvršava.
- Enterprise Single Sign-on - obezbeđuje mogućnost mapiranja autentifikacionih informacija između Windows i ne-Windows sistema.
- Business Activity Monitoring (BAM) - obezbeđuje prikaz informacija o poslovnom procesu koji se izvršava. Informacije su namenjene poslovnim ljudima, tako da je i jezik koji se koristi ne-tehnički.
- Business Activity Services - obezbeđuju podešavanje i upravljanje veza sa poslovnim partnerima.

BizTalk Server 2006 može da primi poruku u XML, flat ili EDI (Electronic Document Interchange) obliku. Bez obzira na format ulazne poruke, BizTalk Engine uvek obrađuje poruku u XML formatu, što zahteva da se definiše šema za konverziju poruke u flat ili EDI formatu.

BizTalk Server 2006 Engine

Da bi omogućila korisnicima da kreiraju poslovni proces koji povezuje više aplikacija, BizTalk Server 2006 Engine komponenta mora da obezbedi dve osnovne stvari: način da implementira i specificira logiku vezanu za poslovni proces, kao i da obezbedi mehanizam za komunikaciju između aplikacija koje koristi poslovni proces. Slika prikazuje komponente koje omogućavaju da se to ostvari.



slika 54: Glavne komponente BizTalk Server 2006 Engine

Poruke se primaju preko Receive Adapter komponente. Različiti adapteri obezbeđuju različite komunikacione mehanizme, tako da poruka može biti dobijena preko Web servisa, čitanjem dokumenta, ili na neki drugi način. Primljena poruka se dalje obrađuje preko Receive Pipeline komponente. U okviru ove komponente mogu se nalaziti druge komponente, koje obavljaju poslove kao što su: konverzija poruke iz osnovnog formata u XML dokument, provera digitalnog potpisa, i slično. Posle toga se poruka isporučuje u bazu podataka, koja se naziva MessageBox, i koja je implemetirana korišćenjem SQL Server softvera.

Logika vezana za poslovni proces je implementirana kao jedna ili više orkestracija, koje sadrže izvršni kod. Orkestracije kreiraju poslovni analitičari ili diveloperi, pri čemu koriste odgovarajući alat da grafički organizuju grupu oblika, kako bi mogli da izraze uslove, petlje, kao i druge oblike ponašanja. Orkestracije mogu opciono da koriste Business Rules Engine, koja obezbeđuje jednostavniji i lakši način da se izraze kompleksniji skupovi pravila vezani za poslovni proces.

Svaka orkestracija kreira Subscriptions, preko koje pokazuje koju vrstu poruke želi da primi. Kada poruka stigne u MessageBox, ona se šalje u ciljnu orkestraciju, koja izvršava potrebnu akciju. Resultat ove obrade uobičajeno je druga poruka, i ona se pamti u MessageBox bazi. U povratku, ova poruka se obrađuje od Send Papeline, koja je može konvertovati iz internog XML formata, koji koristi BizTalk Server 2006, u format koji odgovara odredištu, zatim se poruci može dodati digitalni potpis, i uraditi druga podešavanja. Send Adapter koristeći odgovarajući mehanizam šalje poruku odgovarajućoj aplikaciji.

Potpuno rešenje, koje se kreira pomoću BizTalk Server 2006 Engine komponente, naziva se BizTalk aplikacija, i ona može sadržati različite delove (koji se zovu i artifakti): orkestracije, pipelines, šeme poruka.

Različiti ljudi mogu izvršavati različite funkcije sa BizTalk Server 2006 Engine. Na primer, poslovni analitičar može definisati poslovna pravila i ponašanje koji čine poslovni proces. Takođe, poslovni analitičar može odrediti tok izvršavanja poslovnog procesa, tako što definiše koje informacije treba slati u svaku od uključenih aplikacija i kako se jedan poslovni dokument mapira u drugi. Pošto poslovni analitičar definiše poslovni proces, diveloper može kreirati BizTalk aplikaciju koja ga implementira. Ovo uključuje definisanje XML šema za poslovne dokumente koji će biti korišćeni, detaljno specificiranje mapiranja, kao i kreiranje orkestracija neophodnih da bi se implementirao proces. Administrator podešava komunikaciju između delova, isporučuje aplikaciju i izvršava druge poslove. Sve tri uloge - poslovni analitičar, diveloper i administrator, su neophodni da bi mogle da se kreiraju i održavaju BizTalk Server 2006 aplikacije.

Povezivanje sistema: Adapteri

Efektivna razmena poruka između različitog softvera koji se izvršava na različitim mašinama je najvažniji preduslov da bi se obavilo povezivanje aplikacija. Da bi ispunio taj zahtev, BizTalk Server 2006 Engine podržava više vrsta protokola i formata poruka. Inače, BizTalk Server 2006 Engine interno isključivo radi sa XML dokumentima, jer bez obzira na format u kojem se nalazi poruka koja stiže, ona se uvek prebacuje u XML dokument. Slično, ako primalac dokumenta ne može da prihvati dokument u XML obliku, Engine konvertuje dokument u potreban format.

Adapter predstavlja implementaciju komunikacionog mehanizma, na primer određenog protokola. Diveloper može odrediti koji će se adapter koristiti u određenoj situaciji.

BizTalk Server 2006 uključuje sledeće adaptere:

- Web Services Adapter: omogućava slanje i primanje poruka korišćenjem SOAP poruke, koja se prenosi preko HTTP protokola. Predstavlja osnovni adapter za komunikaciju sa Web servisima.

- File Adapter: omogućava čitanje i pisanje u fajlovi na Windows operativnom sistemu.

- HTTP Adapter: omogućava slanje i primanje informacija korišćenjem HTTP protokola.

- MSMQ Adapter: omogućava slanje i primanje poruka korišćenjem Microsoft Message Queuing (MSMQ).

- MSMQT Adapter: omogućava slanje i primanje poruka korišćenjem *BizTalk Message Queuing (MSMQT)*.

- WebSphere MQ Adapter: omogućava slanje i primanje poruka korišćenjem IBM WebSphere MQ (ranije poznatog kao MQSeries).

- SMTP Adapter: omogućava slanje poruka korišćenjem SMTP.

- POP3 Adapter: omogućava primanje email poruka i zakačenih fajlova korišćenjem POP3 protokola.

- Windows SharePoint Services (WSS) Adapter: dozvoljava pristup i publikovanje dokumenata koji se nalaze u SharePoint bibliotekama dokumenata.

- SQL Adapter: omogućava čitanje i pisanje informacija u SQL Server bazu podataka.

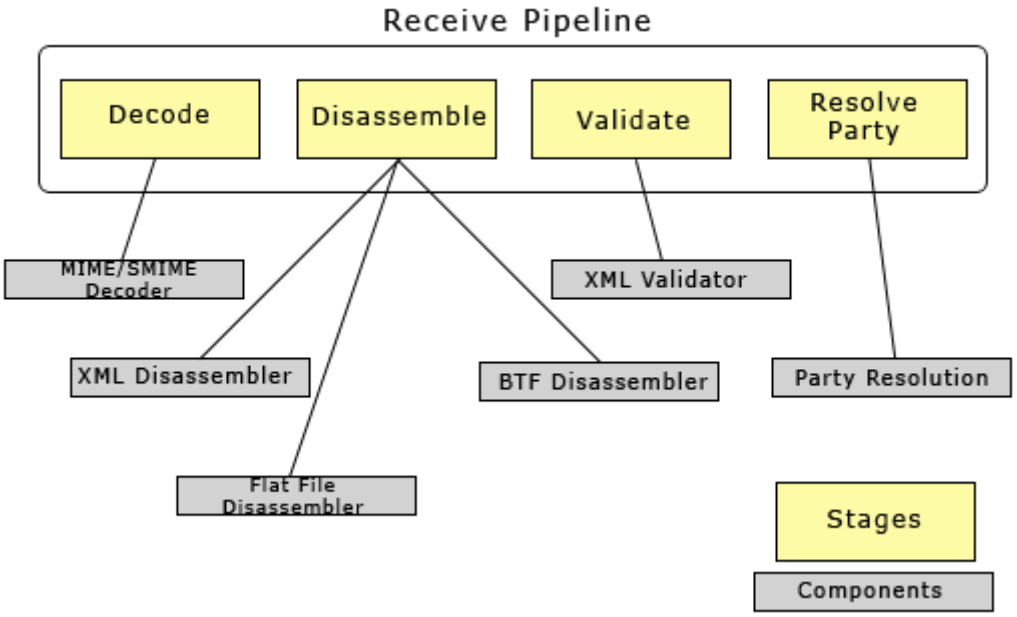
Takođe su dostupni i adapteri za komunikaciju sa često korišćenim aplikacijama od drugih proizvođača, na primer za: Siebel, PeopleSoft, Oracle aplikacije i Oracle baze podataka, JD Edwards OneWorld and EnterpriseOne, TIBCO Rendezvous and Enterprise Messaging Service, Amdocs Clarify. Osim Microsoft kompanije, i njeni partneri takođe su napravili različite adaptere, uključujući i adaptere za Electronic Data Interchange (EDI) i druge.

Ukoliko među postojećim adapterima ne postoji odgovarajući, postoji mogućnost kreiranja novog adaptera. Ovo se postiže korišćenjem Adapter Framework komponente.

Obrada poruka: Pipelines

Aplikacije koje čine poslovni proces komuniciraju međusobno razmenjujući različite vrste dokumenata: otpremnice, fakture, i drugo. Da bi BizTalk Server 2006 aplikacija mogla da izvrši poslovni proces, ona mora biti u stanju da radi sa porukama, koje sadrže ove dokumente.

Obrada poruka se sastoji od više koraka, i izvršava se od strane Pipeline komponente. Dolazeće poruke se obrađuju pomoću Receive Pipeline, a odlazeće pomoću Send pipeline. Pipeline je sastoji od određenog broja faza, koje sadrže jednu ili više .NET ili COM komponenti. BizTalk Server 2006 Engine obezbeđuje nekoliko standardnih komponenti, koji pokrivaju najčešće situacije. Diveloperi mogu kreirati nove komponente kao rešenja za situacije koje nisu podržane standardnim komponentama. Takođe, mogu kreirati i nove Pipelines korišćenjem Pipeline Designer komponente.



slika 55: BizTalk Server 2006 Receive Pipeline

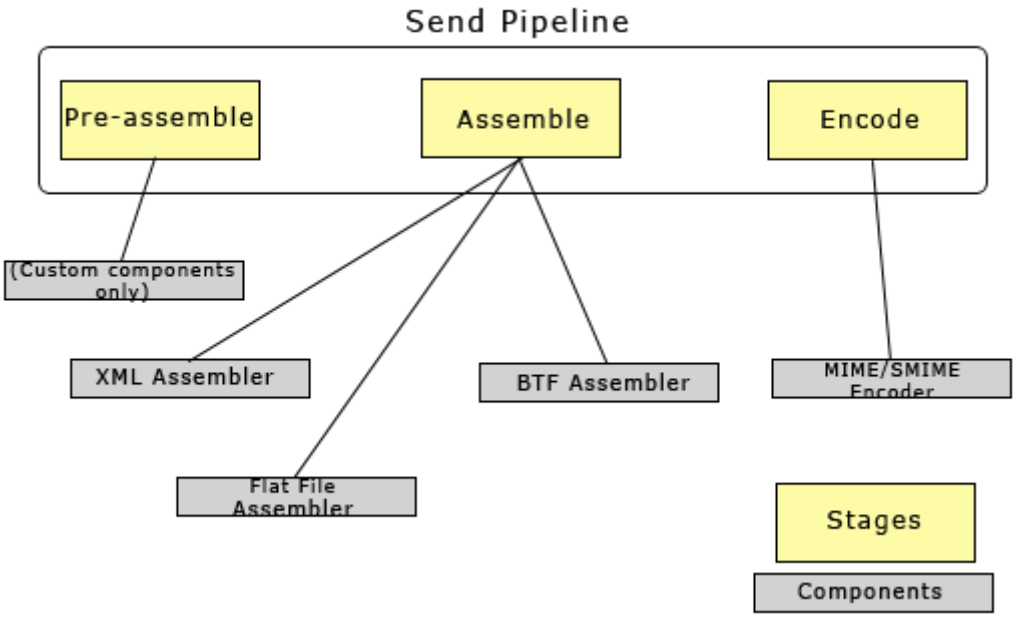
Slika ilustruje faze kod Receive Pipeline i standardne komponente koje se koriste u pojednim fazama:

- Decode: BizTalk Server 2006 obezbeđuje jednu standardnu komponentu za ovu fazu, *MIME/SMIME Decoder*. Ova komponenta upravlja porukama i prikačenim fajlovima u MIME ili Secure MIME (S/MIME) formatu. Komponenta konvertuje obe vrste poruka u XML, i može da dekriptuje S/MIME poruku i proveri digitalni potpis.

- Disassemble: Obezbeđene su tri standardne komponente. Flat File Disassembler komponenta konvertuje standardne fajlove u XML dokumente. *XML Disassembler* parsira dolazeće poruke, koje se nalaze u XML formatu. *XML Disassembler* komponta se retko koristi. Njena svrha je da prihvati poruke koje je definisao BizTalk Framework (BTF), koji je implementiran u BizTalk Server 2000.

- Validate: *XML Validator* komponenta se koristi u ovoj fazi. Njena svrha je da proveri usklađenost XML dokumenta, koji je kreiran u okviru Disassemble faze, sa specificiranom šemom.

- Resolve Party: *Party Resolution* komponenta se izvršava u ovoj fazi, i ona pokušava da odredi identitet pošiljaoca poruke. Ako je poruka digitalno potpisana, potpis se koristi da bi se pronašao Windows Identity u BizTalk Server 2006 konfiguracionoj bazi. Ako poruka sadrži SID od Windows korisnika, taj identitet se koristi. Ako nijedan mehanizam autentifikacije ne uspe, pošiljalac poruke dobija anonimni identitet.



slika 56: BizTalk Server 2006 Send Pipeline

Slika prikazuje faze i sadržane standardne komponente u Send Pipeline:

- Pre-assemble: Ne sadrži standardne komponente. Mogu se napraviti potrebne komponente, ukoliko su neophodne.

- Assemble: Sadrži komponente kao i Disassemble faza kod Receive Pipeline.

- Encode: BizTalk Server 2006 defin iše samo jednu standardnu komponentu za ovu fazu i to *MIME/SMIME Encoder*. Ova komponente pakuje odlazeće poruke u MIME ili S/MIME format. Ako je S/MIME korišćen, poruka može biti digitalno potpisana ili kriptovana.

BizTalk server integracija sa Web servisima

BizTalk Server 2006 prirodno podržava postojeće Web servis standarde. Ova podrška omogućava diveloperima da upotrebljavaju Web servise i kao deo poslovnog procesa, i da publikuju poslovne procese kao Web servise, koji mogu biti korišćeni od strane eksternih aplikacija. Šeme koje kreira BizTalk Server su kompatibilne sa W3C standardima, što omogućava BizTalk serveru da sarađuje sa više različitih tipova eksternih sistema preko postojećih Web servis standarda.

BizTalk Server 2006 omogućava pozivanje eksternih Web servisa iz postojećeg poslovnog procesa, definisanog pomoću orkestracije. Da bi se koristio Web servis iz jedne orkestracije, potrebno je da se:

1. kreira Web referenca - referenca se uobičajeno dodaje na WSDL dokument,

2. kreira Web port u okviru orkestracije - port može biti zahtev-odgovor (request-response) port, kao i samo zahtev (request) port,

3. kreiraju poruke - koje se prosleđuju do i od Web servisa,

4. transformiše poruka u odgovarajući format za slanje odgovora Web servisu,

5. postavi Send i Receive element, koji omogućavaju slanje i prijem zahtevanih informacija.

BizTalk Server orkestracija i BPEL

BizTalk elementi	**BPEL elementi**
Send shape	<invoke> ili <reply>
Receive shape	<receive>
Port	<partnerLinks>, <partnerLink>
Role Link	<partnerLinkType>, <role>
Message Assignment shape	<assign>, <copy>, <from>, <to>
Decide shape	<switch>, <case>, <otherwise>
Delay shape	<wait>
Listen shape	<pick>, <onMessage>, <onAlarm>
Parallel Actions shape	<flow>
Loop shape	<while>
Scope shape	<scope>
Throw Exception shape	<throw>
Compensate shape	<compensate>
Suspend shape	Nije podržano

Call Orchestration shape	Nije podržano
Start Orchestration shape	Nije podržano
Call Rules shape	Nije podržano
Transform shape	Nije podržano
Terminate shape	<terminate>
Compensation Block	<compensationHandler>
Exception Handler	<faultHandler>, <catch>, <catchAll>
Correlation	<correlation>, <correlations>, <correlationSets>, <correlationSet>

Human Workflow Services

Bez obzira na nivo razvoja softvera, najvažnije interakcije u jednoj organizaciji se dešavaju između ljudi. Automatizacija ove vrste komunikacije može poboljšati izvršavanje poslovnih procesa. Veoma često, poslovni proces zahteva ulaz od strane poslovnog korisnika. Primeri su: ulaz podataka, upravljanje izuzecima, startovanje procesa, odabir odluke.

Human Workflow Services (HWS) predstavlja skup mogućnosti u okviru BizTalk servera, koji obezbeđuju da ad-hoc i nestruktuirani dijagrami toka budu dostupni korisnicima preko klijentskih aplikacija.

Ljudi, učesnici u poslovnom procesu, se nazivaju aktori (Actors). Svaki dijagram toka, koji se naziva još i dijagram aktivnosti, se sastoji od jedne ili više akcija, koje su implementirane kao orkestracije. Akcije sadrže jedan ili više zadataka, koji se izvršavaju od strane aktora. Korisnički interfejs za dijagram toka, može biti bilo koji klijent, koji može da uspostavi vezu sa HWS pomoću Web servisa, kao i pomoću Microsoft Office alata, kao što su Word, Outlook, InfoPath.

BizTalk Server 2006 i Windows Workflow Foundation

Windows Workflow Foundation je deo .NET Framework 3.0, koji će se isporučivati kao sastavni deo Vista (ili nekog kasnijeg) operativnog sistema. Namenjen je diveloperima softvera, a ne poslovnim analitičarima. Njegov osnovni cilj je da obezbedi workflow funkcionalnost u okviru jedne aplikacije, što se postiže preko workflow mašine, standardnog skupa aktivnosti za definisanje dijagrama toka, i grafičkog dizajnera, čija je namena da pomogne diveloperima da kreiraju dijagrame toka.

Osnovni cilj BizTalk servera je da obezbedi workflow funkcionalnost pri korišćenju više aplikacija, odnosno da obezbedi integraciju različitih aplikacija, zbog čega veće aplikacije i poslovni procesi, još uvek zahtevaju korišćenje BizTalk servera. U ovom momentu, BizTalk Server 2006, još uvek predstavlja glavnu Microsoft tehnologiju za obezbeđivanje workflow funkcionalnosti, odnosno izvršavanje poslovnih procesa.

8.3 Ostali korišćeni alati i tehnologije

Za razvoj novih aplikacija korišćene su najnovije Microsoft tehnologije i proizvodi.

Microsoft .NET Framework

Microsoft .NET Framework obezbeđuje alate i tehnologije za brzi razvoj različitih vrsta aplikacija, koje se mogu izvršavati na različitim vrstama uređaja.

Dva osnovna dela .NET Framework-a čine:

- CLR (Common Language Runtime)
  CLR predstavlja interfejs između aplikacijskog koda i operativnog sistema, odnosno, obezbeđuje okruženje za izvršavanje aplikacija: JIT kompajliranje; upravljanje sigurnošću; učitavanje klasa iz .NET biblioteke klasa; obezbeđuje debug koda i upravljanje greškama koje se mogu desiti prilikom izvršavanja; komunikaciju sa aplikacijama koje nisu kreirane korišćenjem .NET-a; programiranje preko više niti.
- .NET Class Library
  Sadrži više stotina klasa. Klase su grupisane, radi lakšeg korišćenja, u prostore imena (namespaces), koji sadrže funkcionalno srodne grupe klasa. Sve klase mogu biti upotrebljavane od strane bilo kog .NET programskog jezika. Takođe, biblioteka klasa je proširiva.

.NET aplikacije se ne izvršavaju na isti način kao tradicionalne aplikacije. Umesto da se kompajlira u izvršnu datoteku, koja sadrži mašinski kod, .NET programski kod se kompajlira u MSIL (Microsoft Intermediate Language) i smešta u datoteku koja se naziva assembly. U vreme izvršavanja, assembly se Just-in-Time kompajlira pomoću CLR-a u mašinski kod. Tokom izvršavanja, CLR obezbeđuje: upravljanje memorijom; provere bezbednosti tipa podataka i druge poslove vezane za aplikaciju u vreme njenog izvršavanja.

ASP.NET

ASP.NET tehnologija predstavlja programski okvir, zasnovan na .NET Framework-u, za razvoj Web aplikacija i Web servisa. ASP.NET forme obezbeđuju veoma lak i moćan način za izgradnju dinamičkih Web strana. Takođe, ASP.NET Web servisi obezbeđuju kreiranje delova distribuiranih Web aplikacija.

