Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Matic Bizjak

INTEGRACIJA UPRAVLJANJA POSLOVNIH PROCESOV Z OKOLJEM ZA RAZVOJ APLIKACIJ

diplomsko delo

interdisciplinarni univerzitetni
študijski program prve stopnje
računalništvo in matematika

Mentor: izr. prof. dr. Mojca Ciglarič

Ljubljana, 2017

Besedilo je oblikovano z urejevalnikom besedil LaTeX.

Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Tematika naloge:

Preučite in povzemite osnovne pojme s področja upravljanja poslovnih procesov in sistemov za podporo le temu. Glede na vsebino poslovnega okolja identificirajte najpogostejše možnosti integracije sistemov za upravljanje poslovnih procesov (BPMS) z drugimi sistemi. Opišite probleme pri uporabi BPMS v podjetju ter prednosti, ki bi jih v konkretnem okolju prinesla integracija BPMS z okoljem za razvoj aplikacij. Izberite primeren arhitekturni model ter izdelajte rešitev, ki bo omogočala takšno integracijo za BPMS Ultimus. Rešitev ustrezno ovrednotite.

Najprej bi se zahvalil izr. prof. dr. Mojci Ciglarič za mentorstvo in vse nasvete pri izdelavi diplomske naloge.

Hvala tudi Mateju T., Urošu N. in Mateju B. ter ostalim sodelavcem v podjetju Crea, ki so mi pomagali pri diplomski nalogi in mi predali veliko novega znanja.

Največja zahvala gre moji družini, ki mi je omogočila študij in mi ves čas stala ob strani.

Na koncu pa bi se rad zahvalil še Nadji, ki je ves čas verjela vame in me spodbujala.

Kazalo

Povzetek

Abstract

1 Uvod **1**

2 Zgodovina poslovnih procesov in njihovo upravljanje **3**

2.1 Poslovni proces 3

2.2 Od funkcijske k procesni organiziranosti 4

2.3 Upravljanje poslovnih procesov 5

2.4 Osnovne faze upravljanja poslovnih procesov 8

2.5 Sistemi za upravljanje poslovnih procesov 9

2.6 Možnosti integracije sistemov za upravljanje poslovnih procesov 11

3 Začetno stanje in opis problema **13**

3.1 Kaj ponuja Ultimus BPM Suite? 13

3.2 Opis problema 15

4 Arhitektura, osredotočena na vire **19**

4.1 REST arhitekturni stil 19

4.2 Arhitektura, osredotočena na vire 21

4.3 Utemeljitev izbrane arhitekture 23

5 Uporaba arhitekture, osredotočene na vire, na poslovnih procesih **25**

5.1 Integracija preko spletnih storitev 25
5.2 Koncepti poslovnega procesa 26
5.3 Izzivi 27
5.4 Pregled obstoječih RESTful spletnih storitev 27

6 Načrt in izdelava rešitve **31**
6.1 Operacije Ultimusovega aplikacijskega programskega vmesnika 31
6.2 Viri in vzorci enoličnih naslovov vira 33
6.3 Podprte metode na virih 34
6.4 Opis procesa 34
6.5 Pomen HTTP metod 36
6.6 Izzivi pri izdelavi rešitve 37
6.7 Ovrednotenje rešitve na podlagi arhitekture, osredotočene na vire 39

7 Sklepne ugotovitve **43**
7.1 Možne nadgradnje 44

Literatura **45**

Slike

2.1 Shema poslovnega procesa. 4
2.2 Osnovne faze BPM. 9
2.3 Identificirane možnosti integracije sistemov za upravljanje poslovnih procesov. 12

3.1 UML komponentna shema, ki prikazuje trenutni način komuniciranja spletne aplikacije (A) z izvajalnim okoljem (S). . . . 16
3.2 UML komponentna shema, ki prikazuje komunikacijo aplikacije (A) in izvajalnega okolja (S) preko RESTful spletne storitve (R). 17

6.1 Shema procesa nabave v Ultimus BPM Studiu. 35
6.2 Seznam predodelitev za določen korak. Podatki so v formatu JSON. 38
6.3 Predstavitev vira primerka procesa in enolični naslovi vira bližnjih virov. Podatki so v formatu JSON. 38
6.4 Primer varnega načina komunikacije z RESTful spletno storitvijo preko HTTPS. Podatki so v formatu XML. 39

Tabele

2.1 Razlike med funkcijsko in procesno organiziranostjo podjetja. Povzeto po [11]. 6

5.1 Uspešnost RESTful spletnih storitev pri upoštevanju lastnosti arhitekture, osredotočene na vire. 29

6.1 Podprte HTTP metode na posameznih virih in oblika njihovih relativnih enoličnih naslovov vira. 35

6.2 Primerjava izdelane RESTful spletne storitve in prej omenjenih RESTful spletnih storitev na podlagi upoštevanja lastnosti arhitekture, osredotočene na vire. 41

Seznam uporabljenih kratic

kratica	angleško	slovensko
BPM	business process management	upravljanje poslovnih procesov
WfMS	workflow management system	sistem za upravljanje delovnega toka
BPMS	business process management system	sistem za upravljanje poslovnih procesov
API	application programming interface	aplikacijski programski vmesnik
GUI	graphical user interface	grafični uporabniški vmesnik
BPMN	business process model and notation	standard za modeliranje poslovnih procesov
HTTP	hypertext transfer protocol	protokol za prenos hiperteksta
HTTPS	hypertext transfer protocol secure	protokol za varni prenos hiperteksta
REST	representational state transfer	predstavitveni prenos stanja
ROA	resource oriented architecture	arhitektura, osredotočena na vire
UML	unified modeling language	poenoten jezik za modeliranje
JSON	javaScript object notation	preprost format za izmenjavo podatkov
XML	extensible markup language	razširljiv označevalni jezik

Povzetek

Naslov: INTEGRACIJA UPRAVLJANJA POSLOVNIH PROCESOV Z OKOLJEM ZA RAZVOJ APLIKACIJ

Avtor: Matic Bizjak

Povzetek: V diplomi je opisano, kaj je to poslovni proces, upravljanje poslovih procesov in sistem za upravljanje poslovnih procesov. Raziskane so možnosti integracije z zunanjimi sistemi in predstavljena je arhitektura, osredotočena na vire, ki je uporabljena za razvoj rešitve. Cilj diplomske naloge je na podlagi pridobljenega znanja razviti RESTful spletno storitev, ki izpostavi in doda nove funkcionalnosti k operacijam aplikacijskega programskega vmesnika, uporabljenega za integracijo sistema za upravljanje poslovnih procesov z okoljem za razvoj aplikacij. Rešitev je na koncu primerjana z že obstoječimi RESTful spletnimi storitvami drugih sistemov za upravljanje poslovnih procesov na podlagi arhitekture, osredotočene na vire.

Ključne besede: poslovni proces, BPMS, ROA, RESTful spletna storitev.

Abstract

Title: BUSINESS PROCESSES MANAGEMENT INTEGRATION WITH APPLICATION DEVELOPMENT ENVIRONMENT

Author: Matic Bizjak

Abstract: Bachelor's thesis describes business process, business process management, business process management systems and ways to integrate them into existing applications. Resource oriented architecture is presented and used to develop the solution. The main purpose of this work is to design and develop RESTful web service which exposes and adds new functionalities to application programming interface, which is used to integrate business process management system with software development framework. At the end the solution is compared with existing RESTful web services from different business process management systems in terms of resource oriented architecture.

Keywords: business process, BPMS, ROA, RESTful web service.

Poglavje 1

Uvod

V današnjem času smo priča vedno večji procesni organiziranosti podjetij z namenom povečanja produktivnosti. Za prepoznavanje in izboljševanje procesov podjetja se je uveljavil pristop upravljanja poslovnih procesov. Ta je v večini primerov podprt s sistemom za upravljanje poslovnih procesov. Danes je na trgu veliko različnih ponudnikov teh sistemov, ki nudijo različne možnosti povezovanja z zunanjimi sistemi.

V podjetju Crea izdelujemo spletne aplikacije s podporo sistemov za upravljanje poslovnih procesov. Med drugim že dolgo časa uporabljamo Ultimusov sistem in imamo z njim veliko izkušenj.

Za povezavo tega sistema z našimi aplikacijami uporabljamo aplikacijski programski vmesnik, ki ga ponuja Ultimus. Problem uporabe tega vmesnika je ta, da se lahko uporablja samo na strežniku, kjer je nameščeno izvajalno okolje.

Zaradi povečanja števila uporabnikov in procesov na naših spletnih aplikacijah smo se odločili ločiti strežnik z izvajalnim okoljem od ostalih spletnih aplikacij z namenom razbremenitve strežnika. Operacije Ultimusovega aplikacijskega programskega vmesnika in dodatne funkcionalnosti smo se odločili izpostaviti preko spletne storitve. Za njen razvoj smo uporabili arhitekturo, osredotočeno na vire, ki se je kazala kot primerna že na začetku.

Prednost arhitekture, osredotočene na vire, je majhna kompleksnost, saj

sloni na že obstoječih tehnologijah, ki jih razvijalci poznajo. Izkoristi vse prednosti protokola za prenos hiperteksta (HTTP) in enoličnega naslova vira (URL), tehnologiji na katerih temelji današnji splet. Naš cilj je implementirati spletno storitev skladno z načeli arhitekture, osredotočene na vire.

V 2. poglavju bo predstavljena definicija poslovnega procesa, pristop upravljanja poslovnih procesov in sistemi za upravljanje poslovnih procesov ter možnosti integracije teh sistemov z zunanjimi sistemi. V 3. poglavju si bomo ogledali Ultimus BPM Suite in probleme s katerimi se srečujemo pri uporabi tega orodja. V 4. poglavju bo predstavljen REST arhitekturni stil in arhitektura, osredotočena na vire, na podlagi katere bo razvita rešitev integracije BPMS z okoljem za razvoj aplikacij. V 5. poglavju bodo opisani koncepti poslovnega procesa in predstavljeni izzivi pri razvijanju RESTful spletnih storitev za procesno naravnane sisteme. Primerjali in ocenili bomo RESTful spletne storitve različnih ponudnikov sistemov za upravljanje poslovnih procesov. V 6. poglavju bodo predstavljene operacije aplikacijskega programskega vmesnika in preslikava teh operacij v ustrezne HTTP metode na virih. Opisali bomo vire, vzorce enoličnih naslovov vira in HTTP metode izdelane RESTful spletne storitve. Predstavil bom tudi izzive, s katerimi sem se srečal pri načrtovanju in razvijanju. Rešitev bomo na koncu ocenili na podlagi lastnosti arhitekture, osredotočene na vire.

