Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Maruša Benedik

Modeliranje in izvajanje poslovnih procesov z uporabo BMPN 2

DIPLOMSKO DELO

UNIVERZITETNI ŠTUDIJSKI PROGRAM PRVE STOPNJE RAČUNALNIŠTVO IN INFORMATIKA

Mentor: prof. dr. Matjaž Branko Jurič

Ljubljana, 2013

Rezultati diplomskega dela so intelektualna lastnina avtorja in Fakultete za računalništvo in informatiko Univerze v Ljubljani. Za objavljanje ali izkoriščanje rezultatov diplomskega dela je potrebno pisno soglasje avtorja, Fakultete za računalništvo in informatiko ter mentorja.

Besedilo je oblikovano z urejevalnikom besedil LaTeX.

Univerza *v Ljubljani*
Fakulteta *za računalništvo in informatiko*

Tržaška 25
1001 Ljubljana, Slovenija
telefon: 01 476 84 11
01 476 83 87
faks: 01 426 46 47
01 476 87 11
www.fri.uni-lj.si
e-mail: dekanat@fri.uni-lj.si

Št. naloge: 00063/2013
Datum: 04.03.2013

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

Kandidat: **MARUŠA BENEDIK**

Naslov: **MODELIRANJE IN IZVAJANJE POSLOVNIH PROCESOV Z UPORABO BPMN 2**

BUSINESS PROCESS MODELING AND EXECUTION USING BPMN 2

Vrsta naloge: Diplomsko delo univerzitetnega študija prve stopnje

Tematika naloge:

Predstavite osnove storitveno orientirane arhitekture (SOA) in osnove upravljanja poslovnih procesov (BPM). Podrobno analizirajte izvršljivi jezik BPMN 2 skupaj z metamodelom in serializacijo v XML. Predstavite ključne konstrukte ter potek preoblikovanja abstraktnega procesnega modela v izvršljivi model. Predstavite in primerjajte tri sisteme za izvajanje poslovnih procesov, jBPM, Bonita Open Solution in Activiti. Izdelajte model poslovnega procesa obiska veterinarske klinike in ga pripravite za izvajanje z orodjem Bonita BPM Community.

Mentor:

prof. dr. Matjaž B. Jurič

Fakulteta za računalništvo in informatiko 2 UNIVERZA V LJUBLJANI

Dekan:

prof. dr. Nikolaj Zimic

IZJAVA O AVTORSTVU DIPLOMSKEGA DELA

Spodaj podpisana **Maruša Benedik**,
z vpisno številko **63040009**,

sem avtorica diplomskega dela z naslovom:

Modeliranje in izvajanje poslovnih procesov z uporabo BMPN 2

S svojim podpisom zagotavljam, da:

- sem diplomsko delo izdelala samostojno pod mentorstvom **prof. dr. Matjaža Branka Juriča**,
- 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 16. septembra 2013 Podpis avtorice:

Zahvaljujem se prof. dr. Matjažu B. Juriču za mentorstvo pri izdelavi diplomske naloge. Iskrena hvala tudi vsem, ki so mi tekom študija kakorkoli pomagali in mi nudili oporo.

KAZALO

1 Uvod **1**

2 Storitveno orientirana arhitektura **3**

2.1 Storitve 5

2.2 Kompozicija storitev 6

2.3 Razvojni cikel SOA 7

3 Upravljanje poslovnih procesov **9**

3.1 Življenjski cikel poslovnih procesov 10

3.2 Relacija med SOA in BPM 11

3.3 Vloge v razvojnem procesu 14

4 Modeliranje poslovnih procesov z BPMN 2.0 **17**

4.1 Notacija BPMN 19

4.2 BPMN 2.0 20

4.3 Elementi diagramov poslovnih procesov 21

4.3.1 Dogodki 23

4.3.2 Aktivnosti 25

4.3.3 Prehodi 28

4.3.4 Povezave 29

4.3.5 Plavalne steze 30

4.3.6 Artefakti 30

4.4 Sodelovanje med procesi 31

4.4.1 Primer sodelovanja obiska veterinarske klinike 32

4.5 Diagram koreografije in diagram pogovora 36

4.5.1 Diagram koreografije 36

4.5.2 Diagram pogovora 37

5 Izvajanje poslovnih procesov z BPMN 2.0 **39**

5.1 Izvajanje poslovnih procesov z uporabo procesnih strojev 40
5.2 Tipi skladnosti 42
5.2.1 Ustreznost modeliranja procesov 42
5.2.2 Ustreznost izvajanja procesov 42
5.2.3 Ustreznost izvajanja procesov BPEL 43
5.2.4 Ustreznost modeliranja koreografije 43
5.3 Semantika izvajanja procesov 43
5.4 BPMS s podporo za BPMN 2.0 44
5.4.1 Opis orodij BPMS 48
5.5 Preoblikovanje abstraktnega modela v izvajalni model BPMN 2.0 51
5.5.1 Postavitev meje med poslovnim in tehničnim nivojem modela . . 52
5.5.2 Avtomatizirana in človeško orientirana opravila 52
5.5.3 Določanje poslovnih pravil 53
5.5.4 Preoblikovanje abstraktnega modela procesa obiska veterinarske klinike v izvajalni model BPMN 2.0 56
5.6 Standardizirana shema XML 63

6 Sklepne ugotovitve **65**

A Proces: *Sistem veterinarske klinike* **67**

Seznam slik **75**

Seznam tabel **77**

Literatura **78**

SEZNAM UPORABLJENIH KRATIC

- ACID - Atomicity, Consistency, Isolation, Durability (atomarnost, konsistentnost, izolacija, trajnost)
- API - Application Programming Interface (vmesnik uporabniškega programa)
- BPEL - Business Process Execution Language (jezik za izvajanje poslovnih procesov)
- BPD - Business Process Diagram (diagram poslovnih procesov)
- BPM - Business Process Management (upravljanje poslovnih procesov)
- BPMI - Business Process Management Initiative (iniciativa za upravljanje poslovnih procesov)
- BPMN - Business Process Modeling Notation (notacija modeliranja poslovnih procesov)
- BPMN 2.0 - Business Process Modeling and Notation (modeliranje in notacija poslovnih procesov)
- BPMS - Business Process Management System (sistem za podporo upravljanja procesov)
- BRMS - Business Rule Management System (sistem za izvajanje poslovnih opravil)
- DoDAF - Department of Defense Architecture Framework (arhitekturni okvir Ministrstva za obrambo ZDA)
- ESB - Enterprise Service Bus (storitveno vodilo)
- IT - Information Technology (informacijska tehnologija)
- KPI - Key Performance Indicator (ključni kazalnik uspešnosti)

- LDAP - Lightweight Directory Access Protocol (lahki protokol za dostop do imenikov)
- LSB - Local Service Bus (lokalno storitveno vodilo)
- MEP - Message Exchange Pattern (vzorec za izmenjavo sporočil)
- OMG - Object Management Group (skupina za objektno upravljanje)
- SAP - Systems Applications and Products in Data Processing (sistemske aplikacije in produkti pri procesiranju podatkov)
- SOA - Service Oriented Architecture (storitveno orientirana arhitektura)
- SOAP - Simple Object Access Protocol (enostaven protokol za dostop do objektov)
- UDDI - Universal Description, Discovery and Integration (univerzalen opis, odkritje in integracija)
- UML - Unified Modeling Language (splošno namenski jezik za modeliranje)
- WfMC - Workflow Management Coalition (zveza za upravljanje delovnih tokov)
- WSDL - Web Services Description Language (opisni jezik spletnih storitev)
- XML - Extensible Markup Language (razširljiv označevalni jezik)
- XPath - XML Path Language (jezik XPath)
- XPDL - XML Process Definition Language (jezik XML za definicijo procesov)
- XSD - XML Schema Definition (definicija sheme XML)

POVZETEK

V diplomski nalogi smo najprej predstavili osnove storitveno orientirane arhitekture (SOA) in osnove upravljanja poslovnih procesov (BPM) ter njuno korelacija. Nato smo fokus usmerili na predstavitev izvajalnega BPMN (modeliranje in notacija poslovnih procesov), ki je z verzijo 2.0 postal veliko več kot samo notacija za modeliranje poslovnih procesov, saj je dobil nov spodaj ležeči metamodel in njegovo serializacijo v obliki formata XML. Za potrebe modeliranja z BPMN 2.0 smo predstavili vse konstrukte za izdelavo diagrama sodelovanja, diagrama koreografije in diagrama pogovora. Po izdelavi bolj abstraktnega modela BPMN smo izdelali izvajalni model BPMN 2.0, kjer smo abstraktni model preoblikovali tako, da je bil primeren za izvajanje na procesnem stroju. Običajno je dovolj že, da se za doseganje avtomatizacije poslovnih procesov v abstraktni model dodajo potrebne tehnične podrobnosti. Nato je celoten izvajalni model BPMN 2.0 pripravljen za izvajanje na procesnem stroju. Predstavili smo tri sisteme za podporo upravljanja procesov (BPMS): jBPM, Bonita Open Solution in Activiti. BPMS se uporablja za modeliranje, izvajanje in nadzorovanje poslovnih procesov. S pomočjo programskega orodja Microsoft Visio 2010 smo izdelali abstraktni model BPMN 2.0 za fiktiven primer obiska veterinarske klinike. Za modeliranje in implementacijo izvajalnega modela BPMN 2.0 pa smo uporabili platformo Bonita BPM Community 6.0.2.

Ključne besede:
SOA, BPM, BPMN 2.0, BPMS, Bonita BPM Community

ABSTRACT

In this diploma thesis we firstly represent the basics of Service Oriented Architecture (SOA) and the basics of Business Process Management (BPM) and also their correlation. Then we direct focus on a presentation of executable BPMN (Business Process Modeling and Notation), which has become with version 2.0 much more than just a business process modeling notation, because it has got a new underlying metamodel and its serialization in XML. For the purposes of modeling with BPMN 2.0 we represent all constructs for making the collaboration diagram, the choreography diagram, and the conversation diagram. After the completion of more abstract BPMN model we form executable BPMN 2.0 model, where abstract model is transformed in such a way that it is suitable for the execution on the process engine. To attain the business process automation it is usually enough to supplement needed technical details into abstract model. Afterwards, entire executable BPMN 2.0 model is prepared for the execution on the process engine. We discuss and compare three Business Process Management Systems (BPMS): jBPM, Bonita Open Solution, and Activiti. BPMSes are used to model, execute and monitor the business processes. With help of the software tool Microsoft Visio 2010 we design an abstract BPMN 2.0 model for the fictive example of visiting the veterinary clinic. Finally, we use the platform Bonita BPM Community 6.0.2 for modeling and implementation of the executable BPMN 2.0 model.

Key words:
SOA, BPM, BPMN 2.0, BPMS, Bonita BPM Community

Poglavje 1

UVOD

Organizacije so se v zadnjih letih začele zavedati dejstva, da imajo najboljši pregled nad poslovnimi procesi, če so ti v čim večji meri avtomatizirani, ter da obstaja čim manjša semantična vrzel med modeli poslovnih procesov in dejansko implementiranim informacijskim sistemom. Rešitev za to je storitveno orientirana arhitektura (angl. Service Oriented Architecture - SOA) v navezi z upravljanjem poslovnih procesov (angl. Business Process Management - BPM). BPM in SOA skupaj podpirata celoten življenjski cikel poslovnega procesa. BPM je procesno orientirana disciplina za upravljanje poslovnih procesov, SOA pa je primer arhitekture, ki jo uporablja informacijska tehnologija (angl. Information Technology - IT). Pri tem se (je) za modeliranje poslovnih procesov večinoma uporablja(la) notacija BPMN (angl. Business Process Modeling Notation), za implementacijo pa jezik BPEL (angl. Business Process Execution Language). S tem, ko je BPMN postal tudi izvajalni jezik, pa tako za modeliranje kot za implementacijo poslovnih procesov običajno zadostuje BPMN 2.0 (angl. Business Process Modeling and Notation), ki je z verzijo 2.0 dobil nov spodaj ležeči metamodel in njegovo serializacijo v obliki formata XML.

Diplomska naloga je v grobem razdeljena na tri dele. V prvem delu je predstavljena SOA in njene faze v razvojnem ciklu ter njena povezava s storitvami in kompozicijo storitev. Na kratko je opisana tudi tehnologija spletnih storitev.

V drugem delu je predstavljen BPM in njegova relacija do SOA. Opisane so faze v življenjskem ciklu poslovnih procesov, ki so poimenovane: definicija procesa, modeliranje, simulacija, implementacija, izvajanje, nadzorovanje in optimizacija. Poleg tega so predstavljeni tudi strokovnjaki, ki nastopajo v razvojnem ciklu poslovnih procesov.

Tretji in hkrati najobširnejši del diplomske naloge pokriva podrobno predstavitev izvajalnega BPMN 2.0 ter praktičen primer: obisk veterinarske klinike. Do različice 1.2 je

kratica BPMN pomenila notacija modeliranja poslovnih procesov (angl. Business Process Modeling Notation), v verziji 2.0 pa pomeni modeliranje in notacija poslovnih procesov (angl. Business Process Modeling and Notation). BPMN 2.0 je torej dobil nov metamodel in njegovo serializacijo XML. Najprej so opisani vsi konstrukti diagrama sodelovanja, ki prikazujejo poslovne procese. Opisana sta nova diagrama, ki sta bila dodana v BPMN 2.0, in se imenujeta diagram koreografije in diagram pogovora. Fiktiven primer obiska veterinarske klinike je predstavljen z vsemi tremi omenjenimi diagrami BPMN. Za izdelavo modelov je uporabljeno orodje Microsoft Visio 2010. Nato so predstavljene rešitve za implementacijo in izvajanje modelov BPMN. Za pretvorbo iz abstraktnega modela BPMN v izvajalni model BPMN je običajno dovolj dopis potrebnih tehničnih podrobnosti. S tem se zagotovi avtomatizacija poslovnih procesov, ki se kasneje izvedejo na procesnem stroju (angl. Process Engine). Pri polavtomatiziranih aktivnostih, torej uporabniških opravilih, je potrebna izdelava obrazcev, ki igrajo vlogo veznega člena med procesom in uporabnikom. Za izdelavo izvajalnega modela BPMN 2.0 se uporablja sistem za podporo upravljanja procesov (angl. Business Process Management System - BPMS), ki mora upoštevati tipe skladnosti in semantiko izvajanja poslovnih procesov. Predstavljeni so naslednji trije BPMS-ji: Activiti, jBPM ter Bonita Open Solution. Izdelava izvajalnega modela BPMN in kasnejša implementacija za fiktiven primer obiska veterinarske klinike se nadaljuje v BPMS-ju Bonita BPM Community 6.0.2. Prikazana sta modelirno okolje in portal za upravljanje poslovnih procesov v Boniti BPM ter spletni obrazec za uporabniško opravilo, ki je del praktičnega primera obisk veterinarske klinike. Na koncu je prikazan še izvajalni model BPMN 2.0 v formatu BPMN XML.

Poglavje 2

STORITVENO ORIENTIRANA ARHITEKTURA

V preteklosti je bilo načrtovanje poslovnih informacijskih sistemov predvsem funkcionalno usmerjeno. Ker so imeli ti sistemi množico funkcionalnosti in velike količine podatkov, se jih je prijelo ime silosi. Zaradi samega načina načrtovanja so bile podprte posamezne funkcije oziroma aktivnosti in ne celotni poslovni procesi. To je tudi (bil) osnovni problem silosov [1].

Po definiciji je poslovni proces skupek zaporedno povezanih specifičnih aktivnosti med seboj, ki stremijo k skupnemu cilju, torej doseganju zastavljenih poslovnih rezultatov. Nanj vplivajo dogodki iz zunanjega sveta ali drugih procesov. Aktivnosti se izvajajo popolnoma avtomatično ali pa zahtevajo človeško posredovanje. Učinkovitost procesa in posledično tudi učinkovitost celotne organizacije sta določena s strani vrstnega reda in učinkovitosti aktivnosti. Torej poslovni proces poleg vsega zgoraj napisanega obsega tudi ljudi, stroje in sisteme različnih organizacij, ki želijo doseči skupen poslovni cilj. Iz tega se lahko sklepa, da so poslovni procesi jedro vsake organizacije, njihovo celovito in učinkovito obvladovanje pa igra glavno vlogo pri dolgoročni uspešnosti podjetja [5–7].

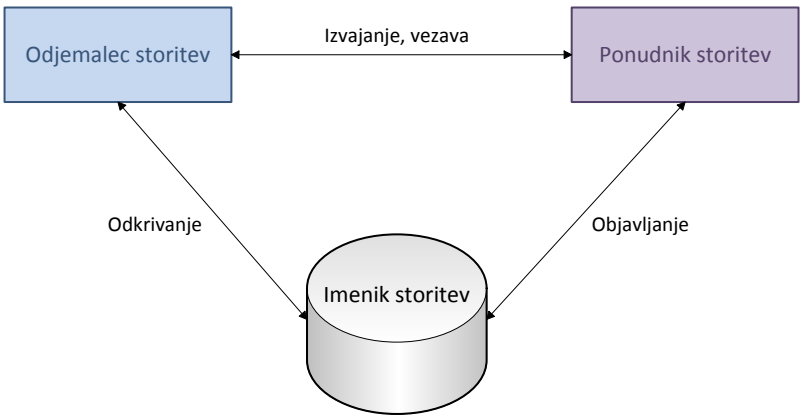
Za zagotavljanje celovitosti poslovnega procesa je potrebno delovanje nad določenim številom aplikacij (silosov), kar s tehničnega vidika predstavlja integracijski problem. Aplikacije celotnega poslovnega procesa so lahko narejene v različnih tehnologijah, nekatere pa so lahko tudi razpršene. Poleg tega mora poslovni proces poskrbeti za ustrezno zaporedje izvajanja aplikacij. Funkcionalnost poslovnega procesa je torej razpršena med več obstoječih sistemov ali z drugimi besedami, funkcionalnost je fragmentirana in tesno sklopljena [1].

Organizacije so se začele čez čas zavedati, da bi bilo idealno, če bi bili poslovni procesi

popolnoma avtomatizirani. To pomeni, da bi bil pri razvoju aplikacije podprt vsak korak poslovnega procesa.

Običajno so se (se še vedno) poslovni procesi modelirali »na papir«. Kot rezultat tega so nastale lepe slike ali diagrami. Pri takšnem načinu razvoja poslovnih procesov je prišlo do semantične vrzeli med modeliranimi diagrami poslovnih procesov in dejanskimi informacijskimi sistemi [2]. Zaradi tega je prihajalo do sprememb pri implementaciji poslovnih procesov, saj velikokrat diagrami niso bili dovolj jasno narisani ali pa sama tehnologija, ki se je uporabila pri implementaciji informacijskega sistema, ni omogočala točno takšne implementacije, kot je bila zamišljena pri modeliranju.

Na podlagi vsega tega je očitno, da je takšen razvoj poslovnih procesov zamuden in kompleksen, enako pa velja tudi za dopolnjevanje in spreminjanje poslovnih procesov. Kot rešitev na te probleme se je pojavila storitveno orientirana arhitektura ali krajše SOA (angl. Service Oriented Architecture).



Slika 2.1: **Osnovni elementi SOA [3].**

Osnovna ideja SOA je, da se nefleksibilne, enotne aplikacije transformira v dinamične in storitveno usmerjene poslovne aplikacije, ki se integrirajo v celovit informacijski sistem. To se doseže z implementacijo paradigme šibkega sklapljanja med modularnimi storitvami. Sklapljanje opiše, kolikšna stopnja sorodnosti obstaja v dekompoziciji med posameznimi elementi (v tem primeru storitvami), ki si niso bratje in sestre. Za samo iskanje oziroma odkrivanje storitev je potreben imenik storitev (angl. Service Registry), ki vsebuje informacije o storitvah, ki jih ponujajo ponudniki storitev. Imenik storitev igra vlogo posrednika informacij med odjemalcem in ponudnikom storitev [3]. Z drugimi besedami, ključni poudarek SOA je na definiciji načina integracije avtonomnih aplikacij (storitev), ki so lahko že obstoječe ali pa so na novo razvite, v celovit informacijski sistem.

Pri tem je poseben poudarek na modelu komunikacije, podpori različnih transportnih mehanizmov, zagotavljanju varnosti in transakcijske identitete, zanesljivosti sporočanja ter koordinaciji in kompoziciji.

S tehnološkega vidika je storitveno orientirana arhitektura le posebna vrsta porazdeljenih sistemov, kjer so storitve komponente sistema. Za implementacijo SOA se trenutno uporablja tehnologija spletnih storitev (angl. Web Services). Zgolj tri osnovne tehnologije: SOAP, WSDL in UDDI, pa za realizacijo opisanih konceptov ne zadoščajo, zato je potrebno uporabiti še druge tehnologije [1].

2.1 Storitve

Storitev je avtonomna aplikacija, ki opravlja neko točno določeno operacijo. Zaradi avtonomnosti lahko to operacijo uporabimo izven konteksta aplikacije, v kateri je ponujena. Storitve si med seboj izmenjujejo samo sporočila (podatke), ne pa tudi obnašanja (razredov oziroma metod).

Vsaka storitev mora imeti natančno definiran vmesnik, do katerega se preko omrežja dostopa z uporabo standardnih protokolov in podatkovnih tipov. Pravzaprav je vmesnik nabor operacij, ki so lahko sinhrone ali asinhrone, oziroma natančen opis sporočil, ki se lahko izmenjajo, ter določa zaporedje sporočil, ki je lahko zelo pomembno. V sinhronem načinu operacija odgovori odjemalcu storitve, ko je procesiranje končano. Odjemalec mora torej počakati na zaključek procesiranja. Ta način se uporabi, kadar procesiranje operacije traja kratek čas. V asinhronem načinu operacija odjemalcu ne vrne odgovora ob koncu procesiranja. Lahko pa vrne potrditev, da odjemalec ve, če je bil klic operacije uspešen. Če je potreben odgovor, potem se od storitve do odjemalca uporabi klic nazaj. V takšnem scenariju je potrebna vzajemnost sporočil.

Vmesnik je torej nekakšna pogodba med storitvijo in njenim odjemalcem in je tudi neodvisen od uporabljene implementacijske tehnologije. Zaradi takšnih lastnosti je vmesnik mogoče uporabljati za povezave brez stanja. Storitve same vzdržujejo svoje stanje, če ga potrebujejo. Vezava med odjemalcem in storitvijo se opravi na dva načina, in sicer statično, kjer se vezava opravi v času prevajanja, ali dinamično, kjer se vezava opravi v času izvajanja.

Storitve si izmenjujejo tipizirana sporočila in jih obdelujejo po načelih šibke sklopljenosti. Šibko sklapljanje storitev se doseže s samoopisovanjem vmesnikov, grobo granulacijo, izmenjavo podatkovnih struktur in podporo za sinhrone in asinhrone komunikacijske modele. Šibko sklopljene storitve so tiste, ki razkrijejo samo potrebne

odvisnosti in zmanjšajo vse vrste drugih nepotrebnih odvisnosti. Minimalne odvisnosti zagotavljajo, da če se storitev spremeni, so spremembe, narejene na drugih odvisnih storitvah, minimalne. Vsebina posameznih sporočil mora biti razumljiva. Nekatera sporočila so med seboj lahko soodvisna. Problem soodvisnosti (korelacije) se lahko reši z identifikatorji, ki so del vsebine, ali pa z umetno generiranimi identifikatorji. Poleg tega morajo imeti storitve sposobnost obdelave dolgo trajajočih sporočil.

Za razvoj resnično uporabljivih storitev je prav tako pomembno, da se nameni pozornost atributom, kot so razpoložljivost, zmogljivost, varnost, itd. To so atributi kakovosti storitve (angl. Quality of Service - QoS) in so pomembni v velikih informacijskih sistemih [1, 2].

2.2 Kompozicija storitev

Prvi korak pri realizaciji SOA je izpostavljanje funkcionalnosti v obliki ustrezno načrtovanih storitev. Na tem mestu se govori o atomarnih (samostojnih) storitvah. Ta vidik realizacije SOA se imenuje tudi pristop od spodaj navzgor (angl. Bottom-Up Approach) in je pogoj, da se lahko preide k drugemu koraku realizacije SOA. Drugi korak je kompozicija storitev ali procesni vidik realizacije SOA, ki se imenuje tudi pristop od zgoraj navzdol (angl. Top-Down Approach). Na tem mestu se govori o kompozitnih (sestavljenih) storitvah, ki so izpeljane iz večjega števila atomarnih storitev, in ni nujno, da se vsebovane atomarne storitve izvajajo v istem sistemu.

Z drugim korakom realizacije SOA, torej s kompozicijo storitev, se uresniči glavni cilj SOA, ki je, kot že rečeno, integracija atomarnih storitev v poslovne procese. S kompozicijo storitev se realizira enoten nivo, ki se imenuje procesni nivo. Ta nivo se lahko vidi kot inovacija SOA, kjer se definirajo poslovni procesi. Zaradi realizacije procesnega (dodatnega) nivoja se dosežejo ključne prednosti SOA, ki so:

– izboljšana odzivnost na poslovne procese,

– vpogled v poslovne procese »v realnem času«,

– enostavnost definicije in sprememb poslovnih procesov.

Seveda pa se omenjene prednosti dosežejo le z učinkovito realizacijo SOA. Procesni vidik realizacije SOA je torej neke vrste lepilo med storitvami in poslovnimi procesi, ki s procesnega vidika predstavljajo ključni sestavni del. Poslovne procese in pripadajoče

sodelovanje, izmenjavo informacij in aktivnosti se izrazi z uporabo ustreznih tehnologij, ki so predvsem ESB (angl. Enterprise Service Bus), BPEL in BPMN [1, 3].

Pri kompoziciji storitev se srečamo s pojmom **orkestracija**, ki pomeni, da si vnaprej definirane kombinacije atomarnih storitev sledijo po točno določenem vrstnem redu v poslovnem procesu. Torej se orkestracija poslovnih procesov (angl. Business Process Orchestration) lahko enostavno definira kot koordinacija storitev v poslovnem procesu na tehnični stopnji. Termin orkestracija kaže na zmožnost zbiranja diskretnih storitev v celoten procesni tok. S tem se zmanjšuje cena in kompleksnost integracije poslovnih procesov [2, 11].

Poleg orkestracije je potrebno omeniti tudi **koreografijo**, ki se nanaša na opazne (javne) interakcije storitev. Koreografija je tip procesa, ki specificira, kako mora odjemalec, ki je lahko uporabnik ali stroj, vzajemno delovati s storitvijo, da se dobi veljaven rezultat. Standardni proces oziroma proces orkestracije definira tok aktivnosti organizacije, koreografija pa formalizira pot, po kateri udeleženci v poslu koordinirajo svoje interakcije. Koreografija torej definira vrstni red, po katerem se pojavljajo naloge. Med udeleženci je fokus usmerjen bolj na izmenjavo informacij kot na samo orkestracijo dela, ki je predstavljena znotraj kroga udeležencev. Na koreografijo se lahko gleda tudi kot na tip poslovne pogodbe oziroma sporazuma med dvema ali več organizacijami. Med odjemalci, ki so običajno spletne storitve, ta sporazum iz globalnega zornega kota opisuje vedenje, ki se opazi od zunaj, in je predstavljeno s sporočili, ki se v urejenem vrstnem redu izmenjujejo med odjemalci. Torej je koreografija definicija pričakovanega obnašanja ali z drugimi besedami, proceduralna poslovna pogodba med vzajemno delujočimi udeleženci. Izmenjava sporočil partnerjem omogoča, da svoje poslovne procese med operacijami planirajo tako, da ne pride do konfliktov. Model koreografije omogoča izpeljavo vmesnikov procesov vsakega procesa poslovnega partnerja [3, 6, 12].

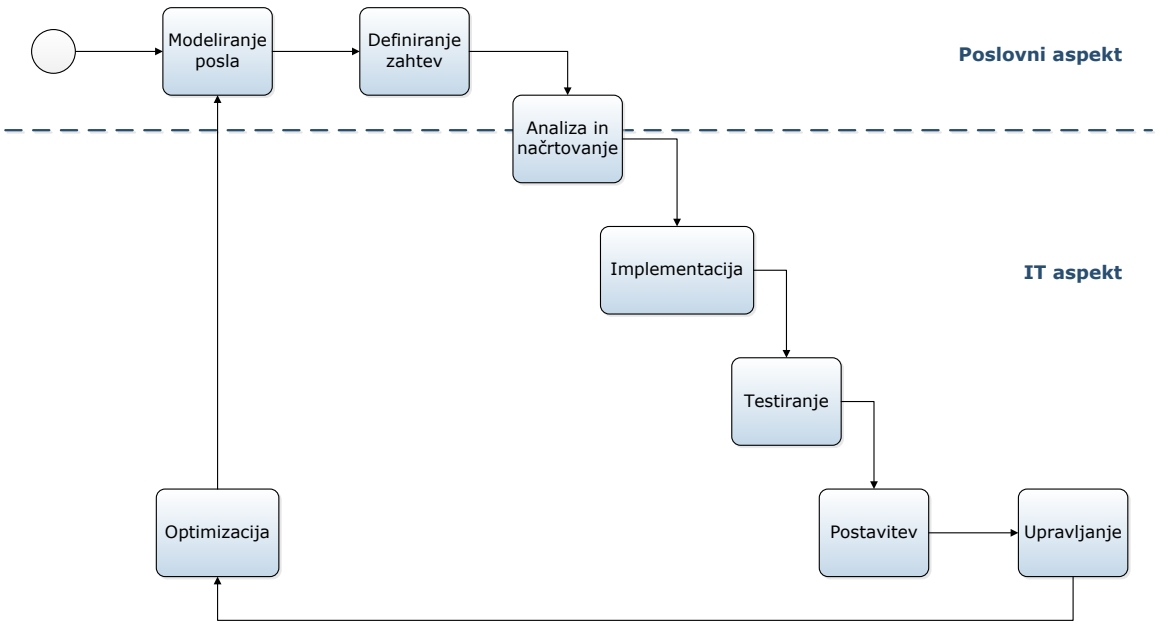
2.3 Razvojni cikel SOA

Faze v razvojnem ciklu SOA se nanašajo na:

1. **Poslovni aspekt** – Poslovne zahteve, ki izhajajo iz procesov, ki morajo biti podprti, so začetna točka razvojnega cikla v SOA.

