Univerza *v Ljubljani*
Fakulteta za računalništvo in informatiko

ANA ŠAŠA

MODEL ZA AVTOMATIZACIJO POSLOVNIH PROCESOV V STORITVENO USMERJENIH SISTEMIH S TEHNOLOGIJAMI OBVLADOVANJA ZNANJA

DOKTORSKA DISERTACIJA

MENTOR: PROF. DR. MARJAN KRISPER

Ljubljana, 2009

Povzetek

Zaradi vse večjih zahtev po učinkovitosti, gospodarnosti in fleksibilnosti poslovnih sistemov je avtomatizacija poslovnih procesov postala pomembno področje, s katerim se ukvarjajo številni raziskovalci. Pristop k avtomatizaciji poslovnih procesov, ki se je v zadnjih letih najbolj uveljavil, je pristop po načelih storitveno usmerjene arhitekture. V storitveno usmerjeni arhitekturi je poslovni proces sestavljen iz storitev, ki predstavljajo različna opravila, ki morajo biti izvedena v poslovnem sistemu. Poslovni proces je navadno implementiran z enim izmed jezikov za izvajanje poslovnih procesov, med katerimi je danes najbolj uveljavljen WS-BPEL (*Web Services Business Process Execution Language*). Sistemi za izvajanje poslovnih procesov izvajajo orkestracijo različnih storitev, ki lahko predstavljajo avtomatizirana opravila ali omogočajo integracijo opravil, ki jih izvajajo ljudje (uporabniška opravila). Integracijo uporabniških opravil v izvajanje poslovnih procesov podajata specifikaciji BPEL4People in WS-HumanTask.

Tema pričujoče doktorske disertacije je doseganje višje ravni avtomatizacije poslovnih procesov v storitveno usmerjenih sistemih. V poslovnih procesih se osredotočamo na uporabniška opravila, kot tisti del opravil, ki niso avtomatizirana. Analizirali smo uporabniška opravila in možnost ter primernost avtomatizacije posameznih vrst ter posameznih sestavnih delov uporabniških opravil v okviru storitve za izvajanje uporabniških opravil. Podajamo arhitekturni model, podprt z metodami, ki razširjajo možnosti njihove avtomatizacije. Vsaka arhitekturna komponenta modela je odgovorna za avtomatizacijo določenega dela uporabniškega opravila. V primerih, ko avtomatizacija s podanim modelom ni možna, je opravilo posredovano v izvedbo ustrezni osebi. Za uporabo modela disertacija opredeli tudi razširitev specifikacij WS-HumanTask. Razširitev omogoča ohranjanje povratne skladnosti z obstoječimi specifikacijami.

Model temelji na zapažanju, da lahko izvajanje poslovnih procesov in uporabniških opravil uporabimo za zajem znanja, na katerem izvajanje temelji, in za zajem znanja, ki med njihovim izvajanjem nastaja. Pri tem je cilj omogočanje neposredne ponovne uporabe znanja pri izvajanju uporabniških opravil in s tem avtomatizacije uporabniških opravil. V ta namen je podana metoda odločanja, ki temelji na večkriterijskem odločanju in odločitveni model gradi v OWL ontologiji.

Model omogoča njeno uporabo v izvajanju uporabniških opravil kot dela izvajanja poslovnih procesov, s čimer smo dosegli višjo raven avtomatizacije uporabniških opravil, ki temeljijo na odločitvah.

Model dosega tudi višjo raven avtomatizacije uporabniških opravil, ki zahtevajo sodelovanje med različnimi udeleženci. Podan je inovativen pristop k opredelitvi protokolov sodelovanja, ki so osnova za sodelovanje med agenti večagentnih sistemov, ki predstavljajo udeležence. Pristop omogoča ločitev protokola in implementacije ter vzpostavitev šibke sklopljenosti protokola in implementacije. To se odraža v tem, da spremembe protokolov ne zahtevajo spremembe implementacije obnašanja agentov, ki protokole uporabljajo za svoje delovanje.

Ker sta v poslovnih sistemih odločanje in sodelovanje med najpomembnejšimi aktivnosti zaposlencev, model izboljšuje pomemben delež uporabniških opravil. Predstavlja pomemben korak naprej k viziji celovite avtomatske podpore poslovnim procesom in podaja celovito rešitev za dvig avtomatizacije uporabniških opravil. Ta pripomore k učinkovitejšemu izvajanju poslovnih procesov in višji sledljivosti odločitev v poslovnem procesu.

S tem pričujoča doktorska disertacija celostno zaokrožuje segment izboljšanja učinkovitosti uporabniških opravil v okviru izvajanja poslovnih procesov v SOA in podaja sklop novih, inovativnih rešitev, ki temeljijo na ontologijah in večagentnih sistemih, katerih skupni cilj je avtomatizacija uporabniških opravil in s tem doseganje višje ravni avtomatizacije poslovnih procesov.

Ključne besede: avtomatizacija poslovnih procesov, storitveno usmerjena arhitektura, večkriterijsko odločanje, ontologije, večagentni sistem.

Abstract

Due to increasing requirements for efficiency, effectiveness and flexibility of business systems, automation of business processes has become an important topic. In the last years, the most successful and predominant approach to business process automation has become the service-oriented architecture approach. In a service-oriented architecture a business process is composed of services, which represent different tasks that have to be performed in a business system. Typically, a business process is implemented with one of the business process execution languages. Among them the prevalent language has become WS-BPEL (*Web Services Business Process Execution Language*). Business process execution systems perform orchestration of different services. Services can represent automated tasks or enable integration of human-performed tasks (human tasks). Integration of human tasks into business process execution is covered by a pair of specifications: BPEL4People and WS-HumanTask.

The topic of this doctoral dissertation is to enable a higher degree of automation of business processes in service-oriented systems. We focus on human tasks, because they represent those tasks of a business process, which are not automated. We define an architectural model supported by innovative methods, which extend the possibilities of automation of human tasks. Every architectural component is responsible for automation of a certain part of a human task. In cases where automation is not enabled by the proposed model, the human task is delegated to the responsible person. For implementation of the model the dissertation also defines an extension to the WS-HumanTask specifications. The extension allows backward compatibility with the existing specifications.

The model is based on observation that business process and human task execution can be seen as an opportunity to capture the knowledge based on which the execution is performed, and to capture the knowledge which is created during the execution. The goal is to enable direct reuse of the knowledge and possible automation for succeeding human task executions. In order to enable this, we define a decision making method, which is based on multi-criteria decision making and develops a decision model in an OWL ontology. The model enables the use of this method in

human task execution as part of business process execution. With this it enables a higher level of automation of human tasks, which are based on decisions.

The model also enables a higher level of automation of human tasks, which require collaboration between different human actors. The dissertation defines an innovative approach to defining business collaboration protocols, which are the basis for collaboration between agents in a multi-agent system, which represent the human actors. The approach enables separation of protocol definition and implementation and, with this, loose coupling of protocols and implementation. This means that changes in a collaboration protocol do not require changes in the implementation of the multi-agent system or the implementation of the behaviour of the agents, which use the protocols in their acts.

As decision making and collaboration are among the most important activities of employees in business systems, the model improves a significant part of human tasks. It represents an important step closer to the vision of end-to-end business process automation. It heightens the degree of automation of human tasks and consequently enables more efficient business process execution. Another important advantage is a higher level of decision traceability.

By this, the doctoral dissertation represents an integral solution to improving efficiency of human tasks within the business process context in service-oriented systems and defines a set of new and innovative methods, which are based on ontologies and multi-agent systems. Their common goal is automation of human tasks and, by this, enabling a higher degree of automation of business processes.

Index Terms: business process automation, service-oriented architecture, multi-criteria decision making, ontologies, multiagent system.

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

doktorske disertacije

Spodaj podpisana Ana Šaša

z vpisno številko 63000280,

sem avtorica doktorske disertacije z naslovom

MODEL ZA AVTOMATIZACIJO POSLOVNIH PROCESOV V STORITVENO USMERJENIH SISTEMIH S TEHNOLOGIJAMI OBVLADOVANJA ZNANJA.

S svojim podpisom zagotavljam, da:

- sem doktorsko disertacijo izdelal/-a samostojno pod vodstvom mentorja (naziv, ime in priimek)

  Prof. Dr. Marjana Krisperja

  in somentorstvom (naziv, ime in priimek)

  _____

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

- in soglašam z javno objavo elektronske oblike doktorske disertacije v zbirki »Dela FRI«.

V Ljubljani, dne 19.6.2009 Podpis avtorja/-ice: _____

Zahvala

Najprej se zahvaljujem mentorju prof. dr. Marjanu Krisperju za vodenje in usmerjanje v času podiplomskega študija in za koristne nasvete pri pisanju doktorske disertacije. Za pomoč s strokovnimi nasveti se zahvaljujem tudi vsem članom Laboratorija za informatiko Fakultete za računalništvo in informatiko Univerze v Ljubljani, kjer je bilo to delo opravljeno.

Iskrena hvala moji mami za vso njeno pomoč in da mi je vedno stala ob strani ter me spodbujala. Hvala mojemu bratu za podporo in spodbudo.

Še posebej se zahvaljujem Matjažu, za njegovo ljubezen, podporo in razumevanje.

Kazalo

Kazalo slik

Kazalo tabel

1 Uvod

Aplikativni in informacijski sistemi so postali ključni temelj delovanja poslovnih sistemov [40]. Posamezna opravila poslovnih procesov pogosto izvajajo zaposlenci v poslovnem sistemu. Ti lahko pri izvajanju opravil uporabljajo aplikativne sisteme. Aplikativni sistemi lahko opravila podpirajo tudi v celoti in vključenost zaposlencev v opravilo ni potrebna. Z vsakim novim aplikativnim sistemom se informacijski sistemi širi in podpira večji del poslovanja [41]. Poslovni sistemi postajajo tako vse bolj odvisni od svojih informacijskih sistemov, poslovni procesi pa vse bolj avtomatizirani.

Pri soočanju z globalno konkurenco morajo poslovni sistemi na eni strani skrbeti za čim nižje stroške poslovanja in na drugi strani za hiter ter proaktiven razvoj novih izdelkov in storitev [30]. Poleg tega poslovni sistemi postajajo vse bolj kompleksni in velik del poslovnih procesov presega meje poslovnega sistema [62]. Tradicionalni pristopi k razvoju informacijskih sistemov in k izdelavi poslovno-informacijskih arhitektur tem zahtevam niso več bili sposobni zadoščati. Kot odgovor je bila predlagana paradigma storitveno usmerjene arhitekture, ki je danes postala prevladujoč pristop k razvoju in obvladovanju informacijskih sistemov. Usmerjena je v poslovne procese in v način, kako naj bi jim informacijski sistemi nudili ustrezno podporo, čim višjo raven avtomatizacije ter omogočali hitro in fleksibilno prilagajanje novim poslovnim zahtevam.

Tema pričujoče doktorske disertacije je doseganje višje ravni avtomatizacije poslovnih procesov v storitveno usmerjenih sistemih. V poslovnih procesih se osredotočamo na opravila, ki jih izvajajo ljudje (uporabniška opravila), kot tisti del opravil, ki niso avtomatizirana. V drugem poglavju opišemo osnovne koncepte in področja. Natančneje predstavimo izvajanje poslovnih procesov v storitveno usmerjenih sistemih. Razčlenimo podporna področja, s pomočjo katerih bomo predlagali rešitev modela za doseganje višje ravni avtomatizacije poslovnih procesov. S tem definiramo izhodišča, na katerih temelji raziskovalno delo.

V tretjem poglavju definiramo področje raziskovalnega dela. Opredelimo problem, opišemo zadano nalogo in postavimo hipoteze raziskovalnega dela. Na osnovi pomena in pregleda

sorodnih raziskav ugotovimo, da je področje doseganja višje ravni avtomatizacije poslovnih procesov s tehnologijami obvladovanja znanja slabo raziskano.

V četrtem poglavju predstavimo uporabniška opravila in njihove značilnosti. Analiziramo različne vrste uporabniških opravil in določimo tiste vrste, ki so smiselne za nadaljnjo avtomatizacijo v okviru izvajanja uporabniških opravil.

Na podlagi izhodišč raziskovalnega dela, zadanih nalog in analize avtomatizacije uporabniških opravil v petem poglavju podamo zasnovo modela za doseganje višje ravni avtomatizacije poslovnih procesov. Opredelimo posamezne arhitekturne komponente modela in njihovo funkcionalnost. V poglavjih šest, sedem in osem sledi natančen opis metod, ki podpirajo zasnovani model pri uresničevanju zadanih ciljev.

V devetem poglavju podajamo zaključke in strnemo teoretične in praktične prispevke te doktorske disertacije. V prilogah podajamo natančnejši formalni opis določenih delov modela in podrobnejše zapise primerov.

2 Osnovna raziskovalna področja

2.1 Ključna področja

2.1.1 Poslovni procesi

Poslovni procesi in avtomatizacija poslovnih procesov

Poslovni proces je množica logično povezanih opravil, ki so lahko avtomatizirana (avtomatizirana opravila) ali jih izvede določena oseba oziroma skupina oseb (uporabniška opravila) in imajo za poslovni sistem določeno poslovno vrednost [41]. Poslovne procese lahko delimo na ključne poslovne procese in na podporne poslovne procese. Ključni poslovni procesi so tisti, ki s svojimi rezultati neposredno ustvarjajo poslovno vrednost. Navadno prečkajo meje poslovnega sistema - začnejo in končajo se zunaj poslovnega sistema, na primer pri stranki ali poslovnemu partnerju. Podporni poslovni procesi so tisti, ki k ustvarjanju poslovne vrednosti ne prispevajo neposredno, temveč tako da omogočajo izvajanje ključnih poslovnih procesov ali prispevajo k učinkovitosti izvajanja ključnih poslovnih procesov [7].

Vrstni red opravil v poslovnem procesu in učinkovitost izvajanja teh opravil sta tesno povezana z učinkovitostjo samega poslovnega sistema. Vedno bolj postaja pomembno, da so poslovni procesi natančno opredeljeni. Formalna opredelitev poslovnih procesov je osnova za doseganje učinkovitosti [41]:

- predstavlja podlago za odkrivanje ozkih grl in optimizacijo izvajanja poslovnih procesov, na primer odpravljanje podvajanj opravil, ki se jih brez formalizacije procesa morda sploh ne bi zavedali,
- predstavlja osnovo za avtomatizacijo poslovnih procesov.

Poslovni sistemi postajajo vse bolj odvisni od svojih informacijskih sistemov. Da so lahko poslovni sistemi uspešni, naj bi njihovi informacijski sistemi omogočali čim boljšo podporo avtomatizaciji poslovanja. Avtomatizacija poslovnih procesov je koncept, ki je v zadnjem

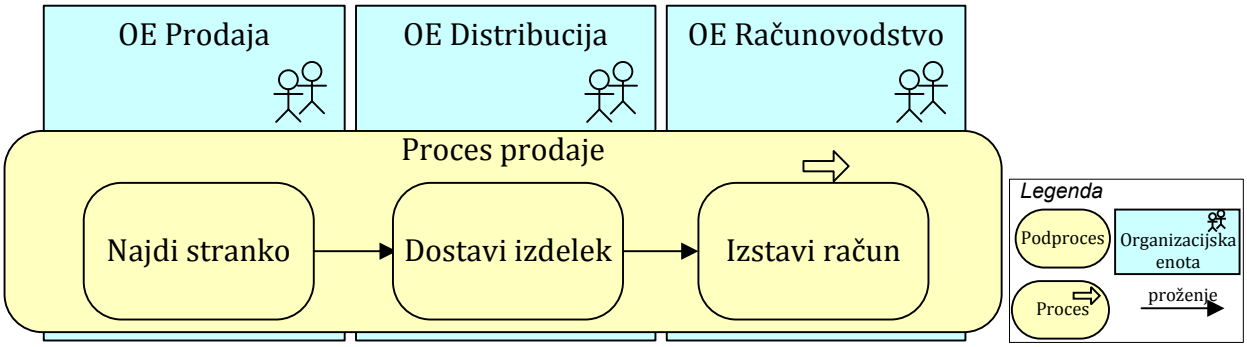
desetletju postal zelo popularen. Pomeni pomembne izboljšave v poslovnih sistemih in je zato danes eno izmed bolj atraktivnih znanstvenih področij informatike, s katerim se ukvarja vrsta raziskovalcev, na primer [6],[15],[17],[18],[21],[37],[38],[40],[43],[45],[48],[50],[60],[65], [68],[72],[80],[101],[102],[109],[111], [112]. Pri tem ne gre zgolj za avtomatizacijo posameznih opravil poslovnega procesa, temveč tudi za avtomatizacijo orkestracije opravil v skladu s formalnimi opredelitvami poslovnih procesov. Sisteme, ki takšno orkestracijo omogočajo, imenujemo sistemi za izvajanje poslovnih procesov. Sistemi za izvajanje poslovnih procesov poleg same avtomatizacije nudijo tudi pomembne informacije o izvajanju poslovnih procesov ter posameznih opravil v procesu in omogočajo nadzor nad potekom procesa, merjenje učinkovitosti ter optimizacijo poslovnih procesov.

Poslovna in funkcionalna usmerjenost poslovnega sistema

Poslovno dogajanje lahko opišemo z množico med seboj bolj ali manj povezanih poslovnih procesov (procesni vidik). V nasprotju s tem tradicionalni pogled na poslovno dogajanje ni procesno usmerjen, temveč je usmerjen funkcionalno (funkcionalni vidik). To pomeni, da poslovno dogajanje opisuje glede na funkcionalna področja, ki so prisotna v poslovnem sistemu. Tipično so poslovni sistemi organizirani v različne organizacijske enote (oddelke, službe ipd.) glede na funkcionalna področja. Vsaka organizacijska enota opravlja določeno funkcijo, ki določa specifična znanja in kompetence in s tem pogojuje tudi strokovno usposobljenost svojih zaposlencev. Pogosto je v tovrstnih poslovnih sistemih prisotna kultura t.im. *funkcionalnih otokov* (*»silosov«*), kjer posamezne organizacijske enote delujejo samostojno z malo ali nič interakcije z drugimi organizacijskimi enotami v istem poslovnem sistemu [53].

Če aktivnosti v poslovnem procesu zahtevajo različna strokovna znanja, proces lahko vključuje zaposlene v različnih organizacijskih enotah (*horizontalen poslovni proces*). Primer je poslovni proces prodaje stranki, ki vključuje več organizacijskih enot: OE Prodaja, OE Distribucija in OE Finance. Na visokem nivoju abstrakcije ga ponazarja Slika 1. Iz vidika funkcionalnih področij se nadzor in odgovornosti skozi proces spreminjajo glede na kompetence posameznih organizacijskih enot. Pri tem obstaja tveganje podvajanja opravil in različnega zaznavanja kvalitete, ki lahko pomeni tudi tveganje zmanjšanja ali izgube nadzora nad kvaliteto. To tveganje je lahko še bolj izrazito ob rigidni hierarhični organizacijski strukturi, ki pogosto predstavlja oviro učinkoviti komunikaciji med različnimi organizacijskimi enotami. Zaradi ovir v

komunikaciji med različnimi organizacijskimi enotami lahko prihaja do zamud, predvsem v točkah, kjer proces prestopa meje organizacijskih enot.



Slika 1: Primer procesa, ki prečka več organizacijskih enot

V nasprotju s funkcionalno usmerjenim pogledom, ki je osredotočen na »kaj kdo stori«, je procesni vidik osredotočen na to »kaj je potrebno narediti«. Ko proces teče skozi različne funkcionalne otoke, je poudarek na tem, da proces teče učinkovito, gospodarno in brez večjih ovir. To pomeni, da je potrebna učinkovita koordinacija in komunikacija med akterji v poslovnem procesu zato, da proces doseže zahtevane cilje. V današnjih poslovnih sistemih ima informacijski sistem, poleg same avtomatizacije poslovnih procesov, nalogo podpore in integracije množice poslovnih procesov z združevanjem funkcionalnih otokov in z omogočanjem čim višje ravni dosegljivosti podatkov v celotnem poslovnem sistemu v realnem času [75].

Sistemi za obvladovanje delovnih tokov

Pojem avtomatizacije poslovnih procesov je tesno povezan s sistemi za obvladovanje delovnih tokov (*Workflow management systems*, WfMS). WfMS so načrtovani z namenom omogočanja bolj učinkovitega dela, integracije heterogenih aplikativnih sistemov in podpore medorganizacijskim procesom v aplikacijah e-trženja [72]. Različni avtorji pojmujejo pojem »delovni tok« zelo različno, na primer kot sinonim za poslovni proces [12], kot specifikacijo procesa, kot programsko opremo, ki implementira in avtomatizira proces, ali kot programsko opremo, ki podpira koordinacijo in sodelovanje med ljudmi, ki izvajajo proces [30]. Kljub temu je tradicionalno pojem »delovni tok« usmerjen v pisarniška opravila [61] in večina WfMS sistemov se ukvarja z avtomatizacijo tistih delov poslovnega procesa, v katerih se dokumenti in informacije ustvarjajo, spreminjajo in brišejo, ter tistih, v katerih se dokumenti, podatki in

opravila prenašajo med različnimi akterji v poslovnem procesu [13],[61],[72]. WfMS sistem torej skrbi za avtomatizacijo poslovnega procesa v celoti ali deloma, pri čemer se dokumenti, informacije ali opravila prenašajo med različnimi akterji v procesu glede na množico pravil. Množica opravil, ki jih mora izvesti uporabnik sistema WfMS, se imenuje delovni seznam. Uporabniku se prikaže v uporabniški aplikaciji.

V nasprotju s tem pa so sistemi za izvajanje poslovnih procesov usmerjeni predvsem v orkestracijo avtomatiziranih (strojno, aplikativno itd.) opravil in v reševanje integracijskih vprašanj. Zaradi potrebe po celoviti podpori poslovnim procesom danes meje med sistemi za obvladovanje delovnih tokov in sistemi za izvajanje poslovnih procesov vedno bolj izginjajo. To pomeni, da je *sistem za izvajanje poslovnih procesov informacijski sistem, ki je zadolžen za prehajanje med opravili poslovnega procesa, pri čemer v okviru posameznega opravila proži ustrezne aplikativne ali strojne sisteme oziroma dodeljujejo opravila uporabnikom*. V ta namen morajo izvajati številne aktivnosti koordiniranja in nadzora, kot so generiranje instanc poslovnega procesa, dodeljevanje aktivnosti "človeškim ali strojnim agentom", ki izvajajo posamezne aktivnosti, generiranje delovnih seznamov, usmerjanje opravil in z njimi povezanih delovnih objektov med agenti, pošiljanje opomnikov človeškim agentom itd. [18]. Uporabniška opravila lahko pomenijo povsem različne vrste dela, od na primer priklapljanja naprave v električni sistem do odločitve o neki zadevi.

2.1.2 Storitveno-usmerjena arhitektura

Kaj je storitveno usmerjena arhitektura?

Kompleksni informacijski sistemi sestavljeni iz množice tehnološko heterogenih aplikativnih sistemov postajajo skozi čas vse manj fleksibilni. Na drugi strani narašča potreba po celoviti podpori poslovnim procesov, ki omogoča hitro prilagajanje ob spreminjajočih se poslovnih zahtevah. To je vodilo k iskanju novih rešitev pri izgradnji informacijskih sistemov. Izkazalo se je, da informacijski sistemi, ki nastanejo kot rezultat tradicionalnih pristopov k razvoju, skozi čas postanejo zelo zahtevni in dragi za vzdrževanje in praviloma ne zmorejo zadoščati zahtevanim potrebam po agilnosti in učinkovitosti. Poslovni sistemi ne zmorejo slediti poslovnim zahtevam, saj se informacijski sistemi niso sposobni prilagoditi tako hitro, kot bi bilo potrebno, in s tem omejujejo zmožnost poslovnega sistema pri odzivanju na tržne zahteve [45]. Poleg tega

posamezni aplikativni sistemi največkrat nudijo podporo posameznim funkcionalnim otokom, medtem ko je podpora horizontalnim poslovnim procesom realizirana s pomočjo integracije na podatkovni ali aplikativni ravni. Semantična razhajanja med poslovnimi uporabniki, ki so bodisi lastniki poslovnih procesov ali akterji v poslovnih procesih, in informatiki, ki skrbijo za implementacijo poslovnih procesov, so velika, zaradi česar je usklajenost poslovne in aplikativne domene manjša, informacijski sistem pa navadno ne omogoča nadzora in optimizacije izvajanja celovitih poslovnih procesov.

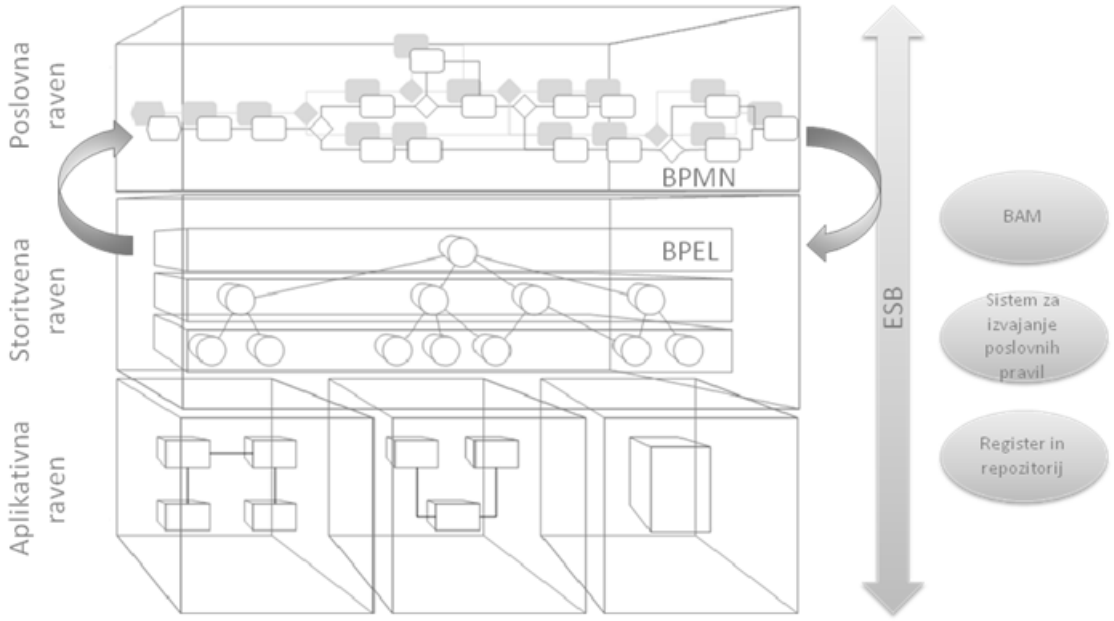
Storitveno usmerjena arhitektura (*service-oriented architecture*, SOA) je sprejeta paradigma razvoja informacijskih sistemov, ki izhaja iz motivacije za odpravo navedenih težav s tradicionalnimi informacijskimi sistemi. Storitveno usmerjena arhitektura je osredotočena na poslovne procese poslovnega sistema in na način, kako bi naj informacijska tehnologija te procese podprla [41]. Pri tem je glavna motivacija povišanje zmožnosti poslovnega sistema za vključevanje morebitnih novih poslovnih zahtev v čim krajšem času, predvsem s ponovno uporabo obstoječe poslovne logike in podatkovnih modelov. Na ta način so stroški, čas in drugi viri za uvedbo novih zahtev minimalni ter zmanjšana tveganja, predvsem v primerjavi s ponovnim razvojem celotnih aplikativnih sistemov [45]. SOA torej ni usmerjena zgolj na tehnološko plat temveč v poslovanje: v poslovno vsebino, poslovne procese, usklajevanje informacijskega sistema s poslovnim sistemom in na optimizacijo poslovnih procesov.

Ključni pojmi storitveno usmerjene arhitekture

Storitveno usmerjena arhitektura med poslovno in aplikativno raven poslovnega sistema uvaja **storitveno raven** (Slika 2). Storitvena raven je sestavljena iz storitev, logičnih enot, ki bi naj ustrezale načelom storitvene usmerjenosti [20]:

- **Šibka sklopljenost** - storitve ohranjajo odnose, ki minimizirajo odvisnosti med njimi in ohranjajo le zavedanje ena druge.
- **Storitvena pogodba** – storitve se držijo komunikacijskega dogovora, ki je določen z enim ali več opisov storitev in podobnih dokumentov.
- **Neodvisnost** – storitev je neodvisna od drugih storitev, in sicer v smislu nadzora nad svojo logiko.

- **Abstrakcija** – z izjemo opisa storitve je logika storitve nedostopna zunanjemu svetu.
- **Ponovna uporaba** – logika je razdeljena, ločena oziroma razbita v različne storitve z namenom možnosti ponovne uporabe.
- Storitve minimizirajo količino informacij, ki pripada določeni aktivnosti – so **brez stanja**.
- **Odkrivanje** – storitve so načrtovane tako, da jih lahko opišemo in najdemo; do njih dostopamo preko temu namenjenih mehanizmov.



Slika 2: Ključni koncepti SOA

SOA temelji na ohlapno povezanih sistemih, združenih v celoto, pri čemer so posamezni deli med seboj neodvisni in lahko tečejo na poljubnih platformah [40]. Storitev je abstraktni vir, ki predstavlja zmožnost izvedbe opravila, ki iz vidika uporabnikov in ponudnikov storitve sestavlja koherentno funkcionalnost. Da je lahko uporabljena, jo mora realizirati konkreten ponudnik [97]. Storitvi lahko pravimo tudi storitveno usmerjena logična enota. Logika, ki jo storitev vsebuje, je namenjena za reševanje določenega problema, katerega kompleksnost je lahko zelo različna [20]. Na primer, gledano iz vidika poslovnega procesa, lahko predstavlja storitev posamezno procesno

opravilo, sklop opravil ali celoten proces. Poleg tega lahko izvajanje ene storitve vključuje izvajanje ene ali več drugih storitev, se pravi, da lahko storitve med seboj sestavljamo.

Ena izmed pomembnejših pridobitev storitvene ravni je, da sestavljivost omogoča nadaljnje strukturiranje storitev v različne ravni. Priporočljivo je, da so nižje storitvene ravni sestavljene iz tehničnih storitev in aplikativnih storitev, ter da so višje storitvene ravni sestavljene iz poslovnih in procesnih storitev. Raven procesnih storitev je namenjena orkestraciji storitev nižjih ravni, tako da se njihove operacije izvedejo v določenem zaporedju. S tem skrbi za zagotavljanje poslovnih pravil in kompozicijske logike. Tako storitveno usmerjena arhitektura skozi procesno raven omogoča tudi podporo avtomatizaciji poslovnih procesov [80].

Pomemben pojem je opis storitve. Opis storitve natančno opredeljuje storitveno pogodbo in omogoča odjemalcu storitve, da storitev uporabi, ne da bi se zavedal same implementacije [74]. Da lahko posamezno storitev uporablja neka druga storitev (ali program) mora imeti svoj opis, ki bo drugi storitvi nudil podatke, kot so operacije, ki jih lahko izvaja, kje se nahaja, kaj so njeni vhodni parametri in kaj lahko pričakuje na izhodu.

Da lahko storitve med seboj komunicirajo, morajo znati med seboj izmenjevati informacije. Za to potrebujejo komunikacijsko ogrodje, ki bo ohranjalo šibko sklopljenost med storitvami. Temu ustreza pošiljanje sporočil, ki so prav tako kot storitve neodvisna. Ko storitev sporočilo pošlje, izgubi nadzor nad sporočilom. Pravimo, da so sporočila neodvisne enote komunikacije.

Tehnologije SOA

Storitvena usmerjenost in storitveno usmerjena arhitektura sta obstajali že pred nastankom spletnih storitev, vendar se je izkazalo, da so spletne storitve zelo primerne in uspešne pri izdelavi rešitev SOA. Vse platforme večjih proizvajalcev podpirajo razvoj storitveno usmerjenih rešitev in večina jih pri tem temelji na spletnih storitvah.

Ena izmed največjih značilnosti spletnih storitev je, da izmenjava podatkov temelji na odprtih standardih. Sporočilo, ki ga ena spletna storitev pošlje drugi, »potuje« preko množice protokolov, ki so globalno standardizirani in sprejeti. Tako je interoperabilno sodelovanje med storitvami (neodvisno od programskega jezika ali platforme) izboljšano s skladnostjo s standardi, kot na primer XML (*Extensible Markup Language*) [88], XML Schema [94],[98], SOAP (*Simple Object*

Access Protocol) [92],[93], UDDI (*Universal Description, Discovery and Integration*) [55],[56], WSDL (*Web Services Description Language*) [95],[96].

Opis storitve se nahaja v datoteki WSDL. Datoteke WSDL lahko hranimo skupaj z nekaterimi drugimi podatki tudi v katerem od registrov storitev UDDI, preko katerih lahko uporabniki poiščejo objavljeno storitev.

Orkestracija spletnih storitev zahteva model, s katerim so poslovni procesi lahko opisani v smislu povezovanja spletnih storitev. V preteklem desetletju je bilo v ta namen razvitih več jezikov. Najbolj znani med njimi so XLANG, WSFL, YAWL in WS-BPEL (*Web Services Business Process Execution Language;* v nadaljevanju BPEL) [8],[57]. BPEL združuje načela XLANG in WSFL in je danes najbolj sprejet in podprt s številnimi orodji. Gre za jezik, ki je bil primarno izdelan za podporo avtomatizacije poslovnih procesov, ki temeljijo na spletnih storitvah. Odlikuje ga razširljivost procesov, prožnost uporabe in enostavnost ter berljivost zapisa. Ker je vsak BPEL proces tudi sam spletna storitev, potrebuje tudi svojo datoteko WSDL. To pomeni, da ga je moč poklicati kot spletno storitev in da lahko posamezne BPEL procese sestavljamo med seboj.

Za modeliranje poslovnih procesov je na voljo več različnih notacij. Priporočena notacija za pristop SOA je BPMN (*Business Process Modeling Notation*) [59]. Vsebuje tako grafične konstrukte za izdelavo diagramov procesov, kot široko množico atributov, ki služijo za natančnejši opis procesa. Ključna prednost jezika BPMN je preslikava v BPEL kodo procesa, kar poenostavi razumevanje in sinhronizacijo med poslovnim vidikom ter tehničnim vidikom poslovnega procesa.

Kljub skladnosti s standardi, lahko še vedno prihaja do različnih odstopanj in do zmanjšanja interoperabilnosti, na primer zaradi nedoslednosti ali posebnosti uporabljene platforme. Organizacija *Web Services Interoperability Organization* je zato podala smernice za doseganje interoperabilnosti v različnih specifikacijah, kot so *WS-I Basic Profile* [106], *Simple Soap Binding Profile* [108] in *Basic Security Profile* [107]. Ti standardi podpirajo neodvisnost sporočil, ki si jih spletne storitve pošiljajo med seboj, in šibko sklopljenost storitev, ki razen opisov storitev ne potrebujejo drugih podatkov, da lahko komunicirajo.

Z namenom uvedbe standardizirane podpore za uporabniška opravila z jezikom BPEL, je bil v organizacijo OASIS predlagan par specifikacij:

- BPEL4People (*WS-BPEL Extension for People*) [4], ki razširjajo jezik BPEL, in
- WS-HumanTask (*Web Services Human Task*) [5], ki določajo vidike implementacije storitev uporabniških opravil.

Predlagane specifikacije predstavljajo celovito rešitev integracije uporabniških opravil v poslovni proces, in sicer tako iz vidika poslovnega procesa, kot z vidika implementacije storitve za izvajanje uporabniških opravil. Obravnavajo na primer različne vidike uporabniških interakcij s procesom, kot so vloge, vzorci interakcij, različni načini integracije uporabniških opravil v proces in operacije odjemalskih aplikacij.

SOA in poslovni procesi

V SOA je poslovni proces sestavljen iz storitev. Z vidika poslovnega procesa storitve predstavljajo različna opravila, ki morajo biti v poslovnem sistemu izvedena [38]. SOA pristop k razvoju informacijskih sistemov temelji na iterativnem pristopu implementacije in optimizacije poslovnih procesov, ki upošteva spreminjanje poslovnih procesov in omogoča fleksibilno vključevanje sprememb v informacijski sistem, ki jih podpira. Pomembna pridobitev SOA so nove tehnologije in jeziki, ki zmanjšujejo semantična razhajanja med poslovnimi procesi in aplikativnimi sistemi [41]:

- Ključno vlogo pri zmanjševanju razhajanj med poslovnimi procesi in aplikativnimi sistemi igrata specifikaciji BPMN in BPEL. Notacija BPMN omogoča modeliranje poslovnih procesov na poslovni ravni in preslikavo v BPEL kodo. Na ta način pridobljeno BPEL kodo informatik dopolni s tehničnimi podrobnostmi in pripravi proces za izvajanje.
- Sistemi za izvajanje poslovnih procesov omogočajo vpogled v izvajanje poslovnih procesov in nadzorovanje poslovnih procesov (*Business Activity Monitoring, BAM*). Nadzorni podatki poslovnim uporabnikom pomenijo dragocene informacije in so med drugim osnova tudi za optimizacijo poslovnih procesov.
- Če se pokaže potreba po spremembi poslovnega procesa, na primer optimizaciji, odgovorni poslovni uporabniki ustrezne spremembe vnesejo v model poslovnega procesa (v notaciji BPMN).

Gre torej za sklenjen krog od modeliranja poslovnih procesov (poslovni uporabniki), preko njihove avtomatizacije (informatiki), izvajanja (sistem za izvajanje poslovnih procesov, akterji v poslovnem procesu, informatiki) in spremljanja (poslovni uporabniki, navadno odgovorni nosilci poslovnih procesov) do optimizacije (poslovni uporabniki, navadno odgovorni nosilci poslovnih procesov), ki spet vodi v modeliranje.

2.2 Podporna področja

2.2.1 Obvladovanje znanja in ontologije

Obvladovanje znanja (*Knowledge Management*) omogoča zajemanje, shranjevanje in širjenje znanja z uporabo informacijske tehnologije [46]. Z naraščajočo kompleksnostjo produktov, z globalizacijo, virtualnimi organizacijami in usmerjenostjo k stranki je postalo pomemben dejavnik uspeha v poslovnih sistemih, in sicer tako znotraj posameznih poslovnih sistemov, kot v povezavi z njihovim okoljem in drugimi poslovnimi sistemi [77]. Če poslovni sistem nima ustreznega sistema za obvladovanje z znanjem, se lahko znanje kljub temu, da je morda že osvojeno, ne prenese na vse zaposlence, ki bi jim koristilo pri delu, ne uporabi, ko bi lahko pripomoglo k reševanju problemov, se ponovno odkriva ali celo popolnoma izgubi oziroma pozabi. Na sisteme, ki podpirajo obvladovanje znanja, bi lahko gledali kot na skupen organizacijski spomin [2], ki integrira neformalno, polformalno in formalno znanje z namenom omogočanja dostopa do skupnega znanja, ponovne uporabe in učinkovitejšega izvajanja individualnih oziroma kolektivnih opravil zaposlencev [19]. Pojem organizacijski spomin (*organizational memory*), ki mu določeni avtorji pravijo tudi skupni spomin (corporate memory) ali organizacijska baza znanja (*organizational knowledge base*), se nanaša na shranjene informacije iz zgodovine poslovnega sistema, s katerimi si lahko pomagamo pri sedanjih odločitvah [85].

Pogosto obvladovanje znanja temelji na uporabi in obvladovanju dokumentov. Tak pristop je sicer preprost, pragmatičen in prinaša hitro vidne učinke, vendar pa je povezan tudi s številnimi pomanjkljivostmi, kot so težave s ponovno uporabo znanja in da se posamezni subjekti znanja lahko pojavljajo v množici različnih dokumentov in formatov [70]. Z namenom učinkovitega in

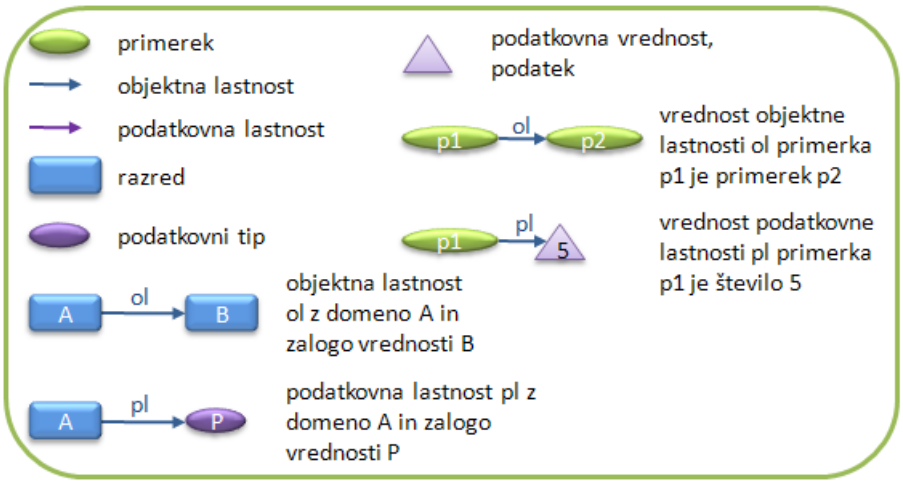
fleksibilnega obvladovanja znanja je potrebno znanje modelirati, strukturirati in določiti ustrezne medsebojne povezave med koncepti [78]. Izkazalo se je, da so primerna tehnologija, ki omogoča odgovor na probleme strukturiranja in modeliranja, ontologije [33],[54]. Zato je eden izmed danes pomembnejših pristopov k obvladovanju znanja, s katerim se ukvarja velik delež raziskovalcev tega področja, pristop z uporabo ontologij [33],[46],[49], [66],[70],[77],[84].

Opredelitev ontologije na področju informatike se razlikuje od klasičnih opredelitev ontologije, ki izhajajo iz filozofije. Beseda ontologija izhaja iz grškega jezika in v filozofiji pomeni *nauk o bitju in o tem kar je*. Na področju informatike ontologija pomeni *sredstvo, ki omogoča komunikacijo in skupno uporabo znanja z zajemanjem skupnega razumevanja izrazov, ki jih lahko uporabljajo tako ljudje kot programi* [46]. Organizacijskemu spominu, ki temelji na ontologijah, lahko pravimo organizacijska ontologija [28].

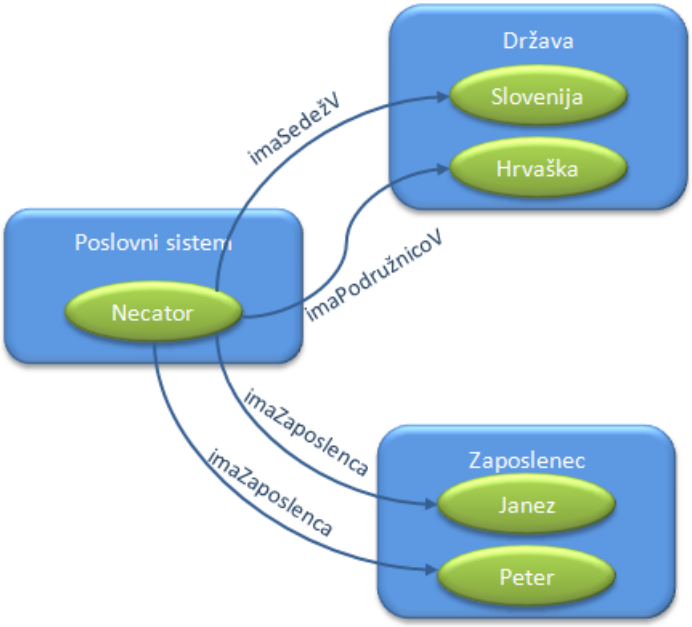
Ontologije se uporabljajo za zajem znanja o določeni domeni, za opis konceptov domene in medsebojnih odnosov oziroma povezav med njimi [33]. Za predstavitev ontologij je na voljo več jezikov, na primer DAML [34], CG [51], RDF [91], OIL [22], DAML+OIL [87] in OWL [89],[90]. Med njimi je danes najbolj razširjen jezik OWL (*Web Ontology Language*), ki je nastal pod okriljem dejavnosti *W3C Sematic Web Activity* z namenom olajšanja dostopa do informacij avtomatiziranim procesom [35]. Čeprav disertacija nima neposredne povezave z vizijo semantičnega spleta, se kot jezik ontologije uporablja OWL, ker omogoča predstavitev široke množice značilnosti ontologije, zaradi visoke ravni aplikativne podpore in ker temelji na jeziku XML. Osnova v XML ga naredi še posebej primernega za uporabo v kontekstu SOA tehnologij, ki v veliki večini prav tako temeljijo na XML.

Glavni koncepti OWL ontologije so razredi, lastnosti in primerki razredov. Lastnosti so binarne relacije nad primerki – med seboj povezujejo dva primerka (objektne lastnosti) oziroma primerku priredijo podatek določenega tipa (podatkovne lastnosti). Razredi vsebujejo formalne opise, ki natančno opredeljujejo pripadnost. Lahko jih razumemo kot množice primerkov. Razredi so lahko organizirani v hierarhijo razredov in podrazredov. Podobno velja tudi za lastnosti in podlastnosti. Ena izmed pomembnejših predpostavk v OWL ontologiji je predpostavka o odprtem svetu: določen primerek je lahko primerek kateregakoli razreda, za katerega v opisu ni podano nasprotno. Poleg podatkovnih in objektnih lastnosti lahko za poljuben koncept opredelimo tudi anotacijske lastnosti, ki so namenjene opombam in jih stroji za sklepanje ne upoštevajo. Slika 3

prikazuje simbole, ki jih bomo v disertaciji uporabljali za slikovni prikaz OWL ontologije. Slika 4 predstavlja primer uporabe simbolov ontologije.



Slika 3: Simboli za slikovno ponazoritev konceptov OWL ontologije



Slika 4: Ponazoritev konceptov ontologije

Na voljo so tri različice jezika OWL, in sicer OWL Lite, OWL DL in OWL Full. V disertaciji uporabljamo jezik OWL DL, ki temelji na opisni logiki in je namenjen uporabnikom, ki želijo na eni strani visoko izraznost in na drugi strani ohranjanje računske celovitosti (vsi zaključki so izračunljivi) ter odločljivosti (računanje se zaključi v končnem času).

Uporabo jezika OWL-DL izpopolnimo z jezikom SWRL (*Semantic Web Rule Language*) [86]. Jezik SWRL je namenjen razširitvi množice aksiomov jezika OWL s pravili tipa: »če drži predpostavka (telo pravila), potem sledi posledica (glava pravila)«. Tako telo kot glava pravila sta lahko sestavljena iz več atomov. Poleg OWL atomov se lahko uporabljajo tudi t.im. *vgrajeni* atomi (*built-ins*), ki izvirajo iz jezikov XPath [99] in XQuery [100], kar omogoča izdelavo bogate množice pravil, na primer z uporabo primerjav ali matematičnih izrazov.

2.2.2 Večkriterijsko odločanje

Večkriterijsko odločanje je že vrsto let področje številnih raziskav. Pričujoče delo obravnava večkriterijsko odločanje, kot je bilo opredeljeno v [14]. Trije ključni koncepti večkriterijskega odločanja so alternative (ali variante), kriteriji (ali parametri) in ocene. Imamo množico alternativ $A = \{a_1, a_2 \dots a_N\}$ in množico kriterijev $X = \{x_1, x_2 \dots x_M\}$, pri čemer je x_i funkcija, ki alternativo a_j preslika v določeno vrednost: $x_i: A \rightarrow D_i$, kjer je D_i zaloga vrednosti i-tega kriterija; $i \in \{1,2..M\}$ in $j \in \{1,2..N\}$. Za vsako alternativo določimo vrednosti kriterijev. Vrednost alternative glede na določen kriterij predstavlja oceno alternative po tem kriteriju. Končna odločitev je racionalna odločitev; to je izbira tiste alternative a_j iz A, ki je najbolj zaželena. Določimo jo s funkcijo koristnosti $v(a)$ in preferenčno relacijo P:

$$v(a) = v\big(x_1(a), x_2(a) \dots x_M(a)\big), a \in A,$$

$$a\ P\ b \ \dots\ a\ imam\ rajši\ kot\ b,$$

$$a\ P\ b \leftrightarrow v(a) > v(b).$$

Funkcija koristnosti več kriterijev ni nujno zvezna, ampak je lahko podana po točkah (zaloga vrednosti ocen je diskretna). V tem primeru lahko na funkcijo koristnosti gledamo kot na množico pravil. Merska lestvica, ki jo pri tem uporabljamo, je lahko kvalitativna ali kvantitativna. V disertaciji se ukvarjamo s kvalitativnim večkriterijskim odločanjem.

Navadno kriterije uredimo v drevo kriterijev, s katerim strukturiramo odločitveni problem. Pri tem združujemo vsebinsko sorodne kriterije. Število naslednikov vozlišč je praviloma omejeno. Razlog za to je zmanjšanje kompleksnosti in večja preglednost ter razumljivost pravil.

Faze odločitvenega procesa so praviloma naslednje: 1. Opredelitev problema, 2. Identifikacija kriterijev - 2.1 Opredelitev kriterijev (Izdelava seznama kriterijev), ki vplivajo na odločitev, 2.2 Izdelava drevesa kriterijev, 2.3 Določitev merskih lestvic, 3. Definicija odločitvenih pravil, 4. Opis alternativ, 5. Vrednotenje alternativ in 6. Analiza alternativ.

2.2.3 Večagentni sistemi

Večagentni sistemi so namenjeni predvsem razvoju velikih sistemov v odprtih in porazdeljenih okoljih, ki so danes vse pogostejša. Vendar lahko agentne tehnologije, metode in teorije uporabimo v raznoraznih področjih, kot so pridobivanje informacij, upravljanje z znanjem, uporabniški vmesniki, elektronsko trgovanje, robotika, računalniške igre itd. Zaradi širokega spektra namenskosti agentov splošno priznani opredelitvi pojmov agent in večagentni sistem ne obstajata. Kljub temu ju avtorji najpogosteje razumejo tako, kot je predstavljeno v nadaljevanju poglavja.

Agent je računalniški sistem, ki deluje v določenem okolju in je v njem sposoben avtonomnih akcij z namenom doseganja svojih ciljev [105]. Avtonomnost agenta pomeni, da vsebuje stanje, ki drugim agentom ni dostopno, in da glede na to stanje sprejema odločitve, brez neposrednega posredovanja drugih agentov ali uporabnika. Da je agent del okolja, pomeni, da je sposoben zaznavati svoje okolje oziroma del svojega okolja, da v njem deluje in s svojim delovanjem nanj vpliva [104].

Med ostalimi lastnostmi, ki jih pogosto pripisujemo agentom, so še reaktivnost, proaktivnost in socialnost. Reaktivnost pomeni, da je agent sposoben zaznavanja okolja in pravočasnega odzivanja na spremembe v njem z namenom, da doseže načrtovane cilje. Proaktivnost je ciljno usmerjeno obnašanje, predvsem v smislu samostojne pobude v pravem trenutku. To v praksi največkrat pomeni, da se agent glede na obstoječe znanje odloča o akcijah, ki jih bo izvajal. Socialnost se navadno nanaša na sposobnost sodelovanja z drugimi agenti in na sposobnost koordinacije in pogajanj v primeru konfliktnih interesov.

Večagentni sistem je sistem, ki je sestavljen iz agentov, ki so v medsebojni interakciji (bodisi sodelujejo bodisi tekmujejo) z namenom doseganja individualnih ali skupnih ciljev [32]. Zaradi podobnosti med agenti v večagentnem sistemu in človeškimi akterji v organizaciji poslovnega

sistema, predvsem v smislu njihovih značilnosti in koordinacije, lahko agente večagentnega sistema uporabimo za predstavitev akterjev v poslovnem sistemu.

Komunikacija med agenti se pogosto navezuje na teorijo govornih aktov. Teorija govornih aktov je komunikacijska teorija, katere začetnik je filozof J. L. Austin [9], kasneje pa je njegovo delo nadaljeval J. Searle [67]. Osnovna predpostavka te teorije pravi, da lahko *izjavo* primerjamo z izvajanjem akcij: če pomeni delovati spreminjati stanje določenih stvari v svetu, je govoriti spreminjanje miselnega stanja sogovornikov. Posledica tega razmišljanja je, da bi naj bilo mogoče določiti osnovne gradnike komunikacije, ki se imenujejo govorni akti (*speech acts*) ali komunikacijski akti.

Na področju agentov in večagentnih sistemov igra pomembno vlogo organizacija FIPA (*The Foundation for Intelligent Physical Agents*), ki izdeluje standarde ter jih skuša narediti združljive s standardi drugih tehnologij. Definirali so že več ključnih specifikacij agentov, med njimi tudi standard za jezik komunikacije agentov FIPA ACL [24] in vrsto protokolov interakcij (*FIPA Interaction Protocols*) [27].

Da lahko koordiniramo skupino heterogenih avtonomnih agentov, morajo ti agenti znati komunicirati v jeziku, ki jim bo vsem razumljiv. Jeziki za komunikacijo med agenti (*Agent Communication Languages, ACL*) so razviti posebej za področje agentov, njihov namen pa je, da bi komunikacijo naredili čim bolj enostavno. Skušajo minimizirati komunikacijo med agenti in se izogniti izčrpnim množicam ad hoc sporočil ter nekoristno razširjenim lestvicam protokolov. Osredotočeni bi naj bili predvsem na način, kako čim bolj učinkovito opisati komunikacijske akte, tako iz sintaktičnega kot semantičnega vidika, ki bo podpiral tudi jezik predstavitve znanj. Vsak ACL zahteva rigorozen standard tako semantike kot tudi pragmatike, saj je pomembno, da agenti pravilno razumejo pomen simbolov in da jih znajo pravilno uporabljati. Jezik komunikacije agentov mora tako vključevati naslednje vidike [105]:

- *sintaksa*, ki določa kako bodo simboli jezika strukturirani;
- *pragmatika*, ki določa uporabo in interpretacijo simbolov;
- *ontologija*, ki določa skupen slovar besed, s katerim bo predstavljena vsebina komunikacije.

V jezikih komunikacije je formalna določitev ontologije nujna, da se lahko agenti med seboj sporazumevajo. Določa, katere besede bodo agenti uporabljali pri komunikaciji in kaj bo njihov pomen (definicije besed). Format, ki se uporablja za izmenjavo znanj je podan v okviru jezika vsebine, neodvisno od jezika ACL.

Določitev jezika komunikacije, ki bo zagotavljal koherentno obnašanje agentov sistema in bo hkrati pokrival tudi ontološki vidik, je težka naloga in nastalo je več jezikov, ki so tako ali drugače skušali ugoditi tem zahtevam. Pred ustanovitvijo FIPA je bil edini poskus standardizacije jezika komunikacije med agenti KQML, iz katerega pa se je kasneje razvilo več različic, ki med seboj niso združljive. V nadaljevanju sta na kratko predstavljena jezika KQML in FIPA-ACL.

KQML (Knowledge Query and Manipulation Language)

KQML [23] je jezik za komunikacijo agentov, ki izhaja iz teorije govornih aktov. Namenjen je kooperaciji kognitivnih agentov. Vsebina sporočila je izraz, ki ustreza formatu KIF (*Knowledge Interchange Format*) in temelji na logiki prvega reda. KQML je hkrati jezik za komunikacijo in protokol visokega nivoja za izmenjavo informacij. Neodvisen je od mehanizma prenosa (TCP/IP, SMTP...), od jezika vsebine (Prolog, SQL, KIF...) ali uporabljene ontologije.

Z vidika načrtovanja lahko ločimo tri nivoje sporočila KQML: nivo vsebine, nivo komunikacije in nivo sporočila. Nivo vsebine dopušča uporabo kateregakoli jezika primernega sistemu, od jezikov izraženih v ASCII do jezikov izraženih z binarno predstavitvijo. Nivo komunikacij kodira množico gradnikov sporočila, ki opisujejo parametre komunikacije, kot so identiteta pošiljavca in prejemnika ter unikatni identifikator komunikacije. Nivo sporočila kodira sporočilo, ki ga prva aplikacija želi posredovati drugi, in predstavlja jedro KQML. Naloga tega nivoja je, da identificira vrsto govornega akta, ki ga pošiljavec doda vsebini. S to vrsto pove ali je sporočilo potrditev, vprašanje, ukaz ali kaj drugega. Poleg tega mora sporočilo vsebovati tudi druge parametre, ki povejo na primer, kateri jezik in katera ontologija se uporabljata. Parametri omogočajo pravilno analizo sporočil, brez potrebe po vpogledu v vsebino.

FIPA-ACL

Podobno kot KQML, tudi FIPA-ACL temelji na teoriji govornih aktov: sporočila so akcije oziroma komunikacijski akti, saj je namen njihovega pošiljanja, da se izvede določena akcija.

Specifikacije FIPA-ACL vsebujejo množico tipov sporočil in opisov njihove uporabe oziroma njihovih učinkov na miselno stanje agentov (pošiljavca ali prejemnika). Poleg tega opredeljujejo vsak komunikacijski akt z naravnim in formalnim opisom, ki temelji na pogojni logiki.

FIPA-ACL je na prvi pogled podobna KQML. Z izjemo nekaterih rezerviranih imen, je njuna sintaksa identična. Tudi FIPA-ACL razlikuje med zunanjim in notranjim jezikom. Zunanji določa pomen sporočila, notranji pa se nanaša na vsebino sporočila.

Najpomembnejša razlika med FIPA-ACL in KQML je v množici različnih tipov sporočil, ki jih določata, ter v izpopolnjenosti semantike. Semantika je namreč največja pomanjkljivost KQML in zato so razvijalci FIPA-ACL temu namenili posebno pozornost.

Na komunikacijske akte lahko med drugim gledamo tudi kot na gradnike pogovora med agenti. Če hočemo, da bo pogovor koherenten, je potrebna formalizacija ne le na nivoju gradnikov, ampak morata biti tudi logika in semantika komunikacijskih aktov razširjena za njihovo nizanje v pogovor, in sicer s pojmom protokola. V specifikacijah najdemo poleg komunikacijskih aktov tudi normativen opis množice protokolov interakcij visokega nivoja, vključno s prošnjo za akcijo, ustanovitvijo pogodbe in več vrst dražb.

Niti FIPA-ACL, niti KQML, ne določata implementacije, ki bi vnaprej predpisovala določen programski jezik ali računalniško okolje. Edina zahteva je, da bo implementacija skladna s specifikacijami jezika.

3 Področje raziskav

3.1 Opredelitev problema

Storitveno usmerjen pristop k avtomatizaciji poslovnih procesov je usmerjen v povezovanje različnih poslovnih, aplikativnih in strojnih storitev [20]. Posamezna opravila poslovnega procesa so lahko bodisi avtomatizirana bodisi jih izvaja določena oseba oziroma skupina oseb. Opravila poslovnega procesa, ki niso avtomatizirana, so uporabniška opravila. Doseganje višje ravni avtomatizacije poslovnih procesov se bo zato nanašalo na možnosti, ki izhajajo iz višje ravni avtomatizacije uporabniških opravil.

Obstoječe SOA rešitve uporabniška opravila realizirajo s pomočjo storitve za izvajanje uporabniških opravil. Z vidika poslovnega procesa je transparentno, ali gre za uporabniško ali avtomatizirano opravilo. Storitev za izvajanje uporabniških opravil je zadolžena za izdelavo seznamov opravil posameznih oseb, za posredovanje opravila ustrezni osebi ter po končanem opravilu za posredovanje rezultatov kot izhodnih podatkov prožene storitve. Osebo, ki je zadolžena za izvedbo uporabniškega opravila, imenujemo lastnik opravila. Potencialnih lastnikov opravila je lahko za posamezno opravilo več.

V SOA sistemih za izvajanje poslovnih procesov lahko opredelimo štiri ravni podpore uporabniškim opravilom:

Raven 1 – Osnovna podpora uporabniškim opravilom:

Uporabniška opravila skupaj z vhodnimi podatki sistem posreduje lastniku opravila. Lastnik opravilo izvede in vnese v sistem zahtevane rezultate opravila, ki so posredovani nazaj procesu.

Raven 2 – Uporabniška opravila s podporo uporabnikom:

Sistem je sposoben lastniku opravila predlagati morebitne odgovore oziroma rezultate opravila, odločitvene modele, urnike itd. skupaj z vhodnimi podatki ali med izvajanjem opravila. S

pomočjo predlogov lastnik opravilo izvede in vnese v sistem zahtevane rezultate opravila, ki so posredovani nazaj procesu.

Raven 3 - Polavtomatizirana uporabniška opravila:

Sistem s podporo polavtomatiziranim uporabniškim opravilom je sposoben izvajati določena uporabniška opravila namesto lastnikov opravila, na primer sprejemanje odločitev ali sklepanje. Tudi če sistem izvede opravilo namesto lastnika opravila, je lahko zahtevana potrditev rezultata s strani lastnika opravila (polavtomatizirana uporabniška opravila).

Raven 4 – Avtomatizirana uporabniška opravila:

Sistem je sposoben izvajanja uporabniškega opravila namesto lastnikov opravil, pri čemer je raven zanesljivosti izvedbe opravila tako visoka, da potrditev ni potrebna.

Obstoječe rešitve za podporo uporabniškim opravilom v SOA sistemih za izvajanje poslovnih procesov podpirajo zgolj uporabniška opravila na najnižji ravni (raven 1). Ker uporabniška opravila predstavljajo tisti del opravil poslovnega procesa, ki so neavtomatizirana, so z vidika izvajanja težavna: zavzemajo veliko časa, procesi so odvisni od razpoložljivega časa zaposlenih, težko je določiti natančen čas njihovega izvajanja, poleg tega pa izvedba uporabniškega opravila na prvi ravni omogoča zgolj sledljivost na podlagi rezultatov opravila, ne pa tudi sledljivosti posameznih korakov izvedbe opravila in njihovega vpliva na izvedbo in rezultat celotnega procesa. Vse to otežuje planiranje in optimizacijo procesov.

Analiza uporabniških opravil, kot tistega dela opravil poslovnega procesa, ki niso avtomatizirana, bi lahko pokazala na vrste uporabniških opravil, za katere bi lahko z ustreznim pristopom dosegli višjo raven njihove avtomatizacije in s tem tudi višjo raven avtomatizacije poslovnih procesov. Področje avtomatizacije poslovnih procesov z višjo ravnjo avtomatizacije izvajanja uporabniških opravil kot delov poslovnih procesov v storitveno usmerjenih sistemih je slabo raziskano in brez koristnih rezultatov, ki bi lahko bistveno pripomogli k učinkovitejšemu izvajanju poslovnih procesov.

3.2 Zadana naloga

Opredelitev modela za doseganje višje ravni avtomatizacije poslovnih procesov z višjo ravnjo avtomatizacije uporabniških opravil zahteva analizo različnih tipov uporabniških opravil in smiselnosti ter možnosti njihove avtomatizacije v okviru storitve za izvajanje uporabniških opravil. V analizi bodo identificirane tiste vrste uporabniških opravil, ki so smiselne za avtomatizacijo v storitvah za izvajanje uporabniških opravil.

Na podlagi rezultatov analize bo predlagan arhitekturni model za doseganje višje ravni avtomatizacije uporabniških opravil v okviru storitve za izvajanje uporabniških opravil. Pri tem bo avtomatizacija uporabniških opravil modela temeljila na večkriterijskem odločanju, ontologijah in večagentnih sistemih. Cilj je v čim večji meri doseči, da za v analizi identificirane vrste uporabniških opravil:

na podlagi vhodnih podatkov v opravilo in znanja v organizacijski ontologiji ter

s pomočjo stroja za sklepanje in večagentnih sistemov

storitev za izvajanje uporabniških opravil (SIUO) izvede uporabniško opravilo v imenu

njegovega lastnika oziroma lastnikov in vrne ustrezen rezultat nazaj primerku poslovnega

procesa oziroma drugemu odjemalcu, ki je storitev za izvajanje uporabniških opravil prožil.

Gre torej za doseganje tretje in četrte ravni podpore uporabniškim opravilom.

Model bo izdelan skladno z obstoječimi specifikacijami BPEL4People in WS-HumanTask. Po potrebi bodo specifikacije razširjene tako, da bodo podpirale integracijo predlaganega modela.

Zaradi prednosti, ki jih prinaša storitveno usmerjena arhitektura je fokus na izboljšanju izvajanja uporabniških opravil v SOA sistemih, čeprav se lahko model uporabi tudi v drugih tipih sistemov za izvajanje uporabniških opravil, na primer tradicionalnih sistemih za obvladovanje delovnih tokov.

Ker model temelji na tehnologijah obvladovanja znanja, bo uporaba modela opredeljena tudi z vidika aktivnosti obvladovanja znanja.

Ena izmed zadanih nalog je z uporabo inovativnih metod, ki bodo podpirale predlagani model, omogočiti sledljivost, in sicer ne le na podlagi rezultatov opravila, ampak tudi sledljivost posameznih korakov izvedbe opravila in njihovega vpliva na izvedbo ter na rezultat celotnega procesa.

3.3 Postavitev hipotez

V disertaciji bomo dokazali veljavnost naslednjih hipotez:

Hipoteza 1

Z definicijo in uporabo arhitekturnega modela, ki temelji na tehnologijah obvladovanja znanja, je možno doseči višjo raven avtomatizacije uporabniških opravil kot dela poslovnih procesov.

Hipotezo 1 v doktorski disertaciji razčlenimo na dve podhipotezi, in sicer na hipotezo 1.1 in hipotezo 1.2.

Hipoteza 1.1

Izdelati je mogoče metodo večkriterijskega odločanja, ki temelji na ontologijah in ki ob uporabi v storitvah za izvajanje uporabniških opravil v SOA sistemih omogoča višjo stopnjo avtomatizacije izvajanja uporabniških opravil, ki temeljijo na odločitvah.

Hipoteza 1.2

Izdelati je mogoče večagentni sistem, ki ob uporabi v storitvah za izvajanje uporabniških opravil v SOA sistemih omogoča koordinacijo uporabniških opravil in avtomatizira sodelovanje, ki temelji na poslovnih protokolih.

Hipoteza 2

Z izdelavo in uporabo modela storitve za izvajanje uporabniških opravil, ki omogoča doseganje višje ravni avtomatizacije uporabniških opravil kot dela poslovnih procesov, je mogoče doseči višjo raven sledljivosti odločitev v poslovnem procesu in višjo raven proaktivnosti.

Hipoteza 3

Aktivnosti opredelitve in uporabe modela za doseganje višje ravni avtomatizacije uporabniških opravil, ki temelji na tehnologijah obvladovanja znanja, dopolnjujejo in razširjajo obstoječe aktivnosti življenjskega cikla sistemov obvladovanja znanja ter zmanjšujejo razhajanja med področjem poslovnih procesov in sistemi obvladovanja znanja z neposredno povezavo med aktivnostmi modeliranja in implementacije poslovnih procesov ter aktivnostmi obvladovanja znanja.

3.4 Sorodne raziskave

S področjem izvajanja uporabniških opravil so se ukvarjali že številni raziskovalci. Disertacija ne skuša predlagati popolnoma novega pristopa k implementaciji uporabniških opravil v okviru sistemov za izvajanje poslovnih procesov, temveč je njen namen izpopolniti obstoječe pristope tako, da bo delo lastnikov opravil čim bolj avtomatizirano. Pregled sorodne literature je pokazal, da je pristopov, ki bi se ukvarjali z možnostmi višje ravni avtomatizacije poslovnih procesov z doseganjem višje ravni avtomatizacije uporabniških opravil v sistemih za izvajanje uporabniških opravil, le malo.

Predlagani model se povezuje z več raziskovalnimi področji, pri čemer pristop, ki bi obravnaval presek področij na način, kot ga obravnava disertacija, ne obstaja. Zato so v nadaljevanju poglavja opisana dela v sklopih, ki se povezujejo s posameznih raziskovalnim področjem oziroma podmnožico obravnavanih raziskovalnih področij.

Abecker in Decker sta opozorila, da je analiza problemov ekspertnih sistemov vodila k razvoju informacijskih sistemov organizacijskih spominov [3]. Model predlagan v disertaciji se od

ekspertnih sistemov [31] razlikuje po tem, da je namen omogočanje višje ravni avtomatizacije uporabniških opravil in podpora koordinaciji več človeških agentov med izvajanjem poslovnega procesa, in ne zgolj omogočanje podpore odločanju posameznemu človeškemu agentu. V disertaciji je uporabljen pojem organizacijskega spomina, ki temelji na ontologijah in združuje ontologije z večkriterijskim odločanjem. Cilj, ki ga s tem skušamo doseči, je razvoj metode, na osnovi katere lahko omogočimo polavtomatsko oziroma avtomatsko izvajanje uporabniških opravil, ki temeljijo na odločanju. Organizacijski spomin je področje, ki mu je bilo posvečena vrsta raziskav, le malo pa je avtorjev, ki so pojem organizacijskega spomina obravnavali v domeni poslovnih procesov na način, kot ga predlagamo v disertaciji. Abecker *et al.* [2] so razvili model sistema organizacijskega spomina, ki je sestavljen iz treh ravni: 1) objektna raven, ki je sestavljena iz raznolikih informacijskih virov in virov znanja; 2) raven opisa znanja, ki omogoča inteligenten dostop do raznovrstnih virov objektne ravni; 3) aplikacijska raven, v kateri so modelirani in izvajani procesi in opravila. Kaathoven *et al.* [42] so sistematsko raziskali integracijo sistemov organizacijskega spomina s sistemi WfMS. Na osnovi tega dela so Reimer *et al.* predlagali model z namenom podpore pisarniškim delavcem pri izvajanju opravil, pri čemer je šlo za aktivno podporo in usmerjanje [63]. Z izjemo dejstva, da ne obravnavajo uporabniških opravil v SOA sistemih, njihov pristop omogoča drugo raven podpore uporabniškim opravilom (glej poglavje 3.1 Opredelitev problema). Poglavitna razlika med navedenimi deli in pričujočo disertacijo je v tem, da fokus disertacije ni na podpori človeških akterjev pri izvajanju opravil, temveč na možnostih nadaljnje avtomatizacije njihovega dela z metodami, ki uporabljajo organizacijski spomin (doseganje tretje in četrte ravni podpore uporabniškim opravilom v SOA sistemih za izvajanje poslovnih procesov).

Fox *et al.* [28] in za njimi Ba *et al*. [10] so se ukvarjali z avtomatizacijo na področju odločanja, ki temelji na organizacijskem spominu. Fox *et al.* so podarili, da vloga informacijskih sistemov naj ne bi bila zgolj v ponujanju preprostih repozitorijev podatkov, temveč v omogočanju naprednejše podpore za neavtomatizirano in avtomatizirano odločanje. Pri tem trdijo, da bi morali informacijski sistemi biti sposobni dajanja odgovorov na poizvedbe eksplicitnih in implicitnih predstavitev v njihovem modelu poslovnega sistema. S ciljem rešiti to vprašanja so predlagali model poslovnega sistema TOVE [28]. Ba et al. so predlagali konceptualni model in spletno-osnovano arhitekturo za implementacijo omrežja znanja, ki pokriva celoten poslovni sistem, z namenom organizacije in integracije komponent znanja iz različnih virov in z namenom

omogočanja boljšega dostopa do informacij nosilcem odločanja, ki so lahko lokacijsko razpršeni. Pristop obravnava eksplicitno znanje, ki je formalno izraženo, in avtomatsko grajenje modelov za scenarije podane s poizvedbami [10]. Pomembne razlike v primerjavi s pričujočo disertacijo so, da odločanje obravnavamo v kontekstu izvajanja poslovnih procesov, ki temelji na načelih storitveno usmerjene arhitekture, da obravnavamo tacitno znanje in preslikavo v eksplicitno znanje ter da predlagamo metodo, ki omogoča avtomatsko poizvedovanje in generiranje odgovorov na poizvedbe in s tem višjo raven avtomatizacije poslovnega procesa. Poleg tega ne skušamo predlagati pristopa k integraciji informacij iz heterogenih virov, ki bi omogočale podporo človeškim akterjem pri njihovem delu, ampak v ta namen implicitno uporabimo SOA integracijske platforme za zbiranje informacij, ki jih nato preslikamo v organizacijski spomin.

Z izboljšanjem poslovnih procesov s pomočjo agentnih tehnologij se je ukvarjala vrsta raziskovalcev, vendar k temu cilju pristopajo na drugačne načine. Večina prispevkov tega področja se ukvarja z različnimi predlogi upravljanja in izboljšanja kompozicije poslovnih procesov z agenti in večagentnimi sistemi, na primer [6],[12],[13],[39] in [108]. Taveter in Wagner [83] sta predlagala t.im. radikalni agentno usmerjen proces (*Radical Agent-Oriented Process*; RAP), ki je osnovan na modeliranju AOR (*Agent-Object-Relationship*) in na metodologiji RAP/AOR, ki stremi k modeliranju, simulaciji in avtomatizaciji poslovnih procesov. Njun pristop je agentno-usmerjen in ne storitveno usmerjen. Drugi avtorji predlagajo implementacijo spletnih storitev z agentnimi tehnologijami z namenom realizacije kompleksne interakcije in koordinacije med storitvami, na primer [48] in [111]. Poleg tega nekateri avtorji obravnavajo upravljanje organizacijskega spomina z večagentnimi sistemi, na primer [1] in [29]. Kljub temu da v našem delu agenti delujejo kot posredniki med poslovnimi procesi, organizacijskim spominom in človeškimi akterji, je njihova glavna vloga omogočanje podpore sodelovanju med različnimi akterji in koordinaciji izvajanja uporabniških opravil.

Deli pričujoče doktorske disertacije so že bili objavljeni v prispevkih [80], [81]in [82].

4 Uporabniška opravila in analiza primernosti njihove avtomatizacije s storitvijo za izvajanje uporabniških opravil (SIUO)

4.1 Opravila

Dogajanje, ki se odvija v poslovnem sistemu, lahko opišemo z množico različnih opravil, ki so lahko bodisi avtomatizirana bodisi jih izvajajo ljudje (zaposleni, stranke itd.). Opravilo (ali aktivnost) ima veliko definicij. V tem delu opravilo razumemo kot *delo, ki mora biti opravljeno, da na podlagi vhodov v opravilo dobimo zahtevane oziroma želene izhode iz opravila.* Zato da je opravilo izvedeno, so lahko potrebni različni viri. Vhod v opravilo mora zato vsebovati vse, kar je potrebno za njegovo izvedbo. Opravilo tako lahko predstavimo kot funkcijo t:

$$t(inp_t) = r_t,$$

(F1)

ki vhode inp_t preslika v izhode r_t.

