UNIVERZA V LJUBLJANI
FAKULTETA ZA RAČUNALNIŠTVO IN INFORMATIKO

Boštjan Senica

PRIMERJAVA METODOLOGIJ IN ORODIJ ZA RAZVOJ IN UPORABO ONTOLOGIJ

DIPLOMSKO DELO NA UNIVERZITETNEM ŠTUDIJU

Mentor: izr. prof. dr. Marjan Krisper

Ljubljana, 2008

ZAHVALA

Za pomoč in nasvete pri pisanju diplomske naloge se zahvaljujem izr. prof. dr. Marjanu Krisperju in asistentu Dejanu Lavbiču ter vsem ostalim, ki so kakorkoli pomagali pri izdelavi tega dela.
Ob dokončanju univerzitetnega študija se zahvaljujem družini za podporo in potrpežljivost v času študija.

KAZALO:

KAZALO SLIK

KAZALO TABEL

UPORABLJENE KRATICE

- BIRT (Business Intelligence Reporting Tool) - orodje za kreiranje poročil na področju poslovnega obveščanja
- DBMS (Database Management System) - sistem za upravljanje podatkovnih baz
- HTML (HyperText Markup Language) - jezik za označevanje nadbesedila
- IEEE (Institute of Electrical and Electronics Engineers) - inštitut inženirjev elektrotehnike in elektronike
- ISBN (International Standard Book Number) - mednarodni standard za številčenje knjig
- MDA (Model Driven Architecture) - metodologija za razvoj programske opreme, kjer se funkcionalnost sistema določi v modelu, ki je neodvisen od platforme
- SOA (Service Oriented Architecture) - storitveno usmerjena arhitektura
- SQL (Structured Query Language) - strukturirani povpraševalni jezik
- UI - umetna inteligenca
- UML (Unified Modeling Language) - standardni jezik za modeliranje na področju razvoja programske opreme
- URI (Uniform Resource Identifier) - standard za identifikacijo in lociranje virov
- W3C (World Wide Web Consortium) - konzorcij, ki razvija tehnologije za svetovni splet

POVZETEK

Namen diplomske naloge je primerjava obstoječih pristopov, metodologij in podpornih orodij za razvoj in uporabo ontologij. Ontologije se na področju informacijske tehnologije uporabljajo za povečanje souporabe in ponovne rabe znanja. V začetku dela je opisano področje ontologij, predstavljena povezava razvoja ontologij z upravljanjem znanja ter prikazana ideja in cilji semantičnega spleta. Opisani so tudi jeziki za predstavitev ontologij in njihov kronološki razvoj.
V nadaljevanju so natančneje opisane metodologije razvoja ontologij, njihovi pristopi in razvojni procesi, ki jih vključujejo. Opredeljeno je ogrodje s kriteriji za ocenjevanje metodologij, na podlagi katerih so zbrani podatki o metodologijah. Kriteriji, ki se nanašajo na procese razvoja ontologije, so povzeti po IEEE 1074-1995 standardu za proces razvoja programske opreme. Primerjava je opravljena s pomočjo programa DEXi, ki omogoča primerjavo variant s pomočjo drevesa kriterijev.
Podobno, kot metodologije, so opisana in predstavljena tudi nekatera orodja za razvoj ontologij. Opredeljen je seznam kriterijev za primerjavo orodij, zbrani so podatki in opravljena primerjava s programom DEXi.
Na koncu je prikazan praktični primer ontologije v orodjih OntoStudio in Protege, ki sta bila v primerjavi ocenjena kot najbolje razvita. V prilogah je dodano poročilo primerjave metodologij in orodij za razvoj ontologij iz programa DEXi.

Ključne besede: ontologija, metodologije za razvoj ontologij, orodja za razvoj ontologij, upravljanje znanja

ABSTRACT

The purpose of the following thesis is to compare existing approaches, methodologies and tools for ontology development as well as for its usage. Ontologies were exploited in computer science to enhance knowledge sharing and reuse. At the beginning of this thesis onotologies are described, the connection between ontology development and knowledge management is presented, and the vision of the Semantic Web is introduced. In addition ontology languages and their chronological development are described.
The thesis continues with description of methodologies for building ontologies, their approaches and included development processes. Evaluation framework with criteria which are used to compare methodologies is defined and data based on this criteria is collected. The criteria referring to ontology development processes are based on IEEE 1074-1995 standard for software development process. The comparison of methodologies is performed in DEXi program which enables comparison of variants based on evaluation framework.
Similar to methodologies some tools for building ontologies are described. Furthermore the evaluation framework with the list of criteria for tool comparision is established, as well as the data about the tools is collected. The comparision is made with DEXi program.
The thesis concludes with an example of ontology implemented within best evaluated tools regarding to described criteria - OntoStudio and Protégé. Two reports from DEXi describing the results of methodologies and tools comparision are included in attachments.

Key words: ontology, methodologies for building ontologies, ontology development tools, knowledge management

1 UVOD

Naša družba se iz industrijske vse bolj sprevrača v družbo, kjer največ pomeni znanje. Znanje je postalo ključen ekonomski resurs, celo pomembnejši od kapitala, naravnih bogastev in dela. Ta, tako imenovana »postindustrijska revolucija«, znanje obravnava kot »intelektualni kapital«. Zato bi morala podjetja kazati enak interes za upravljanje znanja kakor za upravljanje kapitalskih naložb ali delovnih razmerij. Upravljanje znanja se je tako pojavilo kot pomembna strategija za soočanje z novimi izzivi [33].
Vodila za pristop k upravljanju znanja so bila zmanjšanje stroškov ob hkratnem povečanju uspehov, zmanjšanje izgub znanja ob odhodih ključnih ljudi iz podjetij in skrajšanje časa pri iskanju ustreznih odgovorov.
Glavni cilj upravljanja znanja je omogočiti čim bolj učinkovito širjenje oziroma deljenje skupnega znanja, in sicer ne le med ljudmi, temveč tudi med ljudmi in tehnologijo ter med tehnologijo samo. To je mogoče, na primer, s pomočjo programskih agentov, ki komunicirajo z ljudmi ali pa med seboj. Za vse to pa agenti, človeški in programski, potrebujejo nek skupen jezik za sporazumevanje.
Ontologije se uporabljajo v informacijski tehnologiji za povečanje širitve in ponovne uporabe znanja. Kot prvo zagotavljajo skupno in dogovorjeno razumevanje znanja nekega domenskega področja. Po drugi strani pa zajemajo in formalizirajo znanje s povezovanjem človeškega razumevanja simbolov z zmožnostjo njihovega računalniškega procesiranja. Uporaba ontologij za podporo upravljanju znanja prinaša mnogo koristi. Ontologija je tako slovar, ki se uporablja za opis in predstavitev določenega področja, hkrati pa vključuje tudi pomen tega slovarja. V strokovni in znanstveni literaturi ima izraz ontologija številne pomene: od taksonomije (znanje z minimalno hierarhično strukturo), do slovarja (besede in sinonimi), tematskih področij (podpora organizaciji in navigaciji po večji količini podatkov), konceptualnega modela (bolj kompleksno znanje) in teorije logike (zelo bogato, kompleksno, konsistentno znanje).
Pojav pojma ontologija na področju informacijske tehnologije se je začel v začetku zadnjega desetletja prejšnjega stoletja. Sprva sta se razvoj in uporaba ontologij odvijala predvsem na znanstvenem področju (na primer: umetna inteligenca), pozneje pa se je njihova uporaba počasi širila tudi na druga področja. Tako se ontologije že uporabljajo na področjih obvladovanja znanja, pridobivanja in integracije informacij, digitalnih knjižnic, inteligentnih programskih agentov, načrtovanja poslovnih procesov in v povezavi s poslovnimi aplikacijami. Na svetovnem spletu se ontologije uporabljajo v povezavi s spletnimi katalogi (Yahoo!), inteligentnimi iskalniki, za kategorizacijo produktov in njihovih lastnosti pri spletni prodaji (Amazon.com) itd. Nekatere stroke pa že razvijajo tudi standardizirane ontologije, ki jih lahko nato domenski strokovnjaki uporabljajo za skupno rabo in razlago informacij na svojem področju.
V tem času se je pojavilo mnogo različnih pristopov in metodologij za razvoj ontologij, ki skušajo opredeliti aktivnosti razvoja ontologij, podporne aktivnosti razvoja in tudi aktivnosti za obvladovanje ontologije. Obenem pa je bilo razvitih tudi kar nekaj spletnih in samostojnih orodij za razvoj ontologij, ki ponujajo različne nabore funkcionalnosti.
Namen tega dela je predstaviti in primerjati metodologije in orodja za razvoj in uporabo ontologij. Opredeljena bodo merila oziroma seznam kriterijev, ki bodo odražali prednosti ali slabosti opisanih metodologij in orodij. Preverilo se bo, v kolikšni meri so metodologije in orodja na področju razvoja ontologij že dozoreli za širšo rabo v praktičnih primerih.

2 ZGODOVINA

Pojem ontologija izhaja s področja filozofije, ki je povezano s študijem o obstoju. V filozofiji se ontologija obravnava kot veja metafizike, ki se ukvarja z naravo obstoja in obravnava nespremenljive značilnosti sveta. V informacijski tehnologiji pa je ontologija tehnični izraz, ki je bil vpeljan pri nameri modeliranja znanja o določeni domeni ali področju [20].
Pojem ontologije so si prvi prisvojili raziskovalci s področja umetne inteligence, ki so spoznavali uporabnost del iz matematične logike ter dokazovali, da lahko ustvarijo nove ontologije kot računske modele, ki omogočajo nekakšno avtomatsko sklepanje. V osemdesetih letih so na področju umetne inteligence uporabljali izraz ontologija tako za teorijo modeliranja sveta kakor tudi kot del sistemov za upravljanje z znanjem. Nekateri raziskovalci, ki so se sklicevali na ontologije s filozofskega področja, so takšne bolj računske in logične ontologije videli kot nekakšno uporabno filozofijo. Vendar pa ontologije, za razliko od UI sistemov, kjer je veliko špekulativnega sklepanja, ne vsebujejo nobene slabo definirane logike, logike na podlagi verjetnosti ali kakršnega koli sklepanja, ki skuša simulirati človeško zavest.
V začetku devetdesetih let je iz prizadevanj za nastanek splošno uporabnih standardov nastala tehnologija, imenovana plast ontologije, ki je postala standardna komponenta sistemov za upravljanje znanja. Nastalo je veliko dokumentov in spletnih strani z definicijami ontologije kot tehničnega pojma v informacijski tehnologiji. Najbolj znano definicijo je predstavil Tom Gruber, ki ontologijo opredeljuje kot »eksplicitna specifikacija konceptualizacije«. To je v bistvu pomenilo objekte, koncepte in ostale entitete, ki obstajajo na določenem interesnem področju, ter relacije, ki obstajajo med njimi. Čeprav sta izraza specifikacija in konceptualizacija sprožila mnogo razprav o primernosti te definicije, pa je njeno bistvo naslednje:

- ontologija definira (specificira) koncepte in relacije, ki so ustrezne za modeliranje domene,
- specifikacija predstavlja obliko reprezentativnega slovarja (razredov, relacij itd.), ki določa pomen slovarju in formalne omejitve za razumljivo in jasno uporabo [31].

Eden izmed pomislekov pri tej definiciji je, da je preveč splošna in lahko zajema vse od specifikacij preprostih slovarjev pa do logičnih teorij v predikatnih računih. Zato so pozneje to definicijo malce spremenili in razširili v »eksplicitno formalno specifikacijo skupne konceptualizacije«. Izrazi v definiciji pomenijo naslednje:

- »eksplicitno« pomeni, da so tipi uporabljenih konceptov in omejitev pri njihovi rabi jasno in nedvoumno določeni,
- »formalno« se nanaša na dejstvo, da mora biti ontologija razumljiva tudi programskim agentom,
- »skupno« se nanaša na sporazumno znanje, zajeto v ontologiji, ki ne pripada posamezniku, ampak je sprejeto v neki skupini,
- »konceptualizacija« se nanaša na abstrakten model nekega pojava z identificiranjem ustreznih konceptov [9].

Po drugi strani pa obstajajo tudi bolj neformalne definicije ontologije, kot na primer »slovar izrazov in specifikacija njihovih pomenov«.
Te definicije se razlikujejo predvsem v določeni stopnji formaliziranosti in sporazumnega znanja, ki ga ontologija predstavlja. Pomembno je, da je znanje, zajeto v ontologiji, sporazumno potrjeno vsaj s strani določene skupine strokovnjakov, saj je s tem omogočena ponovna uporaba v različnih sistemih za upravljanje z znanjem. Navsezadnje je to glavni

razlog razvijanja ontologij. Po drugi strani pa stopnja formaliziranosti ni vedno splošno določena. Obstajajo namreč tako strogo formalizirane ontologije kakor tudi ontologije, ki so ohlapno izražene z naravnim jezikom [9].

3 KAJ SO ONTOLOGIJE?

Ontologije so namenjene reševanju problemov, ki nastajajo pri uporabi različne terminologije za enake pojme ali pa pri uporabi enakih izrazov za različne pojme. Čeprav izraz ontologija izvira iz filozofije, ima pri rabi v informacijski tehnologiji oziroma na področju umetne inteligence drugačen pomen. Medtem ko se v filozofiji pojem ontologije uporablja v zvezi z naravo obstoja, pa ga v informacijski tehnologiji in na področju umetne inteligence predstavljajo predvsem z Gruberjevo definicijo kot eksplicitna specifikacija konceptualizacije. Ontologija definira skupen slovar pojmov med različnimi sistemi. Poskuša identificirati in premostiti prepreke pri skupni rabi in ponovni uporabi znanja, uporabljenega v programih, ki nastanejo zaradi pomanjkanja soglasij v izrazoslovju in različnih pomenskih interpretacijah modelov na področju neke domene. Neformalno je ontologija sestavljena iz zbirke pojmov in zbirke omejitev, ki so določene tako, da se lahko ti pojmi združujejo. Omejitve tudi omejujejo različne pomenske interpretacije pojmov. Pojmi v ontologijah pomenijo predstavitev nekega koncepta oziroma predstave. Tako je potrebno poudariti, da so v ontologijah predstavljeni koncepti in ne besede. Koncepti, v splošnem, tudi niso specifični posameznim jezikom.

Ontologije so tesno povezane z bazami znanja. Razlika med njimi leži v različnih vlogah, ki jih zavzemajo pri predstavljenem znanju. Ontologije skušajo predstaviti znanje, ki je v večini sporazumno določeno s strani neke skupine ljudi oziroma strokovnjakov dotičnega področja, medtem ko je v bazah znanja znanje, specifično za določen problem, ki ga ta baza rešuje. Ontologije se ukvarjajo s statičnim domenskim znanjem. Tako se znanje v ontologijah malokrat spreminja za razliko od znanja v bazah znanja. Na primer modeliranje ontologije nekega podjetja vključuje koncepte, kot so aktivnost, proces, resurs, medtem ko v bazah znanja nastopajo točno določene aktivnosti, procesi ali resursi, specifični za določeno podjetje. Zato je znanje, zajeto v ontologijah, veliko bolj primerno za ponovno uporabo in delitev med aplikacijami.

Čeprav so ontologije namenjene zajemanju statičnega znanja, so odvisne tudi od aplikacij, ki jih uporabljajo oziroma zahtevajo njihov razvoj. Če dve aplikaciji uporabljata znanje iz iste domene, vendar so njune operacije različne, potem je normalno, da potrebujeta ontologije, ki se med seboj razlikujejo. Čeprav je večina konceptov po navadi skupnih, pa so lahko definirani na različne načine, recimo z različno stopnjo podrobnosti, zajemajo različne poglede ali lastnosti istega koncepta. Različni pogledi oziroma stališča lahko tudi posledično pomenijo, da so enaki koncepti predstavljeni z različno terminologijo. Treba je poudariti, da ne obstaja samo en način snovanja konceptov, pač pa več možnih alternativ.

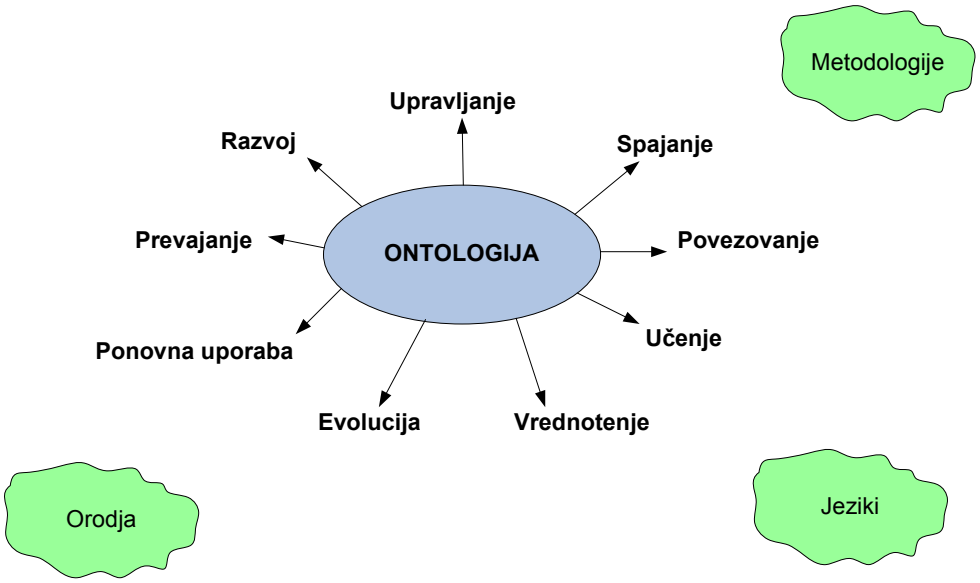
Ontologije igrajo pomembno vlogo pri komunikaciji med inteligentnimi sistemi. Recimo, da neka aplikacija potrebuje drugo za opravljanje določene operacije. Pošiljanje informacij o problemu, ki ga želi rešiti, mora biti opravljeno na takšen način, da jih druga aplikacija razume. Zato je potrebno prevesti informacije iz različnih ontologij znotraj iste domene.

Kompatibilnost med ontologijami pa ni pomembna le pri komunikaciji med aplikacijami, pač pa tudi pri gradnji večjih sistemov iz manjših. Sistemi, bazirani na znanju, imajo kot eno izmed komponent tudi ontologijo. Bazo znanja lahko sestavimo iz manjših samo v primeru, da so ontologije, na katerih so osnovane, med seboj kompatibilne in konsistentne. Če te ontologije niso enake, pomeni, da uporabljajo različne pojme, ali pa aksiomi, ki predstavljajo

omejitve, med seboj niso skladni. Zato ni mogoče zgraditi velikega sistema z vidika ponovne uporabe, če ni razumevanja med slovarji ontologij ali pa, če aksiomi med seboj niso skladni [9, 32].

3.1 Procesi, povezani z ontologijami

Z ontologijami je povezanih mnogo procesov, ki so prikazani na sliki 1. Osnova tem procesom pa je razvoj metodologij, jezikov in programskih orodij.



Slika 1: Procesi, povezani z ontologijami [8]

V tej diplomski nalogi bodo na kratko predstavljeni jeziki za formalen zapis ontologij, natančneje pa metodologije in orodja za razvoj ontologij.

3.2 Uporaba ontologij

Ontologije so del paketa standardov W3C za semantični splet, v katerem se uporabljajo za specifikacijo standardnih konceptualnih slovarjev, v katerih se izmenjujejo podatki med sistemi, zagotavljajo storitve za reševanje poizvedb, objavljajo baze znanja za ponovno uporabo in ponujajo storitve za omogočanje interoperabilnosti med različnimi sistemi ali podatkovnimi bazami. Glavni pomen ontologij je oblikovati predstavitev podatkovnega modeliranja na bolj abstraktnem nivoju, višjem od specifičnih modelov podatkovnih baz (logičnih ali fizičnih). Tako se lahko nad podatki izvajajo poizvedbe, lahko se izvažajo, prevajajo in združujejo med neodvisno razvitimi sistemi in storitvami. Uspešne aplikacije danes omogočajo interoperabilnost podatkovnih baz, iskanje med različnimi podatkovnimi bazami in integracijo spletnih storitev [31].

3.3 Namen ontologij

Razlogi za razvoj ontologij so:

- širitev splošnega razumevanja strukture informacij med ljudmi ali programskimi agenti,
- omogočanje ponovne uporabe znanja določene domene,
- določitev in razjasnitev predpostavk o določeni domeni,
- ločitev domenskega znanja od operativnega,
- analiza domenskega znanja.

Širitev splošnega razumevanja strukture informacij med ljudmi ali programskimi agenti je eden najpomembnejših skupnih ciljev pri razvoju ontologij. Predpostavimo na primer, da imamo več različnih spletnih strani, ki vsebujejo informacije s področja medicine ali pa izvajajo medicinske e-storitve. Če si te strani delijo in objavljajo isto ontologijo izrazoslovja, ki ga uporabljajo, lahko programski agenti povzemajo in združujejo podatke iz teh različnih spletnih strani. Te podatke lahko nato agenti uporabljajo kot odgovore na uporabniške poizvedbe ali kot vhodne podatke za druge aplikacije.
Omogočanje ponovne uporabe znanja določene domene je bilo eno izmed glavnih vodil v začetkih raziskovanja ontologij. Tako lahko na primer neko ontologijo, ki jo je že do potankosti razvila skupina strokovnjakov, drugi preprosto ponovno uporabijo v sklopu svoje domene. Poleg tega lahko pri razvijanju večjih ontologij vključimo več obstoječih ontologij, ki predstavljajo posamezne dele velike ontologije. Mogoče pa je uporabiti tudi neko splošno ontologijo ter jo nato prilagoditi na specifično interesno področje.
Z določitvijo predpostavk znotraj domenskega področja ontologije ločeno od implementacije, lahko ob spremembah znanja o domeni le te preprosto spremenimo. Kodiranje takšnih predpostavk v programski kodi povzroča njihovo težje razumevanje in iskanje, hkrati pa jih je tudi težje spreminjati. Poleg tega pa ločena specifikacija domenskega znanja novim uporabnikom omogoča lažje razumevanje pojmov.
Ločitev domenskega znanja od operativnega je še ena splošna uporabnost ontologij. Tako lahko recimo opišemo oblikovanje produkta iz komponent glede na določen načrt in na podlagi tega implementiramo program, ki to nalogo opravlja, vse skupaj pa je neodvisno od specifikacije produkta in komponent. Če nato razvijemo ontologijo računalniških komponent in karakteristik, lahko uporabimo omenjeni algoritem za izdelavo poljubnih računalnikov. Isti algoritem pa lahko nato uporabimo recimo za izdelovanje prenosnih telefonov, če zraven uporabimo ontologijo s komponentami prenosnih telefonov.
Analiza domenskega znanja se lahko opravlja potem, ko je narejena specifikacija pojmov oziroma terminov. Analiza je zelo koristna tako pri ponovni uporabi znanja, kakor tudi pri razširitvi znanja oziroma ontologij.
Pogosto pa izdelane domenske ontologije niso končni cilj. Razvoj ontologij je povezan z zbirko podatkov in njihovega strukturiranja, ki jih nato uporabljajo računalniški programi. Metode za reševanje problemov, domensko neodvisne aplikacije in programski agenti uporabljajo ontologije ter baze znanja, zgrajene iz ontologij. Za primer vzemimo ontologijo vin ter hrane in primernih kombinacij vrste vina k določenim obrokom. Takšna ontologija je lahko v pomoč več aplikacijam pri upravljanju z restavracijami. Ena aplikacija lahko, recimo, predlaga primerno vino k dnevnim menijem ali pa pomaga natakarjem in gostom pri njihovih poizvedbah, druga pa lahko denimo analizira zaloge vinske kleti ter predlaga, katere vrste vina bi bilo potrebno kupiti glede na prihajajočo ponudbo hrane [16].

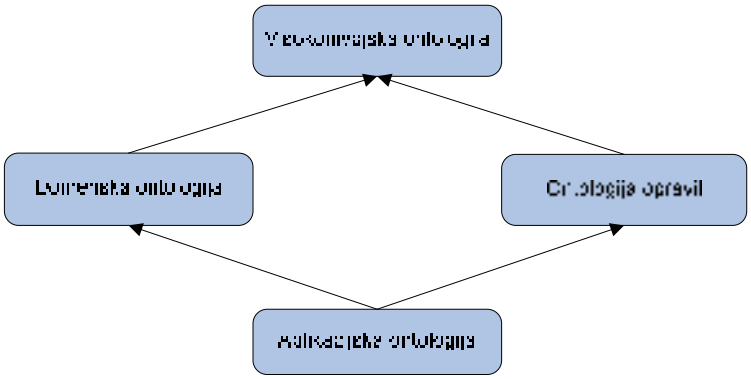
3.4 Komponente ontologije

Na področju računalništva in informacijske tehnologije predstavljajo ontologije zbirko konceptov znotraj neke domene in povezave med temi koncepti. Uporabljajo se za sklepanje o lastnostih domene in tudi za definiranje domene.
Tipične komponente ontologije so:

- **Primerki:** osnovni objekti oziroma instance ontologije;
- **Razredi:** zbirke, koncepti ali tipi objektov. Vsebujejo lahko primerke ali druge razrede;
- **Atributi:** lastnosti, značilnosti, karakteristike ali parametri, ki jih lahko imajo objekti in razredi;
- **Relacije:** načini povezav med objekti oziroma med razredi;
- **Funkcijski izrazi:** kompleksne strukture, formirane iz določenih relacij;
- **Omejitve:** formalno podani opisi, ki morajo biti resnični, če želimo sprejeti nekatere izjave;
- **Pravila:** trditve tipa »če-potem« stavkov, ki opisujejo logične sklepe, izpeljane iz določenih trditev v modelu;
- **Aksiomi:** trditve ali pravila v logičnem modelu, ki skupaj zajemajo celotno teorijo, ki jo ontologija opisuje v določeni domeni;
- **Dogodki:** spremembe atributov ali relacij [19].

