UNIVERZA V LJUBLJANI

FAKULTETA ZA RAČUNALNIŠTVO IN INFORMATIKO

Borut Pirnat

Integracija aplikacij z uporabo Microsoft Biztalk-a

DIPLOMSKO DELO UNIVERZITETNEGA ŠTUDIJA

Mentor: doc. dr. Mojca Ciglarič

Ljubljana, 2008

Zahvala

Najprej bi se rad zahvalil mentorici doc. dr. Mojci Ciglarič za mentorstvo, vodenje in nasvete pri izdelavi diplomske naloge.

Hvala družini in Vesni za vzpodbude v času študija.

Kazalo

Seznam uporabljenih kratic

- API – *Application programming interface:* Aplikacijski programski vmesnik
- BAC – *BizTalk Administration Console:* Upravljalska konzola BizTalk
- BAM – *Business Activity Monitoring:* Sistem za spremljanje izvajana poslovnih procesov
- BPEL – *Business process execution language:* Jezik za formalno specifikacijo poslovnih procesov
- BPM – *Business process management:* Upravljanje poslovnih procesov
- BPMN – *Business process modeling notation:* Notacija za modeliranje poslovnih procesov
- BPMS – *Business process management system:* Sistem za upravljanje poslovnih procesov
- BRC – *Business rule composer:* Orodje za kreiranje poslovnih pravil
- BRE – *Business rules engine:* Mehanizem za izvajanje poslovnih pravil
- DURS – Davčna uprava Republike Slovenije
- EAI – *Enterprise application integration:* Integracija aplikacij
- ESB – *Enterprise service bus:* Storitveno vodilo
- HAT – *Health and Activity Tracking:* Orodje za razhroščevanje aplikacij BizTalk
- HTTP – *Hypertext transfer protocol:* Protokol za prenašanje hiperbesedila
- MIME – *Multipart Internet Mail Extension:* Standard za pripenjanje dvojiških datotek
- MMC - *Microsoft Management Console:* Upravljalska konzola Microsoft
- MSMQ – *Microsoft Message Queue:* Microsoftova izvedba sporočilnih vrst
- OLAP – *Online analitical processing:* Sprotno analitično procesiranje podatkov
- RDZ – Register davčnih zavezancev
- SMTP – *Simple mail transport protocol:* Protokol za prenašanje elektronske pošte
- SOA – *Service oriented arhitecture:* Storitveno usmerjena arhitektura
- SOAP - *Simple object access protocol:* Standarden protokol za izmenjavo sporočil XML po računalniškem omrežju
- URL – *Uniform Resource Locator:* Enoličen naslov v svetovnem spletu
- WCF – *Windows Communication Foundation:* Ogrodje za razvoj in izvajanje storitev v okolju Windows
- WSDL – *Web service description language:* Jezik za opisovanje spletnih storitev
- XML – *Extensible markup language:* Razširljiv označevalni jezik
- XSD – *XML Schema Definition Language:* Jezik za definiranje shem dokumentov XML
- XSLT – *XML Schema Transformation Language:* Jezik za definiranje preslikav med dokumenti XML

Slovar pojmov

- **Avtentikacija:**
  Avtentikacija je v računalništvu proces, v katerem se mora strežnik prepričati, da je uporabnik res tisto, kar trdi, da je.

- **Avtorizacija:**
  Je varnostni koncept, ki dovoljuje dostop do virov samo tistim, ki imajo za to dovoljenje.

- **Koreografija:**
  Je v smislu spletnih storitev specifikacija, kako naj potujejo sporočila med povezanimi, a različnimi programskimi komponentami in aplikacijami, da bi bilo doseženo čim boljše sodelovanje.

- **Mediacija:**
  Je lastnost sistema za integracijo aplikacij. Mediacija pomeni, da se sistem za integracijo aplikacij obnaša kot komunikacijski posrednik med večimi aplikacijami. Vedno, ko se v kateri izmed aplikacij zgodi dogodek, je sistem za integracijo aplikacij o tem obveščen in o dogodku obvesti tudi ostale aplikacije.

- **Orkestracija:**
  Je način sodelovanja storitev, kjer glavna storitev kliče ostale storiteve. Glavna storitev pozna zaporedje akcij in vmesnike ter obliko podatkov, ki jih vrne storitev, ki je bila klicana.

1. Povzetek

Integracija aplikacij z uporabo Microsoft Biztalk-a

Diplomska naloga preučuje povezovanje sistemov, in sicer v šibko sklopljene sisteme. Cilj diplomske naloge je implementacija rešitve s strežnikom BizTalk, ki bo omogočala izmenjavo podatkov med Registrom davčnih zavezancev Slovenije in ostalimi informacijskimi sistemi na Davčni upravi Republike Slovenije ter z zunanjimi sistemi in registri. Uporaba strežnika BizTalk bi omogočila, da bi lahko vse obstoječe informacijske sisteme na Davčni upravi Republike Slovenije povezali med seboj brez velikih prilagoditev teh sistemov, hkrati pa bi bili podatki v informacijskih sistemih Davčne uprave Republike Slovenije ves čas ažurni zaradi avtomatizirane izmenjave podatkov z zunanjimi informacijskimi sistemi in registri.

Diplomska naloga je sestavljena iz teoretičnega in praktičnega dela. Teoretičen del predstavljajo poglavja tri, štiri in pet. V prvem delu so opisane teoretične osnove za izdelavo konkretne rešitve v drugem delu. Prvi del obsega opis upravljanja s poslovnimi procesi in storitveno vodilo, ki sta ključni temi pri izdelavi praktičnega dela. V teoretičnem delu diplomske naloge so opisane tudi najpomembnejše funkcionalnosti produktov podjetij IBM, Oracle in Microsoft, ki podpirajo storitveno vodilo in storitveno usmerjeno arhitekturo. Teoretičen del vsebuje tudi podroben opis arhitekture strežnika BizTalk 2006 in orodij, ki strežniku BizTalk pripadajo. Praktičen del diplomske naloge obsega šesto poglavje. V praktičnem delu je opisana problemska domena, to je vpeljava rešitve na podlagi strežnika BizTalk 2006 in s tem nadomestitev obstoječe rešitve izmenjave podatkov Davčne uprave Republike Slovenije z zunanjimi registri. Predstavljena je analiza trenutnega stanja, načrt rešitve in implementacija aplikacije.

Ključne besede: storitveno vodilo, integracija aplikacij, upravljanje poslovnih procesov, Microsoft BizTalk

Abstract

Enterprise application integration with Microsoft Biztalk

This paper deals with loosely coupled integration of legacy systems and automated data exchange with external information systems. The objective of the paper is to analyze current solution, to prepare project plan and to implement a part of planned system. The solution has two main functions. The first one is to integrate legacy systems at DURS and the second is to automate the exchange of data from the national register of taxpayers with external registers and information systems. With this paper author introduces use of BizTalk Server at DURS.

The paper consists of two parts: the theoretical one and the practical one. The theoretical part includes the description of business process management and enterprise service bus which are important to accomplish second part of this paper. Theoretical part also describes main functionalities of IBM, Oracle and Microsoft software products, which have embedded support for enterprise service bus and service oriented arhitecture. In the theoretical part is also detailed representation of Microsoft BizTalk Server 2006 and its tools. In the practical part of this paper is described the problem which is to be solved with solution developed with BizTalk Server 2006. In the practical part is also described analysis of current situation, project plan and implementation of possible solution.

Keywords: enterprise service bus, enterprise application integration, business process management, Microsoft BizTalk

2. Uvod

Nobena aplikacija ni osamljen otok. Povezovanje sistemov med seboj je postalo nujno potrebno, a povezovanje sistemov pomeni mnogo več kot samo izmenjavanje podatkov med njimi. Z razvojem organizacij in njihovim prehodom na storitveno usmerjeno arhitekturo je postal njihov cilj izgradnja učinkovitih poslovnih procesov, ki povezujejo različne sisteme v povezano celoto. Da pa bi bilo povezovanje sistemov čimbolj univerzalno in prilagodljivo ter da bi bile implementacije povezanih sistemov neodvisne med seboj, poznamo danes različne standardne infrastrukture in tehnologije, ki to omogočajo, to so storitveno usmerjena arhitektura, upravljane s poslovnimi procesi, integracija aplikacij in storitveno vodilo. Z neodvisnim in prilagodljivim povezovanjem sistemov dosežemo šibko sklopljenost.

Cilj diplomske naloge je v praksi z uporabo strežnika Microsoft BizTalk 2006 povezati različne sisteme med seboj. Diplomska naloga predstavi metodologije, arhitekture in tehnologije, ki so pomembne za povezovanje sistemov. Z uporabo znanja pridobljenega v teoretičnem delu naloge sem izdelal analizo obstoječega stanja problemske domene, načrt rešitve in na koncu še implementiral aplikacijo.Pri izdelavi rešitve sem uporabil razvojno okolje strežnika BizTalk 2006 in tudi ostale komponente, ki jih ta sistem ponuja in olajšajo povezovanje sistemov ter ponujajo napredne možnosti spremljanja izvajanja povezovanja, prilagajanje izvajanja spremenljivim potrebam in prilagodljivo razširjanje komponent, ki sodelujejo v procesu povezave.

Diplomska naloga je sestavljena iz logično zaključenih celot, katere sestavljajo poglavja. Prvi del, ki obsega tretje in četrto poglavje opisuje temeljne metodologije, arhitekture in tehnologije namenjene povezovanju sistemov, to so upravljanje s poslovnimi procesi, integracija aplikacij, storitveno vodilo in storitveno usmerjena arhitektura. V okviru poglavja o storitvenem vodilu sem predstavil tudi programsko opremo, ki podpira rešitve problematike povezovanja sistemov. Naslednji del diplomske naloge obsega peto poglavje in podrobneje predstavi strežnik Microsoft BizTalk 2006, princip delovanja, njegove komponente in dodatna orodja namenjena razvoju rešitev z strežnikom BizTalk 2006.Tretji in hkrati zadnji del diplomske naloge zajema šesto poglavje. V tem delu diplomske naloge sem opisal razvoj konkretne rešitve povezovanja sistemov s strežnikom Microsoft BizTalk 2006. V tem delu sem predstavil problemsko domeno, izdelal analizo obstoječega stanja, izdelal načrt rešitve in končno tudi implementiral aplikacijo.

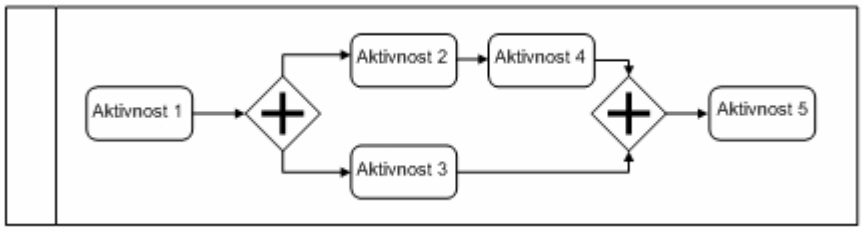
3. Upravljanje poslovnih procesov

3.1. Uvod

Poslovni proces je povezana množica aktivnosti, ki se usklajeno izvajajo v organizacijskem in tehničnem okolju. Izvajanje aktivnosti omogoča organizaciji uresničevanje njenih poslovnih ciljev. Vsak poslovni proces se izvaja v določeni organizaciji, a med svojim izvajanjem hkrati sodeluje s poslovnimi procesi znotraj organizacije in tudi s poslovnimi procesi zunaj organizacije[11].

Upravljanje poslovnih procesov(BPM) je skupek konceptov, metod in tehnik, ki podpirajo analiziranje, načrtovanje in upravljanje poslovnih procesov. Osnovna naloga BPM je eksplicitna predstavitev poslovnih procesov in njihovih aktivnosti ter omejitev, ki se pojavljajo med njihovim izvajanjem.[11]

BPM temelji na dejstvu, da je vsak izdelek, ki ga organizacija ponuja na trgu, rezultat izvajanja zaporedja aktivnosti. Poslovni proces je osnovni koncept, ki organizira izvajanje teh aktivnosti in omogoča razumevanje soodvisnosti aktivnosti. Informacijski sistemi pridobivajo znotraj BPM vse bolj na pomenu, saj je čedalje več aktivnosti, ki jih organizacije izvajajo, informacijsko podprtih. Organizacija lahko doseže prednost pred konkurenti, če uporablja programske produkte, ki ji omogočajo usklajevanje izvajanja aktivnosti v poslovnem procesu. Takšne programske produkte imenujemo sistemi za upravljanje poslovnih procesov(BPMS).

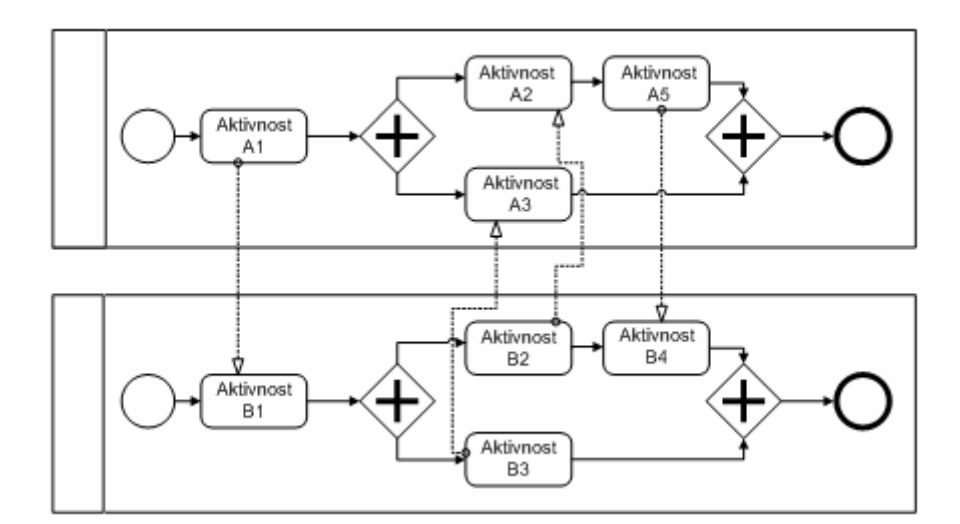


Slika 1 : primer predstavitve poslovnega procesa

Na sliki 1 vidimo predstavitev enostavnega poslovnega procesa z diagramom. Aktivnosti so predstavljene z zaobljenimi pravokotniki, medtem ko sta razdružitev in združitev vej poslovnega procesa predstavljeni z rombom.

Natančneje slika 1 predstavlja model poslovnega procesa. Vsak konkreten primer izvedbe modela poslovnega procesa pa imenujemo instanca poslovnega procesa.

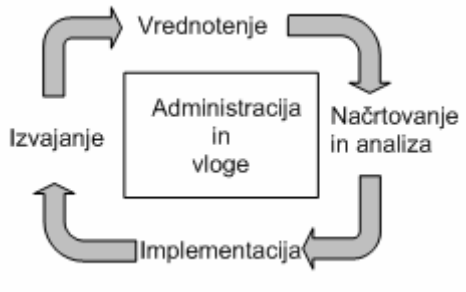
Ker se poslovni proces izvaja znotraj posamezne organizacije, opravlja usklajevanje aktivnosti znotraj poslovnega procesa BPMS. BPMS predstavlja centralno programsko komponento v informacijskem sistemu organizacije.



Slika 2: sodelovanje dveh poslovnih procesov

Slika 2 prikazuje sodelovanje dveh poslovnih procesov. Aktivnosti iz poslovnega procesa A in poslovnega procesa B, ki med seboj sodelujejo, so povezane s črtkanimi črtami, ki predstavljajo tok sporočil med sodelujočima procesoma. Sodelovanje množice poslovnih procesov med seboj imenujemo koreografija. Koreografija predstavlja povezavo med poslovnimi procesi, katera poteka brez centralnega nadzora, ki bi nadziral sodelovanje med poslovnimi procesi. Vsako sodelovanje med poslovnimi procesi temelji le na toku sporočil, zato je potrebno pred povezovanjem poslovnih procesov doseči dogovor o skupni koreografiji, da bi sodelovanje potekalo pravilno.

3.2. Razvojni cikel poslovnega procesa



Slika 3: razvojni cikel poslovnega procesa

3.2.1. Analiza in načrtovanje

Življenjski cikel poslovnega procesa se prične s fazo Analize in načrtovanja, v kateri je opravljen pregled in ocena poslovnega procesa ter njegovega organizacijskega in tehničnega okolja. Temelječ na izdelkih te faze so poslovni procesi identificirani, pregledani, ocenjeni in predstavljeni z modelom poslovnega procesa. Poslovni procesi so v tej fazi predstavljeni grafično, da lahko vsi sodelujoči uporabniki enostavno in učinkovito podajo svoje poglede, ki pripomorejo k izboljšanju poslovnega procesa.

Poleg tehnik modeliranja poslovnega procesa se v tej fazi uporabljajo tudi tehnike kontrole, preverjanja veljavnosti in simuliranja. Najprej se na podlagi raziskav in ocene stanja zgradi formalna predstavitev poslovnega procesa s pomočjo notacije za modeliranje poslovnih procesov(BPMN). Sledi preverjanje pravilnosti formalne predstavitve poslovnega procesa. Simulacija poslovnega procesa se v fazi analize in načrtovanja uporablja z namenom iskanja izvajanja nepravilnih zaporedij aktivnosti znotraj poslovnega procesa, katere bi lahko zmanjšale uporabnost poslovnega procesa.

3.2.2. Implementacija

Ko je izdelan veljaven načrt poslovnega procesa, sledi implementacija. Ker se bom v svojem diplomskem delu omejil le na računalniško podprte poslovne procese, bom v nadaljevanju opisoval le takšne.

V fazi implementacije se izbere platforma, s pomočjo katere se bo izvedla implementacija poslovnega procesa. Model poslovnega procesa se dopolni s tehničnimi podatki BPMS, ki bo zagotavljal izvajanje poslovnega procesa. BPMS je potrebno nastaviti tako, da bo zadostil zahtevam organizacijskega okolja in zahtevam poslovnega procesa, ki ga bo izvajal. BPMS mora zagotoviti tudi ustrezno podporo sodelovanju poslovnih uporabnikov in integraciji z obstoječimi informacijskimi sistemi. Slednje je zelo pomembno, saj ima danes že večina organizacij poslovne procese programsko podprte.

Ko je implementacija v sistemu za upravljanje poslovnih procesov zaključena mora nujno slediti testiranje. V ta namen se uporabljajo preverjene programske metode testiranja. Zelo pomembno vlogo imata tudi integracijski in zmogljivostni test, ki razkrijeta morebitne napake pri izvajanju procesa še preden je ta prešel v fazo izvajanja. Če je postopek testiranja uspešen, se sistem namesti na ciljno izvajalno okolje.

3.2.3. Izvajanje

Po zaključku faze implementacije se lahko prične izvajanje instanc poslovnega procesa. Poslovni procesi se izvajajo z namenom, da bi zadostili poslovnim ciljem organizacije. Sprožitev poslovnega procesa je posledica nekega dogodka, npr. sprejem naročila s strani kupca.

BPMS nadzoruje izvajanje instanc poslovnega procesa v skladu s pravili določenimi z modelom poslovnega procesa. Izvajanje poslovnega procesa mora zagotoviti ustrezno orkestracijo poslovnega procesa in izvajanje aktivnosti poslovnega procesa v skladu z omejitvami določenimi v modelu poslovnega procesa.

Zelo pomembna dejavnost v fazi izvajanja poslovnega procesa je spremljanje in nadziranje izvajanja, saj zagotavlja natančne informacije o stanju poslovnega procesa. Večina BPMS ima zato vgrajene programske komponente, ki omogočajo vizualno spremljanje izvajanja poslovnega procesa. S tem poenostavijo spremljanje in povečajo preglednost nad stanjem posameznih instanc poslovnega procesa

3.2.4. Vrednotenje

V fazi vrednotenja se uporabijo razpoložljive informacije o poslovnem procesu pridobljene v fazi izvajanja z namenom vrednotenja in izboljšanja modela poslovnega procesa in njegove implementacije.

V fazi vrednotenja se lahko ugotovi, da posamezna aktivnost traja predolgo zaradi pomanjkanja zmogljivosti računalniškega sistema na katerem se izvaja poslovni proces. Ta informacija je pomembna tudi za simulacijo v fazi analize in načrtovanja, ki sledi fazi vrednotenja. V tem se opazi cikličen princip razvoja poslovnega procesa.

3.2.5. Administracija in uporabniške vloge

Pri razvoju poslovnega procesa nastane veliko število elementov poslovnega procesa. Ti elementi morajo biti organizirano shranjeni v shrambah, ki omogočajo hitro pridobitev posameznega elementa iz shrambe za potrebe izvajanja in razvoja poslovnega procesa. Poleg zmogljivih shramb pa so za učinkovito izvajanje poslovnih procesov potrebni tudi uporabniki z ustreznim znanjem, ki opravljajo različne vloge v razvojnem ciklu poslovnega procesa.