ASP.NET je sasvim nezavisan od programskog jezika. To znači, za kreiranje Web strana može se koristiti bilo koji jezik koji ima implementiran .NET kompajler, koji omogućava kompajliranje u MSIL.

ASP.NET klasama se može pristupiti preko System.Web prostora imena iz .NET Framework-a.

ASP.NET nudi i sledeće pogodnosti:

- odvojene su aplikaciona logika od prezentacionog sloja aplikacije;
- bogat skup serverskih kontrola, koje automatski generišu HTML kod prilagođen klijentu (browser);
- unapređeno upravljanje stanjem;
- izvršne datoteke su kompajlirane, tako da se izvršavaju brže od interpretiranih skriptova;
- ažuriranje uvedenih Web aplikacija obavlja se u hodu, bez restartovanja Web servera;

- pristup radnom okruženju .NET Framework, koje pojednostavljuje mnoge aspekte Web programiranja;
- mogućnost pravljenja novih serverskih kontrola na osnovu postojećih kontrola;
- ugrađenu bezbednost preko Windows servera ili pomoću drugih metoda provere identiteta korisnika i njegovih prava;
- integrisanost sa ADO.NET-om kako bi se obezbedio pristup bazi podataka i alati za dizajniranje baze podataka u okviru samog Visual Studio .NET-a;
- puna podrška za XML, CSS i druge Web standarde;
- ugrađene mogućnosti za keširanje često traženih Web strani na serveru, lokalizovanje sadržaja za konkretne jezike i kulturološke celine.

ADO.NET

ADO.NET predstavlja skup klasa koje obezbeđuju .NET aplikaciji da čita i menja podatke u bazi podataka, ili nekom drugom izvoru podataka. ADO.NET klasama se može pristupiti preko System.Data prostora imena iz .NET Framework-a. ADO.NET omogućava konzistentan pristup različitim tipovima izvora podataka, uključujući MS SQL Server, OLEDB baze podataka, nerelacione strukture (Exchange Server ili Index Server), i XML dokumente.

Programski jezici Visual C# .NET i Visual Basic .NET

C# .NET i Visual Basic .NET predstavljaju moderne, objektno-orijentisane jezike koji omogućavaju programerima da brzo kreiraju širok spektar aplikacija, od kreiranja različitih komponenti, do aplikacija sistemskog nivoa.

Svi .NET programski jezici koriste .NET Framework biblioteke klasa i CLR sistem za upravljanje. Objedinjene .NET klase obezbeđuju konzistentan (dosledan) metod za pristup bilo kojoj funkcionalnosti koju obezbeđuje platforma.

U većini situacija, svi .NET jezici se mogu veoma efikasno koristiti, tako da izbor programskog jezika zavisi samo od prethodnog iskustva.

Visual Studio .NET

Visual Studio .NET predstavlja kompletan skup razvojnih alata za dizajniranje, kreiranje i isporučivanje ASP.NET Web aplikacija, XML Web servisa, mobilnih i tradicionalnih klijentskih aplikacija. Obezbeđuje da svi .NET programski jezici koriste isto integrisano razvojno okruženje koje je tesno integrisano sa .NET Framework-om.

Visual Studio .NET je jedino razvojno okruženje u potpunosti izgrađeno za XML Web servise.

Microsoft Visio

Microsoft Visio je program koji pomaže da se kreiraju poslovni i tehnički dijagrami koji dokumentuju i organizuju kompleksne ideje, procese i sisteme. Sa Visio-om se mogu praviti blok dijagrami, planovi zgrada, električne instalacije, algoritmi, formulari, mape, mrežni dijagrami, organizacione tabele, konceptualni dijagrami, dijagrami baza podataka.

Microsoft SQL Server 2000

SQL Server 2000 je sistem za upravljanje relacionim bazama podataka (RDBMS). Napravljen je da odgovori potrebama različitih sistema, kako zahtevima za skladištenjem velike količine podataka sistema za obradu podataka ili komercijalnih Web sajtova, tako i zahtevima za jednostavno korišćenje od strane malih firmi ili pojedinaca.

Karakteristike SQL Server 2000:

- Jednostavno administriranje baze podataka, jer funkcije automatskog podešavanja i održavanja omogućavaju administratorima da se usredsrede na druge kritične zadatke;
- Visok nivo dostupnosti, uz pomoć slanja evidencije, rezervnih kopija na mreži i serverskih grupa za premošćavanje otkaza;
- Proširivost do 32 procesora i 64 GB RAM memorije;
- Visoka bezbednost u svakom mrežnom okruženju uz bezbednosne opcije zasnovane na ulogama i uz šifrovanje datoteka i mreže;
- XML podrška omogućava lako čuvanje podataka za upotrebu u Web aplikacijama.
- Mogućnost pretraživanja čitavog teksta;
- Integracija sa drugim serverskim sistemima (Windows Server, SharePoint Portal Server, Commerce Server, Content Management Server, BizTalk Server) ;
- Usluge transformisanja podataka pomoću procedura za isporučivanje, transformaciju i učitavanje podataka iz različitih izvora;
- Usluge analize obimnih i složenih kolekcija podataka uz pomoć višedimenzionalnog skladišta.

9. Primena modeliranja poslovnih procesa u razvoju aplikacija za turističko poslovanje

Profil naručioca – Turistička agencija (turoperator - TO) iz Beograda, osnovana 1990. godine, pruža pretežno usluge organizacije i prodaje aranžmana za letovanja i zimovanja u Srbiji, i drugim državama. Ostvaruje različite oblike poslovne saradnje sa velikim brojem drugih turističkih agencija (subagentima). Vlasnici agencije su izuzetno motivisani za primenu programskih rešenja i Internet servisa (E-mail, WWW) u okviru svakodnevnog poslovanja. Ovaj zaključak proističe iz činjenice da od osnivanja koriste integrisani, turistički informacioni sistem i da su Web aplikaciju uradili 1998. godine. Ponuda agencije je visoko pozicionirana na mašinama za pretraživanje, sa oko 800 dnevnih poseta u letnjim mesecima. Zahvaljujući Web aplikaciji, oko 50% putnika obezbeđuju posredstvom Interneta.

9.1 Specifikacija i analiza zahteva

9.1.1 Poslovni ciljevi i zahtevi

Osnovni ciljevi vlasnika su da se komunikacija sa subagentima unapredi efikasnijom, automatskom razmenom podataka o rezervacijama turističkih aranžmana. Osim toga, zahteva se i funkcionalno redizajniranje postojeće Web aplikacije, sa ciljem da se poveća njena efikasnost.

Svoje potrebe iskazali su u obliku sledećih zahteva:

1. Zadržati i nastaviti korišćenje postojećih programskih rešenja, koja su zasnovana na već proverenom, poslovnom modelu.
2. Kreirati rešenje, sa ciljem da se automatizuju određene funkcije rezervacije i prodaje aranžmana.
3. Omogućiti subagentima da, koristeći novo rešenje, izvrše rezervaciju putovanja i prevoza bez usmenih i pismenih kontakata sa turoperatorom, kao i da dobiju informacije o statusu zahteva za viziranje putnih isprava.

9.1.2 Opis poslovanja turističke agencije

9.1.2.1 Osnovni pojmovi vezani za turističko poslovanje

Turoperator je turistička agencija, sa licencom koja dozvoljava poslovanje vezano za:

- zakup turističkih kapaciteta;
- prodaju turističkih kapaciteta drugim turističkim agencijama, svojim subagentima;
- prodaju pojedinačnih aranžmana;
- organizaciju prevoza u okviru turističkog, ili poslovnog putovanja, a ne uključuje redovan, linijski prevoz.

Subagent je turistička agencija, sa licencom za prodaju pojedinačnih, turističkih aranžmana.

Turistički aranžman može da obuhvati usluge smeštaja, prevoza, transfera, viziranja pasoša, kao i druge vrste usluga, koje su povezane sa turističkim putovanjem.

Ugovarač je osoba koja sklapa ugovor sa agencijom i koja preuzima na sebe plaćanje ugovorenih usluga. Ugovarač i agencija sklapaju ugovor o putovanju.
Sastavni deo ugovora o putovanju je:

- ukupna cena;
- spisak putnika, s osnovnim podacima (prezime i ime, pol, datum rođenja, broj pasoša, adresa;
- datum ugovora;
- spisak usluga uključenih u cenu;
- način i dinamika plaćanja;
- broj noćenja;
- prva i poslednja usluga;
- opšti uslovi putovanja.

Putnik je osoba koja putuje i čije se ime nalazi na vaučeru, koji na osnovu ugovora o putovanju izdaje agencija. Na jednom vaučeru nalaze se imena svih putnika, koji putuju na osnovu jednog ugovora.

Vaučer sadrži sledeće podatke:

1. redni broj putnika
2. pol putnika
3. prezime i ime putnika
4. datum rođenja putnika
5. destinaciju
6. naziv hotela
7. vrstu smeštaja
8. način ishrane
9. broj noćenja
10. podatke o prvoj i poslednjoj usluzi koja se pruža na mestu smeštaja
11. način prevoza
12. sedišta, ukoliko putnik koristi organizovani prevoz i ukoliko je praksa agencije da ih unapred (po redosledu prijavljivanja za tu destinaciju i prevozno sredstvo) odredi…

Prodaja turističkog aranžmana

Obuhvata:

- pružanje usluga smeštaja;
- pružanje usluga prevoza;
- obezbeđenje putnih viza.

U zavisnosti od poslovne politike agencije, usluge se mogu prodavati u obliku "paket aranžmana", ili pojedinačno (samo smeštaj, samo prevoz, samo viziranje).

9.1.2.2 Rezervacija turističkog aranžmana

1. Osnovne usluge

Putnik najčešće, kao osnovnu uslugu bira smeštaj:

1. Hotel, ili objekat privatnog smeštaja (apartman, studio, soba);
2. Vrstu smeštaja (struktura sobe, apartmana);
3. Period boravka
   Moguće su sledeće situacije:
   - U periodima koje određuje Agencija (na primer: desetodnevni, sedmodnevni aranžman)

- Određen broj dana, različit od ranije određenih termina i perioda. Ovaj slučaj je moguć ukoliko u odabranom periodu postoji slobodan smeštaj. Ograničenje je mogućnost organizovanog prevoza, koji tada najčešće nije moguć. Takođe, ograničenje je i nemogućnost korišćenja grupne, turističke vize.

Putnici imaju mogućnost korišćenja organizovanog prevoza i obezbeđenja grupnih viza.

2. Dodatne usluge

Dopunske, ili dodatne usluge su usluge koje se pružaju uz turistički aranžman, a koje nisu uključene u osnovnu cenu aranžmana.
Putnik može odabrati neke od sledećih dodatnih usluga:

- Lično osiguranje;
- Doplatu za ishranu, koja nije uključena u cenu osnovnog aranžmana;
- Izlete i ekskurzije;
- Iznajmljivanje opreme (skijaška oprema, na primer).

3. Ostvarivanje prava na popuste

U određenim slučajevima, pojedini putnici mogu ostvariti pravo na određene popuste:

- Deca do određenog godišta (procenat popusta i godine dece zavise od uslova koje propisuje hotel i mogu se ostvariti samo u hotelskom smeštaju);
- Korišćenje dodatnog ležaja u dvokrevetnoj / trokrevetnoj sobi;
- U slučaju kada dete koristi isti ležaj s pratiocem;
- U slučaju kada se rezervacija i uplata izvrše do određenog roka, a pre početka putovanja.

4. Plaćanje usluga

Usluge turističke agencije najčešće se plaćaju:

- u gotovom;
- bezgotovinski, korišćenjem kreditnih kartica;
- bezgotovinski, korišćenjem čekova;
- korišćenjem bankarskih kredita.

Dinamika plaćanja je:

- u ukupnom iznosu i u ugovorenom roku;
- određeni procenat pri rezervaciji aranžmana, ostatak do početka putovanja;
- u određenom broju rata.

9.1.3 Postojeće aplikacije

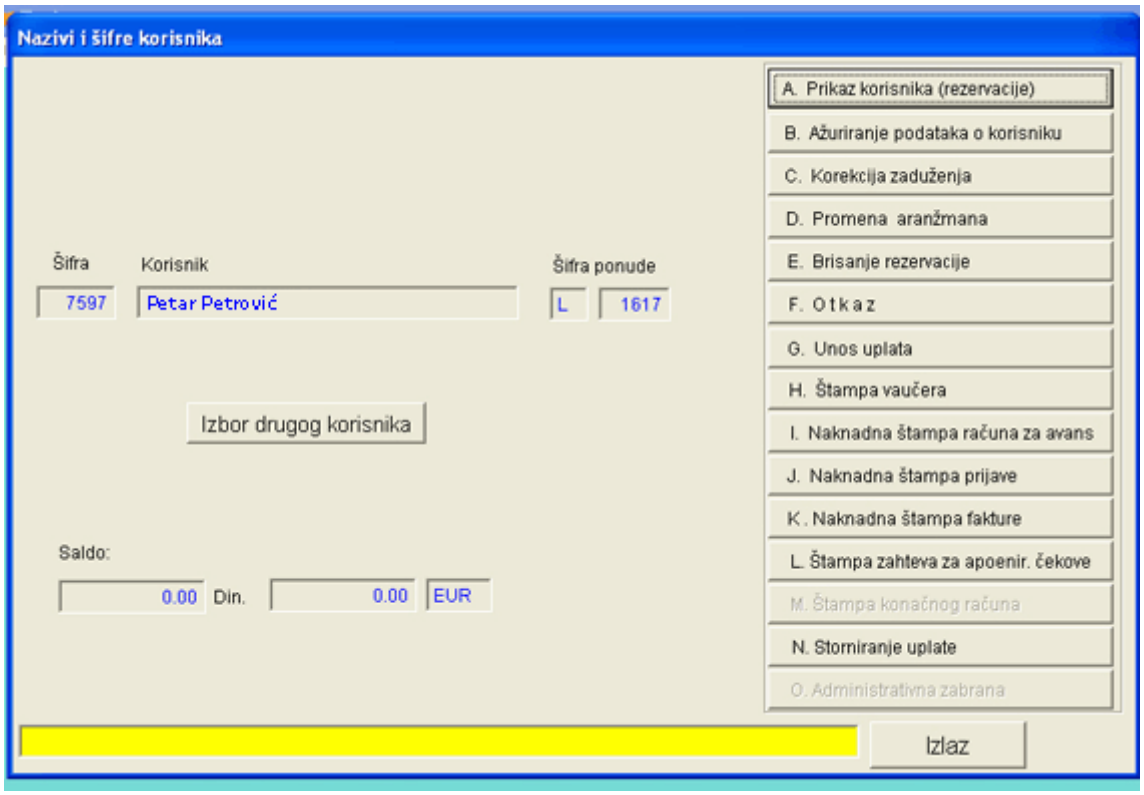
9.1.3.1 Agencijska aplikacija turoperatora

Agencijska aplikacija turoperatora podržava sve potrebne poslovne funkcije, uključujući:

- evidentiranje zakupa smeštajnih kapaciteta, prevoza i dodatnih usluga;
- rezervaciju i prodaju aranžmana, prevoza i dodatnih usluga;
- vođenje evidencije obezbeđenih kolektivnih viza i drugih putnih dokumenata;
- vođenje finansijskog poslovanja sa subagentima.

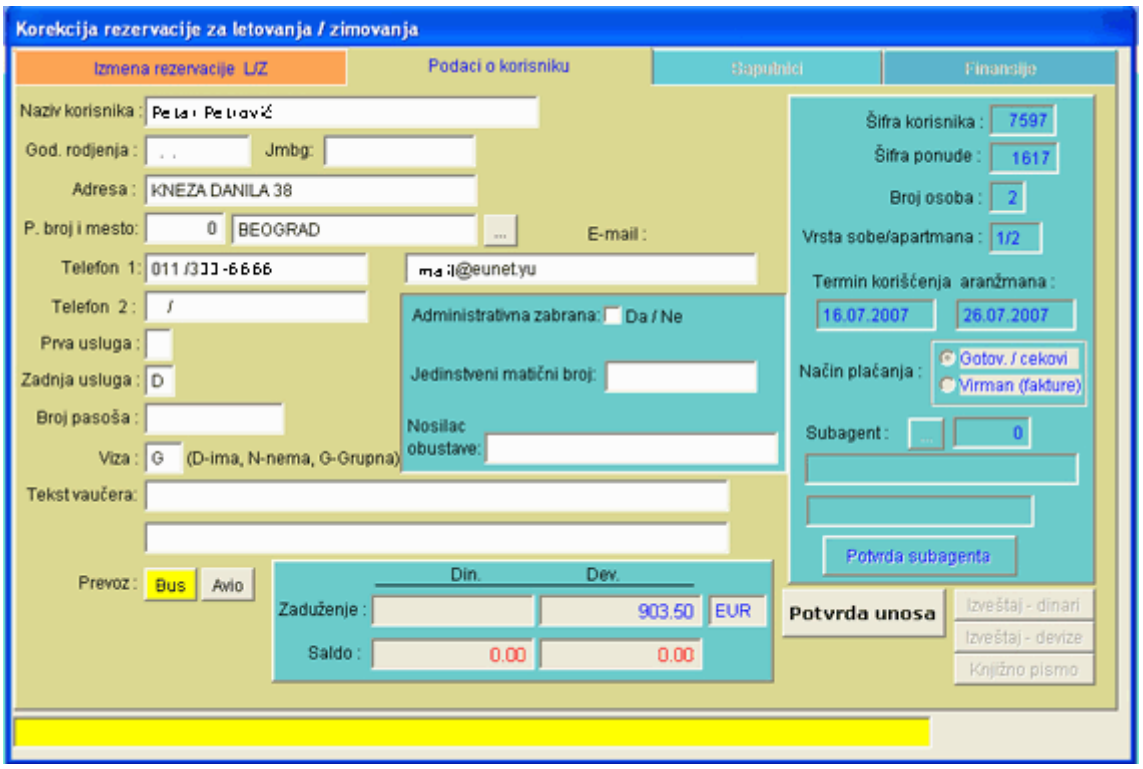
Aplikacija je rađena za mrežno Windows okruženje, a programski jezik je FoxPro. Realizovane funkcije zadovoljavaju sopstvene potrebe poslovanja turoperatora.
Centralni objekat aplikacije je Korisnik usluga. Podtipovi Korisnika usluga su Putnik i Ugovarač. Za Putnika se vezuju sve informacije neophodne za izvršenje ugovora o

putovanju (prevoz, smeštaj, viziranje, dodatne usluge i slično). Za Ugovarača se, dodatno, vezuju podaci o ugovoru o pružanju usluga i podaci o finansijskim transakcijama.
Ilustracije radi, na sledećoj slici prikazana je forma za izbor korisnika usluga, sa menijem za izbor raspoloživih funkcija.



slika 57: Osnovna forma za izbor korisnika i meni za izbor funkcija

Zakupljene smeštajne kapacitete, kao i druge usluge, turoperator prodaje korisnicima - putnicima neposredno, ili preko svojih subagenata.
Ukoliko aranžman, ili uslugu prodaje subagent, komunikacija između turoperatora i subagenta odvija se slanjem faks ili E-mail poruka. Na osnovu dobijenih podataka, turoperator kreira aranžman za putnike subagenata, koristeći sopstvenu aplikaciju.
Na sledećoj slici prikazana je forma za evidentiranje korisnika usluga. Forma uključuje i podatke o subagentu, ukoliko je aranžman prodao subagent.



slika 58: Podaci o korisniku usluga, aranžmanu i finansijskom stanju aranžmana

Nakon definisanja elemenata aranžmana i dodatnih usluga, vrše se potrebni finansijski obračuni, evidencije plaćanja, korekcije zaduženja i slično. Jedna od specifičnosti poslovanja turističke agencije je mogućnost da putnik, nakon pružene usluge, uloži reklamaciju na neku od pruženih usluga (smeštaj, na primer). U tom slučaju, moguć je međusobni dogovor i usaglašavanje novčanog iznosa, koji će agencija isplatiti putniku. Osim toga, moguće su i korekcije zaduženja po drugim osnovima.
Na sledećoj slici prikazana je forma za korekciju zaduženja korisnika usluga.

slika 59: Jedna od formi za evidentiranje finansijskih transakcija

Najveći nedostatak postojećeg aplikativnog rešenja je nepostojanje direktne komunikacije subagent – turoperator: ne postoji mogućnost da subagenti pristupe direktno bazi turoperatora, kako bi proverili raspoloživost smeštaja, prevoza i status viza. Samim tim, ne postoje ni mogućnosti direktne rezervacije i prodaje aranžmana i prevoza, pa se celokupna komunikacija subagent – turoperator odvija pismeno (razmenom E-mail ili faks poruka).

9.1.3.2 Web aplikacija turoperatora

Postojeća Web aplikacija obuhvata prezentacioni deo, vezan za ponudu aranžmana, kao i forme (form-mail obrasce) za prijavu i rezervisanje smeštaja i prevoza. Raspoloživi aranžmani detaljno su opisani i ilustrovani kvalitetnim fotografijama.