2. **IT aspekt** – V teh fazah razvoja se čuti močan vpliv tehnologij, ki se uporabljajo v SOA. Ključnega pomena so načrtovanje in implementacija storitev ter kompozicija storitev. »Postavitev (angl. Deploy)« in »Upravljanje (angl. Manage)« sta fazi, ki

se nanašata na operacije storitev. Ti dve fazi se aktivirata v času izvajanju storitev in vključujeta nadzorovanje zmogljivosti, analizo napak ali ponovno konfiguracijo komponent sistema IT [3].



Slika 2.2: **Faze razvojnega cikla SOA [3].**

SOA minimizira semantično vrzel med modeli poslovnih procesov in dejansko programsko opremo informacijskega sistema, torej zmanjšuje vrzel med IT in poslom. SOA reši tudi problem avtomatizacije poslovnih procesov. To se realizira z izvajalnim BPMN 2.0 ali s preslikovanjem BPMN modelov procesov v izvajalni BPEL in povezovanjem BPEL-a s partnerji oziroma storitvami.

Storitveno orientirana arhitektura je obširen projekt, ki vključuje:

– poslovne aspekte,

– tehnične aspekte,

– organizacijske aspekte.

SOA sklene neke vrste kompromis med poslovnimi in IT aspekti [2].

Poglavje 3

UPRAVLJANJE POSLOVNIH PROCESOV

Organizacije v svojih poslovnih procesih običajno generirajo ogromno število podatkov, ki so pogosto razpršeni in nepregledni. Če pa se ti podatki povežejo v pregledne informacijske strukture, se lahko dobi natančen vpogled v vsako najmanjšo podrobnost poslovanja. S temi informacijami se je možno hitreje prilagajati željam in potrebam trga [4].

BPM (angl. Business Process Management) je organiziran celovit pristop, s katerim želijo podjetja ustvariti prožno organizacijo, ki je sposobna hitrega prilagajanja poslovnih procesov, kjer so slednji sredstva (angl. Assets) organizacije, saj strankam prinašajo vrednost, in doseganja najboljših rezultatov. Pod okrilje BPM spada tudi informacijska tehnologija, ki je ključna pri podpori neprestanega izboljševanja procesov. BPM je torej neke vrste most med poslovno filozofijo, pri kateri je velik poudarek na tem, da podjetje pozna potrebe svojih strank in se jim čim bolj prilagaja, ter informacijsko tehnologijo, ki omogoča spremljanje, analiziranje in usklajevanje poslovanja s strateškimi cilji organizacije. Poslovni proces pod okriljem BPM je pregleden in transparenten. Kadarkoli se lahko ugotovi, kje se pojavljajo ozka grla in zamude, zato se jih lahko hitro odpravi. Zaradi preglednosti se bolje definirajo dolžnosti in vloge v podjetju, zmanjša se čas vodenja, hitreje se odkrije prevare in lažje oceni regulacijske sporazume. BPM postaja kompleksnejši zaradi povečane dinamike poslovnega okolja in večjega povpraševanja po naprednejših storitvah in produktih. Ključna dejavnika uspešnosti sodobnih organizacij sta tako procesni pristop kot upravljanje poslovnih procesov [4, 6, 7].

Na dolgi rok je z uvedbo BPM mogoče spremeniti funkcionalno naravnano organizacijo v procesno naravnano organizacijo. Za slednjo je značilno, da jo opredeljujejo tako

imenovani procesi »od konca do konca« (angl. End-to-End) [4, 6].

BPM definira naslednje faze življenjskega cikla poslovnih procesov: modeliranje, simuliranje, implementacija, izvajanje, nadzorovanje ter optimizacija. V praksi se v preteklosti klasičen pristop k BPM ni izkazal za učinkovitega, saj ni zadostno podpiral vseh naštetih faz življenjskega cikla in ni zagotavljal želene fleksibilnosti. Z vpeljavo storitveno orientirane arhitekture, ki omogoča tesno povezanost poslovnih zahtev z dejansko implementacijo ter zagotavlja dobro podporo izvajanju, se je takšen pristop k BPM izkazal za veliko boljšega od klasičnega pristopa k BPM in je trenutno še vedno najboljši pristop k vpeljavi BPM [5].

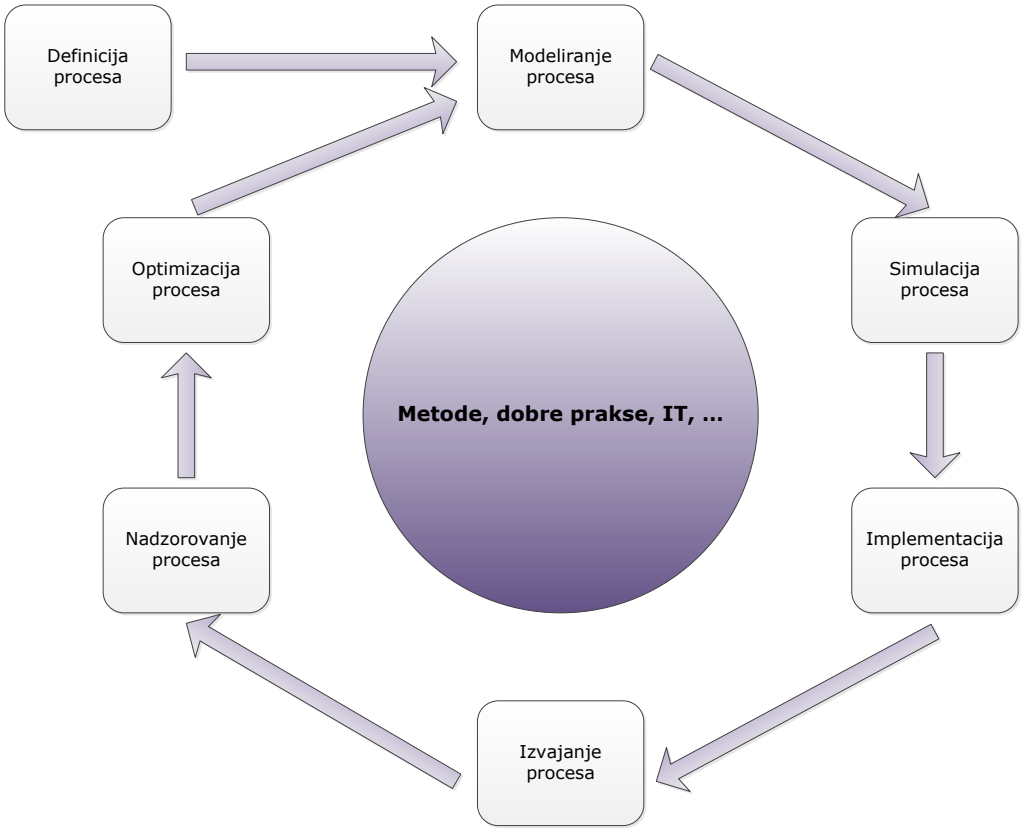
Metodologija BPM se lahko izvaja »na papirju« ali pa se uporabi programsko opremo BPMS (angl. Business Process Management System). Slednja je v veliko pomoč pri avtomatizaciji in neprestanem izboljševanju poslovnih procesov. BPMS je skupek več programskih orodij, s katerimi se poslovne procese načrtuje, avtomatizira, izvaja, nadzoruje in optimizira. Na izvedbeni ravni procesa BPMS poveže ljudi, aplikacije in podatke v eno celoto. BPMS je torej programsko orodje za razvoj procesno naravnanih IT rešitev.

Za izdelavo celostne rešitve morajo biti avtomatizirani poslovni procesi jasno definirani. Sem spadajo aktivnosti, poslovna pravila, vloge in ključna merila uspešnosti. Ad hoc procesov, torej nepredvidljivih procesov, pa ni mogoče uspešno avtomatizirati z obstoječimi BPMS orodji, saj poslovne procese običajno izvajajo in vodijo zaposleni, ki vodijo projekte, rešujejo probleme ali generirajo kreativne ideje [4].

3.1 Življenjski cikel poslovnih procesov

Upravljanje poslovnih procesov je sestavljeno iz več faz, ki pogosto temeljijo na dobrih praksah oziroma standardih (ISO 9000, COBIT, ITIL, itd.). Vsaka faza poslovnega procesa je podprta z določeno informacijsko tehnologijo. Vse faze skupaj tvorijo celoten življenjski cikel poslovnega procesa. Povezanost med fazami v ciklu omogoča, da se procesi nenehno spreminjajo, izboljšujejo oziroma prilagajajo [6].

Življenjski cikel poslovnega procesa se prične z definicijo oziroma analizo procesa in nadaljuje z modeliranjem. Učinkovitost modela je včasih dobro preveriti tudi s simulacijo, ki pomaga pri identifikaciji ozkih grl, ocenitvi cene tekočega procesa ter identifikaciji potencialnih problemov z viri (resursi) in njihovi razporeditvi [2]. Pri pristopu SOA k BPM je za modeliranje poslovnih procesov priporočljivo uporabiti notacijo BPMN, za implementacijo pa jezik BPEL ali izvajalni BPMN 2.0. Ko je poslovni proces v izvajanju, se njegova učinkovitost spremlja s pomočjo ključnih kazalnikov uspešnosti (angl. Key



Slika 3.1: **Življenjski cikel poslovnega procesa [6].**

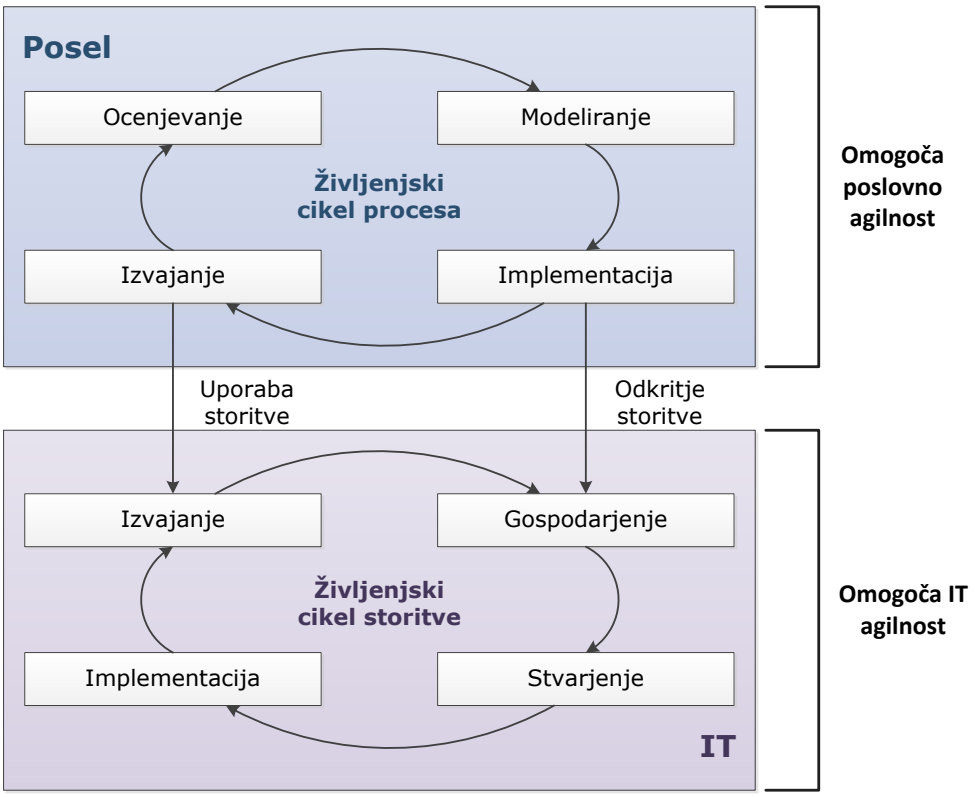
Performance Indicator - KPI), ki se konstantno zbirajo v času izvajanja poslovnega procesa. KPI-ji so osnova za izvedbo optimizacije, če pride do ugotovitve, da izvajajoči poslovni proces ne dosega želene učinkovitosti. Po izboljšavi oziroma optimizaciji procesa je priporočljivo preveriti realno vrednost optimizacije tudi z izvajanjem simulacij. Nato je potrebno popraviti vse spremembe v poslovnem modelu in implementaciji. Na koncu je potrebno dati v uporabo novo verzijo poslovnega procesa. Vsako novo in popravljeno verzijo poslovnega procesa je prav tako potrebno nadzorovati in po potrebi ponoviti cikel [5].

3.2 Relacija med SOA in BPM

BPM in SOA skupaj podpirata celoten življenjski cikel poslovnega procesa. BPM je procesno orientirana disciplina za upravljanje poslovnih procesov, SOA pa je primer arhitekture, ki jo uporablja IT. Za doseganje večje agilnosti BPM organizira ljudi, medtem ko SOA organizira tehnologijo. Procesi v SOA, torej povezane storitve, omogočajo

koordinacijo porazdeljenih sistemov, ki podpirajo poslovne procese, ob enem pa procesi v SOA ne bi smeli biti nikoli zamešani s poslovnimi procesi [2, 7].

Kot se lahko opazi na spodnji sliki, ima SOA odločilno vlogo v fazi implementacije procesa ter fazah izvajanja in nadzorovanja procesa [2].



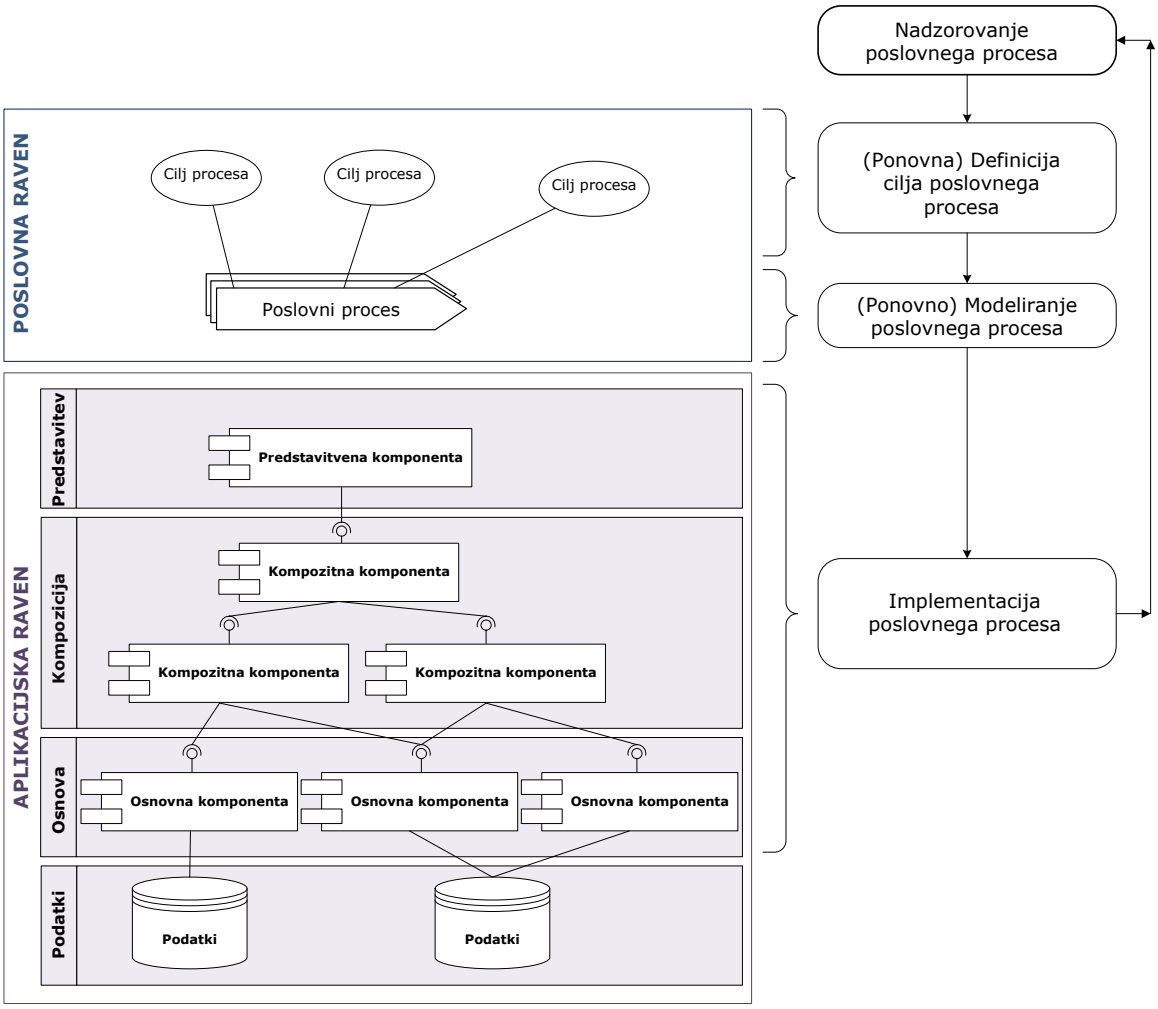
Slika 3.2: **BPM in SOA omogočata poslovno agilnost [2].**

Na podlagi aktivnosti, ki se izvajajo znotraj BPM življenjskega cikla, se torej skupni življenjski cikel SOA in BPM razdeli na poslovno raven in aplikacijsko oziroma IT raven.

Na poslovni ravni je poslovni proces specificiran z uporabo notacije modeliranja procesov. Slednja se ne ozira na podrobnosti podpore planiranega poslovnega informacijskega sistema. Zahtevana funkcionalnost in sistemi IT, ki so že v uporabi, so le grobo identificirani, saj je na tej ravni fokus usmerjen na zahtevane cilje procesa. To se opravi znotraj dveh faz BPM: (ponovna) definicija cilja poslovnega procesa in (ponovno) modeliranje poslovnega procesa.

S prihodom na aplikacijsko oziroma IT raven pride na vrsto SOA, saj je potrebno izvesti implementacijo poslovnega procesa. S poslovne perspektive je integrirana podpora IT za poslovne procese močno zaželena , vendar v večini primerov ni priskrbljena. Ta

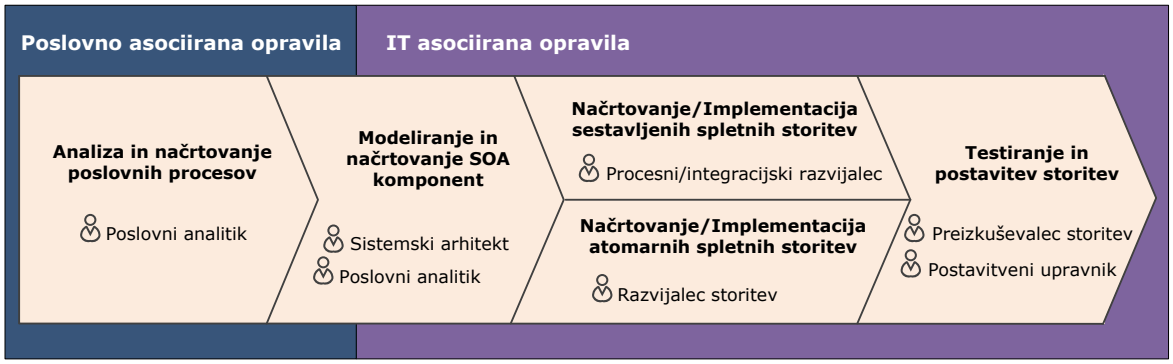
zahteva se še posebno opira na kompozicijsko plast storitveno orientirane arhitekture, kjer se uporablja standard oziroma tehnologija za kompozicijo spletnih storitev, kot je na primer BPEL. To omogoča sestavljanje različnih funkcionalnosti oziroma pridobljenih spletnih storitev v novo sestavljeno storitev ali kompozicijo, vse skladno s tokom poslovnega procesa (angl. Business Process Flow).



Slika 3.3: **Relacija med SOA in BPM [3].**

Po (ponovnem) modeliranju in implementaciji poslovnega procesa se na osnovi vnaprej definiranih ciljev nadzira poslovni proces. Z nadzorom poslovnega procesa se lahko s pomočjo informacij, ki se konstantno zbirajo v času izvajanja implementiranega poslovnega procesa, ugotovi, če poslovni proces zadovoljivo ustreza ciljem poslovnega procesa, oziroma če je potrebna optimizacija [3].

3.3 Vloge v razvojnem procesu



Slika 3.4: **Vloge v razvojnem procesu SOA v navezi z BPM [3].**

Zlasti v *fazi analize in načrtovanja* sta SOA in BPM med seboj zelo povezana, saj je fokus usmerjen na poslovne procese, ki predstavljajo aplikacijsko domeno ciljnega storitveno orientiranega sistema. Storitveno podprte aktivnosti so predstavljene znotraj modelov procesa in ne s primeri uporabe. Modeli poslovnega procesa lahko bazirajo na diagramih BPMN ali diagramih aktivnosti (angl. Activity Diagrams) UML 2 (angl. Unified Modeling Language). Ta faza spada pod okrilje poslovnega analitika (angl. Business Analyst), ki mora biti domenski strokovnjak za poslovne procese, kateri pridejo v poštev pri razvoju dotičnega sistema. Poslovni analitik mora poznati tehnike upravljanja poslovnih procesov, kot sta modeliranje procesov in simulacija. Poleg vsega tega mora imeti tudi osnovno znanje o storitveno orientiranem načrtovanju, saj mora identificirati potencialne storitve in storitveno podprte aktivnosti. V končni fazi to vodi v optimiziran načrt poslovnih procesov, vključno s specifikacijami izvajalnih poslovnih procesov, poznanih tudi kot delovni tokovi.

Na podlagi izvajalnih modelov procesa se ustvari *model komponent SOA* (angl. SOA Component Model). Za realizacijo tega se lahko uporabi samostojen diagram komponent UML (angl. UML Component Diagram), ki obsega vse komponente storitev, ki so zahtevane za implementacijo želene storitveno orientirane podpore IT. Komponente storitev so lahko kompozitne (sestavljene) storitve ali atomarne storitve. Rezultat tega je model komponent SOA, ki odseva načrt storitveno orientirane podpore IT za želeni poslovni proces. V tej fazi mora sistemski arhitekt (angl. System Architect) delati v tesnem sodelovanju s poslovnim analitikom, da se izogne napačnim interpretacijam poslovnih modelov.

V naslednji fazi pride na vrsto *podrobno načrtovanje identificiranih komponent SOA*

s strani dveh različnih tehničnih razvijalcev. Sestavljene storitve se načrtujejo in implementirajo ločeno od atomarnih storitev. Procesni/integracijski razvijalec (angl. Process/Integration Developer) dodela specificirane kompozicije storitev na podlagi modela komponent in modela izvajalnih procesov (angl. Executable Process Model), kar na koncu privede do polno izvedljive BPEL ali BPMN 2.0 definicije procesa. Razvijalec storitev (angl. Service Developer) podrobno načrtuje in implementira atomarne storitve. Razvijalcu storitev predstavlja glavni izziv razširitev že obstoječih aplikacij z vmesnikom spletnih storitev.

V zadnji fazi so storitve *postavljene in testirane v integriranem okolju*. Še posebej je pomembno testiranje in preverjanje specifičnih sodelovanj med različnimi komponentami storitev. Naloge zadnje faze opravljata postavitveni upravnik (angl. Deployment Manager) in preizkuševalec storitev (angl. Service Tester) [3].

Poglavje 4

MODELIRANJE POSLOVNIH PROCESOV Z BPMN 2.0

Modeliranje procesov obsega izdelavo in uporabo modelov. Posamezen model predstavlja sliko oziroma posnetek določenega realnega stanja. Glavni cilj modeliranja je izdelava trenutnega modela procesa (angl. As-Is), v nekaterih primerih tudi izdelava novega modela ali optimiziranega modela obstoječih procesov (angl. To-Be). Pri izdelavi trenutnega modela procesa je potrebno paziti, da se resnično modelira trenutno stanje in ne želja. Pri kompleksnejših poslovnih procesih je priporočljiva razdelitev na več podprocesov. Vsak proces, ki vsebuje podproces, mora priskrbeti zahtevane vhode v podproces in zahtevane izhode iz podprocesa. Načrtuje se optimalen tok procesa (angl. Process Flow) in identificira verjetne alternativne oziroma izjemne scenarije. Ker izjeme prekinejo običajen tok procesa, je potrebno specificirati, kako se bodo te izjeme obravnavale. Dobro je izvesti tudi več skupnih pregledov modela. Pri modeliranju celostne (angl. End-to-End) implementacije poslovnega procesa je potrebno veliko pozornosti nameniti tudi pravilni stopnji granularnosti [2, 5, 6].

Granularnost se sklicuje na »čistost« ali »robatost« aktivnosti. Opiše drobnozrnate (angl. Fine-Grained) ali grobozrnate (angl. Coarse-Grained) karakteristike izvedenih operacij s strani aktivnosti ali podatkov, ki jih sporočijo in/ali priredijo aktivnosti, ali pa so kombinacija obeh. Najbolj drobnozrnata aktivnost je metaforično gledano kot osnovna »Lego kocka« [10].

Modeli poslovnih procesov morajo biti razumljivi in prenosljivi znotraj organizacije, kakor tudi med organizacijami. Zaradi tega jih je priporočljivo izdelati na podlagi standardiziranih notacij, med katerimi sta vodilni UML in BPMN. Največja razlika med njima je v tem, da je UML objektno orientiran, BPMN pa procesno. UML se torej

osredotoča na razvoj programske opreme, BPMN pa na procese. Ker je notacija BPMN procesno orientirana, je primernejša za modeliranje poslovnih procesov in je postala nekakšen de facto standard na področju modeliranja poslovnih procesov [5, 6]. S prihodom BPMN je bil storjen velik napredek v BPM tehnologiji [9].

BPMN model je v bistvu prikaz poslovnih procesov, ki predstavljajo procesno orientiran pogled orkestracije, in je fokusiran na prikaz toka zaporedja [10]. Notacija BPMN je postala priljubljena tudi zaradi preglednosti in razumljivosti nestrokovnjakom. Še večjo prednost pred drugimi notacijami pa ima zato, ker boljša orodja BPMS omogočajo avtomatsko pretvorbo modela BPMN v skelet BPEL ali v izvajalni BPMN [5, 6].

Pri modeliranju poslovnih procesov je s pomočjo intervjujev potrebno odgovoriti na naslednja vprašanja:

1. Katere vloge so potrebne pri poslovnem procesu?
2. Katere aktivnosti so potrebne za celosten poslovni proces?
3. Po kakšnem vrstnem redu se aktivnosti izvajajo?
4. Kdo izvaja aktivnosti?
5. Kateri dokumenti se izmenjujejo?
6. Kako se lahko proces spremeni v prihodnosti zaradi dodajanja/odvzemanja dogodkov [2, 5]?

Model poslovnega procesa je sestavljen iz aktivnosti, ki se s funkcionalno dekompozicijo razčlenjujejo v podrobnejše aktivnosti procesa, dokler razčlenjene aktivnosti niso funkcionalno čiste oziroma atomarne. Torej jih ni več potrebno razčlenjevati naprej, kar zadostuje za modeliranje procesiranja transakcij. Atomarne aktivnosti so najmanjše funkcionalno čiste aktivnosti, ki procesirajo omejeno število vhodov v izhode. Po izvajanju morajo pustiti organizacijo v trdnem stanju in privesti do pomembnega poslovnega rezultata [10].

Pri modeliranju poslovnih procesov se najprej naredi model na najvišjem nivoju. Na najvišjem nivoju vedno obstaja samo en proces. Potem se poslovne procese po nivojih modelira toliko časa, dokler ni vsaka aktivnost procesa atomarna [16, 18]. Na višjih nivojih modela poslovnih procesov naj bi procesi kazali višjo kohezijo (kohezija prikaže, kolikšna je sorodnost med elementi na vsakem nivoju funkcionalne dekompozicije) in ohlapnejše sklapljanje, na nižjih nivojih modela poslovnih procesov pa naj bi kazali nižjo kohezijo in močnejše sklapljanje [10].

4.1 Notacija BPMN

Notacija BPMN (angl. Business Process Modeling Notation) je nastala leta 2004 pod okriljem skupine BPMI (angl. Business Process Management Initiative), katera se je leta 2006 združila s skupino OMG (angl. Object Management Group). Primarni cilj BPMN je zagotoviti standardno notacijo, ki jo z lahkoto razumejo vsi poslovni uporabniki, od poslovnega analitika, ki kreira začetne načrte procesov, do tehničnega razvijalca, ki je odgovoren za implementacijo tehnologije, ki bo izvajala poslovne procese, in tudi do poslovnih ljudi, ki upravljajo in nadzorujejo poslovne procese v organizaciji. Drugi cilj BPMN je razvoj notacije, ki bi se avtomatično preslikala v izvajalni jezik. BPMN torej do dobršne mere premosti vrzel med modelom poslovnega procesa in implementacijo. V obstoječih tehnologijah je bila ta vrzel prevelika [3, 6, 12].

BPMN diagrami izražajo potek izvajanja aktivnosti po korakih. Takšni modeli poslovnih procesov so znani tudi kot abstraktni poslovni procesi in lahko vsebujejo dragocene informacije brez vključevanja nepotrebnih podrobnosti [14]. BPMN obsega samo koncepte modeliranja poslovnih procesov, ne vključuje pa drugih tipov modeliranja, kot so modeliranje poslovnih pravil, podatkovnih ali informacijskih modelov ter strategij. BPMN priskrbi grafične simbole za modeliranje različnih aspektov poslovnih procesov oziroma za izdelavo diagramov poslovnih procesov (angl. Business Process Diagram). Ker BPMN priskrbi preprosto notacijo za modeliranje tako enostavnih kot kompleksnih procesov, se je grafični vidik notacije oblikoval na podlagi predhodnih notacij. Elementi so razvrščeni v posamezne kategorije, da uporabnik lažje prepozna določene elemente BPMN.

Glavni tipi elementov BPMN so:

1. elementi za opredelitev toka dogodkov (angl. Flow Objects),
2. povezovalni elementi (angl. Connecting Objects),
3. plavalne steze (angl. Swim-Lanes),
4. artefakti (angl. Artifacts).

Neodvisni podprocesi, ki sestavljajo nek proces, med seboj sodelujejo s pomočjo pošiljanja in prejemanja sporočil [6].