Pri razvoju poslovnih procesov poznamo različne vloge, in sicer direktor poslovnih procesov, poslovni inženir, načrtovalec poslovnih procesov, poslovni analitik, skrbnik poslovnega procesa, sistemski inženir, razvijalec. Vse te vloge so pri razvoju in izvajanju poslovnih procesov močno prepletene, kar pomeni, da razvojni cikel poslovnih procesov zahteva zelo dobro organizacijsko strukturo organizacije v kateri se izvaja.

Izdelava konkretnega primera v drugem delu diplomske naloge zajema tudi razvoj poslovnega procesa, ki temelji na zmožnostih, ki jih ponuja ESB. Implementacija poslovnega procesa ne bi bila mogoča, če prej ESB ne bi zagotovil komunikacijske infrastrukture za sodelovanje storitev.

Pri razvoju poslovnega procesa sem se držal standardov in navodil, ki so opisana v tem poglavju. Najprej sem analiziral problemsko področje in z uporabo BPMN izdelal načrt poslovnega procesa. V fazi implementacije sem izbral, namestil in primerno nastavil platformo za izvajanje poslovnega procesa. To je v mojem primeru strežnik BizTalk 2006, strežnik Windows 2003 in strežnik Microsoft SQL 2005. Poleg implementacije samega poslovnega procesa sem poskrbel tudi za podporo uporabniškemu sodelovanju s pomočjo orodij za urejanje poslovnih pravil(BRC) in sistema za spremljanje izvajana poslovnih procesov(BAM). V fazi implementacije sem izvedel tudi testiranje poslovnega procesa in se s tem prepričal o pravilnosti delovanja. V fazi vrednotenja poslovnega procesa sem uporabil podatke zbrane v fazi izvajanja poslovnega procesa, da bi ugotovil, če se poslovni proces izvaja dovolj hitro in predvsem, če se vse instance poslovnega procesa uspešno zaključijo. Podatke iz faze vrednotenja sem nato uporabil za izboljšanje poslovnega procesa v naslednjem ciklu.

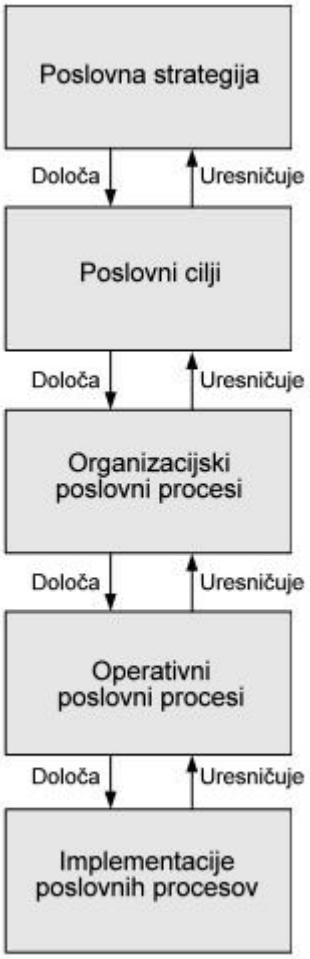
Metodologija BPM se mi zdi zelo pomembna pri razvoju poslovnih procesov, saj omogoča pregleden in obvladljiv proces razvoja. Pomembno je, da obstaja notacija BPMN, ki poenoti načrtovanje poslovnih procesov in se izogne nepotrebnim zapletom pri različnih predstavitvah

modelov poslovnih procesov. Prav tako je razvoj poslovnih procesov zelo olajšan z uporabo sistemov BPMS, ki omogočajo, da se posamezne komponente poslovnih procesov shranjujejo v shrambo in so ponovno uporabljive. Ker je razvoj poslovnih procesov lahko zelo zapleten je uporaba metodologij in tehnik BPM neizogibna in nujna, predvsem kadar je poslovnih procesov več in ti med seboj sodelujejo.

3.3. Klasifikacija poslovnih procesov

Pri upravljanju poslovnih procesov se srečamo z različnimi nivoji abstrakcije, in sicer od poslovne strategije pa vse do implementiranih poslovnih procesov. Vsi nivoji so prikazani na sliki 4.

Na najvišjem nivoju je določena poslovna strategija, ki opisuje dolgoročne razvojne načrte organizacije, ki ji bodo omogočali preživetje in razvoj na trgu. Na drugem nivoju se poslovna strategija razčleni na posamezne poslovne cilje. Poslovni cilji so lahko enostavni ali pa so sestavljeni iz več podciljev. Na tretjem nivoju najdemo organizacijske poslovne procese. To so poslovni procesi, ki so definirani na zelo visokem nivoju abstrakcije. Opredeljeni so opisno, in sicer s pričakovanimi vhodnimi in izhodnimi parametri procesa, z rezultati in s soodvisnostmi z drugimi organizacijskimi poslovnimi procesi. Na prvih treh nivojih se uporabljajo predvsem neformalne tehnike modeliranja. Za opredelitev poslovne strategije, poslovnih ciljev in organizacijskih poslovnih procesov se uporablja predvsem opisovanje in risanje diagramov brez upoštevanja standardov notacije.[11]



Slika 4: nivoji abstrakcije poslovnih procesov

Medtem ko organizacijski poslovni procesi bolj grobo opisujejo delovanje organizacije, operativni poslovni procesi določajo poslovanje podrobneje. Zato pripada ponavadi enemu organizacijskemu poslovnemu procesu več operativnih. V operativnih poslovnih procesih so predstavljene aktivnosti in povezave med njimi s pomočjo modelov poslovnih procesov. Na tej stopnji abstrakcije niso predstavljeni vidiki implementacije, a operativni poslovni procesi predstavljajo osnovo za razvoj implementacije poslovnih procesov. Implementacije poslovnih procesov vsebujejo podatke o izvajanju aktivnosti poslovnega procesa ter tehnične in organizacijske podrobnosti v katerih se bo poslovni proces izvajal. Načinov kako implementirati poslovni proces je več, vse od programiranja procedur in organizacijskih pravil pa do uporabe že razvitih produktov, ki omogočajo razvoj poslovnih procesov.[11]

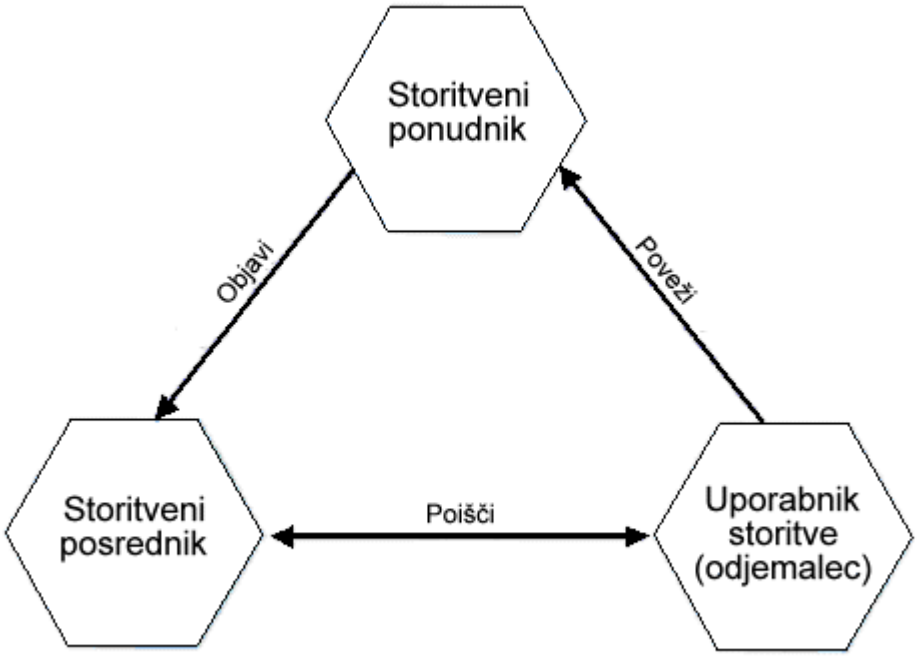
4. Storitveno vodilo

4.1. Storitveno usmerjena arhitektura

Storitveno usmerjena arhitektura(SOA) je arhitektura računalniških sistemov za podporo šibkemu sklopljenju razpršenih storitev, ki komunicirajo in sodelujejo na podlagi dogovorjenih pravil. SOA omogoča različnim aplikacijam sodelovanje in izmenjavo podatkov v poslovnih procesih. V takšnih primerih pravimo, da so aplikacije šibko sklopljene in so neodvisne od operacijskih sistemov in programskih jezikov, ki podpirajo aplikacije.

SOA razdeli funkcionalnost sistema v ločene storitve, katere so lahko porazdeljene po celotnem omrežju in lahko med seboj sodelujejo ter so ponovno uporabljive v različnih poslovnih procesih. Storitve komunicirajo med seboj z izmenjavo sporočil.

Podjetja so dolgo časa iskala način kako integrirati obstoječe aplikacije z namenom, da bi omogočili izvajanje poslovnih procesov, ki pokrivajo vse poslovne zahteve. V ta namen so poskušali z razvojem različnih večinoma neuspešnih arhitektur. Rešitev pa so našli v prilagodljivi in standardizirani arhitekturi, ki podpira medsebojno povezovanje različnih aplikacij in izmenjavo podatkov med njimi. In prav to SOA ponuja. SOA namreč posplošuje in poenostavlja gradnjo poslovnih procesov s povezovanjem različnih med seboj ločenih in neodvisnih storitev. V teh poslovnih procesih lahko sodelujejo različne aplikacije in uporabniki znotraj in zunaj podjetja. Novi poslovni procesi zgrajeni na principih SOA se odlikujejo z veliko prilagodljivostjo in univerzalnostjo.



Slika 5: model storitveno usmerjene arhitekture

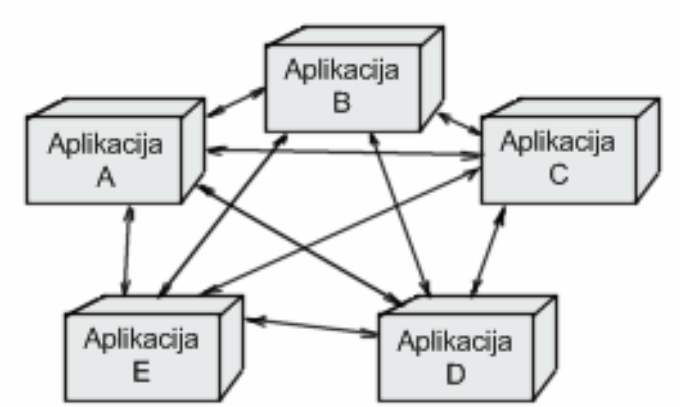
SOA gradi aplikacije in poslovne procese s povezovanjem programskih storitev. Storitve so samostojne enote, ki imajo opredeljeno funkcionalnost. Znotraj posamezne storitve ni vgrajenih klicev drugih storitev. Namesto vgrajenih klicev drugih storitev obstajajo protokoli, ki določajo kako storitve komunicirajo med seboj. Storitve se med seboj povezujejo v poslovne procese ali orkestracije in to je tudi način s katerim komunicirajo.

Glavni cilj SOA je združiti veliko število različnih funkcionalno zaokroženih enot- storitev v nove poslovne procese. Če posamezne storitve opravljajo veliko funkcionalnosti, potrebujemo za izvedbo določenega poslovnega procesa manjše število storitev. Po drugi strani pa nas storitve, ki opravijo veliko funkcionalnosti omejujejo pri prilagodljivosti in ponovni uporabljivosti.

4.2. Integracija aplikacij

Različne aplikacije, ki tečejo v organizacijah, ponavadi niso sposobne komunicirati med seboj v smislu izmenjave podatkov. Takšne aplikacije imenujemo informacijski otoki. Slabosti, ki jih prinaša nepovezanost aplikacij so neučinkovitost, večkratno zajemanje istih podatkov, shranjevanje istih podatkov na različnih lokacijah in nezmožnost avtomatiziranja poslovnih procesov.[9]

Integracija aplikacij je proces povezovanja aplikacij znotraj organizacije z namenom, da se doseže čim višja stopnja avtomatizacije poslovnih procesov pri čemur ni potrebno spreminjati obstoječih aplikacij in podatkovnih struktur. Problem s katerim se integracija aplikacij sooča je, da aplikacije ne tečejo na enakih operacijskih sistemih, da ne uporabljajo enakih podatkovnih baz in da niso implementirane v enakem programskem jeziku.

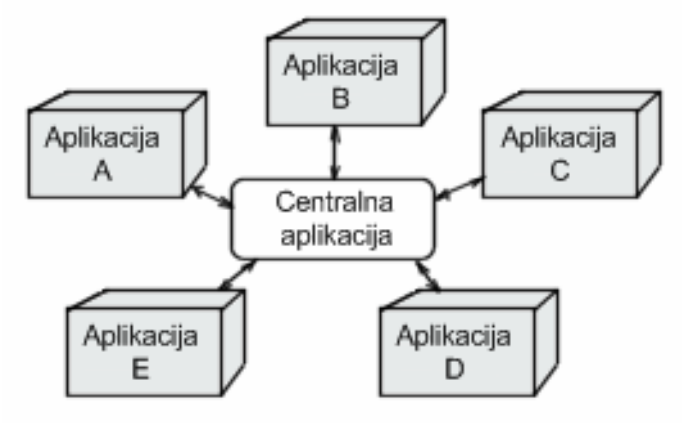


Slika 6: Povezovanje aplikacij po principu točka v točko

Povezovanje aplikacij, ki ne sledi pristopom, ki jih priporoča integracija aplikacij vodi do strukture povezave točka v točko, kjer je vsaka aplikacija povezana z vsako. Povezovanje aplikacij po principu točka v točko je prikazano na sliki 6. Takšen pristop vodi do nestrukturiranega sistema, ki ga je težko vzdrževati in je zato nepriporočljiv. Boljše rešitve ponuja integracija aplikacij. Integracija aplikacij pozna dve temeljni topologiji, in sicer:

- **Zvezdasta topologija**
  Prva integracijska arhitektura, ki je nadomestila povezovanje aplikacij tipa vsak z vsakim je bila zvezdasta topologija. Prikazana je na sliki 7. Zvezdasta integracija aplikacij je sestavljena iz centralne aplikacije, ki sprejema zahteve iz ostalih aplikacij povezanih z njo. Aplikacije v vozliščih so s centralno aplikacijo povezane z uporabo

vmesnikov, ki so nameščeni v vozliščih in so neodvisni od aplikacij. Aplikacija v centralnem vozlišču mora podpirati funkcionalnosti, kot so transformacija, preverjanje in usmerjanje sporočil ter orkestracijo.

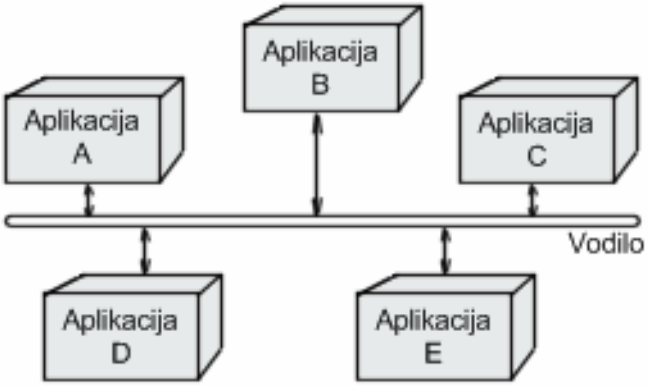


Slika 7: Zvezdasta integracija aplikacij

Integracija aplikacij na podlagi zvezdaste arhitekture je uporabna zaradi prednosti, ki jih prinaša. Prva in največja prednost pred povezovanjem aplikacij s topologijo vsak z vsakim je, da zvezdasta integracija aplikacij zmanjšuje število povezav med aplikacijami. Druga in prav tako zelo pomembna prednost je ta, da centralna aplikacija predstavlja vmesno plast med aplikacijami v vozliščih in s tem omogoča odstranitev in dodajanje novih aplikacij v vozliščih neodvisno od aplikacij v ostalih vozliščih.

Integracija aplikacij z zvezdasto topologijo pa ima tudi slabosti. Izpostavil bi predvsem dve, in sicer je prva ta, da je integracije v celoti odvisna od osrednje aplikacije. V primeru izpada centralne aplikacije je onemogočena vsakršna komunikacija med aplikacijami. Druga slabost pa je v tem, da so centralne aplikacije ponavadi zelo kompleksne in drage. Ker zvezdasta integracija aplikacij ni zadovoljila vseh potreb integracij so razvijalci naredili naslednji korak, to je integracija aplikacij z vodilom.

- **Vodilo**
  Vodilo je hrbtenica razvoja SOA. Vodilo je porazdeljena storitvena arhitektura, ki zagotavlja sporočilno vmesno opremo, pametno usmerjanje sporočil, transformacijo sporočil ter upravljalska orodja za nastavljanje, namestitev in spremljanje izvajanja storitev. Tipična integracija z uporabo vodila je predstavljena na sliki 8.



Slika 8

V bistvu je vodilo množica storitev vmesne opreme, ki ponujajo integracijske funkcionalnosti. Aplikacije se povezujejo na vodilo preko vmesnikov, ki ovijajo sistemske funkcionalnosti vodila in predstavljajo vmesno plast med aplikacijami in vodilom. Vmesniki ponujajo tudi varnostne storitve in omogočajo preslikavo sporočil.

Poznamo več različni vrst integracij aplikacij, v splošnem pa je vsaka integracija določena s štirimi parametri, to so:

- **Model komunikacije**
  Poznamo dva modela komuniciranja, in sicer sinhroni model komuniciranja, pri katerem odjemalec pošlje zahtevo storitvi in ne nadaljuje s procesiranjem dokler ne dobi odgovora. Drugi model komuniciranja je asinhroni. Pri asinhronem modelu komuniciranja odjemalec potem, ko pošlje zahtevo storitvi nadaljuje s procesiranjem. Pri asinhronem modelu komuniciranja odjemalec sprejme odgovor od storitve ter ga obdela in medtem prekine ostalo procesiranje. Sinhrona komunikacija se naprej deli na komunikacijo tipa zahteva/odgovor in enosmerno komunikacijo, asinhrona komunikacija pa se deli na komunikacijo tipa pošlji in pozabi ter objavi in se naroči.

- **Metoda integracije**
  Metoda integracije je pristop uporabljen za izgradnjo zahtev odjemalca do storitve. Poznamo dva pristopa in sicer pristop s sporočili in pristop z vmesniki.

- **Vmesna oprema**
  Vmesna oprema je poseben tip programske opreme, ki omogoča pošiljanje zahtev med programskimi komponentami z uporabo definiranih sporočil ali pa vmesnikov.

- **Storitve**
  Storitve so nepogrešljiva komponenta integracije, saj se integracija zaradi njih sploh implementira.

4.3. Šibka sklopljenost

Eden izmed glavnih principov SOA je šibka sklopljenost. Sklopljenost pomeni raven odvisnosti posameznega programskega modula od drugih programskih modulov. Ob pogostih spremembah programskih modulov, kar je pravilo pri današnjem razvoju programske opreme, je šibka sklopljenost zelo priporočljiv princip razvoja informacijskih sistemov.

Šibka sklopljenost je torej pristop, kjer so integracijski vmesniki aplikacij načrtovani in implementirani tako, da se v čim manjši meri opirajo na način implementacije aplikacij s katerimi komunicirajo. To vodi k temu, da sprememba v implementaciji ene aplikaciji ne pomeni, da je potrebno spremeniti tudi vse ostale aplikacije, ki z njo komunicirajo, da bi še vedno pravilno in učinkovito komunicirale med seboj.

Poznamo različne pomene šibke sklopljenosti, in sicer:

- **Šibka sklopljenost v časovnem smislu** pomeni, da integracija aplikacije deluje tudi, če je v določenem trenutku katera izmed sodelujočih aplikacij v neaktivnem stanju. V tem primeru bo aplikacija, ki je trenutno v izpadu, obdelala vse zahteve potem, ko bo ponovno delovala.

- **Šibka sklopljenost v podatkovnem smislu** pomeni, da sprememba oblike sporočil ene aplikacije ne povzroči nezmožnost komuniciranja z ostalimi aplikacijami.

- **Pri SOA pa šibka sklopljenost** pomeni, da je implementacija storitev, ki sodelujejo v poslovnem procesu skrita in neodvisna od implementacije ostalih storitev s katerimi sodeluje.

4.4. Storitveno vodilo

ESB je eden izmed temeljev SOA, saj predstavlja povezovalno plast med storitvami. Storitev je zelo širok pojem in ni omejena s protokolom(SOAP, HTTP), niti ni nujno, da je storitev opisana v točno določenem formatu(WSDL), čeprav so imeli vsi omenjeni standardi velik doprinos pri razvoju storitvenega vodila in SOA. Storitev je programska komponenta opisana z meta podatki, ki jih aplikacija razume. Meta podatki storitev so objavljeni, da bi omogočili ponovno uporabljivost storitve s strani drugih programskih komponent.[10] Programske komponente, ki bi rade uporabljale določeno storitev, morajo zato poznati le meta podatke storitve ne pa tudi njene implementacije. S takšnim pristopom se srečamo tudi pri dobrem načrtovanju programske opreme, kjer komponente z meta podatki definirajo vmesnike med komponentami in so tako posamezne komponente ponovno uporabljive. Meta podatki storitev se razlikujejo od vmesnikov v tem, da so meta podatki objavljeni, da bi omogočili ponovno uporabo storitev v šibko sklopljenih sistemih, ki so ponavadi povezani v omrežju.