4.2 Kompozicija opravil

Opravila so lahko bodisi elementarna ali sestavljena (kompozitna). Elementarno opravilo je opravilo, ki ga ne moremo razstaviti v podopravila. Sestavljeno opravilo je opravilo, ki je sestavljeno iz enega ali več podopravil. Podopravilo je elementarno ali sestavljeno opravilo. Vsako sestavljeno opravilo ct lahko natančno opredelimo z njegovimi podopravili in kompozicijskimi funkcijami. Kompozicijske funkcije opredeljujejo:

- preslikave iz vhodov v opravilo in/ali izhodov enega ali več podopravil v vhode drugega podopravila ter

- preslikave iz vhodov v opravilo in/ali izhodov enega ali več podopravil v končni izhod (rezultat) opravila.

Opredelimo naslednje simbole (Tabela 1):

inp_t … vhod opravila t r_t … izhod (rezultat) opravila t $\overrightarrow{cf_{ct}} = (cf_{1ct}, cf_{2ct} \dots cf_{(n+1)ct})$ … vektor kompozicijskih funkcij opravila ct	$\overrightarrow{st_{ct}} = (st_{1ct}, st_{2ct} \dots st_{nct})$ … vektor podopravil sestavljenega opravila ct; n je število podopravil sestavljenega opravila ct

Tabela 1: Definicije simbolov za formalni zapis sestavljenih opravil

Kompozicijske funkcije opredelimo z naslednjo definicijo:

$$\forall i \in 1..(n+1): cf_{ict}\left(r_{st_{m_o ct}}, r_{st_{m_2 ct}} \dots r_{st_{m_3 ct}}\right) = cf_{ict}\left(\vec{r}_{cf_{ict}}\right) = \begin{cases} inp_{st_{ict}}, & i \in 1..n \\ r_{ct}, & i = n+1 \end{cases},$$

$$cf_{ict} \in \overrightarrow{cf_{ct}}, k \in 0..(i-1),$$

$$\forall l \in 0..k: m_l \in 0..(i-1), \wedge_{l1 \neq l2}\, m_{l_1} \neq m_{l_2},\ l_1, l_2 \in 0..k.$$

(F2a)

Pri tem so izhodi posameznih podopravil določeni z naslednjo definicijo:

$$r_{st_{m_l ct}} = \begin{cases} inp_{ct}, & m_l = 0 \\ st_{m_l ct}\left(inp_{st_{m_l ct}}\right), & m_l \in 1..(i-1), st_{m_l ct} \in \overrightarrow{st_{ct}} \end{cases}.$$

(F2b)

Rezultat sestavljenega opravila je torej enak rezultatu kompozicijske funkcije $cf_{(n+1)ct}$:

$$ct(inp_{ct}) = cf_{(n+1)ct}\left(\vec{r}_{cf_{(n+1)ct}}\right) = r_{ct}\,.$$

(F2c)

V interesu vsakega poslovnega sistema je, da poslovni procesi tečejo učinkovito in da vključujejo zgolj nujna opravila, saj lahko na ta način delujejo bolj učinkovito in hitreje [41]. Če ne želimo izvajanja redundantnih podopravil, mora veljati tudi naslednje:

$$\forall u \in 0..n: \exists z \in (u+1)..(n+1): r_{st_{uct}} \in \vec{r}_{cf_{zct}}.$$

(F3)

V nasprotnem primeru bi lahko podopravilo dalo rezultat, ki ne bi bil uporabljen za pridobitev končnega rezultata opravila.

Na podlagi definicij F1 in F2 je lahko sestavljeno opravilo natančno določeno z vektorjem podopravil in vektorjem kompozicijskih funkcij. Skupaj sestavljata t.im. kompozicijski par $(\overrightarrow{st}_{ct}, \overrightarrow{cf}_{ct})$ sestavljenega opravila ct:

$$ctup(ct) = (\overrightarrow{st}_{ct}, \overrightarrow{cf}_{ct}) = ((st_{1ct}, st_{2ct} \dots st_{nct}), (cf_{1ct}, cf_{2ct} \dots cf_{(n+1)ct})),$$

(F4)

kjer je $ctup(ct)$ funkcija, ki vrne kompozicijski par sestavljenega opravila ct.

Primer tako opredeljenega sestavljenega opravila je BPEL proces: vsaka storitev, ki jo proži, predstavlja podopravilo, medtem ko BPEL koda implementira kompozicijske funkcije sestavljenega opravila.

Iz definicij (F1-4) sledi, da je kompozicijski par elementarnega opravila et enak:

$$ctup(et) = (\overrightarrow{st}_{et}, \overrightarrow{cf}_{et}) = (\emptyset, (et))$$

(F5)

Nivo opravila t, $lvl(t)$, definiramo z naslednjim:

$$lvl(t) = \begin{cases} 0, \text{ če je } t \text{ elementarno opravilo} \\ max_i \{lvl(st_{it})\} + 1, st_{it} \in \overrightarrow{st}_t, sicer \end{cases}$$

(F6)

Nivo podopravila je torej za vsaj ena nižji od nivoja opravila, ki ga sestavlja, pri čemer je spodnja meja nivojev 0. Z rekurzivnim vstavljanjem kompozicijskih funkcij in podopravil lahko vsako sestavljeno opravilo pretvorimo v enakovredno opravilo sestavljeno zgolj iz elementarnih podopravil. Torej lahko predpostavimo, da če lahko avtomatiziramo kompozicijske funkcije, se problem avtomatizacije opravil spremeni v problem avtomatizacije elementarnih opravil. V obravnavani domeni to pomeni, da bi lahko ob tej predpostavki problem avtomatizacije poslovnih procesov pretvorili v problem avtomatizacije elementarnih uporabniških opravil.

Sestavljeno opravilo lahko prestrukturiramo tako, da kompozicija ne zahteva človeške dejavnosti. Če je le-ta vključena v kompozicijo rezultatov določenih podopravil, jo lahko implementiramo kot uporabniško (pod)opravilo, medtem ko je kompozicija tisto, kar podopravila povezuje med seboj. Na primer, 1) naj sta A in B podopravili sestavljenega opravila C, 2) rezultat opravila C je enak rezultatu opravila A ali rezultatu opravila B, 3) izbiro o tem, kateri izmed rezultatov opravil A in B bo izbran kot končni rezultat opravila C, določi oseba O. V tem primeru lahko uvedemo novo podopravilo D v katerem oseba O odloči o izbiri. Tako postanejo A, B in D podopravila opravila C.

V odvisnosti od zahtevane kompozicije, lahko kompozicijske funkcije avtomatiziramo z uporabo enega izmed jezikov za avtomatizacijo poslovnih procesov, na primer z jezikom BPEL, ali z uporabo druge tehnologije.

V poglavjih 4.3 in 4.4 bomo analizirali možnost in smiselnost avtomatizacije uporabniških opravil v SIUO glede na vrsto ter strukturo uporabniškega opravila.

4.3 Avtomatizacija elementarnih uporabniških opravil

Posamezen primerek (instanca) elementarnega uporabniškega opravila vedno izvede natanko eden lastnik opravila. Če primerek opravila izvedeta dva ali več lastnikov opravila, lahko opravilo razstavimo na podopravila, ki pripadajo posameznemu lastniku opravila.

Na uporabniško opravilo lahko gledamo z dveh zornih kotov: iz zornega kota poslovnega procesa in iz zornega kota lastnika oziroma lastnikov uporabniškega opravila. Iz zornega kota poslovnega

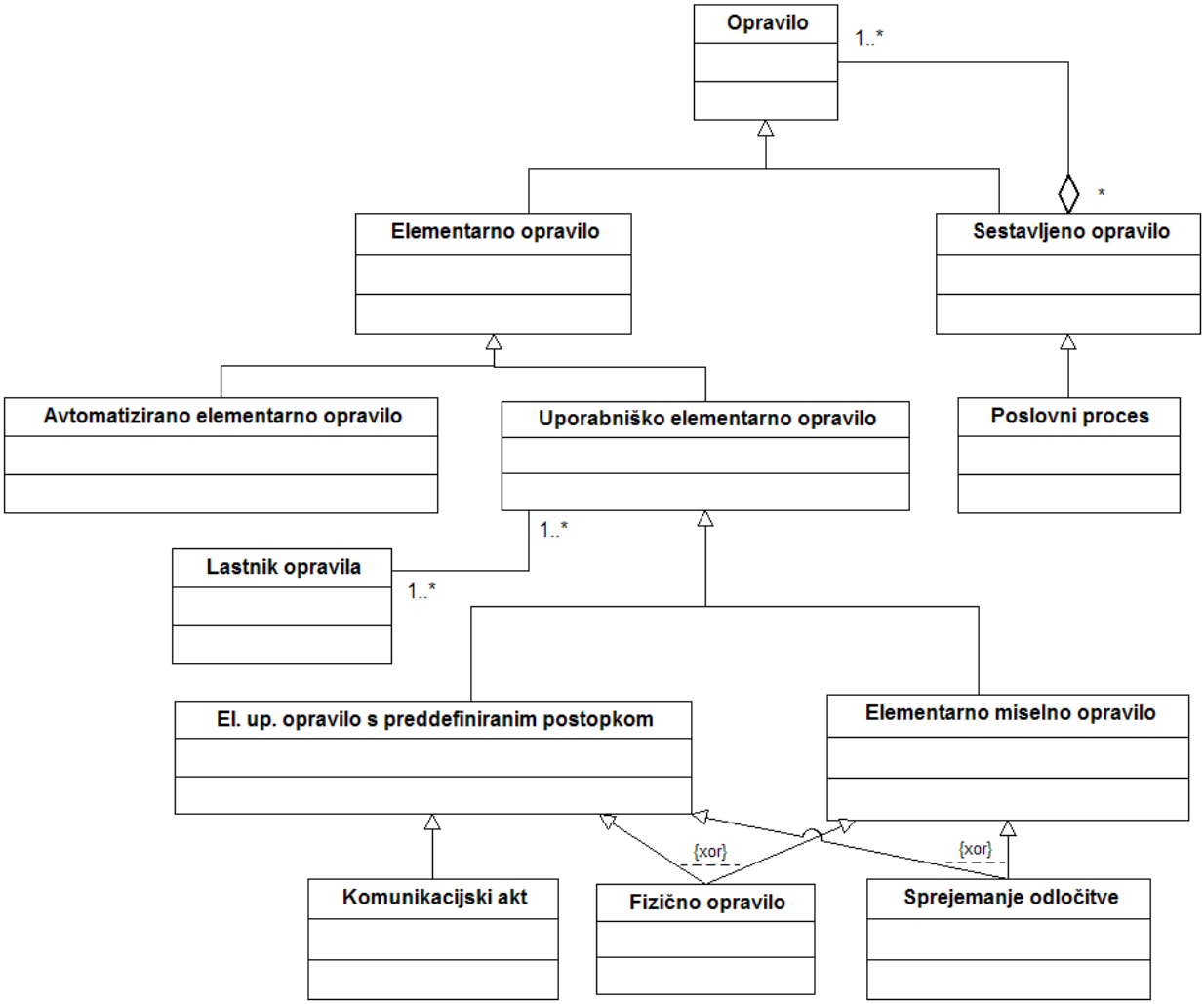
procesa se izvedba uporabniškega opravila s storitvijo za izvajanje uporabniških opravil ne razlikuje od izvedbe drugih storitev, ki jih proces proži, jim poda vhodne podatke in lahko pričakuje izhodne podatke. Implementacija storitve je za proces transparentna, ne glede na to ali je avtomatizirana ali jo izvede lastnik opravila.

Iz zornega kota lastnika uporabniškega opravila lahko opravilo predstavlja različne aktivnosti, ki jih mora izvesti med tem, ko mu je opravilo z vhodnimi podatki dodeljeno, in do takrat, ko vrne zahtevan rezultat opravila. Glede na njihovo naravo, lahko elementarna uporabniška opravila razvrstimo v dve skupini. V prvi skupini so opravila, za katere obstaja preddefiniran postopek, ki določa, kako naj bo opravilo izvedeno. Za tovrstna opravila velja, da ob predpostavki, da lastnik opravila postopek izvede pravilno, rezultat opravila ni odvisen od posameznega lastnika opravila – če bi opravilo pravilno izvedla druga oseba, ki pozna postopek, bi bil rezultat opravila enak. V drugi skupini so opravila, za katere postopek izvedbe ni predhodno določen, ampak je odvisen od posameznika, ki opravilo izvaja, od njegovega znanja, njegovih izkušenj, zaznave sveta itd. Imenujemo jih miselna opravila.

Posebna vrsta elementarnih uporabniških opravil s preddefiniranim postopkom so komunikacijski akti. Komunikacijski akt je elementarno uporabniško opravilo, ki od lastnika opravila zahteva, da komunicira določeno informacijo drugi osebi oziroma skupini oseb. Primer je pošiljanje pisma ali telefonski klic.

Za uporabniška opravila s preddefiniranim postopkom pogosto velja, da bi lahko bila avtomatizirana, na primer z implementacijo ustrezne informacijske podpore, uporabo strojev ali razvojem vmesnikov med obstoječimi informacijskimi sistemi. V fazi implementacije poslovnega procesa lahko avtomatska podpora ni na voljo, je predraga oziroma obstajajo drugi praktični razlogi, zaradi katerih se opravila izvajajo ročno. Včasih je bolj priporočljivo, da določena opravila implementiramo kot uporabniška opravila, ker storitve, ki bi bile potrebne za njihovo avtomatizacijo še niso na voljo, na primer so planirane ali so v fazi izdelave, vendar zaradi tega ne želimo ustaviti celotne implementacije poslovnega procesa. Izkaže se lahko tudi, da bo kasneje mogoče avtomatizirati določeno uporabniško opravilo, ki ga prej nismo mogli, na primer s prihodom nove tehnologije. V teh primerih lahko po izdelavi storitve, ki avtomatizira uporabniško opravilo, klic SIUO iz poslovnega procesa zamenjamo s klicem nove storitve. Na primer, nadzorni sistem, ki služi za aktivacijo določene funkcije na nekem stroju, še nima razvitih

vmesnikov, ki bi aktivacijo omogočali s klicem storitve. Če je aktivacija te funkcije del poslovnega procesa, jo začasno implementiramo kot uporabniško opravilo, v katerem bo zaposleni sprožil funkcijo preko nadzornega sistema. Ko bo storitev z vmesniki razvita, bo v poslovnem procesu klic storitve za izvajanje uporabniških opravil zamenjan sklicem nove storitve[1].



Slika 5: Metamodel opravil

Slika 5 prikazuje metamodel opravil z opisanimi koncepti. Komunikacijski akt, fizično opravilo in sprejemanje odločitve so primeri specializacije uporabniških elementarnih opravil. Sprejemanje odločitve lahko poteka na primer na podlagi preddefiniranih poslovnih pravil ali na

[1] V skladu s specifikacijami WS-HumanTask se klic storitve iz poslovnega procesa ne spremeni (vmesnik se ohrani), ampak se spremeni samo implementacija storitve.

podlagi miselnega procesa tistega, ki odloča. Poslovni proces je primer specializacije sestavljenega opravila.

Če uporabniška opravila s preddefiniranim postopkom niso avtomatizirana, praviloma za to obstajajo praktični razlogi, kot so omejena sredstva, fizične ali tehnološke omejitve. V primeru, da uporabniško opravilo s preddefiniranim postopkom avtomatiziramo, le-to predstavlja storitev neodvisno od posameznih oseb. Zaradi tega avtomatizacija elementarnih uporabniških opravil s preddefiniranim postopkom ne spada v okvir storitve za izvajanje uporabniških opravil.

Na drugi strani težave z avtomatizacijo miselnih uporabniških opravil praviloma ne izhajajo iz praktičnih razlogov, temveč iz dejstva, da njihova izvedba temelji na človeških miselnih procesih. Če bi jih želeli avtomatizirati, bi morala biti SIUO sposobna delovanja namesto lastnikov opravil. To pomeni izvajanje opravila na osnovi mentalnih karakteristik lastnika opravila, na primer njegovega znanja in preferenc. V primeru njihove avtomatizacije, bi bila izvedba tovrstnih opravil še vedno odvisna od posameznega lastnika opravila. Avtomatizacija elementarnih uporabniških opravil v okviru SIUO je tako smiselna za miselna opravila. Najpogostejša vrsta miselnih opravil, s katero se srečujemo v poslovnih sistemih, je sprejemanje odločitev.

4.4 Avtomatizacija sestavljenih opravil

Zaradi načel storitvene usmerjenosti, predvsem ponovne uporabe, šibke sklopljenosti in načela storitev brez stanja, je potrebno določiti primerno raven avtomatizacije kompozicije, ki jo naj nudi storitev za izvajanje uporabniških opravil.

Če storitev za izvajanje uporabniških opravil ne podpira avtomatizacije kompozicije in podpira le elementarna opravila, potem je njihova kompozicija lahko implementirana s poljubnim storitveno usmerjenim jezikom za izvajanje poslovnih procesov ali drugo tehnologijo, če se v danem primeru izkaže kot bolj primerna. Vendar pa takšen pristop ni primeren za nekatere tipe sestavljenih uporabniških opravil:

i) Identificiramo lahko določene vzorce sestavljenih uporabniških opravil, ki se pogosto pojavljajo v različnih sestavljenih uporabniških opravilih. Gre predvsem za opravila, ki

zahtevajo posredovanje uporabniškega opravila med različnimi lastniki opravila na osnovi določenega vzorca. Primer je zaporedno izvajanje enakega uporabniškega opravila, pri čemer vsak lastnik opravila prispeva k končnemu rezultatu sestavljenega opravila.

Za tovrstne vzorce kompozicije opravil, ni primerno, da je njihova implementacija naloga posameznih načrtovalcev poslovnih procesov. Zaradi njihove povezave z uporabniškimi opravili in visoke ponovne uporabe spadajo v okvir storitve za izvajanje uporabniških opravil. V nadaljevanju jih imenujemo uporabniška opravila, ki temeljijo na osnovnih vzorcih kompozicije. Če opravilo, ki temelji na osnovnih vzorcih kompozicije označimo s simbolom cp lahko s pomočjo (2-4) opredelimo potreben pogoj, ki ga izpolnjujejo uporabniška opravila, ki temeljijo na osnovnih vzorcih kompozicije:

$$\forall i_1, i_2 \in 1..n:\ st_{i_1 cp} = st_{i_2 cp} \wedge \forall i_3 \in 1..(n+1):$$

$$cf_{i_3 cp}(x) = x, cf_{i_3 cp} \in \overrightarrow{cf}_{cp}, st_{i_1 cp}, st_{i_2 cp} \in \overrightarrow{st}_{cp} \tag{F7}$$

Na primer, kompozicijske funkcije opravila, ki temelji na vzorcu zaporedne kompozicije (sc), imajo naslednjo obliko:

$$\forall i \in 1..(n+1):\ cf_{isc}\left(r_{st_{(i-1)sc}}\right) = r_{st_{(i-1)sc}} = \begin{cases} inp_{st_{isc}}, i \in 1..n \\ r_{sc},\ i = n+1 \end{cases},\ cf_{isc} \in \overrightarrow{cf}_{sc}, \tag{F8}$$

in v primeru, ko je n=2:

$$\overrightarrow{st}_{sc} = (st_{1sc}, st_{2sc}) = (st_{1sc}, st_{1sc}),$$

$$\overrightarrow{cf}_{sc} = (cf_{1sc}, cf_{2sc}, cf_{3sc}) = (cf_{1sc}, cf_{1sc}, cf_{1sc}),$$

$$sc(inp_{sc}) = r_{sc} = cf_{3sc}(r_{st_{2sc}}) = r_{st_{2sc}} = st_{2sc}(inp_{st_{2sc}}) = st_{1sc}(inp_{st_{2sc}}) =$$

$$st_{1sc}(r_{st_{1sc}}) = st_{1sc}(\ st_{1sc}(r_{st_{0sc}})) =\ st_{1sc}(\ st_{1sc}(inp_{ct})), \tag{F9}$$

To pomeni, da je $sc = st_{1sc} \circ st_{1sc}$.

Obstoječe SOA rešitve za podporo uporabniškim opravilom navadno že podpirajo ponovno uporabne vzorce sestavljenih opravil in ustrezajo pogoju (7), zato jih bomo v modelu le predvideli, medtem ko jih ne bomo posebej obravnavali.

ii) Obstajajo določena sestavljena uporabniška opravila, katerih implementacija z uporabo storitve za izvajanje uporabniških opravil, ki bi podpirala le elementarna uporabniška opravila, ni primerna, ker bodisi ne bi bila v skladu z načeli storitvene usmerjenosti bodisi bi predstavljala težave z zaupnostjo. Gre za sestavljena uporabniška opravila, za katere velja, da mora vhod v podopravilo vključevati rezultate predhodnih podopravil vključno z rezultati predhodnih podopravil istega lastnika. Eden izmed najpogostejših primerov tovrstnih sestavljenih opravil so uporabniška opravila, ki temeljijo na sodelovanju. Uporabniško opravilo, ki temelji na sodelovanju, - sodelovalno opravilo (suo), je sestavljeno iz podopravil dveh tipov:

- komunikacijski akt,
- opravila, namenjena pripravi sporočila za komunikacijo.

V opravilih priprave sporočil, udeleženci naredijo vse, kar je potrebno za to, da lahko sestavijo novo sporočilo, ki ga bodo komunicirali ostalim v sodelovalnem opravilu. To lahko pomeni zelo različne vrste uporabniških opravil. Če udeleženec pozna vsebino, ki jo bo zajemalo sporočilo, potem lahko to opravilo zajema zgolj oblikovanje sporočila na podlagi te vsebine. Če udeleženec vsebine še ne pozna oziroma jo pozna le delno, potem to opravilo zajema tudi pridobivanje informacij, ki jih bodo posredovali v sporočilu, na primer iskanje informacij v informacijskem sistemu, odločanje o neki zadevi na podlagi sporočil prejetih v predhodnih komunikacijah itd. Opravilo priprave sporočila je lahko poljubno uporabniško ali avtomatizirano opravilo, ki je lahko sestavljeno ali elementarno.

Če je $\overrightarrow{st}_{suo}$ vektor podopravil sodelovalnega opravila suo, $st_{isuo} \in \overrightarrow{st}_{suo}$ podopravilo priprave sporočila in to oseba, ki je lastnik podopravila, potem na podlagi (1-2) za sodelovalno opravilo velja naslednje:

$$inp_{st_{isuo}} = cf_{isuo}\left(r_{st_{m_0 suo}}, r_{st_{m_1 suo}} \dots r_{st_{m_k suo}}\right) = \left(r_{st_{m_0 suo}}, r_{st_{m_1 suo}} \dots r_{st_{m_k suo}}\right) = \vec{r}_{cf_{isuo}},$$

(F10)

kjer je $\vec{r}_{cf_{isuo}}$ vektor, ki vsebuje rezultate vseh predhodnih podopravil suo, ki so potrebni za izvedbo podopravila $isuo$. To lahko vključuje tudi rezultate podopravil, ki jih je opravila oseba to. Na primer, pri pogajanjih za ceno, je cena, ki jo udeleženec predlaga odvisna od cen, ki jih je predhodno predlagal in od odgovorov, ki jih je prejel na svoje predloge. To ni težavno, če opravilo ni avtomatizirano in ga izvaja oseba, ki si bi v normalnih okoliščinah (sama ali s pripomočki) zapomnila zgodovino sporočil. V nasprotnem primeru, če SIUO opravilo avtomatizira ter podpira zgolj elementarna uporabniška opravila, implementacija sodelovalnega opravila s pomočjo SIUO odpira vrsto težav. Možni sta dve alternativi:

1) SIUO shranjuje rezultate komunikacijskih aktov sodelovalnega opravila, na primer z ohranjanjem stanja, v dokumentih, podatkovni bazi itd.
2) Zgodovina sodelovanja se posreduje kot vhodni podatek pri proženju SIUO za izvedbo posameznih podopravil sodelovalnega opravila.

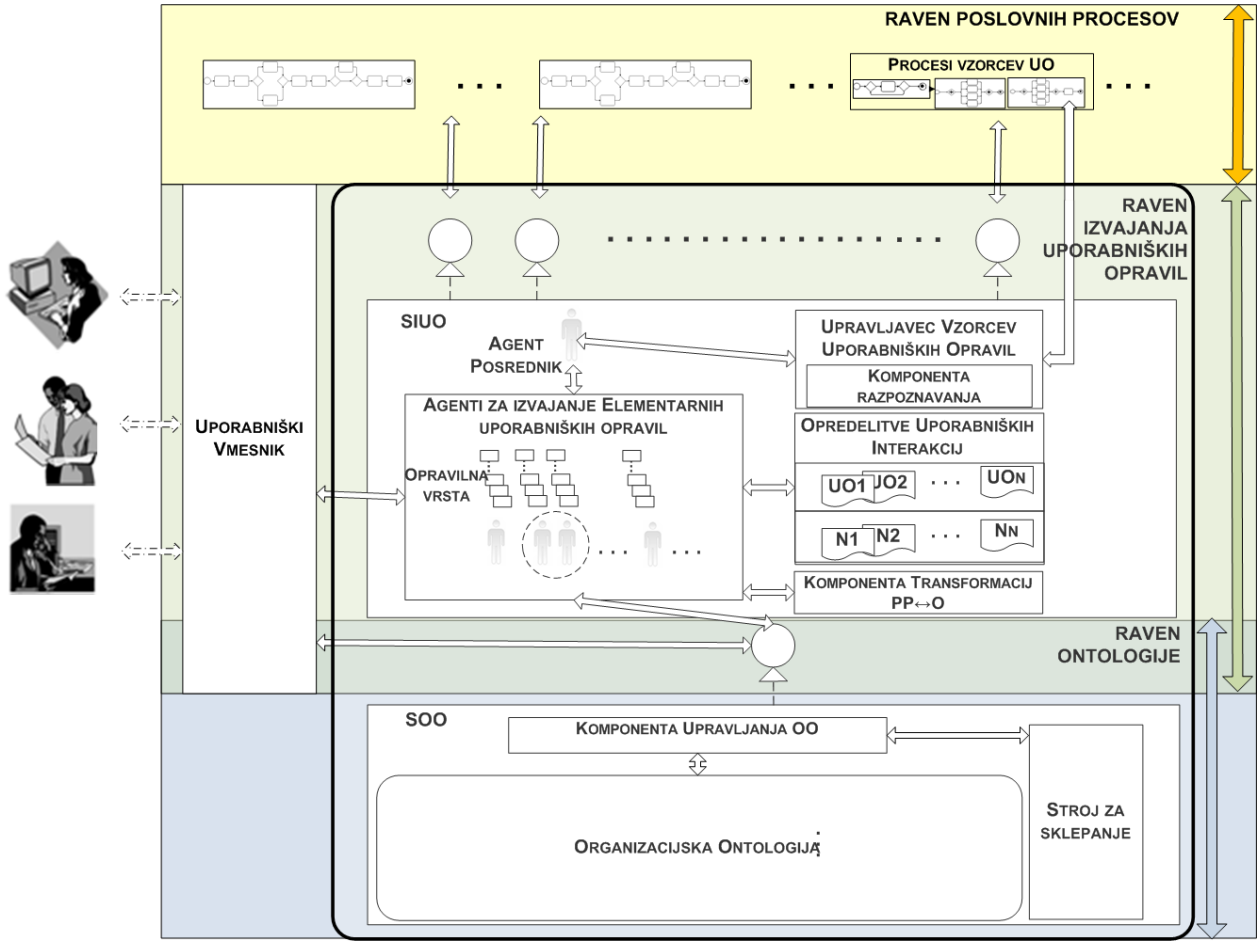
Obe alternativi imata ključne pomanjkljivosti. Prva alternativa ni v skladu z načelom storitev brez stanja [20], medtem kot druga alternativa predstavlja težave z zaupnostjo. Oseba bi morala zaupati drugim o zgodovini sodelovanja, med drugim tudi o svojih predhodnih odgovorih. Zaradi tega storitev za izvajanje uporabniških opravil poleg elementarnih uporabniških opravil ter uporabniških opravil, ki temeljijo na pogostejših vzorcih kompozicije, podpira tudi sodelovalna opravila.

5 Zasnova modela za doseganje višje ravni avtomatizacije poslovnih procesov v storitveno usmerjenih sistemih s pristopi in tehnologijami obvladovanja znanja

V tem poglavju podajamo zasnovo arhitekturnega modela storitve za doseganje višje ravni avtomatizacije poslovnih procesov skozi avtomatizacijo uporabniških opravil. Na podlagi načel storitvene usmerjenosti [20] je bil eden izmed ciljev raziskovalnega dela opredeliti generični arhitekturni model za avtomatizacijo poslovnih procesov (MAPP) skozi avtomatizacijo uporabniških opravil. Del arhitekturnega modela predstavlja arhitektura storitve za izvajanje uporabniških opravil, ki podpira poljubna uporabniška opravila. V odvisnosti od vrste uporabniškega opravila, lahko MAPP omogoča različne nivoje podpore uporabnikom in avtomatizacije njihovih opravil. Kot smo ugotovili v prejšnjem poglavju je avtomatizacija uporabniških opravil smiselna za miselna uporabniška opravila, za sestavljena uporabniška opravila, ki temeljijo na osnovnih vzorcih kompozicije, in za sodelovalna opravila.

Slika 6 prikazuje strukturo predlaganega arhitekturnega modela. Model obsega storitev za izvajanje uporabniških opravil (SIUO) in storitev organizacijske ontologije (SOO). Slika prikazuje tri različne ravni, ki nakazujejo širši kontekst uporabe storitev arhitekturnega modela. Raven poslovnih procesov prikazuje, kako lahko SIUO uporabljamo kot del izvajanja poslovnih procesov. Osnovni vzorci kompozicije so implementirani kot procesi vzorcev uporabniških opravil. Raven izvajanja uporabniških opravil obsega SIUO, uporabniški vmesnik in ljudi, ki opravila izvajajo. SIUO je zadolžena za izvajanje elementarnih in sodelovalnih opravil, medtem ko v primeru opravil, ki temeljijo na osnovnih vzorcih kompozicije, SIUO proži ustrezni poslovni proces, ki ga implementira. Ta spet proži SIUO za izvedbo uporabniških opravil, ki sestavljajo osnovni vzorec kompozicije. Gre torej za rekurzivna proženja, pri čemer je število rekurzivnih klicev odvisno od nivoja opravila. SIUO uporablja SOO. Raven ontologije obsega SOO in tista uporabniška opravila, ki se nanašajo na razvoj in uporabo organizacijske ontologije.

Osrednja komponenta SIUO je večagentni sistem sestavljen iz agenta posrednika in agentov za izvajanje elementarnih uporabniških opravil. Ob proženju SIUO opravilo prejme agent posrednik.



Slika 6: Arhitekturni model

Če gre za uporabniško opravilo, ki temelji na osnovnem vzorcu kompozicije, agent zahtevo posreduje upravljavcu vzorcev uporabniških opravil, ki je zadolžen za proženje ustreznega procesa, ki vzorec implementira. Rezultat proženega procesa posreduje kot izhod opravila v povratnem klicu storitve. Če ne gre za uporabniško opravilo, ki temelji na osnovnem vzorcu kompozicije, določi lastnike opravila in opravilo dodeli agentom za izvajanje elementarnih uporabniških opravil. Če gre za elementarno uporabniško opravilo, ga dodeli enemu izmed agentov. Če gre za sodelovalno opravilo, podopravila dodeli agentom, ki zastopajo posamezne udeležence sodelovanja. Agenti za izvajanje elementarnih opravil vsak zase izvajajo elementarna uporabniška opravila, medtem ko kot skupina lahko izvajajo sodelovalna opravila. Z uvajanjem

enega tipa agenta za izvajanje elementarnih uporabniških opravil, omogočimo, da lahko vsak agent zastopa poljubnega lastnika opravila. Ker je fokus na avtomatizaciji miselnih in sodelovalnih opravil, s tem funkcionalnosti ne omejujemo. Poleg tega je model zasnovan dovolj generično, da je njegova aplikacija primerna za različne implementacije; v primeru, da je v določenem okolju večja specializacija agentov potrebna, se lahko dani večagentni sistem ustrezno dopolni brez sprememb ostalih elementov modela.

Vsak agent za izvajanje elementarnih uporabniških opravil ima svojo opravilno vrsto. Agent posrednik agentom za izvajanje elementarnih uporabniških opravil opravila dodeljuje na podlagi časovnih rokov za izvedbo opravil, predvidenega trajanja in prioritet opravil v njihovih opravilnih vrstah ter opravila, ki ga dodeljuje. Posamezen agent za izvajanje elementarnih opravil med izvajanjem opravila zastopa določenega lastnika opravila, pri čemer pri prehodu na izvajanje drugega opravila lahko prevzame vlogo drugega lastnika, če opravilo zahteva drugega lastnika.

Struktura večagentnega sistema je dinamična. Če agent posrednik ne najde ustreznega agenta za izvedbo določenega opravila, na primer zaradi prezasedenosti, instanciira novega agenta za izvajanje elementarnih uporabniških opravil in mu dodeli opravilo. V nasprotnem primeru, če določen agent ali agenti niso aktivni, lahko zamrejo.

Specifikacije WS-HumanTask določajo opredelitev uporabniških opravil in notifikacij, kot dela opredelitve človeških interakcij. Opredelitev teh interakcij je vezana na posamezen vmesnik storitve. Zahteve po opredelitvi človeških interakcij izvirajo iz poslovnih procesov. Različni poslovni procesi lahko zahtevajo različne človeške interakcije. V ta namen je SIUO zasnovana tako, da lahko vsebuje opredelitve več vrst človeških interakcij. Glede na potrebe poslovnih procesov se lahko dodajajo nove oziroma spreminjajo obstoječe. Zato lahko ima SIUO več vmesnikov storitve. Posamezen vmesnik storitve je namenjen uporabi eni ali več opredelitev uporabniških interakcij.

Agenti za izvajanje elementarnih uporabniških opravil so zadolženi za omogočanje izvajanja elementarnih in sodelovalnih uporabniških opravil. V obeh primerih uporabljajo SOO. Ključna elementa SOO sta organizacijska ontologija in stroj za sklepanje. Agenti za izvajanje elementarnih uporabniških opravil skušajo priti do zahtevanega rezultata opravila z uporabo informacij v organizacijski ontologiji. Če rezultata ne morejo doseči na ta način (v organizacijski

ontologiji ni dovolj informacij), opravila ne morejo avtomatizirati. V tem primeru opravilo posredujejo lastniku opravila preko uporabniškega vmesnika skupaj z relevantnimi informacijami, ki bi lastniku lahko pripomogle pri izvajanju opravila.

Določanje lastnika opravila se ne razlikuje od izvedbe drugih opravil: rezultat je določen z opredelitvijo lastnika opravila, časovni rok, časovni okvir izvajanja in prioriteta so določene na podlagi samega opravila, lastnik pa je administrator poslovnega procesa, iz katerega je bila SIUO prožena. Zato agent posrednik določanje lastnika opravila obravnava kot opravilo in ga dodeli enemu izmed agentov za izvajanje elementarnih uporabniških opravil. V primeru, ko lastnika ne more določiti avtomatsko z uporabo SOO, SIUO proži izjemo. Kako je izjema obravnavana, je odvisno od posameznega poslovnega procesa oziroma drugega uporabnika SIUO. Primer je ponovno proženje SIUO z opravilom namenjenim administratorju poslovnega procesa, ki bodisi določi drugega lastnika oziroma izvede ustrezen ukrep.

V nadaljnjih poglavjih so predstavljene podrobnosti arhitekturnega modela in metode, značilne za posamezne elemente modela, ki omogočajo realizacijo zadanih nalog. V šestem poglavju je podana metoda večkriterijskega odločanja, ki temelji na ontologijah. Ta predstavlja osnovo za avtomatizacijo in polavtomatizacijo miselnih uporabniških opravil, ki temeljijo na sprejemanju odločitev. V sedmem poglavju je podrobneje obravnavana avtomatizacija sodelovalnih opravil. Obe poglavji temeljita predvsem na razvoju in uporabi SOO (organizacijske ontologije ter stroja za sklepanje). V osmem poglavju je zasnova arhitekturnega modela dopolnjena tako, da skupaj z metodami in koncepti, ki so potrebni za avtomatizacijo in polavtomatizacijo (podanimi v šestem in sedmem poglavju), tvori polno rešitev modela za doseganje višje ravni avtomatizacije s tehnologijami obvladovanja znanja.

6 Metoda večkriterijskega odločanja, ki temelji na ontologijah

6.1 Cilji

Cilj poglavja je podajati metodo večkriterijskega odločanja, ki temelji na ontologijah. Ker med elementi OWL ontologij in elementi večkriterijskega odločanja ni neposredne preslikave, je ena izmed zahtev za opredelitev metode določiti nedvoumna pravila, ki določajo, kako elementi ontologije v posameznem koraku ustrezajo konceptom večkriterijskega odločanja.

Predpostavka je, da odločitveni model gradimo na že obstoječi ontologiji. V primeru, da določen koncept, potreben za odločanje, ni zajet v ontologiji, ga je potrebno umestiti v ontologijo. Umestitev novega koncepta v ontologijo je opisana v poglavjih 8.6 Spremembe konceptualne ravni organizacijske ontologije, ki izvirajo iz opredelitve in izvajanja uporabniških opravil in 8.7 Umestitev modela v metodologije obvladovanja znanja, ki temeljijo na ontologijah..

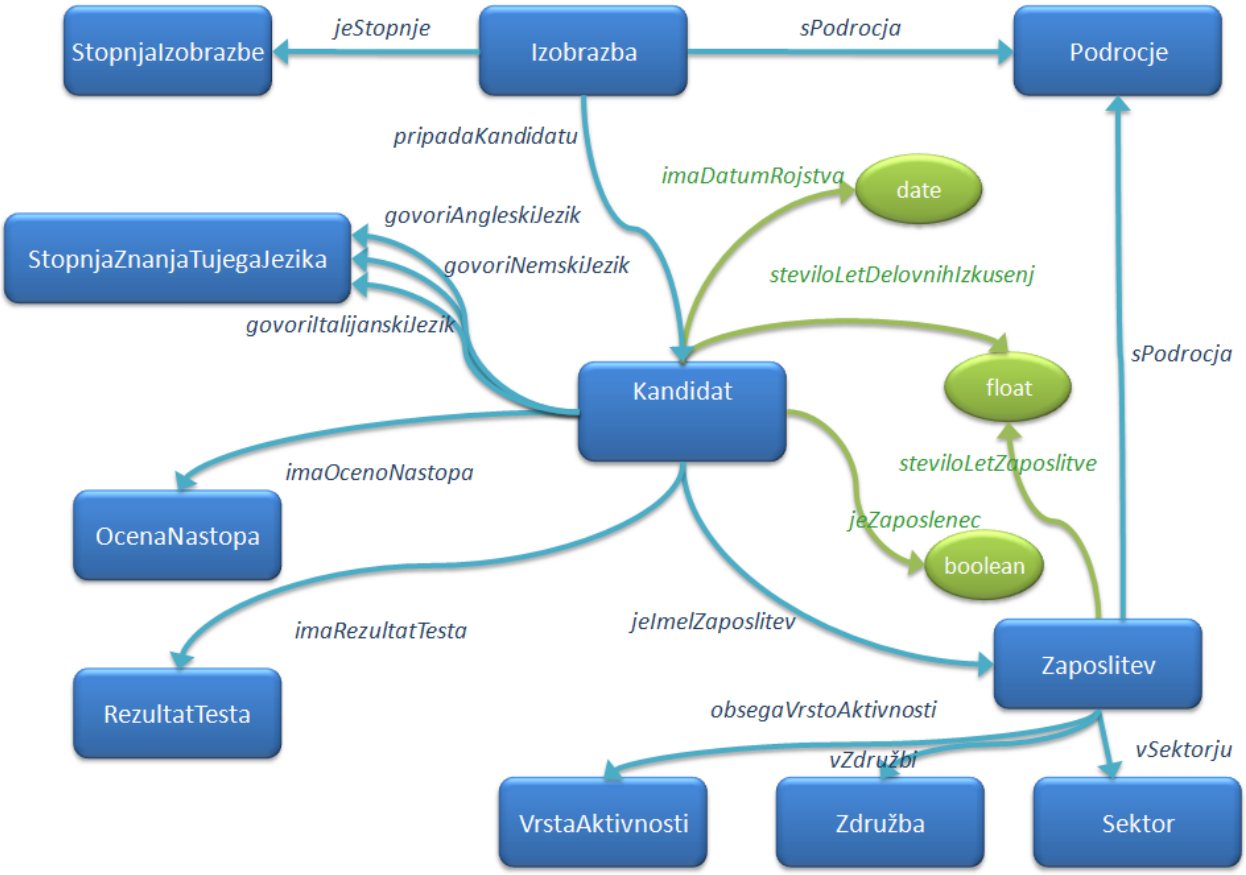
Za ponazoritev postopka so koraki opisani tudi s pomočjo primera izbire najboljšega kandidata za zaposlitev na dano delovno mesto. Primer je zasnovan tako, da je enostaven za razumevanje in da hkrati podpira različne strukture, ki se pojavljajo v ontologijah in so potrebni razlago konceptov. Ontologija, na kateri temelji primer, je prikazana na Sliki 7. Za primerjavo s klasičnim večkriterijskim odločanjem je podan tudi primer odločitvenega drevesa (Slika 8). Na mestih, kjer je potrebna razlaga, ki je dani primer ne omogoča, so podani dodatni primeri. Celoten primer odločitvenega modela zapisanega v OWL ontologiji je podan v Prilogi 3 (Primer odločitvenega modela v OWL ontologiji).

Faze postopka odločanja, ki temelji na ontologijah, so naslednje:

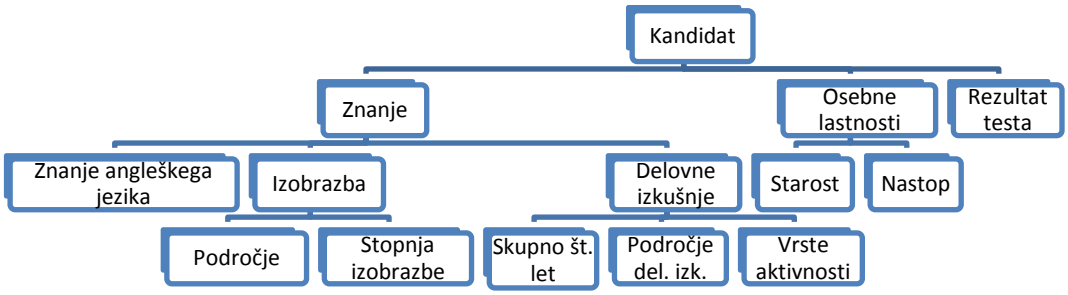
1. *Opredelitev problema*
2. *Identifikacija kriterijev*
3. *Strukturiranje kriterijev in identifikacija vrednosti merske lestvice*
   a. *Identifikacija kriterijev*

   b. *Izdelava podrazredov kriterijev*
4. *Opredelitev odločitvenih pravil*
5. *Opis alternativ*
6. *Vrednotenje alternativ*
7. *Analiza alternativ*

V nadaljevanju poglavja so opisane posamezne faze postopka odločanja.



Slika 7: Primer ontologije



Slika 8: Primer odločitvenega drevesa

6.1 Opredelitev problema

V OWL ontologiji alternative odločanja ustrezajo primerkom določenega razreda. Ker gre za obstoječo ontologijo, razred, ki ustreza primerkom bodisi že obstaja (v ontologiji je podan eksplicitno), bodisi ga lahko opredelimo iz že obstoječih konceptov.

Faza zajema identifikacijo ali opredelitev razreda, ki predstavlja množico alternativ (RA).

V primeru kandidatov za zaposlitev na dano delovno mesto iščemo najboljšega izmed kandidatov. V primeru ontologije je razred, ki predstavlja množico alternativ, razred $Kandidat$.

Primer množice alternativ, ki v ontologiji ni eksplicitno predstavljena, bi lahko našli v odločanju med obstoječimi zaposlenci. Razred alternativ, ki predstavljajo zaposlence, lahko opredelimo z naslednjim izrazom:

$$Zaposlenec \equiv Oseba \wedge \forall jeZaposlenec.\{true\}. \quad \text{(F11)}$$

6.2 Identifikacija kriterijev

Identifikacija kriterijev v ontologiji pomeni identifikacijo vseh tistih razredov in lastnosti, ki vplivajo na odločitev. Ker se odločamo med primerki razreda alternativ, je pogoj, ki ga morajo izpolnjevati kriteriji v ontologiji, da neposredno ali posredno opisujejo primerke razreda alternativ. Zato je potrebno natančno in nedvoumno določiti njihovo povezavo z razredom alternativ.

Za bolj nazorno nadaljnjo razlago določimo naslednje preslikave:

- $RD(L)$… preslikava objektne ali podatkovne lastnosti L v množico vseh razredov njene domene,
- $RZV(OL)$… preslikava objektne lastnosti OL v množico vseh razredov njene zaloge vrednosti,
- $PTZV(PL)$… preslikava podatkovne lastnosti PL v podatkovni tip njene zaloge vrednosti.

Poleg tega naj bo:

- $LD(R)$… preslikava razreda R v množico vseh tistih lastnosti, za katere je R razred njihove domene: $\forall L \in LD(R): R \in RD(L)$,
- $OLZV(R)$… preslikava razreda R v množico vseh tistih objektnih lastnosti, za katere je R razred njihove zaloge vrednosti: $\forall L \in LZV(R): R \in RZV(L)$.

Opredelimo relacijo $povezanZ$ med dvema razredoma. Zanjo naj velja naslednje:

PZ1 $\forall R1 \subseteq \top: R1\ povezanZ\ R1$

PZ2 $R1\ povezanZ\ R2 \equiv$
$\left(\exists OL: R1 \in RD(OL) \wedge R2 \in RZV(OL)\right) \vee (\exists R3: R1\ povezanZ\ R3\ \wedge R3\ povezanZ\ R2)$,

PZ3 $R1\ povezanZ\ R2 \leftrightarrow R2\ povezanZ\ R1$,

kjer je OL objektna lastnost in $R1, R2, R3 \subseteq \top$ razredi. Ugotovimo lahko, da gre za ekvivalenčno relacijo, saj veljajo refleksivnost, tranzitivnost in simetričnost.

Lastnost se nanaša na določen razred oziroma več razredov (razredi domene lastnosti), zato lahko tudi v primeru lastnosti govorimo o povezanosti danega domenskega razreda lastnosti in razreda alternativ. Identifikacija posameznih razredov in lastnosti povezanih z razredom alternativ ne bi pomenila vedno enolične opredelitve kriterija. Možnih primerkov relacij $povezanZ$ med posameznim razredom in razredom alternativ je lahko namreč več. Naj bo RO množica vseh razredov, ki vplivajo na odločitev, in LO množica vseh lastnosti, ki vplivajo na odločitev. Za dani razred lastnosti, ki vplivajo na odločitev, ne moremo določiti zgolj s presekom množic LO in $OLZV(R1)$, saj rezultat ne bi bil vedno pravilen. Na primer, naj za objektno lastnost $L1$ in razred $R1$ velja naslednje:

$$RD(L1) = \{R1\},$$

$$RZV(L1) = \{R1, R2\} \subseteq RO,$$

$$OLZV(R1) = \{L1, L2\} \subseteq LO.$$

(F12)

Presek LO in $LZV(R1)$ je množica $\{L1, L2\}$. Vendar, kljub temu da je lastnost $L1$ opredeljena kot lastnost, ki vpliva na odločitev, to še ne pomeni, da vpliva na odločitev z razredom $R1$, kot razredom zaloge vrednosti. Obstaja namreč tudi možnost, da vpliva na odločitev zgolj z razredom zaloge vrednosti $R2$, medtem ko razred $R1$ vpliva na odločitev z lastnostjo $L2$. To pomeni, da opredelitev množice razredov in množice lastnosti, ki vplivajo na odločitev, ni dovolj za nedvoumno določitev kriterijev.

Predlagamo rešitev, ki kriterije opredeljuje z vektorjem trojic, ki ga imenujemo *veriga povezanosti kriterija*. Trojica $\vec{T}$ verige povezanosti je vektor, sestavljen iz bodisi dveh razredov in objektne lastnosti (objektna trojica), bodisi iz razreda, podatkovnega tipa in podatkovne lastnosti (podatkovna trojica), in sicer je $\vec{T}$ lahko:

- $\vec{T} = (R1, \mathrm{O}L, R2)$, kjer velja:

- da je OL objektna lastnost: $R1 \in RD(OL), R2 \in RZV(OL)$,
- $R1\ povezanZ\ RA$; ali

- $\vec{T} = (R1, PL, D)$, kjer velja:
  - PL je podatkovna lastnost: $R1 \in RD(PL), D = PTZV(PL)$,
  - $R1\ povezanZ\ RA$.

V obeh primerih razred $R1$ imenujemo domenski razred trojice.

Kriteriji opisujejo alternative. Zato je potreben pogoj, da obstaja med domenskim razredom vsake trojice (v verigi povezanosti kriterija) in razredom alternativ RA relacija *povezanZ*. Ker posamezna trojica dopušča možnost več kot ene možne povezave med obema razredoma, so trojice razvrščene v verigo povezanosti $Ch(K, RA) = \overrightarrow{vp}_{K,RA}$, ki enolično določa kriterij.

Opredelimo naslednje relacije za dani vektor trojic $\overrightarrow{vt}$ dolžine d in naravno število i:

- $R1(\overrightarrow{vt}, i) \equiv (\overrightarrow{vt}[i][1] = \overrightarrow{vt}[i-1][1]) \vee (\overrightarrow{vt}[i][1] = \overrightarrow{vt}[i-1][3]), i \in 2..d$,
- $R2(\overrightarrow{vt}, i) \equiv (\overrightarrow{vt}[i][1] = \overrightarrow{vt}[i+1][1]) \vee (\overrightarrow{vt}[i][1] = \overrightarrow{vt}[i+1][3]), i \in 1..(d-1)$,
- $R3(\overrightarrow{vt}, i) \equiv (\overrightarrow{vt}[i][3] = \overrightarrow{vt}[i+1][1]) \vee (\overrightarrow{vt}[i][3] = \overrightarrow{vt}[i+1][3]), i \in 1..(d-1)$,
- $R4(\overrightarrow{vt}, i) \equiv (\overrightarrow{vt}[i][3] = \overrightarrow{vt}[i-1][1]) \vee (\overrightarrow{vt}[i][3] = \overrightarrow{vt}[i-1][3]), i \in 2..d$.

Naj bo l dolžina vektorja verige povezanosti $Ch(K, RA) = \overrightarrow{vp}_{K,RA}$. Za verigo povezanosti veljajo naslednje omejitve:

VP1 Če je $\overrightarrow{vp}_{K,RA}[1]$ objektna trojica, potem velja:

$$\left(\overrightarrow{vp}_{K,RA}[1][1] = RA \wedge R3\left(\overrightarrow{vp}_{K,RA}, 1\right)\right) \vee \left(\overrightarrow{vp}_{K,RA}[1][3] = RA \wedge R2(\overrightarrow{vp}_{K,RA}, 1)\right).$$

Sicer, če je $\overrightarrow{vp}_{K,RA}[1]$ podatkovna trojica, potem velja:

$\overrightarrow{vp}_{K,RA}[1][1] = RA \wedge l = 1.$

VP2 Če je $l > 2$, potem za $\forall i \in 2..(l-1)$:

- $\overrightarrow{vp}_{K,RA}[i]$ je objektna trojica in
- velja natanko eden od naslednjih pogojev:

VP2.a $R1(\overrightarrow{vp}_{K,RA}, i) \wedge R3(\overrightarrow{vp}_{K,RA}, i)$

VP2.b $R2(\overrightarrow{vp}_{K,RA}, i) \wedge R4(\overrightarrow{vp}_{K,RA}, i)$

VP3 Za $\overrightarrow{vp}_{K,RA}[l]$ velja natanko ena izmed naslednjih trditev:

VP3.a $\overrightarrow{vp}_{K,RA}[l]$ je podatkovna trojica. Zanjo velja:

$$R1(\overrightarrow{vp}_{K,RA}, l)$$

VP3.b $\overrightarrow{vp}_{K,RA}[l]$ je objektna trojica. Zanjo velja:

$$(R1(\overrightarrow{vp}_{K,RA}, l) \vee R4(\overrightarrow{vp}_{K,RA}, l))$$

To pomeni, da v $\overrightarrow{vp}_{K,RA}$ obstaja največ ena podatkovna trojica $\overrightarrow{PT}$. Če obstaja, potem velja:

$$\overrightarrow{PT} = \overrightarrow{vp}_{K,RA}[l].$$

Če bi veriga povezanosti vsebovala zaporedje razredov in lastnosti, ne bi bilo vedno jasno, kdaj določen razred nastopa v vlogi domene lastnosti in kdaj v vlogi zaloge vrednosti. Zato predlagana struktura temelji na trojicah.

Kriteriji, kot jih določimo v tej fazi, v klasičnem večkriterijskem odločanju ustrezajo seznamu kriterijev, preden jih strukturiramo v odločitveno drevo. Imenujemo jih tudi *osnovni kriteriji*. V klasičnem večkriterijskem odločanju je to vsebina faze opredelitve kriterijev (glej poglavje 2.2.2 Večkriterijsko odločanje). Po izdelavi odločitvenega drevesa kriterije, identificirane v tej fazi,

najdemo v listih drevesa. Primer kriterija v odločitvenem drevesu je *Stopnja izobrazbe*. Pripadajoči kriterij v ontologiji določimo z naslednjo verigo povezanosti:

$$Ch(Stopnja\ izobrazbe, Kandidat) = \overrightarrow{vp}_{Stopnja\ izobrazbe,Kandidat}$$
$$= \{(Izobrazba, pripadaKandidatu, Kandidat), (Izobrazba, jeStopnje, StopnjaIzobrazbe)\} \quad \text{(F13)}$$

Na enostaven način je moč preveriti, da veriga zadošča omejitvam VP1-VP3.

6.3 Strukturiranje kriterijev in identifikacija vrednosti merske lestvice

Faza strukturiranja kriterijev in identifikacije vrednosti merske lestvice ustreza fazi izdelave drevesa kriterijev in fazi določitve merskih lestvic v klasičnem večkriterijskem odločanju (glej poglavje 2.2.2 Večkriterijsko odločanje). Do razhajanja v imenu faze in do združitve dveh faz v eno prihaja zaradi dejstva, da v ontologiji ni eksplicitnega odločitvenega drevesa, in zaradi dejstva, da strukturiranja kriterijev v ontologiji ne moremo izvajati ločeno od določanja vrednosti merskih lestvic.

Faza zajema dva koraka:

1 identifikacijo izpeljanih kriterijev, ki zajema razvrščanje sorodnih kriterijev v skupine in identifikacijo razredov izpeljanih kriterijev,

2 izdelavo podrazredov kriterijev, ki zajema določitev vrednosti merske lestvice razredov izpeljanih in osnovnih kriterijev in izdelavo podrazredov razredov izpeljanih kriterijev za vsako identificirano vrednost merske lestvice.

Oba koraka sta podrobneje opisana v naslednjih podpoglavjih (6.3.1 in 6.3.2).

6.3.1 Identifikacija izpeljanih kriterijev

Kriterije, ki smo jih identificirali v prejšnji fazi, razvrstimo v več vsebinsko sorodnih skupin. Priporočljivo je, da v posamezni skupini nista več kot dva ali trije kriteriji [14]. Pomembno je, da

določimo maksimalno število posameznih kriterijev v skupini. Skupine predstavljajo izpeljane kriterije prvega nivoja. Za vsak izpeljani kriterij je potrebno določiti razred v ontologiji, ki skupino predstavlja. Pri tem je ciljni razred tisti razred, ki ga kriteriji skupine opisujejo. Imenujemo ga razred izpeljanega kriterija.

Če je število izpeljanih kriterijev, ki jih s tem dobimo, večje od maksimalnega števila kriterijev v skupini, tudi izpeljane kriterije razvrstimo v več vsebinsko povezanih skupin. Te predstavljajo izpeljane kriterije drugega nivoja. Tudi zanje je potrebno določiti pripadajoče razrede v ontologiji. Postopek ponavljamo dokler število kriterijev nivoja ne postane manjše ali enako od maksimalnega števila kriterijev v skupini. Takrat dosežemo nivo (N-1), kjer N predstavlja globino pripadajočega odločitvenega drevesa. Skupina izpeljanih kriterijev nivoja N-1 je predstavljena z razredom alternativ (RA). Razred alternativ se tako nahaja v vrhu odločitvenega drevesa in predstavlja razred izpeljanega kriterija nivoja N. Kriteriji, ki smo jih identificirali v predhodni fazi so, v skladu s terminologijo nivojev, na nivoju 0. V odločitvenem drevesu jih najdemo v listih drevesa.

V danem primeru odločitvenega drevesa lahko najdemo pet izpeljanih kriterijev, in sicer: $Znanje$, $Izobrazba$, $Delovne\ izkušnje$, $Osebne\ lastnosti$ in $Kandidat$.

Če je $SK = \{K1, K2..Kn\}$ množica skupine kriterijev, ki opisujejo izpeljani kriterij, pripadajoči razred R izpeljanega kriterija določimo po naslednjem postopku:

- Če $\exists i \in 1..l_{min}: \forall j \in 1..i: \left(\overrightarrow{vp}_{K1,RA}[j] = \overrightarrow{vp}_{K2,RA}[j] = \cdots = \overrightarrow{vp}_{Kn,RA}[j]\right)$, potem identificiramo začetno skupno verigo povezanosti. l_{min} je število, ki je enako številu trojic v najkrajši izmed verig povezanosti kriterijev $K1$, $K2$ in Kn. Začetna skupna veriga povezanosti je vektor $\overrightarrow{vp}_{SK,RA}$, za katerega velja:

$$\overrightarrow{vp}_{SK,RA} = (\overrightarrow{vp}_{K1,RA}[1], \overrightarrow{vp}_{K1,RA}[2] \ldots \overrightarrow{vp}_{K1,RA}[i]).$$

(F14)

  V tem primeru je lahko razred izbranega kriterija, katerikoli razred v skupni verigi povezanosti. Z namenom nižje kompleksnosti odločitvenega modela in možnosti

opredeljevanja manj kompleksnih odločitvenih pravil v naslednji fazi, priporočamo izbiro naslednjega razreda:

- če je trojica $\overrightarrow{vp}_{SK,RA}[i]$ podatkovna trojica, naj bo izbran razred $\overrightarrow{vp}_{SK,RA}[i][1]$;
- sicer, če je $\overrightarrow{vp}_{SK,RA}[i]$ objektna trojica naj bo izbran razred R, za katerega velja:

$$R = \begin{cases} \overrightarrow{vp}_{SK,RA}[i][1], & \text{če } velja\ R4\left(\overrightarrow{vp}_{K,RA}, i\right)\ in\ i > 1 \\ \overrightarrow{vp}_{SK,RA}[i][3], & \text{če } velja\ R1\left(\overrightarrow{vp}_{K,RA}, i\right)\ in\ i > 1 \\ \overrightarrow{vp}_{SK,RA}[i][1], & \text{če } velja\ \overrightarrow{vp}_{SK,RA}[i][3] = RA\ in\ i = 1 \\ \overrightarrow{vp}_{SK,RA}[i][3], & \text{če } velja\ \overrightarrow{vp}_{SK,RA}[i][1] = RA\ in\ i = 1 \end{cases} \quad \text{(F15)}$$

- V nasprotnem primeru, torej če $\nexists i \in 1..l: \forall j \in 1..i: \left(\overrightarrow{vp}_{K1,RA}[j] = \overrightarrow{vp}_{K2,RA}[j] = \cdots = \overrightarrow{vp}_{Kn,RA}[j]\right)$, je razred izpeljanega kriterija razred RA. Namreč, četudi ni začetnih skupnih delov verig povezanosti, so vse trojice povezane z razredom alternativ, ki je v tem primeru razred izpeljanega kriterija.

Primer izpeljanega kriterija iz odločitvenega drevesa je kriterij $Izobrazba$. Pripadajoča skupina kriterijev je sestavljena iz dveh kriterijev, in sicer $Področje$ in $Stopnja\ izobrazbe$, ki sta v ontologiji zajeta z verigama povezanosti:

$$\overrightarrow{vp}_{Področje,Kandidat} = ((Izobrazba, pripadaKandidatu, Kandidat), (Izobrazba, sPodrocja, Področje)) \quad \text{(F16)}$$

$$\overrightarrow{vp}_{Stopnja\ izobrazbe,Kandidat} = ((Izobrazba, pripadaKandidatu, Kandidat), (Izobrazba, jeStopnje, StopnjaIzobrazbe)). \quad \text{(F17)}$$

Začetna skupna veriga povezanosti je sestavljena iz ene same trojice:

$$\overrightarrow{vp}_{\{Področje,Stopnja\ izobrazbe\},Kandidat} = ((Izobrazba, pripadaKandidatu, Kandidat)). \quad \text{(F18)}$$

Izbrani razred izpeljanega kriterija je razred $\overrightarrow{vp}_{SK,RA}[i][1] = Izobrazba$, saj je dolžina začetne skupne verige povezanosti enaka 1 in $\overrightarrow{vp}_{SK,RA}[i][3] = RA = Kandidat$.

Drug primer izpeljanega kriterija iz odločitvenega drevesa je kriterij $Delovne\ izkušnje$. Pripadajočo skupina kriterijev v tem primeru sestavljajo kriteriji $Skupno\ število\ let$, $Področje\ delovnih\ izkušenj$ in $Vrste\ aktivnosti$. Pri tem se $Skupno\ število\ let$ nanaša na skupno število let vseh delovnih izkušenj in ne na število let delovnih izkušenj posamezne zaposlitve. Kriterijem v ontologiji ustrezajo naslednje verige povezanosti:

- $\overrightarrow{vp}_{Skupno\ število\ let,Kandidat} = ((Kandidat, steviloLetDelovnihIzkusenj, float))$, (F19)
- $\overrightarrow{vp}_{Podrocje\ del.izk.,Kandidat} = ((Kandidat, jeImelZaposlitev, Zaposlitev), (Zaposlitev, sPodrocja, Podrocje))$, (F20)
- $\overrightarrow{vp}_{\text{Vrste aktivnosti},Kandidat} = ((Kandidat, jeImelZaposlitev, Zaposlitev), (Zaposlitev, zahtevaVrstoAktivnosti, VrstaAktivnosti))$. (F21)

V tem primeru je začetna skupna veriga povezanosti prazna, zato je razred izpeljanega kriterija razred $Kandidat$.

6.3.2 Izdelava podrazredov kriterijev

V tem koraku je potrebno določiti možne (kvalitativne) vrednosti izpeljanih kriterijev (merska lestvica vrednosti). Nato je potrebno izdelati skupino podrazredov za vsak razred izpeljanega kriterija. Če želimo manjšo kompleksnost odločitvenih pravil, ki se določajo v naslednji fazi, določimo vrednosti in izdelamo podrazrede tudi za razrede osnovnih kriterijev.

Za vsako možno vrednost kriterija izdelamo en podrazred razreda kriterija. Vsak izmed podrazredov je namenjen primerkom, ki zavzemajo eno izmed možnih vrednosti pripadajočega kriterija. V primeru, da določen razred predstavlja več kot en kriterij, je zanj potrebno izdelati več skupin podrazredov, in sicer za vsak kriterij in njegove vrednosti.

Vsakemu razredu kriterija je potrebno določiti aksiom pokritja za vsako njegovo skupino podrazredov kriterijev. Če je RK razred kriterija in so $R_1, R_2 \dots R_n$ njegovi podrazredi za vrednosti kriterija, je pripadajoči aksiom pokritja:

$$R_1 \sqcup R_2 \sqcup \dots \sqcup R_n \equiv RK.$$

(F22)

Poleg tega je potrebno podrazrede kriterija med seboj določiti kot disjunktne:

$$\forall R_k, R_l : R_k \sqcap R_l \equiv \bot, kjer\ k, l \in 1..n, k \neq l.$$

(F23)

Če vzamemo kot primer izpeljanega kriterija kriterij $Izobrazba$ in pripadajoči razred v ontologiji z enakim imenom, lahko opredelimo naslednje kvalitativne vrednosti kriterija: zelo primerna izobrazba, primerna izobrazba, neprimerna izobrazba. Na podlagi teh vrednosti v ontologiji izdelamo tri pripadajoče podrazrede razreda $Izobrazba$, na primer: $ZeloPrimernaIzobrazba$, $PrimernaIzobrazba$, $NeprimernaIzobrazba$. Aksiom pokritja je naslednji:

$$ZeloPrimernaIzobrazba \sqcup PrimernaIzobrazba \sqcup NeprimernaIzobrazba \equiv Izobrazba.$$

(F24)

Poleg tega je potrebno določiti vse tri podrazrede med seboj kot disjunktne.

6.4 Opredelitev odločitvenih pravil

V ontologiji so odločitvena pravila predstavljena v obliki potrebnih in zadostnih pogojev ali SWRL pravil. Izdelajo se za vse podrazrede osnovnih in izpeljanih kriterijev, ki smo jih izdelali v prejšnji fazi na podlagi opredeljenih možnih vrednosti kriterija. Za vsak podrazred razredov izpeljanega kriterija morajo biti opredeljeni bodisi OWL potrebni in zadostni pogoji ali SWRL pravila, ki natančno določajo pripadnost.

Kriteriji, ki spadajo v isto skupino izpeljanega kriterija, nastopajo pri opisovanju razreda izpeljanega kriterija. Če primerek izpolnjuje potrebne in zadostne pogoje razreda, potem razredu pripada. Pri tem je priporočljiva uporaba koncepta komplementarnega razrede z izrazom $complementOf$, kjer je to mogoče, saj prispeva k nižjemu številu pogojev. Praviloma uporabljamo SWRL pravila tam, kjer želimo ekspresivnost, ki je jezik OWL ne omogoča, na primer za primerjave vrednosti podatkovnih lastnostih. Odločitvena pravila zapisana v jeziku SWRL morajo natančno opredeljevati pogoje, ki določajo, kdaj primerek pripada določenemu podrazredu razreda izpeljanega kriterija. V določenih primerih je zapis pravila v jeziku OWL lahko enakovreden zapisu v jeziku SWRL. Na primer potreben in zadosten pogoj za pripadnost razredu R: $A \sqcap \exists L.B$, je enakovreden SWRL zapisu: $A(?x) \wedge L(?x, ?y) \wedge B(?y) \rightarrow R(?x)$.

Podajmo primer potrebnega in zadostnega pogoja za razred $NeprimernaIzobrazba$:

$$NeprimernaIzobrazba \equiv$$
$$Izobrazba \sqcap \neg PrimernaIzobrazba \sqcap \neg ZeloPrimernaIzobrazba.$$

(F25)

Naj bo eden izmed podrazredov razreda $Kandidat$ za izpeljani kriterij $Delovne\ izkušnje$ razred $KandidatZManjPrimernimiDelovnimiIzkusnjami$. Primer odločitvenega pravila v SWRL jeziku je:

$$Kandidat(?x) \wedge steviloLetDelovnihIzkusenj(?x, ?stLetDI) \wedge$$
$$lessThan(?stLetDI, "8"\text{^}\text{^}integer) \rightarrow$$
$$KandidatZManjPrimernimiDelovnimiIzkusnjami(?x).$$

(F26)

6.5 Opis alternativ

Ker je ena izmed predpostavk, da odločitveni model gradimo na obstoječi ontologiji, so primerki razreda alternativ, med katerimi izbiramo najboljšega, že podani v ontologiji. Opis alternativ v večkriterijskem odločanju tako ni potreben.

V primeru so alternative primerki razreda $Kandidat$.

V nasprotnem primeru, če primerki še niso v ontologiji oziroma še niso primerno opisani, da bi lahko sprejeli odločitev o najboljšem, jih je potrebno dodati oziroma opisati. V skladu z namenom avtomatizacije uporabniškega opravila, ki temelji na odločanju, se alternative lahko dodajo avtomatsko med izvajanjem uporabniškega opravila, in sicer na podlagi procesnih podatkov, ki se preslikajo v ontologijo. To podrobneje opisuje poglavje 8 Organizacijska ontologija in povezanost z vsebinskimi informacijami poslovnega procesa. Če procesni podatki ne zadoščajo za preslikavo, je potrebno dodati primerke v skladu z dano metodologijo obvladovanja organizacijske ontologije (glej poglavje 8).

6.6 Vrednotenje alternativ

Vrednotenje alternativ je izvedeno s pomočjo stroja za sklepanje, ki na podlagi odločitvenih pravil razvrsti primerke v pripadajoče razrede. Na ta način so primerki razvrščeni tudi v podrazred razreda alternativ, ki predstavlja najbolj zaželene primerke, in v druge podrazrede razreda alternativ, ki predstavljajo slabše ovrednotene alternative.

Naj bodo podrazredi razreda alternativ $Kandidat$ za izpeljani kriterij $Kandidat$[2]: $NajboljsiKandidat$, $ZeloDoberKandidat$, $PrimerenKandidat$ in $NeprimerenKandidat$. Če so bila vsa odločitvena pravila v ontologiji opredeljena v fazi opredelitve odločitvenih pravil, bo po uporabi stroja za sklepanje razred $NajboljsiKandidat$ vseboval najboljše kandidate.

6.7 Analiza alternativ

Ker gre za gradnjo odločitvenega modela, ki je podlaga za avtomatsko in polavtomatsko odločanje v uporabniških opravilih, ki so del poslovnega procesa, se aktivnosti analize alternativ nanašajo zgolj na polavtomatizirana uporabniška opravila. Ker je odločanje postavljeno v širši

[2] *Opomba*: Kot je bilo ugotovljeno v predhodnih fazah odločitvenega postopka, razred Kandidat predstavlja tudi razred drugih izpeljanih kriterijev in ima tako tudi druge skupine podrazredov, na primer za kriterij Delovne izkušnje.

kontekst izvajanja poslovnih procesov, se aktivnosti analize alternativ, ki se pojavljajo v klasičnem večkriterijskem odločanju, kot so interaktivno pregledovanje rezultatov, pregledovanje z grafikoni in analiza tipa kaj-če, lahko dopolnijo s proaktivno analizo vpliva odločitve na izvajanje poslovnega procesa. To je seveda mogoče le, če ima lastnik uporabniškega opravila, ustrezne pravice za vpogled v izvajanje poslovnega procesa. Gre za zelo pomembno pridobitev, saj se iz izolirane obravnave posamezne odločitve dvignemo na raven obravnave odločitve v poslovnem procesu, v katerem se odločitev sprejema, in celo na raven celotne ravni poslovnih procesov poslovnega sistema. Analiziramo lahko namreč vpliv odločitve na izvajanje in na rezultate poslovnega procesa ter posledično vpliv na druge poslovne procese, ki se z danim poslovnim procesom povezujejo.

7 Avtomatizacija sodelovalnih opravil

7.1 Cilji

Kot pri vseh sestavljenih opravilih se tudi avtomatizacija sodelovalnih opravil nanaša na avtomatizacijo podopravil in kompozicijskih funkcij. Za izvedbo sodelovalnih opravil agent posrednik opravilo pretvori v zaporedje podopravil. Udeleženci sodelovanja so predstavljeni z agenti za izvajanje elementarnih uporabniških opravil, ki vzdržujejo zgodovino sodelovanja. Zaporedja podopravil so sestavljena iz dveh tipov podopravil, in sicer komunikacijskih aktov in opravil, namenjenih pripravi sporočila za komunikacijo. Lahko pomenijo tudi izvedbo avtomatiziranega opravila. V tem primeru agent proži ustrezno storitev, ki to opravilo avtomatizira.

Opravilo, namenjeno pripravi sporočila za komunikacijo, je lahko uporabniško opravilo. Lahko je avtomatizirano ali ne, pri čemer se njegova avtomatizacija obravnava na enak način kot pri vseh uporabniških opravilih. Lahko je elementarno uporabniško opravilo ali sodelovalno opravilo. Komunikacijski akti so elementarna opravila, v katerih pride do izmenjave sporočil med različnimi udeleženci sodelovanja. V skladu z ugotovitvami poglavja 4.3 Avtomatizacija elementarnih uporabniških opravil, ne spadajo med uporabniška opravila, ki so avtomatizirana z MAPP. Kljub temu je cilj komunikacijskega akta, kadar so komunikacijski akti del sodelovalnih opravil, ne le dostaviti sporočilo prejemniku, ampak s sporočilom drugemu udeležencu (oziroma drugim udeležencem) podati vhodne podatke potrebne za izvedbo naslednjega opravila. Če je le-to opravilo avtomatizirano, za izvedbo sodelovanja v danem opravilu ni potrebe, da bi oseba, ki je lastnik opravila, sporočilo prejela. Dovolj je, če sporočilo prejme agent, ki opravilo izvede v njenem imenu. V primeru, da to opravilo ni avtomatizirano, je posredovano osebi, ki je lastnik opravila, skupaj z vhodnimi podatki, katerih del je tudi sporočilo prejeto v predhodnem komunikacijskem aktu. Iz tega sledi, da v nobenem izmed primerov, osebi, ki je prejemnik sporočila komunikacijskega akta, sporočila ni potrebno eksplicitno posredovati v okviru samega komunikacijskega akta. Zaradi tega so komunikacijski akti, ki so podopravila sodelovalnega

opravila, avtomatizirani kot komunikacija med agenti[3]. Če lastnik opravila kljub temu želi prejemati vsa sporočila, so mu sporočila sodelovalnega opravila posredovana in opravila priprave sporočil izvaja sam.