4.2 BPMN 2.0

Zadnja različica BPMN 2.0 je izšla januarja 2011 in predstavlja največji preskok med izdanimi različicami BPMN. Različica 2.0 temelji na različici 1.2 in razširja njene zmogljivosti v mnogih pogledih. Po novem BPMN 2.0 ni več samo notacija, ampak je tudi izvajalni jezik, kar je razvidno že iz spremembe pomena kratice BPMN. Do različice 1.2 je kratica pomenila *notacija modeliranja poslovnih procesov* (angl. Business Process Modeling Notation), v verziji 2.0 pa pomeni *modeliranje in notacija poslovnih procesov* (angl. Business Process Modeling and Notation). BPMN 2.0 je torej dobil nov metamodel in njegovo serializacijo XML [6]. BPMN 2.0 ima vse možnosti, da postane standard, ki bo v poslovnih procesih nadomestil BPEL, saj omogoča tako grafično modeliranje, kot semantiko izvajanja poslovnih procesov, kar pomeni, da poslovni analitik in tehnični razvijalec uporabljata en sam standard [8].

Pri modeliranju moramo brezpogojno upoštevati nekaj zahtev. Model poslovnega procesa mora biti sintaktično pravilen z ozirom na BPMN 2.0 specifikacijo notacije, kar pomeni, da se uporabljajo samo BPMN 2.0 konstrukti, in da so povezave ali razmerja med njimi pravilni [10].

Implementacija, ki ustvari in prikaže BPMN diagrame, naj bi se z ozirom na povezave in ostala diagramska razmerja med grafičnimi elementi prilagajala specifikacijam in omejitvam. Zagotavljala naj bi skladnost med povezavami in vrednostmi atributov, kjer so povezave, ki so dovoljene ali zahtevane, navedene kot pogojne in bazirajo na atributih skladnih konceptov [12].

Poglavitne novosti, ki jih prinaša BPMN 2.0 so:

– novi elementi modelov procesov,

– diagram koreografije (angl. Choreography Diagram),

– diagram pogovora (angl. Conversation Diagram),

– semantika izvajanja procesov (angl. Execution Semantics),

– tipi skladnosti (angl. Conformance Levels),

– standardizirana shema XML [6].

4.3 Elementi diagramov poslovnih procesov

Elementi so znotraj diagrama poslovnih procesov (angl. Business Process Diagram – BPD) kategorizirani tako, da uporabniki brez problemov razumejo diagrame. Na najvišjem nivoju modela poslovnih procesov naj bi za modeliranje bistvenih točk procesa zadostovali enostavni konstrukti, torej set osnovnih elementov. BPMN je bil ustvarjen z namenom, da bi podpiral modeliranje operativnih procesov in zato priskrbi nekaj elementov, ki ustvarijo nivojsko zgradbo znotraj operativnih procesov: proces na najvišjem nivoju (angl. Top-Level Process), podproces (angl. Subprocess) znotraj tega procesa in opravilo (angl. Task) [2, 16, 18]. Z izpopolnjevanjem osnovnih elementov raste kompleksnost, še vedno pa se ohranja standardna predstavitev diagramov poslovnih procesov.

Osnovni elementi so razdeljeni v štiri kategorije, vsaka pa vsebuje specifičen set elementov:

1. *Elementi za opredelitev toka dogodkov* vsebujejo najbolj pomembne elemente v BPD in se uporabljajo za predstavitev osnovnega obnašanja vsakega procesa. Ti elementi so dogodek (angl. Event), aktivnost (angl. Activity) in prehod (angl. Gateway).

2. *Povezovalni elementi* se uporabljajo za medsebojno povezavo elementov za opredelitev toka dogodkov. Ti elementi so tok zaporedja (angl. Sequence Flow), tok sporočil (angl. Message Flow) in asociacija (angl. Association).

3. *Plavalne steze* se uporabljajo za organizacijo aktivnosti v posamezne vizualne kategorije, tako da ilustrirajo različne odgovornosti ali funkcijske zmožnosti. Med plavalne steze spadata elementa bazen (angl. Pool) in steza (angl. Lane).

4. *Artefakti* se uporabljajo za prikaz dodatnih informacij o poslovnem procesu. Ti elementi vsebujejo podatkovni objekt (angl. Data Object), tekstovni komentar (angl. Annotation) in skupino (angl. Group). Poleg osnovnih artefaktov obstajajo še lastni artefakti (angl. Custom Artifacts), ki se lahko poljubno definirajo [2].

Najpomembnejši **novi elementi** modelov procesov so: nove vrste dogodkov (eskalacija, paralelni sestavljeni dogodek, neprekinjajoči dogodki), opcijski dogodkovni podprocesi, nove vrste prehodov (ekskluzivni in paralelni dogodkovni prehod), grafične oznake za različne vrste aktivnosti in novi podatkovni objekti (podatkovna baza, zbirka podatkov). Eden izmed razlogov za vpeljavo novih elementov je nova podpora izvajalnim poslovnim procesom BPMN 2.0. S tem se je kompleksnost notacije in na njej temelječih modelov še bolj obogatila (BPMN 2.0 vsebuje več kot sto elementov), saj ponuja bolj proceduralno in sporočilno usmerjeno (angl. Message-Level) obnašanje kot prej [10].

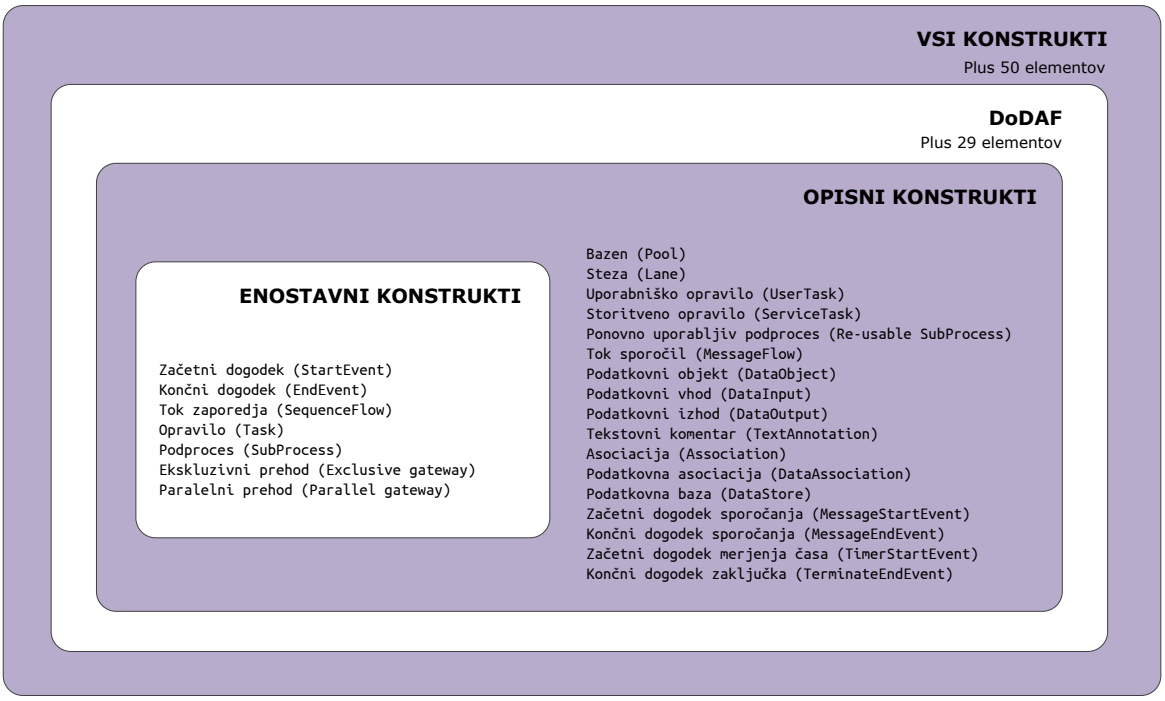
Zaradi povečane kompleksnosti notacije so elementi razvrščeni v tri skupine:

– *Skupina opisnih (angl. Descriptive) elementov* vključuje vizualne elemente in atribute, ki so namenjeni visokonivojskemu modeliranju in dokumentiranju poslovnih procesov. Ti elementi so v večji meri skladni z drugimi notacijami, kot so na primer diagrami poteka, in so zadostni poslovnemu analitiku. V skupino opisnih elementov spadajo preprost začetni in končni dogodek, bazeni in steze, aktivnosti, tok zaporedja in sporočil ter ekskluzivni in paralelni prehod.

– *Skupina analitičnih (angl. Analytic) elementov* vsebuje vse opisne elemente ter dodatne vizualne elemente, ki omogočajo natančno modeliranje poslovnih procesov pri analiziranju, nadziranju in predstavitvi procesov, ki se izvajajo na procesnih strojih. V skupini opisnih in skupini analitičnih elementov je fokus usmerjen na vizualne elemente in minimalno podmnožico podpirajočih atributov/elementov. V skupino analitičnih elementov spadajo dogodki, ki niso zajeti v skupini opisnih elementov, pogojni tokovi, izjeme in vse vrste prehodov. Tako kot skupina opisnih elementov, se tudi skupina analitičnih elementov nanaša na neizvajalne modele in vsebuje samo informacije, ki so vidne v diagramih. V takšnih diagramih ni opisa podatkov, pogojev prehodov ali sporočil, saj to spada v domeno izvajalnega poslovnega procesa.

– *Skupina izvajalnih (angl. Executable) elementov* vsebuje še preostale elemente, ki so potrebni pri modeliranju poslovnih procesov. Fokus je usmerjen na izvajanje modelov procesov, kjer se slednji načrtajo do natančnosti, ki zadostuje za neposredno izvajanje modelov na procesnih strojih. Zaradi tega se določeni elementi iz skupine analitičnih elementov razširjajo z dodatnimi atributi, da zadostijo pogojem skupine izvajalnih elementov [6, 12].

Poleg zgoraj omenjenega kategoriziranja elementov obstaja še ena kategorizacija elementov, ki so jo predstavili pri WfMC (Workflow Management Coalition). Slednja je razdelila elemente v štiri različne kategorije, kjer se enostavni in opisni konstrukti lahko primerjajo s skupino opisnih elementov, DoDAF (arhitekturni okvir Ministrstva za obrambo ZDA) konstrukti se lahko primerjajo s skupino analitičnih elementov, skupina izvajalnih elementov pa se lahko primerja z vsemi preostalimi konstrukti v BPMN 2.0 [18].



Slika 4.1: **Kategorizacija BPMN 2.0 konstruktov s strani WfMC [18].**

4.3.1 Dogodki

V BPMN 2.0 je obseg dogodkov, ki jih notacija pokriva, obširen, zato je notacija BPMN tudi primernejša za modeliranje poslovnih procesov kot njena konkurenca. Dogodki predstavljajo različna stanja, ki so pomembna v poslovnem procesu. Običajno imajo vlogo sprožilca ali čakajo na določeno akcijo. Glede na to, kje se dogodek pojavi v procesu, jih lahko razdelimo v tri skupine:

– *Začetni* (angl. Start) dogodek nakazuje začetek procesa.

– *Vmesni* (angl. Intermediate) dogodek se pojavi med procesom, kot alternativa lahko tudi odlaša z izvajanjem procesa.

– *Končni* (angl. End) dogodek nakazuje konec procesa.

Dogodki so predstavljeni s krogom. Glede na tip dogodka je v sredini kroga določena ikona [2].

Vrsta dogodka / *Tip dogodka*	**Začetni dogodki**	**Vmesni dogodki**			**Končni dogodki**
	Lovljenje dogodka	Lovljenje dogodka	Prekinjajoč mejni dogodek, lovljenje dogodka	Neprekinjajoč mejni dogodek, lovljenje dogodka	Proženje dogodka	Proženje dogodka
Začetni dogodek: Netipiziran dogodek, ki naznanja začetno točko, spremembo stanja ali zaključno stanje.						
Sporočilo: Prejemanje in pošiljanje sporočil.						
Časovni dogodek: Ciklični časovni dogodek, časovni interval, trenutek ali iztečni čas (timeout).						
Eskalacija: Prenos sporočila iz podprocesa v matični proces.						
Pogoj: Reakcija na izpolnitev določenega pogoja.						
Povezava: Konektor, ki prikazuje povezavo k delu procesa, ki je na drugem listu ali zaslonu (Off-page connector). Dva skladna dogodka povezave sta enakovredna toku zaporedja.						
Napaka: Lovljenje in proženje kritičnih napak.						
Preklic: Reakcija na preklicane transakcije ali sprožene razveljavitve.						
Kompenzacija: Obravnavanje ali proženje kompenzacije. Ti dogodki so močno povezani s transakcijami.						
Signal: Signaliziranje različnih procesov. Sprožen signal je lahko večkrat ulovljiv.						
Sestavljen dogodek: Lovljenje enega izmed množice dogodkov. Proženje vseh definiranih dogodkov.						
Paralelni sestavljeni dogodek: Lovljenje vseh dogodkov iz množice paralelnih dogodkov.						
Končni dogodek: Proženje takojšnjega zaključka celotnega procesa.						

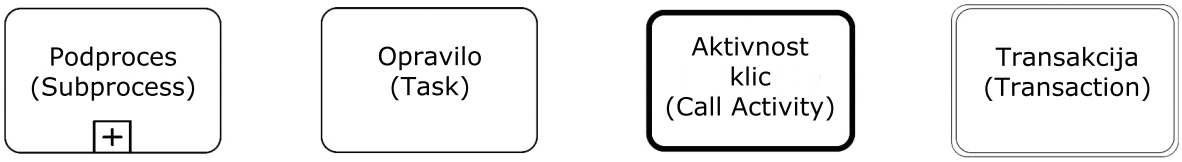
Slika 4.2: **Tipi dogodkov v BPMN 2.0 [21].**

4.3.2 Aktivnosti

Aktivnost označuje posamezno enoto dela poslovnega procesa. Serija zaporednih aktivnosti z jasnimi začetnimi in končnimi točkami predstavlja proces. V diagramu poslovnih procesov je predstavitev procesov hierarhična, torej ima lahko določen proces več podprocesov. Aktivnosti so lahko atomarne ali sestavljene. Atomarne predstavljajo eno samo akcijo in se ne morejo nadalje razstavljati. Sestavljene aktivnosti ali podprocesi potrebujejo še nekaj podaktivnosti za dokončanje dela [2].

Pri dobrem modeliranju poslovnih procesov naj bi bile funkcije in podfunkcije, ki predstavljajo aktivnosti v teku brez začetnih in končnih točk, poimenovane s samostalniki ali samostalniškimi frazami. Procesi in podprocesi, ki predstavljajo aktivnosti z začetnimi in končnimi točkami, naj bi bili poimenovani z glagoli ali glagolskimi frazami [10].

BPMN 2.0 podpira štiri različne tipe aktivnosti: podproces, opravilo, transakcija in aktivnost klic.



Slika 4.3: **Različni tipi aktivnosti v BPMN 2.0 [21].**

Notacija za aktivnost je predstavljena z zaokroženim pravokotnikom. Spreminja se skladno s tipom aktivnosti in s tem, kaj se zgodi znotraj aktivnosti. Aktivnost je lahko označena s simbolom oziroma znakom, ki izpopolni semantiko izvajanja. Znaki za označevanje aktivnosti so naslednji:

– Z *znakom za zaporednost večih instanc* se lahko realizira zanka *for* z *n* iteracijami.

– Z *znakom za Adhoc* se označi podproces, ki je sestavljen iz množice aktivnosti, ki med seboj niso povezane s tokom zaporedja oziroma ni mogoče definirati vrstnega reda izvajanja aktivnosti. Vrstni red aktivnosti določajo izvajalci posameznih aktivnosti med izvajanjem, zato se lahko pojavijo v kateremkoli vrstnem zaporedju.

– Z *znakom za zanko* se prikaže, da gre podproces čez več iteracij med izvajanjem procesa. Z znakom za zanko se torej realizira zanka *while* ali zanka *repeat-until*.

– Z *znakom za kompenzacijo* podproces namiguje, da ima skupek kompenziranih aktivnosti.



Slika 4.4: **Znaki za označevanje aktivnosti [21].**

Podproces je v starševskem procesu označen s plus znakom znotraj zaokroženega pravokotnika. Temu se lahko reče tudi zloženi proces (angl. Collapsed Process), saj znak plus označuje, da je dana aktivnost temeljiteje izdelana v ločenem diagramu. Zloženi proces je lahko namesto v ločenem diagramu predstavljen kot razširjeni proces (angl. Expanded Process), kjer je znotraj velikega zaokroženega pravokotnika izdelan celoten podproces. S podprocesi se model poslovnih procesov razbije na manjše in zato lažje obvladljive dele. Podproces lahko znak za podproces uporablja v konjunkciji s katerimkoli drugim znakom za označevanje aktivnosti.

Aktivnost klic se uporablja kot referenca na globalno definirane procese ali opravila, s čimer se pospešuje ponovna uporaba aktivnosti. Globalni proces ali opravilo je izven obsega definicije procesa, medtem ko je podproces vgrajen znotraj originalne definicije procesa.

Opravilo je atomarna aktivnost v poslovnem procesu. Običajno je opravilo delo, ki ga opravi aplikacija ali uporabnik med izvajanjem procesa. Opravila uporabljajo naslednje znake za označevanje aktivnosti: znak za zanko, znak za kompenzacijo in znak za paralelnost večih instanc. V BPMN 2.0 so opravila olepšana z ikonami, ki predstavljajo specifičen tip opravil. Z ikonami se lažje razume, kaj posamezno opravilo predstavlja. Tipi opravil so naslednji:

- *Storitveno opravilo* je avtomatizirano opravilo, ki prikliče neko vrsto storitve. To je torej poziv neke poslovne funkcije preko vmesnika storitev (angl. Service Interface). Vsako delo, ki se izvede izven procesnega stroja, naj bi bilo prikazano z uporabo storitvenega opravila.

- *Opravilo pošiljanje sporočila* je opravilo, ki pošlje sporočilo zunanjemu udeležencu procesa.

- *Opravilo prejem sporočila* je opravilo, ki pričakuje prejem določenega sporočila od zunanjega udeleženca procesa.

- *Uporabniško opravilo* je opravilo, ki ga izvede uporabnik s pomočjo ustrezne aplika-

cije. Proces čaka toliko časa, dokler uporabnik ne zaključi opravila. Z uporabniškim opravilom je običajno povezana informacija o vlogi in izkušenosti, ki naj bi jo imel določen uporabnik. To opravilo je dejansko polavtomatizirano opravilo.

– *Skriptno opravilo* je avtomatizirano opravilo, ki požene skripto, in je končano, ko se skripta izvede do konca. Skriptni jezik je odvisen od implementacije procesnega stroja.

– *Ročno opravilo* je opravilo, ki se izvede brez pomoči aplikacije ali procesnega stroja.

– *Opravilo poslovno pravilo* omogoča klic sistema za izvajanje poslovnih opravil (angl. Business Rule Management System - BRMS). Je le mehanizem, ki deluje kot vhod v stroj poslovnih pravil (angl. Business Rule Engine), in ki dobi izračune, ki jih priskrbi stroj poslovnih pravil. V modelu poslovnih procesov se ne izvede nobena akcija. Opravilo poslovno pravilo vrne le izračune pravil, katere lahko uporabi naknadna procesna logika [2, 20–22].

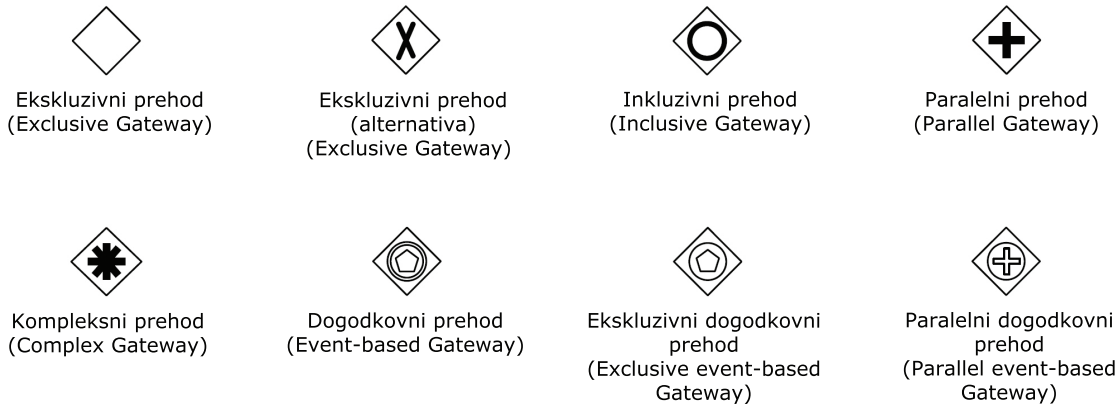


Slika 4.5: **Različni tipi opravil v BPMN 2.0 [21].**

Transakcija je specifičen podproces, katerega obnašanje kontrolira transakcijski protokol. Največkrat se uporablja protokol modela ACID, ki ima naslednje lastnosti: atomarnost (pri procesiranju se izvede vse ali nič), konsistentnost (rezultat sklopa operacij oziroma transakcije je konsistentna sprememba v stanju), izolacija (transakcije se izvajajo neodvisno ena od druge, zato delni rezultati ene transakcije ne smejo biti vidni drugim transakcijam) in trajnost (učinek potrjene transakcije je trajen). Zaradi omenjenih lastnosti transakcij naj bi procesirane operacije, ki so del transakcije, ob vsakem ponovnem izvajanju proizvedle enak rezultat. Če med izvajanjem transakcije pride do prekinitve, mora transakcija vrniti podatke v stanje pred izvedbo podprocesa [10, 21].

4.3.3 Prehodi

Prehodi se uporabljajo za kontrolo medsebojnega delovanja tokov ob vejitvi ali združevanju. V osnovi je prehod neke vrste odločitveno stekališče, kjer se določen tok odloči za razcepitev v dve ali več aktivnosti, ali pa se odloči za združitev dveh ali več aktivnosti v eno samo. Prehodi se lahko interpretirajo tudi kot *if-then* ali *switch* konstrukti. Obstaja tudi možnost definiranja privzete vejitve v primeru, če noben od pogojev ni resničen. V BPMN 2.0 ima prehod obliko diamanta z določeno ikono na sredini, ki predstavlja specifičen prehod.



Slika 4.6: **Različni tipi prehodov v BPMN 2.0 [21].**

Tipi prehodov so naslednji:

– *Ekskluzivni prehod* med več možnimi potmi izbere eno samo. Če ima več tokov resnično vrednost pogoja (*true*), potem ekskluzivni prehod izbere tisti tok zaporedja, ki je v XML definiran kot prvi, ostale pa ignorira. Ekskluzivni prehod ima na razpolago dva prikaza notacije, vendar je priporočljivo uporabljati notacijo z ikono na sredini diamanta.

– *Inkluzivni prehod* omogoča vejitev toka v več paralelnih poti izvajanja ali združevanje večih poti v eno. Pri vejitvi toka zaporedja spusti prehod vsako odhajajočo pot, katere pogoj je resničen. Pri združevanju poti prehod čaka toliko časa, dokler ne prejme vsake pričakovane vhodne poti.

– *Paralelni prehod* se uporablja za kreiranje in sinhronizacijo paralelnih poti. Pri vejitvi toka zaporedja se vse poti oziroma aktivnosti aktivirajo istočasno. Pri združevanju poti prehod čaka toliko časa, dokler se ne končajo vse vhodne aktivnosti.

– *Kompleksni prehod* bazira na kompleksnih izrazih, ki jih definirajo uporabniki s kombinacijo obnašanja vejitve in združevanja. Obnašanje je specificirano v aktivacijskem pogoju (angl. Activation Condition) in v izrazu prehoda. Uporablja se takrat, ko ostali prehodi ne morejo priskrbeti zahtevanih rezultatov.

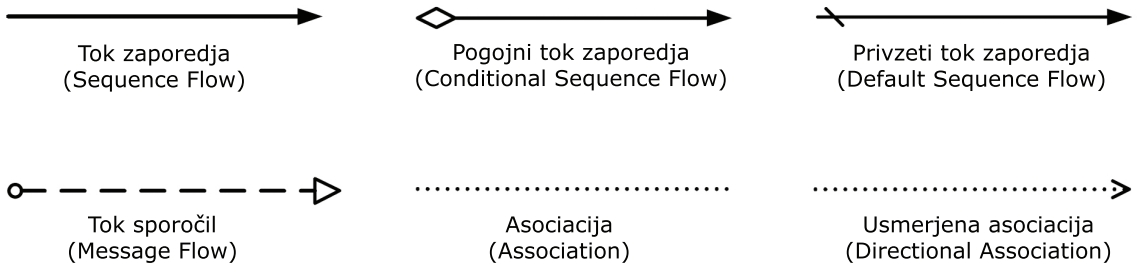
– *Dogodkovni prehod* bazira na specifičnem dogodku in spusti čez vejitev le tisti tok zaporedja, ki vsebuje prožilni dogodek, vse ostale pa ignorira.

– *Ekskluzivni dogodkovni prehod* je zelo podoben navadnemu dogodkovnemu prehodu. Razlika je v tem, da se navadni dogodkovni prehod lahko uporabi kjerkoli v diagramu poslovnih procesov, ekskluzivni dogodkovni prehod pa le na začetku diagrama poslovnih procesov. Poleg tega se pri navadnem dogodkovnem prehodu proces začne (instancira in začne) in nato čaka, da se pojavi določen dogodek. Pri ekskluzivnem dogodkovnem prehodu se proces instancira šele potem, ko se pojavi določen dogodek.

– *Paralelni dogodkovni prehod* je podoben ekskluzivnemu dogodkovnemu prehodu, le da se morajo v tem primeru na prehodu pojaviti vsi definirani dogodki, kateri so potrebni za normalno zaključitev procesa, da se proces lahko začne [2, 12, 21, 22].

4.3.4 Povezave

Povezave povezujejo elemente za opredelitev toka dogodkov, plavalne steze ali artefakte.



Slika 4.7: **Različni tipi povezav v BPMN 2.0 [12].**

Tok zaporedja specificira vrstni red izvajanja aktivnosti in ne sme prečkati meje bazena. Za označevanje privzetih poti se uporablja **privzeti tok zaporedja**. **Pogojni tok zaporedja** se uporablja za vrednotenje pogoja, še preden se tok premakne na naslednjo aktivnost.

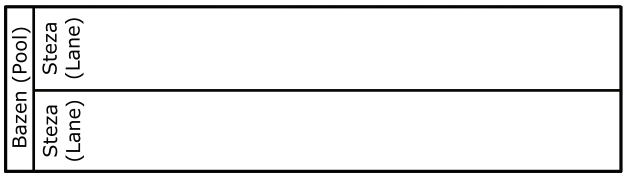
Tok sporočil predstavlja interakcijo med dvema udeležencema v poslovnem procesu, ki sta predstavljena z dvema različnima bazenoma.

Asociacija je specifičen tip povezav, ki se uporablja za vezavo artefaktov na elemente v diagramih poslovnih procesov. Asociacija je lahko tudi usmerjena [2, 21].

4.3.5 Plavalne steze

S plavalnimi stezami so v BPMN prikazani hierarhični organizacijski vidiki. Vsak **bazen** specificira konkretno organizacijo, lahko pa označuje tudi ključno vlogo v organizaciji. Vsak poslovni proces se nahaja v svojem bazenu in zato posledično vsak proces izvede ena sama organizacija.

Steza se uporablja za razdelitev bazena na več delov in predstavlja entiteto znotraj organizacije. Z razdelitvijo bazena na več stez se bolj jasno prikaže, kateri oddelki ali vloge v organizaciji so odgovorni za posamezne aktivnosti [2, 21].



Slika 4.8: **Predstavitev bazenov in stez v BPMN 2.0 [12].**

4.3.6 Artefakti

Artefakti se uporabljajo za prikaz dodatnih informacij o poslovnem procesu in služijo izključno namenu informiranja, torej ne vplivajo na semantiko izvajanja procesa.

Podatkovni objekt poda dodatne informacije o delovanju procesa. To so lahko podrobnosti dokumentov, podatkov ali objektov, ki se uporabljajo pri prenosu med aktivnostmi.

Skupina se običajno uporablja za grupiranje določenih aktivnosti za potrebe dokumentacije in analize. S skupino se lahko označi tudi porazdeljeno transakcijo, ki sega čez več bazenov.

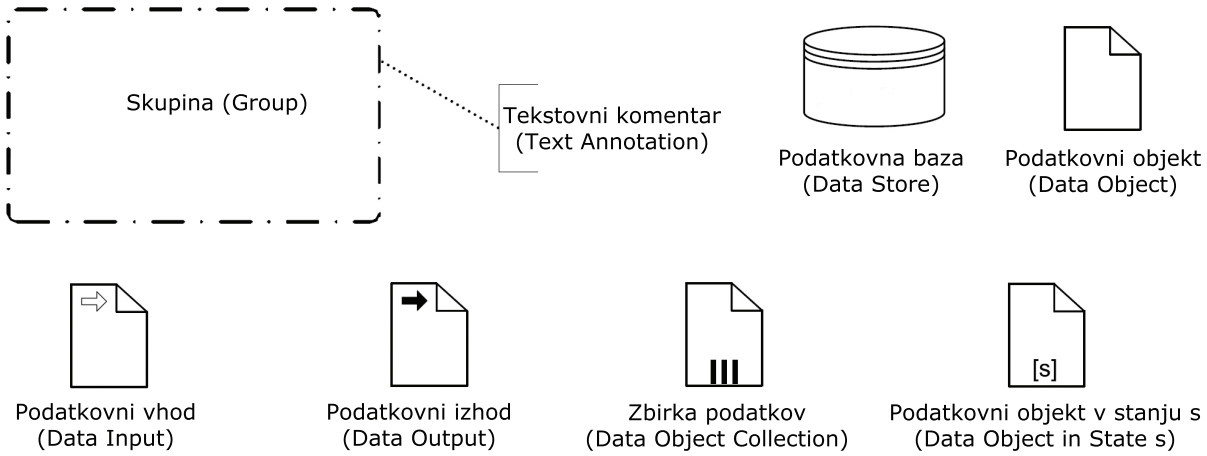
Tekstovni komentar se uporablja za dodajanje dodatnih tekstovnih informacij, ki so v pomoč pri branju diagrama. Tekstovni komentar se z asociacijo lahko poveže na katerikoli objekt v diagramu.