Na tem mestu je potrebno razložiti, kaj pomeni objaviti opis storitve. Opis storitve, ki ga objavi ponudnik storitve je namenjen dostopu odjemalcev do storitve. Storitveno vodilo formalizira objavo z uporabo registra storitev, ki so pripravljene na klice odjemalcev storitev, kateri se želijo povezati s storitvami ponudnikov. Objava ponudnikov in odjemalcev storitev omogoča, da se meta podatki urejajo v registru storitvenega vodila in s tem olajša posodabljanje povezav med storitvami. Kljub temu pa se lahko ponudniki in odjemalci storitev povežejo preko storitvenega vodila ne da bi se prej vpisali v register. Njihovo sodelovanje bo potekalo nemoteno, le da ne bodo izkoristili vseh dinamičnih zmožnosti, ki jih ponuja storitveno vodilo.[10]

Storitveno vodilo vpiše meta podatke o storitvah v register na tri različne načine, in sicer:

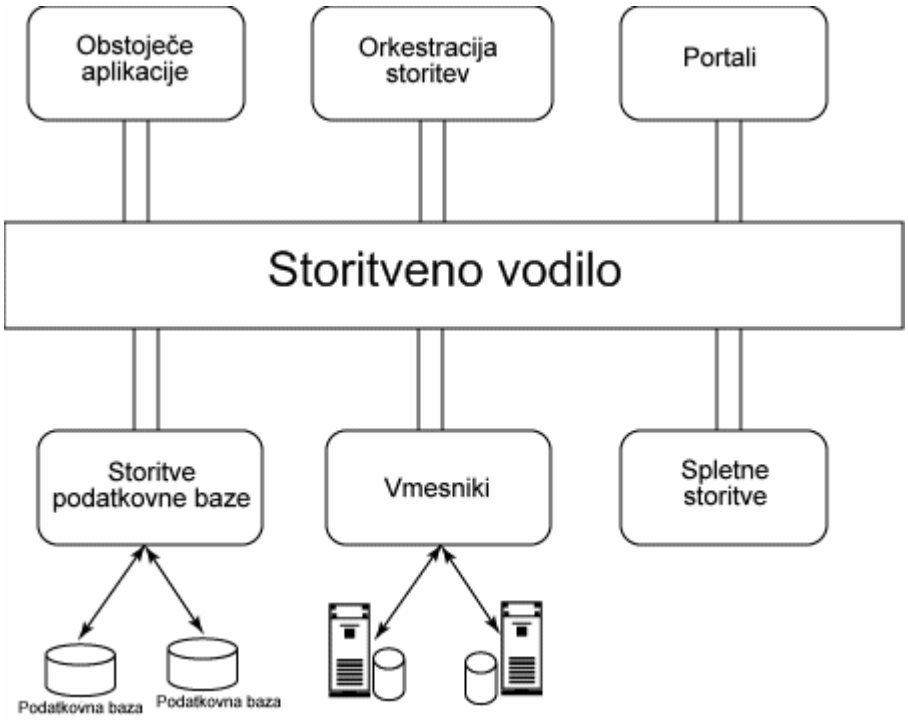
- **Ob nameščanju storitve v produkcijsko okolje**: ob nameščanju storitve se hkrati zapišejo tudi njeni meta podatki v register.

- **Ko je storitev že nameščena v produkcijskem okolju**: meta podatki storitve se vpišejo v register naknadno po opravljeni namestitvi storitve v produkcijsko okolje.

- **Storitveno vodilo samo poišče že nameščene storitve v produkcijskem okolju**: Storitveno vodilo samo avtomatsko poišče že nameščene storitve in sodelovanje med njimi ter na podlagi tega dopolni register.

Storitveno vodilo je le infrastruktura za medsebojno povezovanje storitev, tako da storitveno vodilo ne vključuje tudi poslovne logike ponudnikov in odjemalcev storitev niti ne vključuje vsebnikov, ki gostijo storitve. Gostitelji storitev in samostojne aplikacije se lahko povezujejo na storitveno vodilo na različnih nivojih integracije, odvisno od podprtih protokolov in komunikacijskih standardov, ki jih uporabljajo. Večina vsebnikov, kot sta J2EE aplikacijski strežnik ali Microsoft .NET se integrira z storitvenim vodilom z uporabo HTTP/SOAP protokola, a to ni edini način. Potem, ko storitveno vodilo dostavi podatke do vsebnika, je njegovo delo zaključeno. Znotraj vsebnika se klici storitev posredujejo med različnimi računalniki ali pa se odgovor prebere iz predpomnilnika. To je za storitveno vodilo popolnoma nepomembno. Storitveno vodilo je tudi povezovalna plast mehanizma za izvajanje poslovnih procesov, ki zagotavlja orkestracijo storitev. Mehanizem za izvajanje poslovnih procesov je odgovoren zato, da se pravilne storitve izvajajo v pravilnem vrstnem redu. Mehanizem za izvajanje poslovnih procesov, poverja storitvenemu vodilu odgovornost za prenos in usmerjanje storitvenih zahtev.

Temeljno načelo SOA je, da je implementacija odjemalcev storitev neodvisna od implementacije storitev, katere uporabljajo. Zato ni presenetljivo, da je storitveno vodilo v bistvu neviden za ponudnike in odjemalce storitev, ki storitveno vodilo uporabljajo za povezovanje.[10] Razvijalec lahko pri razvoju odjemalcev uporablja aplikacijske programske vmesnike(API)storitev, ne da bi mu bilo potrebno razmišljati o tem ali gredo njihove zahteve neposredno do storitev ali jih posreduje storitveno vodilo. Podobno je lahko storitev implementirana s katerimkoli razvojnim orodjem v kateremkoli programskem jeziku brez uporabe posebne programske kode, ki bo storitev naredila dostopno preko storitvenega vodila. Navkljub temu pa storitveno vodilo prevzame odgovornost za veliko infrastrukturnih zahtev, katerim bi sicer morali zadostiti v programski kodi storitve in odjemalca. Takšen primer je, ko lahko razvijalci uporabljajo API storitev in jim ni potrebno dodajati logike ki bi skrbela za varnost.



Slika 9: predstavitev modela storitvenega vodila

Storitveno vodilo virtualizira storitve, ki so dostopne preko vodila. Odjemalcu storitve se ni potrebno v svoji aplikacijski logiki niti v namestitvi zavedati programskega jezika, produkcijskega okolja, strojne opreme, mrežnega naslova ali pa trenutne dostopnosti storitve. Pri uporabi storitvenega vodila ni nujno niti, da odjemalec in ponudnik storitve uporabljata isti protokol. Odjemalec se poveže na vodilo, ki nato prevzame vso odgovornost za dostavo zahteve do storitve in je pri tem zelo zanesljivo.[10] Infrastruktura vodila je zelo prilagodljiva in omogoča razširjanje in krčenje vodila v odvisnosti od zahtev okolja, katerega podpira.

Prilagodljivost, ki jo s seboj prinaša SOA in virtualizacija, ki iz nje izhaja je realizirana z dinamičnimi lastnostni storitvenega vodila. V vse meta podatke ter pogoje in omejitve, ki se uporabljajo pri povezovanju storitev in odjemalcev se lahko vpogleduje in se jih uporablja ter spreminja med samim izvajanjem. Na primer: pojavi se nova implementacija neke storitve v regiji, kjer je doslej še ni bilo. Vse zahteve do storitve v isti regiji se preusmerijo na novo storitev, ne da bi bilo potrebno spreminjati nastavitve odjemalcev. Ta prilagodljivost je neposredna posledica vloge, ki jo opravlja register storitvenega vodila. Ker so vsi pomembni podatki storitev in odjemalcev shranjeni v registru, so nam vedno na voljo najnovejši podatki, katere lahko poljubno spreminjamo ne da bi bilo potrebno zato postaviti celoten sistem za določen čas v neaktivno stanje.

Da bi storitveno vodilo doseglo čim višji nivo prilagodljivosti, si storitve zahteve interpretirajo kot sporočila. Nad temi sporočili nato izvajajo poljubne operacije oziroma jih posredujejo naprej. Posredovanje sporočil je sestavni del storitvenega vodila, ki zagotavlja npr. preslikovanje sporočil ali pa pošlje sporočilo drugi storitvi, kadar je odzivni čas prvotno naslovljene predolg. Storitveno vodilo gre v prilagodljivosti še dlje, saj ni potrebno, da je mehanizem ki zagotavlja posredovanje sporočil sestavni del storitvenega vodila, temveč je lahko tudi izdelek drugega proizvajalca.

Posredovanje sporočil je način, s katerim lahko storitveno vodilo zagotovi, da se odjemalec uspešno poveže s storitvijo. Če storitev zahteva določen format podatkovnega polja v

sporočilu in odjemalec uporablja drugačen format za isto polje, potem ponuja mehanizem posredovanja sporočil rešitev. Mehanizem preslika podatkovno polje v sporočilu v drug format, ki ustreza storitvi in tako storitveno vodilo dostavi zahtevo v ustreznem formatu. Enako velja tudi za komunikacijo v nasprotni smeri, od storitve k odjemalcu. Storitveno vodilo reagira dinamično na zahteve odjemalcev in storitev, kadar so le-ti opisani in shranjeni kot meta podatki v registru storitvenega vodila.

Mediacija je lastnost integracije aplikacij. Mediacija je lahko vgrajena v storitveno vodilo, da bi zagotovila spremljanje stanja in vsebine sporočil zahtev in odgovorov, ko ti potujejo skozi sistem. Mediacija lahko preusmerja storitvene zahteve do lokacij drugih ekvivalentnih storitev, kadar so prvotno naslovljene v izpadu ali pa do novih, bližnjih storitev, če se takšne na novo pojavijo. Mediacija lahko tudi preverja sporočila v smislu pravilnosti formata sporočil, vrednosti podatkov ali pa uporabniške avtentikacije in avtorizacije. Z mediacijo in ostalimi sistemskimi upravljalskimi zmogljivostmi storitveno vodilo zagotavlja, da so šibko sklopljeni sistemi lahko nastavljivi tudi v produkcijskem okolju.

Večina zgoraj naštetih zmogljivosti mediacije so temeljne lastnosti storitvenega vodila. Vse značilnosti so zelo enostavno nastavljive. Tako lahko na primer splošno usmerjanje ki temelji na uporabi usmerjevalne tabele spremenimo, da bo uporabljalo posebno tabelo, ki jo določi razvijalec in kot ključ v sporočilih uporabljalo polja, ki jih prav tako določi razvijalec. Storitveno vodilo zagotavlja tudi orodja za spreminjanje nastavitev medsebojnega sodelovanja storitev. To so orodja, ki prikazujejo aktivne storitve, orodja, ki omogočajo povezovanje storitev med seboj, orodja, ki dodajajo določeno varnostno politiko posamezni storitvi ali pa skupini storitev, orodja, ki prepoznajo napake in jih s pomočjo mediacije sama odpravljajo.

V zgornjih odstavkih sta bila velikokrat uporabljena izraza odjemalec in storitev. Oba pojma sta enakovredna partnerja v medsebojni komunikaciji, s tem da je odjemalec tisti partner v tem razmerju, ki vedno začenja komunikacijo. Komunikacija se lahko nadaljuje bodisi s pošiljanjem sporočila s strani odjemalca ali odjemalčevim sprejemom sporočila s strani storitve. Storitveno vodilo podpira veliko različnih tipov komunikacije, to so lahko enosmerno sporočanje, kot tudi sporočanje tipa zahteva/odgovor. Komunikacija je lahko sinhrona kot tudi asinhrona. Komunikacija lahko temelji na principu objavi in se naroči, kjer se odgovor razpošlje vsem odjemalcem, ki so se prijavili da želijo prejeti odgovor. Komunikacija pa lahko temelji tudi na kompleksnem dogodkovno vodenem pristopu, kjer je opazovano zaporedje dogodkov, ki posledično sproži nek dogodek kot rezultat.

Prednosti ESB pred sedanjo implementacijo v moji problemski domeni, ki namesto vodila uporablja zvezdasto omrežje z aplikacijskim strežnikom v središču je več. Storitveno vodilo je enostavno razširljivo. Medtem, ko bi bilo potrebno v zvezdasti infrastrukturi dograditi logiko aplikacij na aplikacijskem strežniku in jo prilagoditi dodatnemu vozlišču, lahko na vodilo preko vmesnikov priklopimo poljubno število novih aplikacij ali storitev. Prednost vodila je tudi v tem, da so vmesniki za priklop večine razširjene programske opreme že razviti in dostopni. ESB je tudi infrastruktura, ki omogoča povezovanje več vodil med seboj, česar zvezdasta integracija aplikacij ne omogoča. ESB nudi tudi veliko podporo spletnim storitvam, kar je še en korak naprej proti čim večji neodvisnosti storitev in šibki sklopljenosti. ESB podpira tudi veliko različnih komunikacijskih modelov, in sicer od objavi/prijavi do sinhrone dvosmerne komunikacije ter asinhrone pošlji in pozabi.

ESB predstavlja zelo prilagodljivo arhitekturo, zato sem se v diplomski nalogi odločil predstaviti njene prednosti pred obstoječo rešitvijo sistema izmenjav na DURS tudi z implementacijo konkretnega primera v drugem delu naloge.

4.4.1. Microsoft in storitveno vodilo

Microsoft ne ponuja produkta, ki je označen samo kot storitveno vodilo, temveč integracijske tehnologije, ki ponujajo vse funkcionalnosti storitvenega vodila in še več. Njihova ponudba na tem področju je osredotočena na strežnik BizTalk, ki je integracijski strežnik in strežnik za izvajanje poslovnih procesov ter na WCF, to je ogrodje namenjeno razvoju varnih in zanesljivih spletnih storitev.[6]

Strežnik BizTalk je integracijski strežnik in strežnik za izvajanje poslovnih procesov. Omogoča šibko sklopljeno povezovanje široke palete sistemov, in sicer MSMQ, SAP, spletnih storitev ter ostalih sistemov in aplikacij. Poleg integracije ponuja BizTalk tudi popolno podporo izvajanju poslovnih procesov, spremljanju izvajanja poslovnih procesov ter razvoju in izvajanju poslovnih pravil.[12]

Strežnik BizTalk omogoča podporo tudi dolgotrajnim transakcijam(njihovo stanje med neaktivnostjo shrani v podatkovno bazo, tako da ne obremenjujejo računalniškega spomina po nepotrebnem) ter prilagodljivim preslikavam sporočil iz ene oblike v drugo.[3]

BizTalk vključuje tudi ostala orodja, ki neposredno ne podpirajo izvajanja poslovnih procesov, ampak prinašajo dodatne funkcionalnosti. Z BAM lahko poslovni analitik določi, katere podatke ki se pojavljajo v poslovnih procesih, bo zbiral in kako jih bo interpretiral. BAM omogoča poslovnim uporabnikom spremljanje katere aktivnosti in procesi se izvajajo ter v kakšnih stanjih so. Strežnik BizTalk beleži stanje vsake instance poslovnega procesa in predstavi njihovo stanje tehnično nepoučenim uporabnikom v obliki Excelovih preglednic.

Strežnik BizTalk ponuja veliko večjo funkcionalnost, kot je samo storitveno vodilo. Uporabniki strežnika BizTalk ga lahko po namestitvi uporabljajo kot produkt, ki ponuja samo funkcionalnosti storitvenega vodila, ali pa ga lahko uporabljajo le za izvajanje poslovnih procesov. Najučinkovitejša pa je seveda kombinacija obojega.

Microsoft daje poudarek razvoju rešitve, ki ponuja več kot le sporočilni sistem storitvenega vodila, saj želi zadostiti širšemu naboru potreb vključno z validacijo in preslikavo sporočil iz ene oblike v drugo, orkestracijo poslovnih procesov, spremljanjem izvajanja poslovnih procesov, razvojem in izvajanjem poslovnih pravil.

4.4.2. IBM WebSphere in storitveno vodilo

Aplikacijski strežnik IBM WebSphere je glavni produkt IBM WebSphere programske platforme in je ključno orodje pri razvoju storitveno usmerjenih rešitev. Jedro aplikacijskega strežnika IBM WebSphere je aplikacijska platforma za Java 2EE in spletne storitve. Aplikacijski strežnik WebSphere je zmogljiva rešitev, ki pomaga pri razvoju, integraciji in upravljanju aplikacij na zahtevo.

Namestitev aplikacijskega strežnika WebSphere je neodvisna od sistemske platforme in lahko teče na vsakem strežniku. Aplikacijski strežnik WebSphere ponuja prilagodljivo, standardizirano in odprto aplikacijsko infrastrukturo potrebno za razvoj storitveno usmerjenih rešitev. Glavne značilnosti so:

- **Enostaven in hiter razvoj ter namestitev:** Aplikacijski strežnik WebSphere pomaga pri razvoju rešitev s svojimi zmogljivimi in enostavno uporabljivimi funkcionalnostmi.

- **Varno, razširljivo in visoko razpoložljivo produkcijsko okolje:** Aplikacijski strežnik WebSphere zagotavlja varno, razširljivo in zanesljivo okolje za izvajanje aplikacij in storitev.

- **Izboljšane komunikacijske storitve:** Aplikacijski strežnik WebSphere pomaga povečati prilagodljivost poslovanja s pomočjo ponovne uporabljivosti storitev, hkrati pa ponuja uporabo obstoječih storitev na nov način.

- **Učinkovito upravljanje aplikacij:** Aplikacijski strežnik WebSphere ponuja učinkovita in enostavna orodja za upravljanje.

Sestavni del aplikacijskega strežnika WebSphere predstavlja tudi storitveno vodilo IBM WebSphere. Storitveno vodilo aplikacijskega strežnika IBM WebSphere ima naslednje značilnosti:

- Podpora različnim komunikacijskim protokolom, kot so TCP/IP, SSL, HTTP.
- Podpora različnim modelom komunikacije: zahteva/odgovor, točka v točko, objavi/naroči.
- Napredna podpora spletnim storitvam.
- Integrirano razvojno okolje WebSphere zagotavlja integrirano, interaktivno in vizualno razvojno okolje za hiter razvoj integracijske logike.
- Izboljšan preslikovalnik formata sporočil, ki podpira vse kompleksne podatkovne tipe in ustreza splošnim standardom.
- Omogoča obravnavo napak in loči med napakami v poslovni logiki in izvajalnem okolju.
- Ponuja napredno podporo razvoju WSDL in XSD dokumentov in podpira uporabo različnih standardiziranih XML shem.
- Omogoča tesno integracijo z ostalimi IBM produkti, da bi se razširil v združen model storitvenega vodila(povezovanje različnih instanc storitvenega vodila).
- Preslikovalnik formatov sporočil se lahko enostavno vgradi znotraj toka sporočila(procesa) in s tem izboljša mediacijo.[2]

4.4.3. Programski paket Oracle SOA Suite in storitveno vodilo

Oraclov produkt na področju SOA je Oracle SOA suite. Oracle SOA Suite je integrirana skupina produktov, ki omogoča razvoj, namestitev in upravljanje programskih rešitev. Oracle SOA suite je produkt, ki ustreza sprejetim standardom. Njegova hitra in enostavna razširljivost omogočata prilagoditev in sodelovanje z obstoječo informacijsko infrastrukturo v organizaciji.

Glavne prednosti Oracle SOA Suite so:

- Oracle SOA Suite je vsestranska rešitev: izboljša varnost, zniža stroške vzdrževanja in poveča skalabilnost.
- Enostavno razširljiva rešitev: omogoča hitro in enostavno modularno razširitev z ostalimi Oracle produkti.

Oracle SOA suite sestavljajo naslednji moduli:

- **Oracle storitveno vodilo:** Storitveno vodilo Oracle je preverjena in razširljiva integracijska platforma SOA, ki omogoča enostavno in standardizirano integracijo. Storitveno vodilo Oracle je načrtovano za povezovanje ter mediacijo in upravljanje sodelovanja med heterogenimi storitvami, aplikacijami, sistemi in drugimi instancami storitvenega vodila v omrežju organizacije. Oracle zagotavlja vgrajen nadzor in upravljanje z zmogljivostmi storitvenega vodila. Storitveno vodilo Oracle je ključna in najpomembnejša komponenta Oracle SOA Suite in predstavlja sporočilno hrbtenico.

- **Oracle JDeveloper:** Oracle JDeveloper je integrirano razvojno okolje za razvoj rešitev SOA in za razvoj javanskih aplikacij. JDeveloper je možno vgraditi v vsa Oraclova in tudi v ostala razvojna okolja.

