UNIVERZA V LJUBLJANI
FAKULTETA ZA RAČUNALNIŠTVO IN INFORMATIKO

Jure Grom

ARHITEKTURE INTEGRACIJE PODEDOVANIH APLIKACIJ

DIPLOMSKO DELO NA UNIVERZITETNEM ŠTUDIJU

Mentor: doc. dr. Mojca Ciglarič

Ljubljana, 2009

Univerza
v Ljubljani

Fakulteta za računalništvo
in informatiko

Tržaška 25
1000 Ljubljana, Slovenija
telefon: 01 476 84 11
faks: 01 426 46 47
www.fri.uni-lj.si
e-mail: dekanat@fri.uni-lj.si

Št. naloge: 01562/2009
Datum: 05.04.2009

Univerza v Ljubljani, Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Kandidat: **JURE GROM**

Naslov: **ARHITEKTURA INTEGRACIJE PODEDOVANIH APLIKACIJ**
LEGACY APPLICATION INTEGRATION ARCHITECTURE

Vrsta naloge: Diplomsko delo univerzitetnega študija

Tematika naloge:

Opišite pomen in področja uporabe aplikacijske integracije. Osredotočite se na prikaz razlik in podobnosti na različnih nivojih integracije in vse nivoje opišite. Analizirajte arhitekturne pristope k integraciji podedovanih aplikacij in poslovnih aplikacij ter primerjajte s storitveno arhitekturo. Na konkretnem primeru predstavite izvedbo integracije podedovanih aplikacij in kritično ovrednotite prednosti in slabosti.

Mentor:
M. Ciglarič
doc. dr. Mojca Ciglarič

Dekan:
Franc Solina
prof. dr. Franc Solina

Fakulteta za računalništvo in informatiko • UNIVERZA V LJUBLJANI • 2

I Z J A V A O A V T O R S T V U

diplomskega dela

Spodaj podpisani/-a Jure Grom,

z vpisno številko 63010035,

sem avtor/-ica diplomskega dela z naslovom:

ARHITEKTURE INTEGRACIJE PODEDOVANIH APLIKACIJ

S svojim podpisom zagotavljam, da:

- sem diplomsko delo izdelal/-a samostojno pod mentorstvom (naziv, ime in priimek)

  doc. dr. Mojce Ciglarič

  in somentorstvom (naziv, ime in priimek)

  _____

- so elektronska oblika diplomskega dela, naslov (slov., angl.), povzetek (slov., angl.) ter ključne besede (slov., angl.) identični s tiskano obliko diplomskega dela

- soglašam z javno objavo elektronske oblike diplomskega dela v zbirki »Dela FRI«.

V Ljubljani, dne 02.06.2009 Podpis avtorja/-ice: _____

Zahvala

Zahvaljujem se mentorici, doc. dr. Mojci Ciglarič, za nasvete in usmeritve pri izdelavi diplomskega dela. Zahvaljujem se tudi sodelavcem iz podjetja Marand Inženiring d.o.o. za koristne nasvete in predloge pri realizaciji končne rešitve.

Hvala družini in Miheli za vso podporo in spodbudo med študijem.

Kazalo

Seznam uporabljenih kratic in pojmov

CRP – Centralni Register Prebivalstva

RPE – Register Prostorskih Enot

OGC – *Open Geospatial Consortium*: neprofitna mednarodna organizacija za razvoj standardov na področju geografskih informacijskih sistemov

WFS – *Web Feature Service Interface Standard*: specifikacija vmesnika in operacij za dostop in obdelavo geografsko značilnih podatkov

VPD – *Virtual Private Database*: navidezno zasebna podatkovna baza

RLS – *Row Level Security*: varnost na nivoju zapisa

EAI – *Enterprise Application Integration*: integracija aplikacij

SOA – *Service Oriented Architecture*: storitveno usmerjena arhitektura

ESB – *Enterprise Service Bus*: storitveno vodilo

ACID – *Atomicity, Consistency, Isolation, Durability*: atomarnost, konsistentnost, izolacija, trajnost

HKOM – prostrano privatno omrežje državnih organov

Hub and spoke – vozlišče in priključki

Messaging bus – sporočilno vodilo

Legacy application/system – podedovana aplikacija/sistem

Enterprise application/system – poslovna aplikacija: aplikacija za pomembnejše, težje naloge oz. opravila s ciljem večje produktivnosti in učinkovitosti

Business process – poslovni proces: povezana pravila in delovni tokovi

Interoperability – interoperabilnost: zmožnost izmenjave informacij in ponovne uporabljivosti različnih sistemov

Povzetek

Potrebe po informacijski podpori v podjetjih nenehno rastejo. Razvoj celotnega lastnega sistema je dolgotrajen in drag, več manjših samostojnih sistemov pa tudi ne zadovoljuje ustrezno vseh potreb. V primeru nakupa celostne rešitve bi pa zaradi splošnosti ostal del le te neizkoriščen. Z integracijo aplikacij dosežemo učinkovito uporabo posameznih aplikacij, saj s povezavo aplikacij dosežemo deljenje poslovne logike med aplikacijami ter odpravimo ponavljajoča opravila. Z dobro načrtovanim procesom integracije pa pridobimo tudi možnost hitrejših vključevanj sprememb v sistem tako z vidika funkcionalnosti kot zmogljivosti.

V diplomski nalogi sem naredil pregled različnih arhitektur za integracijo aplikacij. Večji poudarek je na arhitekturi integracije podedovanih aplikacij in različnih možnosti realizacije integracije. Pojem podedovana aplikacija se uporablja v razširjenem smislu in vanj spadajo vse aplikacije, ki niso bile načrtovane za šibko sklopljene povezave oz. niso storitveno usmerjene. Storitveno usmerjena arhitektura velja kot primer dobrega načina integracije aplikacij in prinaša vrsto dobrih lastnosti. Z opisanimi načini lahko tudi podedovane aplikacije postopoma preoblikujemo v storitveno usmerjene.

Sledi opis integracijske rešitve, ki povezuje podatke Centralnega Registra Prebivalcev (CRP) in Registra Prostorskih Enot (RPE) v nov register. Podatki v registru se dnevno osvežujejo s klici funkcij CRP in RPE, ki so podrobno opisane. Ažurni podatki pa so na voljo preko storitev ostalim aplikacijam.

Ključne besede: arhitektura integracije, podedovane aplikacije, CRP, RPE, SOA

Abstract

Information needs of modern businesses are constantly growing. Own development of a business information system is time consuming and expensive but, as an alternative, using smaller partial solutions doesn't cover all user needs. Decision to buy a universal solution, often results in only part of it being actively and efficiently used. By application integration we manage to achieve efficient use of applications, since by linking them, we enable sharing of business logics among applications and eliminate redundant tasks. When the integration process is well planned and structured we also gain the possibility of faster implementation of changes to the system, and therefore improvements from the point of functionality as well as performance.

My bachelor thesis presents and overview of different types of application integration architectures. Emphasis lays on architecture of legacy applications and different possibilities of integration realization. The term legacy application is used in accordance with its widest definition. The term covers all applications that weren't designed for loosely coupled integration – aren't service oriented. Service oriented architecture presents an example of application integration good practice that enables series of benefits. With described principles of integration we can also gradually transform legacy applications into service oriented.

In the second part of my bachelors' thesis I describe an integration solution that merges data gathered in Central population register (CRP) and in Register of spatial units (RPE) into a new register. Data in the register is refreshed daily by calling services CRP and RPE. These services are described in detail in this chapter. Refreshed data is available through services to other applications.

Keywords: integration architecture, legacy applications, CRP, RPE, SOA

1 Uvod

Danes posamezna aplikacija težko zadosti vsem potrebam podjetja po informacijski podpori. Strokovnjaki ocenjujejo, da sistem ERP pokrije od 30% do 40%, za vse ostalo so potrebne namenske rešitve prilagojene za posamezno podjetje. Potreba po razpoložljivosti in dostopnosti informacij vsak dan narašča in predstavlja izziv razvoju modernih aplikacij.

Marsikatero podjetje uporablja starejše programske rešitve, ki določen segment poslovanja pokrivajo povsem zadovoljivo, vendar pa ne morejo več slediti potrebam po informacijski podpori. V literaturi se tovrstne aplikacije imenuje podedovane aplikacije (legacy applications). Običajno tovrstne aplikacije niso bile zasnovane za integracijo z ostalimi aplikacijami, če pa že, pa je možna le tesna povezava aplikacij na enaki platformi. Ker pa so velikokrat ključne za delovanje podjetja kot celote, si podjetja tudi ne morejo privoščiti hitre zamenjave aplikacij, ponovni razvoj pa tudi ni vedno smiseln. Vsake toliko časa se tudi pokaže potreba po uvedbi nove aplikacije za pokrivanje novih informacijskih potreb. Nove aplikacije temeljijo na novejših tehnologijah in modernejših arhitekturah in niso vedno neposredno združljive z že obstoječimi (podedovanimi) aplikacijami. Za zadostitev kriterijema razpoložljivosti in dostopnosti, je potrebno vse aplikacije, tako podedovane kot nove, med seboj povezati. Informacijska infrastruktura mora stremeti k odpravljanju nepotrebnih oz. odvečnih opravil uporabnikov, kar lahko dosežemo le z dobro izvedeno integracijo aplikacij.

Cilj diplomske naloge je rešitev za ažuriranje podatkov iz Centralnega registra prebivalstva (CRP) in Registra prostorskih enot (RPE) za potrebe aplikacij Onkološkega Inštituta, kot so državni presejalni programi in Register raka. Za uspešno dosego cilja sem najprej naredil pregled možnosti za povezovanje aplikacij ter analiziral prednosti in slabosti posamezne arhitekture. Preučil sem tudi vmesnike obeh registrov, ki so na voljo za povezovanje. Kot uporabnik storitev registrov nisem imel vpliva na funkcionalnosti, ki jih ponujata. S pridobljenim znanjem in upoštevanjem zahtev sem nato izbral najprimernejšo arhitekturo in izdelal rešitev za dnevno ažuriranje podatkov.

V 2. in 3. poglavju diplomskega dela je predstavljen pojem integracije, kakšne so naloge, izzivi in cilji integracije, kje se integracija lahko izvaja in na kakšen način. Predstavljene so nekatere arhitekture, pristopi in tehnologije, ki se lahko uporabijo za reševanje različnih integracijskih problemov.

V 4. poglavju so opisane storitve dveh zbirk podatkov, že omenjena registra CRP in RPE. Z uporabo storitev vmesnikov registrov je mogoč dostop do podatkov. V Centralnem registru prebivalstva se osebni podatki prebivalcev Republike Slovenije zbirajo, obdelujejo, hranijo in uporabljajo z namenom spremljanja stanja prebivalstva. Podatki o bivališču prebivalcev se sklicujejo na podatke iz Registra prostorskih enot. RPE hrani podatke o občinah, naseljih, ulicah, hišnih številkah, upravnih enotah, statističnih okoliših, … in poleg opisnih podatkov vsebuje tudi geografske podatke (koordinate, površina).

5. poglavje opisuje rešitev, ki smo jo razvili v podjetju Marand d.o.o. za Onkološki Inštitut. Na Onkološkem Inštitutu izvajajo državna presejalna programa zgodnjega odkrivanje raka DORA in ZORA ter vzdržujejo Register Raka: Za učinkovito delovanje potrebujejo ažurne podatke o prebivalcih. Podatki so se pred uvedbo rešitve ažurirali enkrat mesečno preko datotek, ki so vsebovale podatke o spremembah. Takšen način je obremenjeval skrbnika posamezne aplikacije, pa tudi časovni interval ni najprimernejši. Nova aplikacija uvaja avtomatično dnevno osveževanje osebnih podatkov z uporabo storitev CRP in RPE.

2 Integracija aplikacij

Potrebe po informacijski podpori nenehno rastejo. Če tudi v nekem trenutku aplikacije zadostno pokrivajo potrebe, se lahko že v naslednjem trenutku pokažejo nove potrebe zaradi novih projektov ali reorganizacije poslovanja. Celostne rešitve so drage in ne ponujajo vedno zadovoljive podpore. Po drugi strani pa se v primeru več manjših aplikacij področja prekrivajo, kar pomeni ponavljanje opravil in podatkov.

Idealna situacija bi bila, ko bi vsaka aplikacija delovala le na področju, ki ga najbolje pokriva, komponente aplikacij bi bile na voljo tudi drugim aplikacijam, podatki pa bi se vnašali le na enem mestu in se ne bi podvajali in z neprestano dostopnostjo do podatkov. K takšnemu stanju stremi proces integracije aplikacij.

Glavna cilja pri integraciji aplikacij sta:

- enkratni vnos podatkov (*Single Data Input*) in
- dostop do informacij z minimalno zakasnitvijo (*Low Latency Information Access*).

Enkratni vnos podatkov je zahteva, da se podatki v sistem vnašajo le enkrat. S tem se zagotavlja konsistenca podatkov ter minimizira napake zaradi morebitnega večkratnega vnosa in lokalnih podatkovnih baz. Dostop do informacij z minimalno zakasnitvijo je pomemben, ker se morajo spremembe v enem delu sistema odraziti tudi v ostalih takoj oz. v najkrajšem možnem času.

2.1 Nivoji integracije

Oblikovanja integracijske arhitekture se običajno lotimo sistematično. Problem integracije razbijemo na več podproblemov, ki ji lažje obvladujemo. Kljub temu, da je vsak projekt integracije edinstven v kombinaciji aplikacij, orodij in tehnologij, lahko integracijo razdelimo na več nivojev, vsak nivo pa lahko predstavlja enega izmed podproblemov. Integracijski proces se običajno prične na najnižjem nivoju, nato pa postopoma napreduje po nivojih. Najpomembnejši so naslednji nivoji [1]:

- nivo integracije podatkov,
- nivo integracije aplikacij,
- nivo integracije poslovnih metod in procesov ter
- integracija predstavitvenega nivoja.

V [2] sta omenjena še dva dodatna nivoja. Prvi je nivo integracije platforme, katere cilj je zagotoviti prenosljivost med različnimi operacijskimi sistemi. Ta cilj je danes zaradi uporabe odprtih standardov relativno enostavno dosegljiv. Problem se lahko pojavi v podedovanih sistemih, ki nimajo več podpore. Drugi je nivo integracije podjetja s podjetjem (B2B). Ta nivo izhaja iz potreb po sodelovanju med podjetji, predpogoj pa je uspešna integracija ključnih aplikacij v podjetju.

2.1.1 Integracija podatkov

Definicija integracije podatkovnega nivoja pravi [2], da se osredotoča na prenos podatkov med aplikacijami s ciljem deljenja istih podatkov različnih aplikacij brez vključevanja poslovne logike. Na tem nivoju se integracija največkrat začne in je tudi s tehničnega vidika enostavna tako za implementacijo kot za razumevanje.

Podatki so običajno shranjeni v podatkovni bazi, dostop do posamezne podatkovne pa je enostaven. Prednost te vrste integracije je, da ne zahteva spremembe aplikacij. Za realizacijo imamo naslednje možnosti:

- aplikaciji A se omogoči dostop do podatkovne baze aplikacije B,
- aplikacija A replicira podatke aplikacije B,
- vzpostavi se centralna podatkovna baza in
- enosmerna integracija.

Čeprav je implementacija dostopa do podatkovne baze relativno enostavna, pa sama integracija ni povsem trivialna. Razlog je v kompleksnosti podatkovne baze in v njihovem številu. Poznati moramo v kakšni obliki so shranjeni podatki, kdaj in kako pridobiti podatke in ne nazadnje poznati je potrebno tudi strukturo podatkovne baze. Le na ta način bo zagotovljena konsistentnost podatkovne baze in podatki v formatu, razumljivem vsem aplikacijam.

Včasih pa zaradi različnih razlogov (varnostna politika, nepodprta tehnologija, …) ne moremo dostopati do podatkovne baze. V tem primeru se lahko podatki izmenjujejo preko datotek ali pa s programskim vmesnikom.

Integracije na nivoju podatkov se lotimo z naslednjimi zaporednimi koraki [2]:

- identifikacija podatkov,
- identifikacija tipov podatkovnih baz,
- definicija modela in
- implementacija integracije.

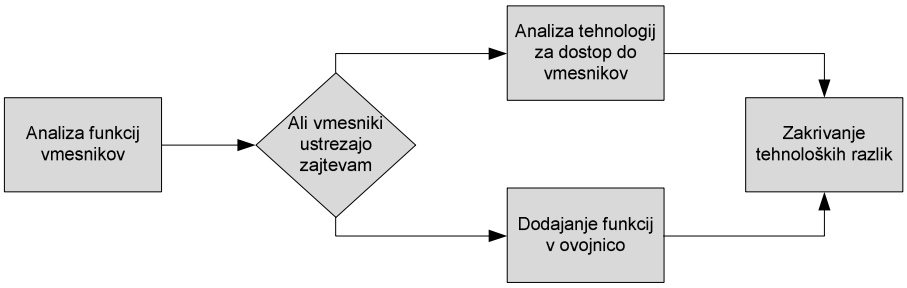
V koraku identifikacije podatkov analiziramo obstoječe ER diagrame in njihove implementacije. Na ta način dobimo vpogled v strukturo podatkov. Pomembno pa je tudi razumevanje semantike podatkov oz. kako aplikacije upravljajo s podatki. Ko razumemo logični model podatkov moramo spoznati tudi fizičnega. To storimo v drugem koraku. V tretjem koraku definiramo model, ki je predstavitev vseh podatkov v organizaciji na visokem nivoju. Model definira, katera množica podatkov predstavlja podporo določenemu delu v organizaciji in ne vsebuje vedno vseh detajlov podatkov, pomembni so le tisti za integracijo. Zadnji korak je sama implementacija.