Kompozicijske funkcije so v sodelovalnih opravilih identične funkcije, zato je poudarek predvsem na zaporedju izvajanja podopravil. Posebnost sodelovalnih opravil je v tem, da se o naslednjem podopravilu dinamično odloča trenutni lastnik podopravila; na primer na podlagi predhodno prejetih sporočil drugih udeležencev sodelovanja se odloči, kaj bo naredil v naslednjem koraku. V primeru pogajanj se odloči, ali predlog sprejeme, ga zavrne, poda nov predlog ali odneha od pogajanj. Pravila, ki določajo, kateri komunikacijski akt lahko udeleženec sodelovanja izvede v danem trenutku, imenujemo protokol sodelovanja. Protokol sodelovanja torej določa legalne vektorje podopravil sodelovalnega opravila.

Za avtomatizacijo sodelovalnih opravil je potrebno opredeliti posamezne protokole sodelovanja in omogočiti doseganje skladnosti s protokoli. Za izvedbo sodelovalnih opravil so potrebni različni protokoli. Posamezen protokol sodelovanja se lahko nanaša na potek različnih sodelovalnih opravil, zato je pomembna lastnost opredelitve protokolov ponovna uporaba v različnih sodelovalnih opravilih. Različni protokoli sodelovanja lahko temeljijo na enakih osnovnih protokolih sodelovanja, ki jih razširjajo z dodatnimi koncepti in pravili, ki so specifične za vsakega izmed njih. Poleg ponovne uporabe protokolov sodelovanja v uporabniških opravilih naj bo mogoča tudi ponovna uporaba in razširjanje protokolov pri opredeljevanju novih protokolov.

Če se naj sodelovalno opravilo izvede v skladu s protokolom sodelovanja, morajo znati tudi agenti, ki delujejo namesto lastnikov opravil, ravnati v skladu s protokoli. Ker so tako posamezniki kot agenti, ki jih predstavljajo, avtonomne entitete, je ravnanje v skladu s protokolom individualna odločitev lastnika opravila, ki prevzame odgovornost v primeru kršenja protokola.

Če bi bili protokoli sodelovanja implementirani kot del agentov, bi dodajanje novega protokola ali spreminjanje obstoječega protokola zahtevalo kompleksne spremembe v večagentnem sistemu

[3] *Opomba*: Če niso del sodelovalnega opravila, so delegirani osebi, ki je lastnik opravila. V tem primeru sporočilo prenese prejemniku sporočila lastnik opravila.

v SIUO. Ker je eden izmed ciljev storitveno usmerjene arhitekture omogočiti poslovnim sistemom hitro odzivnost na spremembe in agilnost, to ne bi bilo sprejemljivo. Poslovni protokoli, ki narekujejo oblike sodelovanja v poslovnem sistemu, so namreč del poslovne ravni in jih je na tej ravni potrebno tudi modelirati. Na eni strani želimo, da lahko protokole sodelovanja modelirajo poslovni uporabniki, in na drugi strani, da so protokoli sodelovanja razumljivi tudi agentom v večagentnem sistemu v SIUO, in sicer tako, da bodo s pomočjo opredelitve protokolov sodelovanja lahko avtomatizirali sodelovalna opravila.

Cilje opredelitve protokolov sodelovanja in zasnove večagentnega sistema, ki omogočajo avtomatizacijo sodelovalnih opravil, lahko povzamemo v naslednjih točkah:

- *Protokoli naj bodo zajeti na način, ki ga razumejo agenti večagentnega sistema v SIUO. Agenti naj bodo zasnovani tako, da ravnajo v skladu s protokolom, če je to v interesu lastnika opravila, v imenu katerega delujejo.*
- *Protokole sodelovanja v poslovnem sistemu naj bo mogoče modelirati na poslovni ravni s strani poslovnih uporabnikov.*
- *Sprememba protokola sodelovanja na poslovni ravni naj se v čim višji meri avtomatsko odraža v implementaciji protokola. Sprememba protokola naj ne zahteva spremembe implementacije obnašanja agentov.*
- *Posamezen protokol sodelovanja naj bo opredeljen tako, da bo mogoča njegova ponovna uporaba v različnih uporabniških opravilih in nadgradnja v različice protokola.*

7.2 Protokoli sodelovanja

Obstajajo različni pristopi k modeliranju protokolov. Nekateri temeljijo na omejevanju možnih zaporedij komunikacijskih aktov z uporabo diagramov prehajanj, na primer [16] in [27], medtem ko drugi temeljijo na logični predstavitvi stanj, ki jih lahko izpeljemo na podlagi akcij izvedenih v predhodnih stanjih, na primer [69] in [110]. V pričujočem delu pri podajanju rešitve, ki bo izpolnjevala zadane cilje, izhajamo iz obeh pristopov in uporabljamo omejitve zaporedij

komunikacijskih aktov za opredelitev protokolov sodelovanja na poslovni ravni in predstavitve stanj za izvajanje protokola v večagentnem sistemu.

Protokol p določa:

1 vektor omejitev $\overrightarrow{c_p}$:

$$\overrightarrow{c_p} = \left(c_{1p}, c_{2p} \dots c_{rp}\right), \forall c_{ip}: T \to R_{c_{ip}},$$

(F27)

kjer je T domena, ki vsebuje opravila, r število funkcij omejitev in $R_{c_{ip}}$ zaloga vrednosti funkcije omejitve c_{ip}; in

2 množico TCP_p:

$$TCP_p = \{(\overrightarrow{pr}_{11}, \overrightarrow{pr}_{12} \dots \overrightarrow{pr}_{1r}), (\overrightarrow{pr}_{21}, \overrightarrow{pr}_{22} \dots \overrightarrow{pr}_{2r}) \dots (\overrightarrow{pr}_{s1}, \overrightarrow{pr}_{s2} \dots \overrightarrow{pr}_{sr})\}, \forall i \in 1..s, \forall j \in 1..r: \overrightarrow{pr}_{ij} \in R_{c_{jp}}^{d_{\overrightarrow{pr}_{ij}}}, s \in N,$$

(F28)

kjer je r število funkcij omejitev in $R_{c_{jp}}^{d_{\overrightarrow{pr}_{ij}}}$ množica vektorjev s številom elementov $d_{\overrightarrow{pr}_{ij}}$, v kateri vsak element vektorja pripada množici $R_{c_{jp}}$ zaloge vrednosti preslikave omejitve c_{jp}.

Protokol sodelovanja v ožjem smislu se ukvarja zgolj z komunikacijskimi akti sodelovalnega opravila. Protokol sodelovanja v širšem smislu lahko določa tudi druga opravila, na primer izvedba transakcije. Vsak komunikacijski akt vključuje pošiljatelja in prejemnike; pošiljatelj je eden, medtem ko je prejemnikov lahko več. Pri vsaki (uspešni[4]) komunikaciji sta prisotna vsaj dva komunikacijska akta: eno pošiljanje in vsaj eno prejemanje sporočila. Vsak tip komunikacijskega akta ima torej komplementaren tip, katerega lastnik je udeleženec nasprotne strani komunikacije.

Naj bo množica TO množica vseh tipov opravil in naj bo TKA:

[4] Če je komunikacija neuspešna, je lahko izvedeno na primer zgolj pošiljanje, medtem ko zaradi napake v komunikaciji prejemnik/-i sporočila ne prejme/-jo.

$$TKA = \{t_1, t_2 \dots t_{n_{PT}}\} \subset TO$$

(F29)

množica tipov komunikacijskih aktov. Poleg tega naj bo $\overrightarrow{st}_{ct} = (st_{1ct}, st_{2ct} \dots st_{nct})$ vektor podopravil sestavljenega opravila ct, v skladu z opredelitivjo (F2) in naj bo $tt: T \to TO$ preslikava opravila v njegov tip opravila. Potem lahko vsakemu sestavljenemu opravilu ct določimo pripadajoči vektor komunikacijskih opravil $\overrightarrow{pst}_{ct} = (pst_{1ct}, pst_{2ct} \dots pst_{mct}), m < n$, n je število vseh podopravil opravila ct. Za $\overrightarrow{pst}_{ct}$ velja:

$$\exists i_1 \in 1..n: st_{i_1ct} = pst_{1ct} \wedge tt(st_{i_1ct}) \in TKA \wedge \forall i_2 \in 1..(i_1 - 1): tt(st_{i_2ct}) \notin TKA$$

(F30a)

in

$$\forall i \in 2..m: \exists j \in i..n: st_{jct} = pst_{ict} \wedge \exists k \in 1..(j-1): st_{kct} = pst_{(i-1)ct} \wedge tt(st_{jct}) \in TKA \wedge tt(st_{kct}) \in TKA \wedge \nexists o: k < o < j: tt(st_{oct}) \in TKA.$$

(F30b)

Označimo s $f_{arg}(\vec{v})$ vektor, ki ga dobimo, če vsak element vektorja $\vec{v} = (v_1, v_1 \dots v_{n_v})$ preslikamo s funkcijo f:

$$f_{arg}(\vec{v}) = (f(v_1), f(v_2) \dots f(v_{n_v})).$$

(F31)

Sodelovalno opravilo ct je skladno s protokolom p, če vektor komunikacijskih opravil sodelovalnega opravila ct izpolnjuje naslednji pogoj:

$$(c_{1p}{}_{arg}(\overrightarrow{pst}_{ct}), c_{2p}{}_{arg}(\overrightarrow{pst}_{ct}) \dots c_{rp}{}_{arg}(\overrightarrow{pst}_{ct})) \in TCP_p.$$

(F32)

Pri tem gre za skladnost s protokolom v ožjem smislu. Če gre za protokol v širšem smislu, ki ne omejuje zgolj komunikacijskih aktov, množico TKA ustrezno razširimo s tipi opravil, ki jih protokol v širšem smislu obravnava.

Različni protokoli omejujejo različne lastnosti podopravil sodelovalnega opravila. V danem kontekstu so minimalne omejitve naslednje:

- vloga lastnika komunikacijskega akta,
- vloge prejemnikov sporočila,
- dovoljeni tipi komunikacijskih aktov.

Označimo omejitev vloge lastnika opravila z c_{1p}. Zaloga vrednosti omejitve je množica vlog udeležencev sodelovanja $R_{c_{1p}}$. Vektor podopravil sodelovalnega opravila $\overrightarrow{st}_{ct}$ lahko preslikamo v vektorje $\overrightarrow{st}_{cp_i ct}$, ki so sestavljeni zgolj iz podopravil posameznih lastnikov:

$$\forall i \in 1..\left|R_{c_{1p}}\right|: \overrightarrow{st}_{cp_i ct} = \left(st_{i_1 ct}, st_{i_2 ct} \dots st_{i_{n'} ct}\right), \forall j \in 1..n': c_{1p}\left(st_{i_j ct}\right) = cp_i \wedge \nexists k \notin \{i_1, i_2 \dots i_{n'}\}: c_{1p}(st_{kct}) = cp_i, cp_i \in R_{c_{1p}}, \quad i_{l1} < i_{l2} \rightarrow l1 < l2,$$

(F33)

pri čemer so $st_{i_1 ct}, st_{i_2 ct} \dots st_{i_{n'} ct}$ in st_{kct} elementi vektorja podopravil $\overrightarrow{st}_{ct}$.

Na podoben način lahko množico TCP_p prevedemo v množice elementov, ki se nanašajo na posamezne lastnike podopravil:

$$TCP_{p\_cp_i} = \left\{\left(\overrightarrow{pr}_{cp_i 12} \dots \overrightarrow{pr}_{cp_i 1r}\right), \left(\overrightarrow{pr}_{cp_i 22} \dots \overrightarrow{pr}_{cp_i 2r}\right) \dots \left(\overrightarrow{pr}_{cp_i s2} \dots \overrightarrow{pr}_{cp_i sr}\right)\right\}, \forall k \in 1..s, \forall j \in 1..r: \overrightarrow{pr}_{cp_i kj} \in R_{c_{jp}}^{\left|\overrightarrow{pr}_{cp_i kj}\right|},$$

(F34)

kjer so vektorji $\overrightarrow{pr}_{cp_i ij}$ sestavljeni iz omejitev komunikacijskih aktov, ki jih izvede vloga cp_i. To pomeni, da je z zornega kota lastnika opravila z vlogo cp_i sodelovalno opravilo skladno s protokolom p, če izpolnjuje naslednji pogoj:

$$\left(c_{2p_{arg}}(\overrightarrow{pst}_{cp_i ct}), c_{3p_{arg}}(\overrightarrow{pst}_{cp_i ct}) \dots c_{rp_{arg}}(\overrightarrow{pst}_{cp_i ct})\right) \in TCP_{p\_cp_i}.$$

(F35)

Na podlagi povedanega lahko opredelimo funkcijo $lt_{cp_i}(\overrightarrow{ft}) = \overrightarrow{lt}_{ft}$ z naslednjim:

$$\forall j \in 1..d_{\overrightarrow{lt}_{ft}}: \exists z \in 1..\left|TCP_{p\_cp_i}\right|, \forall l \in 2..r: (\forall k \in 1..d_{\overrightarrow{ft}}: c_{lp}(ft_k) = pr_{kcp_i zl}) \wedge c_{lp}(lt_{jft}) = pr_{(d_{\overrightarrow{ft}}+1)cp_i zl},\ lt_{aft} \neq lt_{bft} \rightarrow a \neq b,$$

(F36)

kjer je $d_{\overrightarrow{lt}_{ft}}$ število elementov vektorja opravil $\overrightarrow{lt}_{ft}$, $d_{\overrightarrow{ft}}$ število elementov vektorja opravil $\overrightarrow{ft}$, $ft_k \in T$ k-ti element vektorja $\overrightarrow{ft}$, r število funkcij omejitev, $lt_{jft} \in T$ j-ti element vektorja $\overrightarrow{lt}_{ft}$ in $pr_{kcp_i zl} \in \overrightarrow{pr}_{cp_i zl}$ k-ti element vektorja $\overrightarrow{pr}_{cp_i zl}$. Protokol torej za vsakega udeleženca določa funkcijo lt_{cp_i}, ki na podlagi vektorja $\overrightarrow{ft}$, ki vsebuje komunikacijske akte, ki so jih že izvedli med sodelovanjem, vrne vektor $\overrightarrow{lt}_{ft}$, ki vsebuje vse dovoljene komunikacijske akte, ki jih lahko udeleženec izvede v naslednjem komunikacijskem aktu sodelovalnega opravila, če želi delovati v skladu s protokolom. Imenujmo jo *funkcija veljavnih komunikacijskih aktov*.

Kot rešitev, ki bo izpolnjevala zadane cilje, predlagamo koncept knjižnice protokolov, ogrodje modeliranja protokolov sodelovanja na poslovni ravni in preslikavo med opredelitvami protokolov na poslovni ravni in knjižnico protokolov. Pomembna prednost predlaganega pristopa k implementaciji protokolov sodelovanja je, da ne predpisuje uporabe specifičnega jezika komunikacije agentov ACL ali specifične specifikacije protokolov. Zasnovan je dovolj generično, da omogoča implementacijo poljubnih protokolov sodelovanja s poljubnim jezikom ACL.

V nadaljnjih poglavjih sta najprej podana primera, ki jih bomo uporabljali za ponazoritev razlage, nato je opisana knjižnica protokolov in zatem modeliranje protokolov na poslovni ravni skupaj s preslikavo med poslovno ravnjo in ravnjo organizacijske ontologije protokolov sodelovanja.

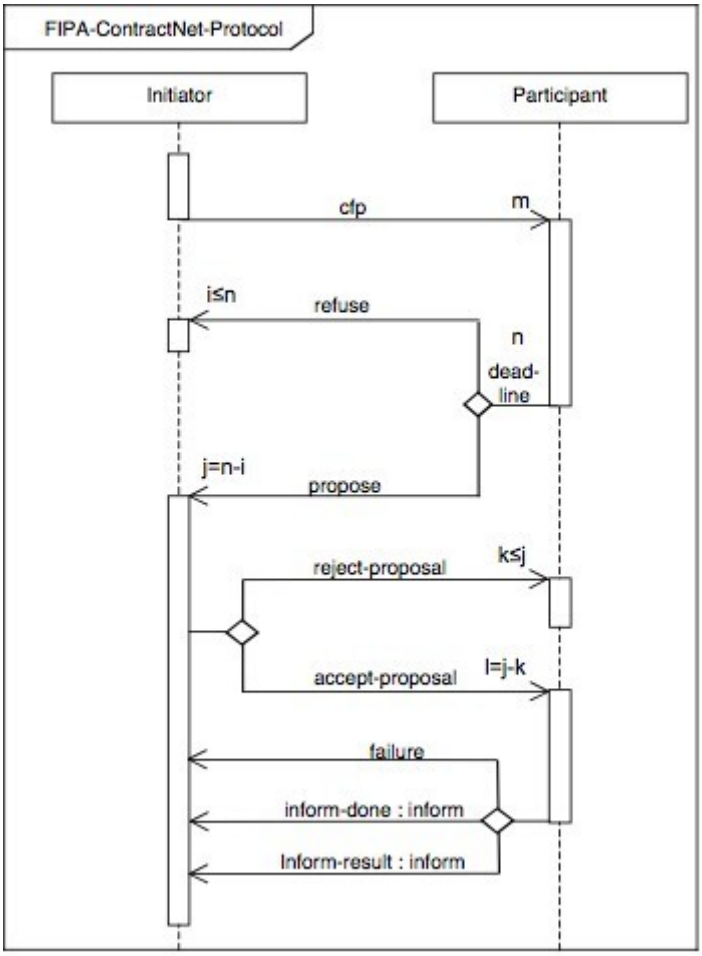
7.3 Opis primerov

Opredeljevanje protokolov sodelovanja bomo ponazorili na podlagi dveh primerov protokolov, ki temeljita na specifikacijah FIPA: protokol pogodbene mreže *(FIPA Contract Net Interaction Protocol*) [25] in protokol angleških avkcij (*FIPA English Auction Interaction Protocol*) [26]. V tem poglavju bomo opisali potek obeh primerov protokolov.

Ker primere podajamo za lažjo razlago opredeljevanja protokolov v organizacijski ontologiji in poslovni ravni, izjemnih primerov ne bomo posebej obravnavali.

7.3.1 Protokol pogodbene mreže (Contract Net)

V FIPA protokolu pogodbene mreže določen agent prevzame vlogo iniciatorja (ali upravljavca), ki želi najti enega ali več agentov, ki so najprimernejši za izvedbo določenega opravila. Najprimernejše agente lahko izbere na podlagi različnih kriterijev, kot so cena, čas, v katerem lahko opravilo končajo, enakovredna porazdelitev opravil itd.



Slika 9: FIPA protokol interakcij *Contract Net*

Potek protokola pogodbene mreže ponazarja Slika 9 v notaciji UML, razširjeni za področje večagentnih sistemov [58]. Iniciator protokola pošlje m klicev za predloge (*call for proposals, CFP*), ki določajo opravilo in pogoje, ki jih iniciator postavlja za izvedbo opravila. Udeleženci, ki prejmejo CFP, so potencialni pogodbeniki. Udeleženci lahko odgovorijo s predlogom ali odklonitvijo v določenem časovnem roku.

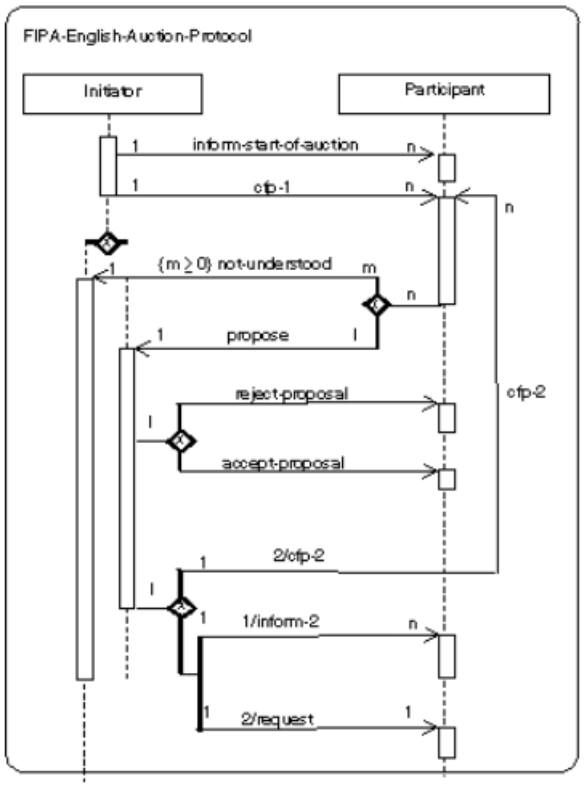
Po izteku časovnega roka iniciator oceni prejete predloge udeležencev in izbere agente, ki bodo izvajali opravilo. Izbranih je lahko nič, eden ali več agentov. Agentom, ki so izbrani za izvedbo opravila, iniciator pošlje sporočilo o sprejemu njihovega predloga. Udeležencem, ki so poslali predlog in niso bili izbrani, pošlje sporočilo o zavrnitvi predloga. Ko izbrani udeleženec oziroma udeleženci zaključi/-jo opravilo, iniciatorju pošlje/-jo obvestilo o izvedenem opravilu (*inform-*

done) ali obvestilo o rezultatih (*inform-result*). Če udeleženec ne uspe izvesti opravila, pošlje sporočilo o neuspehu (*failure*).

7.3.2 Protokol angleških avkcij

V FIPA protokolu angleških avkcij agent iniciator (dražbar) skuša najti tržno ceno blaga. Avkcijo začne tako, da najprej predlaga ceno, za katero ocenjuje, da je nižja od tržne vrednosti. Nato ceno postopoma zvišuje glede na ponudbe. Vsakič, ko objavi ceno, dražbar počaka, da vidi, če je kdo izmed kupcev pripravljen plačati predlagano ceno. Takoj ko eden izmed kupcev odgovori, da je ceno pripravljen sprejeti, dražbar objavi višjo ceno blaga. Avkcija se nadaljuje dokler ni več kupca, ki bi bil pripravljen blago kupiti po objavljeni ceni. Takrat se avkcija konča. Če zadnja sprejeta cena presega rezervirano ceno, ki je znana zgolj dražbeniku, je blago prodano kupcu, ki jo je sprejel po ponujeni ceni oziroma ceni, ki je višja od ponujene. V nasprotnem primeru, če je najvišja sprejeta cena manjša od rezervirane cene dražbenika, potem blago ni prodano.

Slika 10 ponazarja FIPA protokol angleških avkcij. Protokol se začne ko agent inciator pošlje sporočilo o začetku avkcije vsem udeležencem. Zatem jim pošlje sporočilo CFP, ki določa blago in izklicno ceno. Drugi udeleženci avkcije (potencialni kupci) nato lahko pošljejo predlog cene, ki je lahko enaka ali višja izklicni ceni. Za razliko od avkcij v resničnem svetu, ki na primer potekajo v enem prostoru, agenti ne vedo, kdaj je nekdo ceno sprejel oziroma ali je bil sprejet njihov predlog. Zato določimo časovni rok, v katerem inciator sprejema odgovore. Zatem iniciator na vsak predlog pošlje odgovor, v katerem ga sprejme ali zavrne. Če ne prejme predlogov, se avkcija konča. Če je končna cena višja od rezervirane cene, iniciator kupcu, katerega predlog je sprejet, pošlje zahtevo po izvedbi transakcije.



Slika 10: FIPA protokol interakcij angleških avkcij

7.4 Knjižnica protokolov sodelovanja

Knjižnica protokolov sodelovanja je del organizacijske ontologije. Omogoča določanje minimalnih omejitev in njihovo razširjanje z drugimi omejitvami. Sestavljena je iz več hierarhično strukturiranih OWL ontologij. Pravimo, da posamezna OWL ontologija uporablja drugo OWL ontologijo, če jo uvaža *(owl:imports*) in razširja. Knjižnica protokolov ima naslednje osnovne karakteristike:

- Funkcije veljavnih tipov komunikacijskih aktov so implementirane s SWRL pravili. Vse ostale omejitve so implementirane s podatkovnimi ali objektnimi lastnostmi razredov. Domene lastnosti so razredi, ki predstavljajo komunikacijske akte.

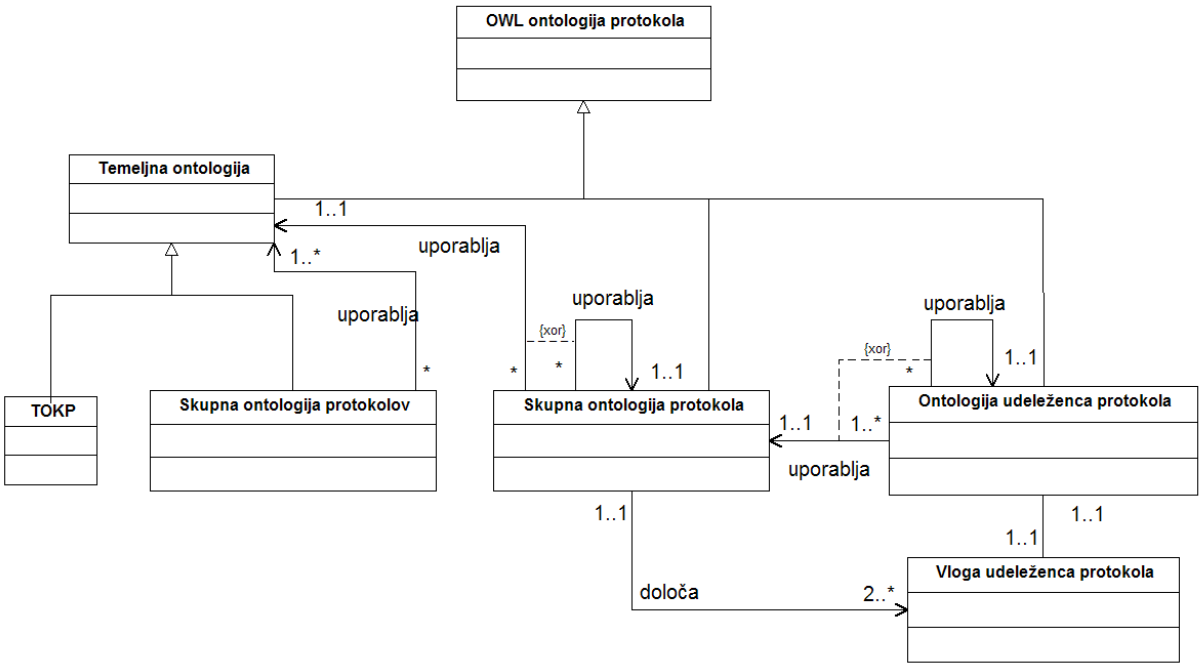
- Temeljni razredi in lastnosti protokolov so zajeti v *temeljni ontologiji knjižnice protokolov* ($TOKP$). Temeljna jo imenujemo zato, ker jo neposredno ali posredno uporabljajo vse druge ontologije knjižnice protokolov.

- Če več protokolov uporablja določene skupne koncepte, ki razširjajo temeljno ontologijo knjižnice protokolov, na primer z dodatnimi omejitvami, opredelimo *skupno ontologijo protokolov*, ki jo lahko uporabljajo. V tem primeru temeljno ontologijo knjižnice protokolov uporabljajo posredno.

- Vsak protokol je zajet s *skupno ontologijo protokola* in množico *ontologij udeležencev*:

    - V *skupni ontologiji protokola* so zajete vloge udeležencev in koncepti omejitev, ki so značilni za posamezen protokol. Skupna ontologija protokola uporablja bodisi temeljno ontologijo knjižnice protokolov bodisi eno izmed skupnih ontologij protokolov.

    - *Ontologije udeležencev* določajo funkcije veljavnih tipov komunikacijskih aktov za posamezno vlogo udeleženca.



Slika 11: Struktura OWL ontologij knjižnice protokolov

Slika 11 ponazarja razmerja med različnimi vrstami OWL ontologij knjižnice protokolov.

7.4.1 Temeljna ontologija knjižnice protokolov

Temeljna ontologija knjižnice protokolov določa temeljne razrede, primerke in lastnosti, ki so potrebni za vzpostavitev skupne osnove opredelitve protokolov. Razrede in lastnosti temeljne ontologije knjižnice protokolov ponazarja Slika 12. Za $TOKP$ so značilni razredi:

- Temeljni razred $Protokol\_Thing$, ki je podrazred razreda $Thing$ $(owl{:}Thing)$ in vsebuje vse primerke konceptov protokola – vsak razred obravnavan v protokolu je podrazred tega temeljnega razreda.

- Temeljni razred komunikacijskih aktov $KomunikacijskiAkt$, ki vsebuje temeljna podrazreda $Posiljanje$ in $Prejem$. Je podrazred razreda $Opravilo$, ki je del skupne ontologije organizacijske ontologije (glej poglavje 8.2 Struktura organizacijske ontologije). S tem je možna uporaba ontologije protokola v širšem kontekstu in določanje drugih opravil (poleg komunikacijskih) za razširitve protokolov, na primer za primere, ko protokol ne določa le komunikacijskih aktov, ampak tudi tipe opravil za pripravo sporočila. Velja:

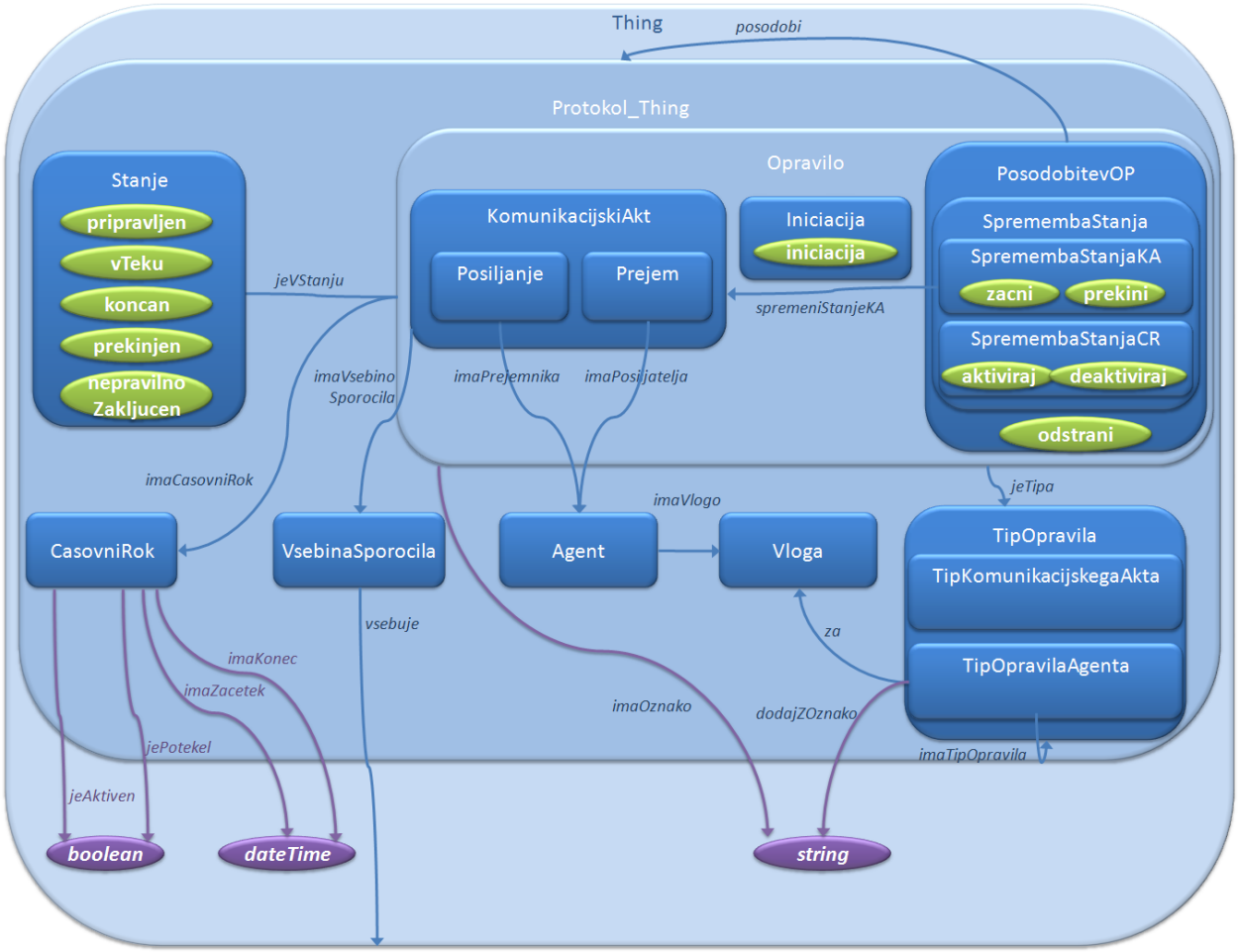
$$Posiljanje \sqcup Prejem \equiv KomunikacijskiAkt;\ Posiljanje \sqcap Prejem \equiv \bot.$$

(F37)

- Temeljni razred agentov $Agent$, ki je namenjen zajemu primerkov agentov, ki zastopajo udeležence sodelovanja.

- Temeljni razred $Vloga$, ki namenjena zajemu primerkov vlog udeležencev sodelovanja.

- Temeljni razred stanj $Stanje$, ki vsebuje osnovne primerke stanj, in sicer $pripravljen$, $vTeku$, $končan$, $prekinjen$ in $nepravilnoZakljucen$. Namenjen je določanju stanja primerkov opravila.



Slika 12: Temeljna ontologija knjižnice protokolov

- Temeljni razred vsebine sporočila $VsebinaSporocila$, v katerem lahko v skupnih ontologijah protokolov ali protokola za posamezne komunikacijske akte podrobneje opredelimo značilne razrede vsebin sporočil.

- Temeljni razred $PosodobitevOP$, ki je podrazred razreda $Opravilo$, ker določa opravila posodobitve ontologije protokola. Namenjen je opredeljevanju primerkov posodobitev ontologije. Vsebuje razred $SpremembaStanja$ s podrazredoma $SpremembaStanjaKA$ in $SpremembaStanjaCR$, ki sta namenjena opredeljevanju funkcij veljavnih tipov komunikacijskih aktov. Namen razredov in njihovih primerkov je podrobneje opisan v poglavju 7.4.3 Ontologija udeleženca protokola.

- Temeljni razred $Iniciacija$ s primerkom $iniciacija$, ki služi za opredelitev začetka sodelovanja. Njun namen je podrobneje opisan v poglavju 7.4.3 Ontologija udeleženca protokola.

- Temeljni razred tipov opravil $TipOpravila$ ima dva podrazreda: $TipKomunikacijskegaAkta$ in $TipOpravilaAgenta$. V razredu $TipKomunikacijskegaAkta$ vsak primerek predstavlja določen tip komunikacijskega akta, medtem ko razred $TipOpravilaAgenta$ vsebuje primerek za vsak možen par agenta z določeno vlogo in tipa opravila. Pri tem sta vrednost lastnosti $imaVlogo$ agenta in vrednost lastnosti za tipa opravila agenta enaki. Razred $TipOpravila$ se uporablja predvsem za določitev nekaterih vrst posodobitev ontologije protokola med njegovim izvajanjem. Tudi ti razredi so podrobneje opisani v poglavju 7.4.3 Ontologija udeleženca protokola.

- Temeljni razred $CasovniRok$, ki je namenjen opredeljevanju časovnih omejitev opravil in časovnih omejitev med sodelovanjem, ki niso vezana na opravila.

TOKP določa tudi temeljne lastnosti namenjene opredeljevanju funkcije veljavnih komunikacijskih aktov protokola, in sicer:

- lastnosti omejitev vlog udeležencev sodelovanja: $imaPosiljatelja$, ki je funkcionalna lastnost in določa pošiljatelja sporočila, ter $imaPrejemnika$, ki določa prejemnike sporočila; pri tem velja:

$$RD(imaPosiljatelja) = \{Posiljanje\},$$

$$RD(imaPrejemnika) = \{Prejem\},$$

$$RZV(imaPosiljatelja) = RZV(imaPrejemnika) = \{Agent\},$$

(F38)

- lastnost $jeVStanju$, ki določa stanje komunikacijskega akta in je funkcionalna lastnost; vrednost lastnosti $vStanju$ primerka $iniciacija$ je $pripravljen$;

- lastnost $jeTipa$, ki določa tip opravila, na primer komunikacijskega akta, in je funkcionalna lastnost;

- lastnost $imaVsebinoSporocila$, ki določa primerek sporočila posameznega komunikacijskega akta, in lastnost $vsebuje$, ki primerku razreda $VsebinaSporocila$ določa primerke, ki jih vsebina sporočila vsebuje; zaloga vrednosti lastnosti $vsebuje$ je $Thing$, kar pomeni, da lahko vsebuje poljuben primerek OWL ontologije; s tem je omogočena uporaba primerkov vsebine v drugih ontologijah organizacijske ontologije, ki jih protokol uporablja oziroma uporabljajo protokol;

- lastnosti $posodobi$, $dodajZOznako$, $spremeniStanjeKA$ in $spremeniStanjeCR$[5], ki določajo izvedbo določenih akcij posodobitev ontologije; lastnosti in njihov namen so podrobneje opisane v poglavju 7.4.3 Ontologija udeleženca protokola;

- lastnost $imaOznako$, ki določa oznako opravila;

- lastnost $imaCasovniRok$, ki določa časovni rok opravila;

- lastnosti $imaZacetek$ in $imaKonec$, ki določata lastnosti primerka časovnega roka; omogočata enostavno razširjanje s podlastnostmi, kot so najzgodnejši začetek in najkasnejši začetek ipd.;

- lastnost $jeAktiven$ je pomožna lastnost, ki določa, če je časovni rok aktiven v danem trenutku in da ga je potrebno spremljati; če primerek časovnega roka ni aktiven (vrednost $false$ lastnosti $jeAktiven$), potem ga ne spremljamo in ostale lastnosti časovnega roka nimajo pomena;

- lastnost $jePotekel$, ki določa, ali je časovni rok že potekel.

Bolj podrobno je temeljna ontologija knjižnice protokolov podana v prilogi (Priloga 4: Temeljna ontologija knjižnice protokolov).

[5] Opomba: Lastnost $spremeniStanjeCR$ zaradi večje preglednosti na sliki ni prikazana (Slika 12). Njena domena je razred $SpremembaStanjaCR$. Njena zaloga vrednosti je razred $CasovniRok$.

7.4.2 Skupna ontologija protokola in skupna ontologija protokolov

Skupne ontologije več protokolov in skupne ontologije posameznih protokolov so strukturirane na enak način, pri čemer se prve nanašajo na skupino protokolov, druge pa na specifičen protokol. V nadaljevanju tega podpoglavja govorimo o obeh vrstah kot o skupnih ontologijah.

V skupni ontologiji so zajete vloge, komunikacijski akti in morebitne druge omejitve, specifične za protokol oziroma za skupino protokolov. Podrobneje opredeljujejo koncepte temeljne ontologije (ali druge skupne ontologije, če jo uporabljajo) in opredeljujejo dodatne koncepte.

Protokoli lahko specializirajo ali dodajajo vloge udeležencev. Posamezna vloga udeležencev je zajeta s primerkom razreda $Vloga$, lahko pa tudi z podrazredom razreda $Agent$, katerega vrednost lastnosti $imaVlogo$ opredeljuje potreben pogoj in s tem vlogo, ki jo igrajo agenti razreda.

Protokole sodelovanja s predlaganim pristopom lahko opredelimo na podlagi poljubnih komunikacijskih aktov, saj niso omejeni na uporabo določenih vrst komunikacijskih aktov. Primerki komunikacijskih aktov so v ontologiji zajeti bodisi kot podrazredi razreda $Pošiljanje$ bodisi podrazredi razreda $Prejem$, in sicer neposredno ali posredno (preko hierarhije podrazredov). Primerki tipov komunikacijskih aktov so zajeti kot primerki razreda $TipKomunikacijskegaAkta$. Za primerke razreda $KomunikacijskiAkt$ velja, da imajo določeno lastnost $jeTipa$, katere vrednost je primerek razreda $TipKomunikacijskegaAkta$.

Predpostavka je, da agenti, ki delujejo na podlagi protokolov, razumejo pomen posameznega tipa komunikacijskega akta in tega kako ga lahko izvedejo. Pri tem se lahko opremo na standardne komunikacijske akte, na primer [23] ali [24], in na predloge doseganja višje ravni razumevanja različnih komunikacijskih aktov med različnimi agenti, na primer na [11] ali [52]. Enako velja, če protokol določa tudi druga opravila, ki niso komunikacijski akti. V primeru, da agent ne razume pomena določenega komunikacijskega akta oziroma drugega opravila, t.j. ga ne zna izvesti, je opravilo delegirano osebi, ki jo agent zastopa. V tem primeru elementarno opravilo sodelovalnega opravila ni avtomatizirano. Kljub temu je avtomatizirana njihova kompozicija, kot jo narekuje protokol sodelovanja.

Skupna ontologija protokola lahko opredeljuje omejitve, ki bodisi razširjajo omejitve drugih skupnih ontologij ali temeljne ontologije knjižnice protokolov bodisi so specifične za protokol oziroma skupino protokolov skupne ontologije. Omejitve so implementirane kot lastnosti med razredom komunikacijskega akta oziroma enim izmed njegovih podrazredov in zalogo vrednosti omejitve. Zaloga vrednosti je lahko bodisi razred bodisi podatkovni tip. V skupni ontologiji protokola so tako opredeljeni razredi vlog udeležencev, lastnosti omejitev in morebitni razredi zalog vrednosti lastnosti omejitev, ki niso opredeljeni v temeljni ontologiji, ki jo skupna ontologija protokola uporablja.

Primer skupne ontologije protokolov

Primer skupne ontologije protokolov je skupna ontologija protokolov za FIPA protokole interakcij. FIPA protokoli interakcij [27] uporabljajo tipe komunikacijskih aktov določene s specifikacijami FIPA *Communicative Act Library Specification* [24].

Skupna ontologija FIPA protokolov interakcij razširja temeljno ontologijo knjižnice protokolov s podrazredi razredov $Posiljanje$ in $Prejem$ glede na specifikacije tipov komunikacijskih aktov FIPA. Skladno s tem določa tudi primerke razreda $TipOpravila$. Pri tem je pomembna lastnost razredov komunikacijskih aktov, da predstavljajo množico primerkov elementarnih opravil. V skladu z opredelitvami komunikacijskega akta kot elementarnega opravila, je potrebno izdelati za vsak tip FIPA komunikacijskega akta dva tipa komunikacijskega akta knjižnice protokolov, in sicer enega za pošiljanje sporočila in drugega za prejem sporočila. Na primer, FIPA komunikacijski akt Obveščanje (*Inform*) se v skupni ontologiji FIPA protokolov knjižnice protokolov predstavi z dvema tipoma, in sicer $PosiljanjeObvestila$, ki je podrazred razreda $Posiljanje$, in $PrejemObvestila$, ki je podrazred razreda $Prejem$. Delitev na dva tipa je potrebna, saj se tipi komunikacijskih aktov v knjižnici protokolov nanašajo na elementarne komunikacijske akte in so namenjeni uporabi s strani posameznih udeležencev. Za posameznega udeleženca mora protokol določati, kdaj naj obvestilo pošlje in kdaj ga lahko pričakuje od drugega udeleženca protokola, če le-ta deluje v skladu s protokolom.

Druga, enakovredna rešitev, ki ne bi zahtevala ločevanja na elementarne akte pošiljanja in prejemanja, bi bila z opredelitvijo razreda $Komunikacija$, ki bi pomenila pošiljanje in prejem določenega sporočila. Takšen pristop bi zahteval drugačno opredelitev SWRL pravil protokola v

ontologijah udeležencev protokola (glej poglavje 7.4.3 Ontologija udeleženca protokola), in sicer bi pri vsakem opravilu, ki ga naj udeleženec po protokolu izvede, morali posebej določiti, kdaj je udeleženec pošiljatelj in kdaj je prejemnik.

Poleg razširjanja temeljne ontologije knjižnice protokolov s tipi FIPA komunikacijskih aktov lahko skupna ontologija FIPA protokolov razširja tudi vloge udeležencev in vsebine sporočil povezane z komunikacijskimi akti, ki jih določa. V vseh FIPA protokolih je prisotna vloga *Iniciator* (*Initiator*), v veliki večini pa tudi vloga *Udeleženec* (*Participant*). Vlogi *Iniciator* in *Udeleženec* lahko tako zajamemo v skupni ontologiji FIPA protokolov, kot primerka razreda $Vloga$: $iniciator$ in $udelezenec$. Poleg tega lahko opredelimo podrazreda $AgentIniciator$ in $AgentUdeleženec$ razreda $Agent$, ki jima omejimo vrednosti lastnosti $imaVlogo$ z ustreznim primerkom vloge.

Posamezni tipi komunikacijskih aktov določajo posebne vsebine sporočila. Tako je na primer vsebina sporočila FIPA komunikacijskega akta Obvesti (*Inform*) tipa Trditev (*Proposition*). Zato skupna ontologija FIPA protokolov podrobneje določa razred $VsebinaSporocila$, tako da opredeljuje podrazrede, ki ustrezajo posameznim vrstam vsebine sporočil. Skladno s tem razrede komunikacijskih aktov podrobneje opredeli z lastnostjo $imaVsebinoSporocila$. Na primer, razredoma $PosiljanjeObvestila$ in $PrejemObvestila$ določimo naslednjo omejitev:

$$\forall imaVsebinoSporocila.Trditev.$$

(F39)

Poleg tega podrobneje opredeljuje tudi vsebine sporočila. Na primer, predlog vsebuje primerek razreda $Opravilo$. Omejitev razreda $Predlog$ je tako:

$$\exists vsebuje.Opravilo.$$

(F40)

Primer: Skupna ontologija protokola pogodbene mreže

Primer skupne ontologije posameznega protokola je skupna ontologija FIPA protokola pogodbene mreže. Ker spada v skupino FIPA protokolov, uporablja skupno ontologijo FIPA protokolov. Ne zahteva nobenih posebnosti glede posebnih tipov komunikacijskih aktov ali vlog

udeležencev. Opredeljuje primerek časovnega roka, ki ga bomo v nadaljnjih primerih poimenovali CR1. Poleg tega opredeljuje tudi disjunktna razreda $ZaželenoOpravilo$ in $NezazelenoOpravilo$ ter disjunktna razreda $SprejetoOpravilo$ in $ZavrnjenoOpravilo$.

Primer: Skupna ontologija protokola angleških avkcij

Podobno kot skupna ontologija protokola pogodbene mreže, tudi skupna ontologija protokola angleških avkcij uporablja skupno ontologijo FIPA protokolov. Tudi protokol angleških avkcij zahteva določene posebnosti, in sicer predvsem natančneje določa vsebino sporočil. Gre torej za omejitve, ki izhajajo iz vsebine sporočil komunikacijskih aktov. Pojavljajo se na dveh mestih, in sicer ko udeleženec pošlje predlog, mora biti ponujena cena enaka ali višja od trenutne izklicne cene, ter ob koncu avkcije, ko je prodaja odvisna od tega, ali je sprejeta cena višja ali enaka od rezervirane cene. Izklicna cena je določena v sporočilu CFP, ki ga iniciator (dražbenik) pošlje udeležencem (potencialnim kupcem).

Poleg tega protokol angleških avkcij obravnava dodatna opravila, in sicer *prodaja* in *izvedba transakcije*. Prodaja je povezana s tistim blagom, ki predstavlja predmet prodaje. Kot protokol pogodbene mreže tudi protokol angleških avkcij zahteva opredelitev primerka časovnega roka. Tudi tega poimenujmo CR1. Če povzamemo, so koncepti, ki jih je potrebno zajeti v ontologiji protokola angleških avkcij, zato da bo le-ta nudila zadostno podlago za opredelitev pravil, naslednji:

- Rezervirana cena:

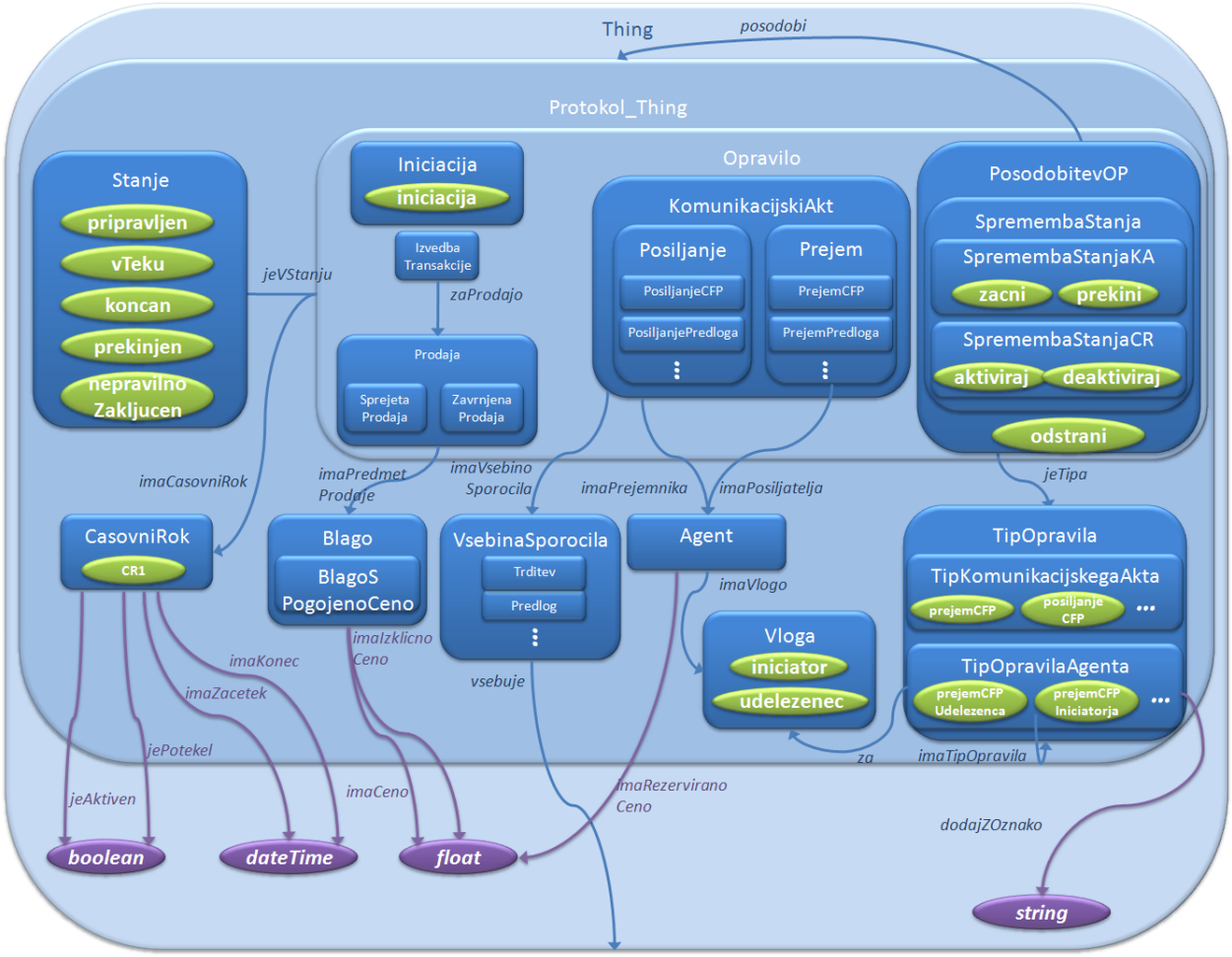
  Rezervirana cena je določena ob začetku protokola in se skozi potek sodelovanja ne spreminja. Rezervirana cena ni prisotna kot vsebina izmenjanih sporočil, temveč je določena v omejitvi zadnjega komunikacijskega akta, ki je naloga iniciatorja.

- Blago, izklicna cena blaga, rezervirana cena:

  Vsebina sporočila komunikacijskega akta CFP in vsebina sporočila komunikacijskega akta predloga (*Propose*) sta sestavljeni iz para elementov: prvi element določa opravilo, ki bi ga naj naslovnik izvedel, medtem ko drugi element postavlja pogoj za to opravilo. Opravilo, ki ga v okviru protokola angleških avkcij vsebujeta, je primerek razreda $Prodaja$, omejen z

določitvijo lastnosti $imaPredmetProdaje$ na zalogo vrednosti $Blago$. Za izpolnjevanje pogoja opredelimo podrazred razreda $Blago$, v katerem so primerki, za katere je cena višja od izklicne cene.

- Poleg tega je potrebno opredeliti ustrezne koncepte za namen opredelitve izvedbe transakcije in pošiljanja obvestila o začetku in koncu avkcije.



Slika 13: Skupna ontologija FIPA protokola angleških avkcij

Za podporo dodatnih konceptov skupno ontologijo FIPA protokolov (ki razširja temeljno ontologijo), v skupni ontologiji protokola angleških avkcij razširimo z dodatnimi razredi, lastnostmi in omejitvami razredov. Možnih implementacij razširitev je lahko več. Primer

implementacije skupne ontologije FIPA protokola angleških avkcij z uporabljeno skupno ontologijo FIPA protokolov ter posredno s temeljno ontologijo knjižnice protokolov ponazarja Slika 13.

Ključni dodatni razredi so naslednji:

- $Blago$, ki je podrazred razreda protokola $Protokol\_P\_Thing$. Ima podatkovno lastnost $imaCeno$, ki določa ceno blaga.

- $Prodaja$, ki je podrazred razreda $Opravilo$ in ima objektno lastnost $imaPredmetProdaje$, ki določa blago, ki je predmet prodaje, ter podrazreda $SprejetaProdaja$ in $ZavrnjenaProdaja$.

- Razredi, ki predstavljajo del ontologije, ki je uvožena iz skupne ontologije FIPA protokolov: podrazredi razredov $Pošiljanje$ in $Prejem$, kot so $PošiljanjeCFP$, $PrejemPredloga$…., in podrazredi razreda $VsebinaSporocila$, kot so $Trditev$, $Predlog$…

- Razred $IzvedbaTransakcije$ in lastnost $zaProdajo$, ki določi na katero prodajo se transakcija nanaša.

Poleg razredov, primerkov in lastnosti, prikazanih na sliki, v skupni ontologiji FIPA protokola angleških avkcij veljata tudi omejitvi razredov CFP in Predlog: ∃vsebuje. Prodaja. Razred $Avkcija$ z lastnostma $seJeZačela$ in $jeZaključena$ zaradi večje preglednosti na sliki ni prikazan (Slika 13). Iz istega razloga niso prikazane vse lastnosti, ki jih lahko sicer vidimo na sliki temeljne ontologije knjižnice protokolov (Slika 12).

7.4.3 Ontologija udeleženca protokola

Ontologije udeležencev določajo funkcije veljavnih komunikacijskih aktov (F36) s pravili SWRL. Praviloma ontologije udeležencev ne razširjajo skupne ontologije protokola z novimi koncepti, temveč zgolj s SWRL pravili. Protokol sodelovanja namreč narekuje veljavne oblike sodelovanja, kar pomeni da koncepti, ki jih obravnava zgolj posamezen udeleženec, in drugi zanje ne vedo, niso del sodelovanja in s tem ne določajo veljavnih oblik sodelovanja. To ne vključuje konceptov na nivoju celotnega protokola, ki so uporabljeni le v ontologiji enega izmed

udeležencev. Namreč, določen koncept je lahko znan le enemu izmed udeležencev, medtem ko je pomemben in določen na nivoju celotnega sodelovalnega opravila. V tem primeru je koncept del skupne ontologije protokola, medtem ko je na ravni primerkov del le ene izmed ontologij udeležencev protokola. Poleg udeleženca protokola, ki ga upošteva v funkcijah veljavnih komunikacijskih aktov, je pomemben za tisto entiteto, ki sodelovalno opravilo proži. Primer je rezervirana cena v FIPA protokolu angleških avkcij.

Ontologije udeležencev lahko razširjajo skupno ontologijo protokola z novimi koncepti zgolj v primerih, kadar lahko udeleženec izvede enega izmed več veljavnih komunikacijskih aktov. Takrat lahko v ontologiji udeleženca protokola opredelimo disjunktne razrede, ki v povezavi z določeno omejitvijo v vseh možnih komunikacijskih aktih, pomenijo, da se lahko izvede le eno izmed veljavnih opravil.

Z uporabo SWRL pravil želimo doseči, da agent (udeleženec protokola) med izvajanjem sodelovalnega opravila na podlagi svoje ontologije, ki vključuje že izvedene komunikacijske akte in SWRL pravila, s proženjem ustrezne operacije SOO pridobi množico tistih komunikacijskih aktov, ki jih lahko izvede v naslednjem koraku.

Poleg tipov komunikacijskih aktov, vlog udeležencev in drugih omejitev komunikacijskih aktov, igrajo pri določanju funkcije veljavnih komunikacijskih aktov ključno vlogo primerki razreda $PosodobitevOP$ ter lastnosti $posodobi$, $spremeniStanjeKA$ in $dodajZOznako$. V nadaljevanju so podani razlogi za njihovo uvedbo in opisan njihov pomen.

SWRL pravila so namenjena monotonemu sklepanju [86], kar pomeni, da lahko na podlagi SWRL pravil stroj za sklepanje pride zgolj do monotonih sklepov. Zaradi tega SWRL pravil ne moremo uporabljati za spreminjanje obstoječih informacij v ontologiji. Na primer, podajmo naslednje pravilo:

Če se je primerek $pka1$ komunikacijskega akta $KA1$ končal, t.j. je v stanju $koncan$, in je primerek $pka2$ drugega komunikacijskega akta $KA2$ v stanju $vTeku$, potem naj se primerek $pka2$ prestavi v stanje $prekinjen$.

V jeziku SWRL se bi pravilo glasilo:

$$KA1(?x1) \wedge jeVStanju(?x1, koncan) \wedge KA2(?x2) \wedge jeVStanju(?x2, vTeku) \rightarrow jeVStanju(?x2, prekinjen).$$

(F41)

Ob izpolnjevanju pogojnega dela pravila, bi rezultat pravila dodal vrednost $prekinjen$ lastnosti $jeVStanju$ vsem primerkom razreda $KA2$, ki bi izpolnjevali pogoj v telesu pravila. To ne bi spremenilo obstoječe vrednosti lastnosti $jeVStanju$ primerka, ampak bi vrednost zgolj dodalo. Ker je lastnost $jeVStanju$ funkcionalna lastnost, bi ontologija s tem postala nekonsistentna. Za spremembo stanja bi bilo potrebno zbrisati obstoječo vrednost lastnosti in dodati novo vrednost.

Samo izvajanje pravil protokola ne more zagotoviti vseh potrebnih informacij. Na primer izvedba komunikacijskega akta sama po sebi ne bo rezultirala v spremembi stanja komunikacijskega akta iz $vTeku$ v $koncan$. Med izvajanjem komunikacijskih aktov protokola ontologija agenta vsebuje tudi ontologijo protokola. Agent je zadolžen za posodabljanje svoje ontologije skladno z informacijami, ki jih pridobiva s svojim delovanjem in ki so rezultat njegovega delovanja. To pomeni, da agent v ontologiji posodablja tudi stanje komunikacijskih aktov, ki jih izvaja. V ta namen temeljna ontologija knjižnice protokola določa razred $PosodobitevOP$ in lastnost $dodajZOznako$. Primerki razreda $PosodobitevOP$ so namenjeni temu, da agentu, ki protokol izvaja, povedo, kateri komunikacijski akt naj začne, prekine oziroma ga odstrani. Če te vrste posodobitev ne zadoščajo, lahko dopolnimo razred $PosodobitevOP$ z novimi primerki. Vrednost lastnosti $dodajZOznako$ primerka razreda $TipOpravilaAgenta$ agentu pove, da je potrebno dodati nov primerek komunikacijskega akta v stanju $pripravljen$ in z enako vrednostjo lastnosti $jeTipa$, kot ima vrednost lastnosti $imaTipOpravila$ dani primerek razreda $TipOpravilaAgenta$.

V danem delu ne želimo predpisati obnašanja agentov večagentnega sistema, temveč želimo podati predlog, ki omogoča, da agenti razumejo protokol, tako da lahko delujejo v skladu z njim, če oseba, ki jo agent zastopa, to želi. S tem je predlog zasnovan splošno in ga lahko uporabljamo kot razširitev poljubne implementacije agentov. Zato določamo množico predpostavk značilnosti agentov, ki lahko delujejo v skladu z danim pristopom. Kako so te značilnosti vključene v celotno obnašanje agenta, je stvar konkretne implementacije.

Predpostavke za uporabo predlaganega pristopa kot dela obnašanja agentov so naslednje:

- Agent zna izvesti tipe komunikacijskih aktov oziroma drugih opravil določenih s protokolom. V primeru, da opravila, ki ga določa protokol, ne pozna, opravilo z morebitnimi omejitvami delegira osebi, ki jo zastopa.

- Za tipe komunikacijskih aktov, ki jih agent zna izvesti, razume njihove lastnosti, ki so podane z ontologijo, in jih pri izvedbi komunikacijskih aktov ustrezno upošteva.

- Agent ob začetku sodelovalnega opravila ontologijo vloge udeleženca protokola, ki jo igra, presname iz knjižnice protokola in jo vključi v svojo ontologijo.

- Agent v svoji ontologiji izvaja naslednje posodobitve:

    - Po izvedenem opravilu primerku, ki to opravilo predstavlja, spremeni lastnost $jeVStanju$ v vrednost $koncan$.

    - Če agent začne izvajati opravilo, potem primerku, ki ga predstavlja v ontologiji, spremeni vrednost lastnosti $jeVStanju$ v $vTeku$.

    - Če agent prekine izvajanje opravila, potem primerku, ki ga predstavlja v ontologiji, spremeni vrednost lastnosti $jeVStanju$ v $prekinjen$.

    - Lastnost $dodajZOznako$ razume kot navodilo, ki določa zahtevo po dodajanju novega primerka oziroma novih primerkov opravila v ontologijo protokola. Pri tem velja naslednje:

        - Primerki imajo določeno vrednost lastnosti $jeTipa$, ki je enaka vrednosti lastnosti $imaTipOpravila$ danega primerka $TipOpravilaAgenta$. Pri tem je potrebno novemu primerku določiti vrednost $pripravljen$ lastnosti $jeVStanju$.

        - Dodani komunikacijski akt ima oznako enako kot je vrednost lastnosti $dodajZOznako$.

        - Vsak dodan komunikacijski akt ima bodisi lastnost $imaPrejemnika$ ali $imaPosiljatelja$, kar je odvisno od tega ali gre za pošiljanje ali prejem.

Vrednost lastnosti za tipa opravila agenta določa vrednost lastnosti $imaPrejemnika$ ali $imaPosiljatelja$ dodanega primerka komunikacijskega akta. Za vsakega agenta, ki ima ustrezno vlogo, agent izdela eden komunikacijski akt v stanju $pripravljen$.

- Vrednosti lastnosti $spremeniStanjeKA$ primerkov razreda $SpremembaStanjaKA$ razume kot navodilo povezano z izvedbo komunikacijskega akta, in sicer:
  - vrednost lastnosti $spremeniStanjeKA$ primerka $zacni$ pomeni, da je komunikacijski akt, ki je določen z vrednostjo lastnosti, eden izmed veljavnih komunikacijskih aktov, ki jih lahko agent začne izvajati;
  - vrednost lastnosti $spremeniStanjeKA$ primerka $prekini$ pomeni, da je izvajanje komunikacijskega akta, ki ga primerek vrednosti določa, potrebno prekiniti.
- Vrednost lastnosti $posodobi$ primerka $odstrani$ agent razume kot navodilo, da je potrebno primerek, ki je vrednost lastnosti, odstraniti iz ontologije.
- Agent je sposoben spremljanja časovnih rokov. Če časovni rok nima določenega roka začetka in je aktiven (vrednost lastnosti $jeAktiven$ je $true$), mora agent spremljati rok končanja. Po preteku roka konca nastavi vrednost lastnosti $jePotekel$ na *true*. Če ima časovni rok določen tudi rok začetka, po preteku roka začetka, nastavi vrednost lastnosti $jeAktiven$ na *true*. Pri tem je predpostavka, da so ob inicializaciji protokola, vrednosti danih lastnosti časovnih rokov ažurne.
- Vrednosti lastnosti $spremeniStanjeCR$ primerkov razreda $SpremembaStanjaCR$ razume kot navodilo za spremembo aktivnosti časovnega roka:
  - vrednost lastnosti $spremeniStanjeCR$ primerka $aktiviraj$ pomeni, da je primerku razreda $CasovniRok$, ki ga vrednost določa, potrebno spremeniti vrednost lastnosti $jeAktiven$ na $true$;

- vrednost lastnosti $spremeniStanjeCR$ primerka $deaktiviraj$ pomeni, da je primerku razreda $CasovniRok$, ki ga vrednost določa, potrebno spremeniti vrednost lastnosti $jeAktiven$ na $false$;

Primer zapisa FIPA protokola pogodbene mreže za agenta iniciatorja s pravili jezika SWRL

Primer zapisa FIPA protokola pogodbene mreže za vlogo agenta iniciatorja v jeziku SWRL je podan v spodnji tabeli (Tabela 2).

Pravilo 1	$Iniciacija(?i) \wedge jeVStanju(?i, koncan) \wedge TipOpravilaAgenta(?toa1) \wedge za(?toa1, udelezenec) \wedge imaTipOpravila(?toa1, posiljanjeCFP) \wedge \rightarrow posodobi(odstrani, ?i) \wedge dodajZOznako(?toa1, 1)$
Pravilo 2	$jeVStanju(?ka, pripravljen) \wedge imaOznako(?ka, ?o) \wedge swrlb{:}stringEqualIgnoreCase(?o, 1) \wedge TipOpravilaAgenta(?toa1) \wedge za(?toa1, udelezenec) \wedge imaTipOpravila(?toa1, prejemPredloga) \wedge TipOpravilaAgenta(?toa2) \wedge za(?toa2, udelezenec) \wedge imaTipOpravila(?toa2, prejemOdklonitve) \rightarrow spremeniStanjeKA(zacni, ?ka) \wedge dodajZOznako(?toa1, 2) \wedge dodajZOznako(?toa2, 3)$

Pravilo 3	$jeVStanju(?ka, pripravljen) \wedge imaOznako(?ka, ?o)$ $\wedge\ swrlb{:}stringEqualIgnoreCase(?o, 2)$ $\wedge\ imaOznako(?pka, ?o1)$ $\wedge\ swrlb{:}stringEqualIgnoreCase(?o1, 1)$ $\wedge\ jeVStanju(?pka, koncan)$ $\wedge\ imaVsebinoSporocila(?pka, ?v1) \wedge vsebuje(?v1, ?op1)$ $\wedge\ imaPrejemnika(?pka, ?pre)$ $\wedge\ imaPosiljatelja(?ka, ?pre)$ $\wedge\ imaVsebinoSporocila(?ka, ?v2) \wedge vsebuje(?v2, ?op2)$ $\wedge\ sameAs(?op1, ?op2) \wedge TipOpravilaAgenta(?toa2)$ $\wedge\ TipOpravilaAgenta(?toa3) \wedge za(?toa2, udelezenec)$ $\wedge\ imaTipOpravila(?toa2, posiljanjeSprejemaPredloga)$ $\wedge\ za(?toa3, udelezenec)$ $\wedge\ imaTipOpravila(?toa3, posiljanjeZavrnitvePredloga)$ $\rightarrow\ spremeniStanjeKA(zacni, ?ka)$ $\wedge\ imaCasovniRok(?ka, CR1)$ $\wedge\ spremeniStanjeCR(aktiviraj, CR1)$ $\wedge\ dodajZOznako(?toa2, 4) \wedge dodajZOznako(?toa3, 5)$

Pravilo 4	$jeVStanju(?ka, pripravljen) \wedge imaOznako(?ka, ?o)$ $\wedge\ swrlb{:}stringEqualIgnoreCase(?o, 3)$ $\wedge\ imaOznako(?pka, ?o1)$ $\wedge\ swrlb{:}stringEqualIgnoreCase(?o1, 1)$ $\wedge\ jeVStanju(?pka, koncan)$ $\wedge\ imaVsebinoSporocila(?pka, ?v1) \wedge vsebuje(?v1, ?op1)$ $\wedge\ imaPrejemnika(?pka, ?pre)$ $\wedge\ imaPosiljatelja(?ka, ?pre)$ $\wedge\ imaVsebinoSporocila(?ka, ?v2) \wedge vsebuje(?v2, ?op2)$ $\wedge\ sameAs(?op1, ?op2)$ $\rightarrow spremeniStanjeKA(zacni, ?ka)$ $\wedge\ imaCasovniRok(?ka, CR1)$ $\wedge\ spremeniStanjeCR(aktiviraj, CR1)$
Pravilo 5	$jeVStanju(?ka, pripravljen) \wedge imaOznako(?ka, ?o)$ $\wedge\ swrlb{:}stringEqualIgnoreCase(?o, 4)$ $\wedge\ imaOznako(?pka, ?o1)$ $\wedge\ swrlb{:}stringEqualIgnoreCase(?o1, 2)$ $\wedge\ jeVStanju(?pka, koncan)$ $\wedge\ imaVsebinoSporocila(?pka, ?v1) \wedge vsebuje(?v1, ?r)$ $\wedge\ imaPosiljatelja(?pka, ?pre)$ $\wedge\ imaPrejemnika(?ka, ?pre)$ $\wedge\ imaVsebinoSporocila(?ka, ?v2) \wedge SprejetoOpravilo(?r)$ $\wedge\ jeAktiven(CR1, true) \wedge jePotekel(CR1, true)$ $\rightarrow spremeniStanjeKA(zacni, ?ka) \wedge vsebuje(?v2, ?r)$

Pravilo 6	$jeVStanju(?ka, pripravljen) \wedge imaOznako(?ka, ?o)$ $\wedge\ swrlb{:}stringEqualIgnoreCase(?o, 5)$ $\wedge\ imaOznako(?pka, ?o1)$ $\wedge\ swrlb{:}stringEqualIgnoreCase(?o1, 2)$ $\wedge\ jeVStanju(?pka, koncan)$ $\wedge\ imaVsebinoSporocila(?pka, ?v1) \wedge vsebuje(?v1, ?r)$ $\wedge\ imaPosiljatelja(?pka, ?pre)$ $\wedge\ imaPrejemnika(?ka, ?pre)$ $\wedge\ imaVsebinoSporocila(?ka, ?v2)$ $\wedge\ ZavrnjenoOpravilo(?r) \wedge jeAktiven(CR1, true)$ $\wedge\ jePotekel(CR1, true)$ $\rightarrow spremeniStanjeKA(zacni, ?ka) \wedge vsebuje(?v2, ?r)$

Tabela 2: Zapis FIPA protokola Contract Net s pravili v jeziku SWRL

Primer zapisa FIPA protokola angleških avkcij za vlogo agenta iniciatorja s pravili jezika SWRL

Primer zapisa FIPA protokola angleških avkcij v jeziku SWRL za vlogo agenta iniciatorja je podan v spodnji tabeli (Tabela 3).