- **Mehanizem za izvajanje poslovnih pravil Oracle Business Rules:** Oracle Business Rules omogoča razvoj prilagodljivejših aplikacij in poslovnih procesov, saj lahko poslovni uporabniki in ostali nerazvijalci enostavno definirajo in spreminjajo poslovna pravila z uporabo spletnega vmesnika. Oracle Business Rules zagotavlja hitrejše in enostavnejše ter predvsem cenejše spreminjanje poslovne logike v aplikacijah.

- **Razvojno orodje za orkestracije Oracle BPEL Process Manager:** Oracle BPEL Process Manager omogoča organizacijam orkestriranje različnih aplikacij in spletnih storitev v poslovne procese.

- **Orodje za razvoj integracij Oracle Business-to-Business Integration:** Dandanes predstavlja tesno sodelovanje s poslovnimi partnerji veliko poslovno prednost. Oracle Business-to-Business Integration uporablja pri povezovanju standardne protokole, da bi omogočil enostavno integracijo z namenom avtomatiziranja poslovnih procesov v katerih sodelujejo poslovni partnerji.

- **Sistem za spremljanje izvajanja poslovnih procesov Oracle Business Activity Monitoring:** Oracle Business Activity Monitoring (Oracle BAM) je popolna rešitev, ki omogoča razvoj interaktivnih rešitev, ki delujejo v realnem času in omogočajo spremljanje izvajanja poslovnih procesov in storitev. Oracle BAM ponuja vodstvu podjetij informacije, ki so ključnega pomena za njihove poslovne odločitve.

- **Upravljalec spletnih storitev Oracle Web Services Manager:** Večina organizacij se odloči za implementacijo SOA, da bi dosegla večjo učinkovitost z ponovno uporabljivimi storitvami. Ko so ponovno uporabljive storitve prisotne v organizaciji, je ključnega pomena spremljanje njihove uporabljivosti, sicer lahko hitro izgubimo nadzor nad SOA. Oracle Web Services Manager je rešitev za spremljanje sodelovanja med storitvami in zagotavlja, da je njihova uporabljivost ves čas pod nadzorom.[8]

4.4.4. Primerjava produktov

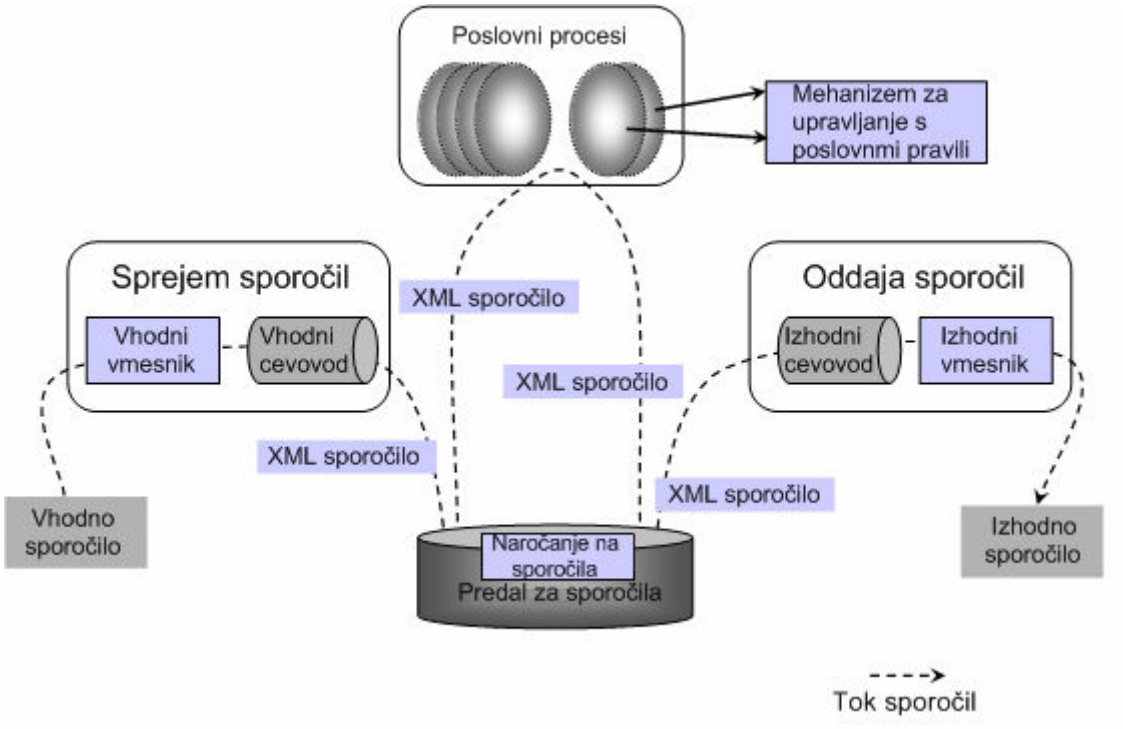
Vsi trije produkti Microsoft BizTalk, IBM Websphere in Oracle SOA Suite ponujajo temeljne rešitve za problematiko s katero se ukvarja ta diplomska naloga, to je storitveno vodilo in razvoj poslovnih procesov, ki temelji na uporabi storitvenega vodila. Vsi trije produkti omogočajo povezovanje sistemov z uporabo storitvenega vodila, ki predstavlja sporočilno hrbtenico med sistemi. Na temelju storitvenega vodila, ki je sporočilno infrastrukturo za poslovne procese, omogočajo navedeni produkti razvoj poslovnih procesov. Poslovni procesi lahko tečejo med različnimi sistemi. Vsako izmed navedenih orodij ima svoje integrirano razvojno okolje za razvoj poslovne logike v obliki poslovnih procesov. Razvoj s strežnikom Microsoft BizTalk temelji na .NET ogrodju, ostala produkta pa uporabljata javansko platformo.

Medtem ko lahko IBM Websphere in Oracle SOA suite tečeta na različnih operacijskih sistemih(Windows, AIX, Solaris, Linux) in uporabljata za podporo podatkovne baze različnih proizvajalcev(IBM DB2, podatkovna baza Oracle, strežnik Microsoft SQL), je Microsoft BizTalk popolnoma vezan na Microsoftovo platformo. Microsoft BizTalk potrebuje za svoje delovanje podporo operacijskega sistema Windows Server in podatkovno bazo Microsoft SQL.

Vsi trije produkti ponujajo podporo spletnim storitvam. Omogočajo enostavne spletne storitve z uporabo SOAP, prav tako pa omogočajo napredne spletne storitve, ki podpirajo najbolj pomembne standarde spletnih storitev, kot so WS-Security, WS-Addressing in WS-Reliable messaging. IBM Websphere in Microsoft BizTalk omogočata uporabo naprednih spletnih storitev z uporabo dodatnih vmesnikov, in sicer Websphere z Websphere Web Services Feature Pack, BizTalk pa z adapterjem WCF. Storitveno vodilo Oracle pa ima že vgrajeno podporo standardom WS-Security, WS-Addressing in WS-Reliable messaging.

5. Strežnik Microsoft BizTalk 2006

Strežnik Microsoft BizTalk 2006 omogoča uporabnikom, ki želijo razviti poslovne procese kateri bi zajemali sodelovanje več aplikacij, dva pomembna mehanizma. Prvi je enostaven način načrtovanja in implementacije logike poslovnega procesa, drugi mehanizem pa je namenjen komunikaciji med aplikacijami, ki bodo sodelovale v poslovnem procesu.



Slika 10 : Prikaz arhitekture strežnika BizTalk

Kot je razvidno iz slike 10 so sporočila sprejeta v sistem s pomočjo vhodnega vmesnika. Nato se sporočilo obdela v vhodnem cevovodu(pretvorba v XML format, preveri se pravilnost, preveri se digitalni podpis...). Nato se sporočilo shrani v predal za sporočila. Poslovna logika je implementirana kot eden ali več poslovnih procesov, ki se povežejo z mehanizmom za izvajanje poslovnih procesov.

Vsak poslovni proces v sistemu se prijavi na določen tip sporočila, tako da je za vsako sporočilo, ki pride v predal za sporočila točno določeno kateremu poslovnemu procesu mora biti posredovano. Ko proces prejme sporočilo, izvede aktivnosti, ki so določene v definiciji poslovnega procesa. Ko se proces zaključi, ponavadi kot rezultat, vrne novo sporočilo in ga shrani v predal sporočil. To sporočilo se obdela v izhodnem cevovodu in se pošlje iz sistema s pomočjo izhodnega vmesnika.[1]

5.1. Povezovanje sistemov

Nepogrešljiva zahteva integracije računalniških sistemov je učinkovita izmenjava sporočil med različnimi aplikacijami, ki tečejo na različnih računalniških sistemih. Zato strežnik BizTalk podpira različne protokole in formate sporočil. Kot je opisano v nadaljevanju je velik del funkcionalnosti strežnika BizTalk namenjen prav povezovanju sistemov.

5.1.1. Vmesniki(pošiljanje in sprejemanje sporočil)

Ker se strežnik BizTalk zanaša na uporabo vmesnikov, je sposoben komunicirati z velikim številom različnih aplikacij. Vmesnik je implementiran komunikacijski mehanizem v strežniku BizTalk, in sicer poznamo v strežniku BizTalk 2006 naslednje standardne vmesnike:

- **Vmesnik za spletne storitve**: omogoča pošiljanje in sprejemanje sporočil z uporabo protokolov SOAP in HTTP. Ker je SOAP temelnji protokol spletnih storitev, je ta vmesnik najpomembnejši element strežnika BizTalk pri komunikaciji v SOA.

- **Datotečni vmesnik**: omogoča branje in pisanje datotek v datotečnem sistemu Windows. Ti vmesniki so pogosto uporabljeni, ker so aplikacije ponavadi na istem datotečnem sistemu, bodisi lokalno ali pa preko računalniškega omrežja.

- **Vmesnik HTTP**: omogoča pošiljanje in sprejemanje sporočil z uporabo HTTP. Strežnik BizTalk izpostavi enega ali več URL na katere lahko ostale aplikacije pošiljajo sporočila in preko katerih tudi BizTalk pošilja sporočila drugim aplikacijam.

- **Vmesnik za sporočilne vrste**: omogoča pošiljanje in sprejemanje sporočil preko sporočilnih vrst.

- **Vmesnik za SMTP**: omogoča pošiljanje in sprejemanje sporočil z uporabo SMTP.

- **Vmesnik SQL**: omogoča branje in pisanje podatkov iz in v podatkovno bazo.

Poleg vseh naštetih vmesnikov, ki jih podpira strežnik BizTalk, pa omogoča tudi razvoj lastnih vmesnikov in dodajanje drugih vmesnikov, ki niso standardno vgrajeni v strežnik BizTalk.[12]

Vmesniki so temelj povezljivosti, ki jo strežnik BizTalk omogoča in so edini način, da se aplikacije in storitve povežejo na ESB strežnika BizTalk. Uporaba vmesnikov omogoča enostaven priklop nove aplikacije na ESB, saj je potrebno le izbrati pravilni vmesnik in ga pravilno nastaviti. Pomembno vlogo ima vmesnik za spletne storitve. Ta vmesnik omogoča klicanje spletnih storitev in se s tem še bolj približa SOA, saj lahko vsako aplikacijo ovijemo v spletno storitev.[4]

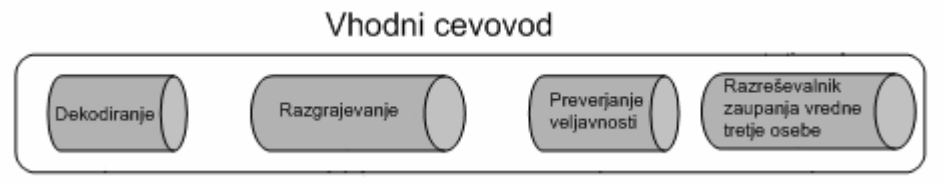
V implementaciji, ki sem jo izvedel v okviru tega diplomskega dela, sem zaradi takšnih poslovnih zahtev uporabil le datotečni vmesnik in vmesnik SQL. Ob tem sem se prepričal, da omogočata hitro in prilagodljivo povezavo.

5.1.2. Cevovodi(obdelava sporočil)

Poslovne aplikacije, ki sodelujejo, komunicirajo z izmenjavo različnih tipov dokumentov, to so naročilnice, računi, ponudbe,… . Zato je za strežnik BizTalk zelo pomembno, da zna ustrezno obdelovati sporočila, ki predstavljajo takšne dokumente. Takšna obdelava poteka v

več stopnjah in v ta namen strežnik BizTalk uporablja mehanizem imenovan cevovod. Obstajata vhodni cevovod za vhodna sporočila ter izhodni cevovod za izhodna sporočila.

Najbolj enostaven primer obdelave sporočila v vhodnem cevovodu je pretvorba sporočila v XML format, ker strežnik BizTalk interno uporablja le XML format sporočil, kar pa ni nujno pravilo za zunanje aplikacije.



Slika 11: Vhodni cevovod

Strežnik BizTalk tudi med vhodnimi in izhodnimi cevovodi ponuja standardne vgrajene komponente za posamezne stopnje cevovoda, a hkrati omogoča tudi implementacijo komponent po lastnih zahtevah. Standardne stopnje vhodnega cevovoda, ki jih prikazuje tudi slika 11 so:

- **Dekodiranje**: v tej stopnji cevovoda se uporablja MIME dekodiranje. Ta komponenta zna pretvoriti sporočilo ter njegove priponke v XML format in tudi preveriti veljavnost digitalnega podpisa.

- **Razgrajevanje:** v tej fazi se najpogosteje uporabljata dve vgrajeni komponenti, in sicer je prva razgrajevalnik tekstovnih datotek, ki zna iz logičnih zapisov v datoteki razbrati sporočila in za posamezno sporočilo ustvariti XML dokument. Druga komponenta je razstavljalec XML datotek. Ta komponenta pa razbije prejeti XML dokument v enega ali več XML sporočil. Tukaj je potrebno zaradi jasnosti povdariti, da lahko posamezen dokument vsebuje več sporočil.

- **Preverjanje veljavnosti:** v tej fazi se uporabi XML preverjalnik, ki preveri, če XML sporočilo ustvarjeno v prejšnji stopnji cevovoda ustreza zahtevam določenim v shemi. Shemo mora razvijalec predhodno določiti.

- **Razreševalnik zaupanja vredne tretje osebe:** v tej stopnji cevovoda poskuša edina vgrajena komponenta ugotoviti kdo je pošiljatelj sporočila.

Slika 12: izhodni cevovod

Prav tako kot gredo vhodna sporočila skozi vhodni cevovod, gredo tudi izhodna sporočila skozi izhodni cevovod preden zapustijo sistem. Izhodni cevovod je prikazan na sliki 12 in vsebuje naslednje stopnje:

- **Priprava na sestavljanje:** v tej stopnji ni določenih nobenih standardnih komponent.

- **Sestavljanje:** Podobno kot ima faza razstavljanja v vhodnem cevovodu dve najpogosteje uporabljane standardne komponente, ima tudi faza sestavljanja v

izhodnem cevovodu sestavljalec tekstovnih datotek in sestavljalec XML datotek. Sestavljalec tekstovnih datotek sestavi iz XML sporočil tekstovne datoteke, medtem ko sestavljalec XML datotek doda ovojnico XML datoteki ali naredi ostale spremembe.

- **Kodiranje:** v tej stopnji se uporablja le ena standardna komponenta, in sicer MIME kodirnik. Ta komponenta zakodira izhodno sporočilo v MIME format.

Mehanizem vhodnega in izhodnega cevovoda se je izkazal za zelo uporabnega. Cevovod ti omogoča, da lahko narediš že takoj ob sprejemu nujne transformacije nad dokumenti, tako da so sporočila, ki iz teh dokumentov izhajajo, povsem prilagojena potrebam poslovne logike. Prav komponenta v vhodnem cevovodu, ki ima funkcijo razgrajevanja logičnih zapisov dokumenta v več sporočil ima funkcionalnost, ki je bila ena izmed temeljnih zahtev implementacije, ki sem jo izdelal v praktičnem delu naloge. Pomembne se mi zdijo tudi komponente, ki se ukvarjajo z varnostjo. Te komponente so temelj, da se na ESB lahko povežejo ne le aplikacije znotraj organizacije temveč tudi aplikacije drugih organizacij preko medmrežja.

5.1.3. Naročanje na sporočila

Ko sporočilo prispe skozi vhodni vmesnik in vhodni cevovod v sistem, se postavi vprašanje kaj narediti s sporočilom. Ponavadi sporočilo sproži poslovni proces ali pa je vhod v poslovni proces v kasnejši fazi. Včasih pa potuje sporočilo neposredno skozi izhodni cevovod v izhodni vmesnik in naprej iz sistema, ne da bi sprožil izvajanje poslovne logike. V vsakem primeru pa je potrebno določiti pot, ki jo bo moralo sporočilo opraviti. To je v BizTalku rešeno z mehanizmom, ki se imenuje naročanje na sporočila.

Ko sporočilo pride ven iz vhodnega cevovoda se mu dodeli kontekst, ki nosi različne lastnosti sporočila. Poslovni proces ali pa izhodni cevovod se lahko naročita na sporočilo na podlagi lastnosti določenih v kontekstu. Lastnosti na podlagi katerih se sporočilo usmerja v poslovnem procesu so lahko ime vhodnega vmesnika, čas ko je sporočilo prispelo v sistem ali pa celo vsebina sporočila.

5.2. Definiranje poslovnih procesov

Pošiljanje sporočil med aplikacijami je temeljna funkcionalnost strežnika BizTalk, a ne tudi najbolj pomembna. Glavni cilj je izdelava in izvajanje poslovnih procesov, ki se opirajo na sodelujoče aplikacije. Za definiranje poslovnih procesov strežnik BizTalk uporablja orkestracijo, za kreiranje in izvajanje poslovnih pravil pa uporablja mehanizem za izvajanje poslovnih pravil.

Logiko poslovnih procesov je možno implementirati neposredno s pisanjem programske kode, a to postane s povečevanjem kompleksnosti poslovnih procesov zelo dolgotrajno in zahtevno opravilo. Strežnik BizTalk ne uporablja takšnega pristopa, temveč namesto tega podpira izgradnjo poslovnih procesov grafično. Poleg tega, da je ta pristop hitrejši je tudi preglednejši in omogoča enostaven način spreminjanja poslovnih procesov.

Vsak razvijalec poslovnih procesov se pri uporabi strežnik BizTalk opira na tri temeljne funkcionalnosti, in sicer urejevalnik BizTalk za kreiranje XML shem, preslikovalnik BizTalk za definiranje preslikav med shemami in urejevalnik orkestracij za izgradnjo logike poslovnega procesa.

- **Kreiranje shem: urejevalnik BizTalk**
  BizTalk interno uporablja samo XML dokumente in vsak dokument mora ustrezati pravilom določene sheme. Zato mora obstajati način za definiranje shem. BizTalk ima v ta namen orodje urejevalnik BizTalk, ki omogoča hitro izgradnjo XSD shem.

- **Preslikovanje shem: preslikovalnik BizTalk**
  Orkerstracija, ki je implementacija poslovnega procesa, v večini primerov tekom svojega izvajanja sprejme in odda dokumente. V veliko primerih se zgodi, da je potrebno prejeti dokument prilagoditi svojim potrebam ali pa notranji delovni dokument potrebam prejemnika.

Preslikovalnik BizTalk je orodje, ki omogoča grafično izdelovanje transformacij med dvema XSD shemama. Vse transformacije med XSD shemami v BizTalku temeljijo na standardu XSLT.

Preslikave med elementi v XML shemah so lahko enostavne, to pomeni da se vrednost elementa iz izvorne sheme prenese v vrednost elementa ponorne sheme. Poznamo pa tudi kompleksne transformacije, ki jih naredimo s pomočjo posebnih elementov. BizTalk ima vgrajeno veliko število različnih takšnih elementov, ki spadajo v naslednje skupine:

- **Matematični:** ti elementi izvedejo matematične operacije nad elementi vhodnega dokumenta in rezultat priredijo elementu ciljnega dokumenta.

- **Pretvorni:** elementi iz te skupine pretvarjajo številčne vrednosti v tekstovne in obratno.

- **Logični:** logični elementi izvajajo logične operacije nad elementi vhodnega dokumenta. Uporabni so za odločanje o tem ali bomo določeno vrednost priredili elementu ali atributu ciljnega dokumenta.

- **Kumulativni:** kumulativni elementi računajo vsote, povprečne vrednosti, najmanjše in največje vrednosti v vhodnih dokumentih in rezultate priredijo elementom v ciljnem dokumentu.

- **Podatkovni:** elementi iz te skupine lahko dostopajo do podatkov v podatkovni bazi.

5.2.1. Definiranje poslovnih procesov: Urejevalnik orkestracij

Pri razvoju poslovnih procesov s strežnikom BizTalk lahko razvijalec pri določitvi poslovnih aktivnosti in povezav med njimi ter omejitev uporabi urejevalnik orkestracij, ki je grafično orodje. Urejevalnik orkestracij ponuja naslednje gradnike:

- **Sprejemni element –** poslovnemu procesu omogoča sprejemanje sporočil. Lahko vsebuje tudi filter, ki določa katera sporočila bodo sprejeta.