3.5 Klasifikacija ontologij

Obstajajo različne klasifikacije ontologij. Klasifikacijo, ki za glavni kriterij uporablja konceptualizacijo, je predstavil Guarino. Predlagal je razvoj različnih vrst ontologij glede na nivo posplošenosti.



Slika 2: Klasifikacija ontologij [33]

Kot kaže slika 2, obstajajo naslednje vrste ontologij:

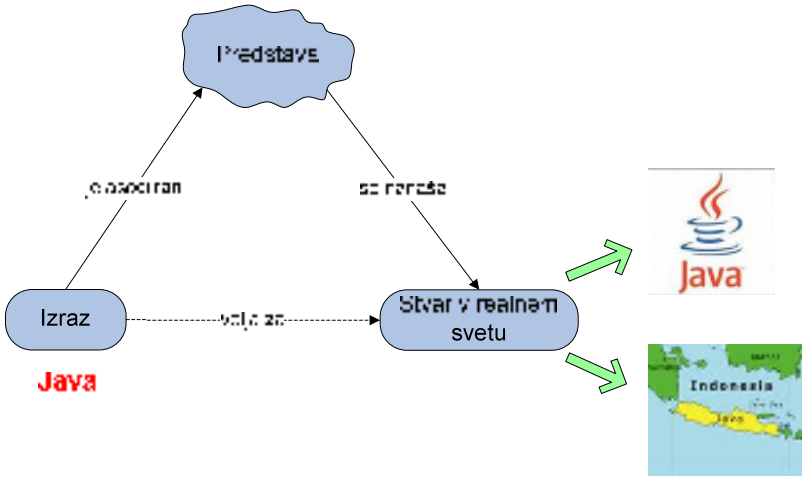
- **Visokonivojske ontologije** opisujejo zelo posplošene koncepte, kot so prostor, čas, dogodek, ki so neodvisni od določenih problemov ali domen. Takšne univerzalne visokonivojske ontologije služijo veliki množici ljudi in aplikacij. Olajšajo namreč (pol)avtomatično integracijo ali kombinacijo različnih ontologij, ki so povezane z isto visokonivojsko ontologijo. Obstajajo že tudi standardizirane višje ontologije za splošno rabo, kot so Dublin Core, OpenCyc, SUMO ali DOLCE.

- **Domenske ontologije** opisujejo slovar, povezan z neko specifično domeno (avtomobili, vina). Razvijejo se s specializacijo konceptov, predstavljenih pri visokonivojskih ontologijah.
- **Ontologije opravil** opisujejo slovar, povezan z nekim splošnim opravilom (prodaja, pitje). Tudi te ontologije se razvijejo s specializacijo konceptov, predstavljenih pri visokonivojskih ontologijah.
- **Aplikacijske ontologije** so najbolj specifične ontologije. Koncepti v aplikacijskih ontologijah se po navadi skladajo z vlogami v domenskih ontologijah ob izvrševanju neke določene aktivnosti. Tako so aplikacijske ontologije specializacija domenskih ontologij in ontologij opravil. Predstavljajo osnovo za implementacijo aplikacij z določeno domeno in področjem delovanja [33].

Nižjenivojske ontologije (domenske, aplikacijske, ontologije opravil) predstavljajo bolj specifične koncepte, zato so med seboj pogosto nezdružljive. Ko se sistemi, ki so osnovani na teh ontologijah, razširijo, je potrebno ontologije velikokrat združiti v bolj splošne predstave, kar predstavlja velik izziv snovalcem ontologij. V isti domeni ontologij pa se lahko pojavljajo tudi različne ontologije zaradi različnega dojemanja domene na podlagi kulturnega ozadja, izobraženosti, ideologije ali pa zaradi uporabljenega drugačnega predstavitvenega jezika. Obstajajo že študije o splošnih tehnikah za združevanje ontologij, ki pa so trenutno še bolj na teoretičnem nivoju.

3.6 Komunikacija z ontologijami

Za predstavitev interakcije med simboli ali besedami, predstavami in stvarmi iz realnega sveta se uporablja tako imenovani »pomenski trikotnik«. Ta ilustrira dejstvo, da obstaja povezava med besedami, predstavami in stvarmi, čeprav besede ne morejo v celoti zajeti bistva predstav ali stvari iz realnega sveta. Zveza med besedami in stvarmi je posredna. Besede tako dobijo pomen šele, ko si razlagalec ob besedi ustvari predstavo ter s tem poveže to besedo s stvarjo v realnem svetu.



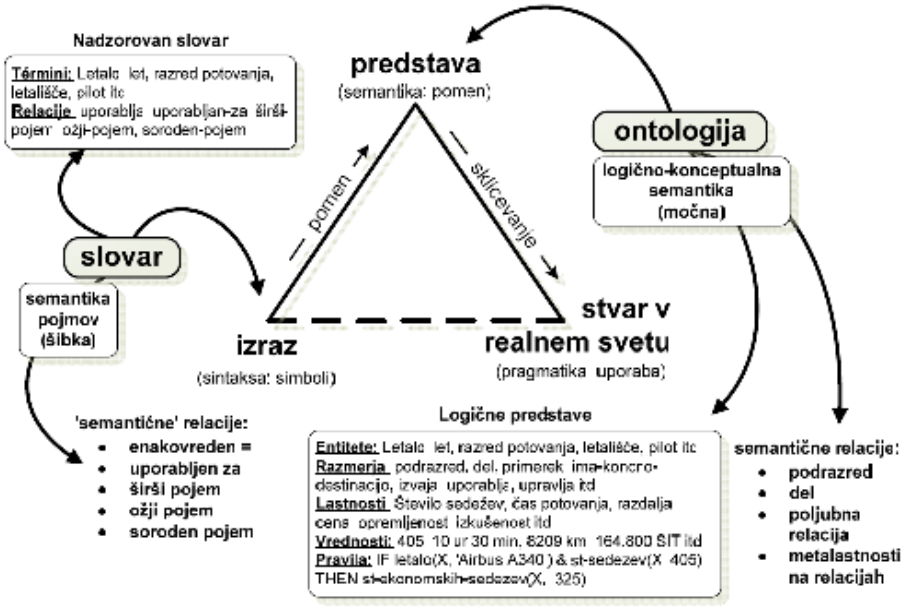
Slika 3: Pomenski trikotnik

Na sliki 3 so prikazane vse tri komponente pomenskega trikotnika. Ob izrazih si lahko ljudje ustvarimo povsem različne predstave za stvari iz realnega sveta, na primer ob besedi »Java« lahko nekateri pomislijo na programski jezik, drugi pa na del Indonezije. Povezava med

izrazom in stvarjo v realnem svetu je prikazana s črtkano črto, saj za njuno povezavo človek potrebuje predstavo.
Podobno pa velja tudi za programske agente, katerih znanje temelji na specifičnih domenskih ontologijah. Ti prav tako, kot ljudje v naravnem jeziku, izmenjujejo sporočila, osnovana na nekem komunikacijskem protokolu. Programski agent, ki za svojo semantično osnovo uporablja ontologijo s področja geografije, se bo tako ob izrazu »Java« nanašal na povsem drugo stvar kot agent z ontologijo s področja programskih jezikov.
Ljudje in programski agenti uporabljajo svoje koncepte v procesu sklepanja z namenom zmanjšanja izbire možnih asociacij [33].

3.6.1 Razlika med slovarjem in ontologijo

Kot prikazuje slika 4, se slovar v splošnem ukvarja z levo stranjo trikotnika (izrazi in predstave), medtem ko je ontologija večinoma osredotočena na desno stran (predstave in stvari v realnem svetu).



Slika 4: Vloga slovarja in ontologije v trikotniku pomena [4]

Slovar je v prvi vrsti klasifikacijski prostor v določeni domeni ali množici domen ali celo v širšem svetu, predvsem za potrebe iskanja in pridobivanja informacij. Semantika klasifikacijskega prostora je tako relativno šibka, s preprostimi semantičnimi relacijami med izrazi. Ti so strukturirani s taksonomijo relacij. Vse kar moramo poznati o izrazu v slovarju je, da se semantično razlikuje od ostalih (s čimer izločimo dvoumnost) in da je širši oziroma ožji pojem kot drugi.
Na drugi strani želi ontologija predstaviti zapleteno semantiko predstav in relacij med predstavami, njihovimi lastnostmi, vrednostmi, omejitvami in pravili. Namen ontologije se razlikuje od namena slovarja, saj ontologija skuša zajeti in predstaviti pomen določenih domen, množice domen ali celotnega sveta, ker skuša eksplicitno stimulirati razumevanje, ki ga ima človek v svojih mentalnih modelih problemske domene ali sveta. V nasprotju s slovarjem želimo pri ontologijah konceptualno semantiko izraziti na čim bolj natančen, obsežen in konsistenten način [4].

3.7 Podatkovno modeliranje in ontologije

V nasprotju s podatkovnim modeliranjem je glavna prednost ontologij v neodvisnosti od programskih rešitev, saj je ontologija sestavljena iz sorazmerno splošnega znanja, ki ga lahko uporabljamo pri različnih opravilih in v različnih programih. Korak bližje konceptu ontologij je globalni konceptualni podatkovni model, ki je tudi neodvisen od programskih rešitev.
Računalniška ontologija predstavlja dogovor o konceptualizaciji ter vsebuje slovar (terminov in oznak) in opredelitve konceptov ter njihove medsebojne odvisnosti v določeni domeni. Podatkovni model, v nasprotju z ontologijo, predstavlja strukturo in integriteto podatkovnih elementov praviloma v določenem programu, ki ta podatkovni model uporablja. Konceptualizacija in uporaba slovarja podatkovnega modela tako ni nujno namenjena za souporabo v več različnih programih.
Kot primer omenimo ontologijo knjigarne, kjer na primer obstaja pravilo, ki pravi, da je vsaka knjiga enolično določena s številko ISBN. Vsi programi, ki uporabljajo to interpretacijo ontologije knjigarne, morajo upoštevati zgornje pravilo. Programi za podporo dela v knjigarni, ki ne predvidevajo številke ISBN za vsako knjigo, ne morejo uporabiti oziroma souporabiti ontologijo knjigarne. Brez tako definirane ontologije dva programa med seboj ne moreta komunicirati (ni souporabe slovarja in pravil, ki veljajo za določeno domeno, med dvema programoma) [4, 25].

3.8 Modeliranje podatkovnih shem in ontoloških modelov

Podatkovni modeli, kot so podatkovne baze ali sheme XML, določajo strukturo in integriteto podatkov. Izdelava podatkovnih modelov je v podjetju po navadi odvisna od določenih zahtev in opravil, ki morajo biti v podjetju izvedena. Semantiko podatkovnih modelov v večini primerov določajo neformalni dogovori med razvijalci in uporabniki podatkovnega modela. Najdemo jo zakodirano v samih programih, ki ta podatkovni model uporabljajo.
V nasprotju s podatkovnimi modeli, ki so odvisni od opravil in usmerjeni k implementaciji, so ontologije že v osnovi in po opredelitvi čim bolj splošne in kar se le da neodvisne od opravil. Bolj kot se ontologija približa idealu formalnega in dogovorjenega zapisa, večja je raven souporabe.
V nadaljevanju bom predstavil, kako (formalno izražena) pravila določene domene vplivajo na splošnost modeliranega znanja. Spodnji seznam predstavlja elemente, kjer se ontologije razlikujejo od podatkovnih modelov:

- **Nivo delovanja**
  Pravila problemske domene lahko zapišemo na nižji ravni, usmerjeni k implementaciji, z uporabo podatkovnih tipov, primarnih ključev itd. Na višjem nivoju pa lahko vključimo bolj abstraktna pravila, kot so popolnost, togost, identiteta, ki so neodvisna od ravni implementacije. Bolj ko so pravila o problemski domeni abstraktna, bolj so le-ta splošna in ponovno uporabljiva na več mestih.
- **Moč izražanja**
  Namen jezikov za manipulacijo s podatki, kot je SQL, je ohranitev integritete podatkov in uporaba splošnih konstruktov, na primer ključev. V splošnem morajo pravila o domeni izražati ne samo integritete podatkov, ampak tudi konceptualizacijo domene. Tako mora jezik za izražanje pravil o problemskem področju vsebovati konstrukte, s katerimi lahko izrazimo ostale pomenske omejitve, kot je taksonomija ali podpora sklepanju, kar najdemo na primer pri jezikih za predstavitev ontologij (DAML+OIL in OWL).

- **Povezanost z uporabnikom, namenom in ciljem**
  Podatkovni model se zelo dobro prilega določenim ciljem in potrebam uporabnikov, ki program uporabljajo. Jasno je, da na ta račun trpi splošnost konceptualizacije, ki je odvisna od granularnosti procesa modeliranja, odločitev, ali določeno dejstvo modeliramo kot razred ali atribut, uporabo tipa objekta iz slovarja ali ne itd.
- **Razširljivost**
  Pri konceptualizaciji ontologije problemskega področja je pomembno, da elemente obravnavamo ločeno od problema ali opravil in tako omogočimo uporabo in razširitev slovarja v splošnih primerih, ki jih pri gradnji mogoče nismo predvideli [4, 25].

4 UPRAVLJANJE ZNANJA IN RAZVOJ ONTOLOGIJ

V zadnjih letih se je zavedanje o pomembnosti razvoja ontologij razširilo tudi izven meja umetne inteligence, na področja elektronskega poslovanja in informacijskih sistemov. Medtem pa je vloga znanja s strani mnogih raziskovalcev prevzela odločilno vlogo oziroma pridobitev za preživetje in napredek podjetij. Poleg tega je postal razvoj metodologij za povečevanje vpliva organizacijskega znanja glavna skrb discipline, ki ji pravimo upravljanje z znanjem. Ker je primarna osredotočenost tako upravljanja znanja kot gradnje ontologij znanje in načini, kako ga lahko delimo, uporabljamo oziroma ponovno uporabimo, se porajata naslednji vprašanji:

- Kako lahko aktivnosti upravljanja z znanjem podpremo z ontologijami?
- Kaj se lahko metodologije za inženiring ontologij naučijo iz upravljanja z znanjem?

Medtem ko je mnogo strokovnjakov skušalo odgovoriti na prvo vprašanje, pa odgovora na drugo vprašanje še ni bilo. Ob pomanjkanju sporazumnih usmeritev oziroma napotkov ali pa enotnega procesnega modela za metodologije inženiringa ontologij, tako vsaka razvojna ekipa sledi svojim razvojnim principom, fazam in oblikovanju. Takšno stanje ovira razvoj poenotenega razumevanja znotraj in med razvojnimi ekipami, onemogoča razširitev obstoječih ontologij s strani drugih strokovnjakov in ponovno uporabo ontologij za razvoj drugih ontologij ali za končne aplikacije. Izkazalo se je, da je vzrok mnogih problemov, s katerimi se srečujejo razvijalci ontologij, prav pomanjkanje enotnega razvojnega modela. Zato je postal glavni cilj razvoj konceptualnega modela za metodologije inženiringa ontologij na podlagi upravljanja z znanjem [7].

4.1 Inženiring ontologij

Ontologije skušajo, kot je razvidno tudi iz definicije eksplicitne specifikacije konceptualizacije, olajšati komunikacijo med ljudmi in organizacijami ter sodelovanje med sistemi in razvojem sistemov. Kljub zavedanju o pomembnosti ontologij pri razvoju različnih poslovnih ali neposlovnih aplikacij, pa je razvoj ontologij še vedno bolj na nivoju spretnosti, kot pa inženirska aktivnost. Cilj inženiringa ontologij je učinkovita podpora razvoju in uporabi ontologij skozi faze njihovega življenjskega cikla: vrednotenje, validacija, vzdrževanje, razvoj, povezovanje, integracija, delitev in ponovna uporaba. Ker inženiring ontologij še ni dobro razvit, še ne obstajajo splošne metodologije inženiringa ontologij, niti konceptualni modeli za te metodologije [7].

4.2 Upravljanje z znanjem

Upravljanje z znanjem lahko splošno definiramo kot sistematično in organizacijsko opredeljen proces za pridobivanje, organiziranje in komuniciranje tako implicitnega kot tudi eksplicitnega znanja zaposlenih, z namenom omogočanja uporabe znanja drugim za večjo učinkovitost in produktivnost. Jedro upravljanja z znanjem je Nonakin model ustvarjanja organizacijskega znanja. Na podlagi tega modela se znanje kreira skozi nenehno interakcijo med implicitnim in eksplicitnim znanjem. Medtem ko je eksplicitno znanje tisto, ki je lahko predstavljeno ali zapisano s sporočilnimi sredstvi, pa je implicitno znanje osebno znanje, povezano z individualnimi izkušnjami in vključuje tudi osebna prepričanja, poglede in vrednote. Medsebojni vpliv teh dveh tipov znanj je vključen v štiri osnovne procese pretvorbe znanj:

- **Socializacija** (iz implicitnega v implicitno znanje) je proces delitve implicitnega znanja s pomočjo sodelovanja oziroma medsebojnega vplivanja med posamezniki. Rezultat tega procesa je ustvarjanje skupnih pogledov in delitve izkušenj, čemur pravimo tudi simpatizirano znanje.
- **Eksternalizacija** (iz implicitnega znanja v eksplicitno) je proces zajemanja skupnega implicitnega znanja v eksplicitne koncepte, ki jih lahko izražamo na razumljiv način. Rezultat tega procesa je konceptualno znanje.
- **Kombinacija** (iz eksplicitnega v eksplicitno znanje) je proces, pri katerem je znanje, ki je že na voljo, kombinirano ali preoblikovano v novo eksplicitno znanje. Kombinacija vključuje tudi sortiranje, prekategoriziranje, širitev in urejanje novega znanja. Rezultat procesa je sistematično znanje.
- **Internalizacija** (iz eksplicitnega v implicitno znanje) je proces, pri katerem se eksplicitno znanje pretvarja v implicitno. To se izvaja z vključevanjem eksplicitnega znanja v konkretne zadeve, kot so akcije, postopki, prakse, izdelki ali sistemi. Ta proces zato vključuje tudi uporabnost novoustvarjenega znanja in je močno povezan s procesom učenja ob delu. Rezultat te internalizacije je operativno znanje [7].

Kot je prikazano v tabeli 1, je ustvarjanje znanja kontinuiran in dinamičen proces.

		V	
		Implicitno znanje	**Eksplicitno znanje**
Iz	**Implicitno znanje**	Proces: **Socializacija** Rezultat: **Simpatizirano znanje**	Proces: **Eksternalizacija** Rezultat: **Konceptualno znanje**
	Eksplicitno znanje	Proces: **Internalizacija** Rezultat: **Operativno znanje**	Proces: **Kombinacija** Rezultat: **Sistematično znanje**

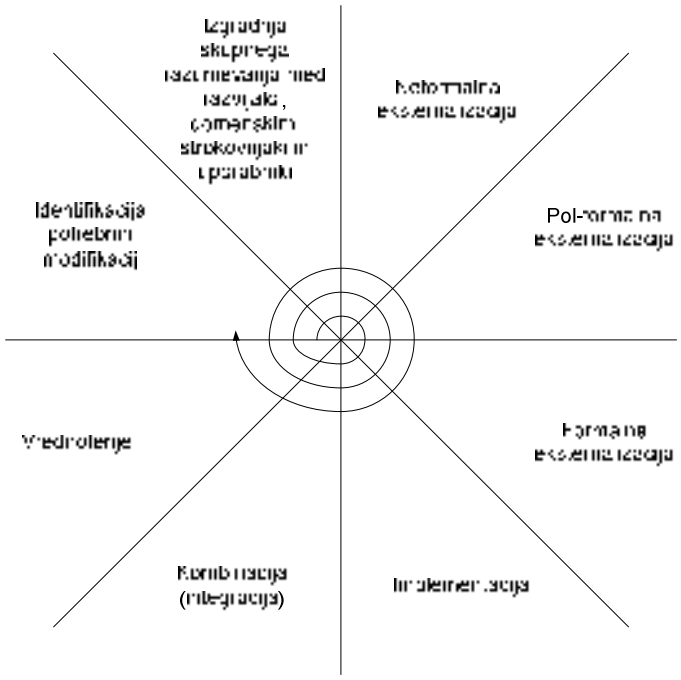
Tabela 1: Pretvorba znanj [7]

4.3 Procesni model na osnovi upravljanja z znanjem za metodologije inženiringa ontologij

V definiciji ontologije eksplicitna specifikacija konceptualizacije je bilo veliko vprašanj glede pomena konceptualizacije. Nekateri so jo obravnavali kot skupno razumevanje določenega področja, ontologijo pa kot eksplicitno posledico tega razumevanja. Poudarek na konceptualizaciji kot skupnem razumevanju izvira iz dejstva, da lahko zaradi različnih potreb obstaja na enem področju mnogo različnih pogledov in predpostavk o tem, kaj je pomembno. Vsakdo uporablja drugačen žargon, vsak lahko uporablja drugačne koncepte strukture in metode. Zato bi moral biti razvoj ontologij skupen dosežek, ki odseva izkušnje in poglede ljudi, ki pri tem razvoju sodelujejo.
Tu pridejo do izraza tudi relacije med izrazom, konceptom in stvarmi v realnem svetu, ki so bile omenjene že v prejšnjih poglavjih. Konceptualizacija se tako tiče bolj konceptov kot pa stvari v realnem svetu. Ker sta koncept in stvar v realnem svetu neodvisna, pa je za njuno povezavo potrebno dodatno znanje. To znanje in večina znanja, ki zadeva domene, je implicitno in obsega mnogo dejavnikov, kot so osebne izkušnje in organizacijska pravila. Poleg tega pa takšno znanje tudi pogosto ni sistematizirano.
Zato se mora razvoj ontologij na osnovi konceptualizacije začeti z ustvarjanjem skupnega razumevanja med domenskimi strokovnjaki, razvijalci in vzdrževalci ontologij ter uporabniki in to nato tudi dokumentirati. Vse te aktivnosti za razvoj ontologij so po svoji naravi socialne, ker vključujejo mnogo skupin ljudi, ki imajo vsaka svoje vidike in organizacijska okolja. Razvoj ontologij je tako politični proces, ki zahteva usklajevanja med udeleženimi skupinami. Glede na vse to in ker predstavljajo ontologije znanje domen, lahko na razvoj ontologij na podlagi konceptualizacije gledamo kot na poseben primer Nonakinega modela ustvarjanja znanja. V nadaljevanju sem opisal vse štiri pretvorbe znanja, ki veljajo za razvoj ontologij.

Socializacija: vsak razvoj ontologij na podlagi konceptualizacije bi se moral začeti z identifikacijo aktivnosti, v katerih so zajeti nameni ontologije in vključeni ljudje. Naslednja aktivnost je ustvariti skupno razumevanje izbrane domene med vpletenimi ljudmi. To vključuje vzpostavitev dogovora glede stvari, ki naj bi bile zajete v predstavitvi domene, in glede pomena domenskih predstav. Obe aktivnosti, identifikacija in ustvarjanje skupnega razumevanja, se nanašata na delitev implicitnega znanja vpletenih ljudi.
Eksternalizacija: po uskladitvi skupnega razumevanja je naslednja aktivnost eksternalizacija oziroma zapis le-tega. Eden glavnih problemov pri ponovni uporabi ali delitvi ontologij je namreč, da osnovne predpostavke niso bile dovolj jasno zapisane in dorečene. Eksternalizacija vključuje tudi vpeljavo ontoloških obvez, s čimer se zagotovi, da se določen model, napisan v določenem jeziku, uporablja samo tako, kot je predvidel razvijalec modela. V povezavi z inženiringom ontologij ločimo tri tipe eksternalizacije. Prva, neformalna eksternalizacija, je predstavitev konceptualizacije v naravnem jeziku. Druga, polformalna eksternalizacija, je predstavitev konceptualizacije v jeziku, neodvisnem od implementacije, kot je na primer UML. Tretja, formalna eksternalizacija, pa je proces za zapis konceptualizacije s formalnim jezikom. Pri tem je za validacijo ontologije na nivoju znanja pomembno, da že vmesne predstavitve preverijo tudi domenski strokovnjaki in končni uporabniki.
Kombinacija: v kontekstu razvoja ontologij se kombinacija nanaša na proces, pri katerem so deli obstoječe ontologije ponovno uporabljeni pri snovanju nove ontologije ali pa pri združevanju več ontologij v eno samo.
Internalizacija: ta proces skuša zapisati ontologijo v računalniški jezik. Vendar pa formalizacija znanja za računalniško podprte sisteme ni dovolj. Takšni sistemi lahko izkoriščajo koncepte samo v povezavi z operacijami in pravili, povezanimi z njimi, ki jih je

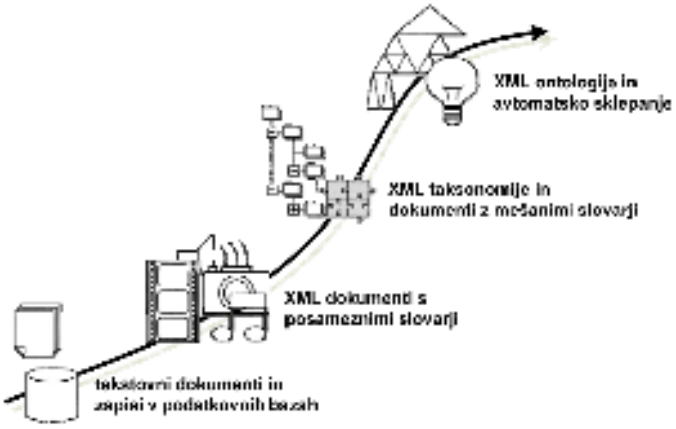
tudi potrebno določiti. Ta pravila vključujejo semantiko, ki skupaj s formalnimi koncepti ontologije tvori celotno ontologijo za računalniški sistem. Poleg tega mora biti takšna ontologija tehnično preverjena in ocenjena pravilnost njene uporabe. Glede na rezultate lahko sledi nov cikel procesov pretvorbe znanja za vzdrževanje nove ontologije. Če postanejo ontologije kritične za uspeh več poslovnih aplikacij, je potrebno oceniti tudi njihovo poslovno vrednost [7].