2.1.2 Integracija aplikacij

Cilj integracije aplikacij je deljenje poslovne logike in ne zgolj deljenje podatkov kot na nivoju integracije podatkov. Takšen pristop je včasih celo nujen, ker so želeni podatki zaščiteni s plastjo poslovne logike in neposreden dostop do podatkov ne sme biti dovoljen.

Integracijo aplikacij lahko implementiramo npr. z uporabo vmesnikov API ali pa s spletnimi storitvami. Zaželena lastnost vmesnika je, da zakrije tehnološko specifičnost in da zagotavlja interoperabilnost. Vmesnik je pogodba med aplikacijami, ki ne spreminja svoje bistvene funkcionalnosti, dokler se pogodba ne spremeni.

Integracije aplikacij se lotimo po korakih, kot ji prikazuje slika 1.



Slika 1. Koraki integracije aplikacij.

V koraku analize funkcij oz. funkcionalnosti dostopnih preko vmesnikov preverimo, ali imamo primeren in omogočen dostop do vseh potrebnih funkcionalnosti aplikacije, ki jih potrebujemo za integracijo. Včasih je dostop do določene funkcionalnosti možno realizirati na več načinov, zato je potrebno poiskati najbolj ustreznega potrebam. V primeru ustreznih funkcionalnosti naredimo pregled tehnologij dostopa oz. klicev funkcij. API je lahko npr. realiziran sinhrono ali asinhrono, uporablja lahko npr. CORBO ali DCOM, … Če pa ne moremo dostopati do vseh funkcionalnosti aplikacije, bo le to potrebno nadgraditi oz. ji dodati nove vmesnike za manjkajoče funkcionalnosti v ovojnico aplikacije. V zadnjem koraku poskrbimo za maskiranje tehnoloških razlik med integriranimi aplikacijami. To lahko storimo npr. z navideznimi komponentami. Vsaka navidezna komponenta predstavlja posamezno aplikacijo in uporablja specifično tehnologijo za dostope do API-ja.

2.1.3 Integracija poslovnih metod in procesov

Deljenje podatkov in poslovne logike na tem nivoju omogoča povezavo obstoječih procesov, ki so lahko v različnih aplikacijah, v nov, kompleksnejši in avtomatiziran proces. Z integracijo na tem nivoju dosežemo, da obstoječe aplikacije delujejo na način, kot bi delovala ena sama, z vsemi funkcionalnosti obstoječih. Prednost integracije je tudi, da ni potrebno takšnega sistema začeti razvijati na novo, ampak lahko uporabimo obstoječe aplikacije.

Z integracijo poslovnih metod in procesov dosežemo večjo fleksibilnost sistema in lažje uveljavljanje sprememb v poslovnih procesih in nove poslovne procese. Za modeliranje poslovnih procesov v storitveno usmerjenih arhitekturah se uporablja jezik BPEL (*Business Process Execution Language*), samo izvajanje procesov pa nadzorujeta oz. usmerjata

komponenti posrednik (*broker*) in orkestracija procesov (*orchestration*). Če ne uporabljamo naštetih komponent in vmesne opreme, imamo lahko težave s prevelikim številom povezav od točke do točke med posameznimi procesi v aplikacijah.

Pri integraciji poslovnih metod in procesov sledimo naslednjim zaporednim korakom:

- definicija visoko nivojskih vmesnikov,
- povezovanje operacij in
- implementacija.

Definicija visoko nivojskih vmesnikov je zahtevno opravilo, ker je potrebno definirati vmesnik, ki je dovolj fleksibilen, da bo zadostoval tudi prihajajočim potrebam. Pri definiranju je dobro podpreti vse zahteve integriranega sistema, prilagoditi arhitekturo organizaciji in imeti v mislih tudi prihajajoči razvoj oz. potrebe. Rezultat definicije je globalni model organizacije. V naslednjem koraku določimo, kako naj bodo povezane posamezne funkcionalnosti visoko nivojskih vmesnikov med seboj in poskušamo čim bolj ponovno uporabiti že obstoječe funkcije aplikacij.

2.1.4 Integracija predstavitvenega nivoja

Pogosto, ko naredimo integracijo poslovnih metod in procesov, nadaljujemo z integracijo predstavitvenega nivoja. Uporabniku ponudimo novo aplikacijo oz. nov poenoten uporabniški vmesnik, ki uporablja nove poslovne metode in procese, ki so rezultat prejšnjega nivoja. Na ta način se odpravi neprijetno preklapljanje med aplikacijami.

Novi predstavitveni nivo uporablja funkcionalnosti združenih poslovnih procesov in je šibko sklopljen z obstoječimi aplikacijami in se ne zaveda podrobnosti posameznega izvajanja procesa. Tako se pokaže možnost, kako lahko v prihodnosti zamenjamo podedovane sisteme z novejšimi, brez vliva na sistem kot celoto.

K integraciji predstavitvenega nivoja pristopimo z naslednjimi koraki:

- izbira uporabniških vmesnikov za prenovo,
- analiza in zasnova novih uporabniški vmesnikov,
- povezava uporabniških vmesnikov na poslovni nivo in
- implementacija.

2.2 Pristop k integraciji

Pri oblikovanju integracijske rešitve moramo strmeti k naslednjim ciljem:

- Interoperabilno modularno ogrodje, ki omogoča nadgradnje.
- Uporabno odprtih, platformno neodvisnih tehnologij.
- Generična dostopnost in odprti vmesniki.

Cilji so dosegljivi, imajo pa lahko tudi precejšnjo ceno. Pri integraciji aplikacij oz. načrtovanju integracije so se za dobre prakse izkazali naslednji principi [2]:

- zasnova za prihodnost – zasnova arhitekture naj bo takšna, da omogoča v prihodnosti razširitve in prilagoditve,
- integriteta in abstrakcija vmesnikov – dobra zasnova vmesnikov pomeni, da jih ne bo potrebno spreminjati in spremembe v logiki bodo zahtevale spremembe le v eni aplikaciji,
- več dostopov do enakih funkcionalnosti – uporabimo tistega, ki najbolj ustreza našim potrebam,
- zmogljivost – nekatere (podedovane) aplikacije niso bile zasnovane za nova bremena, kar lahko povzroča težave,
- varnost – nekatere (podedovane) aplikacije niso potrebovale posebne obravnave varnosti, ker so delovale v zaprtih okoljih, z integracijo pa se le to spremeni.

Aplikacije lahko integriramo na dva načina, lahko so šibko sklopljene (*loose coupled*) ali pa tesno sklopljene (*tight coupled*). Pri aplikacijah, ki so tesno sklopljene lahko naletimo na naslednje težave:

- sprememba v eni aplikaciji pomeni spremembo tudi v ostalih aplikacijah,
- poslovna logika posameznega modula je težko razumljiva, če je izvzeta iz celote,
- module težko ponovno uporabimo in
- module težko preizkušamo, ker so tesno povezani z ostalimi.

Šibko sklopljene aplikacije imajo dobro definirane vmesnike, ki so platformno neodvisni, fleksibilni in se ne spreminjajo. To pomeni, da spremembe v modulih oz. poslovni logiki ne pomenijo sprememb v ostalih. Šibko sklopljene aplikacije so odraz dobrega načrtovanja in nas vodijo do interoperabilnosti. Čeprav imajo tesno sklopljene aplikacije večjo zmogljivost, pa prednosti šibko sklopljenih aplikacij, ki se odražajo tudi v razvoju in vzdrževanju aplikacij, odtehtajo ceno zmanjšanja zmogljivosti.

2.3 Integracijska platforma

Vmesna oprema (*middleware*) je širok pojem, ki lahko pomeni veliko različnih stvari. V kontekstu integracij aplikacij se z vmesno opremo označujejo rešitve za medaplikacijsko komunikacijo v heterogenih okoljih. Vmesna oprema ponuja infrastrukturo za povezovanje različnih aplikacij, kot tudi različnih plasti v večplastnih aplikacijah.

Vmesna oprema je lahko tudi dobra integracijska platforma. Pri izbiri primerne je treba upoštevati naslednje kriterije [2]:

- podpora integraciji z obstoječimi aplikacijami,
- podpora odprtim standardom in tehnologijam,
- robustnost, zmogljivost in nadgradljivost,

- prenosljivost in interoperabilnost,
- cena in podpora,
- obstoječa znanja in
- zrelost.

Trenutno sta ustrezni in najbolj razširjeni platformi .NET in J2EE. Obe podpirata več nivojske aplikacije, razvoj komponent in vse potrebne funkcionalnosti za poslovne aplikacije. J2EE je relativno odprta platforma, katere specifikacijo določa *Java Community Process*. Podpira le programski jezik Java, je pa neodvisna od okolja izvajanja. Prenosljivost ni 100% zagotovljena, ker posamezni proizvajalci vpeljejo določene specifičnosti. Na drugi strani za razvoj .NET skrbi Microsoft, kar pomeni omejenost pri izbiri izvajalnega okolja. Sicer obstajajo odprtokodne rešitve, ki pa ne sledijo tempu razvoja platforme pa tudi podpora je vprašljiva.

3 Arhitekture integracije aplikacij

Razširjenost XML in široka podpora spletnim storitvam v platformah .NET in J2EE sta dejavnika, ki sta omogočila uveljavitev storitveno usmerjene arhitekture – SOA. Glavna prednost SOA kot moderne arhitekture integracije aplikacij je, da omogoča hitrejše in učinkovitejše uvajanje sprememb v sistem zaradi prilagodljivosti potrebam podjetja. Hitro uvajanje sprememb je mogoče, ker so posamezne funkcionalnosti sistema razdeljene na storitve, te pa so smiselno povezane v celoto prek poslovnih procesov. S povezovanjem storitev v poslovne procese se zmanjšuje tudi semantični prepad med zaznavanjem sistema z vidika informatikov in z vidika vodstva podjetja. SOA ima tudi visok nivo interoperabilnosti, kar je zaželena lastnost aplikacij oz. sistemov.

SOA trenutno velja za obetaven pristop k integraciji aplikacij. Razvitih je bilo veliko načinov kako obstoječe sisteme vključiti oz. spremeniti v storitveno usmerjene. V tem poglavju so opisani nekateri od teh načinov. Vsak način predstavlja arhitekturo rešitve, predstavljene pa so tudi pozitivne in negativne lastnosti posamezne arhitekture.

Po definiciji, arhitektura integracije določa celostno strukturo, logične komponente in relacije med aplikacijami, ki jih želimo integrirati [2]. Razvrstitve in opisi arhitektur integracij v tem poglavju se pretežno zgledujejo po [3].

3.1 Integracija podedovanih aplikacij

Termin podedovana aplikacija se nanaša na aplikacije, ki niso bile načrtovane za šibko sklopljeno integracijo. Definicija podedovane aplikacije v [2] pravi, da je to aplikacija, ki izpolnjuje naslednja pogoja:

1. Aplikacija se uporablja in je še vedno uporabna.
2. Arhitektura aplikacije se smatra za zastarelo.

Najpogostejši predstavniki podedovanih aplikacij so eno nivojske aplikacije (*mainframe-based systems*) in aplikacije tipa odjemalec/strežnik. V skupino podedovanih aplikacij lahko uvrstimo tudi nekatere zunanje aplikacije. Kot zunanje podedovane aplikacije pojmujemo tiste aplikacije, pri katerih nimamo vpliva na način integracije. Podedovane zunanje aplikacije ne uporabljajo modernejših načinov integracije, kakršen je npr. SOA. V kontekstu SOA lahko opredelimo kot podedovano aplikacijo vsako pred obdobjem SOA [3].

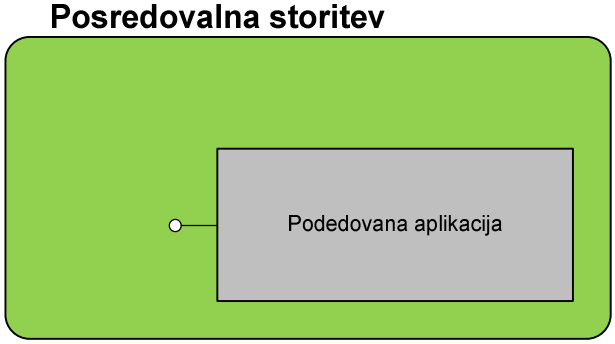
Podedovane aplikacije so pomembne za tekoče delovanje podjetja, kar pomeni, da je zamenjava ali nadgradnja le teh zahteven projekt, saj morajo biti podedovane aplikacije vedno na razpolago. Pogosto se v takšnih primerih ravna tako, da se z novimi aplikacijami le razširi obstoječe oz. se obstoječim dodajo nove funkcionalnosti.

Zaradi velikega pomena podedovanih aplikacij so zelo pomembni načini, kako jih spremeniti v storitveno orientirane.

Podedovane aplikacije lahko spremenimo v storitveno orientirane spremenimo prek:

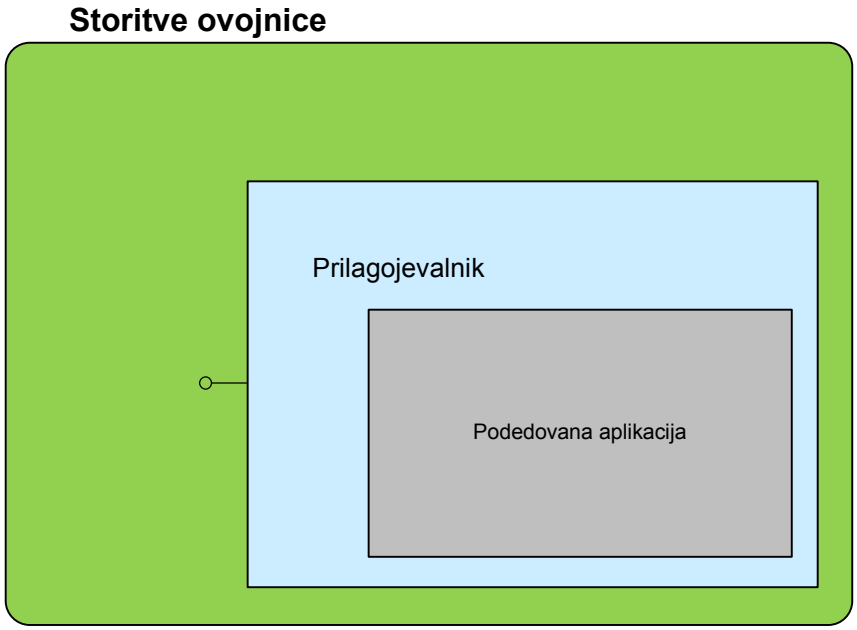
- Posredovalnih storitev (*proxy services*)
- Storitev ovojnice (*wrapper services*)
- Koordinacijskih storitev (*coordination services*)

Posredovalno storitev dobimo s preslikavo vmesnika komponente v spletno storitev. Obstajajo tudi orodja, ki ta postopek poenostavijo. Ta orodja omogočajo generiranje datotek WSDL na podlagi vmesnikov aplikacije. Z datoteko WSDL dobimo vmesnik spletne storitve, od vmesnika do implementacije pa ni več daleč. Kljub enostavnosti postopka, pa je spletna storitev lahko neučinkovita, kadar vmesnik komponente ali komponenta sama nista bila zasnovana za spletno storitev. To lahko povzroča težave pri delovanju.



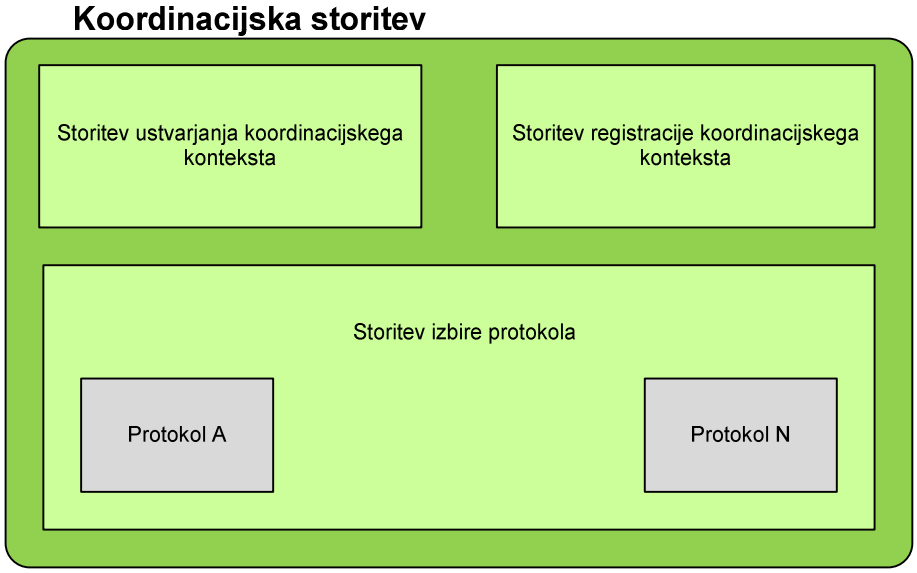
Slika 2. Shema posredovalne storitve.

V primeru, ko želimo izpostaviti le določene funkcionalnosti aplikacije oz. komponente, govorimo o storitvah ovojnice. Eden izmed načinov, kako lahko izpostavimo le določene funkcionalnosti aplikacije, je modifikacija generirane posredovalne storitve. V SOA arhitekturah se ta tip storitev imenuje tudi prilagoditvena storitev (*adapter services*). Storitve ovojnice so za razliko od posredovalnih storitev namensko razvite, prilagojene in ne preslikavajo zgolj vmesnika podedovane aplikacije, temveč predstavljajo tudi fasado funkcionalnostim podedovane aplikacije in omogočajo lažji dostop oz. poenostavitev klicev funkcij podedovane aplikacije. Kljub temu, pa storitev ovojnice ne dodaja nove poslovne logike.