Procesi običajno uporabljajo podatke, ki se ustvarijo pred začetkom procesa. Za prikaz takšnih podatkov se uporablja **podatkovni vhod**.

Če se med izvajanje procesa ustvarijo izhodni podatkovni objekti, katere bodo uporabili drugi procesi, potem se za prikaz takšnih podatkov uporablja **podatkovni izhod**.

Podatkovna baza predstavlja informacijski sistem, ki hrani podatke, ali skladišče podatkov.

Zbirka podatkov ne predstavlja individualnega podatkovnega objekta, ampak skupek podatkov, kot je na primer seznam podatkov [2, 21].



Slika 4.9: **Artefakti v BPMN 2.0 [12].**

4.4 Sodelovanje med procesi

Ker BPMS platforme prevzamejo obseg nadzora, mora biti model diagrama sodelovanja prikazan kot zloženi bazeni (angl. Collapsed Pools). Zaradi zahteve po natančnem prikazu različnih področij odgovornosti morajo biti aktivnosti ločene s primernimi bazeni za udeležence in stezami znotraj bazena za vloge. S tem je zagotovljeno, da so aktivnosti, ki predstavljajo transakcije, ustrezno ločene med seboj, in da vsako izvajalno operacijo izvede specifičen vir oziroma resurs. Diagram poslovnega procesa mora biti znotraj lastnega bazena platforme BPMS, ki (kot glavni udeleženec) vsebuje steze za procesni stroj in katerokoli uporabniško vlogo, ki sodeluje z aplikacijo [10].

Z BPMN se lahko izražajo poslovni procesi med večimi organizacijami, ki sodelujejo med seboj. Znotraj procesa se uporablja tok zaporedja, ki definira zaporedje izvajanja aktivnosti. Znotraj organizacije se ustvarijo postopki in pravila, ki zagotavljajo, da so

aktivnosti izvedene tako, kot je specificirano v modelu poslovnih procesov. Drugačna zgodba se odvija pri poslovnih partnerjih, s katerimi posamezni procesi sodelujejo. Poslovni partnerji med seboj običajno ne razkrivajo vrstnega reda izvajanja aktivnosti v svojih procesih, zato se za medsebojno sodelovanje med procesi uporablja izključno tok sporočil. Ko poslovni partner dobi sporočilo, točno ve, kako bo to sporočilo vplivalo na njegove poslovne procese, saj je seznanjen z zgradbo svojih procesov.

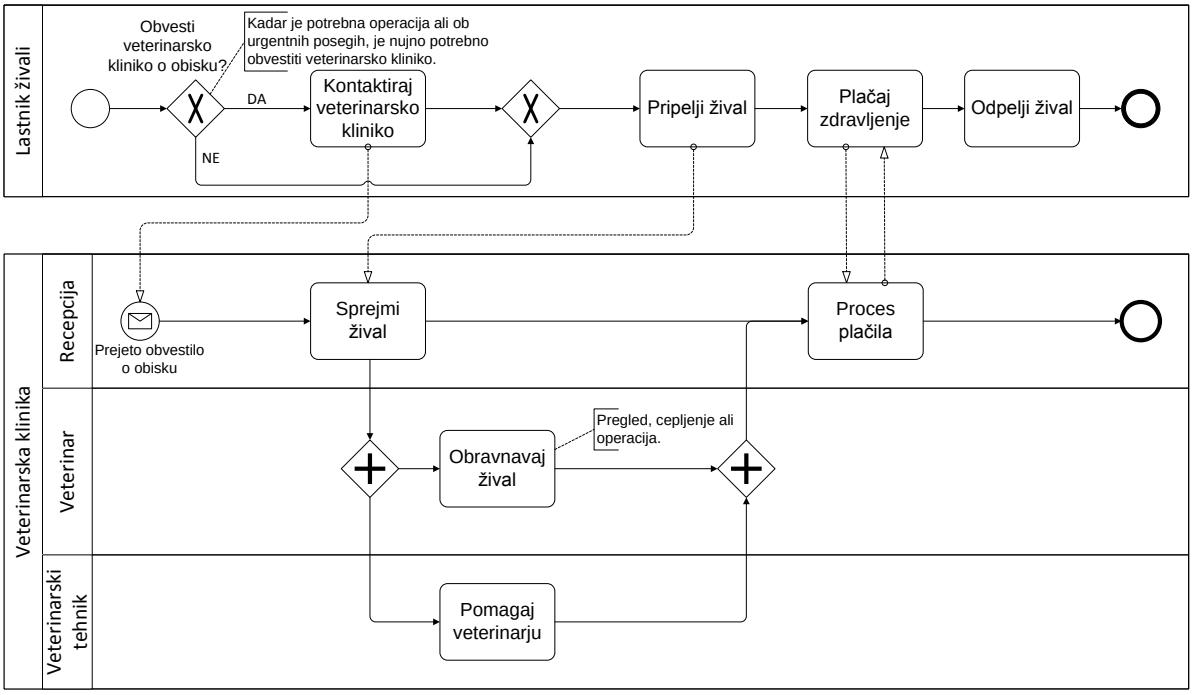
Pri najbolj abstraktnem načinu sodelovanja med poslovnimi partnerji organizacija pri poslovnih partnerjih vidi samo vloge in tokove sporočil med procesi. Nobena informacija o notranjih procesih poslovnih partnerjev ni na razpolago in prav tako nima nobenega pomena vrstni red tokov sporočil med organizacijami. Takšne vloge poslovnih partnerjev so predstavljene s praznimi bazeni, ki se imenujejo bazeni črni zaboj (angl. Black Box Pools). Diagrami sodelovanja, ki vsebujejo bazene črni zaboj, zagotavljajo visokonivojski pogled na procese.

V bazenih, ki predstavljajo zunanjega poslovnega partnerja, je lahko vidno zunanje komunikacijsko obnašanje procesa, ki teče v organizaciji poslovnega partnerja. Takšni bazeni so poimenovani bazeni beli zaboj (angl. White Box Pool). Ena prednost tega je sledenje principu skrivanja nepotrebnih informacij, tako da kompleksnost notranjega poslovnega procesa ne poveča kompleksnosti celotnega procesa. Druga prednost tega je, da podjetje ne razkriva svojih notranjih procesov zunanjemu svetu, saj so pomembno premoženje podjetja. Ker se v takšnih primerih iz zunanjega sveta opazi samo komunikacijsko obnašanje procesa, je takšen proces, ki je omejen le na svoje komunikacijske aktivnosti, poimenovan **javni proces** (angl. Public Process). Včasih poslovni partner razkrije svoj celoten notranji proces. Slednje je storjeno z dodajanjem aktivnosti in v nekaterih primerih tudi z dodajanjem kontrolnih struktur (angl. Control Structures) v javni proces poslovnega partnerja. Takšen proces je poimenovan **privatni proces** (angl. Private Process) in vsebuje vse aktivnosti, ki so sprejete znotraj podjetja. Privatni proces torej realizira orkestracijo procesa [21].

4.4.1 Primer sodelovanja obiska veterinarske klinike

Kot praktičen primer je z diagrami sodelovanja BPMN prikazan model obiska veterinarske klinike za male živali. Prvi korak pri modeliranju procesov je izdelava enostavnega visokonivojskega diagrama sodelovanja, ki ga večinoma sestavljajo le enostavni konstrukti BPMN. Visokonivojski diagram sodelovanja je najabstraktnejši med vsemi diagrami sodelovanja, ki se izdelajo pri modeliranju poslovnih procesov. Takšen diagram je potreben za osnovno razumevanje poslovnih procesov.

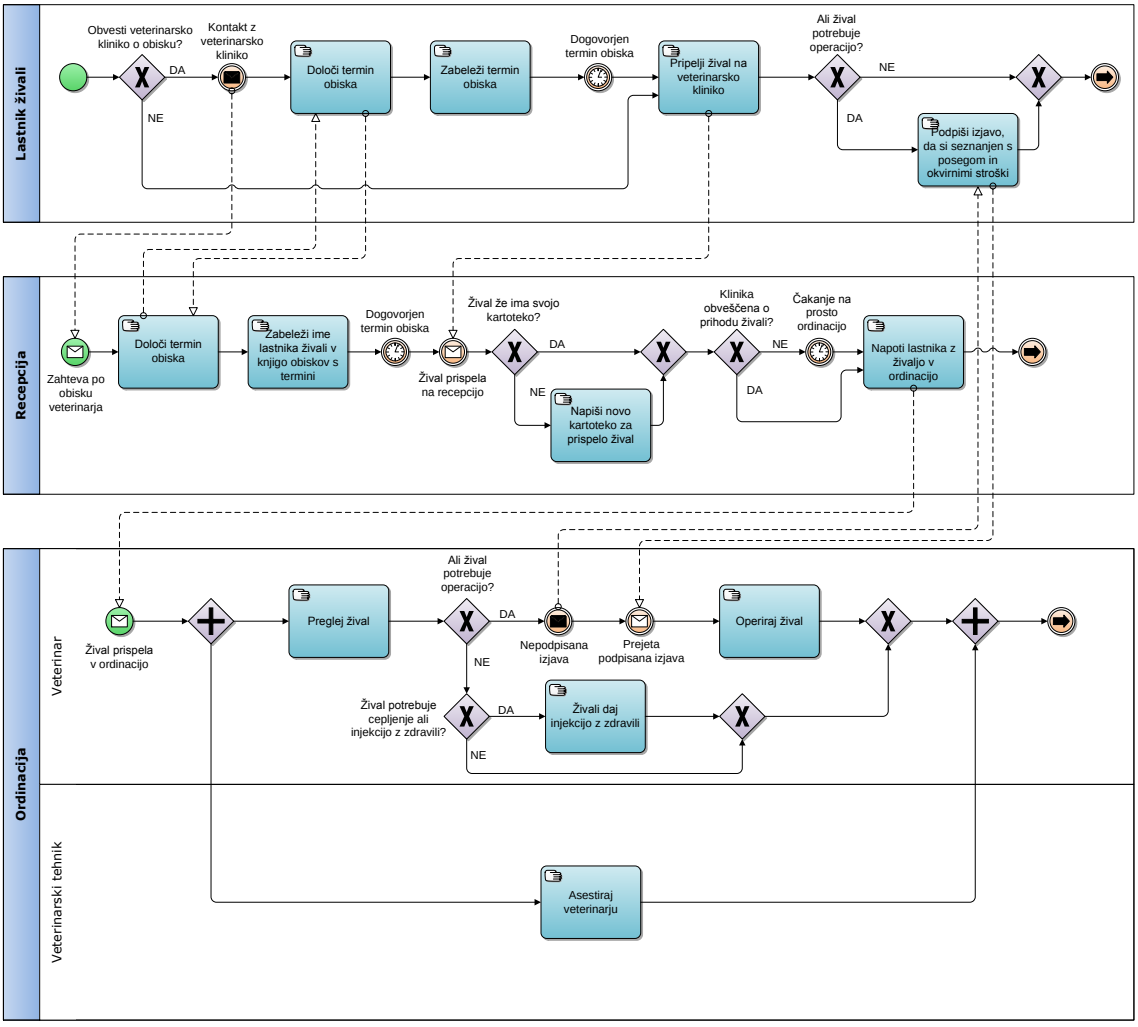
Kot je prikazano na sliki 4.10, se model prične z ugotovitvijo, da domača žival potrebuje veterinarsko oskrbo. Lastnik živali se lahko po telefonu ali elektronski pošti predhodno zmeni z receptorjem veterinarske klinike za točno uro obiska. Lastnik domače živali se lahko odloči tudi za to, da svojega ljubljenčka odpelje naravnost na veterinarsko kliniko. Kadar domača žival potrebuje operacijo ali urgenten poseg, je nujno potrebno obvestiti veterinarsko kliniko o obisku, da se lahko določi najbolj primeren termin, oziroma da se ob urgentnih primerih lahko osebje veterinarske klinike pravočasno organizira, tako da ob prihodu na veterinarsko kliniko živalski pacient dobi takojšnjo oskrbo. Ko na kliniko prispeta domači ljubljenček in njegov lastnik, najprej na recepciji veterinarske klinike sprejmejo žival, potem pa lastnika in njegovo žival pospremijo v ordinacijo, kjer veterinar in veterinarski tehnik pregledata žival ter jo ustrezno oskrbita. Ko je domača žival pregledana in veterinarsko oskrbljena, lastnik živali plača zdravljenje in odpelje svojega ljubljenčka domov. V primeru da domača žival potrebuje operacijo, lastnik pusti ljubljenčka na kliniki in ga odpelje domov šele potem, ko minejo kritične ure, včasih tudi dnevi, po operaciji. Z odhodom domov se poslovni proces konča.



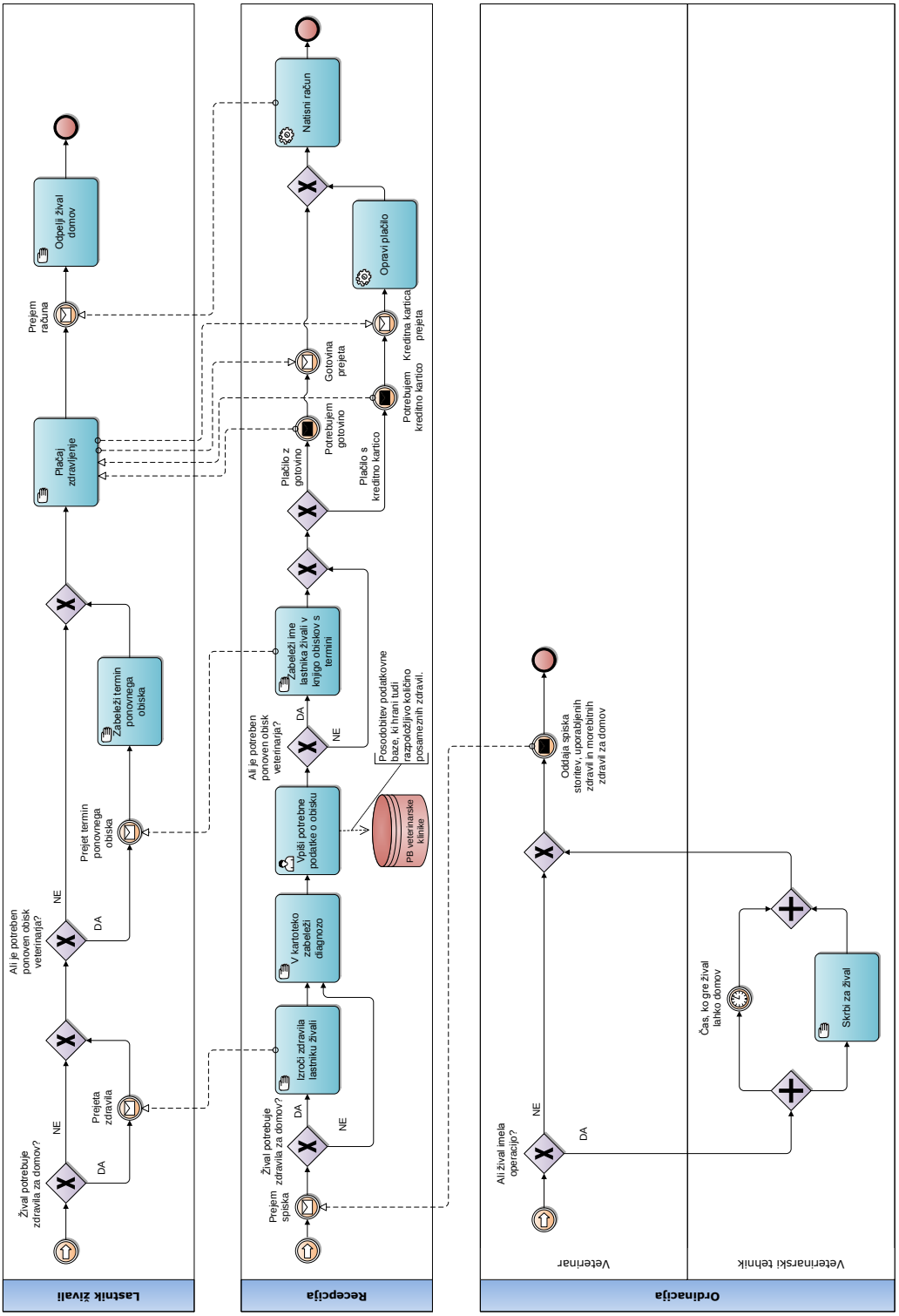
Slika 4.10: **Model obiska veterinarske klinike z vidika visokonivojskega diagrama sodelovanja BPMN.**

Naslednji korak pri modeliranju je izdelava podrobnega diagrama sodelovanja BPMN. Pri obširnih modelih poslovnih procesov je potrebno modeliranje večih nivojev, v primeru

omenjenega modela obiska veterinarske klinike pa je dovolj zgolj izdelava enega samega podrobnega diagrama sodelovanja BPMN. Podroben diagram sodelovanja BPMN je lahko sestavljen iz vseh konstruktov BPMN 2.0. Prikazan je veliko bolj podroben potek kot pri visokonivojskem diagramu BPMN. Posamezne aktivnosti so prikazane z ustreznimi ikonami, ki ponazarjajo določene vrste opravil, medtem ko imajo pri visokonivojskem diagramu sodelovanja BPMN vse aktivnosti enotno ikono.



Slika 4.11: **Prvi del modela obiska veterinarske klinike z vidika podrobnega diagrama sodelovanja BPMN.**



Slika 4.12: **Drugi del modela obiska veterinarske klinike z vidika podrobnega diagrama sodelovanja BPMN.**

4.5 Diagram koreografije in diagram pogovora

BPMN 2.0 poleg modelov (interni procesi, sodelovanje, javni procesi), ki so jih podpirale predhodne verzije BPMN, podpira tudi model koreografije in model pogovora.

4.5.1 Diagram koreografije

Diagram koreografije BPMN je sestavljen iz aktivnosti, ki se med seboj povezujejo s tokom zaporedja. Te aktivnosti so sestavljene iz ene ali več interakcij med udeleženci. Interakcije so pogosto opisane kot vzorec za izmenjavo sporočil (angl. Message Exchange Pattern) oziroma MEP-i. Nekateri MEP-i vsebujejo samo eno sporočilo, večina MEP-ov pa vsebuje dve sporočili, ki predstavljata zahtevo in odgovor. Obstaja celo kompleksnejši MEP, ki vsebuje tudi sporočilo napake.

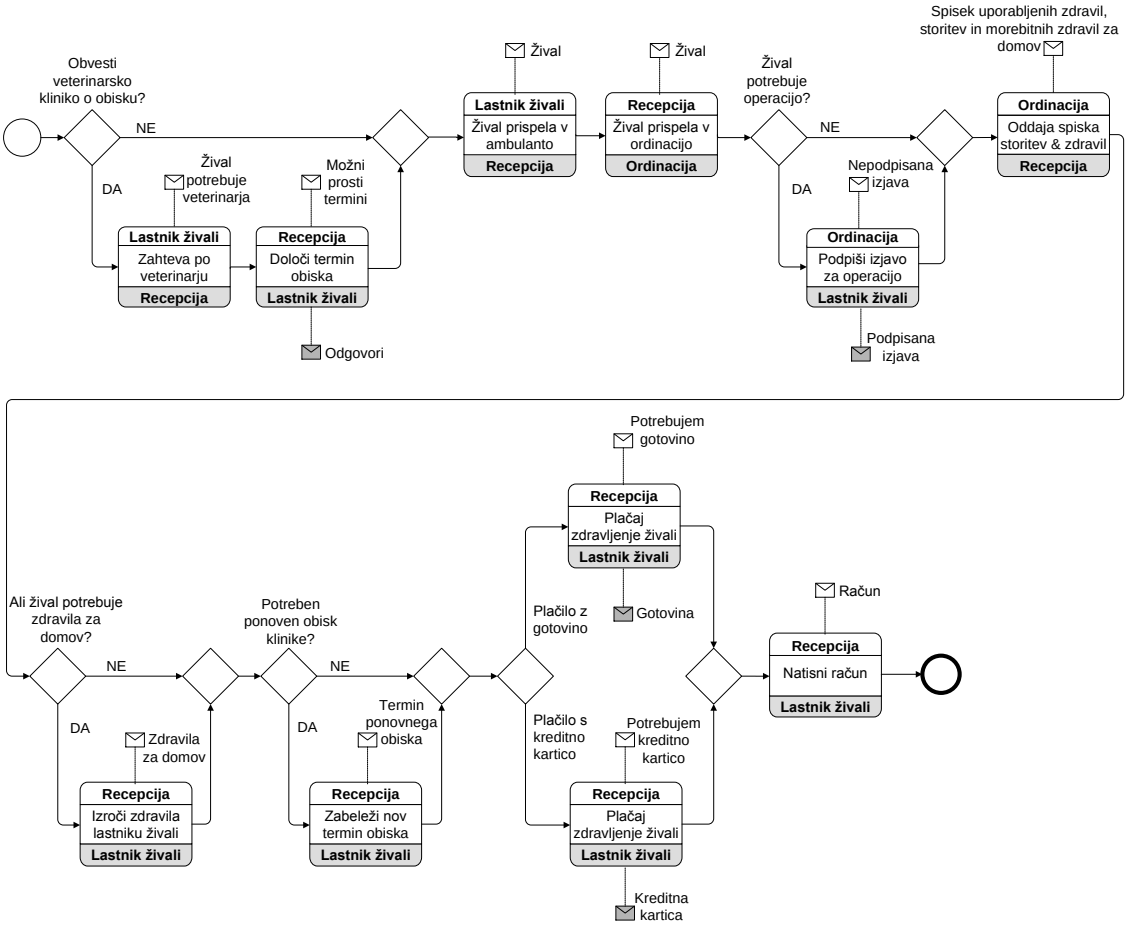
V BPMN 2.0 so koreografije zasnovane tako, da dovoljujejo samostoječe in skalabilne modele interakcij udeležencev. Odkar BPMN omogoča še druge poglede na modele poslovnih procesov, se koreografije načrtujejo tako, da se prilegajo znotraj diagramov sodelovanja in s tem prikažejo razmerje med koreografijo in orkestracijo procesov.

Ključ do razumevanja in uporabe koreografij v BPMN-ju je v njihovem razmerju do bazenov. Bazeni so grafične predstavitve udeležencev, koreografija pa definira zaporedje interakcij med udeleženci. Vsak korak oziroma aktivnost v koreografiji vključuje dva ali več udeležencev. To pomeni, da je v BPMN-ju koreografija definirana zunaj vsakega posameznega bazena in obstaja vsaj med dvema bazenoma. Vsak udeleženec predvideva status koreografije na podlagi sporočil, ki so bila prejeta in poslana med udeleženci. Če sta v koreografiji samo dva udeleženca, se oba zavedata, kdo je odgovoren za pošiljanje naslednjega sporočila. Če so v koreografiji trije ali več udeležencev, potem je pomembno, da so aktivnosti koreografije zvrščene v takem zaporedju, da udeleženci vedo, kdaj so odgovorni za iniciacijo interakcij.

Za potrebe koreografije je v BPMN 2.0 predstavljen nov element *opravilo koreografije* (angl. Choreography Task). Predstavlja atomarno enoto oziroma aktivnost znotraj koreografije in definira en MEP. Poleg opravila koreografije so del diagrama koreografije lahko še naslednji elementi: podopravilo koreografije (angl. Sub-Choreography), ki je lahko zloženo (angl. Collapsed) ali razširjeno (angl. Expanded), globalno opravilo koreografije (angl. Global Choreography Task), prehodi, dogodki in povezave [2, 6, 12].

Diagram koreografije za primer obiska veterinarske klinike je prikazan na sliki 4.13 in predstavlja le drugačen zorni kot pogleda na procese kot diagram sodelovanja, spodaj ležeči semantični model pa je isti. V tem diagramu so vsebovana samo opravila, ki preko

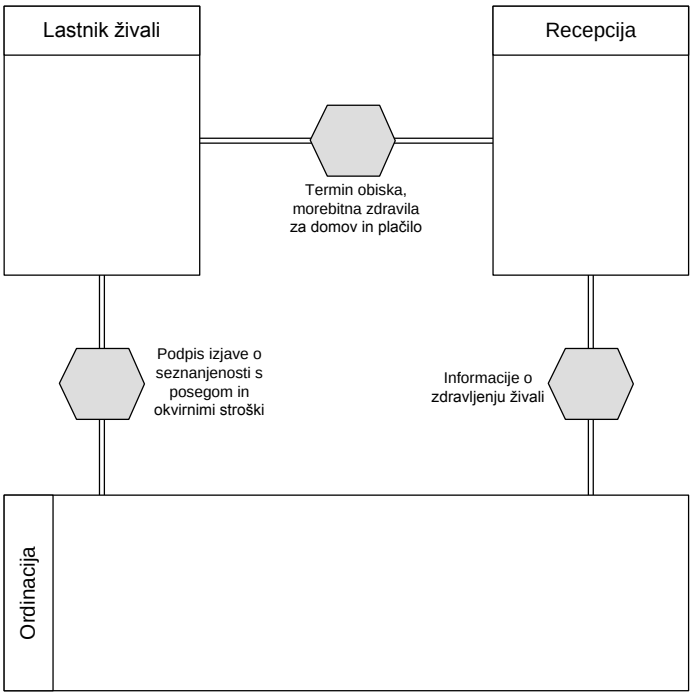
toka sporočil komunicirajo z različnimi udeleženci procesa (*Lastnik živali*, *Recepcija* in *Ordinacija*). Vsi ostali koraki so skriti.



Slika 4.13: **Obisk veterinarske klinike z vidika diagrama koreografije.**

4.5.2 Diagram pogovora

Diagram pogovora je podoben diagramu sodelovanja. Razlika med njima je v tem, da steze v diagramu pogovora ne vsebujejo procesov in med stezami ni dovoljeno vstavljanje koreografij. Poleg tega sta v diagramu pogovora dodana dva nova gradnika: *pogovor* (angl. Conversation) in *povezava pogovora* (angl. Conversation Link). Pogovor med udeleženci se prikaže z gradnikom pogovora, ki s povezavo pogovora povezuje vsaj dva bazena. Bazeni predstavljajo posamezne udeležence v pogovoru [6, 12].



Slika 4.14: **Obisk veterinarske klinike z vidika diagrama pogovora.**

Poglavje 5

IZVAJANJE POSLOVNIH PROCESOV Z BPMN 2.0

Izvajalni poslovni procesi bazirajo na diagramu, ki po korakih predstavlja izvajalni tok (angl. Executable Flow). Takšen diagram lahko izgleda skoraj identično kot diagram, ki predstavlja abstraktni poslovni proces, vendar je drugačen že v tem, da potrebuje višjo stopnjo tehničnih podrobnosti. BPMN 2.0 zagotavlja vizualizacijo s poslovno orientirano notacijo za jezike XML, ki so namenjeni izvajanju poslovnih procesov. Poleg tega je pomembno razumeti, da model izvajalnih poslovnih procesov povzame perspektivo enega samega sistema. Torej je celoten potek procesa obravnavan z vidika enega BPMS-ja in v primeru komunikacije med dvema zunanjima strankama BPMS običajno nima nobene informacije o korakih, ki se izvedejo pri komunikaciji. Enolično modeliranje takšnih korakov je nemogoče v izvajalnih poslovnih procesih. Dodatna razlika med abstraktnim in izvajalnim modelom poslovnega procesa je, da so izvajalni procesi del življenjskega cikla razvoja programske opreme, torej so pod nadzorom tehničnih razvijalcev in potrebujejo enotno testiranje kot vsak drug del programske opreme. Ključni pomen izvajalnega BPMN je, da služi kot izmenjevalni format [14, 19].

Izvajalni BPMN izvaja vse nivoje modela poslovnih procesov, ne samo najnižjega in najbolj podrobnega. Prav zaradi tega so nivoji znotraj modela poslovnih procesov striktno hierarhični. To pomeni, da morajo biti definicije aktivnosti na najvišjem nivoju skladne s podprocesi in/ali opravili, ki jih vsebujejo. Pri izvajanju se nadzor premika od aktivnosti na najvišjem nivoju do aktivnosti na nižjem nivoju in potem nazaj, ko se procesi inicializirajo in zaključijo [16].

Za izvajanje procesov se uporabljata dva pristopa: eden je z uporabo konvencionalnih tehnik in tehnologij, drugi pa z uporabo procesnih strojev (angl. Process Engine).

Pristop z uporabo konvencionalnih tehnik in tehnologij (na primer Java ali .NET) ima kar nekaj omejitev, saj so takšne rešitve težko prenosljive v druga izvajalna okolja, težko se prilagajajo spremembam procesov (proces je implementiran v obliki programske kode), izvajanja procesov ni mogoče realizirati na podlagi modelov procesov in ne podpirajo drugih aktivnosti, ki so del BPM.

Pristop z uporabo procesnih strojev zgoraj omenjene probleme pri pristopu s konvencionalnimi tehnikami in tehnologijami izniči ali vsaj minimizira, če so modeli BPMN obogateni z dodatnimi informacijami. Procesni stroj je sestavni del sistema za podporo upravljanja procesov (BPMS). Različni BPMS-ji se skupaj s podpornimi orodji združujejo v pakete (angl. Suites), ki podpirajo vse faze v BPM. V povezavi s procesnimi stroji se največkrat srečamo z jezikom BPEL, ki je (bil) zaslužen za izvajanje poslovnih procesov. Ker BPEL ne zagotavlja standardiziranega prikaza poslovnih procesov, se za to uporablja BPMN, saj je omogočena preslikavo iz BPMN v BPEL. Pri sodobnem BPMS-ju s podporo za BPMN 2.0 se lahko znotraj BPMS-ja izdela ali uvozi model procesa BPMN in se ga izvaja neposredno brez preslikave v BPEL [6].