Pravilo 1	$Iniciacija(?\,i) \wedge jeVStanju(?\,i, koncan) \wedge TipOpravilaAgenta(?\,toa1)$ $\wedge\ za(?\,toa1, udelezenec)$ $\wedge\ imaTipOpravila(?\,toa1, posiljanjeObvestila)$ $\wedge\ TipOpravilaAgenta(?\,toa3) \wedge za(?\,toa3, udelezenec)$ $\wedge\ imaTipOpravila(?\,toa3, posiljanjeObvestila)$ $\rightarrow dodajZOznako(?\,toa1, 1) \wedge dodajZOznako(?\,toa3, 6)$ $\wedge\ posodobi(odstrani, ?\,i)$
Pravilo 2	$jeVStanju(?\,ka, pripravljen) \wedge imaOznako(?\,ka, ?\,o)$ $\wedge\ swrlb{:}stringEqualIgnoreCase(?\,o, 1)$ $\wedge\ TipOpravilaAgenta(?\,toa) \wedge za(?\,toa, udelezenec)$ $\wedge\ imaTipOpravila(?\,toa, posiljanjeCFP) \wedge Avkcija(?\,a)$ $\wedge\ seJeZacela(?\,a, true) \wedge imaVsebinoSporocila(?\,ka, ?\,vs)$ $\rightarrow spremeniStanjeKA(zacni, ?\,ka)$ $\wedge\ dodajZOznako(?\,toa, 2) \wedge vsebuje(?\,vs, ?\,a)$

Pravilo 3	$jeVStanju(?ka, pripravljen) \wedge imaOznako(?ka, ?o)$ $\wedge\ swrlb{:}stringEqualIgnoreCase(?o, 2) \wedge Prodaja(?p)$ $\wedge\ imaPredmetProdaje(?p, ?b) \wedge Blago(?b)$ $\wedge\ imaIzklicnoCeno(?b, ?c)$ $\wedge\ imaVsebinoSporocila(?ka, ?vs)$ $\wedge\ imaOznako(?pka, ?o1)$ $\wedge\ swrlb{:}stringEqualIgnoreCase(?o, 1)$ $\wedge\ jeVStanju(?pka, koncan) \wedge TipOpravilaAgenta(?toa)$ $\wedge\ za(?toa, udelezenec)$ $\wedge\ imaTipOpravila(?toa, prejemPredloga)$ $\rightarrow spremeniStanjeKA(zacni, ?ka) \wedge vsebuje(?vs, ?p)$ $\wedge\ dodajZOznako(?toa, 3)$

Pravilo 4	$jeVStanju(?ka, pripravljen) \wedge imaOznako(?ka, ?o)$ $\wedge\ swrlb{:}stringEqualIgnoreCase(?o, 3) \wedge Prodaja(?p)$ $\wedge\ imaPredmetProdaje(?p, ?b) \wedge Blago(?b)$ $\wedge\ imaOznako(?pka, ?opka) \wedge jeVStanju(?pka, koncan)$ $\wedge\ swrlb{:}stringEqualIgnoreCase(?opka, 2)$ $\wedge\ imaPrejemnika(?pka, ?pr) \wedge imaPosiljatelja(?ka, ?pr)$ $\wedge\ imaVsebinoSporocila(?pka, ?vs) \wedge vsebuje(?vs, ?p)$ $\wedge\ imaVsebinoSporocila(?ka, ?vs1) \wedge vsebuje(?vs1, ?p1)$ $\wedge\ Prodaja(?p1) \wedge sameAs(?p1, ?p) \wedge imaCeno(?p1, ?c)$ $\wedge\ imaIzklicnoCeno(?p1, ?ic)$ $\wedge\ swrlb{:}greaterThanOrEqual(?c, ?ic)$ $\wedge\ TipOpravilaAgenta(?toa1) \wedge za(?toa1, udelezenec)$ $\wedge\ imaTipOpravila(?toa1, posiljanjeSprejemaPredloga)$ $\wedge\ TipOpravilaAgenta(?toa2) \wedge za(?toa2, udelezenec)$ $\wedge\ imaTipOpravila(?toa2, posiljanjeZavrnitvePredloga)$ $\rightarrow spremeniStanjeKA(zacni, ?ka)$ $\wedge\ imaCasovniRok(?ka, CR1)$ $\wedge\ spremeniStanjeCR(aktiviraj, CR1)$ $\wedge\ dodajZOznako(?toa1, 4) \wedge dodajZOznako(?toa2, 5)$

Pravilo 5	$jeVStanju(?ka, pripravljen) \wedge imaOznako(?ka, ?o)$ $\wedge\ swrlb{:}stringEqualIgnoreCase(?o, 4)$ $\wedge\ SprejetaProdaja(?p) \wedge imaPredmetProdaje(?p, ?b)$ $\wedge\ Blago(?b) \wedge imaOznako(?pka, ?opka)$ $\wedge\ jeVStanju(?pka, koncan)$ $\wedge\ swrlb{:}stringEqualIgnoreCase(?opka, 3)$ $\wedge\ imaPosiljatelja(?pka, ?pr) \wedge imaPrejemnika(?ka, ?pr)$ $\wedge\ imaVsebinoSporocila(?ka, ?vs2)$ $\wedge\ imaVsebinoSporocila(?pka, ?vs) \wedge vsebuje(?vs, ?p)$ $\wedge\ TipOpravilaAgenta(?toa) \wedge za(?toa, udelezenec)$ $\wedge\ imaTipOpravila(?toa, posiljanjeCFP)$ $\wedge\ TipOpravilaAgenta(?toa2) \wedge za(?toa2, udelezenec)$ $\wedge\ imaTipOpravila(?toa2, posiljanjeZahteve)$ $\wedge\ jeAktiven(CR1, true) \wedge jePotekel(CR1, true)$ $\rightarrow spremeniStanjeKA(zacni, ?ka) \wedge dodajZOznako(?toa, 2)$ $\wedge\ dodajZOznako(?toa2, 7) \wedge vsebuje(?vs2, ?p)$ $\wedge\ spremeniStanjeCR(deaktiviraj, CR1)$

Pravilo 6	$jeVStanju(?ka, pripravljen) \wedge imaOznako(?ka, ?o)$ $\wedge swrlb{:}stringEqualIgnoreCase(?o, 2) \wedge Prodaja(?p)$ $\wedge imaPredmetProdaje(?p, ?b) \wedge Blago(?b)$ $\wedge imaIzklicnoCeno(?b, ?ic) \wedge Prodaja(?p1)$ $\wedge imaPredmetProdaje(?p1, ?b) \wedge Blago(?b)$ $\wedge imaCeno(?b, ?c) \wedge swrlb{:}greaterThan(?ic, ?c)$ $\wedge imaOznako(?pka, ?o1)$ $\wedge swrlb{:}stringEqualIgnoreCase(?o, 4)$ $\wedge jeVStanju(?pka, koncan)$ $\wedge imaVsebinoSporocila(?pka, ?v1) \wedge vsebuje(?v1, ?p)$ $\wedge imaVsebinoSporocila(?ka, ?v2)$ $\wedge TipOpravilaAgenta(?toa) \wedge za(?toa, udelezenec)$ $\wedge imaTipOpravila(?toa, prejemPredloga)$ $\rightarrow spremeniStanjeKA(zacni, ?ka) \wedge vsebuje(?v2, ?p1)$ $\wedge dodajZOznako(?toa, 3)$
Pravilo 7	$jeVStanju(?ka, pripravljen) \wedge imaOznako(?ka, ?o)$ $\wedge swrlb{:}stringEqualIgnoreCase(?o, 5)$ $\wedge ZavrnjenaProdaja(?p) \wedge imaPredmetProdaje(?p, ?b)$ $\wedge Blago(?b) \wedge imaOznako(?pka, ?opka)$ $\wedge jeVStanju(?pka, koncan)$ $\wedge swrlb{:}stringEqualIgnoreCase(?opka, 3)$ $\wedge imaPosiljatelja(?pka, ?pr) \wedge imaPrejemnika(?ka, ?pr)$ $\wedge imaVsebinoSporocila(?ka, ?vs2)$ $\wedge imaVsebinoSporocila(?pka, ?vs) \wedge vsebuje(?vs, ?p)$ $\wedge jeAktiven(CR1, true) \wedge jePotekel(CR1, true)$ $\rightarrow spremeniStanjeKA(zacni, ?ka) \wedge vsebuje(?vs2, ?p)$

Pravilo 8	$jeVStanju(?ka, pripravljen) \wedge imaOznako(?ka, ?o) \wedge swrlb{:}stringEqualIgnoreCase(?o, 6) \wedge Avkcija(?a) \wedge jeZakljucena(?a, true) \wedge imaVsebinoSporocila(?ka, ?vs) \wedge \wedge jeAktiven(CR1, true) jePotekel(CR1, true) \rightarrow vsebuje(?vs, ?a) \wedge spremeniStanjeKA(zacni, ?ka)$
Pravilo 9	$jeVStanju(?ka, pripravljen) \wedge imaOznako(?ka, ?o) \wedge swrlb{:}stringEqualIgnoreCase(?o, 7) \wedge IzvedbaTransakcije(?it) \wedge zaProdajo(?it, ?p) \wedge imaVsebinoSporocila(?ka, ?vs) \wedge imaPredmetProdaje(?p, ?b) \wedge Blago(?b) \wedge imaIzklicnoCeno(?b, ?ic) \wedge imaCeno(?b, ?c) \wedge imaOznako(?pka, ?op) \wedge swrlb{:}stringEqualIgnoreCase(?op, 4) \wedge jeVStanju(?pka, koncan) \wedge imaVsebinoSporocila(?pka, ?v1) \wedge vsebuje(?v1, ?p) \wedge AgentIniciator(?a) \wedge imaRezerviranoCeno(?a, ?rc) \wedge swrlb{:}greaterThan(?ic, ?c) \wedge swrlb{:}greaterThan(?c, ?rc) \wedge jeAktiven(CR1, true) \wedge jePotekel(CR1, true) \rightarrow vsebuje(?vs, ?it) \wedge spremeniStanjeKA(zacni, ?ka)$

Tabela 3: Zapis FIPA protokola angleških avkcij s pravili v jeziku SWRL

7.5 Modeliranje protokolov sodelovanja na poslovni ravni in preslikava v knjižnico protokolov

Namen poglavja modeliranje protokolov sodelovanja na poslovni ravni in preslikava v množico protokolov je prikazati, da se protokoli zajeti v knjižnici protokolov, ki je del organizacijske ontologije, lahko zajamejo na poslovni ravni s strani poslovnih uporabnikov ter kako lahko k

temu pristopimo. Podana so pravila preslikav iz opredelitev na poslovni ravni v opredelitve na ravni organizacijske ontologije, in sicer tako, da je mogoča čim višja raven avtomatizacije preslikav.

Protokole sodelovanja na poslovni ravni bi lahko zajeli na različne načine, na primer v enem izmed jezikov modeliranja poslovnih procesov, na primer BPMN, kot del opredelitve poslovno-informacijske arhitekture, na primer v jeziku Archimate [47], ali kot del modela poslovnih pravil. V poglavju je zajem pravil na poslovni ravni zasnovan čim bolj splošno z namenom, da podpira natanko tiste koncepte, ki so potrebni za opredelitev potrebnega protokola sodelovanja, in zato, da ga je mogoče prenesti in uporabiti tudi v drugih okoljih modeliranja poslovne ravni.

Pri opredeljevanju protokolov na poslovni ravni izhajamo iz minimalnih omejitev komunikacijskih aktov (tipi komunikacijskih aktov, vloga lastnika, vloge prejemnikov sporočila), ki jim zaradi pogostosti uporabe dodamo časovne omejitve in omejitve vsebine. Gre za določanje pravil, ki opredeljujejo funkcije veljavnih komunikacijskih aktov. Pri tem na poslovni ravni pravil ne določamo za posamezne komunikacijske akte, temveč za primerke komunikacije. Posamezen primerek komunikacije v okviru sodelovalnega opravila razumemo kot komunikacijski akt tipa pošlji, v katerem je sporočilo poslano enemu ali več prejemnikom, in komunikacijske akte tipa prejmi, v katerih prejemniki sporočilo prejmejo. Poslovni uporabnik namreč določa komunikacijo kot celoto in ne posameznih elementarnih opravil. Na podlagi omejitev vlog udeležencev, pravila preslikamo v ontologije posameznih udeležencev sodelovanja na nivoju knjižnice protokolov (7.2 Protokoli sodelovanja: F29-32). Za vsako komunikacijo oziroma skupino enakovrednih primerkov komunikacij določimo, kdaj jo je dovoljeno izvesti, kdo jo lahko izvede in vse morebitne dodatne omejitve. Enakovredni primerki komunikacije so primerki, ki se razlikujejo zgolj po pošiljatelju - imajo enako funkcijo v sodelovalnem opravilu, enako vsebino, isti čas pošiljanja, istega prejemnika oziroma iste prejemnike ter različne pošiljatelje. Do njih lahko pride, kadar je pošiljatelj določen z vlogo, ki ji ustreza več oseb.

Opredeljevanje protokolov sodelovanja na poslovni ravni temelji na uporabi konceptov organizacijske ontologije, ki so uporabniku predstavljeni v njemu razumljivi obliki. V primerih, ko obstoječi koncepti ontologije za opredelitev protokola ne zadoščajo, poslovni uporabnik opredeli nove koncepte. Te lahko opredeli v povezavi z obstoječimi omejitvami in koncepti. Na primer, če množica vlog pošiljateljev ne ustreza, določi novo vlogo pošiljatelja. Če protokol

vsebuje koncepte, ki v ontologiji še niso zajeti, se ob preslikavi v knjižnico protokolov prožijo aktivnosti obvladovanja znanja, kar je podrobneje opisano v poglavju 8.6 Spremembe konceptualne ravni organizacijske ontologije, ki izvirajo iz opredelitve in izvajanja uporabniških opravil.

Podatki, ki jih poslovni uporabnik opredeli za komunikacijo oziroma skupino enakovrednih komunikacij, so podani v naslednjih sklopih:

- *Oznaka komunikacije*:

  Poslovni uporabnik vsaki komunikaciji oziroma skupini enakovrednih primerkov komunikacije v protokolu sodelovanja določi enolično oznako.

- *Tip komunikacije*:

  Komunikacija je določenega tipa. Pri tem so tipi komunikacije enaki tipom parov komplementarnih komunikacijskih aktov. Priporočen pristop je, da poslovni uporabnik ustrezni tip izbere iz vnaprej določene množice tipov komunikacije.

- *Pošiljatelj*:

  Potrebno je določiti pošiljatelja, ki ga opredelimo z vlogo, ki jo igra v protokolu.

  Priporočen pristop je, da poslovni uporabnik ustrezno vlogo izbere iz vnaprej določene množice vlog.

- *Prejemniki*:

  Potrebno je določiti prejemnike, ki jih opredelimo z vlogami, ki jih igrajo v protokolu.

  Priporočen pristop je, da poslovni uporabnik ustrezne vloge izbere iz vnaprej določene množice različnih vlog.

- *Časovna omejitev končanja opravila*:

  Izvedba komunikacije je lahko časovno pogojena. Časovna omejitev pomeni, da se lahko komunikacija oziroma skupina enakovrednih komunikacij izvede do časovnega roka. Lahko

je podana eksplicitno ob opredelitvi protokola, na primer dva dni, ali simbolično z oznako, ki dobi vrednost ob izvedbi protokola.

- *Predpogoj*:

  Predpogoj določa pogoj za začetek komunikacije, ki je vezan na izvajanje določenega primerka druge komunikacije, na časovno omejitev začetka oziroma na drugo omejitev.

  V primeru predpogoja, ki se sklicuje na stanje drugega primerka komunikacije, se poslovni uporabnik nanj sklicuje z oznako komunikacije in določi v kakšnem stanju mora biti, da je predpogoj izpolnjen. Najpogostejši primer predpogoja je, da mora biti določena komunikacija končana. Poslovni uporabnik lahko izbira med stanji *pripravljen*, *končan*, *v teku*, *prekinjen* ali *nepravilno zaključen*. Posebnosti protokolov lahko zahtevajo, da poslovni uporabnik doda tudi druga stanja.

  Časovna omejitev začetka komunikacije je lahko dveh vrst, in sicer: komunikacija se ne sme izvesti pred določenim časovnim rokom ali komunikacija se lahko izvede po določenem roku.

  Druge omejitve podane v polju predpogoj se lahko navezujejo na primer na prejeta in poslana sporočila.

- *Omejitev vsebine*:

  V določenih primerih je potrebno omejiti vsebino sporočila komunikacije. Omejitev vsebine določa koncepte vsebine sporočila in njihove omejitve: vsebina je lahko omejena z določenim konceptom, ki ga mora vsebovati, ali z določeno lastnostjo, ki jo mora primerek vsebine imeti. Pri tem izhajamo iz obstoječih razredov, primerkov in lastnosti zajetih v organizacijski ontologiji, ki so uporabniku predstavljeni v njemu razumljivi obliki.

  Oblikovanje ustrezne vsebine je prav tako predpogoj za izvedbo opravila. Zaradi večje preglednosti ga navajamo ločeno. Če ji ne moremo zadovoljiti, se komunikacija ne izvede.

- Druge omejitve:

  Morebitne druge omejitve poslovni uporabnik poda v obliki teksta.

V primeru, da predlagani koncepti oziroma množica obstoječih konceptov poslovnemu uporabniku ne zadoščajo, dodaja nove koncepte. Na primer, če predlagani tipi komunikacije ali vloge ne zadoščajo, poslovni uporabnik doda nov tip komunikacije oziroma novo vlogo.

Poleg omejitev lahko določimo tudi *posledice* komunikacije.

7.5.1 Primer

Tabela 4 podaja primer opredelitve protokola pogodbene mreže na poslovni ravni. Predpostavka je, da imajo omejitve ustrezne koncepte določene v skupni ontologiji FIPA protokolov. Za določitev posameznih pravil lahko ugotovimo naslednje:

- Pravilo 1 določa prvo komunikacijo protokola sodelovanja, ki je tipa CFP, in vloge pošiljateljev ter prejemnikov. Omejuje vsebino, ki naj vsebuje opredelitev opravila. Omejitev je privzeta lastnost tipa komunikacije CFP, ki izhaja iz skupne ontologije FIPA protokolov, zato je poslovnemu uporabniku ni potrebno posebej določiti.

- Pravilo 2 določa komunikacijo med udeležencem in iniciatorjem, in sicer za primer, ko udeleženec na CFP odgovori s predlogom. Komunikacija je omejena s časovnim rokom z oznako CR1. Pravilo omejuje tudi vsebino, ki naj vsebuje opravilo, ki ga je udeleženec prejel CFP od iniciatorja. Za udeleženca je opravilo v tem primeru zaželeno. V ontologiji udeleženca je v ta namen opredeljen razred $ZazelenoOpravilo$. Predpogoj za izvedbo komunikacije je, da je komunikacija 1 končana.

- Pravilo 3 določa komunikacijo med udeležencem in iniciatorjem, in sicer za primer, ko udeleženec CFP odkloni. Kot v pravilu 2, je omejena komunikacija s časovnim rokom z oznako CR1 in vsebina na opravilo, ki ga je udeleženec prejel od inciatorja v komunikaciji 1. Za udeleženca opravilo v tem primeru ni zaželeno. V ontologiji udeleženca je v ta namen opredeljen razred $NezazelenoOpravilo$. Razreda $ZazelenoOpravilo$ in $NezazelenoOpravilo$ sta disjunktna podrazreda razreda Opravilo. Ker je lahko zadoščeno le eni izmed omejitev vsebine pravil 2 in 3, se vedno izvede le ena izmed obeh komunikacij. Predpogoj za izvedbo komunikacije je, da je komunikacija 1 končana.

- Pravilo 4 opisuje komunikacijo sprejema predloga. Poleg vlog pošiljatelja in prejemnika je omejena vsebina, ki vsebuje opravilo, ki ga je iniciator prejel v okviru komunikacije pravila 2 od udeleženca in je sprejeto. V ontologiji je sprejeto opravilo predstavljeno kot podrazred $SprejetoOpravilo$ razreda $Opravilo$. Pogoj za proženje pravila je pretek časovnega roka CR1.

- Pravilo 5 opisuje komunikacijo zavrnitve predloga. Poleg vlog pošiljatelja in prejemnika je omejena vsebina, ki vsebuje opravilo, ki ga je iniciator prejel v okviru komunikacije pravila 2 od udeleženca in je sprejeto. V ontologiji je sprejeto opravilo predstavljeno kot podrazred $ZavrnjenoOpravilo$ razreda $Opravilo$. Pogoj za proženje pravila je pretek časovnega roka CR1.

Tabela 5 podaja primer opredelitve protokola angleških avkcij na poslovni ravni. Kot pri opredelitvi protokola pogodbene mreže, je predpostavka tudi tukaj, da imajo omejitve ustrezne koncepte določene v skupni ontologiji FIPA protokolov. Za določitev posameznih pravil lahko ugotovimo naslednje:

- Pravilo 1 določa prvo komunikacijo protokola sodelovanja, ki je tipa Obvestilo, in vloge pošiljateljev ter prejemnikov. Omejuje vsebino, ki naj vsebuje obvestilo o začetku avkcije.

- Pravilo 2 določa komunikacijo tipa CFP. Poleg vlog pošiljatelja in prejemnikov omejuje vsebino, ki mora vsebovati blago in izklicno ceno blaga. Izklicna cena je v ontologiji predstavljena kot podatkovna lastnost razreda $Blago$. Koncepta sta specifična za dani protokol sodelovanja in zato nista del skupne ontologije FIPA protokolov. Zato ju poslovni uporabnik doda kot nova koncepta vsebine. Pravilo se izvede pod pogojem, da je iniciator že poslal obvestilo o začetku avkcije, ali pod pogojem, da je pred tem isti agent poslal sprejem predloga podanega s pravilom 4 in da je nova izklicna cena blaga večja od sprejete predlagane cene.

- Pravilo 3 določa komunikacijo tipa Predlog. Poleg vlog pošiljatelja in prejemnikov postavlja časovni rok in omejuje vsebino. To pomeni, da je časovni rok s tem aktiviran. Predpogoji pravila so da je komunikacija 2 končana. Predmet prodaje mora biti isti, kot predmet prodaje

v CFP komunikacije 2. Predlagana cena mora biti večja ali enaka od cene, ki jo je določil iniciator v CFP.

- Pravilo 4 opisuje komunikacijo sprejema predloga. Poleg vlog pošiljatelja in prejemnika je omejena vsebina, ki vsebuje sprejeti predlog. Gre za specializacijo koncepta predloga, ki je v ontologiji predstavljena kot podrazred $SprejetiPredlog$ razreda $Predlog$. Ta že predstavlja del skupne ontologije FIPA protokolov. Predpogoj za proženje komunikacije je pretek časovnega roka. Predpogoj, ki izhaja iz vsebine je, da je prodaja sprejeta in da je bila prejeta od agenta z vlogo Udeleženec v komunikaciji 3. Temu bo lahko zadoščeno samo za tiste Udeležence, katerih predlog je sprejet. Posledica komunikacijskega akta je, da CR1 ni več aktiven (spremljanje CR1 se preneha).

- Pravilo 5 opisuje komunikacijo zavrnitve predloga. Poleg vlog pošiljatelja in prejemnika je omejena vsebina, ki vsebuje zavrnjeni predlog. Gre za specializacijo koncepta predloga, ki je v ontologiji predstavljena kot podrazred $ZavrnjeniPredlog$ razreda $Predlog$. Ta že predstavlja del skupne ontologije FIPA protokolov. Predpogoj za proženje pravila je pretek časovnega roka. Predpogoj, ki izhaja iz vsebine je, da je prodaja zavrnjena in da je bila prejeta od agenta z vlogo Udeleženec v komunikaciji 3. Temu bo lahko zadoščeno samo za tiste Udeležence, katerih predlog je zavrnjen.

- Pravilo 6 opisuje komunikacijo pošiljanja obvestila. Omejuje vsebino, ki naj vsebuje obvestilo o začetku avkcije. Predpogoj je potek časovnega roka CR1, če je le ta aktiven.

- Pravilo 7 opisuje komunikacijo pošiljanja zahteve. Poleg vlog pošiljatelja in prejemnika je omejena vsebina, ki vsebuje zahtevo po opravilu *izvedba transakcije*. Izvedba transakcije je v ontologiji predstavljena kot podrazred $IzvedbaTransakcije$ razreda $Opravilo$. Ker je specifična za dani protokol, ni del skupne ontologije FIPA protokolov in jo poslovni uporabnik doda z opredelitvijo pravila. Pogoj za izvedbo komunikacije je, da je iniciator poslal sprejem predloga udeležencu, da je potekel časovni rok, ki je aktiven, da je izklicna cena blaga večja kot sprejeta cena ter da je cena blaga večja od rezervirane cene.

Oznaka komunikacije	Tip komunikacije	Pošiljatelj	Prejemnik	Časovna omejitev konca	Omejitev vsebine	Predpogoj	Posledica
1	CFP	Iniciator	Udeleženec		Opravilo.		
2	Predlog	Udeleženec	Iniciator	CR1	Opravilo: Opravilo je prejel od vloge Iniciator v 1. in Opravilo je zaželeno.	1 je končan.	
3	Odklonitev	Udeleženec	Iniciator	CR1	Opravilo: Opravilo je prejel od vloge Iniciator v 1. in Opravilo ni zaželeno.	1 je končan.	
4	Sprejem predloga	Iniciator	Udelezenec		Opravilo: Opravilo je prejel od vloge Udeleženec v 2. in Opravilo je sprejeto.	CR1 se je iztekel.	
5	Zavrnitev	Iniciator	Udelezenec		Opravilo: Opravilo je prejel od vloge Udeleženec	CR1 se je iztekel	

	predloga				v 2. in Opravilo ni sprejeto.		

Tabela 4: Primer opredelitve protokola pogodbene mreže

Oznaka komunikacije	Tip komunikacije	Pošiljatelj	Prejemnik	Časovna omejitev konca	Omejitev vsebine	Predpogoj	Posledica
1	Obvestilo	Iniciator	Udeleženec		Obvestilo: Obvestilo o začetku avkcije.		
2	CFP	Iniciator	Udeleženec		Prodaja: • predmet prodaje je Blago. • Blago ima določeno izklicno ceno.	1 je končan.	
					Prodaja: • predmet prodaje je Blago, ki ima določeno izklicno ceno, • enaka kot Prodaja poslana vlogi Udeleženec v	4 je končan.	

					4, ki ima blago s Ceno. • Izklicna cena > Cena.		
3	Predlog	Udeleženec	Iniciator	CR1	Prodaja: • predmet prodaje je Blago: - je enako kot Blago prejeto od vloge Iniciator v 2, - Blago ima določeno ceno. Cena ≥ izklicna cena.	2 je končan.	
4	Sprejem predloga	Iniciator	Udeleženec		Prodaja: • predmet prodaje je Blago, ki ima določeno ceno, • enaka kot Prodaja prejeta od vloge Udeleženec v 3, • predlog prodaje je sprejet.	3 je končan. CR1 je aktiven in je potekel.	CR1 ni več aktiven.
5	Zavrnitev predloga	Iniciator	Udeleženec		Prodaja: • predmet prodaje je Blago, ki ima določeno ceno, • enaka kot Prodaja prejeta od vloge Udeleženec v 3, • predlog prodaje ni sprejet.	3 je končan. CR1 je aktiven in je potekel.	

6	Obvestilo	Iniciator	Udelezenec		Obvestilo o zaključku avkcije.	CR1 je aktiven in je potekel.	
7	Zahteva	Iniciator	Udeleženec		Izvedba transakcije: • za Prodaja: - enaka kot Prodaja poslana udeležencu v 4, - ima predmet prodaje Blago, ki ima ceno < izklicna cena in rezervirana cena < cena.	CR1 je aktiven in je potekel.	

Tabela 5: Primer opredelitve protokola angleških avkcij

7.5.2 Opredelitev preslikave

Pravila preslikave med pravili opredeljenimi na poslovni ravni in SWRL pravili protokola, se razlikujejo v odvisnosti od tega ali gre za pravilo, ki označuje začetne komunikacijski akte sodelovanja, ali pravilo, ki označuje nadaljnje komunikacijske akte.

Če pravilo ne opredeljuje predpogojev, gre za začetno komunikacijo. Začetna komunikacija se začne po zaključku akcij iniciacije protokola, ki jih izvede agent, na primer uvoz ontologije protokola v svojo ontologijo. Ko začetno akcijo konča, podobno kot za druge akcije v ontologijo zabeleži novo stanje te akcije ($koncan$). V SWRL pravilu je končanje začetne akcije pogoj za začetek prve komunikacije.

Podobno je tudi v primerih, ko se predpogoj ne nanaša na stanje drugih komunikacij, ampak zgolj na časovno omejitev začetka. V tem primeru je komunikacija lahko ali ne začetna, saj se lahko druge komunikacije na primer končajo pred veljavnim časom njenega začetka. Kadar se predpogoj nanaša zgolj na časovno omejitev začetka komunikacije, je v SWRL pravilu končanje začetne akcije pogoj prav tako pogoj za začetek dane komunikacije.

Za namenom določanja preslikav, bomo kadar ne želimo določiti konkretnega primerka, razreda ali lastnosti, za oznako uporabljali simbol v prelomljenih oklepajih - ⟨ ⟩. Podajmo naslednje simbole, ki jih bomo uporabljali za opredelitev preslikav:

- $\langle tka \rangle$ označuje primerek razreda $TipKomunikacijskegaAkta$, ki je določen v polju tip komunikacije danega pravila poslovnega uporabnika. Poslovni uporabnik določi tip komunikacije, ki v ontologiji protokola ustreza paru komplementarnih tipov komunikacijskih aktov (pošiljanje in prejem). V ontologiji udeleženca, ki mu je poslovni uporabnik določil vlogo pošiljatelja, se preslika v ustrezni primerek, ki opisuje komunikacijski akt pošiljanja, v ontologiji udeleženca, ki igra vlogo prejemnika se preslika v ustrezni primerek, ki opisuje komunikacijski akt prejema. Na primer, v pravilu 1 opredelitve protokola pogodbenih mrež (Tabela 4) v ontologiji udeleženca Iniciator $\langle tka \rangle$ ustreza primerku $pošiljanjeCFP$, medtem ko v ontologiji udeleženca Udeleženec $\langle tka \rangle$ ustreza primerku $prejemCFP$.

- $\langle v \rangle$ ustreza primerku vloge, ki označuje vlogo udeleženca nasprotne strani komunikacije. Na primer, v pravilu 1 opredelitve protokola pogodbenih mrež (Tabela 4) v ontologiji

udeleženca Iniciator $\langle v \rangle$ ustreza primerku u$delezenec$, v ontologiji udeleženca Udeleženec pa primerku $iniciator$.

- ? ka označuje spremenljivko, ki predstavlja komunikacijski akt.
- $\langle cr \rangle$ označuje primerek časovnega roka. Določa ga polje časovne omejitve konca ali polje predpogoj, ki se nanaša na časovno omejitev začetka komunikacije.
- $\langle R \rangle$ označuje razred, ki predstavlja koncept, ki bi ga naj sporočilo komunikacije vsebovalo. Določa ga polje omejitev vsebine.
- $\langle o \rangle$ označuje niz oznake komunikacije opredeljen v polju Oznaka komunikacije. Komunikacijski akti, ki izhajajo iz iste komunikacije, imajo enako oznako.
- $\langle lastnost \rangle$ označuje določeno lastnost koncepta vsebine sporočila, ki je podana kot del omejitve vsebine.
- Enaki simboli elementov pravil (spremenljivk, primerkov, razredov in lastnosti), ki jih uporabljamo v različnih pravilih preslikave, se nanašajo na iste elemente končnega SWRL pravila. Kadar so podani znotraj prelomljenih oklepajev, na primer $\langle ? v \rangle$, se enaka oznaka znotraj posameznega pravila preslikave nanaša na isto entiteto (spremenljivko, primerek, razred, lastnost) v končnem SWRL pravilu, medtem ko v različnih pravilih preslikave ne pomeni iste entitete in se zato preslika v različne oznake (spremenljivk, primerkov, razredov ali lastnosti) v končnem SWRL pravilu.

Privzema se, da so različni izrazi v glavi in telesu pravila ločeni z znakom $\wedge$, zato v opredelitvi preslikav ob posameznih izrazih tega posebej ne navajamo.

Vsako pravilo protokola, ki ga opredeli poslovni uporabnik, se praviloma v vsaki ontologiji udeleženca preslika v eno SWRL pravilo. Izjema je pravilo, ki vsebuje dve različni skupini predpogojev – komunikacija se lahko izvede, če držijo eni ali drugi predpogoji. V tem primeru se pravilo preslika v toliko SWRL pravil, kot je skupin predpogojev. Na primer, pravilo 2 v protokolu angleških avkcij (Tabela 5) ima dve skupini predpogojev, zato se preslika v dve SWRL pravili. Pri tem se v preslikavi v posamezno SWRL pravilo upošteva skupni del pravila in posamezno skupino predpogojev.

Poleg tega izdelamo eno dodatno SWRL pravilo po naslednjem pravilu preslikave:

PP1 Telo SWRL pravila vsebuje izraz z naslednjo strukturo:

$$Iniciacija(?\,i) \wedge\ jeVStanju(?\,i, koncan),$$

medtem ko glava SWRL pravila vsebuje izraz:

$$posodobi(odstrani, ?\,i).$$

V telo istega SWRL pravila za vsako komunikacijo k, ki je začetna komunikacija ali komunikacija, katere predpogoj ne vsebuje pogoja o stanju katere druge komunikacije, dodamo izraz:

$$TipOpravilaAgenta(\langle ?\,toa_k\rangle) \wedge za(\langle ?\,toa_k\rangle, \langle v_k\rangle) \wedge imaTipOpravila(\langle ?\,toa_k\rangle, \langle tka_k\rangle)$$

in v glavo SWRL pravila izraz:

$$dodajZOznako(\langle ?\,toa_k\rangle, k).$$

Pri tem imata simbola $\langle v_k\rangle$ in $\langle tka_k\rangle$ enak pomen kot simbola $\langle v\rangle$ in $\langle tka\rangle$, s tem da se nanašata na pravilo z oznako k.

Na primer, v protokolu pogodbene mreže je v dani preslikavi obravnavano pravilo komunikacije z oznako 1, ki opisuje začetno komunikacijo. Rezultat preslikave je pravilo 1 v tabeli SWRL pravil pogodbene mreže ontologije vloge Iniciator (Tabela 2).

Pravila preslikave iz pravila, ki ga določi poslovni uporabnik, v SWRL pravilo komunikacijskega akta za posameznega udeleženca sodelovanja so naslednja (PP2-PP9):

PP2 Telo SWRL pravila vsebuje naslednji izraz:

$$jeVStanju(?\,ka, pripravljen) \wedge imaOznako(?\,ka, \langle ?\,o\rangle) \wedge swrlb{:}\,stringEqualIgnoreCase(\langle ?\,o\rangle, \langle o\rangle).$$

PP3 Časovna omejitev konca opravila se preslika v izraz:

$$imaCasovniRok(?\,ka, \langle cr\rangle) \wedge spremeniStanjeCR(aktiviraj, \langle cr\rangle),$$

ki je v glavi SWRL pravila.

Primer najdemo v preslikavi komunikacije 3 protokola pogodbene mreže (Tabela 4) v SWRL pravilo 4 tega protokola (Tabela 2).

PP4 Omejitev vsebine z določenim konceptom, ki ga mora sporočilo vsebovati, in brez dodatnih omejitev tega koncepta se preslika v izraz v telesu pravila:

$$\langle R \rangle (\langle ?\, r \rangle) \wedge imaVsebinoSporocila(?\, ka, \langle ?\, v \rangle)$$

in v izraz:

$$vsebuje(\langle ?\, v \rangle, \langle ?\, r \rangle),$$

ki je v ontologiji pošiljatelja sporočila del glave SWRL pravila in v ontologiji prejemnika sporočila del telesa SWRL pravila.

Primer najdemo v preslikavi komunikacije 1 protokola angleških avkcij (Tabela 5) v SWRL pravilo 2 tega protokola (Tabela 3).

PP5 Omejitev vsebine sporočila na vsebine prejete od določenih udeležencev oziroma poslane v predhodnih komunikacijskih aktih določenim udeležencev se preslika v naslednji izraz v telesu SWRL pravila:

$$\langle R \rangle (\langle ?\, r \rangle).$$

Preostanek pravila preslikave je različen za različne vloge udeležencev komunikacije. V ontologiji udeleženca z vlogo pošiljatelja je naslednji:

- Če gre za omejitev vsebine pošiljanja z vsebino in pošiljateljem predhodno prejetega sporočila, se omejitev preslika v izraz v telesu SWRL pravila:

$$\begin{aligned} imaOznako(\langle ?\, pka \rangle, \langle ?\, o \rangle) \wedge swrlb{:}\, stringEqualIgnoreCase(\langle ?\, o \rangle, \langle po \rangle) \\ \wedge jeVStanju(\langle ?\, pka \rangle, koncan) \\ \wedge imaVsebinoSporocila(\langle ?\, pka \rangle, \langle ?\, v1 \rangle) \wedge vsebuje(\langle ?\, v1 \rangle, \langle ?\, r \rangle) \\ \wedge imaPosiljatelja(\langle ?\, pka \rangle, \langle ?\, pre \rangle) \\ \wedge imaVsebinoSporocila(?\, ka, \langle ?\, v2 \rangle). \end{aligned}$$

Če gre poleg tega tudi za omejitev, da je prejemnik sporočila isti kot je bil pošiljatelj, dodamo v telo SWRL pravila še izraz:

$$imaPrejemnika(?\, ka, \langle ?\, pre \rangle).$$

- Če gre za omejitev vsebine pošiljanja z vsebino predhodno poslanega sporočila, se omejitev preslika v izraz v telesu SWRL pravila:

$$imaOznako(\langle ?\,pka\rangle, \langle ?\,o\rangle) \wedge swrlb{:}\,stringEqualIgnoreCase(\langle ?\,o\rangle, \langle po\rangle) \wedge jeVStanju(\langle ?\,pka\rangle, koncan) \wedge imaVsebinoSporocila(\langle ?\,pka\rangle, \langle ?\,v1\rangle) \wedge vsebuje(\langle ?\,v1\rangle, \langle ?\,r\rangle) \wedge imaVsebinoSporocila(?\,ka, \langle ?\,v2\rangle).$$

Če gre poleg tega tudi za omejitev, da je prejemnik sporočila isti kot je bil prejemnik predhodnega sporočila z dano vsebino, dodamo v telo SWRL pravila še izraz:

$$imaPrejemnika(?\,ka, \langle pre\rangle) \wedge imaPrejemnika(\langle ?\,pka\rangle, \langle ?\,pre\rangle).$$

V obeh primerih (omejitev s predhodno poslano ali predhodno prejeto vsebino) v ontologiji udeleženca z vlogo pošiljatelja nastopi v glavi SWRL pravila izraz:

$$vsebuje(\langle ?\,v2\rangle, \langle ?\,r\rangle).$$

V ontologiji udeleženca z vlogo prejemnika je pravilo preslikave naslednje:

- Če gre za omejitev vsebine prejema sporočila z vsebino in pošiljateljem predhodno prejetega sporočila, se omejitev preslika v izraz v telesu SWRL pravila:

$$imaOznako(\langle ?\,pka\rangle, \langle ?\,o\rangle) \wedge swrlb{:}\,stringEqualIgnoreCase(\langle ?\,o\rangle, \langle po\rangle) \wedge jeVStanju(\langle ?\,pka\rangle, koncan) \wedge imaVsebinoSporocila(\langle ?\,pka\rangle, \langle ?\,v1\rangle) \wedge vsebuje(\langle ?\,v1\rangle, \langle ?\,r\rangle) \wedge imaVsebinoSporocila(?\,ka, \langle ?\,v2\rangle) \wedge vsebuje(\langle ?\,v2\rangle, \langle ?\,r1\rangle) \wedge sameAs(\langle ?\,r\rangle, \langle ?\,r1\rangle).$$

Če gre poleg tega tudi za omejitev pošiljatelja sporočila s to vsebino, dodamo v telo SWRL pravila še izraz:

$$imaPosiljatelja(\langle ?\,pka\rangle, \langle pre\rangle).$$

Če mora biti pošiljatelj isti, kot je bil pošiljatelj predhodno prejetega sporočila, v telo SWRL pravila dodamo še:

$$imaPosiljatelja(?\,ka, \langle pre\rangle).$$

- Če gre za omejitev vsebine prejema sporočila z vsebino predhodno poslanega sporočila, se omejitev preslika v izraz v telesu SWRL pravila:

$imaOznako(\langle ?\,pka\rangle, \langle ?\,o\rangle) \wedge swrlb{:}\,stringEqualIgnoreCase(\langle ?\,o\rangle, \langle po\rangle) \wedge$
$jeVStanju(\langle ?\,pka\rangle, koncan) \wedge imaVsebinoSporocila(\langle ?\,pka\rangle, \langle ?\,v1\rangle) \wedge$
$vsebuje(\langle ?\,v1\rangle, \langle ?\,r\rangle) \wedge imaVsebinoSporocila(?\,ka, \langle ?\,v2\rangle) \wedge$
$vsebuje(\langle ?\,v2\rangle, \langle ?\,r1\rangle) \wedge sameAs(\langle ?\,r\rangle, \langle ?\,r1\rangle)$.

Če gre poleg tega tudi za omejitev, da je pošiljatelj sporočila isti kot je bil prejemnik predhodno poslanega sporočila, dodamo v telo SWRL pravila še izraz:

$imaPrejemnika(\langle ?\,pka\rangle, \langle ?\,pre\rangle) \wedge imaPosiljatelja(?\,ka, \langle ?\,pre\rangle)$.

$\langle po\rangle$ je oznaka predhodno poslanega sporočila.

Vsak izmed navedenih izrazov je lahko razširjen z morebitnimi dodatnimi omejitvami vsebine (glej pravilo preslikave PP6).
Primer lahko najdemo v preslikavi pravila 3 protokola angleških akcij (Tabela 5) v SWRL pravila iniciatorja. Omejitev vsebine se preslika v:

$$\begin{aligned} Prodaja(?\,p) &\wedge imaPredmetProdaje(?\,p, ?\,b) \wedge Blago(?\,b) \\ &\wedge imaOznako(?\,pka, ?\,opka) \wedge jeVStanju(?\,pka, koncan) \\ &\wedge swrlb{:}\,stringEqualIgnoreCase(?\,opka, 2) \\ &\wedge imaPrejemnika(?\,pka, ?\,pr) \wedge imaPosiljatelja(?\,ka, ?\,pr) \\ &\wedge imaVsebinoSporocila(?\,pka, ?\,vs) \wedge vsebuje(?\,vs, ?\,p) \\ &\wedge imaVsebinoSporocila(?\,ka, ?\,vs1) \wedge vsebuje(?\,vs1, ?\,p1) \\ &\wedge sameAs(?\,p1, ?\,p) \wedge imaCeno(?\,p1, ?\,c) \\ &\wedge imaIzklicnoCeno(?\,p, ?\,ic) \\ &\wedge swrlb{:}\,greaterThanOrEqual(?\,c, ?\,ic). \end{aligned}$$

Omejitev vsebine v povezavi s sporočili poslanimi oziroma prejetimi v predhodnih komunikacijskih aktih se lahko nanaša tudi na sporočilo kot celoto. V tem primeru SWRL pravila za izpoljnevanje tega pogoja ne določajo lastnosti $vsebuje$, ampak preverjajo enakost vrednosti lastnosti $imaVsebinoSporočila$ obeh komunikacijskih aktov.

PP6 Morebitne druge omejitve konceptov vsebine, se preslikajo v lastnosti primerkov, ki jim ustrezajo. Posamezna omejitev se preslika v naslednji izraz v telesu SWRL pravila ontologije udeleženca z vlogo pošiljatelja sporočila:

$$\langle R\rangle(\langle ?\,r\rangle) \wedge \langle lastnost\rangle(\langle ?\,r\rangle, \langle l\rangle).$$

$\langle l\rangle$ označuje primerek ali podatek, ki je vrednost lastnosti $\langle lastnost\rangle$. Dovoljene so tudi kompleksnejše strukture. Tudi $\langle l\rangle$ je lahko omejen z vrednostjo njegovih lastnosti ali pripadnostjo razredu/-om.

Primer je omejitev vsebine v pravilu 6 protokola angleških avkcij (Tabela 5) se preslika v naslednje:

$$Prodaja(?\,p)\ \wedge imaPredmetProdaje(?\,p, ?\,b)\ \wedge Blago(?\,b)$$

PP7 Omejitev stanja predhodnega komunikacijskega akta se preslika v naslednji izraz v telesu SWRL pravila:

$$imaOznako(\langle ?\,ppka\rangle, \langle ?\,o\rangle)\ \wedge\ jeVStanju(\langle ?\,ppka\rangle, \langle stanje\rangle) \wedge swrlb{:}\,stringEqualIgnoreCase(?\,o, \langle po\rangle),$$

kjer se $\langle po\rangle$ nanaša na oznako predhodnega pravila opredelitve komunikacije.

PP8 Predpogoj, ki se nanaša na pretek časovnega roka se preslika v naslednji izraz v telesu SWRL pravila:

$$jeAktiven(\langle cr\rangle, true)\ \wedge\ jePotekel(\langle cr\rangle, true).$$

PP9 Glava SWRL pravila vsebuje izraz:

$$spremeniStanjeKA(zacni, ?\,ka).$$

PP10 Če oznaka danega pravila poslovnega uporabnika nastopa v katerem izmed predpogojev drugih pravil, ki določajo stanje $vTeku,\ koncan$, $prekinjen$ ali $napacnoZakljucen$, ali pri omejitvi vsebine, potem za vsako tovrstno pravilo poslovnega uporabnika v telo SWRL pravila dodamo izraz:

$$TipOpravilaAgenta(\langle ?\,toa\rangle) \wedge za(\langle ?\,toa\rangle, \langle v\rangle) \wedge imaTipOpravila(\langle ?\,toa\rangle, \langle tka\rangle)$$

in v glavo SWRL pravila izraz:

$$dodajZOznako(\langle ?\,toa\rangle, \langle o\rangle).$$

$\langle tka\rangle$ je določen s tipom komunikacije pravila, v katerem je dana komunikacija v predpogoju. $\langle v\rangle$ se prav tako nanaša na pravilo, v katerem je dana komunikacija v predpogoju. Opisuje nasprotnega udeleženca komunikacije od udeleženca ontologije udeleženca, v katero poteka preslikava. Na primer, v pravilu 2 opredelitve protokola angleških avkcij (Tabela 5) predpogoj določa, da mora biti komunikacija z oznako 1 končana. Torej bi opredelitvi SWRL pravila, ki opisuje pravilo 1, dodali v ontologiji udeleženca z vlogo Iniciator v telo SWRL izraz:

$$TipOpravilaAgenta(?\,toa) \wedge za(?\,toa, udelezenec) \wedge imaTipOpravila(?\,toa, posiljanjeCFP)$$

in v glavo SWRL pravila izraz:

$$dodajZOznako(?\,toa, 2).$$

To pomeni, da se za vsako pravilo, katerega predpogoj je končanje danega komunikacijskega akta, izdela ustrezen izraz, ki zahteva pripravo naslednjega (potencialnega[6]) akta.

Poleg tega dodamo izraze, če se katera izmed komunikacij, ki na opisan način vsebujejo obravnavano komunikacijo v predpogoju, prav tako pojavlja v katerem izmed predpogojev ostalih komunikacij, ki določajo stanje $pripravljen$. V tem primeru za vsako izmed takšnih pravil dodamo dodatna izraza z enako strukturo, in sicer v telo SWRL pravila:

$$TipOpravilaAgenta(\langle ?\,toa_k\rangle) \wedge za(\langle ?\,toa_k\rangle, \langle v_k\rangle) \wedge imaTipOpravila(\langle ?\,toa_k\rangle, \langle tka_k\rangle)$$

in v glavo SWRL pravila izraz:

$$dodajZOznako(\langle ?\,toa_k\rangle, k).$$

$\langle v_k\rangle$ in $\langle tka_k\rangle$ se v tem primeru nanašata na pravilo z oznako k, v katerem se pojavlja opisani predpogoj.

[6] Potencialnega pravimo zato, ker je lahko zaključek dane akcije, le eden izmed predpogojev za začetek druge akcije.

PP11 Posledica se preslika v glavo SWRL pravila. Najpogosteje gre za prekinitev komunikacijskega akta ali za deaktivacijo aktivne časovne omejitve, ki jo želimo prenehati spremljati. V prvem primeru se omejitev preslika v izraz:

$$spremeniStanjeKA(prekini, \langle ?\, ka' \rangle) \wedge imaOznako(\langle ?\, ka' \rangle, \langle o' \rangle),$$

kjer $\langle o' \rangle$ določa oznako komunikacijskega akta, ki ga je potrebno prekiniti.

V drugem primeru se omejitev preslika v izraz:

$$spremeniStanjeCR(deaktivniraj, \langle cr' \rangle),$$

kjer $\langle cr' \rangle$ določa oznako časovnega roka, ki ga je potrebno deaktivirati.

Posledica lahko določa tudi druge koncepte, ki se preslikajo v koncepte ontologije, ki jim ustrezajo.

Na osnovi opisanih primerov lahko ugotovimo, da lahko večji del opredelitev pravil protokola poslovni uporabnik izdela na podlagi obstoječe ontologije. V primeru, ko to ne zadošča, se prožijo aktivnosti evolucije ontologije. Primer opredelitve protokola pogodbenih mrež tega ne zahteva, saj zadošča skupna ontologija FIPA protokolov, medtem ko je v primeru opredelitve protokola angleških avkcij potrebno določene koncepte dodati, v kolikor v ontologiji še ne obstajajo.

Če spremembe protokolov temeljijo na obstoječih konceptih ontologije, na primer spremembe vrstnega reda pravil, pogojev izvajanja, časovnega roka itd., se preslikave sprememb v knjižnico protokolov lahko izvedejo avtomatsko. Sprememba večagentnega sistema pri tem ni potrebna. V nasprotnem primeru se prožijo aktivnosti evolucije ontologije. Če novi protokol ali sprememba obstoječega protokola zahteva določena opravila, ki jih obstoječi agenti ne morejo izvesti (niso del njihovega obnašanja), lahko sposobnosti agentov nadgradimo tako, da bodo znali izvajati tudi tovrstna opravila. V nasprotnem primeru agenti opravilo posredujejo osebi, ki je lastnik opravila in ga zastopajo.

8 Organizacijska ontologija in povezanost z vsebinskimi informacijami poslovnega procesa

8.1 Cilji

Organizacijska ontologija predstavlja temelj za omogočanje avtomatizacije uporabniških opravil. Zajemati mora tiste informacije, ki pripomorejo k doseganju tega cilja. Ker so uporabniška opravila vezana na posamezne lastnike opravil, njihove informacije in poznavanje domene, naj bo organizacijska ontologija strukturirana tako, da bo na eni strani omogočala ločevanje informacij, ki pripadajo posameznim lastnikom opravil, in na drugi strani omogočala ponovno uporabo tistih informacij, do katerih lahko dostopa več uporabnikov.

Organizacijska ontologija ne predstavlja entitete, ki je statična in se ne spreminja [49]. Tako poslovni sistem in okolje, v katerem deluje, sta izpostavljena spremembam, ki se morajo odražati tudi v organizacijski ontologiji. Med izvajanjem procesov nastajajo nove informacije, ki so lahko pomembne za vključitev v organizacijsko ontologijo. Ob proženju SIUO naj bi primerek poslovnega procesa posredoval vse procesne podatke, ki so osnova za nove informacije, s pomočjo katerih lahko lastnik opravila opravilo izvede. Uporabniška opravila naj bodo zasnovana tako, da bo na podlagi njihove opredelitve omogočena preslikava vhodnih podatkov v koncepte organizacijske ontologije v izvajalnem času.

Za omogočanje avtomatizacije uporabniških opravil je potrebna opredelitev pričakovanih rezultatov uporabniškega opravila na način, ki bo omogočal avtomatsko izvajanje opravila z namenom doseganja opredeljenih rezultatov. Pričakovani rezultati so lahko opredeljeni statično (z uporabniškim opravilom) ali dinamično (so pogojeni s posameznimi primerki uporabniškega opravila). Uporabniško opravilo naj bo zasnovano tako, da bo podpiralo takšno opredelitev rezultatov.

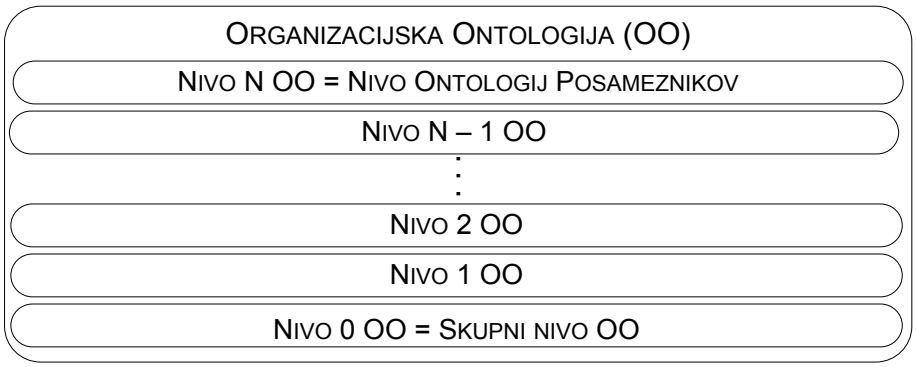
Za izvedbo sodelovalnih opravil je poleg tega potrebno v uporabniškem opravilu določiti, kateri protokol določa njegovo izvajanje in kdo so osebe, ki igrajo posamezne vloge v protokolu sodelovanja. Za omogočanje avtomatizacije sodelovalnih opravil naj bo določitev

protokola in vlog podprta na način, ki bo agentom SIUO omogočal odkrivanje ustreznega protokola iz knjižnice protokolov organizacijske ontologije in implicitno dodelitev vlog posameznim agentom, ki osebe zastopajo.

Organizacijska ontologija se lahko spreminja na nivoju primerkov in na konceptualnem nivoju. Potrebno je opredeliti, kako razvoj in izvajanje uporabniških opravil kot dela poslovnih procesov vplivata na spremembe v organizacijski ontologiji, ter kako te spremembe sovpadajo s pristopi in metodologijami razvoja sistemov za obvladovanje znanja, ki temeljijo na ontologijah.

8.2 Struktura organizacijske ontologije

Slika 14 ponazarja osnovno strukturo organizacijske ontologije. Organizacijska ontologija je strukturirana hierarhično. Hierarhična struktura je predlagana zaradi prednosti, ki jih omogoča, predvsem zaradi zmanjšanja kompleksnosti iskanja in doseganja višje ravni ponovne uporabe shranjenih informacij [46].



Slika 14: Struktura organizacijske ontologije

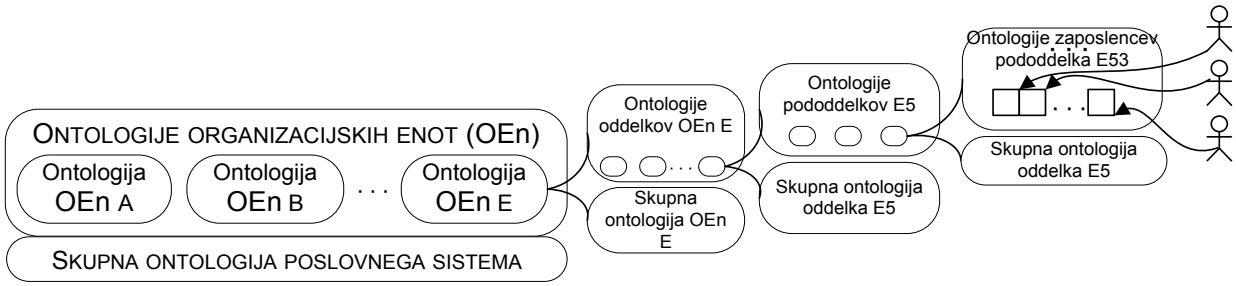
Hierarhičnost organizacijske ontologije pomeni, da lahko pod določenimi pogoji ontologija višjega nivoja uporablja ontologijo nižjega nivoja. Hierarhičnost je omogočena s konceptom uvoza, ki je del specifikacij jezika OWL (*owl:imports*). OWL ontologija lahko uvozi druge ontologije in s tem pridobi dostop do entitet, izrazov in aksiomov te ontologije. S tem koncept uvoza predstavlja osnovo za modularizacijo OWL ontologij [90].

Poimenujmo posamezen element organizacijske strukture poslovnega sistema *organizacijski element* (*OE*). Organizacijski element lahko tako predstavlja celoten poslovni sistem, oddelek, vlogo, skupino ali katerikoli drug možen element, ki se pojavlja v organizaciji poslovnega sistema. Organizacijska ontologija je sestavljena iz ontologij, ki pripadajo različnim organizacijskim elementom. Vsaka ontologija organizacijske ontologije pripada natanko enemu organizacijskemu elementu. Pogoj, za razvrščanje ontologij v različne nivoje je naslednji:

$$\check{C}e\ je\ OE_1\ del\ OE_2,\ OE_1\ lahko\ uporablja\ skupno\ ontologijo\ OE_2.$$

Na primer, če je OE_1 vloga, ki deluje v oddelku OE_2, potem lahko uporablja informacije zajete v skupni ontologiji oddelka OE_2. Skupni nivo organizacijske ontologije predstavlja skupne informacije poslovnega sistema kot celote, na primer poslovno politiko in organizacijsko strukturo poslovnega sistema. Najvišji nivo organizacijske ontologije zajema informacije posameznikov v poslovnem sistemu.

Slika 15 ponazarja primer hierarhične strukture organizacijske ontologije poslovnega sistema, ki se na najvišjem nivoju deli na pet organizacijskih enot (A,B,C,D in E), organizacijske enote se delijo na oddelke in ti na pododdelke. Slika na vsakem nivoju prikazuje delitev ene ontologije. Pri tem je potrebno poudariti, da gre zgolj za primer, in da je delitev le redko tako preprosta. Na primer v primeru matrične organizacijske strukture, je hierarhična struktura bolj zapletena.



Slika 15: Primer strukture organizacijske ontologije

8.3 Vzpostavitev strukture organizacijske ontologije

Struktura organizacijske ontologije je torej odvisna od dejanske organizacije poslovnega sistema. Posledično je od nje odvisna tudi sama vzpostavitev organizacijske ontologije. V nadaljevanju so podani koraki vzpostavitve organizacijske ontologije:

1 Postavitev ontologije temeljne organizacijske strukture

Temelj organizacijske strukture vključuje razvoj osnovnih organizacijskih konceptov in vlog v ontologiji. Ti osnovni koncepti so naslednji:

- Osnovna razreda OE in $Oseba$:

  Razred OE je potreben kot osnovni nadrazred za vse organizacijske elemente, ki se v poslovnem sistemu pojavljajo. Razred Oseba je potreben za vzpostavite povezav med osebami, zaposlenimi v poslovnem sistemu in OE, v katerih delujejo.

- Osnovni objektni lastnosti $pripada$ in $jeSestavljenIz$, pri katerih je razred OE tako domena kot zaloga vrednosti:

  Če sta dva primerka razreda OE povezana z lastnostjo $pripada$, to pomeni, da je prvi organizacijski element del drugega oziroma da mu pripada; lastnost $jeSestavljenIz$ je inverzna lastnost lastnosti *pripada*.

2 Opredelitev tipov OE

Korak zahteva identifikacijo in implementacijo vseh organizacijskih elementov kot podrazredov razreda OE, njihovo razvrstitev v ustrezno taksonomijo, opredelitev vseh potrebnih podlastnosti lastnosti $pripada$ in $jeSestavljenIz$ skupaj z določanjem njihovih domen in zalog vrednosti z na novo opredeljenimi koncepti. Primeri podrazredov OE so $PoslovniSistem$, $Oddelek$, $Poddodelek$ in $Vloga$, pri čemer na primer pododdelek neposredno pripada oddelku, vloga pa lahko neposredno pripada pododdelku, oddelku in poslovnemu sistemu kot celoti. Primer vloge, ki pripada oddelku, je vloga administrator podatkovnih baz, ki pripada oddelku Informatika. Primer vloge, ki neposredno pripada poslovnemu sistemu kot celoti, je vloga direktor poslovnega sistema, ki ni del nobenega podrazreda OE.

Poleg tega je potrebno opredeliti tudi objektno lastnost, ki povezuje domenski razred $Oseba$ z razredom zaloge vrednosti, ki je eden izmed opredeljenih podrazredov razreda OE. Najpogostejši primer takšne lastnosti je $imaVlogo$ z razredom zaloge vrednosti $Vloga$.

3 Implementacija ontologije organizacijske strukture

Korak zajema razvoj primerkov, ki spadajo v predhodnem koraku opredeljene razrede, in povezav med njimi. Če je poslovni sistem razdeljen v tri organizacijske enote A, B in C, je primer ene izmed aktivnosti tega koraka dodajanje treh primerkov, ki predstavljajo posamezne organizacijske enote, v razred, ki predstavlja organizacijske enote.

4 Izdelava ogrodja organizacijske ontologije

Korak lahko opišemo z naslednjimi točkami:

T1. Za $\forall x \in OE$ izdelaj prazno ontologijo.

T2. Za $\forall x \in OE$:

Za $\forall y: (x, y) \in jeSestavljenIz$ dodaj uvoz ontologije organizacijskega elementa x v ontologijo organizacijskega elementa y.

8.4 Preslikava nosilcev informacij uporabniškega opravila v raven primerkov organizacijske ontologije

8.4.1 Preslikava med nosilci informacij podatkov uporabniškega opravila in organizacijsko ontologijo

Podatek je podlaga informacije. Podatek je golo dejstvo, simbol za neko vsebino opredeljen brez širšega konteksta. Informacija je že obdelan podatek, ki izraža vrednostni odnos uporabnika do podatka [44]. Vhodni podatki v opravilo bi naj vsebovali vse potrebne procesne podatke, ki so podlaga informacijam potrebnim za izvedbo opravila. Za to, da jih lahko uporabimo skupaj z informacijami v organizacijski ontologiji in nad njimi izvajamo sklepanje, jih je potrebno pravilno umestiti v kontekst ontologije lastnika opravila. Ti podatki predstavljajo raven primerkov v ontologiji. Potrebno je natančno opredeliti preslikavo med

vhodnimi podatki na eni strani in primerki ter lastnostmi med njimi v organizacijski ontologiji na drugi strani. Cilj opredelitve preslikave je avtomatska pretvorba vhodnih podatkov v koncepte ontologije, ki omogoča združitev novih informacij z obstoječimi. Pri tem se nanašamo na ontologijo, ki pripada lastniku opravila, bodisi je to posameznik, skupina posameznikov, oddelek itd. Med izvajanjem sodelovalnega opravila postane del ontologije lastnika opravila tudi ontologija ustreznega udeleženca protokola. Preslikava se zato izvede, ko je opredeljen konkreten lastnik opravila.

XML shema, ki podaja strukturo vhodnih podatkov, in konceptualna raven ontologije predstavljata podlago za opredelitev preslikave. Dele vhodnih podatkov, ki predstavljajo vhod v preslikavo v dani koncept ontologije, imenujemo nosilci informacij procesnih podatkov. Nosilec informacije je lahko katerikoli del vhodnih podatkov, na primer posamezen element, pojav dveh elementov, pojav elementa v drugem elementu itd. Predpostavljamo, da pri opredelitvi preslikav delamo z že obstoječo ontologijo, v kateri so na konceptualnem nivoju opredeljeni vsi razredi in lastnosti, ki so zaloga vrednosti preslikav. Primere, ko pri opredelitvi preslikave ugotovimo, da določen koncept v ontologiji ne obstaja, obravnava poglavje 8.6 Spremembe konceptualne ravni organizacijske ontologije, ki izvirajo iz opredelitve in izvajanja uporabniških opravil.

Z vidika organizacijske ontologije lahko za določene koncepte ontologije avtomatsko polnjenje s primerki in njihovimi lastnostmi ni zaželeno, saj predstavljajo kritične informacije. Čeprav je preslikava pravilno opredeljena, lahko nastopijo različni dejavniki, ki informacije naredijo manj zanesljive, na primer morebitne napake pri pridobivanju podatkov v poslovnem procesu. Napačne informacije bi lahko imele negativne posledice za poslovni sistem, zato je za kritične koncepte v ontologiji, pred samim vnosom novih primerkov v ontologijo, potrebna potrditev s strani vloge upravljavec znanja za domeno, v katero spada kritični koncept. Kritičnost se nanaša na organizacijsko ontologijo in ne na posamezno uporabniško opravilo, zato se določi v organizacijski ontologiji: za posamezne koncepte ontologije, na primer z anotacijsko lastnostjo, ali za posamezne ontologije v celoti.

Včasih so informacije koristne samo za tekoče uporabniško opravilo, medtem ko njihovo trajno hranjenje v organizacijski ontologiji ni potrebno. V teh primerih naj se vhodni podatki preslikajo v ontologijo, se združijo z informacijami v obstoječi ontologiji in se uporabijo za sklepanje, po končanem opravilu pa se ne shranijo v ontologijo. V primeru, če se kasneje pojavi potreba, po analizi rezultatov posameznih opravil, lahko na podlagi sledi poslovnega

procesa pridobimo vhodne podatke posredovane SIUO ter s pomočjo preslikave ponovno ugotovimo, zakaj je prišlo do določenega sklepa. Ker je pojav vezan na posamezna uporabniška opravila, se zahteva opredeli skupaj s preslikavo.

V primeru, da v ontologiji že obstaja informacija, ki ni skladna z novo informacijo, lahko nova informacija bodisi zamenja predhodno informacijo, bodisi se nova informacija zavrže. Lahko je zaželena tudi potrditev zamenjave z novo informacijo s strani upravljavca znanja domene. Potrditev je za upravljavca znanja opravilo, ki je sorodno opravilu potrditve vnosa kritičnih informacij. Pri tem gre za informacije, ki se nanašajo na podatkovne ali objektne lastnosti. Pravila o tem, ali naj se določena informacija zavrže ali ne, so lahko odvisna od posameznih konceptov ontologije, od vira informacije ali kombinacije obojega. Tako se lahko opredelijo v posameznih konceptih ontologije, na primer z uporabo anotacijskih lastnosti, na ravni posameznih ontologij kot celote, v uporabniških opravilih ali drugih morebitnih virih. V primeru uporabniških opravil se lahko opredelijo za uporabniško opravilo kot celoto, za posamezno preslikavo nosilcev informacij procesnih podatkov v ontologijo, za posamezen sklop uporabniških opravil ali na globalnem nivoju SIUO za vsa uporabniška opravila. V primeru opredelitve pravil za različne vire, kot so uporabniška opravila, se pravila sicer lahko opredelijo v sami organizacijski ontologiji, vendar bi to lahko pomenilo visoko kompleksnost varnostnih mehanizmov, ki bi razlikovala med klici SOO iz različnih uporabniških opravil. Zaradi tega za primere uporabniških opravil, ki so notranji poslovnemu sistemu, predlagamo opredelitev tovrstnih pravil v okviru uporabniških opravil.

Pri opredelitvi pravil preslikave je potrebna posebna pazljivost in skrb za ohranjanje konsistentnosti celotne organizacijske ontologije. Preslikava iz procesnih podatkov v ontologijo je lahko opredeljena na različnih nivojih granularnosti:

- na globalnem nivoju storitve SIUO:

  Na nivoju SIUO so opredeljena pravila, ki za vsa uporabniška opravila določajo, kateri elementi procesnih podatkov se preslikajo v katere koncepte ontologije. Pravila, opredeljena na ta način, zahtevajo manj dela, predvsem v primerjavi z opredeljevanjem pravil za vsako uporabniško opravilo. Vendar pa zahtevajo višjo mero discipline pri generiranju XML shem vhodnih podatkov, saj bi napačno zasnovana shema pomenila napačno preslikavo podatkov. Posledično je izdelava XML shem izpostavljena omejitvam in manj fleksibilna. V primeru, da se zaradi omejitev, ustrezne sheme ne da izdelati, bi bilo potrebno kompromiranje bodisi organizacijske ontologije, bodisi XML sheme, ki bi

tako lahko podpirala nepravilne strukture podatkov. Če ne eno ne drugo ne bi bilo sprejemljivo, bi bilo potrebno spremeniti pravila ali organizacijsko ontologijo, kar lahko povzroči verižno reakcijo zahtevanih sprememb v drugih XML shemah in organizacijski ontologiji.

Druga pomanjkljivost pristopa izhaja iz dejstva, da lahko ima določen podatek za različne posameznike in v različnih kontekstih različen pomen. S pravili opredeljenimi na globalnem nivoju SIUO, je to težje doseči. V določenih primerih se v ta namen lahko prilagodi sama XML shema podatkov. S tem posledično prihaja do novih omejitev ter do povečane kompleksnosti. Tak pristop je zato primeren zgolj za manjše ontologije ali v kombinaciji z drugimi pristopi.

- posamezno uporabniško opravilo:

  Pravila, ki določajo preslikavo na podlagi XML sheme elementov vhodnih podatkov v ontologijo, so določena v posameznem uporabniškem opravilu. Pravila so prilagojena posamezni shemi, zaradi česar je mogoča večja fleksibilnost preslikav. Pristop je semantično bogatejši, saj omogoča opredeljevanje informacij glede na posameznega lastnika opravila in glede na kontekst, v katerem se informacije pojavljajo.

- sklopi uporabniških opravil:

  Če več uporabniških opravil uporablja vhodne podatke na podlagi istih elementov XML shem, lahko pravila preslikave določimo za sklop uporabniških opravil. Pristop je prilagojen posamezni shemi, zaradi česar je v primerjavi s pravili opredeljenimi na globalnem nivoju mogoča večja fleksibilnost preslikav. V primerjavi z opredelitvijo pravil za posamezna opravila, je upoštevanje konteksta in lastnikov opravil na nižji ravni, medtem ko je opredeljevanje preslikav nekoliko poenostavljeno, saj jih ni potrebno določati za vsako opravilo posebej.

Skladno s prednostmi in pomanjkljivostmi različnih granularnosti opredeljevanja pravil preslikav predlagamo naslednji pristop:

1) Pravila se opredelijo za posamezno uporabniško opravilo, v primerih če:

- se XML shema procesnih podatkov ne uporablja v drugih opravilih in/ali
- kadar je preslikava specifična za kontekst oziroma lastnika opravila;

2) Če obstaja sklop uporabniških opravil, ki uporabljajo procesne podatke na podlagi istih elementov XML shem in preslikava za celoten sklop uporabniških opravil omogoča dovolj specifike glede na kontekst in lastnike opravil, se pravila opredelijo za ustrezen sklop uporabniških opravil. Pri določanju preslikave posameznega uporabniškega opravila v sklopu se opredelitev opravila v SIUO sklicuje na preslikavo sklopa.

3) Opcijsko lahko opredelimo tudi pravila na globalnem nivoju. Namenjena so za primere, v katerih uporabniško opravilo ni del nobenega sklopa uporabniških opravil, za katerega bi bila opredeljena preslikava, niti uporabniško opravilo samo ne opredeljuje preslikave. Gre za koncept privzetih preslikav opravila, zato je pri njihovi uporabi potrebna posebna pazljivost. Ker je za koncepte ontologije opredeljeno, če je potrebna posebna potrditev spremembe s strani upravljavca znanja domene, je s tem dosežena dodatna varnost, če privzeta pravila v vseh primerih ne opredeljujejo ustrezne preslikave. V primerih zahtevane potrditve, delujejo kot predlog upravljavcu znanja domene.

Tako je lahko z vidika posameznega uporabniškega opravila preslikava opredeljena bodisi kot del uporabniškega opravila, bodisi je v uporabniškem opravilu podan sklic na preslikavo bodisi preslikava ni opredeljena. Če preslikava ni opredeljena, to pomeni, da se uporabijo pravila globalnega nivoja oziroma, če pravil globalnega nivoja ne uporabljamo, da se vhodni podatki ne preslikajo v ontologijo. V zadnjem primeru se opravilo izvaja na podlagi informacij, ki že obstajajo v organizacijski ontologiji in informacij lastnika opravila, če opravilo ni avtomatizirano.

Posamezen nosilec informacije v vhodnih podatkih lahko predstavlja različne primerke oziroma lastnosti v organizacijski ontologiji. Pri tem se zgledujemo po pristopu JXML2OWL [64], ki smo ga posplošili in dopolnili za specifiko domene. Cilj preslikave je lahko primerek, podatkovna ali objektna lastnost. Z vidika ciljnega koncepta ontologije ločimo naslednje tipe preslikav:

- Preslikava v nov primerek:

  Nosilec informacije v vhodnih podatkih predstavlja nov primerek v ontologiji. Za preslikavo je potrebno opredeliti naslednje:

  - kateri del vhodnega sporočila pomeni nov primerek v ontologiji (je nosilec informacije),

- poimenovanje primerka,
- OWL razred, katerega element bo nov primerek.

Opredelimo lahko več kot eden OWL razred, ki jim nov primerek pripada.

Če je delov vhodnega sporočila, ki identificirajo nove primerke v ontologiji več, je potrebno identificirati imena za vsakega izmed njih. Na primer, če gre za več elementov posameznega vozlišča, se ime posameznih primerkov nanaša na posamezne elemente vozlišča.

Vsi primerki bodo umeščeni v vse podane OWL razrede.

- Preslikava v podatkovno lastnost:

  Nosilec informacije lahko predstavlja nov primerek podatkovne lastnosti, pri čemer primerek oziroma primerki, ki jim določimo lastnost, v ontologiji lahko že obstajajo ali ne. Za preslikavo v podatkovno lastnost je potrebno opredeliti naslednje:

  - OWL primerek, ki pripada domenskemu razredu podatkovne lastnosti,
  - podatek, ki opisuje podatkovno lastnost primerka,
  - OWL podatkovno lastnost.

  OWL primerek določimo v odvisnosti od tega ali v ontologiji že obstaja ali ne:

  - Če OWL primerek v ontologiji še ne obstaja, ga določimo s preslikavo v nov primerek;
  - Če OWL primerek v ontologiji že obstaja, ga identificiramo na enega izmed naslednjih načinov:
    - s pomočjo nosilcev informacij v vhodnih podatkih:

      Primer je, ko vhodni podatki opisujejo podatke v zvezi z novimi izdelki blagovne znamke, pri čemer je ime blagovne znamke podano kot del vhodnih podatkov. S pomočjo vhodnih podatkov, ki opisujejo blagovno

znamko, lahko določimo ime OWL primerka, ki jo predstavlja v ontologiji.

- s konkretnim poimenovanjem OWL primerkov:

  Primer je, ko vhodni podatki opisujejo število izpadov omrežne infrastrukture v zadnjem letu, pri čemer podatek o infrastrukturi ni podan kot del vhodnih podatkov, saj je privzeto, da lastnik opravila ve, za katero infrastrukturo gre. V tem primeru kot OWL primerek opredelimo s poimenovanjem tistega primerka, ki predstavlja omrežno infrastrukturo.

Če je na ta način identificiranih več primerkov, je potrebno v vhodnih podatkih identificirati podatke, ki opisujejo podatkovno lastnost, za vsakega izmed primerkov.

Podatek, ki opisuje podatkovno lastnost primerka, je določen z določenim vhodnim podatkom.

OWL podatkovno lastnost določimo z njenim eksplicitnim poimenovanjem.

- Preslikava v objektno lastnost:

  Nosilec informacije lahko predstavlja novo objektne lastnosti med dvema primerkoma. Za preslikavo v objektno lastnost je potrebno opredeliti naslednje:

  - OWL primerek, ki pripada domenskemu razredu objektne lastnosti,
  - OWL primerek, ki pripada razredu zaloge vrednosti objektne lastnosti,
  - OWL objektno lastnost.

  Vsakega izmed OWL primerkov lahko določimo na enak način, kot pri določanju OWL primerka, ki pripada domenskemu razredu podatkovne lastnosti pri preslikavi v podatkovno lastnost:

  - Če OWL primerek v ontologiji še ne obstaja, ga določimo s preslikavo v nov primerek.

- Če OWL primerek v ontologiji že obstaja, ga identificiramo s pomočjo nosilcev informacij v vhodnih podatkih ali s konkretnim poimenovanjem OWL primerkov.

Navadno je znano, ali vhodni podatki opisujejo obstoječe ali nove primerke. Določeni OWL razredi ne dovoljujejo dodajanja novih primerkov v poljubnih uporabniških opravilih, ampak samo v določenih, v katerih ima lastnik opravila ustrezna pooblastila. Primer so razredi, ki vsebujejo v poslovnem sistemu vnaprej določene koncepte, kot so organizacijske enote ali tuji jeziki. V primeru, da ni znano, ali so primerki že obstoječi ali ne, predpostavimo da gre za nove primerke. Če se pri preslikavi izkaže, da primerki niso novi, se pri opredeljevanju lastnosti uporabijo že obstoječi primerki. Preslikava v nov primerek, ki že obstaja, se ignorira.