Slika 5: Visokonivojski cikel procesnega modela inženiringa ontologij [7]

5 SEMANTIKA PODATKOV

Danes se soočamo z vprašanjem, kako zgraditi splet podatkov, ki bi ga lahko računalniki samodejno obdelovali. Najprej moramo spremeniti pogled na podatke. V preteklosti so bili podatki shranjeni in zaklenjeni v zaščitenih programih ter pri sami obdelavi niso igrali glavne vloge. Takšni podatki imajo veliko pomanjkljivost v odvisnosti od njihove obdelave oziroma v odvisnosti programske opreme od dobrih podatkov. Ker so nekateri spoznali, da takšen pristop ni pravilen, so začeli razmišljati v drugo smer. Velik uspeh je doživel internet in z njim XML ter v zadnjih časih zelo priljubljen semantični splet, kjer se vedno bolj osredotočamo na podatke in ne na programe. Ključni element pri samodejni računalniški obdelavi podatkov pa je priprava »pametnejših« podatkov.
Slika 6 prikazuje stopnjo inteligence, ki so jo podatki pridobivali skozi čas. Začnemo s takšnimi, ki za nadaljnje sklepanje vsebujejo zelo malo semantičnih informacij, in končamo pri ontologijah.

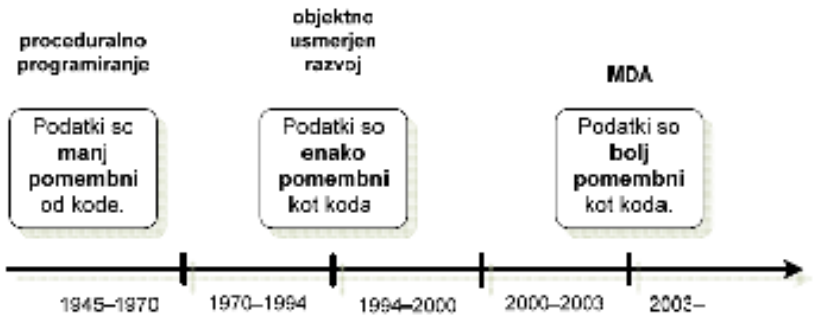


Slika 6: Naraščanje »inteligence« v podatkih [4]

- **Tekst in podatkovne baze (pred XML)**
  Začetna faza, kjer so podatki pripadali določenemu programu. Inteligenca je bila zato skrita v samem programu in ne v podatkih.
- **XML dokumenti za določeno domeno**
  Faza, kjer podatki v določeni problemski domeni dosežejo določeno neodvisnost od programov. Podatki so sedaj dovolj »inteligentni«, da se lahko prenašajo med različnimi programi znotraj določene problemske domene. Kot primer lahko navedemo XML standarde v zdravstvu ali zavarovalništvu.
- **Taksonomije in dokumenti z mešanimi slovarji**
  Podatke lahko združimo iz različnih domen in jih tudi ustrezno klasificiramo v obliki hierarhične taksonomije. Pravzaprav lahko klasifikacijo uporabljamo za odkrivanje podatkov, preprosta razmerja med kategorijami in taksonomijo in za povezovanje ter združevanje podatkov. Podatki so v tej fazi dovolj »inteligentni«, da jih lahko preprosto odkrijemo in razumno združimo z ostalimi podatki.
- **Ontologije in pravila**
  Na tej ravni lahko iz obstoječih podatkov izpeljemo nove s pomočjo logičnih pravil. Podatki so sedaj že na dovolj visoki ravni, da jih lahko opišemo z dejanskimi medsebojnimi povezavami in naprednimi formalizmi, s pomočjo katerih nad to »semantično algebro« izvajamo logične izračune. Takšne podatke lahko povezujemo na bolj atomarnem nivoju in izvajamo analize podatkov z majhnimi zrni. V tem primeru podatki ne predstavljajo zgolj surovine, ampak so del nekakšnega naprednega mikrokozmosa. Primer takšne predstavitve podatkov je samodejni prevod dokumenta iz enega problemskega področja v ekvivalentnega oziroma čim bolj podoben dokument v drugi problemski domeni.

Ko govorimo o pogledu razvijalcev na evolucijo podatkov, lahko na sliki 7 vidimo vpliv podatkov na razvoj programske opreme. Proceduralno programiranje je osredotočeno na funkcionalno dekompozicijo opravil. Podatke so spreminjale in obvladovale procedure, kjer je veljalo, da so podatki manj pomembni od kode. Pri objektno usmerjenem razvoju se pojavijo enkapsulacija, dedovanje in polimorfizem. Ideja o varovanju podatkov s pomočjo rezerviranih besed in metod za dostop povzroči dvig ravni pomembnosti podatkov, tako da postanejo enakovredni metodam, ki s temi podatki manipulirajo. S prihodom XML-ja, MDA-ja (Model Driven Architecture - metodologija za razvoj programske opreme, kjer se funkcionalnost

sistema določi v modelu, ki je neodvisen od platforme) in podpore metapodatkom v času izvajanja vstopamo v obdobje, kjer podatki postajajo bolj pomembni od same kode. V ospredje ne prihaja zgolj sintaktična pravilnost, ampak predvsem semantična [4].



Slika 7: Pogled razvijalcev na podatke [4]

6 SEMANTIČNI SPLET

Že dolgo je bilo jasno, da potrebujemo nov pristop za upravljanje z informacijami in podatki. Tehnološki razvoj zadnjih nekaj desetletij, vključno s svetovnim spletom, je vsakomur omogočil dostop do mnogo več informacij, kot jih lahko uspešno doume in nadzira. Študije so pokazale, da se je obseg informacij na svetovnem spletu samo med leti 2000 in 2003 povečal za trikrat. Zato so nujno potrebne nove tehnike, ki nam bodo pomagale najti smisel iz vsega tega, poiskati tisto, kar želimo vedeti in odstraniti ostalo, izluščiti in povzeti tisto, kar je pomembno, in nam pomagale razumeti povezave med relevantnimi informacijami. Velik izziv za organizacije je tudi povečati produktivnost delavcev, ki upravljajo z znanjem, teh pa je v razvitem svetu vse več.

Hkrati pa je povezovanje oziroma integracija informacij glavni izziv strokovnjakom s področja informacijske tehnologije. Stroški integracije informacij so tako znotraj organizacij kot tudi z zunanjimi partnerji zelo veliki. Integracija informacij v organizacijah je pomembna predvsem za doseganje boljšega razumevanja poslovanja skozi podatke in za razumevanje povezav med podatki. Poleg tega pa prihaja tudi do potrebe po povezovanju aplikacij, z možnostjo skupne rabe podatkov, informacij ter poslovne logike. K temu so prispevale težnje po integraciji novih tehnologij različnih proizvajalcev. Rešitev je nastala s konceptom storitveno orientirane arhitekture (SOA), kjer so poslovne funkcije implementirane kot šibko povezane storitve. Takšen način zagotavlja bolj fleksibilno in šibko povezane vire kot v klasični sistemski arhitekturi in omogoča tudi ponovno uporabo. Spletne storitve so primarni način implementacije storitveno usmerjene arhitekture. Pri tem pa je potrebno identificirati in integrirati potrebne storitve ter obenem omogočiti skupno rabo podatkov med njimi.

Učinkovita implementacija storitev, upravljanja in integracije informacij ter integracija aplikacij zahteva kot osnovo temeljne procese in informacije, ki morajo biti opisani semantično, kar pomeni, da morajo biti računalniki sposobni sami razumeti njihov opis. To pa je osnovna ideja semantičnega spleta, ki se je prvič pojavila leta 1999 (Berners Lee). Zadnja leta je viden močan razvoj teh idej predvsem s strani konzorcija W3C. W3C je standardiziral prve jezike za podporo semantičnega spleta, obenem pa poteka tudi razvoj drugih potrebnih tehnologij, kot so procesiranje naravnega jezika, upravljanje z znanjem in upravljanje z ontologijami [11, 30].

6.1 Cilj semantičnega spleta

Semantični splet in tehnologije semantičnega spleta ponujajo nove pristope k upravljanju z informacijami in procesi, pri čemer je osnovni princip kreiranje in uporaba semantičnih metapodatkov.

Metapodatki lahko nastopajo na dveh nivojih. Na eni strani lahko opisujejo dokument, na primer spletno stran, ali pa del dokumenta, na primer odstavek. Po drugi strani pa lahko metapodatki opisujejo entitete znotraj dokumenta, na primer osebo ali podjetje. V obeh primerih je pomembno, da so metapodatki semantični, s čimer opišejo vsebino dokumenta (namen, povezava z drugimi dokumenti) ali pa entitete znotraj njega. To je v nasprotju z metapodatki v današnjem spletu, ki so vključeni v HTML in opisujejo obliko, v kateri naj bodo informacije predstavljene. Z uporabo HTML-ja lahko določimo, da bo neka beseda prikazana odebeljeno ali rdeče barve, ne moremo pa določiti, da neka vrednost označuje ceno produkta, ime avtorja ali kaj podobnega. Obstaja pa veliko dodatnih storitev, ki omogočajo takšne metapodatke.

Najprej lahko uredimo in poiščemo informacije glede na njihov pomen in ne samo tekst. Z uporabo semantike sistemi razumejo, kdaj sta neka beseda in fraza ekvivalentni. Pri iskanju imena »France Prešeren« lahko tako dobimo enako primeren dokument kot pri besedni zvezi »Največji slovenski pesnik«. Nasprotno pa lahko tudi razlikujejo v primerih, ko je ista beseda uporabljena za različne pomene. Če iščemo povezave z besedo »Jaguar« v kontekstu avtomobilske industrije, lahko sistem izpusti povezave na dokumente o divjih zvereh. Kadar je o neki temi možno poiskati zelo malo, pa lahko sistem poskuša poiskati informacije o semantično povezanih temah.

Z uporabo semantike lahko izboljšamo način predstavitve informacij. Najprej lahko pri iskanju namesto linearnega seznama zadetkov izpišemo rezultate, zbrane po pomenu. Tako ob iskanju informacij o besedi »Jaguar« dobimo dokumente zbrane glede na to, ali so povezani z avtomobili, divjimi zvermi ali še s kakšnimi drugimi temami. Lahko pa postorimo še več z uporabo semantike za strnitev informacij iz vseh relevantnih dokumentov, pri tem odstranimo podvajanja in informacije povzamemo, kjer je to primerno. Povezave med ključnimi entitetami v dokumentih so lahko predstavljene celo vizualno. Podpora vsemu temu je zmožnost sklepanja, to je ustvarjanje povzetkov iz obstoječega znanja za kreiranje novega znanja.

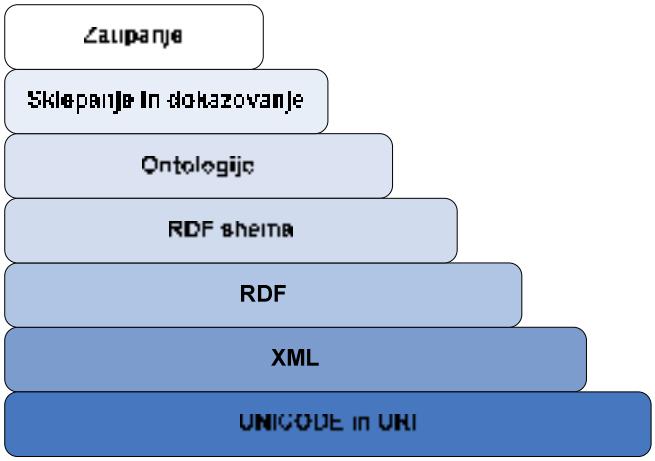
Uporaba semantičnih metapodatkov je nujna tudi za integracijo informacij iz heterogenih virov znotraj ene organizacije ali med več organizacijami. Tipično so uporabljene različne sheme za opis in klasifikacijo informacij, v informacijah pa se uporablja tudi različna terminologija. S kreiranjem povezav med različnimi shemami je mogoče ustvariti poenoten pogled in omogočiti medsebojno delovanje med procesi, ki uporabljajo te informacije.

Semantični opis lahko prav tako priredimo procesom, ki predstavljajo spletno storitev. Če funkcionalnost spletne storitve opišemo semantično, lahko takšno storitev veliko preprosteje najdemo. Če so obstoječe spletne storitve opremljene z metapodatki, ki opisujejo njihovo funkcionalnost in kontekst, lahko neko novo spletno storitev kreiramo avtomatsko s kombinacijo obstoječih spletnih storitev. Uporaba takšnih semantičnih opisov je skoraj nujno potrebna za dosego velikega obsega implementacij v sklopu storitveno orientirane arhitekture (SOA).

Vse pridobitve semantičnega spleta so v veliki meri povezane z ontologijami. Tako največ raziskovanja poteka ravno na področju spletnih ontologij, saj predstavljajo način obvladovanja raznovrstnih predstavitev spletnih virov. Domenski model iz ontologije se lahko uporabi kot enotna struktura za skupno predstavitev in semantiko informacij. Prav tako se v viziji semantičnega spleta doseže jasen pomen dialoga med oddaljenimi aplikacijami ali agenti s pomočjo uporabe ontologije [11, 30].

6.2 Arhitektura semantičnega spleta

Semantični splet je podprt z zbirko tehnologij, orodij in standardov, ki oblikujejo osnovne gradnike sistema, ki lahko podpira vizijo spleta, obogatenega s pomenom. Semantični splet razvija nivojsko arhitekturo, ki je predstavljena na sliki 8.



Slika 8: Arhitektura semantičnega spleta [3]

- **Unicode in URI**: Unicode je standard za predstavitev računalniških znakov, URI pa je standard za identifikacijo in lociranje virov (na primer spletne strani). Skupaj predstavljata osnovo za predstavitev znakov, uporabljenih v večini svetovnih jezikov, in za identifikacijo virov.
- **XML**: Jezik XML in z njim povezani standardi, kot recimo sheme in imenska polja, predstavljajo osnovna sredstva za strukturiranje podatkov na spletu, toda brez dodatnega opisa pomena podatkov. Ti standardi so že zelo dobro uveljavljeni na spletu.
- **RDF**: RDF (ogrodje za opis virov) je prvi sloj v sklopu semantičnega spleta. RDF je preprosto ogrodje za predstavitev metapodatkov, ki za identifikacijo spletnih virov uporablja URI in grafični model za opis relacij med viri. Pri tem je možnih več sintaktičnih predstavitev, vključno s standardnim XML formatom.
- **RDF shema**: RDF shema je preprost tip modelirnega jezika za opisovanje razredov virov in lastnosti povezav med njimi v osnovnem RDF modelu. Omogoča preprosto ogrodje za sklepanje o različnih tipih virov.
- **Ontologije**: Ontologije omogočajo bogatejši jezik za zagotavljanje bolj kompleksnih omejitev na tipih virov in njihovih lastnostih.
- **Logika in dokazovanje**: Omogočata avtomatičen sistem sklepanja, ki se nahaja nad plastjo ontologij, za izvrševanje dodatnih povzetkov in sklepov. Poleg tega lahko programski agenti z uporabo takšnih sistemov izvajajo odločitve o tem, ali določen vir zadošča danim zahtevam.
- **Zaupanje**: Najvišji nivo sklada se nanaša na zaupanje, ki ga semantični splet lahko podpira. Ta komponenta je še slabo razvita, njena vizija pa je omogočiti ljudem vprašanja o zanesljivosti informacij na spletu, s čimer bi dosegli neko potrditev o njihovi kakovosti.

Na samem vrhu sklada se nahajajo programi, ki uporabljajo vse spodaj ležeče tehnologije in jih lahko imenujemo »inteligentne aplikacije«, saj opravila opravljajo na naprednejši način in prinašajo bolj »inteligentne« storitve [3, 4].

6.3 Programski agenti

Prava moč semantičnega spleta se bo pokazala, ko bo ustvarjenih mnogo programov, ki bodo zbirali spletno vsebino iz različnih virov, procesirali informacije in izmenjevali rezultate z drugimi programi. Učinkovitost takšnih programskih agentov bo naraščala eksponentno z večanjem obsega spletnih vsebin, ki jih bodo lahko brali računalniki, in večanjem števila avtomatiziranih storitev. Semantični splet podpira takšno sinergijo, da tudi agenti, ki niso bili izrecno narejeni za vzajemno delovanje, lahko med seboj izmenjujejo podatke, če so ti le opremljeni s semantiko [30].

7 JEZIKI ZA PREDSTAVITEV ONTOLOGIJ

Jeziki za predstavitev ontologij so formalni jeziki, ki jih na področju računalništva in umetne inteligence uporabljamo za konstruiranje ontologij. Omogočajo zajemanje znanja o določenih domenah in pogosto vključujejo tudi pravila za sklepanje, ki podpirajo procesiranje tega znanja.
Od leta 1990 dalje je bilo razvitih kar nekaj jezikov za predstavitev znanja oziroma ontologij. Razdelimo jih v dve skupini, in sicer v skupino osnovnih jezikov za predstavitev znanja ter v skupino označevalnih jezikov za predstavitev ontologij, ki večinoma bazirajo na XML sintaksi.

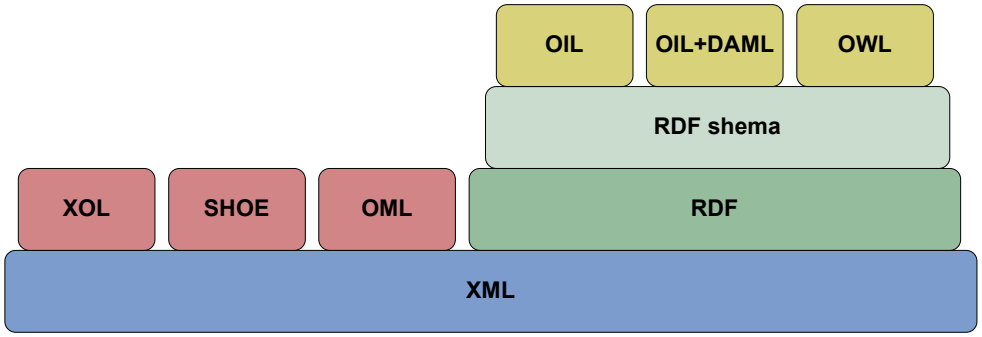
7.1 Osnovni jeziki za predstavitev ontologij

Cycl je formalni jezik, katerega sintaksa izvira iz predikatne logike prvega reda. Uporablja se za opis splošnega znanja. Vsebuje naslednje izrazne tipe: konstante, formule, funkcije, spremenljivke in količinske tipe. Njegove prednosti so izraznost, natančnost in neodvisna predstavitev od uporabe.
KIF je računalniški jezik za izmenjavo znanja med različnimi programi in ima deklarativno semantiko (pomen izrazov v predstavitvi je razumljiv brez uporabe prevajalca za omenjene izraze). S svojo splošnostjo (predstavitev poljubnih stavkov v obliki predikatnega računa prvega reda) omogoča predstavitev znanja o znanju, opredelitev objektov, funkcij in relacij. Ne vključuje pa podpore odločanju. Na osnovi jezika KIF je nastal tudi jezik Ontolingua.
Ontolingua je jeziku KIF dodala samo dodatne sklope aksiomov.
LOOM je bil razvit vzporedno z Ontolinguo in je bil sprva namenjen samo predstavitvi znanja. Bazira na opisni logiki in produkcijskih pravilih in omogoča avtomatično klasifikacijo konceptov. Z njim lahko predstavimo naslednje komponente ontologij: koncepte, taksonomije, n-terne relacije, funkcije, aksiome in pravila.
OCML je strukturni jezik, ki zagotavlja mehanizme za izražanje relacij, funkcij, pravil, razredov in primerkov. Za bolj učinkovito izvrševanje jezika vključuje tudi logične mehanizme sklepanja. Razvit je bil za razvoj izvrševalnih ontologij in modelov pri metodah za reševanje problemov. Podoben je jeziku Ontolingua, s katerim je tudi združljiv.

FLogic je jezik za predstavitev znanja in ontologij in je kombinacija strukturnega jezika in predikatne logike prvega reda. Omogoča predstavitev konceptov, taksonomij, binarnih relacij, funkcij, aksiomov in pravil. Poleg tega pa v povezavi z objektno orientiranimi jeziki omogoča predstavitev kompleksnih objektov, dedovanja, polimorfizma, poizvedovanja in enkapsulacije. FLogic je v podobni relaciji z objektno usmerjenim programiranjem, kot je klasični predikatni račun z relacijskimi bazami.
OKBC je protokol, ki omogoča dostop do različnih sistemov za predstavitev znanja. S tem omogoča enoten model predstavitve znanja in temelji na splošni konceptualizaciji razredov, primerkov, dedovanja itd. [17, 18, 29]

7.2 Označevalni jeziki za predstavitev ontologij

Z razcvetom interneta se je začel razvoj jezikov za predstavitev ontologij, ki so se razvijali na osnovi značilnosti svetovnega spleta, njihova največja vrednost pa se kaže v kontekstu semantičnega spleta. Najbolj znani so naslednji jeziki, ki bazirajo na sintaksi XML: XOL (Ontology Exchange Language), SHOE (Simple HTML Ontology Extension), OML (Ontology Markup Language), RDF (Resource Description Framework) in RDF Schema. Pozneje so bili razviti še trije jeziki, ki razširjajo funkcionalnosti RDF(S) (združena RDF in RDF shema). To so OIL (Ontology Inference Layer), DAML+OIL in OWL (Web Ontology Language). Ti jeziki in relacije med njimi so prikazani na sliki 9.



Slika 9: Označevalni jeziki za predstavitev ontologij [1]

7.2.1 XML(eXtended Markup Language)

XML primarno ni jezik za predstavitev ontologij, je pa osnova za več drugih jezikov. Uporabnikom omogoča kreiranje lastnih oznak in atributov, definiranje strukture podatkov (vključno z gnezdenjem), zbiranje podatkov iz dokumentov in razvoj aplikacij za testiranje strukturne pravilnosti dokumentov XML. XML je preprost za analiziranje in razčlenjevanje, njegova sintaksa je dobro definirana in je ljudem razumljiv za branje. Vendar pa jezik XML ne pove ničesar o semantiki podatkov. Na svetovnem spletu je postal standard za opis podatkov, jezikom za predstavitev ontologij pa služi kot preprost način za predstavitev njihove sintakse [17].

7.2.2 XOL (Ontology Exchange Language)

Združenje bioinformatike v ZDA je oblikovalo jezik XOL za izmenjavo definicij ontologij med heterogenimi programskimi sistemi znotraj njihove domene. Raziskovalci so ga razvili zaradi potreb strokovnjakov v bioinformatiki. Za osnovo jezika XOL so izbrali jezika Ontolingua in OML in združili visoko izraznost jezika OKBC ter sintakse OML-ja na osnovi jezika XML. XOL je funkcionalno zelo omejen in ne podpira mehanizmov sklepanja. Za razvoj ontologij v jeziku XOL ne obstaja nobeno orodje, vendar je osnovan na jeziku XML. Za oblikovanje XOL dokumentov lahko uporabljamo tudi urejevalnike XML-ja [1, 18].

7.2.3 SHOE (Simple HTML Ontology Extension)

SHOE je bil prvotno razvit kot razširitev jezika HTML, ki je vključeval dodane semantične podatke HTML ali drugim spletnim dokumentom, kar je omogočalo strojno branje dokumentov. Uporabljal je drugačne oznake kot HTML in omogočal vključitev ontologij v HTML dokumente. Z razvojem jezika XML, kot standardnega jezika za izmenjavo informacij na spletu, je bila osnovna sintaksa SHOE prilagojena jeziku XML. Programski agenti so lahko s pomočjo jezika SHOE zbirali informacije o spletnih straneh in dokumentih, izboljšali mehanizme iskanja in kopičili znanje. Ta proces je bil sestavljen iz treh faz: definiranje ontologije, označitev strani HTML s semantičnimi informacijami iz ontologij o vsebini dokumenta in kreiranje programskega agenta za osveževanje informacij s pomočjo iskanja preko vseh obstoječih strani [1, 17, 18].

7.2.4 OML (Ontology Markup Language)

OML je bil razvit delno na osnovi jezika SHOE, zato imata tudi veliko podobnih lastnosti. Obstajajo štirje nivoji jezika OML. Jedro OML je povezano z logičnimi vidiki jezika in je vključeno v vse ostale nivoje. Osnovni OML se uporablja za povezavo do jezika RDF(S). Okrajšani OML vključuje funkcionalnosti za kreiranje konceptualnih diagramov. Standardni OML pa je najbolj izrazna verzija jezika. Za jezik OML, razen urejevalnikov jezika XML, ne obstajajo posebna orodja za razvoj ontologij [1, 18].