5.1 Izvajanje poslovnih procesov z uporabo procesnih strojev

Pri poslovnih procesih, kjer še vedno obstaja potreba po jeziku BPEL, predstavlja avtomatizirana preslikava modela BPMN v BPEL enega izmed ključnih korakov v življenjskem ciklu, saj odpravlja razkorak med obema domenama in omogoča tesno poravnanost poslovnih zahtev in implementacije. Tehnični razvijalci izvedejo le potrebne dopolnitve implementacije. Če se realizira nov model BPMN, se preprosto opravi spojitev obeh verzij. Komunikacija pa lahko poteka tudi v obratni smeri. V primeru, da postopek implementacije zahteva dodajanje aktivnosti, se lahko te spremembe posredujejo nazaj. Ta dvosmerna komunikacija se imenuje krožno potovanje (angl. Round-Tripping) med abstraktnim modelom procesa in izvajalnim modelom procesa. Na ta način naj bi bila v vsakem trenutku zagotovljena ažurnost modela BPMN. V praksi to ni tako enostavno kot izgleda v teoriji, saj je podpora za krožno potovanje med čistim modeliranjem in izvajalnim okoljem BPMS večinoma pomanjkljiva. V današnjih časih je za omogočanje izvajanja procesov bolj običajno zanašanje na skriptne jezike in tehnološko specifične razširitve standarda BPMN [5, 10].

Zaradi omogočenega izvajanja procesov je kompleksnost BPMN 2.0 močno narasla.

BPMN 2.0 je značilno kompleksnejši od BPEL-a, vendar je neposredna preslikava v BPEL mogoča samo za podskupino BPMN 2.0, ki pa je najpomembnejša pri razvoju poslovnih procesov. Kljub temu, da je z BPMN 2.0 prišlo kar nekaj novosti, nekatere značilnosti še vedno nimajo izvajalnega okolja (angl. Runtime Environment), saj je skladna domena izven območja BPEL-a in bi se zato morala razviti nova vmesna oprema (angl. Middleware) [9].

V osnovi obstajata dva načina preslikave med BPMN in BPEL. Preslikana koda BPEL ima lahko strukturo grafa (angl. Graph Structure), kjer je celoten proces gnezden znotraj elementa tok (angl. Flow), zaporedje aktivnosti pa je določeno s povezavami (angl. Link). Drugi način preslikave, ko ima preslikana koda BPEL blokovno strukturo (angl. Block Structure), pa se tok zaporedja preslika s pomočjo elementa zaporedje (angl. Sequence). V resničnem življenju se na žalost pogosto izkaže, da prehod med fazo modeliranja (BPMN) in fazo implementacije (BPEL) ni preprost. Notacija BPMN in jezik BPEL sta si konceptualno preveč različna. Pred preslikavo iz BPMN v BPEL je potrebno priskrbeti dodatne informacije o poslovnem procesu in nad vsako aktivnostjo se mora izvesti funkcionalna dekompozicijo, dokler se ne doseže stopnje, ko so vse aktivnosti atomarne. Šele potem lahko sledi preslikovanje, ko se vsak objekt BPMN transformira v ekvivalent BPEL-a, če seveda obstaja.

BPEL je tipično blokovno strukturiran jezik (*if-then-else, repeat-until, case-otherwise, itd.*) in je namenjen zaporednemu izvajanju aktivnosti, kar pomeni, da ne pozna tako imenovanih ukazov GOTO. S pomočjo aktivnosti *While* je mogoče definirati preproste strukturne cikle, ni pa mogoča uporaba nestrukturiranih oziroma arbitrarnih ciklov (cikli z različnim številom vhodnih in izhodnih povezav). To pomeni, da z nobenim načinom preslikave iz BPMN v BPEL ni mogoče direktno preslikati arbitrarnih ciklov. Čeprav je BPEL mogočen jezik, je težek za uporabo. Poslovni ljudje, kot sta poslovni analitik in sistemski arhitekt, načeloma ne razumejo prikaza XML jezika BPEL, ker je preveč IT centriran in ne podpira konstruktov, ki se jih običajno rabi v poslovnih procesih.

Po drugi strani pa notacija BPMN pri modeliranju postavlja zelo malo omejitev, saj se procesi modelirajo v obliki grafov, torej se uporabljajo tudi arbitrarni cikli. BPMN je prosto strukturiran in mu manjkajo konstrukti *case, repeat, until, do-while,* itd., katere vsebuje jezik BPEL. Zaradi tega se jezik BPEL uporablja predvsem za tehnično orientirane procese, torej predvsem za orkestracijo spletnih storitev. Za razliko od BPEL-a pri izvajalnem BPMN ni nujno, da je implementacija spletna storitev, ki bazira na SOAP, vendar pa mora biti sočasna (angl. Synchronous) ter mora imeti zahteve (angl. Requests) in odgovore (angl. Responses), ki so v XML-ju predstavljeni kot sporočila. Zaradi uvedbe

semantike izvajanja poslovnih procesov v BPMN verziji 2.0, so se BPMS-ji usmerili v izvajalni BPMN, ki solidno nadomešča jezik BPEL [5, 13, 20].

5.2 Tipi skladnosti

Pred izdajo BPMN 2.0 ni bilo definirano, katere zahteve mora podpirati programsko orodje, da bo skladno z BPMN. V verziji 2.0 pa so merila ustreznosti za ugotavljanje skladnosti programskega orodja s specifikacijo BPMN jasno definirana. Definirani so štirje tipi skladnosti:

- ustreznost modeliranja procesov (angl. Process Modeling Conformance),
- ustreznost modeliranja koreografije (angl. Choreography Modeling Conformance),
- ustreznost izvajanja procesov BPEL (angl. BPEL Process Execution Conformance),
- ustreznost izvajanja procesov (angl. Process Execution Conformance) [6, 12].

5.2.1 Ustreznost modeliranja procesov

Implementacije, ki trdijo, da podpirajo tip ustreznosti modeliranja procesov, morajo podpirati naslednje pakete BPMN:

- Osnovni elementi BPMN, ki obsegajo tiste, ki so definirani v paketih infrastrukture, fundacije, skupnosti in storitev.
- Diagrami procesov, ki obsegajo elemente definirane v paketih procesov, aktivnosti, podatkov in človeške interakcije (angl. Human Interaction).
- Diagrami sodelovanja, ki obsegajo bazene (angl. Pools) in tok sporočil (angl. Message Flow).
- Diagrami pogovora, ki obsegajo bazene, pogovore (angl. Conversations) in povezave pogovorov (angl. Conversation Links) [12].

5.2.2 Ustreznost izvajanja procesov

Orodja BPMN 2.0, ki trdijo, da podpirajo tip ustreznosti izvajanja procesov, morajo polno podpirati in interpretirati operacijsko semantiko in življenjski cikel aktivnosti. Neoperacijski elementi naj bi se s strani implementacije ignorirali. Implementacija tipa

te ustreznosti mora polno podpirati in interpretirati spodaj ležeči metamodel. Poleg tega mora tip ustreznosti izvajanja procesov podpirati uvoz tipov diagrama procesov BPMN, vključno z njegovo definicijo sodelovanja [12].

5.2.3 Ustreznost izvajanja procesov BPEL

To je poseben tip ustreznosti izvajanja procesov, ki podpira preslikavo med BPMN in BPEL [12].

5.2.4 Ustreznost modeliranja koreografije

Implementacije, ki podpirajo tip ustreznosti modeliranja koreografije, morajo podpirati naslednje pakete BPMN:

- Osnovni elementi BPMN, ki obsegajo tiste, ki so definirani v paketih infrastrukture, fundacije, skupnosti in storitev.
- Diagrami sodelovanja, ki obsegajo bazene (angl. Pools) in tok sporočil (angl. Message Flow).
- Diagrami koreografije, ki obsegajo elemente, definirane v paketih koreografije in v sami koreografiji.

Orodja naj bi polno podpirala in interpretirala grafično in izvajalno semantiko, ki obsega elemente in tipe diagramov koreografije. Implementacija naj bi prav tako podpirala uvoz/izvoz tipov diagramov koreografije in tipov diagramov sodelovanja, ki upodabljajo koreografijo znotraj sodelovanja, da omogočijo prenos definicij koreografije. S tem se lahko definicije BPMN prenašajo med orodji različnih prodajalcev [12].

5.3 Semantika izvajanja procesov

Z BPMN verzijo 2.0 se je uvedla točno določena semantika (opisana je neformalno, v tekstovni obliki), ki je potrebna za izvajanje poslovnih procesov. Z drugimi besedami, opis interpretacije in opis izvajanja modelov BPMN sta jasna in natančna, medtem ko v prejšnjih verzijah BPMN takšne podrobnosti izvajanja procesov niso bile jasno definirane.

Obstajajo tudi elementi, pri katerih je priskrbljen samo konceptualni oziroma abstraktni model, ki pa ne opiše podrobnosti, potrebnih za izvajanje na procesnem stroju. To so neoperacijski (angl. Non-Operational) elementi, katere lahko implementacije ignorirajo

ali pa razširijo semantiko neopracijskih elementov in jih s tem naredijo izvedljive. Takšni elementi imajo večinoma le namen dokumentiranja.

Semantika izvajanja procesov je namenjena razvijalcem orodij za implementacijo simulacij, animacij in izvajanja poslovnih procesov na procesnih strojih. Razvijalci orodij, ki podpirajo BPMN 2.0, se morajo strogo držati standarda definirane semantike izvajanja, saj je za dosego medobratovalnosti (angl. Interoperability) potrebno, da se ohrani standardna semantika in sintaksa [6, 12].

Pomemben aspekt pri semantiki izvajanja je, kdaj naj se proces instancira. Če je v procesu en sam začetni dogodek, potem se proces instancira, ko se pojavi začetni dogodek. V bolj kompleksnih diagramih poslovnih procesov je lahko več začetnih dogodkov. V takšnih primerih BPMN trdi, da so si začetni dogodki alternative, torej se proces instancira, kadarkoli se pojavi eden izmed možnih začetnih dogodkov. V procesih, kjer je za zagon potrebnih več začetnih dogodkov, morajo biti začetni dogodki med seboj soodnosni. Soodnosnost oziroma korelacija je potrebna zato, da se dogodki pričvrstijo na instance procesa [21].

5.4 BPMS s podporo za BPMN 2.0

Pred BPMN 2.0 je bilo na razpolago več lastniških rešitev, kjer je vsako programsko orodje ponujalo svoj pristop k informatizaciji poslovnih procesov. S prihodom BPMN verzije 2.0 se je poenotila semantika izvajanja procesov, dosti lastniških programskih orodij s svojimi rešitvami za informatizacijo poslovnih procesov pa se je znašlo v zagati, saj je bila njihova lastna semantika izvajanja procesov drugačna od predpisane semantike izvajanja procesov v BPMN 2.0. Semantika izvajanja procesov z ozirom na avtomatizacijo izvajanja omogoča razvijalcem orodij enotno interpretacijo posameznih elementov BPMN. Informacijska tehnologija torej lahko podpre BPMN 2.0 pri modeliranju procesov in izvajanju procesov.

Številna splošna orodja, kot na primer Microsoft Visio, podpirajo le modeliranje procesov v BPMN, zato so specifična BPMN modelirna orodja (običajno del BPMS-ja) učinkovitejša. Splošna orodja BPMN so primernejša za podjetja, ki uporabljajo tudi druge notacije [6]. Orodja, ki podpirajo le modeliranje procesov v BPMN, običajno ne dopuščajo pravega hierarhičnega (nivojskega) modeliranja. V takšnih orodjih vsi modeli procesov, razen tisti na najnižjem nivoju, služijo le lažjemu razumevanju. Ker se tem vrstam orodij ni treba prilagajati temu, na kašen način BPMN definira hierarhične procese za pravilno delovanje, mora organizacija oziroma podjetje samo definirati,

kateri modeli procesov se bodo razvili kot avtomatizirani procesi. V takšnih primerih se običajno uporablja eno orodje za modeliranje procesov, ki služi samo kot informacija, in drugo orodje za modeliranje enega samega nivoja BPMN, ki je namenjen avtomatizaciji. Neposredna povezava med modeli procesov in avtomatiziranimi modeli procesov (angl. Automation Models) se prekine, ko je narejen izvoz. Za skladnost modelov procesov mora organizacija ročno vzdrževati in sinhronizirati modele v obeh orodjih [16].

Modelirno orodje, ki podpira semantiko izvajanja procesov BPMN 2.0, mora vključevati podporo za uporabo vsaj tistih elementov in atributov, ki so specificirani v zahtevah za tip skladnosti ustreznost izvajanja procesov (angl. Process Execution Conformance). Podrobnosti za izvajanje morajo biti predstavljene kot atributi in informacije v serializiranem izrazu modela XML.

BPMS orodja, ki podpirajo izvajalni BPMN 2.0, morajo zadostiti še nekaterim preostalim zahtevam, ki diktirajo, kako se v modelu izražajo informacije procesiranja, vmesniki storitvenih sklicev (angl. Service Invocation Interfaces) in poslovna logika (angl. Business Logic). Te zahteve so:

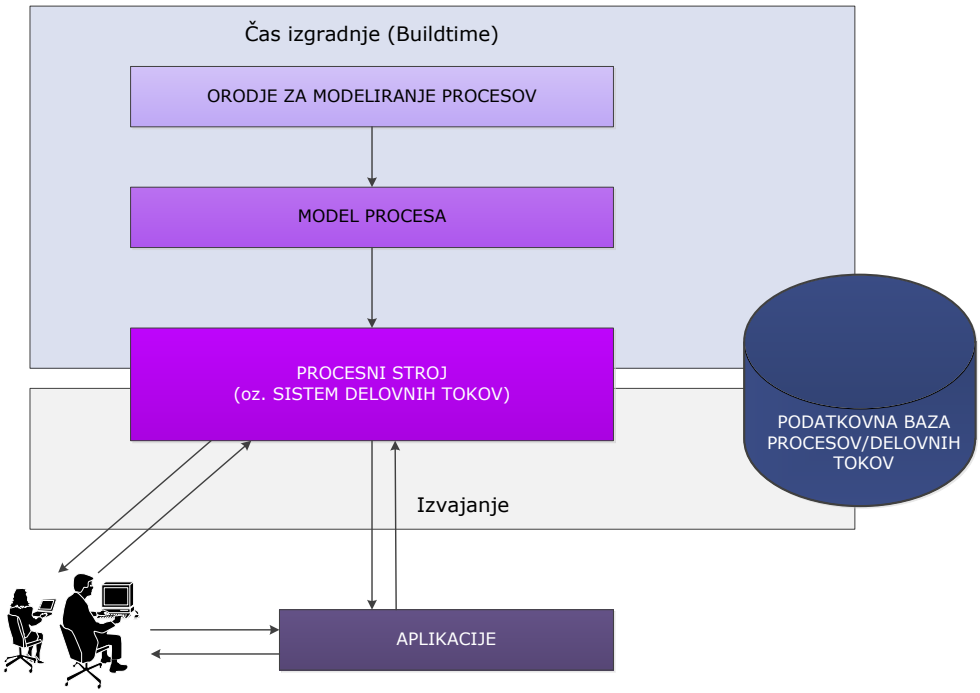
– Jezik definicije podatkovnega tipa mora biti XSD (definicija sheme XML), ki zagotovi ustreznost izmenjanih sporočil k zahtevanim strukturam sporočil.

– Jezik definicije vmesnika storitev mora biti WSDL, torej so sklici storitev za ali pa kot spletne storitve.

– Jezik podatkovnih asociacij mora biti XPath, ki standardizira skriptni jezik, kjerkoli oziroma kadarkoli že ga je potrebno uporabiti [10].

BPMS-ji običajno podpirajo nivojsko modeliranje procesov z uporabo enega samega integriranega modela z izkoriščanjem sposobnosti BPMN-ja, ki omogoča uporabo vstavljenih podprocesov (angl. Embedded Subprocesses). Model poslovnih procesov BPMN, ki je postavljen kot avtomatizirana rešitev, mora biti pravilen in popoln, kar pomeni, da so modeli procesov na vsakem nivoju skladni z modeli procesov na višjem nivoju, v katerem so vsebovani. Izdelava konsistentnih nivojev zahteva stopnjo zavezanosti podrobnostim, kar mnogim organizacijam povzroča težave [16].

Sistemi za podporo upravljanja procesov (BPMS) so običajno sestavljeni iz štirih glavnih komponent: procesni stroj, orodja za modeliranje procesov (angl. Process Modeling Tools), orodja za spremljanje procesov (angl. Process Monitoring Tools) in orodja za upravljanje življenjskega cikla procesov (angl. Process Life-Cycle Management Tools).

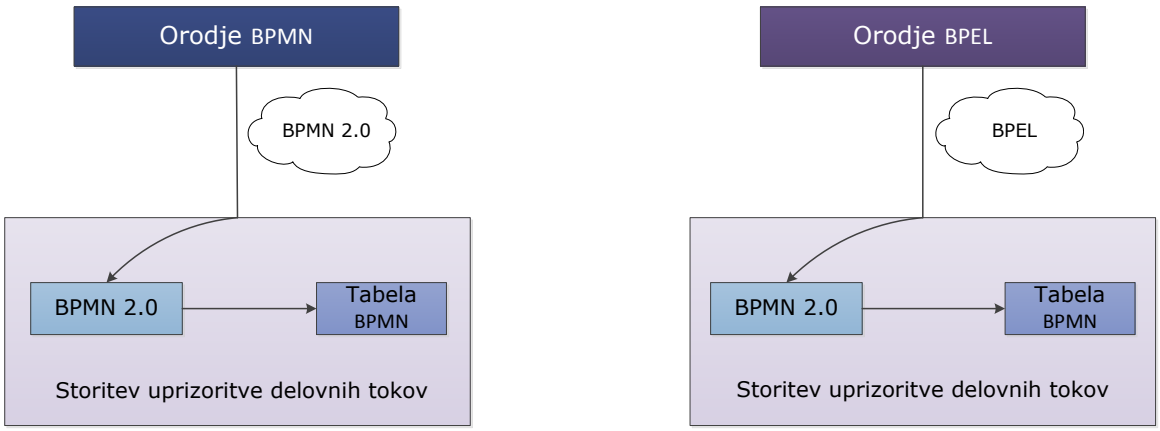


Slika 5.1: **Osnovna sestava BPMS-ja [9].**

Procesni stroj je najpomembnejši del BPMS-ja, saj je osnova za paket upravljanja poslovnih procesov (angl. Business Process Management Suite). Zmožen je izvajanja modelov poslovnih procesov z vidiki kakovosti (angl. Quality Aspects) kot so razpoložljivost, učinkovitost in robustnost. Ko procesni stroj izvaja proces, se v ozadju izvedejo avtomatizirane aktivnosti, izvedejo se prehodi in ustvarijo se uporabniška opravila, ki se dodelijo konfiguriranim uporabnikom in skupinam. Osnovni sestavni del procesnega stroja je dober **upravljalnik stanja procesov** (angl. Process State Manager), kateri mora vzdrževati tekoči poslovni proces v spominu, dokler proces ne doseže svojega končnega stanja. To lahko traja tudi dalj časa, kot je na primer en dan ali en teden. V tem času mora poslovni proces imeti sposobnost sprejemanja novih zahtev, obravnavanja dogodkov in napredovanja v definiciji poslovnega procesa. Ker je stanje procesa shranjeno v podatkovni bazi, lahko procesni stroj nadaljuje izvajanje procesov tudi po zaustavitvi sistema na strežniku zaradi strojne napake ali običajnega vzdrževanja. Drugi bistveni sestavni del procesnega stroja je **upravljalnik človeških opravil** (angl. Human Task Manager), ki podpira funkcionalnost poteka dela oziroma dodeljevanje uporabniških ali skupinskih opravil med izvajanjem procesa. Upravljalnik človeških opravil naj bi bil torej sposoben dodeljevanja opravil specifičnim uporabnikom ali skupinam uporabnikov [18].

Izvirni metamodel stroja BPMN 2.0 (lastni procesni metamodel stroja) lahko vsebuje

podporo za jezik BPEL. Prodajalec stroja BPEL, ki priskrbi podporo uvozu modelov poslovnega procesa BPMN 2.0, lahko postane prodajalec stroja BPMN 2.0, ki bazira na istem stroju izvajanja (angl. Execution Engine). Popolnoma jasno je, da se vmesni model BPEL generira iz modela BPMN 2.0, še preden je izveden dejanski uvoz. Uporabniki pa se vmesnega formata BPEL niti ne zavedajo. Sedaj je mnogo bolj kot izvirni metamodel procesnega stroja pomembna njegova robustnost, učinkovitost, skalabilnost itd. Možno je, da se bodo čez čas stroji BPEL razvili in razširili tako, da bodo podpirali ključne značilnosti BPMN 2.0, ki manjkajo v BPEL-u in strojih BPEL danes. Obstaja tudi velika verjetnost, da noben prodajalec ne bo podpiral celotnega BPMN 2.0 v svojih produktih zaradi visoke stopnje kompleksnosti [9].



Slika 5.2: **Stroj BPMN 2.0 na katerem lahko bazira tako orodje BPEL kot orodje BPMN 2.0 [9].**

Večina implementacij storitev v realnih BPMS-jih uporablja neko vrsto storitvenih adapterjev (angl. Service Adapter) ali priključkov (angl. Connector), ki so vsebovani v paketu znotraj BPMS-ja. Z uporabo čarovnika v modelirnem okolju BPMS (angl. BPMS Design Environment) se preko storitvenega adapterja vzpostavi vmesnik storitev. Rezultat tega je obširen XML (vse vrste podatkovnih vhodov, izhodov, preslikav, sporočil itd.), ki ga generira orodje pri izvozu BPMN-ja. Orodje naj bi prav tako razkrilo vmesnik storitve vsakega adapterja v obliki uvožene datoteke XSD, ki se sklicuje na model poslovnih procesov [20].

5.4.1 Opis orodij BPMS

Obstaja kar nekaj orodij BPMS, ki podpirajo BPMN 2.0, in prav tako večina teh orodij podpira programski jezik Java. Nekatera izmed takšnih orodij BPMS so Activiti, Oracle BPM 11g, IBM Web Sphere, Bonita Open Solution in jBPM. Popolnoma lastniško orodje je le IBM Web Sphere, Bonita Open Solution pa poleg lastniške različice ponuja še odprtokodno različico z omejenim številom modulov. Activiti, jBPM in Oracle BPM 11g so prosto dostopna orodja BPMS. Vsa omenjena orodja podpirajo semantiko izvajanja procesov BPMN 2.0, le IBM Web Sphere uporablja BPMN za modeliranje, BPEL pa za izvajanje [6].

jBPM

jBPM je odprtokodni procesni stroj, napisan v Javi, ki ga je izdal JBoss Community pod licenco ASL (licenca Apache). Od vsega začetka je podpiral svoj procesni jezik jPDL, z verzijo 5.0 pa je dobil podporo za izvajanje poslovnih procesov, ki so izdelani v BPMN 2.0. Poleg tega se je z verzijo 5.0 projekt jBPM združil s projektom JBoss Drools (odprtokodno ogrodje za upravljanje poslovnih pravil) in nadomestil Drools Flow kot jezik toka pravil v ogrodju Drools.

jBPM za razvijalce priskrbi modelirno orodje, ki bazira na projektu Onyx, ter orodje Eclipse. Končni uporabniki, torej poslovni uporabniki, lahko preko spletne aplikacije ustvarijo, postavijo, izvajajo in upravljajo poslovne procese preko celotnega življenjskega cikla procesa. jBPM 5.0 vsebuje tudi močno podporo za integracijo dogodkov in poslovnih pravil ter podporo za bolj napredne in fleksibilne poslovne procese. Pri človeški interakciji se uporablja neodvisen standard WS-HT (WebService-HumanTask) [15, 24].

Activiti

Procesni stroj Activiti je srce projekta Activiti, ki je ugledal luč sveta, ko sta dva ključna razvijalca pri jBPM, Tom Baeyens in Joram Barrez, leta 2010 zapustila Red Hat in se zaposlila pri Alfrescu ter začela z Activitijem. Slednji bazira na podlagi njunih izkušenj poteka dela pri jBPM, sama osnovna programska koda, ki je napisana v programskem jeziku Java, pa je spisana na novo in ne bazira na nobeni programski kodi jBPM. Activiti se trguje pod licenco Apache, razvijajo pa ga zaposleni pri Alfrescu in razvijalci iz skupnosti Activiti (angl. Activiti Community). Activiti je torej odprtokodni stroj delovnih tokov (angl. Open-Source Workflow Engine), ki lahko izvaja poslovne procese, predstavljene v BPMN 2.0.

Activiti teče v kateremkoli Java okolju, kot sta Spring, Standalone JDBC in JTA, na strežniku ali v oblaku. Spretno se integrira z okoljem Spring. Je ekstremno hiter in lahek (angl. Lightweight) ter bazira na enostavnih konceptih. Activiti vsebuje tudi skrite Java poslušalce dogodkov (angl. Event Listeners) za razdruževanje tehničnih podrobnosti programske opreme iz diagramov na poslovni ravni. Ponuja tudi možnost testiranja izvajanja procesov v izolaciji z enostavnim osnovnim testom (angl. Plain Unit Test) [14].

Bonita Open Solution

Bonita je bila ustvarjena leta 2001 v francoskem INRIA (National Institute for Research in Computer Science) in se je potem nekaj let razvijala v francoski družbi Bull. Leta 2009 pa je razvoj prevzelo podjetje Bonita Soft.

Bonita je odprtokodni BPMS, ki podpira modeliranje, razvoj, izvajanje in nadzor poslovnih procesov. Združuje tri rešitve v eno samo: inovativen studio za modeliranje procesov, zmogljiv stroj BPM in prizadeven vmesnik za končne uporabnike. Stroj BPM je Java API (vmesnik uporabniškega programa), ki dovoljuje obstoj programskih interakcij s poslovnimi procesi.

Bonita BPM ponuja enostavno, intuitivno in grafično rešitev. Inovativen pristop prejemanja obvestil (angl. Inbox) ponuja uporabniku enostaven način za upravljanje opravil in sodelovanje. Definiran model BPMN se samodejno preslika v obrazce spletne aplikacije, kjer so vnosna polja skladna z definiranimi atributi v modelu BPMN. Bonita je dovolj fleksibilna, da se prilagodi katerikoli arhitekturi informacijskih sistemov in dovolj zmogljiva, da vzdrži intenzivno obremenitev. Omogoča integracijo z obstoječimi rešitvami, kot so spletne storitve, različne podatkovne baze, LDAP, sporočilni sistemi, Pay Pal, SAP itd. [6, 19].

Primerjava orodij BPMS

	Bonita	**Activiti**	**jBPM**
Integrirano razvojno okolje	Da	Ne	Ne
Razširitev z Java poslušalci dogodkov	Ne	Da	Ne
Podpora poslovnim pravilom	Da	Da – osnovna integracija s strojem pravil Drools.	Da – integracija s strojem pravil Drools na večih nivojih.
Podpora za urejanje obrazcev	Da	Da	Da
Način razvoja poslovnih procesov	Procesi in obrazci se definirajo s tehniko klikanja in vlečenja (angl. Click and Drag), torej je mogoče razviti proces brez kodiranja v Javi.	S pomočjo Java API-ja, ki zahteva vsaj nekaj kodiranja v Javi.	S pomočjo Java API-ja, ki zahteva vsaj nekaj kodiranja v Javi.
Nadzor nad kodo poslovnih procesov	Delni nadzor, saj je koda običajno generirana s strani razvojnega orodja.	Popoln nadzor nad kodo.	Popoln nadzor nad kodo.
Podpora neodvisnemu standardu WS-HT	Ne	Ne	Da
Podpora za simulacijo	Da	Ne	Da
Modeliranje diagramov z ročnimi opravili	Ne	Da – med izvajanjem se ročno opravilo preskoči.	Da – med izvajanjem se ročno opravilo preskoči.
Asinhrono izvajanje procesov	Da	Da	Da

Tabela 5.1: **Primerjava orodij BPMS.**

Kot je razvidno iz tabele 5.1 je samo Bonita integrirano razvojno orodje, kjer se poslovni procesi in obrazci definirajo s tehniko klikanja in vlečenja konstruktov. Bonita je torej veliko bolj primerna za razvijalce z manjšim tehničnim znanjem, saj je za razvoj poslovnega procesa v Activitiju ali jBPM-ju potrebna določena stopnja tehničnega znanja oziroma znanja programiranja v programskem jeziku Java. V Boniti je posledično tudi zaradi tehnike klikanja in vlečenja konstruktov, kjer se za vsak konstrukt generira določena koda, le delni nadzor nad kodo poslovnih procesov. Activiti in jBPM omogočata popoln nadzor nad kodo, saj je razvoj poslovnih procesov in s tem tudi kodiranje ter dostop do kode omogočen s pomočjo Java API-ja. Vse trije BPMS-ji omogočajo asinhrono izvajanje procesov, imajo podporo za poslovna pravila in podporo za urejanje obrazcev.

Activiti omogoča razširitev z Java poslušalci dogodkov. jBPM je edini BPMS, ki podpira neodvisen standard WS-HT za uporabniška opravila, katera se zaradi neodvisnega standarda lahko brez težav prenesejo v katerikoli BPMS, ki podpira standard WS-HT. Bonita je edina, ki ne podpira modelov z ročnimi opravili, Activiti pa edini, ki ne omogoča simulacije poslovnih procesov.

Bonita je platforma, ki je primernejša za manj izkušene razvijalce poslovnih procesov, ki jim zadostujejo enostavnejši poslovni procesi. Pri bolj neobičajnih in kompleksnejših poslovnih procesih, predvsem pa v primeru, da je pomemben popoln dostop do kode poslovnih procesov, je primerneje izbrati Activiti ali jBPM [14, 19, 24].

5.5 Preoblikovanje abstraktnega modela v izvajalni model BPMN 2.0

Pri dobro načrtovanem modelu izvajalnih procesov je premalo, da je model poslovnih procesov zgrajen iz izvajalnih elementov BPMN 2.0, ki so pogoj za izvedljivost modela. Model izvajalnih procesov mora kazati tudi želene značilnosti, ki so potrebne za dosego zastavljenega cilja, in vsebovati elemente, ki se nedvoumno preslikajo v implementirane komponente.