Broj poseta je zadovoljavajući (u proseku oko 800 poseta dnevno, sa rastućim trendom). Postojeće form-mailove za prijavu i rezervisanje aranžmana koriste budući putnici, ali i subagenti. Rezervacija se ne može izvršiti direktno, već to ručno radi turoperator, na osnovu podataka dobijenih putem elektronske poruke, koji je generisan popunjavanjem form-mail obrasca.

9.1.4 Predlog rešenja

Da bi se zadovoljili postavljeni ciljevi i zahtevi, potrebno je kreirati poslovni proces "Rezervacija aranžmana", koji će sadržati sve elemente neophodne za automatizaciju analiziranog turističkog poslovanja. Novokreirani poslovni proces će, kao svoje sastavne delove, podržati funkcionalnost postojećih aplikacija.

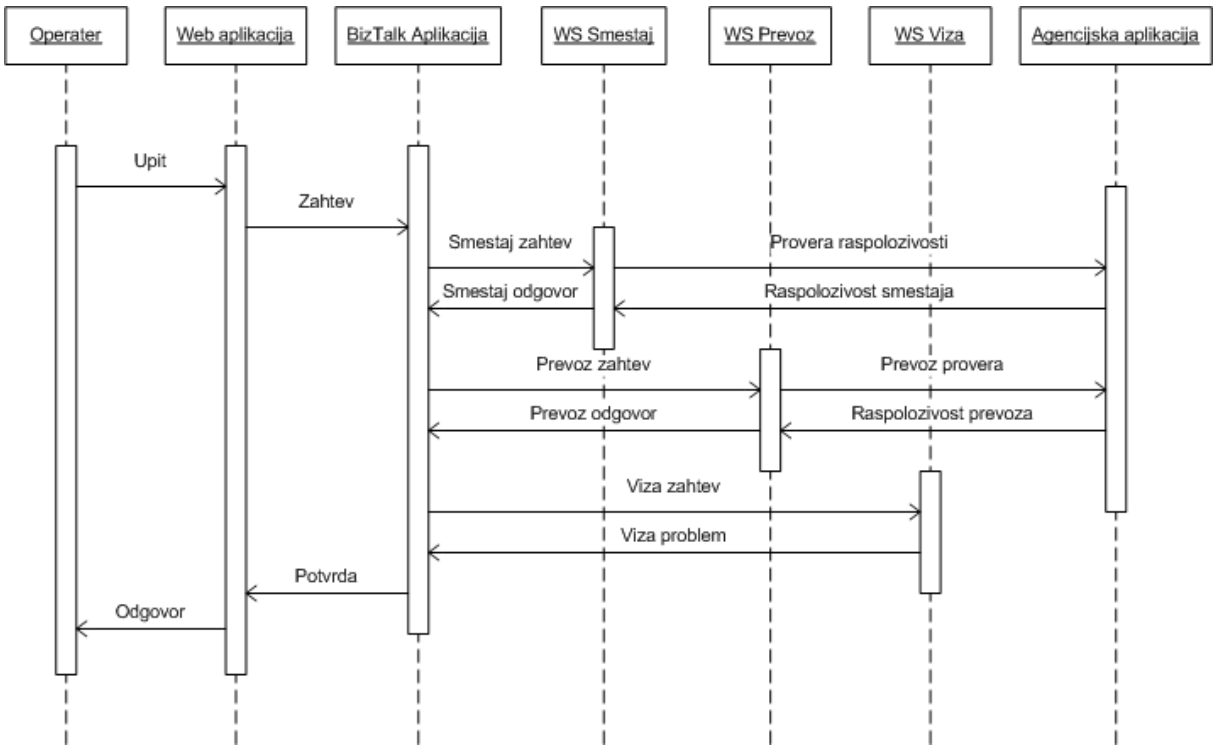
Kreiranje poslovnog procesa obuhvatiće:

- Model poslovnog procesa, grafički prikazan pomoću BPMN dijagrama.
- Obezbeđenje komunikacije između postojećih i novih aplikacija, kao i dodavanje mogućnosti automatizacije poslovnog procesa, izradom sledećih Web servisa:
    - Web servis koji će omogućiti rezervisanje, kao i otkaz smeštaja u nekoj od ponuđenih turističkih destinacija;
    - Web servis koji će omogućiti rezervisanje, kao i otkaz prevoza u neku od ponuđenih turističkih destinacija;
    - Web servis koji će omogućiti slanje obaveštenja o zahtevu za vizu, kao i obaveštenje o problemu u vezi sa prosleđenim zahtevom za vizu.
- Generisanje koda koji će omogućiti izvršavanje poslovnog procesa, a na osnovu kreiranog modela poslovnog procesa.
- Kreiranje nove Web aplikacije, portalskog tipa.
    - Web aplikacija treba da omogući jednostavan pregled postojećih smeštajnih kapaciteta, organizovanog prevoza i viziranja pasoša;
    - Web aplikacija treba da omogući da operater: izabere zahtevanu destinaciju, datum i period korišćenja aranžmana; upiše broj putnika koji će putovati, prevoz koji će koristiti, zahtev za vizu, kao i ostale neophodne podatke;
    - Web aplikacija treba da omogući iniciranje, odnosno startovanje instance poslovnog procesa;
    - Web aplikacija treba da bude dostupna isključivo operaterima turoperatora i operaterima subagenata.

- Kreiranje BizTalk Server aplikacije, koja će omogućiti izvršavanje i nadgledanje poslovnog procesa.

Izvršavanje poslovnog procesa "Rezervacija aranžmana" obavljaće se sledećim redosledom:

- Operater koristeći Web aplikaciju, a na osnovu zahteva putnika, bira podatke vezane za rezervaciju smeštaja (destinaciju, hotel, period, vrstu smeštaja i broj potrebnih soba), rezervaciju prevoza (destinaciju, datum polaska, datum povratka, ukupan broj sedišta), i viziranje pasoša.
- Web aplikacija kreira odgovarajuće XML poruke i šalje ih BizTalk procesnoj mašini.
- BizTalk aplikacija, na osnovu pristiglih poruka, inicira kreiranje poslovnog procesa.
- Poslovni proces poziva Web servis za rezervaciju smeštaja, koji u postojećoj bazi podataka izvršava rezervaciju, i vraća podatak o raspoloživosti zahtevanog smeštaja.
- Ukoliko postoji zahtev, poslovni proces poziva Web servis za rezervaciju prevoza, koji u postojećoj bazi podataka izvršava rezervaciju, i vraća podatak o obavljenog rezervaciji prevoza.
- Ako je potrebno da se obezbedi putna viza, poslovni proces poziva Web servis za viziranje pasoša, koji šalje informaciju o zahtevu za vizu. Poslovni proces vraća povratnu poruku Web aplikaciji o izvršenim rezervacijama.

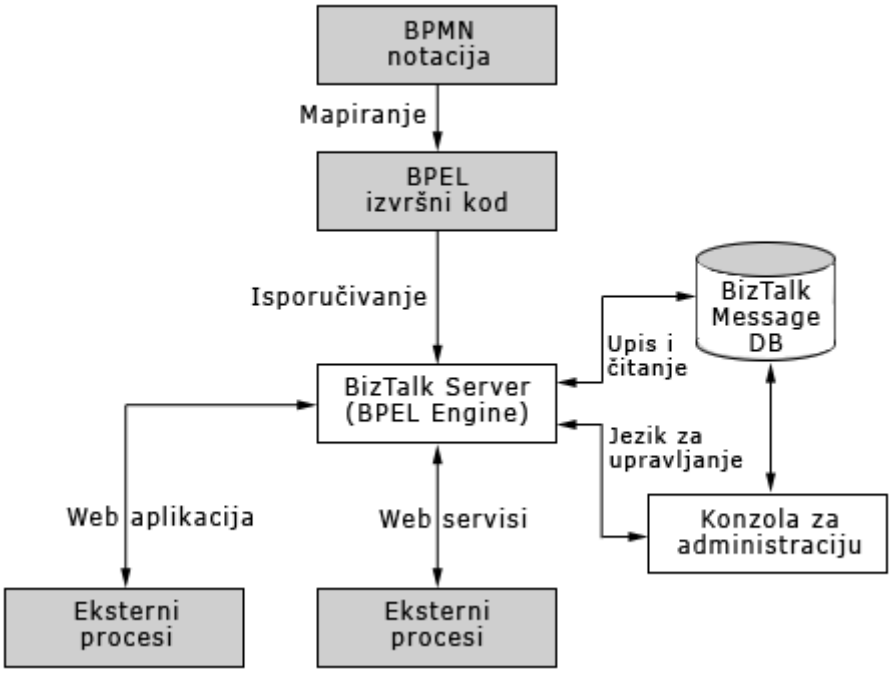


slika 60: Predlog rešenja predstavljen sistemskim dijagramom sekvenci

Za realizaciju predloženog rešenja koristiće se najnovije metode i dostupne tehnologije:

- za kreiranje i formalizaciju poslovnog procesa koristiće se BPMN dijagrami;
- kreirani BPMN dijagram biće mapiran u odgovarajući izvršni BPEL kod;
- za kreiranje potrebnih Web aplikacije i Web servisa koristiće se ASP.NET tehnologija;
- izvršavanje i nadgledanje poslovnog procesa obavljaće BizTalk server.

Način realizacije predloženog rešenja prikazan je na sledećoj slici.



slika 61: Način realizacije predloženog rešenja turističkog poslovanja

9.2 Razvoj rešenja

Razvoj rešenja je prošao kroz sledeće faze:

1. Dizajniranje BPMN modela;
2. Kreiranje Web servisa;
3. Generisanje BPEL na osnovu BPMN modela;
4. Razvoj Web portala "Rezervacioni sistem";
5. Kreiranje BizTalk Server 2006 aplikacije.

9.2.1 Dizajniranje BPMN modela

Na osnovu analize zahteva, pomoću BPMN modela, kreiran je poslovni proces "Rezervacija turističkog aranžmana".

Putnik, Subagent i Turoperator su prikazani pomoću Pool elemenata i predstavljaju učesnike u procesu.

Proces se startuje nakon prijema zahteva za obezbeđenje turističkog aranžmana, zbog čega je startovanje procesa predstavljeno Start Event elementom tipa Message.

Prijem zahteva se može sastojati od sledećih pojedinačnih zahteva: rezerviši smeštaj, rezerviši prevoz, i obezbedi vizu. Obrada pristiglih zahteva je predstavljena preko paralelnih Gateway elemenata, koji dozvoljavaju prolazak kroz svaku granu za koju je ispunjen postavljeni uslov.

Ukoliko stigne zahtev da je potrebno da se obezbedi smeštaj, obavlja se zadatak "Rezerviši smeštaj", predstavljen preko Task elementa, tipa Service, koji poziva operaciju "SmestajRezervacija" Web servisa "Smeštaj", čiji je svrha da rezerviše smeštaj na odabranoj destinaciji, u odabranom terminu, i po definisanoj strukturi soba. Web servis vraća poruku o uspešno ili neuspešno obavljenoj rezervaciji smeštaja. U slučaju potrebe da se otkaže već rezervisani smeštaj, obavlja se zadatak "Otkaži smeštaj", predstavljen preko Task elementa, tipa Service, koji poziva operaciju "SmestajOtkaz" Web servisa "Smeštaj". Ovaj zadatak se obavlja kao kompenzaciona aktivnost od prethodno obavljenog zadatka za rezervaciju smeštaja.

Ukoliko stigne zahtev da je potrebno da se obezbedi prevoz, obavlja se zadatak "Rezerviši prevoz", predstavljen preko Task elementa, tipa Service, koji poziva operaciju "PrevozRezervacija" Web servisa "Prevoz", čiji je svrha da rezerviše prevoz za odabranu destinaciju, u odabranom terminu, i za određeni broj osoba. Web servis vraća poruku o uspešno ili neuspešno obavljenoj rezervaciji prevoza. U slučaju potrebe da se otkaže već rezervisani prevoz, obavlja se zadatak "Otkaži prevoz", predstavljen preko Task elementa, tipa Service, koji poziva operaciju "PrevozOtkaz" Web servisa "Prevoz". Ovaj zadatak se obavlja kao kompenzaciona aktivnost od prethodno obavljenog zadatka za rezervaciju prevoza.

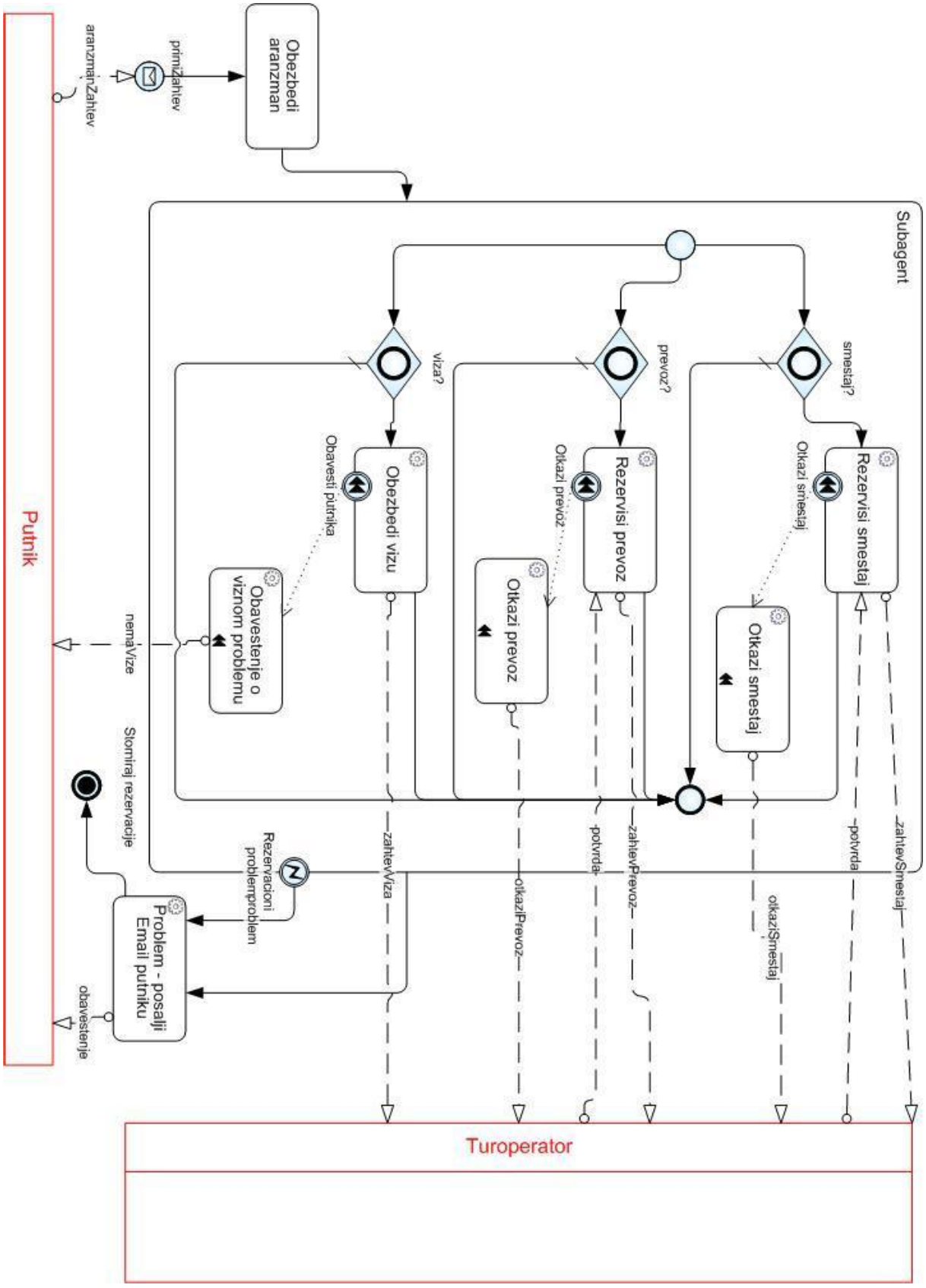
Zadatak "Obezbedi vizu" izvršava se ukoliko stigne zahtev da je potrebno obezbeđenje vize. Zadatak je prikazan pomoću Task elementa, Service tipa. Zadatak poziva operaciju "VizaRezervacija" Web servisa "Viza", koji treba da prosledi informaciju o zahtevu da se obezbedi viza. Ako se desi neki problem kod obezbeđenja vize, obavlja se zadatak

"Obaveštenje o viznom problemu", koji poziva operaciju "VizaProblem" Web servisa "Viza". Zadatak se obavlja kao kompenzaciona aktivnost.

Komunikacija Task elemenata, Service tipa, sa Web servisima je predstavljena preko objekata veza tipa Message Flow, koji se prikazuju isprekidanom linijom sa nepopunjenom strelicom. Ovi elementi se koriste da prikažu tok poruka između dva odvojena učesnika u procesu.

Ukoliko se desi neki izuzetak vezan za rezervaciju aranžmana, podiže se događaj, predstavljen pomoću Exception Event elementa. Događaj prosleđuje kontrolu zadatku "Problem - pošalji Email putniku", koji poziva operaciju "EmailInfo" Web servisa "EmailNote", koji šalje elektronsku poruku o postojanju problema.

BPMN model poslovnog procesa "Rezervacija turističkog aranžmana", kreiran korišćenjem ITP Process Modeler za Microsoft Visio, predstavljen je na sledećoj slici.



slika 62: BPMN dijagram za proces turističkog poslovanja

9.2.2 Kreiranje Web servisa

Za obezbeđenje komunikacije između postojećih i novih aplikacija, kao i dodavanje mogućnosti automatizacije poslovnog procesa, izgrađeno je nekoliko Web servisa. Opisi Web servisa su dostupni preko WSDL. WSDL specificira operacije i portove koje Web servis sadrži, poruke koje treba da prihvati, kao i tipove podataka.

9.2.2.1 Web servis "Smeštaj"

Web servis "Smeštaj" omogućava rezervisanje, kao i otkaz smeštaja u nekoj od ponuđenih turističkih destinacija, što je definisano preko WSDL elementa *portType*.

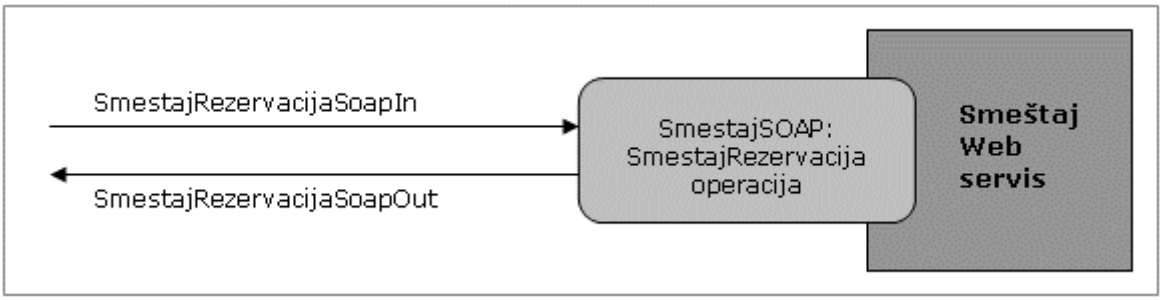
```
...
  <wsdl:portType name="SmestajSoap">
    <wsdl:operation name="SmestajRezervacija">
      <wsdl:documentation
xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/">Web metod rezerviše smeštaj
u određenom hotelu, u određenom periodu, za određeni broj putnika, koji će
u hotelu koristiti zasebne ležajeve.</wsdl:documentation>
      <wsdl:input message="tns:SmestajRezervacijaSoapIn" />
      <wsdl:output message="tns:SmestajRezervacijaSoapOut" />
    </wsdl:operation>
    <wsdl:operation name="SmestajOtkaz">
      <wsdl:documentation
xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/">Web metod Otkaži (storniraj)
smeštaj ima zadatak da otkaže (stornira) rezervisani
smeštaj.</wsdl:documentation>
      <wsdl:input message="tns:SmestajOtkazSoapIn" />
      <wsdl:output message="tns:SmestajOtkazSoapOut" />
    </wsdl:operation>
  </wsdl:portType>
...
```

portType element se sastoji od dve operacije: SmestajRezervacija i SmestajOtkaz.

SmestajRezervacija operacija

Korišćenje Web servisa "Smeštaj" za rezervaciju:

1. Koristeći Web aplikaciju, operater bira: destinaciju, hotel, period, vrstu smeštaja i broj potrebnih soba;
2. Web aplikacija kreira XML dokument i šalje ga na odredište, gde će ga čitati BizTalk server;
3. BizTalk server čita pristigli XML dokument i poziva na izvršavanje Web servis "Smeštaj";
4. Operacija "SmestajRezervacija" od Web servisa "Smeštaj" izvršava rezervaciju zahtevanog smeštaja;
5. Operacija "SmestajRezervacija" od Web servisa "Smeštaj" vraća podatak o raspoloživosti zahtevanog smeštaja.



slika 63: SmestajRezervacija operacija

SmeštajRezervacija je sinhrona request/response operacija, koja za zadate ulazne podatke (subagent, broj ugovora, destinacija, hotel, datum početka korišćenja smeštaja, trajanje korišćenja smeštaja, struktura soba) vraća informaciju o mogućnosti rezervacije smeštaja. Input i output poruke su u WSDL dokumentu definisane na sledeći način:

```
...
<wsdl:message name="SmestajRezervacijaSoapIn">
  <wsdl:part name="parameters" element="tns:SmestajRezervacija" />
</wsdl:message>

<wsdl:message name="SmestajRezervacijaSoapOut">
 <wsdl:part name="parameters" element="tns:SmestajRezervacijaResponse" />
</wsdl:message>
...
```

Tipovi ulazne i izlazne poruke su definisani preko *types* elementa.