- **Oddajni element** – poslovnemu procesu omogoča pošiljanje sporočil.

- **Vmesniški element –** določa kako se pošiljajo sporočila. Vsak vmesniški element je lahko povezan bodisi s sprejemnim elementov, bodisi z oddajnim elementom.

- **Odločitveni element -** predstavlja če-potem-sicer programski konstrukt in omogoča poslovnemu procesu izvajanje na podlagi vrednosti izbranih pogojev.

- **Zančni element –** predstavlja programski konstrukt zanke in omogoča ponavljanje izvajanja določenih aktivnosti, dokler je zadoščeno določenemu pogoju.

- **Element konstrukcije sporočil –** omogoča kreiranje sporočil.

- **Transformacijski element –** pretvarja podatke iz sheme enega dokumenta v shemo drugega dokumenta. Pretvorba se izvede s sprožitvijo prej v preslikovalniku BizTalk definirane preslikave.

- **Element vzporednih aktivnosti –** vzporedno izvaja dve ali več zaporedij aktivnosti. Aktivnost, ki sledi elementu vzporednih aktivnosti se ne začne, dokler niso zaključeni vse veje izvajanja znotraj elementa vzporednih aktivnosti.

- **Element dosega –** omogoča združevanje operacij v transakcije in definiranje lovilcev izjem za omejevanje napak v izvajanju.

- **Element prirejanja vrednosti sporočil –** s pomočjo tega elementa lahko razvijalec priredi vrednosti spremenljivki v orkestraciji.

Spletne storitve omogočajo aplikacijam, da si izmenjujejo dokumente z uporabo SOAP in imajo velik vpliv na integracijo različnih aplikacij. Urejevalnik orkestracij omogoča implementacijo klica spletne storitve iz orkestracije. Podobno lahko posamezno aktivnost orkestracije izpostavimo kot spletno storitev. Hiter vzpon uporabe spletnih storitev ima tudi velik vpliv na to, kako so definirani poslovni procesi. Dokler poteka komunikacije med poslovnimi procesi, ki tečejo na enakih platformah se ne srečamo s problemi. Kaj pa se zgodi, če bi radi sodelovanje med poslovnimi procesi, ki tečejo na različni platformah? V takšnih primerih je zelo uporabno imeti način, kako opisati lastnosti poslovnih procesov, da v njih lahko sodelujejo tudi drugi.

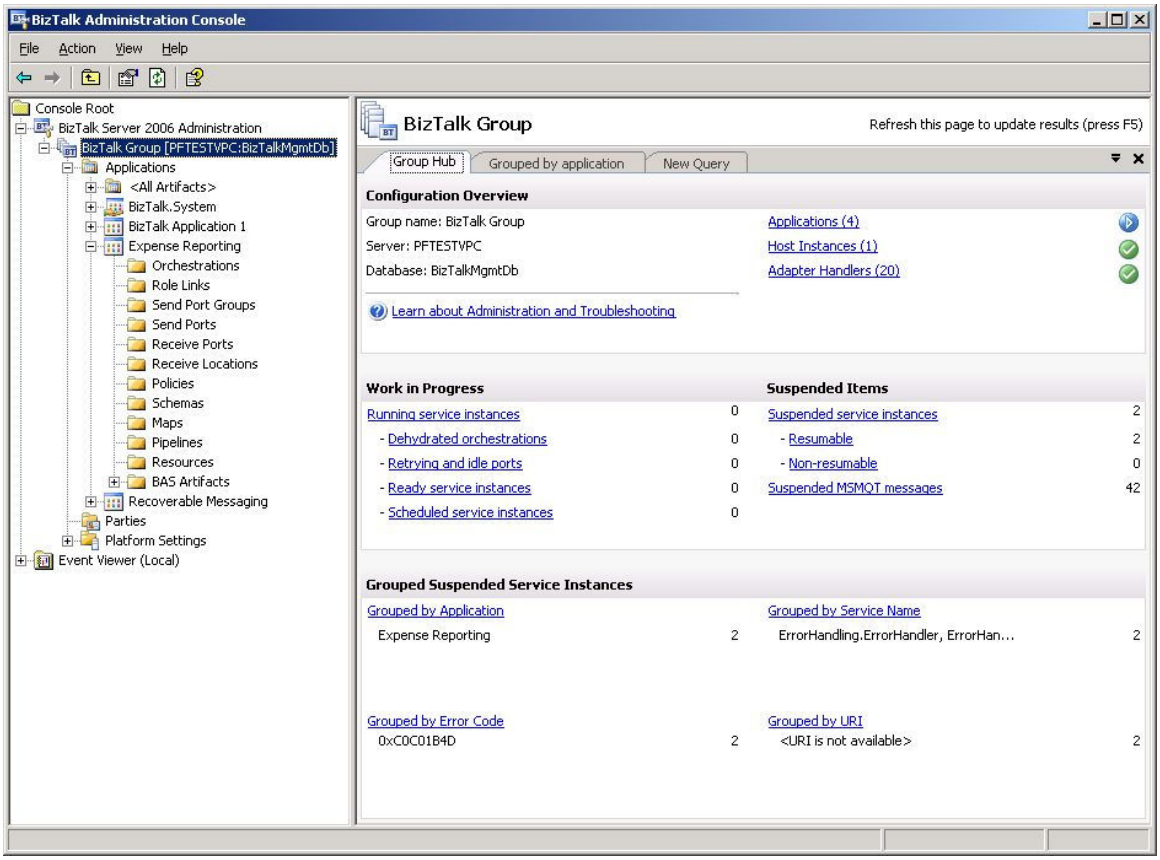
5.3. Upravljanje aplikacij

Glavno in najbolj pomembno orodje strežnika BizTalk 2006 je upravljalska konzola BizTalk(BAC), ki temelji na upravljalski konzoli Microsoft(MMC) in omogoča administracijo strežnika BizTalk. Izmed mnogih funkcionalnosti, ki jih BAC podpira so naslednje najpomembnejše:

- **Nameščanje aplikacij BizTalk:** Pri strežniku BizTalk 2006 je na področju namestitve narejen velik napredek v primerjavi s prejšnjo verzijo. V nasprotju s strežnikom BizTalk 2004, ki ni podpiral povezovanja različnih komponent rešitve med seboj, omogoča strežnik BizTalk 2006 celovito upravljanje z aplikacijami. S pomočjo BAC lahko administrator kreira aplikacije in jih namešča na enega ali več strežnikov.

- **Konfiguracija aplikacij BizTalk:** Medtem ko razvijalci razvijajo aplikacije, delujejo predvsem na logičnem nivoju. Če se razvijalec odloči, da bo njegova aplikacija komunicirala z drugo aplikacijo s pomočjo protokola HTTP, bo razvijalec s seznama preprosto izbral možnost HTTP, ne da bi ga skrbelo kateri URL bo uporabljen. Podobno razvijalec določi, da bo izhodni cevovod dodal digitalen podpis na sporočilo in se ne bo ukvarjal s tem, kateri ključ se bo zato uporabil. Vendar se morajo kljub vsem vse te podrobnosti nekje vendarle določiti in nastaviti. BAC je orodje, ki je namenjeno tem opravilom.

- **Nadzorovanje BizTalk aplikacij:** Z uporabo BAC lahko administrator spremlja operacije aplikacij. Kot prikazuje primer na sliki 13 lahko informacije o stanju aplikacij spremljamo na različne načine. Namesto, da bi morali administratorji iskati, kje se pojavlja problem, komponenta z različnimi barvami olajša opažanje napak.



Slika 13 : Upravljalska konzola BizTalk

BAC ponuja tudi druge funkcionalnosti. Administratorji lahko na primer dodajajo nove računalnike in jih določijo kot gostitelje aplikacij BizTalk, medtem ko se aplikacije nemoteno izvajajo. Da bi aplikacije prepoznale te spremembe pa jih sploh ni potrebno zaustavljati in ponovno zagnati. Funkcije BAC pa so dostopne tudi programsko in omogočajo administratorjem kreiranje skript za avtomatizacijo postopkov, ki jih BAC izvaja. To je zelo uporabno pri večjem številu obsežnih aplikacij.

Kot sem že povedal je BAC najbolj pomembna in osnovna komponenta strežnika BizTalk 2006. V bistvu je BAC orodje, ki upravlja ESB. S pomočjo BAC lahko uporabnik izvaja vse funkcionalnosti, ki pripadajo ESB. Tako lahko z BAC definiramo vmesnike in cevovode,

usmerjanje sporočil na podlagi različnih lastnosti, dodajamo povezave novih aplikacij na vodilo.

5.4. Mehanizem za izvajanje poslovnih pravil

Urejevalnik orkestracij skupaj z urejevalnikom BizTalk in preslikovalnikom BizTalk zagotavlja učinkovito podporo razvoju poslovnih procesov in poslovnih pravil, ki jih procesi uporabljajo. Vendarle pa je ponavadi bolj enostavno in učinkovito imeti enostavnejši način za definiranje in vzdrževanje poslovnih pravil. Da bi to omogočili, so strežniku BizTalk dodali orodje za definiranje poslovnih pravil(BRC) in mehanizem za izvajanje poslovnih pravil (BRE).

Primer kjer se pokaže prednost BRE je, če želimo napisati zelo kompleksno poslovno pravilo. V takšnem primeru se v izračunu uporabijo podatki iz različnih virov in v različnih formatih. Pisanje takšnega poslovnega pravila v obliki pogojnih stavkov v programski kodi ali s pomočjo odločitvenega elementa v urejevalniku orkestracij bi bilo mogoče, a zagotovo ne lahko delo. Vsekakor pa bi bilo vzdrževanje takšnih poslovnih pravil zelo zamudno in bi ob vsaki spremembi zahtevalo nepotrebno ponovno nalaganje celotnega poslovnega procesa v produkcijsko okolje. Poleg tega bi bilo potrebno ponovno zagnati strežnik BizTalk, da bi le ta zaznal spremembe. Ob tem je dejstvo, da se poslovna pravila v poslovanju zelo pogosto spreminjajo. Če pa je poslovno pravilo implementirano s pomočjo BRE, ga lahko spremenimo ne da bi bilo zato potrebno ponovno zagnati katerikoli modul strežnika BizTalk. Vse kar je potrebno storiti je spremeniti poslovno pravilo v BRE in ga dodati v množico že obstoječih pravil in spremembe bodo vidne takoj. Medtem ko so poslovni procesi ponavadi vzdrževani s strani razvijalcev, lahko poslovna pravila zaradi njihove preproste implementacije z uporabo BRC spreminjajo poslovni uporabniki.

Definiranje vsakega poslovnega pravila se začne z definiranjem besednjaka. Besednjak je množica uporabniku prijaznih imen za podatek ali pa operacijo nad temi podatki. Ko je besednjak enkrat definiran, lahko kreiramo poslovne politike, ki uporabljajo določen besednjak. Vsaka poslovna politika lahko vsebuje eno ali več poslovnih pravil. Poslovna pravila gradimo s pomočjo terminov definiranih v besednjaku in s pomočjo logičnih operatorjev. Kot podatkovni vir v poslovnih pravilih lahko uporabimo podatke v XML dokumentih ali pa podatke v podatkovnih bazah.

Za izvajanje poslovnih pravil je potrebno v poslovni proces dodati element za klicanje poslovnih pravil. Ta nato kreira instance BRE, določi katera poslovna politika se bo izvedla in poda vhodne podatke, kot je na primer XML dokument. Poslovna pravila pa se lahko kličejo tudi programsko z uporabo .NET objektnega modela.

Poslovna pravila so zelo pomemben del poslovne logike in se tudi pogosto spreminjajo. Zato je zelo pomembno, da obstaja mehanizem, ki podpira implementacijo poslovnih pravil in je ločen od jedra strežnika BizTalk. Pomembno je tudi da za spreminjanje poslovnih pravil ni potreben strokovnjak s tehničnim poznavanjem mehanizma za izvajanje poslovnih pravil, temveč lahko s pomočjo BRC poslovna pravila dodaja in spreminja poslovni uporabnik z vsebinskim znanjem. Funkcionalnost BRC sem uporabil tudi pri izdelavi primera. Po mojem mnenju sta BRC in BRE dve komponenti strežnika BizTalk, ki zelo veliko pripomoreta k njegovi uporabnosti in razširjenosti, saj sta enostavna za uporabo, a vseeno ponujata vso potrebno funkcionalnost.

5.5. Sistem za spremljanje izvajanja poslovnih procesov

V poslovnem svetu je veliko situacij, ko bi poslovni uporabniki radi vpogledali v stanje poslovnega procesa. Da bi zadostili vsem različnim potrebam potrebujemo splošno ogrodje za sledenje dogajanju v določenem poslovnem procesu. Točno to pa ponuja BAM. BAM je eno izmed orodij, ki razširjajo funkcionalnost strežnika BizTalk 2006.

BAM omogoča spremljanje dogodkov in podatkov, ki jih ustvarijo aplikacije BizTalk. Te informacije so dostopne na različne načine, in sicer:

- Preko Excelovih preglednic.
- Preko spletnega portala BAM.
- Preko sporočilnih storitev strežnika Microsoft SQL.

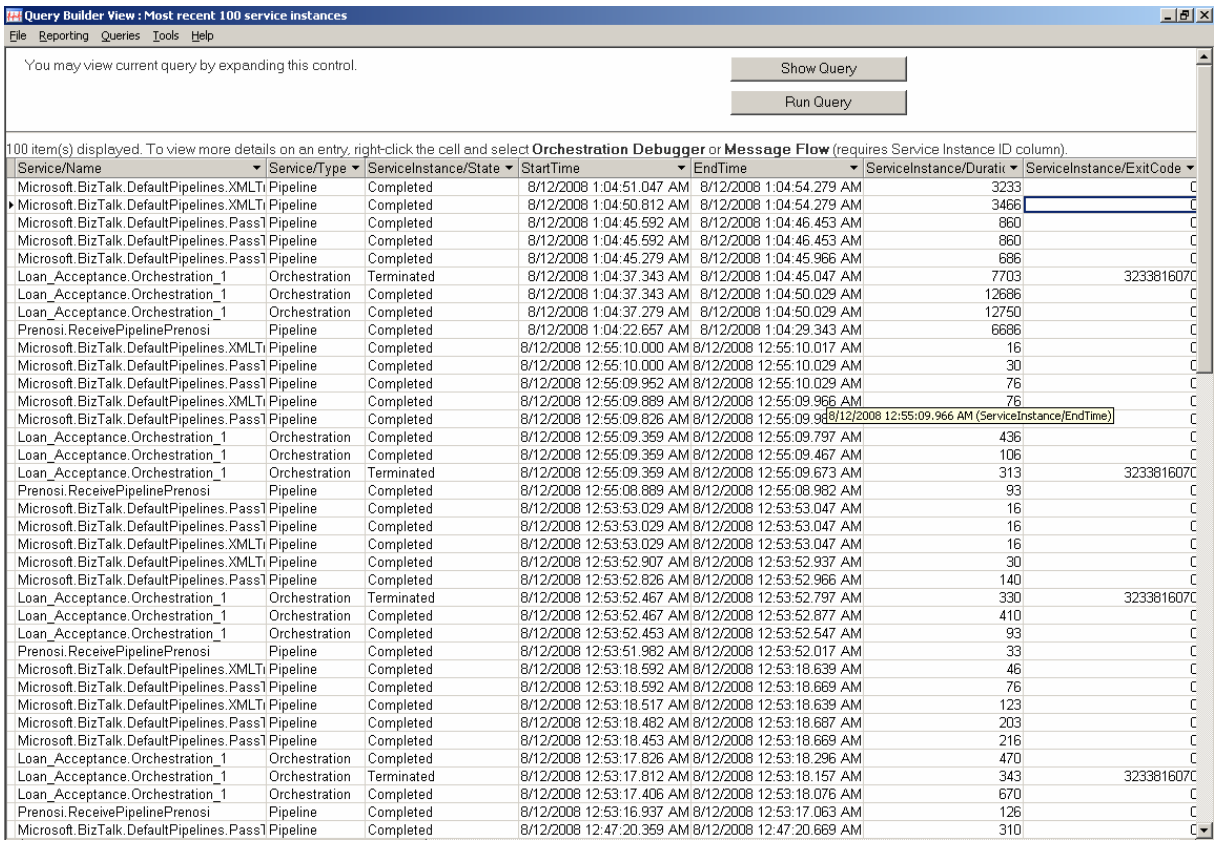
Ker BAM temelji na sprotnem analitičnem procesiranju podatkov(OLAP), so podatki ves čas ažurni in omogočajo poslovnim uporabnikom in vodilnim ljudem v organizacijah verodostojne podatke za odločanje. Prav vodilni delavci so tisti, ki jih agregirani podatki poslovnih procesov najbolj zanimajo, in to samo pomembni podatki brez podrobnosti. Po mojem mnenju BAM to zelo dobro podpira. Mislim pa da BAM ni nujno namenjen samo pridobivanju podatkov na podlagi katerih bodo sprejete odločitve, temveč je uporaben tudi za razvijalce. BAM omogoča razvoj različnih pogledov na podatke, kar je zelo uporaben pripomoček pri razvoju in iskanju napak v izvajanju poslovnih procesov s strežnikom BizTalk.

5.6. Orodje za razhroščevanje aplikacij BizTalk

Orodje za razhroščevanje aplikacij BizTalk (HAT) je orodje, ki omogoča razhroščevanje poslovnih procesov. Aplikacije razvite z BizTalkom izvajajo veliko različnih funkcionalnosti: sprejemajo in pošiljajo sporočila, obdelujejo sporočila znotraj poslovnega procesa, komunicirajo z različnimi sistemi z uporabo različnih protokolov… Zelo uporabno je, če ob tem ohranimo nadzor nad dogajanjem znotraj procesa še posebno, če se pojavi kakšna napaka.

HAT zagotavlja grafični vmesnik za dostop do informacij o aplikacijah, ki se izvajajo v strežniku BizTalk. Informacije nam povedo, kdaj se je posamezen poslovni proces začel in kdaj se je zaključil, povedo nam kdaj se je posamezna aktivnost znotraj procesa začela in kdaj končala, kdaj je bilo posamezno sporočilo poslano in kdaj sprejeto ter kakšna je bila vsebina sporočila. Razvijalec lahko celo določi točke znotraj procesa, na katerih se bo proces zaustavil in mu omogočil, da razišče kakšne vrednosti imajo parametri, ki ga zanimajo.

HAT omogoča, da lahko pridobimo seznam aktivnosti s poljubno poizvedbo in se s tem popolnoma prilagaja potrebam razvijalca. Na sliki 14 je prikazan takšen seznam aktivnosti in pripadajoče informacije.