Jedro izvajalnega modela poslovnih procesov so različne vrste avtomatiziranih aktivnosti. Aktivnosti v modelu poslovnih procesov so dejansko abstrakcija komponent izvajalnih storitev (angl. Executing Service Components). Zaradi tega so primerno poimenovane in konfigurirane glede na implementirano platformo. **Opravilo poslovno pravilo** (angl. Business Rule Task) je abstrakcija stroja poslovnih pravil (angl. Business Rule Engine), ki se ga kliče kot tip odločitvene storitve. **Uporabniško opravilo** (angl. User Task) je abstrakcija človeškega poteka dela (angl. Human Workflow), ki realizira uporabniški vmesnik. Aktivnost klic (angl. Call Activity) se prezrcali v podproces BPMN, kjer komponenta, ki implementira elemente in aktivnosti, slednje tudi izvaja. Aktivnost klic je torej lahko abstrakcija drugega procesa BPMN, ki ga implementira BPMS, ali proces BPEL, ki ga implementira stroj BPEL, ali pa proces lokalnega storitvenega vodila (LSB), ki ga implementira LSB. **Storitveno opravilo** (angl. Service Task) je abstrakcija zunanje referenčne storitve, na primer klica WSDL. To je lahko vse, kar je odkrito. Enostaven primer tega je storitev oznanilo (angl. Notification Service), ki skrbi za pošiljanje elektronskih sporočil, in je obenem alternativa uporabi opravila pošiljanje (angl. Send Task) [10].

5.5.1 Postavitev meje med poslovnim in tehničnim nivojem modela

Proces gre običajno čez serije dekompozicij, preden doseže stanje za implementacijo [2]. Ni pomembno, kako organizacija definira posamezne nivoje modelov poslovnih procesov, na nekem specifičnem nivoju je s stališča avtomatizacije ekstremno pomembno, da obstaja meja med nivojem, ki je takoj nad tokom avtomatizacije (angl. Automation Flow), in nivojem, kjer je sam tok avtomatizacije.

Procesi opravijo delo znotraj aktivnosti, torej znotraj podprocesa ali opravila. Podproces je v bistvu posoda za diagram poslovnih procesov in je nekakšen način za razvrščanje zbirke elementov toka. S tega stališča sam podproces ne opravlja dejanskega dela, ampak je delo opravljeno znotraj opravil, ki jih podproces vsebuje. BPMN definira naslednja opravila, ki podpirajo **avtomatizacijo** procesa: storitveno opravilo, opravilo prejem sporočila, opravilo pošiljanje sporočila, skriptno opravilo, opravilo poslovno pravilo in uporabniško opravilo. Ta opravila predstavljajo tehnično implementacijo toka, zato naj bi bile te podrobnosti pod nivojem, katerega želi videti nekdo, ki razume le poslovni proces.

Za maksimalno razumljivost procesov mora organizacija ločiti najnižji nivo poslovne predstavitve procesa od tehnične implementacije, ki podpira ta tok procesa. To se najlažje doseže z dogovorom, da najnižji nivo poslovne predstavitve procesa vsebuje samo podprocese in nobenih opravil. Ti podprocesi predstavljajo najbolj atomarne aktivnosti na nivoju poslovne predstavitve. Z vsakim od teh podprocesov naj bi bil tok procesa definiran tako, da bi predstavljal nivo tehnične avtomatizacije tega podprocesa. Ti tehnični podprocesi so sestavljeni iz kombinacije podprocesov in avtomatiziranih opravil. Na tem nivoju diagrama poslovnih procesov so tehnične podrobnosti kot so: uporaba sporočanja, zajem storitvenih opravil ter preslikovanje transakcij in podatkovnih tokov (angl. Data Flows). Organizaciji ta pristop omogoča vzdrževanje toka procesa na nivoju poslovne predstavitve procesa, ki je popolnoma konsistenten in obenem ločen od tehničnih elementov procesa. To je tudi poglavitni korak za katerokoli organizacijo, ki želi doseči razumljivost in izvedljivost, ki sta obljubljeni v BPMN-ju [16].

5.5.2 Avtomatizirana in človeško orientirana opravila

V svetu BPM je proces lahko avtomatiziran ali človeško orientiran. Slednji večinoma zahtevajo človeško posredovanje, da se proces premakne iz enega stanja v drugega. Za razlikovanje med avtomatiziranimi in ročnimi aktivnostmi se uporabljajo različni atributi [2].

Vsak BPMS si drugače predstavlja človeška opravila (angl. Human Task) in spremljajoče načrtovalsko orodje. V modelu BPMN 2.0 so prenosljiva samo tista človeška opravila, ki podpirajo WS-Human Task. Edini del izvajalnega BPMN 2.0, ki je prenosljiv med različnimi orodji, je logika toka in ne implementacija opravil.

Pri človeški interakciji s procesom je človeško opravilo v lasti vloge, ki zagotavlja, da je opravilo pravočasno izvedeno. Pri poslovnem opravilu ni pomembno, kako se dovrši delo, saj se to lahko zelo hitro spremeni. Pomembno pa je, kdaj se delo izvede, saj se najprej opravi eno opravilo, šele nato se kliče naslednje opravilo. S takšnim časovnim zaporedjem se ohranja čisto sporočilo diagrama poslovnih procesov.

Če proces upravlja procesni stroj, potem je potrebno vedeti, če je opravilo *ročno* ali *avtomatizirano* (*uporabniško* ali *sistemsko*), saj so regulatorji za vsako vrsto opravila drugačni. BPMN specificira dva različna tipa opravil, kjer je za dokončanje opravila potrebna človeška interakcija: ročno opravilo in uporabniško opravilo.

Uporabniško opravilo izvede in upravlja stroj poslovnih procesov v tekočem času, atributi uporabniškega opravila pa potrebujejo človeško interakcijo, pri čemer se za dovršitev človeških nalog največkrat uporabljajo uporabniški vmesniki v obliki obrazcev. Uporabniško opravilo je torej opravilo, kjer človek izvede neko opravilo s pomočjo programske aplikacije. Življenjski cikel takšnega opravila upravlja upravljalnik opravil in se običajno izvede v kontekstu procesa.

Za razliko od uporabniškega opravila ročnega opravila ne izvede in ne upravlja stroj poslovnih procesov, ampak se vse opravi ročno. Stroj torej ne sledi začetku in zaključku opravila. To so opravila, ki za dovršitev ne potrebujejo interakcije s sistemom.

Sistemska opravila in opravila poslovno pravilo pomagajo pri uporabniških opravilih v stroju poslovnih procesov. Sistemsko opravilo je čista IT aktivnost, ki je lahko del uporabniškega opravila. To je lahko opravilo, ki na primer ob določenem času preveri slabe podatke ali preveri opravila, ki niso bila končana v zahtevanem času [23].

5.5.3 Določanje poslovnih pravil

Eden od bistvenih elementov procesa je definicija poti, katerim se lahko sledi od aktivnosti do aktivnosti v procesu. Medtem ko se nekatere poti enostavno sprožijo z zaključitvijo aktivnosti, je bolj običajno, da imajo poti omejitve, ki omejujejo okoliščine, v katerih proces lahko prečka posamezno pot. Pod temi omejitvami ležijo poslovna pravila, ki čakajo, da se jih odkrije in obravnava. V BPMN-ju se z uporabo posameznih elementov lahko na poteh znotraj procesov definira takšne omejitve. Zbirka takšnih elementov BPMN omogoča organizaciji, da sama vpelje standarde za preslikovanje različnih predstavitev

v okviru poslovnih pravil (angl. Business Rules Framework), kjer se ta pravila lahko zajamejo. Seveda pa mora organizacija upoštevati tudi zmožnosti ciljne platforme.

Izraz tok se znotraj BPMN-ja uporablja za opis prenašanja kontrole iz ene aktivnosti na drugo. Nadzor toka se lahko izvaja na dva načina:

1. Proces specificira neko vrsto notranje omejenosti, s katero se mora seznaniti, da se procesiranje lahko nadaljuje po izbranem toku.

2. Proces specificira, da se mora zgoditi določen zunanji dogodek, da se procesiranje lahko nadaljuje.

Notranja omejenost je eden od prostorov, kjer vzajemno delujejo poslovna pravila in poslovni procesi. Sam standard BPMN ne definira ničesar o poslovnih pravilih in podobnih omejenostih. Organizacijam je prepuščeno, da same osnujejo dogovore o vzajemnem delovanju med poslovnimi pravili in procesi. Razmerje med nadzorom toka in poslovnimi pravili se najlažje vidi pri pogojnih tokovih (angl. Conditional Flows). Pogojni tokovi vsebujejo pogoj, ki določa, ali se bo proces nadaljeval po toku ali ne. Ta pogoj je poslovno pravilo. V BPMN so pogojni tokovi predstavljeni kot samostojni elementi ali kot izhodi iz različnih vrst prehodov.

Pogojni dogodek (angl. Condition Event) se obnaša podobno kot zgoraj omenjeni pogoji, torej drugače kot običajni dogodki. Pogojni dogodek je pravzaprav način vstavljanja poslovnega pravila na sredino toka. Edini namen tega dogodka je, da testira pogoj, ki definira dogodek. Tudi tu je pogoj poslovno pravilo. Ključna razlika med pogojnim dogodkom in pogojnim tokom je v tem, da pogojni dogodek čaka toliko časa, da se izpolni pogoj, medtem ko se pogojni tok testira le enkrat in to takrat, ko tok doseže pogoj. Ob neizpolnjenem pogoju pogojni tok ne nadaljuje po poti, na kateri pogoj ni bil izpolnjen. Pogoj znotraj toka procesa obstaja zato, da se naredi odločitev o tem, katerim potem znotraj procesa se bo sledilo. Ta odločitev je nekaj edinstvenega za določeno instanco procesa in ne more biti narejena izven te instance procesa.

Vsak poslovni proces osnuje področje, ki vsebuje množico informacijskih konceptov. V BPMN se ti informacijski koncepti imenujejo podatkovni elementi (angl. Data Items). V procesu so grafično prikazani s simbolom podatkovnega objekta ali pa so enostavno definirani kot del definicije procesa. Vrednosti podatkovnega elementa so lahko postavljene na različne načine:

– Informacija se prenese v proces takrat, ko se kliče instanca procesa, atributi pa se preslikajo v podatkovne objekte.

– Prejmejo se dogodki, njihove lastnosti pa se preslikajo v podatkovne objekte.

– Rezultati aktivnosti se preslikajo v podatkovne objekte.

– Obnovi se informacija iz podatkovne baze.

V mnogih situacijah je dovolj le pisanje pravil nadzora procesa (angl. Process Control Rules), ki uporabljajo podatke v procesu, ki so vsebovani znotraj podatkovnih elementov. Kakorkoli, ta pristop ima tudi nekaj omejitev.

Pri pisanju pravil kontrole procesa morajo biti vsi podatki, ki se uporabijo, del procesa. Pri enostavnih pravilih so ti podatki običajno že predstavljeni v procesu, pri kompleksnejših pravilih pa je potrebna definicija dodatnih podatkov v procesu in vstavljanje dodatnih aktivnosti za obnovo podatkov. Z uporabo tega pristopa se mora zaradi definiranja novih podatkovnih elementov in dodajanja novih aktivnosti spremeniti sam poslovni proces.

V mnogih primerih je pri uporabi pravil nadzora procesa testirana vrednost v poslovnem pravilu dinamično pridobljena. Pri tem je pomembno, da se vedno uporabi najnovejšo vrednost zahtevanih podatkov. Podatki, ki se pridobijo s strani kompleksnejših pravil in se iz podatkovne baze ne obnovijo na enostaven način, zahtevajo dinamično ponovno vrednotenje teh pravil, kadarkoli se zahtevajo podatki. Vgrajevanje takšnih pravil nadzora procesa zahteva, da je izpeljava definirana na vsaki lokaciji, kjer se uporabi. Zaradi tega organizacije velikokrat razširijo pravila, ki izhajajo iz nespecifičnih informacij posameznega procesa. Eden izmed načinov za dosego tega cilja so dodatne ločene storitve, ki poganjajo pravila, potrebna za dinamično pridobivanje informacij. Klic takšne storitve se ponovi pred vsakim testom informacij. Opisan pristop se uporablja na skoraj vseh izvajalnih platformah, ki bazirajo na BPMN-ju.

Standard BPMN 2.0 priskrbi en element, ki se nanaša le na poslovna pravila. Ostali elementi standarda niso eksplicitno definirani, da bi bili točke vmesnikov (angl. Interfaces Points) za poslovna pravila, in se nanašajo le na preference in način obdelave poslovnih pravil, ki jih določijo posamezne organizacije. Element, ki se nanaša le na poslovna pravila, je posebna vrsta opravila: opravilo poslovno pravilo. Ironično je ta element za organizacije najmanj uporaben element pri integraciji poslovnih pravil v procese, ki bazirajo na BPMN-ju. Opravilo poslovno pravilo je funkcionalno identično storitvenemu opravilu, saj ne priskrbi nobenega dodatnega obnašanja. Nekatere organizacije uporabljajo to opravilo le kot grafičen prikaz, da se določen klic storitev uporablja za klic poslovnih pravil. Vendar je uporaba tega elementa problematična:

– Za poslovna pravila naj bi bilo bolj primerno, da se uporabljajo po potrebi, kot da so sprožena eksplicitno.

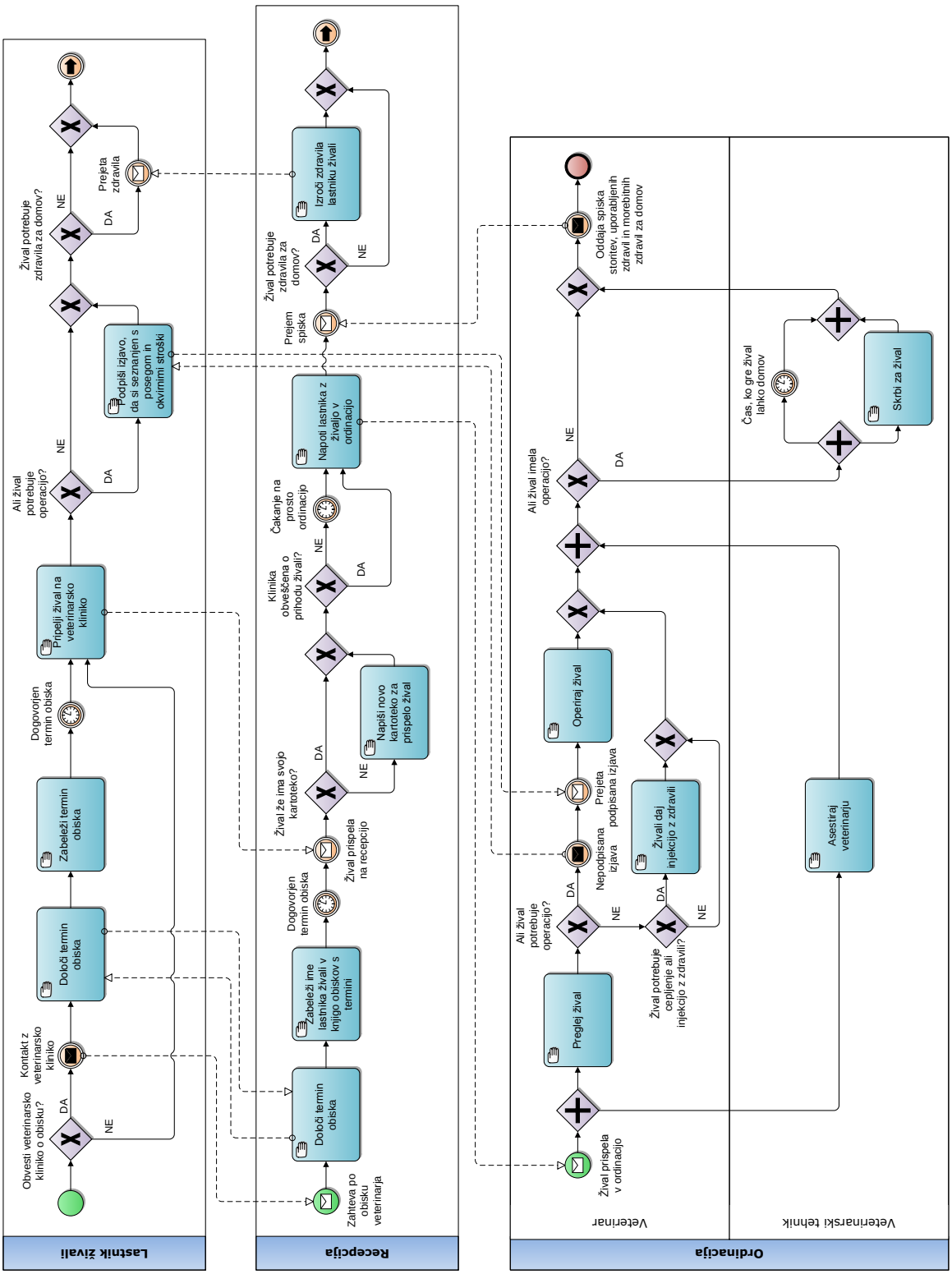
– Z uporabo opravila poslovno pravilo organizacija v svoje tokove procesov vstavlja tehnološko obvezo, in če se delo, ki mora biti opravljeno, razvije preko meja platforme poslovnih pravil, potem uporaba opravila poslovno pravilo znotraj procesnega toka postane problematična.

Običajno je za organizacijo najboljše, da se izogiba uporabi opravila poslovno pravilo in namesto tega uporablja storitveno opravilo [17].

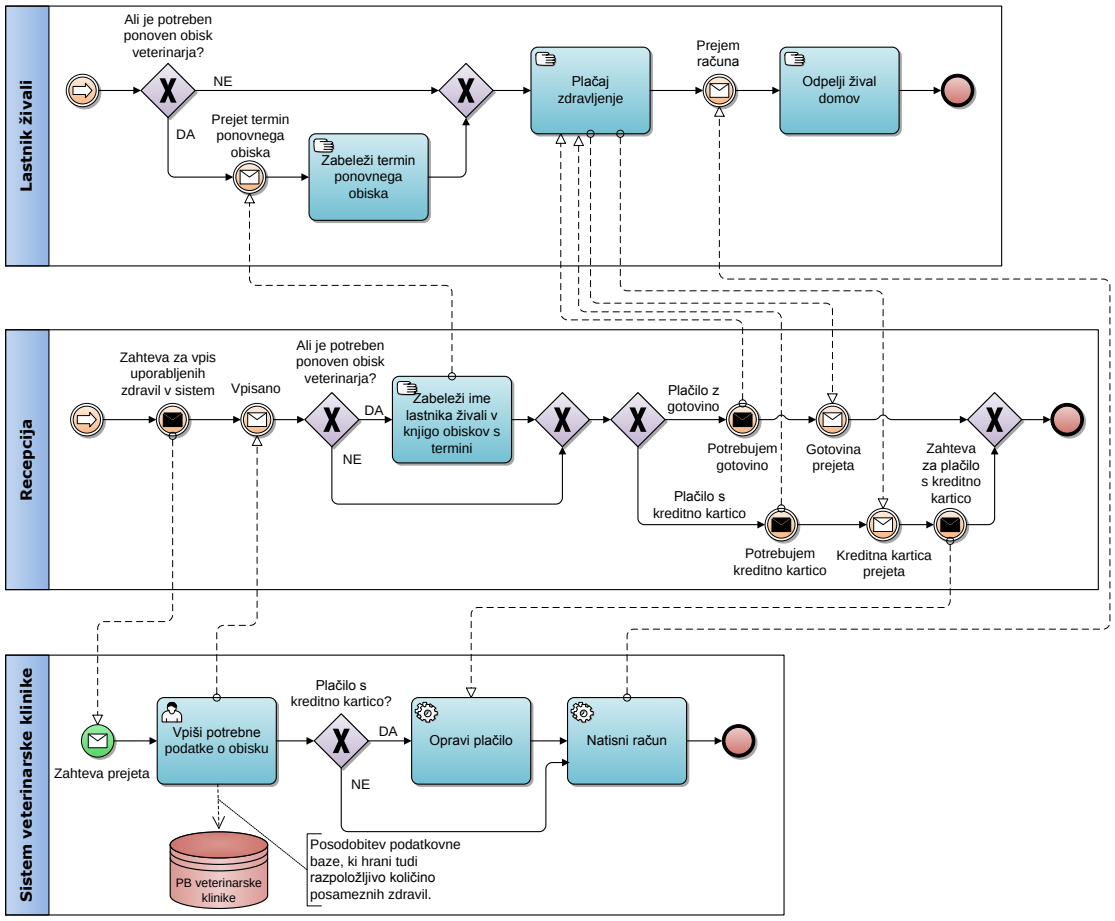
5.5.4 Preoblikovanje abstraktnega modela procesa obiska veterinarske klinike v izvajalni model BPMN 2.0

Diagram sodelovanja BPMN, ki je prikazan na slikah 4.11 in 4.12, ima vse avtomatizirane aktivnosti v bazenu *Recepcija*, ostala dva bazena pa vsebujeta le ročna opravila. Ker bazen *Recepcija* vsebuje tako opravila, ki nimajo nič skupnega s samim informacijskim sistemom veterinarske klinike, kot tudi opravila, ki so del informacijskega sistema, se izdela nov diagram sodelovanja BPMN, kjer se bazen *Recepcija* nadomesti z bazenom *Recepcija* in bazenom *Sistem veterinarske klinike*. Z novim diagramom sodelovanja BPMN, ki je prikazan na slikah 5.3 in 5.4, se avtomatizirana opravila ločijo od neavtomatiziranih.

Na podlagi izdelanega diagrama sodelovanja BPMN za primer obiska veterinarske klinike se vidi, da je za izdelavo izvajalnega modela BPMN in kasnejšo implementacijo edini smiseln del modela bazen *Sistem veterinarske klinike*. Ostale dele modela bi procesni stroj enostavno preskočil ali pa bi razširil njihovo semantiko in jih naredil izvedljive. Neizvajalni deli modela BPMN večinoma služijo boljšemu razumevanju celotnega poslovnega procesa. Izvajalni model BPMN 2.0, ki ga bo obdelal procesni stroj, torej vsebuje le konstrukte, ki so vsebovani v bazenu *Sistem veterinarske klinike*. Prav tako zaradi enostavnosti primera ni postavljene meje med poslovnim in tehničnim nivojem modela.

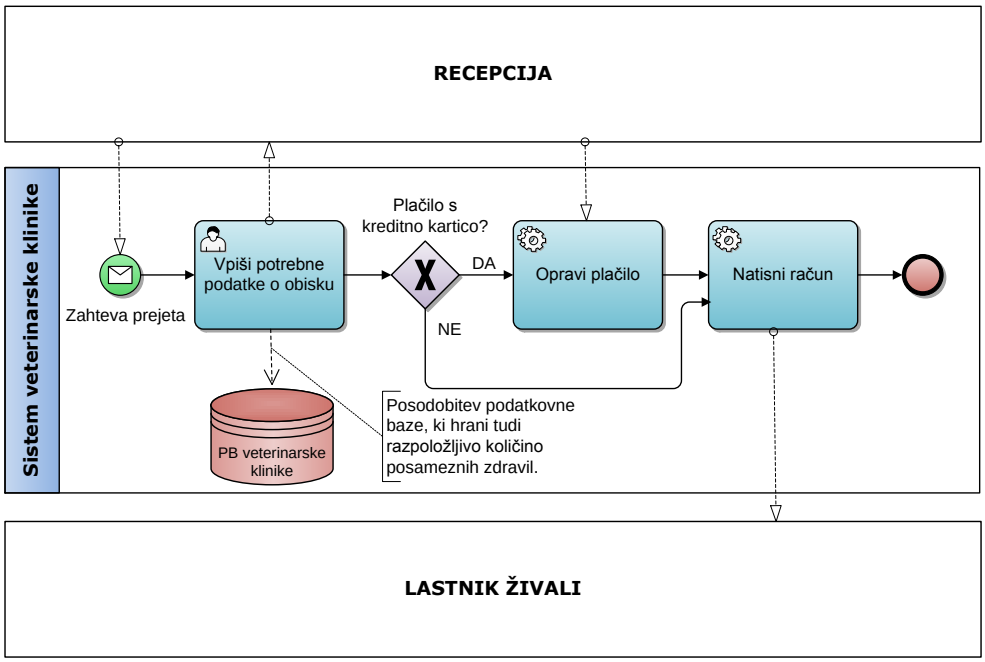


Slika 5.3: **Prvi del modela obiska veterinarske klinike z vidika diagrama sodelovanja BPMN 2.0.**



Slika 5.4: **Drugi del modela obiska veterinarske klinike z vidika diagrama sodelovanja BPMN 2.0.**

Kot je prikazano na sliki 5.5, se proces v bazenu *Sistem veterinarske klinike* začne, ko mora receptor oziroma zaposleni na recepciji vpisati potrebne podatke o obisku veterinarske klinike kot so ime živali, vrsta domače živali, priimek in ime lastnika, datum obiska, vrsta veterinarske storitve ter uporabljena in predpisana zdravila za domačega ljubljenčka. Vpis podatkov se izvede preko obrazca. Po potrditvi vpisa podatkov se posodobi podatkovna baza veterinarske klinike. Lastnik domače živali nato poravna stroške zdravljenja. Na razpolago ima plačilo z gotovino ali plačilo s kreditno kartico. Če se lastnik živali odloči za plačilo s kreditno kartico, se informacijski sistem preko vmesnika storitvenega opravila poveže z bančnim sistemom, da se opravi bančna transakcija. Ob uspešno končanem plačilu se sproži storitveno opravilo za tiskanje računa. Ko se račun natisne, se proces v bazenu *Sistem veterinarske klinike* zaključi.



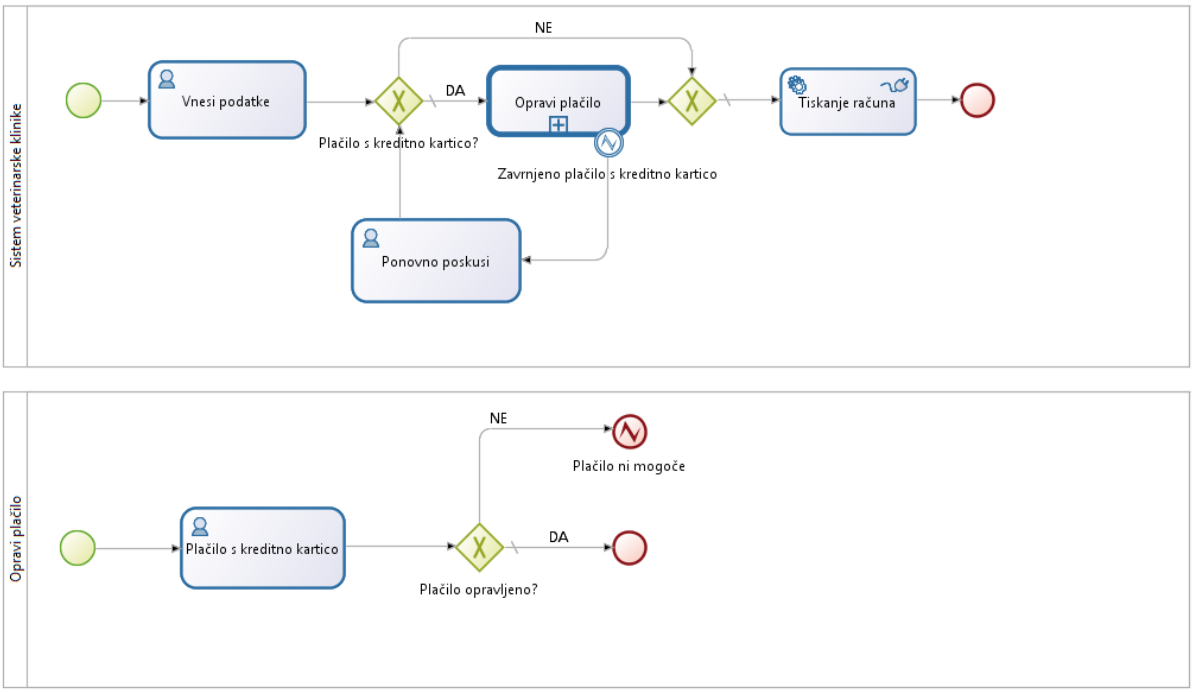
Slika 5.5: **Izvajalni del modela BPMN 2.0 za primer obiska veterinarske klinike.**

Vsi modeli BPMN 2.0, ki so predstavljeni do te točke, so izdelani v orodju Microsoft Visio 2010. Od tukaj naprej je za modeliranje izvajalnega modela BPMN 2.0 uporabljeno orodje Bonita BPM Community verzije 6.0.2, torej pravi BPMS.