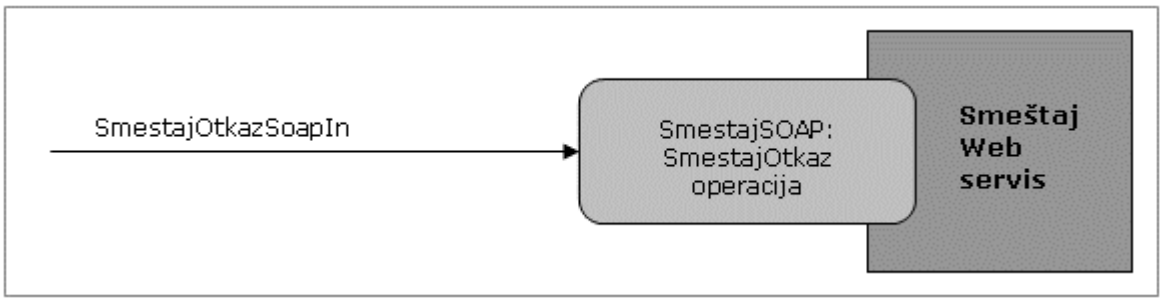
```
...

  <s:element name="SmestajRezervacija">
        <s:complexType>
          <s:sequence>
            <s:element minOccurs="1" maxOccurs="1" name="subagent"
type="s:int" />
            <s:element minOccurs="0" maxOccurs="1" name="brojUgovora"
type="s:string" />
            <s:element minOccurs="1" maxOccurs="1" name="destinacija"
type="s:int" />
            <s:element minOccurs="1" maxOccurs="1" name="hotel"
type="s:int" />
            <s:element minOccurs="1" maxOccurs="1" name="datum"
type="s:dateTime" />
            <s:element minOccurs="1" maxOccurs="1" name="trajanje"
type="s:int" />
            <s:element minOccurs="0" maxOccurs="1" name="strukturaSoba">
              <s:complexType>
                <s:sequence>
                  <s:element ref="s:schema" />
                  <s:any />
                </s:sequence>
              </s:complexType>
            </s:element>
          </s:sequence>
```

```
        </s:complexType>
      </s:element>
      <s:element name="SmestajRezervacijaResponse">
        <s:complexType>
          <s:sequence>
            <s:element minOccurs="1" maxOccurs="1"
name="SmestajRezervacijaResult" type="s:boolean" />
          </s:sequence>
        </s:complexType>
      </s:element>
...
```

SmestajOtkaz operacija

Korišćenje Web servisa "Smeštaj" za otkaz smeštaja:

1. Koristeći Web aplikaciju, operater bira broj ugovora;
2. Web aplikacija kreira XML dokument i šalje ga na odredište, gde će ga čitati BizTalk server;
3. BizTalk server čita pristigli XML dokument i poziva na izvršavanje Web servis "Smeštaj";
4. Operacija "SmestajOtkaz" od Web servisa "Smeštaj" stornira ranije kreiranu rezervaciju smeštaja.



slika 64: SmestajOtkaz operacija

SmeštajOtkaz je request operacija, koja sadrži samo input poruku. Operacija za zadate ulazne podatke (subagent, broj ugovora) stornira ranije kreiranu rezervaciju smeštaja. Input i output poruke su u WSDL dokumentu definisane na sledeći način:

```
...
  <wsdl:message name="SmestajOtkazSoapIn">
    <wsdl:part name="parameters" element="tns:SmestajOtkaz" />
  </wsdl:message>
  <wsdl:message name="SmestajOtkazSoapOut">
    <wsdl:part name="parameters" element="tns:SmestajOtkazResponse" />
  </wsdl:message>
...
```

Tip ulazne poruke je definisan preko types elementa.

```
...
```

```
<s:element name="SmestajOtkaz">
        <s:complexType>
          <s:sequence>
            <s:element minOccurs="1" maxOccurs="1" name="subagent"
type="s:int" />
            <s:element minOccurs="0" maxOccurs="1" name="brojUgovora"
type="s:string" />
          </s:sequence>
        </s:complexType>
      </s:element>
…
```

9.2.2.2 Web servis "Prevoz"

Web servis "Prevoz" omogućava rezervisanje, kao i otkaz prevoza u neku od ponuđenih turističkih destinacija, što je definisano preko WSDL elementa *portType*.

```
...

  <wsdl:portType name="PrevozSoap">
    <wsdl:operation name="PrevozRezervacija">
      <wsdl:documentation
xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/">Web metod rezerviše prevoz u
određenom autobusu, sa određenim datumom odlaska i sa određenim datumom
povratka, za određeni broj putnika, koji će u autobusu koristiti zasebna
sedišta.</wsdl:documentation>
      <wsdl:input message="tns:PrevozRezervacijaSoapIn" />
      <wsdl:output message="tns:PrevozRezervacijaSoapOut" />
    </wsdl:operation>
    <wsdl:operation name="PrevozOtkaz">
      <wsdl:documentation
xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/">Web metod Otkaži (storniraj)
prevoz ima zadatak da otkaže (stornira) rezervisana
sedišta.</wsdl:documentation>
      <wsdl:input message="tns:PrevozOtkazSoapIn" />
      <wsdl:output message="tns:PrevozOtkazSoapOut" />
    </wsdl:operation>
  </wsdl:portType>

...
```

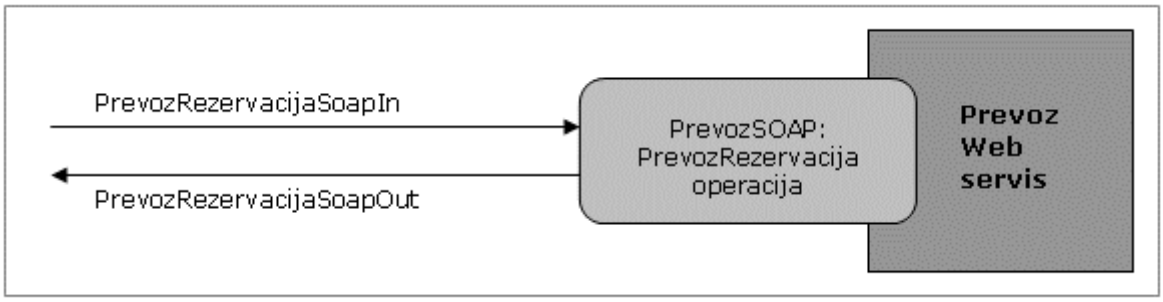
portType element se sastoji od dve operacije: PrevozRezervacija i PrevozOtkaz.

PrevozRezervacija operacija

Korišćenje Web servisa "Prevoz" za rezervaciju:

1. Koristeći Web aplikaciju, operater bira: destinaciju, datum polaska, datum povratka, ukupan broj sedišta;
2. Web aplikacija kreira XML dokument i šalje ga na odredište, gde će ga čitati BizTalk server;

3. BizTalk server čita pristigli XML dokument i poziva na izvršavanje Web servis "Prevoz";
4. Operacija "PrevozRezervacija" od Web servisa "Prevoz" izvršava rezervaciju prevoza;
5. Operacija "PrevozRezervacija" od Web servisa "Prevoz" vraća podatak o rezervaciji zahtevanog prevoza.



slika 65: PrevozRezervacija operacija

PrevozRezervacija je sinhrona request/response operacija, koja za zadate ulazne podatke (subagent, broj ugovora, destinacija, datum odlaska, datum povratka, ukupan broj sedišta za rezervaciju) vraća informaciju o mogućnosti rezervacije prevoza. Input i output poruke su u WSDL dokumentu definisane na sledeći način:

```
...
<wsdl:message name="PrevozRezervacijaSoapIn">
  <wsdl:part name="parameters" element="tns:PrevozRezervacija" />
</wsdl:message>
<wsdl:message name="PrevozRezervacijaSoapOut">
  <wsdl:part name="parameters" element="tns:PrevozRezervacijaResponse" />
</wsdl:message>
...
```

Tipovi ulazne i izlazne poruke su definisani preko *types* elementa.

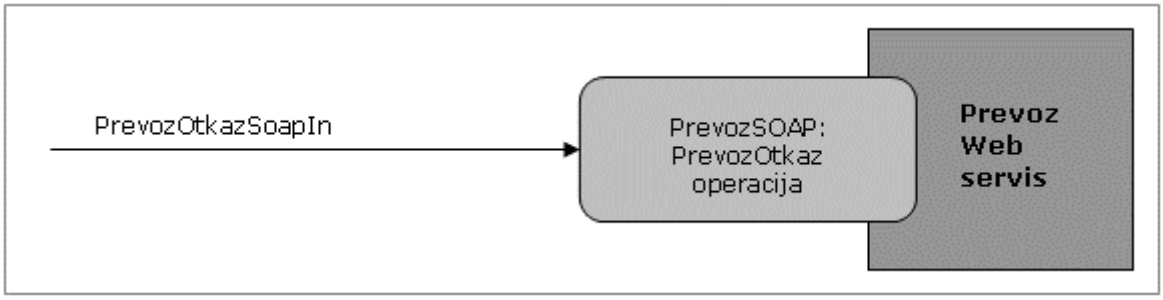
```
...

      <s:element name="PrevozRezervacija">
        <s:complexType>
          <s:sequence>
            <s:element minOccurs="1" maxOccurs="1" name="subagent"
type="s:int" />
            <s:element minOccurs="0" maxOccurs="1" name="brojUgovora"
type="s:string" />
            <s:element minOccurs="1" maxOccurs="1" name="destinacija"
type="s:int" />
            <s:element minOccurs="1" maxOccurs="1" name="datumOdlaska"
type="s:dateTime" />
            <s:element minOccurs="1" maxOccurs="1" name="datumPovratka"
type="s:dateTime" />
            <s:element minOccurs="1" maxOccurs="1"
name="ukupanBrojSedista" type="s:int" />
          </s:sequence>
```

```
        </s:complexType>
      </s:element>
      <s:element name="PrevozRezervacijaResponse">
        <s:complexType>
          <s:sequence>
            <s:element minOccurs="1" maxOccurs="1"
name="PrevozRezervacijaResult" type="s:boolean" />
          </s:sequence>
        </s:complexType>
      </s:element>
...
```

PrevozOtkaz operacija

Korišćenje Web servisa "Prevoz" za otkaz :

1. Koristeći Web aplikaciju, operater bira broj ugovora;
2. Web aplikacija kreira XML dokument i šalje ga na odredište, gde će ga čitati BizTalk server;
3. BizTalk server čita pristigli XML dokument i poziva na izvršavanje Web servis "Prevoz";
4. Operacija "PrevozOtkaz" od Web servisa "Prevoz" stornira ranije kreiranu rezervaciju prevoza.



slika 66: PrevozOtkaz operacija

PrevozOtkaz je request operacija, koja sadrži samo input poruku. Operacija za zadate ulazne podatke (subagent, broj ugovora) stornira ranije kreiranu rezervaciju prevoza. Input i output poruke su u WSDL dokumentu definisane na sledeći način:

```
...
  <wsdl:message name="PrevozOtkazSoapIn">
    <wsdl:part name="parameters" element="tns:PrevozOtkaz" />
  </wsdl:message>
  <wsdl:message name="PrevozOtkazSoapOut">
    <wsdl:part name="parameters" element="tns:PrevozOtkazResponse" />
  </wsdl:message>
...
```

Tip ulazne poruke je definisan preko *types* elementa.

```
...

      <s:element name="PrevozOtkaz">
        <s:complexType>
          <s:sequence>
            <s:element minOccurs="1" maxOccurs="1" name="subagent"
type="s:int" />
            <s:element minOccurs="0" maxOccurs="1" name="brojUgovora"
type="s:string" />
          </s:sequence>
        </s:complexType>
      </s:element>
...
```

9.2.2.3 Web servis "Viza"

Web servis "Viza" omogućava slanje obaveštenja o zahtevu za vizu, kao i obaveštenje o problemu u vezi sa prosleđenim zahtevom za vizu, što je definisano preko WSDL elementa *portType*.

```
...

  <wsdl:portType name="VizaSoap">
    <wsdl:operation name="VizaRezervacija">
      <wsdl:documentation
xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/">Web servis omogucava slanje
zahteva za rezervaciju vize.</wsdl:documentation>
      <wsdl:input message="tns:VizaRezervacijaSoapIn" />
      <wsdl:output message="tns:VizaRezervacijaSoapOut" />
    </wsdl:operation>
    <wsdl:operation name="VizaProblem">
      <wsdl:documentation
xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/">Web servis salje obavestenje
o problemu u vezi rezervacije vize.</wsdl:documentation>
      <wsdl:input message="tns:VizaProblemSoapIn" />
      <wsdl:output message="tns:VizaProblemSoapOut" />
    </wsdl:operation>
  </wsdl:portType>

...
```

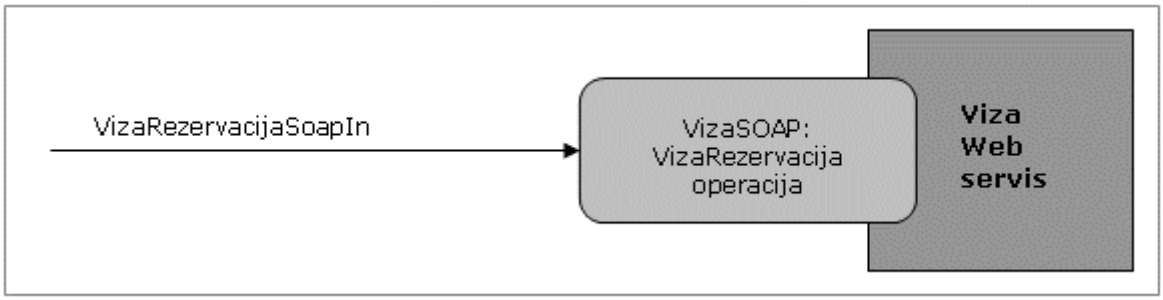
portType element se sastoji od dve operacije: VizaRezervacija i VizaProblem.

VizaRezervacija operacija

Korišćenje Web servisa "Viza" za rezervaciju:

1. Koristeći Web aplikaciju, operater unosi podatke o putniku za koga treba obezbediti putnu vizu za određenu zemlju;
2. Web aplikacija kreira XML dokument i šalje ga na odredište, gde će ga čitati BizTalk server;
3. BizTalk server čita pristigli XML dokument i poziva na izvršavanje Web servis "Viza";

4. Operacija "VizaRezervacija" od Web servisa "Viza" šalje informaciju o zahtevu za vizu.



slika 67: VizaRezervacija operacija

VizaRezervacija je request operacija, koja za zadate ulazne podatke (subagent, broj ugovora) šalje informaciju o zahtevu za vizu. Input i output poruke su u WSDL dokumentu definisane na sledeći način:

```
...
  <wsdl:message name="VizaRezervacijaSoapIn">
    <wsdl:part name="parameters" element="tns:VizaRezervacija" />
  </wsdl:message>
  <wsdl:message name="VizaRezervacijaSoapOut">
    <wsdl:part name="parameters" element="tns:VizaRezervacijaResponse" />
  </wsdl:message>
...
```

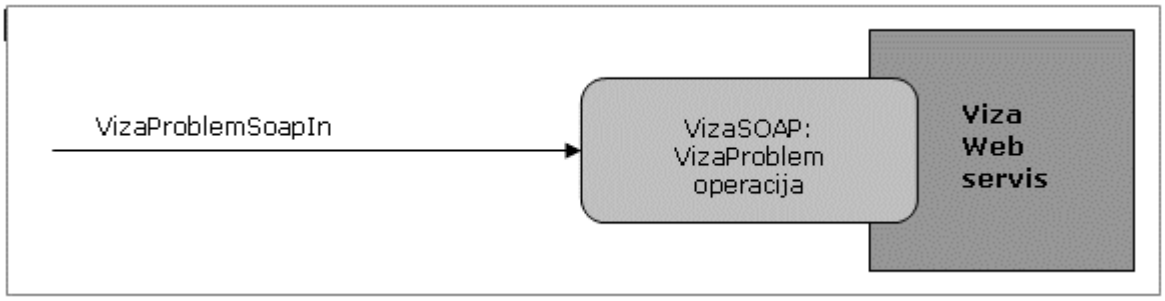
Tip ulazne poruke je definisan preko *types* elementa.

```
...

      <s:element name="VizaRezervacija">
        <s:complexType>
          <s:sequence>
            <s:element minOccurs="1" maxOccurs="1" name="subagent"
type="s:int" />
            <s:element minOccurs="0" maxOccurs="1" name="brojUgovora"
type="s:string" />
            <s:element minOccurs="1" maxOccurs="1"
name="vizaRezervacijaZahtev" type="s:boolean" />
          </s:sequence>
        </s:complexType>
      </s:element>
      <s:element name="VizaRezervacijaResponse">
        <s:complexType />
      </s:element>
...
```

VizaProblem operacija

Korišćenje Web servisa "Viza" za obaveštenje o viznom problemu:

1. Koristeći Web aplikaciju, operater unosi broj ugovora i na osnovu odluke konzulata evidentira da je zahtev za vizu odbijen;
2. Web aplikacija kreira XML dokument i šalje ga na odredište, gde će ga čitati BizTalk server;
3. BizTalk server čita pristigli XML dokument i poziva na izvršavanje Web servis "Viza";
4. Operacija "VizaProblem" od Web servisa "Viza" šalje informaciju o postojanju problema u vezi sa vizom.



slika 68: VizaProblem operacija

VizaProblem je request operacija, koja sadrži samo input poruku. Operacija za zadate ulazne podatke (subagent, broj ugovora) šalje obaveštenje o postojanju problema u vezi sa zahtevom za vizu. Input i output poruke su u WSDL dokumentu definisane na sledeći način:

```
...
  <wsdl:message name="VizaProblemSoapIn">
    <wsdl:part name="parameters" element="tns:VizaProblem" />
  </wsdl:message>
  <wsdl:message name="VizaProblemSoapOut">
    <wsdl:part name="parameters" element="tns:VizaProblemResponse" />
  </wsdl:message>
...
```

Tip ulazne poruke je definisan preko *types* elementa.

```
...

      <s:element name="VizaProblem">
        <s:complexType>
          <s:sequence>
            <s:element minOccurs="1" maxOccurs="1" name="subagent"
type="s:int" />
            <s:element minOccurs="0" maxOccurs="1" name="brojUgovora"
type="s:string" />
            <s:element minOccurs="1" maxOccurs="1"
name="vizaProblemObavestenje" type="s:boolean" />
          </s:sequence>
        </s:complexType>
      </s:element>
...
```

9.2.2.4 Web servis "EmailNote"

Web servis EmailNote omogućava slanje email poruke putniku, što je definisano preko WSDL elementa *portType*.

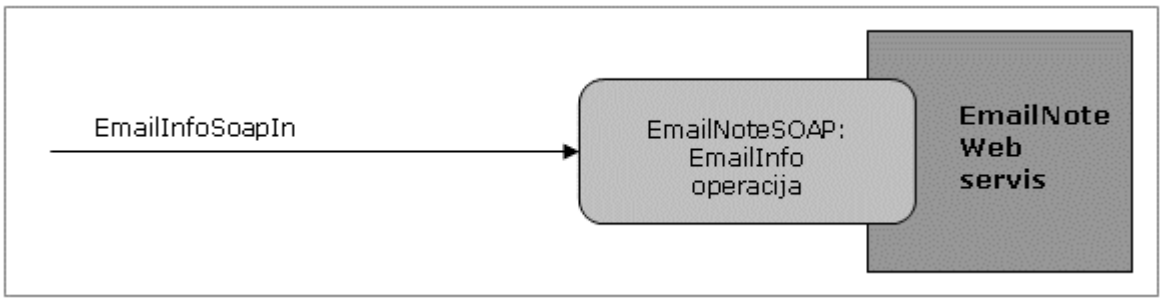
```
...

  <wsdl:portType name="EmailNoteSoap">
    <wsdl:operation name="EmailInfo">
      <wsdl:documentation
xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/">Web metod Email notification
ima zadatak da posalje email poruku klijentu.</wsdl:documentation>
      <wsdl:input message="tns:EmailInfoSoapIn" />
      <wsdl:output message="tns:EmailInfoSoapOut" />
    </wsdl:operation>
  </wsdl:portType>

...
```

portType element se sastoji od jedne operacije EmailInfo.