7.2.5 RDF (Resource Description Framework)

Jezik RDF je bil razvit strani W3C za kreiranje metapodatkov, ki opisujejo spletne vire. Med jezikoma RDF in XML je močna povezava. Eden glavnih ciljev jezika RDF je namreč specificirati semantiko podatkov v XML zapisu in to na standardiziran način. RDF zagotavlja mehanizme za eksplicitno predstavitev storitev, procesov in poslovnih modelov, poleg tega pa omogoča prepoznavanje nejasnih informacij. Podatkovni model jezika RDF sestavljajo trije tipi objektov, osebek (subjekt), lastnost in predmet (objekt), ki nastopajo skupaj v trojicah. Osebki so entitete, ki jih naslavljamo z URI naslovi. URI (Universal Resource Identifier) naslovi se uporabljajo podobno kot povezave na svetovnem spletu (URL-ji, spletni naslovi, so najbolj pogost tip URI-ja). Lastnosti definirajo določene karakteristike, poglede, atribute ali povezave, ki opisujejo osebek. Predmeti pa določajo vrednost lastnosti preko določenega vira, ki je tudi URI naslov. Z URI naslovom so lahko določene tudi lastnosti, kar omogoča vsakomur kreiranje nove lastnosti z določitvijo URI naslova zanjo nekje na spletu [1, 18, 28].
Podatkovni model RDF pa ne vsebuje mehanizmov za definiranje povezav med lastnostmi in viri, ne vključuje pa tudi aksiomov in mehanizmov sklepanja. Prednost jezika RDF je v uporabi XML-ja in URI-ja, s čimer je omogočena ponovna uporaba entitet na svetovnem spletu.

```
<rdf:RDF
 xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:terms="http://purl.org/dc/terms/">
        <rdf:Description rdf:about="urn:x-states:New%20York">
                        <terms:alternative>NY</terms:alternative>
        </rdf:Description>
</rdf:RDF>
```

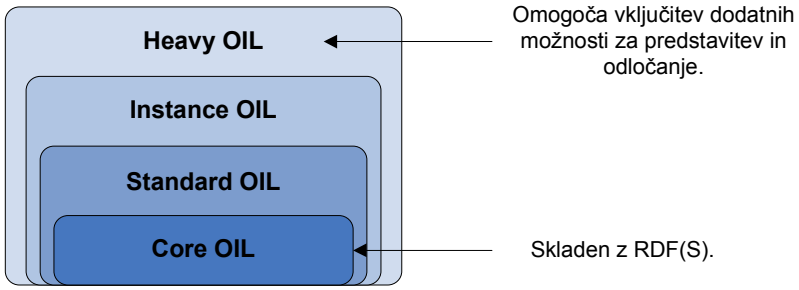
Na primeru vidimo zapis stavka: Država New York ima okrajšavo NY. Pri tem je »New York« osebek stavka, »ima okrajšavo« lastnost in »NY« predmet.
URI naslov "urn:x-states:New%20York" je naslov, ki označuje državo New York, URI naslov http://purl.org/dc/terms/alternative pa je naslov lastnosti [28].

7.2.6 RDF shema

RDF shema je opisni jezik, ki razširja jezik DRF in omogoča definiranje RDF shem. Podatkovni model RDFS vsebuje mehanizme za definiranje povezav med atributi in viri. Poleg tega razširja jezik RDF z dodatnimi načrtovalskimi gradniki, ki jih pogosto najdemo pri jezikih za predstavitev ontologij. To so razredi, dedovanje razredov, dedovanje lastnosti, domene in obseg omejitev. V takšnem jeziku ne moremo predstaviti aksiomov in edini mehanizem sklepanja, ki je na voljo, temelji le na obstoječih povezavah med razredi.
Jezika RDF in RDF shema nastopata skupaj pod oznako RDF(S). Jeziku RDF(S) primanjkuje nekaj operatorjev za formiranje bogatejših konceptov in boljšo izraznost. Vendar pa so se bogatejši in bolj izrazni jeziki razvili na podlagi RDF(S) (OIL, OIL+DAML, OWL). RDF(S) se uporablja kot predstavitveni format v številnih orodjih in projektih, kot so Protege, OntoStudio in podobni [1, 4, 18].

7.2.7 OIL (Ontology Interchange Language)

OIL je bil razvit v sklopu projekta On-To-Knowledge in omogoča medsebojno sodelovanje med spletnimi viri na podlagi semantike. Njegova sintaksa in semantika bazirata na obstoječih tehnologijah XOL, OKBC in RDF(S). Zagotavlja veliko prvin modeliranja, ki se uporabljajo tako v ontologijah z opisno logiko, kot tudi v strukturiranih ontologijah. Njegova semantika je preprosta in dobro definirana, omogoča pa tudi avtomatično sklepanje. OIL ima štiri plasti, pri čemer je v vsaki višji plasti dodanih nekaj funkcionalnosti spodnje plasti. To omogoča agentom, ki so sposobni procesirati samo nižje plasti, da lahko še vedno delno razumejo ontologije, ki so opisane na višjem nivoju. Najnižja plast je v celoti združljiva z RDF(S), kar pomeni, da lahko vsak agent za RDF(S) procesira tudi ontologije v OIL.
Za kreiranje ontologij s pomočjo jezika OIL so na voljo orodja, kot so OILEd, Protege in WebODE. Sintaksa OIL pa je lahko poleg jezika XML izražena tudi z ASCII znaki.



Slika 10: Plasti jezika OIL

7.2.8 DAML+OIL

DAML+OIL je bil razvit s sodelovanjem ameriške agencije DARPA in Evropske unije in omogoča medsebojno sodelovanje med XML dokumenti na podlagi semantike. To je eden izmed semantično najbolj izraznih jezikov za opis dokumentov na svetovnem spletu. DAML+OIL je bil razvit na osnovi RDF(S) in tudi jezika OIL, s katerim sta si podobna in imata enake cilje. Tudi za DAML+OIL obstaja mnogo orodij in urejevalnikov (OILEd, WebODE, Protege, OntoEdit) [1, 4].

7.2.9 OWL (Web Ontology Language)

OWL je skupina jezikov za predstavitev znanja, ki omogočajo kreiranje ontologij. Nastal je leta 2003 pod okriljem W3C. OWL je sestavljen iz treh nivojev: OWL Lite, OWL DL in OWL Full. Jeziki bazirajo na dveh večinoma združljivih semantikah. OWL Lite in OWL DL bazirata na opisni logiki z dobro razumljivimi lastnostmi. OWL Full pa uporablja novejši semantični model, ki omogoča tudi združljivost z RDF shemo. OWL ontologije večinoma uporabljajo RDF in sintakso XML.
OWL je zgrajen na konceptih DAML+OIL, saj ima, podobno kot predhodnik, razrede, lastnosti, omejitve nad lastnostmi in primerke razredov ter lastnosti. OWL, podobno kot DAML+OIL, za opisovanje razredov in informacij o podatkovnih tipih (iz XML sheme) uporablja konstrukta »podrazred« in »nepovezan« ter omogoča logične kombinacije izrazov nad razredi (presek, unija, komplement), pa tudi razvrščanje sistemov v sezname.
Podatki, opisani v OWL ontologijah, so interpretirani kot sklopi primerkov in sklopi lastnosti, ki povezujejo primerke med seboj. OWL ontologije sestavljajo sklopi aksiomov, ki postavljajo omejitve na primerke (razrede) in tipe relacij, ki so dovoljene med njimi. Ti aksiomi določajo semantiko in tako omogočajo sistemom povzemanje dodatnih informacij glede na podane semantične podatke.
OWL si preprosto lahko predstavljamo kot množico trojčkov RDF, vendar le-ti uporabljajo slovar OWL, ki jim daje v okviru jezika OWL poseben pomen.
OWL je trenutno ena izmed osnovnih tehnologij in najbolj izrazen jezik za predstavitev znanja na semantičnem spletu. Uporablja se tako na akademskem kot tudi na komercialnem področju.

Nivoji jezika OWL:

Slika 11: Nivoji jezika OWL [4]

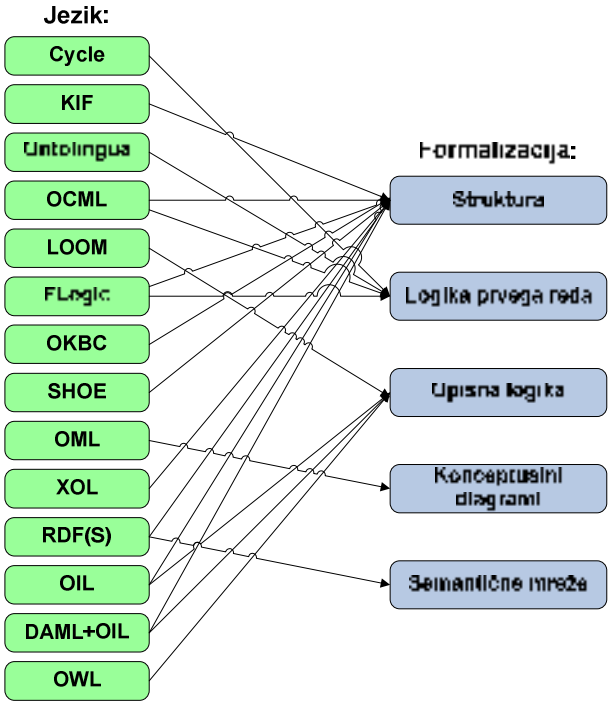
- **OWL Lite** je primeren za uporabnike, ki primarno potrebujejo le klasifikacijo hierarhije in preproste omejitve. Pri omejitvi številnosti sta tako dovoljeni le vrednosti 0 in 1. Prvotno je bilo mišljeno, da bo preprosteje razviti orodja za OWL Lite kot za jezike na ostalih dveh nivojih. Toda s kompleksnimi kombinacijami gradnikov jezika OWL Lite je možno sestaviti tudi večino konstruktov jezika OWL DL. Zato ni razvoj orodij za OWL Lite veliko preprostejši, posledično pa se ta jezik tudi veliko ne uporablja. Še vedno pa ima OWL Lite večjo izrazno moč kot shema RDF.

- **OWL DL** je bil razvit za zagotovitev največje možne izraznosti, pri čemer se še ohranja celovitost izračunljivosti (vsi zaključki bodo zanesljivo izračunani), rešljivost (vsi izračuni bodo dokončani v končnem času) in prisotnost praktičnih mehanizmov sklepanja. OWL DL vključuje vse konstrukte jezika OWL, vendar so nekateri lahko uporabljeni samo pod določenimi pogoji. Oznako OWL DL je jezik dobil zaradi skladnosti z opisno logiko.
- **OWL Full** bazira na drugačni semantiki kot jezika OWL Lite in OWL DL in je bil osnovan za ohranitev skladnosti z RDF shemo. Namenjen je uporabnikom, ki želijo največjo mogočo izraznost in sintaktično svobodo jezika RDF, vendar brez zagotavljanja celovitosti izračunavanja. V jeziku OWL Full je lahko, na primer, razred obravnavan kot množica posameznih razredov ali kot posameznik. To v jeziku OWL DL ni dovoljeno. OWL Full omogoča dodajanje pomena ontologijam, že definiranim v jeziku RDF ali obeh ostalih OWL jezikih. Zaradi velike izraznosti je malo verjetno, da bi kakšno orodje uspelo zagotoviti podporo celotnemu mehanizmu sklepanja jezika OWL.

Opisane ravni OWL predstavljajo naraščajočo stopnjo izraznosti, kar pomeni, da višje ravni vsebujejo funkcionalnosti nižjih. Trditve, predstavljene na nižjem nivoju, so veljavne tudi na višjem, nasprotno pa to ne velja [4, 24].

7.3 Formalizacija jezikov

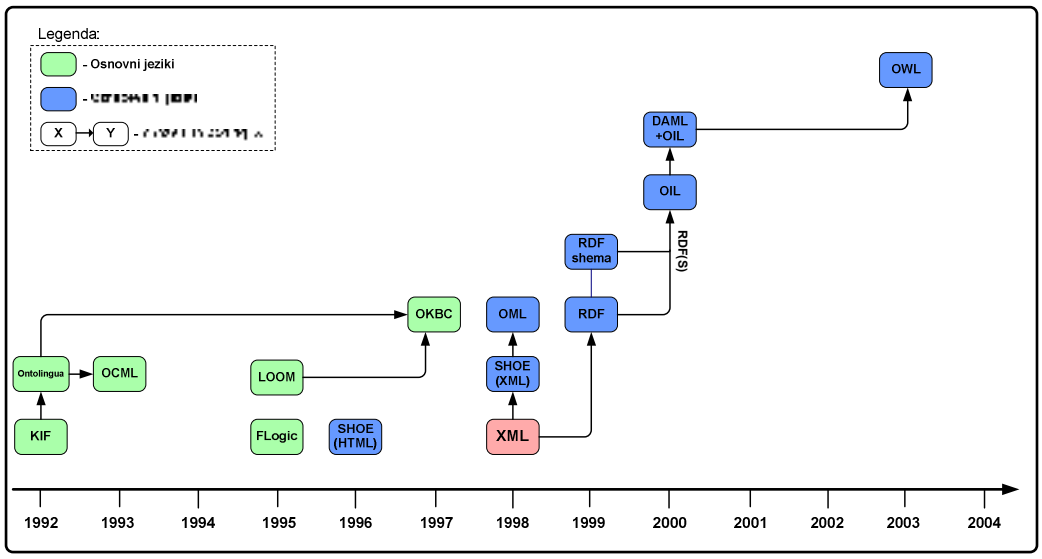
Na sliki 12 so predstavljeni načini formalizacije posameznih jezikov za predstavitev ontologij.



Slika 12: Načini formalizacije jezikov za predstavitev ontologij [2]

Omeniti velja, da je pri razvoju in implementaciji ontologije potrebno najprej razmisliti, kaj bo aplikacija zahtevala v smislu izraznosti in mehanizmov sklepanja, saj ne omogočajo vsi jeziki predstavitev enakih komponent in sklepov na enak način. Predstavitev in sklepanja na osnovi osnovnih informacij, kot so koncepti, taksonomije in binarne relacije, pogosto niso dovolj, če želimo razviti kompleksnejše ontologije.

7.4 Kronološki pregled razvoja jezikov za predstavitev ontologij



Slika 13: Kronološki pregled razvoja jezikov za predstavitev ontologij

Na sliki 13 je prikazan kronološki pregled razvoja jezikov za predstavitev ontologij. Predstavljeni so tudi vplivi, ki so jih imeli določeni jeziki na razvoj novih jezikov.

8 METODOLOGIJE ZA RAZVOJ ONTOLOGIJ

Že na področju upravljanja z znanjem je veljalo splošno prepričanje, da metodologije za razvoj sistemov znanja pripomorejo razvojnim projektom k uspešni realizaciji ciljev v določenem času. S prihodom ontologij na področje sistemov znanja je podoben metodološki pristop k strukturiranju procesov za razvoj ontologij sledil tudi na tem področju.
Veliko pristopov je bilo predstavljenih za razvoj ontologij. Nekateri med njimi opisujejo gradnjo ontologij od samega začetka ali ponovno uporabo obstoječih ontologij, drugi pa pristope h gradnji ontologij s pomočjo preureditve drugih ontologij, na primer, z združevanjem.
Čeprav je lahko po navadi skupina uporabnikov ontologij razmeroma velika, je na drugi strani pri razvoju precej manjša skupina domenskih strokovnjakov, ki zagotovijo model znanja, in inženirjev ontologij, ki to znanje strukturirajo in formalizirajo.

8.1 Definicija metodologije za razvoj ontologij

Po EMRIS (Enotna metodologija razvoja informacijskih sistemov) [14] metodologija zajema vse, kar redno počnemo, da bi dosegli želen rezultat, torej izdelek ali storitev, ki je cilj našega dela. V primeru razvoja programske opreme to ne pomeni zgolj postopkov, ki so neposredno povezani z razvojem (na primer analiza, načrtovanje), temveč zajema tudi podporne postopke, načine komunikacije med sodelujočimi, pravila odločanja in podobno. V tem oziru lahko metodologijo opredelimo tudi kot množico dogovorov (konvencij), s katerimi se projektna skupina/organizacija strinja.
Metodologija je kot takšna prežeta s filozofijo, načeli, idejami in pogledi organizacije in njenih članov, kar še zlasti poudarja njeno sociološko komponento. Metodologija namreč ni nekaj, kar lahko nastane neodvisno od ljudi, katerim je namenjena. Čeprav obstajajo številne formalne metodologije, ki temeljijo na preizkušenih postopkih, ki so se v praksi izkazali kot dobri, si jih organizacije, ko jih privzamejo, vedno prilagodijo po svoji meri, tako da ustrezajo njihovemu načinu dela ter percepciji domene, za katero so vzpostavljene. Z uporabo metodologije se uporabniki učijo in pridobivajo nove izkušnje, s čimer se posledično bogati tudi metodologija sama.
IEEE definira metodologijo kot obsežno in poenoteno serijo tehnik ali metod, ki kreirajo splošno teorijo, kako naj bo izvedena kopica umsko intenzivnega dela. Metodologija naj bi definirala objektivno zbirko kriterijev za določanje, če so rezultati postopka sprejemljive kakovosti [11].
V nasprotju z metodologijo je metoda urejen proces ali postopek, uporabljen pri razvoju produkta ali izvedbi storitve. Tehnika pa je tehnični postopek, uporabljen za dosego cilja.
Proces je funkcija, ki mora biti izvedena v življenjskem ciklu programske opreme in je sestavljen iz aktivnosti. Naloga pa je dobro definirano delo, dodeljeno enemu ali več projektnim članom. Podobne naloge se povezujejo v aktivnosti.

8.1.1 Metodologija

Metodologije za razvoj ontologij morajo upoštevati naslednje tri tipe aktivnosti:

- **Aktivnosti upravljanja z ontologijo**
  Postopki za aktivnosti upravljanja z ontologijo morajo vsebovati načrt razvrščanja nalog pri razvoju ontologije. Poleg tega je potrebno definirati kontrolne mehanizme in korake za zagotavljanje kakovosti.

- **Aktivnosti razvoja ontologije**
  Pri razvoju ontologije je pomembno, da so definirani postopki za študije o izvedljivosti in okolju, v katerega bo ontologija integrirana. Po odločitvi o začetku gradnje mora inženir ontologij določiti postopke ter konceptualizirati, formalizirati in implementirati ontologijo. Na koncu je potrebno za uporabnike in inženirje pripraviti še napotke za vzdrževanje, uporabo in nadaljnji razvoj ontologije.

- **Spremljevalne aktivnosti**
  Za podporo pri razvoju ontologij je potrebno izvesti veliko spremljevalnih aktivnosti. Te vključujejo pridobivanje znanja, vrednotenje, integracijo, združevanje, razvrščanje in upravljanje s konfiguracijami. Te aktivnosti se izvajajo skozi vse korake razvojnega in upravljalskega procesa. Pridobivanje znanja se lahko izvaja na centraliziran ali decentraliziran način. Učenje ontologij na način podpore ročnemu pridobivanju znanja z metodami strojnega učenja [11].

8.1.2 Dokumentacija

Pomembno je, da so rezultati opravljenih aktivnosti dokumentirani. V poznejših fazah razvoja to pomaga pri sledenju, zakaj so bile določene aktivnosti izvedene. Dokumentiranje rezultatov je lahko poenostavljeno z uporabo primernih programskih orodij. Nivo dokumentacije se lahko glede na metodologijo spreminja. Nekatere metodologije zahtevajo samo dokumentiranje rezultatov razvojnega procesa, medtem ko druge upoštevajo tudi odločitveni proces [11].

8.1.3 Vrednotenje

V okviru razvoja ontologij naj bi vrednotenje omogočalo sredstvo za merjenje kakovosti kreirane ontologije. To je pri ontologijah še zlasti težko, ker so odločitve modeliranja v večini primerov subjektivne. Zato se skušamo čim bolj osredotočiti na vrednotenje, ki izhaja iz statističnih podatkov [11].

8.2 Ogrodje za ocenjevanje metodologij

V tem poglavju bodo predstavljeni glavni kriteriji za primerjavo različnih pristopov oziroma metodologij za razvoj ontologij od začetka ali za ponovno uporabo že obstoječih. Ti kriteriji so:

1. **Načrt strategije izgradnje**
   - *1.1. Načrt življenjskega cikla*
     Življenjski cikel predstavlja niz stopenj, skozi katerega mora ontologija v njenem življenjskem obdobju in opisuje aktivnosti, ki morajo biti izvedene v vsaki stopnji, ter povezanost med stopnjami.
   - *1.2. Strategija upoštevanja aplikacije*
     Pomeni stopnjo odvisnosti ontologije glede na aplikacijo, ki to ontologijo uporablja. Metodologije so razdeljene v tri razrede:
     - Aplikacijsko odvisne, ki predstavljajo pristop gradnje ontologije na podlagi aplikacijske baze znanja s procesom abstrakcije.
     - Delno odvisne od aplikacije, ki v fazi specifikacije vključujejo možne scenarije uporabe ontologije.
     - Aplikacijsko neodvisne, kjer je proces popolnoma neodvisen od poznejše uporabe ontologije.
   - *1.3. Uporaba jedrnih oziroma visokonivojskih ontologij*
     Omogoča uporabo jedrne ali visokonivojske ontologije kot začetno točko pri razvoju domenske ontologije.
   - *1.4. Strategija identifikacije konceptov*
     Obstajajo tri strategije identifikacije konceptov: od najbolj konkretnega do najbolj abstraktnega ali koncept od spodaj navzgor (ang. bottom-up), od najbolj abstraktnega do najbolj konkretnega ali koncept od zgoraj navzdol (ang. top-down) in od najbolj primernih do abstraktnih in konkretnih ali koncept od sredine navzven (ang. middle-out).

2. **Načrt procesa razvoja ontologije**
   Osnovan je na ogrodju, ki povzema IEEE 1074-1995 standard za proces razvoja programske opreme. Ta proces je razčlenjen na več drugih procesov, ki so sestavljeni iz aktivnosti.

2.1. Proces upravljanja projekta

Postavljeno mora biti ogrodje projekta in zagotovljena dovolj visoka stopnja upravljanja skozi celoten življenjski cikel produkta. V ta proces sodijo aktivnosti vzpostavitve projekta, spremljanje in kontrola projekta ter upravljanje kakovosti ontologije.

2.2. Razvojno orientirani procesi

To so procesi produkcije, namestitve, delovanja, vzdrževanja in upokojitve ontologije. Razdeljeni so v tri skupine:

- Procesi pred razvojem

  Potekajo pred samim začetkom razvoja ontologije. Vključujejo aktivnosti, povezane s proučevanjem okolja za namestitev ontologije in proučevanjem izvedljivosti. S proučevanjem okolja se doseže možnost natančnejšega definiranja zahtev, proučevanje izvedljivosti pa preprečuje, da se projekt razvoja ontologije ne bi zaključil zaradi neizvedljivih zahtev.

- Procesi razvoja

  To so procesi, ki morajo biti izvršeni za izgradnjo ontologije. Ti procesi vključujejo: obdelavo zahtev, ki vključuje iterativne aktivnosti za izdelavo specifikacije zahtev; proces konstruiranja, katerega cilj je razvoj razumljive in dobro organizirane predstavitve ontologije, ki mora zadoščati vsem zahtevam; proces implementacije, ki pretvori oblikovano predstavitev ontologije v implementacijski jezik.

- Procesi po razvoju

  Povezani so z namestitvijo, delovanjem, podporo, vzdrževanjem in upokojitvijo ontologije. Izvajajo se po izgradnji ontologije.

2.3. Spremljevalni procesi

Potrebni so za uspešno dokončanje projekta izgradnje ontologije. Zagotavljajo dokončanje in kakovost projektnih funkcij. Spremljevalni procesi se izvajajo sočasno z razvojno orientiranimi procesi in vključujejo aktivnosti, ki ne razvijajo ontologije, so pa potrebni za vzpostavitev uspešnega sistema. Te aktivnosti pokrivajo procese pridobivanja znanja, kontrole in ocenjevanja, upravljanja s konfiguracijo ontologije, dokumentiranja in usposabljanja.

3. Uporaba metodologije

Nanaša se na uporabo metodologije na projektih, sprejemanje ontologije s strani drugih razvojnih skupin, število ontologij, razvitih po tej metodologiji, domenah teh ontologij in v katerih sistemih se le-te uporabljajo.

4. Tehnološka podpora

Pomembno je vedeti, ali obstajajo orodja, ki podpirajo metodologijo delno ali v celoti. Orodja lahko močno pripomorejo k preprostejšemu, bolj sistematičnemu in preglednejšemu razvoju ontologij [6, 13].

8.3 Opis metodologij za gradnjo ontologij

Seznam opisanih metodologij za gradnjo ontologij:

- Cyc,
- Enterprise Ontology (metodologija Usholda in Kinga),
- TOVE (metodologija Gruningerja in Foxa),
- KACTUS,
- METHONTOLOGY,
- SENSUS,
- CommonKADS,
- DOGMA,
- On-To-Knowledge,
- HCOME,
- DILIGENT,
- UPON.

8.3.1 Cyc

Cyc metodologija se je razvila iz izkušenj pri razvoju Cyc baze znanja, ki vsebuje ogromno splošnega znanja. Faze izgradnje ontologije so:

- Prva faza predvideva ročni zapis eksplicitnega in implicitnega znanja, ki nastopa v virih znanja, brez pomoči sistemov za procesiranje naravnega jezika ali sistemov učenja.
- Druga faza predvideva kodiranje znanja s pomočjo orodij, ki uporabljajo znanje, že shranjeno v bazi znanja Cyc.
- Tretja faza se izvaja pretežno s podpornimi orodji. Ljudje pomagajo le pri priporočanju primernih virov znanja in razlagi najtežjih delov teksta.

V vsaki fazi se opravljata dve aktivnosti:

- Razvoj predstavitve znanja in visokonivojske ontologije, ki vključuje najbolj abstraktne koncepte. Termini, kot so atribut ali vrednost atributa, so primeri terminov za predstavitev znanja.
- Predstavitev ostalega znanja z uporabo teh terminov.

Metodologija Cyc se je uporabljala v raziskovalnih programih za razvoj visokozmogljivih baz znanja za izboljšanje tehnologije o tem, kako računalniki pridobivajo, predstavljajo in manipulirajo z znanjem.
Metodologija se uporablja samo za gradnjo baz znanja Cyc. Vseeno pa ima Cyc tudi nekaj teorij, ki prikazujejo znanje v različnih domenah in iz različnih pogledov.
Za aplikacije, v katerih se uporabljajo ontologije Cyc, obstaja več modulov in mehanizmov sklepanja, vgrajenih v bazo znanja Cyc. Eden izmed njih je sistem za integracijo heterogenih podatkovnih baz. Ta modul povezuje Cyc slovar s podatkovnimi shemami ostalih baz. S tem so podatki, zajeti v bazah, interpretirani na način, kot so v slovarju Cyc. Modul za iskanje znanja v zajetih informacijah omogoča izvajanje poizvedb v naravnem jeziku, naslednji modul pa omogoča razširitev baze znanja Cyc z informacijami iz svetovnega spleta. Poleg tega je razvitih tudi nekaj Cyc agentov, ki vsi zajemajo osnovno znanje iz baze Cyc, ostalo pa iz specifičnih domen agentov.