8.4.2 Poizvedbe organizacijske ontologije na podlagi opredelitve uporabniškega opravila

Eden izmed pomembnejših elementov uporabniškega opravila so predstavitveni elementi (*presentation elements*), ki podajajo ime, temo in opis opravila, in so namenjeni prikazu opravila njegovemu lastniku [5]. Skupaj z vhodnimi podatki predstavljajo osnovo za predstavitev opravila njegovemu lastniku. Za to, da bi lahko podprli avtomatsko izvajanje uporabniškega opravila s pomočjo sklepanja nad organizacijsko ontologijo, je potrebno v uporabniškem opravilu z jezikom ontologije opredeliti, kaj je pričakovani rezultat. Ker je ontologija sredstvo za zajemanje skupnega razumevanja izrazov, ki jih lahko uporabljajo tako ljudje kot programi [46], se na ta način opredeljen pričakovani rezultat lahko uporabi tako za morebitno avtomatsko izvedbo opravila kot za osnovo za predstavitev opredelitve pričakovanega rezultata v predstavitvi opravila njegovemu lastniku. Zato se lahko uporablja skupaj s predstavitvenimi elementi.

Opis pričakovanega rezultata lahko razumemo kot vprašanje, na katerega bo rezultat podal odgovor, ali kot na izjavo, ki jo poskušamo zadovoljiti oziroma potrditi njeno resničnost. Izvedbo opravila lahko torej razumemo kot iskanje odgovora na vprašanje ali kot poskus potrjevanja izjave. Primer opravila, ki ga lahko opišemo s prispodobo iskanja odgovora na vprašanje, je opravilo, v katerem lastnik opravila skuša odgovoriti na vprašanje »Kdo je najboljši kandidat za zaposlitev na delovno mesto XY?« na podlagi vhodnih podatkov, ki

opisujejo kandidate za zaposlitev. Primer opravila, ki ga lahko opišemo s prispodobo potrjevanja resničnosti izjave, je opravilo, v katerem lastnik opravila skuša potrditi izjavo: »Izdelek A bo dostavljen stranki S do dne 1.2.2009.« na podlagi vhodnih podatkov, ki podajajo stranko, izdelek in naslov stranke. V primeru, da z izvedbo opravila odgovor na vprašanje ni najden oziroma da izjava ni potrjena, je opravilo neuspešno.

Vsak izmed navedenih načinov opisa pričakovanega rezultata se lahko preoblikuje v enakovreden opis drugega tipa: vprašanje lahko preoblikujemo v izjavo ter izjavo lahko preoblikujemo v vprašanje. Zato v OWL ontologiji tako vprašanje kot izjavo predstavimo s poizvedbo nad ontologijo.

V primeru vprašanja »Kdo je najboljši kandidat za zaposlitev na delovno mesto XY?« gre za miselno opravilo. Če je način, kako lastnik opravila pride do rezultata, zajet v ontologiji, bi s poizvedbo nad organizacijsko ontologijo prišli do rezultata. V primeru izjave »Izdelek A bo dostavljen stranki S do dne 1.2.2009.« gre za fizično opravilo. Če izdelek ni bil dostavljen k stranki oziroma če ontologija ne vsebuje informacij o tem, poizvedba nad ontologijo ne bo vrnila pozitivnega rezultata. Zaradi tega bo kreiran nov primerek uporabniškega opravila, ki bo posredovan ustreznemu lastniku opravila. Ko bo izdelek dostavljen, bo lastnik opravila potrdil, da je opravilo izvedeno. Če je opredeljena preslikava med rezultatom opravila in organizacijsko ontologijo, se bo informacija o tem vnesla v ontologijo. S tem bosta izjava in tako rezultat opravila potrjena.

V obeh primerih gre torej za poizvedbo nad organizacijsko ontologijo, ki lahko vključuje določene elemente iz vhodnih podatkov uporabniškega opravila, na primer identifikator izdelka. Uporabniško opravilo mora tako opredeliti poizvedbo, ki opisuje pričakovani rezultat, in podatek o tem, ali se rezultat hrani v ontologijo ali ne.

8.4.3 Predlog razširitve specifikacij WS-HumanTask

Specifikacije WS-HumanTask [5] opredeljujejo, kako je sestavljeno uporabniško opravilo in kako se povezuje s vhodnimi ter izhodnimi podatki pri klicu storitve, ki uporabniško opravilo implementira. Na podlagi konceptov opredeljenih v predhodnih poglavjih 8.4.1 in 8.4.2, je potrebno razširiti specifikacije opredelitve uporabniškega opravila, tako da:

- bo definicija uporabniškega opravila omogočala opredelitev preslikave nosilcev informacij procesnih podatkov v primerke organizacijske ontologije ali opredelitev sklica na tovrstno preslikavo;
- bo moč opredeliti preslikavo nosilcev informacij procesnih podatkov v primerke organizacijske ontologije na ravni posameznega sklopa uporabniških opravil;
- bo moč opredeliti globalna pravila preslikave nosilcev informacij procesnih podatkov v primerke organizacijske ontologije, ki veljajo v primerih, ko za uporabniško opravilo ni opredeljena preslikava niti na nivoju opravila niti za sklop opravil, v katerega spada;
- bo moč opredeliti pravila, ki določajo, ali se v primeru, da v ontologiji že obstaja informacija, ki ni skladna z novo informacijo, nova informacija zavrže ali zamenja predhodno informacijo, in sicer na naslednjih nivojih granularnosti: preslikava v podatkovno ali objektno lastnost, uporabniško opravilo, sklop uporabniških opravil ali globalni nivo SIOU;
- bo moč opredeliti pričakovani rezultat uporabniškega opravila,
- bo za sodelovalna opravila moč opredeliti protokol sodelovanja in določiti povezave med vlogami protokola ter lastniki opravila.

Osnovni element specifikacij WS-HumanTask, ki podpira opredelitev človeških interakcij, je element <*humanInteractions*>. V tem elementu so opredeljena uporabniška opravila in notifikacije.

Za namen opredelitve razširitev uporabljamo sintakso, ki je pogosta v različnih specifikacijah, na primer [4],[5],[57],[95] in drugih. Sintaksa opisuje XML besednjak posameznik fragmentov XML, in sicer [57]:

- Sintaksa je prikazana kot primerek XML, kjer so namesto vrednosti podani podatkovni tipi.
- Deli v poudarjenem tisku predhodno še niso bili uvedeni ali prikazujejo posebej pomembne dele primera.
- <-- opis --> je polje elementov iz imenskega prostora »other«, na primer ##other v XSD,

- Posameznim elementom in atributom so pripeti znaki, in sicer:
    - »?« - 0 ali 1 element/atribut,
    - »*« - 0 ali več elementov/atributov,
    - »+« - 1 ali več elementov/atributov.
    - Poleg navedenih oznak dodajamo, da če elementu oziroma atributu ne sledi nobeden izmed znakov »?«, »*« in »+«, to pomeni, da mora biti prisoten natanko eden element oziroma atribut.
- Znaka »[« in »]« označujeta, da je elemente in atribute, ki se nahajajo med njima, potrebno obravnavati kot skupino glede na znake »?«, »*« in »+«.
- Elementi in atributi, ki so ločeni z znakom | in grupirani z »(» in »)« pomenijo sintaktične alternative.
- Predpone imenskih prostorov XML označujejo imenske prostore opredeljenega elementa.

Sintaksa razširjenih specifikacij elementa <*humanInteractions*> je podana v spodnji tabeli (Tabela 6). S poudarjenim tistkom so prikazani elementi in atributi razširitve.

```
<?xml version="1.0" encoding="UTF-8"?>
<htd:humanInteractions ...
      ...
      xmlns:htoo = "http://www.fri.uni-lj.si/siuo/oo">
      ...
      <htoo:ontologyTranslationElements>?
      ...
      </htoo:ontologyTranslationElements>

      <htd:extensions>
        <htd:extension namespace="http://www.fri.uni-lj.si/siuo/oo"
mustUnderstand="yes"/>
      </htd:extensions>
```

```
<htd:import namespace="anyURI"?
        location="anyURI"?
        importType="anyURI" />*

 <htd:logicalPeopleGroups>?

   …

</htd:logicalPeopleGroups>

<htd:tasks>?

   <htoo:ontologyTranslationElements>?
        ...
   </htoo:ontologyTranslationElements>

   <htoo: taskGroupOntologyTranslationElements>*
     ...
   </htoo: taskGroupOntologyTranslationElements>


   <htd:task name="NCName">+
        [<htoo:ontologyTranslationElements>
        ...
        </htoo:ontologyTranslationElements> |

        <htoo: taskGroupOntologyTranslationName>
        ...
        </htoo: taskGroupOntologyTranslationName>]?

        <htoo:ontologyOutputQuery >+
        ...
        </htoo:ontologyOutputQuery>

        <htoo:protocolName>?
        ...
        </htoo:protocolName >


        <htd:interface ...
        ...

        <htd:peopleAssignments>
            <htd:potentialOwners> +
               <htoo:protocolRoleName>?
               ...
```

```
                    </htoo:protocolRoleName>
                    <htd:from>
                    ...
                    <htd:from>
                  <htd:potentialOwners> +

        </htd:task>

      </htd:tasks>

       <htd:notifications>?
            ...
       </htd:notifications>
    </htd:humanInteractions>
```

Tabela 6: Sintaksa razširjenih specifikacij WS-HumanTask

Razširitev zajema naslednje:

- XML imenski prostor razširitve:

  Opredelitev človeških interakcij mora za opredelitev preslikav nosilcev informacij uporabniških opravil v koncepte ravni primerkov ontologije uporabljati elemente razširjenih specifikacij. V ta namen uporabimo mehanizem razširjanja z XML domenskim prostorom, ki omogoča razširitev specifikacij WS-HumanTask s povratnim ohranjanjem skladnosti. To je pomembna lastnost, saj predlagane razširitve ne kršijo skladnosti s specifikacijami WS-HumanTask in omogočajo uporabo orodij, ki podpirajo WS-HumanTask in razširitve ne poznajo. V ta namen smo uvedli URI domenskega prostora *http://www.fri.uni-lj.si/siuo/oo*, za katerega uporabljamo okrajšavo *htoo*.

- Element <*htoo:ontologyTranslationElements*> :

  Element <*htoo:ontologyTranslationElements*> vsebuje pravila preslikave nosilcev informacij procesnih podatkov v primerke organizacijske ontologije in je lahko opredeljen kot:

  - podelement elementa <*htd:humanInteractions*> za globalno opredeljene preslikave;
  - podelement elementa <*htd:task*> za preslikave, ki se nanašajo na posamezno opravilo;

- podelement elementa <*htoo:taskGroupOntologyTranslationElements*> za preslikave, ki se nanašajo na posamezen sklop opravil.

V primeru, da so za dano uporabniško opravilo opredeljene preslikave na več ravneh hkrati (globalno, sklop uporabniških opravil, posamezno uporabniško opravilo), se upoštevajo elementi <*htoo:ontologyTranslationElements*> nižjih ravni, ostali se ignorirajo:

- Če so podani elementi na vseh treh ravneh, se globalni element preslikave in element preslikave v sklopu uporabniških opravil ignorirata; upošteva se element preslikave, ki je del opredelitve danega uporabniškega opravila.
- Če sta podana elementa globalne ravni in sklopa uporabniških opravil, se za dano uporabniško opravilo upošteva element preslikave v sklopu uporabniških opravil, v katerega spada.
- Če sta podana elementa preslikave v sklopu uporabniških opravil in v danem uporabniškem opravilu, se element v sklopu ignorira in se upošteva element preslikave v danem uporabniškem opravilu.
  Določanje preslikav v elementu preslikav je podrobneje opisano v poglavju Sintaksa opredelitve preslikav med nosilci informacij procesnih podatkov in primerki organizacijske ontologije.

- Element <*htoo: taskGroupOntologyTranslationName*>:

  Element se pojavlja znotraj elementa opravila <*hdt:task*> in predstavlja sklic na preslikavo, ki je opredeljena za sklop uporabniških opravil. Predstavlja alternativni element elementu <*htoo:ontologyTranslationElements*>.

- Element <*htoo: taskGroupOntologyTranslationElements*>*:*

  Element je namenjen opredeljevanju preslikav za sklop uporabniških opravil. Sintaksa elementa je podrobneje opisana v poglavju Element preslikave za sklop uporabniških opravil.

- Element **<***htoo:ontologyOutputQuery*>:

Element je namenjen določanju pričakovanega rezultata z opredelitvijo poizvedbe za organizacijsko ontologijo. Je obvezen element in se določa kot podelement posameznega uporabniškega opravila.

- Element <*htoo:protocolName*>:

  Če gre za sodelovalno opravilo, s tem elementom podamo ime protokola.

- Element <*htoo:protocolRoleName*>:

  Če gre za sodelovalno opravilo, je potrebno določiti poleg potencialnih lastnikov tudi vlogo, ki jo igrajo pri sodelovanju. Pri tem morajo vloge, ki jih dodelimo potencialnim lastnikom, ustrezati vlogam opredeljenim s samim protokolom, in sicer mora biti za vsako vlogo protokola opredeljena skupina potencialnih lastnikov. Protokol, na katerega se vloge nanašajo, je določen v elementu <*htoo:protocolName*>. Primer določitve potencialnih lastnikov vlogam Iniciator in Udelezenec protokola je podan v spodnji tabeli (Tabela 7).

```
<htd:potentialOwners>
        <htoo:protocolRoleName>Iniciator </htoo:protocolRoleName>
        <htd:from>
           htd:getInput("ZahtevaIzbireIzvajalca")/odgovornaOseba
        </htd:from>
     </htd:potentialOwners>
     <htd:potentialOwners>
        <htoo:protocolRoleName>Udelezenec </htoo:protocolRoleName>
        <htd:from>
           htd:getInput("ZahtevaIzbireIzvajalca")/usposobljeniZaposlenci
        </htd:from>
</htd:potentialOwners>
```

Tabela 7: Primer določitve potencialnih lastnikov vlogam protokola

Sintaksa opredelitve preslikav med nosilci informacij procesnih podatkov in primerki organizacijske ontologije

Sintakso elementa *<htoo:ontologyTranslationElements>*, ki opredeljuje preslikave med procesnimi podatki in koncepti organizacijske ontologije podaja spodnja tabela (Tabela 8).

```
<htoo:ontologyTranslationElements queryLanguage="anyURI"?>
  <htoo:HTOOQueryParameters>?
    < htoo:HTOOQueryParameter name=”NCName” type=”xsd:string”> +
      izraz
    < /htoo:HTOOQueryParameter>
  </htoo:HTOOQueryParameters>
  <htoo:translateToOWLIndividual>*
    <htoo:elementForOWLIndividual type="QName">
      izraz
    </htoo:elementForOWLIndividual>
    <htoo:IDForOWLIndividual type="QName">
      izraz
    </htoo:IDForOWLIndividual>
    <htoo:OWLClass type="anyURI">
      URI
    </htoo:OWLClass>
  </htoo:translateToOWLIndividual>

  <htoo:translateToOWLDatatypeProperty>*
    <htoo:toODPWithNewIndividual >
        <htoo:elementForOWLIndividual type="QName">
          izraz
        </htoo:elementForOWLIndividual>
        <htoo:IDForOWLIndividual type="QName">
          izraz
        </htoo:IDForOWLIndividual>
        <htoo:OWLClass type="anyURI">
          URI
```

```
        </htoo:OWLClass>

    </htoo:toODPWithNewIndividual > |

    <htoo:referenceToTranslationToIndividual name=”QName” /> |

    <htoo:IDOfExistingOWLIndividualFromInput type="QName">
            izraz
    </htoo: IDOfExistingOWLIndividualFromInput > |

    <htoo:OWLIndividual type="anyURI">
       URI
    </htoo:OWLIndividual>


    <htoo:OWLDatatypeProperty type="anyURI">
       URI
    </htoo:OWLDatatypeProperty>
    <htoo:datatypeValue type="QName">
       izraz
    </htoo:datatypeValue>
</htoo:translateToOWLDatatypeProperty>

<htoo:translateToOWLObjectProperty>*

     <htoo:fromNewOWLIndividual>
       <htoo:elementForOWLIndividual type="QName">
          izraz
       </htoo:elementForOWLIndividual>
       <htoo:IDForOWLIndividual type="QName">
          izraz
       </htoo:IDForOWLIndividual>
       <htoo:OWLClass type="anyURI">
          URI
       </htoo:OWLClass>
    </htoo:fromNewOWLIndividual > |
```

```
<htoo:referenceToTranslationFromIndividual name=”QName”/ > |

<htoo:fromIDOfExistingOWLIndividualFromInput type="QName">
   izraz
</htoo:fromIDOfExistingOWLIndividualFromInput> |

<htoo:fromOWLIndividual type="anyURI">
   URI
</htoo:fromOWLIndividual>

<htoo:toNewOWLIndividual>
   <htoo:elementForOWLIndividual type="QName">
      izraz
   </htoo:elementForOWLIndividual>
   <htoo:IDForOWLIndividual type="QName">
      izraz
   </htoo:IDForOWLIndividual>
   <htoo:OWLClass type="anyURI">
      URI
   </htoo:OWLClass>
</htoo:toNewOWLIndividual> |

<htoo:referenceToTranslationToIndividual name=”QName” /> |

<htoo:toIDOfExistingOWLIndividualFromInput type="QName">
   izraz
</htoo:toIDOfExistingOWLIndividualFromInput> |

<htoo:toOWLIndividual type="anyURI">
   URI
</htoo:toOWLIndividual>

<htoo:OWLObjectProperty type="anyURI">
   URI
</htoo:OWLObjectProperty>
```

```
    </htoo:translateToOWLObjectProperty>

</htoo:ontologyTranslationElements>
```

Tabela 8: Sintaksa elementov opredelitve preslikav med procesnimi podatki in koncepti organizacijske ontologije

Za element <*htoo:ontologyTranslationElements*> so opredeljeni naslednji elementi in atributi:

- Atribut *queryLanguage*: Atribut določa jezik poizvedb uporabljenih v elementih preslikav (<*htoo: translateToOWLIndividual*>, <*htoo: translateToOWLDatatypeProperty*> in <*htoo: translateToOWLObjectProperty*>) in v elementu <*htoo:HTOOQueryParameters*>. Atribut je opcijski. Če ni podan, je privzeti jezik podedovan od najožjega elementa, v katerem se nahaja.

- Element <*htoo:HTOOQueryParameters*>:

  Element opredeljuje parametre (spremenljivke), ki jih lahko uporabljamo v izrazih opredeljenih v elementih preslikav (<*htoo: translateToOWLIndividual*>, <*htoo: translateToOWLDatatypeProperty*> in <*htoo:translateToOWLObjectProperty*>). Je opcijski in če je prisoten, mora vsebovati vsaj eden element <*htoo: HTOOQueryParameter*>. Sintaksa sklicevanja na parameter je $*imeParametra*.

  Tabela 9 podaja primer opredelitve parametrov.

```
<htoo:HTOOQueryParameters>
      <htoo:HTOOQueryParameter name="kandidat" type="xsd:string">
        htd:getInput("ZahtevaIzbireKandidataZaDelovnoMesto")/seznamKandidatov/kandidat
      </htoo:HTOOQueryParameter>
  </htoo:HTOOQueryParameters>
```

Tabela 9: Primer opredelitve parametrov

- Element <*translateToOWLIndividual*>: Element določa preslikavo v primerek OWL ontologije. Sestavljen je iz treh podelementov, in sicer:

  - Element <*htoo:elementForOWLIndividual*>, ki določa nosilca informacij procesnih podatkov z izrazom v danem izraznem jeziku,

- Element <*htoo:IDForOWLIndividual*>, ki določa enolično ime primerka, ki ga bo nosil v ontologiji novi primerek. Element je prav tako podan kot izraz v danem izraznem jeziku, pri čemer se izraz nanaša na element <*htoo:elementForOWLIndividual*> istega vozlišča nadelementa <*htoo:translateToOWLIndividual*>.

- Element <*htoo:OWLClass*>, ki določa URI razredov v ontologiji, v katere bo novi primerek umeščen. Elementov OWLClass je lahko več, in sicer za vsak razred, v katerega novi primerek spada.

Element lahko ima določen tudi opcijski atribut name, ki, če je podan, mora biti znotraj posameznega elementa <*htoo:ontologyTranslationElements*> enoličen.

V primeru, ko je v danih vhodnih podatkih več vozlišč nosilca informacij <*htoo:elementForOWLIndividual*>, se preslikava izvede za vsakega izmed njih. Element <*htoo:IDForOWLIndividual*> se v vsaki preslikavi izvede za posamezno vozlišče. Rezulati preslikave so v tem primeru primerki, ki so vsi uvrščeni v vsakega izmed razredov <*htoo:OWLClass*>.

Tabela 10 podaja primer opredelitve preslikave v nov primerek. Vidimo lahko, da se v opredelitvi elementov <*htoo:elementForOWLIndividual*> in <*htoo:IDForOWLIndividual*> sklicuje na parameter kandidat. V primeru, da je število vozlišč, ki ustrezajo izrazu opredeljenim s parametrom kandidat, več kot ena, se izvede toliko preslikav kolikor je takšnih vozlišč. Pri tem se elementa <*htoo:elementForOWLIndividual*> in <*htoo:IDForOWLIndividual*> posameznega primerka preslikave nanašata na isto vozlišče.

```
<htoo:translateToOWLIndividual name="preslikajKandidate">
        <htoo:elementForOWLIndividual type="xsd:string">
           $kandidat
        </htoo:elementForOWLIndividual>
        <htoo:IDForOWLIndividual type="xsd:string">
           $kandidat/identifikacija/EMSO
        </htoo:IDForOWLIndividual>
        <htoo:OWLClass>http://www.oo.si/kandidati.owl#Kandidat</htoo:OWLClass>
</htoo:translateToOWLIndividual>
```

Tabela 10: Primer opredelitve preslikave iz nosilca informacij procesnih podatkov v OWL primerek

- *<htoo:translateToOWLDatatypeProperty>*: Element določa preslikavo v podatkovno lastnost primerka ontologije. Sestavljen je iz treh podelementov, in sicer:

  - Element, ki določa primerek, ki mu bo lastnost dodeljena. Določen je z enim izmed štirih alternativnih elementov. Prvi (*<htoo:toODPWithNewIndividual>*), ki ustreza dodajanju novega primerka, je po strukturi podelementov enak elementu preslikave v novi primerek *<htoo:translateToOWLIndividual>*.
    Drugi (*<htoo:referenceToTranslationToIndividual*) je podan z elementom, katerega atribut *name* je enak atributu *name* enega izmed elementov *<htoo:translateToOWLIndividual>*.
    Tretji (*<htoo:IDOfExistingOWLIndividualFromInput>*) vsebuje izraz, ki iz vhodnih podatkov pridobi identifikator primerka iz ontlogije.
    Četrti (*<htoo:OWLIndividual>*) vsebuje URI primerka v ontologiji.
  - Element *<htoo:OWLDatatypeProperty>*, ki določa URI podatkovne lastnosti v ontologiji.

  - Element *<htoo:datatypeValue>*, ki določa izraz, na podlagi katerega se iz vhodnih podatkov pridobi podatek, ki predstavlja vrednost podatkovne lastnosti primerka. V primeru, ko se izraz elementa, ki določa primerek, ki mu bo lastnost dodeljena, ustreza več vozliščem, v posameznem primerku preslikave element *<htoo:datatypeValue>* nanaša na posameznega izmed njih. Če izraz, ki ga določa ustreza zgolj enemu vozlišču, potem vsi primerku dobijo enako vrednost lastnosti. Če ustreza več vozliščem, mora veljati, da se nanaša na isti sklop vozlišč kot izrazi, ki določajo primerek. V nasprotnem primeru, preslikava ni veljavna.

  Tabela 11 podaja primer opredelitve preslikave v podatkovno lastnost skupnega števila mesecev delovnih izkušenj posameznega kandidata. Gre za preslikavo v podatkovno lastnost novega primerka, ki bo preslikan na podlagi reference na preslikavo *preslikajKandidate*, ki jo podaja Tabela 10. Praviloma je v seznamu kandidatov več vozlišč, ki ustrezajo parametru kandidat. Element *<htoo:datatypeValue>* za posamezen primerek preslikave nanaša na isto vozlišče kot izraza *<htoo:elementForOWLIndividual>* in *<htoo:IDForOWLIndividual>* v preslikavi *preslikajKandidate*.

<htoo:translateToOWLDatatypeProperty>

```
<htoo:referenceToTranslationToIndividual name="preslikajKandidate" />
<htoo:OWLDatatypeProperty>http://www.oo.si/kandidati.owl#steviloLetDelovnih
Izkusenj </htoo:OWLDatatypeProperty>
<htoo:datatypeValue type="xsd:string">
    fn:sum(for $j in $kandidat/delovneIzkusnje return $j/obdobjeVMesecih)
</htoo:datatypeValue>
</htoo:translateToOWLDatatypeProperty>
```

Tabela 11: Primer opredelitve preslikave iz nosilca informacij procesnih podatkov v OWL podatkovno lastnost

- *<htoo:translateToOWLObjectProperty>*: Element določa preslikavo v objektno lastnost primerka ontologije. Sestavljen je iz treh podelementov:

    - Element, ki določa primerek, ki mu bo lastnost dodeljena, in element, ki določa primerek, ki bo vrednost lastnosti. Oba primerka sta lahko določena z enim izmed štirih alternativnih elementov. Prvi, ki ustreza dodajanju novega primerka, je po strukturi podelementov enak elementu preslikave v novi primerek <htoo:translateToOWLIndividual>). Drugi je podan z elementom, katerega atribut name je enak atributu name enega izmed elementov <htoo:translateToOWLIndividual>. Tretji vsebuje izraz, ki iz vhodnih podatkov pridobi identifikator primerka iz ontlogije. Četrti vsebuje URI primerka v ontologiji.

    - Element <htoo:OWLObjectProperty>, ki določa URI objektne lastnosti v ontologiji.

  Tabela 12 podaja primer opredelitve preslikave v objektno lastnost, ki določa stopnjo znanja angleškega jezika posameznega kandidata. Primerki, za katere bo lastnost določena, so opredeljeni s preslikavo *preslikajKandidate* (Tabela 10) .

```
<htoo:translateToOWLObjectProperty>
    <htoo:referenceToTranslationToNewIndividual name="preslikajKandidate" />
    <htoo:toIDOfExistingOWLIndividualFromInput>
      for $j in $kandidat/tujiJeziki/tujJezik
      return
```

```
            if(fn:compare($j/sifra,'en-UK')=0) then
fn:concat('http://www.oo.si/kandidati.owl#',fn:data($j/p1:stopnja))
            else ()
        </htoo:toIDOfExistingOWLIndividualFromInput>
        <htoo:OWLObjectProperty> http://www.oo.si/kandidati.owl#govoriAngleskiJezik
</htoo:OWLObjectProperty>
    </htoo:translateToOWLObjectProperty>
```

Tabela 12: Primer opredelitve preslikave iz nosilca informacij procesnih podatkov v OWL objektno lastnost

Element preslikave za sklop uporabniških opravil

Njegova sintaksa je naslednja:

```
<htoo: taskGroupOntologyTranslationElements name="NCName">

        < htoo: taskGroupName>
        …
        </htoo: taskGroupName>

        < htoo: ontologyTranslationElements queryLanguage="anyURI"?>
        …
        </ htoo: ontologyTranslationElements>

</ htoo: taskGroupOntologyTranslationElements >
```

Tabela 13: Sintaksa elementa preslikave za sklop uporabniških opravil

Element preslikave za sklop uporabniških opravil je sestavljen iz dveh podelementov. Prvi določa ime preslikave in drugi določa preslikavo. Sintaksa elementa, ki določa preslikavo, je enaka sintaksi elementa <*htoo:ontologyTranslationElements*>, ki jo podaja Tabela 8.

8.5 Predlagana metoda odločanja kot mehanizem eksternalizacije implicitnega znanja v eksplicitnega

Eksternalizacija je mehanizem, ki omogoča, da se implicitno znanje zaposlencev spremeni v eksplicitno in se shrani v organizacijski spomin [78]. Predlagana metoda odločanja predstavlja mehanizem za eksternalizacijo znanja, potrebnega za sprejemanje odločitve.

Eksternalizacija odločitvenega modela se nanaša na faze 2 do 5 odločitvenega postopka (glej poglavje 6 Metoda večkriterijskega odločanja, ki temelji na ontologijah). Opredelitev problema je določena s primerkom uporabniškega opravila. Poslovni proces ob proženju uporabniškega opravila posreduje vhodne podatke, ki med drugim opredeljujejo tudi pričakovani rezultat uporabniškega opravila. Ta določa razred alternativ.

Navadno je namen eksternalizacije znanja v organizacijskem spominu omogočanje ponovne uporabe in skupne uporabe znanja med različnimi zaposlenci in poenostavljanje dostopa do informacij [19]. Pomembno pa je poudariti, da je v primeru eksternalizacije odločitvenih modelov v uporabniških opravilih in za uporabo v uporabniških opravilih primarni namen drugačen. Čeprav gre za omogočanje ponovne uporabe znanja, se le-ta na prvem mestu ne nanaša na ponovno uporabo s strani zaposlencev, temveč za omogočanje avtomatizacije odločitev kot dela poslovnega procesa.

Organizacijska ontologija je zgrajena hierarhično in vsaka ontologija, ki jo sestavlja, pripada točno določenemu organizacijskemu elementu. Tako so tudi odločitveni modeli lahko zasnovani hierarhično. Pripadajo lahko posamezniku ali so skupni več zaposlencem.

Posamezna ontologija je zgrajena iz dveh ravni, in sicer splošne in odločitvene ravni. Splošna raven vsebuje elemente uvoženih ontologij in elemente domene organizacijskega elementa, ki mu pripada. Odločitvena raven nadgrajuje splošno raven z odločitvenimi modeli.

Namen eksplicitnega zapisa odločitvenih modelov, ki so skupni več zaposlencem, je poleg avtomatizacije, tudi v ponovni uporabi skupnega znanja. V nasprotju s tem namen eksplicitnega zapisa odločitvenih modelov, ki so vezani na posamezno osebo, tudi v primeru, ko odločitev ni avtomatizirana, ni v omogočanju skupne uporabe znanja. Odločitveni modeli, ki pripadajo ontologiji posameznika, so v veliki meri subjektivni in so zato, kadar odločitev ni avtomatizirana, praviloma namenjeni ponovni uporabi odločitvenega modela s strani iste osebe.

Če bi bila skupna uporaba odločitvenega modela posameznika v določenem primeru kljub temu zaželena, na primer če bi se določen odločitveni model izkazal kot posebej primeren in bi prerasel v splošno sprejet model odločanja na nivoju določene organizacijske enote ali poslovnega sistema kot celote, je seveda to prav tako posreden namen, čeprav je v danem kontekstu bolj izjema kot pravilo.

8.6 Spremembe konceptualne ravni organizacijske ontologije, ki izvirajo iz opredelitve in izvajanja uporabniških opravil

Pri opredeljevanju in izvajanju uporabniških opravil ter preslikavi protokolov sodelovanja iz poslovne ravni v knjižnico protokolov se lahko zgodi, da informacije zajete v ontologiji ne zadoščajo. Gre za naslednje primere:

- Ob opredeljevanju preslikav med nosilci informacij procesnih podatkov in ontologijo se lahko zgodi, da koncept, v katerega bi želeli preslikati določen element vhodnega sporočila oziroma ki bi ga potrebovali za preslikavo v izhodno sporočilo, v ontologiji ne obstaja oziroma da njegova opredelitev ni več aktualna.

- Pri izdelavi odločitvenih modelov ugotovimo, da ustreznih konceptov, ki bi bilo potrebni za odločitev, v ontologiji ni. V teh primerih, so potrebne spremembe konceptualne ravni ontologije. Poleg tega tudi sama izdelava odločitvenega modela v ontologiji (glej poglavje 6 Metoda večkriterijskega odločanja, ki temelji na ontologijah) zahteva spremembe na konceptualni ravni ontologije, in sicer izdelavo podrazredov kriterijev in opredelitev odločitvenih pravil.

- Pri opredeljevanju protokola na poslovni ravni, kjer temeljimo na konceptih zajetih v organizacijski ontologiji, lahko ti ne zadoščajo. Zato poslovni uporabnik doda nov koncept, ki nima preslikave v raven ontologije. Poleg tega tudi sama implementacija ontologij protokola zahteva spremembe na konceptualni ravni ontologije, in sicer razširitve temeljne ontologije knjižnice protokolov, drugih skupnih ontologij protokolov in implementacijo SWRL pravil v ontologijah udeležencev.

V vseh navedenih primerih gre za obvladovanje spremembe konceptualne ravni ontologije. To je odvisno od uporabljene metodologije. Umestitev navedenih dogodkov v metodologije obvladovanja znanja je podana v naslednjem podpoglavju (8.7).

8.7 Umestitev modela v metodologije obvladovanja znanja, ki temeljijo na ontologijah

Predlagani model temelji na uporabi obstoječe ontologije in je komplementaren s pristopi in metodologijami razvoja sistemov za obvladovanje znanja, ki temeljijo na ontologijah. Gre za pomembno lastnost, saj ga lahko uporabljamo v skladu z izbrano metodologijo in pristopi k razvoju, vzdrževanju in uporabi sistemov za obvladovanje znanja. Za razumevanje predlaganega modela v okviru celotnega življenjskega cikla sistemov obvladovanja znanja je v nadaljevanju podana umestitev predlaganih postopkov v dve priznani metodologiji, in sicer:

- v metodologijo On-To-Knowledge [70],[77],[76],[78],
- v proces evolucije ontologij, ki so ga predlagali Maedche et al. [49].

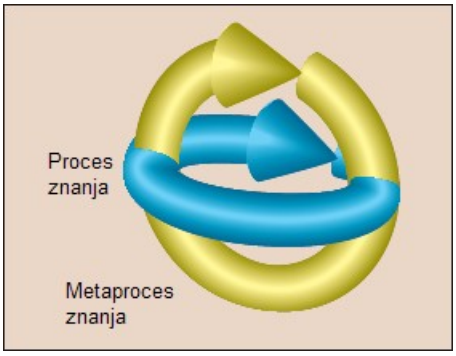
Metodologija On-To-Knowledge je izbrana kot primer metodologije, ki se ukvarja s celotnim življenjskim ciklom sistema za obvladovanje znanja, ki temelji na ontologijah. Na primeru metodologije On-To-Knowledge bomo prikazali, kako se določeni dogodki in aktivnosti opredeljevanja in izvajanja uporabniških opravil kot dela poslovnega procesa navezujejo v kontekst celotnega življenjskega cikla sistemov obvladovanja znanja.

Maedche et al. so avtorji procesa evolucije ontologije, ki se ne ukvarja s celotnim razvojem in uporabo ontologije, temveč podrobneje obravnava prav fazo evolucije. Z podrobnejšo umestitvijo modela predlaganega v disertaciji v proces evolucije, bomo ponazorili obvladovanje konceptualne ravni ontologije, v primerih ko obstoječa ontologija za potrebe uporabniških opravil ne zadošča.

8.7.1 Razvoj in izvajanje uporabniških opravil po metodologiji *On-To-Knowledge*

On-To-Knowledge ločuje med dvema procesoma obvladovanja znanja, ki bi se naj izvajala ločeno z namenom doseganja jasne identifikacije različnih aktivnosti in problemov obvladovanja znanja (Slika 16). Prvi proces naslavlja vidike uvajanja novega informacijskega sistema obvladovanja znanja v poslovni sistem ter vidike njegovega vzdrževanja (metaproces znanja) in predstavlja razširitev ter izboljšavo medotodologije CommonKADS [66]. Drugi

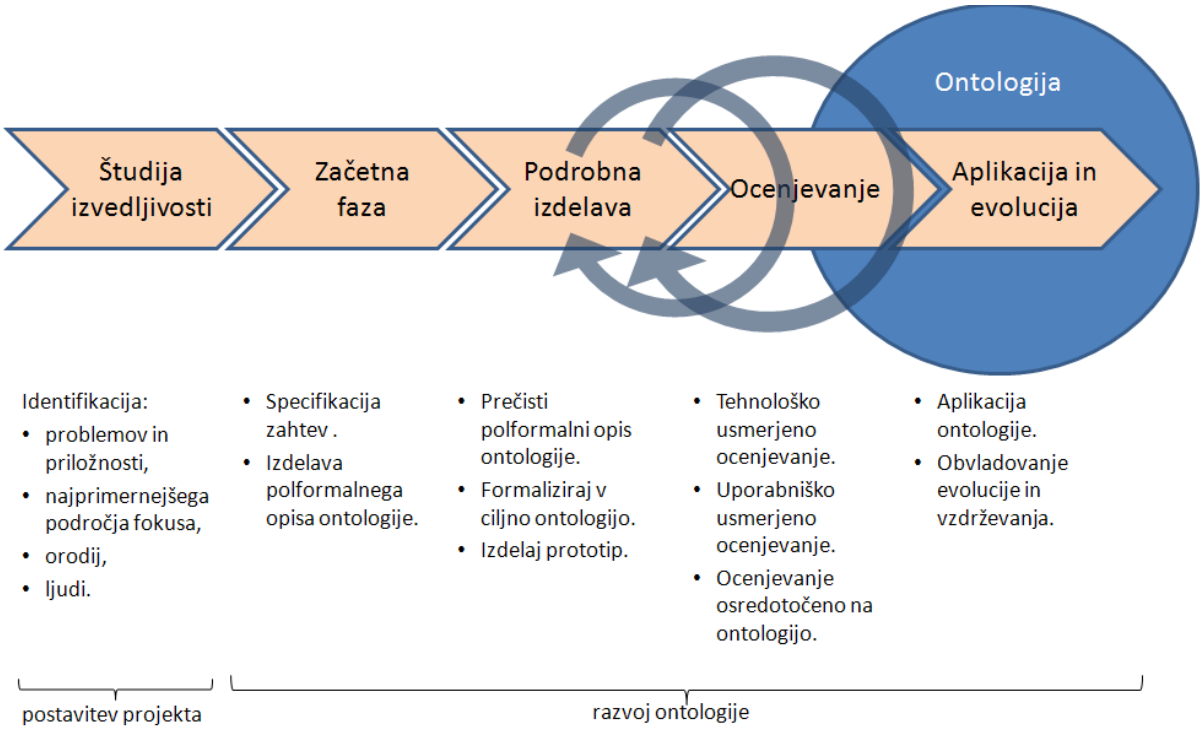
proces naslavlja upravljanje obstoječe rešitve informacijskega sistema obvladovanja znanja (proces znanja).



Slika 16: Ločevanje procesov obvladovanja znanja po metodologiji On-To-Knowledge

V nadaljevanju sta opisana oba procesa obvladovanja znanja skupaj z umestitvijo različnih delov modela,ki ga podajamo v disertaciji.

Metaproces znanja



Slika 17: Metaproces znanja

Slika 17 povzema metaproces znanja. Organizacijska ontologija, ki jo obravnavamo v disertaciji, ima hierarhično strukturo opredeljeno z organizacijsko strukturo poslovnega sistema. Metaproces znanja se izvaja za posamezne ontologije v organizacijski ontologiji.

Vzpostavitev strukture organizacijske ontologije je del metaprocesa znanja (glej pogavje 8.3) in je pogoj za izvedbo metaprocesov znanja ostalih ontologij v organizacijski ontologiji.

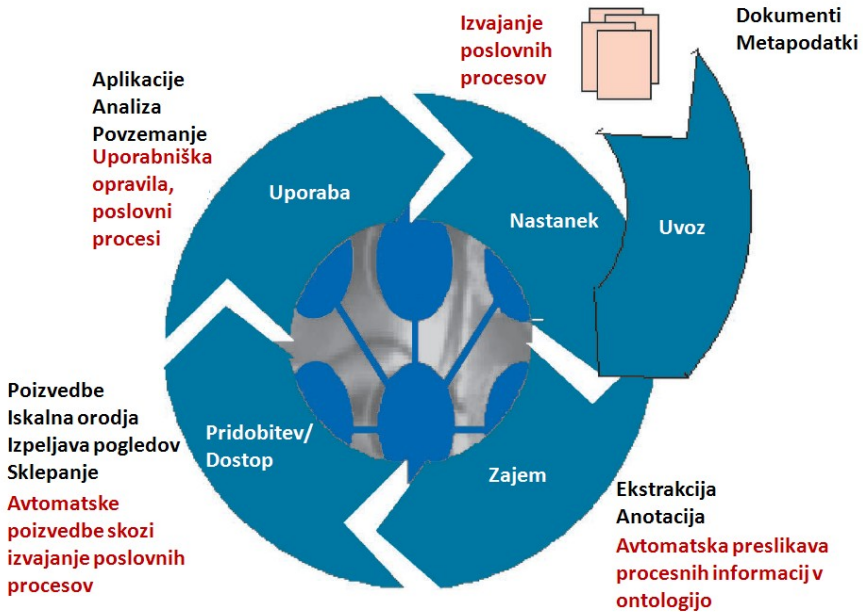
Z izjemo vzpostavitve strukture organizacijske ontologije, ki je pogoj za uporabo modela, model MAPP obravnava delo z obstoječo ontologijo. Določene aktivnosti povezane z opredelitvami uporabniških opravil prožijo aktivnosti faze aplikacije in evolucije ontologije. Faza aplikacije ontologije se nanaša na uporabo sistemov, ki temeljijo na ontologijah. Aplikacija ontologije je stičiščna točka med metaprocesom znanja in procesom znanja.

Evolucija ontologije se nanaša na aktivnosti, povezane s spremembami ontologije z namenom odražanja sprememb v resničnem svetu. Opredeljena morajo biti stroga pravila, ki določajo postopke dodajanja, brisanja in posodabljanja konceptov ontologije. V okviru razvoja in izvajanja poslovnih procesov in uporabniških opravil se aktivnosti evolucije prožijo bodisi ob novih uporabniških opravilih, ob spremembah uporabniškega opravila ali ob spremembah poslovnih procesov (glej poglavje 8.6). Na primer, v okviru opredelitve novega uporabniškega opravila je potrebno določiti preslikave med nosilci informacij vhodnih in izhodnih podatkov ter koncepti ontologije. V primeru, da določen koncept, ki ga želimo uporabiti v uporabniškem opravilu, v ontologiji ne obstaja, se proži aktivnost evolucije ontologije, v kateri gre za dodajanje novega koncepta. Do posodabljanja lahko pride ob spremembah poslovnega sistema oziroma njegovega okolja, ki vplivajo na poslovne procese oziroma na uporabniška opravila. Do brisanja koncepta lahko pride v primeru, če ugotovimo, da določen koncept ni niti eksplicitno niti implicitno relevanten za nobeno uporabniško opravilo (in morebitne druge uporabnike ontologije).

Spremembe evolucije ontologije lahko obvladujemo skozi nov cikel razvoja – od faze podrobne izdelave, ocenjevanja in aplikacije ter evolucije (Slika 17) – oziroma kot to opredeljujejo pravila in strategije evolucije.

Proces znanja

Ko je sistem za obvladovanje znanja implementiran v poslovnem sistemu, procesi znanja krožijo skozi korake prikazane na spodnji sliki (Slika 18). Procesni koraki odražajo tipične korake, ki jih upravljavci znanja opravljajo v okviru svojih vsakodnevnih opravil. Gre torej za korake od nastanka znanja, preko njegovega zajema v ontologijo, do dostopanja do tega znanja in njegove nadaljnje uporabe.



Slika 18: Proces znanja

Opomba: Z rdečo so navedene posebnosti značilne za proces znanja z uporabo MAPP

Predlagan model MAPP se tesno povezuje s procesom znanja in ga v nadgrajuje ter izboljšuje. V nadaljevanju so opisani posamezni koraki, skupaj s posebnostmi, ki se nanašajo na proces znanja z uporabo MAPP:

- Nastanek ali uvoz:

  Ustvarjanje znanja je del vsakdana upravljavcev znanja. Poslovni procesi opisujejo dejavnost poslovnega sistema. Lahko pravimo tudi, da znanje nastaja skozi izvajanje poslovnih procesov. Stopnje ustvarjenega znanja se razlikujejo od formalnih do neformalnih, na primer od prostega teksta in dokumentov brez vnaprej opredeljene strukture, do vsebinsko strukturiranega teksta ali strogo določeno XML strukturo vsebin. Uvoženo znanje je zahtevnejše za kasnejši uvoz v ontologijo, saj ga je v ta namen potrebno praviloma dodatno opremiti z metapodatki.

  V storitveno usmerjenih sistemih implementacija poslovnih procesov zahteva opredelitev formalnih XML struktur. Znanje torej nastaja v okviru nadzorovanega izvajanja poslovnih procesov na podlagi formaliziranih struktur.

- Zajem:

Ustvarjeno znanje je potrebno na pravilen način umestiti v ontologijo. Metodologija je osredotočena na zajem znanja z izdelavo metapodatkov v skladu z ontologijami, kar imenujejo anotacijski proces. Zajem znanja pomeni zajem bistvene vsebine ustvarjenih elementov znanja. V ta namen so izdelali tudi posebno orodje, ki omogoča označevanje (anotacijo) vsebin, na primer določenih izrazov v dokumentu.

MAPP podpira zajem znanja na podlagi preslikav med nosilci informacij vhodnih in izhodnih podatkov uporabniških opravil. Preslikave omogočajo, da se znanje, ki je ustvarjeno skozi izvajanje poslovnega procesa in je vezano na formalizirano strukturo, avtomatsko prenese v ontologijo. Pri tem gre za preslikavo v raven primerkov ontologije. Za posamezno preslikavo ali sklop preslikav lahko na podlagi predloga razširitve specifikacij WS-HumanTask določimo, ali je vnos avtomatski ali zahteva potrditev s strani ustreznega upravljavca znanja. V tem primeru se proži nova instanca uporabniškega opravila namenjena upravljavcu znanja, ki zajem potrdi, popravi in potrdi ali zavrne. Gre torej za pomemben korak naprej na področju zajema znanja, saj omogoča izkoriščanje pristopa izvajanja poslovnih procesov v skladu s storitveno usmerjeno arhitekturo za avtomatski ali polavtomatski zajem primerkov ontologije in njihovih lastnosti namesto ročnega označevanja v bolj ali manj strukturiranih dokumentih.

- Priklic/Dostop:

Priklic in dostop se nanašata na izpolnjevanje iskanj in poizvedb v ontologiji, ki ga izvaja upravljavec znanja. Največkrat se v ta namen uporabljajo namenski uporabniški vmesniki, same poizvedbe in iskanja pa so lahko dopolnjene tudi z različnimi vpogledi v ontologijo. Posebej pomembna je možnost uporabe strojev za sklepanje, s katerimi lahko na podlagi danega znanja v ontologiji pridemo do novih sklepov.

Model predlagan v pričujoči disertaciji nadgrajuje možnosti pridobitve in dostopa do znanja z opredelitvijo poizvedb kot dela opredelitve uporabniškega opravila. Pri tem so lahko poizvedbe določene kot izraz, ki lahko vsebuje spremenljivke. Med samim izvajanjem uporabniškega opravila se na podlagi vhodnih podatkov, ki jih je posredovala instanca poslovnega procesa, ki je opravilo prožila, poizvedbe dinamično sestavijo. Na podlagi vhodnih podatkov, ki jih preslikamo v ontologijo, obstoječega znanja zajetega v ontologiji in stroja za sklepanje lahko najdemo odgovor na poizvedbo.

Gre torej za dinamične poizvedbe, ki se prožijo na podlagi podatkov izvajajoče se instance poslovnega procesa, in za avtomatsko integracijo s strojem za sklepanje in z obstoječim znanjem. Namen tovrstne integracije je pridobitev novih informacij potrebnih za izvedbo opravila. Če je v ontologiji zajeto znanje za izvedbo opravila in so na voljo informacije potrebne kot vhod v opravilo, se lahko opravilo izvede popolnoma avtomatizirano. V skladu z opredelitvijo opravila v (F1), $t(inp_t) = r_t$, to pomeni, da je sama funkcija t zajeta v ontologiji, medtem ko je vhod v opravilo sestavljen iz informacij, ki so že zajete v ontologiji in/ali informacij posredovanih v uporabniško opravilo iz poslovnega procesa in ki so preslikane v koncepte ontologije na podlagi preslikav določenih v opredelitvi opravila. Primer funkcije t implementirane v ontologiji je odločitveni model večkriterijskega odločanja, ki temelji na ontologijah (glej poglavje 6 Metoda večkriterijskega odločanja, ki temelji na ontologijah).

Sama uporaba znanja je tako implicitno integrirana v izvajanje poslovnega procesa in avtomatsko uporabljena tam, kjer je potrebna oziroma kjer lahko pripomore k izvajanju. Uporabniku torej, če ne želi, ni potrebno iskati po različnih vpogledih v ontologiji, saj je uporaba ontologije v tem primeru za uporabnika popolnoma transparentna.

Če ustrezni vhodi v opravilo niso na voljo ali če znanje potrebno za izvedbo opravila v ontologiji ni implementirano, opravilo ni avtomatizirano. V tem primeru lahko delni rezultati poizvedbe služijo kot pomoč lastniku opravila.

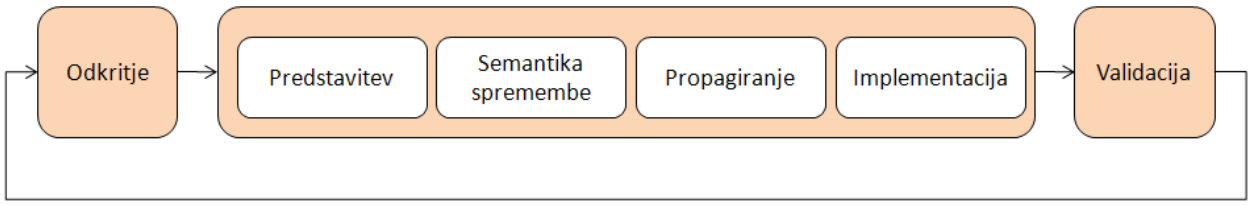
- Uporaba:

Znanje zajeto v ontologiji ne služi zgolj za priklic oziroma dostop do tega znanja v prihodnosti, temveč tudi za njegovo nadaljnjo uporabo. Pri tem so pomembna načela za učinkovito ponovno uporabo predvsem proaktivni dostop, personalizacija in tesna integracija z nadaljnjimi aplikacijami ontologij.

Organizacijska ontologija je zasnovana tako, da omogoča visoko raven integracije različnih ontologij, ki pripadajo različnim organizacijskim elementom, in visoko raven personalizacije, saj je vsaka ontologija v organizacijski ontologiji namensko izdelana za uporabnika oziroma skupino uporabnikov. Hierarhična struktura in pripadnost posameznim organizacijskim elementom sta ključna koncepta, ki omogočata visoko raven ponovne uporabe. Na proaktivni dostop se model nanaša s pomočjo opredelitve

uporabniškega opravila, v katerem lahko določimo, katere informacije lahko potencialno služijo za izvedbo opravila.

8.7.2 Razvoj in izvajanje uporabniških opravil v procesu evolucije ontologije po Maedche et al.



Slika 19: Evolucija ontologije [49]

Evolucija ontologije po Maedche et al. je časovno prilagajanje ontologije spreminjajočim se zahtevam in konsistentno propagiranje sprememb odvisnim artefaktom [49]. Njihov proces evolucije ontologije je sestavljen iz šestih faz, ki so prikazane na zgornji sliki (Slika 19). V nadaljevanju so opisane posamezne faze, v katere so umeščeni postopki, ki izhajajo iz razvoja in izvajanja uporabniških opravil:

- Odkritje:

  Spremembe, ki jih je potrebno implementirati v ontologiji, lahko imajo različne izvore, na primer: spremembe v okolju ali v poslovnem sistemu, ki se morajo odražati v ontologiji, uporaba ontologije pokaže na redundantne koncepte, v fazi validacije so odkrite neoptimalne strukture konsistentne ontologije itd. Na voljo so različne metode, ki omogočajo samoprilagajanje sistema in proaktivne predloge za izboljšanje ontologije, na primer odkrivanje sprememb na podlagi hevristik, ki skušajo izboljšati ontologijo na podlagi njene strukture, odkrivanje sprememb na podlagi analize obstoječih primerkov ali odkrivanje sprememb na podlagi analize uporabe ontologije.

  Ob razvoju in uporabi MAPP so lahko odkrite potrebne spremembe konceptualne ravni ontologije v naslednjih primerih:

  - ob opredeljevanju preslikav med nosilci informacij procesnih podatkov in ontologijo,

- ob izdelavi odločitvenih modelov po predlaganem postopku,
- pri opredeljevanju protokola sodelovanja.

Ti primeri so podrobneje opisani v poglavju 8.6 Spremembe konceptualne ravni organizacijske ontologije, ki izvirajo iz opredelitve in izvajanja uporabniških opravil.

- Predstavitev:

  Spremembo je potrebno predstaviti v ustreznem formatu. Spremembe so lahko elementarne, na primer dodajanje koncepta, odstranitev koncepta, določanje zaloge vrednosti lastnosti, ali sestavljene, na primer premestitev razreda iz enega nadrazreda k drugemu.

  Predstavitev sprememb, ki izvirajo iz opredelitve ali izvajanja uporabniških opravil s MAPP, v splošnem nima posebnosti. Izjema je predstavitev sprememb v ontologiji pri izdelavi odločitvenih modelov, ki je opisana v okviru faz izdelave podrazredov kriterijev in opredelitve odločitvenih pravil (glej poglavje 6 Metoda večkriterijskega odločanja, ki temelji na ontologijah) in pri izdelavi ontologij protokolov, ki razširjajo skupne ontologije (glej poglavje 7.4 Knjižnica protokolov sodelovanja).

- Semantika spremembe:

  Sprememba ontologije lahko povzroči nekonsistentnost v drugih delih ontologije. Lahko gre za semantično ali sintaktično nekonsistentnost. Do semantične nekonsistentnosti pride, kadar se spremeni pomen entitete v ontologiji. Do sintaktične nekonsistentnosti pride, če so uporabljene nedefinirane entitete na konceptualni ravni ali ravni primerkov ali če so kršene omejitve modela ontologije. Težave z nekonsistentnostjo lahko odpravimo tako, da zahtevamo nove spremembe. V velikih ontologijah to lahko postane težavno. Za posodabljanje ontologije, tako da preide v želeno konsistentno stanje, se uporabljajo različne evolucijske strategije [73].

  V modelu MAPP je poenostavljeno obvladovanje sprememb, ki izvirajo iz izdelave odločitvenih modelov (dodajanje podrazredov izpeljanih kriterijev in opredelitev odločitvenih pravil) in iz izdelave ontologij protokolov. Ker gre zgolj za dodajanje brez vpliva na druge obstoječe koncepte, v ontologijah posameznikov faza semantike spremembe ni potrebna.

Drugih posebnosti v fazi semantike spremembe ob uporabi modela MAPP ni.

- Propagiranje:

  Faza propagiranja bi naj po posodabljanju ontologije omogočila avtomatski prehod vseh odvisnih elementov v konsistentno stanje. Sprememba ontologije lahko povzroči spremembe v odvisnih ontologijah, in posledično težave z nekonsistentnostjo konceptov v odvisnih ontologijah. Problem rešimo z rekurzivnim izvajanjem procesa evolucije na odvisnih ontologijah. Poleg tega je potrebno upoštevati tudi semantično nekonsistentnost in vpliv na aplikativne sisteme, ki uporabljajo spremenjeno ontologijo.

  Predlagana organizacijska ontologija v MAPP je strukturirana hierarhično. Ontologija, ki pripada določenemu organizacijskemu elementu, lahko uporablja ontologije tistih organizacijskih elementov, v katere njen lastnik spada v organizacijski strukturi poslovnega sistema. To pomeni, da je od njih odvisna.

  Uporabniška opravila med drugim opredeljujejo preslikave med nosilci informacij procesnih podatkov ter organizacijsko ontologijo. Sprememba v ontologiji, na katero se preslikava nanaša, lahko povzroči nepravilne preslikave procesnih podatkov v ontologijo in s tem nepravilnosti na ravni primerkov ontologije. Poleg tega lahko nepravilna preslikava v izhodne podatke uporabniškega opravila povzroči napačno izvajanje poslovnega procesa. Ob spremembi je potrebno torej natančno preveriti vpliv spremembe na opredelitev preslikav v ontologiji in jih ustrezno prilagoditi.

- Implementacija:

  Da se izognemo nezaželenim spremembam, sistem pred potrditvijo sprememb v ontologijo za uporabnika generira seznam vseh sprememb in vpliva sprememb ontologije. Uporabnik na podlagi seznama spremembe potrdi ali zavrne. Kadar je potrebno izvesti več sprememb naenkrat, zato da ontologija preide iz enega v drugo konsistentno stanje, je pomembno, da se takšni sklopi sprememb izvedejo kot transakcija.

  Posebnosti v fazi implementacije spremembe ob uporabi modela MAPP ni.

- Validacija:

  Faza validacije se nanaša na uporabo ontologije po potrjeni spremembi in na možnost vrnitve v predhodno stanje, če uporabniki ugotovijo, da novo stanje ni pravilno. Različni

uporabniki imajo lahko različne poglede na določeno področje, ki ga ontologija opisuje, ali potrdijo spremembo, ki jo niso pravilno razumeli. Po potrditvi spremembe, se skozi uporabo lahko ugotovi, da sprememba ni bila pravilna.

V MAPP se, po implementaciji spremembe v ontologiji, v naslednjih primerkih izvedbe uporabniških opravil uporabi nova različica ontologije. Ob prehodu iz ene različice ontologije v drugo je pomembno, da se uporabniška opravila, ki spremenjeno ontologijo uporabljajo in so sicer avtomatizirana, v obdobju validacije izvajajo kot polavtomatizirana. Obdobje validacije je lahko podano z rokom konca ali s številom primerkov uporabniških opravil, ki se morajo izvesti za validacijo.

8.8 Analiza avtomatizacije vrst odločanja kot dela izvajanja poslovnih procesov z uporabo modela za avtomatizacijo uporabniških opravil in metode večkriterijskega odločanja, ki temelji na ontologijah

Za sprejem odločitve z uporabo metode opredeljene v poglavju 6 je potrebno naslednje:

- V ontologiji so zajete alternative.
- V ontologiji so zajete informacije o alternativah, ki predstavljajo kriterije odločanja,
- V ontologiji je izdelan odločitveni model.

Na podlagi tega, lahko ločimo lahko dve vrsti odločanja:

- Ob izvedbi uporabniškega opravila, ki zahteva odločitev, so alternative zajete in opisane v ontologiji tako, da so na voljo vse potrebne informacije, ki opisujejo kriterije odločanja. Naloga lastnika pri izvedbi opravila je tako izdelati odločitveni model oziroma uporabiti že obstoječ odločitveni model.
- Ob izvedbi uporabniškega opravila, ki zahteva odločitev, informacije o alternativah niso zadostno opisane v ontologiji. Za sprejetje odločitve mora lastnik opravila opredeliti informacije o alternativah, ki opisujejo kriterije. Nato lastnik izdela odločitveni model oziroma uporabi že obstoječi odločitveni model.

Z uporabo modela, opredeljenega v tem delu, pridobimo več pri odločitvah prve vrste, ki so lahko teoretično popolnoma avtomatizirana. Ko oseba, ki sprejema odločitev, opredeli odločitveni model, so lahko odločitve avtomatizirane dokler ne želi spremeniti odločitvenega modela, na primer v primeru potrebe po spremembi kriterijev, posameznih odločitvenih pravil itd. V nadaljevanju sta podana primera za obe vrsti odločanja.

8.8.1 Primer prve vrste odločanja

Primer prve vrste odločanja lahko najdemo v procesu izbire najboljšega kandidata za zaposlitev na določeno delovno mesto. Primer procesa je sestavljen iz naslednjih korakov:

1) Objava prostega delovnega mesta,

2) Zbiranje prijav kandidatov,

3) Izbira najboljših 10 kandidatov,

4) Pošiljanje vabil na intervjuje izbranim kandidatom,

5) Izvedba intervjujev in testov,

6) Ocenitev kandidatov in izbira najprimernejšega.

Najprej bomo opisali kako se proces izvaja brez uvedbe modela MAPP in nato kako implementacija modela izboljša njegovo izvajanje.

Večina opravil poslovnega procesa izbire najboljšega kandidata je uporabniških opravil. Začne se z objavo prostega delovnega mesta na spletni strani podjetja, ki je izvedena avtomatsko po prenehanju zaposlitve osebe, ki je bila predhodno zaposlena na delovnem mestu, ali ob ustanovitvi novega delovnega mesta v poslovnem sistemu. Nato sledi zbiranje prijav kandidatov, pri čemer kandidati izpolnjujejo obrazce za prijavo preko spletne aplikacije. Po preteku časovnega roka zbiranja prijav se proži opravilo izbire najboljših deset kandidatov. Gre za opravilo sestavljeno iz dveh podopravil. V prvem so odstranjeni kandidati, ki ne zadoščajo formalnim zahtevam delovnega mesta. V drugem odgovorna oseba oceni prijave kandidatov, ki formalno izpolnjujejo pogoje za delovno mesto, in izbere najprimernejših deset kandidatov. Sledi avtomatsko pošiljanje elektronske pošte s povabilom na intervju izbranim desetim kandidatom. Zatem so izvedeni intervjuji. Na vsakem intervjuju so prisotni dva ali trije zaposlenci: vodja organizacijske enote, ki išče kandidata za delovno

mesto, zaposlenec v organizacijski enoti, ki išče kandidata za delovno mesto in bo sodeloval z bodočim novim zaposlencem, in po potrebi še tretji zaposlenec, ki je lahko bodisi oseba iz kadrovske službe, bodisi drugi kompetentni zaposlenec poslovnega sistema. Intervju zajema tudi reševanje vnaprej pripravljenega testa, ki ga nato pregleda ustrezni zaposlenec in vrne rezultat uporabniškega opravila procesu. Vsak izmed prisotnih iz intervjuja izbere določene podatke o intervjuvancu in jih vnese kot zahtevani rezultat uporabniškega opravila. Zadnje opravilo je prav tako uporabniško opravilo, ki ga praviloma izvedejo vsak zase vodja organizacijske enote, ki išče kandidata za delovno mesto ali njegov namestnik, ter ostali zaposlenci, ki so bili prisotni na intervjuju. Zahteva izbiro najprimernejšega kandidata na podlagi informacij iz prijav kandidatov in informacij zbranih na intervjujih. Končna izbira najprimernejšega kandidata se izvede nato avtomatsko, glede na odločitve posameznih odločevalcev in vpliva, ki jo imajo na končno odločitev (na primer odločitev vodje oddelka ima večjo utež kot odločitev podrejenega).

Z vidika poslovnega sistema, ki išče najboljšega kandidata, so avtomatizirana opravila objava prostega delovnega mesta, zbiranje prijav kandidatov, pošiljanje vabil kandidatom in končna odločitev. Ostala opravila so uporabniška opravila:

- Izbira desetih najboljših kandidatov je uporabniško opravilo, ki zahteva izbiro najprimernejših kandidatov, ki bodo povabljeni na intervju, in tako zahteva odločitev.

- Izvedba intervjujev spada med opravila, ki jih ne moremo avtomatizirati, saj vključujejo zbiranje informacij na podlagi pogovora s posameznim kandidatom.

- Testiranje je sestavljeno iz dveh delov, in sicer reševanja testov in ocenjevanja testov. Reševanja testov ne moremo avtomatizirati, saj kandidati niso del poslovnega sistema in jih ne moremo zastopati z agenti. Ocenjevanje testov se z vidika poslovnega sistema potencialno da avtomatizirati, na primer če kandidati teste rešujejo v aplikaciji in naloge v testu zahtevajo izbiro pravilnega odgovora iz vnaprej pripravljene množice odgovorov. Ocenjevanje testiranja spada v uporabniška opravila s preddefiniranim postopkom, saj bi naj bila pravilnost odgovorov neodvisna od osebe, ki odgovore pregleduje. Ta bi naj imela zadostno znanje in naj bi rešene teste pregledala kompetentno, zato je ob pravilni izvedbi pregleda testiranja rezultat neodvisen od osebe, ki teste ocenjuje. To pomeni, da bi avtomatsko testiranje ne spadalo v storitev za izvajanje uporabniških opravil.

- Ocenitev kandidatov in izbira najprimernejšega je uporabniško opravilo, ki zahteva odločitev o najprimernejšem kandidatu na podlagi zbranih informacij o kandidatih.

Z implementacijo in uporabo modela za doseganja višje ravni avtomatizacije uporabniških opravil podanega v tem delu, lahko tako izboljšamo izvedbo dveh opravil, in sicer Izbira najboljših 10 kandidatov in Ocenitev kandidatov in izbira najprimernejšega. Obe opravili lahko potencialno popolnoma avtomatiziramo. Gre za sorodni opravili, pri čemer odločitveni model prvega temelji na manj informacijah o kandidatih, drugi pa tudi na dodatnih informacijah pridobljenih na intervjujih in testih. Pri tem je odločitveni model praviloma vezan na določeno delovno mesto ali skupino enakovrednih delovnih mest (delovna mesta istega tipa), na primer če v organizacijski enoti potrebujejo več delovnih mest z enakimi kompetencami, na primer pet razvijalcev v programskem jeziku Java. Za primerke poslovnega procesa, ki se nanašajo na isto delovno mesto oziroma delovno mesto določenega tipa, se uporablja isti odločitveni model. Ko je takšen primerek prožen prvič odločitveni model še ni izdelan. Oseba, ki sprejema odločitev, ga izdela. Nato lahko agent, ki zastopa lastnika opravila, s pomočjo SOO sprejme odločitev. Pri tem je predpostavka, da so informacije o kandidatih zajete v ontologiji. Ob proženju uporabniškega opravila so posredovane kot vhodni podatki v SIUO in preslikane v organizacijsko ontologijo. Ko je v organizacijski ontologiji izdelan tudi odločitveni model lastnika opravila, so s pomočjo stroja za sklepanje kandidati razvrščeni po primernosti. V naslednjih izvedbah primerkov za delovno mesto istega tipa, se odločitveni model ponovno uporabi in posredovanje osebe, ki je lastnik opravila, ni več potrebno. Seveda če lastnik opravila želi, lahko rezultat dobi v pregled in potrditev. V tem primeru gre za polavtomatizirano opravilo.

8.8.2 Primer druge vrste odločanja

V drugi vrsti odločanja je opis alternativ po določenih kriterijih naloga lastnika opravila, ki sprejme odločitev, saj ni zajeta v okviru predhodnih korakov poslovnega procesa in tako ne more biti posredovana z vhodnimi podatki. Način kako so kriteriji opisani je odvisen od posamezne osebe, ki odločitev sprejema, in je lahko za različne osebe drugačen. Primer takšnega odločanja lahko najdemo v procesu ocenjevanja člankov na konferenci. Primer takšnega procesa je sestavljen iz naslednjih korakov:

1) Prejemanje člankov,

2) Pošiljanje potrdila o prejetih člankih avtorjem,

3) Pregled in ocenjevanje člankov,

4) Izbira sprejetih člankov,

5) Obveščanje avtorjev,

6) Prejem končnih verzij člankov.

Najprej bomo opisali, kako se proces izvaja brez uvedbe MAPP, in nato, kako implementacija modela izboljša njegovo izvajanje.

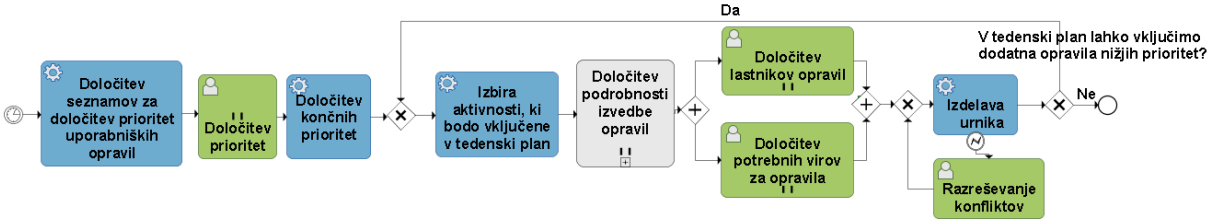
Prejemanje člankov poteka preko spletne aplikacije, v kateri lahko avtorji članke oddajo v elektronski obliki. Za vsak prejeti članek je avtorjem avtomatsko poslano potrdilo o uspešnem prejemu članka. Po preteku časovnega roka za oddajo, so članki posredovani recenzentom, ki jih pregledajo in ocenijo. Vsak članek je posredovan trem recenzentom. Gre torej za vzporedno izvedbo več primerkov uporabniškega opravila, katerih lastniki so recenzenti. Rezultat uporabniškega opravila pregleda in ocenitve članka je predlog recenzenta o tem, ali naj bo članek sprejet ali ne. Ko vsi recenzenti oddajo ocene člankov, so izbrani članki z najvišjimi povprečnimi ocenami. V danem primeru je to avtomatska storitev. Nato so avtorji obveščeni o tem, ali je bil njihov članek sprejet ali ne. Obveščanje je realizirano avtomatsko s pošiljanjem obvestila preko elektronske pošte. Sledi prejem končnih verzij člankov preko spletne aplikacije. Proces ocenjevanja člankov se s tem zaključi.

Z vidika poslovnega sistema, ki izbira članke za konferenco, so avtomatizirana opravila prejemanje člankov, pošiljanje potrdila o prejetih člankih avtorjem, izbira sprejetih člankov, obveščanje avtorjev in prejem končnih verzij člankov. Uporabniška opravila poslovnega procesa so torej primerki opravila pregled in ocenjevanje člankov. Končni rezultat tega opravila je odločitev recenzenta o tem, ali naj bo članek sprejet ali zavrnjen. V primerjavi s predhodnim primerom izbire najboljšega kandidata za zaposlitev, v tem primeru alternative (članki) v ontologiji niso opisani enakovredno, ampak je njihov opis odvisen od posameznega recenzenta. V primeru izbire najboljšega kandidata za zaposlitev so vsi kandidati in informacije v zvezi z njimi, ki predstavljajo osnovo za odločitev, po izvedbi preslikave procesnih podatkov v ontologijo že zajete v ontologiji. Razlika je le v odločitvenih modelih posameznih odločevalcev. V nasprotju s tem pa se v primeru ocenjevanja člankov razlikujejo tudi informacije o samih alternativah, ki so del ontologij posameznih odločevalcev. Na primer

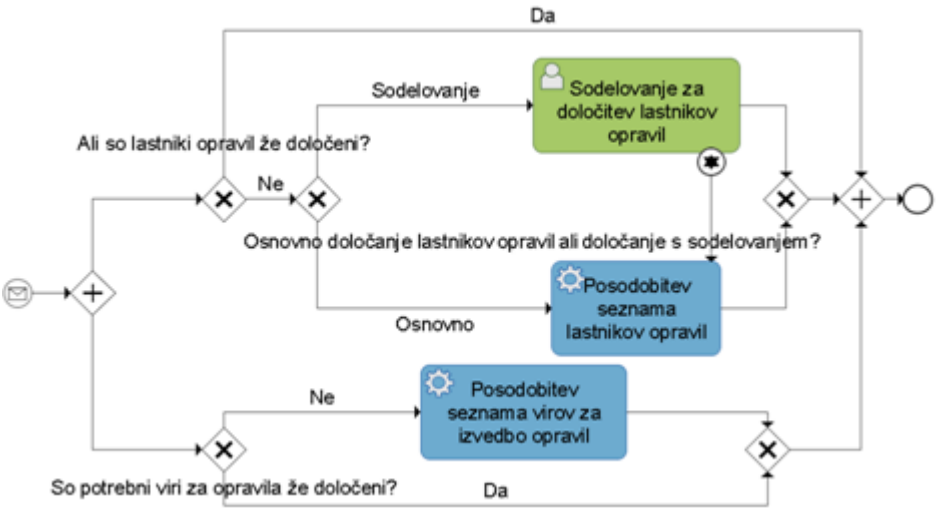
ob pregledu članka, recenzent zajame informacije, ki se mu zdijo relevantne za sprejem članka. Te informacije so v odvisnosti od posameznih recenzentov lahko različne. Na podlagi teh informacij recenzent opredeli kriterije in odločitveni model. V tem primeru odločitev ne more biti avtomatizirana, saj je vedno potreben opis alternativ, ki ga izvede lastnik opravila. Kljub temu pa ima pristop pomembno prednost v možnosti ponovne uporabe izdelanega odločitvenega modela.

8.9 Študij primera *Izdelava tedenskega plana*

Kot primer bomo obravnavali poslovni proces *Izdelava tedenskega plana* v poslovnem sistemu ABC. Model procesa je prikazan na spodnji sliki (Slika 20). Slika 21 prikazuje njegov podproces *Določitev podrobnosti izvedbe opravil*.



Slika 20: Poslovni proces določanja tedenskega plana v notaciji BPMN



Slika 21: Podproces Določitev podrobnosti izvedbe opravil v notaciji BPMN

Velik del procesa *Izdelava tedenskega plana* predstavljajo uporabniška opravila. Na slikah so uporabniška opravila zasenčena z zeleno barvo, avtomatizirana opravila pa z modro barvo. Proces se izvaja enkrat tedensko ob vnaprej določenem času.

Izdelava tedenskega plana zajema obravnavo različnih predlogov opravil, ki so kandidati za vključitev v tedenski plan. Določeni predlogi opravil že vključujejo prioriteto opravila. Tistim, ki prioritete še nimajo podane, se jo doda v opravilu *Določitev prioritet*. *Določitev prioritet* je uporabniško opravilo, v katerem lastnik opravila določi prioritete posameznim predlogom opravil, ki so mu posredovani kot vhod v opravilo. Praviloma so lastniki opravil *Določitev prioritet* osebe odgovorne za opravila, ki jim določajo prioritete, na primer vodje organizacijskih enot. Ker lahko posamezno opravilo zadeva več kot eno organizacijsko enoto, je lahko več oseb odgovornih za določitev prioritet. V primeru, ko so za posamezen predlog opravila določene različne prioritete, je potrebno posebej določiti končne prioritete. Te se določijo v opravilu *Določitev končnih prioritet*, ki je avtomatizirano opravilo. Končne prioritete se določi glede na vpletenost posameznih organizacijskih enot v opravilo. V naslednji aktivnosti so avtomatsko izbrani predlogi opravil z najvišjimi prioritetami (*Izbira aktivnosti, ki bodo vključene v tedenski plan*). Pred izdelavo urnika opravil je potrebno določiti še vire, potrebne za izvedbo opravil, in osebe, ki bodo opravila izvedle. Včasih so ti podatki že določeni s predlogom opravila, pogosto pa jih je potrebno določiti v okviru izdelave tedenskega plana.