EmailInfo operacija

Korišćenje Web servisa "EmailNote" za slanje elektronske poruke:

1. Koristeći Web aplikaciju, operater bira broj ugovora;
2. Web aplikacija kreira XML dokument i šalje ga na odredište, gde će ga čitati BizTalk server;
3. BizTalk server čita pristigli XML dokument i poziva na izvršavanje Web servis "EmailNote";
4. Operacija "EmailInfo" od Web servisa "EmailNote" šalje elektronsku poruku putniku.



slika 69: EmailInfo operacija

VizaRezervacija je request operacija, koja za zadate ulazne podatke (subagent, broj ugovora) šalje elektronsku poruku putniku. Input i output poruke su u WSDL dokumentu definisane na sledeći način:

```
...
  <wsdl:message name="EmailInfoSoapIn">
    <wsdl:part name="parameters" element="tns:EmailInfo" />
  </wsdl:message>
```

```
  <wsdl:message name="EmailInfoSoapOut">
    <wsdl:part name="parameters" element="tns:EmailInfoResponse" />
  </wsdl:message>
...
```

Tip ulazne poruke je definisan preko *types* elementa.

```
...

      <s:element name="EmailInfo">
        <s:complexType>
          <s:sequence>
            <s:element minOccurs="1" maxOccurs="1" name="subagent"
type="s:int" />
            <s:element minOccurs="0" maxOccurs="1" name="brojUgovora"
type="s:string" />
          </s:sequence>
        </s:complexType>
      </s:element>
...
```

9.2.3 Generisanje BPEL na osnovu BPMN modela

Na osnovu kreiranog BPMN modela i postojanja potrebnih Web servisa, kreiran je BPEL izvršni kod. BPEL kod je generisan korišćenjem ITP programa za modeliranje procesa. Generisani BPEL kod nije sadržao dovoljno informacija, koje bi omogućile jednostavan prelazak na sledeći korak u kreiranju poslovnog procesa, a to je kreiranje izvršne verzije poslovnog procesa. Zbog toga je bilo potrebno da se naknadno ubace potrebne informacije u BPEL kod, ili da se potpuno definisanje poslovnog procesa obavi naknadno, kao što je to i urađeno, prilikom kreiranja orkestracije u BizTalk serveru.

Operativna verzija BPEL koda, usaglašena sa BPMN dijagramom i sa svim elementima neophodnim za kreiranje BizTalk orkestracije, je predstavljena na sledeći način:

```
<?xml version="1.0"?>
<process xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:q2="http://www.turoperator.co.yu"
xmlns:q1="http://BT_TA_RezervacioniSistem.PropertySchema.PropertySchema"
xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
name="TA_RezervacioniSistem.TA_RezervacioniSistemm"
targetNamespace="http://www.turoperator.co.yu"
xmlns="http://schemas.xmlsoap.org/ws/2003/03/business-process/">

<!-- definicije partnerLink elemenata, koji ukazuju na Web servise, koje poziva
poslovni proces -->
  <partnerLinks>
    <partnerLink name="Prevoz_WS" partnerLinkType="q2:Prevoz_WSType"
partnerRole="portRole" />
    <partnerLink name="Email_WS" partnerLinkType="q2:Email_WSType"
partnerRole="portRole" />
    <partnerLink name="PrevozOtkaz_WS" partnerLinkType="q2:PrevozOtkaz_WSType"
partnerRole="portRole" />
    <partnerLink name="Aranzman_Zahtev" partnerLinkType="q2:Aranzman_ZahtevType"
myRole="portRole" />
```

```
    <partnerLink name="SmestajOtkaz_WS" partnerLinkType="q2:SmestajOtkaz_WSType"
partnerRole="portRole" />
    <partnerLink name="Smestaj_WS" partnerLinkType="q2:Smestaj_WSType"
partnerRole="portRole" />
    <partnerLink name="Viza_WS" partnerLinkType="q2:Viza_WSType"
partnerRole="portRole" />
    <partnerLink name="VizaProblem_WS" partnerLinkType="q2:VizaProblem_WSType"
partnerRole="portRole" />
  </partnerLinks>

<!-- definicije varijabli (koriste se za pamćenje poruka, koje se razmenjuju između
partnera u poslovnom procesu, ili za čuvanje podataka koji se odnose na stanje
procesa) -->
  <variables>
    <variable name="primiZahtev"
messageType="q2:__messagetype_BT_TA_RezervacioniSistem_AranzmanZahtev" />
    <variable name="msgPrevozRezervacija_File"
messageType="q2:__messagetype_BT_TA_RezervacioniSistem_PrevozRezervacija_PrevozReze
rvacija_Zahtev" />
    <variable name="msgPrevozRezervacija_Zahtev"
messageType="q2:__messagetype_BT_TA_RezervacioniSistem_PrevozRezervacija_PrevozReze
rvacija_Zahtev" />
    <variable name="msgPrevozRezervacija_Odgovor"
messageType="q2:__messagetype_BT_TA_RezervacioniSistem_PrevozRezervacija_PrevozReze
rvacija_Odgovor" />
    <variable name="msgPrevozOtkaz_Zahtev"
messageType="q2:__messagetype_BT_TA_RezervacioniSistem_PrevozOtkaz" />
    <variable name="msgSmestajRezervacija_File"
messageType="q2:__messagetype_BT_TA_RezervacioniSistem_SmestajRezervacija_SmestajRe
zervacija_Zahtev" />
    <variable name="msgSmestajRezervacija_Zahtev"
messageType="q2:__messagetype_BT_TA_RezervacioniSistem_SmestajRezervacija_SmestajRe
zervacija_Zahtev" />
    <variable name="msgSmestajRezervacija_Odgovor"
messageType="q2:__messagetype_BT_TA_RezervacioniSistem_SmestajRezervacija_SmestajRe
zervacija_Odgovor" />
    <variable name="msgSmestajOtkaz_Zahtev"
messageType="q2:__messagetype_BT_TA_RezervacioniSistem_SmestajOtkaz" />
    <variable name="msgEmailInfo_Zahtev"
messageType="q2:__messagetype_BT_TA_RezervacioniSistem_EmailInfo" />
    <variable name="msgPrevozOtkaz_File"
messageType="q2:__messagetype_BT_TA_RezervacioniSistem_PrevozOtkaz" />
    <variable name="msgSmestajOtkaz_File"
messageType="q2:__messagetype_BT_TA_RezervacioniSistem_SmestajOtkaz" />
    <variable name="msgVizaRezervacija_File"
messageType="q2:__messagetype_BT_TA_RezervacioniSistem_VizaRezervacija" />
    <variable name="msgVizaOtkaz_File"
messageType="q2:__messagetype_BT_TA_RezervacioniSistem_VizaProblem" />
    <variable name="msgVizaRezervacija_Zahtev"
messageType="q2:__messagetype_BT_TA_RezervacioniSistem_VizaRezervacija" />
    <variable name="msgVizaOtkaz_Zahtev"
messageType="q2:__messagetype_BT_TA_RezervacioniSistem_VizaProblem" />
  </variables>

<!-- definicija korelacionog seta, koja će omogućiti povezivanje sa kreiranom
instancom poslovnog procesa, u našem slučaju povezivanje po osnovu broja ugovora i
šifre subagenta; svojstva za korelacioni set su definisana u okviru BizTalk server
aplikacije -->
  <correlationSets>
    <correlationSet properties="q1:brojUgovora q1:subagent"
name="InstancaOrkestracije" />
  </correlationSets>

<!-- definicija poslovnog procesa -->
  <sequence>

<!-- poslovni proces čeka dolazak odgovarajuće poruke (receive tag), koju će
obraditi operacija Operation_1 na portu Aranzman_Zahtev_1; poruka se pamti pomoću
```

```
varijable primiZahtev; atribut createInstance naređuje procesnoj mašini da kreira
novu instancu poslovnog procesa; BPEL Receive tag je nastao od BPMN Start Event
elementa tipa Message -->
    <receive partnerLink="Aranzman_Zahtev" portType="q2:Aranzman_Zahtev_1"
operation="Operation_1" variable="primiZahtev" createInstance="yes">

<!-- inicijalizacija korelacije -->
      <correlations>
        <correlation set="InstancaOrkestracije" initiate="yes" />
      </correlations>
    </receive>
    <scope name="Transaction_Rezervacija" variableAccessSerializable="no">
      <faultHandlers>
        <catchAll>
          <empty />
        </catchAll>
      </faultHandlers>

<!-- flow tag omogućava definisanje aktivnosti koje će biti izvršavane paralelno;
kreiran je mapiranjem BPMN paralenih Gateway elemenata; switch tagovi omogućavaju
grananja u okviru flow elementa; flow tag koji prethodi svakom switch tagu se
koristi u slučajevima kada je moguće da se izvrši više od jedne paralelne grane-->
      <flow>

<!-- obrada zahteva za prevoz -->
        <flow>
         <switch>
<!-- ukoliko postoji, obrada zahteva za obezbeđenje prevoza -->
          <case condition="/prevozZahtev/aln:Boolean">
            <scope name="Transaction_Prevoz" variableAccessSerializable="no">
<!-- compensationHandler sekcija omogućava otkazivanje rezervisanog prevoza, i to
pozivanjem Web servisa preko invoke taga -->
              <compensationHandler>
                <sequence>
                  <receive partnerLink="Aranzman_Zahtev"
portType="q2:Aranzman_Zahtev_1" operation="Operation_3"
variable="msgPrevozOtkaz_File">
                    <correlations>
                      <correlation set="InstancaOrkestracije" />
                    </correlations>
                  </receive>
                  <assign>
                    <copy>
                      <from variable="msgPrevozOtkaz_File" part="part" />
                      <to variable="msgPrevozOtkaz_Zahtev" part="part" />
                    </copy>
                  </assign>
                  <invoke partnerLink="PrevozOtkaz_WS"
portType="q2:PrevozOtkaz_WS_1" operation="Operation_1"
inputVariable="msgPrevozOtkaz_Zahtev" />
                </sequence>
              </compensationHandler>
              <sequence>
                <receive partnerLink="Aranzman_Zahtev"
portType="q2:Aranzman_Zahtev_1" operation="Operation_2"
variable="msgPrevozRezervacija_File">
                  <correlations>
                    <correlation set="InstancaOrkestracije" />
                  </correlations>
                </receive>
<!-- assign tag omogućava kopiranje vrednosti varijabli -->
                <assign>
                  <copy>
                    <from variable="msgPrevozRezervacija_File" part="part" />
                    <to variable="msgPrevozRezervacija_Zahtev" part="part" />
                  </copy>
                </assign>
<!—- invoke tag omogućava poziv Web servisa, koji je definisan preko atributa
```

```
portType, atribut inputVariable definiše ulaznu poruku, a tag outputVariable
varijablu preko koje će biti prihvaćen rezultat -->
                <invoke partnerLink="Prevoz_WS" portType="q2:Prevoz_WS_1"
operation="Operation_1" inputVariable="msgPrevozRezervacija_Zahtev"
outputVariable="msgPrevozRezervacija_Odgovor" />
              </sequence>
            </scope>
          </case>
         </switch>
        </flow>

<!-- obrada zahteva za smeštaj -->
        <flow>
         <switch>
<!-- ukoliko postoji, obrada zahteva za obezbeđenje smeštaja -->
          <case condition="/smestajZahtev/aln:Boolean">
            <scope name="Transaction_ff0e6143_ff37_4381_9b2c_Smestaj"
variableAccessSerializable="no">
              <compensationHandler>
                <sequence>
                  <receive partnerLink="Aranzman_Zahtev"
portType="q2:Aranzman_Zahtev_1" operation="Operation_5"
variable="msgSmestajOtkaz_File">
                    <correlations>
                      <correlation set="InstancaOrkestracije" />
                    </correlations>
                  </receive>
                  <assign>
                    <copy>
                      <from variable="msgSmestajOtkaz_File" part="part" />
                      <to variable="msgSmestajOtkaz_Zahtev" part="part" />
                    </copy>
                  </assign>
                  <invoke partnerLink="SmestajOtkaz_WS"
portType="q2:SmestajOtkaz_WS_1" operation="Operation_1"
inputVariable="msgSmestajOtkaz_Zahtev" />
                </sequence>
              </compensationHandler>
              <sequence>
                <receive partnerLink="Aranzman_Zahtev"
portType="q2:Aranzman_Zahtev_1" operation="Operation_4"
variable="msgSmestajRezervacija_File">
                  <correlations>
                    <correlation set="InstancaOrkestracije" />
                  </correlations>
                </receive>
                <assign>
                  <copy>
                    <from variable="msgSmestajRezervacija_File" part="part" />
                    <to variable="msgSmestajRezervacija_Zahtev" part="part" />
                  </copy>
                </assign>
                <invoke partnerLink="Smestaj_WS" portType="q2:Smestaj_WS_1"
operation="Operation_1" inputVariable="msgSmestajRezervacija_Zahtev"
outputVariable="msgSmestajRezervacija_Odgovor" />
              </sequence>
            </scope>
          </case>
         </switch>
        </flow>

<!-- obrada zahteva za vizu -->
        <flow>
         <switch>
<!-- ukoliko postoji, obrada zahteva za obezbeđenje vize -->
          <case condition="/vizaZahtev/aln:Boolean">
            <scope name="Transaction_360318c0_66a3_4dfb_abf5_Viza"
variableAccessSerializable="no">
```

```
              <compensationHandler>
                <sequence>
                  <receive partnerLink="Aranzman_Zahtev"
portType="q2:Aranzman_Zahtev_1" operation="Operation_7"
variable="msgVizaOtkaz_File">
                    <correlations>
                      <correlation set="InstancaOrkestracije" />
                    </correlations>
                  </receive>
                  <assign>
                    <copy>
                      <from variable="msgVizaOtkaz_File" part="part" />
                      <to variable="msgVizaOtkaz_Zahtev" part="part" />
                    </copy>
                  </assign>
                  <invoke partnerLink="VizaProblem_WS"
portType="q2:VizaProblem_WS_1" operation="Operation_1"
inputVariable="msgVizaOtkaz_Zahtev" />
                </sequence>
              </compensationHandler>
              <sequence>
                <receive partnerLink="Aranzman_Zahtev"
portType="q2:Aranzman_Zahtev_1" operation="Operation_6"
variable="msgVizaRezervacija_File">
                  <correlations>
                    <correlation set="InstancaOrkestracije" />
                  </correlations>
                </receive>
                <assign>
                  <copy>
                    <from variable="msgVizaRezervacija_File" part="part" />
                    <to variable="msgVizaRezervacija_Zahtev" part="part" />
                  </copy>
                </assign>
                <invoke partnerLink="Viza_WS" portType="q2:Viza_WS_1"
operation="Operation_1" inputVariable="msgVizaRezervacija_Zahtev" />
              </sequence>
            </scope>
          </case>
         </switch>
        </flow>

      </flow>
    </scope>

<!-- dodeljivanje vrednosti varijablama -->
    <assign>
      <copy>
        <from />
        <to variable="msgEmailInfo_Zahtev" part="part" />
      </copy>
      <copy>
        <from variable="primiZahtev" property="q1:subagent" />
        <to variable="msgEmailInfo_Zahtev" part="part" query="/*[local-
name()='EmailInfo_Zahtev' and namespace-
uri()='http://BT_TA_RezervacioniSistem.EmailInfo']/*[local-name()='subagent' and
namespace-uri()='']" />
      </copy>
      <copy>
        <from variable="primiZahtev" property="q1:brojUgovora" />
        <to variable="msgEmailInfo_Zahtev" part="part" query="/*[local-
name()='EmailInfo_Zahtev' and namespace-
uri()='http://BT_TA_RezervacioniSistem.EmailInfo']/*[local-name()='brojUgovora' and
namespace-uri()='']" />
      </copy>
    </assign>

<!-- poziv Web servisa koji šalje email -->
```

```
    <invoke partnerLink="Email_WS" portType="q2:Email_WS_1" operation="Operation_1"
inputVariable="msgEmailInfo_Zahtev">
      <correlations>
        <correlation set="InstancaOrkestracije" pattern="out" />
      </correlations>
    </invoke>
<!-- terminate tag označava kraj izvršavanja procesa; nastao mapiranjem BPMN End
Event elementa tipa Terminate -->
    <terminate />
  </sequence>
</process>
```

9.2.4 Razvoj Web aplikacije

Web aplikacija "Rezervacioni sistem" treba da omogući nezavisno rezervisanje smeštaja i prevoza za putnike turoperatora i njegovih subagenata. Funkcije rezervisanja smeštaja i prevoza treba da se obave u on-line režimu, nad jedinstvenom bazom podataka, čime se potpuno isključuju mogućnosti višestruke prodaje istih kapaciteta. Osim postignute konzistentnosti baze podataka, razvojem Web aplikacije omogućiće se i sledeće pogodnosti:

- U svakom momentu je poznato stanje raspoloživih i prodatih kapaciteta. Te informacije omogućavaju turoperatoru da efikasnije upravlja glavnim resursima agencije – smeštajnim i prevoznim kapacitetima, što je izuzetno značajno u periodima intenzivne prodaje aranžmana (zima, leto). Na drugoj strani, subagenti prodaju samo kapacitete koji su trenutno i stvarno raspoloživi, čime se isključuju slučajevi pogrešnih prodaja.
- Nestaje potreba za stalnom usmenom i pismenom komunikacijom osoblja turoperatora i subagenata, a vezano za rezervisanje smeštaja i prevoza.
- Rezervacioni sistem je raspoloživ 7x24h sedmično, čime se eliminišu problemi neusaglašenosti radnog vremena turoperatora i subagenata.

Poseban problem organizovanja putovanja je obezbeđenje putnih viza, koje su najčešće potrebne za većinu destinacija i većini putnika. S obzirom da je reč o zimskim i letnjim aranžmanima, problemi zbog gužvi, neefikasnosti i ležernosti konzulata stranih zemalja postaju još izraženiji. Navedene činjenice, kao i broj raznovrsnih dokumenata, koje je potrebno pripremiti za različite kategorije putnika, zahtevaju stalnu razmenu informacija između turoperatora i subagenata. Sa namerom da se napori povezani za viziranjem smanje, a poslovi praćenja statusa viza pojedinačnih putnika olakšaju, odlučeno je da sastavni deo Web aplikacije Rezervacioni sistem bude i modul: Viziranje putnih isprava. Prvobitna ideja bila je da se korišćenjem novokreiranog Web servisa omogući automatska komunikacija i razmena potrebnih podataka turoperatora sa konzulatima, i dalje: sa subagentima. Međutim, organizacija rada konzulata i njihova nezainteresovanost onemogućili su planiranu i potpunu automatizaciju.

Na osnovu analize trenutnih mogućnosti, odlučeno je da se praćenje statusa viza delimično olakša, automatizacijom komunikacije turoperatora i subagenata. Informaciju o statusu vize (prihvaćen ili odbijen zahtev) osoblje turoperatora unosi ručno, na osnovu čega novokreirani Web servis automatski prosleđuje elektronsku poruku subagentu, sa odgovarajućom porukom o statusu vize putnika.

Za potrebe automatizacije poslovnih funkcija saradnje turoperatora i subagenata kreirana je Web aplikacija "Rezervacioni sistem".

Novi deo aplikacije zahteva autentifikaciju, jer je dostupan samo određenim korisnicima, odnosno prijavljenim subagentima i operaterima turoperatora.

Odlučeno je da novi deo aplikacije bude portalskog tipa, jer je osnovna namena Web portala da predstavlja centralizovanu tačku pristupa za pronalaženje i upravljanje informacijama, a za izvršavanje različitih poslova preko Web portala koristi se Web browser. Takođe, Web portali predstavljaju vrlo pogodan način za prikupljanje informacija sa različitih izvora i njihovo koncentrisanje na jedno mesto.