Poglavje 2

Zgodovina poslovnih procesov in njihovo upravljanje

V poglavju bomo opisali definicijo poslovnega procesa, prehod iz funkcijske v procesno organiziranost, pristop upravljanja poslovnih procesov in njegove osnovne faze. Seznanili se bomo tudi s sistemi za upravljanje poslovnih procesov in vgrajenimi možnostmi povezovanja teh sistemov z že obstoječimi aplikacijami v podjetju.

2.1 Poslovni proces

Najprej si poglejmo osnovno definicijo poslovnega procesa. Med svojim raziskovanjem literature sem našel več definicij poslovnega procesa različnih avtorjev.

„Poslovni proces je aktivnost oziroma skupek aktivnosti, ki vzame dobaviteljeve vhode, jim doda vrednost in proizvede rezultat za notranjo ali zunanjo stranko“ [10].

„Poslovni proces opredeljujemo kot zbirko dejavnosti, ki zahteva eno ali več vrst vložkov in ustvarja rezultat, ki za odjemalca pomeni neko vrednost“ [9].

„Poslovni proces opredeljujemo kot skupek logično povezanih izvajalskih

in nadzornih postopkov in aktivnosti, katerih posledica oziroma izid je načrtovani izdelek ali storitev“ [11].

Vsem trem navedenim definicijam je skupen vhod, aktivnosti, ki ustvarjajo dodano vrednost in izhod iz poslovnega procesa. Tako splošno definicijo prikazuje slika 2.1.

Slika 2.1: Shema poslovnega procesa.

Razumevanje in nadzor sta ključna pri uspešnem delovanju procesa. Če imamo nadzor nad vhodom v proces in razumemo celoten potek izvajanja procesa, potem imamo nadzor tudi nad izhodom iz procesa, ki predstavlja vrednost za odjemalca. Značilnosti dobrega procesa so [11]:

- orientiranost na odjemalca,
- dvigovanje dodane vrednosti proizvodov (izdelkov, storitev),
- znani in sposobni lastnik,
- razumevanje in sprejemanje s strani vseh sodelujočih v procesu,
- merljiva učinkovitost in uspešnost,
- neprestano izboljševanje.

2.2 Od funkcijske k procesni organiziranosti

Že leta 1776 je Adam Smith [15] analiziral uspeh delitve dela in specializacije na primeru izdelave bucik. Proces izdelave bucik je razdelil na približno 18 korakov, kjer je dopuščal možnost, da dva ali tri korake naredi isti delavec. Zaporedje korakov se začne s pridobitvijo žice, nato se žico zravna, potem se jo odreže, četrti korak je ostrenje žice, peti korak pa je brusenje dela žice, kamor nato pride glava bucike. Tudi izdelava glave bucike je sestavljena iz več korakov in tako naprej.

Maksimalna zgornja meja števila izdelanih bucik na dan za neizkušenega delavca je 20 bucik. V primeru, ko delavec opravlja samo en ali par korakov, namesto izdelave cele bucike, lahko 10 delavcev proizvede tudi do 48000 bucik na dan. Kar nanese 4800 bucik na enega delavca v enem dnevu. Kot je razvidno iz primera, osredotočanje na svoje ključne sposobnosti in masovna industrijska proizvodnja povečata podjetju konkurenčnost. Zaradi tega je v industrijski dobi prevladovala funkcijska, oddelčna organiziranost podjetja.

Taka podjetja si lahko predstavljamo kot zbirko organizacijskih enot, ki so zadolžene vsaka za svojo funkcijo v podjetju. Primeri takšnih funkcij oziroma organizacijskih enot so prodaja, finance, nabava, proizvodnja, kakovost in distribucija.

Funkcijska organiziranost je smiselna, če lahko aktivnosti razdelimo na enostavne, preproste, ponovljive korake v masovni proizvodnji in so potrebe kupcev standardizirane. Prednost takšne organiziranosti je v visoki specializaciji in izkoriščeni zmogljivosti.

Ko se pričakovanja kupcev premaknejo od standardiziranih h kupcu prilagojenim izdelkom ter se masovna proizvodnja zamenja s fleksibilno, funkcijska organiziranost ni več smiselna. Skupaj z manjšo življenjsko dobo izdelkov in veliko hitrostjo tehnoloških sprememb prihaja do nujnosti procesne organiziranosti.

2.3 Upravljanje poslovnih procesov

Upravljanje poslovnih procesov (ang. business process management – BPM) je „poslovni pristop k upravljanju sprememb pri prenavljanju poslovnih procesov“ [11]. Je kombinacija metod, orodij in tehnologij, ki jih uporabljamo za načrtovanje, analiziranje, izvajanje in merjenje poslovnega procesa z namenom povečanja produktivnosti in zmožnosti prilagajanja spremembam na trgu. Zmotno je prepričanje, da je BPM aplikacija ali stvar, ki jo podjetje kupi; je dejavnost, ki jo izvajamo.

Z definicijo upravljanja poslovnih procesov ne predpisujemo nobenih kon-

	Tradicionalno podjetje	**Procesno podjetje**
Poslovni izid	poslovna funkcija	poslovni proces
Organizacijska enota	oddelek	delovna skupina
Opis dela	ozko določen	širok
Osredotočenost	nadrejeni	stranka
Temelj nadomestila	aktivnost	rezultat
Vloga managementa	nadzor	mentorstvo
Ključna oseba	direktor poslovne funkcije	lastnik procesa
Poslovna kultura	konfliktno naravnana	sodelovanje

Tabela 2.1: Razlike med funkcijsko in procesno organiziranostjo podjetja. Povzeto po [11].

kretnih metodologij ali tehnologij. Vse to je odvisno od izbire podjetja, ki se odloči za izvajanje tega pristopa. Metode in orodja, ki so vključene v upravljanje poslovnih procesov in s katerimi podjetja lahko ustrezno odgovarjajo na spremembe poslovnega okolja so [11]:

- opredelitev strategije projekta prenove poslovanja, usklajevanje s strateškimi cilji podjetja in spremljanje učinkov (metoda ključnih dejavnikov uspeha),
- upravljanje znanja, upravljanje s spremembami in sprotno analiziranje poslovanja in spreminjanje rezultatov oziroma upravičenosti sprememb (metoda uravnoteženih kazalnikov), metoda spremljanja stroškov po aktivnostih,
- prenavljanje in izboljševanje poslovnih procesov (preurejanje poslovnih procesov, orodja za modeliranje poslovnih procesov, simulacije in analiziranje procesov),
- sistemi izboljševanja kakovosti (celovito upravljanje kakovosti, ISO certificiranje, poslovna odličnost ...),

- informatizacija poslovnih procesov podjetja (metodologije razvijanja informacijskih sistemov, orodja za modeliranje podatkov in razvijanje programskih rešitev ...),
- krmiljenje in spremljanje izvajanja delovnih procesov (Workflow Management Systems – WfMS), upravljanje z dokumenti,
- modeli in rešitve najboljše prakse (repozitorij podatkov, procesov in poslovnih pravil, referenčni modeli procesov, celovite programske rešitve ...).

Iz teh metod in pristopov lahko izpostavimo osnovne lastnosti pristopa upravljanja poslovnih procesov:

- Je procesno orientiran in zahteva poglobljeno znanje poslovnih procesov organizacije.
- Zahteva stalno izboljševanje in optimizacijo poslovnih procesov.
- Ponuja vpogled v poslovni proces v realnem času, omogoča splošno razumevanje procesa kot celote. Proces postane transparenten in možnosti za človeške napake in izigravanje se zmanjšajo.
- Podpira sodelovanje različnih kadrov kot so poslovni strokovnjaki in informatiki. Vključuje kupce, sisteme, dobavitelje in partnerje.
- Spreminja način delovanja organizacij iz papirnatih v računalniško vodene procese.
- S pomočjo programske opreme omogoča avtomatizacijo izvajanja procesov brez časovne zamude. Poleg tega pa omogoča, da se procesi izvajajo istočasno in neodvisno drug od drugega.

Sicer z definicijo BPM ne predpisujemo uporabe nikakršne programske opreme, vendar je ta dandanes nepogrešljiva. Zato je upravljanje poslovnih procesov „vedno bolj usmerjeno v razvoj platforme za integracijo poslovne strategije in poslovnih procesov s ključno arhitekturo podjetja“ [11]. Programsko

opremo, ki nam omogoča to povezovanje, imenujemo sistem za upravljanje poslovnih procesov (ang. business process management system – BPMS).

2.4 Osnovne faze upravljanja poslovnih procesov

Do sedaj je bilo predlaganih več modelov življenjskega cikla upravljanja poslovnih procesov. Življenjski cikel vsebuje različne faze, katerim naj bi sledili pri izvajanju BPM projektov [12]. V splošnem lahko prepoznamo 6 osnovnih faz upravljanja poslovnih procesov:

1. **Definiranje ciljev:** Najprej morajo izvajalci BPM projekta definirati svoje cilje, ki jih poskušajo doseči z izvedbo projekta.

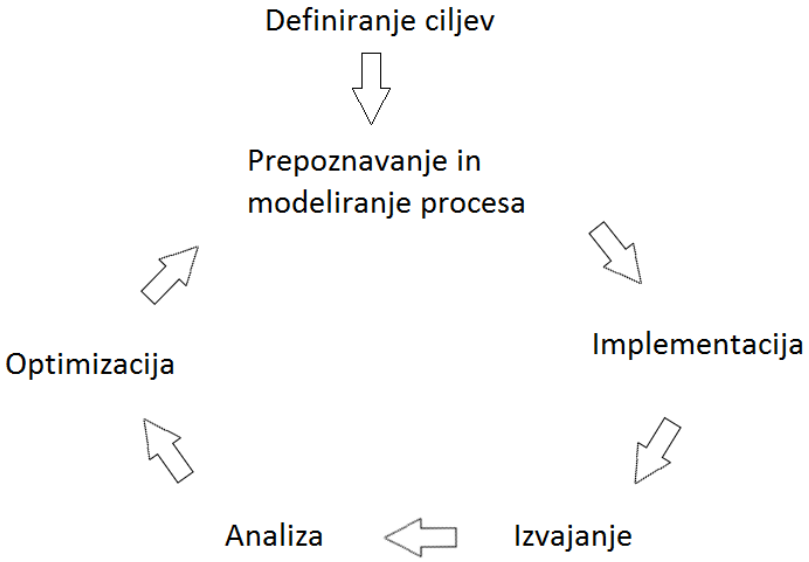
2. **Prepoznavanje in modeliranje procesa:** Cilj druge faze je prepoznati procese, ki prinesejo pozitivne spremembe v kontekstu BPM projekta. V nadaljevanju lastnik ali analitik modelira nov proces ali prilagaja obstoječega. Visoko nivojski načrt procesa, ki predstavlja skupek nalog, je ponavadi zgrajen s pomočjo grafičnih urejevalnikov.