Projekt Cyc je močno povečal prepoznavnost tehnologij na področju inženiringa znanja, obenem pa je bila njegova pomanjkljivost ta, da je želel doseči cilj modeliranja »celega sveta«. Ta cilj so pozneje znižali in kompleksno nalogo razdelili na manjše dele, tudi z razdelitvijo visokonivojske ontologije Cyc [6, 18].

8.3.2 Enterprise Ontology (metodologija Usholda in Kinga)

Metodologija Enterprise Ontology, imenovana tudi metodologija Usholda in Kinga, je nastala ob razvoju ontologije za modeliranje procesov. Metodologija predpisuje tudi navodila in korake za razvoj ontologij:

1. **Identificirati namen**
   Pomembno je jasno vedeti, za kaj se ontologija gradi in na kakšne načine se bo uporabljala.
2. **Gradnja ontologije**
   2.1. Zajem ontologije
   - Identifikacija ključnih konceptov in relacij v interesni domeni. Pomembno se je osredotočiti na koncepte in ne na besede, ki jih opisujejo.
   - Kreiranje natančnih in jasnih tekstovnih definicij za te koncepte in relacije.
   - Identifikacija pojmov, ki se nanašajo na te koncepte in relacije.

   Na koncu tega koraka je pomembno, da se razvijalci strinjajo z zajeto terminologijo.
   Avtorji so za ta korak raje uporabljali pristop od sredine navzven, kot pa, da bi najprej iskali najbolj splošne ali najbolj podrobne koncepte. Najprej so identificirali najbolj pomembne koncepte, ki se nato uporabijo še za oblikovanje ostale hierarhije s pomočjo generalizacije in specializacije.

   2.2. Kodiranje
   Vključuje ekspliciten zajem pridobljenega znanja s formalnim jezikom.

   2.3. Integracija obstoječe ontologije
   Med procesom zajemanja in kodiranja ontologije je potrebno premisliti o uporabi in vključevanju že obstoječih ontologij.
3. **Ocenjevanje**
   Pri tej fazi avtorji tehnično presodijo ontologijo, povezano programsko okolje in dokumentacijo glede na specifikacijo zahtev.
4. **Dokumentacija**
   Priporočena so navodila za dokumentiranje ontologij, ki naj bi se razlikovala glede na različne tipe in namene ontologije.

Pomanjkljivosti metodologije Enterprise Ontology so odsotnost procesa upravljanja projekta in procesov, ki se odvijajo pred in po razvoju ontologije. Poleg tega so nekateri koraki, kot je zajemanje znanja, premalo specificirani.
Najpomembnejši projekt, razvit s to metodologijo, je Enterprise Ontology, ki zajema zbirko pojmov in definicij, primernih za podjetja.
Najpomembnejši orodji za podporo tej metodologiji sta Enterprise Toolset in Ontolingua Server. Enterprise Toolset uporablja arhitekturo na podlagi agentov, ki omogočajo integracijo in združljivost z drugimi orodji. Komponente orodja so: Procedure Builder za zajem procesnih modelov, Agent Toolkit za podporo razvoja agentov, Task Manager za integracijo in vizualizacijo procesov in Enterprise Ontology za komunikacijo.
Tudi ta metodologija je vplivala na poznejše pristope in tudi oblikovanje mnogih poznejših urejevalnikov ontologij [6, 11, 13, 18].

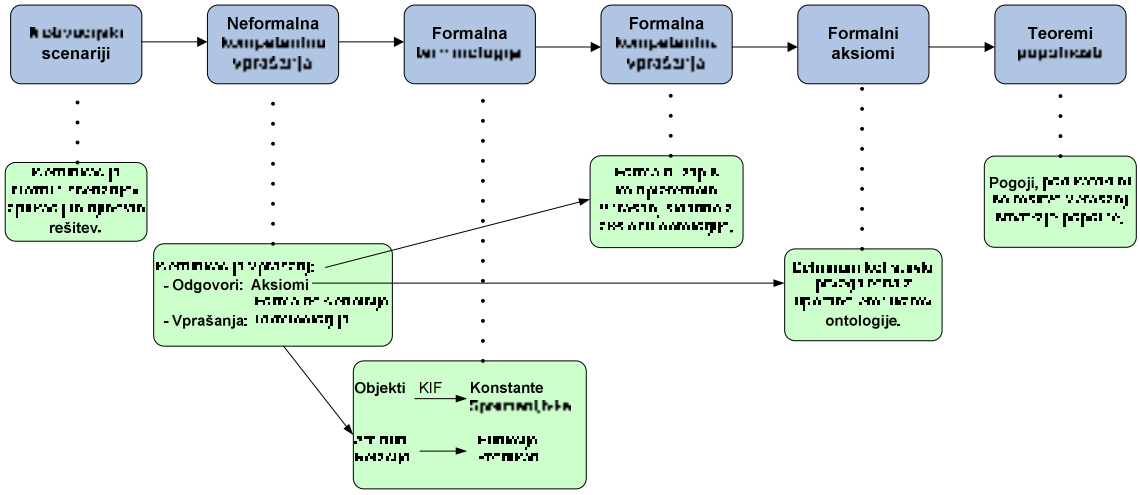
8.3.3 TOVE (metodologija Gruningerja in Foxa)

Ta metodologija je nastala na podlagi izkušenj iz projekta TOVE, kjer so razvili ontologijo znotraj domene modeliranja poslovnih procesov in aktivnosti.
Prvenstveno vključuje izgradnjo logičnega modela znanja, ki je pozneje opisan v sklopu ontologije. Ta model ni zgrajen direktno, ampak je najprej izdelan neformalen opis na podlagi začetne specifikacije ontologije, nato pa je ta opis formaliziran. Metodologija predlaga naslednje korake:

1. **Zajem motivacijskih scenarijev**
   Po mnenju Gruningerja in Foxa naj bi bil razvoj ontologij spodbujen s scenariji, ki izhajajo iz aplikacij. Ti scenariji so problemi ali primeri, ki še niso vključeni v obstoječih ontologijah. Takšen scenarij obenem ponuja tudi možne intuitivne rešitve problema. Te rešitve tvorijo neformalno semantiko za objekte in relacije, ki bodo pozneje vključene v ontologijo.
   Vsak osnutek nove ontologije ali razširitve ontologije naj bi opisoval enega ali več motivacijskih scenarijev ter pripadajoče rešitve.
2. **Formulacija neformalnih kompetentnih vprašanj**
   Ta vprašanja bazirajo na osnovi zajetih scenarijev v prejšnjem koraku in jih lahko obravnavamo kot izrazne zahteve v obliki vprašanj. Ontologija si mora biti sposobna predstavljati ta vprašanja s pomočjo svojih pojmov in mora biti sposobna tvoriti odgovore z uporabo aksiomov in definicij. To so neformalna kompetentna vprašanja, ker še niso izražena v formalnem jeziku ontologije.
   Kompetentna vprašanja so razslojena in odgovor na eno vprašanje se lahko uporabi za odgovor na bolj splošna vprašanja iz iste ali druge ontologije s pomočjo operacij kompozicije in dekompozicije. To je tudi način prepoznavanja že predstavljenega znanja za ponovno uporabo ali integracijo ontologij.
   Vprašanja služijo kot omejitve, kaj lahko ontologija vsebuje. Za razliko od drugih metodologij, kjer se osnuje določen model ontologije s pripadajočimi zahtevami. Z eno zbirko vprašanj pa ni nujno povezana samo ena ontologija. Kompetentna vprašanja se lahko uporabijo tudi za oceno, ali ontologija vključuje vse začetne zahteve.
3. **Določanje terminologije ontologije s formalnim jezikom**
   3.1. Zbiranje neformalne terminologije
      Ko so na voljo kompetentna vprašanja, lahko iz njih izluščimo uporabljene pojme. Ti pojmi bodo služili kot osnova za specifikacijo terminologije v formalnem jeziku.
   3.2. Specifikacija formalne terminologije
      Na osnovi določenih neformalnih kompetentnih vprašanj za novo ontologijo se določi terminologija ontologije s pomočjo jezika, kot je recimo KIF. Ti pojmi bodo pozneje v koraku 5 omogočali izražanje definicij in omejitev s pomočjo aksiomov.
4. **Oblikovanje formalnih kompetentnih vprašanj s pomočjo terminologije ontologije**
   Na podlagi neformalnih kompetentnih vprašanj in terminologije se formalno definirajo kompetentna vprašanja.
5. **Specifikacija aksiomov in definicij za pojme s pomočjo formalnega jezika**
   Aksiomi v ontologiji določajo definicije pojmov ontologije in omejitve pri njihovi razlagi. Preprosto določanje zbirke objektov ali zbirke osnovnih pojmov v logiki prvega reda zase ne določa ontologije. Aksiomi morajo biti določeni za definiranje semantike in pomena teh pojmov.
   Če so predlagani aksiomi nezadostni za predstavitev formalnih kompetentnih vprašanj in opis odgovorov na ta vprašanja, potem je potrebno ontologiji dodajati nove objekte in aksiome. Razvoj aksiomov ontologije je tako iterativen proces.

6. **Vzpostavitev pogojev za določanje popolnosti ontologije**
   Na osnovi formalno podanih kompetentnih vprašanj je potrebno definirati pogoje, pod katerimi so rešitve teh vprašanj popolne.

Tudi ta metodologija ne vključuje procesov pred in po razvoju ontologije ter procesov upravljanja projekta. Tehnike in aktivnosti niso opisane natančno. Je pa pristop uporabe kompetentnih vprašanj, na katere mora biti ontologija sposobna odgovoriti, zelo praktičen in priročen, še zlasti, če se z njim ukvarjajo domenski strokovnjaki, ki imajo manj znanja modeliranja. Zaradi tega je bil ta pristop uporabljen tudi v nekaterih poznejših metodologijah [6, 13].



Slika 14: Razvojne faze metodologije TOVE [6]

8.3.4 KACTUS

Metodologija je nastala pod vodstvom Bernarasa v sklopu projekta Esprit KACTUS. Eden izmed ciljev projekta KACTUS je raziskovanje možnosti ponovne uporabe znanja v kompleksnih tehničnih sistemih s pomočjo ontologij.
Pristop razvoja ontologij je pogojen z razvojem aplikacij. Tako se vedno ob novi aplikaciji zgradi tudi ontologija, ki vključuje znanje, potrebno za to aplikacijo. Ontologije so lahko razvite s pomočjo ponovne uporabe drugih ontologij, lahko pa so vključene tudi v ontologije poznejših aplikacij. Vsakokrat, ko je razvita nova aplikacija, se izvedejo naslednji koraki:

1. Specifikacija aplikacije, ki predstavi kontekst aplikacije in pogled na komponente, ki jih skuša aplikacija oblikovati.
2. Osnovni načrt, ki bazira na relevantnih kategorijah visokonivojskih ontologij. Tu se uporablja seznam pojmov in opravil iz prejšnjega koraka za dosego različnih pogledov na globalni model v skladu z določenimi kategorijami visokonivojskih ontologij. Proces načrtovanja vključuje tudi iskanje ontologij, razvitih za druge aplikacije, ki so preurejene in razširjene za uporabo v novih aplikacijah.
3. Preurejanje in strukturiranje ontologije z namenom doseganja končnega modela. Princip minimalnega povezovanja lahko zagotovi, da moduli med seboj niso preveč odvisni, pa vendar čim bolj skladni. S tem dosežemo tudi največjo homogenost znotraj modulov.

Ta metodologija se je prvotno uporabljala v domeni električnih omrežij. Metodologija ni definirana v podrobnosti. Življenjski cikel ontologij, razvitih s to metodologijo, je podoben

življenjskemu ciklu aplikacije, na katero se nanaša. Zaradi tega je tudi proces razvoja ontologij aplikacijsko odvisen [6, 13, 18].

8.3.5 METHONTOLOGY

Metodologija METHONTOLOGY je bila razvita v laboratoriju za umetno inteligenco na univerzi v Madridu. Ogrodje metodologije omogoča konstruiranje ontologij na nivoju znanja. Za večjo uporabnost pa zajema nekaj idej iz bolj razvite discipline razvoja programske opreme. Razvojni proces je namreč osnovan na aktivnostih IEEE standarda za razvoj programske opreme. Te aktivnosti so razporejene v življenjskem ciklu METHONTOLOGY ontologij, ki zajema stopnje, skozi katere gre ontologija v svojem življenjskem času in aktivnosti, ki se morajo izvesti na vsaki stopnji.

Razvojni proces ontologij
Proces se nanaša na aktivnosti, ki se izvajajo pri gradnji ontologij. Aktivnosti delimo v tri kategorije:

- **Aktivnosti upravljanja z ontologijo:** vključujejo načrtovanje, kontrolo in zagotavljanje kakovosti. Načrtovanje zajema identifikacijo nalog, ki jih je potrebno opraviti, njihovo pripravo ter izračun potrebnega časa in virov (ljudje, programska oprema, strojna oprema). Aktivnost kontrole skozi celoten življenjski cikel ontologije zagotavlja, da se bodo aktivnosti izvedle, kot je bilo načrtovano, ter da ni odstopanj. Aktivnost zagotavljanja kakovosti pa preverja kakovost proizvodov metodologije (ontologija, programska oprema, dokumentacija).
- **Razvojno usmerjene aktivnosti ontologije:** razdeljene so v aktivnosti pred razvojem, aktivnosti razvoja in aktivnosti po razvoju.
  Pred razvojem se preuči okolje, v katerem se bo ontologija uporabljala, ter izvedljivost razvoja.
  Med razvojem se najprej v aktivnosti specifikacije navedejo razlogi za gradnjo ontologije, predvidena uporaba ontologije in njeni končni uporabniki. V aktivnosti konceptualizacije se domensko znanje strukturira v razumljive modele na nivoju znanja. Formalizacija te konceptualne modele pretvori v formalne modele. Na koncu se formalni modeli implementirajo v aktivnosti implementacije.
  Po razvoju aktivnost vzdrževanja skrbi za posodabljanje in morebitne popravke ontologij.
- **Podporne aktivnosti:** izvajajo se sočasno z razvojno usmerjenimi aktivnostmi. Delijo se na pet aktivnosti. Aktivnost zajemanja znanja ima cilj pridobiti znanje od raznih strokovnjakov ali s pomočjo avtomatskega učenja ontologij. Vrednotenje ocenjuje razvite ontologije, programsko opremo in dokumentacijo, če so v skladu z zahtevami. Aktivnost integracije se odvija v primeru, če se ponovno uporabljajo druge ontologije ali celo več ontologij, če se ontologije spajajo ali združujejo. Spajanje ustvari novo ontologijo iz drugih, združevanje pa samo vzpostavi povezave med obstoječimi ontologijami in tako ohranja njihovo obliko. Aktivnost dokumentiranja zajema razlago vsake dokončane stopnje in izdelka, aktivnost upravljanja s konfiguracijami pa zajema verzije ontologij, programske opreme in dokumentacije z namenom kontrole sprememb.

Proces razvoja predstavlja aktivnosti, ki morajo biti izvedene, vendar pa ne opisuje, kdaj katera aktivnost nastopi in kako si morajo slediti. To je določeno v drugem delu metodologije, življenjskem ciklu ontologije, ki vzpostavlja stopnje in aktivnosti v življenjskem obdobju ontologije [13, 26].

Življenjski cikel ontologije

Življenjski cikel ontologije razporeja aktivnosti razvoja. Življenjski cikel je cikličen in bazira na razvijajočih se prototipih. To omogoča inkrementalni razvoj ontologij, kar pa dovoljuje tudi bolj zgodnje preverjanje skladnosti in preurejanje ontologije v razvoju. Vsak cikel se začne z načrtovanjem aktivnosti, ki zajemajo načrtovanje potrebnih opravil in virov. Zatem se začnejo aktivnosti razvoja, začenši s specifikacijo. Vzporedno se začnejo odvijati tudi aktivnosti upravljanja, kontrola in zagotavljanje kakovosti, ter podporne aktivnosti, zajemanje znanja, integracija, vrednotenje, dokumentiranje in upravljanje konfiguracij.

V vsakem ciklu gre prototip ontologije skozi vse razvojne aktivnosti (specifikacija, konceptualizacija, formalizacija, implementacija in vzdrževanje). Vsakemu ciklu sledi nov, ki upošteva zaključke vrednotenja oziroma ocenjevanja, dokler ne postane prototip dovolj zrel, da zadošča zahtevam vrednotenja. V vsakem razvojnem ciklu se odvijajo naslednji koraki:

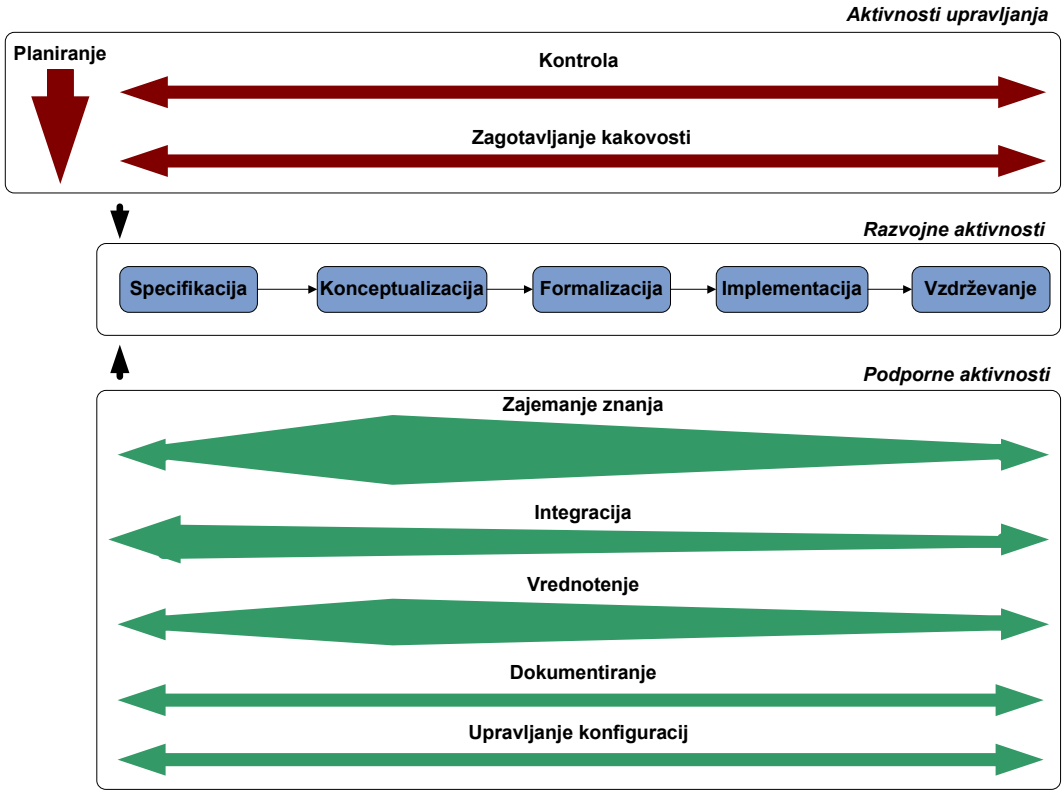
- specifikacija prototipa,
- izgradnja konceptualnega modela iz znanja, pridobljenega v sklopu aktivnosti zajemanja znanja, ki se večinoma odvija med konceptualizacijo,
- formalizacija konceptualnega modela,
- implementacija formaliziranega konceptualnega modela (to se lahko izvede avtomatično, če je omogočen avtomatičen prevod iz formalizacije v jezik za implementacijo ontologije),
- vzdrževanje pridobljene ontologije, kar lahko privede do novega razvojnega cikla, če zahteve še niso dosežene ali pa se pojavijo nove.

Kot je bilo povedano, se podporne aktivnosti in aktivnosti upravljanja odvijajo vzporedno z razvojnimi aktivnostmi. Vključenost podpornih aktivnosti pa ni enakomerna skozi celoten življenjski cikel ontologije. Aktivnosti zajemanja znanja, integracije in vrednotenja se bolj intenzivno izvajajo med fazo konceptualizacije ontologije, ki je tudi glavna faza metodologije (slika 15). Na začetku razvoja se namreč zajame največ znanja, ontologije se integrirajo na konceptualnem nivoju pred implementacijo, boljše pa je tudi v fazi konceptualizacije izvesti natančno vrednotenje, s čimer se lažje izognemo poznejšim napakam.

Vse povezave med aktivnostmi, ki so bile opisane do sedaj, sodijo med notranje odvisnosti znotraj istega razvojnega procesa ontologije. Na drugi strani pa obstajajo tudi odvisnosti med aktivnostmi razvojnih procesov različnih ontologij. Pri tem lahko govorimo o prepletanju življenjskih ciklov ontologij. Te odvisnosti pridejo do izraza predvsem pri ontologijah znotraj iste domene, ko prihaja do ponovne uporabe, integriranja, spajanja in združevanja ontologij [26].

Metodologija METHONTOLOGY se uporablja v različnih domenah za razvoj mnogih ontologij in za podporo mnogim aplikacijam.

Glavno orodje za podporo metodologije METHONTOLOGY je WebODE, za njeno implementacijo pa se lahko uporabljajo tudi orodja ODE, Protege in OntoEdit.



Slika 15: Razvojni proces in življenjski cikel metodologije METHONTOLOGY [26]

8.3.6 SENSUS

Ontologija SENSUS

Ta ontologija se uporablja za procesiranje naravnih jezikov. Obsega široko osnovano konceptualno strukturo za razvoj strojnih prevajalnikov. Vsebina je bila zbrana s povzemanjem in spajanjem informacij iz različnih elektronskih virov znanja. Baza ontologije je bila razvita s spajanjem dveh visokonivojskih lingvističnih ontologij in semantičnih kategorij slovarja. Pozneje sta bila dodana še španski in japonski besednjak.

SENSUS obsega več kot 70.000 konceptov, ki so organizirani v hierarhijo glede na njihov nivo abstrakcije. Vključuje tako pojme na visokem in srednjem nivoju abstrakcije, na splošno pa ne vključuje pojmov iz specifičnih domen. Za izgradnjo domenskih ontologij se domenski pojmi povezujejo z ontologijo SENSUS, nepotrebni pojmi v ontologiji SENSUS pa se odstranijo.

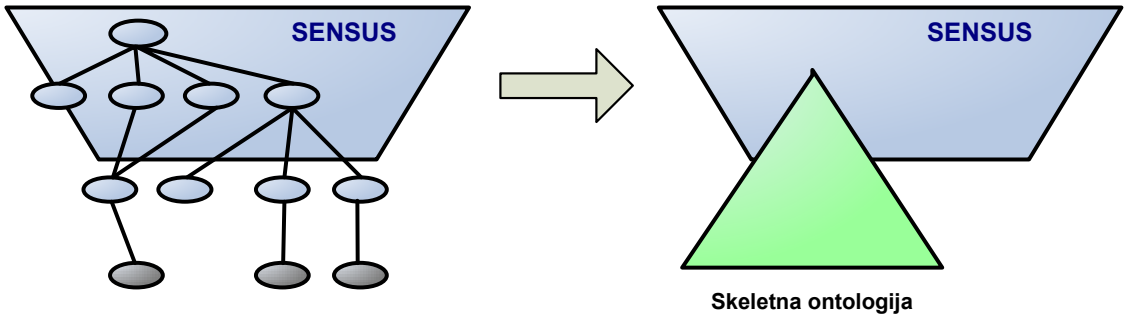
Razvojni proces metodologije SENSUS

Metodologija uporablja koncept od sredine navzven za izpeljevanje domenskih ontologij iz večje ontologije. Koraki, ki nastopijo pri razvoju ontologije v določeni domeni, so naslednji:

1. Za osnovo (seme) se vzame zbirka pojmov;
2. Pojmi se ročno povežejo z ontologijo SENSUS;
3. Vključeni so vsi koncepti in poti od osnovnih pojmov do korena ontologije SENSUS;
4. Vključijo se še dodatni pojmi, ki so relevantni znotraj določene domene;

5. Na koncu se za vozlišča, skozi katera gre veliko poti, vključi v ontologijo njihovo celotno poddrevo. To je osnovano na ideji, da v primeru, ko je veliko vozlišč poddrevesa relevantnih, potem so zelo verjetno relevantna tudi ostala vozlišča poddrevesa. Ta korak se izvaja ročno, saj za sprejemanje odločitev potrebuje nekaj razumevanja o domeni. Očitno bo skozi visokonivojska vozlišča ontologije šlo vedno mnogo poti, vendar je malokrat primerno vključiti njihova celotna poddrevesa.