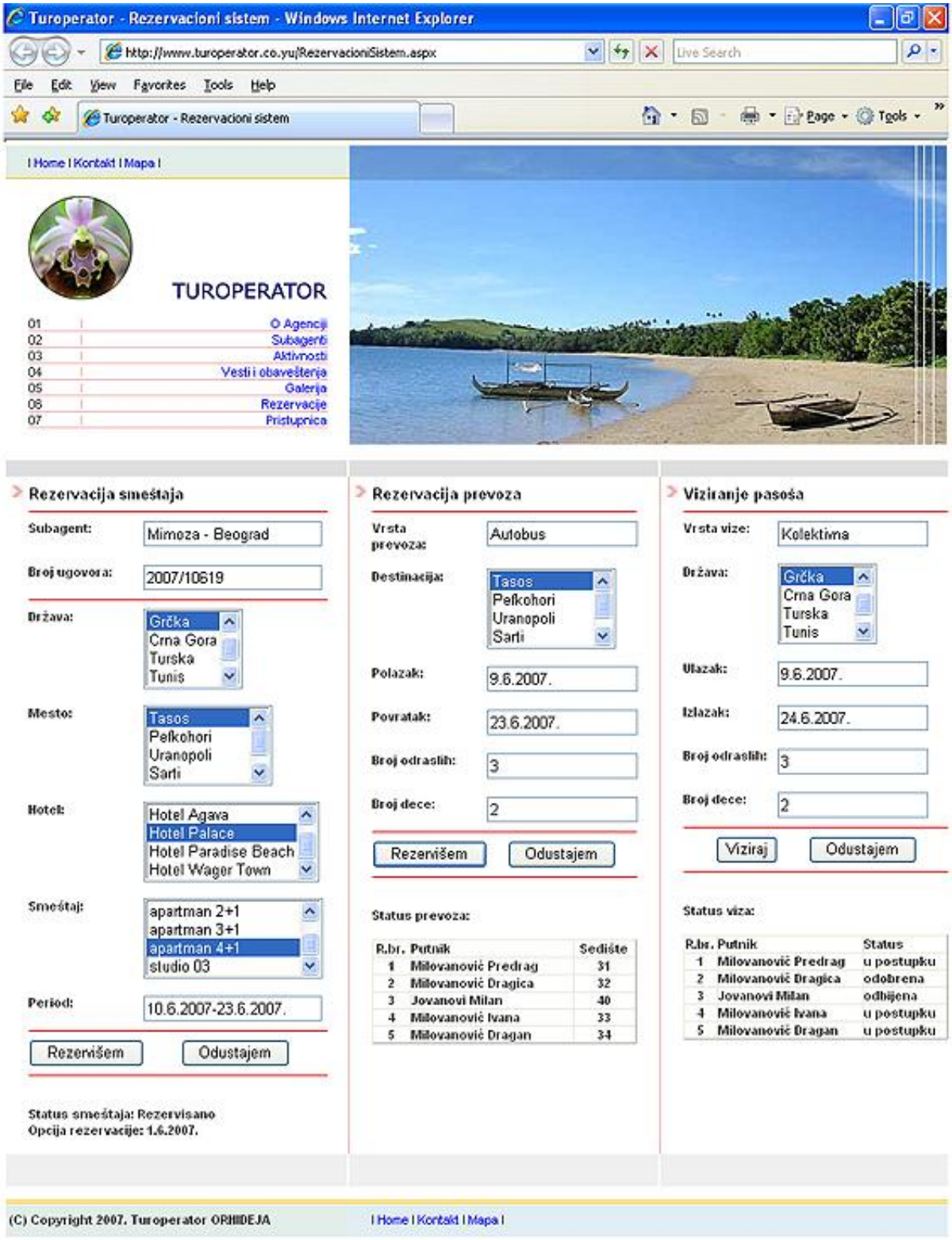
Web portal "Rezervacioni sistem" je kreiran kao višeslojna ASP.NET Web aplikacija. Ovakav način modularnog programiranja može poboljšati i olakšati održavanje, tako što izoluje ostatak Web aplikacije od promena napravljenih u bilo kom delu sistema.

Prezentacioni sloj uključuje sve što je specifično za korisnički interfejs: Web forme (.aspx strane), korisničke kontrole (.ascx) i pripadajuće code-behind fajlove (.aspx.cs i .ascx.cs), koji sadrže programsku logiku, kao i grafičke fajlove koji su sastavni delovi korisničkog interfejsa.

Srednji sloj, odnosno poslovni sloj, sadrži klase koje sprovode definisana poslovna pravila.

Sloj za pristup podacima je realizovan preko klasa, koje pomoću ADO.NET tehnologije pristupaju potrebnim bazama podataka. Korišćenje ADO.NET tehnologije veoma pojednostavljuje pristup različitim izvorima podataka.

Osnovni ekran Web portala sadrži delove, koji omogućavaju, od strane operatera, izbor zahtevane destinacije, datum i period korišćenja aranžmana, definisanje broja putnika koji će putovati, izbor prevoza, izbor vize, kao i ostalih neophodnih podataka.



slika 70 - Osnovni ekran Web aplikacije

Koristeći prikazani ekran, operater na osnovu zahteva putnika, vrši odabir podataka. Na osnovu odabranih podataka, u skladu sa definisanim šemama, kreiraju se XML dokumenti, koji se šalju do odredišta, gde će ih čitati BizTalk aplikacija.

Prikaz sadržaja Zahtev.xml dokumenta.

```
<?xml version="1.0" encoding="utf-8"?>
<aranzmanZahtev xmlns="http://BT_TA_RezervacioniSistem.aranzmanZahtev">
  <smestajZahtev>true</smestajZahtev>
  <prevozZahtev>false</prevozZahtev>
  <vizaZahtev>false</vizaZahtev>
</aranzmanZahtev>
```

Prikaz sadržaja Zahtev_Smestaj.xml dokumenta.

```
<?xml version="1.0" encoding="utf-8"?>
<SmestajRezervacija_Zahtev
xmlns="http://BT_TA_RezervacioniSistem.SmestajRezervacija">
  <subagent>7</subagent>
  <brojUgovora>123</brojUgovora>
  <destinacija>100</destinacija>
  <hotel>1</hotel>
  <datum>2007-06-10T00:00:00+01:00</datum>
  <trajanje>14</trajanje>
  <strukturaSoba>
    <tipSobe>4+1</tipSobe>
    <brojSoba>1</brojSoba>
  </strukturaSoba>
</SmestajRezervacija_Zahtev>
```

9.2.5 Kreiranje BizTalk Server 2006 aplikacije

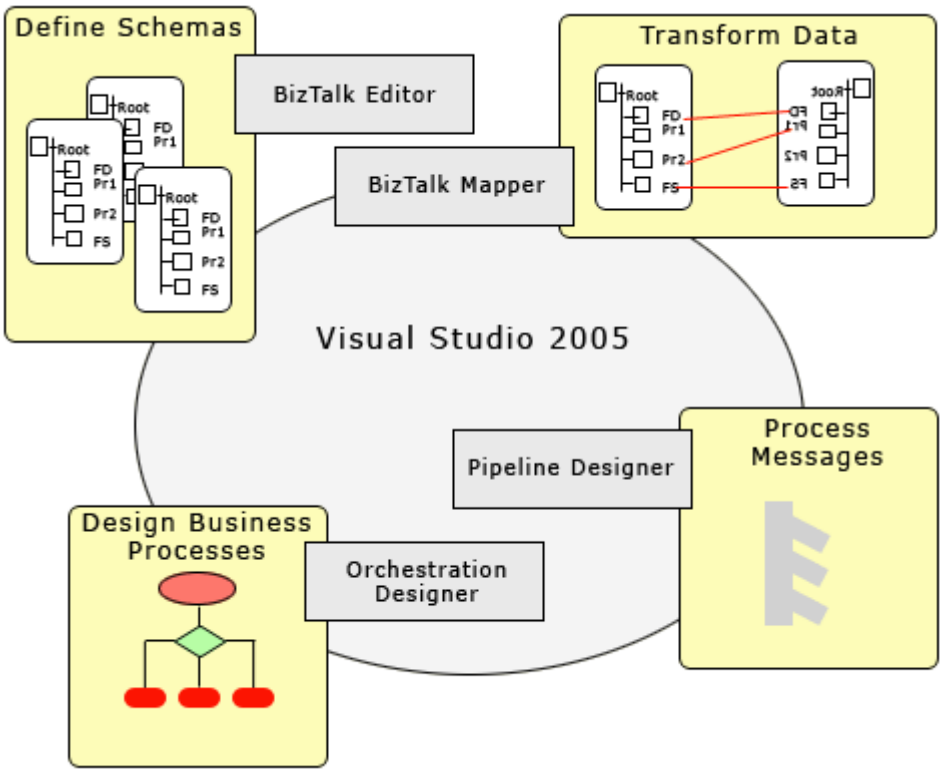
Kreirani BPEL kod treba da se izvrši pomoću neke od procesnih mašina, u našem slučaju je izabran je BizTalk Server 2006.

BizTalk razvojni alati su dostupni u okviru Visual Studio .NET okruženja. Proces instalacije BizTalk Server 2006, dodaje u BizTalk Projects folder u New Project dijalog boks. BizTalk Projects folder sadrži šablone za kreiranje sledećih tipova projekata:

- *Empty BizTalk Server Project*. Koristi se kada želimo da kreiramo novu BizTalk Server 2006 aplikaciju.
- *BizTalk Server Migration Project*. Ovaj tip projekta pokreće BizTalk Migration Wizard, jer mu je osnovna namena prebacivanje starijih verzija BizTalk aplikacija u verziju 2006.
- *BizTalk Server Human Workflow Project*. Koristi se za kreiranje nove Human Workflow aplikacije.
- *BizTalk Server BPEL Import Project*. Omogućava uvoženje BPEL, WSDL i XSD datoteka u BizTalk Project pomoću BPEL Import Wizard.

Prilikom rada na BizTalk projektu, u okviru Visual Studio, mogu se koristi sledeći razvojni alati:

- *BizTalk Editor.* Omogućava da se definišu šeme. Sve poruke koje BizTalk server obrađuje moraju biti predstavljene preko XSD šema. XSD šeme opisuju tipove podataka svakog sloga i polja koji su sastavni delovi poruke.
- *BizTalk Mapper.* Omogućava mapiranje izvorne u odredišnu šemu, pri čemu se mogu definisati transformacije između polja sa podacima u okviru poruke.
- *Pipeline Designer.* Koristi se za obradu ulaznih i izlaznih poruka, korišćenjem operacija kao što su enkripcija, dekripcija, kompresija, preformatiranje i validacija.
- *Orchestration Designer.* Omogućava kreiranje orkestracija za modeliranje poslovnih procesa.



slika 71: BizTalk Server 2006 razvojni alati u okviru Visual Studio 2005 okruženja

Kreiranje BizTalk aplikacije od napravljenog BPEL izvršnog koda je relativno jednostavno. BizTalk obezbeđuje šablon BizTalk Server BPEL Import Project, koji koristeći čarobnjaka korak po korak izvršava uvoz BPEL koda. Da bi uvoz mogao da se obavi, neophodno je, prilikom kreiranja BPEL koda, obratiti pažnju na sledeće:

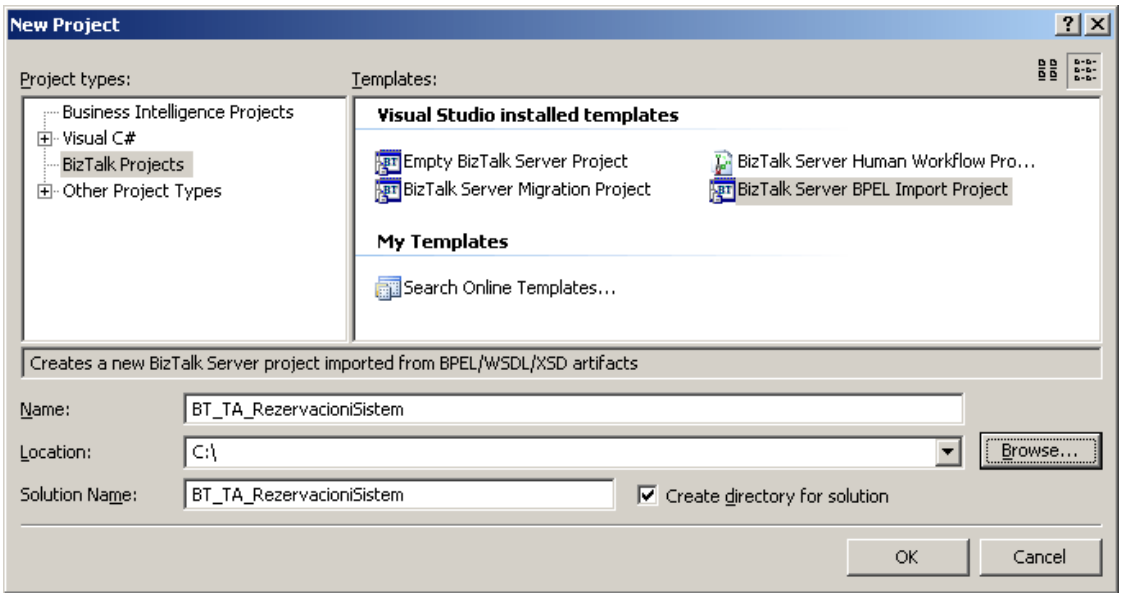
- Svojstvo Name od WSDL Definition čvora i ime BPEL proces čvora ne smeju biti isti;

- BPEL kod ne sme da sadrži rezervisane reči iz XLANG jezika (BizTalk ne podržava prirodno BPEL, njegov osnovni jezik je XLANG, preko kojeg su interno zapamćene i predstavljene orkestracije, i koji je specijalno kreiran i podržan od strane BizTalk servera);

- XSD šeme ne smeju sadržati kompleksne tipove.

Da bi se kreirala BizTalk aplikacija od napravljenog BPEL izvršnog koda, urađeno je sledeće:

1. Kreiran je novi BizTalk Server Import Project u Visual Studio .NET okruženju i dato mu je ime BT_TA_RezervacioniSistem.

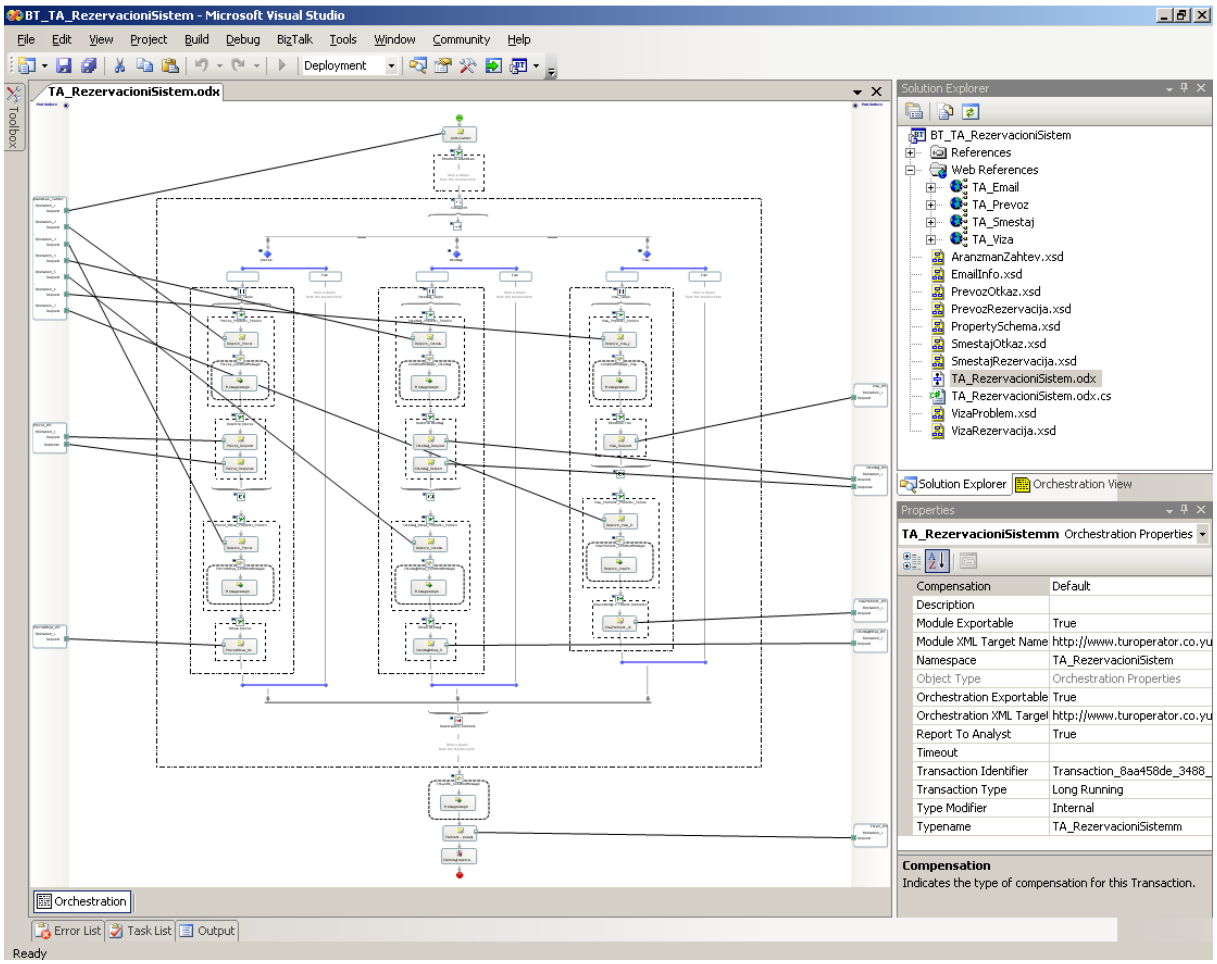


slika 72: Kreiranje BizTalk BPEL Import Project aplikacije

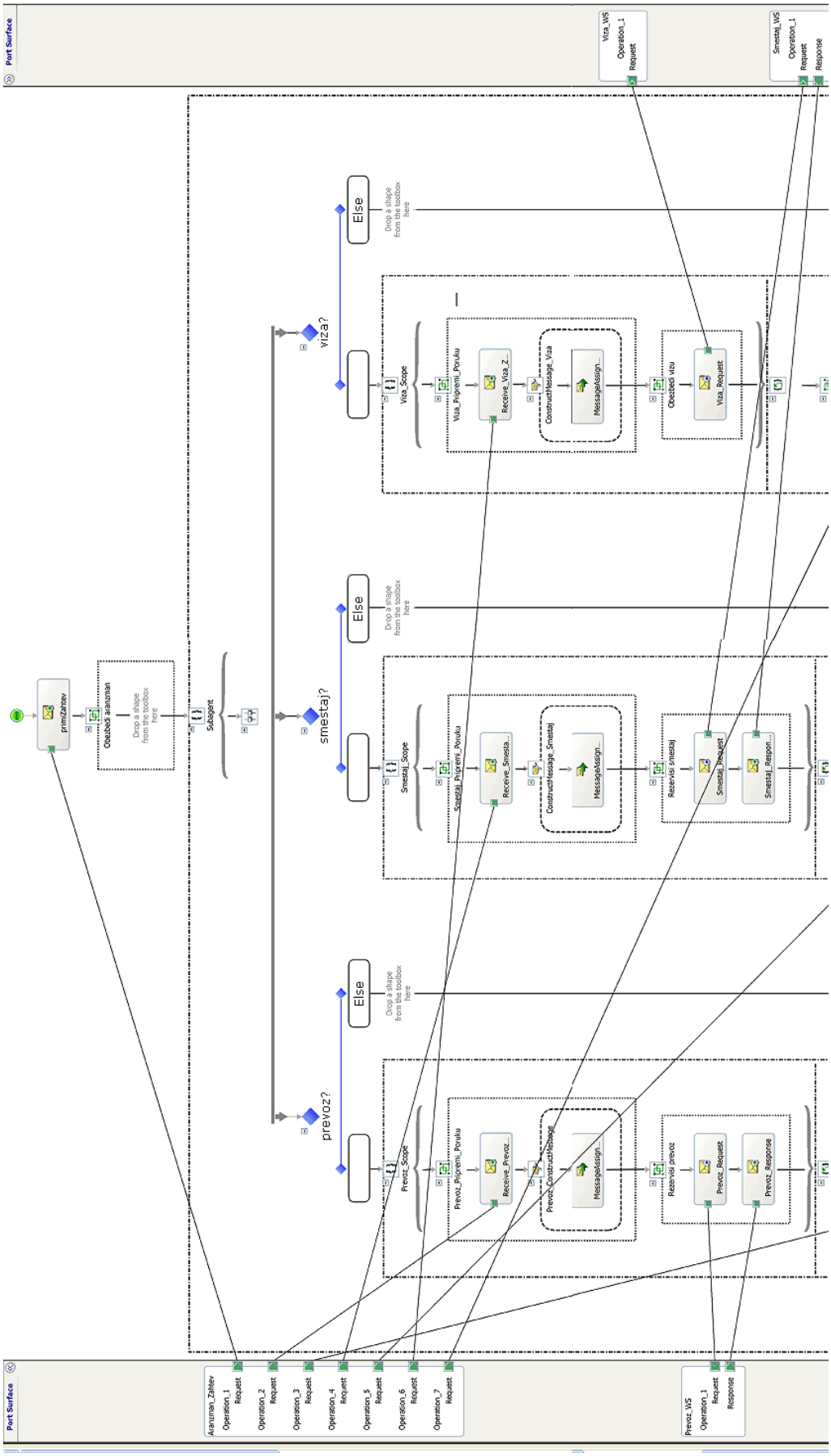
2. Posle završetka rada BPEL Import čarobnjaka, u Solution Explorer mogu se videti kreirane šeme i jedna orkestracija, koja predstavlja vizuelnu prezentaciju poslovnog procesa. BPEL dokument je konvertovan u orkestraciju (.odx datoteku), koja sadrži logiku poslovnog procesa i deklaracije za elemente kao što su Partner Links (Port i Role Link u BizTalk serveru) i varijable (poruke u BizTalk serveru).