Slika 3. Shema storitve ovojnice.

V primeru, da pri integraciji podedovanih potrebujemo podporo transakcijam, moramo implementirati koordinacijske storitve. Koordinacijske storitve implementirajo model opredeljen v specifikaciji WS-Coordination [4], ki je del specifikacije WS-Transaction [5]. Model predvideva 3 storitve in sicer za ustvarjanje konteksta (*context creation*), registracijo koordinacijskega konteksta (*registration for coordination context*) in izbiro protokola (*protocol selection*). S koordinacijskimi storitvami dosežemo sposobnost izvajanja transakcij z množico lastnosti ACID in omogočimo dodajanje nove poslovne logike v integrirane aplikacije. Lastnosti transakcij ACID so atomarnost, konsistentnost, izolacija in trajnost. Zagotavljajo zanesljivo izvajanje transakcije.

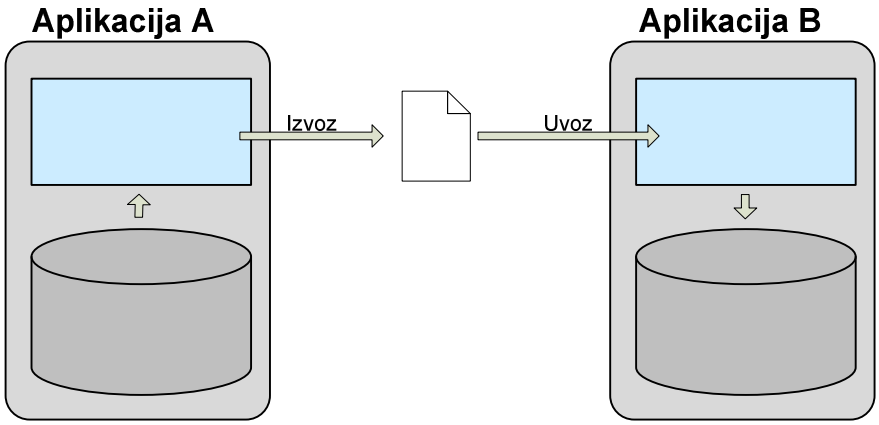


Slika 4. Shema koordinacijske storitve.

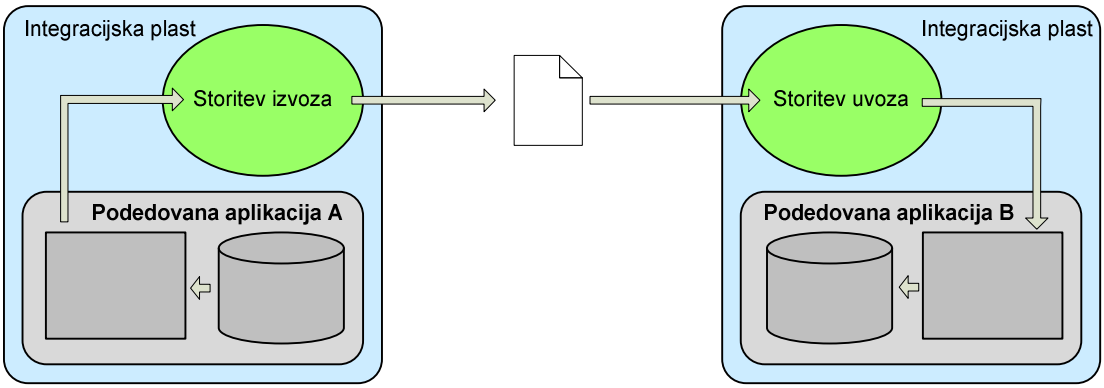
3.1.1 Arhitektura enosmerne integracije

Arhitektura enosmerne integracije predstavlja relativno enostavno in poceni rešitev na katero minimalno vplivajo v aplikacijah uporabljene tehnologije. Zaradi slednjega se enosmerna integracija pogosto uporablja še danes. Najbolj pogosta modela implementacije enosmerne integracije sta paketni uvoz/izvoz in neposreden dostop do podatkov.

Paketni uvoz/izvoz je način pri katerem ena aplikacija izvede izvoz podatkov v določenem formatu, npr. v nepovezano datoteko ali datoteko XML, druga aplikacija pa te podatke uvozi. Proces izvoza in uvoza lahko poteka bodisi avtomatično bodisi na zahtevo. Način je primeren za integracijo aplikacij, ki ne zahtevajo izmenjave podatkov v realnem času. Čeprav paketni uvoz/izvoz implementiramo relativno poceni, je lahko drag za vzdrževanje. To še posebej velja, kadar proces integracije poteka na zahtevo.



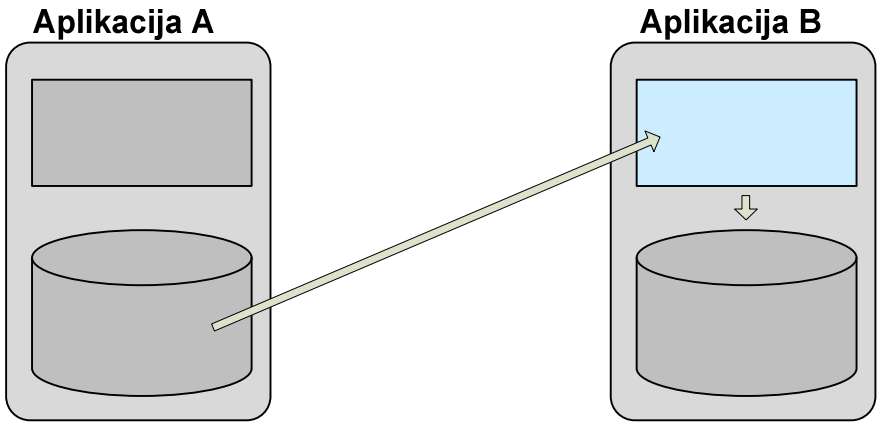
Slika 5. Shema za paketni uvoz/izvoz.



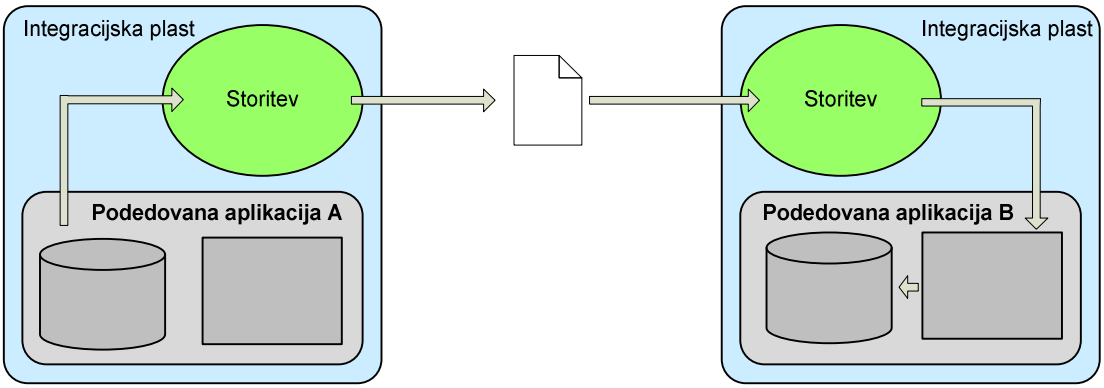
Slika 6. Shema storitveno usmerjenega paketnega uvoza/izvoza.

V primeru neposrednega dostopa do podatkov aplikacije, ki deli svoje podatke, običajno ni potrebno spreminjati, mogoče pa je, da so potrebni popravki podatkovne baze. V podatkovni bazi je potrebno nastavi pravice dostopa za novega odjemalca, lahko se naredi tudi posebne poglede na tabele podatkov. Način neposrednega dostopa do podatkov je primeren, kjer je poslovna logika implementirana v težkih odjemalcih in jo ne moremo uporabiti za integracijo. Prav tako je pristop primeren, kadar je poslovna logika implementirana v obliki baznih procedur, funkcij in prožilcev, saj jo lahko uporablja oz. je dostopna tudi integrirani aplikaciji.

Prednost neposrednega dostopa v primerjavi s paketnim uvozom/izvozom je, da omogoča izmenjavo podatkov v realnem času, njegova pomanjkljivost pa je, da povečuje obremenjenost podatkovne baze.



Slika 7. Shema neposrednega dostopa do podatkov.



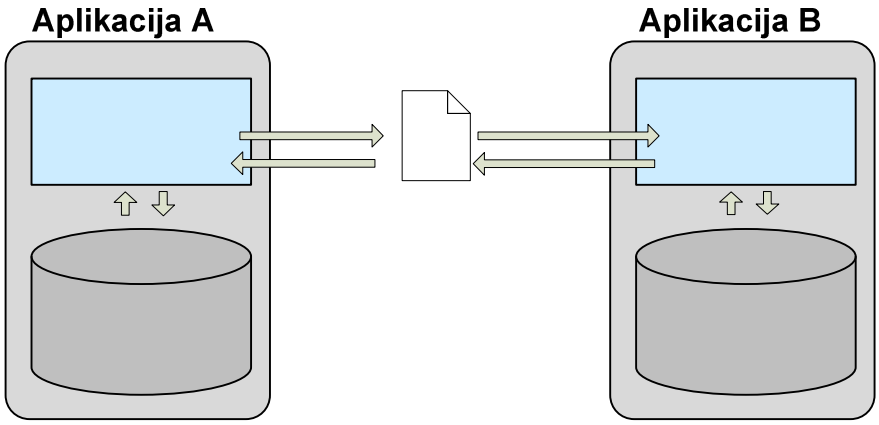
Slika 8. Shema storitveno usmerjenega neposrednega dostopa do podatkov.

3.1.2 Arhitektura integracije od točke do točke

Arhitektura od točke do točke je zelo razširjen način integracije aplikacij. Aplikacije se poveže z vzpostavitvijo neposrednega komunikacijskega kanala. To omogoča precejšnjo kontrolo nad posredovanjem in obdelavo podatkov, dosežemo pa tudi visok nivo interoperabilnosti.

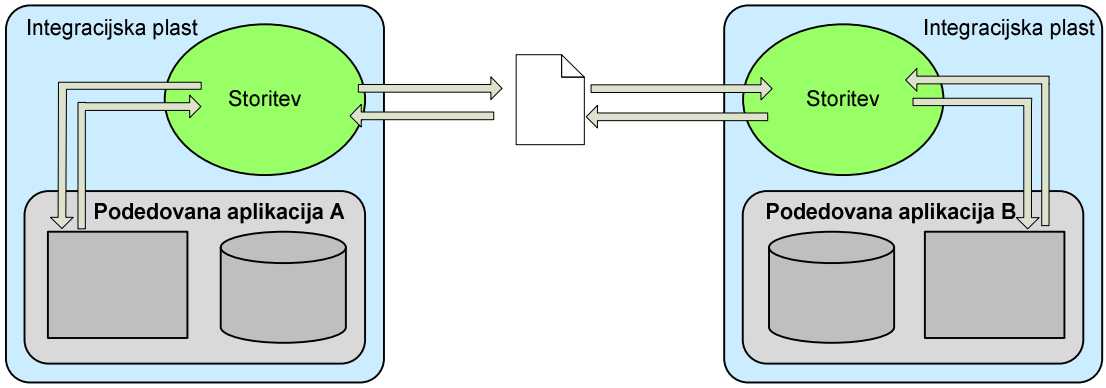
Povezovanje aplikacij, ki so razvite na enakih platformah, s to metodo je relativno enostaven postopek. Običajno gre za razširitev vsake aplikacije s funkcionalnostmi, ki so potrebne za integracijo. Razvijalec take povezave ima popoln nadzor nad prenosom podatkov. Sama integracija lahko postane nov, kompleksen in med-aplikacijski proces z možnostjo dolgotrajnih transakcij. Pri implementaciji integracije, lahko uporabimo tudi specifične tehnologije platforme, ki jo uporabljajo aplikacije. Slednje je lahko obenem prednost in slabost. Z uporabo specifičnih funkcionalnosti platforme si lahko poenostavimo in optimiziramo marsikateri postopek, ker integrirane aplikacije uporabljajo enako platformo in ni potrebno skrbeti za ohranjanje splošnosti. Po drugi strani pa s tem preprečimo integracijo z aplikacijami, ki uporabljajo drugačne platforme in bi jih morda želeli integrirati v prihodnosti.

Predvsem pa je zahteva po homogenosti platform aplikacij, povezanih med seboj z metodologijo od točke do točke, velika omejitev integracije.



Slika 9. Shema poveze aplikacij enake platforme.

Za aplikacije, ki ne uporabljajo enake platforme, je potrebno poiskati alternativni način integracije. Uporabimo lahko odprte tehnologije ali naredimo vmesni člen – vmesnik oz. prilagojevalnik (*adapter*). Vmesnik skrbi za komunikacijo med aplikacijami in rešuje probleme zaradi različnih platform. Najprimernejša implementacija vmesnika je spletna storitev. Več vmesnikov, tudi različnih aplikacij, združimo v integracijsko plast.



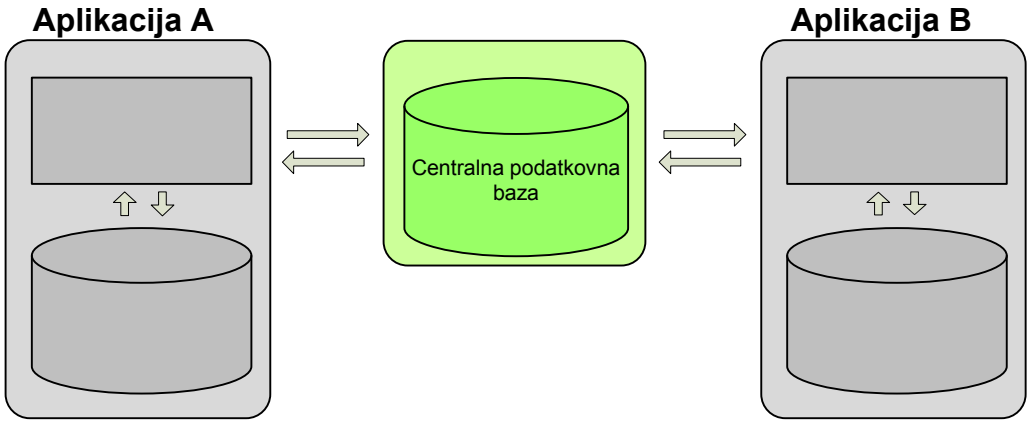
Slika 10. Shema povezave aplikacij različnih platform.

Slabost integracije od točke do točke je, da z naraščanjem sodelujočih aplikacij eksponentno narašča kompleksnost. To ima za posledico težko upravljanje in vzdrževanje. Prav tako lahko pride do problema integritete podatkov.

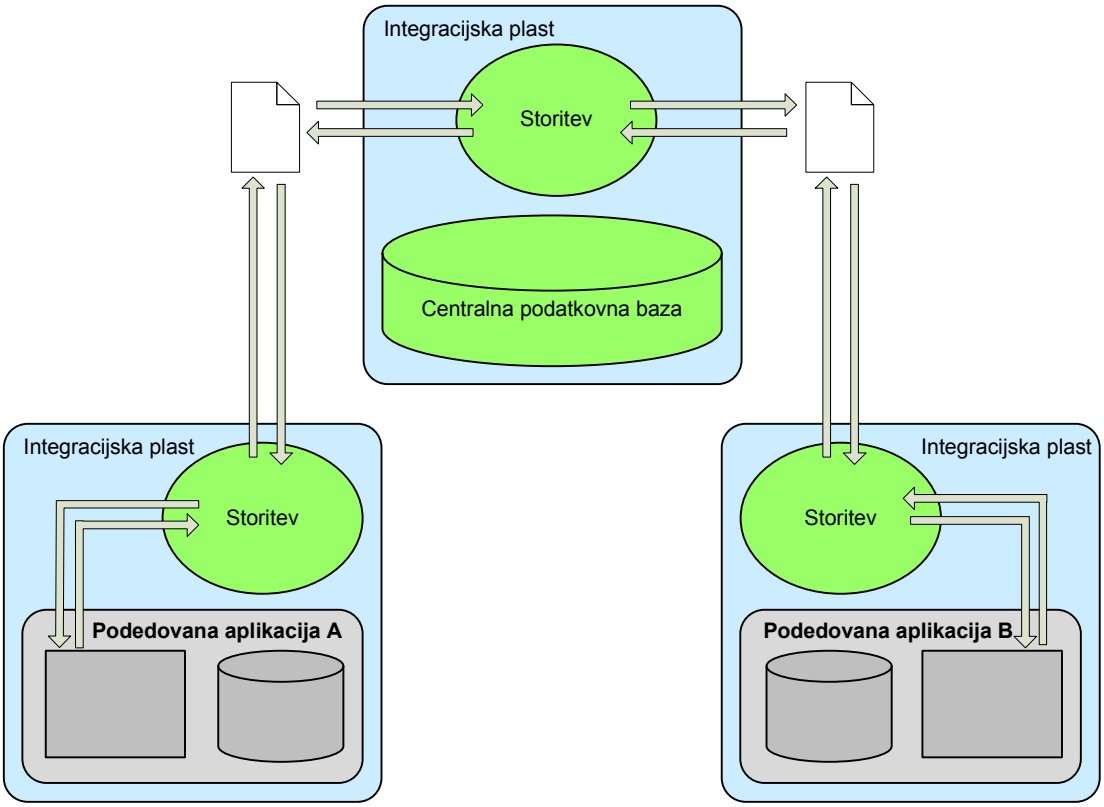
3.1.3 Arhitektura centralne podatkovne baze

Ena izmed možnosti povezovanja aplikacij je tudi centralni repozitorij podatkov – podatkovna baza. Z deljenjem podatkov v podatkovni bazi lahko pridobijo vse aplikacije, posebno še, če je v podatkovni bazi shranjena tudi poslovna logika v obliki baznih procedur in prožilcev.