Ta pristop spodbuja tudi skupno rabo znanja, saj je enaka osnovna ontologija uporabljena za razvoj ontologij v določenih domenah.
Metodologija SENSUS ima tako, glede na druge metodologije, svojevrsten pristop k razvoju ontologij. Metodologija ni opisana v podrobnosti in tudi ne obsega aktivnosti pred in po razvoju ter aktivnosti upravljanja. Strategija razvoja je delno odvisna od aplikacije, saj se osnovni pojmi za razvoj nove ontologije določajo na osnovi aplikacije.
S pomočjo metodologije SENSUS je bila razvita ontologija za načrtovanje zračnega bojevanja za vojsko. Vsebuje seznam osnovnih elementov, ki opisujejo bojne načrte, vključno s scenariji, udeleženci in poveljniki. Ta ontologija vključuje tudi ontologije orožja, splošnih sistemov, goriv itd. [6, 13, 18, 33]



Slika 16: SENSUS pristop k razvoju ontologij [6]

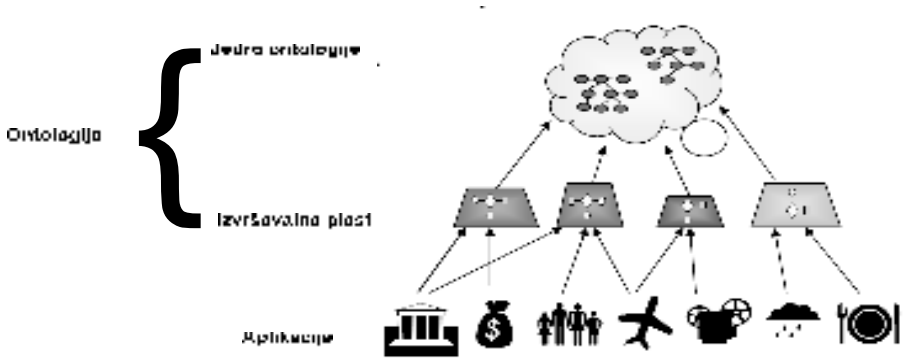
8.3.7 CommonKADS

CommonKADS ni prava metodologija za razvoj ontologij. Razvita je bila v sklopu evropskega projekta ESPRIT. Pokriva poglede od upravljanja z znanjem, analize in inženiringa znanja, do oblikovanja in implementacije informacijskih sistemov na osnovi znanja. CommonKADS je osredotočen bolj na začetne faze razvoja aplikacij za upravljanje z znanjem. Za razvijalce programske opreme ponuja standard za razvoj sistemov, ki zagotavljajo visokokakovostne rešitve na podlagi ponovno uporabljivih komponent. Upravljalcem z znanjem pa ponuja CommonKADS metode za kreiranje podrobnih opisov nalog znotraj celotnega poslovnega procesa in tehnike za izčrpno analizo, razvoj in shranjevanje znanja. CommonKADS uporablja tudi notacijo, ki je združljiva z jezikom UML. Metodologijo CommonKADS podpirajo orodja PC PACK in KADS22, ki omogočata sodelovanje inženirjev znanja in domenskih strokovnjakov. To metodologijo smo omenili, ker je služila kot osnova nekaterim drugim metodologijam za razvoj ontologij, kot je metodologija On-To-Knowledge [11, 33].

8.3.8 DOGMA

Metodologija DOGMA je pristop k razvoju ontologij razvila na podlagi podatkovnih baz. Ontologije tu delimo na dva dela:

- **Jedro ontologije:** obsega možne konceptualizacije domene iz realnega sveta. Sestavljeno je iz atomarnih predikatov ali tako imenovanih leksonov. Leksoni so izraženi s trojčki: <Pojem1, Funkcija, Pojem2>.
- **Izvrševalna plast:** sestavljajo jo pravila in omejitve, ki služijo za komunikacijo z aplikacijami. Za svojo osnovo uporabljajo pojme iz jedra ontologije. Ta plast služi kot posrednik med aplikacijami in ontologijo. Tako se lahko za različne aplikacije pripravijo različni interpretacijski pogledi na ontologijo.

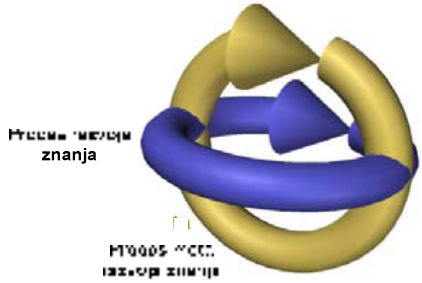


Slika 17: DOGMA pristop k razvoju ontologij [27]

Za podporo metodologiji DOGMA sta bili razviti tudi orodji DOGMA Server in DOGMA Modeler. DOGMA Server je namenjen shranjevanju, DOGMA Modeler pa modeliranju in urejanju jedra ontologije in izvrševalne plasti [27].

8.3.9 On-To-Knowledge

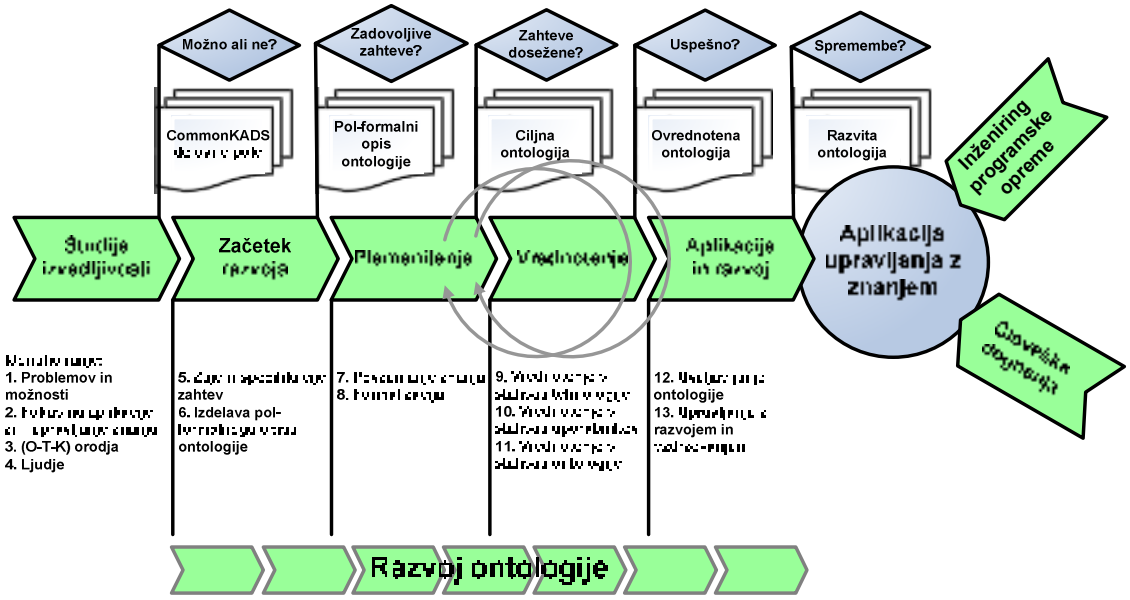
Metodologija On-To-Knowledge je bila razvita v sklopu evropskega projekta On-To-Knowledge za predstavitev in vzdrževanje aplikacij, ki upravljajo z znanjem na podlagi ontologij v podjetjih. Metodologija se v splošnem deli na dva tipa procesov, razvoj znanja in metarazvoj znanja (slika 18). Pri tem se proces razvoja znanja ukvarja predvsem z uporabo ontologij, medtem ko se metarazvoj znanja ukvarja z začetnim razvojem in vzpostavitvijo ontologij.



Slika 18: Ortogonalna procesa razvoja metodologije On-To-Knowledge [33]

Proces metarazvoja znanja

Na sliki 19 je prikazan proces metarazvoja znanja, ki je sestavljen iz petih glavnih korakov. Vsak korak vsebuje številne podkorake, na koncu zahteva odločitev o zadostitvi zahtevam ter specifične rezultate. Glavni tok predstavlja korake (faze), ki vodijo do aplikacije za upravljanje znanja na podlagi ontologije. Te faze so: študija izvedljivosti, začetek razvoja, plemenitenje, vrednotenje ter aplikacija in razvoj. Pod vsako fazo so navedeni najpomembnejši podkoraki. Vsaka zastavica z dokumenti nad fazami označuje glavne rezultate oziroma izdelke koraka. Odločitvene točke nad zastavicami označujejo najpomembnejše odločitve, ki morajo biti sprejete pred napredovanjem procesa v naslednjo fazo. Rezultati oziroma produkti posamezne faze po navadi služijo za podlago odločitvi. Faze plemenitenje, vrednotenje ter aplikacija in razvoj se tipično izvajajo v iterativnih ciklih. Na razvoj takšnih aplikacij imajo vpliv tudi proces inženiringa programske opreme ter človeška dognanja [6, 33].

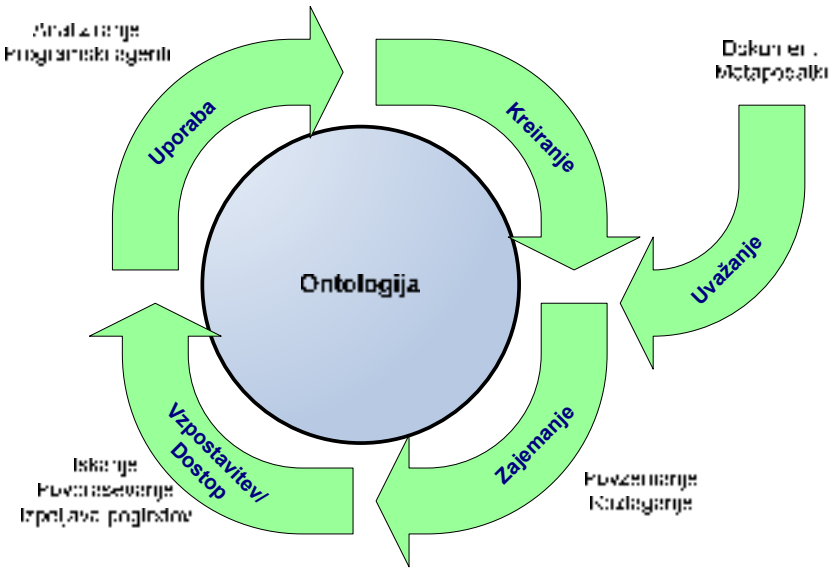


Slika 19: Koraki metaprocesa razvoja metodologije On-To-Knowledge [33]

Proces razvoja znanja

Ko je aplikacija za upravljanje znanja popolnoma implementirana v organizaciji, pa procesi razvoja znanja pravzaprav krožijo po naslednjih korakih:

- Kreiranje znanja ter morebitno vzporedno uvažanje dokumentov in metapodatkov. Pri tem morajo biti vsebine prilagojene tako, da zadoščajo pravilom podjetja oziroma infrastrukturi upravljanja z znanjem.
- Nato mora biti znanje zajeto, pri čemer se pojasni pomembnost ali povezanost znanja z obstoječim znanjem, kreirajo pa se tudi povezovalni metapodatki.
- Vzpostavitev znanja in omogočen dostop do njega mora zadovoljiti preprosta povpraševanja.
- Uporaba znanja pa zajema tekočo uporabo znanja v podjetjih za analize, povzemanja ter njegovo uporabo v programskih agentih.



Slika 20: Proces razvoja znanja metodologije On-To-Knowledge [33]

Ostala dognanja projekta On-To-Knowledge:

- napotki domenskim strokovnjakom morajo biti izredno strokovni,

sodelovanje pri inženiringu ontologij zahteva fizično prisotnost in napredno podporo orodij,

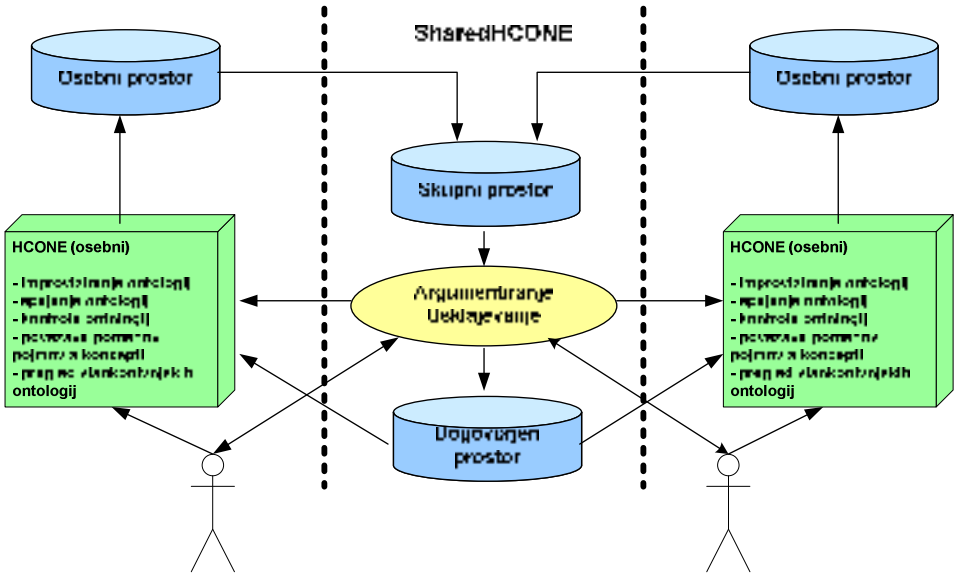
- Intenzivno razmišljanje o razvoju je v veliko pomoč v zgodnejših fazah inženiringa ontologij, še zlasti za domenske strokovnjake s slabšim znanjem modeliranja [6, 33].

On-To-Knowledge metodologija je zelo podrobno opisana, vključuje tudi aktivnosti upravljanja projekta ter aktivnosti pred in po razvoju. Pri svojem razvoju se je opirala tudi na že razvite metodologije, kot so: CommonKADS, ENTERPRISE in TOVE.
OntoEdit je orodje, ki omogoča izčrpno podporo metodologiji On-To-Knowledge. Vključuje napredno okolje za inženiring ontologij, ki podpira vse faze metodologije.

8.3.10 HCOME

Metodologija HCOME zagovarja dinamičen proces razvoja ontologij, ki se po navadi začne z neko grobo osnovo ontologije, ta pa je pozneje popravljena, izboljšana, obogatena in dopolnjena s podrobnostmi. Razvoj ontologije mora biti podprt skozi njen celoten življenjski cikel, ki se na koncu odraža v uporabni ontologiji. Metodologija skuša spodbujati strokovnjake na področju upravljanja znanja h kontinuiranemu urejanju svojih formalnih konceptualizacij pri vsakodnevnih opravilih. Tu je poudarek na decentraliziranem razvoju in vrednotenju ontologij v kontekstu večjih skupin strokovnjakov za upravljanje znanja.
Metodologijo podpirata orodji HCONE in SharedHCONE, ki predstavljata okolje za decentraliziran razvoj ontologij in spodbujata sodelovanje med strokovnjaki in kontinuirano delo. Tu so na voljo trije prostori za shranjevanje ontologij:

- **Osebni prostor:** tu lahko uporabniki kreirajo in združujejo ontologije, kontrolirajo verzije ontologij in povezujejo pojme s koncepti.
- **Skupni prostor:** tu se lahko shranjujejo razvijajoče ontologije in so na voljo skupni rabi. Skupni prostor je na voljo vsem sodelujočim.
- **Dogovorjeni prostor:** po razpravah in dogovarjanjih o ustreznosti ontologije se usklajena ontologija prenese v dogovorjen prostor.



Slika 21: Decentraliziran model razvoja ontologij HCOME [12]

Končni cilj metodologije je tako razvoj usklajene in razumljive ontologije za učinkovito upravljanje z znanjem. Za doseganje tega pa morajo biti opravila za upravljanje ontologij vključena v cikle modeliranja in črpanja informacij. Stopnje in opravila življenjskega cikla ontologij metodologije HCOME so predstavljeni v tabeli 2. Razvoj ontologij v sklopu te metodologije je torej iterativen in kontinuiran [11, 12].

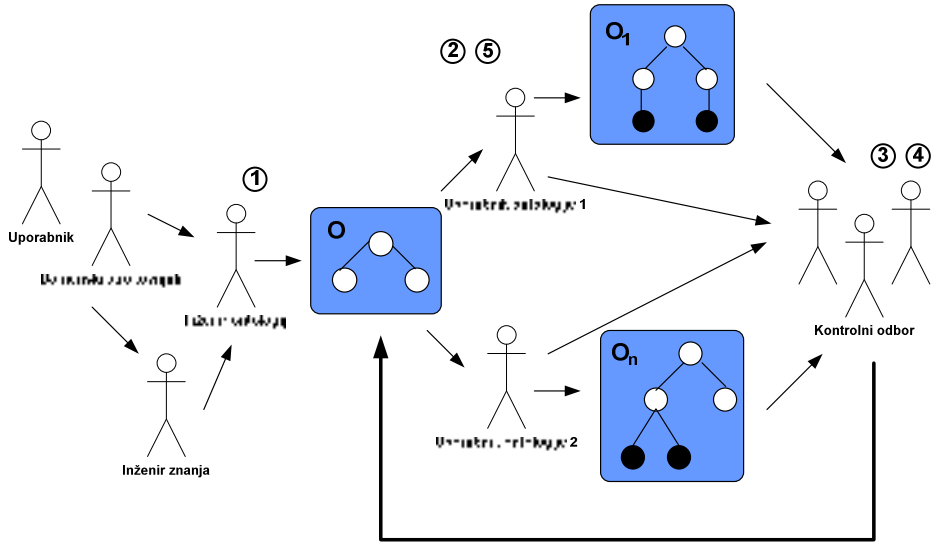
Faze življenjskega cikla ontologije	Procesi	Opravila
Specifikacija	Definiranje ciljev in področja ter iskanje virov znanja	• Razprava o potrebah (S) • Izdelava dokumentov (S) • Izbor sodelavcev • Specifikacija področja in ciljev ontologije (S)
Konceptualizacija	Pridobivanje znanja	• Zajem in knjižnic ontologij (O) • Pregled višjenivojskih ontologij (O) • Posvetovanje z domenskimi strokovnjaki (S)
	Razvoj in vzdrževanje ontologije	• Improviziranje (O) • Upravljanje konceptualizacij (O) • Združevanje različnih verzij (O) • Primerjava lastnih verzij (O) • Generalizacija/specializacija verzij (O) • Dodajanje dokumentacije (O)
Izkoriščanje	Uporaba ontologije	• Iskanje po ontologijah (O) • Uporaba v aplikacijah
	Vrednotenje ontologije	• Skupinsko obravnavanje kritik (S) • Primerjava različnih verzij (S) • Uporaba dogovorjenih ontologij (S) • Vključevanje izvlečkov razprav v ontologije (S)

Tabela 2: Življenjski cikel ontologij metodologije HCOME [12]

Opravila v tabeli se izvajajo v sklopu osebnih prostorov razvijalcev (označena z (O)) ali v sklopu skupnih prostorov (označena s (S)).

8.3.11 DILIGENT

Procesni model DILIGENT je bil izdelan za scenarije dinamične souporabe znanja, ki potrebujejo procesni model za obvladovanje pogosto spreminjajočih uporabniških potreb. Pri razvoju ontologij v metodologiji DILIGENT sodeluje veliko strokovnjakov z različnim in med seboj dopolnjujočim znanjem. Vsi skupaj pa so vključeni v gradnjo iste ontologije. V decentraliziranih okoljih, kot je semantični splet, lahko ti strokovnjaki pripadajo celo konkurenčnim organizacijam in so geografsko razpršeni. Pogosto so domenski strokovnjaki, ki sodelujejo pri gradnji ontologij, tudi njeni uporabniki, vseeno pa je končna skupina uporabnikov veliko večja od razvijalcev.
Osnovna ontologija se zgradi v okviru manjše skupine razvijalcev. Ta ontologija je nato na voljo vsem in uporabniki imajo možnost lokalne uporabe in spreminjanja za lastne potrebe. Centralni odbor razvijalcev pa vzdržuje in skrbi za kakovost osnovne ontologije, hkrati pa je odgovoren tudi za nadgrajevanje. Nadgrajevanje je tipično posledica lokalnih sprememb, ki jih opravijo uporabniki, in se nato prenesejo na osnovno ontologijo. Ontologija se tako razvija preko predlaganih sprememb pod nadzorom centralnega odbora.



Slika 22: Proces decentraliziranega razvoja ontologij metodologije DILIGENT [10]

Razvoj ontologij z DILIGENT metodologijo je sestavljena iz petih glavnih korakov:

1. **Gradnja:** proces se začne z gradnjo začetne ontologije, pri kateri sodeluje manjša skupina domenskih strokovnjakov, uporabnikov in inženirjev ontologij. Pri tem skušajo najti majhno in sporazumno prvo verzijo skupne ontologije.
2. **Lokalno prilagajanje:** ko je osnovna ontologija na voljo, jo lahko uporabniki uporabljajo ter jo prilagajajo lastnim potrebam. Pri tem lahko prosto spreminjajo lokalno verzijo ontologije, ne pa tudi skupne ontologije, za katero pa kontrolni odbor zbira predloge za spremembe ter beleži lokalne prilagoditve.
3. **Analiza:** kontrolni odbor analizira uporabniške predloge za spremembe in skuša najti podobnosti med ontologijami uporabnikov. Pri tem je glavna aktivnost odločanje, katere spremembe bodo dodane naslednji verziji ontologije. Poizkuša se najti ravnotežje med potrebami in zahtevami uporabnikov.
4. **Popravek ontologije:** kontrolni odbor mora redno popravljati skupno ontologijo, da se lokalne verzije ne oddaljijo preveč od skupne. Cilj nadgrajevanja oziroma popravljanja ontologije je tako čim bolj izenačiti ontologijo s potrebami uporabnikov

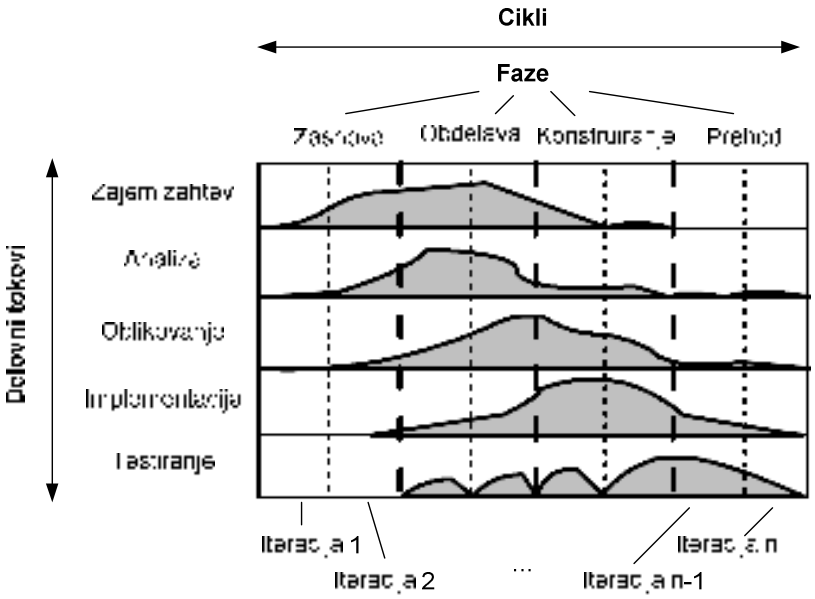
in s tem dosego večje sprejemljivosti skupne ontologije. Pri tem sodelujejo uporabniki s predlogi za spremembe, domenski strokovnjaki potrdijo skladnost spremembe z domeno, inženirji ontologij pa skrbijo za tehnično rešitev spremembe.

5. **Lokalno posodabljanje:** ko je na voljo nova verzija skupne ontologije, lahko uporabniki posodobijo svojo lokalno verzijo in s tem uporabljajo na novo predstavljeno znanje [10, 11].

8.3.12 UPON

UPON je inkrementalna metodologija za razvoj ontologij, ki je osnovana na poenotenem procesu za razvoj programske opreme (Unified Software Development Process) in podprta z modelirnim jezikom UML. Od ostalih metodologij se razlikuje po tem, da je iterativna, inkrementalna in razvoj bazira na primerih uporabe. Osnovanje razvoja na primerih uporabe ima namen izdelati ontologije, ki bodo čim bolj služile uporabnikom, tako ljudem kot tudi programskim agentom. Narava razvojnega procesa je iterativna, kar omogoča razvijalcem, da se osredotočijo na določen del ontologije, in inkrementalna, saj se ontologija skozi proces razširja in vedno bolj osredotoča na podrobnosti.

Razvojni proces se ponavlja skozi serijo ciklov, ki vodijo do končne ontologije. Metodologija UPON je sestavljena iz ciklov, faz, iteracij in delovnih tokov. Vsak cikel je sestavljen iz štirih faz: zasnova, obdelava, konstruiranje in faza prehoda. Rezultat cikla je nova verzija ontologije. Vsaka faza se naprej deli na iteracije. Med vsako iteracijo se odvija pet delovnih tokov: zajem zahtev, analiza, oblikovanje, implementacija in testiranje. Začetne faze so bolj osredotočene na zajem zahtev, poznejše iteracije pa razširjajo in dopolnjujejo ontologijo. Na sliki 23 je predstavljen razvojni proces metodologije UPON.



Slika 23: Razvojni proces metodologije UPON [15]

Delovni tokovi metodologije UPON

1. Zajem zahtev je proces definiranja znanja, ki naj bo zajeto v ontologiji. Glavni cilj pri tem je doseči sporazum med oblikovalci, inženirji znanja in končnimi uporabniki. Koraki pri tem so:
    - definiranje interesne domene,
    - definiranje razloga (motivacijskega scenarija) in namena za razvoj,
    - definiranje področja in obsega delovanja,
    - identificiranje kompetentnih vprašanj,
    - izdelava diagramov primerov uporabe.
2. Analiza vključuje pregled in strukturiranje zahtev, zajetih v prejšnjem delovnem toku. Koraki pri tem so:
    - preučevanje morebitne uporabe že obstoječih virov,
    - identifikacija relevantnih pojmov,
    - definiranje konceptov.

    Rezultat tega delovnega toka je model analize.
3. Oblikovanje, katerega koraki so:
    - strukturiranje konceptov in relacij (kreiranje razrednih diagramov),
    - realizacija primerov uporabe (kreiranje diagramov zaporedja in sodelovanja).
4. Implementacija zajema formalizacijo ontologije v izbranem jeziku in implementacijo po komponentah. Komponente so lahko formalizirane v različnih jezikih in notacijah, če le imajo dovolj veliko izrazno moč. Rezultat delovnega toka je model implementacije.
5. Testiranje omogoča preverjanje, ali so v ontologiji pravilno zajete vse specifikacije. Pri tem je razvitih več testnih primerov, ki testirajo skladnost z zahtevami [15].