V Boniti je izdelan poenostavljen fiktiven model procesa *Sistem veterinarske klinike*. Receptor najprej v uporabniškem opravilu *Vnesi podatke* preko obrazca vnese potrebne podatke v sistem. Lastnik živali se mora odločiti ali bo plačal zdravljenje z gotovino ali s

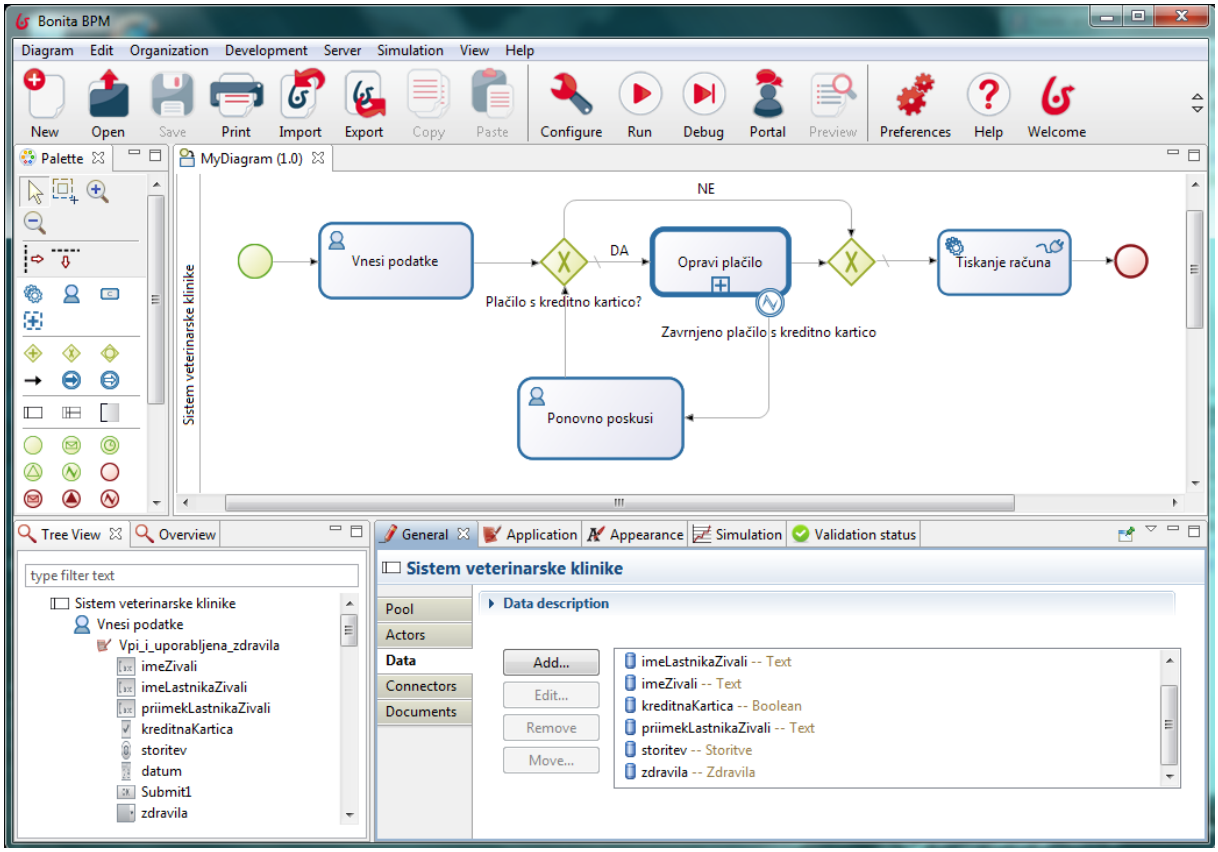
kreditno kartico in to sporočiti receptorju, ki preko obrazca sporoči sistemu želeni način plačila. Kot je prikazano na sliki 5.6, je storitveno opravilo *Opravi plačilo* nadomeščeno z aktivnostjo klic *Opravi plačilo*. Poleg tega iz opravila *Opravi plačilo* vodita dva toka zaporedja, eden za primer uspešnega zaključka opravila in drugi za primer, če pride do napake pri plačilu s kreditno kartico. Ker je poenostavljenemu informacijskemu sistemu nekako potrebno sporočiti, če v primeru neuspešnega zaključka opravila *Opravi plačilo* lastnik živali želi še enkrat poskusiti s plačilom preko kreditne kartice, je v ta namen v diagram dodano uporabniško opravilo *Ponovno poskusi*. Na ta način gre tok zaporedja lahko še enkrat čez aktivnost klic *Opravi plačilo*. Če plačilo s kreditno kartico nikakor ne uspe, mora lastnik živali zdravljenje plačati v gotovini. Na koncu se preko storitvenega opravila *Tiskanje računa* natisne račun.

Storitev klic *Opravi plačilo* kliče proces oziroma bazen *Opravi plačilo*, ki bi bil v realnosti del bančnega sistema. Zaradi lažje predstavitve plačila s kreditno kartico je popolnoma avtomatizirano opravilo nadomeščeno z uporabniškim opravilom *Plačilo s kreditno kartico*, kjer se preko obrazca sistemu sporoči ali je plačilo opravljeno ali ne. Če plačilo ni opravljeno, se proces *Opravi plačilo* konča s končnim dogodkom za napako, ki procesu *Sistem veterinarske klinike* sporoči, da plačilo ni bilo uspešno izvedeno.



Slika 5.6: **Model BPMN 2.0, izdelan v Boniti, za proces fiktivnega sistema veterinarske klinike.**

Pri izdelavi izvajalnega modela BPMN 2.0 je potrebno poleg definiranja posameznih spremenljivk, ki se uporabljajo v procesu, pravilno definirati tudi prehode in pogojne tokove zaporedja, da se pri izvajanju izbere primerna pot. Za izvajalne modele BPMN 2.0 je tudi pomembno, da se v vsakem procesu definirajo pobudniki, ki bodo sodelovali pri procesu. Proces *Sistem veterinarske klinike* ima na primer pobudnika, ki se imenuje receptor. Pobudniki so odgovorni za posamezna uporabniška opravila, ki se ne morejo izvesti, če nimajo dodeljenega vsaj enega pobudnika. Vsak končni dogodek z napako je potrebno s posebno kodo za napako povezati s primernim začetnim ali mejnim dogodkom za napako. Popolnoma avtomatizirana opravila (storitveno opravilo, opravilo prejem sporočila, opravilo pošiljanje sporočila, skriptno opravilo in opravilo poslovno pravilo) je potrebno preko priključkov povezati s primerno integracijo, kot sta na primer skripta ali WSDL storitev (Bonita omogoča integracijo z velikim številom obstoječih rešitev).

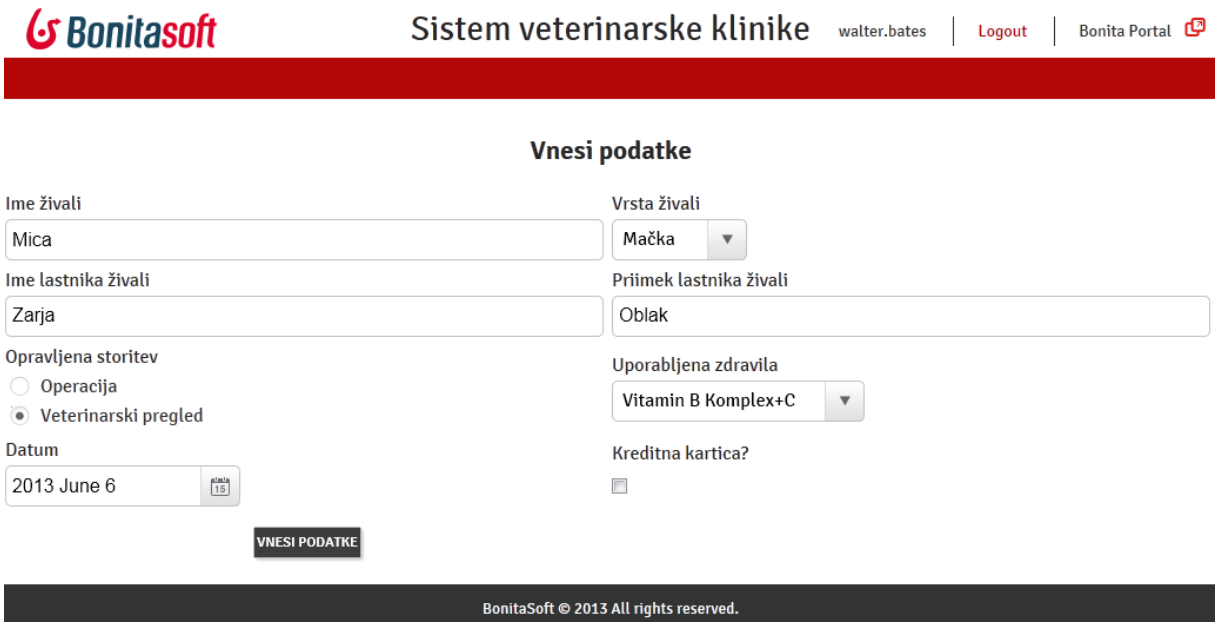


Slika 5.7: **Prikaz modelirnega okolja v Boniti BPM.**

Na sliki 5.7 je prikazan izgled modelirnega okolja v Boniti BPM Community 6.0.2. V desnem kotu spodaj je prikazan del podatkovnih spremenljivk, ki so potrebne za izvajanje procesa *Sistem veterinarske klinike*. V levem kotu spodaj pa so prikazani različni elementi obrazca, ki se prikaže ob začetku izvajanja uporabniškega opravila *Vnesi podatke*. Bonita

BPM ima dobro podporo za izdelavo obrazcev, ki pa kot taki niso del tehnologije BPMN 2.0, ampak služijo kot podpora za implementacijo uporabniških opravil v BPMS-jih.

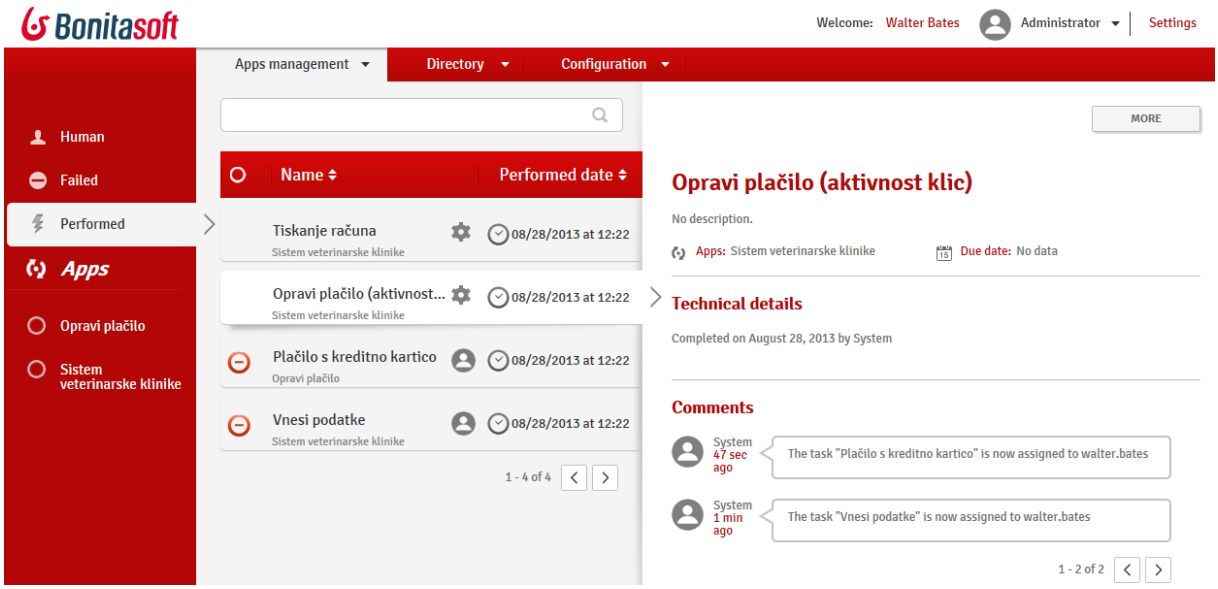
Za pravilno delujočo implementacijo procesov je potrebno za vsako uporabniško opravilo izdelati še obrazec za vnos potrebnih podatkov. Sicer Bonita BPM priskrbi šablono za prikaz obrazcev, vendar omogoča tudi izdelavo popolnoma svojega izgleda obrazca. Na sliki 5.8 je prikazan obrazec za uporabniško opravilo *Vnesi podatke* s primarnim izgledom obrazcev v Boniti BPM.



Slika 5.8: **Obrazec za vnos podatkov uporabniškega opravila *Vnesi podatke* v Boniti BPM.**

Preko Bonitinega portala za upravljanje s poslovnimi procesi je možno sledenje aktivnostim, ki se ob implementaciji izvedejo. Primarno se je na Bonitin portal možno prijaviti kot administrator ali kot uporabnik. Posamezni uporabniki lahko vidijo samo zapise svojih opravil, ki so jih izvedli, administrator pa ima vpogled na vse, kar se dogaja na portalu.

Na sliki 5.9 je prikazan primer Bonitinega portala ob uspešno končanem izvajanju poslovnega procesa *Obisk veterinarske klinike*. Na portal je prijavljen administrator, ki ima vpogled na zapise vseh aktivnosti, ki so se izvedle. Lepo je vidno, da se izvedeta dva poslovna procesa, ki sta *Opravi plačilo* in *Sistem veterinarske klinike*, saj se v procesu *Sistem veterinarske klinike* preko aktivnosti klic *Opravi plačilo* kliče proces *Opravi plačilo*.



Slika 5.9: **Izgled Bonitinega portala za upravljanje s poslovnimi procesi.**

5.6 Standardizirana shema XML

Šele z BPMN različico 2.0 je na voljo standardiziran format za izmenjavo diagramov, kljub temu da je BPMN že od samega začetka standard, ki definira obliko in pomen posameznih simbolov. Predhodne verzije BPMN niso vsebovale sheme XML za izvoz in uvoz modelov, čeprav je združenje WfMC ponudilo standardiziran format XPDL (angl. XML Process Definition Language), ki je namenjen prenosljivosti poslovnih modelov med različnimi orodji. V BPMN 2.0 je problem s prenosljivostjo poslovnih modelov rešen z definicijo standardizirane sheme XML za izmenjavo izvajalnih in neizvajalnih modelov BPMN XML. Izvajalni BPMN 2.0 priskrbi jezik XML kot izmenjevalni format za podatke v procesu in za njegove interakcije z logiko toka [6].

Zaradi boljše podpore specifikacije bo v prihodnosti, oziroma je že, prevladal format BPMN XML. Po idealnem scenariju naj bi to postal način, s katerim bi bile podrobnosti implementacije vključene v izvajalni model, kot tudi način, po katerem bi se lahko modeli BPMN izmenjevali med različnimi orodji BPMS. Vendar v realnosti temu ni točno tako. Shema je kompleksna in trpi zaradi enakih nejasnosti, kot so tiste, ki so prisotne v sami specifikaciji (na primer: ravnanje s podatkovnimi objekti, klic storitev itd.). To je tudi razlog, da je posvojitev te sintakse počasna in neenotna. Mogoče se bo rešitev teh problemov dosegla z uporabo semantičnih spletnih storitev (angl. Semantic Web Technologies) [10].

Za fiktiven primer poslovnega procesa *Sistem veterinarske klinike* je izvajalni model v

formatu BPMN XML prikazan v Dodatku A.

Poglavje 6

SKLEPNE UGOTOVITVE

Organizacije, ki stopajo s časom, se iz funkcionalno naravnanih usmerjajo v procesno naravnane organizacije s pomočjo pristopa BPM, ki je v korelaciji s SOA. S tem se ustvarjajo prožne organizacije, ki imajo sposobnost hitrega prilagajanja poslovnih procesov, kar je v današnjih časih bistvenega pomena. Za realizacijo poslovnih procesov je najbolj primeren BPMN 2.0.

Modeliranje in implementacija poslovnih procesov je kompleksno delo, od katerega je odvisno, kako učinkovito bo izvajanje. Za zagotavljanje optimalnosti pa je dobra definicija poslovnih procesov še pomembnejša od modeliranja in implementacije. Z izvajalnim BPMN 2.0 se je zmanjšala vrzel med modeliranjem in implementacijo poslovnih procesov, saj je BPMN 2.0 za razliko od predhodnih različic BPMN-ja postal ne le notacija, ampak tudi izvajalni jezik za modele poslovnih procesov. BPMN 2.0 je v teoriji skoraj popoln način za modeliranje in implementacijo poslovnih procesov, a v realnosti je slika precej drugačna.

Izvajalni modeli BPMN naj bi se brez problemov prenašali med različnimi platformami BPMS. Kljub dogovorjenim tipom skladnosti in semantiki izvajanja poslovnih procesov pa nekatere platforme BPMS modelom dodajo nekaj svojih specifik, ki običajno niso razumljive drugim platformam. Zaradi takšnih dodatnih specifik je od vsake posamezne platforme odvisno, ali uvozi brez napak model BPMN, ki je bil izdelan na drugi platformi BPMS, ali ne. Poleg tega posamezni BPMS-ji podpirajo le del konstruktov BPMN 2.0 s semantiko izvajanja. Modeliranje poenostavljenega fiktivnega primera na platformi Bonita BPM Community je pokazalo, da Bonita omogoča dokaj enostavno modeliranje poslovnih procesov. Ena izmed pomanjkljivosti je ta, da ni mogoče izdelati procesa, ki vsebuje ročno opravilo. Bonita torej podpira le modeliranje avtomatiziranih in uporabniških opravil v procesu. Poleg tega ni mogoča izdelava procesa z navadnim podprocesom,

ampak je namesto tega potrebno uporabiti aktivnost klic. Zadnja opažena pomanjkljivost je, da med artefakti, ki so sestavni del BPMN 2.0, Bonita ponuja le tekstovni komentar. Sicer pa je izdelava izvajalnega modela BPMN enostavna. Ena izmed dobrih lastnosti Bonite BPM je enostavna izdelava obrazcev za uporabniška opravila, saj platforma ponuja dober urejevalnik za obrazce. Izgled obrazcev ni vezan le na primarno obliko, ampak se lahko s pomočjo kode HTML po meri prilagaja. Po implementaciji in izvajanju poslovnih procesov je na Bonitinem portalu za upravljanje poslovnih procesov možen enostaven pregled podatkov izvedenih aktivnosti ter določanje pravic posameznim uporabnikom, ki rokujejo z uporabniškimi opravili.

Bonita BPM Community je bila za izdelavo fiktivnega primera poslovnega procesa *Sistem veterinarske klinike* ena izmed najboljših izbir med BPMS-ji, saj je omogočila dobro in enostavno uporabniško izkušnjo ter zmogljiv procesni stroj. Kljub temu pa zaradi odsotnosti podpore nekaterim konstruktom ni bila primerna za izdelavo celotnega abstraktnega modela za primer obiska veterinarske klinike. Na podlagi abstraktnih modelov primera, ki smo ga modelirali v Microsoft Visio 2010, so bili predstavljeni nekateri konstrukti BPMN 2.0 in trije različni diagrami: diagram sodelovanja, diagram pogovora in diagram koreografije.

Čeprav BPMN 2.0 ponuja več kot sto različnih konstruktov za modeliranje, se tudi pri obsežnih modelih uporablja le del konstruktov, saj so nekateri konstrukti uporabni le v zelo specifičnih modelih. Različni BPMS-ji torej ravno zaradi specifičnosti in manjše uporabnosti določenih konstruktov podpirajo le tiste konstrukte BPMN 2.0, ki se najbolj obsežno uporabljajo pri modeliranju poslovnih procesov. Popolnoma avtomatizirana opravila se lahko v celoti prenašajo med različnimi BPMS-ji brez posebnih težav, če noben izmed vključenih BPMS-jev ne doda posebnih specifik svojim modelom. Po drugi strani polavtomatizirana oziroma uporabniška opravila omogočajo le prenos logike toka, implementacija uporabniških opravil pa je prepuščena vsakemu posameznemu BPMS-ju. To je mogoče rešiti z enotnim standardom. Takšen standard je standard WS-HT, vendar je realizacija tega standarda zelo kompleksna, kar je tudi razlog, da ga podpira le majhno število platform BPMS. Ko bodo BPMS-ji podpirali še enoten standard za celoten prenos uporabniških opravil med različnimi BPMS-ji, bodo modeli poslovnih procesov BPMN 2.0 resnično prenosljivi, vsaj v kontekstu z najbolj uporabljivimi konstrukti BPMN 2.0, ki bi jih podpirali vsi udeleženi BPMS-ji. BPMN 2.0 je kompleksen jezik in platforme BPMS se mu le počasi prilagajajo. BPMS-ji imajo pred seboj še dolgo pot, preden bodo resnično v celoti podpirali izvajalni BPMN 2.0, oziroma obstaja velika verjetnost, da ga zaradi visoke kompleksnosti ne bodo nikoli v celoti podpirali.

Dodatek A

PROCES: *Sistem veterinarske klinike*

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <model:definitions xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xmlns:bonitaConnector="http://www.bonitasoft.org/studio/connector/definition/6.0"
      xmlns:dc="http://www.omg.org/spec/DD/20100524/DC" xmlns:di="http://www.omg.
      org/spec/BPMN/20100524/DI" xmlns:di_1="http://www.omg.org/spec/DD/20100524/DI"
      xmlns:java="http://jcp.org/en/jsr/detail?id=270" xmlns:model="http://www.omg.
      org/spec/BPMN/20100524/MODEL" xsi:schemaLocation="schemaLocation http://www.omg.
      org/spec/BPMN/20100524/MODEL schemas/BPMN20.xsd" exporter="BonitaSoft" exporterVersion=
      "6.0.2" expressionLanguage="http://groovy.codehaus.org/" targetNamespace="
      http://bonitasoft.com/_C9hcIf7pEeKj3Z50sywxHw">
3   <model:import importType="http://www.w3.org/2001/XMLSchema" location="
        connectorDefs/scripting-groovy.defconnectors.xsd" namespace="http://www.bonitasoft.
        org/studio/connector/definition/6.0"/>
4   <model:collaboration id="_C9hcIf7pEeKj3Z50sywxHw">
5     <model:participant id="_fvUXERJyEeO7dLOhtwHdzg" name="Sistem veterinarske klinike"
          processRef="_DL3aIP7pEeKj3Z50sywxHw"/>
6     <model:participant id="_DVTC8P7pEeKj3Z50sywxHw" name="Employee actor">
7       <model:documentation>This is an example of actor that is mapped to any ACME users<
            /model:documentation>
8     </model:participant>
9     <model:participant id="_gRqC4AnREeO5s8M9bY-Egw" name="Receptor"/>
10    <model:participant id="_f2VlFRJyEeO7dLOhtwHdzg" name="Opravi plačilo" processRef="
          _MdzSUAnbEeO5s8M9bY-Egw"/>
11    <model:participant id="_GDX_sA3EEeOw5LUhyu_WZg" name="hoj"/>
12  </model:collaboration>
13  <model:process id="_DL3aIP7pEeKj3Z50sywxHw" name="Sistem veterinarske klinike">
14    <model:ioSpecification id="_fvadsBJyEeO7dLOhtwHdzg">
15      <model:dataInput id="_fvbEwBJyEeO7dLOhtwHdzg" itemSubjectRef="_VQG1AAzGEeOy3Jek3EKZ4A
            "/>
16      <model:dataInput id="_fvdhAxJyEeO7dLOhtwHdzg" itemSubjectRef="_2ibYMAzwEeOCxcUUuhGWDw
            "/>
17      <model:dataInput id="_fveIFBJyEeO7dLOhtwHdzg" itemSubjectRef="_ztkjkAzzEeOCxcUUuhGWDw
            "/>
18      <model:dataInput id="_fveIGhJyEeO7dLOhtwHdzg" itemSubjectRef="_DYwDAAzOEeOCxcUUuhGWDw
            "/>
19      <model:dataInput id="_fveIIBJyEeO7dLOhtwHdzg" itemSubjectRef="_F8fyIAzOEeOCxcUUuhGWDw
            "/>
20      <model:dataInput id="_fveIJhJyEeO7dLOhtwHdzg" itemSubjectRef="_HIxYAAzOEeOCxcUUuhGWDw
            "/>
21      <model:dataInput id="_fveILBJyEeO7dLOhtwHdzg" itemSubjectRef="_in4XEA15EeO9cbH4334SDg
            "/>
22      <model:dataInput id="_fveIMhJyEeO7dLOhtwHdzg" itemSubjectRef="__bP78A_CEeOUDM6UcVOF9Q
            "/>
23      <model:inputSet id="_fvbrOBJyEeO7dLOhtwHdzg">
24        <model:dataInputRefs>_fvbEwBJyEeO7dLOhtwHdzg</model:dataInputRefs>
```

```
25        </model:inputSet>
26        <model:inputSet id="_fvdhBBJyEeO7dL0htwHdzg">
27          <model:dataInputRefs>_fvdhAxJyEeO7dL0htwHdzg</model:dataInputRefs>
28        </model:inputSet>
29        <model:inputSet id="_fveIFRJyEeO7dL0htwHdzg">
30          <model:dataInputRefs>_fveIFBJyEeO7dL0htwHdzg</model:dataInputRefs>
31        </model:inputSet>
32        <model:inputSet id="_fveIGxJyEeO7dL0htwHdzg">
33          <model:dataInputRefs>_fveIGhJyEeO7dL0htwHdzg</model:dataInputRefs>
34        </model:inputSet>
35        <model:inputSet id="_fveIIRJyEeO7dL0htwHdzg">
36          <model:dataInputRefs>_fveIIBJyEeO7dL0htwHdzg</model:dataInputRefs>
37        </model:inputSet>
38        <model:inputSet id="_fveIJxJyEeO7dL0htwHdzg">
39          <model:dataInputRefs>_fveIJhJyEeO7dL0htwHdzg</model:dataInputRefs>
40        </model:inputSet>
41        <model:inputSet id="_fveILRJyEeO7dL0htwHdzg">
42          <model:dataInputRefs>_fveILBJyEeO7dL0htwHdzg</model:dataInputRefs>
43        </model:inputSet>
44        <model:inputSet id="_fveIMxJyEeO7dL0htwHdzg">
45          <model:dataInputRefs>_fveIMhJyEeO7dL0htwHdzg</model:dataInputRefs>
46        </model:inputSet>
47        <model:outputSet id="_fvevIBJyEeO7dL0htwHdzg"/>
48      </model:ioSpecification>
49      <model:dataObject id="DataObject_fvZ2oBJyEeO7dL0htwHdzg_VQG1AAzGEeOy3Jek3EKZ4A" name="
            kreditnaKartica" isCollection="false" itemSubjectRef="_VQG1AAzGEeOy3Jek3EKZ4A"/>
50      <model:dataObject id="DataObject_fvdhAhJyEeO7dL0htwHdzg_2ibYMAzwEeOCxcUUuhGWDw" name="
            datum" isCollection="false" itemSubjectRef="_2ibYMAzwEeOCxcUUuhGWDw"/>
51      <model:dataObject id="DataObject_fveIExJyEeO7dL0htwHdzg_ztkjkAzzEeOCxcUUuhGWDw" name="
            zdravila" isCollection="false" itemSubjectRef="_ztkjkAzzEeOCxcUUuhGWDw"/>
52      <model:dataObject id="DataObject_fveIGRJyEeO7dL0htwHdzg_DYwDAAzOEeOCxcUUuhGWDw" name="
            imeLastnikaZivali" isCollection="false" itemSubjectRef="_DYwDAAzOEeOCxcUUuhGWDw"/>
53      <model:dataObject id="DataObject_fveIHxJyEeO7dL0htwHdzg_F8fyIAzOEeOCxcUUuhGWDw" name="
            priimekLastnikaZivali" isCollection="false" itemSubjectRef="_F8fyIAzOEeOCxcUUuhGWDw
            "/>
54      <model:dataObject id="DataObject_fveIJRJyEeO7dL0htwHdzg_HIxYAAzOEeOCxcUUuhGWDw" name="
            imeZivali" isCollection="false" itemSubjectRef="_HIxYAAzOEeOCxcUUuhGWDw"/>
55      <model:dataObject id="DataObject_fveIKxJyEeO7dL0htwHdzg_in4XEA15EeO9cbH4334SDg" name="
            storitev" isCollection="false" itemSubjectRef="_in4XEA15EeO9cbH4334SDg"/>
56      <model:dataObject id="DataObject_fveIMRJyEeO7dL0htwHdzg__bP78A_CEeOUDM6UcVOF9Q" name="
            zival" isCollection="false" itemSubjectRef="__bP78A_CEeOUDM6UcVOF9Q"/>
57      <model:userTask id="_TLEnIAnREeO5s8M9bY-Egw" name="Vnesi podatke">
58        <model:performer id="_fwWROBJyEeO7dL0htwHdzg">
59          <model:resourceRef>_gRqC4AnREeO5s8M9bY-Egw</model:resourceRef>
60        </model:performer>
61      </model:userTask>
62      <model:startEvent id="_S8IEOAnUEeO5s8M9bY-Egw" name="Začetek"/>
63      <model:exclusiveGateway id="_-Eki8AnaEeO5s8M9bY-Egw" name="Plačilo s kreditno kartico?"
            default="_YtcfAAnbEeO5s8M9bY-Egw"/>
64      <model:callActivity id="_MiYQUAnbEeO5s8M9bY-Egw" name="Opravi plačilo" calledElement="
            Opravi plačilo">
65        <model:dataInputAssociation id="_fwZ8MBJyEeO7dL0htwHdzg">
66          <model:assignment>
67            <model:from id="_fwZ8MRJyEeO7dL0htwHdzg">_VQG1AAzGEeOy3Jek3EKZ4A</model:from>
68            <model:to id="_fwy9wBJyEeO7dL0htwHdzg">_pfGRAA4-EeOSvYyPCyjKDQ</model:to>
69          </model:assignment>
70        </model:dataInputAssociation>
71        <model:dataOutputAssociation id="_fwzkOBJyEeO7dL0htwHdzg">
72          <model:assignment>
73            <model:from id="_fxBnQBJyEeO7dL0htwHdzg">_pfGRAA4-EeOSvYyPCyjKDQ</model:from>
74            <model:to id="_fxBnQRJyEeO7dL0htwHdzg">_VQG1AAzGEeOy3Jek3EKZ4A</model:to>
75          </model:assignment>
76        </model:dataOutputAssociation>
77      </model:callActivity>
```