3. **Implementacija:** Tretja faza je pretvorba visoko nivojskega načrta procesa v izvršljivo obliko procesa, ki vključuje tudi zaposlene in sredstva, s katerimi zaposleni upravljajo.

4. **Izvajanje:** Po končani fazi implementacije je izvršljiva oblika procesa zagnana in proizvaja dodatno vrednost za organizacijo ali željen izdelek za naročnika.

5. **Analiza:** Med izvajanjem opazujemo proces in pridobivamo podatke o procesu. Za merjenje uspešnosti se merijo ključni indikatorji uspešnosti, ki določajo nivo uspešnosti delovanja procesa. To so lahko čas trajanja poslovnega procesa, število zaključenih instanc procesa v določenem časovnem obdobju, število napak pri izvajanju in drugi.

6. **Optimizacija:** Na podlagi podatkov, pridobljenih v prejšnji fazi, v primeru mogoče izboljšave določenega dela procesa, izvedemo spremembe.



Slika 2.2: Osnovne faze BPM.

V takem življenjskem ciklu upravljanja poslovnih procesov lahko prepoznamo potrebo po BPMS. „Sistemi za upravljanje poslovnih procesov podpirajo definiranje oz. modeliranje, izvajanje in analiziranje poslovnih procesov“ [6]. Nudijo tehnološko podporo življenjskemu ciklu poslovnega procesa.

2.5 Sistemi za upravljanje poslovnih procesov

Obstaja veliko ponudnikov sistemov za upravljanje poslovnih procesov in nikakršnih predpisov, kaj mora programska oprema vključevati, da se kvalificira za to kategorijo. Kar pomeni, da si lahko vsak ponudnik BPMS po svoje predstavlja, kaj spada pod področje upravljanja poslovnih procesov. Zato so tudi razlike med temi sistemi različnih ponudnikov zelo velike. Nekateri

sistemi so bolj uporabni za procese z velikim številom interakcij z zunanjimi sistemi, drugi pa so bolj osredotočajo na procese z veliko interakcijami med ljudmi.

Sistemi za upravljanje poslovnih procesov naj bi v splošnem vključevali sistem za upravljanje delovnega toka (WfMS), okolje za izdelavo poslovnih procesnih pravil, integracijo z drugimi aplikacijami, uporabniški vmesnik in programsko opremo za opazovanje in merjenje uspešnosti izvajanja poslovnega procesa.

Sistemi za upravljanje poslovnih procesov so lahko videni tudi kot združitev nekaterih že prej obstoječih tehnologij in konceptov v eno. Predvsem vidni sta združitvi sistema za upravljanje delovnega toka in orodja za povezovanje aplikacij. „Za idejo povezovanja aplikacij stoji potreba po avtomatizaciji. Med koncem osemdesetih in devetdesetih let prejšnjega stoletja so se podjetja vedno bolj zanašala na aplikacije, katerih vloga je bila podpirati različne poslovne funkcije“ [5].

Problem povezovanja aplikacij se pojavi, ko hočemo vse korake različnih poslovnih funkcij (in različne aplikacije) povezati v razumljiv in neprekinjen proces. „Ko so se sistemi za upravljanje delovnega toka razvijali in je potreba po povezovanju aplikacij rasla, so podjetja začela ugotavljati, da bi lahko bili ti sistemi uporabni ne samo za nadzorovanje in pošiljanje informacij do ljudi, ki so vključeni v proces, ampak tudi za definiranje poslovne logike, potrebne za integracijo raznovrstnih in porazdeljenih sistemov“ [5].

Glavna pomankljivost sistemov za upravljanje delovnega toka v primerjavi z orodji za povezovanje aplikacij je bila podpora raznovrstnim aplikacijam preko različnih priključkov. Zato je bil naslednji logični korak združiti ta orodja s sistemi za upravljanje delovnega toka. Ti sistemi danes ponujajo podporo različnim aplikacijam preko že vgrajenih priključkov.

2.6 Možnosti integracije sistemov za upravljanje poslovnih procesov

Ob pregledu različnih sistemov za upravljanje poslovnih procesov sem identificiral različne možnosti integracij z zunanjimi aplikacijami.

Prva možnost integracije procesa je direktno s podatkovno bazo. Poslovni proces lahko preko priključka za podatkovno bazo dostopa do baz različnih zalednih sistemov in tako pridobiva podatke. To je uporabljeno predvsem v sistemih, ki zbirajo in interpretirajo velike količine podatkov.

Druga možnost je integracija s strežnikom elektronskih sporočil. Izvajalno okolje sistema za upravljanje poslovnih procesov spremlja strežnik elektronske pošte in čaka na prejeto elektronsko pošto. Ko elektronska pošta pride do strežnika, ta avtomatsko zažene poslovni proces in podatki so lahko pridobljeni iz elektronske pošte ter prikazani na procesni formi. Tak način integracije se uporablja v aplikacijah, kjer na primer podpora dobi elektronsko pošto, ki samodejno začne proces odpravljanja napake, svetovanja ...

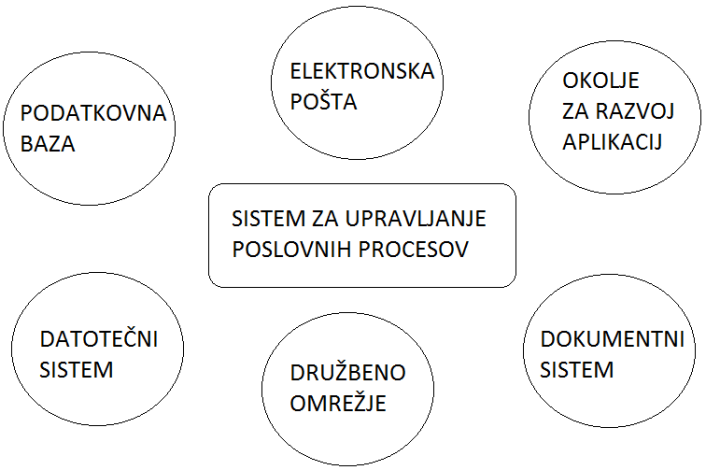
Tretja možnost je integracija sistema za upravljanje poslovnih procesov z okoljem za razvoj aplikacij. Taka integracija je največkrat podprta preko spletnih storitev ali preko aplikacijskega programskega vmesnika (ang. application programming interface – API). Spletne storitve omogočajo integracijo z drugimi poslovnimi procesi in nasploh z vsemi aplikacijami in sistemi, ki poznajo spletne storitve. Spletne storitve omogočajo napredne povezave s sistemi, za katere nimamo priključka (kot v prejšnjih dveh primerih, kjer uporabljamo priključek za podatkovno bazo in priključek za strežnik elektronske pošte). Zahtevajo pa več znanja in programiranja kot integracija preko že sprogramiranih priključkov. Poleg spletnih storitev pa lahko integracija poteka preko že omenjenih aplikacijskih programskih vmesnikov. Vmesnik vsebuje strukture in metode, s pomočjo katerih programska koda komunicira s poslovnimi procesi. Ta način integracije je predvsem uporaben za aplikacije, ki za svoje delovanje uporabljajo funkcionalnosti, ki jih ponuja BPMS. Aplikacije so omejene na programske jezike, v katerih so na-

pisani aplikacijski programski vmesniki. Različni sistemi ponujajo podporo aplikacijam v različnih programskih jezikih.

Še ena možnost integracije procesa pa je integracija z datotečnim sistemom. BPMS nadzoruje datotečne mape in ob zaznavi dodane datoteke sproži poslovni proces. Ta integracija je uporabna pri procesih, ki pridobijo informacije iz podatkovnega vira, ga spremenijo v format, primeren shranjevanju in shranijo informacije v podatkovno bazo.

S pojavitvijo in vzponom družbenih omrežij so se pojavili sistemi, ki omogočajo integracijo z le-temi. BPMS omogoča interakcijo s procesi preko dogodkov, kot so komentiranje ali objavljanje na družbenih omrežjih.

Omenimo še zadnjo možnost integracije, in sicer z dokumentnim sistemom. BPMS ponuja priključke za povezavo z različnimi dokumentnimi sistemi, ki ponujajo napredno upravljanje z dokumenti. Procesi lahko tako dokumente prenašajo, spreminjajo, iščejo in brišejo iz dokumentnega sistema.



Slika 2.3: Identificirane možnosti integracije sistemov za upravljanje poslovnih procesov.

Za nas najbolj pomembna možnost integracije je integracija z okoljem za razvoj aplikacij. V nadaljevanju si bomo pogledali, kako integracijo uporabljamo v našem podjetju in katere probleme nam povzroča tako povezovanje aplikacije s sistemom za upravljanje poslovnih procesov.

Poglavje 3

Začetno stanje in opis problema

V prejšnjem poglavju so bili predstavljeni sistemi za upravljanje poslovnih procesov in njihove možnosti integracije z zunanjimi aplikacijami. V tretjem poglavju bomo opisali Ultimus BPM Suite sistem za upravljanje poslovnih procesov, ki ga uporabljamo v našem podjetju. Na koncu bo predstavljen problem integracije sistema z okoljem za razvoj aplikacij.

3.1 Kaj ponuja Ultimus BPM Suite?

Zaenkrat si poglejmo, katere module vsebuje in kaj ponuja Ultimus BPM Suite [17, 16].

- **Ultimus BPM Server:** je izvajalno okolje, kamor so nameščeni poslovni procesi, ko je verzija poslovnega procesa zaključena in objavljena v modulu Ultimus BPM Studio. Skrbi za orkestracijo dogodkov v procesih in med njimi, poleg tega pa je zadolžen za integracijo z obstoječimi sistemi za upravljanje s podatki.

- **Ultimus BPM Studio:** je razvijalsko okolje, ki se uporablja za načrtovanje, razvijanje, simuliranje in testiranje vsakega poslovnega procesa. Modul je sestavljen iz odjemalca in strežnika. Posamezen odjemalec dostopa do strežnika, kjer je shranjen proces in ga ureja. Ker je

strežnik skupen, lahko naenkrat en proces razvija samo en razvijalec, spremembe pa so takoj vidne vsem. Kljub temu pa je možno hkratno delo na več procesih.

- **Ultimus Process Designer:** je orodje za modeliranje poslovnega procesa, namenjeno predvsem lastnikom podjetij, analitikom in upravljalcem procesov. Nudi enake zmogljivosti modeliranja kot Ultimus BPM Studio, vendar omogoča modeliranje brez potrebe razvijalcev. Analitiki in upravljalci lahko najprej modelirajo, analizirajo in avtomatizirajo poslovni proces, šele potem pa ga predajo razvijalcu za pretvorbo iz visoko nivojskega načrta procesa v izvršljivo obliko.