Izbira izvajalca opravila (lastnika opravila) je trivialna, če je za izvedbo opravila usposobljen le določeni zaposlenec. Kljub temu so pogostejši primeri, ko je za posamezno opravilo usposobljenih več zaposlencev. V teh primerih lahko osebo, ki bi naj izvedla opravilo, določimo na tri načine:

1 Odgovorna oseba, na primer vodja organizacijske enote, razvrsti usposobljene zaposlence od najprimernejšega oziroma najprimernejše skupine zaposlencev za izvedbo danega opravila do najmanj primernega zaposlenca oziroma skupine zaposlencev. Gre za opravilo, ki zahteva odločitev. Odvisna je od preferenc odločevalca in strategije, ki jo uporablja za svoje podrejene. Odločitev tako lahko izvede na podlagi njihovih predhodnih rezultatov in izkušenj pri podobnih opravilih.

2 Odgovorna oseba, na primer vodja organizacijske enote, določi potencialne lastnike opravil, pri čemer se konkretni izvajalec določi pred izvedbo opravila. Odgovorna oseba na primer določi potencialne lastnike z vlogo V. Ko je uporabniško opravilo potrebno

izvesti je posredovano vsem osebam, ki igrajo vlogo V. Prvi izmed njih, ki opravilo prevzame, postane lastnik opravila. S tem je opravilo odstranjeno s seznama opravil ostalih potencialnih lastnikov (ostalih oseb, ki igrajo vlogo V).

3 Tretji način določanja zaposlencev za izvedbo določenega opravila je s sodelovanjem z zaposlenci, ki so usposobljeni za izvedbo opravila. Primeri, kdaj je ta pristop potreben in kako ga je potrebno izvajati, je določen z poslovnimi pravili in politikami poslovnega sistema. Primer je, kadar je določeno opravilo potrebno izvesti s službenim potovanjem. V tem primeru ima na primer posamezni zaposlenec možnost, da se odloči ali bo opravilo izvedel ali ne, in poda natančnejši predlog, kdaj ga je pripravljen izvesti (znotraj podanega časovnega intervala opravila). V tem primeru odgovorna oseba ustreznim zaposlencem pošlje sporočilo CFP. Vsak izmed njih nato bodisi odkloni ali pošlje predlog. Odgovorna oseba nato na podlagi prejetih predlogov in morebitnih drugih kriterijev o zaposlencih, izbere tistega, ki ga bo zadolžil za lastnika opravila in ga obvesti. V primeru, da nihče ne odgovori pozitivno (vsi ustrezni zaposlenci odklonijo), se proži izjema. Kako se posreduje v takšnih primerih je prav tako določeno v poslovnih pravilih. Na primer, odgovorna oseba lahko v tem primeru določi zaposlenca, ne glede na to ali je odklonil ali ne.

Prvi in drugi način sta obravnavana v opravilu *Določitev lastnikov opravil* (Slika 20). Pri tem je v posameznem primerku tega opravila lastniku opravila posredovan seznam vseh opravil, ki jim morajo določiti izvajalce.

Tretji način je zajet v opravilu *Sodelovanje za določitev lastnikov opravil*.

Podproces *Določitev podrobnosti izvedbe opravil* se izvede za vsako opravilo, ki je izbrano za vključitev v tedenski plan. V okviru tega podprocesa so izdelani seznami opravil, ki jim je potrebno določiti vire, in seznami opravil, ki jim je potrebno določiti izvajalce; gre za eden seznam za vsako odgovorno osebo, ki opravilom dodeljuje potrebne vire, in eden seznam za vsako odgovorno osebo, ki opravilom dodeljuje izvajalce. Vsa opravila ne zahtevajo posebnih virov. Med opravili, ki jih najpogosteje zahtevajo, so fizična opravila. Ti seznami so del vhodnih podatkov opravil *Določitev lastnikov opravil* in *Določitev potrebnih virov za opravila*. V primeru, ko je za določitev izvajalca potrebno sodelovanje, se izvede sodelovanje in določi izvajalca. V tem primeru opravilo ni vključeno v nobenega izmed seznamov za določitev lastnikov opravil.

Ko so ovrednoteni lastniki opravil od najprimernejših do najmanj primernih in določeni potrebni viri, se izdela urnik na podlagi časovnih omejitev, razpoložljivih virov in z iskanjem najvišje povprečne vrednosti pri določanju najprimernejših kandidatov za lastnike opravil. Gre za avtomatizirano opravilo. V primeru konfliktnih situacij med opravili, se proži napaka in je obveščena odgovorna oseba. Ta mora identificirati razlog za napako in izvesti ustrezen ukrep. Najpogosteje je težava v pomanjkanju virov ali prezasedenosti zaposlencev.

Če urnik dopušča vključitev dodatnih opravil v tedenski plan, se proces vrne v opravilo *Izbira aktivnosti, ki bodo vključene v tedenski plan*, ki začne nov cikel z predlogi opravil nižje prioritete. Če urnik ne dopušča vključitve dodatnih opravil v tedenski plan se proces zaključi.

Analizirajmo uporabniška opravila poslovnega procesa in možnosti njihove avtomatizacije z modelom za avtomatizacijo uporabniških opravil:

- *Določitev prioritet*:

  Način kako se določajo prioritete opravil je opredeljen s poslovnimi pravili, na primer glede na kritičnost za poslovni proces. Dovolj natančno opredeljena pravila za določitev prioritet pomenijo, da je postopek izvedbe opravila preddefiniran. Avtomatizacija določitve prioritet torej ne bi spadala v SIUO, saj bi naj bilo opravilo neodvisno od posameznega lastnika opravila. Z vidika posamezne organizacijske enote so sicer prioritete lahko drugačne, vendar bi naj tudi to določala poslovna pravila. Prav tako bi naj poslovna pravila določala, kako se na podlagi prioritet opravil v posameznih organizacijskih enotah, določijo globalne prioritete, kar je že avtomatsko opravilo (*Določitev končnih prioritet*).

- *Sodelovanje za določitev lastnikov opravil*:

  V primerih, ko je za določitev lastnikov opravil potrebno sodelovanje, je sodelovanje določeno s pravili sodelovanja. Ta se lahko z uporabo modela za avtomatizacijo uporabniških opravil preslikajo v pravila v knjižnici protokolov. Izvedba sodelovanja je tako lahko avtomatizirana z agenti večagentnega sistema in njihovo uporabo SOO, ki implementira knjižnico protokolov. V primeru določanja zaposlenca, ki bo opravilo izvedel na službenem potovanju, gre za uporabo protokola pogodbene mreže. Njegov zapis z vidika agenta, ki zastopa odgovorno osebo in tako predstavlja vlogo iniciatorja, podaja Tabela 2.

- *Določitev lastnikov opravil*:

  Določitev lastnikov opravil se lahko določi z vlogo, ki jo igra ustrezni zaposlenec, ali z razvrstitvijo ustreznih zaposlencev oziroma skupine zaposlencev, če je potreben več kot eden izvajalec opravila, od najprimernejšega do najmanj primernega. Ločimo lahko opravila, ki so izrednega značaja in vnaprej opredeljena opravila. Za vnaprej opredeljena opravila velja, da so ponovljivega značaja in imajo opredelitev v SIUO. Zanje so potencialni lastniki določeni že s samim opravilom. To pomeni, da je vloga že določena z opravilom. Odgovorna oseba, ki za konkretno opravilo potencialne lastnike razvrsti od najprimernejšega do najmanj primernega torej izvaja razvrščanje na podlagi množice potencialnih lastnikov. Pri tem gre v večini primerov za prvo vrsto odločanja, saj se lastnik opravila *Določitev lastnikov opravil* praviloma odloča na podlagi znanja, preteklih izkušenj in uspešnosti zaposlencev. Če so te informacije del ontologije, je lahko opravilo *Določitev lastnikov opravil* po opredelitvi odločitvenega modela popolnoma avtomatizirano. Ob prvi izvedbi uporabniškega opravila za določitev izvajalcev vnaprej opredeljenega opravila izdela odločitveni model, na podlagi katerega določi ustrezne lastnike opravil. V vseh nadaljnjih izvedbah določitve lastnikov opravil za isto vrsto opravila je lahko odločitev avtomatizirana. Primer podmnožice odločitvenih pravil za določitev lastnikov za opravilo razreševanja napake na sistemu XS12 je podan v spodnji tabeli (Tabela 14).

  V primeru izrednih opravil potencialni lastniki niso opredeljeni vnaprej, ampak jih mora določiti odgovorna oseba. V tem primeru zaradi nepredvidljive narave izrednih opravil določitev lastnikov ni avtomatizirana.

$$NapakaSrednjePrioritete(?x) \ \wedge \ SrednjeIzkusenIPSzXS12(?z) \ \wedge$$

$$\rightarrow \ imaClana1(?b,?z) \ \wedge \ imaClana2(?b,?y)$$
$$\wedge \ imaNajprimernejsoSkupinoZaRazresitev(?x,?b)$$

$$NapakaSrednjePrioritete(?x) \wedge SrednjeIzkusenIPSzXS12(?z) \wedge UspešenIPSzXS12(?z) \wedge igraVlogo(?z,?a) \wedge lahkoRazresi(?a,?x) \wedge IPSZačetnikzXS12(?y) \wedge igraVlogo(?y,?ay) \wedge lahkoRazresi(?ay,?x) \wedge SkupinaIPS(?b) \rightarrow imaClana1(?b,?z) \wedge imaClana2(?b,?y) \wedge imaNajprimernejsoSkupinoZaRazresitev(?x,?b)$$

$$NapakaSrednjePrioritete(?x) \wedge ZeloIzkusenIPSzXS12(?z) \wedge VisokoUspešenIPSzXS12(?z) \wedge igraVlogo(?z,?a) \wedge lahkoRazresi(?a,?x) \wedge IPSZačetnikzXS12(?y) \wedge igraVlogo(?y,?ay) \wedge lahkoRazresi(?ay,?x) \wedge SkupinaIPS(?b) \rightarrow imaClana1(?b,?z) \wedge imaClana2(?b,?y) \wedge imaZeloPrimernoSkupinoZaRazresitev(?x,?b)$$

$$NapakaSrednjePrioritete(?x) \wedge ZeloIzkusenIPSzXS12(?z) \wedge VisokoUspešenIPSzXS12(?z) \wedge igraVlogo(?z,?a) \wedge lahkoRazresi(?a,?x) \wedge IPSBrezIzkusenjzXS12(?y) \wedge igraVlogo(?y,?ay) \wedge lahkoRazresi(?ay,?x) \wedge SkupinaIPS(?b) \rightarrow imaClana1(?b,?z) \wedge imaClana2(?b,?y) \wedge imaZeloPrimernoSkupinoZaRazresitev(?x,?b)$$

$$NapakaSrednjePrioritete(?x) \wedge ZeloIzkusenIPSzXS12(?z) \wedge VisokoUspešenIPSzXS12(?z) \wedge igraVlogo(?z,?a) \wedge lahkoRazresi(?a,?x) \wedge ZeloIzkusenIPSzXS12(?y) \wedge VisokoUspešenIPSzXS12(?y) \wedge igraVlogo(?y,?ay) \wedge lahkoRazresi(?ay,?x) \wedge differentFrom(?z,?y) \wedge SkupinaIPS(?b)$$

$$\rightarrow imaClana1(?b,?z) \wedge imaClana2(?b,?z) \wedge imaSprejemljivoSkupinoZaRazresitev(?x,?b)$$

$$NapakaSrednjePrioritete(?x) \wedge IPSZačetnikzXS12(?z) \wedge igraVlogo(?z,?a) \wedge lahkoRazresi(?a,?x) \wedge IPSZačetnikzXS12(?y) \wedge differentFrom(?z,?y) \wedge igraVlogo(?y,?ay) \wedge lahkoRazresi(?ay,?x) \wedge SkupinaIPS(?b)$$

$$\rightarrow imaClana1(?b,?z) \wedge imaClana2(?b,?y) \wedge imaManjPrimernoSkupinoZaRazresitev(?x,?b)$$

Tabela 14: Primer odločitvenih opravil za določitev lastnikov opravil

- *Določitev potrebnih virov za opravila:*

  Odgovorna oseba določi potrebne vire za opravila. Podobno kot pri določitvi lastnikov opravil, so tudi potrebni viri lahko določeni z opredelitvijo opravila. Tisti, ki določa potrebne vire, jih v primeru, da gre za izbiro med več alternativami, ki ustrezajo, lahko razvrsti od najprimernejšega do najmanj primernega, čeprav pri virih navadno zadostuje alokacija enega izmed množice razpoložljivih ustreznih virov. Pri tem zaradi raznovrstnosti opravil in različnih situacij, v katerih so potrebna, določitev potrebnih virov za opravila ne avtomatiziramo.

- *Razreševanje konfliktov:*

  Razreševanje konflikta je uporabniško opravilo, ki pa zaradi nepredvidljive narave ni avtomatizirano.

Na podlagi navedenega lahko zaključimo, da uporaba MAPP avtomatizira dva tipa uporabniških opravil v poslovnem procesu. Eden izmed tipov uporabniških opravil procesa ni primeren za avtomatizacijo v okviru storitve za izvajanje uporabniških opravil (glej poglavje 4 Uporabniška opravila in analiza primernosti njihove avtomatizacije s storitvijo za izvajanje uporabniških opravil (SIUO)). Dva tipa zaradi nepredvidljive narave nista avtomatizirana. To pomeni, da za dani proces model lahko avtomatizira polovico tipov uporabniških opravil, ki so smiselna za avtomatizacijo v okviru SIUO. Opravilo *Določitev lastnikov opravil* podaja primer prve vrste odločanja, ki je lahko popolnoma avtomatizirano. Opravilo *Sodelovanje za*

določitev lastnikov opravil je sodelovalno opravilo, ki je prav tako lahko popolnoma avtomatizirano: avtomatizirane so lahko tako kompozicijske funkcije elementarnih opravil kot elementarna opravila, ki zahtevajo odločitve zaposlencev o tem ali so pripravljeni izvesti opravilo ali ne ter odločitev odgovorne osebe o končni izbiri zaposlenca. S primerom smo tako ponazorili, kako lahko model za avtomatizacijo uporabniških opravil pomembno pripomore k višji ravni avtomatizacije poslovnega procesa.

8.10 Sledljivost in proaktivnost

Poleg izboljšanja učinkovitosti poslovnih procesov z doseganjem višje ravni avtomatizacije, model za avtomatizacijo uporabniških opravil pomeni tudi nekatere druge pridobitve v izvajanju poslovnih procesov. Pomembna lastnost odločanja je zmožnost razlage o tem, kako je bila odločitev sprejeta (sledljivost odločitve). V kontekstu poslovnih procesov in storitveno usmerjene arhitekture je to še posebej pomembno, ko preučujemo izvedbo posameznih primerkov poslovnih procesov. V obstoječih pristopih k implementaciji uporabniških opravil v SOA sistemih je rezultat opravila posredovan primerku poslovnega procesa, ki je opravilo prožilo, pri čemer je zmožnost ugotavljanja tega, kako je lastnik opravila prišel do rezultata zelo omejena. V primeru odločitve, ki je rezultat miselnega procesa lastnika opravila, razlaga ni mogoča. Tudi v primeru odločitve, pri kateri je lastnik opravila uporabil določen odločitveni sistem in z njegovo pomočjo prišel do rezultata, je povezavo med sprejeto odločitvijo in poslovnim procesov težko ugotoviti. V tem primeru prav tako ni neposredne povezave med procesnimi podatki, ki so pogosto nosilci informacij za odločitev, in uporabljenim odločitvenim sistemom. To pomeni, da je moral lastnik opravila te podatke ustrezno prenesti v uporabljeni odločitveni sistem. Takšen pristop je izpostavljen človeškim napakam ter napakam, ki lahko izvirajo iz napačne interpretacije procesnih podatkov ter kriterijev v odločitvenem modelu.

Z uporabo predlaganega modela, lahko s pomočjo preslikav opredeljenih v uporabniških opravilih in odločitvenega modela lastnika opravila, natančno določimo kako in na podlagi katerih informacij je le-ta prišel do odločitve in kako je to vplivalo na dani primerek poslovnega procesa. Preko rezultata opravila, preslikav med nosilci informacij procesnih podatkov ter organizacijsko ontologijo in odločitvenega modela uporabljenega pri izvedbi uporabniškega opravila lahko analiziramo vpliv odločitve na želenem nivoju podrobnosti, od

odločitve kot celote do posameznih kriterijev oziroma sklopov kriterijev, ki so na odločitev vplivali. To pomeni, da lahko določimo natančen razlog za določen zaključek primerka poslovnega procesa v odvisnosti od sprejete odločitve. S tem je povišana raven granularnosti nadzora in analize poslovnih procesov, proučujemo lahko morebitne izboljšave poslovnih procesov in posameznih odločitvenih modelov. Prav tako je mogoča analiza vpliva posledic na izvajanje drugih poslovnih procesov, ki so odvisni od rezultatov procesa, ki je uporabniško opravilo prožilo, v odvisnosti posamezne odločitve.

Poleg tega model omogoča, da v času izvedbe opravila analiziramo različne možne izide primerka poslovnega procesa, ki je opravilo prožilo, ter od njega odvisnih primerkov poslovnih procesov, kot nadgradnjo kaj-če analize odločitvenega modela. Če v uporabniški vmesnik, ki implementira izvedbo uporabniškega opravila, integriramo model procesa in potek primerka procesa do točke, ko je bilo uporabniško opravilo proženo, lahko uporabnik ugotavlja, kako bo njegov odločitveni model vplival na izid procesa in posredno z njim povezanih procesov. S tem je omogočena višja raven proaktivnosti v izvajanju poslovnih procesov, ki je predvsem pomembna z vidika zmanjšanja tveganj pri izvedbi poslovnih procesov.

9 Zaključek

Z naraščajočim pomenom avtomatizacije poslovnih procesov raziskovalci zelo raznolikih področij iščejo nove možnosti doseganja izboljšanja obstoječega stanja, na primer na področjih programskih in spletnih tehnologij, inteligentnih senzorjev, umetne inteligence, robotike, strojništva itd. Pričujoča doktorska disertacija opredeljuje inovativni model za doseganje višje ravni avtomatizacije na podlagi tehnologij obvladovanja znanja, pri čemer obvladovanje znanja prenese v kontekst izvajanja poslovnih procesov in uporabniških opravil. Namen je avtomatizacija in polavtomatizacija določenih vrst uporabniških opravil, in sicer se osredotoča na tista uporabniška opravila, ki jih je smiselno avtomatizirati v okviru storitve za izvajanje uporabniških opravil. Identificiramo tri vrste uporabniških opravil, ki so primerna za avtomatizacijo v okviru SIUO: miselna uporabniška opravila na nivoju elementarnih uporabniških opravil, ter sodelovalna opravila in opravila, ki temeljijo na osnovnih vzorcih kompozicije, na nivoju sestavljenih uporabniških opravil. Model uvaja pomembne pridobitve. Z namenom avtomatizacije miselnih in sodelovalnih opravil, opredeljuje organizacijsko ontologijo, v kateri igrajo ključno vlogo odločitveni modeli, ki pripadajo različnim organizacijskim elementom. Ker je odločanje ena izmed najpomembnejših aktivnosti v poslovnih sistemih, se model nanaša na izboljšanje velikega dela miselnih opravil. Z namenom podpore in avtomatizacije sodelovalnih opravil je model razširjen z agentnimi tehnologijami in knjižnico protokolov, ki je integrirana v organizacijski ontologiji. V določenih primerih je zaradi kritičnih vprašanj odgovornosti polavtomatizacija, ki zahteva eksplicitno potrditev s strani lastnika opravila, primernejša od avtomatizacije. Z razširjenimi možnostmi avtomatizacije uporabniških opravil in s tem izboljšanja njihovega časa izvajanja ob enaki kakovosti izvedbe in enako kakovostnih rezultatih, je izboljšana avtomatizacija izvajanja poslovnih procesov. S tem prispevki doktorske disertacije pomenijo pomemben korak bliže k viziji celovite avtomatizacije poslovnih procesov. Poleg tega je pomemben prispevek modela omogočanje višje ravni sledljivosti odločitev, in sicer od razlage o tem, kako je bila odločitev sprejeta, do proaktivne identifikacije posledic odločitve na izvedbo poslovnega procesa in njegovih rezultatov.

S predstavljenimi rezultati smo potrdili vse v tretjem poglavju postavljene hipoteze:

- Hipoteza 1:

Z definicijo in uporabo arhitekturnega modela, ki temelji na tehnologijah obvladovanja znanja, je možno doseči višjo raven avtomatizacije uporabniških opravil kot dela poslovnih procesov.

Zasnovo arhitekturnega modela smo opredelili v petem poglavju. Ključni uporabljeni tehnologiji obvladovanja znanja sta ontologije in večagentni sistemi. V šestem poglavju smo podali metodo odločanja, ki temelji na ontologijah. V sedmem poglavju smo podali metodo za avtomatizacijo sodelovalnih opravil s pomočjo večagentnega sistema in ontologij. V osmem poglavju smo zasnovo arhitekturnega modela dopolnili, tako da omogoča vključitev metod podanih v šestem in sedmem poglavju, in prikazali na kakšen način model dosega avtomatizacijo in polavtomatizacijo miselnih in sodelovalnih uporabniških opravil.

- Hipoteza 1.1:

Izdelati je mogoče metodo večkriterijskega odločanja, ki temelji na ontologijah in ki ob uporabi v storitvah za izvajanje uporabniških opravil v SOA sistemih omogoča višjo stopnjo avtomatizacije izvajanja uporabniških opravil, ki temeljijo na odločitvah.

Metoda večkriterijskega odločanja, ki temelji na ontologijah, je podana v šestem poglavju. V osmem poglavju je opisan način, kako se uporabi v organizacijski ontologiji, kako je odločanje na podlagi v ontologiji zapisanih odločitvenih modelov integrirano v izvajanje uporabniških opravil in kako se s tem doseže avtomatsko oziroma polavtomatsko izvajanje uporabniških opravil, ki temeljijo na odločitvah.

- Hipoteza 1.2:

Izdelati je mogoče večagentni sistem, ki ob uporabi v storitvah za izvajanje uporabniških opravil v SOA sistemih omogoča koordinacijo uporabniških opravil in avtomatizira sodelovanje, ki temelji na poslovnih protokolih.

V petem poglavju je v zasnovi arhitekturnega modela podana struktura večagentnega sistema in vloge posameznih agentov, ki izvajajo koordinacijo uporabniških opravil. V sedmem poglavju je funkcionalnost večagentnega sistema nadgrajena in je opisano, kako dosega avtomatizacijo sodelovalnih uporabniških opravil. Opredeljen je koncept knjižnice protokolov, ki določajo pravila sodelovanja. Večagentni sistem je podan na način, ki omogoča implementacijo s poljubno tehnologijo in po poljubni metodologiji, pri čemer so opredeljene

zahteve, ki jim mora ustrezati organizacijska ontologija in zahteve, ki jih morajo zadovoljevati agenti z namenom doseganja avtomatizacije sodelovalnih pravil.

- Hipoteza 2:

Z izdelavo in uporabo modela storitve za izvajanje uporabniških opravil, ki omogoča doseganje višje ravni avtomatizacije uporabniških opravil kot dela poslovnih procesov, je mogoče doseči višjo raven sledljivosti odločitev v poslovnem procesu in višjo raven proaktivnosti.

Model storitve za izvajanje uporabniških opravil, ki omogoča doseganje višje ravni avtomatizacije uporabniških opravil kot dela poslovnih procesov, smo podali v okviru potrjevanja hipoteze 1. Možnost doseganja višje ravni sledljivosti odločitev z uporabo modela, smo opisali v poglavju 6.7 Analiza alternativ in v poglavju 8.10 Sledljivost in proaktivnost. Gre za sledljivost v povezavi z izvajanjem primerka poslovnega procesa, pri čemer lahko analiziramo rezultat procesa v odvisnosti od odločitve sprejete v uporabniškem opravilu. Preko rezultata opravila, preslikav med nosilci informacij procesnih podatkov ter organizacijsko ontologijo in odločitvenega modela uporabljenega pri izvedbi uporabniškega opravila lahko analiziramo vpliv odločitve na želenem nivoju podrobnosti, od odločitve kot celote do posameznih kriterijev oziroma sklopov kriterijev, ki so na odločitev vplivali. Prav tako je mogoča analiza vpliva posledic na izvajanje drugih poslovnih procesov, ki so odvisni od rezultatov procesa, ki je uporabniško opravilo prožilo. Poleg tega model omogoča, da v času izvedbe opravila analiziramo različne možne izide instance poslovnega procesa, ki je opravilo prožila, ter od nje odvisnih instanc poslovnih procesov, kot nadgradnjo kaj-če analize odločitvenega modela. S tem je omogočena višja raven proaktivnosti v izvajanju poslovnih procesov.

- Hipoteza 3:

Aktivnosti opredelitve in uporabe modela za doseganje višje ravni avtomatizacije uporabniških opravil, ki temelji na tehnologijah obvladovanja znanja, dopolnjujejo in razširjajo obstoječe aktivnosti življenjskega cikla sistemov obvladovanja znanja ter zmanjšujejo razhajanja med področjem poslovnih procesov in sistemi obvladovanja znanja z neposredno povezavo med aktivnostmi modeliranja in implementacije poslovnih procesov ter aktivnostmi obvladovanja znanja.

V osmem poglavju smo opisali organizacijsko ontologijo in njeno povezanost s poslovnim procesom. Opredelili smo aktivnosti, ki so potrebne za doseganje višje ravni avtomatizacije, ter kako se povezujejo s aktivnostmi opredelitve in izvajanja poslovnih procesov. Posebej smo obravnavali aktivnosti, ki se nanašajo na področje obvladovanja znanja in življenjski cikel sistemov obvladovanja znanja, ki temeljijo na ontologijah. Ugotovili smo, da se nanašajo predvsem na aktivnosti evolucije ontologije. Umestili smo jih v dve metodologiji: On-To-Knowledge, ki podaja celoten življenjski cikel sistemov za obvladovanje znanja, ki temeljijo na ontologijah, in metodologijo po Maedche et al., ki podrobneje obravnava evolucijo ontologije.

Na koncu podajamo strnjen pregled prispevkov te doktorske disertacije:

1 Podana je metoda odločanja, ki temelji na večkriterijskem odločanju in odločitveni model gradi v OWL ontologiji. Metoda prispeva tako k samemu razvoju ontologij kot uporabi ontologije, ki ji dvigne praktično vrednost predvsem v smislu doseganja višje ravni avtomatizacije odločanja.

2 Iz metode odločanja, ki temelji na večkriterijskem odločanju in odločitveni model gradi v OWL ontologiji, je izpeljan model, ki omogoča njeno uporabo v izvajanju uporabniških opravil kot dela izvajanja poslovnih procesov. S tem metoda dvigne avtomatizacijo poslovnih procesov na višjo raven.

3 Podan je inovativen pristop k opredelitvi protokolov sodelovanja, ki so osnova za sodelovanje med agenti večagentnih sistemov. Pristop temelji na predpostavki, da so poslovni protokoli sodelovanja, kot del poslovnega sistema, izpostavljeni spremembam in zato njihova neposredna implementacija v obnašanju agentov večagentnega sistema ni primerna. Pristop omogoča ločitev protokola od implementacije ter vzpostavitev šibke sklopljenosti protokola in implementacije. To se odraža v tem, da spremembe protokolov ne zahtevajo ali minimizirajo spremembe implementacije obnašanja agentov, ki protokole uporabljajo za svoje delovanje.

4 Pristop k opredelitvi protokolov sodelovanja iz točke tri omogoča hierarhično opredeljevanje protokolov in s tem višjo raven ponovne uporabe protokolov sodelovanja ter enostavno opredeljevanje različic protokola.

5 Opredeljeni model omogoča vključitev delovanja skladno s protokoli sodelovanja iz prejšnje točke v izvajanje uporabniških opravil in s tem avtomatizacijo sodelovanja.

6 Opredeljena je razširitev specifikacij WS-HumanTask, ki omogoča uporabo modela skladno z obstoječimi specifikacijami in orodji, ki jih podpirajo.

7 Uporaba modela vključuje aktivnosti povezane z obvladovanjem znanja. Te se lahko uporabljajo kot del aktivnosti razvoja in vzdrževanja sistemov za obvladovanje znanja. Podana je njihova umestitev v metodologijo On-To-Knowledge razvoja sistemov za obvladovanje znanja, ki temeljijo na ontologijah, in v proces evolucije ontologije podan v Medche et al. [49].

8 Identificirani sta vrsti odločanja, kot dela izvajanja poslovnih procesov, ter analizirane možnosti njune avtomatizacije in polavtomatizacije s podano metodo odločanja, ki temelji na večkriterijskem odločanju in je zgrajena v OWL ontologiji.

9 Podana je klasifikacija ravni podpore uporabniškim opravilom v SOA sistemih za izvajanje poslovnih procesov.

10 Podana je analiza uporabniških opravil ter možnosti in primernost avtomatizacije posameznih vrst ter posameznih sestavnih delov uporabniških opravil v okviru storitve za izvajanje uporabniških opravil.

S tem pričujoča doktorska disertacija podaja celostni doprinos k izboljšanju avtomatizacije uporabniških opravil v okviru izvajanja poslovnih procesov. S potrditvijo hipotez in izvedbo zadanih nalog predstavlja pomemben interdisciplinaren doprinos na področju avtomatizacije poslovnih procesov, storitveno usmerjenih arhitektur, večagentnih sistemov, odločitvenih sistemov in sistemov za obvladovanje znanja.

Literatura

[1] A. Abecker, A. Bernardi, L. van Elst, "Agent Technology For Distributed Organizational Memories: The Frodo Project," v zborniku *5th International Conference On Enterprise Information Systems*, Angers, France, 2003, str. 3-10.

[2] A. Abecker, A. Bernardi, K. Hinkelmann, O. Kühn, M. Sintek, "Toward a technology for organizational memories," *IEEE Intelligent Systems*, 13(3): 40–48, 1998.

[3] A. Abecker, S. Decker, "Organizational memory: Knowledge acquisition, integration and retrieval issues," v zborniku: *XPS-99 / 5. Deutsche Tagung Wissensbasierte Systeme*, Wurzburg, Nemčija, 1999, str. 113-124.

[4] A. Agrawal, M. Amend, M. Das, M. Ford, C. Keller, M. Kloppmann, D. Konig, F. Leymann, R. Muller, G. Pfau, K. Plosser, R. Rangaswamy, A. Rickayzen, M. Rowley, P. Schmidt, I. Trickovic, A. Yiu, M. Zeller, "WS-BPEL Extension for People (BPEL4People), Version 1.0," junij 2007, dostopno na: http://www.ibm.com/developerworks/webservices/library/specification/ws-bpel4people/.

[5] A. Agrawal, M. Amend, M. Das, M. Ford, C. Keller, M. Kloppmann, D. Konig, F. Leymann, R. Muller, G. Pfau, K. Plosser, R. Rangaswamy, A. Rickayzen, M. Rowley, P. Schmidt, I. Trickovic, A. Yiu, M. Zeller, "Web Services Human Task (WS-HumanTask), Version 1.0," junij 2007, dostopno na: http://www.ibm.com/developerworks/webservices/library/specification/ws-bpel4people/.

[6] S. Ali, B. Soh, T. Torabi, "A Novel Approach Toward Integration of Rules Into Business Process Using An Agent-Oriented Framework," *IEEE Transactions on Industrial Informatics,* 2(3): 145-154, 2006.

[7] S. Alter, *Information Systems: Foundation of e-business*, Prentice Hall College, 2001.

[8] T. Andrews, F. Curbera, H. Dholakia, Y. Goland, J. Klein, F. Leymann, K. Liu, D. Roller, D. Smith, S. Thatte, I. Trickovic, S. Weerawarana, "Business Process Execution

Language for Web Services, Version 1.1," maj 2003, dostopno na: http://xml.coverpages.org/BPELv11-May052003Final.pdf.

[9] J.L. Austin, *How to Do Things With Words*, Harvard University Press, Cambridge, UK, 1962.

[10] S. Ba, K.R. Lang, A.B. Whinston, "Compositional enterprise modeling and decision support," *Information Systems and E-Business Management*, Vol. 6(2): 137-160, marec 2008.

[11] J. Bermúdez, A**.** Goñi, A. Illarramendi, M.I. Bagüés, "Interoperation among agent-based information systems through a communication acts ontology," *Information Systems*, Vol. 32(8): 1121-1144, december 2007.

[12] M.B. Blake, "Agent-Based Workflow Configuration and Management of On-line Services," v zborniku *International Conference on Electronic Commerce Research (ICECR-4)*, Dallas, TX, 2001, str. 567-588.

[13] M.B. Blake, H. Gomaa, "Agent-oriented compositional approaches to services-based cross-organizational workflow," *Decision Support Systems*, Vol. 40(1): 31-50, julij 2005.

[14] M. Bohanec, V. Rajkovič, "Multi-attribute decision modeling: Industrial appliactions of DEX," *Informatica*, Vol. 23(4): 487-491, oktober 1999.

[15] J. Bosch, S. Friedrichs, S. Jung, J. Helbig, A. Scherdin, "Service Orientation in the Enterprise," *IEEE Computer*, 40(11): 51-56, 2007.

[16] J.M. Bradshaw, S. Dutfield, P. Benoit, J.D. Woolley, "KAoS: toward an industrial-strength open agent architecture," v: J.M. Bradshaw (urd.), *Software Agents*, AAAI Press/The MIT Press, Cambridge, MA, 1997.

[17] C.K. Chang, J. Zhang, K.H. Chang, "Survey of computer-supported collaboration in support of business processes," *International Journal of Business Process Integration and Management*, 1(2): 76-100, 2006.

[18] A. Cichocki, "Workflow and Process Automation: Concepts and Technology", Kluwer Academic Publishers, 1998.

[19] R. Dieng, O. Corby, A. Giboin, M. Ribiere, “Methods and tools for corporate knowledge management, “ *International Journal of Human-Computer Studies*, 51(3): 567–598, 1999.

[20] T. Erl, *Service-Oriented Architecture: Concepts, Technology and Design*, Prentice Hall, Upper Saddle River, NJ, 2005.

[21] J. Estublier, S. Sanlaville, “Extensible Process Support Environments for Web Services Orchestration”, *International Journal of Web Services Practices*, 1(1-2): 30-39, 2005.

[22] D. Fensel, F. van Harmelen, I. Horrocks, D.L. McGuinness, P.F. Patel-Schneider, “OIL: an ontology infrastructure for the Semantic Web,” *IEEE Intelligent Systems*, Vol. 16 (2): 38–45, 2001.

[23] T. Finin, R. Fritzson, D. McKay, R. McEntire, “KQML as an Agent Communication Language, ” v zborniku *Proceedings of the Third International Conference on Information and Knowledge Management*, ACM Press, Gaithersburg, Maryland, ZDA, november 1994, str. 456-463.

[24] Foundation for Intelligent Physical Agents, "FIPA Agent Communication Language Specifications," dostopno na: http://www.fipa.org/repository/aclspecs.html.

[25] Foundation for Intelligent Physical Agents, "FIPA Contract Net Interaction Protocol Specification," december 2002, dostopno na: http://www.fipa.org/specs/fipa00029/SC00029H.html.

[26] Foundation for Intelligent Physical Agents, "FIPA English Auction Interaction Protocol Specification," avgust 2001, dostopno na: http://www.fipa.org/specs/fipa00031/XC00031F.html.

[27] Foundation for Intelligent Physical Agents, "FIPA Interaction Protocol Library Specification," 2003, dostopno na: http://www.fipa.org/repository/ips.php3.

[28] M.S. Fox, M. Barbuceanu, M. Gruninger, J. Lin, "An Organisation Ontology for Enterprise Modeling," v: M. Prietula, K. Carley, and L. Gasser (urd.), *Simulating*

Organizations: Computational Models of Institutions and Groups, AAAI/MIT Press, Menlo Park, CA, 1998.

[29] F. Gandon, A. Poggi, G. Rimassa, P. Turci, "Multi-Agents Corporate Memory Management System," *Journal of Applied Artificial Intelligence*, Vol. 16(9&10): 699-720, oktober/december 2002.

[30] D. Georgakopoulos, M. Hornick, and A. Sheth, "An Overview of Workflow Management: From Process Modeling to Workflow Automation Infrastructure," *Distributed and Parallel Databases*, Vol. 3(2): 119-153, april 1995.

[31] J.C. Giarratano, G. Riley, *Expert Systems, Principles and Programing*, Brooks/Cole, Pacific Grove, CA, 2005.

[32] P. Giorgini, B. Henderson-Sellers, "Agent-Oriented methodologies: An introduction," v: B. Henderson-Sellers, P. Giorgini (urd.), *Agent- Oriented methodologies*, Idea Group Inc., Hershley, PA, 2005.

[33] T. Gruber, "Towards principles for design of ontologies used for knowledge sharing," *International Journal of Human-Computer Studies*, Vol. 43(5-6): 907–928, november/december 1995.

[34] J. Hendler, D. McGuinness, "The DARPA Agent Markup Language, " *IEEE Intelligent Systems*, Vol. 15 (6): 67–73, 2000.

[35] I. Horrocks, P.F. Patel-Schneider, F. van Harmelen, "From SHIQ and RDF to OWL: the making of a Web Ontology Language," *Web Semantics: Science, Services and Agents on the World Wide Web*, Vol. 1(1): 7-23, 2003.

[36] I. Horrocks, C. Welty, Digital Libraries and web-based information systems," v: F. Baader, D.L McGuiness, D. Nardi, D. (Eds.), *Description Logic Handbook: Theory, Implementation and Applications*, Cambridge University Press, Cambridge, UK, 2003.

[37] M.-E. Iacob, H. Jonkers, M.M. Lankhorst, M.W.A. Steen, “Service-oriented enterprise modeling and analysis: a case study”, *International Journal of Business Process Integration and Management*, 2(1): 26-37, 2007.

[38] F. Jammes, H. Smit, "Service-Oriented Paradigms in Industrial Automation," *IEEE Transactions on Industrial Informatics,* 1(1): 62-70, 2005.

[39] N.R. Jennings, T.J. Norman, P. Faratin, "Autonomous Agents for Business Process Management," *International Journal of Applied Artificial Intelligence*, Vol. 14(2): 145-189, 2000.

[40] M.B. Jurič, B. Mathew, and P. Sarang, *Business Process Execution Language for Web Services*, 2. izdaja, Packt Publishing, Birmingham, UK, 2006.

[41] M.B. Jurič, K. Pant, *Business Process Driven SOA using BPMN and BPEL: From Business Process Modeling to Orchestration and Service Oriented Architecture*, Packt Publishing, Birmingham, UK, 2008.

[42] R. van Kaathoven, M. A. Jeusfeld, M. Staudt, U. Reimer, "Organizational Memory Supported Workflow Management," v zborniku *4. Int. Tagung Wirtschaftsinformatik*, *Electronic Business Engineering*, Physica-Verlag, Heidelberg, Nemčija, 1999, str. 543-563

[43] A.P. Kalogeras, J.V. Gialelis, C.E. Alexakos, M.J. Georgoudakis, S.A. Koubias, "Vertical Integration of Enterprise Industrial Systems Utilizing Web Services", *IEEE Transactions on Industrial Informatics,* 2(2): 120-128, 2006.

[44] B. Kavčič, *Poslovno komuniciranje*, Celje: Visoka komercialna šola, 2006.

[45] D. Krafzig, K. Banke, D. Slama, *Enterprise SOA: Service-Oriented Architecture Best Practices*, Prentice Hall PTR, 2004.

[46] L.F. Lai, "A knowledge engineering approach to knowledge management, " *Information Sciences*, Vol. 177(19): 4072-4094, oktober 2007.

[47] M. Lankhorst, *Enterprise Architecture at Work: Modelling, Communication and Analysis*, Springer, 2005.

[48] Y. Li, K-M Chao, M. Younas, Y. Huang, X. Lu, "Modeling e-marketplaces with multi-agents Web services", v zborniku *11th Int. Conf. on Parallel and Distributed Systems*, Fukuoka, Japan, 2005.

[49] A. Maedche, B. Motik, L. Stojanovic, R. Studer, R. Volz, "Ontologies for Enterprise Knowledge Management, " *IEEE Intelligent Systems*, 18(2): 26-33, 2003.

[50] J. McGovern, O. Sims, A. Jain, M. Little, *Enterprise Service Oriented Architectures: Concepts, Challenges, Recommendations*, Springer, 2006.

[51] G.W. Mineau, M. Bernard, J.F. Sowa (urd.), *Conceptual Graphs for Knowledge Representation*, Lecture Notes in AI 699, Springer-Verlag, Berlin, 1993.

[52] S.A. Moore, "**A** Foundation for Flexible Automated Electronic Communication," *Information Systems Research*, Vol. 12(1): 34-62, 2001.

[53] A. Nugroho, "Business Process vs. Functional Silos," dostopno na: http://processdriven.livejournal.com/.

[54] D. O'Leary, "Using AI in knowledge management: Knowledge bases and ontologies, " *IEEE Intelligent Systems*, Vol. 13(3): 34-39, 1998.

[55] OASIS, "OASIS UDDI Version 2 Specifications," julij 2002, dostopno na: http://www.oasis-open.org/committees/uddi-spec/doc/tcspecs.htm.

[56] OASIS, "UDDI Version 3.0.2, UDDI Spec Technical Committee Draft," oktober 2004, dostopno na: http://uddi.org/pubs/uddi_v3.htm.

[57] OASIS, "Web Services Business Process Execution Language Version 2.0, OASIS Standard, " april 2007, dostopno na: http://docs.oasis-open.org/wsbpel/2.0/wsbpel-v2.0.html.

[58] J. Odell, H. Van Dyke Parunak, B. Bauer, "Representing Agent Interaction Protocols in UML," v: P. Ciancarini, M Wooldridge (urd.), *Agent-Oriented Software Engineering*, Springer, Berlin, 2001.

[59] OMG, "Business Process Modeling Notation, " dostopno na: http://www.bpmn.org/.

[60] M.P. Papazoglou, P. Traverso, S. Dustdar, F. Leyman, "Service-Oriented Computing: State of the Art and Research Challenges," *IEEE Computer*, 40(11): 38-45, 2007.

[61] C. Plesums, "Introduction to Workflow," v L. Fischer (urd.), *Workflow Handbook 2002*, Future Strategies Inc., Lighthouse Point, FL, 2002.

[62] C. Plesums, "Workflow in the world of BPM: Are they the same?" v L. Fischer (urd), *Workflow Handbook 2005,* Future Strategies Inc., Lighthouse Point, FL, 2005.

[63] U. Reimer, A. Magelish, M. Staudt, "EULE: A Knowledge-Based System to Support Business Processes," *Knowledge-Based Systems*, Vol. 13(5): 235-331, oktober 2000.

[64] T. Rodrigues, P. Rosa, J. Cardoso, "Mapping XML to existing OWL ontologies," v zborniku *International Conference WWW/Internet 2006*, Murcia, Španija, 2006, str.72-77.

[65] J. Schiefer, H. Roth, A. Schatten, "Auditable WSBPEL: probing and monitoring of business processes with web services", *International Journal of Business Process Integration and Management*, 2(1): 60-73, 2007.

[66] G. Schreiber, H. Akkermans, A. Anjewierden, R. de Hoog, N. Shadbolt, W. Van de Velde, B. Wielinga, *Knowledge Engineering and Management - The CommonKADS Methodology*, The MIT Press, 1999.

[67] J. Searle, *Speech Acts: An Essay in the Philosophy of Language*, Cambridge University Press, 1969.

[68] H. Shah, M. El Kourdi, "Frameworks for Enterprise Architecture," *IT Professional*, 9(5): 42-46, 2007.

[69] M.P. Singh, "A social semantics for agent communication languages," v: J.G. Carbonell, J. Siekmann, F. Dignum, M. Greaves (urd.), *Issues in Agent Communication*, Springer, Berlin 2000.

[70] S. Staab, H.-P. Schnurr, R. Studer, Y. Sure, "Knowledge processes and ontologies," *IEEE Intelligent Systems*, 16(1): 26-34, 2001.

[71] B. Steffen, P. Narayan, "Full Life-Cycle Support for End-to-End Processes," *IEEE Computer*, 40(11): 64-73, 2007.

[72] E.A. Stohr, J.L. Zhao, "Workflow Automation: Overview and Research Issues", *Information Systems Frontiers*, 3(3): 281-296, 2001.

[73] L. Stojanovic, A. Maedche, B. Motik, N. Stojanovic, "User-driven Ontology Evolution Management," v zborniku *13th European Conference on Knowledge Engineering and Management (EKAW 2002)*, Springer-Verlag, 2002, str. 285-300.

[74] Z. Stojanovic, A. Dahanayake, *Service-Oriented Software System Engineering: Challenges and Practices*, Idea Group Inc., 2005.

[75] D.M. Strong , O. Volkoff, "A Roadmap for Enterprise System Implementation," *IEEE Computer*, Vol. 37 (6), junij 2004, str. 22 - 29.

[76] Y. Sure, S. Staab, R. Studer, "Methodoloy for development and employment of ontology based knowledge management applications", SIGMOD Record-Web Edition, Vol 31(4), v: R. Meersman, A. Sheth (urd.), *Special Section on Semantic Web and Data Management*, 2002. Dostopno na: http://www.acm.org/sigmod/record.

[77] Y. Sure, R. Studer, "On-To-Knowledge Methodology – Final Version", *EU-IST Project IST-1999-10132 On-To-Knowledge*, 2002, dostopno na: www.aifb.uni-karlsruhe.de/WBS/ysu/publications/.

[78] Y. Sure, *Methodoloy, Tools & Case Studies for Ontology based Knowledge Management*, doktorska disertacija, Fakultät für Wirtschaftswissenschaften, Universität Karlsruhe.

[79] A. Šaša, *Modeliranje večagentnih sistemov: diplomsko delo*, Ljubljana, 2005.

[80] A. Šaša, M. B. Jurič, M. Krisper, "Agents and People Activities in Web-services based Business processes", v zborniku *2007 IEEE International Conference on Digital Ecosystems and Technologies (IEEE-DEST 2007)*, Cairns, Australia, 2007.

[81] A. Šaša, M. B. Jurič, M. Krisper, "Service-Oriented Framework for Human Task Support and Automation," *IEEE Transactions on Industrial Informatics,* Vol. 4(4): 292-302, 2008.

[82] A. Šaša, V. Rajkovič, "Ontology-Based Decision Making and Support," v zborniku *9. mednarodne multikonference Informacijska družba 2006,* Ljubljana, oktober 2006.

[83] K. Taveter, G. Wagner, "Towards Radical Agent-Oriented Software Engineering Process Based on AOR Modelling," v: B. Henderson-Sellers, P. Giorgini (urd.), *Agent-oriented Methodologies*, Idea Group Inc., Hershey, PA, 2005.

[84] M. Uschold, M. Gruninger, "Ontologies: Principles, Methods and Applications," *The Knowledge Engineering Revolution*, 11(2): 93-136, 1996.

[85] J. Walsh, G.R. Ungson, "Organizational memory, " *Academy of Management Review*, 16(1): 57-91, 1991.

[86] W3C, "SWRL: A Semantic Web Rule Language. Combining OWL and RuleML," maj 2004, dostopno na: http://www.w3.org/Submission/SWRL/.

[87] W3C, "DAML+OIL (March 2001) Reference Description," marec 2001, dostopno na: http://www.w3.org/TR/daml+oil-reference.

[88] W3C, "Extensible Markup Language (XML)," dostopno na: http://www.w3.org/XML/.

[89] W3C, "OWL Web Ontology Language," februar 2004, dostopno na: http://www.w3.org/TR/owl-ref/.

[90] W3C, "OWL 2 Web Ontology Language: Structural Specification and Functional-Style Syntax, W3C Working Draft, " december 2008, dostopno na: http://www.w3.org/TR/owl2-syntax/.

[91] W3C, "Resource Description Framework (RDF)," 2004, dostopno na: http://www.w3.org/RDF/.

[92] W3C, "Simple Object Access Protocol, " dostopno na: http://www.w3.org/TR/soap/.

[93] W3C, "SOAP Version 1.2 specification," dostopno na: http://www.w3.org/TR/soap12/.

[94] W3C, "W3C XML Schema Definition Language (XSD) 1.1, " junij 2008, dostopno na: http://www.w3.org/TR/xmlschema11-1/ in http://www.w3.org/TR/xmlschema11-2/.

[95] W3C, "Web Services Description Language (WSDL) 1.1," marec 2001, dostopno na: http://www.w3.org/TR/wsdl.

[96] W3C, "Web Services Description Language (WSDL) Version 2.0, W3C Recommendation," junij 2007, dostopno na: http://www.w3.org/TR/wsdl20/.

[97] W3C, "Web Services Glossary, " februar 2004, dostopno na: http://www.w3.org/TR/ws-gloss/.

[98] W3C, "XML Schema," oktober 2004, dostopno na: http://www.w3.org/TR/xmlschema-1/ in http://www.w3.org/TR/xmlschema-2/.

[99] W3C, “XML Path Language (XPath) 2.0,” januar 2007, dostopno na: http://www.w3.org/TR/xpath20/.

[100] W3C, “XQuery 1.0: An XML Query Language,” januar 2007, dostopno na: http://www.w3.org/TR/xquery/.

[101] W.M.P. Van Der Aalst, M. Beisiegel, K. M. Van Hee, D. Konig, C. Stahl, “An SOA-based architecture framework”, *International Journal of Business Process Integration and Management*, 2(2): 91-101, 2007.

[102] W.M.P. Van Der Aalst, F. Leymann, W. Reisig, “The role of business processes in service oriented architectures,” *International Journal of Business Process Integration and Management*, 2(2): 75-80, 2007.

[103] WfMC Workflow Management Coalition, http://www.wfmc.org.

[104] M. Wooldridge, "Agent Based Software Engineering, *IEE Proceedings on Software Engineering*, Vol. 144(1): 26 - 37 , februar 1997.

[105] M. Wooldridge, *An Introduction to Multi Agent Systems*, Wiley, Chicester, UK, 2002.

[106] WS-I, “Basic Profile,” dostopno na: http://www.ws-i.org/deliverables/workinggroup.aspx?wg=basicprofile.

[107] WS-I, “Basic Security Profile Version 1.0,” marec 2007, dostopno na: http://www.ws-i.org/Profiles/BasicSecurityProfile-1.0.html.

[108] WS-I, “Simple SOAP Binding Profile Version 1.0, ” avgust 2004, dostopno na: http://www.ws-i.org/Profiles/SimpleSoapBindingProfile-1.0.html.

[109] Q. Xu, R. Qiu, F. Xu, "Agent-Based Workflow Coordination for Supply Chain Management," *Transactions of Nanjing University of Aeronautics & Astronautics*, 20(1): 112-117, 2003.

[110] P. Yolum, M.P. Singh, “Commitment-based enhancement of e-commerce protocols,” v zborniku *IEEE 9th International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises*, Gaithersburg, MD, 2000, str. 278–283.

[111] T.I. Zhang, H. Jiang, “A Framework of Incorporating Software Agents into SOA,” v zborniku *Artificial Intelligence and Soft Computing (ASC 2005)*, Benidorm, Spain, 2005.

[112] R. Zurawski, "Integration Technologies for Industrial Automated Systems: Challenges and Trends," R. Zurawski, *Integration Technologies for Industrial Automated Systems (Industrial Information Technology)*, CRC Press, Boca Raton, FL, 2006.

Priloga 1: XML shema razširitev specifikacij WS-Human Task

HTOO.xsd

```
<?xml version="1.0" encoding="windows-1252" ?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
            xmlns="http://www.fri.uni-lj.si/siuo/oo"
            xmlns:htd="http://www.example.org/WS-HT"
            targetNamespace="http://www.fri.uni-lj.si/siuo/oo"
            elementFormDefault="qualified"
            blockDefault="#all">

    <xsd:import namespace = "http://www.example.org/WS-HT" schemaLocation = "ws-
humantask.xsd"/>

    <xsd:element name="ontologyTranslationElements"
type="tOntologyTranslationElements"/>
    <xsd:element name="taskGroupOntologyTranslationElements"
type="tTaskGroupOntologyTranslationElements"/>
    <xsd:element name="HTOOQueryParameters" type="tHTOOQueryParameters" />
    <xsd:element name="outputQuery" type="xsd:string" />
    <xsd:element name="taskGroupOntologyTranslationName" type="xsd:string" />
    <xsd:element name="protocolName" type="xsd:string" />
    <xsd:element name="protocolRoleName" type="xsd:string" />

    <xsd:attribute name="overrideExisting" type="xsd:boolean" default="false" />
    <xsd:attribute name="confirmationRequired" type="xsd:boolean" default="false"
/>

    <xsd:complexType name="tTaskGroupOntologyTranslationElements">
        <xsd:sequence>
            <xsd:element name="taskGroupName" type="xsd:string" />
            <xsd:element name="ontologyTranslationElements"
type="tOntologyTranslationElements"/>
        </xsd:sequence>
        <xsd:attribute ref="overrideExisting"/>
        <xsd:attribute ref="confirmationRequired"/>
    </xsd:complexType>

    <xsd:complexType name="tTaskInGroup">
        <xsd:attribute name="taskName" type="xsd:NCName" use="required" />
    </xsd:complexType>

    <xsd:complexType name="tHTOOQuery" mixed="true">
        <xsd:attribute name="type" type="xsd:QName" use="required" />
    </xsd:complexType>

    <xsd:complexType name="tHTOOQueryParameter" mixed="true">
        <xsd:complexContent>
            <xsd:extension base="tHTOOQuery">
                <xsd:attribute name="name" type="xsd:NCName" use="required" />
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>

    <xsd:complexType name="tHTOOQueryParameters">
        <xsd:sequence>
```

```
            <xsd:element name="HTOOQueryParameter" type="tHTOOQueryParameter"
minOccurs="1" maxOccurs="unbounded"/>
        </xsd:sequence>
    </xsd:complexType>

    <xsd:complexType name="tOntologyTranslationElements">
        <xsd:sequence>
            <xsd:element name="translateToOWLIndividual"
type="tTranslateToOWLIndividual" minOccurs="0" maxOccurs="unbounded"/>
            <xsd:element name="translateToOWLDatatypeProperty"
type="tTranslateToOWLDatatypeProperty" minOccurs="0" maxOccurs="unbounded"/>
            <xsd:element name="translateToOWLObjectProperty"
type="tTranslateToOWLObjectProperty" minOccurs="0" maxOccurs="unbounded"/>
        </xsd:sequence>
        <xsd:attribute name="queryLanguage" type="xsd:anyURI" />
    </xsd:complexType>

    <xsd:complexType name="tTranslateToOWLIndividual" >
        <xsd:sequence>
            <xsd:element name="elementForOWLIndividual" type="tHTOOQuery" />
            <xsd:element name="IDForOWLIndividual" type="tHTOOQuery" />
            <xsd:element name="OWLClass" type="xsd:anyURI" />
        </xsd:sequence>
        <xsd:attribute name="name" type="xsd:NCName" use="optional" />
    </xsd:complexType>

    <xsd:complexType name="tToODPWithNewIndividual">
        <xsd:complexContent>
            <xsd:extension base="tTranslateToOWLIndividual">
                <xsd:sequence>
                    <xsd:element name="datatypeValue" type="tHTOOQuery" />
                </xsd:sequence>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>

    <xsd:complexType name="tReferenceToTranslation" >
        <xsd:attribute name="name" type="xsd:QName" use="required" />
    </xsd:complexType>

    <xsd:complexType name="tTranslateToOWLDatatypeProperty" >
        <xsd:sequence>
            <xsd:choice>
                 <xsd:element name="toODPWithNewIndividual"
type="tTranslateToOWLIndividual" />
                 <xsd:element name="referenceToTranslationToIndividual"
type="tReferenceToTranslation" />
                 <xsd:element name="IDOfExistingOWLIndividualFromInput"
type="tHTOOQuery" />
                 <xsd:element name="OWLIndividual" type="xsd:anyURI" />
            </xsd:choice>

            <xsd:element name="OWLDatatypeProperty" type="xsd:anyURI" />
            <xsd:element name="datatypeValue" type="tHTOOQuery" />
        </xsd:sequence>
   </xsd:complexType>

   <xsd:complexType name="tTranslateToOWLObjectProperty" >
        <xsd:sequence>
            <xsd:choice>
                 <xsd:element name="fromNewOWLIndividual"
type="tTranslateToOWLIndividual" />
                 <xsd:element name="referenceToTranslationToNewIndividual"
type="tReferenceToTranslation" />
                 <xsd:element name="fromIDOfExistingOWLIndividualFromInput"
type="tHTOOQuery" />
                 <xsd:element name="fromOWLIndividual" type="xsd:anyURI" />
```

```
            </xsd:choice>

            <xsd:choice>
               <xsd:element name="toNewOWLIndividual"
type="tTranslateToOWLIndividual"/>
               <xsd:element name="referenceToTranslationFromIndividual"
type="tReferenceToTranslation" />
               <xsd:element name="toIDOfExistingOWLIndividualFromInput"
type="tHTOOQuery" />
               <xsd:element name="toOWLIndividual" type="xsd:anyURI" />
            </xsd:choice>
            <xsd:element name="OWLObjectProperty" type="xsd:anyURI" />
         </xsd:sequence>
   </xsd:complexType>
</xsd:schema>
```

Priloga 2: WSDL datoteka in OWL ontologija primera

Opredelitev WSDL datoteke SIUO (Zaposlovanje.wsdl)

```
<?xml version="1.0" encoding="UTF-8"?>
<wsdl:definitions name="Zaposlovanje"
    targetNamespace="http://www.primer.si/zaposlovanje"
    xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
    xmlns:tns="http://www.primer.si/zaposlovanje"
    xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
        xmlns:plnk="http://schemas.xmlsoap.org/ws/2003/05/partner-link/">

       <wsdl:types>
            <xsd:schema xmlns="http://www.w3.org/2001/XMLSchema">
            <xsd:import namespace="http://www.primer.si/zaposlovanje"
schemaLocation="kandidati.xsd" />
        </xsd:schema
          </wsdl:types>

       <wsdl:message name="SeznamKandidatovZaDelovnoMesto">
           <wsdl:part name="SeznamKandidatovZaDelovnoMesto"
       element="tns:seznamKandidatovZaDelovnoMesto" />
       </wsdl:message>

       <wsdl:message name="NajprimernejsiKandidat">
           <wsdl:part name="NajprimernejsiKandidat" type="xsd:string" />
       </wsdl:message>

       <wsdl:portType name="IzborNajprimernejsegaKandidataPT">
           <wsdl:operation name="izberiNajprimernejsegaKandidata">
               <wsdl:input message="tns:SeznamKandidatovZaDelovnoMesto" />
           </wsdl:operation>
           <wsdl:operation name="escalate">
               <wsdl:input message="tns:Kandidati" />
           </wsdl:operation>
       </wsdl:portType>

       <wsdl:portType name="IzborNajprimernejsegaKandidataCallbackPT">
           <wsdl:operation name="najprimernejsiKandidat">
               <wsdl:input message="tns:NajprimernejsiKandidat" />
           </wsdl:operation>
       </wsdl:portType>
</wsdl:definitions>
```

Opredelitev XML sheme uporabljene v WSDL datoteki (Kandidati.xsd)

```
<?xml version="1.0" encoding="windows-1252" ?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
            xmlns="http://www.primer.si/zaposlovanje"
            targetNamespace="http://www.primer.si/zaposlovanje"
            elementFormDefault="qualified" >
```

```
  <xsd:element name="seznamKandidatovZaDelovnoMesto">
            <xsd:complexType>
                <xsd:sequence>
                    <xsd:element name="delovnoMesto" type="xsd:string"/>
                    <xsd:element name="kandidati" type="kandidatiType"/>
                </xsd:sequence>
            </xsd:complexType>
        </xsd:element>

  <xsd:complexType name="kandidatiType">
    <xsd:sequence>
        <xsd:element name="kandidat" type="kandidatType"/>
    </xsd:sequence>
  </xsd:complexType>
  <xsd:complexType name="kandidatType">
      <xsd:sequence>
        <xsd:element name="identifikacija" type="identifikacijaType" minOccurs="1"
maxOccurs="1"/>
        <xsd:element name="izobrazevanje" minOccurs="0">
            <xsd:complexType>
                <xsd:sequence>
                    <xsd:element name="izobrazba" type="izobrazbaType"
maxOccurs="unbounded"
                                 minOccurs="1"/>
                </xsd:sequence>
            </xsd:complexType>
        </xsd:element>
        <xsd:element name="delovneIzkusnje" type="delovneIzkusnjeType"
minOccurs="0"/>
        <xsd:element name="tujiJeziki" minOccurs="0">
            <xsd:complexType>
                    <xsd:sequence>
                        <xsd:element name="tujJezik" type="tujJezikType"
maxOccurs="unbounded"
                                     minOccurs="1"/>
                    </xsd:sequence>
            </xsd:complexType>
        </xsd:element>
        <xsd:element name="osebneLastnosti" type="osebneLastnostiType"
minOccurs="0"/>
      </xsd:sequence>
  </xsd:complexType>

  <xsd:complexType name="identifikacijaType">
        <xsd:sequence>
            <xsd:element name="ime" type="xsd:string"/>
            <xsd:element name="primek" type="xsd:string"/>
            <xsd:element name="EMSO" type="xsd:string"/>
            <xsd:element name="kontaktniPodatki" type="kontaktniPodatkiType"
minOccurs="0"/>
            <xsd:element name="datumRojstva" type="xsd:date" minOccurs="0"/>
            <xsd:element name="spol" minOccurs="0">
                <xsd:simpleType>
                    <xsd:restriction base="xsd:string">
                        <xsd:enumeration value="Z"/>
                        <xsd:enumeration value="M"/>
                        <xsd:enumeration value="NA"/>
                    </xsd:restriction>
                </xsd:simpleType>
            </xsd:element>
            <xsd:element name="nacionalnost" minOccurs="0" maxOccurs="unbounded"
type="xsd:string" />
        </xsd:sequence>
  </xsd:complexType>

  <xsd:complexType name="kontaktniPodatkiType">
        <xsd:sequence>
```

```
            <xsd:element name="naslov" type="naslovType" />
            <xsd:element name="telefon" type="xsd:string"/>
            <xsd:element name="fax" type="xsd:string"/>
            <xsd:element name="mobilnaSt" type="xsd:string"/>
            <xsd:element name="email" type="xsd:string"/>
        </xsd:sequence>
</xsd:complexType>

  <xsd:complexType name="naslovType">
    <xsd:sequence>
      <xsd:element name="ulica" minOccurs="0" maxOccurs="1" type="xsd:string" />
      <xsd:element name="kraj" minOccurs="0" maxOccurs="1" type="xsd:string" />
      <xsd:element name="postnaSt" minOccurs="0" maxOccurs="1" type="xsd:string" />
      <xsd:element name="drzava" minOccurs="0" maxOccurs="1" type="xsd:string" />
    </xsd:sequence>
  </xsd:complexType>

   <xsd:complexType name="izobrazbaType">
    <xsd:sequence>
        <xsd:element name="datumPridobitve" type="xsd:date"/>
        <xsd:element name="naziv" type="xsd:string"/>
        <xsd:element name="stopnja" type="xsd:int"/>
        <xsd:element name="ustanova" type="xsd:string"/>
        <xsd:element name="podrocje" type="xsd:string"/>
    </xsd:sequence>
  </xsd:complexType>

  <xsd:complexType name="delovneIzkusnjeType">
        <xsd:sequence>
            <xsd:element name="obdobjeVMesecih" type="xsd:int"/>
            <xsd:element name="vrsteAktivnosti">
                <xsd:complexType>
                    <xsd:sequence>
                        <xsd:element name="vrstaAktivnosti">
                            <xsd:simpleType>
                                <xsd:restriction base="xsd:string">
                                    <xsd:enumeration value="vodenje"/>
                                    <xsd:enumeration value="projektnoVodenje"/>
                                    <xsd:enumeration value="planiranje"/>
                                    <xsd:enumeration value="porocanje"/>
                                    <xsd:enumeration value="..."/>
                                    <xsd:enumeration value="programiranje"/>
                    </xsd:restriction>
                </xsd:simpleType>
                        </xsd:element>
                    </xsd:sequence>
                </xsd:complexType>

                        </xsd:element>
            <xsd:element name="delovnoMesto" type="xsd:string"/>
            <xsd:element name="delodajalec" type="xsd:string"/>
            <xsd:element name="sektor" type="xsd:string"/>
        </xsd:sequence>
  </xsd:complexType>

  <xsd:complexType name="tujJezikType">
        <xsd:sequence>
            <xsd:element name="sifra" type="xsd:language" minOccurs="0"/>
            <xsd:element name="jezik" type="xsd:string" minOccurs="0"/>
            <xsd:element name="stopnja" type="xsd:string"
minOccurs="0"/>
       </xsd:sequence>
   </xsd:complexType>

   <xsd:complexType name="osebneLastnostiType">
           <xsd:sequence>
               <xsd:element name="ocenaNaTestu" minOccurs="1">
```

```
                        <xsd:simpleType>
                    <xsd:restriction base="xsd:int">
                        <xsd:enumeration value="1"/>
                        <xsd:enumeration value="2"/>
                        <xsd:enumeration value="3"/>
                        <xsd:enumeration value="4"/>
                        <xsd:enumeration value="5"/>
                    </xsd:restriction>
                </xsd:simpleType>
                        </xsd:element>
                        <xsd:element name="nastop" minOccurs="0">
                            <xsd:simpleType>
                                <xsd:restriction base="xsd:string">
                                    <xsd:enumeration value="podpovprecno"/>
                                    <xsd:enumeration value="povprecno"/>
                                    <xsd:enumeration value="dobro"/>
                                    <xsd:enumeration value="zeloDobro"/>
                                </xsd:restriction>
                            </xsd:simpleType>
                         </xsd:element>
                </xsd:sequence>
        </xsd:complexType>
    </xsd:schema>
```

Opredelitev OWL ontologije (Kandidati.owl)

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


<!DOCTYPE rdf:RDF [
    <!ENTITY owl "http://www.w3.org/2002/07/owl#" >
    <!ENTITY xsd "http://www.w3.org/2001/XMLSchema#" >
    <!ENTITY owl2xml "http://www.w3.org/2006/12/owl2-xml#" >
    <!ENTITY Kandidati "http://www.htoo.org/dm/Kandidati.owl#" >
    <!ENTITY rdfs "http://www.w3.org/2000/01/rdf-schema#" >
    <!ENTITY rdf "http://www.w3.org/1999/02/22-rdf-syntax-ns#" >
]>

<rdf:RDF xmlns="http://www.htoo.org/dm/Kandidati.owl#"
     xml:base="http://www.htoo.org/dm/Kandidati.owl"
     xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
     xmlns:owl2xml="http://www.w3.org/2006/12/owl2-xml#"
     xmlns:Kandidati="http://www.htoo.org/dm/Kandidati.owl#"
     xmlns:owl="http://www.w3.org/2002/07/owl#"
     xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
     xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#">
    <owl:Ontology rdf:about=""/>

    <!--

///////////////////////////////////////////////////////////////////////////////////
////
    //
    // Objektne lastnosti
    //

///////////////////////////////////////////////////////////////////////////////////
////
     -->

    <!-- http://www.htoo.org/dm/Kandidati.owl#govoriAngleskiJezik -->

    <owl:ObjectProperty rdf:about="#govoriAngleskiJezik">
        <rdf:type rdf:resource="&owl;FunctionalProperty"/>
        <rdfs:domain rdf:resource="#Kandidat"/>
```

```
    <rdfs:range rdf:resource="#StopnjaZnanjaTujegaJezika"/>
</owl:ObjectProperty>

<!-- http://www.htoo.org/dm/Kandidati.owl#govoriItalijanskiJezik -->

<owl:ObjectProperty rdf:about="#govoriItalijanskiJezik">
    <rdf:type rdf:resource="&owl;FunctionalProperty"/>
    <rdfs:domain rdf:resource="#Kandidat"/>
    <rdfs:range rdf:resource="#StopnjaZnanjaTujegaJezika"/>
</owl:ObjectProperty>

<!-- http://www.htoo.org/dm/Kandidati.owl#govoriNemskiJezik -->

<owl:ObjectProperty rdf:about="#govoriNemskiJezik">
    <rdf:type rdf:resource="&owl;FunctionalProperty"/>
    <rdfs:domain rdf:resource="#Kandidat"/>
    <rdfs:range rdf:resource="#StopnjaZnanjaTujegaJezika"/>
</owl:ObjectProperty>

<!-- http://www.htoo.org/dm/Kandidati.owl#imaOcenoNastopa -->

<owl:ObjectProperty rdf:about="#imaOcenoNastopa">
    <rdf:type rdf:resource="&owl;FunctionalProperty"/>
    <rdfs:domain rdf:resource="#Kandidat"/>
    <rdfs:range rdf:resource="#OcenaNastopa"/>
</owl:ObjectProperty>

<!-- http://www.htoo.org/dm/Kandidati.owl#imaRezultatTesta -->

<owl:ObjectProperty rdf:about="#imaRezultatTesta">
    <rdf:type rdf:resource="&owl;FunctionalProperty"/>
    <rdfs:domain rdf:resource="#Kandidat"/>
    <rdfs:range rdf:resource="#RezultatTesta"/>
</owl:ObjectProperty>

<!-- http://www.htoo.org/dm/Kandidati.owl#jeImelZaposlitev -->

<owl:ObjectProperty rdf:about="#jeImelZaposlitev">
    <rdf:type rdf:resource="&owl;InverseFunctionalProperty"/>
    <rdfs:domain rdf:resource="#Kandidat"/>
    <rdfs:range rdf:resource="#Zaposlitev"/>
</owl:ObjectProperty>

<!-- http://www.htoo.org/dm/Kandidati.owl#jeStopnje -->

<owl:ObjectProperty rdf:about="#jeStopnje">
    <rdf:type rdf:resource="&owl;FunctionalProperty"/>
    <rdfs:domain rdf:resource="#Izobrazba"/>
    <rdfs:range rdf:resource="#StopnjaIzobrazbe"/>
</owl:ObjectProperty>

<!-- http://www.htoo.org/dm/Kandidati.owl#obsegaVrstoAktivnosti -->

<owl:ObjectProperty rdf:about="#obsegaVrstoAktivnosti">
    <rdfs:range rdf:resource="#VrstaAktivnosti"/>
    <rdfs:domain rdf:resource="#Zaposlitev"/>
</owl:ObjectProperty>

<!-- http://www.htoo.org/dm/Kandidati.owl#pripadaKandidatu -->

<owl:ObjectProperty rdf:about="#pripadaKandidatu">
    <rdfs:domain rdf:resource="#Izobrazba"/>
    <rdfs:range rdf:resource="#Kandidat"/>
</owl:ObjectProperty>

<!-- http://www.htoo.org/dm/Kandidati.owl#sPodrocja -->
```

```
    <owl:ObjectProperty rdf:about="#sPodrocja">
        <rdfs:range rdf:resource="#Podrocje"/>
        <rdfs:domain>
            <owl:Class>
                <owl:unionOf rdf:parseType="Collection">
                    <rdf:Description rdf:about="#Izobrazba"/>
                    <rdf:Description rdf:about="#Zaposlitev"/>
                </owl:unionOf>
            </owl:Class>
        </rdfs:domain>
    </owl:ObjectProperty>

    <!-- http://www.htoo.org/dm/Kandidati.owl#vSektorju -->

    <owl:ObjectProperty rdf:about="#vSektorju">
        <rdf:type rdf:resource="&owl;FunctionalProperty"/>
        <rdfs:range rdf:resource="#Sektor"/>
        <rdfs:domain rdf:resource="#Zaposlitev"/>
    </owl:ObjectProperty>

    <!-- http://www.htoo.org/dm/Kandidati.owl#vZdruzbi -->

    <owl:ObjectProperty rdf:about="#vZdruzbi">
        <rdf:type rdf:resource="&owl;FunctionalProperty"/>
        <rdfs:domain rdf:resource="#Zaposlitev"/>
        <rdfs:range rdf:resource="#Zdruzba"/>
    </owl:ObjectProperty>

    <!--

///////////////////////////////////////////////////////////////////////////////////
////
    //
    // Podatkovne lastnosti
    //

///////////////////////////////////////////////////////////////////////////////////
////
     -->

    <!-- http://www.htoo.org/dm/Kandidati.owl#imaDatumRojstva -->

    <owl:DatatypeProperty rdf:about="#imaDatumRojstva">
        <rdf:type rdf:resource="&owl;FunctionalProperty"/>
        <rdfs:domain rdf:resource="#Kandidat"/>
        <rdfs:range rdf:resource="&xsd;date"/>
    </owl:DatatypeProperty>

    <!-- http://www.htoo.org/dm/Kandidati.owl#jeZaposlenec -->

    <owl:DatatypeProperty rdf:about="#jeZaposlenec">
        <rdf:type rdf:resource="&owl;FunctionalProperty"/>
        <rdfs:domain rdf:resource="#Kandidat"/>
        <rdfs:range rdf:resource="&xsd;boolean"/>
    </owl:DatatypeProperty>

    <!-- http://www.htoo.org/dm/Kandidati.owl#steviloLetDelovnihIzkusenj -->

    <owl:DatatypeProperty rdf:about="#steviloLetDelovnihIzkusenj">
        <rdf:type rdf:resource="&owl;FunctionalProperty"/>
        <rdfs:domain rdf:resource="#Kandidat"/>
        <rdfs:range rdf:resource="&xsd;float"/>
    </owl:DatatypeProperty>



    <!-- http://www.htoo.org/dm/Kandidati.owl#steviloLetZaposlitve -->
```

```
    <owl:DatatypeProperty rdf:about="#steviloLetZaposlitve">
        <rdf:type rdf:resource="&owl;FunctionalProperty"/>
        <rdfs:domain rdf:resource="#Zaposlitev"/>
        <rdfs:range rdf:resource="&xsd;float"/>
    </owl:DatatypeProperty>

    <!--

///////////////////////////////////////////////////////////////////////////////////
////
    //
    // Razredi
    //