3. Kreirana orkestracije je morala da se doradi, odnosno da se:

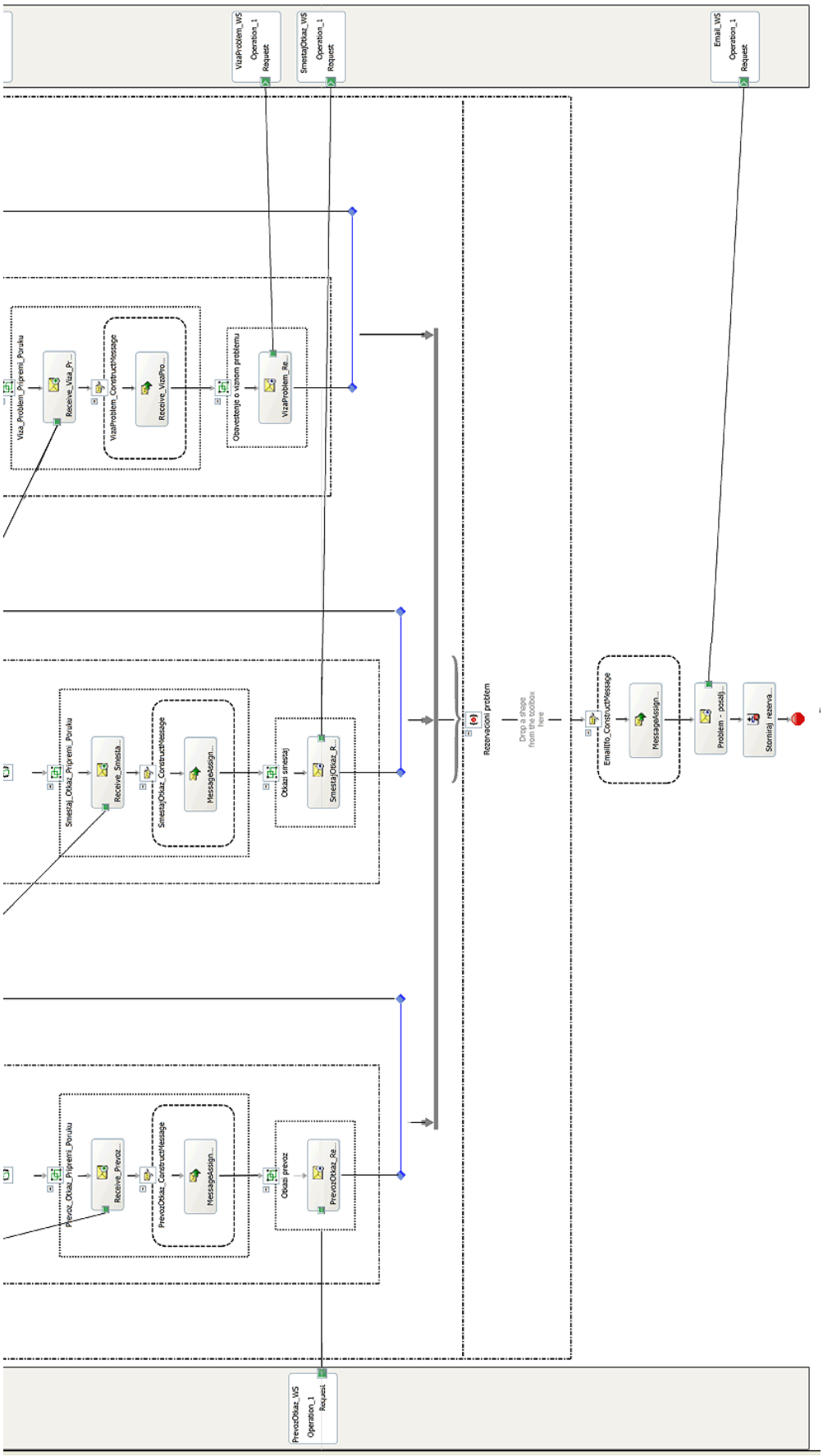
   - definišu portovi za preuzimanje pristiglih XML datoteka,
   - kreiraju poruke koji će biti popunjene pročitanim XML podacima,
   - na osnovu kreiranih poruka konstruišu poruke koje će nositi ulazne podatke kod poziva Web servisa,
   - definišu korelacioni tipovi i korelacioni setovi, koji omogućavaju da se pristigla poruka upari sa kreiranom instancom orkestracije.



slika 73: Kreirana BizTalk server orkestracija za poslovni proces turističkog poslovanja u Visual Studio 2005 razvojnom alatu



slika 74: Detaljan prikaz BizTalk Server orkestracije - 1. deo



slika 75: Detaljan prikaz BizTalk Server orkestracije - 2. deo

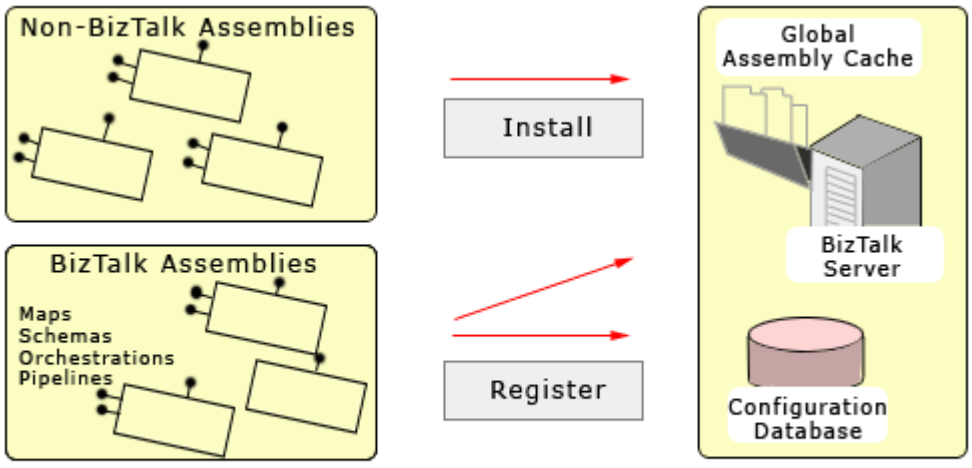
Bitno je naglasiti da se većina BizTalk server poslovnih procesa, upravo kreira pomoću orkestracije, kao nova BizTalk server aplikacija, kojoj nije prethodilo kreiranje BPMN dijagrama, kao ni prebacivanje BPMN dijagrama u BPEL jezik. Uočena su dva bitna razloga. Prvi razlog proističe iz činjenice da mali broj poslovnih korisnika poznaje BPMN notaciju, pa čak i ako je poznaje, nije spreman da se bavi svim detaljima vezanim za implementaciju poslovnog procesa, što podrazumeva poznavanje i definisanje podataka neophodnih da bi se BPMN dijagram prebacio u BPEL jezik. Drugi razlog je činjenica da nije podržano jednoznačno mapiranje BPMN u BPEL izvršni kod, što je i konstatovano napred u tekstu.

Posledica navedenog je da se kreiranjem poslovnih procesa i dalje bave diveloperi, a ne oni koji ih najbolje poznaju, a to su poslovni analitičari.

Ista su iskustva i kada se koristi Oracle BPEL Process Manager softvera, koji je, kao i BizTalk server, veoma korišćen kao procesna mašina.

Drugi način, sa kojim se srećemo u praksi, je da se poslovni proces definiše korišćenjem BPMN dijagrama, pomoću nekog od dostupnih alata, koji obezbeđuju grafičko okruženje za dizajn i razvoj poslovnih procesa. Zatim se, kreirani poslovni proces, eksportuje u izvršni kod prilagođen odabranoj procesnoj mašini (BizTalk, Oracle BPEL), na kojoj će se kasnije i izvršavati. Operaciji za eksport mora da prethodi definisanje detalja vezanih za implementaciju: povezivanje sa Web servisima, koji će se pozivati, kao i kreiranje šema za poruke, koje će se razmenjivati. Takođe, BPMN dijagram ne može da sadrži elemente koji se ne mogu direktno mapirati u jezik odabrane procesne mašine. Osim toga, potrebno je da se posle eksport operacije, kreirana orkestracija doradi, da bi proces mogao da se izvršava.

9.3 Isporučivanje i administracija rešenja



slika 76: Grafički prikaz instalacije BizTalk aplikacije

Da bi BizTalk aplikacija mogla da se izvršava, prvo je potrebno da se iskompajlira i izgradi, što rezultuje kreiranjem .NET sklopa (assembly). Posle kreiranja izvršnog sklopa, potrebno je da se aplikacija isporuči. Isporuka aplikacije se sastoji od sledećih koraka:

1. instaliranje sklopa (DLL) u Global Assembly Cache (GAC);

2. registrovanje sklopa u BizTalk konfiguracionu bazu.

BizTalk Server 2006 poseduje nekoliko alata za nadgledanje i upravljanje isporučenim aplikacijama, otkrivanje grešaka, kontrolisanje sigurnosnih podešavanja:

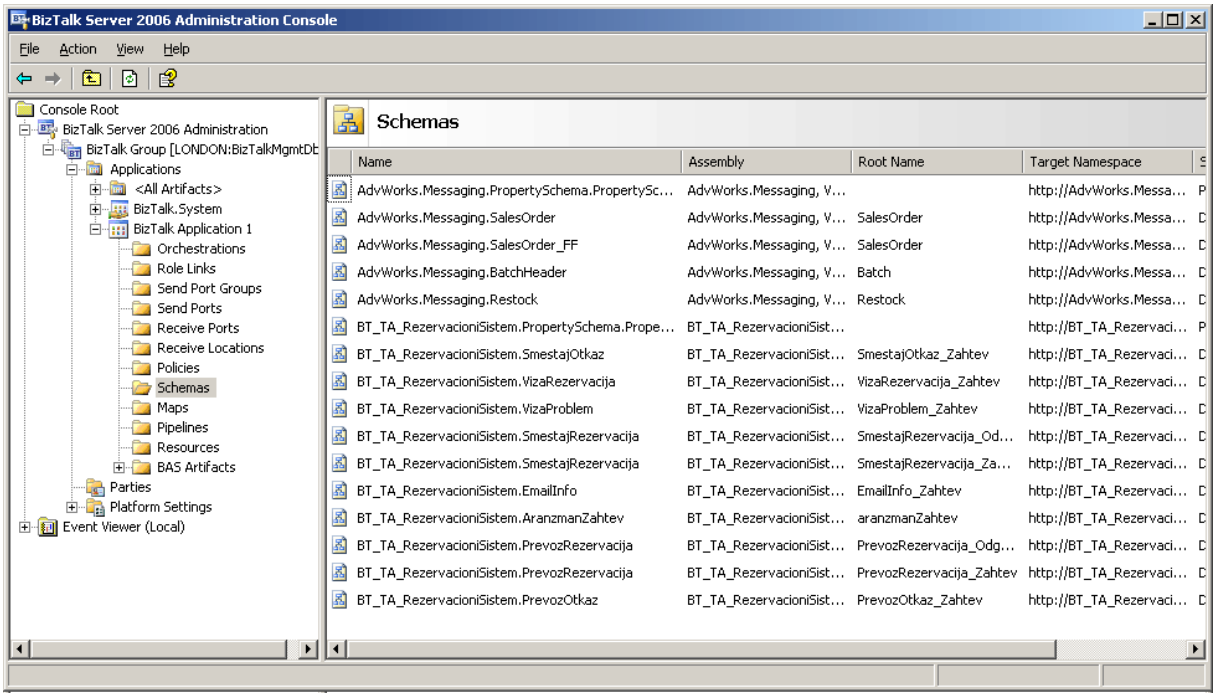
- BizTalk Server 2006 Administration Console - Predstavlja Microsoft Management (MMC) snap-in i osnovni je alat za upravljanje BizTalk serverom. Preko korisničkog interfejsa ove konzole moguće je obaviti niz administrativnih poslova: pregledati sklopove i pripadajuće artifakte koji su isporučeni u BizTalk konfiguracionu bazu, upravljanje BizTalk grupama, otkrivanje grešaka, upravljanje različitim podešavanjima.

- Health and Activity Tracking (HAT) - Omogućava praćenje stanja BizTalk poruka i orkestracija i pruža pomoć u identifikovanju grešaka i uskih grla u BizTalk Server okruženju. Koristi se za pregled tehničkih detalja određene orkestracije ili poruke, kao i za praćenje jedne protoka određene poruke od početka do kraju u okviru BizTalk sistema. Poslovni korisnici mogu prregledati, nadgledati i pretraživati poruke i procese, kao i pamtiti kreirane upite zbog ponovnog korišćenja.

- Business Activity Services (BAS) Web sajt - Predstavlja Windows SharePoint Services Web sajt, i omogućava poslovnim korisnicima preko interfejsa, jednostavnog za korišćenje, interakciju sa poslovnim partnerima i poslovnim procesima.

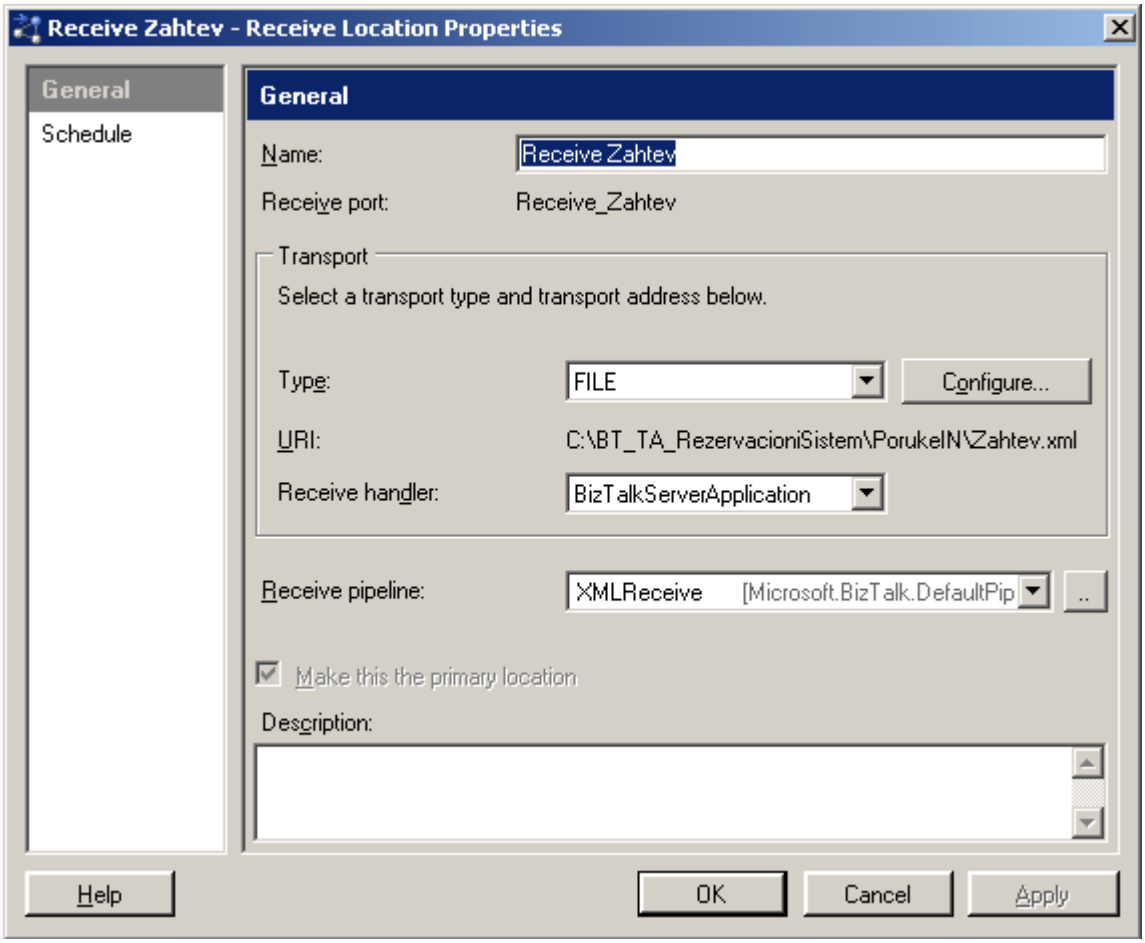
- Business Activity Monitoring (BAM) - Omogućava poslovnim korisnicima nadgledanje poslovnih procesa, koji se izvršavaju.

Da bi BizTalk orkestracija mogla da se izvršava, potrebno je da se startuje pomoću BizTalk Server 2006 Administration Console. Kod našeg procesa, trenutno startovanje orkestracije nije bilo moguće, jer nisu bila zadovoljena dva važna preduslova: svi portovi definisani u orkestraciji moraju biti povezani za fizičkim portovima, i neohodno je da orkestracija bude povezana sa Host aplikacijom. Host predstavlja kontejner za različite BizTalk entitete, uključujući orkestracije i upravljanje primanjem, obradom i slanjem poruka.



slika 77: BizTalk Server 2006 Administration Console

Bilo je neophodno da se definišu Receive Locations i Receive Ports, i da se nakon toga status za definisane Receive Locations stavi na Enabled.

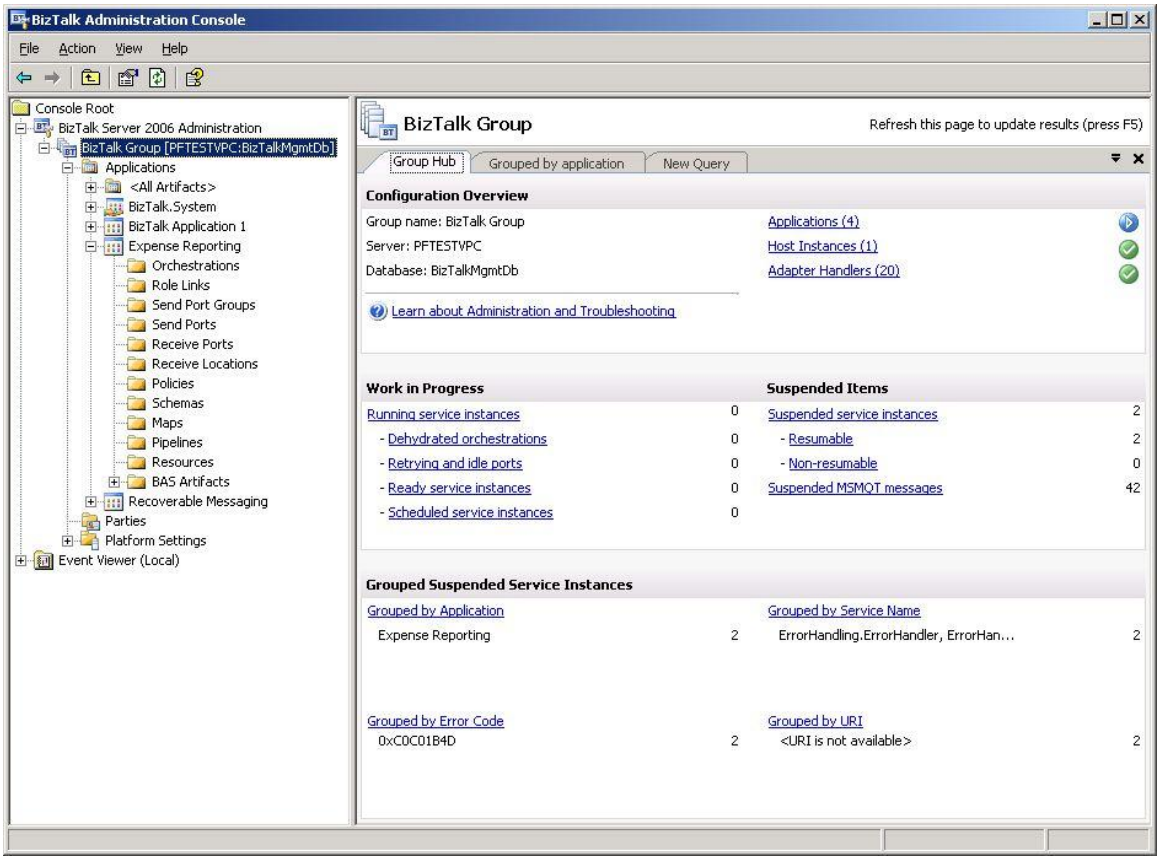


slika 78: Podešavanje Receive Location svojstava u Administratrion Console

Nakon startovanja BizTalk aplikacije, moguće je korišćenjem alata Health and Activity Tracking (HAT) i Business Activity Monitoring (BAM) nadgledati i upravljati orkestracijom koja se izvršava, kao i pregledati informacije o poslovnom procesu koji se izvršava. Ove alate mogu koristiti i diveloperi i poslovni analitičari, pri čemu je BAM više namenjen poslovnim analitičarima.

BizTalk Administration Console uključuje Group Hub stranu, koje prikazuje informacije o stanju BizTalk servera. Group Hub strana obezbeđuje sledeće:

- pregled operacija koje se izvršavaju,
- pregled suspendovanih operacija,
- mogućnost izvršenja obimnih operacija (terminate, suspend, resume) na instancama servisa,
- mogućnost izgradnje i pamćenja posebnih upita, koji vraćaju informacije o porukama i instancama servisa.



slika 79: Nadgledanje statusa BizTalk aplikacija pomoću Group Hub strane

9.4 Ostvareni ciljevi i prednosti rešenja

Izgrađeno rešenje za automatizaciju poslovanja turoperatora i njegovih subagenata nalazi se u fazi završnih testiranja i stabilizacije.