- **Ultimus BPM Database:** modul, ki mora biti ustrezno nastavljen z izbrano podatkovno bazo, je osnovni del Ultimus BPM okolja in vsebuje procesne in poslovne podatke, ki so ključni za izvajanje. Podatki so uporabljeni za izvajanje procesnih korakov, izpolnjevanje procesnih obrazcev, usmerjanje delovnega toka, pošiljanje opomnikov in generiranje poročil.

- **Ultimus organization charts:** je modul, ki omogoča grafično načrtovanje in predstavitev strukture podjetja. Ta vsebuje vsa delovna mesta, razmerja njihovih podrejenosti in nadrejenosti ter pripadnosti različnim oddelkom in področjem podjetja. Na podlagi te strukture se poslovni proces zaveda svojih izvajalcev in poslovnim procesom pametno usmerjati naloge posameznikom oz skupinam.

- **Ultimus Business Organization Database** je podatkovni vir, ki vsebuje podatke o strukturi podjetja, kreirani v Ultimus Organization Charts. Ponavadi je podatkovni vir ločen od Ultimus BPM Database.

- **Ultimus Flobots and FloStations:** Ultimusovi floboti (ang. workflow robots) omogočajo (že vgrajeno) integracijo poslovnih procesov z obstoječimi aplikacijami, ki je na voljo s klikom in brez kakršnegakoli

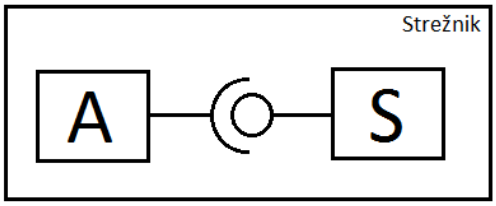
programiranja. Floboti se z lahkoto povežejo s sistemi, kot so spletne storitve, podatkovne baze, elektronska pošta, Excel in Word.

- **Ultimus System Administrator** in **Ultimus Process Administrator:** sta poleg modula Ultimus Organization Charts uporabljena za nadziranje in konfiguriranje Ultimus BPM Server modula.
- **Ultimus Director:** je grafični uporabniški vmesnik (ang. graphical user interface – GUI), ki omogoča izdelavo in upravljanje s poslovnimi pravili, povezanimi z avtomatiziranim poslovnim procesom.
- **Ultimus Enterprise Integration Kit (EIK)** ponuja orodje in poglobljeno tehnično dokumentacijo za razširjanje Ultimus BPM Server izvajalnega okolja. Omogoča integracijo z okoljem za razvoj aplikacij preko aplikacijskega programskega vmesnika, ki je zbirka programske kode.

3.2 Opis problema

Naš problem je posledica omejitve aplikacijskega programskega vmesnika, ki ga ponuja Ultimus za dostop do svojega izvajalnega okolja. Ultimus podpira namestitev izvajalnega okolja na strežniške verzije operacijskega sistema Windows. Da lahko spletna aplikacija dostopa do izvajalnega okolja preko aplikacijskega programskega vmesnika, mora biti spletna aplikacija nameščena na istem strežniku, kjer je nameščeno izvajalno okolje. Kot spletno aplikacijo imenujem aplikacijo, sestavljeno iz dela, ki teče na strežniku in dela, ki teče na brskalniku. To pomeni, da morajo biti naše spletne aplikacije nameščene na strežniku, na katerem je nameščeno tudi Ultimusovo izvajalno okolje. Povezavo med spletno aplikacijo (označena z A) in izvajalnim okoljem (označeno s S) prikazuje slika 3.1. Na njej vidimo, da sta spletna aplikacija in izvajalno okolje na istem strežniku.

Ločitev spletne aplikacije in izvajalnega okolja ima veliko prednosti. Prva je ta, da z namestitvijo aplikacij na drug strežnik razbremenimo strežnik, na



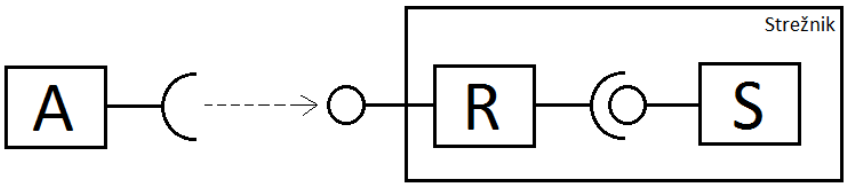
Slika 3.1: UML komponentna shema, ki prikazuje trenutni način komuniciranja spletne aplikacije (A) z izvajalnim okoljem (S).

katerem teče Ultimusovo izvajalno okolje. S tem se nam poenostavi tudi vzdrževanje spletnih aplikacij. Spremembe na aplikacijskem strežniku ne vplivajo na strežnik z izvajalnim okoljem in obratno. Povečana bo tudi skalabilnost naših aplikacij, saj lahko v primeru povečanja procesov ali uporabnikov lažje dodamo aplikacijski strežnik.

Pozitivne posledice bodo vidne tudi pri razvijanju aplikacij. Trenutno se aplikacija razvija na strežnikih z nameščenim Ultimusovim izvajalnim okoljem. Razvijalci morajo zato uporabljati virtualni dostop do strežnika za razvijanje aplikacije. Sedaj pa bo razvijalcem omogočeno razvijanje iz lokalnega računalnika.

Spletnim aplikacijam (A) bomo omogočili uporabo aplikacijskega programskega vmesnika in dodatnih funkcionalnosti na daljavo preko RESTful spletne storitve (R), kot prikazuje slika 3.2. Opazimo lahko, da aplikacija (A) in izvajalno okolje (S) nista več na istem strežniku. RESTful spletna storitev (R) predstavlja vmesni del, ki omogoča izvajanje operacij tudi iz naprav, ki nimajo nameščenega Ultimusovega izvajalnega okolja.

S tem bomo pridobili tudi neodvisnost od okolja za razvijanje aplikacij. Aplikacijski programski vmesnik se lahko uporablja samo v Microsoft .NET okolju za razvijanje aplikacij, RESTful spletna storitev pa bo dostopna iz katerega koli okolja za razvijanje aplikacij (in celo neposredno iz brskalnika).



Slika 3.2: UML komponentna shema, ki prikazuje komunikacijo aplikacije (A) in izvajalnega okolja (S) preko RESTful spletne storitve (R).

Seznanili smo se s trenutnim načinom dela in opisali probleme, ki jih srečujemo. V nadaljevanju bomo razvijali RESTful spletno storitev, ki bo omogočala povezavo BPMS z okoljem za razvoj aplikacij. Za razvoj RESTful spletne storitve bomo sledili načelom arhitekture, osredotočene na vire, ki bo predstavljena v naslednjem poglavju.

Poglavje 4

Arhitektura, osredotočena na vire

V prejšnjem poglavju smo opisali problem uporabe aplikacijskega programskega vmesnika. V tem poglavju si bomo pogledali REST arhitekturni stil in iz njega izpeljano arhitekturo, osredotočeno na vire, ki nam bo služila kot smernica za razvoj RESTful spletne storitve.

4.1 REST arhitekturni stil

REST označuje arhitekturni stil, ki je sestavljen iz „množice omejitev različnih arhitekturnih stilov“ [7]. Spletne storitve so RESTful, če dobro upoštevajo REST omejitve, ki jih je predstavil Roy Fielding v svoji doktorski disertaciji. REST vsebuje 6 glavnih omejitev [7]:

1. *Client-Server*: Odjemalec-strežnik omejitev spodbuja ločevanje skrbi pri razvijanju vmesnika od skrbi pri shranjevanju podatkov. Strežnik ponuja določene storitve in čaka na zahteve za te storitve. Odjemalec, ki želi izvajati storitev, pošlje zahtevo na strežnik. Strežnik nato zavrne ali izvede prejeti zahtevo in pošlje odgovor nazaj odjemalcu. Ta ločitev omogoča neodvisno razvijanje komponent in izboljša skalabilnost s poenostavitvijo komponent.

2. *Stateless*: Omejitev določa, da komunikacija med odjemalcem in strežnikom ne sme pomniti stanja, kar pomeni da mora vsaka zahteva odjemalca vsebovati vse potrebne podatke iz katerih lahko strežnik *razume* zahtevo in ne uporablja stanja, shranjenega na strežniku. Stanje seje je v celoti shranjeno na odjemalcu.

3. *Cache*: Predpomnilnik ima nalogo shranjevanja odgovorov na prejšnje zahteve, če so le-te označene, da se lahko shranijo. Shranjeni odgovori so lahko uporabljeni za kasnejše zahteve, ki bodo po vsej verjetnosti pripeljale do enakih odgovorov. Predpomnilniška komponenta ima možnost delno ali v celoti odstraniti nekatere interakcije in tako izboljšati storilnost in hitrost komuniciranja med strežnikom in odjemalcem.

4. *Uniform Interface*: Značilnost enotnega vmesnika je glavna prednost REST, saj je celotna arhitektura s tem poenostavljena in so interakcije med komponentami bolj pregledne.

5. *Layered System*: v takem sistemu vsak sloj ponuja storitve višjemu in uporablja storitve nižjega sloja. S tem se zmanjša sklopljenost različnih slojev ter poveča enostavnost njihove ponovne uporabe.

6. *Code-On-Demand*: Zadnja omejitev je opcijska in omogoča, da je funkcionalnost odjemalca, ki ima dostop do virov, ne pa navodil, kako jih obdelati, razširjena. Odjemalec pošlje zahtevo na strežnik, ta pa pošlje nazaj odgovor s kodo, ki vsebuje navodila za obdelavo virov. Odjemalec nato lokalno izvede prejeto kodo. To poenostavi odjemalca, ker zmanjša število potrebnih že implementiranih funkcionalnosti, vendar zmanjša preglednost interakcij.

Kot množica omejitev je REST zelo splošen. Nič glede teh omejitev ni vezano na mehaniko protokola za prenos hiperteksta (HTTP) ali strukture enoličnih naslovov vira (URL).