Pristopa k implementaciji arhitekture centralne podatkovne baze sta dva. Prva možnost je, da so v skupni podatkovni bazi shranjeni samo skupni podatki. Na ta način preprečimo ponavljanje podatkov. Druga možnost pa, da so v njej shranjeni vsi podatki.



Slika 11. Shema centralne podatkovne baze.



Slika 12. Shema storitveno usmerjene centralne podatkovne baze.

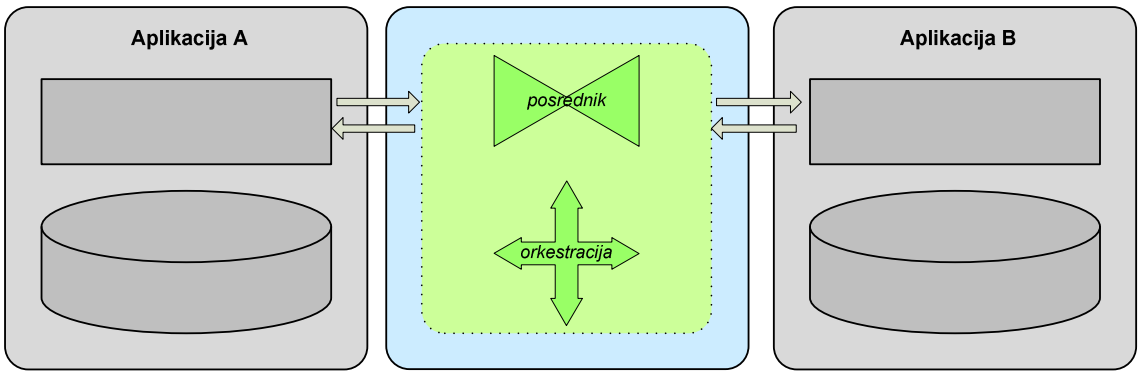
Ker se vsi podatki prenašajo preko centralne podatkovne baze, obstaja nevarnost, da podatkovna baza postane ozko grlo. To je potrebno upoštevati pri načrtovanju. Problem je tudi nevarnost izpada centralne podatkovne baze, saj to onemogoči delovanje vseh aplikacij, vključenih v integracijo.

3.2 Integracija (poslovnih) aplikacij

Podjetja za svoje delovanje potrebujejo več informacijskih sistemov, ki pokrivajo posamezne segmente poslovanja. Namenske rešitve za integracijo informacijskih sistemov zadostijo potrebam, vendar pa njihov največji problem to, da rešujejo točno določen problem,

problematično je tudi njihovo drago vzdrževanje. V primeru, da je potrebno spremeniti oz. nadomestiti del celotnega sistema, je treba popraviti tudi integracijsko rešitev. Kot primer dobre praksa se je pokazalo oblikovanje odprte in od ponudnika neodvisne integracijske rešitve. Uveljavitev XML, popularizacija spletnih storitev kot tehnološke platforme in integracija poslovnih aplikacij EAI kot model integracije sta dala nov zagon integracijskemu procesu.

Integracijske rešitve, ki slonijo na modelu EAI, so procesno orientirane. Vmesna oprema ponuja ogrodje za storitve, ki predstavljajo procese. Za opis procesov se je najbolj razširjena uporaba specifikacije BPEL4WS. Skupaj s standardoma *WS-Transaction* in *WS-Coordination* predstavlja BPEL4WS ogrodje za povezavo procesov v nove kompleksnejše procese ter za upravljanje novih kompleksnejših procesov. Dobra stran BPEL4WS je tudi odprta usmerjenost in neodvisnost od implementacije, kar zagotavlja prenosljivost, ponovno uporabljivost in neodvisnost od sprememb implementacije. Velik pomen pri uvajanju kompleksnejših poslovnih procesov imajo koordinacijske storitve, saj omogočajo dalj časa trajajoče transakcije. Posamezne poslovne procese se povezuje v nove, kompleksnejše poslovne procese z orodjem za orkestracijo poslovnih procesov.



Slika 13. Shema modela EAI.

V modelu EAI imata ključno vlogo komponenti posrednik (*broker*) in orodje za orkestracijo poslovnih procesov (*orchestration engine*). Glavna naloga posrednika je, da sporočila izvora prispejo v pravilni obliki do ponora. Posrednik preoblikuje sporočila, združuje dokumente iz različnih virov ali dopolni dokument z vsebino iz različnih aplikacij. Orkestracija skrbi za usmerjanje sporočil med posredniki, integracijo aplikacij za pridobivanje podatkov in integriranje obstoječe poslovne logike.

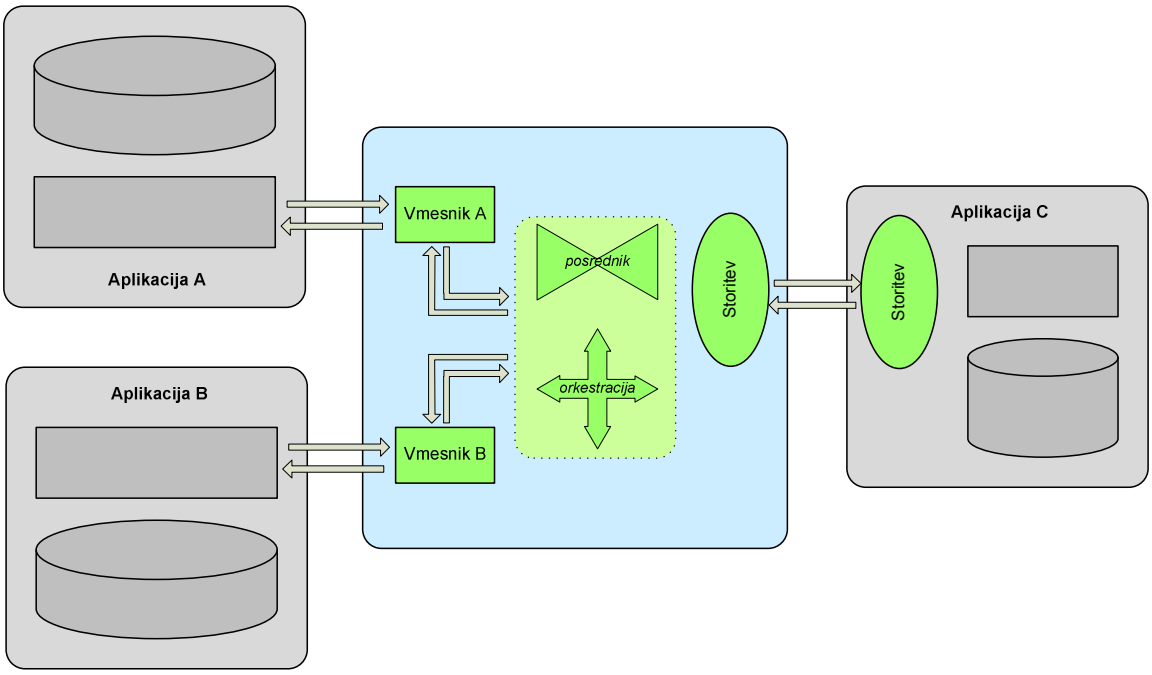
Vsako sporočilo ima stanje, ki je lahko permanentno ali procesno. V primeru permanentnega stanja sporočilo služi za prenos podatkov, če pa je stanje procesno, to pomeni da stanje sporočila določa tudi stanje izvajanja poslovnega procesa.

Zaradi sodelovanja posrednika v procesu integracije se pri orkestraciji aplikacijam ni potrebno zavedati ostalih aplikacij vključenih v integracijo in njihove vloge v procesu integracije, kar je dobra lastnost orkestracije. Naloga posrednika je tudi prilagajanje sporočila različnim aplikacijam.

Večina EAI rešitev uporablja asinhrona sporočilna ogrodja. Na ta način se odpre prostor za povezovanje med sodelujočimi aplikacijami v integracijskem procesu. V EAI okolju je XML najboljša izbira za prenos podatkov. Široka podpora spletnim storitvam je omogočila vključitev XML strukture podatkov v takšna okolja. Pri implementaciji spletnih storitev je treba biti pozoren, ker ima vsaka tehnološka platforma nekaj specifičnosti pri implementaciji, kar lahko onemogoča komunikacijo med rešitvami različnih proizvajalcev oz. različnih platform . Rešitev ponuja specifikacija *WS-Interoperability*, ki daje smernice, kako definirati spletne storitve, da bodo neodvisne od platforme.

3.2.1 Vozlišče in priključki

Vozlišče in priključki (*hub and spoke*) je arhitektura s centralizirano integracijsko logiko, ki skrbi za orkestracijo in posredništvo med vsemi povezanimi aplikacijami. S serijo vmesnikov (priključki) lahko priključimo na strežnik veliko raznolikih aplikacij, ki jih povezuje orkestracija. S povezavo aplikacije na vozlišče odpade potreba po poznavanju povezanih aplikacij med seboj z vidika aplikacije. Posredniki s svojo funkcionalnostjo omogočajo ena proti ena, ena proti mnogo in mnogo proti mnogo kombinacij povezav podatkov. Ker je procesiranje centralizirano, se zmanjša potencialna redundanca podatkov in obseg obdelave podatkov. Tudi sam nadzor je bolj enostaven. Čeprav ima centralizirana rešitev številne prednosti, pa ima tudi pomanjkljivosti. Centralizirana arhitektura lahko predstavlja ozko grlo, pa tudi kritično točko odpovedi. V primeru odpovedi strežnika, bi bilo onemogočeno delovanje vsem aplikacijam povezanih z integracijo vozlišča in priključkov.



Slika 14. Shema arhitekture integracije vozlišče in priključki.

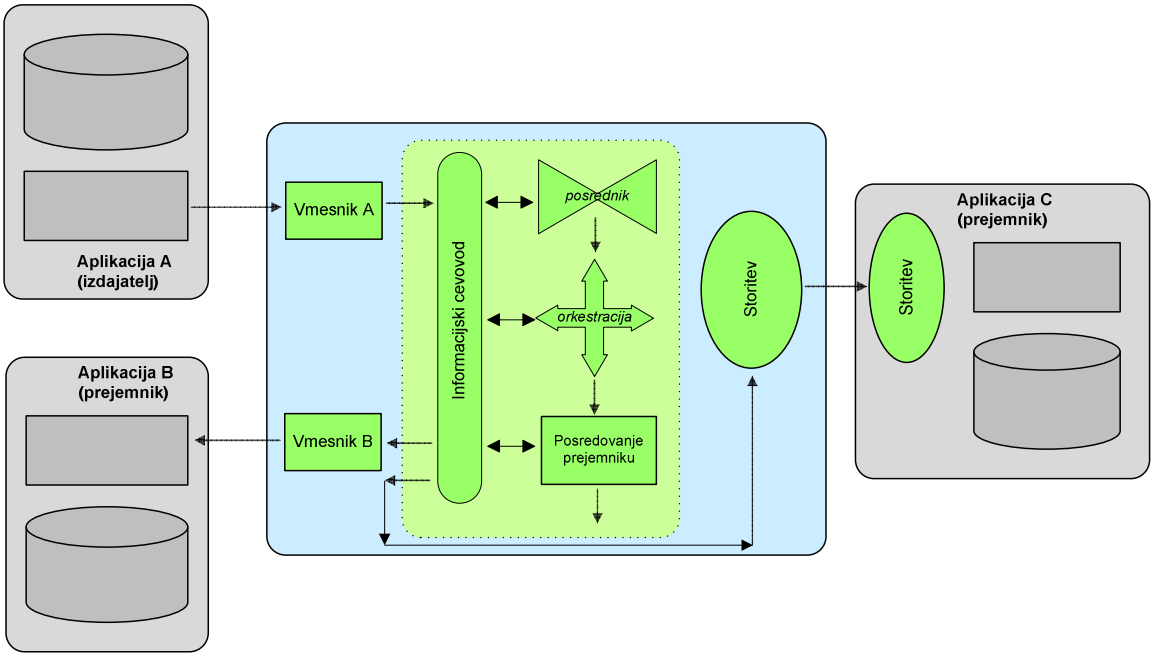
3.2.2 Sporočilno vodilo

Arhitektura sporočilno vodilo (*messaging bus*) je v literaturi poznana tudi pod imenom objavi in naroči (*publish and subscribe*) in informacijsko vodilo (*information bus*). Je podobna že

omenjeni arhitekturi vozlišča in priključkov, ker prav tako uporablja centraliziran tip integracije. Pomembna razlika je v obdelavi podatkov, ki v primeru arhitekture sporočilno vodilo ni centralizirana.

Integrirane aplikacije imajo vlogo bodisi prejemnik (*subscriber*) bodisi izdajatelj (*publisher*). Vsak izvor podatkov je lahko v vlogi izdajatelja, ostale aplikacije pa se lahko naročijo oz. prijavijo za prejem podatkov/sporočila. Sporočilo, v kontekstu sporočilnega vodila, je najmanjši sklop podatkov, ki se prenaša. Integracijska logika poskrbi za prenos podatkov ovitih v sporočilo izdajatelja do vseh naročenih aplikacij. Sporočila potujejo po sporočilnih kanalih. Kanali so povezave med aplikacijami in so del konfiguracije sporočilnega sistema, tvorijo informacijski cevovod. Prejemnik sporočilo sprejme in izlušči podatke, ga obdela, nato pa morebiti znova objavi oz. izda sporočilo. Pot sporočila od izdajateljev do naročnikov določa orkestracija.

Z uporabo arhitekture sporočilno vodilo dosežemo decentralizirano obdelavo podatkov. Decentralizacija poveča potrebe po administraciji, zato moramo paziti, da ostane znotraj meja obvladljivega.



Slika 15. Shema arhitekture integracije sporočilnega vodila.

Integracije na osnovi sporočil zelo podrobno pokriva [6].

3.3 Storitveno usmerjena integracija

Storitveno usmerjena integracijska arhitektura je okolje, ki poenoti raznolike aplikacije z uporabo množice standardiziranih vmesnikov. V storitveno usmerjeni arhitekturi so vmesniki običajno spletne storitve. Kvaliteta spletnih storitev je velikega pomena, ker spletne storitve predstavljajo vstopno točko za zunanje aplikacije. Spletne storitve so tako kritičen del poslovnih procesov, ki so definirani znotraj storitveno usmerjene integracijske arhitekture.

Za uspešno storitveno usmerjeno arhitekturo pa niso dovolj zgolj kvalitetne spletne storitve in sledenje že opisanim postopkom. Potrebno je upoštevati tudi strategije za [3]:

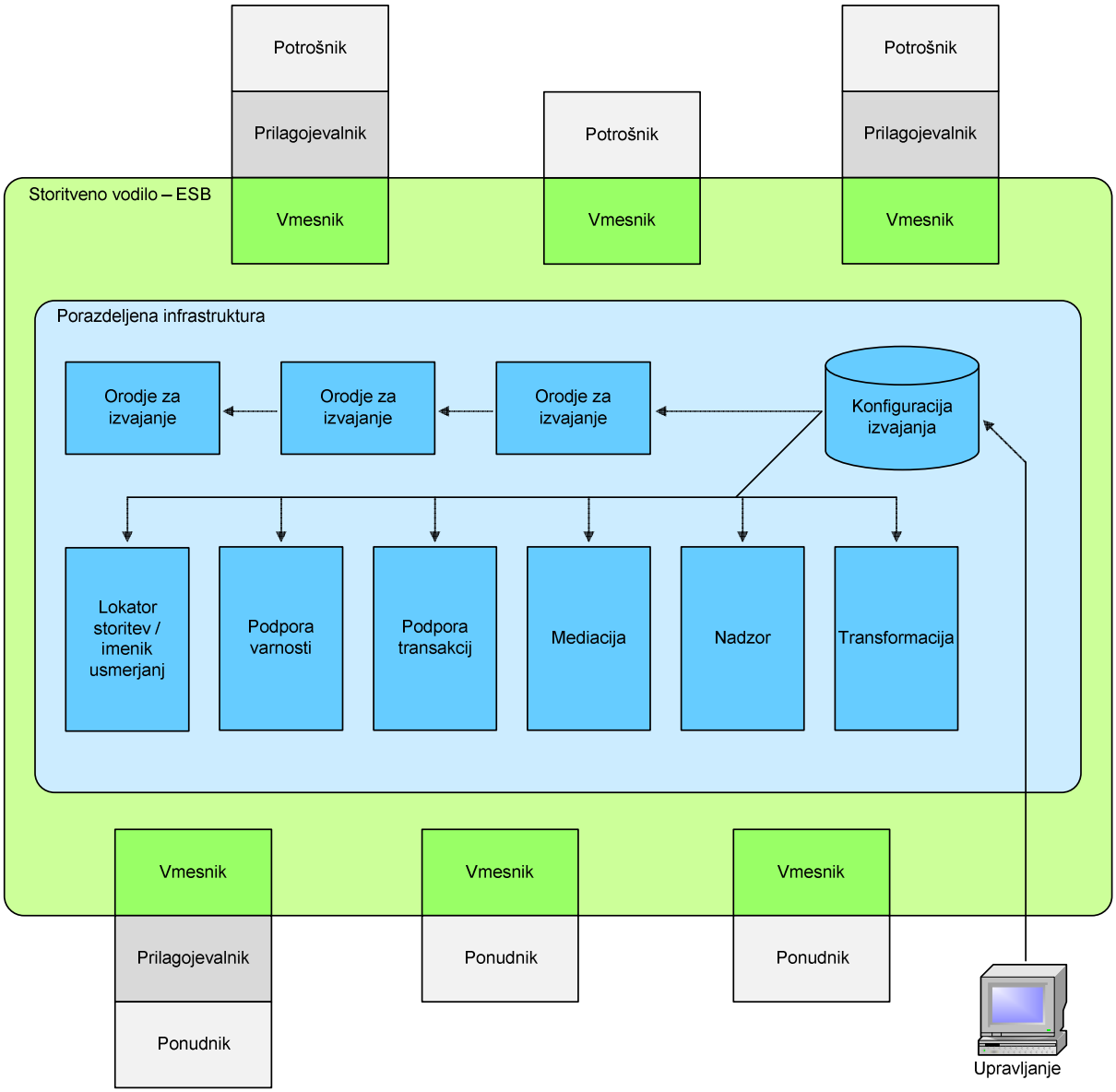
- racionalizacijo integracijskih končnih vmesnikov,
- optimizacijo končnih storitev,
- integracijo podedovanih arhitektur,
- integracijo poslovnih sistemov in
- integracijo varnosti spletnih storitev.