Service/Name	Service/Type	ServiceInstance/State	StartTime	EndTime	ServiceInstance/Duratic	ServiceInstance/ExitCode
Microsoft.BizTalk.DefaultPipelines.XMLTi	Pipeline	Completed	8/12/2008 1:04:51.047 AM	8/12/2008 1:04:54.279 AM	3233	0
Microsoft.BizTalk.DefaultPipelines.XMLTi	Pipeline	Completed	8/12/2008 1:04:50.812 AM	8/12/2008 1:04:54.279 AM	3466	0
Microsoft.BizTalk.DefaultPipelines.Pass1	Pipeline	Completed	8/12/2008 1:04:45.592 AM	8/12/2008 1:04:46.453 AM	860	0
Microsoft.BizTalk.DefaultPipelines.Pass1	Pipeline	Completed	8/12/2008 1:04:45.592 AM	8/12/2008 1:04:46.453 AM	860	0
Microsoft.BizTalk.DefaultPipelines.Pass1	Pipeline	Completed	8/12/2008 1:04:45.279 AM	8/12/2008 1:04:45.966 AM	686	0
Loan_Acceptance.Orchestration_1	Orchestration	Terminated	8/12/2008 1:04:37.343 AM	8/12/2008 1:04:45.047 AM	7703	3233816070
Loan_Acceptance.Orchestration_1	Orchestration	Completed	8/12/2008 1:04:37.343 AM	8/12/2008 1:04:50.029 AM	12686	0
Loan_Acceptance.Orchestration_1	Orchestration	Completed	8/12/2008 1:04:37.279 AM	8/12/2008 1:04:50.029 AM	12750	0
Prenosi.ReceivePipelinePrenosi	Pipeline	Completed	8/12/2008 1:04:22.657 AM	8/12/2008 1:04:29.343 AM	6686	0
Microsoft.BizTalk.DefaultPipelines.XMLTi	Pipeline	Completed	8/12/2008 12:55:10.000 AM	8/12/2008 12:55:10.017 AM	16	0
Microsoft.BizTalk.DefaultPipelines.Pass1	Pipeline	Completed	8/12/2008 12:55:10.000 AM	8/12/2008 12:55:10.029 AM	30	0
Microsoft.BizTalk.DefaultPipelines.Pass1	Pipeline	Completed	8/12/2008 12:55:09.952 AM	8/12/2008 12:55:10.029 AM	76	0
Microsoft.BizTalk.DefaultPipelines.XMLTi	Pipeline	Completed	8/12/2008 12:55:09.889 AM	8/12/2008 12:55:09.966 AM	76	0
Microsoft.BizTalk.DefaultPipelines.Pass1	Pipeline	Completed	8/12/2008 12:55:09.826 AM	8/12/2008 12:55:09.96		0
Loan_Acceptance.Orchestration_1	Orchestration	Completed	8/12/2008 12:55:09.359 AM	8/12/2008 12:55:09.797 AM	436	0
Loan_Acceptance.Orchestration_1	Orchestration	Completed	8/12/2008 12:55:09.359 AM	8/12/2008 12:55:09.467 AM	106	0
Loan_Acceptance.Orchestration_1	Orchestration	Terminated	8/12/2008 12:55:09.359 AM	8/12/2008 12:55:09.673 AM	313	3233816070
Prenosi.ReceivePipelinePrenosi	Pipeline	Completed	8/12/2008 12:55:08.889 AM	8/12/2008 12:55:08.982 AM	93	0
Microsoft.BizTalk.DefaultPipelines.Pass1	Pipeline	Completed	8/12/2008 12:53:53.029 AM	8/12/2008 12:53:53.047 AM	16	0
Microsoft.BizTalk.DefaultPipelines.Pass1	Pipeline	Completed	8/12/2008 12:53:53.029 AM	8/12/2008 12:53:53.047 AM	16	0
Microsoft.BizTalk.DefaultPipelines.XMLTi	Pipeline	Completed	8/12/2008 12:53:53.029 AM	8/12/2008 12:53:53.047 AM	16	0
Microsoft.BizTalk.DefaultPipelines.XMLTi	Pipeline	Completed	8/12/2008 12:53:52.907 AM	8/12/2008 12:53:52.937 AM	30	0
Microsoft.BizTalk.DefaultPipelines.Pass1	Pipeline	Completed	8/12/2008 12:53:52.826 AM	8/12/2008 12:53:52.966 AM	140	0
Loan_Acceptance.Orchestration_1	Orchestration	Terminated	8/12/2008 12:53:52.467 AM	8/12/2008 12:53:52.797 AM	330	3233816070
Loan_Acceptance.Orchestration_1	Orchestration	Completed	8/12/2008 12:53:52.467 AM	8/12/2008 12:53:52.877 AM	410	0
Loan_Acceptance.Orchestration_1	Orchestration	Completed	8/12/2008 12:53:52.453 AM	8/12/2008 12:53:52.547 AM	93	0
Prenosi.ReceivePipelinePrenosi	Pipeline	Completed	8/12/2008 12:53:51.982 AM	8/12/2008 12:53:52.017 AM	33	0
Microsoft.BizTalk.DefaultPipelines.XMLTi	Pipeline	Completed	8/12/2008 12:53:18.592 AM	8/12/2008 12:53:18.639 AM	46	0
Microsoft.BizTalk.DefaultPipelines.Pass1	Pipeline	Completed	8/12/2008 12:53:18.592 AM	8/12/2008 12:53:18.669 AM	76	0
Microsoft.BizTalk.DefaultPipelines.XMLTi	Pipeline	Completed	8/12/2008 12:53:18.517 AM	8/12/2008 12:53:18.639 AM	123	0
Microsoft.BizTalk.DefaultPipelines.Pass1	Pipeline	Completed	8/12/2008 12:53:18.482 AM	8/12/2008 12:53:18.687 AM	203	0
Microsoft.BizTalk.DefaultPipelines.Pass1	Pipeline	Completed	8/12/2008 12:53:18.453 AM	8/12/2008 12:53:18.669 AM	216	0
Loan_Acceptance.Orchestration_1	Orchestration	Completed	8/12/2008 12:53:17.826 AM	8/12/2008 12:53:18.296 AM	470	0
Loan_Acceptance.Orchestration_1	Orchestration	Terminated	8/12/2008 12:53:17.812 AM	8/12/2008 12:53:18.157 AM	343	3233816070
Loan_Acceptance.Orchestration_1	Orchestration	Completed	8/12/2008 12:53:17.406 AM	8/12/2008 12:53:18.076 AM	670	0
Prenosi.ReceivePipelinePrenosi	Pipeline	Completed	8/12/2008 12:53:16.937 AM	8/12/2008 12:53:17.063 AM	126	0
Microsoft.BizTalk.DefaultPipelines.Pass1	Pipeline	Completed	8/12/2008 12:47:20.359 AM	8/12/2008 12:47:20.669 AM	310	0

Slika 14: seznam aktivnosti in podatki o posameznih aktivnostih v orodju HAT

Moje izkušnje pri razvoju poslovnih procesov na platformi strežnik BizTalk so, da se poslovnih procesov ne da razvijati brez orodja, kakršen je HAT. HAT omogoča popolno natančno diagnosticiranje napak, izvajanje poslovnega procesa korak za korakom in ob tem spremljanje stanja posameznih elementov poslovnega procesa in podatkov v sporočilih, ki sodelujejo v poslovnem procesu. HAT je prilagodljivo orodje, saj razvijalcu omogoča prikaz aktivnosti in njenih podatkov na podlagi poizvedbe, ki jo razvijalec napiše sam. Tako si lahko HAT prilagodi povsem po svojih potrebah.

6. Problemska domena

V drugem, praktičnem delu diplomske naloge se bom lotil reševanja konkretnega problema s pomočjo teorije in programskega orodja strežnik BizTalk 2006, ki sem jih opisal v poglavjih 2,3 in 4. Lotil sem se analize obstoječega stanja, izdelave načrta možne rešitve in tudi implementacije.

6.1. Analiza obstoječega stanja

Problemska domena moje diplomske naloge je Register davčnih zavezancev (RDZ), ki ga vodi Davčna uprava Republike Slovenije(DURS). RDZ je največji register v Sloveniji. V RDZ se vodijo podatki, ki jih potrebuje DURS za svoje delo za vse davčne zavezance v Sloveniji, tako za fizične kot za pravne osebe.

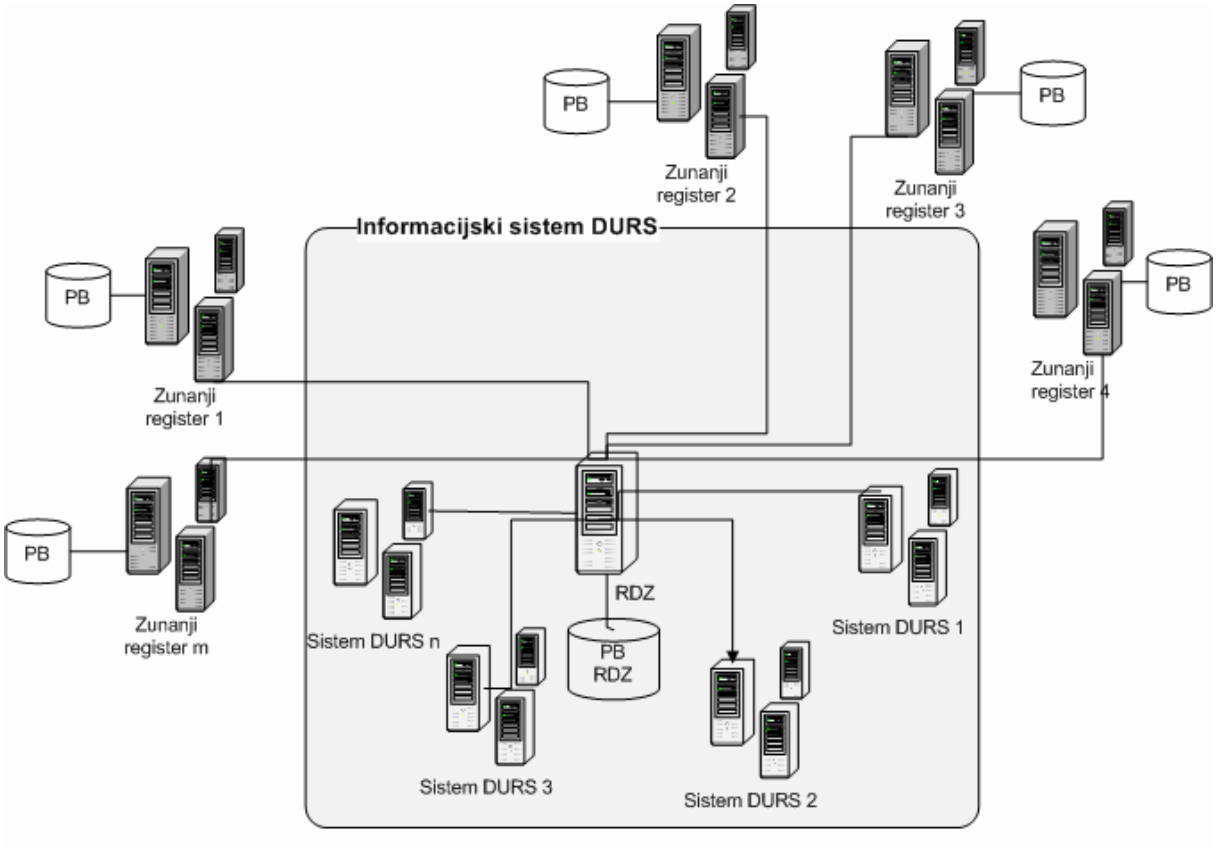
V RDZ se vodijo osnovni podatki o zavezancih, podatki o njihovih naslovih, zaposlitvah, naložbah, zaposlenih … Torej podatki, ki so zelo pomembni za delo na davčnih izpostavah in uradih. S pomočjo podatkov iz RDZ se izračunavajo davki, dohodnina, davčne olajšave in ostali izračuni. Zelo pomembno pa je, da so podatki v RDZ čim bolj ažurni. Da bi zadostili temu pogoju se RDZ povezuje z drugimi registri v Sloveniji. Tako RDZ pridobiva podatke iz Centralnega registra prebivalstva, Poslovnega registra Slovenije, Registra transakcijskih računov in ostalih registrov.

Podatki se iz RDZ tudi izvažajo v zaledne sisteme DURS in na posamezne davčne izpostave in urade ter v zunanje registre.

V sistemu izmenjav RDZ se izvaja tudi zelo podrobno beleženje. Za vsak korak, ki se izvede med procesom izmenjave je potrebo zabeležiti, če je bil uspešno izveden. Ob morebitni napaki je potrebno zabeležiti čas napake in vzrok napake. Beleženja vzroka napake pomeni, da se zabeležijo podatki, ki so se prenašali v trenutku pojavitve napake. Beleženje napak je pomembno z vidika zanesljivosti izmenjave. Ob dobrem beleženju se ne more zgoditi, da se kakšen podatek ali pa skupina podatkov ne bi prenesla in o tem ne bi obstajal noben zapis.

O uspešnosti izmenjav je potrebno obveščati odgovorne ljudi, da lahko ustrezno ukrepajo ob morebitnih napakah pri prenosu. Obveščanje poteka tako, da se generirajo ustrezna poročila, ta pa se posredujejo odgovornim osebam. Kako in kdaj se pošiljajo poročila je odvisno od posamezne izmenjave, ampak v splošnem se poročila o napakah pošljejo takoj, ko se napaka pojavi. Poročilo o uspešnih prenosih pa se pošlje odgovornim osebam enkrat dnevno.

Trenutno opravlja funkcijo upravljanja izmenjav aplikacija, katero je po naročilu DURS razvilo podjetje IXTLAN-TEAM. Aplikacija upravlja izmenjave in nima vgrajene poslovne logike. Poslovna logika je implementirana na podatkovni bazi z shranjenimi procedurami. Aplikacija ima urnik in sproža izmenjave glede na urnik, skrbi da se izvaja beleženje izmenjav, aplikacija zagotavlja ponovljivost izmenjav in generira poročila, s katerimi obvešča uporabnike, ki so odgovorni za izmenjave.



Slika 15: prikaz modela izmenjav

Kot je razvidno iz slike 15 je RDZ povezan z različnimi sistemi. Izmenjave med njimi potekajo v različnih časovnih intervalih in uporabljajo različne tehnologije in protokole. Izmenjave z nekaterimi registri so enosmerno, kar pomeni da gredo podatki samo od zunanjih registrov v RDZ. Nekatere izmenjave pa so dvosmerne.

Sistemi izmenjujejo podatke, ki ustrezajo vnaprej dogovorjenim shemam. Aplikacija lahko tako preverja pravilnost podatkov in s tem prepreči, da bi se v RDZ vpisali neveljavni podatki.

6.2. Načrt rešitve

Glavni cilj diplomske naloge je izdelava rešitve, ki nakazuje možnost nadomestitve sedanje aplikacije, katero uporabljajo na DURS za izmenjave podatkov med RDZ in ostalimi sistemi. Rešitev s pomočjo strežnika BizTalk ima prednost ker lahko neposredno v procesu izmenjave pripravi podatke in jih posreduje ostalim sistemom na DURS. BizTalk je enostavno razširljiv zaradi principa vmesnikov katerega podpira, saj omogoča enostavno priključitev na podatkovno bazo kateregakoli proizvajalca in tudi na ostale razširjene informacijske sisteme.

Proces izmenjave, ki sem jo implementiral izmenjuje podatke z zunanjimi sistemi. Ti so naslednji:

- Vhodni
  - Aplikacija dobi vhodne podatke o zavezancih iz zunanjega registra.

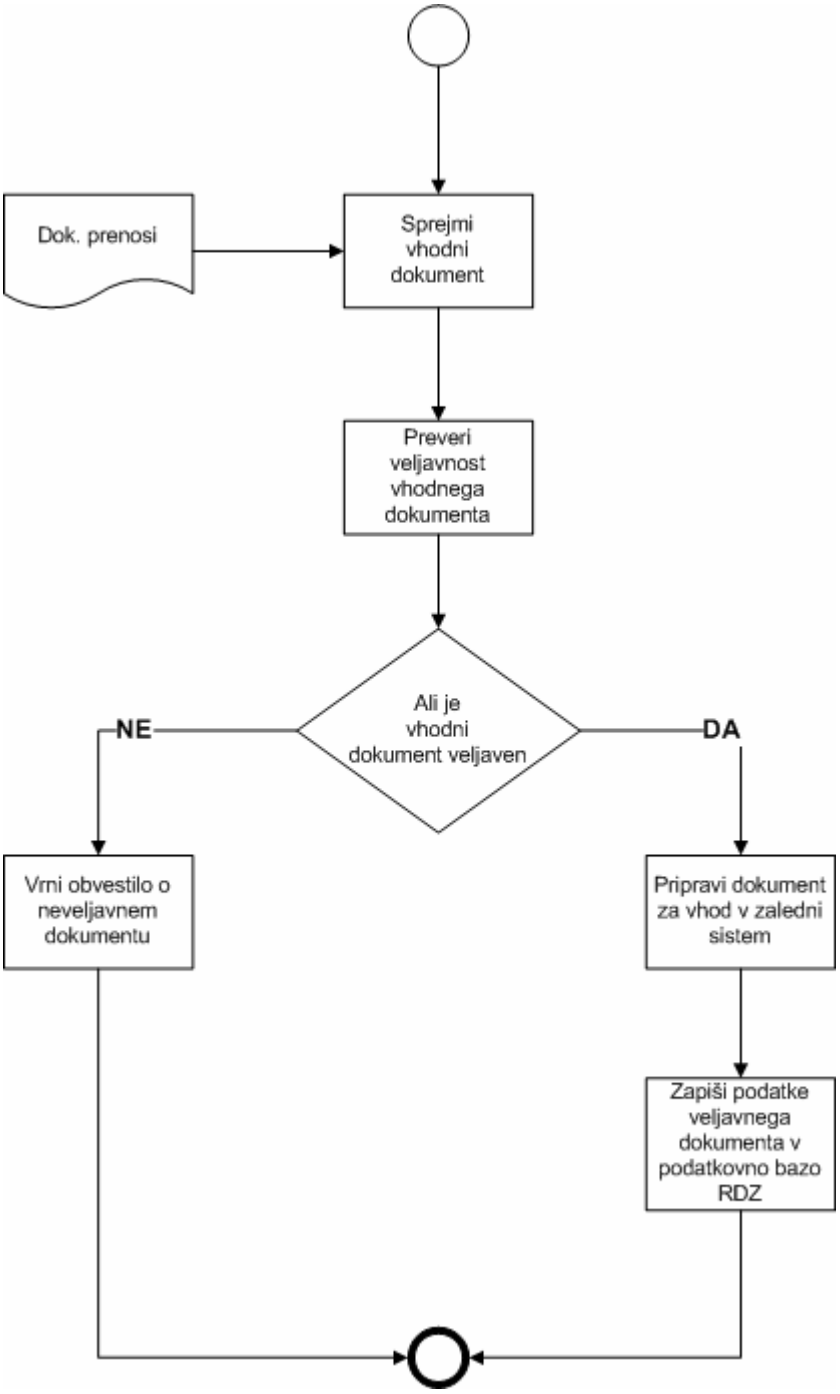
- Izhodni
  - Aplikacija odloži veljavne podatke o posameznem zavezancu v obliki XML datoteke v mapo. Dokument je namenjen vhodu v zaledni sistem.
  - Aplikacija pošlje veljavne podatke o posameznem zavezancu podatkovni bazi, ki jih zapiše v podatkovno bazo RDZ.

Izmenjava, katero sem implementiral v tej diplomski nalogi, poteka po naslednjem scenariju, in sicer se v vnaprej določeno mapo na datotečnem sistemu odloži XML datoteka, ki ustreza predpisani XSD shemi in vsebuje podatke o zavezancih iz zunanjega registra. Rešitev je implementirana tako, da sistem ves čas preverja stanje v tej mapi in takoj, ko ugotovi, da se v mapi nahaja datoteka jo aplikacija prevzame in prične se obdelava. V prejeti datoteki so lahko podatki o enemu ali več zavezancih. Med sprejemanjem XML datoteke aplikacija poskrbi, da se vhodna XML datoteka razgradi v več novih XML datotek, tako da vsaka vsebuje zapis le o enem zavezancu. Te datoteke se sedaj v sistemu obravnavajo kot sporočila oziroma zahteve. V sistemu je toliko novih sporočil, kolikor je bilo logičnih zapisov v vhodni datoteki. V naslednjem koraku se preveri vsebina posameznega sporočila. Aplikacija na podlagi poslovnih pravil, katera so implementirana s pomočjo BRE, preveri če so podatki, ki so pomembni za obdelavo, prisotni v sporočilu.

Če so podatki prisotni, lahko sporočilo nadaljuje, sicer pa se označi kot neveljavno sporočilo. V odločitveni veji v procesu se nato ločijo veljavna sporočila od neveljavnih. Neveljavna sporočila se odložijo na določeno mesto v datotečnem sistemu v obliki XML datoteke, ki vsebuje vse podatke, tako da lahko uporabnik, ki nadzoruje prenose pregleda sporočilo in ugotovi zakaj je bilo označeno kot neveljavno. Obdelava veljavnih sporočil nemoteno poteka dalje, in sicer se veljavna sporočila odložijo v vnaprej določeno mapo na datotečnem sistemu. Ta sporočila imajo definirano obliko, ki ustreza vhodu v zaledni sistem DURS. Kasneje zaledni sitem iz te mape prebere datoteke in jih uporabi za svoje potrebe. S tem se pospeši izvajanje prenosov, saj ni več potrebno prenašati podatke, ki so bili sprejeti v uvozu preko RDZ v zaledne sisteme, temveč se lahko podatki za zaledne sisteme pripravijo že med izvajanjem samega prenosa in se zalednim sistemom dostavijo neposredno. Če se pojavi potreba po drugačni shemi ali celo po drugačnem formatu sporočila, se lahko to v sami aplikaciji s pomočjo preslikav in izhodnih vmesnikov enostavno in hitro spremeni.

Vsa veljavna sporočila se naprej pošljejo v komponento, katera kreira nova sporočila z uporabo podatkov v veljavnih sporočilih. Komponenta uporabi podatke iz prvotnih sporočil, jih preoblikuje in preslika v polja novega sporočila. S tem postopkom se ustvarijo nova sporočila, ki so namenjena izključno zapisu v podatkovno bazo RDZ. Vsako novo sporočilo je uporabljeno kot vhodni parameter pri klicu shranjene procedure na podatkovni bazi RDZ. Klic procedure zapiše nove podatke v RDZ.

Predstavljena rešitev obsega tudi uporabo dodatnih modulov strežnika BizTalk kot sta BRE, ki je uporabljena za implementacijo poslovnih pravil. Druga uporabljena komponenta v sistemu je BAM, ki na temelju OLAP omogoča spremljanje izvajanja v aplikaciji. BAM nadomešča kreiranje poročil o uspešnosti izvedbe prenosov v sedanji aplikaciji nameščeni na DURS. BAM omogoča ažurno spremljanje informacij o tem koliko prenosov je bilo uspešno in koliko neuspešno izvedenih.