4.2 Arhitektura, osredotočena na vire

Skladno z REST omejitvami in tehnologijami spleta (HTTP, URL) je bila predstavljena arhitektura, osredotočena na vire (ang. Resource Oriented Architecture – ROA), ki je „dober primer RESTful arhitekture“ [14]. ROA ima 4 koncepte [14]:

- **Vir**: Vir je lahko karkoli, kar je dovolj pomembno za samostojno referenco.
- **Naslov vira**: vsak vir mora imeti vsaj eno *ime* oziroma naslov v obliki enoličnega naslova vira (URL). V primeru da vir nima svojega naslova, ga tretiramo kot del podatkov, ki opisuje nek drug vir. Naslovi so strukturirani na predvidljiv način.
- **Predstavitev virov**: viri so lahko predstavljeni v različnih formatih, kot so XML, slika, graf, spletna stran in tako naprej.
- **Povezanost virov**: Viri so med seboj povezani tako, da zraven predstavitvenih podatkov vsebujejo še naslove bližnjih virov.

Glavne lastnosti arhitekture, osredotočene na vire, so [14]:

- **Direktna dostopnost preko enoličnega naslova vira**: aplikacija mora izpostavljati enolične naslove vira za vsak podatek, ki ga streže. Ta lastnost omogoča, da dva različna uporabnika s pomočjo enakega naslova prideta do enakega vira. To lastnost ima tudi HTTP, na katerem sloni današnji splet. Zato si med seboj lahko pošiljamo enolične naslove vira, ki nas vodijo do enakih virov pri tem pa ni pomembno, s kakšnim zaporedjem smo do njega prišli. Vzemimo za primer sliko, ki jo želimo poslati prijatelju. Najprej v iskalnik vtipkamo besedno zvezo, nato pa kliknemo na sliko, ki jo hočemo poslati. Našemu prijatelju ni potrebno ponoviti vseh teh korakov, ampak v brskalniku samo odpre enolični naslov vira, ki smo mu ga poslali in bo prišel do iste slike z drugačnim zaporedjem korakov kot mi. Ta lastnost nam omogoča tudi

shranjevanje virov (npr. spletnih strani) v predpomnilniku, s katerim zmanjšamo količino podatkov, ki se prenašajo preko spleta.

- **Brez stanja**: to lastnost smo srečali že pri REST omejitvah, in sicer pomeni, da ima vsaka HTTP zahteva vse potrebne informacije za obdelavo na strežniku. Poleg tega se strežnik ne zanaša na informacije iz prejšnjih zahtev. Velikokrat se nam na spletu zgodi (npr. pri plačevanju računov preko spletne banke), da gumb „Nazaj“ v brskalniku ne deluje. Take spletne aplikacije pričakujejo zahteve v določenem vrstnem redu in s tem kršijo omenjeno lastnost.

- **Povezanost**: viri so med seboj povezani. Spletna storitev je povezana, če lahko spremenimo njeno stanje samo preko povezav, ki jih ponuja poleg predstavitvenih podatkov vira. To lastnost ima tudi splet, ki je zaradi tega zelo lahek za uporabo. Večina ljudi začne brskati na začetni strani brskalnika in se po spletu premika s pomočjo povezav (ne da bi na novo vpisoval enolične naslove vira).

- **Enoten vmesnik**: HTTP ponuja 4 glavne metode, s pomočjo katerih lahko upravljamo vire. Prva je **GET**, ki je uporabljena za pridobivanje informacij o viru (predstavitev vira). Druga je **PUT**, ki je uporabljena za kreiranje novega vira in spreminjanje obstoječega. Tretja je **DELETE**, ki jo uporabimo za brisanje vira. In zadnja je **POST**, ki je uporabna za kreiranje novih virov v relaciji z njihovimi *starši* oz. nadrejenimi viri. Razlika med PUT in POST (v primeru kreiranja novega vira) je ta, da je ob metodi POST strežnik zadolžen za izračun lokacije novega vira, pri metodi PUT pa odjemalec določi enolični naslov novega vira.

4.3 Utemeljitev izbrane arhitekture

Pri razvoju spletne storitve z upoštevanjem pravil arhitekture, osredotočene na vire, dobimo določene pozitivne lastnosti. Uporaba že znanih tehnologij spleta (HTTP, URL) omogoča hitrejše razvijanje spletne storitve. Povezanost in enotni vmesnik omogočata hitro razumevanje spletne storitve, brez dolge dokumentacije. Preko povezav do bližnjih virov so vidni so vsi viri spletne storitve, podprte HTTP metode pa prikazujejo mogoče operacije na virih. Razvijanje po načelih arhitekture, osredotočene na vire, prinese neodvisnost od jezika za izmenjavo podatkov. Najbolj pogosto se uporabljata JSON ali XML. Lastnost brez stanja poveča preglednost spletne storitve, saj morajo biti vsi potrebni podatki vključeni v zahtevo. Omogočena je pridobitev predstavitvenih podatkov vseh virov, ne glede na stanje spletne storitve. Direktna dostopnost preko enoličnega naslova vira pa omogoča, da enostavno dostopamo do predstavitvenih podatkov vseh virov, ki jih ponuja spletna storitev.

Poglavje 5

Uporaba arhitekture, osredotočene na vire, na poslovnih procesih

V prejšnjem poglavju je bila predstavljena arhitektura, osredotočena na vire, ki opisuje, kako uporabiti REST omejitve na tehnologijah spleta. Ta arhitektura predstavlja filozofijo, kako naj bi bile razvite RESTful spletne storitve. V tem poglavju bomo pogledali, s katerimi izzivi se srečamo pri uporabi arhitekture, osredotočene na vire, za razvoj RESTful spletne storitve v kontekstu poslovnih procesov. Opisali bomo koncepte poslovnih procesov ter pregledali in ocenili ustreznost obstoječeih RESTful spletnih storitev različnih ponudnikov sistemov za upravljanje poslovnih procesov.

5.1 Integracija preko spletnih storitev

Pozornost pri razvijanju integracije preko spletnih storitev se je usmerjala predvsem v SOA spletne storitve. SOA spletne storitve so osredotočene na operacije, medtem ko so RESTful spletne storitve osredotočene na vire in njihove predstavitve. Zgodovinsko so se najprej pojavile SOA spletne storitve, zato so imeli sistemi za upravljanje poslovnih procesov podprto in-

tegracijo predvsem preko SOA spletnih storitev. SOA spletne storitve izkoriščajo HTTP samo za prenos SOAP sporočil, RESTful spletne storitve pa HTTP uporabljajo kot glavni nosilec informacij ter bolje izkoristijo tehnologije HTTP (enolični naslov vira, HTTP metode).

Ob pojavitvi REST omejitev in začetkov razvijanja RESTful spletnih storitev ni prišlo do novih standardov. Razvijalci so si lahko po svoje prestavljali implementacijo RESTful spletnih storitev z obstoječimi tehnologijami (HTTP, URL). Zaradi tega je veliko tako imenovanih RESTful spletnih storitev, orientiranih na operacije namesto na vire. S tem pride do kršitve ROA. To sem opazil tudi ob raziskovanju RESTful spletnih storitev, ki jih ponujajo različni ponudniki sistemov za upravljanje poslovnih procesov.

5.2 Koncepti poslovnega procesa

V poslovnem procesu lahko prepoznamo naslednje koncepte [18]:

1. **Proces** je množica aktivnosti. Sestavljen je iz korakov, ki si sledijo po določenem vrstnem redu. Iz procesa nastanejo primerki procesa.

2. **Primerek procesa** je ustvarjen na podlagi procesa in je enolično določen z identifikatorjem. Naenkrat je lahko aktivnih več primerkov, ki so neodvisni en od drugega. Tudi izvedeni koraki se lahko razlikujejo glede na uporabniške podatke v primerku procesa. Tako kot se lahko spreminjajo uporabniški podatki, se lahko spreminja tudi stanje.

3. **Stanje** je podatek vezan na primerek procesa in korak. Primerek procesa ali korak sta lahko v več stanjih (aktiven, preklican, zaključen). Stanje primerka procesa ponavadi spreminjajo koraki. Na primer ob zaključku vseh predvidenih korakov postane tudi primerek procesa zaključen.

4. **Korak** je del procesa, ki vsebuje nalogo. Naloga je lahko dodeljena uporabniku, če pa je korak avtomatiziran, je naloga na primer klic spletne storitve ali izvedba programske kode.

5. **Usmerjanje** s poslovnimi pravili določa potek primerka procesa skozi množico korakov. Pravila aktivirajo naslednje korake na podlagi pridobljenih uporabniških podatkov.

5.3 Izzivi

Izzivi integracije poslovnih procesov preko spletnih storitev na podlagi arhitekture, osredotočene na vire, so [18]:

1. Preslikava procesnih konceptov v ustrezne vire in konstruiranje njihovih enoličnih naslovov vira.

2. Konstruiranje HTTP zahtev in odgovorov za različne tipe virov.

3. Določevanje ustreznih pomenov HTTP metod na posameznih virih.

V naslednjem poglavju bom podrobneje opisal izzive, s katerimi sem se srečal med načrtovanjem in razvijanjem RESTful spletne storitve.

5.4 Pregled obstoječih RESTful spletnih storitev

Na podlagi primerjave različnih sistemov za upravljanje poslovnih procesov [13] sem izbral tiste, ki imajo podprte RESTful spletne storitve in ocenil ustreznost le-teh na podlagi arhitekture, osredotočene na vire. Primerjal sem RESTful spletne storitve naslednjih sistemov za upravljanje poslovnih procesov: Bonita BPM 7.5 [2], jBPM 6.1 [3], ProcessMaker 3.2 [4] in Activiti 6 [1]. Med izbranimi sistemi ni Ultimus BPM Suite sistema, ker ne ponuja RESTful spletne storitve. Za oceno ustreznosti bomo vzeli lastnosti arhitekture, osredotočene na vire, opisane v poglavju 4.

5.4.1 Direktna dostopnost preko enoličnega naslova vira

Ta lastnost pomeni, da morajo biti podatki, ki jih streže spletna storitev, dostopni preko enoličnega naslova vira. Pri izbranih spletnih storitvah sem preveril, če so informacije, ki jih strežejo dostopne preko enoličnih naslovov vira. Vse REST spletne storitve, podprte s strani izbranih BPMS, ustrezno upoštevajo to lastnost. Izpostavljajo enolične naslove virov za modelirane procese, primerke procesov, korake in uporabniške podatke primerka procesa.

5.4.2 Brez stanja

Spletna storitev brez pomnenja stanja zagotavlja, da lahko odjemalec zahteve izvaja v poljubnem vrstnem redu. Spletna storitev v vsaki HTTP zahtevi dobi vse potrebne podatke za obdelavo zahteve in ne shranjuje kakršnega koli stanja o odjemalcu. Bonita BPM in ProcessMaker kršita to lastnost, saj je pred katerim koli klicem potrebno opraviti prijavo v njihov sistem. Sistem si nato zapomni uporabniške podatke in šele nato lahko prijavljen uporabnik iuporablja RESTful spletno storitev. Po zaključeni uporabi spletne storitve BPMS pričakuje odjavo iz sistema. Vrstni red klicev je vnaprej določen. jBPM in Activity pa upoštevata to lastnost, saj zahtevata avtentikacijske podatke v glavi HTTP sporočila. S tem omogočata poljuben vrstni red zahtev.