Storitveno usmerjena arhitektura (SOA) obljublja tudi bolj učinkovit razvoj aplikacij z več ponovno uporabljivih komponent. Zaradi večje modularnosti je lažje postopno uvajanje rešitve v okolje kakor tudi samo vzdrževanje aplikacije. Zaradi postopnega uvajanja integracijske rešitve hitreje pridemo do delnih rezultatov integracije.

3.3.1 Storitveno vodilo

Storitveno vodilo ali ESB (*Enterprise Service Bus*) je čista implementacija storitveno usmerjenih principov. Ne implementira SOA, nudi pa ustrezno okolje oz. ogrodje za implementacijo SOA. Integracijsko ogrodje je vrsta standardiziranih spletnih storitev, ki ponujajo tradicionalne funkcionalnosti orkestracije in posrednikov. Standardizirane spletne storitve zagotavljajo neodvisnost od posameznih implementacij različnih ponudnikov rešitev. Prek neodvisnosti ESB zagotavlja interoperabilnost, varnost in nadzor. Več informacij o ESB je v [7] in [1].

Čeprav mogoče na prvi pogled ni opaziti bistvene razlike med storitvenim in sporočilnim vodilom, pa je razlika vendarle pomembna. Ključna razlika je v načinu interakcije potrošnika oz. odjemalca s ponudnikom. V primeru storitvenega vodila se potrošniku ni potrebno zavedati vseh podrobnosti klica storitve, ker SOA zagotavlja izboljšan način dostopa oz. klicev storitev [8].



Slika 16. Shema arhitekture storitvenega vodila.

Osnovno arhitekturo storitvenega vodila z najbolj običajnimi komponentami prikazuje slika 16. Orodje za izvajanje (*run-time engine*) skrbi za komunikacijo med ponudniki (*providers*) in potrošniki (*consumers*). Njegova naloga je prenašanje sporočil in klici ustreznih storitev, kar je precej podobno principu objavi in naroči. Lahko gre tudi za namenska orodja za delo s poslovnimi pravili in orkestracijo procesov. Lokator storitev in imenik usmerjanj vsebuje naslove storitev in varnostno politiko za dostop do teh storitev. Poslovni procesi zahtevajo podporo transakcij, kar zagotavlja komponenta za podporo transakcij. Komponenta za mediacijo je posebno ogrodje, ki omogoča obravnavo neskladji v podatkovnih tipih in neskladij v protokolih interakcij. Nadzor v storitvenem vodilu se izvaja tako na nivoju prometa med servisi, kot tudi na nivoju poslovnih procesov.

Najpogostejše so naslednje implementacije storitvenega vodila:

- samostojno storitveno vodilo (*Stand-alone ESB*),

- storitveno vodilo kot vsebnik storitev (*ESB as a Service Container*) in
- storitveno vodilo kot ogrodje (*ESB as a Framework*).

Samostojno storitveno vodilo je arhitektura, ki je bila uporabljena v prvih implementacijah storitvenega vodila. Je podobna arhitekturi vozlišča in priključkov, le da v primeru storitvenega vodila vozlišče prevzame tudi nalogo mediatorja. Prednost takšnega pristopa je hkrati tudi slabost. Vpliv implementacije vodila je minimalen na ponudnike in potrošnike storitev integracije, po drugi strani pa smo omejeni s funkcionalnostmi vodila pri ustvarjanju storitev.

Rešitev za odpravo pomanjkljivosti je implementacija storitvenega vodila kot vsebnika storitev. Gre za razširitev funkcionalnosti aplikacijskega strežnika, ki je izvajalno okolje za storitve. S tem pa postanejo na voljo vsi mehanizmi aplikacijskega strežnika, kar omogoča izboljšano logiko vozlišča oz. vodila.

Storitveno vodilo kot ogrodje je posledica samostojnega razvoja kot alternativa nakupu rešitve. Običajno gre za ovijanje komponent na vmesni opremi z namenom približati se arhitekturi storitvenega vodila. Glavna prednost je storitvenega vodila je velika neodvisnost, saj je rešitev razvita za vmesno opremo, ki jo ponuja veliko proizvajalcev. Obenem pa lahko pomeni tudi veliko investicijo v lasten razvoj.

4 Registri

Register je uradni seznam z določenimi podatki, namenjen evidenci. Registra, ki ju obravnava diplomska naloga (CRP in RPE) spadata v okvir javne uprave. Sta informacijska sistema oz. zbirki podatkov v javni upravi. Sodobni tovrstni registri omogočajo elektronsko poslovanje z državljani in podjetji, ter učinkovito poslovanje oz. povezovanje med različnimi registri. Registri pokrivajo različna področja. Za spremljanje prebivalstva sta vzpostavljena npr. Matični register in Register stalnega prebivalstva, podatki o podjetjih se zbirajo v Poslovnem registru, podatki o nepremičninah v Zemljiški knjigi,… Dostop do nekaterih storitev naštetih registrov je mogoč le iz omrežja HKOM.

Pri implementaciji naše rešitve smo uporabljali storitve Registra prostorskih enot in storitve Centralnega registra prebivalstva, zato bom podrobneje predstavil ta dva registra.

4.1 Register prostorskih enot – RPE

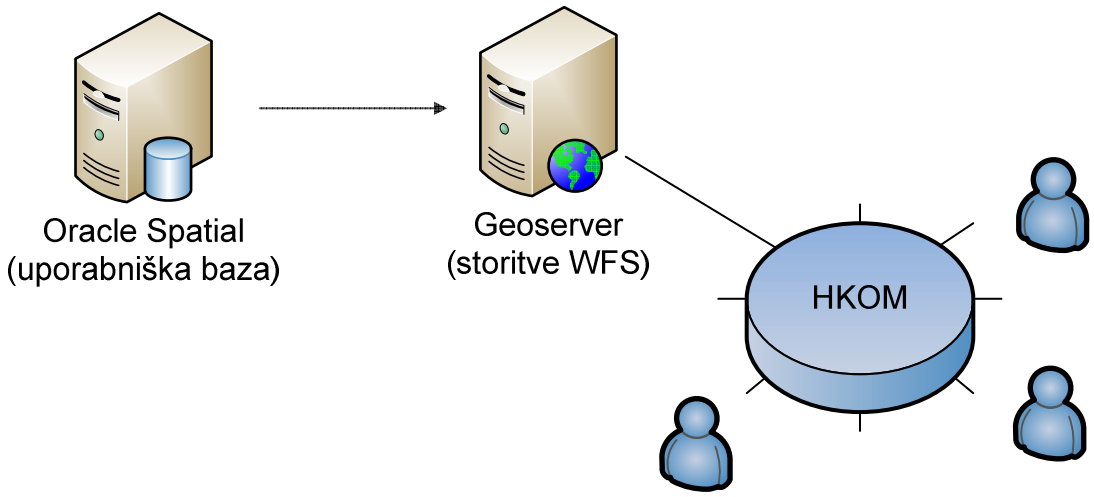
Geodetska uprava Republike Slovenije (GURS) upravlja z naslednjimi registri [9]:

- Zemljiški kataster,
- Kataster stavb,
- Zbirni kataster gospodarske javne infrastrukture in
- Register prostorski enot.

Dostop do registrov je različen. Nekateri podatki so na voljo brezplačno, drugi so plačljivi, nekatere storitve pa so na voljo le iz omrežja HKOM.

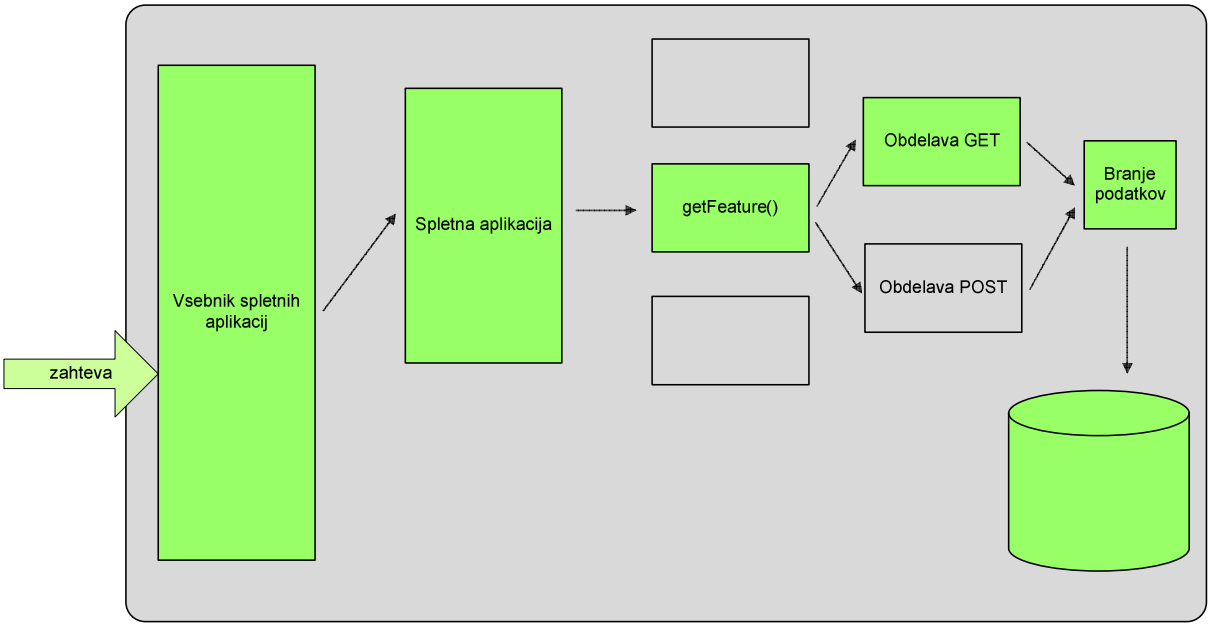
RPE [10] je bil vzpostavljen leta 1995 z združitvijo podatkov Registra območij teritorialnih enot in evidence hišnih številk. Vsebuje lokacijske in opisne podatke o osnovnih (hišna številka, statistični okoliš, občina, upravna enota, …) in dodatnih prostorski enotah (ulica, volilna enota, šolski okoliš, …). Uporabniki dostopajo do podatkov preko uporabniške baze, ki je kopija centralne baze. Kopiranje centralne baze podatkov se izvaja enkrat dnevno.

Dostop do podatkov je mogoč s spletnimi storitvami, ki so razvite v skladu *OGC OpenGIS* standardi. Za posredovanje opisnih podatkov se uporablja spletna storitev za prostorske značilnosti, ki jo opredeljuje standard *Web Feature Service* (WFS) [11]. Shema sistema je prikazana na sliki 17.



Slika 17. Shema sistema za posredovanje opisnih prostorskih podatkov.

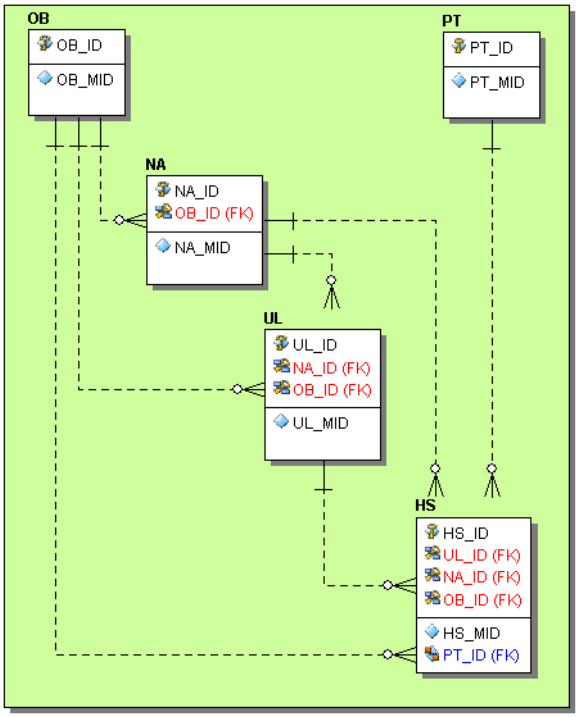
Geoserver je J2EE aplikacija, kar pomeni, da za delovanje potrebuje javanski aplikacijski strežnik. Storitev WFS je implementirana kot spletna aplikacija – *servlet*. Klic spletne aplikacije se lahko izvede na dva načina, s POST ali GET. Klic GET je primernejši za uporabo v aplikacijah, POST pa za končnega uporabnika npr. v spletnih uporabniških aplikacijah. Podatki, ki jih spletna aplikacija posreduje v odgovoru so odvisni od klicane metode. Klicana metoda določa vrsto prostorske enote in pogoje iskanja. Pogoji so združeni v filter po standardu WFS. Funkcija filtra je ekvivalentna stavkom WHERE v vprašanjih SQL, zato je za uspešno sestavo filtra potrebno poznavanje strukture podatkov. Natančna navodila za uporabo storitev GURS-a so podana v uporabniški dokumentaciji [12], več informacij o Geoserverju oz. odprto kodni implementaciji pa je na voljo na spletnih straneh [13].



Slika 18. Potek obdelave zahteve za storitev WFS.

Začetno stanje RPE je bilo vzpostavljeno na datum 01.01.1995 z združitvijo podatkov dveh registrov. To velja za vse prostorske enote, izjema so le podatki o poštnih enotah, ki so bili vzpostavljeni 15.04.2005. Pred tem so obstajali le podatki o poštnih okoliših.

Vsaka prostorska enota ima dva identifikatorja, MID in ID. MID označuje medresorski identifikator, ID pa je identifikator v RPE. Ker ID ni enolični identifikator na nivoju posamezne prostorske enote, je potrebno upoštevati tudi ID nadrejenih prostorskih enot. Ulica ima nadrejeni enoti naselje in občina. Vsi trije ID-ji (ulice, naselja in občine) skupaj enolično določajo ulico.



Slika 19. Shema delnega podatkovnega modela RPE.

Podatki o posamezni prostorski enoti se skozi čas spreminjajo in se beležijo z natančnostjo sekunde. To pomeni, da je neko stanje prostorske enote veljavno do 9:59:00 , od 10:00:00 naprej nekega dne pa velja novo, spremenjeno stanje. Spremenijo se lahko opisni podatki ali geografski podatki prostorske enote.

Spremembe se označi po naslednjem šifrantu sprememb:

- začetno stanje,
- uvedba/določitev/imenovanje,
- ukinitev,
- preštevilčenje/uvedba uličnega sistema,
- premik anotacije,
- premik centroida,
- sprememba območja,

- sprememba pripadnosti,
- sprememba opisnih podatkov in
- ostalo.

Spremembe kot so preštevilčenje/uvedba uličnega sistema in sprememba pripadnosti lahko povzroči tudi spremembo ID-ja prostorske enote oz. hierarhije prostorskih enot. V vseh ostalih primerih zadeva sprememba le atribute prostorske enote.

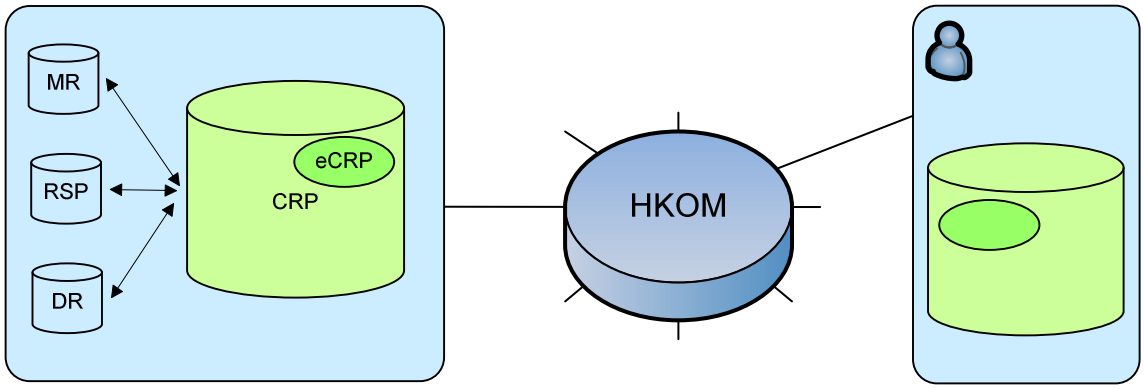
4.2 Centralni register prebivalstva – CRP

Centralni register prebivalstva je osrednja zbirka podatkov o prebivalstvu države in je kot tak pomemben člen v elektronskem poslovanju oz. e-storitvah javne uprave.