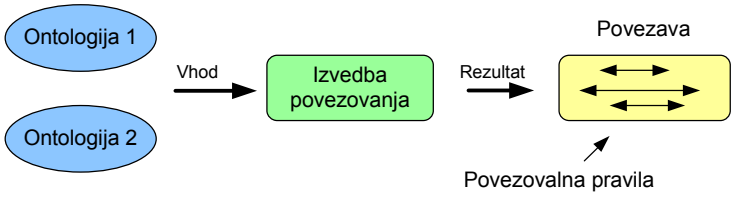
V primerjavi z IEEE standardom razvoja programske opreme metodologija UPON vključuje podporo procesom razvoja, dokumentiranja, vrednotenja (s primeri testiranjem primerov uporabe) in zajemanja zahtev. Ne vključuje pa procesa upravljanja projekta in procesov pred in po razvoju ontologij.
Prednost metodologije UPON je v podpori mnogih orodij za načrtovanje v jeziku UML, kot sta Rational Rose in Microsoft Visio.

8.4 Pristopi posredovanja med ontologijami

Ontologije so specifikacije znanja za skupno rabo, zato bi lahko bile iste ontologije uporabljene za opisovanje različnih podatkovnih virov, kot so spletne strani, XML dokumenti in relacijske baze. Toda to ne rešuje problema v celoti, saj je težko pričakovati, da bi se kdaj med ljudmi ali organizacijami osnovala enotna skupna terminologija ali ontologija. Pričakovati je pojav različnih ontologij, za njihovo vzajemno delovanje pa je potrebno uskladiti razlike med njimi. Usklajevanju teh razlik pravimo posredovanje med ontologijami.
Posredovanje med ontologijami omogoča ponovno uporabo znanja med aplikacijami na semantičnem spletu in tudi sodelovanje med različnimi organizacijami. V kontekstu upravljanja s semantičnim znanjem je posredovanje med ontologijami pomembno za omogočanje skupne rabe podatkov med heterogenimi bazami znanja in za omogočanje aplikacijam ponovno uporabo podatkov iz različnih baz znanja. Še eno pomembno področje posredovanja med ontologijami pa so semantične spletne storitve. Tisti, ki nudi storitev, in tisti, ki po njej povprašuje, uporabljata pogosto različno terminologijo, zato je posredovanje nujno za omogočanje komunikacije med različnimi poslovnimi partnerji.

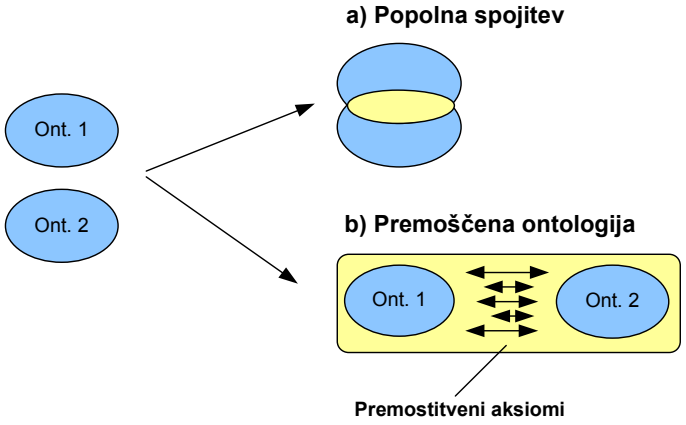
Pristopi posredovanja med ontologijami so:

- **Povezovanje ontologij** je specifikacija semantičnega prekrivanja dveh ontologij. Povezava oziroma skladnost med različnimi entitetami dveh ontologij se po navadi izraža z aksiomi, ki so zapisani v specifičnem jeziku povezovanja. Tri glavne faze povezovalnega procesa so: iskanje povezav, predstavitev povezav in izvedba povezovanja. Eno izmed orodij za podporo povezovanja ontologij je OntoMap.



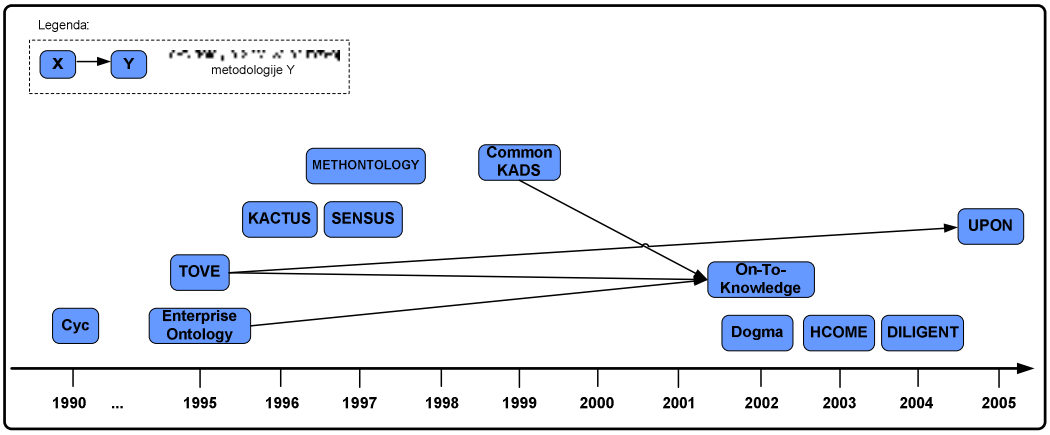
Slika 24: Povezovanje ontologij [11]

- **Uravnavanje ontologij** je proces iskanja podobnosti med dvema ontologijama. Rezultat primerjave je specifikacija podobnosti med ontologijama. Uravnavanje ontologij se pogosto izvaja avtomatično s pomočjo programskih orodij.
- **Spajanje ontologij** je kreiranje ene ontologije iz dveh obstoječih. Novonastala ontologija poenoti in zamenja prejšnji ontologiji. Obstajata dva pristopa spajanja. V prvem pristopu predstavljata vhod v proces dve ontologiji, rezultat procesa pa je nova, spojena ontologija, ki zajema obe originalni ontologiji. V drugem pristopu pa je rezultat premoščena ontologija, ki vključuje originalni ontologiji ter skladnosti med njima specificira s pomočjo premostitvenih aksiomov. Eno izmed orodij za spajanje ontologij je OntoMerge [11].



Slika 25: Spajanje ontologij [11]

8.5 Kronološki pregled razvoja metodologij



Slika 26: Kronološki pregled razvoja metodologij za razvoj ontologij

Pri kronološkem pregledu razvoja metodologij velja omeniti, da sta metodologiji TOVE in Enterprise Ontology orali ledino metodologij za razvoj ontologij in vplivali na razvoj več poznejših metodologij. Poleg tega pa se je večina metodologij zgledovala tudi po bolj razviti disciplini razvoja programske opreme, kjer so bile metodologije razvoja že bolj podrobno določene in posamezni koraki bolj razčlenjeni. Pri tem je imel največji vpliv standard IEEE 1074-1995, ki opisuje procese razvoja programske opreme. Razvoj metodologije UPON pa se je pozneje opiral na proces razvoja programske opreme Unified Software Development Proces.

8.6 Primerjava metodologij

Primerjava metodologij je bila opravljena s pomočjo programa DEXi, ki omogoča primerjavo variant preko drevesa kriterijev in omogoča tudi izdelavo poročil primerjave. V primerjavo so bile vključene vse opisane metodologije z izjemo metodologije CommonKADS, ki ni tipična metodologija za razvoj ontologij. Kriteriji za primerjavo metodologij so bili opisani v poglavju: Ogrodje za ocenjevanje metodologij. Glavni del kriterijev sodi v sklop načrta procesa razvoja ontologije in izhaja iz IEEE 1074-1995 standarda za proces razvoja programske opreme. Temu je dodan še sklop kriterijev o strategiji gradnje in pa kriterija razširjenosti uporabe metodologije in podpore programskih orodij.

8.6.1 Predstavitev in razlaga uteži kriterijev

Kriteriji ocenjevanja metodologij so razvrščeni v drevesno strukturo, vsak kriterij pa ima določeno utež, ki predstavlja vpliv tega kriterija na skupno oceno razvitosti metodologije. Lokalna vrednost uteži pomeni vpliv kriterija v sklopu ene veje drevesa kriterijev, globalna vrednost uteži pa pomeni vpliv kriterija na celotno oceno metodologije. Kriteriji in uteži so predstavljeni na sliki 27.

Kriterij	Lokalne	Globalne
Razvitost metodologije		
Strategija gradnje	6	6
Plan življenjskega cikla	100	6
Odvisnost razvoja od aplikacije	0	0
Strategija identificiranja konceptov	0	0
Uporaba jedrne ontologije	0	0
Definiranost procesa razvoja ontologije	66	66
Upravljanje projekta	10	7
Zasnova projekta	24	2
Nadzorovanje	38	3
Upravljanje kakovosti ontologije	38	3
Razvojno orientirani procesi	70	46
Procesi pred razvojem	17	8
Proučevanje okolja	50	4
Proučevanje izvedljivosti	50	4
Procesi razvoja	58	27
Obdelava zahtev	30	8
Oblikovanje	39	11
Implementacija	30	8
Procesi po razvoju	25	12
Namestitev	25	3
Delovanje	16	2
Podpora	20	2
Vzdrževanje	35	4
Upokojitev	4	1
Spremljevalni procesi	20	13
Zajem znanja	29	4
Vrednotenje	25	3
Upravljanje konfiguracij	11	2
Dokumentiranje	25	3
Usposabljanje	9	1
Uporaba metodologije	9	9
Podpora orodja	18	18

Slika 27: Kriteriji za ocenjevanje razvitosti metodologij

Skupna ocena metodologije je sestavljena iz sklopov strategije gradnje in definiranosti procesa razvoja ontologije ter kriterijev razširjenosti uporabe metodologije in podpore programskih orodij za razvoj ontologije. Pri tem igra največjo vlogo definiranost procesa razvoja ontologije, saj zajema vse faze gradnje ontologije. Na skupno oceno ima velik vpliv tudi obstoj programskega orodja, ki lahko razvijalcem olajša delo in prihrani čas.

V sklopu kriterijev strategije gradnje je edini kriterij, ki vpliva na oceno, obstoj načrta življenjskega cikla, saj so ostali kriteriji odvisni od pristopa metodologije k razvoju in ne kažejo razvitosti metodologij.

V sklopu kriterijev definiranosti razvoja ontologije so najbolj pomembni razvojno orientirani procesi, v sklopu katerih se aplikacija razvije, sledijo pa spremljevalni procesi, ki podpirajo razvojno orientirane procese.

V sklopu razvojno orientiranih procesov imajo največjo težo procesi razvoja. Pri spremljevalnih procesih so najpomembnejši procesi zajema znanja, vrednotenja in dokumentiranja, nekoliko manj pa procesa upravljanja konfiguracij in dokumentiranja.

8.6.2 Podatki o metodologijah

Zbrani podatki o metodologijah se nahajajo v tabeli 3. V sklopu kriterijev načrta procesa razvoja ontologije so možne vrednosti kriterijev »Ni predvideno«, »Predvideno« in »Opisano«. Vrednost »Opisano« pomeni, da je ta korak v metodologiji podrobno opisan z razdelitvijo na opravila in dodeljenimi odgovornostmi, kdo in kdaj je odgovoren za katero opravilo. Vrednost »Predvideno« pomeni, da je ta aktivnost zajeta in obravnavana v metodologiji, vendar ni podrobno opisana. Vrednost »Ni predvideno« pa pomeni, da metodologija tega koraka ne vključuje v svojem razvojnem procesu.

Lastnost				Cyc	Enterprise Ontology	TOVE	KACTUS	METHONTOLOGY	SENSUS	DOGMA	On-To-Knowledge	HCOME	DILIGENT	UPON
Strategija gradnje			Plan življenjskega cikla	Razvoj prototipov	Ni predvideno	Razvoj prototipov	Razvoj prototipov	Razvoj prototipov	Ni predvideno	Ni predvideno	Inkrementalno, razvoj prototipov	Inkrementalno, iterativno	Inkrementalno iterativno	Inkrementalno, iterativno
			Odvisnost razvoja od aplikacije	Ni odvisen	Ni odvisen	Delno odvisen	Odvisen	Ni odvisen	Delno odvisen	Odvisen	Odvisen	Ni odvisen	Ni odvisen	Odvisen
			Strategija identifikacije konceptov	Ni definirano	Od sredine navzven	Od sredine navzven	Od zgoraj navzdol	Od sredine navzven	Od sredine navzven	Od zgoraj navzdol	Odvisno od aplikacije	Od sredine navzven	Od sredine navzven	Od sredine navzven
			Uporaba jedrne ontologije	DA	Ne	Ne	Ne	Ne	Da	Da	Odvisno od virov projekta	Da	Da	Ne
Načrt procesa razvoja ontologije	Upravljanje projekta		Zasnova	Ni predvideno	Ni predvideno	Ni predvideno	Ni predvideno	Ni predvideno	Ni predvideno	Ni predvideno	Opisano	Ni predvideno	Ni predvideno	Ni predvideno
			Nadzorovanje	Ni predvideno	Ni predvideno	Ni predvideno	Ni predvideno	Predvideno	Ni predvideno	Ni predvideno	Opisano	Ni predvideno	Ni predvideno	Ni predvideno
			Upravljanje kakovosti ontologije	Ni predvideno	Ni predvideno	Ni predvideno	Ni predvideno	Predvideno	Ni predvideno	Ni predvideno	Opisano	Ni predvideno	Ni predvideno	Ni predvideno
	Razvojno orientirani procesi	Pred razvojem	Proučevanje okolja	Ni predvideno	Ni predvideno	Ni predvideno	Ni predvideno	Ni predvideno	Ni predvideno	Ni predvideno	Predvideno	Ni predvideno	Ni predvideno	Ni predvideno
			Proučevanje izvedljivosti	Ni predvideno	Ni predvideno	Ni predvideno	Ni predvideno	Ni predvideno	Ni predvideno	Ni predvideno	Opisano	Ni predvideno	Ni predvideno	Ni predvideno
		Razvojni procesi	Obdelava zahtev	Ni predvideno	Predvideno	Opisano	Predvideno	Opisano	Predvideno	Predvideno	Opisano	Predvideno	Predvideno	Opisano
			Oblikovanje	Ni predvideno	Ni predvideno	Opisano	Opisano	Opisano	Ni predvideno	Predvideno	Opisano	Opisano	Opisano	Opisano
			Implementacija	Predvideno	Predvideno	Opisano	Predvideno	Opisano	Opisano	Predvideno	Opisano	Predvideno	Predvideno	Opisano
		Po razvoju	Namestitev	Ni predvideno	Ni predvideno	Ni predvideno	Ni predvideno	Ni predvideno	Ni predvideno	Ni predvideno	Predvideno	Ni predvideno	Ni predvideno	Ni predvideno
			Delovanje	Ni predvideno	Ni predvideno	Ni predvideno	Ni predvideno	Ni predvideno	Ni predvideno	Ni predvideno	Opisano	Ni predvideno	Ni predvideno	Ni predvideno
			Podpora	Ni predvideno	Ni predvideno	Ni predvideno	Ni predvideno	Ni predvideno	Ni predvideno	Ni predvideno	Opisano	Ni predvideno	Ni predvideno	Ni predvideno
			Vzdrževanje	Ni predvideno	Ni predvideno	Ni predvideno	Ni predvideno	Predvideno	Ni predvideno	Ni predvideno	Predvideno	Predvideno	Predvideno	Ni predvideno
			Upokojitev	Ni predvideno	Ni predvideno	Ni predvideno	Ni predvideno	Ni predvideno	Ni predvideno	Ni predvideno	Ni predvideno	Ni predvideno	Ni predvideno	Ni predvideno
	Spremljevalni procesi		Zajem znanja	Predvideno	Predvideno	Predvideno	Ni predvideno	Opisano	Ni predvideno	Predvideno	Opisano	Predvideno	Predvideno	Opisano
			Vrednotenje	Ni predvideno	Predvideno	Predvideno	Ni predvideno	Opisano	Ni predvideno	Ni predvideno	Predvideno	Predvideno	Predvideno	Opisano
			Upravljanje konfiguracij	Ni predvideno	Ni predvideno	Ni predvideno	Ni predvideno	Opisano	Ni predvideno	Predvideno	Predvideno	Predvideno	Ni predvideno	Ni predvideno
			Dokumentiranje	Predvideno	Predvideno	Predvideno	Ni predvideno	Opisano	Ni predvideno	Ni predvideno	Opisano	Predvideno	Ni predvideno	Opisano
			Usposabljanje	Ni predvideno	Ni predvideno	Ni predvideno	Ni predvideno	Ni predvideno	Ni predvideno	Ni predvideno	Opisano	Ni predvideno	Ni predvideno	Ni predvideno
Uporaba metodologije				Delno razširjena	Na ožjem področju	Na ožjem področju	Delno razširjena	Na več področjih	Delno razširjena	Delno razširjena	Na več področjih	Na ožjem področju	Delno razširjena	Delno razširjena
Podpora orodij				Orodje Cyc	Ontolingua Server (delno)	Ni orodij	Ni orodij	WebODE, ODE, Protege (delno)	OntoSaurus (delno)	DOGMA Modeler, DOGMA Server	OntoEdit, OntoStudio (delno)	HCONE, SharedHCONE	OntoStudio (delno)	Rational Rose, Microsoft Visio

Tabela 3: Zbrani podatki o metodologijah po določenih kriterijih [6, 13, 18]

8.6.3 Rezultati primerjave metodologij in ocena razvitosti

V tabeli 4 so predstavljeni rezultati primerjave razvitosti metodologij glede na opisane kriterije in zbrane podatke iz tabele 3. Podrobni rezultati in podatki primerjave so prikazani v prilogi 1 (poročilo primerjave iz orodja DEXi).

Metodologija	Cyc	Enterprise Ontology	TOVE	KACTUS	METHONTOLOGY	SENSUS	DOGMA	On-To-Knowledge	HCOME	DILIGENT	UPON
Razvitost	Slabo razvita	Slabo razvita	Srednje razvita	Srednje razvita	Dobro razvita	Slabo razvita	Slabo razvita	Dobro razvita	Srednje razvita	Srednje razvita	Srednje razvita

Tabela 4: Rezultati primerjave metodologij

Iz tabele rezultatov je razvidno, da se je razvitost ontologij skozi čas izboljševala. Vedno večja pa je bila tudi podpora programskih orodij. Pri oceni razvitosti izstopata metodologiji METHONTOLOGY in On-To-Knowledge, ki sta ocenjeni kot dobro razviti, pokrivata največ korakov razvoja, njuna uporaba je razširjena in kar je zelo pomembno, imata tudi dobro podporo orodij. Metodologiji Cyc in Enterprise Ontology, prvi razviti metodologiji, imata razumljivo slabše opisan proces razvoja, temu primerna pa je tudi podpora orodij. Pri metodologiji TOVE je razvojni proces že podrobneje opisan, vendar, podobno kot za metodologijo KACTUS, ne obstaja podpornih orodij. KACTUS tudi ne obravnava nobenih spremljevalnih procesov. METHONTOLOGY metodologija je prva metodologija, ki ima podrobneje opisane tako razvojne kot tudi spremljevalne procese in dobro podporo orodja WebODE. Metodologiji SENSUS in DOGMA imata malo specifičen pristop k razvoju ontologij, vendar je njun razvojni proces slabše opisan in ne obravnava niti načrta življenjskega cikla metodologije. On-To-Knowledge metodologija vključuje in opisuje največ faz razvoja, pri čemer ima edina podrobneje opisan tudi proces upravljanja projekta. V okviru razvoja metodologije je nastalo tudi orodje OntoEdit, ki pokriva vse faze metodologije. Metodologije HCOME, DILIGENT in UPON so dokaj dobro razvite in pokrivajo razvojne in spremljevalne procese. Pri tem je prednost metodologij HCOME in DILIGENT v decentraliziranem pristopu razvoja ontologij, prednost metodologije UPON pa podpora dobro razvitih orodij UML.

Na splošno lahko rečemo, da nobena metodologija ne pokriva vseh vidikov inženiringa ontologij ter da obstaja še mnogo odprtih točk, ki bi jih bilo mogoče še izboljšati. Največja razlika v primerjavi z razvitejšimi metodologijami za razvoj programske opreme je tu opazna predvsem pri zelo slabi vključenosti procesov upravljanja projekta in procesov pred ter po razvoju ontologije. Tako je celoten proces razvoja ontologij večinoma osredotočen na procese razvoja (obdelava zahtev, oblikovanje, implementacija) in na potrebne spremljevalne procese, manjka pa podrobneje opisan načrt poznejše uporabe in vzdrževanja ontologij. Večinoma pa je v metodologijah definiran načrt življenjskega cikla razvoja, ki omogoča bolj sistematičen razvoj.

Vidimo, da je število različnih metodologij in s tem tudi različnih pristopov k razvoju veliko. Nekatere med njimi so odvisne od aplikacij (KACTUS, On-To-Knowledge, UPON), druge gradijo bolj splošne ontologije (METHONTOLOGY, HCOME, DILIGENT), nekatere imajo za osnovo neko visokonivojsko ontologijo (SENSUS, CYC), druge oblikujejo koncepte od začetka itd. To sicer po eni strani pomeni, da je možno izbirati najprimernejši pristop razvoja ontologije glede na okolje in domeno ontologije. Za gradnjo ontologije tako ni vedno najprimernejša najbolj razvita metodologija, ampak tista, ki najbolj ustreza naravi problema. Vendar pa razvoj metodologij ne bi smel iti v smeri, da se konstruirajo nove metodologije glede na potrebe aplikacij. Bolj bi bilo potrebno težiti k razvoju poenotenega modela metodologije. Takšen model bi lahko vseboval korake, kjer bi bilo mogoče izbrati med

različnimi metodami glede na obravnavan problem. Najbolj primeren način za dosego tega cilja bi bilo sodelovanje med različnimi razvojnimi skupinami.
Trenutno še ne obstaja veliko opisanih dobrih praks z metodologijami, ki bi opisovala posamezne razvojne dosežke. S tem sta razvoj in vzdrževanje ontologij odvisna predvsem od sposobnosti razvijalcev. Tudi za decentraliziran razvoj ontologij še ne obstaja podrobnejših navodil za lažje poenotenje različnih pogledov razvijalcev na isto domeno.
Naslednja slabost obstoječih metodologij je premajhna vključenost v širši procesni model, ki vključuje tudi ljudi človeške in tehnološke vidike upravljanja znanja. Ti vidiki so pomembni za bolj celovit razvoj upravljanja znanja. Eden izmed njih je, na primer, ocenjevanje stroškov za razvoj ontologij.

Če povzamemo, so glavne pomanjkljivosti obstoječih metodologij za razvoj ontologij naslednje:

- podroben opis vseh faz,
- vzdrževanje ontologij,
- decentraliziran razvoj ontologij,
- dobre prakse,
- podpora odločanju pri različnih interesih,
- preveč različnih pristopov,
- vključenost v širši poslovni procesni model,
- ocenjevanje stroškov,
- pomanjkanje podpore orodij.

9 ORODJA ZA RAZVOJ ONTOLOGIJ

V zadnjih letih je bilo razvitih mnogo orodij za razvoj ontologij. Pri odločitvi o gradnji nove ontologije in izbiri primernega orodja pa se je potrebno vprašati na kar nekaj vprašanj [5]:

- Ali orodje podpira razvojni proces ontologije?
- Kako se ontologije shranjujejo (v podatkovnih bazah ali v ASCII datotekah)?
- Ali vključuje orodje mehanizem sklepanja?
- Ali ima orodje prevajalnike v različne jezike ontologij?
- Kako lahko aplikacije sodelujejo s strežniki ontologij?

9.1 Pregled različnih vrst orodij za gradnjo in uporabo ontologij

Na področju razvoja ontologij je nastalo že nekaj vrst različnih orodij in okolij za gradnjo in uporabo ontologij:

- **Orodja za razvoj ontologij**
  Vključujejo orodja in okolja za gradnjo novih ontologij od samega začetka ali s ponovno uporabo obstoječih ontologij. Ta orodja po navadi vključujejo urejanje, preiskovanje, dokumentiranje, uvažanje in izvažanje v različnih formatih ter grafični prikaz ontologij.
- **Orodja za spajanje in integracijo ontologij**
  Ta orodja omogočajo reševanje problemov pri spajanju in integraciji različnih ontologij v isti domeni. Takšna potreba se pojavi v želji po izboljšanju kakovosti obstoječih ontologij znotraj ene domene ali, na primer, pri povezovanju podjetij.

- **Orodja za vrednotenje ontologij**
  To so podporna orodja, ki zagotavljajo določen nivo kakovosti ontologij in z njimi povezane tehnologije. Zagotavljanje kakovosti je pomembno za zmanjševanje možnih problemov pri integraciji ontologij v aplikacije.
- **Orodja za shranjevanje in iskanje po ontologijah**
  Ta orodja omogočajo uporabo in izvedbo poizvedb po ontologijah.
- **Orodja za učenje ontologij**
  Uporabljajo se za (pol)avtomatično izpeljavo ontologij iz tekstov v naravnih jezikih [5].

V nadaljevanju bodo podrobneje predstavljena orodja za razvoj ontologij.