5.4.3 Povezanost

Povezanost virov je lastnost spletne storitve, ki omogoča spreminjanje virov samo preko povezav, ki jih ponuja poleg predstavitvenih podatkov virov. Za omenjene RESTful spletne storitve sem preveril, če vsebujejo poleg predstavitvenih podatkov virov še povezave za bližnje vire. Bonita BPM, ProcessMaker in jBPM poleg predstavitvenih podatkov ne vsebujejo nobenih enoličnih naslovov vira bližnjih virov. Activity pa upošteva to lastnost in poleg predstavitvenih podatkov koraka vsebuje še enolične naslove vira za iz-

vedbo operacij na tem koraku, primerek procesa, h kateremu korak pripada, in za predhodni korak.

5.4.4 Enoten vmesnik

Za spletno storitev lahko rečemo, da ima enoten vmesnik, če uporablja vsaj 4 glavne HTTP metode (GET, POST, PUT, DELETE) in so uporabljene v skladu z njihovim namenom (opisanim v prejšnjem poglavju). To lastnost krši jBPM REST spletna storitev, ker uporablja samo HTTP metodi POST in GET. GET uporablja za pridobivanje informacij o viru, POST pa za vse podprte operacije na virih. Ta način spominja na SOA spletne storitve, ki so osredotočene na operacije. ProcessMaker REST spletna storitev ne sledi pravilni uporabi POST in PUT metode. POST metodo uporablja za ustvarjanje vira na že znanem naslovu, kar je skladno s PUT metodo. Ostali dve RESTful spletni storitvi uporabljata vse 4 glavne HTTP metode (GET, POST, PUT, DELETE) v skladu z njihovim namenom.

RESTful spletne storitve BPMS sistemov	Direktna povezanost preko enoličnega naslova vira	Brez stanja	Povezanost	Enoten vmesnik
Bonita BPM	OK	X	X	OK
ProcessMaker	OK	X	X	X
jBPM	OK	OK	X	X
Activiti	OK	OK	OK	OK

Tabela 5.1: Uspešnost RESTful spletnih storitev pri upoštevanju lastnosti arhitekture, osredotočene na vire.

Struktura enoličnega naslova vira

Poleg naštetih lastnosti sem opazil še eno pomanjkljivost. V ROA so viri dostopni preko enoličnega naslova vira. Dobra praksa je, da je ta naslov sestavljen iz samostalnikov, ki predstavljajo vire. jBPM in ProcessMaker ne upoštevata te prakse, ker za določene operacije (ustvarjanje in zaključevanje

primerka procesa, preklic ali predodeljevanje) uporabita ime operacije namesto samostalnika, ki predstavlja vir. Za primer lahko vzamemo predodeljevanje primerka procesa. Relativni enolični naslov vira je `/cases/{app_uid}/reassign-case`, ki nakazuje operacijo. Bolj primeren relativni URL bi bil `/cases/{app_uid}/reassignment`. V tem primeru bi bil vir predodelitev primerka procesa, na katerem bi lahko bile podprte še ostale HTTP metode za spreminjanje vira ali brisanje vira, v kolikor bi bilo to smiselno.

Na podlagi dobro ocenjenih RESTful spletnih storitev, ki jih ponujajo sistemi za upravljanje poslovnih procesov sem naredil načrt za svojo rešitev integracije Ultimusovega sistema za upravljanje poslovnih procesov z okoljem za razvoj aplikacij. V podjetju želimo obdržati Ultimus BPM Suite, zato so naši napori usmerjeni k izdelavi RESTful spletne storitve za ta BPMS. Izdelava rešitve bo predstavljena v naslednjem poglavju.

Poglavje 6

Načrt in izdelava rešitve

V prejšnjem poglavju smo si pogledali, kateri so koncepti poslovnega procesa, in katere izzive imamo pri uporabi arhitekture, osredotočene na vire, za razvoj RESTful spletnih storitev procesno naravnanih sistemov. Pregledali in ocenili smo obstoječe RESTful spletne storitve različnih ponudnikov sistemov za upravljanje poslovnih procesov. V tem poglavju si bomo pogledali, katere operacije Ultimusovega aplikacijskega programskega vmesnika smo podprli in kako smo preslikali različne operacije v vire. Rešitev integracije Ultimusovega sistema z okoljem za razvoj aplikacij bomo ocenili po enakih kriterijih kakor RESTful spletne storitve v prejšnjem poglavju in primerjali te RESTful spletne storitve z našo RESTful spletno storitvijo.

6.1 Operacije Ultimusovega aplikacijskega programskega vmesnika

Ultimus preko aplikacijskega programskega vmesnika ponuja naslednje operacije:

1. Inicializacija začetnega koraka: izvajalno okolje si za vsak proces shrani identifikator začetnega koraka. Naloga, ki jo vsebuje začetni korak, nima prejemnika. Ko se korak zaključi, se ustvari nov primerek procesa,

ta pa se premakne v prvi korak, ki vsebuje nalogo s prejemnikom. Primerek procesa vsebuje sistemske in uporabniške podatke. Sistemske podatke (številka primerka, verzija, začetni uporabnik procesa) določi izvajalno okolje, uporabniške podatke pa določajo uporabniki. Vsak proces ima lahko več primerkov, ki se lahko na podlagi definiranih procesnih pravil, odvijajo drugače. Proces je sestavljen iz več korakov, ki imajo vsak svoja pravila.

2. Inicializacija nadaljnjih korakov procesa: poleg začetnega koraka ima proces nadaljnje korake. Vsak korak vsebuje nalogo, ki je dodeljena različnim prejemnikom. Ob inicializaciji koraka se naloga dodeli določenemu prejemniku. Koraki se inicializirajo po procesnih pravilih, ki so določeni za posamezne korake. Za pravilno usmerjanje procesa po korakih skrbi izvajalno okolje.

3. Spreminjanje uporabniških podatkov v procesu: uporabniški podatki primerka procesa se lahko spreminjajo v različnih korakih. Te podatke ponavadi vnašajo uporabniki procesa preko uporabniškega vmesnika oz. procesnih form. Na podlagi teh podatkov izvajalno okolje usmerja primerek v naslednje korake. Podatki so shranjeni v obliki XML dokumenta.

4. Predodeljevanje naloge: vsak korak vsebuje nalogo za določenega uporabnika. Ultimus omogoča, da se naloge predodelijo na druge uporabnike.

5. Zaključevanje koraka: ob zaključku aktivnega koraka izvajalno okolje na podlagi definiranih procesnih pravil in uporabniških podatkov primerka procesa usmeri primerek v naslednji korak. Zaključen je lahko samo aktiven korak.

6. Preklic koraka: se lahko izvede le na aktivnem koraku. S to operacijo ustavimo izvajanje koraka.

6.2 Viri in vzorci enoličnih naslovov vira

RESTful spletna storitev vsebuje več virov:

1. Proces: glavna zadolžitev vira procesa je ustvarjanje novih virov primerka procesa. Preko vira procesa je implementirana operacija inicializacije začetnega koraka za določen proces. Relativni vzorec enoličnega naslova vira [8], preko katerega je vir procesa dostopen, je oblike `/process/{processName}`. S parametrom `processName` enolično določimo vir procesa, s katerim želimo upravljati. Je hierarhično najvišji vir, nadrejen vsem ostalim, kar se pozna tudi v strukturi njegovega enoličnega naslova vira.

2. Primerek procesa: omogoča ogled sistemskih podatkov primerka procesa. Je podrejen viru procesa, zato je njegov relativni vzorec enoličnega naslova vira oblike `/process/{processName}/incident/{incidentNumber}`. S parametroma `processName` in `incidentNumber` natančno določimo primerek procesa.

3. Korak: predstavlja korak primerka procesa. Njegov relativni vzorec enoličnega naslova vira je oblike `process/{processName}/incident/{incidentNumber}/task/{taskId}`. Parameter `taskId` predstavlja enoličen identifikator koraka. Korak je nadrejen še trem virom, ki so dovolj pomembni za lastno referenco. To so seznam predodelitev, predodelitev koraka in zaključek koraka.

4. Seznam predodelitev: je vir, na katerem je omogočena izdelava nove predodelitve koraka. Relativni enolični naslov seznama predodelitve je `/process/{processName}/incident/{incidentNumber}/task/{taskId}/reassignments`.

5. Predodelitev koraka: novo ustvarjen vir predodelitve vsebuje podatke o predodelitvi koraka iz enega uporabnika k drugemu. Relativni enolični naslov vira predodelitve koraka je `/process/{processName}/incident/`

`{incidentNumber}/task/{taskId}/reassignments/{id}`. Parameter `id` je enolični identifikator predodelitve koraka.

6. Zaključek koraka: je vir, ki ga ustvarimo, ko hočemo zaključiti korak. Vsebuje podatke o tem, kdo je zaključil korak, kdaj in vpisan komentar. Relativni enolični naslov vira zaključka koraka je `/process/{processName}/incident/{incidentNumber}/task/{taskId}/completion`.

7. Preklic koraka: se ustvari, ko želimo preklicati aktiven korak. Relativni enolični naslov vira preklica koraka je `/process/{processName}/incident/{incidentNumber}/task/{taskId}/cancellation`.

Takšna struktura enoličnih naslovov vira dobro poudari razmerje med posameznimi viri. Ker želimo dostopati do korakov med izvajanjem procesa, nas zanimajo koraki v kontekstu posameznega primerka procesa. Usmerjanje na podlagi procesnih pravil modeliramo kot enolične naslove vira bližnjih virov, ki so podani zraven predstavitvenih podatkov. Odjemalec lahko na podlagi teh enoličnih naslovov vira sledi poteku procesa po korakih.

6.3 Podprte metode na virih

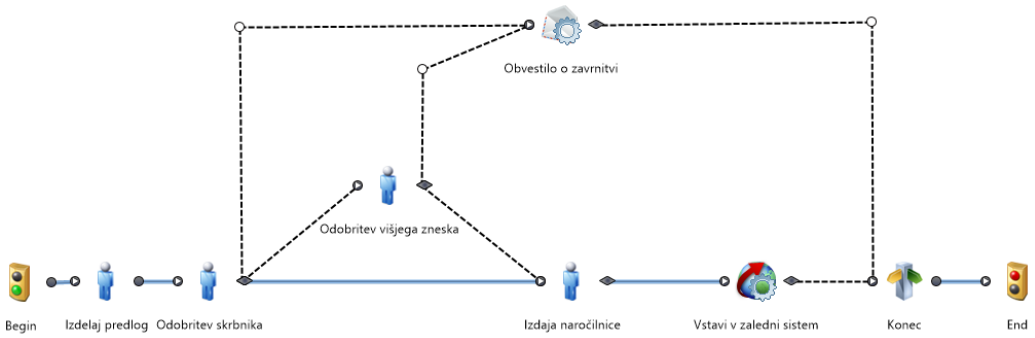
Sedaj, ko imamo pred seboj vse vire, ki jih streže naša RESTful spletna storitev, si poglejmo še to, katere HTTP metode podpirajo viri. Tabela 6.1 prikazuje podprte HTTP metode na posameznih virih.