///////////////////////////////////////////////////////////////////////////////////
////
     -->

    <!-- http://www.htoo.org/dm/Kandidati.owl#Izobrazba -->

    <owl:Class rdf:about="#Izobrazba">
        <owl:disjointWith rdf:resource="#Kandidat"/>
        <owl:disjointWith rdf:resource="#OcenaNastopa"/>
        <owl:disjointWith rdf:resource="#Podrocje"/>
        <owl:disjointWith rdf:resource="#RezultatTesta"/>
        <owl:disjointWith rdf:resource="#Sektor"/>
        <owl:disjointWith rdf:resource="#StopnjaIzobrazbe"/>
        <owl:disjointWith rdf:resource="#StopnjaZnanjaTujegaJezika"/>
        <owl:disjointWith rdf:resource="#VrstaAktivnosti"/>
        <owl:disjointWith rdf:resource="#Zaposlitev"/>
        <owl:disjointWith rdf:resource="#Zdruzba"/>
    </owl:Class>

    <!-- http://www.htoo.org/dm/Kandidati.owl#Kandidat -->

    <owl:Class rdf:about="#Kandidat">
        <owl:disjointWith rdf:resource="#OcenaNastopa"/>
        <owl:disjointWith rdf:resource="#Podrocje"/>
        <owl:disjointWith rdf:resource="#RezultatTesta"/>
        <owl:disjointWith rdf:resource="#Sektor"/>
        <owl:disjointWith rdf:resource="#StopnjaIzobrazbe"/>
        <owl:disjointWith rdf:resource="#StopnjaZnanjaTujegaJezika"/>
        <owl:disjointWith rdf:resource="#VrstaAktivnosti"/>
        <owl:disjointWith rdf:resource="#Zaposlitev"/>
        <owl:disjointWith rdf:resource="#Zdruzba"/>
    </owl:Class>

    <!-- http://www.htoo.org/dm/Kandidati.owl#OcenaNastopa -->

    <owl:Class rdf:about="#OcenaNastopa">
        <owl:equivalentClass>
            <owl:Class>
                <owl:oneOf rdf:parseType="Collection">
                    <rdf:Description rdf:about="#primerno"/>
                    <rdf:Description rdf:about="#nePrimerno"/>
                    <rdf:Description rdf:about="#manjPrimerno"/>
                    <rdf:Description rdf:about="#odlicno"/>
                </owl:oneOf>
            </owl:Class>
        </owl:equivalentClass>
        <owl:disjointWith rdf:resource="#Podrocje"/>
        <owl:disjointWith rdf:resource="#RezultatTesta"/>
        <owl:disjointWith rdf:resource="#Sektor"/>
        <owl:disjointWith rdf:resource="#StopnjaIzobrazbe"/>
        <owl:disjointWith rdf:resource="#StopnjaZnanjaTujegaJezika"/>
        <owl:disjointWith rdf:resource="#VrstaAktivnosti"/>
```

```
    <owl:disjointWith rdf:resource="#Zaposlitev"/>
    <owl:disjointWith rdf:resource="#Zdruzba"/>
</owl:Class>

<!-- http://www.htoo.org/dm/Kandidati.owl#Podrocje -->

<owl:Class rdf:about="#Podrocje">
    <owl:equivalentClass>
        <owl:Class>
            <owl:oneOf rdf:parseType="Collection">
                <rdf:Description rdf:about="#fizika"/>
                <rdf:Description rdf:about="#ekonomija"/>
                <rdf:Description rdf:about="#trzenje"/>
                <rdf:Description rdf:about="#pravo"/>
                <rdf:Description rdf:about="#racunalnistvo"/>
                <rdf:Description rdf:about="#farmacija"/>
                <rdf:Description rdf:about="#matematika"/>
                <rdf:Description rdf:about="#biologija"/>
                <rdf:Description rdf:about="#informatika"/>
                <rdf:Description rdf:about="#elektrotehnika"/>
            </owl:oneOf>
        </owl:Class>
    </owl:equivalentClass>
    <owl:disjointWith rdf:resource="#RezultatTesta"/>
    <owl:disjointWith rdf:resource="#Sektor"/>
    <owl:disjointWith rdf:resource="#StopnjaIzobrazbe"/>
    <owl:disjointWith rdf:resource="#StopnjaZnanjaTujegaJezika"/>
    <owl:disjointWith rdf:resource="#VrstaAktivnosti"/>
    <owl:disjointWith rdf:resource="#Zaposlitev"/>
    <owl:disjointWith rdf:resource="#Zdruzba"/>
</owl:Class>

<!-- http://www.htoo.org/dm/Kandidati.owl#RezultatTesta -->

<owl:Class rdf:about="#RezultatTesta">
    <owl:equivalentClass>
        <owl:Class>
            <owl:oneOf rdf:parseType="Collection">
                <rdf:Description rdf:about="#ocena_1"/>
                <rdf:Description rdf:about="#ocena_2"/>
                <rdf:Description rdf:about="#ocena_5"/>
                <rdf:Description rdf:about="#ocena_3"/>
                <rdf:Description rdf:about="#ocena_4"/>
            </owl:oneOf>
        </owl:Class>
    </owl:equivalentClass>
    <owl:disjointWith rdf:resource="#Sektor"/>
    <owl:disjointWith rdf:resource="#StopnjaIzobrazbe"/>
    <owl:disjointWith rdf:resource="#StopnjaZnanjaTujegaJezika"/>
    <owl:disjointWith rdf:resource="#VrstaAktivnosti"/>
    <owl:disjointWith rdf:resource="#Zaposlitev"/>
    <owl:disjointWith rdf:resource="#Zdruzba"/>
</owl:Class>

<!-- http://www.htoo.org/dm/Kandidati.owl#Sektor -->

<owl:Class rdf:about="#Sektor">
    <owl:disjointWith rdf:resource="#StopnjaIzobrazbe"/>
    <owl:disjointWith rdf:resource="#StopnjaZnanjaTujegaJezika"/>
    <owl:disjointWith rdf:resource="#VrstaAktivnosti"/>
    <owl:disjointWith rdf:resource="#Zaposlitev"/>
    <owl:disjointWith rdf:resource="#Zdruzba"/>
</owl:Class>

<!-- http://www.htoo.org/dm/Kandidati.owl#StopnjaIzobrazbe -->

<owl:Class rdf:about="#StopnjaIzobrazbe">
```

```
    <owl:equivalentClass>
        <owl:Class>
            <owl:oneOf rdf:parseType="Collection">
                <rdf:Description rdf:about="#ISCED_1"/>
                <rdf:Description rdf:about="#ISCED_3"/>
                <rdf:Description rdf:about="#ISCED_4"/>
                <rdf:Description rdf:about="#ISCED_2"/>
                <rdf:Description rdf:about="#ISCED_6"/>
                <rdf:Description rdf:about="#ISCED_5"/>
                <rdf:Description rdf:about="#ISCED_0"/>
            </owl:oneOf>
        </owl:Class>
    </owl:equivalentClass>
    <owl:disjointWith rdf:resource="#StopnjaZnanjaTujegaJezika"/>
    <owl:disjointWith rdf:resource="#VrstaAktivnosti"/>
    <owl:disjointWith rdf:resource="#Zaposlitev"/>
    <owl:disjointWith rdf:resource="#Zdruzba"/>
</owl:Class>

<!-- http://www.htoo.org/dm/Kandidati.owl#StopnjaZnanjaTujegaJezika -->

<owl:Class rdf:about="#StopnjaZnanjaTujegaJezika">
    <owl:equivalentClass>
        <owl:Class>
            <owl:oneOf rdf:parseType="Collection">
                <rdf:Description rdf:about="#tekoce"/>
                <rdf:Description rdf:about="#dobro"/>
                <rdf:Description rdf:about="#osnovno"/>
                <rdf:Description rdf:about="#zacetno"/>
                <rdf:Description rdf:about="#brezPredznanja"/>
            </owl:oneOf>
        </owl:Class>
    </owl:equivalentClass>
    <owl:disjointWith rdf:resource="#VrstaAktivnosti"/>
    <owl:disjointWith rdf:resource="#Zaposlitev"/>
    <owl:disjointWith rdf:resource="#Zdruzba"/>
</owl:Class>

<!-- http://www.htoo.org/dm/Kandidati.owl#VrstaAktivnosti -->

<owl:Class rdf:about="#VrstaAktivnosti">
    <owl:equivalentClass>
        <owl:Class>
            <owl:oneOf rdf:parseType="Collection">
                <rdf:Description rdf:about="#vodenjeITprojektov"/>
                <rdf:Description rdf:about="#programiranje"/>
                <rdf:Description rdf:about="#knjigovodenje"/>
                <rdf:Description rdf:about="#PR"/>
                <rdf:Description rdf:about="#skupinskoDelo"/>
                <rdf:Description rdf:about="#razvojUporabniskihVmesnikov"/>
                <rdf:Description rdf:about="#administriranjePodatkovneBaze"/>
                <rdf:Description rdf:about="#nacrtovanjeArhitekturIS"/>
                <rdf:Description rdf:about="#razvojSpletnihStoritev"/>
                <rdf:Description rdf:about="#poucevanje"/>
                <rdf:Description rdf:about="#poslovnoKomuniciranje"/>
            </owl:oneOf>
        </owl:Class>
    </owl:equivalentClass>
    <owl:disjointWith rdf:resource="#Zaposlitev"/>
    <owl:disjointWith rdf:resource="#Zdruzba"/>
</owl:Class>

<!-- http://www.htoo.org/dm/Kandidati.owl#Zaposlitev -->

<owl:Class rdf:about="#Zaposlitev">
    <owl:disjointWith rdf:resource="#Zdruzba"/>
</owl:Class>
```

```
    <!-- http://www.htoo.org/dm/Kandidati.owl#Zdruzba -->

    <owl:Class rdf:about="#Zdruzba"/>


    <!--

///////////////////////////////////////////////////////////////////////////////////
////
    //
    // Primerki
    //

///////////////////////////////////////////////////////////////////////////////////
////
     -->

    <!-- http://www.htoo.org/dm/Kandidati.owl#ISCED_0 -->
    <StopnjaIzobrazbe rdf:about="#ISCED_0"/>
    <!-- http://www.htoo.org/dm/Kandidati.owl#ISCED_1 -->
    <StopnjaIzobrazbe rdf:about="#ISCED_1"/>
    <!-- http://www.htoo.org/dm/Kandidati.owl#ISCED_2 -->
    <StopnjaIzobrazbe rdf:about="#ISCED_2"/>
    <!-- http://www.htoo.org/dm/Kandidati.owl#ISCED_3 -->
    <StopnjaIzobrazbe rdf:about="#ISCED_3"/>
    <!-- http://www.htoo.org/dm/Kandidati.owl#ISCED_4 -->
    <StopnjaIzobrazbe rdf:about="#ISCED_4"/>
    <!-- http://www.htoo.org/dm/Kandidati.owl#ISCED_5 -->
    <StopnjaIzobrazbe rdf:about="#ISCED_5"/>
    <!-- http://www.htoo.org/dm/Kandidati.owl#ISCED_6 -->
    <StopnjaIzobrazbe rdf:about="#ISCED_6"/>
    <!-- http://www.htoo.org/dm/Kandidati.owl#PR -->
    <VrstaAktivnosti rdf:about="#PR"/>
    <!-- http://www.htoo.org/dm/Kandidati.owl#administriranjePodatkovneBaze -->
    <VrstaAktivnosti rdf:about="#administriranjePodatkovneBaze"/>
    <!-- http://www.htoo.org/dm/Kandidati.owl#biologija -->
    <Podrocje rdf:about="#biologija"/>
    <!-- http://www.htoo.org/dm/Kandidati.owl#brezPredznanja -->
    <StopnjaZnanjaTujegaJezika rdf:about="#brezPredznanja"/>
    <!-- http://www.htoo.org/dm/Kandidati.owl#dobro -->
    <StopnjaZnanjaTujegaJezika rdf:about="#dobro"/>
    <!-- http://www.htoo.org/dm/Kandidati.owl#ekonomija -->
    <Podrocje rdf:about="#ekonomija"/>
    <!-- http://www.htoo.org/dm/Kandidati.owl#elektrotehnika -->
    <Podrocje rdf:about="#elektrotehnika"/>
    <!-- http://www.htoo.org/dm/Kandidati.owl#farmacija -->
    <Podrocje rdf:about="#farmacija"/>
    <!-- http://www.htoo.org/dm/Kandidati.owl#fizika -->
    <Podrocje rdf:about="#fizika"/>
    <!-- http://www.htoo.org/dm/Kandidati.owl#informatika -->
    <Podrocje rdf:about="#informatika"/>
    <!-- http://www.htoo.org/dm/Kandidati.owl#knjigovodenje -->
    <VrstaAktivnosti rdf:about="#knjigovodenje"/>
    <!-- http://www.htoo.org/dm/Kandidati.owl#manjPrimerno -->
    <OcenaNastopa rdf:about="#manjPrimerno"/>
    <!-- http://www.htoo.org/dm/Kandidati.owl#matematika -->
    <Podrocje rdf:about="#matematika"/>
    <!-- http://www.htoo.org/dm/Kandidati.owl#nacrtovanjeArhitekturIS -->
    <VrstaAktivnosti rdf:about="#nacrtovanjeArhitekturIS"/>
    <!-- http://www.htoo.org/dm/Kandidati.owl#nePrimerno -->
    <OcenaNastopa rdf:about="#nePrimerno"/>
    <!-- http://www.htoo.org/dm/Kandidati.owl#ocena_1 -->
    <RezultatTesta rdf:about="#ocena_1"/>
    <!-- http://www.htoo.org/dm/Kandidati.owl#ocena_2 -->
    <RezultatTesta rdf:about="#ocena_2"/>
    <!-- http://www.htoo.org/dm/Kandidati.owl#ocena_3 -->
```

```
    <RezultatTesta rdf:about="#ocena_3"/>
    <!-- http://www.htoo.org/dm/Kandidati.owl#ocena_4 -->
    <RezultatTesta rdf:about="#ocena_4"/>
    <!-- http://www.htoo.org/dm/Kandidati.owl#ocena_5 -->
    <RezultatTesta rdf:about="#ocena_5"/>
    <!-- http://www.htoo.org/dm/Kandidati.owl#odlicno -->
    <OcenaNastopa rdf:about="#odlicno"/>
    <!-- http://www.htoo.org/dm/Kandidati.owl#osnovno -->
    <StopnjaZnanjaTujegaJezika rdf:about="#osnovno"/>
    <!-- http://www.htoo.org/dm/Kandidati.owl#poslovnoKomuniciranje -->
    <VrstaAktivnosti rdf:about="#poslovnoKomuniciranje"/>
    <!-- http://www.htoo.org/dm/Kandidati.owl#poucevanje -->
    <VrstaAktivnosti rdf:about="#poucevanje"/>
    <!-- http://www.htoo.org/dm/Kandidati.owl#pravo -->
    <Podrocje rdf:about="#pravo"/>
    <!-- http://www.htoo.org/dm/Kandidati.owl#primerno -->
    <OcenaNastopa rdf:about="#primerno"/>

    <!-- http://www.htoo.org/dm/Kandidati.owl#programiranje -->
    <VrstaAktivnosti rdf:about="#programiranje"/>
    <!-- http://www.htoo.org/dm/Kandidati.owl#racunalnistvo -->
    <Podrocje rdf:about="#racunalnistvo"/>
    <!-- http://www.htoo.org/dm/Kandidati.owl#razvojSpletnihStoritev -->
    <VrstaAktivnosti rdf:about="#razvojSpletnihStoritev"/>
    <!-- http://www.htoo.org/dm/Kandidati.owl#razvojUporabniskihVmesnikov -->
    <VrstaAktivnosti rdf:about="#razvojUporabniskihVmesnikov"/>
    <!-- http://www.htoo.org/dm/Kandidati.owl#skupinskoDelo -->
    <VrstaAktivnosti rdf:about="#skupinskoDelo"/>
    <!-- http://www.htoo.org/dm/Kandidati.owl#tekoce -->
    <StopnjaZnanjaTujegaJezika rdf:about="#tekoce"/>
    <!-- http://www.htoo.org/dm/Kandidati.owl#trzenje -->
    <Podrocje rdf:about="#trzenje"/>
    <!-- http://www.htoo.org/dm/Kandidati.owl#vodenjeITprojektov -->
    <VrstaAktivnosti rdf:about="#vodenjeITprojektov"/>
    <!-- http://www.htoo.org/dm/Kandidati.owl#zacetno -->
    <StopnjaZnanjaTujegaJezika rdf:about="#zacetno"/>
    <!--

///////////////////////////////////////////////////////////////////////////////////
////
    //
    // Aksiomi
    //

///////////////////////////////////////////////////////////////////////////////////
////
     -->

    <rdf:Description>
        <rdf:type rdf:resource="&owl;AllDifferent"/>
        <owl:distinctMembers rdf:parseType="Collection">
            <rdf:Description rdf:about="#vodenjeITprojektov"/>
            <rdf:Description rdf:about="#programiranje"/>
            <rdf:Description rdf:about="#knjigovodenje"/>
            <rdf:Description rdf:about="#PR"/>
            <rdf:Description rdf:about="#skupinskoDelo"/>
            <rdf:Description rdf:about="#razvojUporabniskihVmesnikov"/>
            <rdf:Description rdf:about="#administriranjePodatkovneBaze"/>
            <rdf:Description rdf:about="#nacrtovanjeArhitekturIS"/>
            <rdf:Description rdf:about="#razvojSpletnihStoritev"/>
            <rdf:Description rdf:about="#poucevanje"/>
            <rdf:Description rdf:about="#poslovnoKomuniciranje"/>
        </owl:distinctMembers>
    </rdf:Description>
    <rdf:Description>
        <rdf:type rdf:resource="&owl;AllDifferent"/>
        <owl:distinctMembers rdf:parseType="Collection">
```