Slika 16: načrt procesa prenosa

6.3. Implementacija

6.3.1. Sistemske zahteve

Potrebna programska oprema, ki mora biti nameščena za razvoj rešitve je naslednja:

- Strežnik Microsoft Windows 2003
- Microsoft Visual Studio 2005
- Strežnik Microsoft SQL 2005
- Microsoft Office Excel
- Strežnik Microsoft BizTalk 2006

6.3.2. Sheme dokumentov

V aplikaciji so bile uporabljene sheme XML dokumentov. Sheme so bile uporabljene za definiranje oblike in tipov podatkov uporabljenih v sporočilih v aplikaciji. Sheme se uporabljajo tudi za preverjanje pravilnosti sporočil in za razgraditev sporočil v vhodnem cevovodu. Uporabljene so bile naslednje sheme:

- *Prenos.xsd*
- *PrenosEnv.xsd*
- *SQLService.xsd*

Sheme so dodane v prilogi.

Uporaba shem XSD je pri razvoju rešitev zelo uporabna, saj lahko posamezno shemo uporabiš v različnih orodjih za različne namene. Tako povečaš njeno uporabnost in imaš hkrati dober nadzor, saj je posamezna shema definirana le na enem mestu in je tako njeno vzdrževanje enostavno.

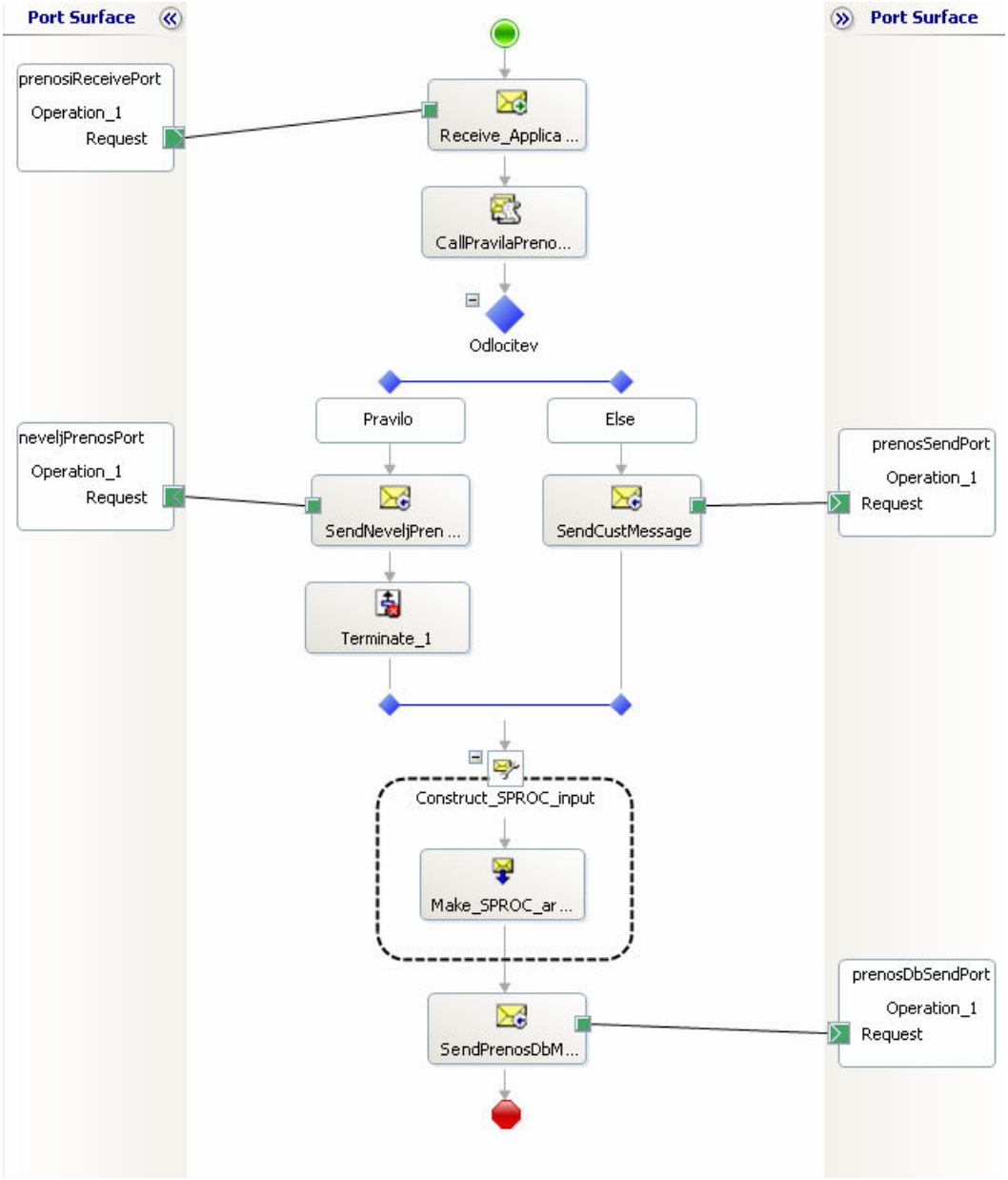
6.3.3. Proces prenosa

Proces prenosa je prikazan na sliki 17. Proces je zaporedje aktivnosti, ki so med seboj logično povezane. Vsaka izmed aktivnosti izvaja določen del funkcionalnosti. Aktivnosti v levem in desnem sivem območju so logični vhodni in izhodni vmesniki in skrbijo za komunikacijo z zunanjimi sistemi, medtem ko aktivnosti v osrednjem področju izvajajo poslovno logiko.

V procesu nastopajo tri sporočila in sicer:

- *PrenosiMessage*, ki ustreza shemi *PrenosEnv.xsd.*
- *PrenosMessage*, ki ustreza shemi *Prenos.xsd.*
- *PrenosReqMessage*, ki ustreza shemi *SQLService.xsd.*

Za razvoj poslovnega procesa na strežniku se uporablja razvojno okolje Microsoft Visual Studio z vgrajeno podporo za razvoj poslovnih procesov. Razvoj je zaradi prijaznega grafičnega vmesnika enostaven. V orodni vrstici se nahajajo vse potrebne komponente za razvoj poslovnih procesov, katere je potrebno še povezati med seboj. Razvojno okolje ima vgrajeno tudi logiko, ki ne dovoli povezati dva gradnika poslovnega procesa med seboj, dokler nimata oba določenega istega tipa dokumenta nad katerim bosta izvajala operacije. Tako razvojno okolje že samo preverja pravilnost poslovnega procesa.

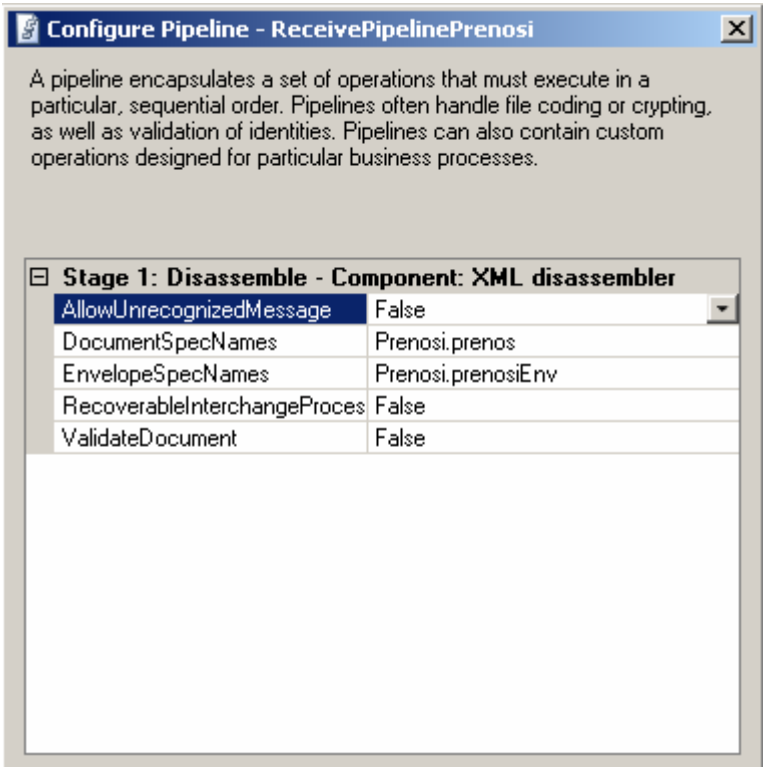


Slika 17: prikaz poslovnega procesa prenosa podatkov v RDZ iz zunanjega registra

6.3.4. Sprejemni cevovod

V vhodnem vmesniku *PrenosiReceivePort* je implementiran cevovod *ReceivePipelinePrenosi*. V vhodnem cevovodu se izvede razgraditev vhodne datoteke, katera

vsebuje enega ali več zapisov v sporočila s posameznim zapisom. Rezultat procesiranja v cevovodu je množica sporočil, ki vsebuje podatke o posameznem zavezancu. Ta sporočila ustrezajo shemi *Prenos.xsd*. Cevovod potrebuje za svoje delovanje shemi *PrenosEnv.xsd* in *Prenos.xsd.* Pri tem je sporočila tipa *PrenosEnv.xsd* ovojnica za sporočila tipa *Prenos.xsd*. Nastavitve vhodnega cevovoda so prikazane na sliki 18.

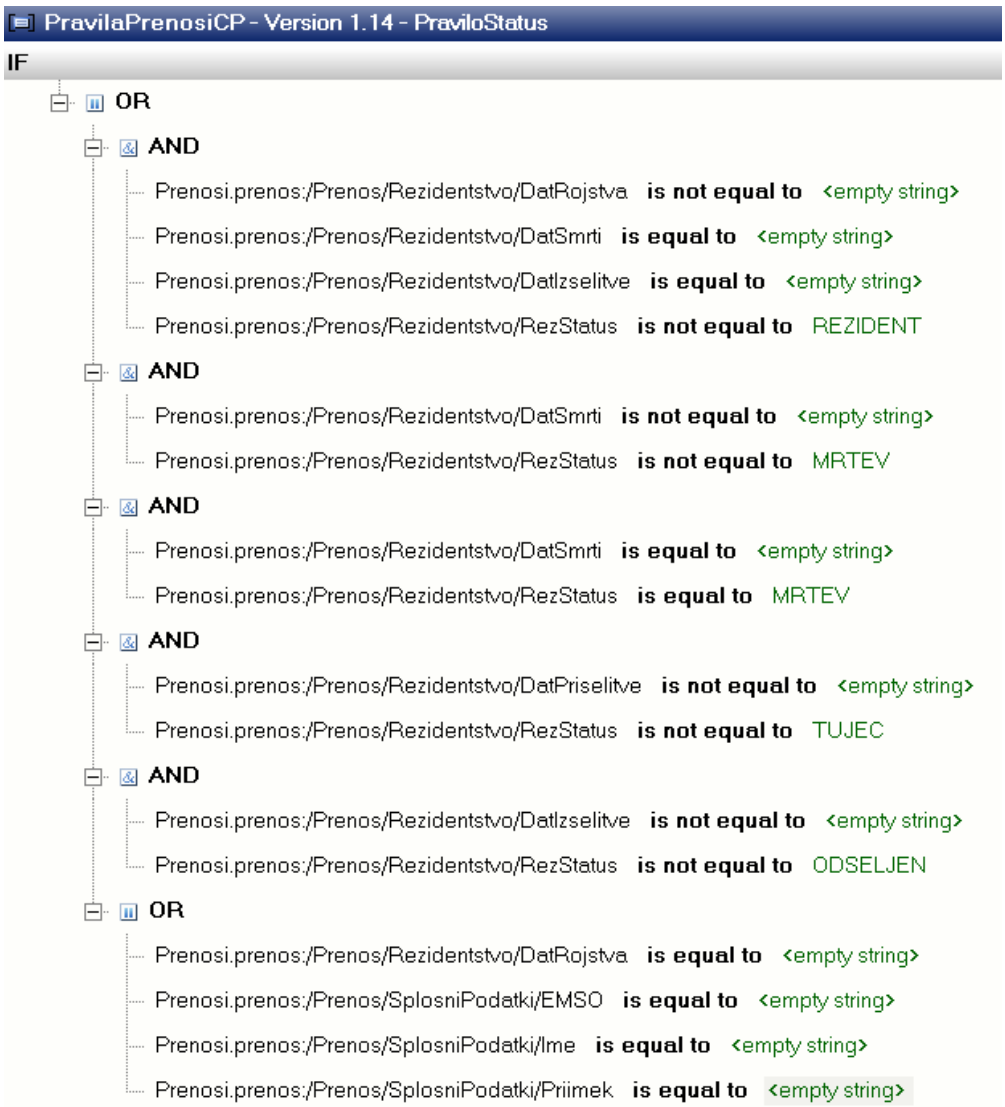


Slika 18: nastavitve sprejemnega cevovoda

6.3.5. Poslovna pravila

Funkcija komponente *CallPravilaPrenosCP* v procesu prenosa je, da se poveže z BRE in za podane vhodne podatke v BRE pridobi od BRE rezultat ovrednotenja poslovnih pravil za določeno sporočilo. Vrnjena vrednost se uporabi v kasnejši aktivnosti procesa. Poslovna pravila v procesu so namenjena predvsem preverjanju, če so v sporočilu prisotna obvezna polja in če ima polje *RezStatus* pravilno vrednost glede na polja *DatRojstva*, *DatSmrti*, *DatPriselitve* in *DatIzselitve*. Logika poslovnih pravil je implementirana z orodjem BRC in je vidna iz slike 19.

Če se katerikoli izmed pogojev iz množice poslovnih pravil na sliki 19 izračuna kot logično pravilen, se nastavi polje *Status* v sporočilu *Prenos* na vrednost NEVELJAVEN, sicer pa ostane njegova vrednost nespremenjena.



Slika 19: Poslovna pravila implementirana z orodjem Business Rules Composer

6.3.6. Vejitev v procesu prenosa

Na podlagi rezultata ovrednotenja poslovnih pravil se v sporočilo v polje *Status* zapiše vrednost NEVELJAVNO oziroma ostane v tem polju prvotna vrednost. Tok sporočila v procesu je odvisen od vrednosti polja *Status*, in sicer če je status različen od NEVELJAVEN se proces nadaljuje po normalni poti, sicer pa se sporočilo odloži v mapo v kateri se nahajajo neveljavna sporočila in se instanca tega procesa zaključi.

Odločitev se izvede v elementu poslovnega procesa, ki je namenjen odločanju in se v aplikaciji imenuje *Odlocitev*.

Strežnik BizTalk sicer omogoča, da se lahko do vsakega polja v sporočilu programsko dostopa s pomočjo xPath. Ponuja pa BizTalk tudi možnost, da lahko posamezno polje v sporočilu izpostavimo in postane tako dostopno po imenu in lahko na vsakem mestu v aplikaciji preberemo oziroma mu nastavimo vrednost. Takšen pristop je uporabljen na mestu v procesu, kjer pride do odločitve. Implementacija izpostavitve polja je prikazana v prilogi.

6.3.7. Izločitev neveljavnih sporočil

Sporočila, ki se jim je pri ovrednotenju poslovnih pravil polje *Status* nastavilo na vrednost NEVELJAVNO nadaljujejo pot v veji, ki odloži neveljavna sporočila v mapo na datotečnem sistemu z namenom, da uporabnik pregleda te dokumente in lahko ugotovi zakaj je prišlo do izločitve. Za odložitev sporočila na datotečni sistem se uporabi izhodni vmesnik *NeveljPrenosPort.*

Odložitev neveljavnih sporočil v posebno mapo na datotečnem sistemu je nujno zaradi diagnosticiranja vzroka, zakaj je prišlo do neveljavnosti sporočila. Če so vsa neveljavna sporočila na enem mestu in vsebujejo vse podatke je najlažje ugotoviti vzrok napake.

6.3.8. Odložitev veljavnih sporočil

Vsa sporočila, ki po ovrednotenju vrednosti njihovih polj ne postanejo neveljavna, se odložijo v mapo na datotečnem sistemu, kjer na datoteke čaka zaledni sistem. Za odložitev sporočila na datotečni sistem se uporabi izhodni vmesnik *SinglePrenosSendPort.*

Za implementacijo konkretnega primera je poslovna zahteva določala, da naj se sporočila, ki predstavljajo vhod v zaledni sistem odlagajo v posebno mapo. Če bi se zahteve spremenile ali pa bi radi integracijo z aplikacijo, ki podpira različen protokol izmenjave sporočil, je za prilagoditev potrebno le spremeniti tip vmesnika. Poslovna logika, ki pripravi sporočilo primerno za vhod v sistem na podlagi poljubne XSD sheme, v poslovnem procesu namreč že obstaja.

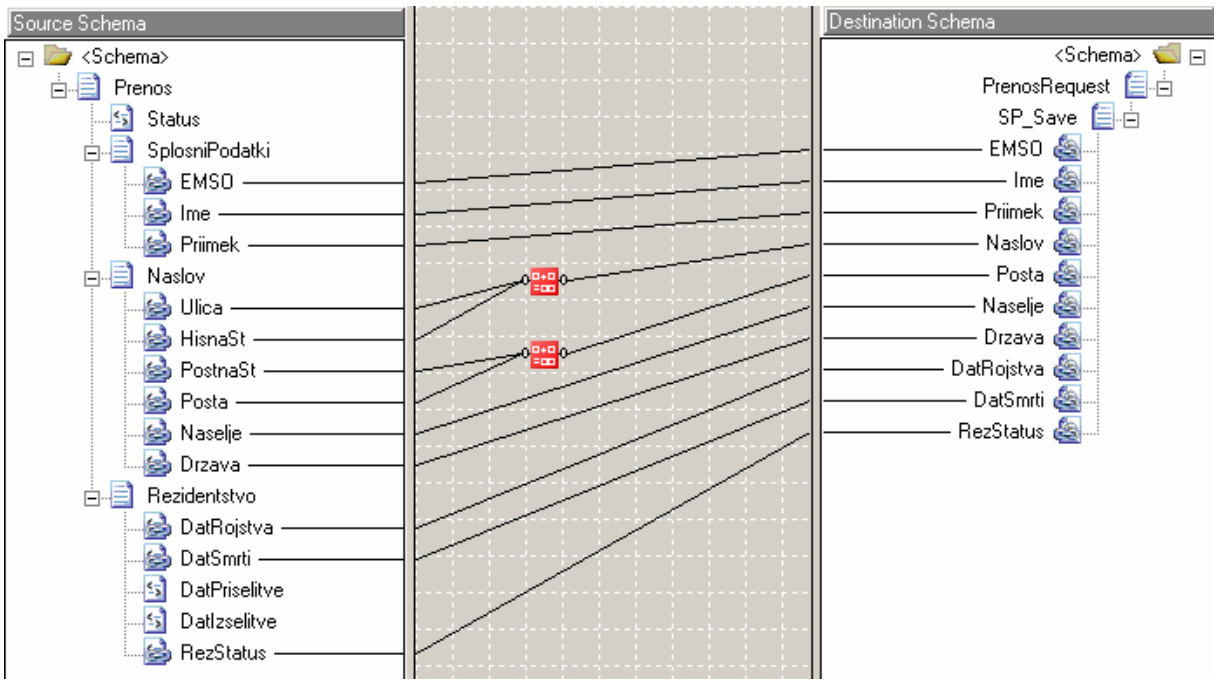
6.3.9. Kreiranje novega sporočila

Ker je za vpis podatkov zavezanca v podatkovno bazo RDZ potrebna drugačna oblika sporočila od prvotne oblike, je bilo v aplikaciji potrebno kreirati novo sporočilo, ki ustreza shemi *SQLService.xsd* in je namenjeno klicu shranjene procedure na podatkovni bazi, ki podatke iz sporočila zapiše v ustrezno tabelo. Za napolnitev podatkovnih polj novega sporočila se uporabi preslikava, ki preslika podatke iz vhodnega dokumenta, to je v tem primeru vhodno sporočilo v sistem v novo kreiran dokument. Preslikava je opisana v točki 6.3.10.

6.3.10. Preslikava sporočil

V postopku kreiranja novega sporočila se uporabi komponenta, ki preslika podatke originalnega sporočila v podatke sporočila namenjenega zapisu v podatkovno bazo. Kako in katera polja se preslikajo je prikazano na sliki 20.

Vhod v preslikavo	**Izhod iz preslikave**
Prenos.xsd	SQLService.xsd



Slika 20: preslikava polj med shemama Prenos.xsd in SQLService.xsd.

6.3.11. Zapis podatkov v podatkovno bazo

Glavni namen prenosa podatkov je zapis podatkov iz zunanjih sistemov in registrov v podatkovno bazo RDZ. Za zapis podatkov v podatkovno bazo poskrbi logični vmesnik *prenosDbSendPort* in pripadajoč fizični vmesnik *PrenosDbSendPort*. Za proženje in predajo vhodnih podatkov shranjeni proceduri je namenjeno sporočilo *PrenosReqMessage*.