6.4 Opis procesa

RESTful spletno storitev sem testiral na izdelanem testnem procesu. Slika 6.1 prikazuje proces, modeliran v orodju Ultimus BPM Studio. Delovanje RESTful spletne storitve si bomo pogledali na primeru procesa nabave. Najprej se v koraku *Izdelaj predlog* izdela predlog za nabavo. Ikona človeka predstavlja človeški korak, kar pomeni, da je korak dodeljen človeškemu uporabniku. Po

VIR	VZOREC ENOLIČNEGA NASLOVA VIRA	PODPRTE METODE
proces	/process/{processName}	GET, POST
primerek procesa	/process/{processName}/incident/{incidentNumber}	GET, DELETE
korak	/process/{processName}/incident/{incidentNumber}/task/{taskId}	GET, PUT
seznam predodelitev	/process/{processName}/incident/{incidentNumber}/task/{taskId}/reassignments	GET, POST
predodelitev koraka	/process/{processName}/incident/{incidentNumber}/task/{taskId}/reassignments/{id}	GET
zaključek	/process/{processName}/incident/{incidentNumber}/task/{taskId}/completion	GET, PUT
preklic	/process/{processName}/incident/{incidentNumber}/task/{taskId}/cancellation	GET, PUT

Tabela 6.1: Podprte HTTP metode na posameznih virih in oblika njihovih relativnih enoličnih naslovov vira.



Slika 6.1: Shema procesa nabave v Ultimus BPM Studiu.

vnosu podatkov in zaključku prvega koraka mora predlog odobriti skrbnik. V kolikor skrbnik ne potrdi predloga, gre proces v samodejni korak *Obvestilo o zavrnitvi*. V tem koraku se samodejno pošlje elektronsko sporočilo avtorju predloga. V primeru odobritve skrbnika ima proces dve možnosti napredovanja v naslednji korak. Če znesek predloga presega mejo 1500, gre proces v korak *Odobritev višjega zneska*, drugače gre v korak *Izdaja naročilnice*. Višji skrbnik lahko zopet potrdi ali zavrne predlog. Po končanem koraku izdaje naročilnice se izvede samodejni korak *Vstavi v zaledni sistem*, ki je tudi zadnji korak procesa.

Za nas pomembni so predvsem človeški koraki, ki niso samodejni in jih moramo ročno spreminjati in zaključevati, kar nam omogoča RESTful spletna storitev.

6.5 Pomen HTTP metod

Semantični pomen HTTP metod na posameznih virih si bomo pogledali s pomočjo procesa nabave. Vsi vzorci enoličnih naslovov vira v nadaljevanju so relativni. Za začetek moramo ustvariti nov primerek procesa. To storimo s HTTP POST zahtevo na `process/Nabava`. HTTP odgovor nam pošlje enolični naslov vira novo izdelanega vira primerka procesa, na primer `process/Nabava/incident/1`.

V prvem koraku moramo spremeniti uporabniške podatke, kamor vnesemo znesek predloga nabave. To storimo s HTTP PUT zahtevo na `process/Nabava/incident/1/task/123`, v telo HTTP zahteve pa vnesemo nov XML dokument, ki predstavlja uporabniške podatke. PUT metoda nam omogoča spreminjanje uporabniških podatkov aktivnega koraka.

Ko imamo vse podatke vnešene in izdelan predlog za nabavo, moramo zaključiti korak. Korak zaključimo s HTTP PUT zahtevo na `process/Nabava/incident/1/task/123/completion`. S tem ustvarimo vir zaključka na že znanem enoličnem naslovu vira.

Namesto zaključitve koraka pa lahko izvedemo tudi preklic aktivnega koraka, ob katerem se celoten primerek procesa prekliče. Vir preklica se ustvari s HTTP PUT zahtevo na `process/Nabava/incident/1/task/123/cancellation`.

Predstavljajmo si, da hočemo korak *Izdaja naročilnice* predodeliti k drugemu uporabniku, saj je trenutni dodeljeni uporabnik odsoten. Nova predodelitev koraka se ustvari s HTTP POST zahtevo na enolični naslov vira seznama predodelitev. HTTP GET zahteva na enolični naslov vira seznama predodelitev nam vrne vse predhodne predodelitve določenega koraka.

Recimo, da se je naš primerek procesa nabave končal (zaključil ali pre-

klical). Ta primerek procesa nabave lahko sedaj tudi izbrišemo. Izbris tega vira izvedemo s HTTP DELETE zahtevo na `process/Nabava/incident/1`.

6.6 Izzivi pri izdelavi rešitve

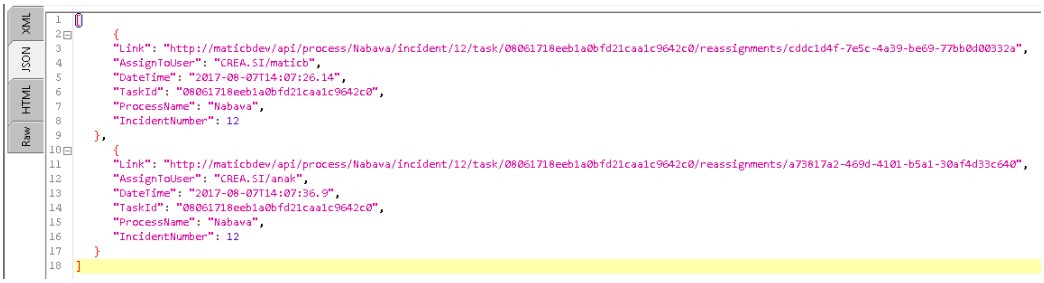
Izpostavil bom tri izzive, na katere sem naletel ob izdelavi rešitve. Ob že omenjenih izzivih integracije poslovnih procesov preko spletnih storitev na podlagi ROA, naštetih v 5. poglavju, sem se srečal še z dvema, ki bosta opisana v nadaljevanju.

6.6.1 Preslikava operacije v vir

Za ustvarjanje vira (seznam predodelitev) iz navadne operacije (predodelitev koraka), kot jo ponuja Ultimusovo orodje, je potrebno dodatno shranjevanje podatkov. Ultimusovo orodje ne shranjuje podatkov o prejšnjih predodelitvah koraka, zato ne ponuja dobre sledljivosti. Ponuja samo uporabniško ime uporabnika, kateremu je trenutno dodeljen korak. V skladu z ROA sem ustvaril vir predodelitve koraka. Za vsako predodelitev se še vedno izvede operacija, vendar se podatki o predodelitvi zapišejo v prav za ta namen izdelano tabelo v podatkovni bazi. Tako sem izvedel shranjevanje podatkov o viru predodelitve koraka. Glavna pridobitev pretvorbe v vir je torej sledljivost. Sedaj si lahko podatke o vseh izvedenih predodelitvah tudi ogledamo, kar prej ni bilo možno. Slika 6.2 prikazuje seznam vseh predodelitev za določen korak.

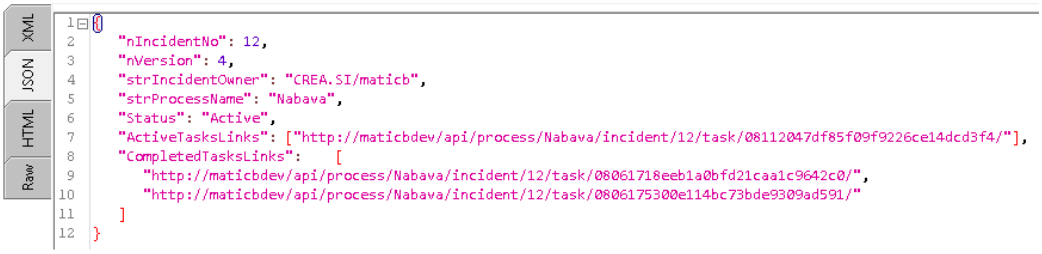
6.6.2 Povezanost RESTful spletne storitve

Da bi razvil povezano spletno storitev, sem moral dodati pridobivanje podatkov in računanje bližnjih virov. Ultimus na primer ne omogoča pridobivanja aktivnih primerkov procesa za posamezen proces preko aplikacijskega programskega vmesnika. To sem rešil tako, da sem pridobil podatke iz Ultimusove podatkovne baze in na podlagi teh podatkov izračunal bližnje vire



Slika 6.2: Seznam predodelitev za določen korak. Podatki so v formatu JSON.

posameznega procesa. Slika 6.3 prikazuje bližnje vire za posamezen primerek procesa.



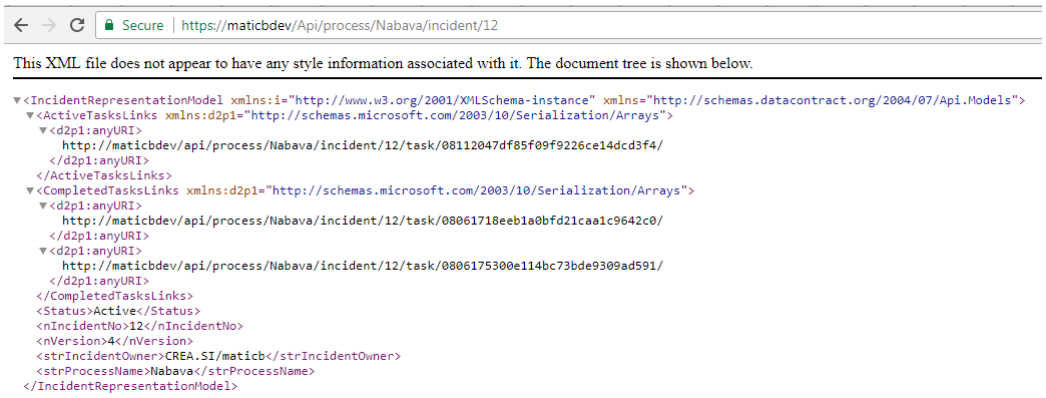
Slika 6.3: Predstavitev vira primerka procesa in enolični naslovi vira bližnjih virov. Podatki so v formatu JSON.