CRP združuje podatke iz naslednjih virov:

- Matični register za vodenje podatkov o rojstvih, porokah in smrtih,
- podatki o razvezah iz sodišč,
- Register stalnega prebivalstva za podatke o stalnem in začasnem prebivališču s povezavo na Register prostorskih enot in
- Register davčnih zavezancev.

Uporabniki podatkov so npr. Statistični urad RS, upravne enote, Davčna uprava RS, zdravstvo, sodstvo, … Do lastnih podatkov lahko dostopajo tudi državljani s kvalificiranim digitalnim potrdilom in tako preverijo, katere podatke o njih hrani CRP. Ostali uporabniki imajo dostop do podatkov omejen skladno z zakonsko podlago, ki jim dovoljuje vpogled.



Slika 20. Shema povezave med CRP in uporabnikom.

Zunanji uporabniki do podatkov CRP dostopajo prek storitev eCRP. Nabor podatkov, ki jih storitve posredujejo uporabnikom, je določen z zakonsko podlago in zabeležen v kontrolni tabeli pravic uporabnikov. Vsi klici storitev s strani uporabnikov se beležijo v dnevnike. Beleži se kdo, kdaj in s kakšnim namenom je klical storitev, vhodni ter izhodni podatki. Enako mero sledljivosti uporabe osebnih podatkov je potrebno zagotoviti tudi v aplikacijah, ki dostopajo do podatkov CRP na uporabnikovi strani.

Storitve eCRP so implementirane v okolju podatkovne baze Oracle v programskem jeziku PL/SQL.

Do storitev eCRP je mogoče dostopati v omrežju HKOM in z neposredno podatkovno povezavo do podatkovne baze (*Oracle database link*).

Podrobnosti o vseh storitvah in načinih izvajanja klicev storitev so podani v uporabniški dokumentaciji [14].

Pomembnejše storitve CRP so npr. vpogled v stanje osebe prek EMŠO-ja, funkcije za vzdrževanje zastavic in posredovanje sprememb.

Rezultat klica storitve za vpogled v stanje osebe prek EMŠO-ja so ažurni podatki o osebi na podlagi podanega EMŠO-ja. Neka oseba ima lahko več EMŠO-jev, aktiven pa je samo eden. Storitev preverja tudi ali uporabnik uporablja aktivni EMŠO. V primeru, da ga ne, v rezultatu vrne aktivni EMŠO, naloga uporabnika pa je, da to primerno obravnava.

Z uporabo storitve za posredovanje sprememb uporabnik CRP vzdržuje ažurno stanje v lastni evidenci. CRP dnevno pripravi paket s podatki oseb, katerim so se podatki spremenili s tem omogoči uporabniku vpogled vanje. Uporabnik s storitvami za vzdrževanje stanja zastavic označi osebe, za katere se mu bodo posredovale spremembe. Preko kontrolnih tabel uporabnik preveri pripravljenost paketa (tabela sprememb). Ko je paket pripravljen, ga uporabnik prekopira v lastno podatkovno bazo. V kontrolnih tabelah se nato paket označi kot prevzet. Ažuriranje statusov v kontrolnih tabelah se opravlja s storitvami za nadzor prenosa paketa sprememb.

Zamenjava EMŠO-ja je poseben dogodek, na katerega je potrebno biti pozoren pri implementaciji uporabe storitev. Zaradi možnosti zamenjave EMŠO-ja, je ta neprimeren za uporabo kot primarni ključ v relacijski bazi, čeprav naj bi enolično določal osebo.

Vzroki za spremembe EMŠO-ja so naslednji:

- EMŠO je bil določen na napačni datum rojstva,
- EMŠO je določen z napačnim spolom,
- uporabljata se dva EMŠO-ja in se izloči manj uporabljani.

Kako naj se obravnava zamenjava EMŠO-ja in kateri so možni scenariji je natančneje opredeljeno v uporabniški dokumentaciji storitev eCRP [14].

5 Register prebivalstva in prostorskih enot

Onkološki Inštitut (OI) v okviru svojega delovanja izvaja dejavnost vodenja registrov raka in epidemiologije. V Registru raka za Slovenijo se izvaja zbiranje in obdelava podatkov o vseh novih primerih raka (incidenci) in o preživetju bolnikov z rakom v Sloveniji. Epidemiološka služba izvaja tudi Državni presejalni program zgodnjega odkrivanja predrakavih sprememb materničnega vratu (ZORA) in Državni presejalni program za raka dojk (DORA). V nadaljnjem besedilu bom Register raka, ZORA in DORA imenoval končne uporabniške aplikacije. Za svoje delovanje končne uporabniške aplikacije potrebujejo ažurno in točno bazo podatkov o prebivalstvu Republike Slovenije.

Podatke o prebivalstvu vodi Centralni register prebivalstva (CRP). Podatkih o prebivališču v CRP se sklicujejo na Register prostorskih enot (RPE). Ker gre v CRP-ju za osebne podatke, imajo končne uporabniške aplikacije različne zakonsko določene pravice dostopa do njih. Register raka ima pravico dostopa do podatkov vseh bolnikov z rakom, DORA za ženske med 50. in 69. letom in ZORA za ženske nad 20. letom.

Namen registra prebivalstva in prostorskih enot (RPPE), ki smo ga uvedli na Onkološkem inštitutu, je zagotoviti enoten dostop do podatkov, kakor tudi povezati podatke, ki so znani posameznim končnim uporabniškim aplikacijam. Posamezna končna aplikacija ima različne zahteve oz. potrebe po podatkih RPPE-ja. Register raka potrebuje točne geografske podatke RPE-ja za natančnejše spremljanje prostorske incidence raka, presejalni programi pa potrebujejo ažurne podatke o prebivališču oseb, ki jih uporabljajo pri pripravi vabil na presejalni test.

Ker končne aplikacije uporabljajo RPPE, so presejalnim programom na voljo tudi podatki iz Registra raka, ki niso del CRP-ja. Takšen podatek je npr. obolelost za rakom. Na ta način je RPPE omogočil izločitev določenih skupin oseb iz seznama kandidatov za test (npr. že obolelih za rakom). Ker več končnih aplikacij uporablja RPPE in ne neposredno CRP in RPE, je odpravljeno tudi nepotrebno obremenjevanje omrežja, ki bi nastopilo v primeru ločenega osveževanja podatkov. Prav tako se je odpravilo večkratno delo osveževanja podatkov v vsaki aplikaciji.

5.1 Predstavitev problema

Pred uvedbo RPPE je vsaka končna aplikacija ločeno osveževala podatke s pomočjo mesečnih datotek. Mesečne datoteke so vsebovale vse spremembe v podatkih RPE in CRP. Podatke teh datotek je bilo potrebno uvoziti v podatkovno bazo končne aplikacije, najprej podatke RPE, nato še CRP. Ker imajo datoteke za izmenjavo točno določeno strukturo, je vsakršna sprememba strukture nujno pomenila tudi spremembo aplikacije za uvoz podatkov, tudi če ni šlo za podatke relevantne za registre. Aplikacije so bile razvite v različnih obdobjih in za različne potrebe po uvoženih podatkih. Ker se zaradi tega razlikujejo podatkovni modeli, se razlikujejo tudi aplikacije za uvoz. To pomeni ločeno vzdrževanje vsake aplikacije za uvoz

podatkov. Uvozi so zahtevali interakcijo uporabnika oz. skrbnika sistema in so ga po nepotrebnem obremenjevali. Kljub temu, da je šlo za enak nabor podatkov, je bilo potrebno uvoz opraviti za vsak register.

Tudi interval osveževanja za presejalna programa ni bil najprimernejši, ker je obstajala možnost, da se je v vmesnem času vabljena oseba preimenovala, preselila ali celo umrla.

Včasih se je pojavila tudi potreba po takojšnjem zajemu podatkov iz CRP. Takšen primer je npr. dodajanje osebe, ki še ni v naboru osveževanja. Pred uvedbo RPPE-ja se je v takšnih primerih osebo označilo kot začasen/neuraden vpis. Kasneje je bilo potrebno tako osebo povezati s podatki iz CRP.

Bistvene želene lastnosti pri uvedbi RPPE-ja so bile naslednje:

- avtomatično dnevno osveževanje podatkov z uporabo storitev CRP in RPE in s tem zagotavljanje ažurnosti in konsistence podatkov,
- minimalna interakcija uporabnika (le v primeru kritičnih napak oz. nepredvidenih napak, ki jih aplikacija ne more rešiti samostojno),
- zmanjšati odvisnost od sprememb strukture podatkov CRP in RPE,
- zagotoviti dostop posameznim končnim aplikacijam le do upravičenih podatkov in
- takojšen vpogled v podatke za neko osebo na podlagi EMŠO-ja.

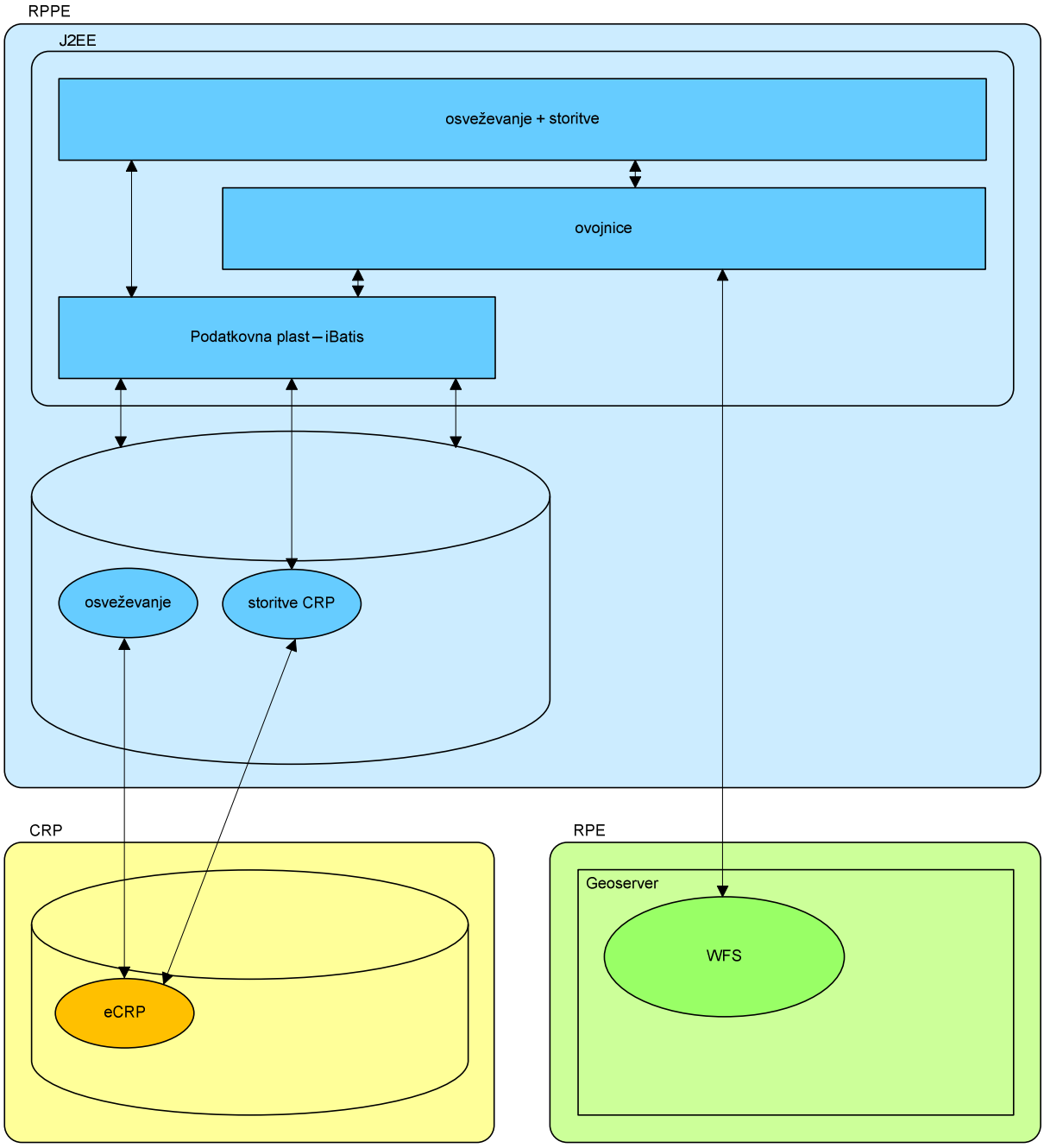
Sodelavec v ekipi je poskrbel za podatkovni model aplikacije in predlagal več-fazni postopek osveževanja podatkov. Moja naloga pa je bila implementacija replikacije podatkov z uporabo storitev RPE in CRP ter razvoj algoritma za osveževanje podatkov v podatkovnem modelu RPPE.

5.2 Implementacija osveževanja podatkov v RPPE

Integracija RPPE z registroma CRP in RPE poteka na podatkovnem nivoju in sicer z replikacijo podatkov CRP in RPE. Za replikacijo podatkov se uporabljajo storitve CRP in RPE, predstavljene prejšnjem poglavju. To je edini način za dostopanje do podatkov registrov.

Arhitekturno je replikacija podatkov integracija od točke do točke. Ker gre samo za dva registra z neposredno povezavo, je število povezav majhno in obvladljivo. Za storitve RPE sem razvil storitve ovojnice. Na ta način sem prilagodil storitve potrebam algoritma za osveževanje in ovil specifičen način klicev funkcij in parametre v ovojnico. Podobno sem naredil tudi z nekaterimi storitvami CRP.

Sama replikacija podatkov CRP je implementirana v podatkovni bazi Oracle. Takšen način je bil najprimernejši, ker se na obeh straneh uporablja enaka podatkovna baza, za povezavo pa se uporablja neposredna povezava podatkovne baze. Replikacija je, zaradi neposredne podatkovne povezave, le osvežitev materializiranega pogleda s klici kontrolnih procedur.



Slika 21. Struktura aplikacije za osveževanje podatkov in storitev RPPE.

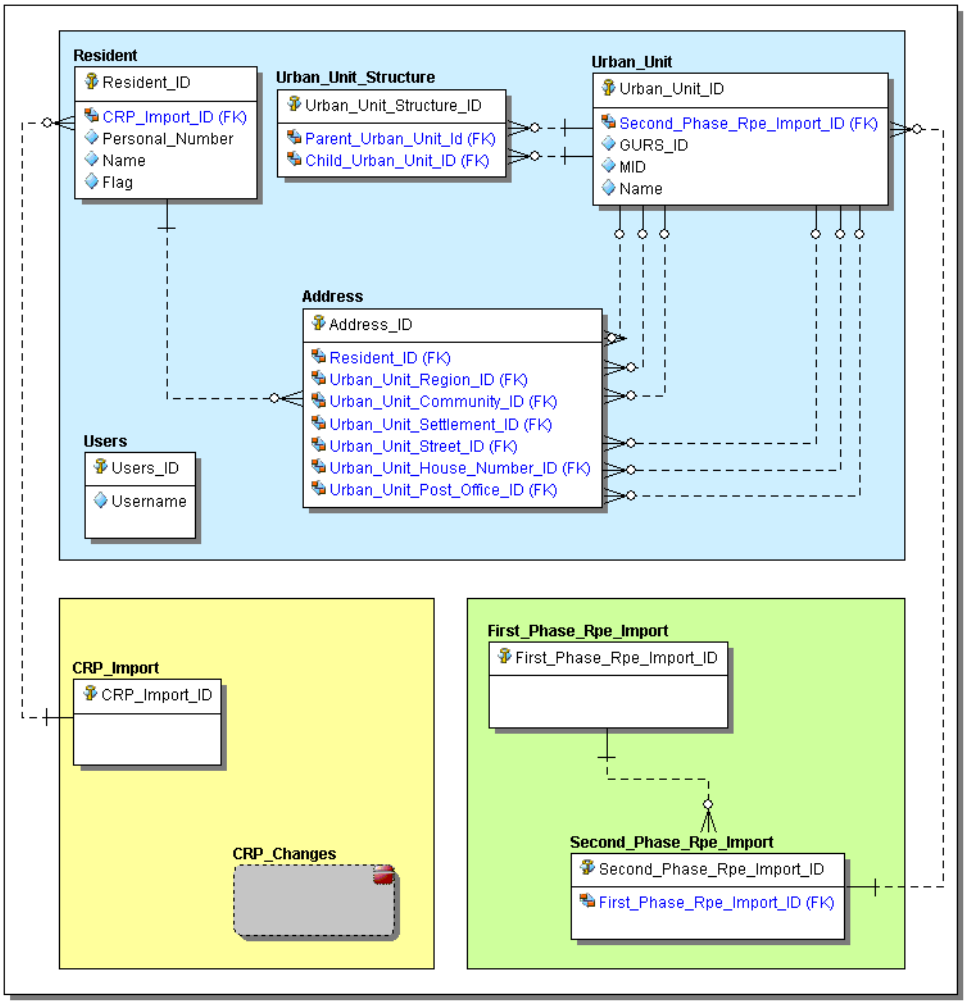
Podatkovni model RPPE je sestavljen iz treh delov:

- uvoz CRP,
- uvoz RPE in
- katalog RPPE.

Dela za uvoz sta namenjena, kot že ime pove, uvozu in hranjenju nespremenjenih podatkov iz CRP (entiteta *CRP_Import*) in RPE (entiteti *First_Phase_Rpe_Import* in *Second_Phase_Rpe_Import*). Za uvoz RPE sta potrebni dve fazi, ker je več podatkov

združenih v eno datoteko XML. V prvi fazi je potrebno podatke uvoziti, v naslednji fazi pa razčleniti, medtem ko so CRP podatki že razčlenjeni.