```
78    <model:boundaryEvent id="_Z7UnYAnbEeO5s8M9bY-Egw" name="Zavrnjeno plačilo s kreditno
          kartico" attachedToRef="_Z7UnYAnbEeO5s8M9bY-Egw" cancelActivity="true">
79      <model:errorEventDefinition id="eventdef-Zavrnjeno plačilo s kreditno
            kartico_fxC1YBJyEeO7dL0htwHdzg" errorRef="Ni placila"/>
80    </model:boundaryEvent>
81    <model:exclusiveGateway id="_oTJzIAncEeO5s8M9bY-Egw" name="Zbirni prehod" default="
          _njR_8AndEeO5s8M9bY-Egw"/>
82    <model:endEvent id="_p2SkYAndEeO5s8M9bY-Egw" name="Konec"/>
83    <model:serviceTask id="_FHBxsAneEeO5s8M9bY-Egw" name="Tiskanje računa" implementation="
          BonitaConnector" operationRef="Execscripting-groovy">
84      <model:ioSpecification id="_f2RToBJyEeO7dL0htwHdzg">
85        <model:dataInput id="_f2RToRJyEeO7dL0htwHdzg" itemSubjectRef="scripting-
              groovyConnectorInput"/>
86        <model:dataOutput id="_f2RToxJyEeO7dL0htwHdzg" itemSubjectRef="scripting-
              groovyConnectorOutput"/>
87        <model:inputSet id="_f2RTohJyEeO7dL0htwHdzg">
88          <model:dataInputRefs>_f2RToRJyEeO7dL0htwHdzg</model:dataInputRefs>
89        </model:inputSet>
90        <model:outputSet id="_f2RTpBJyEeO7dL0htwHdzg">
91          <model:dataOutputRefs>_f2RToxJyEeO7dL0htwHdzg</model:dataOutputRefs>
92        </model:outputSet>
93      </model:ioSpecification>
94      <model:dataInputAssociation>
95        <model:targetRef>_f2RToRJyEeO7dL0htwHdzg</model:targetRef>
96        <model:assignment>
97          <model:from xsi:type="model:tFormalExpression" id="_f2R6sBJyEeO7dL0htwHdzg"
                evaluatesToTypeRef="java:java.lang.Object">println &amp;quot;Tiskalnik tiska rač
                un.&amp;quot;</model:from>
98          <model:to id="_f2R6sRJyEeO7dL0htwHdzg">getDataInput('_f2RToRJyEeO7dL0htwHdzg')
                /bonitaConnector:script</model:to>
99        </model:assignment>
100     </model:dataInputAssociation>
101   </model:serviceTask>
102   <model:userTask id="_bMVn8A48EeOSvYyPCyjKDQ" name="Ponovno poskusi">
103     <model:performer id="_f2R6shJyEeO7dL0htwHdzg">
104       <model:resourceRef>_gRqC4AnREeO5s8M9bY_Egw</model:resourceRef>
105     </model:performer>
106   </model:userTask>
107   <model:sequenceFlow id="_av_CsAnUEeO5s8M9bY-Egw" name="" sourceRef="_S8IEOAnUEeO5s8M9bY
          -Egw" targetRef="_TLEnIAnREeO5s8M9bY-Egw">
108     <model:conditionExpression xsi:type="model:tFormalExpression" id="
            _f2Tv4BJyEeO7dL0htwHdzg" evaluatesToTypeRef="java:java.lang.Boolean" language="
            http://www.w3.org/1999/XPath"></model:conditionExpression>
109   </model:sequenceFlow>
110   <model:sequenceFlow id="_YtcfAAnbEeO5s8M9bY-Egw" name="DA" sourceRef="_-
          Eki8AnaEeO5s8M9bY-Egw" targetRef="_MiYQUAnbEeO5s8M9bY-Egw">
111     <model:conditionExpression xsi:type="model:tFormalExpression" id="_f2U-
            ABJyEeO7dL0htwHdzg" evaluatesToTypeRef="java:java.lang.Boolean" language="
            http://www.w3.org/1999/XPath">kreditnaKartica==true</model:conditionExpression>
112   </model:sequenceFlow>
113   <model:sequenceFlow id="_qjW7MAncEeO5s8M9bY-Egw" name="" sourceRef="_MiYQUAnbEeO5s8M9bY
          -Egw" targetRef="_oTJzIAncEeO5s8M9bY-Egw">
114     <model:conditionExpression xsi:type="model:tFormalExpression" id="_f2U-
            ARJyEeO7dL0htwHdzg" evaluatesToTypeRef="java:java.lang.Boolean" language="
            http://www.w3.org/1999/XPath"></model:conditionExpression>
115   </model:sequenceFlow>
116   <model:sequenceFlow id="_sXLCMAncEeO5s8M9bY-Egw" name="NE" sourceRef="_-
          Eki8AnaEeO5s8M9bY-Egw" targetRef="_oTJzIAncEeO5s8M9bY-Egw">
117     <model:conditionExpression xsi:type="model:tFormalExpression" id="
            _f2VlEBJyEeO7dL0htwHdzg" evaluatesToTypeRef="java:java.lang.Boolean" language="
            http://www.w3.org/1999/XPath">kreditnaKartica==false</model:conditionExpression>
118   </model:sequenceFlow>
119   <model:sequenceFlow id="_njR_8AndEeO5s8M9bY-Egw" name="" sourceRef="_oTJzIAncEeO5s8M9bY
          -Egw" targetRef="_FHBxsAneEeO5s8M9bY-Egw">
```

```
120       <model:conditionExpression xsi:type="model:tFormalExpression" id="
              _f2VlERJyEeO7dL0htwHdzg" evaluatesToTypeRef="java:java.lang.Boolean" language="
              http://www.w3.org/1999/XPath"></model:conditionExpression>
121     </model:sequenceFlow>
122     <model:sequenceFlow id="_smfbIAndEeO5s8M9bY-Egw" name="" sourceRef="_FHBxsAneEeO5s8M9bY
            -Egw" targetRef="_p2SkYAndEeO5s8M9bY-Egw">
123       <model:conditionExpression xsi:type="model:tFormalExpression" id="
              _f2VlEhJyEeO7dL0htwHdzg" evaluatesToTypeRef="java:java.lang.Boolean" language="
              http://www.w3.org/1999/XPath"></model:conditionExpression>
124     </model:sequenceFlow>
125     <model:sequenceFlow id="_c6E8IA48EeOSvYyPCyjKDQ" name="" sourceRef="_Z7UnYAnbEeO5s8M9bY
            -Egw" targetRef="_bMVn8A48EeOSvYyPCyjKDQ"/>
126     <model:sequenceFlow id="_wvit4A5AEeODMICosmCfsg" name="" sourceRef="
            _bMVn8A48EeOSvYyPCyjKDQ" targetRef="_-Eki8AnaEeO5s8M9bY-Egw">
127       <model:conditionExpression xsi:type="model:tFormalExpression" id="
              _f2VlExJyEeO7dL0htwHdzg" evaluatesToTypeRef="java:java.lang.Boolean" language="
              http://www.w3.org/1999/XPath"></model:conditionExpression>
128     </model:sequenceFlow>
129     <model:sequenceFlow id="_y0DjUA5AEeODMICosmCfsg" name="" sourceRef="_TLEnIAnREeO5s8M9bY
            -Egw" targetRef="_-Eki8AnaEeO5s8M9bY-Egw">
130       <model:conditionExpression xsi:type="model:tFormalExpression" id="
              _f2VlFBJyEeO7dL0htwHdzg" evaluatesToTypeRef="java:java.lang.Boolean" language="
              http://www.w3.org/1999/XPath"></model:conditionExpression>
131     </model:sequenceFlow>
132   </model:process>
133   <model:itemDefinition id="_VQG1AAzGEeOy3Jek3EKZ4A" structureRef="java:java.lang.Boolean"/
          >
134   <model:itemDefinition id="_2ibYMAzwEeOCxcUUuhGWDw" structureRef="java:java.util.Date"/>
135   <model:itemDefinition id="_ztkjkAzzEeOCxcUUuhGWDw" structureRef="java:java.lang.String"/>
136   <model:itemDefinition id="_DYwDAAz0EeOCxcUUuhGWDw" structureRef="java:java.lang.String"/>
137   <model:itemDefinition id="_F8fyIAz0EeOCxcUUuhGWDw" structureRef="java:java.lang.String"/>
138   <model:itemDefinition id="_HIxYAAz0EeOCxcUUuhGWDw" structureRef="java:java.lang.String"/>
139   <model:itemDefinition id="_in4XEA15EeO9cbH4334SDg" structureRef="java:java.lang.String"/>
140   <model:itemDefinition id="__bP78A_CEeOUDM6UcVOF9Q" structureRef="java:java.lang.String"/>
141   <model:itemDefinition id="scripting-groovyConnectorInput" structureRef="
          bonitaConnector:scripting-groovyInputType"/>
142   <model:message id="scripting-groovyConnectorMessageInput" itemRef="scripting-
          groovyConnectorInput"/>
143   <model:itemDefinition id="scripting-groovyConnectorOutput" structureRef="
          bonitaConnector:scripting-groovyOutputType"/>
144   <model:message id="scripting-groovyConnectorMessageOutput" itemRef="scripting-
          groovyConnectorOutput"/>
145   <model:interface id="scripting-groovy_Bonita_Connector_Interface" name="scripting-
          groovy_Bonita_Connector_Interface">
146     <model:operation id="Execscripting-groovy" name="Execscripting-groovy">
147       <model:inMessageRef>scripting-groovyConnectorMessageInput</model:inMessageRef>
148       <model:outMessageRef>scripting-groovyConnectorMessageOutput</model:outMessageRef>
149     </model:operation>
150   </model:interface>
151   <model:process id="_MdzSUAnbEeO5s8M9bY-Egw" name="Opravi plačilo">
152     <model:ioSpecification id="_f2VlFxJyEeO7dL0htwHdzg">
153       <model:dataInput id="_f2VlGBJyEeO7dL0htwHdzg" itemSubjectRef="_nHvqoA3nEeOw5LUhyu_WZg
              "/>
154       <model:dataInput id="_f2WMIhJyEeO7dL0htwHdzg" itemSubjectRef="_HEvJYAzwEeOCxcUUuhGWDw
              "/>
155       <model:dataInput id="_f2WMKBJyEeO7dL0htwHdzg" itemSubjectRef="_pfGRAA4-EeOSvYyPCyjKDQ
              "/>
156       <model:inputSet id="_f2VlGRJyEeO7dL0htwHdzg">
157         <model:dataInputRefs>_f2VlGBJyEeO7dL0htwHdzg</model:dataInputRefs>
158       </model:inputSet>
159       <model:inputSet id="_f2WMIxJyEeO7dL0htwHdzg">
160         <model:dataInputRefs>_f2WMIhJyEeO7dL0htwHdzg</model:dataInputRefs>
161       </model:inputSet>
162       <model:inputSet id="_f2WMKRJyEeO7dL0htwHdzg">
```

```
163         <model:dataInputRefs>_f2WMKBJyEeO7dLOhtwHdzg</model:dataInputRefs>
164       </model:inputSet>
165       <model:outputSet id="_f2WMLRJyEeO7dLOhtwHdzg"/>
166     </model:ioSpecification>
167     <model:dataObject id="DataObject_f2VlFhJyEeO7dLOhtwHdzg_nHvqoA3nEeOw5LUhyu_WZg" name="
            opravljeno" isCollection="false" itemSubjectRef="_nHvqoA3nEeOw5LUhyu_WZg"/>
168     <model:dataObject id="DataObject_f2WMIRJyEeO7dLOhtwHdzg_HEvJYAzwEeOCxcUUuhGWDw" name="
            stevilkaKartice" isCollection="false" itemSubjectRef="_HEvJYAzwEeOCxcUUuhGWDw"/>
169     <model:dataObject id="DataObject_f2WMJxJyEeO7dLOhtwHdzg_pfGRAA4-EeOSvYyPCyjKDQ" name="
            kreditna" isCollection="false" itemSubjectRef="_pfGRAA4-EeOSvYyPCyjKDQ"/>
170     <model:startEvent id="_pLgz8AnbEeO5s8M9bY-Egw" name="ZacetekPlacilo"/>
171     <model:exclusiveGateway id="_C7lfwAncEeO5s8M9bY-Egw" name="Plačilo opravljeno?" default
            ="_OhAqwAncEeO5s8M9bY-Egw"/>
172     <model:endEvent id="_NAHfwAncEeO5s8M9bY-Egw" name="konecPlacilo"/>
173     <model:endEvent id="_SIGtQAncEeO5s8M9bY-Egw" name="Plačilo ni mogoče">
174       <model:errorEventDefinition id="Ni placila_f2WzMBJyEeO7dLOhtwHdzg" errorRef="Ni
              placila"/>
175     </model:endEvent>
176     <model:userTask id="_42tKQA3oEeOw5LUhyu_WZg" name="Plačilo s kreditno kartico">
177       <model:performer id="_f2WzMRJyEeO7dLOhtwHdzg">
178         <model:resourceRef>_GDX_sA3EEeOw5LUhyu_WZg</model:resourceRef>
179       </model:performer>
180     </model:userTask>
181     <model:sequenceFlow id="_Ad-NOAncEeO5s8M9bY-Egw" name="" sourceRef="_pLgz8AnbEeO5s8M9bY
            -Egw" targetRef="_42tKQA3oEeOw5LUhyu_WZg">
182       <model:conditionExpression xsi:type="model:tFormalExpression" id="
              _f2WzMhJyEeO7dLOhtwHdzg" evaluatesToTypeRef="java:java.lang.Boolean" language="
              http://www.w3.org/1999/XPath"></model:conditionExpression>
183     </model:sequenceFlow>
184     <model:sequenceFlow id="_LQCpQAncEeO5s8M9bY-Egw" name="" sourceRef="
            _42tKQA3oEeOw5LUhyu_WZg" targetRef="_C7lfwAncEeO5s8M9bY-Egw">
185       <model:conditionExpression xsi:type="model:tFormalExpression" id="
              _f2XaQBJyEeO7dLOhtwHdzg" evaluatesToTypeRef="java:java.lang.Boolean" language="
              http://www.w3.org/1999/XPath"></model:conditionExpression>
186     </model:sequenceFlow>
187     <model:sequenceFlow id="_OhAqwAncEeO5s8M9bY-Egw" name="DA" sourceRef="
            _C7lfwAncEeO5s8M9bY-Egw" targetRef="_NAHfwAncEeO5s8M9bY-Egw">
188       <model:conditionExpression xsi:type="model:tFormalExpression" id="
              _f2XaQRJyEeO7dLOhtwHdzg" evaluatesToTypeRef="java:java.lang.Boolean" language="
              http://www.w3.org/1999/XPath"></model:conditionExpression>
189     </model:sequenceFlow>
190     <model:sequenceFlow id="_VU4LQAncEeO5s8M9bY-Egw" name="NE" sourceRef="
            _C7lfwAncEeO5s8M9bY-Egw" targetRef="_SIGtQAncEeO5s8M9bY-Egw">
191       <model:conditionExpression xsi:type="model:tFormalExpression" id="
              _f2XaQhJyEeO7dLOhtwHdzg" evaluatesToTypeRef="java:java.lang.Boolean" language="
              http://www.w3.org/1999/XPath">opravljeno==false</model:conditionExpression>
192     </model:sequenceFlow>
193   </model:process>
194   <model:itemDefinition id="_nHvqoA3nEeOw5LUhyu_WZg" structureRef="java:java.lang.Boolean"/
          >
195   <model:itemDefinition id="_HEvJYAzwEeOCxcUUuhGWDw" structureRef="java:java.lang.Integer"/
          >
196   <model:itemDefinition id="_pfGRAA4-EeOSvYyPCyjKDQ" structureRef="java:java.lang.Boolean"/
          >
197   <di:BPMNDiagram name="MyDiagram">
198     <di:BPMNPlane id="plane__C9hcIf7pEeKj3Z50sywxHw" bpmnElement="_C9hcIf7pEeKj3Z50sywxHw">
199       <di:BPMNShape id="_DPNpEP7pEeKj3Z50sywxHw" bpmnElement="_DL3aIP7pEeKj3Z50sywxHw"
              isHorizontal="true">
200         <dc:Bounds height="305.0" width="1000.0" x="30.0" y="30.0"/>
201       </di:BPMNShape>
202       <di:BPMNShape id="_TLIRgAnREeO5s8M9bY-Egw" bpmnElement="_TLEnIAnREeO5s8M9bY-Egw">
203         <dc:Bounds height="67.0" width="134.0" x="152.0" y="76.0"/>
204       </di:BPMNShape>
205       <di:BPMNShape id="_S8M9UAnUEeO5s8M9bY-Egw" bpmnElement="_S8IEOAnUEeO5s8M9bY-Egw">
```

```
206         <dc:Bounds height="30.0" width="30.0" x="83.0" y="95.0"/>
207       </di:BPMNShape>
208       <di:BPMNShape id="_-EnmQAnaEeO5s8M9bY-Egw" bpmnElement="_-Eki8AnaEeO5s8M9bY-Egw">
209         <dc:Bounds height="43.0" width="43.0" x="342.0" y="89.0"/>
210       </di:BPMNShape>
211       <di:BPMNShape id="_MiaskAnbEeO5s8M9bY-Egw" bpmnElement="_MiYQUAnbEeO5s8M9bY-Egw">
212         <dc:Bounds height="61.0" width="122.0" x="437.0" y="80.0"/>
213       </di:BPMNShape>
214       <di:BPMNShape id="_Z7XqsAnbEeO5s8M9bY-Egw" bpmnElement="_Z7UnYAnbEeO5s8M9bY-Egw">
215         <dc:Bounds height="25.0" width="25.0" x="528.0" y="133.0"/>
216       </di:BPMNShape>
217       <di:BPMNShape id="_oTNdgAncEeO5s8M9bY-Egw" bpmnElement="_oTJzIAncEeO5s8M9bY-Egw">
218         <dc:Bounds height="43.0" width="43.0" x="589.0" y="89.0"/>
219       </di:BPMNShape>
220       <di:BPMNShape id="_p2TygAndEeO5s8M9bY-Egw" bpmnElement="_p2SkYAndEeO5s8M9bY-Egw">
221         <dc:Bounds height="30.0" width="30.0" x="836.0" y="95.0"/>
222       </di:BPMNShape>
223       <di:BPMNShape id="_FHE1AAneEeO5s8M9bY-Egw" bpmnElement="_FHBxsAneEeO5s8M9bY-Egw">
224         <dc:Bounds height="58.0" width="116.0" x="684.0" y="82.0"/>
225       </di:BPMNShape>
226       <di:BPMNShape id="_bMgnEA48EeOSvYyPCyjKDQ" bpmnElement="_bMVn8A48EeOSvYyPCyjKDQ">
227         <dc:Bounds height="72.0" width="144.0" x="323.0" y="209.0"/>
228       </di:BPMNShape>
229       <di:BPMNEdge id="_awBe8AnUEeO5s8M9bY-Egw" bpmnElement="_av_CsAnUEeO5s8M9bY-Egw">
230         <di_1:waypoint x="113.0" y="110.0"/>
231         <di_1:waypoint x="152.0" y="110.0"/>
232       </di:BPMNEdge>
233       <di:BPMNEdge id="_Yte7QAnbEeO5s8M9bY-Egw" bpmnElement="_YtcfAAnbEeO5s8M9bY-Egw">
234         <di_1:waypoint x="384.0" y="110.0"/>
235         <di_1:waypoint x="437.0" y="110.0"/>
236       </di:BPMNEdge>
237       <di:BPMNEdge id="_qjZ-gAncEeO5s8M9bY-Egw" bpmnElement="_qjW7MAncEeO5s8M9bY-Egw">
238         <di_1:waypoint x="559.0" y="111.0"/>
239         <di_1:waypoint x="589.0" y="111.0"/>
240       </di:BPMNEdge>
241       <di:BPMNEdge id="_sXXPcAncEeO5s8M9bY-Egw" bpmnElement="_sXLCMAncEeO5s8M9bY-Egw">
242         <di_1:waypoint x="363.0" y="89.0"/>
243         <di_1:waypoint x="363.0" y="57.0"/>
244         <di_1:waypoint x="610.0" y="57.0"/>
245         <di_1:waypoint x="610.0" y="89.0"/>
246       </di:BPMNEdge>
247       <di:BPMNEdge id="_njT1IAndEeO5s8M9bY-Egw" bpmnElement="_njR_8AndEeO5s8M9bY-Egw">
248         <di_1:waypoint x="632.0" y="109.0"/>
249         <di_1:waypoint x="684.0" y="109.0"/>
250       </di:BPMNEdge>
251       <di:BPMNEdge id="_smiecAndEeO5s8M9bY-Egw" bpmnElement="_smfbIAndEeO5s8M9bY-Egw">
252         <di_1:waypoint x="800.0" y="109.0"/>
253         <di_1:waypoint x="836.0" y="109.0"/>
254       </di:BPMNEdge>
255       <di:BPMNEdge id="_c6MQ4A48EeOSvYyPCyjKDQ" bpmnElement="_c6E8IA48EeOSvYyPCyjKDQ">
256         <di_1:waypoint x="539.0" y="158.0"/>
257         <di_1:waypoint x="539.0" y="244.0"/>
258         <di_1:waypoint x="467.0" y="244.0"/>
259       </di:BPMNEdge>
260       <di:BPMNEdge id="_wvqCoA5AEeODMICosmCfsg" bpmnElement="_wvit4A5AEeODMICosmCfsg">
261         <di_1:waypoint x="364.0" y="209.0"/>
262         <di_1:waypoint x="364.0" y="131.0"/>
263       </di:BPMNEdge>
264       <di:BPMNEdge id="_y0GmoA5AEeODMICosmCfsg" bpmnElement="_y0DjUA5AEeODMICosmCfsg">
265         <di_1:waypoint x="286.0" y="111.0"/>
266         <di_1:waypoint x="342.0" y="111.0"/>
267       </di:BPMNEdge>
268       <di:BPMNShape id="_MiTX0AnbEeO5s8M9bY-Egw" bpmnElement="_MdzSUAnbEeO5s8M9bY-Egw"
            isHorizontal="true">
```

```
269         <dc:Bounds height="250.0" width="1000.0" x="30.0" y="355.0"/>
270       </di:BPMNShape>
271       <di:BPMNShape id="_pLj3QAnbEeO5s8M9bY-Egw" bpmnElement="_pLgz8AnbEeO5s8M9bY-Egw">
272         <dc:Bounds height="30.0" width="30.0" x="78.0" y="472.0"/>
273       </di:BPMNShape>
274       <di:BPMNShape id="_C7ojEAncEeO5s8M9bY-Egw" bpmnElement="_C7lfwAncEeO5s8M9bY-Egw">
275         <dc:Bounds height="43.0" width="43.0" x="410.0" y="465.0"/>
276       </di:BPMNShape>
277       <di:BPMNShape id="_NAKjEAncEeO5s8M9bY-Egw" bpmnElement="_NAHfwAncEeO5s8M9bY-Egw">
278         <dc:Bounds height="30.0" width="30.0" x="543.0" y="472.0"/>
279       </di:BPMNShape>
280       <di:BPMNShape id="_SIJwkAncEeO5s8M9bY-Egw" bpmnElement="_SIGtQAncEeO5s8M9bY-Egw">
281         <dc:Bounds height="30.0" width="30.0" x="543.0" y="374.0"/>
282       </di:BPMNShape>
283       <di:BPMNShape id="_4227QA3oEeOw5LUhyu_WZg" bpmnElement="_42tKQA3oEeOw5LUhyu_WZg">
284         <dc:Bounds height="70.0" width="140.0" x="179.0" y="452.0"/>
285       </di:BPMNShape>
286       <di:BPMNEdge id="_AeAqEAncEeO5s8M9bY-Egw" bpmnElement="_Ad-N0AncEeO5s8M9bY-Egw">
287         <di_1:waypoint x="108.0" y="487.0"/>
288         <di_1:waypoint x="179.0" y="487.0"/>
289       </di:BPMNEdge>
290       <di:BPMNEdge id="_LQFFgAncEeO5s8M9bY-Egw" bpmnElement="_LQCpQAncEeO5s8M9bY-Egw">
291         <di_1:waypoint x="319.0" y="485.0"/>
292         <di_1:waypoint x="411.0" y="485.0"/>
293       </di:BPMNEdge>
294       <di:BPMNEdge id="_OhDHAAncEeO5s8M9bY-Egw" bpmnElement="_OhAqwAncEeO5s8M9bY-Egw">
295         <di_1:waypoint x="453.0" y="486.0"/>
296         <di_1:waypoint x="543.0" y="486.0"/>
297       </di:BPMNEdge>
298       <di:BPMNEdge id="_VVG0wAncEeO5s8M9bY-Egw" bpmnElement="_VU4LQAncEeO5s8M9bY-Egw">
299         <di_1:waypoint x="431.0" y="465.0"/>
300         <di_1:waypoint x="431.0" y="386.0"/>
301         <di_1:waypoint x="543.0" y="386.0"/>
302       </di:BPMNEdge>
303     </di:BPMNPlane>
304   </di:BPMNDiagram>
305 </model:definitions>
```

Slike

2.1 **Osnovni elementi SOA [3].** 4
2.2 **Faze razvojnega cikla SOA [3].** 8

3.1 **Življenjski cikel poslovnega procesa [6].** 11
3.2 **BPM in SOA omogočata poslovno agilnost [2].** 12
3.3 **Relacija med SOA in BPM [3].** 13
3.4 **Vloge v razvojnem procesu SOA v navezi z BPM [3].** 14

4.1 **Kategorizacija BPMN 2.0 konstruktov s strani WfMC [18].** 23
4.2 **Tipi dogodkov v BPMN 2.0 [21].** 24
4.3 **Različni tipi aktivnosti v BPMN 2.0 [21].** 25
4.4 **Znaki za označevanje aktivnosti [21].** 26
4.5 **Različni tipi opravil v BPMN 2.0 [21].** 27
4.6 **Različni tipi prehodov v BPMN 2.0 [21].** 28
4.7 **Različni tipi povezav v BPMN 2.0 [12].** 29
4.8 **Predstavitev bazenov in stez v BPMN 2.0 [12].** 30
4.9 **Artefakti v BPMN 2.0 [12].** 31
4.10 **Model obiska veterinarske klinike z vidika visokonivojskega diagrama sodelovanja BPMN.** 33
4.11 **Prvi del modela obiska veterinarske klinike z vidika podrobnega diagrama sodelovanja BPMN.** 34
4.12 **Drugi del modela obiska veterinarske klinike z vidika podrobnega diagrama sodelovanja BPMN.** 35
4.13 **Obisk veterinarske klinike z vidika diagrama koreografije.** 37
4.14 **Obisk veterinarske klinike z vidika diagrama pogovora.** 38

5.1 **Osnovna sestava BPMS-ja [9].** 46
5.2 **Stroj BPMN 2.0 na katerem lahko bazira tako orodje BPEL kot orodje BPMN 2.0 [9].** 47

5.3 **Prvi del modela obiska veterinarske klinike z vidika diagrama sodelovanja BPMN 2.0.** 57
5.4 **Drugi del modela obiska veterinarske klinike z vidika diagrama sodelovanja BPMN 2.0.** 58
5.5 **Izvajalni del modela BPMN 2.0 za primer obiska veterinarske klinike.** 59
5.6 **Model BPMN 2.0, izdelan v Boniti, za proces fiktivnega sistema veterinarske klinike.** 60
5.7 **Prikaz modelirnega okolja v Boniti BPM.** 61
5.8 **Obrazec za vnos podatkov uporabniškega opravila *Vnesi podatke* v Boniti BPM.** 62
5.9 **Izgled Bonitinega portala za upravljanje s poslovnimi procesi.** 63

Tabele

5.1 **Primerjava orodij BPMS.** 50

Literatura

[1] M. B. Jurič. *Storitvena arhitektura - zgolj kompozicija spletnih storitev?* Dostopno na: www.soa.si/juric/soa_ss.pdf (april 2013).

[2] M. B. Jurič, P. Kapil. *Business Process Driven SOA using BPMN and BPEL*. Pact Publishing Ltd., 2008.

[3] S. Abeck. *Advanced Web Applications (prosojnice predavanj)*. C&M, Institute of Telematics, KIT, 2009.

[4] J. Jerele. *Kako obvladati procese*. MonitorPro, strani 24-26, pomlad 2011.

[5] M. Križevnik, M. B. Jurič. *Modeliranje in izvajanje poslovnih procesov v storitveno orientiranih arhitekturah*. Uporabna informatika, številka 3, strani 137-147, 2009.

[6] G. Polančič, G. Jošt. *Analiza upravljanja poslovnih procesov z BPMN 2.0*. Uporabna informatika, številka 3, strani 153-163, 2012.

[7] R. K. L. Ko. *A computer scientist's introductory guide to Business Process Management (BPM)*. Crossroads, Summer 2009/Vol.15, No.4, strani 11-18.

[8] A. Barkhordarian, F. Demuth, K. Hamann, M. Hoang, S. Weichler, S. Zaplata. *Migrability of BPMN 2.0 Process Instances (prosojnice predavanj)*. December 2011. Dostopno na: http://www.zirpins.de/WESOA11/Programme_files/C3-WESOA11-Migratability.pdf (maj 2013).

[9] F. Leymann. *BPEL vs BPMN 2.0: Should You Care?*. Oktober 2010. Dostopno na: http://bpt.hpi.uni-potsdam.de/pub/BPMN2010/Program/bpmn2010_leymann.pdf (maj 2013).

[10] L. Dugan, N. Palmer. *Making a BPMN 2.0 Model Executable*. BPMN 2.0 Handbook second edition, strani 71-91, 2012. Dostopno na: http://bpmnhandbook.com/04_papers/BPMN_Handbook_Free_Chapter_Making_BPMN_Model_Executable.pdf (maj 2013).

[11] B. Nguyen. *Business Process Execution Language (BPEL): Definition & How it creates composite WebServices*. Maj 2011. Dostopno na: http://trongbang86.blogspot.com/2011/05/business-process-execution-language.html (maj 2013).

[12] OMG. *Business Process Model and Notation (BPMN)*, version 2.0, januar 2011. Dostopno na: http://www.omg.org/spec/BPMN/2.0/PDF/ (maj 2013).

[13] G. Smink. *BPEL vs BPMN 2.0: which do I choose?*. Maj 2012. Dostopno na: http://www.jointherebels.nl/blog/bpm/item/153-choice-between-bpel-and-bpmn20.html (maj 2013).

[14] Activiti™. *Frequently Asked Questions about Activiti*. Dostopno na: http://www.activiti.org/faq.html (junij 2013).

[15] Wikipedia. *jBPM*. Dostopno na: http://en.wikipedia.org/wiki/JBPM (avgust 2013).

[16] N. McWhorter. *Taking BPMN to Automation: Bumps in the Road – Part 1: Linking Levels*. Dostopno na: http://strategicvaluepartners.com/archives/42 (junij 2013).

[17] N. McWhorter. *Taking BPMN to Automation: Bumps in the Road – Part 2: Flow Control and Business Rules*. Dostopno na: http://strategicvaluepartners.com/archives/231 (junij 2013).

[18] T. Rademakers, R. van Liempd. *Activiti in action*. MEAP Edition. Dostopno na: http://www.mif.vu.lt/~donatas/PSArchitekturaProjektavimas/Library/BP/2010%20BPMN%202.0%20whats%20in%20it%20for%20developers%20(Activiti).pdf (junij 2013).

[19] BonitaSoft. Dostopno na: http://www.bonitasoft.com (junij 2013).

[20] B. Silver. *Executable BPMN 2.0*. November 2011. Dostopno na: http://brsilver.com/executable-bpmn-2-0 (junij 2013).

[21] M. Weske. *Business Process Management. Concepts, Languages, Architectures*. 2. Izdaja, 2012.

[22] F. Marchioni. *BPMN tutorial for beginners*. Dostopno na: http://www.mastertheboss.com/bpmn-20/bpmn-tutorial-for-beginners (junij 2013).

[23] Q. J. Sarafinchan. *BPMN 2.0 Activity : Types of Tasks*. April 2013. Dostopno na: http://www.splatfx.com/bpmn-2-0-activity-types-of-tasks (junij 2013).

[24] JBoss Community. *jBPM*. Dostopno na: http://www.jboss.org/jbpm (avgust 2013).