```
            <rdf:Description rdf:about="#primerno"/>
            <rdf:Description rdf:about="#nePrimerno"/>
            <rdf:Description rdf:about="#manjPrimerno"/>
            <rdf:Description rdf:about="#odlicno"/>
        </owl:distinctMembers>
    </rdf:Description>
    <rdf:Description>
        <rdf:type rdf:resource="&owl;AllDifferent"/>
        <owl:distinctMembers rdf:parseType="Collection">
            <rdf:Description rdf:about="#fizika"/>
            <rdf:Description rdf:about="#ekonomija"/>
            <rdf:Description rdf:about="#trzenje"/>
            <rdf:Description rdf:about="#pravo"/>
            <rdf:Description rdf:about="#racunalnistvo"/>
            <rdf:Description rdf:about="#farmacija"/>
            <rdf:Description rdf:about="#matematika"/>
            <rdf:Description rdf:about="#biologija"/>
            <rdf:Description rdf:about="#informatika"/>
            <rdf:Description rdf:about="#elektrotehnika"/>
        </owl:distinctMembers>
    </rdf:Description>
    <rdf:Description>
        <rdf:type rdf:resource="&owl;AllDifferent"/>
        <owl:distinctMembers rdf:parseType="Collection">
            <rdf:Description rdf:about="#ISCED_1"/>
            <rdf:Description rdf:about="#ISCED_3"/>
            <rdf:Description rdf:about="#ISCED_4"/>
            <rdf:Description rdf:about="#ISCED_2"/>
            <rdf:Description rdf:about="#ISCED_6"/>
            <rdf:Description rdf:about="#ISCED_5"/>
            <rdf:Description rdf:about="#ISCED_0"/>
        </owl:distinctMembers>
    </rdf:Description>
    <rdf:Description>
        <rdf:type rdf:resource="&owl;AllDifferent"/>
        <owl:distinctMembers rdf:parseType="Collection">
            <rdf:Description rdf:about="#tekoce"/>
            <rdf:Description rdf:about="#dobro"/>
            <rdf:Description rdf:about="#osnovno"/>
            <rdf:Description rdf:about="#zacetno"/>
            <rdf:Description rdf:about="#brezPredznanja"/>
        </owl:distinctMembers>
    </rdf:Description>
    <rdf:Description>
        <rdf:type rdf:resource="&owl;AllDifferent"/>
        <owl:distinctMembers rdf:parseType="Collection">
            <rdf:Description rdf:about="#ocena_1"/>
            <rdf:Description rdf:about="#ocena_2"/>
            <rdf:Description rdf:about="#ocena_5"/>
            <rdf:Description rdf:about="#ocena_3"/>
            <rdf:Description rdf:about="#ocena_4"/>
        </owl:distinctMembers>
    </rdf:Description>
</rdf:RDF>
```

Priloga 3: Primer odločitvenega modela v OWL ontologiji

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


<!DOCTYPE rdf:RDF [
    <!ENTITY owl "http://www.w3.org/2002/07/owl#" >
    <!ENTITY swrl "http://www.w3.org/2003/11/swrl#" >
    <!ENTITY swrlb "http://www.w3.org/2003/11/swrlb#" >
    <!ENTITY xsd "http://www.w3.org/2001/XMLSchema#" >
    <!ENTITY owl2xml "http://www.w3.org/2006/12/owl2-xml#" >
    <!ENTITY Kandidati "http://www.htoo.org/dm/Kandidati.owl#" >
    <!ENTITY rdfs "http://www.w3.org/2000/01/rdf-schema#" >
    <!ENTITY rdf "http://www.w3.org/1999/02/22-rdf-syntax-ns#" >
]>


<rdf:RDF xmlns="http://www.htoo.org/dm/Kandidati.owl#"
     xml:base="http://www.htoo.org/dm/Kandidati.owl"
     xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
     xmlns:swrl="http://www.w3.org/2003/11/swrl#"
     xmlns:owl2xml="http://www.w3.org/2006/12/owl2-xml#"
     xmlns:Kandidati="http://www.htoo.org/dm/Kandidati.owl#"
     xmlns:owl="http://www.w3.org/2002/07/owl#"
     xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
     xmlns:swrlb="http://www.w3.org/2003/11/swrlb#"
     xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#">
    <owl:Ontology rdf:about=""/>

    <!--

///////////////////////////////////////////////////////////////////////////////////////
////
    //
    // Objektne lastnosti
    //

///////////////////////////////////////////////////////////////////////////////////////
////
     -->


    <!-- http://www.htoo.org/dm/Kandidati.owl#govoriAngleskiJezik -->

    <owl:ObjectProperty rdf:about="#govoriAngleskiJezik">
        <rdf:type rdf:resource="&owl;FunctionalProperty"/>
        <rdfs:domain rdf:resource="#Kandidat"/>
        <rdfs:range rdf:resource="#StopnjaZnanjaTujegaJezika"/>
    </owl:ObjectProperty>

    <!-- http://www.htoo.org/dm/Kandidati.owl#govoriItalijanskiJezik -->

    <owl:ObjectProperty rdf:about="#govoriItalijanskiJezik">
        <rdf:type rdf:resource="&owl;FunctionalProperty"/>
        <rdfs:domain rdf:resource="#Kandidat"/>
        <rdfs:range rdf:resource="#StopnjaZnanjaTujegaJezika"/>
```

```
</owl:ObjectProperty>

<!-- http://www.htoo.org/dm/Kandidati.owl#govoriNemskiJezik -->

<owl:ObjectProperty rdf:about="#govoriNemskiJezik">
    <rdf:type rdf:resource="&owl;FunctionalProperty"/>
    <rdfs:domain rdf:resource="#Kandidat"/>
    <rdfs:range rdf:resource="#StopnjaZnanjaTujegaJezika"/>
</owl:ObjectProperty>

<!-- http://www.htoo.org/dm/Kandidati.owl#imaIzobrazbo -->

<owl:ObjectProperty rdf:about="#imaIzobrazbo">
    <rdf:type rdf:resource="&owl;InverseFunctionalProperty"/>
    <owl:inverseOf rdf:resource="#pripadaKandidatu"/>
</owl:ObjectProperty>

<!-- http://www.htoo.org/dm/Kandidati.owl#imaOcenoNastopa -->

<owl:ObjectProperty rdf:about="#imaOcenoNastopa">
    <rdf:type rdf:resource="&owl;FunctionalProperty"/>
    <rdfs:domain rdf:resource="#Kandidat"/>
    <rdfs:range rdf:resource="#OcenaNastopa"/>
</owl:ObjectProperty>

<!-- http://www.htoo.org/dm/Kandidati.owl#imaRezultatTesta -->

<owl:ObjectProperty rdf:about="#imaRezultatTesta">
    <rdf:type rdf:resource="&owl;FunctionalProperty"/>
    <rdfs:domain rdf:resource="#Kandidat"/>
    <rdfs:range rdf:resource="#RezultatTesta"/>
</owl:ObjectProperty>

<!-- http://www.htoo.org/dm/Kandidati.owl#jeImelZaposlitev -->

<owl:ObjectProperty rdf:about="#jeImelZaposlitev">
    <rdf:type rdf:resource="&owl;InverseFunctionalProperty"/>
    <rdfs:domain rdf:resource="#Kandidat"/>
    <rdfs:range rdf:resource="#Zaposlitev"/>
</owl:ObjectProperty>

<!-- http://www.htoo.org/dm/Kandidati.owl#jeStopnje -->

<owl:ObjectProperty rdf:about="#jeStopnje">
    <rdf:type rdf:resource="&owl;FunctionalProperty"/>
    <rdfs:domain rdf:resource="#Izobrazba"/>
    <rdfs:range rdf:resource="#StopnjaIzobrazbe"/>
</owl:ObjectProperty>

<!-- http://www.htoo.org/dm/Kandidati.owl#obsegaVrstoAktivnosti -->

<owl:ObjectProperty rdf:about="#obsegaVrstoAktivnosti">
    <rdfs:range rdf:resource="#VrstaAktivnosti"/>
    <rdfs:domain rdf:resource="#Zaposlitev"/>
</owl:ObjectProperty>

<!-- http://www.htoo.org/dm/Kandidati.owl#pripadaKandidatu -->

<owl:ObjectProperty rdf:about="#pripadaKandidatu">
    <rdfs:domain rdf:resource="#Izobrazba"/>
    <rdfs:range rdf:resource="#Kandidat"/>
</owl:ObjectProperty>

<!-- http://www.htoo.org/dm/Kandidati.owl#sPodrocja -->

<owl:ObjectProperty rdf:about="#sPodrocja">
    <rdfs:range rdf:resource="#Podrocje"/>
```

```
        <rdfs:domain>
            <owl:Class>
                <owl:unionOf rdf:parseType="Collection">
                    <rdf:Description rdf:about="#Izobrazba"/>
                    <rdf:Description rdf:about="#Zaposlitev"/>
                </owl:unionOf>
            </owl:Class>
        </rdfs:domain>
    </owl:ObjectProperty>

    <!-- http://www.htoo.org/dm/Kandidati.owl#vSektorju -->

    <owl:ObjectProperty rdf:about="#vSektorju">
        <rdf:type rdf:resource="&owl;FunctionalProperty"/>
        <rdfs:range rdf:resource="#Sektor"/>
        <rdfs:domain rdf:resource="#Zaposlitev"/>
    </owl:ObjectProperty>

    <!-- http://www.htoo.org/dm/Kandidati.owl#vZdruzbi -->

    <owl:ObjectProperty rdf:about="#vZdruzbi">
        <rdf:type rdf:resource="&owl;FunctionalProperty"/>
        <rdfs:domain rdf:resource="#Zaposlitev"/>
        <rdfs:range rdf:resource="#Zdruzba"/>
    </owl:ObjectProperty>

    <!--

////////////////////////////////////////////////////////////////////////////////
////
    //
    // Podatkovne lastnosti
    //

////////////////////////////////////////////////////////////////////////////////
////
     -->

    <!-- http://www.htoo.org/dm/Kandidati.owl#imaDatumRojstva -->

    <owl:DatatypeProperty rdf:about="#imaDatumRojstva">
        <rdf:type rdf:resource="&owl;FunctionalProperty"/>
        <rdfs:domain rdf:resource="#Kandidat"/>
        <rdfs:range rdf:resource="&xsd;date"/>
    </owl:DatatypeProperty>

    <!-- http://www.htoo.org/dm/Kandidati.owl#jeZaposlenec -->

    <owl:DatatypeProperty rdf:about="#jeZaposlenec">
        <rdf:type rdf:resource="&owl;FunctionalProperty"/>
        <rdfs:domain rdf:resource="#Kandidat"/>
        <rdfs:range rdf:resource="&xsd;boolean"/>
    </owl:DatatypeProperty>

    <!-- http://www.htoo.org/dm/Kandidati.owl#steviloLetDelovnihIzkusenj -->

    <owl:DatatypeProperty rdf:about="#steviloLetDelovnihIzkusenj">
        <rdf:type rdf:resource="&owl;FunctionalProperty"/>
        <rdfs:domain rdf:resource="#Kandidat"/>
        <rdfs:range rdf:resource="&xsd;float"/>
    </owl:DatatypeProperty>

    <!-- http://www.htoo.org/dm/Kandidati.owl#steviloLetZaposlitve -->

    <owl:DatatypeProperty rdf:about="#steviloLetZaposlitve">
        <rdf:type rdf:resource="&owl;FunctionalProperty"/>
        <rdfs:domain rdf:resource="#Zaposlitev"/>
```

```
        <rdfs:range rdf:resource="&xsd;float"/>
    </owl:DatatypeProperty>

    <!--

///////////////////////////////////////////////////////////////////////////////////
////
    //
    // Razredi
    //

///////////////////////////////////////////////////////////////////////////////////
////
     -->

    <!-- http://www.htoo.org/dm/Kandidati.owl#Izobrazba -->

    <owl:Class rdf:about="#Izobrazba">
        <owl:equivalentClass>
            <owl:Class>
                <owl:unionOf rdf:parseType="Collection">
                    <rdf:Description rdf:about="#NeprimernaIzobrazba"/>
                    <rdf:Description rdf:about="#PrimernaIzobrazba"/>
                    <rdf:Description rdf:about="#ZeloPrimernaIzobrazba"/>
                </owl:unionOf>
            </owl:Class>
        </owl:equivalentClass>
        <owl:disjointWith rdf:resource="#Kandidat"/>
        <owl:disjointWith rdf:resource="#OcenaNastopa"/>
        <owl:disjointWith rdf:resource="#Podrocje"/>
        <owl:disjointWith rdf:resource="#RezultatTesta"/>
        <owl:disjointWith rdf:resource="#Sektor"/>
        <owl:disjointWith rdf:resource="#StopnjaIzobrazbe"/>
        <owl:disjointWith rdf:resource="#StopnjaZnanjaTujegaJezika"/>
        <owl:disjointWith rdf:resource="#VrstaAktivnosti"/>
        <owl:disjointWith rdf:resource="#Zaposlitev"/>
        <owl:disjointWith rdf:resource="#Zdruzba"/>
    </owl:Class>

    <!-- http://www.htoo.org/dm/Kandidati.owl#Kandidat -->

    <owl:Class rdf:about="#Kandidat">
        <owl:equivalentClass>
            <owl:Class>
                <owl:unionOf rdf:parseType="Collection">
                    <rdf:Description
rdf:about="#KandidatSPrimernimRezultatomTesta"/>
                    <rdf:Description
rdf:about="#KandidatZManjPrimernimRezultatomTesta"/>
                    <rdf:Description
rdf:about="#KandidatZNajprimernejsimRezultatomTesta"/>
                </owl:unionOf>
            </owl:Class>
        </owl:equivalentClass>
        <owl:equivalentClass>
            <owl:Class>
                <owl:unionOf rdf:parseType="Collection">
                    <rdf:Description
rdf:about="#KandidatSPrimernimiOsebnimiLastnostmi"/>
                    <rdf:Description
rdf:about="#KandidatZManjPrimernimiOsebnimiLastnostmi"/>
                    <rdf:Description
rdf:about="#KandidatZNajprimernejsimiOsebnimiLastnostmi"/>
                </owl:unionOf>
            </owl:Class>
        </owl:equivalentClass>
        <owl:equivalentClass>
```

```
            <owl:Class>
                <owl:unionOf rdf:parseType="Collection">
                    <rdf:Description rdf:about="#KandidatSPrimernimNastopom"/>
                    <rdf:Description rdf:about="#KandidatZManjPrimernimNastopom"/>
                    <rdf:Description rdf:about="#KandidatZNePrimernimNastopom"/>
                    <rdf:Description rdf:about="#KandidatZOdlicnimNastopom"/>
                </owl:unionOf>
            </owl:Class>
        </owl:equivalentClass>
        <owl:equivalentClass>
            <owl:Class>
                <owl:unionOf rdf:parseType="Collection">
                    <rdf:Description
rdf:about="#KandidatSPrimernimZnanjemTujegaJezika"/>
                    <rdf:Description
rdf:about="#KandidatZNeprimernimZnanjemTujegaJezika"/>
                </owl:unionOf>
            </owl:Class>
        </owl:equivalentClass>
        <owl:equivalentClass>
            <owl:Class>
                <owl:unionOf rdf:parseType="Collection">
                    <rdf:Description rdf:about="#KandidatSPrimernoIzobrazbo"/>
                    <rdf:Description rdf:about="#KandidatZNeprimernoIzobrazbo"/>
                    <rdf:Description rdf:about="#KandidatZZeloPrimernoIzobrazbo"/>
                </owl:unionOf>
            </owl:Class>
        </owl:equivalentClass>
        <owl:equivalentClass>
            <owl:Class>
                <owl:unionOf rdf:parseType="Collection">
                    <rdf:Description rdf:about="#KandidatSPrimernimPredznanjem"/>
                    <rdf:Description
rdf:about="#KandidatZManjPrimernimPredznanjem"/>
                    <rdf:Description
rdf:about="#KandidatZNajprimernejsimPredznanjem"/>
                    <rdf:Description
rdf:about="#KandidatZZeloPrimernimPredznanjem"/>
                </owl:unionOf>
            </owl:Class>
        </owl:equivalentClass>
        <owl:equivalentClass>
            <owl:Class>
                <owl:unionOf rdf:parseType="Collection">
                    <rdf:Description
rdf:about="#KandidatSPrimernimiDelovnimiIzkusnjami"/>
                    <rdf:Description
rdf:about="#KandidatZManjPrimernimiDelovnimiIzkusnjami"/>
                    <rdf:Description
rdf:about="#KandidatZZeloPrimernimiDelovnimiIzkusnjami"/>
                </owl:unionOf>
            </owl:Class>
        </owl:equivalentClass>
        <owl:equivalentClass>
            <owl:Class>
                <owl:unionOf rdf:parseType="Collection">
                    <rdf:Description rdf:about="#KandidatManjPrimerneStarosti"/>
                    <rdf:Description rdf:about="#KandidatNajprimernejseStarosti"/>
                    <rdf:Description rdf:about="#KandidatPrimerneStarosti"/>
                </owl:unionOf>
            </owl:Class>
        </owl:equivalentClass>
        <owl:equivalentClass>
            <owl:Class>
                <owl:unionOf rdf:parseType="Collection">
                    <rdf:Description rdf:about="#NajboljsiKandidat"/>
                    <rdf:Description rdf:about="#NeprimerenKandidat"/>
```

```
                <rdf:Description rdf:about="#PrimerenKandidat"/>
                <rdf:Description rdf:about="#ZeloDoberKandidat"/>
            </owl:unionOf>
        </owl:Class>
    </owl:equivalentClass>
    <owl:disjointWith rdf:resource="#OcenaNastopa"/>
    <owl:disjointWith rdf:resource="#Podrocje"/>
    <owl:disjointWith rdf:resource="#RezultatTesta"/>
    <owl:disjointWith rdf:resource="#Sektor"/>
    <owl:disjointWith rdf:resource="#StopnjaIzobrazbe"/>
    <owl:disjointWith rdf:resource="#StopnjaZnanjaTujegaJezika"/>
    <owl:disjointWith rdf:resource="#VrstaAktivnosti"/>
    <owl:disjointWith rdf:resource="#Zaposlitev"/>
    <owl:disjointWith rdf:resource="#Zdruzba"/>
</owl:Class>

<!-- http://www.htoo.org/dm/Kandidati.owl#KandidatManjPrimerneStarosti -->

<owl:Class rdf:about="#KandidatManjPrimerneStarosti">
    <rdfs:subClassOf rdf:resource="#Kandidat"/>
    <owl:disjointWith rdf:resource="#KandidatNajprimernejseStarosti"/>
    <owl:disjointWith rdf:resource="#KandidatPrimerneStarosti"/>
</owl:Class>

<!-- http://www.htoo.org/dm/Kandidati.owl#KandidatNajprimernejseStarosti -->

<owl:Class rdf:about="#KandidatNajprimernejseStarosti">
    <rdfs:subClassOf rdf:resource="#Kandidat"/>
    <owl:disjointWith rdf:resource="#KandidatPrimerneStarosti"/>
</owl:Class>

<!-- http://www.htoo.org/dm/Kandidati.owl#KandidatPrimerneStarosti -->

<owl:Class rdf:about="#KandidatPrimerneStarosti">
    <rdfs:subClassOf rdf:resource="#Kandidat"/>
</owl:Class>

<!-- http://www.htoo.org/dm/Kandidati.owl#KandidatSPrimernimNastopom -->

<owl:Class rdf:about="#KandidatSPrimernimNastopom">
    <owl:equivalentClass>
        <owl:Class>
            <owl:intersectionOf rdf:parseType="Collection">
                <rdf:Description rdf:about="#Kandidat"/>
                <owl:Restriction>
                    <owl:onProperty rdf:resource="#imaOcenoNastopa"/>
                    <owl:allValuesFrom>
                        <owl:Class>
                            <owl:oneOf rdf:parseType="Collection">
                                <rdf:Description rdf:about="#primerno"/>
                            </owl:oneOf>
                        </owl:Class>
                    </owl:allValuesFrom>
                </owl:Restriction>
            </owl:intersectionOf>
        </owl:Class>
    </owl:equivalentClass>
    <rdfs:subClassOf rdf:resource="#Kandidat"/>
    <owl:disjointWith rdf:resource="#KandidatZManjPrimernimNastopom"/>
    <owl:disjointWith rdf:resource="#KandidatZNePrimernimNastopom"/>
    <owl:disjointWith rdf:resource="#KandidatZOdlicnimNastopom"/>
</owl:Class>



<!-- http://www.htoo.org/dm/Kandidati.owl#KandidatSPrimernimPredznanjem -->
```

```
    <owl:Class rdf:about="#KandidatSPrimernimPredznanjem">
        <owl:equivalentClass>
            <owl:Class>
                <owl:intersectionOf rdf:parseType="Collection">
                    <rdf:Description
rdf:about="#KandidatSPrimernimiDelovnimiIzkusnjami"/>
                    <rdf:Description rdf:about="#KandidatSPrimernoIzobrazbo"/>
                    <owl:Class>
                        <owl:complementOf
rdf:resource="#KandidatZNajprimernejsimPredznanjem"/>
                    </owl:Class>
                    <owl:Class>
                        <owl:complementOf
rdf:resource="#KandidatZZeloPrimernimPredznanjem"/>
                    </owl:Class>
                </owl:intersectionOf>
            </owl:Class>
        </owl:equivalentClass>
        <rdfs:subClassOf rdf:resource="#Kandidat"/>
        <owl:disjointWith rdf:resource="#KandidatZManjPrimernimPredznanjem"/>
        <owl:disjointWith rdf:resource="#KandidatZNajprimernejsimPredznanjem"/>
        <owl:disjointWith rdf:resource="#KandidatZZeloPrimernimPredznanjem"/>
    </owl:Class>

    <!-- http://www.htoo.org/dm/Kandidati.owl#KandidatSPrimernimRezultatomTesta -->

    <owl:Class rdf:about="#KandidatSPrimernimRezultatomTesta">
        <owl:equivalentClass>
            <owl:Class>
                <owl:intersectionOf rdf:parseType="Collection">
                    <rdf:Description rdf:about="#Kandidat"/>
                    <owl:Restriction>
                        <owl:onProperty rdf:resource="#imaRezultatTesta"/>
                        <owl:allValuesFrom>
                            <owl:Class>
                                <owl:oneOf rdf:parseType="Collection">
                                    <rdf:Description rdf:about="#ocena_3"/>
                                </owl:oneOf>
                            </owl:Class>
                        </owl:allValuesFrom>
                    </owl:Restriction>
                </owl:intersectionOf>
            </owl:Class>
        </owl:equivalentClass>
        <rdfs:subClassOf rdf:resource="#Kandidat"/>
        <owl:disjointWith rdf:resource="#KandidatZManjPrimernimRezultatomTesta"/>
        <owl:disjointWith rdf:resource="#KandidatZNajprimernejsimRezultatomTesta"/>
    </owl:Class>

    <!-- http://www.htoo.org/dm/Kandidati.owl#KandidatSPrimernimZnanjemTujegaJezika
-->

    <owl:Class rdf:about="#KandidatSPrimernimZnanjemTujegaJezika">
        <owl:equivalentClass>
            <owl:Class>
                <owl:intersectionOf rdf:parseType="Collection">
                    <rdf:Description rdf:about="#Kandidat"/>
                    <owl:Restriction>
                        <owl:onProperty rdf:resource="#govoriAngleskiJezik"/>
                        <owl:allValuesFrom>
                            <owl:Class>
                                <owl:oneOf rdf:parseType="Collection">
                                    <rdf:Description rdf:about="#tekoce"/>
                                    <rdf:Description rdf:about="#dobro"/>
                                </owl:oneOf>
                            </owl:Class>
                        </owl:allValuesFrom>
```

```
                    </owl:Restriction>
                </owl:intersectionOf>
            </owl:Class>
        </owl:equivalentClass>
        <rdfs:subClassOf rdf:resource="#Kandidat"/>
        <owl:disjointWith rdf:resource="#KandidatZNeprimernimZnanjemTujegaJezika"/>
    </owl:Class>

    <!--
http://www.htoo.org/dm/Kandidati.owl#KandidatSPrimernimiDelovnimiIzkusnjami -->

    <owl:Class rdf:about="#KandidatSPrimernimiDelovnimiIzkusnjami">
        <rdfs:subClassOf rdf:resource="#Kandidat"/>
        <owl:disjointWith
rdf:resource="#KandidatZManjPrimernimiDelovnimiIzkusnjami"/>
        <owl:disjointWith
rdf:resource="#KandidatZZeloPrimernimiDelovnimiIzkusnjami"/>
    </owl:Class>

    <!-- http://www.htoo.org/dm/Kandidati.owl#KandidatSPrimernimiOsebnimiLastnostmi
-->

    <owl:Class rdf:about="#KandidatSPrimernimiOsebnimiLastnostmi">
        <owl:equivalentClass>
            <owl:Class>
                <owl:intersectionOf rdf:parseType="Collection">
                    <rdf:Description rdf:about="#KandidatSPrimernimNastopom"/>
                    <owl:Class>
                        <owl:unionOf rdf:parseType="Collection">
                            <rdf:Description
rdf:about="#KandidatNajprimernejseStarosti"/>
                            <rdf:Description
rdf:about="#KandidatPrimerneStarosti"/>
                        </owl:unionOf>
                    </owl:Class>
                </owl:intersectionOf>
            </owl:Class>
        </owl:equivalentClass>
        <rdfs:subClassOf rdf:resource="#Kandidat"/>
        <owl:disjointWith
rdf:resource="#KandidatZManjPrimernimiOsebnimiLastnostmi"/>
        <owl:disjointWith
rdf:resource="#KandidatZNajprimernejsimiOsebnimiLastnostmi"/>
    </owl:Class>


    <!-- http://www.htoo.org/dm/Kandidati.owl#KandidatSPrimernoIzobrazbo -->

    <owl:Class rdf:about="#KandidatSPrimernoIzobrazbo">
        <owl:equivalentClass>
            <owl:Class>
                <owl:intersectionOf rdf:parseType="Collection">
                    <rdf:Description rdf:about="#Kandidat"/>
                    <owl:Class>
                        <owl:complementOf
rdf:resource="#KandidatZZeloPrimernoIzobrazbo"/>
                    </owl:Class>
                    <owl:Restriction>
                        <owl:onProperty rdf:resource="#imaIzobrazbo"/>
                        <owl:someValuesFrom rdf:resource="#PrimernaIzobrazba"/>
                    </owl:Restriction>
                </owl:intersectionOf>
            </owl:Class>
        </owl:equivalentClass>
        <rdfs:subClassOf rdf:resource="#Kandidat"/>
        <owl:disjointWith rdf:resource="#KandidatZNeprimernoIzobrazbo"/>
        <owl:disjointWith rdf:resource="#KandidatZZeloPrimernoIzobrazbo"/>
```

```
    </owl:Class>

    <!-- http://www.htoo.org/dm/Kandidati.owl#KandidatZManjPrimernimNastopom -->

    <owl:Class rdf:about="#KandidatZManjPrimernimNastopom">
        <owl:equivalentClass>
            <owl:Class>
                <owl:intersectionOf rdf:parseType="Collection">
                    <rdf:Description rdf:about="#Kandidat"/>
                    <owl:Restriction>
                        <owl:onProperty rdf:resource="#imaOcenoNastopa"/>
                        <owl:allValuesFrom>
                            <owl:Class>
                                <owl:oneOf rdf:parseType="Collection">
                                    <rdf:Description rdf:about="#manjPrimerno"/>
                                </owl:oneOf>
                            </owl:Class>
                        </owl:allValuesFrom>
                    </owl:Restriction>
                </owl:intersectionOf>
            </owl:Class>
        </owl:equivalentClass>
        <rdfs:subClassOf rdf:resource="#Kandidat"/>
        <owl:disjointWith rdf:resource="#KandidatZNePrimernimNastopom"/>
        <owl:disjointWith rdf:resource="#KandidatZOdlicnimNastopom"/>
    </owl:Class>

    <!-- http://www.htoo.org/dm/Kandidati.owl#KandidatZManjPrimernimPredznanjem -->

    <owl:Class rdf:about="#KandidatZManjPrimernimPredznanjem">
        <owl:equivalentClass>
            <owl:Class>
                <owl:intersectionOf rdf:parseType="Collection">
                    <rdf:Description rdf:about="#Kandidat"/>
                    <owl:Class>
                        <owl:complementOf
rdf:resource="#KandidatSPrimernimPredznanjem"/>
                    </owl:Class>
                    <owl:Class>
                        <owl:complementOf
rdf:resource="#KandidatZNajprimernejsimPredznanjem"/>
                    </owl:Class>
                    <owl:Class>
                        <owl:complementOf
rdf:resource="#KandidatZZeloPrimernimPredznanjem"/>
                    </owl:Class>
                </owl:intersectionOf>
            </owl:Class>
        </owl:equivalentClass>
        <rdfs:subClassOf rdf:resource="#Kandidat"/>
        <owl:disjointWith rdf:resource="#KandidatZNajprimernejsimPredznanjem"/>
        <owl:disjointWith rdf:resource="#KandidatZZeloPrimernimPredznanjem"/>
    </owl:Class>

    <!-- http://www.htoo.org/dm/Kandidati.owl#KandidatZManjPrimernimRezultatomTesta
-->

    <owl:Class rdf:about="#KandidatZManjPrimernimRezultatomTesta">
        <owl:equivalentClass>
            <owl:Class>
                <owl:intersectionOf rdf:parseType="Collection">
                    <rdf:Description rdf:about="#Kandidat"/>
                    <owl:Restriction>
                        <owl:onProperty rdf:resource="#imaRezultatTesta"/>
                        <owl:allValuesFrom>
                            <owl:Class>
                                <owl:oneOf rdf:parseType="Collection">
```

```
                                    <rdf:Description rdf:about="#ocena_1"/>
                                    <rdf:Description rdf:about="#ocena_2"/>
                                </owl:oneOf>
                            </owl:Class>
                        </owl:allValuesFrom>
                    </owl:Restriction>
                </owl:intersectionOf>
            </owl:Class>
        </owl:equivalentClass>
        <rdfs:subClassOf rdf:resource="#Kandidat"/>
        <owl:disjointWith rdf:resource="#KandidatZNajprimernejsimRezultatomTesta"/>
    </owl:Class>



    <!--
http://www.htoo.org/dm/Kandidati.owl#KandidatZManjPrimernimiDelovnimiIzkusnjami -->

    <owl:Class rdf:about="#KandidatZManjPrimernimiDelovnimiIzkusnjami">
        <owl:equivalentClass>
            <owl:Class>
                <owl:unionOf rdf:parseType="Collection">
                    <owl:Class>
                        <owl:intersectionOf rdf:parseType="Collection">
                            <rdf:Description rdf:about="#Kandidat"/>
                            <owl:Restriction>
                                <owl:onProperty rdf:resource="#jeImelZaposlitev"/>
                                <owl:allValuesFrom>
                                    <owl:Class>
                                        <owl:intersectionOf
rdf:parseType="Collection">
                                            <rdf:Description
rdf:about="#Zaposlitev"/>
                                            <owl:Class>
                                                <owl:complementOf>
                                                    <owl:Restriction>
                                                        <owl:onProperty
rdf:resource="#obsegaVrstoAktivnosti"/>
                                                        <owl:someValuesFrom>
                                                            <owl:Class>
                                                                <owl:oneOf
rdf:parseType="Collection">

<rdf:Description rdf:about="#nacrtovanjeArhitekturIS"/>
                                                                </owl:oneOf>
                                                            </owl:Class>
                                                        </owl:someValuesFrom>
                                                    </owl:Restriction>
                                                </owl:complementOf>
                                            </owl:Class>
                                        </owl:intersectionOf>
                                    </owl:Class>
                                </owl:allValuesFrom>
                            </owl:Restriction>
                        </owl:intersectionOf>
                    </owl:Class>
                    <owl:Class>
                        <owl:intersectionOf rdf:parseType="Collection">
                            <rdf:Description rdf:about="#Kandidat"/>
                            <owl:Restriction>
                                <owl:onProperty rdf:resource="#jeImelZaposlitev"/>
                                <owl:allValuesFrom>
                                    <owl:Class>
                                        <owl:intersectionOf
rdf:parseType="Collection">
                                            <rdf:Description
rdf:about="#Zaposlitev"/>
```

```
                                        <owl:Class>
                                            <owl:complementOf>
                                                <owl:Restriction>
                                                    <owl:onProperty
rdf:resource="#obsegaVrstoAktivnosti"/>
                                                    <owl:someValuesFrom>
                                                        <owl:Class>
                                                            <owl:oneOf
rdf:parseType="Collection">

<rdf:Description rdf:about="#razvojSpletnihStoritev"/>
                                                            </owl:oneOf>
                                                        </owl:Class>
                                                    </owl:someValuesFrom>
                                                </owl:Restriction>
                                            </owl:complementOf>
                                        </owl:Class>
                                    </owl:intersectionOf>
                                </owl:Class>
                            </owl:allValuesFrom>
                        </owl:Restriction>
                    </owl:intersectionOf>
                </owl:Class>
                <owl:Class>
                    <owl:intersectionOf rdf:parseType="Collection">
                        <rdf:Description rdf:about="#Kandidat"/>
                        <owl:Restriction>
                            <owl:onProperty rdf:resource="#jeImelZaposlitev"/>
                            <owl:allValuesFrom>
                                <owl:Class>
                                    <owl:complementOf>
                                        <owl:Restriction>
                                            <owl:onProperty
rdf:resource="#obsegaVrstoAktivnosti"/>
                                            <owl:someValuesFrom>
                                                <owl:Class>
                                                    <owl:oneOf
rdf:parseType="Collection">
                                                        <rdf:Description
rdf:about="#poslovnoKomuniciranje"/>
                                                    </owl:oneOf>
                                                </owl:Class>
                                            </owl:someValuesFrom>
                                        </owl:Restriction>
                                    </owl:complementOf>
                                </owl:Class>
                            </owl:allValuesFrom>
                        </owl:Restriction>
                    </owl:intersectionOf>
                </owl:Class>
                <owl:Class>
                    <owl:intersectionOf rdf:parseType="Collection">
                        <rdf:Description rdf:about="#Kandidat"/>
                        <owl:Restriction>
                            <owl:onProperty rdf:resource="#jeImelZaposlitev"/>
                            <owl:allValuesFrom>
                                <owl:Class>
                                    <owl:complementOf>
                                        <owl:Restriction>
                                            <owl:onProperty
rdf:resource="#obsegaVrstoAktivnosti"/>
                                            <owl:someValuesFrom>
                                                <owl:Class>
                                                    <owl:oneOf
rdf:parseType="Collection">
                                                        <rdf:Description
rdf:about="#skupinskoDelo"/>
```

```
                                                    </owl:oneOf>
                                                </owl:Class>
                                            </owl:someValuesFrom>
                                        </owl:Restriction>
                                    </owl:complementOf>
                                </owl:Class>
                            </owl:allValuesFrom>
                        </owl:Restriction>
                    </owl:intersectionOf>
                </owl:Class>
                <owl:Class>
                    <owl:intersectionOf rdf:parseType="Collection">
                        <rdf:Description rdf:about="#Kandidat"/>
                        <owl:Restriction>
                            <owl:onProperty rdf:resource="#jeImelZaposlitev"/>
                            <owl:allValuesFrom>
                                <owl:Class>
                                    <owl:complementOf>
                                        <owl:Restriction>
                                            <owl:onProperty
rdf:resource="#obsegaVrstoAktivnosti"/>
                                            <owl:someValuesFrom>
                                                <owl:Class>
                                                    <owl:oneOf
rdf:parseType="Collection">
                                                        <rdf:Description
rdf:about="#vodenjeITprojektov"/>
                                                    </owl:oneOf>
                                                </owl:Class>
                                            </owl:someValuesFrom>
                                        </owl:Restriction>
                                    </owl:complementOf>
                                </owl:Class>
                            </owl:allValuesFrom>
                        </owl:Restriction>
                    </owl:intersectionOf>
                </owl:Class>
                <owl:Class>
                    <owl:intersectionOf rdf:parseType="Collection">
                        <rdf:Description rdf:about="#Kandidat"/>
                        <owl:Restriction>
                            <owl:onProperty rdf:resource="#jeImelZaposlitev"/>
                            <owl:allValuesFrom>
                                <owl:Class>
                                    <owl:complementOf>
                                        <owl:Restriction>
                                            <owl:onProperty
rdf:resource="#sPodrocja"/>
                                            <owl:someValuesFrom>
                                                <owl:Class>
                                                    <owl:oneOf
rdf:parseType="Collection">
                                                        <rdf:Description
rdf:about="#racunalnistvo"/>
                                                        <rdf:Description
rdf:about="#informatika"/>
                                                    </owl:oneOf>
                                                </owl:Class>
                                            </owl:someValuesFrom>
                                        </owl:Restriction>
                                    </owl:complementOf>
                                </owl:Class>
                            </owl:allValuesFrom>
                        </owl:Restriction>
                    </owl:intersectionOf>
                </owl:Class>
            </owl:unionOf>
```

```
            </owl:Class>
        </owl:equivalentClass>
        <rdfs:subClassOf rdf:resource="#Kandidat"/>
        <owl:disjointWith
rdf:resource="#KandidatZZeloPrimernimiDelovnimiIzkusnjami"/>
    </owl:Class>

    <!--
http://www.htoo.org/dm/Kandidati.owl#KandidatZManjPrimernimiOsebnimiLastnostmi -->

    <owl:Class rdf:about="#KandidatZManjPrimernimiOsebnimiLastnostmi">
        <owl:equivalentClass>
            <owl:Class>
                <owl:intersectionOf rdf:parseType="Collection">
                    <rdf:Description rdf:about="#Kandidat"/>
                    <owl:Class>
                        <owl:complementOf
rdf:resource="#KandidatSPrimernimiOsebnimiLastnostmi"/>
                    </owl:Class>
                    <owl:Class>
                        <owl:complementOf
rdf:resource="#KandidatZNajprimernejsimiOsebnimiLastnostmi"/>
                    </owl:Class>
                </owl:intersectionOf>
            </owl:Class>
        </owl:equivalentClass>
        <rdfs:subClassOf rdf:resource="#Kandidat"/>
        <owl:disjointWith
rdf:resource="#KandidatZNajprimernejsimiOsebnimiLastnostmi"/>
    </owl:Class>

    <!-- http://www.htoo.org/dm/Kandidati.owl#KandidatZNajprimernejsimPredznanjem -
->

    <owl:Class rdf:about="#KandidatZNajprimernejsimPredznanjem">
        <owl:equivalentClass>
            <owl:Class>
                <owl:intersectionOf rdf:parseType="Collection">
                    <rdf:Description rdf:about="#Kandidat"/>
                    <rdf:Description
rdf:about="#KandidatSPrimernimZnanjemTujegaJezika"/>
                    <rdf:Description
rdf:about="#KandidatZZeloPrimernimiDelovnimiIzkusnjami"/>
                    <rdf:Description rdf:about="#KandidatZZeloPrimernoIzobrazbo"/>
                </owl:intersectionOf>
            </owl:Class>
        </owl:equivalentClass>
        <rdfs:subClassOf rdf:resource="#Kandidat"/>
        <owl:disjointWith rdf:resource="#KandidatZZeloPrimernimPredznanjem"/>
    </owl:Class>

    <!--
http://www.htoo.org/dm/Kandidati.owl#KandidatZNajprimernejsimRezultatomTesta -->

    <owl:Class rdf:about="#KandidatZNajprimernejsimRezultatomTesta">
        <owl:equivalentClass>
            <owl:Class>
                <owl:intersectionOf rdf:parseType="Collection">
                    <rdf:Description rdf:about="#Kandidat"/>
                    <owl:Restriction>
                        <owl:onProperty rdf:resource="#imaRezultatTesta"/>
                        <owl:allValuesFrom>
                            <owl:Class>
                                <owl:oneOf rdf:parseType="Collection">
                                    <rdf:Description rdf:about="#ocena_5"/>
                                    <rdf:Description rdf:about="#ocena_4"/>
                                </owl:oneOf>
```

```
                            </owl:Class>
                        </owl:allValuesFrom>
                    </owl:Restriction>
                </owl:intersectionOf>
            </owl:Class>
        </owl:equivalentClass>
        <rdfs:subClassOf rdf:resource="#Kandidat"/>
    </owl:Class>

    <!--
http://www.htoo.org/dm/Kandidati.owl#KandidatZNajprimernejsimiOsebnimiLastnostmi --
>

    <owl:Class rdf:about="#KandidatZNajprimernejsimiOsebnimiLastnostmi">
        <owl:equivalentClass>
            <owl:Class>
                <owl:intersectionOf rdf:parseType="Collection">
                    <rdf:Description rdf:about="#KandidatZOdlicnimNastopom"/>
                    <owl:Class>
                        <owl:unionOf rdf:parseType="Collection">
                            <rdf:Description
rdf:about="#KandidatNajprimernejseStarosti"/>
                            <rdf:Description
rdf:about="#KandidatPrimerneStarosti"/>
                        </owl:unionOf>
                    </owl:Class>
                </owl:intersectionOf>
            </owl:Class>
        </owl:equivalentClass>
        <rdfs:subClassOf rdf:resource="#Kandidat"/>
    </owl:Class>

    <!-- http://www.htoo.org/dm/Kandidati.owl#KandidatZNePrimernimNastopom -->

    <owl:Class rdf:about="#KandidatZNePrimernimNastopom">
        <owl:equivalentClass>
            <owl:Class>
                <owl:intersectionOf rdf:parseType="Collection">
                    <rdf:Description rdf:about="#Kandidat"/>
                    <owl:Restriction>
                        <owl:onProperty rdf:resource="#imaOcenoNastopa"/>
                        <owl:allValuesFrom>
                            <owl:Class>
                                <owl:oneOf rdf:parseType="Collection">
                                    <rdf:Description rdf:about="#nePrimerno"/>
                                </owl:oneOf>
                            </owl:Class>
                        </owl:allValuesFrom>
                    </owl:Restriction>
                </owl:intersectionOf>
            </owl:Class>
        </owl:equivalentClass>
        <rdfs:subClassOf rdf:resource="#Kandidat"/>
        <owl:disjointWith rdf:resource="#KandidatZOdlicnimNastopom"/>
    </owl:Class>

    <!--
http://www.htoo.org/dm/Kandidati.owl#KandidatZNeprimernimZnanjemTujegaJezika -->

    <owl:Class rdf:about="#KandidatZNeprimernimZnanjemTujegaJezika">
        <owl:equivalentClass>
            <owl:Class>
                <owl:intersectionOf rdf:parseType="Collection">
                    <rdf:Description rdf:about="#Kandidat"/>
                    <owl:Restriction>
                        <owl:onProperty rdf:resource="#govoriAngleskiJezik"/>
                        <owl:allValuesFrom>
```

```
                            <owl:Class>
                                <owl:oneOf rdf:parseType="Collection">
                                    <rdf:Description rdf:about="#osnovno"/>
                                    <rdf:Description rdf:about="#zacetno"/>
                                    <rdf:Description rdf:about="#brezPredznanja"/>
                                </owl:oneOf>
                            </owl:Class>
                        </owl:allValuesFrom>
                    </owl:Restriction>
                </owl:intersectionOf>
            </owl:Class>
        </owl:equivalentClass>
        <rdfs:subClassOf rdf:resource="#Kandidat"/>
    </owl:Class>

    <!-- http://www.htoo.org/dm/Kandidati.owl#KandidatZNeprimernoIzobrazbo -->

    <owl:Class rdf:about="#KandidatZNeprimernoIzobrazbo">
        <owl:equivalentClass>
            <owl:Class>
                <owl:intersectionOf rdf:parseType="Collection">
                    <rdf:Description rdf:about="#Kandidat"/>
                    <owl:Restriction>
                        <owl:onProperty rdf:resource="#imaIzobrazbo"/>
                        <owl:allValuesFrom rdf:resource="#NeprimernaIzobrazba"/>
                    </owl:Restriction>
                </owl:intersectionOf>
            </owl:Class>
        </owl:equivalentClass>
        <rdfs:subClassOf rdf:resource="#Kandidat"/>
        <owl:disjointWith rdf:resource="#KandidatZZeloPrimernoIzobrazbo"/>
    </owl:Class>

    <!-- http://www.htoo.org/dm/Kandidati.owl#KandidatZOdlicnimNastopom -->

    <owl:Class rdf:about="#KandidatZOdlicnimNastopom">
        <owl:equivalentClass>
            <owl:Class>
                <owl:intersectionOf rdf:parseType="Collection">
                    <rdf:Description rdf:about="#Kandidat"/>
                    <owl:Restriction>
                        <owl:onProperty rdf:resource="#imaOcenoNastopa"/>
                        <owl:allValuesFrom>
                            <owl:Class>
                                <owl:oneOf rdf:parseType="Collection">
                                    <rdf:Description rdf:about="#odlicno"/>
                                </owl:oneOf>
                            </owl:Class>
                        </owl:allValuesFrom>
                    </owl:Restriction>
                </owl:intersectionOf>
            </owl:Class>
        </owl:equivalentClass>
        <rdfs:subClassOf rdf:resource="#Kandidat"/>
    </owl:Class>

    <!-- http://www.htoo.org/dm/Kandidati.owl#KandidatZZeloPrimernimPredznanjem -->

    <owl:Class rdf:about="#KandidatZZeloPrimernimPredznanjem">
        <owl:equivalentClass>
            <owl:Class>
                <owl:intersectionOf rdf:parseType="Collection">
                    <rdf:Description rdf:about="#Kandidat"/>
                    <rdf:Description
rdf:about="#KandidatSPrimernimZnanjemTujegaJezika"/>
                    <owl:Class>
                        <owl:unionOf rdf:parseType="Collection">
```

```
                            <owl:Class>
                                <owl:intersectionOf rdf:parseType="Collection">
                                    <rdf:Description
rdf:about="#KandidatSPrimernimiDelovnimiIzkusnjami"/>
                                    <rdf:Description
rdf:about="#KandidatZZeloPrimernoIzobrazbo"/>
                                </owl:intersectionOf>
                            </owl:Class>
                            <owl:Class>
                                <owl:intersectionOf rdf:parseType="Collection">
                                    <rdf:Description
rdf:about="#KandidatSPrimernoIzobrazbo"/>
                                    <rdf:Description
rdf:about="#KandidatZZeloPrimernimiDelovnimiIzkusnjami"/>
                                </owl:intersectionOf>
                            </owl:Class>
                        </owl:unionOf>
                    </owl:Class>
                    <owl:Class>
                        <owl:complementOf
rdf:resource="#KandidatZNajprimernejsimPredznanjem"/>
                    </owl:Class>
                </owl:intersectionOf>
            </owl:Class>
        </owl:equivalentClass>
        <rdfs:subClassOf rdf:resource="#Kandidat"/>
    </owl:Class>

    <!--
http://www.htoo.org/dm/Kandidati.owl#KandidatZZeloPrimernimiDelovnimiIzkusnjami -->

    <owl:Class rdf:about="#KandidatZZeloPrimernimiDelovnimiIzkusnjami">
        <rdfs:subClassOf rdf:resource="#Kandidat"/>
    </owl:Class>

    <!-- http://www.htoo.org/dm/Kandidati.owl#KandidatZZeloPrimernoIzobrazbo -->

    <owl:Class rdf:about="#KandidatZZeloPrimernoIzobrazbo">
        <owl:equivalentClass>
            <owl:Class>
                <owl:intersectionOf rdf:parseType="Collection">
                    <rdf:Description rdf:about="#Kandidat"/>
                    <owl:Restriction>
                        <owl:onProperty rdf:resource="#imaIzobrazbo"/>
                        <owl:someValuesFrom rdf:resource="#ZeloPrimernaIzobrazba"/>
                    </owl:Restriction>
                </owl:intersectionOf>
            </owl:Class>
        </owl:equivalentClass>
        <rdfs:subClassOf rdf:resource="#Kandidat"/>
    </owl:Class>



    <!-- http://www.htoo.org/dm/Kandidati.owl#NajboljsiKandidat -->

    <owl:Class rdf:about="#NajboljsiKandidat">
        <owl:equivalentClass>
            <owl:Class>
                <owl:unionOf rdf:parseType="Collection">
                    <owl:Class>
                        <owl:intersectionOf rdf:parseType="Collection">
                            <rdf:Description rdf:about="#Kandidat"/>
                            <rdf:Description
rdf:about="#KandidatSPrimernimiOsebnimiLastnostmi"/>
                            <rdf:Description
rdf:about="#KandidatZNajprimernejsimPredznanjem"/>
```

```
                            <rdf:Description
rdf:about="#KandidatZNajprimernejsimRezultatomTesta"/>
                        </owl:intersectionOf>
                    </owl:Class>
                    <owl:Class>
                        <owl:intersectionOf rdf:parseType="Collection">
                            <rdf:Description rdf:about="#Kandidat"/>
                            <rdf:Description
rdf:about="#KandidatZNajprimernejsimPredznanjem"/>
                            <rdf:Description
rdf:about="#KandidatZNajprimernejsimRezultatomTesta"/>
                            <rdf:Description
rdf:about="#KandidatZNajprimernejsimiOsebnimiLastnostmi"/>
                        </owl:intersectionOf>
                    </owl:Class>
                    <owl:Class>
                        <owl:intersectionOf rdf:parseType="Collection">
                            <rdf:Description rdf:about="#Kandidat"/>
                            <rdf:Description
rdf:about="#KandidatZNajprimernejsimRezultatomTesta"/>
                            <rdf:Description
rdf:about="#KandidatZNajprimernejsimiOsebnimiLastnostmi"/>
                            <rdf:Description
rdf:about="#KandidatZZeloPrimernimPredznanjem"/>
                        </owl:intersectionOf>
                    </owl:Class>
                </owl:unionOf>
            </owl:Class>
        </owl:equivalentClass>
        <rdfs:subClassOf rdf:resource="#Kandidat"/>
        <owl:disjointWith rdf:resource="#NeprimerenKandidat"/>
        <owl:disjointWith rdf:resource="#PrimerenKandidat"/>
        <owl:disjointWith rdf:resource="#ZeloDoberKandidat"/>
    </owl:Class>

    <!-- http://www.htoo.org/dm/Kandidati.owl#NeprimerenKandidat -->

    <owl:Class rdf:about="#NeprimerenKandidat">
        <owl:equivalentClass>
            <owl:Class>
                <owl:intersectionOf rdf:parseType="Collection">
                    <rdf:Description rdf:about="#Kandidat"/>
                    <owl:Class>
                        <owl:complementOf rdf:resource="#NajboljsiKandidat"/>
                    </owl:Class>
                    <owl:Class>
                        <owl:complementOf rdf:resource="#PrimerenKandidat"/>
                    </owl:Class>
                    <owl:Class>
                        <owl:complementOf rdf:resource="#ZeloDoberKandidat"/>
                    </owl:Class>
                </owl:intersectionOf>
            </owl:Class>
        </owl:equivalentClass>
        <rdfs:subClassOf rdf:resource="#Kandidat"/>
        <owl:disjointWith rdf:resource="#PrimerenKandidat"/>
        <owl:disjointWith rdf:resource="#ZeloDoberKandidat"/>
    </owl:Class>

    <!-- http://www.htoo.org/dm/Kandidati.owl#NeprimernaIzobrazba -->

    <owl:Class rdf:about="#NeprimernaIzobrazba">
        <owl:equivalentClass>
            <owl:Class>
                <owl:intersectionOf rdf:parseType="Collection">
                    <rdf:Description rdf:about="#Izobrazba"/>
                    <owl:Class>
```

```
                    <owl:complementOf rdf:resource="#PrimernaIzobrazba"/>
                </owl:Class>
                <owl:Class>
                    <owl:complementOf rdf:resource="#ZeloPrimernaIzobrazba"/>
                </owl:Class>
            </owl:intersectionOf>
        </owl:Class>
    </owl:equivalentClass>
    <rdfs:subClassOf rdf:resource="#Izobrazba"/>
    <owl:disjointWith rdf:resource="#PrimernaIzobrazba"/>
    <owl:disjointWith rdf:resource="#ZeloPrimernaIzobrazba"/>
</owl:Class>

<!-- http://www.htoo.org/dm/Kandidati.owl#OcenaNastopa -->

<owl:Class rdf:about="#OcenaNastopa">
    <owl:equivalentClass>
        <owl:Class>
            <owl:oneOf rdf:parseType="Collection">
                <rdf:Description rdf:about="#primerno"/>
                <rdf:Description rdf:about="#nePrimerno"/>
                <rdf:Description rdf:about="#manjPrimerno"/>
                <rdf:Description rdf:about="#odlicno"/>
            </owl:oneOf>
        </owl:Class>
    </owl:equivalentClass>
    <owl:disjointWith rdf:resource="#Podrocje"/>
    <owl:disjointWith rdf:resource="#RezultatTesta"/>
    <owl:disjointWith rdf:resource="#Sektor"/>
    <owl:disjointWith rdf:resource="#StopnjaIzobrazbe"/>
    <owl:disjointWith rdf:resource="#StopnjaZnanjaTujegaJezika"/>
    <owl:disjointWith rdf:resource="#VrstaAktivnosti"/>
    <owl:disjointWith rdf:resource="#Zaposlitev"/>
    <owl:disjointWith rdf:resource="#Zdruzba"/>
</owl:Class>

<!-- http://www.htoo.org/dm/Kandidati.owl#Podrocje -->

<owl:Class rdf:about="#Podrocje">
    <owl:equivalentClass>
        <owl:Class>
            <owl:oneOf rdf:parseType="Collection">
                <rdf:Description rdf:about="#fizika"/>
                <rdf:Description rdf:about="#ekonomija"/>
                <rdf:Description rdf:about="#trzenje"/>
                <rdf:Description rdf:about="#pravo"/>
                <rdf:Description rdf:about="#racunalnistvo"/>
                <rdf:Description rdf:about="#farmacija"/>
                <rdf:Description rdf:about="#matematika"/>
                <rdf:Description rdf:about="#biologija"/>
                <rdf:Description rdf:about="#informatika"/>
                <rdf:Description rdf:about="#elektrotehnika"/>
            </owl:oneOf>
        </owl:Class>
    </owl:equivalentClass>
    <owl:disjointWith rdf:resource="#RezultatTesta"/>
    <owl:disjointWith rdf:resource="#Sektor"/>
    <owl:disjointWith rdf:resource="#StopnjaIzobrazbe"/>
    <owl:disjointWith rdf:resource="#StopnjaZnanjaTujegaJezika"/>
    <owl:disjointWith rdf:resource="#VrstaAktivnosti"/>
    <owl:disjointWith rdf:resource="#Zaposlitev"/>
    <owl:disjointWith rdf:resource="#Zdruzba"/>
</owl:Class>

<!-- http://www.htoo.org/dm/Kandidati.owl#PrimerenKandidat -->

<owl:Class rdf:about="#PrimerenKandidat">
```

```
        <owl:equivalentClass>
            <owl:Class>
                <owl:unionOf rdf:parseType="Collection">
                    <owl:Class>
                        <owl:intersectionOf rdf:parseType="Collection">
                            <rdf:Description rdf:about="#Kandidat"/>
                            <rdf:Description
rdf:about="#KandidatSPrimernimPredznanjem"/>
                            <rdf:Description
rdf:about="#KandidatSPrimernimRezultatomTesta"/>
                            <rdf:Description
rdf:about="#KandidatSPrimernimiOsebnimiLastnostmi"/>
                        </owl:intersectionOf>
                    </owl:Class>
                    <owl:Class>
                        <owl:intersectionOf rdf:parseType="Collection">
                            <rdf:Description rdf:about="#Kandidat"/>
                            <rdf:Description
rdf:about="#KandidatSPrimernimPredznanjem"/>
                            <rdf:Description
rdf:about="#KandidatSPrimernimRezultatomTesta"/>
                            <rdf:Description
rdf:about="#KandidatZNajprimernejsimiOsebnimiLastnostmi"/>
                        </owl:intersectionOf>
                    </owl:Class>
                    <owl:Class>
                        <owl:intersectionOf rdf:parseType="Collection">
                            <rdf:Description rdf:about="#Kandidat"/>
                            <rdf:Description
rdf:about="#KandidatSPrimernimPredznanjem"/>
                            <rdf:Description
rdf:about="#KandidatSPrimernimiOsebnimiLastnostmi"/>
                            <rdf:Description
rdf:about="#KandidatZNajprimernejsimRezultatomTesta"/>
                        </owl:intersectionOf>
                    </owl:Class>
                    <owl:Class>
                        <owl:intersectionOf rdf:parseType="Collection">
                            <rdf:Description rdf:about="#Kandidat"/>
                            <rdf:Description
rdf:about="#KandidatZManjPrimernimiOsebnimiLastnostmi"/>
                            <rdf:Description
rdf:about="#KandidatZNajprimernejsimPredznanjem"/>
                            <rdf:Description
rdf:about="#KandidatZNajprimernejsimRezultatomTesta"/>
                        </owl:intersectionOf>
                    </owl:Class>
                </owl:unionOf>
            </owl:Class>
        </owl:equivalentClass>
        <rdfs:subClassOf rdf:resource="#Kandidat"/>
        <owl:disjointWith rdf:resource="#ZeloDoberKandidat"/>
    </owl:Class>

    <!-- http://www.htoo.org/dm/Kandidati.owl#PrimernaIzobrazba -->

    <owl:Class rdf:about="#PrimernaIzobrazba">
        <owl:equivalentClass>
            <owl:Class>
                <owl:intersectionOf rdf:parseType="Collection">
                    <rdf:Description rdf:about="#Izobrazba"/>
                    <owl:Class>
                        <owl:complementOf rdf:resource="#ZeloPrimernaIzobrazba"/>
                    </owl:Class>
                    <owl:Restriction>
                        <owl:onProperty rdf:resource="#jeStopnje"/>
                        <owl:someValuesFrom>
```

```
                        <owl:Class>
                            <owl:oneOf rdf:parseType="Collection">
                                <rdf:Description rdf:about="#ISCED_5"/>
                                <rdf:Description rdf:about="#ISCED_6"/>
                            </owl:oneOf>
                        </owl:Class>
                    </owl:someValuesFrom>
                </owl:Restriction>
                <owl:Restriction>
                    <owl:onProperty rdf:resource="#sPodrocja"/>
                    <owl:someValuesFrom>
                        <owl:Class>
                            <owl:oneOf rdf:parseType="Collection">
                                <rdf:Description rdf:about="#matematika"/>
                            </owl:oneOf>
                        </owl:Class>
                    </owl:someValuesFrom>
                </owl:Restriction>
            </owl:intersectionOf>
        </owl:Class>
    </owl:equivalentClass>
    <rdfs:subClassOf rdf:resource="#Izobrazba"/>
    <owl:disjointWith rdf:resource="#ZeloPrimernaIzobrazba"/>
</owl:Class>

<!-- http://www.htoo.org/dm/Kandidati.owl#RezultatTesta -->

<owl:Class rdf:about="#RezultatTesta">
    <owl:equivalentClass>
        <owl:Class>
            <owl:oneOf rdf:parseType="Collection">
                <rdf:Description rdf:about="#ocena_1"/>
                <rdf:Description rdf:about="#ocena_2"/>
                <rdf:Description rdf:about="#ocena_5"/>
                <rdf:Description rdf:about="#ocena_3"/>
                <rdf:Description rdf:about="#ocena_4"/>
            </owl:oneOf>
        </owl:Class>
    </owl:equivalentClass>
    <owl:disjointWith rdf:resource="#Sektor"/>
    <owl:disjointWith rdf:resource="#StopnjaIzobrazbe"/>
    <owl:disjointWith rdf:resource="#StopnjaZnanjaTujegaJezika"/>
    <owl:disjointWith rdf:resource="#VrstaAktivnosti"/>
    <owl:disjointWith rdf:resource="#Zaposlitev"/>
    <owl:disjointWith rdf:resource="#Zdruzba"/>
</owl:Class>

<!-- http://www.htoo.org/dm/Kandidati.owl#Sektor -->

<owl:Class rdf:about="#Sektor">
    <owl:disjointWith rdf:resource="#StopnjaIzobrazbe"/>
    <owl:disjointWith rdf:resource="#StopnjaZnanjaTujegaJezika"/>
    <owl:disjointWith rdf:resource="#VrstaAktivnosti"/>
    <owl:disjointWith rdf:resource="#Zaposlitev"/>
    <owl:disjointWith rdf:resource="#Zdruzba"/>
</owl:Class>

<!-- http://www.htoo.org/dm/Kandidati.owl#StopnjaIzobrazbe -->

<owl:Class rdf:about="#StopnjaIzobrazbe">
    <owl:equivalentClass>
        <owl:Class>
            <owl:oneOf rdf:parseType="Collection">
                <rdf:Description rdf:about="#ISCED_1"/>
                <rdf:Description rdf:about="#ISCED_3"/>
                <rdf:Description rdf:about="#ISCED_4"/>
                <rdf:Description rdf:about="#ISCED_2"/>
```

```
                <rdf:Description rdf:about="#ISCED_6"/>
                <rdf:Description rdf:about="#ISCED_5"/>
                <rdf:Description rdf:about="#ISCED_0"/>
            </owl:oneOf>
        </owl:Class>
    </owl:equivalentClass>
    <owl:disjointWith rdf:resource="#StopnjaZnanjaTujegaJezika"/>
    <owl:disjointWith rdf:resource="#VrstaAktivnosti"/>
    <owl:disjointWith rdf:resource="#Zaposlitev"/>
    <owl:disjointWith rdf:resource="#Zdruzba"/>
</owl:Class>

<!-- http://www.htoo.org/dm/Kandidati.owl#StopnjaZnanjaTujegaJezika -->

<owl:Class rdf:about="#StopnjaZnanjaTujegaJezika">
    <owl:equivalentClass>
        <owl:Class>
            <owl:oneOf rdf:parseType="Collection">
                <rdf:Description rdf:about="#tekoce"/>
                <rdf:Description rdf:about="#dobro"/>
                <rdf:Description rdf:about="#osnovno"/>
                <rdf:Description rdf:about="#zacetno"/>
                <rdf:Description rdf:about="#brezPredznanja"/>
            </owl:oneOf>
        </owl:Class>
    </owl:equivalentClass>
    <owl:disjointWith rdf:resource="#VrstaAktivnosti"/>
    <owl:disjointWith rdf:resource="#Zaposlitev"/>
    <owl:disjointWith rdf:resource="#Zdruzba"/>
</owl:Class>

<!-- http://www.htoo.org/dm/Kandidati.owl#VrstaAktivnosti -->

<owl:Class rdf:about="#VrstaAktivnosti">
    <owl:equivalentClass>
        <owl:Class>
            <owl:oneOf rdf:parseType="Collection">
                <rdf:Description rdf:about="#vodenjeITprojektov"/>
                <rdf:Description rdf:about="#programiranje"/>
                <rdf:Description rdf:about="#knjigovodenje"/>
                <rdf:Description rdf:about="#PR"/>
                <rdf:Description rdf:about="#skupinskoDelo"/>
                <rdf:Description rdf:about="#razvojUporabniskihVmesnikov"/>
                <rdf:Description rdf:about="#administriranjePodatkovneBaze"/>
                <rdf:Description rdf:about="#nacrtovanjeArhitekturIS"/>
                <rdf:Description rdf:about="#razvojSpletnihStoritev"/>
                <rdf:Description rdf:about="#poucevanje"/>
                <rdf:Description rdf:about="#poslovnoKomuniciranje"/>
            </owl:oneOf>
        </owl:Class>
    </owl:equivalentClass>
    <owl:disjointWith rdf:resource="#Zaposlitev"/>
    <owl:disjointWith rdf:resource="#Zdruzba"/>
</owl:Class>

<!-- http://www.htoo.org/dm/Kandidati.owl#Zaposlitev -->

<owl:Class rdf:about="#Zaposlitev">
    <owl:disjointWith rdf:resource="#Zdruzba"/>
</owl:Class>

<!-- http://www.htoo.org/dm/Kandidati.owl#Zdruzba -->

<owl:Class rdf:about="#Zdruzba"/>
```

```
    <!-- http://www.htoo.org/dm/Kandidati.owl#ZeloDoberKandidat -->

    <owl:Class rdf:about="#ZeloDoberKandidat">
        <owl:equivalentClass>
            <owl:Class>
                <owl:unionOf rdf:parseType="Collection">
                    <owl:Class>
                        <owl:intersectionOf rdf:parseType="Collection">
                            <rdf:Description rdf:about="#Kandidat"/>
                            <rdf:Description
rdf:about="#KandidatSPrimernimPredznanjem"/>
                            <rdf:Description
rdf:about="#KandidatZNajprimernejsimRezultatomTesta"/>
                            <rdf:Description
rdf:about="#KandidatZNajprimernejsimiOsebnimiLastnostmi"/>
                        </owl:intersectionOf>
                    </owl:Class>
                    <owl:Class>
                        <owl:intersectionOf rdf:parseType="Collection">
                            <rdf:Description rdf:about="#Kandidat"/>
                            <rdf:Description
rdf:about="#KandidatSPrimernimRezultatomTesta"/>
                            <rdf:Description
rdf:about="#KandidatSPrimernimiOsebnimiLastnostmi"/>
                            <rdf:Description
rdf:about="#KandidatZNajprimernejsimPredznanjem"/>
                        </owl:intersectionOf>
                    </owl:Class>
                    <owl:Class>
                        <owl:intersectionOf rdf:parseType="Collection">
                            <rdf:Description rdf:about="#Kandidat"/>
                            <rdf:Description
rdf:about="#KandidatSPrimernimRezultatomTesta"/>
                            <rdf:Description
rdf:about="#KandidatSPrimernimiOsebnimiLastnostmi"/>
                            <rdf:Description
rdf:about="#KandidatZZeloPrimernimPredznanjem"/>
                        </owl:intersectionOf>
                    </owl:Class>
                    <owl:Class>
                        <owl:intersectionOf rdf:parseType="Collection">
                            <rdf:Description rdf:about="#Kandidat"/>
                            <rdf:Description
rdf:about="#KandidatSPrimernimRezultatomTesta"/>
                            <rdf:Description
rdf:about="#KandidatZNajprimernejsimPredznanjem"/>
                            <rdf:Description
rdf:about="#KandidatZNajprimernejsimiOsebnimiLastnostmi"/>
                        </owl:intersectionOf>
                    </owl:Class>
                    <owl:Class>
                        <owl:intersectionOf rdf:parseType="Collection">
                            <rdf:Description rdf:about="#Kandidat"/>
                            <rdf:Description
rdf:about="#KandidatSPrimernimRezultatomTesta"/>
                            <rdf:Description
rdf:about="#KandidatZNajprimernejsimiOsebnimiLastnostmi"/>
                            <rdf:Description
rdf:about="#KandidatZZeloPrimernimPredznanjem"/>
                        </owl:intersectionOf>
                    </owl:Class>
                    <owl:Class>
                        <owl:intersectionOf rdf:parseType="Collection">
                            <rdf:Description rdf:about="#Kandidat"/>
                            <rdf:Description
rdf:about="#KandidatSPrimernimiOsebnimiLastnostmi"/>
```

```
                            <rdf:Description
rdf:about="#KandidatZNajprimernejsimRezultatomTesta"/>
                            <rdf:Description
rdf:about="#KandidatZZeloPrimernimPredznanjem"/>
                        </owl:intersectionOf>
                    </owl:Class>
                </owl:unionOf>
            </owl:Class>
        </owl:equivalentClass>
        <rdfs:subClassOf rdf:resource="#Kandidat"/>
    </owl:Class>

    <!-- http://www.htoo.org/dm/Kandidati.owl#ZeloPrimernaIzobrazba -->

    <owl:Class rdf:about="#ZeloPrimernaIzobrazba">
        <owl:equivalentClass>
            <owl:Class>
                <owl:intersectionOf rdf:parseType="Collection">
                    <rdf:Description rdf:about="#Izobrazba"/>
                    <owl:Restriction>
                        <owl:onProperty rdf:resource="#jeStopnje"/>
                        <owl:someValuesFrom>
                            <owl:Class>
                                <owl:oneOf rdf:parseType="Collection">
                                    <rdf:Description rdf:about="#ISCED_5"/>
                                    <rdf:Description rdf:about="#ISCED_6"/>
                                </owl:oneOf>
                            </owl:Class>
                        </owl:someValuesFrom>
                    </owl:Restriction>
                    <owl:Restriction>
                        <owl:onProperty rdf:resource="#sPodrocja"/>
                        <owl:someValuesFrom>
                            <owl:Class>
                                <owl:oneOf rdf:parseType="Collection">
                                    <rdf:Description rdf:about="#racunalnistvo"/>
                                    <rdf:Description rdf:about="#informatika"/>
                                </owl:oneOf>
                            </owl:Class>
                        </owl:someValuesFrom>
                    </owl:Restriction>
                </owl:intersectionOf>
            </owl:Class>
        </owl:equivalentClass>
        <rdfs:subClassOf rdf:resource="#Izobrazba"/>
    </owl:Class>

    <!-- http://www.htoo.org/dm/Kandidati.owl#ZeloPrimernoPodrocje -->

    <owl:Class rdf:about="#ZeloPrimernoPodrocje">
        <owl:equivalentClass>
            <owl:Class>
                <owl:oneOf rdf:parseType="Collection">
                    <rdf:Description rdf:about="#racunalnistvo"/>
                    <rdf:Description rdf:about="#informatika"/>
                </owl:oneOf>
            </owl:Class>
        </owl:equivalentClass>
        <rdfs:subClassOf rdf:resource="#Podrocje"/>
    </owl:Class>

    <!--

///////////////////////////////////////////////////////////////////////////////
////
    //
    // Primerki
```

```
    //

////////////////////////////////////////////////////////////////////////////////
////
     -->

    <!-- http://www.htoo.org/dm/Kandidati.owl#ISCED_0 -->
    <StopnjaIzobrazbe rdf:about="#ISCED_0"/>
    <!-- http://www.htoo.org/dm/Kandidati.owl#ISCED_1 -->
    <StopnjaIzobrazbe rdf:about="#ISCED_1"/>
    <!-- http://www.htoo.org/dm/Kandidati.owl#ISCED_2 -->
    <StopnjaIzobrazbe rdf:about="#ISCED_2"/>
    <!-- http://www.htoo.org/dm/Kandidati.owl#ISCED_3 -->
    <StopnjaIzobrazbe rdf:about="#ISCED_3"/>
    <!-- http://www.htoo.org/dm/Kandidati.owl#ISCED_4 -->
    <StopnjaIzobrazbe rdf:about="#ISCED_4"/>
    <!-- http://www.htoo.org/dm/Kandidati.owl#ISCED_5 -->
    <StopnjaIzobrazbe rdf:about="#ISCED_5"/>
    <!-- http://www.htoo.org/dm/Kandidati.owl#ISCED_6 -->
    <StopnjaIzobrazbe rdf:about="#ISCED_6"/>
    <!-- http://www.htoo.org/dm/Kandidati.owl#PR -->
    <VrstaAktivnosti rdf:about="#PR"/>
    <!-- http://www.htoo.org/dm/Kandidati.owl#administriranjePodatkovneBaze -->
    <VrstaAktivnosti rdf:about="#administriranjePodatkovneBaze"/>
    <!-- http://www.htoo.org/dm/Kandidati.owl#biologija -->
    <Podrocje rdf:about="#biologija"/>
    <!-- http://www.htoo.org/dm/Kandidati.owl#brezPredznanja -->
    <StopnjaZnanjaTujegaJezika rdf:about="#brezPredznanja"/>
    <!-- http://www.htoo.org/dm/Kandidati.owl#dobro -->
    <StopnjaZnanjaTujegaJezika rdf:about="#dobro"/>
    <!-- http://www.htoo.org/dm/Kandidati.owl#ekonomija -->
    <Podrocje rdf:about="#ekonomija"/>
    <!-- http://www.htoo.org/dm/Kandidati.owl#elektrotehnika -->
    <Podrocje rdf:about="#elektrotehnika"/>
    <!-- http://www.htoo.org/dm/Kandidati.owl#farmacija -->
    <Podrocje rdf:about="#farmacija"/>
    <!-- http://www.htoo.org/dm/Kandidati.owl#fizika -->
    <Podrocje rdf:about="#fizika"/>
    <!-- http://www.htoo.org/dm/Kandidati.owl#informatika -->
    <ZeloPrimernoPodrocje rdf:about="#informatika">
        <rdf:type rdf:resource="#Podrocje"/>
    </ZeloPrimernoPodrocje>
    <!-- http://www.htoo.org/dm/Kandidati.owl#knjigovodenje -->
    <VrstaAktivnosti rdf:about="#knjigovodenje"/>
    <!-- http://www.htoo.org/dm/Kandidati.owl#manjPrimerno -->
    <OcenaNastopa rdf:about="#manjPrimerno"/>
    <!-- http://www.htoo.org/dm/Kandidati.owl#matematika -->
    <Podrocje rdf:about="#matematika"/>
    <!-- http://www.htoo.org/dm/Kandidati.owl#nacrtovanjeArhitekturIS -->
    <VrstaAktivnosti rdf:about="#nacrtovanjeArhitekturIS"/>
    <!-- http://www.htoo.org/dm/Kandidati.owl#nePrimerno -->
    <OcenaNastopa rdf:about="#nePrimerno"/>
    <!-- http://www.htoo.org/dm/Kandidati.owl#ocena_1 -->
    <RezultatTesta rdf:about="#ocena_1"/>
    <!-- http://www.htoo.org/dm/Kandidati.owl#ocena_2 -->
    <RezultatTesta rdf:about="#ocena_2"/>
    <!-- http://www.htoo.org/dm/Kandidati.owl#ocena_3 -->
    <RezultatTesta rdf:about="#ocena_3"/>
    <!-- http://www.htoo.org/dm/Kandidati.owl#ocena_4 -->
    <RezultatTesta rdf:about="#ocena_4"/>
    <!-- http://www.htoo.org/dm/Kandidati.owl#ocena_5 -->
    <RezultatTesta rdf:about="#ocena_5"/>
    <!-- http://www.htoo.org/dm/Kandidati.owl#odlicno -->
    <OcenaNastopa rdf:about="#odlicno"/>
    <!-- http://www.htoo.org/dm/Kandidati.owl#osnovno -->
    <StopnjaZnanjaTujegaJezika rdf:about="#osnovno"/>
    <!-- http://www.htoo.org/dm/Kandidati.owl#poslovnoKomuniciranje -->
```

```
    <VrstaAktivnosti rdf:about="#poslovnoKomuniciranje"/>
    <!-- http://www.htoo.org/dm/Kandidati.owl#poucevanje -->
    <VrstaAktivnosti rdf:about="#poucevanje"/>
    <!-- http://www.htoo.org/dm/Kandidati.owl#pravo -->
    <Podrocje rdf:about="#pravo"/>
    <!-- http://www.htoo.org/dm/Kandidati.owl#primerno -->
    <OcenaNastopa rdf:about="#primerno"/>
    <!-- http://www.htoo.org/dm/Kandidati.owl#programiranje -->
    <VrstaAktivnosti rdf:about="#programiranje"/>
    <!-- http://www.htoo.org/dm/Kandidati.owl#racunalnistvo -->
    <Podrocje rdf:about="#racunalnistvo">
        <rdf:type rdf:resource="#ZeloPrimernoPodrocje"/>
    </Podrocje>
    <!-- http://www.htoo.org/dm/Kandidati.owl#razvojSpletnihStoritev -->
    <VrstaAktivnosti rdf:about="#razvojSpletnihStoritev"/>
    <!-- http://www.htoo.org/dm/Kandidati.owl#razvojUporabniskihVmesnikov -->
    <VrstaAktivnosti rdf:about="#razvojUporabniskihVmesnikov"/>
    <!-- http://www.htoo.org/dm/Kandidati.owl#skupinskoDelo -->
    <VrstaAktivnosti rdf:about="#skupinskoDelo"/>
    <!-- http://www.htoo.org/dm/Kandidati.owl#tekoce -->
    <StopnjaZnanjaTujegaJezika rdf:about="#tekoce"/>
    <!-- http://www.htoo.org/dm/Kandidati.owl#trzenje -->
    <Podrocje rdf:about="#trzenje"/>
    <!-- http://www.htoo.org/dm/Kandidati.owl#vodenjeITprojektov -->
    <VrstaAktivnosti rdf:about="#vodenjeITprojektov"/>
    <!-- http://www.htoo.org/dm/Kandidati.owl#zacetno -->
    <StopnjaZnanjaTujegaJezika rdf:about="#zacetno"/>

    <!--

///////////////////////////////////////////////////////////////////////////////////
////
    //
    // Axiomi
    //

///////////////////////////////////////////////////////////////////////////////////
////
     -->

    <rdf:Description>
        <rdf:type rdf:resource="&owl;AllDifferent"/>
        <owl:distinctMembers rdf:parseType="Collection">
            <rdf:Description rdf:about="#primerno"/>
            <rdf:Description rdf:about="#nePrimerno"/>
            <rdf:Description rdf:about="#manjPrimerno"/>
            <rdf:Description rdf:about="#odlicno"/>
        </owl:distinctMembers>
    </rdf:Description>
    <rdf:Description>
        <rdf:type rdf:resource="&owl;AllDifferent"/>
        <owl:distinctMembers rdf:parseType="Collection">
            <rdf:Description rdf:about="#ISCED_1"/>
            <rdf:Description rdf:about="#ISCED_3"/>
            <rdf:Description rdf:about="#ISCED_4"/>
            <rdf:Description rdf:about="#ISCED_2"/>
            <rdf:Description rdf:about="#ISCED_6"/>
            <rdf:Description rdf:about="#ISCED_5"/>
            <rdf:Description rdf:about="#ISCED_0"/>
        </owl:distinctMembers>
    </rdf:Description>
    <rdf:Description>
        <rdf:type rdf:resource="&owl;AllDifferent"/>
        <owl:distinctMembers rdf:parseType="Collection">
            <rdf:Description rdf:about="#vodenjeITprojektov"/>
            <rdf:Description rdf:about="#programiranje"/>
            <rdf:Description rdf:about="#knjigovodenje"/>
```

```
            <rdf:Description rdf:about="#PR"/>
            <rdf:Description rdf:about="#skupinskoDelo"/>
            <rdf:Description rdf:about="#razvojUporabniskihVmesnikov"/>
            <rdf:Description rdf:about="#administriranjePodatkovneBaze"/>
            <rdf:Description rdf:about="#nacrtovanjeArhitekturIS"/>
            <rdf:Description rdf:about="#razvojSpletnihStoritev"/>
            <rdf:Description rdf:about="#poucevanje"/>
            <rdf:Description rdf:about="#poslovnoKomuniciranje"/>
        </owl:distinctMembers>
    </rdf:Description>
    <rdf:Description>
        <rdf:type rdf:resource="&owl;AllDifferent"/>
        <owl:distinctMembers rdf:parseType="Collection">
            <rdf:Description rdf:about="#tekoce"/>
            <rdf:Description rdf:about="#dobro"/>
            <rdf:Description rdf:about="#osnovno"/>
            <rdf:Description rdf:about="#zacetno"/>
            <rdf:Description rdf:about="#brezPredznanja"/>
        </owl:distinctMembers>
    </rdf:Description>
    <rdf:Description>
        <rdf:type rdf:resource="&owl;AllDifferent"/>
        <owl:distinctMembers rdf:parseType="Collection">
            <rdf:Description rdf:about="#ocena_1"/>
            <rdf:Description rdf:about="#ocena_2"/>
            <rdf:Description rdf:about="#ocena_5"/>
            <rdf:Description rdf:about="#ocena_3"/>
            <rdf:Description rdf:about="#ocena_4"/>
        </owl:distinctMembers>
    </rdf:Description>
    <rdf:Description>
        <rdf:type rdf:resource="&owl;AllDifferent"/>
        <owl:distinctMembers rdf:parseType="Collection">
            <rdf:Description rdf:about="#fizika"/>
            <rdf:Description rdf:about="#ekonomija"/>
            <rdf:Description rdf:about="#trzenje"/>
            <rdf:Description rdf:about="#pravo"/>
            <rdf:Description rdf:about="#racunalnistvo"/>
            <rdf:Description rdf:about="#farmacija"/>
            <rdf:Description rdf:about="#matematika"/>
            <rdf:Description rdf:about="#biologija"/>
            <rdf:Description rdf:about="#informatika"/>
            <rdf:Description rdf:about="#elektrotehnika"/>
        </owl:distinctMembers>
    </rdf:Description>


    <!--

///////////////////////////////////////////////////////////////////////////////
////
    //
    // SWRL pravila
    //

///////////////////////////////////////////////////////////////////////////////
////
     -->

    <swrl:Variable rdf:about="#poslovnoKomuniciranje"/>
    <swrl:Variable rdf:about="#y1"/>
    <swrl:Variable rdf:about="#vodenjeITprojektov"/>
    <swrl:Variable rdf:about="#kDR"/>
    <swrl:Variable rdf:about="#p5"/>
    <swrl:Variable rdf:about="#y5"/>
    <swrl:Variable rdf:about="#razvojSpletnihStoritev"/>
    <swrl:Variable rdf:about="#skupinskoDelo"/>
```

```
    <swrl:Variable rdf:about="#y4"/>
    <swrl:Variable rdf:about="#nacrtovanjeArhitekturIS"/>
    <swrl:Variable rdf:about="#p1"/>
    <swrl:Variable rdf:about="#p2"/>
    <swrl:Variable rdf:about="#y3"/>
    <swrl:Variable rdf:about="#p4"/>
    <swrl:Variable rdf:about="#y2"/>
    <swrl:Variable rdf:about="#p3"/>
    <swrl:Variable rdf:about="#x"/>
    <swrl:Variable rdf:about="#stLetDI"/>
    <swrl:Variable rdf:about="#dtY"/>
    <swrl:Imp>
        <swrl:head>
            <swrl:AtomList>
                <rdf:rest rdf:resource="&rdf;nil"/>
                <rdf:first>
                    <swrl:ClassAtom>
                        <swrl:classPredicate
rdf:resource="#KandidatSPrimernimiDelovnimiIzkusnjami"/>
                        <swrl:argument1 rdf:resource="#x"/>
                    </swrl:ClassAtom>
                </rdf:first>
            </swrl:AtomList>
        </swrl:head>
        <swrl:body>
            <swrl:AtomList>
                <rdf:rest>
                    <swrl:AtomList>
                        <rdf:first>
                            <swrl:ClassAtom>
                                <swrl:classPredicate
rdf:resource="#ZeloPrimernoPodrocje"/>
                                <swrl:argument1 rdf:resource="#p1"/>
                            </swrl:ClassAtom>
                        </rdf:first>
                        <rdf:rest>
                            <swrl:AtomList>
                                <rdf:rest>
                                    <swrl:AtomList>
                                        <rdf:first>
                                            <swrl:ClassAtom>
                                                <swrl:classPredicate
rdf:resource="#ZeloPrimernoPodrocje"/>
                                                <swrl:argument1
rdf:resource="#p3"/>
                                            </swrl:ClassAtom>
                                        </rdf:first>
                                        <rdf:rest>
                                            <swrl:AtomList>
                                                <rdf:rest>
                                                    <swrl:AtomList>
                                                        <rdf:rest>
                                                            <swrl:AtomList>
                                                                <rdf:first>

<swrl:IndividualPropertyAtom>
<swrl:propertyPredicate rdf:resource="#jeImelZaposlitev"/>
<swrl:argument1 rdf:resource="#x"/>
<swrl:argument2 rdf:resource="#y1"/>
</swrl:IndividualPropertyAtom>
</rdf:first>
<rdf:rest>
<swrl:AtomList>
<rdf:rest>
<swrl:AtomList>
<rdf:first>
<swrl:IndividualPropertyAtom>
```

```
<swrl:propertyPredicate rdf:resource="#jeImelZaposlitev"/>
<swrl:argument1 rdf:resource="#x"/>
<swrl:argument2 rdf:resource="#y3"/>
</swrl:IndividualPropertyAtom>
</rdf:first>
<rdf:rest>
<swrl:AtomList>
<rdf:first>
<swrl:IndividualPropertyAtom>
<swrl:propertyPredicate rdf:resource="#jeImelZaposlitev"/>
<swrl:argument1 rdf:resource="#x"/>
<swrl:argument2 rdf:resource="#y4"/>
</swrl:IndividualPropertyAtom>
</rdf:first>
<rdf:rest>
<swrl:AtomList>
<rdf:first>
<swrl:IndividualPropertyAtom>
<swrl:propertyPredicate rdf:resource="#jeImelZaposlitev"/>
<swrl:argument1 rdf:resource="#x"/>
<swrl:argument2 rdf:resource="#y5"/>
</swrl:IndividualPropertyAtom>
</rdf:first>
<rdf:rest>
<swrl:AtomList>
<rdf:rest>
<swrl:AtomList>
<rdf:first>
<swrl:IndividualPropertyAtom>
<swrl:propertyPredicate rdf:resource="#obsegaVrstoAktivnosti"/>
<swrl:argument2 rdf:resource="#poslovnoKomuniciranje"/>
<swrl:argument1 rdf:resource="#y2"/>
</swrl:IndividualPropertyAtom>
</rdf:first>
<rdf:rest>
<swrl:AtomList>
<rdf:rest>
<swrl:AtomList>
<rdf:rest>
<swrl:AtomList>
<rdf:rest>
<swrl:AtomList>
<rdf:first>
<swrl:IndividualPropertyAtom>
<swrl:argument2 rdf:resource="#p1"/>
<swrl:propertyPredicate rdf:resource="#sPodrocja"/>
<swrl:argument1 rdf:resource="#y1"/>
</swrl:IndividualPropertyAtom>
</rdf:first>
<rdf:rest>
<swrl:AtomList>
<rdf:rest>
<swrl:AtomList>
<rdf:rest>
<swrl:AtomList>
<rdf:first>
<swrl:IndividualPropertyAtom>
<swrl:argument2 rdf:resource="#p4"/>
<swrl:propertyPredicate rdf:resource="#sPodrocja"/>
<swrl:argument1 rdf:resource="#y4"/>
</swrl:IndividualPropertyAtom>
</rdf:first>
<rdf:rest>
<swrl:AtomList>
<rdf:first>
<swrl:IndividualPropertyAtom>
<swrl:argument2 rdf:resource="#p5"/>
```

```
<swrl:propertyPredicate rdf:resource="#sPodrocja"/>
<swrl:argument1 rdf:resource="#y5"/>
</swrl:IndividualPropertyAtom>
</rdf:first>
<rdf:rest>
<swrl:AtomList>
<rdf:rest>
<swrl:AtomList>
<rdf:rest>
<swrl:AtomList>
<rdf:rest rdf:resource="&rdf;nil"/>
<rdf:first>
<swrl:BuiltinAtom>
<swrl:builtin rdf:resource="&swrlb;greaterThanOrEqual"/>
<swrl:arguments>
<rdf:Description>
<rdf:type rdf:resource="&rdf;List"/>
<rdf:first rdf:resource="#stLetDI"/>
<rdf:rest>
<rdf:Description>
<rdf:type rdf:resource="&rdf;List"/>
<rdf:first rdf:datatype="&xsd;integer">8</rdf:first>
<rdf:rest rdf:resource="&rdf;nil"/>
</rdf:Description>
</rdf:rest>
</rdf:Description>
</swrl:arguments>
</swrl:BuiltinAtom>
</rdf:first>
</swrl:AtomList>
</rdf:rest>
<rdf:first>
<swrl:BuiltinAtom>
<swrl:builtin rdf:resource="&swrlb;lessThan"/>
<swrl:arguments>
<rdf:Description>
<rdf:type rdf:resource="&rdf;List"/>
<rdf:first rdf:resource="#stLetDI"/>
<rdf:rest>
<rdf:Description>
<rdf:type rdf:resource="&rdf;List"/>
<rdf:first rdf:datatype="&xsd;integer">10</rdf:first>
<rdf:rest rdf:resource="&rdf;nil"/>
</rdf:Description>
</rdf:rest>
</rdf:Description>
</swrl:arguments>
</swrl:BuiltinAtom>
</rdf:first>
</swrl:AtomList>
</rdf:rest>
<rdf:first>
<swrl:DatavaluedPropertyAtom>
<swrl:argument2 rdf:resource="#stLetDI"/>
<swrl:propertyPredicate rdf:resource="#steviloLetDelovnihIzkusenj"/>
<swrl:argument1 rdf:resource="#x"/>
</swrl:DatavaluedPropertyAtom>
</rdf:first>
</swrl:AtomList>
</rdf:rest>
</swrl:AtomList>
</rdf:rest>
</swrl:AtomList>
</rdf:rest>
<rdf:first>
<swrl:IndividualPropertyAtom>
<swrl:argument2 rdf:resource="#p3"/>
```

```
<swrl:propertyPredicate rdf:resource="#sPodrocja"/>
<swrl:argument1 rdf:resource="#y3"/>
</swrl:IndividualPropertyAtom>
</rdf:first>
</swrl:AtomList>
</rdf:rest>
<rdf:first>
<swrl:IndividualPropertyAtom>
<swrl:argument2 rdf:resource="#p2"/>
<swrl:propertyPredicate rdf:resource="#sPodrocja"/>
<swrl:argument1 rdf:resource="#y2"/>
</swrl:IndividualPropertyAtom>
</rdf:first>
</swrl:AtomList>
</rdf:rest>
</swrl:AtomList>
</rdf:rest>
<rdf:first>
<swrl:IndividualPropertyAtom>
<swrl:propertyPredicate rdf:resource="#obsegaVrstoAktivnosti"/>
<swrl:argument2 rdf:resource="#razvojSpletnihStoritev"/>
<swrl:argument1 rdf:resource="#y5"/>
</swrl:IndividualPropertyAtom>
</rdf:first>
</swrl:AtomList>
</rdf:rest>
<rdf:first>
<swrl:IndividualPropertyAtom>
<swrl:argument2 rdf:resource="#nacrtovanjeArhitekturIS"/>
<swrl:propertyPredicate rdf:resource="#obsegaVrstoAktivnosti"/>
<swrl:argument1 rdf:resource="#y4"/>
</swrl:IndividualPropertyAtom>
</rdf:first>
</swrl:AtomList>
</rdf:rest>
<rdf:first>
<swrl:IndividualPropertyAtom>
<swrl:propertyPredicate rdf:resource="#obsegaVrstoAktivnosti"/>
<swrl:argument2 rdf:resource="#skupinskoDelo"/>
<swrl:argument1 rdf:resource="#y3"/>
</swrl:IndividualPropertyAtom>
</rdf:first>
</swrl:AtomList>
</rdf:rest>
</swrl:AtomList>
</rdf:rest>
<rdf:first>
<swrl:IndividualPropertyAtom>
<swrl:propertyPredicate rdf:resource="#obsegaVrstoAktivnosti"/>
<swrl:argument2 rdf:resource="#vodenjeITprojektov"/>
<swrl:argument1 rdf:resource="#y1"/>
</swrl:IndividualPropertyAtom>
</rdf:first>
</swrl:AtomList>
</rdf:rest>
</swrl:AtomList>
</rdf:rest>
</swrl:AtomList>
</rdf:rest>
</swrl:AtomList>
</rdf:rest>
<rdf:first>
<swrl:IndividualPropertyAtom>
<swrl:propertyPredicate rdf:resource="#jeImelZaposlitev"/>
<swrl:argument1 rdf:resource="#x"/>
<swrl:argument2 rdf:resource="#y2"/>
```

```
</swrl:IndividualPropertyAtom>
</rdf:first>

</swrl:AtomList>
                                                                    </rdf:rest>
                                                                </swrl:AtomList>
                                                            </rdf:rest>
                                                            <rdf:first>
                                                                <swrl:ClassAtom>

<swrl:classPredicate rdf:resource="#ZeloPrimernoPodrocje"/>
                                                                    <swrl:argument1
rdf:resource="#p5"/>
                                                                </swrl:ClassAtom>
                                                            </rdf:first>
                                                        </swrl:AtomList>
                                                    </rdf:rest>
                                                    <rdf:first>
                                                        <swrl:ClassAtom>
                                                            <swrl:classPredicate
rdf:resource="#ZeloPrimernoPodrocje"/>
                                                            <swrl:argument1
rdf:resource="#p4"/>
                                                        </swrl:ClassAtom>
                                                    </rdf:first>
                                                </swrl:AtomList>
                                            </rdf:rest>
                                        </swrl:AtomList>
                                    </rdf:rest>
                                    <rdf:first>
                                        <swrl:ClassAtom>
                                            <swrl:classPredicate
rdf:resource="#ZeloPrimernoPodrocje"/>
                                            <swrl:argument1 rdf:resource="#p2"/>
                                        </swrl:ClassAtom>
                                    </rdf:first>
                                </swrl:AtomList>
                            </rdf:rest>
                        </swrl:AtomList>
                    </rdf:rest>
                    <rdf:first>
                        <swrl:ClassAtom>
                            <swrl:classPredicate rdf:resource="#Kandidat"/>
                            <swrl:argument1 rdf:resource="#x"/>
                        </swrl:ClassAtom>
                    </rdf:first>
                </swrl:AtomList>
            </swrl:body>
        </swrl:Imp>
        <swrl:Imp>
            <swrl:body>
                <swrl:AtomList>
                    <rdf:rest>
                        <swrl:AtomList>
                            <rdf:first>
                                <swrl:DatavaluedPropertyAtom>
                                    <swrl:argument2 rdf:resource="#dtY"/>
                                    <swrl:propertyPredicate
rdf:resource="#imaDatumRojstva"/>
                                    <swrl:argument1 rdf:resource="#kDR"/>
                                </swrl:DatavaluedPropertyAtom>
                            </rdf:first>
                            <rdf:rest>
                                <swrl:AtomList>
                                    <rdf:first>
                                        <swrl:BuiltinAtom>
```

```
                                        <swrl:builtin
rdf:resource="&swrlb;lessThan"/>
                                        <swrl:arguments>
                                            <rdf:Description>
                                                <rdf:type
rdf:resource="&rdf;List"/>
                                                <rdf:first rdf:resource="#dtY"/>
                                                <rdf:rest>
                                                    <rdf:Description>
                                                        <rdf:type
rdf:resource="&rdf;List"/>
                                                        <rdf:first
rdf:datatype="&xsd;date">1984-05-30</rdf:first>
                                                        <rdf:rest
rdf:resource="&rdf;nil"/>
                                                    </rdf:Description>
                                                </rdf:rest>
                                            </rdf:Description>
                                        </swrl:arguments>
                                    </swrl:BuiltinAtom>
                                </rdf:first>
                                <rdf:rest>
                                    <swrl:AtomList>
                                        <rdf:rest rdf:resource="&rdf;nil"/>
                                        <rdf:first>
                                            <swrl:BuiltinAtom>
                                                <swrl:builtin
rdf:resource="&swrlb;greaterThan"/>
                                                <swrl:arguments>
                                                    <rdf:Description>
                                                        <rdf:type
rdf:resource="&rdf;List"/>
                                                        <rdf:first
rdf:resource="#dtY"/>
                                                        <rdf:rest>
                                                            <rdf:Description>
                                                                <rdf:type
rdf:resource="&rdf;List"/>
                                                                <rdf:first
rdf:datatype="&xsd;date">1974-05-30</rdf:first>
                                                                <rdf:rest
rdf:resource="&rdf;nil"/>
                                                            </rdf:Description>
                                                        </rdf:rest>
                                                    </rdf:Description>
                                                </swrl:arguments>
                                            </swrl:BuiltinAtom>
                                        </rdf:first>
                                    </swrl:AtomList>
                                </rdf:rest>
                            </swrl:AtomList>
                        </rdf:rest>
                    </swrl:AtomList>
                </rdf:rest>
                <rdf:first>
                    <swrl:ClassAtom>
                        <swrl:classPredicate rdf:resource="#Kandidat"/>
                        <swrl:argument1 rdf:resource="#kDR"/>
                    </swrl:ClassAtom>
                </rdf:first>
            </swrl:AtomList>
        </swrl:body>
        <swrl:head>
            <swrl:AtomList>
                <rdf:rest rdf:resource="&rdf;nil"/>
                <rdf:first>
                    <swrl:ClassAtom>
```

```
                        <swrl:classPredicate
rdf:resource="#KandidatPrimerneStarosti"/>
                        <swrl:argument1 rdf:resource="#kDR"/>
                    </swrl:ClassAtom>
                </rdf:first>
            </swrl:AtomList>
        </swrl:head>
    </swrl:Imp>
    <swrl:Imp>
        <swrl:head>
            <swrl:AtomList>
                <rdf:rest rdf:resource="&rdf;nil"/>
                <rdf:first>
                    <swrl:ClassAtom>
                        <swrl:classPredicate
rdf:resource="#KandidatManjPrimerneStarosti"/>
                        <swrl:argument1 rdf:resource="#kDR"/>
                    </swrl:ClassAtom>
                </rdf:first>
            </swrl:AtomList>
        </swrl:head>
        <swrl:body>
            <swrl:AtomList>
                <rdf:rest>
                    <swrl:AtomList>
                        <rdf:rest>
                            <swrl:AtomList>
                                <rdf:rest rdf:resource="&rdf;nil"/>
                                <rdf:first>
                                    <swrl:BuiltinAtom>
                                        <swrl:builtin
rdf:resource="&swrlb;greaterThan"/>
                                        <swrl:arguments>
                                            <rdf:Description>
                                                <rdf:type
rdf:resource="&rdf;List"/>
                                                <rdf:first rdf:resource="#dtY"/>
                                                <rdf:rest>
                                                    <rdf:Description>
                                                        <rdf:type
rdf:resource="&rdf;List"/>
                                                        <rdf:first
rdf:datatype="&xsd;date">1984-05-30</rdf:first>
                                                        <rdf:rest
rdf:resource="&rdf;nil"/>
                                                    </rdf:Description>
                                                </rdf:rest>
                                            </rdf:Description>
                                        </swrl:arguments>
                                    </swrl:BuiltinAtom>
                                </rdf:first>
                            </swrl:AtomList>
                        </rdf:rest>
                        <rdf:first>
                            <swrl:DatavaluedPropertyAtom>
                                <swrl:argument2 rdf:resource="#dtY"/>
                                <swrl:propertyPredicate
rdf:resource="#imaDatumRojstva"/>
                                <swrl:argument1 rdf:resource="#kDR"/>
                            </swrl:DatavaluedPropertyAtom>
                        </rdf:first>
                    </swrl:AtomList>
                </rdf:rest>
                <rdf:first>
                    <swrl:ClassAtom>
                        <swrl:classPredicate rdf:resource="#Kandidat"/>
                        <swrl:argument1 rdf:resource="#kDR"/>
```

```
                    </swrl:ClassAtom>
                </rdf:first>
            </swrl:AtomList>
        </swrl:body>
    </swrl:Imp>
    <swrl:Imp>
        <swrl:head>
            <swrl:AtomList>
                <rdf:rest rdf:resource="&rdf;nil"/>
                <rdf:first>
                    <swrl:ClassAtom>
                        <swrl:classPredicate
rdf:resource="#KandidatZZeloPrimernimiDelovnimiIzkusnjami"/>
                        <swrl:argument1 rdf:resource="#x"/>
                    </swrl:ClassAtom>
                </rdf:first>
            </swrl:AtomList>
        </swrl:head>
        <swrl:body>
            <swrl:AtomList>
                <rdf:first>
                    <swrl:ClassAtom>
                        <swrl:classPredicate rdf:resource="#Kandidat"/>
                        <swrl:argument1 rdf:resource="#x"/>
                    </swrl:ClassAtom>
                </rdf:first>
                <rdf:rest>
                    <swrl:AtomList>
                        <rdf:rest>
                            <swrl:AtomList>
                                <rdf:rest>
                                    <swrl:AtomList>
                                        <rdf:first>
                                            <swrl:ClassAtom>
                                                <swrl:classPredicate
rdf:resource="#ZeloPrimernoPodrocje"/>
                                                <swrl:argument1
rdf:resource="#p3"/>
                                            </swrl:ClassAtom>
                                        </rdf:first>
                                        <rdf:rest>
                                            <swrl:AtomList>
                                                <rdf:first>
                                                    <swrl:ClassAtom>
                                                        <swrl:classPredicate
rdf:resource="#ZeloPrimernoPodrocje"/>
                                                        <swrl:argument1
rdf:resource="#p4"/>
                                                    </swrl:ClassAtom>
                                                </rdf:first>
                                                <rdf:rest>
                                                    <swrl:AtomList>
                                                        <rdf:first>
                                                            <swrl:ClassAtom>

<swrl:classPredicate rdf:resource="#ZeloPrimernoPodrocje"/>
                                                                <swrl:argument1
rdf:resource="#p5"/>
                                                            </swrl:ClassAtom>
                                                        </rdf:first>
                                                        <rdf:rest>
                                                            <swrl:AtomList>
                                                                <rdf:rest>
                                                                    <swrl:AtomList>
<rdf:rest>
<swrl:AtomList>
<rdf:first>
```

```
<swrl:IndividualPropertyAtom>
<swrl:propertyPredicate rdf:resource="#jeImelZaposlitev"/>
<swrl:argument1 rdf:resource="#x"/>
<swrl:argument2 rdf:resource="#y3"/>
</swrl:IndividualPropertyAtom>
</rdf:first>
<rdf:rest>
<swrl:AtomList>
<rdf:rest>
<swrl:AtomList>
<rdf:first>
<swrl:IndividualPropertyAtom>
<swrl:propertyPredicate rdf:resource="#jeImelZaposlitev"/>
<swrl:argument1 rdf:resource="#x"/>
<swrl:argument2 rdf:resource="#y5"/>
</swrl:IndividualPropertyAtom>
</rdf:first>
<rdf:rest>
<swrl:AtomList>
<rdf:rest>
<swrl:AtomList>
<rdf:first>
<swrl:IndividualPropertyAtom>
<swrl:propertyPredicate rdf:resource="#obsegaVrstoAktivnosti"/>
<swrl:argument2 rdf:resource="#poslovnoKomuniciranje"/>
<swrl:argument1 rdf:resource="#y2"/>
</swrl:IndividualPropertyAtom>
</rdf:first>
<rdf:rest>
<swrl:AtomList>
<rdf:rest>
<swrl:AtomList>
<rdf:first>
<swrl:IndividualPropertyAtom>
<swrl:argument2 rdf:resource="#nacrtovanjeArhitekturIS"/>
<swrl:propertyPredicate rdf:resource="#obsegaVrstoAktivnosti"/>
<swrl:argument1 rdf:resource="#y4"/>
</swrl:IndividualPropertyAtom>
</rdf:first>
<rdf:rest>
<swrl:AtomList>
<rdf:rest>

<swrl:AtomList>
<rdf:rest>
<swrl:AtomList>
<rdf:rest>
<swrl:AtomList>
<rdf:first>
<swrl:IndividualPropertyAtom>
<swrl:argument2 rdf:resource="#p3"/>
<swrl:propertyPredicate rdf:resource="#sPodrocja"/>
<swrl:argument1 rdf:resource="#y3"/>
</swrl:IndividualPropertyAtom>
</rdf:first>
<rdf:rest>
<swrl:AtomList>
<rdf:first>
<swrl:IndividualPropertyAtom>
<swrl:argument2 rdf:resource="#p4"/>
<swrl:propertyPredicate rdf:resource="#sPodrocja"/>
<swrl:argument1 rdf:resource="#y4"/>
</swrl:IndividualPropertyAtom>
</rdf:first>
<rdf:rest>
<swrl:AtomList>
<rdf:first>
```

```
<swrl:IndividualPropertyAtom>
<swrl:argument2 rdf:resource="#p5"/>
<swrl:propertyPredicate rdf:resource="#sPodrocja"/>
<swrl:argument1 rdf:resource="#y5"/>
</swrl:IndividualPropertyAtom>
</rdf:first>
<rdf:rest>
<swrl:AtomList>
<rdf:first>
<swrl:DatavaluedPropertyAtom>
<swrl:argument2 rdf:resource="#stLetDI"/>
<swrl:propertyPredicate rdf:resource="#steviloLetDelovnihIzkusenj"/>
<swrl:argument1 rdf:resource="#x"/>
</swrl:DatavaluedPropertyAtom>
</rdf:first>
<rdf:rest>
<swrl:AtomList>
<rdf:rest rdf:resource="&rdf;nil"/>
<rdf:first>
<swrl:BuiltinAtom>
<swrl:builtin rdf:resource="&swrlb;greaterThanOrEqual"/>
<swrl:arguments>
<rdf:Description>
<rdf:type rdf:resource="&rdf;List"/>
<rdf:first rdf:resource="#stLetDI"/>
<rdf:rest>
<rdf:Description>
<rdf:type rdf:resource="&rdf;List"/>
<rdf:first rdf:datatype="&xsd;integer">10</rdf:first>
<rdf:rest rdf:resource="&rdf;nil"/>
</rdf:Description>
</rdf:rest>
</rdf:Description>
</swrl:arguments>
</swrl:BuiltinAtom>

</rdf:first>
</swrl:AtomList>
</rdf:rest>
</swrl:AtomList>
</rdf:rest>
</swrl:AtomList>
</rdf:rest>
</swrl:AtomList>
</rdf:rest>
</swrl:AtomList>
</rdf:rest>
<rdf:first>
<swrl:IndividualPropertyAtom>
<swrl:argument2 rdf:resource="#p2"/>
<swrl:propertyPredicate rdf:resource="#sPodrocja"/>
<swrl:argument1 rdf:resource="#y2"/>
</swrl:IndividualPropertyAtom>
</rdf:first>
</swrl:AtomList>
</rdf:rest>
<rdf:first>
<swrl:IndividualPropertyAtom>
<swrl:argument2 rdf:resource="#p1"/>
<swrl:propertyPredicate rdf:resource="#sPodrocja"/>
<swrl:argument1 rdf:resource="#y1"/>
</swrl:IndividualPropertyAtom>
</rdf:first>
</swrl:AtomList>
</rdf:rest>
<rdf:first>
<swrl:IndividualPropertyAtom>
```

```
<swrl:propertyPredicate rdf:resource="#obsegaVrstoAktivnosti"/>
<swrl:argument2 rdf:resource="#razvojSpletnihStoritev"/>
<swrl:argument1 rdf:resource="#y5"/>
</swrl:IndividualPropertyAtom>
</rdf:first>
</swrl:AtomList>
</rdf:rest>
</swrl:AtomList>
</rdf:rest>
<rdf:first>
<swrl:IndividualPropertyAtom>
<swrl:propertyPredicate rdf:resource="#obsegaVrstoAktivnosti"/>
<swrl:argument2 rdf:resource="#skupinskoDelo"/>
<swrl:argument1 rdf:resource="#y3"/>
</swrl:IndividualPropertyAtom>
</rdf:first>
</swrl:AtomList>
</rdf:rest>
</swrl:AtomList>
</rdf:rest>
<rdf:first>
<swrl:IndividualPropertyAtom>
<swrl:propertyPredicate rdf:resource="#obsegaVrstoAktivnosti"/>
<swrl:argument2 rdf:resource="#vodenjeITprojektov"/>
<swrl:argument1 rdf:resource="#y1"/>
</swrl:IndividualPropertyAtom>
</rdf:first>
</swrl:AtomList>
</rdf:rest>
</swrl:AtomList>
</rdf:rest>
<rdf:first>
<swrl:IndividualPropertyAtom>
<swrl:propertyPredicate rdf:resource="#jeImelZaposlitev"/>
<swrl:argument1 rdf:resource="#x"/>
<swrl:argument2 rdf:resource="#y4"/>
</swrl:IndividualPropertyAtom>
</rdf:first>
</swrl:AtomList>
</rdf:rest>
</swrl:AtomList>
</rdf:rest>
<rdf:first>
<swrl:IndividualPropertyAtom>
<swrl:propertyPredicate rdf:resource="#jeImelZaposlitev"/>
<swrl:argument1 rdf:resource="#x"/>
<swrl:argument2 rdf:resource="#y2"/>
</swrl:IndividualPropertyAtom>
</rdf:first>
</swrl:AtomList>
</rdf:rest>
<rdf:first>
<swrl:IndividualPropertyAtom>
<swrl:propertyPredicate rdf:resource="#jeImelZaposlitev"/>
<swrl:argument1 rdf:resource="#x"/>
<swrl:argument2 rdf:resource="#y1"/>
</swrl:IndividualPropertyAtom>
</rdf:first>
</swrl:AtomList>
                                                        </rdf:rest>
                                                    </swrl:AtomList>
                                                </rdf:rest>
                                            </swrl:AtomList>
                                        </rdf:rest>
                                    </swrl:AtomList>
                                </rdf:rest>
                                <rdf:first>
```

```
                                    <swrl:ClassAtom>
                                        <swrl:classPredicate
rdf:resource="#ZeloPrimernoPodrocje"/>
                                        <swrl:argument1 rdf:resource="#p2"/>
                                    </swrl:ClassAtom>
                                </rdf:first>
                            </swrl:AtomList>
                        </rdf:rest>
                        <rdf:first>
                            <swrl:ClassAtom>
                                <swrl:classPredicate
rdf:resource="#ZeloPrimernoPodrocje"/>
                                <swrl:argument1 rdf:resource="#p1"/>
                            </swrl:ClassAtom>
                        </rdf:first>
                    </swrl:AtomList>
                </rdf:rest>
            </swrl:AtomList>
        </swrl:body>
    </swrl:Imp>
    <swrl:Imp>
        <swrl:body>
            <swrl:AtomList>
                <rdf:rest>
                    <swrl:AtomList>
                        <rdf:first>
                            <swrl:DatavaluedPropertyAtom>
                                <swrl:argument2 rdf:resource="#stLetDI"/>
                                <swrl:propertyPredicate
rdf:resource="#steviloLetDelovnihIzkusenj"/>
                                <swrl:argument1 rdf:resource="#x"/>
                            </swrl:DatavaluedPropertyAtom>
                        </rdf:first>
                        <rdf:rest>
                            <swrl:AtomList>
                                <rdf:rest rdf:resource="&rdf;nil"/>
                                <rdf:first>
                                    <swrl:BuiltinAtom>
                                        <swrl:builtin
rdf:resource="&swrlb;lessThan"/>
                                        <swrl:arguments>
                                            <rdf:Description>
                                                <rdf:type
rdf:resource="&rdf;List"/>
                                                <rdf:first
rdf:resource="#stLetDI"/>
                                                <rdf:rest>
                                                    <rdf:Description>
                                                        <rdf:type
rdf:resource="&rdf;List"/>
                                                        <rdf:first
rdf:datatype="&xsd;integer">8</rdf:first>
                                                        <rdf:rest
rdf:resource="&rdf;nil"/>
                                                    </rdf:Description>
                                                </rdf:rest>
                                            </rdf:Description>
                                        </swrl:arguments>
                                    </swrl:BuiltinAtom>
                                </rdf:first>
                            </swrl:AtomList>
                        </rdf:rest>
                    </swrl:AtomList>
                </rdf:rest>
                <rdf:first>
                    <swrl:ClassAtom>
                        <swrl:classPredicate rdf:resource="#Kandidat"/>
```

```
                        <swrl:argument1 rdf:resource="#x"/>
                    </swrl:ClassAtom>
                </rdf:first>
            </swrl:AtomList>
        </swrl:body>
        <swrl:head>
            <swrl:AtomList>
                <rdf:rest rdf:resource="&rdf;nil"/>
                <rdf:first>
                    <swrl:ClassAtom>
                        <swrl:classPredicate
rdf:resource="#KandidatZManjPrimernimiDelovnimiIzkusnjami"/>
                        <swrl:argument1 rdf:resource="#x"/>
                    </swrl:ClassAtom>
                </rdf:first>
            </swrl:AtomList>
        </swrl:head>
    </swrl:Imp>
    <swrl:Imp>
        <swrl:head>
            <swrl:AtomList>
                <rdf:rest rdf:resource="&rdf;nil"/>
                <rdf:first>
                    <swrl:ClassAtom>
                        <swrl:classPredicate
rdf:resource="#KandidatNajprimernejseStarosti"/>
                        <swrl:argument1 rdf:resource="#kDR"/>
                    </swrl:ClassAtom>
                </rdf:first>
            </swrl:AtomList>
        </swrl:head>
        <swrl:body>
            <swrl:AtomList>
                <rdf:first>
                    <swrl:DatavaluedPropertyAtom>
                        <swrl:argument2 rdf:resource="#dtY"/>
                        <swrl:propertyPredicate rdf:resource="#imaDatumRojstva"/>
                        <swrl:argument1 rdf:resource="#kDR"/>
                    </swrl:DatavaluedPropertyAtom>
                </rdf:first>
                <rdf:rest>
                    <swrl:AtomList>
                        <rdf:rest rdf:resource="&rdf;nil"/>
                        <rdf:first>
                            <swrl:BuiltinAtom>
                                <swrl:builtin rdf:resource="&swrlb;lessThan"/>
                                <swrl:arguments>
                                    <rdf:Description>
                                        <rdf:type rdf:resource="&rdf;List"/>
                                        <rdf:first rdf:resource="#dtY"/>
                                        <rdf:rest>
                                            <rdf:Description>
                                                <rdf:type
rdf:resource="&rdf;List"/>
                                                <rdf:first
rdf:datatype="&xsd;date">1974-05-30</rdf:first>
                                                <rdf:rest rdf:resource="&rdf;nil"/>
                                            </rdf:Description>
                                        </rdf:rest>
                                    </rdf:Description>
                                </swrl:arguments>
                            </swrl:BuiltinAtom>
                        </rdf:first>
                    </swrl:AtomList>
                </rdf:rest>
            </swrl:AtomList>
        </swrl:body>
```

```
    </swrl:Imp>
</rdf:RDF>
```

Priloga 4: Temeljna ontologija knjižnice protokolov

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


<!DOCTYPE rdf:RDF [
    <!ENTITY owl "http://www.w3.org/2002/07/owl#" >
    <!ENTITY swrl "http://www.w3.org/2003/11/swrl#" >
    <!ENTITY swrlb "http://www.w3.org/2003/11/swrlb#" >
    <!ENTITY xsd "http://www.w3.org/2001/XMLSchema#" >
    <!ENTITY owl2xml "http://www.w3.org/2006/12/owl2-xml#" >
    <!ENTITY rdfs "http://www.w3.org/2000/01/rdf-schema#" >
    <!ENTITY TOKP "http://www.owl-ontologies.com/TOKP.owl#" >
    <!ENTITY rdf "http://www.w3.org/1999/02/22-rdf-syntax-ns#" >
    <!ENTITY protege "http://protege.stanford.edu/plugins/owl/protege#" >
    <!ENTITY xsp "http://www.owl-ontologies.com/2005/08/07/xsp.owl#" >
]>

<rdf:RDF xmlns="http://www.owl-ontologies.com/TOKP.owl#"
     xml:base="http://www.owl-ontologies.com/TOKP.owl"
     xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
     xmlns:swrl="http://www.w3.org/2003/11/swrl#"
     xmlns:protege="http://protege.stanford.edu/plugins/owl/protege#"
     xmlns:owl2xml="http://www.w3.org/2006/12/owl2-xml#"
     xmlns:xsp="http://www.owl-ontologies.com/2005/08/07/xsp.owl#"
     xmlns:TOKP="http://www.owl-ontologies.com/TOKP.owl#"
     xmlns:owl="http://www.w3.org/2002/07/owl#"
     xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
     xmlns:swrlb="http://www.w3.org/2003/11/swrlb#"
     xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#">
    <owl:Ontology rdf:about=""/>

    <!--

////////////////////////////////////////////////////////////////////////////////////
////
    //
    // Objektne lastnosti
    //

////////////////////////////////////////////////////////////////////////////////////
////
     -->

    <!-- http://www.owl-ontologies.com/TOKP.owl#imaCasovniRok -->
    <owl:ObjectProperty rdf:about="#imaCasovniRok">
        <rdfs:range rdf:resource="#CasovniRok"/>
        <rdfs:domain rdf:resource="#Opravilo"/>
    </owl:ObjectProperty>
    <!-- http://www.owl-ontologies.com/TOKP.owl#imaPosiljatelja -->
    <owl:ObjectProperty rdf:about="#imaPosiljatelja">
        <rdf:type rdf:resource="&owl;FunctionalProperty"/>
        <rdfs:range rdf:resource="#Agent"/>
        <rdfs:domain rdf:resource="#Prejem"/>
    </owl:ObjectProperty>
    <!-- http://www.owl-ontologies.com/TOKP.owl#imaPrejemnika -->
    <owl:ObjectProperty rdf:about="#imaPrejemnika">
        <rdfs:range rdf:resource="#Agent"/>
        <rdfs:domain rdf:resource="#Posiljanje"/>
```

```
</owl:ObjectProperty>
<!-- http://www.owl-ontologies.com/TOKP.owl#imaTipOpravila -->
<owl:ObjectProperty rdf:about="#imaTipOpravila">
    <rdf:type rdf:resource="&owl;FunctionalProperty"/>
    <rdfs:domain rdf:resource="#TipOpravilaAgenta"/>
    <rdfs:range>
        <owl:Class>
            <owl:intersectionOf rdf:parseType="Collection">
                <rdf:Description rdf:about="#TipOpravila"/>
                <owl:Class>
                    <owl:complementOf rdf:resource="#TipOpravilaAgenta"/>
                </owl:Class>
            </owl:intersectionOf>
        </owl:Class>
    </rdfs:range>
</owl:ObjectProperty>
<!-- http://www.owl-ontologies.com/TOKP.owl#imaVlogo -->
<owl:ObjectProperty rdf:about="#imaVlogo">
    <rdfs:domain rdf:resource="#Agent"/>
    <rdfs:range rdf:resource="#Vloga"/>
</owl:ObjectProperty>
<!-- http://www.owl-ontologies.com/TOKP.owl#imaVsebinoSporocila -->
<owl:ObjectProperty rdf:about="#imaVsebinoSporocila">
    <rdfs:domain rdf:resource="#KomunikacijskiAkt"/>
    <rdfs:range rdf:resource="#VsebinaSporocila"/>
</owl:ObjectProperty>
<!-- http://www.owl-ontologies.com/TOKP.owl#jeTipa -->
<owl:ObjectProperty rdf:about="#jeTipa">
    <rdf:type rdf:resource="&owl;FunctionalProperty"/>
    <rdfs:domain rdf:resource="#Opravilo"/>
    <rdfs:range rdf:resource="#TipOpravila"/>
</owl:ObjectProperty>
<!-- http://www.owl-ontologies.com/TOKP.owl#jeVStanju -->
<owl:ObjectProperty rdf:about="#jeVStanju">
    <rdf:type rdf:resource="&owl;FunctionalProperty"/>
    <rdfs:domain rdf:resource="#Opravilo"/>
    <rdfs:range rdf:resource="#Stanje"/>
</owl:ObjectProperty>
<!-- http://www.owl-ontologies.com/TOKP.owl#posodobi -->
<owl:ObjectProperty rdf:about="#posodobi">
    <rdfs:domain rdf:resource="#PosodobitevOP"/>
</owl:ObjectProperty>
<!-- http://www.owl-ontologies.com/TOKP.owl#spremeniStanjeCR -->
<owl:ObjectProperty rdf:about="#spremeniStanjeCR">
    <rdfs:range rdf:resource="#CasovniRok"/>
    <rdfs:domain rdf:resource="#SpremembaStanjaCR"/>
    <rdfs:subPropertyOf rdf:resource="#posodobi"/>
</owl:ObjectProperty>
<!-- http://www.owl-ontologies.com/TOKP.owl#spremeniStanjeKA -->
<owl:ObjectProperty rdf:about="#spremeniStanjeKA">
    <rdfs:range rdf:resource="#Opravilo"/>
    <rdfs:domain rdf:resource="#SpremembaStanjaKA"/>
    <rdfs:subPropertyOf rdf:resource="#posodobi"/>
</owl:ObjectProperty>
<!-- http://www.owl-ontologies.com/TOKP.owl#vsebuje -->
<owl:ObjectProperty rdf:about="#vsebuje">
    <rdfs:domain rdf:resource="#VsebinaSporocila"/>
</owl:ObjectProperty>
<!-- http://www.owl-ontologies.com/TOKP.owl#za -->
<owl:ObjectProperty rdf:about="#za">
    <rdf:type rdf:resource="&owl;FunctionalProperty"/>
    <rdfs:domain rdf:resource="#TipOpravilaAgenta"/>
    <rdfs:range rdf:resource="#Vloga"/>
</owl:ObjectProperty>


<!--
```

```
///////////////////////////////////////////////////////////////////////////////
////
    //
    // Podatkovne lastnosti
    //

///////////////////////////////////////////////////////////////////////////////
////
     -->

    <!-- http://www.owl-ontologies.com/TOKP.owl#dodajZOznako -->
    <owl:DatatypeProperty rdf:about="#dodajZOznako">
        <rdfs:domain rdf:resource="#TipOpravila"/>
        <rdfs:range rdf:resource="&xsd;string"/>
    </owl:DatatypeProperty>
    <!-- http://www.owl-ontologies.com/TOKP.owl#imaKonec -->
    <owl:DatatypeProperty rdf:about="#imaKonec">
        <rdf:type rdf:resource="&owl;FunctionalProperty"/>
        <rdfs:domain rdf:resource="#CasovniRok"/>
        <rdfs:range rdf:resource="&xsd;dateTime"/>
    </owl:DatatypeProperty>
    <!-- http://www.owl-ontologies.com/TOKP.owl#imaOznako -->
    <owl:DatatypeProperty rdf:about="#imaOznako">
        <rdf:type rdf:resource="&owl;FunctionalProperty"/>
        <rdfs:domain rdf:resource="#Opravilo"/>
        <rdfs:range rdf:resource="&xsd;string"/>
    </owl:DatatypeProperty>
    <!-- http://www.owl-ontologies.com/TOKP.owl#imaZacetek -->
    <owl:DatatypeProperty rdf:about="#imaZacetek">
        <rdf:type rdf:resource="&owl;FunctionalProperty"/>
        <rdfs:domain rdf:resource="#CasovniRok"/>
        <rdfs:range rdf:resource="&xsd;dateTime"/>
    </owl:DatatypeProperty>
    <!-- http://www.owl-ontologies.com/TOKP.owl#jeAktiven -->
    <owl:DatatypeProperty rdf:about="#jeAktiven">
        <rdf:type rdf:resource="&owl;FunctionalProperty"/>
        <rdfs:domain rdf:resource="#CasovniRok"/>
        <rdfs:range rdf:resource="&xsd;boolean"/>
    </owl:DatatypeProperty>
    <!-- http://www.owl-ontologies.com/TOKP.owl#jePotekel -->
    <owl:DatatypeProperty rdf:about="#jePotekel">
        <rdf:type rdf:resource="&owl;FunctionalProperty"/>
        <rdfs:domain rdf:resource="#CasovniRok"/>
        <rdfs:range rdf:resource="&xsd;boolean"/>
    </owl:DatatypeProperty>

    <!--

///////////////////////////////////////////////////////////////////////////////
////
    //
    // Razredi
    //

///////////////////////////////////////////////////////////////////////////////
////
     -->

    <!-- http://www.owl-ontologies.com/TOKP.owl#Agent -->
    <owl:Class rdf:about="#Agent">
        <rdfs:subClassOf rdf:resource="#Protokol_Thing"/>
    </owl:Class>
    <!-- http://www.owl-ontologies.com/TOKP.owl#CasovniRok -->
    <owl:Class rdf:about="#CasovniRok">
        <rdfs:subClassOf rdf:resource="#Protokol_Thing"/>
    </owl:Class>
```

```
<!-- http://www.owl-ontologies.com/TOKP.owl#Iniciacija -->
<owl:Class rdf:about="#Iniciacija">
    <rdfs:subClassOf rdf:resource="#Opravilo"/>
    <owl:disjointWith rdf:resource="#KomunikacijskiAkt"/>
    <owl:disjointWith rdf:resource="#PosodobitevOP"/>
</owl:Class>
<!-- http://www.owl-ontologies.com/TOKP.owl#KomunikacijskiAkt -->
<owl:Class rdf:about="#KomunikacijskiAkt">
    <owl:equivalentClass>
        <owl:Class>
            <owl:unionOf rdf:parseType="Collection">
                <rdf:Description rdf:about="#Posiljanje"/>
                <rdf:Description rdf:about="#Prejem"/>
            </owl:unionOf>
        </owl:Class>
    </owl:equivalentClass>
    <rdfs:subClassOf rdf:resource="#Opravilo"/>
    <owl:disjointWith rdf:resource="#PosodobitevOP"/>
</owl:Class>
<!-- http://www.owl-ontologies.com/TOKP.owl#Opravilo -->
<owl:Class rdf:about="#Opravilo">
    <rdfs:subClassOf rdf:resource="#Protokol_Thing"/>
</owl:Class>
<!-- http://www.owl-ontologies.com/TOKP.owl#Posiljanje -->
<owl:Class rdf:about="#Posiljanje">
    <rdfs:subClassOf rdf:resource="#KomunikacijskiAkt"/>
    <owl:disjointWith rdf:resource="#Prejem"/>
</owl:Class>
<!-- http://www.owl-ontologies.com/TOKP.owl#PosodobitevOP -->
<owl:Class rdf:about="#PosodobitevOP">
    <rdfs:subClassOf rdf:resource="#Opravilo"/>
</owl:Class>
<!-- http://www.owl-ontologies.com/TOKP.owl#Prejem -->
<owl:Class rdf:about="#Prejem">
    <rdfs:subClassOf rdf:resource="#KomunikacijskiAkt"/>
</owl:Class>
<!-- http://www.owl-ontologies.com/TOKP.owl#Protokol -->
<owl:Class rdf:about="#Protokol">
    <rdfs:subClassOf rdf:resource="#Protokol_Thing"/>
</owl:Class>
<!-- http://www.owl-ontologies.com/TOKP.owl#Protokol_Thing -->
<owl:Class rdf:about="#Protokol_Thing"/>
<!-- http://www.owl-ontologies.com/TOKP.owl#SpremembaStanja -->
<owl:Class rdf:about="#SpremembaStanja">
    <rdfs:subClassOf rdf:resource="#PosodobitevOP"/>
</owl:Class>
<!-- http://www.owl-ontologies.com/TOKP.owl#SpremembaStanjaCR -->
<owl:Class rdf:about="#SpremembaStanjaCR">
    <rdfs:subClassOf rdf:resource="#SpremembaStanja"/>
</owl:Class>
<!-- http://www.owl-ontologies.com/TOKP.owl#SpremembaStanjaKA -->
<owl:Class rdf:about="#SpremembaStanjaKA">
    <rdfs:subClassOf rdf:resource="#SpremembaStanja"/>
</owl:Class>
<!-- http://www.owl-ontologies.com/TOKP.owl#Stanje -->
<owl:Class rdf:about="#Stanje">
    <rdfs:subClassOf rdf:resource="#Protokol_Thing"/>
</owl:Class>
<!-- http://www.owl-ontologies.com/TOKP.owl#TipKomunikacijskegaAkta -->
<owl:Class rdf:about="#TipKomunikacijskegaAkta">
    <rdfs:subClassOf rdf:resource="#TipOpravila"/>
</owl:Class>
<!-- http://www.owl-ontologies.com/TOKP.owl#TipOpravila -->
<owl:Class rdf:about="#TipOpravila">
    <rdfs:subClassOf rdf:resource="#Protokol_Thing"/>
</owl:Class>
<!-- http://www.owl-ontologies.com/TOKP.owl#TipOpravilaAgenta -->
```

```
    <owl:Class rdf:about="#TipOpravilaAgenta">
        <rdfs:subClassOf rdf:resource="#TipOpravila"/>
    </owl:Class>
    <!-- http://www.owl-ontologies.com/TOKP.owl#TipPosiljanje -->
    <owl:Class rdf:about="#TipPosiljanje">
        <rdfs:subClassOf rdf:resource="#TipKomunikacijskegaAkta"/>
    </owl:Class>
    <!-- http://www.owl-ontologies.com/TOKP.owl#TipPrejem -->
    <owl:Class rdf:about="#TipPrejem">
        <rdfs:subClassOf rdf:resource="#TipKomunikacijskegaAkta"/>
    </owl:Class>
    <!-- http://www.owl-ontologies.com/TOKP.owl#Vloga -->
    <owl:Class rdf:about="#Vloga">
        <rdfs:subClassOf rdf:resource="#Protokol_Thing"/>
    </owl:Class>
    <!-- http://www.owl-ontologies.com/TOKP.owl#VsebinaSporocila -->
    <owl:Class rdf:about="#VsebinaSporocila">
        <rdfs:subClassOf rdf:resource="#Protokol_Thing"/>
    </owl:Class>

    <!--

////////////////////////////////////////////////////////////////////////////////
////
    //
    // Primerki
    //

////////////////////////////////////////////////////////////////////////////////
////
     -->

    <!-- http://www.owl-ontologies.com/TOKP.owl#aktiviraj -->
    <SpremembaStanjaCR rdf:about="#aktiviraj"/>
    <!-- http://www.owl-ontologies.com/TOKP.owl#deaktiviraj -->
    <SpremembaStanjaCR rdf:about="#deaktiviraj"/>
    <!-- http://www.owl-ontologies.com/TOKP.owl#iniciacija -->
    <Iniciacija rdf:about="#iniciacija">
        <jeVStanju rdf:resource="#pripravljen"/>
    </Iniciacija>
    <!-- http://www.owl-ontologies.com/TOKP.owl#koncan -->
    <Stanje rdf:about="#koncan"/>
    <!-- http://www.owl-ontologies.com/TOKP.owl#nepravilnoZakljucen -->
    <Stanje rdf:about="#nepravilnoZakljucen"/>
    <!-- http://www.owl-ontologies.com/TOKP.owl#odstrani -->
    <PosodobitevOP rdf:about="#odstrani"/>
    <!-- http://www.owl-ontologies.com/TOKP.owl#prekini -->
    <SpremembaStanjaKA rdf:about="#prekini"/>
    <!-- http://www.owl-ontologies.com/TOKP.owl#prekinjen -->
    <Stanje rdf:about="#prekinjen"/>
    <!-- http://www.owl-ontologies.com/TOKP.owl#pripravljen -->
    <Stanje rdf:about="#pripravljen"/>
    <!-- http://www.owl-ontologies.com/TOKP.owl#protokol -->
    <Protokol rdf:about="#protokol"/>
    <!-- http://www.owl-ontologies.com/TOKP.owl#vTeku -->
    <Stanje rdf:about="#vTeku"/>
    <!-- http://www.owl-ontologies.com/TOKP.owl#zacni -->
    <SpremembaStanjaKA rdf:about="#zacni"/>
</rdf:RDF>
```