Ker se podatki pri pretvorbi spreminjajo, hranjenje izvornih podatkov olajša sledljivost uvoza sprememb. V katalogu RPPE so shranjeni podatki, do katerih dostopajo končne aplikacije. To so podatki o osebah (*Resident*) in naslovih (*Address*). Naslov je razdeljen na več enot, ki ustrezajo prostorskim enotam RPE. Vsaka enota je povezana na prostorsko enoto (*Urban_Unit*). Hierarhično strukturo ureja entiteta struktura prostorskih enot (*Urban_Unit_Structure*). Shemo podatkovnega modela prikazuje slika 22 (izpuščeni so nekateri atributi entitet).



Slika 22. Shema podatkovnega modela RPPE.

Vsaka od entitet v katalogu RPPE ima še svojo kopijo, v kateri se hrani zgodovina podatkov. Na ta način RPPE hrani poleg trenutno veljavnega stanja tudi stanja v preteklosti. Podatki se v nobenem primeru ne brišejo saj so stanja v preteklosti pomembna za statistične obdelave.

Implementacija osveževanja je realizirana z dvema aplikacijama, J2EE in bazno aplikacijo. Bazna aplikacija je potrebna za prenos podatkov iz CRP oziroma za klice storitev eCRP.

Same storitve eCRP zahtevajo implementacijo v programskem jeziku PL/SQL. J2EE aplikacija skrbi za prenos sprememb RPE, nato pa v zaporedju najprej ažurira v RPPE podatke RPE, nato pa še podatke CRP. Tudi samo ažuriranje RPE mora biti v zaporedju po tipih prostorskih enot, zaradi hierarhične odvisnosti le teh. Ustrezno zaporedje je naslednje: podatki o poštah, občinah, naseljih, ulicah in hišnih številkah. Del J2EE aplikacije so tudi storitve RPPE, ki jih uporabljajo končne uporabniške aplikacije.

5.2.1 Implementacija osveževanja podatkov RPE

Za register RPPE so pomembne le storitve za podatke oz. prostorske enote, katere uporablja CRP pri naslovih oseb.

Nazivi storitev so naslednji:

- UL_VS in UL_SPR za podatke o ulicah,
- PT_VS in PT_SPR za podatke o poštnih številkah in
- HS_VS in HS:SPR za podatke o hišnih številkah.

Zaradi podatkovnega modela RPE in potreb statistične obdelave podatkov, RPPE uporablja še dve storitvi:

- OB_VS in OB_SPR za podatke o občinah in
- NA_VS in NA_SPR za podatke o naseljih.

Priponi _VS in _SPR pri nazivih se nanašata na storitev, ki bodisi vrača veljavno stanje (VS), bodisi spremembe (SPR) registra RPE za posamezno prostorsko enoto v nekem obdobju. Storitev s pripono SPR torej vrača tudi stanja registra v preteklosti.

S klicem storitev WFS lahko dostopamo do podatkov o prostorskih enotah. Storitev WFS kličemo s filtrom, kot ga določa standard *OpenGIS Filter Encoding* [11]. V filtru podamo naziv storitve glede na to, katere podatke potrebujemo. Z dodajanjem kriterijev v filtru omejimo množico podatkov v odgovoru storitve.

Vsem prostorskim enotam so skupni naslednji atributi (podatkovni tip za datum in ura):

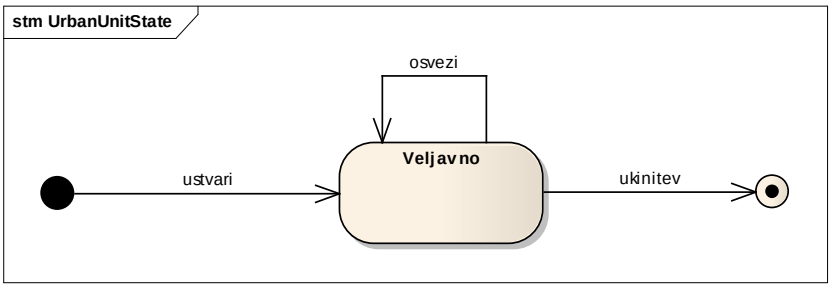
- čas_vpisa,
- čas_veljavnosti in
- vrsta_spremembe.

Ti atributi nastopajo v parih, ki označujejo stanje za začetek in konec. Filter za zajem dnevnih sprememb prostorskih enot je zasnovan tako, da uporablja atributa za čas_vpisa. Servis sprejme za vhodni parameter le datum (brez ure), kar je nerodno, saj se lahko zgodi, da je bila neka sprememba vpisana ob 00:00:00. Takšna sprememba bi bila zajeta dvakrat, zato je bilo potrebno biti pri razčlembi podatkov pozoren in zanemariti podatke ob prvem zajemu. Rezultat klica servisa je datoteka XML, ki vsebuje podatke o spremembah prostorskih enot.

V prvi fazi aplikacija za osveževanje izvrši klic storitve WFS s filtrom, ki zajame vse spremembe od dne zadnjega uspešnega uvoza do preteklega dne. Klic se izvrši za posamezno prostorsko enoto. Odgovor servisa, datoteka XML, se shrani v podatkovno bazo (tabela *First_Phase_Rpe_Import*). V tej fazi lahko pride do napake pri klicu oz. izvajanju storitve WFS, ki jo je mogoče odpraviti z zakasnjenim ponovnim klicem. Druga napaka pa se lahko dogodi pri zapisovanju podatkov v podatkovno bazo. V takšnem primeru aplikacija pošlje obvestilo o napaki in zaključi s procesom osveževanja.

V drugi fazi uvoza sprememb se prenesena datoteka XML razčleni in podatki se zapišejo v tabelo *Second_Phase_Rpe_Import*. Določijo se še povezave na šifrant za vrsto spremembe. Tabela vsebuje še referenčne atribute za vse prostorske enote, ki se nastavijo v tretji fazi uvoza in služijo sledljivosti uvoza.

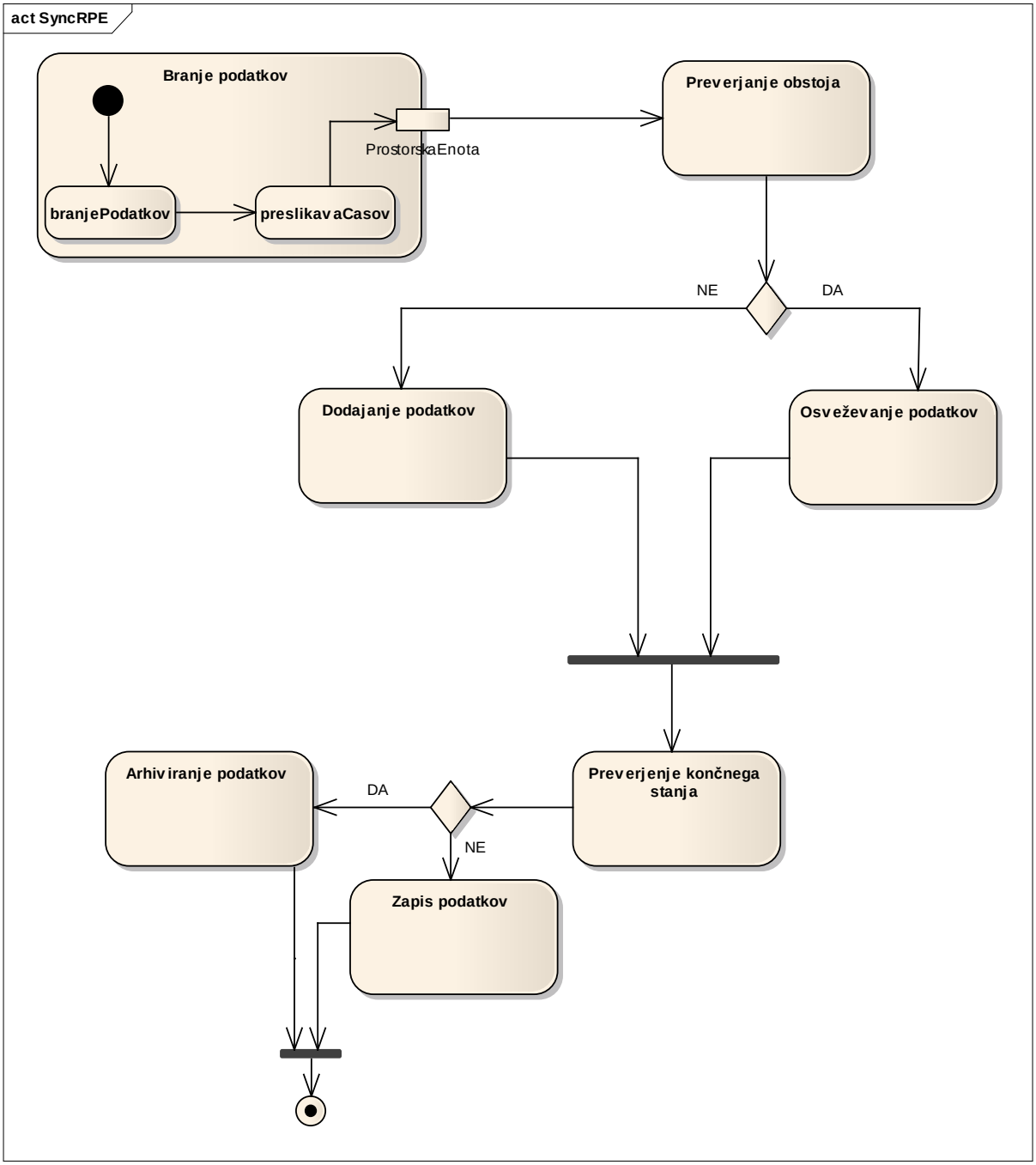
Stanje v RPPE je potrebno ažurirati glede na prenesene spremembe v podatkih prostorskih enot. Ker ima RPE večjo natančnost sledenja spremembam (možnih je tudi več sprememb v enem dnevu), kot je potrebna za RPPE, je bilo potrebno narediti funkcijo, ki preslika stanje v RPE v stanje RPPE. Stanje posamezne prostorske enote prikazuje slika 23.



Slika 23. Diagram prehajanja stanj prostorske enote v RPPE.

Prostorska enota ima tri stanja, začetno (pred uvedbo), veljavno in končno (arhivsko). V veljavno stanje prostorska enota preide od ustanovitvi oz. je določeno v inicialnem stanju. Prehod osveži vse spremembe v podatkih prostorske enote, tudi hierarhične. Prehod ukinitev iz stanja veljavno ne pomeni izbrisa iz tabele, ampak le nastavitev atributa za prenehanje veljavnosti. Na ta način je mogoče iz zgodovine podatkov pridobiti katerokoli stanje v preteklosti.

Algoritem za usklajevanje stanja je prikazan na diagramu aktivnosti na sliki 24. V aktivnosti *Branje podatkov* se pripravi objekt prostorske enote in izvede preslikava časovnih atributov. Glede na obstoj prostorske enote v RPPE se določi ali gre za osveževanje ali za ustvarjanje nove. Ali prostorska enota obstaja se preveri tako, da se poišče v katalogu RPPE veljavna prostorska enota, ki ima enak MID. Ker se spremembe na prostorskih enotah označujejo s pari za začetek in konec, je potrebno preveriti (aktivnost *Preverjanje končnega stanja*) ali je potrebno upoštevati informacijo o končnem stanju spremembe. Takšne sprememba je *ukinitev*. V drugih primerih se končna sprememba zanemari, ker osveževanje podatkov poskrbi za pravilno obravnavo sprememb v tabeli zgodovine.



Slika 24. Diagram aktivnosti uvoza sprememb RPE.

Spremembe podatkov RPE so zabeležene s sekundno natančnostjo, medtem ko RPPE uporablja dnevno natančnost. Zaradi različne frekvence beleženja sprememb je potrebno časovne atribute preoblikovati.

Pri časovni preslikavi so bila upoštevana naslednja pravila:

- Za atribut čas_veljavnost_od počisti podatke o uri.
- Če ima atribut čas_veljavnost_do podatke o uri različne od 23:59:59, potem odštej en dan.

- Če je čas_veljavnosti_do manjši od čas_veljavnosti_od potem je čas_veljavnosti_do enak čas_veljavnosti_od.

Ta pravila omogočajo tudi, da se dnevne spremembe samo prepisujejo, ne dodajajo pa se novi zapisi v tabelo zgodovine.

5.2.2 Implementacija osveževanja podatkov CRP

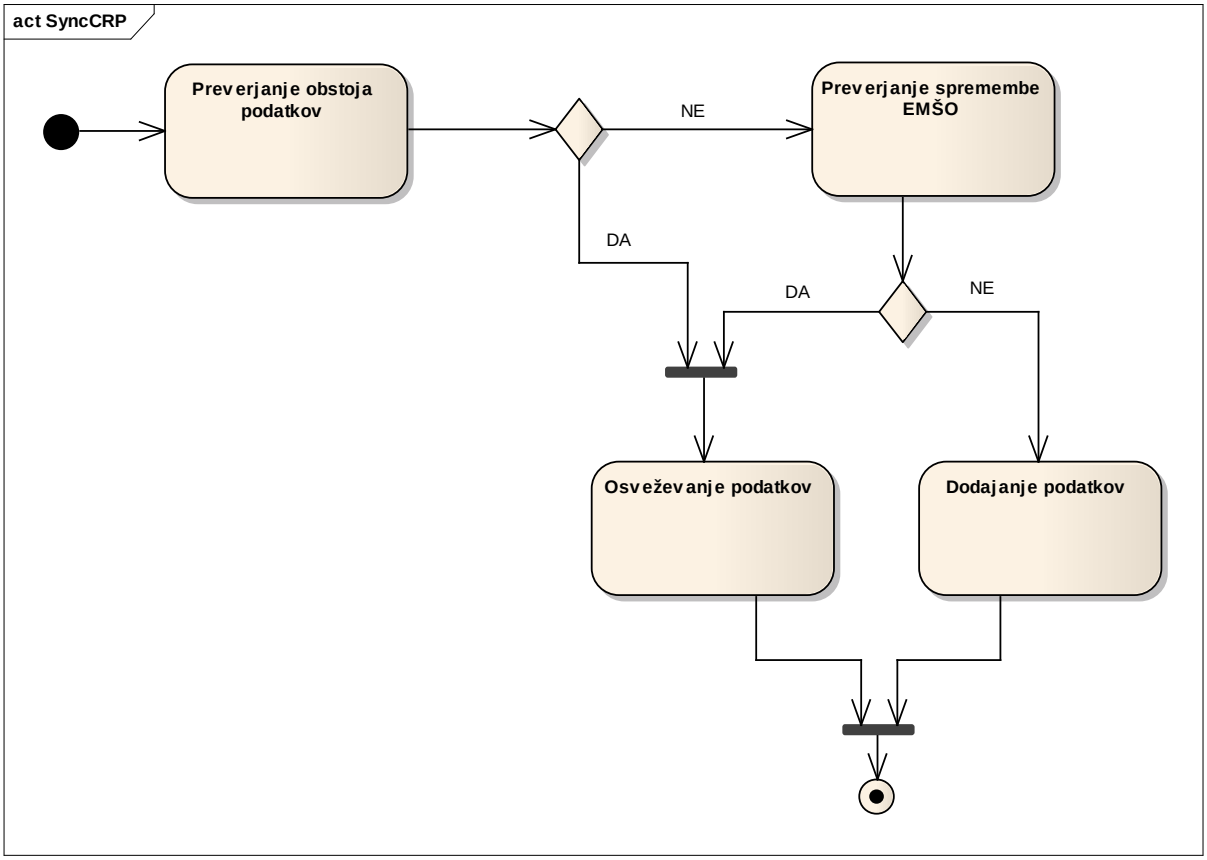
Osveževanje podatkov CRP poteka paketno. Spremembe v podatkih se zbirajo v posebni tabeli. Tabela zastavic določa za katere osebe se zbirajo podatki o spremembah. Tabela zastavic se vzdržuje z namenskimi funkcijami. Če je v nekem dnevu več sprememb za določeno osebo, se te posredujejo skupaj v enem zapisu. Ob koncu dneva se podatki pripravijo in paket se označi kot pripravljen za prenos.

V podatkovni bazi RPPE je definirano opravilo, ki izvede klic bazne procedure vsak dan ob 01:00. Bazna procedura je razvita v programskem jeziku PL/SQL. Njena naloga je osvežitev podatkov v materializiranem pogledu ter kopiranje podatkov o spremembah v lokalno kopijo sprememb. Ustrezno mora klicati tudi storitve eCRP za kontrolo prenosa podatkov. Končen rezultat prenosa paketa sprememb so podatki v tabeli *CRP_Import*. V paketu sprememb je za posamezno osebo navedeno le zadnje stanje, za kar poskrbi že CRP. To poenostavi ažuriranje RPPE, ker ni potrebno preverjanje in transformiranje atributov o časovni veljavnosti sprememb, kot je to potrebno pri uvozu RPE sprememb.

Pogoj za ažuriranje RPPE s spremembami CRP je uspešno zaključeno ažuriranje podatkov RPE. J2EE aplikacija skrbi za to, da se ažuriranje začne šele po prenesenem paketu dnevnih sprememb CRP-a in uspešno zaključeni tretji fazi uvoza sprememb RPE, ki zagotavlja uspešen uvoz sprememb CRP.