Razvojno okolje namenjeno razvoju aplikacij BizTalk ima zelo dobro podprto integracijo s podatkovnimi bazami. Okolje pripravi vse potrebne gradnike za izvedbo branja ali pisanja v podatkovno bazo na podlagi podatkov o shranjenih procedurah ali tabelah. Naloga razvijalca je le, da zagotovi sporočilo na podlagi ustrezne sheme kot vhod v element, ki zapisuje ali pa bere podatke v podatkovno bazo. Ko prispe veljavno sporočilo v element, le ta prebere podatke iz sporočila in jih sam zapiše v podatkovno bazo, ali pa podatke iz podatkovne baze prebere in jih zapiše v sporočilo.

6.3.12. Obravnavanje vhodnih datotek, ki ne ustrezajo shemi

Ker lahko v sistem prenosa nenamerno pride tudi kakšna neprimerna datoteka, lahko pride do nepredvidene napake. Takšni primeri so, ko katera izmed komponent v procesu pričakuje sporočilo, ki ustreza točno določeni shemi, a prispe drugačno sporočilo. Za takšne primere sem razvil pomožno aplikacijo, ki te datoteke prepozna in jih shrani v mapo na datotečnem sistemu. Tako lahko uporabnik spremlja, če se pojavljajo v sistemu napačne datoteke in ustrezno ukrepa.

6.3.13. Povezava med logičnimi in fizičnimi komponentami

Vsak logični vmesnik v procesu izmenjav s slike 17 je povezan z fizičnim vmesnikom. Pri razvoju aplikacije v strežniku BizTalk se v urejevalniku orkestracij kreirajo le logični vmesniki. V tej točki se ne definirajo nobene podrobne nastavitve, temveč se le te določijo fizičnim vmesnikom v BAC, kjer se tudi izvede povezava med logičnimi in fizičnimi vmesniki. Povezave v aplikaciji so:

Logični vmesnik	**Fizični vmesnik**	**Vhodni/Izhodni**	**Vrsta prenosa**
prenosiReceivePort	PrenosiReceivePort	Vhodni	Datoteka
neveljPrenosPort	NeveljPrenosPort	Izhodni	Datoteka
prenosSendPort	SinglePrenosSendPort	Izhodni	Datoteka
prenosDbSendPort	PrenosDbSendPort	Izhodni	SQL

Nastavitve fizičnih vmesnikov:

- PrenosiReceivePort(vhodni vmesnik)

Nastavitev	**Vrednost**
Transport type	FILE
URI	<Pot do mape, kjer se odlagajo vhodne datoteke>
Receive handler	BizTalkServerApplication
Receive pipeline	ReceivePipelinePrenosi
File mask	*.xml

- NeveljPrenosPort(izhodni vmesnik)

Nastavitev	**Vrednost**
Transport type	FILE
URI	<Pot do mape, kjer se odlagajo neveljavne datoteke>
Send handler	BizTalkServerApplication
Send pipeline	PassThruTransmit
File name	err%MessageID%.xml

- SinglePrenosSendPort

Nastavitev	**Vrednost**
Transport type	FILE
URI	<Pot do mape, kjer se odlagajo datoteke namenjene vhodu v zaledni sistem>
Send handler	BizTalkServerApplication
Send pipeline	PassThruTransmit
File name	%MessageID%.xml

- PrenosDbSendPort

Nastavitev	**Vrednost**
Transport type	SQL
URI	SQL://<ime strežnika>/<ime podatkovne baze>
Send handler	BizTalkServerApplication

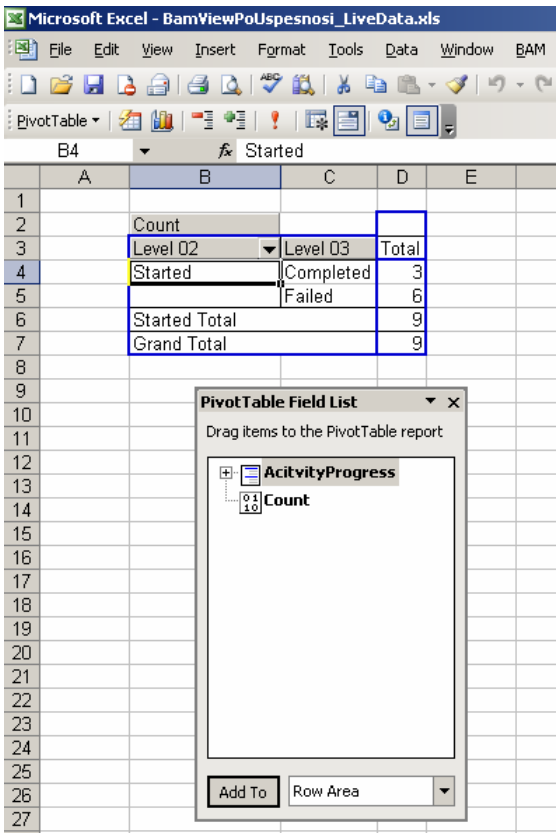
Send pipeline	XMLTransmit
Connection string	Provider=SQLOLEDB.1;IntegratedSecurityProvider=SQLOLE DB.1;Integrated Security=SSPI;Persist Security Info=False;Initial Catalog=<ime podatkovne baze>;Data Source=<ime strežnika>
Document target namespace	http://Prenosi

6.3.14. Spremljanje izvajanja prenosov za poslovne uporabnike

Za spremljanje izvajanja prenosov, je aplikaciji dodan modul za spremljanje števila uspešnih in neuspešnih prenosov. Modul je zelo enostaven za uporabo, saj je implementiran kot vrtilna tabela v Microsof Excelu. Tako ima vsak uporabnik, ki ga zanima stanje prenosov, dostop do ažurnih podatkov o številu uspešnih ali neuspešnih prenosov. Izgled modula je na sliki 21.

Podatki o izvajanju poslovnih procesov so zelo pomembni za sprejemanje poslovnih odločitev. Z uporabo modula BAM lahko razvijemo takšen pogled na podatke, kakršnega določen uporabnik potrebuje. Sam sem uporabil spremljanje podatkov z modulom BAM tudi pri samem razvoju poslovnega procesa. Pogled na podatke sem nastavil svojim potrebam in tako sem lahko spremljal ali se poslovni proces izvaja pravilno.

Prednost modula BAM je da se lahko z njegovo pomočjo razvijejo različni pogledi na isto množico podatkov. Tako so zadoščene poslovne potrebe različnih skupin uporabnikov. Vmesnik za dostop do podatkov je za uporabnike enostaven in prijazen, ker izvaja z uporabo preglednic v Microsoft Office Excel.



Slika 21: BAM modul za spremljanje števila uspešno in neuspešno izvedenih prenosov

7. Sklepne ugotovitve

Z izdelavo diplomske naloge sem dobro spoznal in tudi v praksi uporabil strežnik BizTalk 2006. Strežnik BizTalk je sistem, ki ponuja veliko funkcionalnosti in je popolnoma usmerjen k razširljivosti, kar dokazuje s tem da se ponudniki ostalih informacijskih sistemov zavzemajo za razvoj vmesnikov s katerimi bi se lahko povezali na vodilo strežnika BizTalk. Povezovanje sistemov, v informacijski tehnologiji in tudi širše v gospodarstvu, vse bolj pridobiva na veljavi, zaradi teženj k zmanjševanju stroškov in poenostavitvi poslovanja.

Tehnologija storitvenega vodila na katero se BizTalk opira je zelo odprta arhitektura in omogoča nenehen razvoj in širitev obsega integracij popolnoma različnih sistemov. Veliko dodano vrednost vidim v tem, da lahko organizacije izvajajo poslovne procese, v katerih sodelujejo poljubne aplikacije in sistemi znotraj organizacije, kot tudi širše. Ta funkcionalnost temelji na storitvenem vodilu. Poslovni procesi, ki tečejo preko več različnih informacijskih sistemov ali celo organizacij so tudi varni predvsem pa razširljivi in prilagodljivi.

V diplomskem delu sem se najprej seznanil z upravljanjem poslovnih procesov, kar se je izkazalo, da je teoretični temelj za sistematičen razvoj rešitev s strežnikom BizTalk. Teoretično sem preučil tudi področje storitveno usmerjene arhitekture in integracije aplikacij.

Zaradi sodelovanja podjetja IXTLAN-TEAM, ki mi je omogočilo dostop do njihovih gradiv, sem lahko izdelal natančen pregled in analizo obstoječega stanja in zajel uporabniške zahteve. V fazi načrtovanja sem razvil model poslovnega procesa z uporabo orodja Microsoft Office Visio. Izdelava načrta mi ni povzročala večjih težav, ker je analiza jasna in pregledna in ker poslovni proces ni preveč kompleksen.

Razvoj rešitve sem začel z namestitvijo in nastavitvijo parametrov strežnika BizTalk. Nameščanje strežnika je enostaven in v veliki meri avtomatiziran postopek. Privzete nastavitve strežnika BizTalk omogočajo, da lahko razvijalec takoj prične z razvojem rešitev. Spreminjanje naprednejših nastavitev pa je lahko precej zamuden in zapleten postopek, saj so kakšne nastavitve v orodjih preveč skrite. Upravljalska konzola omogoča centralen in celovit pregled nad vsemi nameščenimi aplikacijami, kar mi je precej olajšalo delo, saj je večina temeljnih funkcionalnosti za upravljanje aplikacij združenih na enem mestu. Razvojno okolje strežnika BizTalk je integrirano v Visual Studio. Ta pristop se mi zdi zelo dober, ker lahko vse komponente za strežnik BizTalk, ki jih je potrebno razviti po meri, uporabnik razvija v istem razvojem okolju kot samo rešitev BizTalk.

Rešitve katero sem v okviru te diplomske naloge predstavil in implementiral zelo dobro uresničuje uporabniške zahteve, njena prava prednost pred obstoječo rešitvijo pa se bo pokazala, ko bo implementirana v celoti. Razvoj aplikacij s strežnikom BizTalk ni zapleten in razvijalcu ponuja veliko že razvitih komponent, kar proces razvoja pohitri. Zato menim, da strežnik BizTalk za svojo ceno ponuja zelo dobro razvojno in produkcijsko okolje.

8. Priloge

8.1. Sheme dokumentov

8.1.1. Prenos.xsd

```
<?xml version="1.0" encoding="utf-16"?>
<xs:schema xmlns:ns0="http://Prenosi.PropertySchema.PropertySchema"
xmlns:b="http://schemas.microsoft.com/BizTalk/2003"
xmlns="http://Prenosi.CP" targetNamespace="http://Prenosi.CP"
xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:annotation>
    <xs:appinfo>
      <b:imports xmlns:b="http://schemas.microsoft.com/BizTalk/2003">
        <b:namespace prefix="ns0"
uri="http://Prenosi.PropertySchema.PropertySchema"
location=".\propertyschema.xsd" />
      </b:imports>
    </xs:appinfo>
  </xs:annotation>
  <xs:element name="Prenos">
    <xs:annotation>
      <xs:appinfo>
        <b:properties>
          <b:property name="ns0:Status" xpath="/*[local-name()='Prenos' and
namespace-uri()='http://Prenosi.CP']/*[local-name()='Status' and namespace-
uri()='']" />
          <b:property distinguished="true" xpath="/*[local-name()='Prenos'
and namespace-uri()='http://Prenosi.CP']/*[local-name()='Status' and
namespace-uri()='']" />
        </b:properties>
      </xs:appinfo>
    </xs:annotation>
    <xs:complexType>
      <xs:sequence>
        <xs:element name="Status" type="xs:string" />
        <xs:element name="SplosniPodatki">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="EMSO" type="xs:string" />
              <xs:element name="Ime" type="xs:string" />
              <xs:element name="Priimek" type="xs:string" />
            </xs:sequence>
          </xs:complexType>
        </xs:element>
        <xs:element name="Naslov">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="Ulica" type="xs:string" />
              <xs:element name="HisnaSt" type="xs:string" />
              <xs:element name="PostnaSt" type="xs:string" />
              <xs:element name="Posta" type="xs:string" />
              <xs:element name="Naselje" type="xs:string" />
              <xs:element name="Drzava" type="xs:string" />
```

```
            </xs:sequence>
          </xs:complexType>
        </xs:element>
        <xs:element name="Rezidentstvo">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="DatRojstva" type="xs:string" />
              <xs:element name="DatSmrti" type="xs:string" />
              <xs:element name="DatPriselitve" type="xs:string" />
              <xs:element name="DatIzselitve" type="xs:string" />
              <xs:element name="RezStatus" type="xs:string" />
            </xs:sequence>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

8.1.2. ZavezanecEnv.xsd

```
<?xml version="1.0" encoding="utf-16"?>
<xs:schema xmlns:ns0="http://Prenosi.CP"
xmlns:b="http://schemas.microsoft.com/BizTalk/2003"
targetNamespace="http://Prenosi.CPEnv"
xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:import schemaLocation=".\prenos.xsd" namespace="http://Prenosi.CP" />
  <xs:annotation>
    <xs:appinfo>
      <b:schemaInfo is_envelope="yes"
xmlns:b="http://schemas.microsoft.com/BizTalk/2003" />
      <b:references>
        <b:reference targetNamespace="http://Prenosi.CP" />
      </b:references>
    </xs:appinfo>
  </xs:annotation>
  <xs:element name="Prenosi">
    <xs:annotation>
      <xs:appinfo>
        <b:recordInfo body_xpath="/*[local-name()='Prenosi' and namespace-
uri()='http://Prenosi.CPEnv']" rootTypeName="Prenosi" />
      </xs:appinfo>
    </xs:annotation>
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="ns0:Prenos" />
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

8.1.3. SQLService.xsd

```
<?xml version="1.0"?>
```

```
<xs:schema attributeFormDefault="unqualified"
elementFormDefault="qualified" targetNamespace="http://Prenosi"
version="1.0" xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:annotation>
    <xs:appinfo>
      <msbtssql:sqlScript value="exec [SP_Save] @EMSO=&amp;quot; &amp;quot;,
@Ime=&amp;quot; &amp;quot;, @Priimek=&amp;quot; &amp;quot;, @Naslov=&amp;quot; &amp;quot;,
@Posta=&amp;quot; &amp;quot;, @Naselje=&amp;quot; &amp;quot;, @Drzava=&amp;quot; &amp;quot;,
@DatRojstva=&amp;quot; &amp;quot;, @DatSmrti=&amp;quot; &amp;quot;, @RezStatus=&amp;quot;
&amp;quot;" xmlns:msbtssql="http://schemas.microsoft.com/BizTalk/2003" />
    </xs:appinfo>
  </xs:annotation>
  <xs:element name="PrenosRequest">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="SP_Save">
          <xs:complexType>
            <xs:attribute name="EMSO" type="xs:string" />
            <xs:attribute name="Ime" type="xs:string" />
            <xs:attribute name="Priimek" type="xs:string" />
            <xs:attribute name="Naslov" type="xs:string" />
            <xs:attribute name="Posta" type="xs:string" />
            <xs:attribute name="Naselje" type="xs:string" />
            <xs:attribute name="Drzava" type="xs:string" />
            <xs:attribute name="DatRojstva" type="xs:string" />
            <xs:attribute name="DatSmrti" type="xs:string" />
            <xs:attribute name="RezStatus" type="xs:string" />
          </xs:complexType>
        </xs:element>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="PrenosResponse">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="Success" type="xs:anyType" />
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

8.2. Podatkovna baza

8.2.1. Tabela

```
USE [Dav_Sistem]
GO
/****** Object:  Table [dbo].[Oseba]    Script Date: 08/05/2008 00:16:53
******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[Oseba](
[EMSO][nvarchar](13) COLLATE SQL_Latin1_General_CP1_CI_AS NOT NULL,
[Ime] [nvarchar](70) COLLATE SQL_Latin1_General_CP1_CI_AS NOT NULL,
```

```
[Priimek][nvarchar](70) COLLATE SQL_Latin1_General_CP1_CI_AS NOT NULL,
[Naslov][nvarchar](200) COLLATE SQL_Latin1_General_CP1_CI_AS NOT NULL,
[Posta][nvarchar](100) COLLATE SQL_Latin1_General_CP1_CI_AS NOT NULL,
[Naselje][nvarchar](100) COLLATE SQL_Latin1_General_CP1_CI_AS NOT NULL,
[Drzava][nvarchar](100) COLLATE SQL_Latin1_General_CP1_CI_AS NOT NULL,
[DatRojstva][nvarchar](10) COLLATE SQL_Latin1_General_CP1_CI_AS NOT NULL,
[DatSmrti][nvarchar](10) COLLATE SQL_Latin1_General_CP1_CI_AS NOT NULL,
[RezStatus][nvarchar](20) COLLATE SQL_Latin1_General_CP1_CI_AS NOT NULL
CONSTRAINT [PK_Customers] PRIMARY KEY CLUSTERED
(
[EMSO] ASC
)WITH (IGNORE_DUP_KEY = OFF) ON [PRIMARY]
) ON [PRIMARY]
```

8.2.2. Shranjena procedura

```
set ANSI_NULLS ON
set QUOTED_IDENTIFIER ON
go

ALTER PROCEDURE [dbo].[SP_Save]
   @EMSO          as nvarchar(13),
   @Ime                as nvarchar(70),
   @Priimek       as nvarchar(70),
   @Naslov        as nvarchar(200),
   @Posta         as nvarchar(100),
   @Naselje       as nvarchar(100),
   @Drzava        as nvarchar(100),
   @DatRojstva    as nvarchar(10),
   @DatSmrti      as nvarchar(10),
   @RezStatus     as nvarchar(20)
 AS


insert into Oseba (EMSO, Ime, Priimek, Naslov, Posta, Naselje, Drzava,
DatRojstva, DatSmrti, RezStatus)
values (@EMSO, @Ime, @Priimek, @Naslov, @Posta, @Naselje, @Drzava,
@DatRojstva, @DatSmrti, @RezStatus)
```

8.3. Izpostavitev polja iz sporočila

8.3.1. Sporočilo *Prenos*

```
<?xml version="1.0" encoding="utf-16"?>
<xs:schema xmlns:b="http://schemas.microsoft.com/BizTalk/2003"
xmlns="http://Prenosi.PropertySchema.PropertySchema"
targetNamespace="http://Prenosi.PropertySchema.PropertySchema"
xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:annotation>
    <xs:appinfo>
```

```
      <b:schemaInfo schema_type="property"
xmlns:b="http://schemas.microsoft.com/BizTalk/2003" />
    </xs:appinfo>
  </xs:annotation>
  <xs:element name="Property1" type="xs:string">
    <xs:annotation>
      <xs:appinfo>
        <b:fieldInfo propertyGuid="9481cb4a-3d05-43de-87b3-f225afa35755" />
      </xs:appinfo>
    </xs:annotation>
  </xs:element>
  <xs:element name="Status" type="xs:string">
    <xs:annotation>
      <xs:appinfo>
        <b:fieldInfo propertyGuid="62085a68-504c-4ca7-ad37-44c4762fd2ba" />
      </xs:appinfo>
    </xs:annotation>
  </xs:element>
</xs:schema>
```

9. Seznam uporabljenih virov

[1] M. Beckner, B. Gross, B. O'Reilly, "BizTalk 2006 Recipes", NewYork: Apress, 2006

[2] (2007) IBM Corporation: IBM WebSphere Enterprise Service Bus, Version 6.1. Dostopno na : http://www-01.ibm.com/software/integration/wsesb/

[3] D. Jefford, K.B. Smith, E. Fairweather, "Professional BizTalk Server 2006", Indianapolis, Wrox, 2007

[4] Microsoft Corporation: Microsoft ESB Guidance for BizTalk Server 2006. Dostopno na: http://msdn.microsoft.com/en-us/library/cc487894.aspx

[5] Microsoft Corporation: Microsoft BizTalk Server 2006 Help. Dostopno na: http://msdn.microsoft.com/en-us/library/aa548004.aspx

[6] (2006) Microsoft Corporation: Understanding BizTalk Server 2006. Dostopno na: http://www.microsoft.com/biztalk/techinfo/whitepapers/understanding.mspx

[7] Microsoft Corporation: Microsoft Terminology Translations. Dostopno na: http://www.microsoft.com/globaldev/tools/MILSGlossary.mspx

[8] (2007) Oracle Corporation: The "Mastering" SOA. Dostopno na: http://www.oracle.com/technology/tech/soa/mastering-soa-series/index.html

[9] W. A. Ruh, F. X. Maginnis, W. J. Brown, "Enterprise Application Integration", New York: Wiley Tech Brief Series, 2000

[10] M. T. Schmidt, B. Hutchison, P. Lambros, R. Phippen: "The Enterprise Service Bus: Making service-oriented architecture real", IBM Systems journal, št. 44, 2005

[11] M. Weske, "Business Process Management", Potsdam: Springer – Verlag 2007

[12] D. Woolston, "Foundations of BizTalk Server 2006", New York: Apress, 2007

Izjava o avtorstvu diplomskega dela

Izjavljam, da sem diplomsko nalogo izdelal samostojno in ob navedenih virih pod mentorstvom doc. dr. Mojce Ciglarič ter soglašam, da je diplomsko delo v elektronski obliki dostopno v zbirki 'Dela FRI'.

Borut Pirnat