Dosadašnje iskustvo primene novog rešenja pokazuje da su glavni ciljevi postignuti:

- Omogućeno je korišćenje postojećih aplikacija, bez značajnijih izmena, kako u pogledu programskih rešenja, tako i u pogledu postojećih struktura baze podataka.
- Omogućeno je dinamičko korišćenje postojeće Web aplikacije turoperatora u smislu provere slobodnih kapaciteta smeštaja i prevoza.
- Izradom potrebnih Web servisa automatizovana je komunikacija turoperator – subagent, što značajno doprinosi efikasnosti i tačnosti poslovanja. Osim toga, onemogućena je nekontrolisana prodaja istih smeštajnih kapaciteta i nekontrolisana prodaja usluga prevoza.
- Kreirano rešenje je fleksibilno, proširivo i predstavlja primenljivu osnovu za dalji razvoj u budućnosti.

10. Analiza mogućnosti automatizacije modeliranja poslovnih procesa

Realni sistemi su najčešće, po svojoj prirodi, složeni. To se odražava i na poslovne procese, koji se odvijaju unutar njih. Konstatacija da je "proces složen", u kontekstu ovog rada, najčešće je rezultat sticaja nekih od sledećih okolnosti:

- Poslovni proces je objektivno, po svojoj prirodi složen.
- Za poslovni proces vezuje se veći broj učesnika, čije je uloge potrebno precizno naznačiti: u pogledu odgovornosti za akcije i podatke; u okviru vremenske dimenzije; sinhronizacije međusobnih aktivnosti i slično.
- Za pojedine poslovne procese postoje različiti, često suprotstavljeni, interesi zainteresovanih, što može znatno da otežа analizu i ispravno i objektivno modeliranje poslovnog procesa.
- Dešava se da je poslovni proces izrazito nestruktuiran. Samim tim, takav poslovni proces često je nemoguće modelirati na način, kojim bi se omogućila njegova automatizacija i tako dalje.

Nažalost, problemi složenosti poslovnih procesa nisu jedini. Na putu do uspešnih modela poslovnih procesa, potrebno je savladati još neke prepreke:

- Objektivna raznovrsnost poslovnih procesa zahteva dosta široka opšta i specijalizovana znanja divelopera. Do njih dolazi sopstvenim radom i uz pomoć stručnjaka iz oblasti koja je predmet analize.
- Diveloper mora da ostvari dobru saradnju sa predstavnicima budućih korisnika sistema, i to u svim fazama analize i modeliranja.
- Rezultat zajedničkog rada trebalo bi da bude formalno iskazan model poslovnih procesa. Od dobijenog modela očekuje se da verno predstavi analizirani proces i da bude razumljiv svim učesnicima analize.

Najčešći oblici formalnih modela poslovnih procesa obuhvataju grafičke prikaze, dopunjene određenim tekstualnim sadržajima i objašnjenjima modela. Sa ciljem da se postupci modeliranja učine efikasnijim i sa namerom da se izvrši standardizacija modeliranja poslovnih procesa, do sada je razvijeno i preporučeno za korišćenje više grafičkih jezika i standarda. Najpoznatiji i najnoviji su:

- Unified Modelling Language (UML) i
- Business Process Modelling Notation (BPMN)

Činjenica da postoji više različitih grafičkih notifikacija i standarda za modeliranje poslovnih procesa govori da je teško odabrati i opredeliti se za "najbolji". Pre se može govoriti o izboru, koji je primeren problemu koji se rešava. Pri tome, mogu se uzeti u obzir neki od sledećih kriterijuma:

- Da li zadovoljava potrebe ciljne grupe kojoj je namenjen?
- Da li je jednostavan za korišćenje?
- Ima li dovoljno grafičkih elemenata za jednoznačan prikaz modela?
- Da li obezbeđuje neophodan nivo razumljivosti za sve učesnike u procesu modeliranja?
- Da li je jednostavan za kreiranje i ažuriranje grafičkih modela?
- Da li postoji dovoljno dobrih alata za kreiranje i dokumentovanje modela i slično?

Poredeći BPMN i UML, osnovne razlike između njih su:

- UML je namenjen softverskim analitičarima, dizajnerima i diveloperima, jer nudi specifikaciju, vizuelizaciju i mogućnost dokumentovanja artifakta za kompjuterski zasnovane sisteme,
- BPMN koriste poslovni analitičari, arhitekte sistema i softverski inženjeri, jer nude specifikaciju, vizuelizaciju i mogućnost dokumentovanja artifakta vezanih za poslovni domen,

- BPMN i UML zasnivaju se na različitim paradigmama:
    - UML je zasnovan na objektno orijentisanom pristupu,
    - BPMN je zasnovan na procesnom pristupu,
    - u pogledu nivou implementacije, UML jeziku nedostaje mogućnost implementacije poslovnih, visoko konceptualnih modela.

Na osnovu navedenih karakteristika BPMN i UML, preporuka je da se za potpuno modeliranje poslovnih procesa koriste BPMN dijagrami. Svrha kreiranja BPMN dijagrama je dvostruka: da omogući pogled iz poslovnog ugla na poslovni proces, a zatim, da omogući generisanje izvršnog koda pomoću BPEL jezika.

Mapiranje BPMN u BPEL jezik opisano je detaljnim specifikacijama, čiji je cilj da se premosti jaz između grafičkog dizajna i izvršavanja poslovnog procesa na procesnoj mašini.

Tokom rada na ovoj tezi pokazalo se da je najčešće nemoguće da se poslovni procesi, opisani grafičkim simbolima, automatski prevedu u neki od programskih kodova, odnosno pokazalo se da je mapiranje samo delimično obavljeno.

Problem nemogućnosti jednoznačnog mapiranja BPMN notacije i BPEL izvršni jezik, proizvođači softvera rešavaju na sledeći način:

- Grafički predstave BPEL proces i nazovu ga BPMN (što znači da koriste samo BPEL jezik)
- Koriste poseban alat za kreiranje BPMN dijagrama, a neki drugi za formiranje BPEL procesa.
- Od kreiranog BPMN dijagrama kreira se izvršni kod, prilagođen unapred odabranoj procesnoj mašini, pri čemu se, po pravilu, zahteva da BPMN ne sadrži elemente, koji se ne mogu direktno prevesti u izvršni kod.

Da bi se postojećim alatima generisao BPEL od BPMN, potrebno je da se reše dva problema:

- Prvi, da se dodaju svi detalji vezani za implementaciju poslovnog procesa pre nego što se BPMN izveze u BPEL, što često dovodi do uobičajenog pitanja "Da li očekujete da će poslovni analitičar postati programer, ili obrnuto, da li očekujete da će programer postati poslovni analitičar?";
- i drugi, zbog loše struktuiranog BPMN grafikona, ponekad izvoz nije moguć, ili za posledicu ima netačan BPEL kod, koji se ne može izvršavati na procesnoj mašini.

Problemi koji se javljaju kod generisanja BPEL koda na osnovu BPMN dijagrama su karakteristični kod svih modela na visokom nivou. Modeli na visokom nivou moraju da se na odgovarajući način sinhronizuju sa modelima na nižem nivou, što svakako nije jednostavno uraditi, osim u retkim situacijama kada je moguće obaviti mapiranje jedan-na-jedan. Uopšteno govoreći, mnogo dvosmislenosti se može naći u BPMN specifikaciji vezano za mapiranje u BPEL jezik.

Sa BPM stanovišta, poslovni proces je sastavljen od različitih podprocesa, koji mogu biti ugnježdeni ili ulančani međusobno. Svaki podproces može biti vlasništvo ili može biti izvršavan od strane različitih poslovnih jedinica. Za razliku od BPM, BPEL ne poznaje koncept podprocesa. To ne znači da se korišćenjem BPEL ne može kreirati proces od ugnježdenih i ulančanih komponenti. Takođe, BPEL ne podržava koncepte ljudi - učesnika identifikovanih preko uloga, grupa ili korisničkih imena.

Činjenica je da, bez obzira na postojanje različitih nedostataka, neki standard ili specifikacija biće prihvaćena, ukoliko postoji dobar softver koji ih primenjuje, i ukoliko postoji literatura sa detaljnim opisima i objašnjenjima.

Softveri koji kreiraju i izvršavaju BPEL poslovne procese, moraju da zadovolje više zahteva:

- jednostavno korišćenje;
- jednostavno kreiranje tokova, kao i njihova promena i brisanje;
- jednostavno testiranje;
- mogućnost nadgledanja izvršavanja;
- sposobnost za simultano upravljanje velikim brojem tokova;
- jednostavna integracija sa Web servisima i slično.

S obzirom da na tržištu softvera postoji više proizvoda, koji zadovoljavaju napred navedene zahteve, budućnost BPEL izvršnog jezika je verovatno obezbeđena.

Konkretnom primenom navedenih metodologija, standarda i tehnoloških rešenja modelirani su poslovni procesi poslovanja turističke agencije. Njeno poslovanje posmatrano je iz ugla samostalnog rada, da bi se zatim, zbog prirode posla, modelirali poslovni procesi vezani za saradnju sa drugim turističkim agencijama - subagentima.

Za definisanje poslovnog procesa rada turističke agencije urađen je BPMN dijagram.

Na osnovu urađenog BPMN modela i prethodno kreiranih, potrebnih Web servisa, generisan je BPEL izvršni kod. BPEL kod generisan je korišćenjem istog alata, ITP programa za modeliranje procesa.

Generisani BPEL kod nije sadržao dovoljno informacija za kreiranje izvršne verzije poslovnog procesa. Zbog toga bilo je neophodno:

- da se naknadno ubace potrebne informacije u BPEL kod, ili
- da se potpuno definisanje poslovnog procesa obavi kasnije, kao što je to i urađeno, a nakon kreiranja orkestracije.

Orkestracija je urađena pomoću BizTalk servera. Kreirana orkestracije je morala da se doradi, odnosno:

- da se definišu portovi za preuzimanje pristiglih XML datoteka,
- da se kreiraju poruke koje će biti popunjene pročitanim XML podacima,
- da se, na osnovu kreiranih poruka konstruišu nove poruke, za prenos podataka do Web servisa,
- da se definišu korelacioni tipovi i korelacioni setovi, koji omogućavaju uparivanje pristigle poruke sa kreiranom instancom orkestracije.

Bitno je naglasiti da se, nezavisno od procesne mašine koja se koristi, većina poslovnih procesa upravo kreira pomoću orkestracije, a bez prethodno kreiranog BPMN dijagrama i bez prebacivanja BPMN dijagrama u BPEL jezik. Uočena su dva bitna razloga. Prvi razlog proističe iz činjenice da mali broj poslovnih korisnika poznaje BPMN notaciju, pa čak i ako je poznaje, nije spreman da se bavi svim detaljima vezanim za implementaciju poslovnog procesa, što podrazumeva poznavanje i definisanje podataka neophodnih da bi se BPMN dijagram prebacio u BPEL jezik. Drugi razlog je činjenica da nije podržano jednoznačno mapiranje BPMN u BPEL izvršni kod, što je i konstatovano napred u tekstu.

Posledica navedenog je da se kreiranjem poslovnih procesa i dalje više bave diveloperi, a ne oni koji ih najbolje poznaju, a to su poslovni analitičari.

I pored činjenice da je dosta napora uloženo u razvoj standarda i softvera, željeni rezultati nisu u potpunosti postignuti u pogledu automatizacije modeliranja i izvršavanja poslovnih procesa. S obzirom na aktuelnost oblasti, realno je u bliskoj budućnosti očekivati dalji napredak, koji će omogućiti potpunu automatizaciju modeliranja poslovnih procesa.

11. Zaključak

U okviru ove magistarske teze data je analiza postojećih koncepata, dizajna, arhitekture i specifikacija standarda za modeliranje poslovnih procesa u B2B aplikacijama elektronskog poslovanja, sa posebnim naglaskom na analizu mogućnosti automatizacije izvršavanja poslovnih procesa.

U radu su prikazane sve faze modeliranja poslovnog procesa, od njegovog kreiranja, do isporučivanja, izvršavanja i administracije. Eksperimentalno su provereni softverski alati i metode na primeru svih faza modeliranja poslovnog procesa u razvoju aplikacije za turističko poslovanje.

Modeliranje poslovnih procesa (BPM) predstavlja rešenje za mnoge probleme, koji se javljaju u savremenom poslovanju. BPM omogućava upravljanje poslovima na način kako ih vide učesnici, pri čemu se poslovi posmatraju kao celina, a ne kao niz funkcionalnih delova. BPM usklađuje različite sastavne delove poslovnog procesa: postojeće poslovne sisteme, poslovna pravila, aktivnosti korisnika, integrišući ih dizajnom i okruženjem za izvršavanje. Takođe, BPM omogućava korišćenje novog, implementacionog stila, razvoj rešenja sa malo programskog koda, ili u potpunosti bez koda, sa ugrađenom pretpostavkom pojave promena u poslovanju u budućnosti.

Oblast modeliranja poslovnih procesa je složena, s obzirom da su poslovni procesi, sami po sebi i svojoj prirodi, složeni. Bilo je mnogo pokušaja da se formalizuje i standardizuje potpuna semantika za modelovanje scenarija iz realnog sveta. Iz različitih razloga, mnogi pokušaji su propali.

Kao vodeći standardi u oblasti modeliranja poslovnih procesa izdvojili su se BPMN, grafička notacija za prikaz poslovnih procesa, i BPEL, jezik za opis izvršavanja poslovnih procesa. Međutim, i kod primene ovih standarda, koji su verovatno najpotpuniji i najčešće se koriste, postoje određeni problemi. Problemi se javljaju kod generisanja BPEL koda na osnovu BPMN dijagrama, i karakteristični su za sve modele na visokom nivou. Modeli na visokom nivou moraju da se na odgovarajući način sinhronizuju sa modelima na nižem nivou, što svakako nije jednostavno uraditi, osim u retkim situacijama kada je moguće obaviti mapiranje u odnosu jedan-na-jedan.

Trenutno se problem prelaska iz jedne u drugu fazu modeliranja poslovnog procesa rešava tako, što se od kreiranog BPMN dijagrama kreira BPEL izvršni kod, prilagođen određenoj procesnoj mašini. Da bi to bilo moguće, uslov je da BPMN ne sadrži elemente koji se ne mogu direktno mapirati.

Generalno, kod većine aktuelnih informatičkih standarda, postoje određene nedorečenosti i dvosmislenosti, što dovodi do problema u njihovoj implementaciji. Vodeće svetske softverske kompanije uočene probleme najčešće rešavaju nadogradnjom postojećih modela. Ovako nastala rešenja još više otežavaju povezivanje i automatizaciju.

Prateći trenutne trendove, očekuje se da je programiranje, zasnovano na procesnom pristupu, sledeći korak u razvoju softverskog inženjeringa. Pri tome, procesno orijentisano programiranje ne treba da zameni, već da dopuni postojeće pristupe programiranju, kao što su strukturno, objektno i komponentno orijentisano programiranje.

Naučni i stručni doprinosi magistarske teze su:

- Analiza mogućnosti automatskog povezivanja različitih faza životnog ciklusa modelovanja poslovnih procesa u elektronskom poslovanju, primenom postojećih

standarda i tehnologija.

- Pregled i analiza postojećih standarda za modelovanje poslovnih procesa.
- Razmatranje problema i zaključci, vezani za konverzije BPMN dijagrama u BPEL izvršni jezik.
- Primena metodologije modelovanja poslovnih procesa pomoću izabranih alata u realizaciji jednog poslovnog procesa.

Istraživački rad, kao nastavak započetog rada na ovoj magistarskoj tezi, može se nastaviti u sledećim oblastima:

- Poboljšanja kod modeliranja poslovnih procesa vezana za automatizaciju njihovog izvršavanja.
- Istraživanja vezana za programiranje zasnovano na procesnom pristupu, odnosno "programiranje na veliko".

Literatura

[BPEL4People] Joint White Paper by IBM and SAP, "*WS-BPEL Extension for People – BPEL4People*", juli 2005.

[BPEL4WS 1.1] BEA Systems, IBM, Microsoft, SAP AG and Siebel Systems, "*Business Process Execution Language for Web Services Version 1.1*", Maj 2003, http://www-128.ibm.com/developerworks/library/specification/ws-bpel/, http://ifr.sap.com/bpel4ws/

[BPMN 1.0] *Business Process Management Initiative (BPMI): Business Process Modeling Notation (BPMN)*, Version 1.0, BPMI.org, May 2004.

[BPMN2BPEL] C. Ouyang, W.M.P. van der Aalst, M. Dumas, and A.H.M. ter Hofstede. "*Translating BPMN to BPEL*", Technical report, BPM Center Report BPM-06-02, 2006.

[Chappell BPM] Chappell, D., "*Microsoft and BPM: A Technology Overview*", Microsoft corporation, Avgust 2005.

[Chappell BTS] Chappell, D., "*Understanding BizTalk Server 2006*", Microsoft corporation, March 2007

[Dev SOA] Emig C., Weisser J., Abeck S., "*Development of SOA-Based Software Systems – an Evolutionary Programming Approach*", IEEE Computer Society (AICT/ICIW 2006)

[Haberl] Haberl, Stephen:" *Business Process Desciption Languages",* http://www.cis.unisa.edu.au/~cissh/research/webflow/bpdl.html

[Havey] Havey, Mike, "*Essential Business Process Modeling*", O'Reilly, 2005, ISBN: 0-596-00843-0

[He] He, H.; Haas, H.; Orchard, D.: "*Web Services Architecture Usage Scenarios*", W3C Working Group Note 11 February 2004. available at: http://www.w3.org/TR/ws-arch-scenarios/.

[IBM SOA] IBM Corporation, "*Patterns: Service-Oriented Architecture and Web Services*", IBM RedBooks (April 2004), ftp://www.redbooks.ibm.com/redbooks/SG246303/

[Juric] Juric M., "*Business Process Execution Language for Web Services Second Edition*", Packt Publishing, 2006, ISBN 1-904811-81-7

[Kerschberg] Kerschberg L., "*Methods for Information Systems Engineering: Knowledge Management and E-Business*", Information Systems Department, George Mason University

[Manes] Manes, Anne T., "*Web Services: A Manager's Guide*", Addison Wesley 2003, 0-321-18577-3

[P4] W. M. P. van der Aalst, A. H. M ter Hofstede, B. Kiepuszewski, and A. P. Barrios, "*Workflow Patterns*" Technical report, Eindhoven University of Technology, Eindhoven, 2003, Distributed and Parallel Databases, 14(1):5-51, 2003.

[P4 BPEL]W.M.P. van der Aalst, M. Dumas, A.H.M. ter Hofstede, N. Russell, H.M.W Verbeek, and P. Wohed, "*Life After BPEL?*" In M. Bravetti et al., editor, *In Proc. of the*

2nd Int. Workshop on Web Services and Formal Methods (WS-FM), volume 3670 of *LNCS*, pages 35–50. Springer Verlag, 2005.

[P4 BPMN] P. Wohed, W.M.P. van der Aalst, M. Dumas, A.H.M. ter Hofstede, N. Russell, "*Pattern-based Analysis of BPMN*", BPM Center Report BPM-05-26, BPMcenter.org, 2005.

[PS SR] Popović S.: "*Integracija aplikacija elektronskog poslovanja korišćenjem Web servisa*", specijalistički rad, FON, Beograd 2006.

[Rosen] Rosen, Mike, "*BPM and SOA: Where Does One End and the Other Begin?*"; Wilson Consalting Group, 2006.

[Silver] Silver, Bruce, "*BPM and SOA: One Technology, Two Communities*", BPMS Watch, 2005

[SOAP] *Simple Object Access Protocol,* http://www.w3.org/TR/2003/REC-soap12-part1-20030624/

[UDDI] *Universal Description, Discovery, and Integration*, http://www.uddi.org

[Web Services Arch] "*Web Services Architecture W3C Working Draft 11 February 2004.*", http://www.w3.org/TR/2004/NOTE-ws-arch-20040211/

[White BPMN] White, Stephen A., "*Introduction to BPMN*", IBM Cooperation, 2004.

[White BPMN BPEL] White, Stephen A., "*Using BPMN to Model a BPEL Process*", BPTrends, 3(3):1-18, 2005.

[White P4] White, Stephen A., "*Process Modeling Notations and Workflow Patterns*", BPTrends, Mart 2004.

[WS SOA] "*From Web Services to SOA and Everything in Between: The Journey Begins*", www.webservices.org (May 2005)

[WS-BPEL 2.0] *Web Service Business Process Execution Language Version 2.0, Working Draft*, July 2005, OASIS Technical Committee, http://www.oasis-open.org/committees/wsbpel

[WSDL 1.1] *Web Services Description Language (WSDL) Version 1.1*, W3C Note, http://www.w3.org/TR/2001/NOTE-wsdl-20010315

[XML Schema Part 1] *XML Schema Part 1: Structures*, W3C Recommendation, October 2004, http://www.w3.org/TR/xmlschema-1/

[XML Schema Part 2] *XML Schema Part 2: Datatypes*, W3C Recommendation, October 2004, http://www.w3.org/TR/xmlschema-2/

[XMLSpec] *XML Specification*, W3C Recommendation, February 1998, http://www.w3.org/TR/1998/REC-xml-19980210

[XPATH 1.0] *XML Path Language (XPath) Version 1.0*, W3C Recommendation, November 1999, http://www.w3.org/TR/1999/REC-xpath-19991116