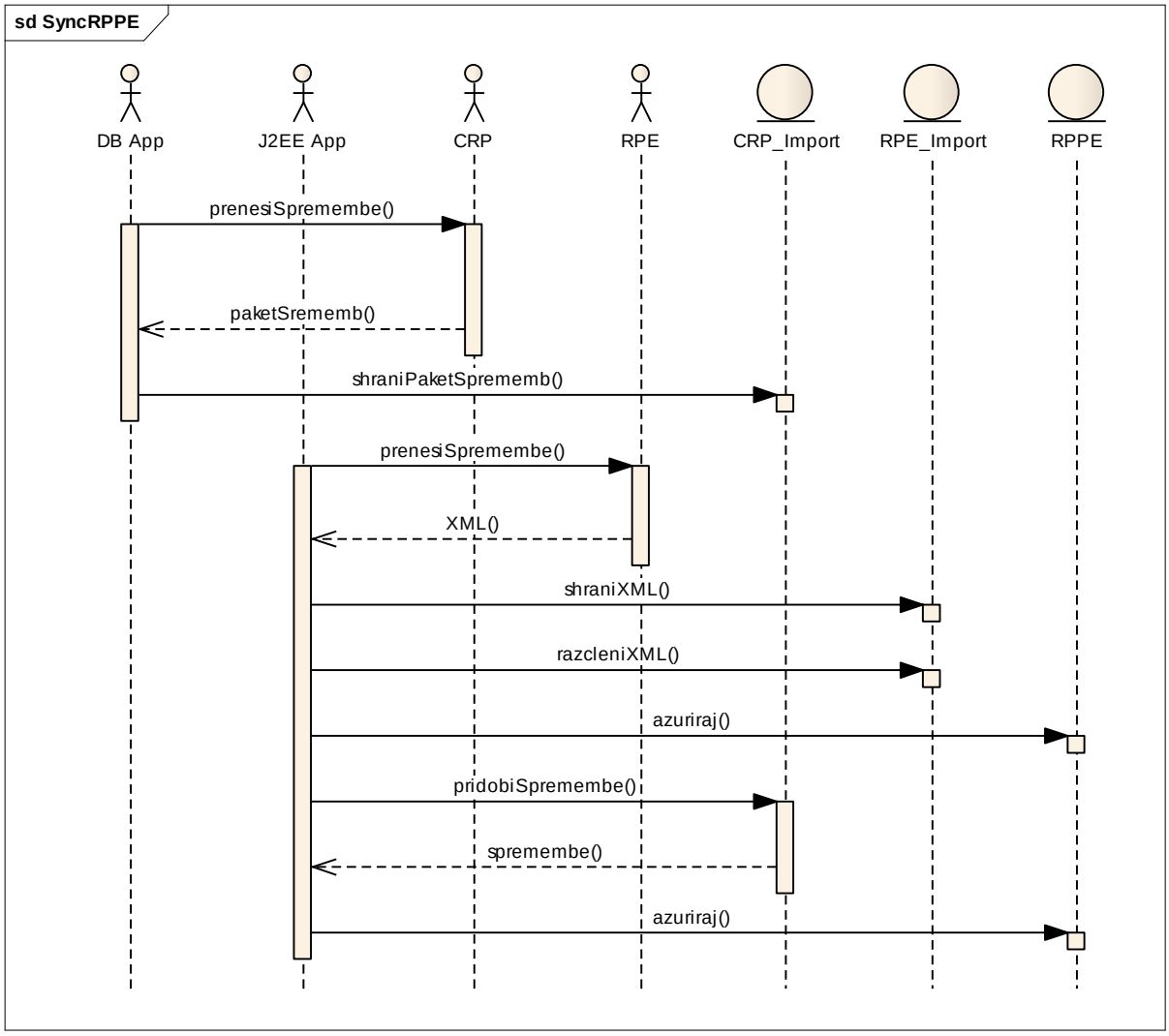
V tabeli *CRP_Import* so združeni vsi opisni podatki in podatki o naslovih. Te podatke je potrebno primerjati s podatki o osebi v tabeli *Resident* in podatki o naslovih v tabeli *Address*. Če je prišlo do spremembe, je potrebno ažurirati stanje tudi v tabeli zgodovine. Tako je zagotovljena časovna konsistentnost podatkov.

Posebno pozornost in obravnavo pri ažuriranju stanje RPPE je bila namenjena spremembi EMŠO-ja. Postopki obravnave sprememb EMŠO-ja so natančno definirani v uporabniški dokumentaciji storitev CRP [14]. Shema algoritma ažuriranja je prikazana na sliki 25.



Slika 25. Diagram aktivnosti uvoza sprememb CRP.

Na sliki 26 je prikaz postopka celotnega dnevnega osveževanja stanja RPPE.



Slika 26. Sekvenčni diagram uvoza sprememb.

5.2.3 Storitve RPPE

Poleg vzdrževanja aktualnega stanja v podatkovni bazi RPPE je naloga registra RPPE omogočiti tudi dostop do podatkov končnim uporabniškim aplikacijam.

Trenutno so razvite naslednje storitve:

- iskanje osebe,
- vpogled v podatke na podlagi EMŠO-ja,
- iskanje ulice,
- iskanje hišne številke in
- iskanje pošte.

Storitev za iskanje osebe omogoča iskanje na podlagi naziva osebe in datuma rojstva ali EMŠO-ja. Rezultat klica so podatki o osebi in naslovih prebivališča. Vedno pa je rezultat natanko ena oseba. Če je zadetkov v podatkih več, storitev povzroči izjemo - indikator več zadetkov. Na ta način je onemogočeno preiskovanje osebnih podatkov.

Vpogled v podatke z EMŠO-jem je storitev ovojnice storitve eCRP. Storitev ovojnice preveri obstoj EMŠO-ja v tabeli zastavic. Če v tabeli zastavic EMŠO manjka, storitev poskrbi za dodajanje EMŠO-ja v tabelo zastavic. Nato se izvede klic storitve eCRP za vpogled v podatke. V primeru uspešnega klica so na voljo podatki o osebi, ki se dodajo v RPPE. Storitev ovojnice poskrbi tudi za sledljivost klica storitve eCRP za vpogled, kar je tudi zahteva CRP.

Storitve za iskanje pošte, ulice in hišne številke omogočajo, v primeru nepopolnih osebnih podatkov, vpis veljavnega naslova k začasnim podatkom osebe. To je včasih potrebno, da se omogoči nadaljnje delo, dokler se ne ugotovi točna identiteta osebe.

Storitve so implementirane v okviru J2EE aplikacije in sicer kot lokalna javanska zrna (*local EJB*). Dostop do storitev je možen le tistim aplikacijam, ki se izvajajo na istem aplikacijskem strežniku.

5.2.4 Varovanje podatkov

RPPE vsebuje osebne podatke, ki jih prenaša iz CRP. Različne končne aplikacije imajo različen zakonsko določen vpogled v te podatke. Ker RPPE združuje te podatke, je bilo potrebno zagotoviti, da nobena od končnih aplikacij po uvedbi RPPE nima večjega nabora dostopa do osebnih podatkov oz. omejiti dostop do podatkov. Poleg dostopnosti do podatkov, je potrebno zagotoviti tudi sledenje dostopa do podatkov. Beleženje dostopov zahteva Zakon o varovanju osebnih podatkov. Rešuje ga vsaka končna aplikacija posebej, zato sledenje dostopa do osebnih podatkov ni del RPPE.

Pri implementaciji varovanja podatkov smo se tako osredotočili le na omejevanje dostopa. Vsaka od končnih aplikacij uporablja drugačno uporabniško ime za prijavo v podatkovno bazo. Vsi uporabniki podatkovne baze imajo dovoljenje za izvajanje poizvedb SQL tipa beri na shemi RPPE. Ta dovoljenja so vezana na celotno tabelo in ne omogočajo omejevanja dostopa do posameznih vrstic v tabeli.

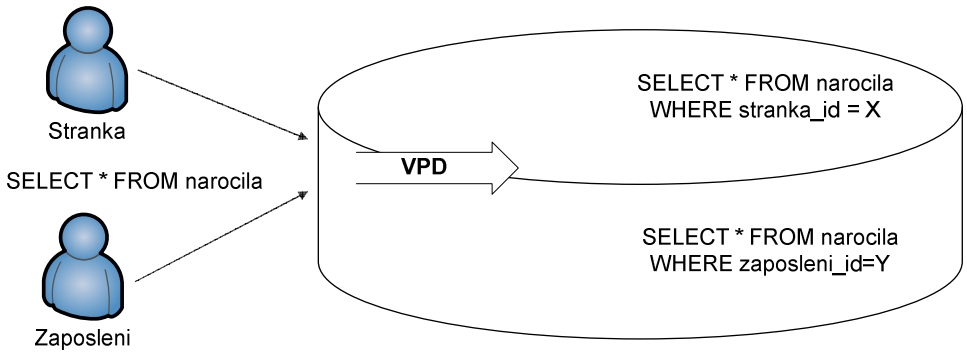
Ena izmed možnosti za zagotovitev omejitve dostopa je, da aplikacija sama filtrira podatke, vse pa beleži v dnevnik. Ta rešitev ni najprimernejša, ker pomeni zahtevnejše vzdrževanje aplikacij, možne pa so tudi napake. Mogoče je tudi implementirati poglede nad tabelami in urediti pravice dostopa do teh baznih objektov. Problem pogledov je, da jih za zadovoljitev varnostnih zahtev potrebujemo veliko, uporabniki pa lahko še vedno morebiti dostopajo do tabel. Boljša rešitev je navidezna zasebna podatkovna baza (VPD – *Virtual Private Database*) [15], ki nadzor dostopov do posameznih vrstic v tabeli prenese na podatkovno bazo.

VPD je del Oracle podatkovnega strežnika in uporablja pristop varovanja dostopa do vrstice (*Row Level Security* – RLS) in aplikacijski kontekst. VPD dinamično spreminja poizvedbe SQL uporabnika na naslednji način:

1. Strežnik prebere aplikacijski kontekst uporabnika, ki se nastavi ob prijavi.
2. Kliče se funkcija PL/SQL, ki jo določa politika dostopa.
3. Funkcija vrne predikat, ki kvalificira določen nabor podatkov.

4. Uporabnikovo poizvedba SQL se modificira s predikatom in izvrši.

Shema delovanja je prikazana tudi na sliki 27.



Slika 27. Prikaz delovanja VPD. Različna uporabnika izvršita enako poizvedbo SQL, vendar VPD dinamično doda pogoj k poizvedbi, kar se odraža v različnih rezultatih poizvedbe.

Podatkovni strežnik vsebuje nabor vnaprej definiranih atributov aplikacijskega konteksta. Eden izmed atributov je tudi SESSION_USER, ki določa uporabnika.

Prednosti VPD so:

- dinamična varnost,
- posameznim objektom v podatkovni bazi lahko določimo več politik,
- uporabnik ne more zaobiti varnostne politike, ker za to skrbi strežnik in
- enostavno spreminjanje pravil, brez posega v uporabniško aplikacijo.

Problem pri uporabi VPD se lahko pojavi pri pisanju dnevnika na nivoju podatkovne baze oz. pri pisanju funkcij za vodenje dnevnika. Ker aplikacije, ki uporabljajo RPPE, same skrbijo za dnevnik, se ta problem v našem primeru ni pojavil.

Sama implementacija VPD oz. funkcije PL/SQL z nazivom *rppePermission* za potrebe RPPE je zahtevala, da se poveže vrednost atributa aplikacijskega konteksta SESSION_USER s podatki, do katerih ima uporabnik/aplikacija dostop. To smo rešili z razširitvijo tabele osebnih podatkov z novim stolpcem zastavice. Vrednost stolpca zastavice določa katere aplikacije imajo dostop do te vrstice.

6 Sklepne ugotovitve

Z uspešno integracijo s CRP in RPE smo dosegli poenostavitev postopka osveževanja podatkov v RPPE. Z avtomatizacijo postopka, smo dosegli višjo frekvenco osveževanja podatkov in zmanjšali vpliv človeškega faktorja pri postopku. Poleg razbremenitve skrbnika aplikacije, je to pomembno zaradi zagotavljanja ažurnosti in varovanja osebnih podatkov v RPPE. Zaradi dnevnega osveževanja podatkov, so ti vedno aktualni. Ker ima RPPE centralno arhitekturo oz. povezovalno vlogo tudi med aplikacijami znotraj Onkološkega Inštituta, se na ta način podatki o osebah posameznih končnih uporabniških aplikacij delijo in ni potrebe po ponavljanju postopka osveževanja podatkov v vsaki uporabniški aplikaciji.

S pregledom arhitektur integracij sem dobil vpogled v prednosti in slabosti posamezne arhitekture integracije. Na podlagi analize ključnih elementov vsake integracije, sem se odločil, da bom implementiral povezavo s CRP in RPE neposredno s storitvijo ovojnice. Na ta način sem lahko prilagodil klice storitev potrebam RPPE oz. logiki za osveževanje podatkov v RPPE. Prednost storitve ovojnice je tudi, da jo lahko logika za osveževanje podatkov uporablja, četudi se bodo morda storitve CRP ali RPE v prihodnosti spremenile. Z vzpostavitvijo RPPE smo pridobili tudi možnost razvoja novih funkcij, ki jih CRP in RPE ne omogočata in so prilagojene potrebam uporabnikov ter jim zagotavljajo večjo učinkovitost dela.

Glavna pomanjkljivost arhitekture integracije RPPE izhaja iz temeljne pomanjkljivosti, ki jo prinaša centralizirano usmerjena arhitektura. Centralni del predstavlja kritično točko odpovedi. Po naši oceni je takšen način vseeno boljši, kot če bi uporabljali neposredno storitve CRP in RPE, saj z replikacijo podatkov izločimo vpliv možnosti izpada povezave do zunanjih sistemov. Slabost predstavlja tudi dnevno osveževanje Izvaja se na zahtevo po podatkih za pretekli dan, zaradi česar se lahko zgodi npr., da podatki o spremembah še niso prekopirani v uporabniško bazo in postopek osveževanja teh podatkov ne zajame.

Kljub temu, da smo se trudili izločiti vpliv človeškega faktorja, le ta še vedno ostaja v vhodnih podatkih v RPPE. Podatke v CRP in RPE vnašajo ljudje in lahko pride do napake pri vnosu. Posledica take napake je, da iskanega podatka uporabniki ne bodo našli oz. algoritem osveževanja podatke osveži na nepredviden način. Na človeški faktor ne moremo vplivati, napake je težko zaznati, povzročijo pa lahko nezaželeno nekonsistentnost podatkov.

Pri implementaciji storitev RPPE sem se opiral na idejo po čim večji ponovni uporabljivosti oz. čim lažji vključitvi storitev v moderne arhitekture kot je SOA. Zato sem storitve implementiral v obliki Java zrn (*Enterprise Java Beans*).

Integracija RPPE s CRP in RPE je integracija na prvem nivoju, se pravi podatkovnem nivoju. Z uvajanjem elektronskega poslovanja v javno upravo je pričakovati večje število storitev, med njimi tudi npr. storitev CRP.

Delo na RPPE se lahko nadaljuje na izboljšani integracijami z ostalimi aplikacijami na Onkološkem Inštitutu. Trenutno se z aplikacijami povezuje na drugem nivoju. Precej operacij, ki se tičejo dela s podatki iz RPPE, je v aplikacijah podobnih, zato bi bilo smiselno to logiko vključiti v RPPE in implementirati kot spletne storitve. Kasneje pa bi lahko storitve povezali in predstavili kot proces, kar bi pomenilo, da bi se približali zaključku integracije tudi po nivojih.

Literatura

[1] **Jurič, Matjaz B., in drugi.** *SOA Approach to Integration.* s.l. : Packt Publishing Ltd., 2007. ISBN 978-1-904811-17-6.

[2] **Jurič, Matjaž B., in drugi.** *Professional J2EE EAI.* Birmingham : Wrox Press, 2001. ISBN 1-861005-44-X.

[3] **Erl, Thomas.** *Service Oriented Architecture: Field Guide to Integrating XML and Web Services.* New Jersey : Prentice Hall, 2004. ISBN 0-1 3-142898-5.

[4] OASIS. *The WS-Coordination specification.* 16. april 2007. Dostopno na: http://docs.oasis-open.org/ws-tx/wstx-wscoor-1.1-spec-os.pdf.

[5] OASIS. *Web Services Transaction.* 12. julij 2007. Dostopno na: http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=ws-tx.

[6] **Hohpe, Gregor in Woolf, Bobby.** *Enterprise integration patterns.* s.l. : Addison Wesley, 2003. ISBN 0-321-20068-3.

[7] **Rosen, Mike, in drugi.** *Applied SOA: Service-Oriented Architecture and Design Strategies.* s.l. : Wiley Publishing, 2008. ISBN 978-0-470-22365-9.

[8] **Binildas, C. A.** *Service Oriented Java Business Integration.* s.l. : Packt Publishing Ltd., 2008. ISBN 978-1-847194-40-4.

[9] Postor. [Elektronski] http://prostor.gov.si/.

[10] Register prostorskih enot. 2009. Dostopno na: http://prostor.gov.si/vstop/sistem_zbirk_prostorskih_podatkov/nepremicnine/register_prostorskih_enot/.

[11] OGC - Web Feature Service. *Open Geospatial Consortium.* 2009. Dostopno na: http://www.opengeospatial.org/standards/is.

[12] **GURS.** Posredovanje atributnih in grafičnih podatkov preko spletnih storitev. s.l. : interna dokumentacija, 2008.

[13] GeoServer. 2009. Dostopno na: http://geoserver.org.

[14] **MJU.** Povezovanje CRP z velikim uporabnikom. s.l. : interna dokumentacija, 2008.

[15] Oracle Database Security Guide 11g. December 2008. Dostopno na: http://download.oracle.com/docs/cd/B28359_01/network.111/b28531.pdf, str.7-1.