6.6.3 Varna komunikacija

Rešitev uporablja osnovno avtentikacijo (ang. basic authentication). V glavi HTTP zahteve mora biti prisoten *Authentication* parameter, ki vsebuje informacijo o vrsti avtentikacije in avtentikacijske podatke. Rešitvi sem dodal preverjanje tega parametra ob vsaki prejeti zahtevi. Na podlagi avtentikacijskih podatkov se spletna storitev odloči za izvedbo ali zavrnitev HTTP zahteve.

Poleg osnovne avtentikacije pa RESTful spletna storitev omogoča komunikacijo preko varnega protokola HTTPS. Na strežnik sem namestil certifikat, s katerim se naša rešitev predstavi kot zaupanja vredna. Podatki, poslani na

našo spletno storitev, se šifrirajo, zato nam zadošča osnovna avtentikacija. Slika 6.4 prikazuje varni način komunikacije preko HTTPS. Podatke smo tokrat prejeli kar preko brskalnika, kar je še ena prednost RESTful spletne storitve. Vidimo lahko, da rešitev za predstavitev podatkov poleg JSON formata podpira tudi XML.



Slika 6.4: Primer varnega načina komunikacije z RESTful spletno storitvijo preko HTTPS. Podatki so v formatu XML.

6.7 Ovrednotenje rešitve na podlagi arhitekture, osredotočene na vire

Izdelano rešitev bomo ocenili na podlagi lastnosti arhitekture, osredotočene na vire, tako kot smo to storili z različnimi RESTful spletnimi storitvami v prejšnjem poglavju.

6.7.1 Direktna dostopnost preko enoličnega naslova vira

Rešitev ponuja dostop do podatkov, ki jih streže, preko enoličnega naslova vira. Vsi viri so dostopni preko svojega enoličnega vira naslova.

6.7.2 Brez stanja

Rešitev o odjemalcu, ki izvaja operacije na virih, ne shranjuje nobenega stanja. V vsaki HTTP zahtevi morajo biti prisotni vsi podatki, ki jih spletna storitev potrebuje za obdelavo zahteve. Rešitev zahteva avtentikacijske podatke o odjemalcu, ki izvaja operacijo, v glavi HTTP zahteve.

6.7.3 Povezanost

Vsi viri poleg podatkov vsebujejo še povezave na bližnje vire. Rešitev ponudi odjemalcu enolične naslove vira, na katere lahko odjemalec v prihodnosti pošilja HTTP zahteve. Bližnji viri za primerek procesa so aktivni in zaključeni koraki primerka procesa. Rešitev omogoča spreminjanje virov samo preko povezav, ki jih ponuja poleg predstavitvenih podatkov virov.

6.7.4 Enoten vmesnik

Rešitev uporablja vse 4 glavne HTTP metode v skladu z njihovim namenom. GET se uporablja za pridobivanje podatkov o viru, npr. pridobivanje podatkov o koraku. POST se uporablja za ustvarjanje vira na nadrejenem viru, kjer URL novega vira ni znan. Primer je ustvarjanje nove predodelitve koraka, ki se ustvari na viru seznama predodelitev. URL novo ustvarjene predodelitve ni vnaprej znan; ob ustvaritvi ga spletna storitev sporoči v glavi HTTP odgovora. Poleg tega nastanek novega vira predodelitve ni idempotentna, saj se ob vsaki HTTP POST zahtevi ustvari nova predodelitev. PUT metoda je uporabljena v primeru novo ustvarjenega zaključka koraka. Enolični naslov vira zaključka koraka je vnaprej znan, ustvaritev vira zaključka pa je idempotentna, saj vsakemu koraku pripada največ en zaključek. Z metodo DELETE pa se izbriše zaključen ali preklican primerek procesa.

RESTful spletne storitve BPMS sistemov	Direktna povezanost preko enoličnega naslova vira	Brez stanja	Povezanost	Enoten vmesnik
Bonita BPM	OK	X	X	OK
ProcessMaker	OK	X	X	X
jBPM	OK	OK	X	X
Activiti	OK	OK	OK	OK
Izdelana rešitev	OK	OK	OK	OK

Tabela 6.2: Primerjava izdelane RESTful spletne storitve in prej omenjenih RESTful spletnih storitev na podlagi upoštevanja lastnosti arhitekture, osredotočene na vire.

Poglavje 7

Sklepne ugotovitve

V diplomski nalogi sem se seznanil s pristopom in sistemi upravljanja poslovnih procesov. Raziskal sem možnosti integracije BPMS z zunanjimi sistemi. Opisal sem Ultimus BPM Suite in analiziral probleme, ki jih srečujemo pri uporabi tega orodja. Predstavil sem REST arhitekturni stil in preučil arhitekturo, osredotočeno na vire, ki je primerna za razvoj RESTful spletnih storitev. Ocenil sem upoštevanje te arhitekture s strani RESTful spletnih storitev štirih sistemov za upravljanje poslovnih procesov. Na podlagi dobro ocenjenih RESTful spletnih storitev sem naredil načrt za svojo rešitev.

Predstavil sem operacije Ultimusovega aplikacijskega programskega vmesnika in prikazal preslikavo teh operacij v HTTP metode na ustreznih virih. Arhitektura, osredotočena na vire, se je pokazala kot pravilna odločitev za razvoj rešitve integracije Ultimusovega sistema z okoljem za razvoj aplikacij. V diplomi sem ugotovil, kateri podatki so dovolj pomembni za svojo referenco v rešitvi in kako je potrebno modelirati operacije povezane s procesi kot vire v ROA. S takim razvijanjem spletne storitve smo pridobili dobro sledljivost preteklih operacij, kar je pomembno za naše podjetje. Utemeljil sem strukturo uporabljenih enoličnih naslovov vira in ustreznost uporabljenih HTTP metod na posameznih virih. Za namen testiranja sem naredil proces nabave, na katerem sem preizkusil delovanje svoje rešitve. Na koncu sem rešitev tudi ocenil na podlagi arhitekture, osredotočene na vire, in dokazal, da sem jo

ustrezno upošteval.

Z izdelano RESTful spletno storitvijo sem omogočili ločitev spletnih aplikacij in izvajalnega okolja. Sedaj lahko izvajamo operacije Ultimusovega aplikacijskega programskega vmesnika, ne glede na to, ali je na konkretnem računalniku nameščeno izvajalno okolje ali ne. Za varno komunikacijo z RESTful spletno storitvijo preko HTTPS sem na strežnik namestil še certifikat.

7.1 Možne nadgradnje

Izdelana RESTful spletna storitev ima prostor za nadgradnje. Prva možna smiselna nadgradnja je podpreti več operacij Ultimusovega aplikacijskega programskega vmesnika. S trenutno rešitvijo sem začrtal smernice razvoja RESTful spletne storitve, ki so uporabne tudi za dodatne operacije.

Rešitev je izdelana na splošnih konceptih poslovnega procesa in bi lahko bila razširjena tudi na druge sisteme za upravljanje poslovnih procesov. RESTful spletna storitev bi lahko dodatno podpirala operacije na procesih, ki se izvajajo v izvajalnem okolju drugega BPMS, poleg Ultimusa.

Literatura

[1] Activiti 6 – dokumentacija RESTful spletne storitve. Dostopno na: `https://www.activiti.org/userguide/#_rest_api`. [Dostopano: 23. 7. 2017].

[2] Bonita BPM 7.5 – dokumentacija RESTful spletne storitve. Dostopno na: `http://documentation.bonitasoft.com/?page=bpm-api`. [Dostopano: 23. 7. 2017].

[3] jBPM 6.1 – dokumentacija RESTful spletne storitve. Dostopno na: `https://docs.jboss.org/jbpm/v6.1/userguide/jBPMRemoteAPI.html#d0e13597`. [Dostopano: 23. 7. 2017].

[4] ProcessMaker 3.2 – dokumentacija RESTful spletne storitve. Dostopno na: `http://wiki.processmaker.com/3.1/developer_info`. [Dostopano: 23. 7. 2017].

[5] G. Alonso, F. Casati, H. Kuno, in V. Machiraju. *Web Services: Concepts, Architectures and Applications*. Springer – Verlag, 2004.

[6] V. Bosilj Vukšić, L. Brkić, in M. Baranović. Business process management systems selection guidelines: Theory and practice. V zborniku *2016 39th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*, str. 1476 – 1481. IEEE, 2016.

[7] R. T. Fielding. *Architectural Styles and the Design of Network–based Software Architectures.* Doktorska disertacija, University of California, Irvine, 2000.

[8] J. Gregorio, R. Fielding, M. Hadley, M. Nottingham, in D. Orchard. Uri template. RFC 6570, Internet Engineering Task Force (IETF), 2012.

[9] M. Hammer in J. Champy. *Preurejanje podjetja: Manifest revolucije v poslovanju.* Gospodarski vestnik, 1995.

[10] H. J. Harrington. *Business process improvement: the breakthrough strategy for total quality, productivity, and competitiveness.* McGraw – Hill, 1991.

[11] A. Kovačič in V. Vukšić Bosilj. *Management poslovnih procesov: prenova in informatizacija poslovanja s praktičnimi primeri.* GV Založba, 2005.

[12] R. Macedo de Morais, S. Kazan, S. Dallavalle de Pádua, in A. Lucirton Costa. An analysis of bpm lifecycles: from a literature review to a framework proposal. *Business Process Management Journal*, št. 20, zv. 3, str. 412 – 432, 2014.

[13] A. Meidan, J.A. García – García, M.J. Escalona, in I. Ramos. A survey on business processes management suites. *Computer Standards and Interfaces*, št. 51, str. 71 – 86, 2017.

[14] L. Richardson in S. Ruby. *RESTful Web Services.* O'Reilly Media, Inc., 2007.

[15] A. Smith. An inquiry into the nature and causes of the wealth of nations. Dostopno na: `http://www.econlib.org/library/Smith/smWN1.html`, 1904. [Dostopano: 18. 4. 2017].

[16] Ultimus, Inc. *Ultimus Adaptive BPM Suite 8.3 Product Documentation*, 2012.

[17] Ultimus, Inc. *Ultimus Adaptive BPM Suite: Enterprise Business Process Management Software Platform*, 2014. Dostopno na: `http://www.ultimus.com/hs-fs/hub/54405/file-15725049-pdf/docs/product-technology/ultimus_adaptive_bpm_suite_v8_product_brief.pdf`. [Dostopano: 22. 4. 2017].

[18] X. Xu, L. Zhu, Y. Liu, in M. Staples. Resource–oriented architecture for business processes. V zborniku *Asia–Pacific Software Engineering Conference*, str. 395 – 402. IEEE, 2008.