9.2 Ogrodje za primerjavo in ocenjevanje orodij

Kriteriji, ki bodo uporabljeni za primerjavo orodij za razvoj ontologij, so razdeljeni v naslednje skupine:

- **Osnovni podatki o orodjih**
  Vključujejo podatke o razvijalcih in verziji orodja. Ti podatki ne vplivajo na oceno orodja.
- **Prijaznost do uporabnika**
  To je najpomembnejši sklop kriterijev. Vključuje podatke o dostopnosti orodja (prosto dostopno ali plačljivo), o modernosti uporabniškega vmesnika, o podpori orodja metodologiji in poda informacijo, ali se orodje še razvija sočasno z novimi tehnologijami.
- **Arhitektura in razširljivost orodja**
  Vključuje podatke o arhitekturi orodja (samostojno, odjemalec/strežnik, trinivojska arhitektura), ali je orodje razširljivo z dodatnimi moduli in funkcionalnostmi, kako so shranjene ontologije (podatkovna baza ali ASCII datoteke) in, ali je sistem združljiv z drugimi orodji, na primer za spajanje, shranjevanje ali sklepanje.
- **Predstavitev znanja**
  Vključuje podatke o načinu modeliranja znanja v orodju in podpori jezikov za gradnjo aksiomov. Poleg tega pa je dodan seznam možnih jezikov za uvoz ali izvoz ontologij.
- **Vključene storitve sklepanja**
  Vsebuje podatke o tem, ali ima orodje vgrajene mehanizme sklepanja oziroma ima za to možnost uporabe zunanjih orodij. Preveri se, če ima orodje vključene mehanizme za preverjanje konsistentnosti in omejitev, in če lahko avtomatično klasificira koncepte v taksonomijo konceptov.
- **Uporabnost**
  V to kategorijo sodijo kriteriji o obstoju grafičnih urejevalnikov za kreiranje konceptov taksonomij in relacij. Poleg tega so podane informacije o možnosti skupinskega dela ter prisotnost knjižnic ontologij [5].

9.3 Opis orodij za razvoj ontologij

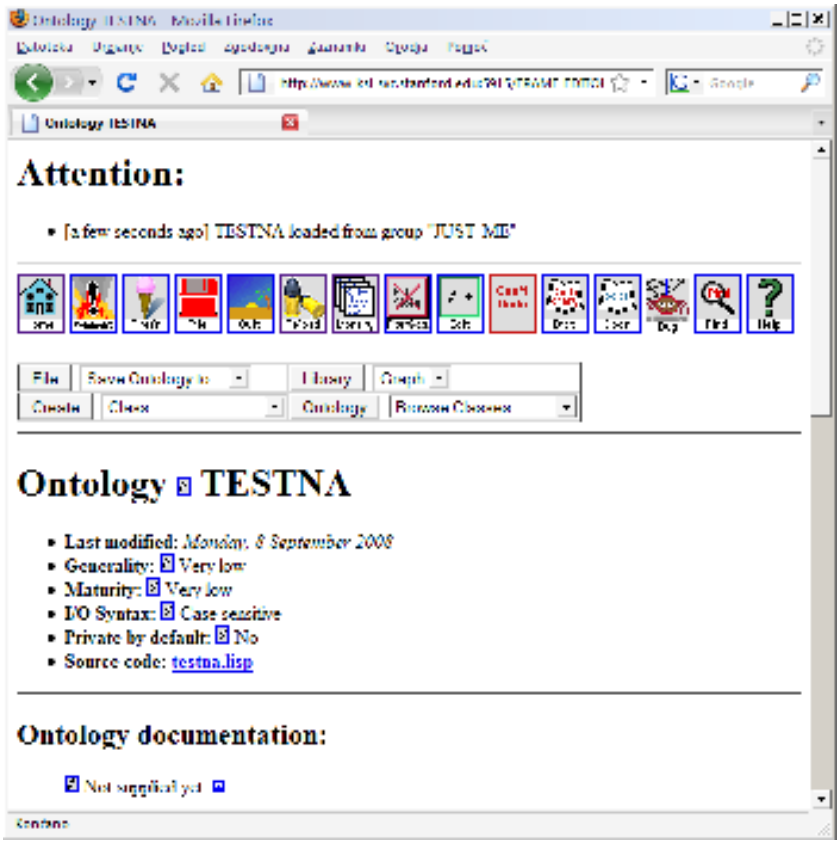
V tem poglavju bodo predstavljena nekatera orodja in okolja za razvoj ontologij:

- Ontolingua Server,
- WebOnto,

- OntoEdit,
- OntoStudio,
- WebODE,
- Protege.

9.3.1 Ontolingua Server

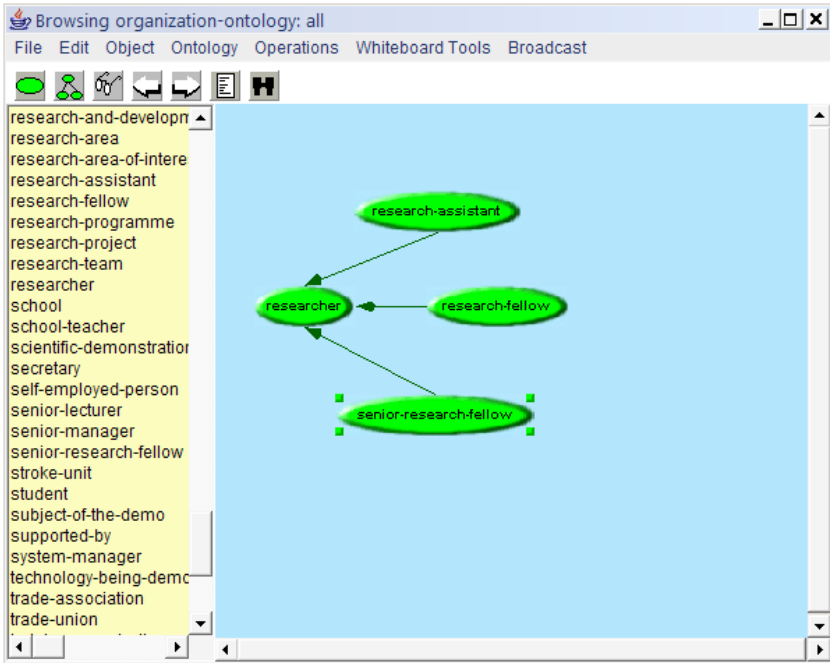
Ontolingua Server je bilo prvo razvito orodje za razvoj ontologij. Razvito je bilo na univerzi Stanford University. Ontolingua Server se je pojavil v začetku devetdesetih let z namenom olajšati razvoj ontologij v jeziku Ontolingua oziroma KIF s pomočjo spletne aplikacije na podlagi obrazcev. Najprej je bil glavni modul orodja urejevalnik ontologij, pozneje pa so bili v okolje dodani še drugi moduli, kot so modul za iskanje podobnosti, OKBC strežnik in Chimaera (orodje za spajanje ontologij).
Orodja in storitve Ontolingua Server podpirajo gradnjo ontologij za skupno rabo med različnimi skupinami uporabnikov. Arhitektura serverja ontologij omogoča dostop do knjižnic ontologij, prevajalnikov v različne jezike in urejevalnika za kreiranje in iskanje ontologij. Oddaljeni razvijalci lahko tako iščejo in urejajo ontologije, oddaljene ali lokalne aplikacije pa lahko dostopajo do katere koli ontologije v knjižnici ontologij.
Ontolingua Server je prosto dostopno spletno orodje, omogoča poljubno število sočasnih uporabnikov. Orodje spodbuja tudi skupinsko delo, saj se pri razvoju ontologije kreira seja, v kateri lahko sodeluje več razvijalcev.
To orodje je bilo namenjeno predvsem raziskovalnih aktivnostim in ne predvideva posebne razširljivosti funkcionalnosti [5, 18].



Slika 28: Orodje Ontolingua Server

9.3.2 WebOnto

WebOnto je bil razvit leta 1997 na inštitutu Knowledge Media Institute na univerzi Open University v Angliji. Omogoča sodelovanje več razvijalcev pri iskanju, gradnji in urejanju ontologij na podlagi jezika za modeliranje znanja OCML. Glavna prednost orodja je podpora skupinskemu urejanju ontologij, ki omogoča sinhrone in asinhrone diskusije o razvijajočih ontologijah. Omogoča komuniciranje med inženirji ontologij in domenskimi strokovnjaki. Skupaj s spremljevalnim orodjem Tadzebao podpira grafično urejanje ontologij in argumentiranje med oblikovalci s pomočjo teksta, slik in ročnih skic.
WebOnto je prosto dostopno spletno orodje, ki je na voljo vsem razvijalcem ontologij. V sklopu orodja je vključena knjižnica z mnogimi ontologijami, do vseh pa je omogočen prost dostop.
WebOnto je zgrajen v obliki Java appleta, ki je povezan s spletnim strežnikom. Orodje ni razširljivo in ne vključuje naprednejših mehanizmov sklepanja [5, 18].



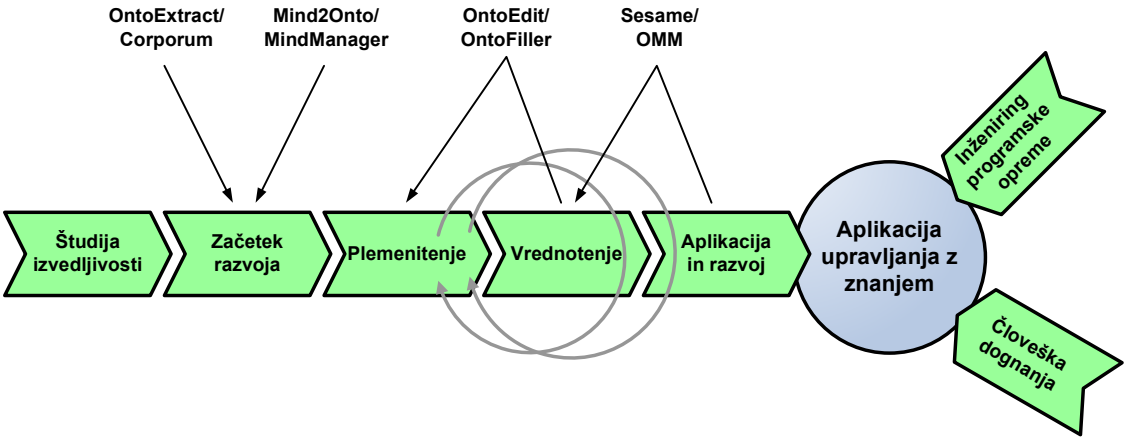
Slika 29: Orodje WebOnto

9.3.3 OntoEdit

V okviru projekta On-To-Knowledge je bilo na univerzi Karlsruhe University razvito orodje OntoEdit za neposredno podporo metodologiji razvoja ontologij On-To-Knowledge. OntoEdit je okolje za inženiring ontologij, ki temelji na prilagodljivem ogrodju, ki omogoča dodajanje dodatnih modulov. S tem je mogoče preprosto razširjati funkcionalnost in uporabljati orodja za različne scenarije.
OntoEdit podpira modeliranje konceptov, relacij in aksiomov, neodvisno od jezikov za predstavitev ontologij. Več grafičnih pogledov na strukturo znanja pa podpira modeliranje ontologije skozi več faz življenjskega cikla razvoja ontologij.
Orodje omogoča razvoj in vzdrževanje ontologij, urejanje hierarhij konceptov ali razredov, iskanje po ontologijah. Z dodatnimi moduli so vključeni še mehanizmi sklepanja ter mehanizmi za uvažanje in izvažanje ontologij različnih formatov, kot so FLogic, XML,

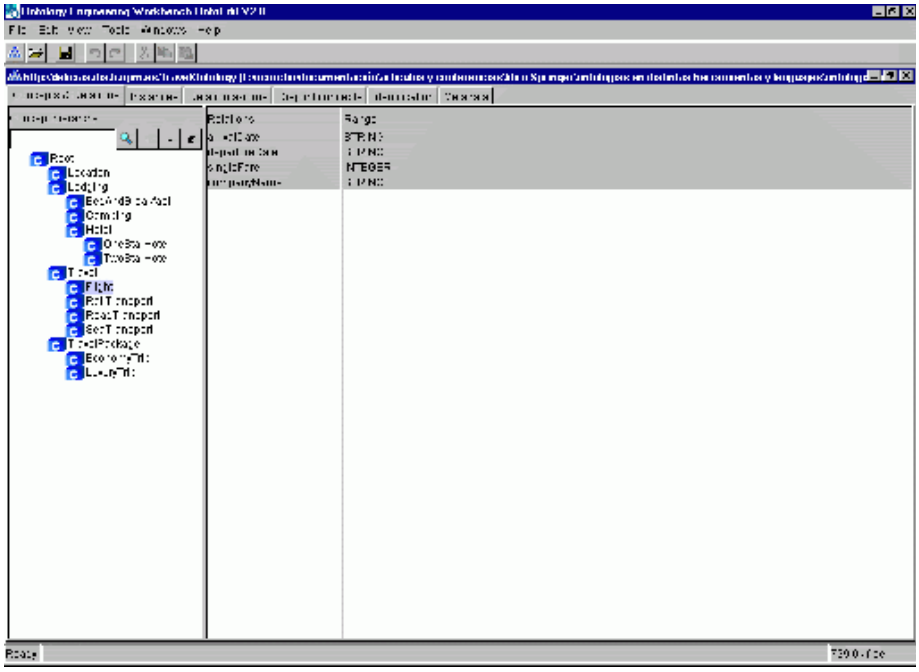
RDF(S) in DAML+OIL. OntoEdit je na voljo v dveh različicah, OntoEdit Free, ki je na voljo vsem, in OntoEdit Professional, ki je plačljivo orodje.
Za podporo različnih faz življenjskega cikla razvoja ontologij po metodologiji On-To-Knowledge so bila razvita še nekatera dodatna orodja, ki so prikazana na sliki 30:

- OntoExtract/Corporum: orodji za avtomatično povzemanje konceptov iz teksta za konstruiranje preprostih ontologij na osnovi lingvističnih sposobnosti orodja Corporum;
- MindManager/Mind2Onto: orodji, ki podpirata postopek zbiranja idej pri skupinskem iskanju rešitev;
- OntoEdit/OntoFiller: orodji za interaktivno gradnjo ontologij in shranjevanje v okolje Sesame;
- Sesame/OMM: orodji za shranjevanje ontologij in beleženje sprememb pri skupinskem razvoju ontologij [5, 33].



Slika 30: Orodja za podporo fazam metodologije On-To-Knowledge

Poleg orodij za razvoj je bilo razvitih tudi nekaj orodij, namenjenih uporabnikom ontologij za iskanje in navigacijo po ontologijah in za skupno uporabo znanja, kot so Spectacle, QuizRDF in OntoShare.



Slika 31: Orodje OntoEdit

9.3.4 OntoStudio

OntoStudio je profesionalno razvojno okolje za gradnjo, testiranje in vzdrževanje ontologij, ki je bilo razvito kot naslednik orodja OntoEdit v podjetju Ontoprise. OntoStudio omogoča semantično združevanje več heterogenih virov podatkov in njihovo povezovanje v ontologije. Uvažajo se lahko podatki v različnih formatih.

OntoStudio je osnovan na razširjenem odprtokodnem ogrodju Eclipse. Tako ima to orodje modularno zgradbo, ki je poznana velikemu številu razvijalcev. Vsebuje bogato zbirko funkcionalnosti, ki pa se lahko razširjajo z dodatnimi moduli.

OntoStudio podpira urejanje ontologij v jezikih RDF in FLogic, v prihodnje pa bo na voljo tudi integriran urejevalnik jezika OWL. OntoStudio zagotavlja ločen vidik modeliranja za vsak jezik zase, kar omogoča specifičen grafični uporabniški vmesnik glede na izbran jezik. Urejevalnik jezika FLogic vključuje tudi funkcionalnosti označevanja sintakse, navigacije in avtomatskega dokončanja kode.

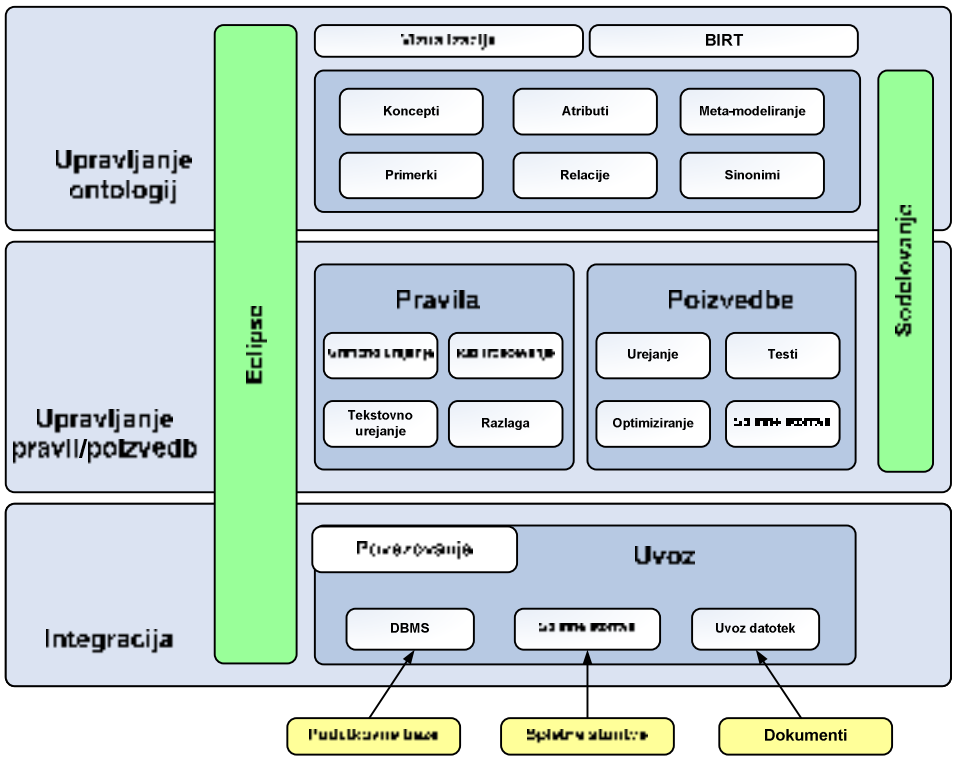
OntoStudio omogoča tudi podporo skupinskemu razvoju. Za to je namenjen poseben strežnik OntoBroker Collaboration Server, preko katerega lahko razvijalci kreirajo, urejajo in omogočajo skupno rabo ontologij ali njihovih delov.

Za izrazna pravila, ki predstavljajo glavno moč ontologij, vključuje OntoStudio mnogo funkcionalnosti za urejanje in upravljanje pravil. Pravila se lahko urejajo preko tekstovnega urejevalnika ali pa grafičnega urejevalnika, ki omogoča kreiranje pravil z grafičnim povezovanjem elementov ontologije. Poleg pravil so lahko dodane tudi razlage o njihovem pomenu. Pri večjem številu pravil so lahko ta urejena v hierarhično strukturo in opremljena z metapodatki.

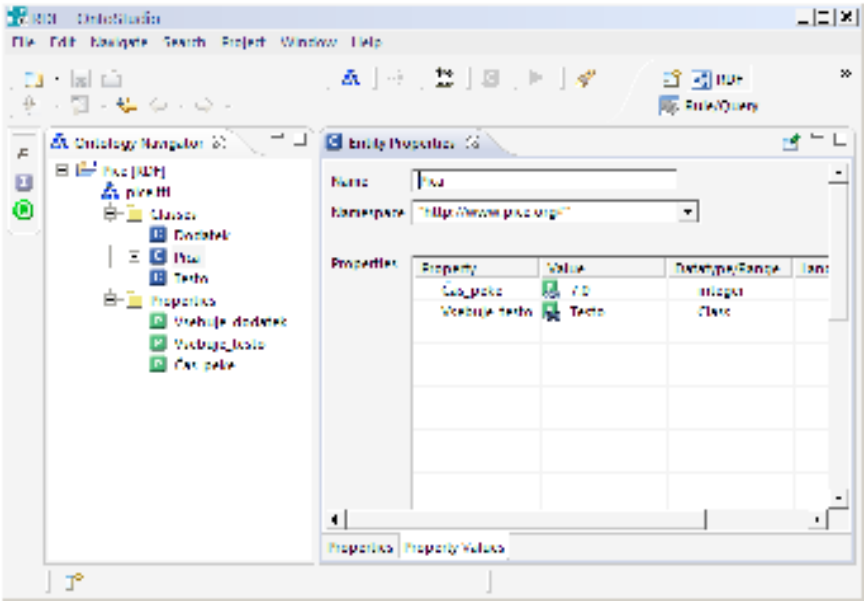
OntoStudio omogoča izvajanje povpraševanj nad ontologijami s pomočjo SPARQL ali FLogic poizvedb, za katere je na voljo tudi grafični urejevalnik. Poizvedbe se lahko izvajajo in testirajo s pomočjo vgrajenega mehanizma sklepanja.

Za testiranje celotnih ontologij je mogoče kreirati zbirke testov in pričakovanih rezultatov, s katerimi je mogoče redno preverjati konsistentnost in skladnost ontologij. Ob negativnih rezultatih testov je na voljo orodje za razhroščevanje, ki beleži nastanek rezultatov.
OntoStudio vsebuje tudi mnogo priključkov za podatkovne baze (Oracle 11g, IBM DB2, Microsoft SQL Server), dokumente (Excel, Outlook, UML), datotečne sisteme, aplikacije in spletne storitve. Napredno povezovalno orodje omogoča semantično integracijo heterogenih podatkovnih virov v ontologijo.
V okolje OntoStudio sta vključeni tudi orodji:

- **OntoBroker** vsebuje mehanizem sklepanja in repozitorij za shranjevanje ontologij. S shranjenimi ontologijami je možno manipulirati s pomočjo vmesnikov za upravljanje in povpraševanje. Poizvedbe se pošljejo v OntoBroker, ki s pomočjo mehanizma sklepanja izpelje odgovore preko formalne semantike ontologij. Pri tem so vključeni vsi priključeni podatkovni viri. OntoBroker nudi podporo jezikom RDF, RDF(S), OWL in FLogic.
- **SemanticGuide** je orodje, namenjeno uporabnikom. Omogoča hitro in učinkovito reševanje problemov pri odločanju in svetovanje pri intenzivnem procesiranju znanja. SemanticGuide zbira znanje strokovnjakov, ki je nato na voljo vsem uporabnikom. Orodje pomaga pripeljati uporabnike do pravih rešitev in želenih informacij [21].



Slika 32: Funkcionalnosti orodja OntoStudio



Slika 33: Orodje OntoStudio

9.3.5 WebODE

WebODE je delovno okolje za inženiring ontologij, ki je bilo razvito v laboratoriju Artificial Intelligence Lab na univerzi Technical University v Madridu. Zagotavlja različne storitve, povezane z ontologijami, in pokriva ali podpira večino aktivnosti, povezanih s procesom razvoja ontologij in njihove uporabe. Pri tem se navezuje na aktivnosti metodologije razvoja ontologij METHONTOLOGY.
WebODE je prosto dostopno orodje s spletnim vmesnikom, ki je osnovano na trinivojski arhitekturi (nivo podatkovne baze, nivo aplikacijskega strežnika, nivo uporabniškega vmesnika). Aplikacijski strežnik omogoča dobro razširljivost, preprosto uporabo obstoječih storitev in dodajanje novih. Ontologije so shranjene v relacijski podatkovni bazi.

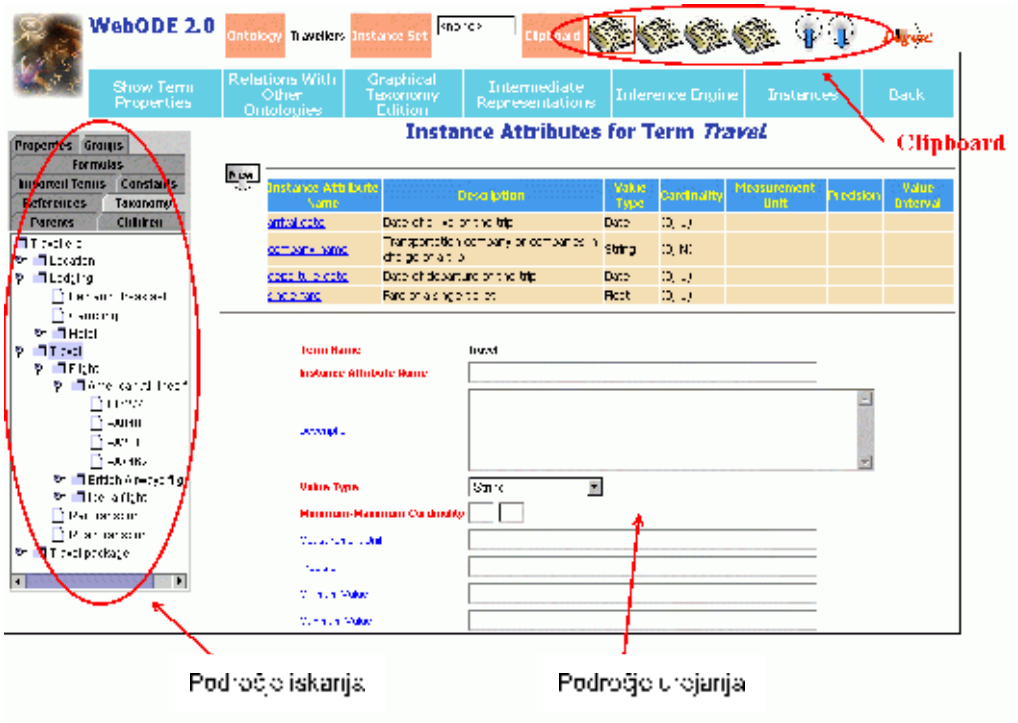
Slika 34: Arhitektura orodja WebODE

Ontologije v WebODE so predstavljene z dobro izraznim modelom znanja, ki vključuje komponente, kot so koncepti, skupine konceptov, taksonomije, binarne relacije, konstante, formule, aksiomi, pravila, primerki in reference. Omogočeno je tudi uvažanje pojmov iz drugih ontologij.

WebODE vključuje dobro definiran storitveno orientiran vmesnik za dostop do ontologij, ki omogoča preprosto integracijo z drugimi sistemi. Povezovanje z drugimi ontologijami pa je mogoče tudi z avtomatičnimi storitvami izvažanja in uvažanja iz drugih sistemov. Izvoz je mogoč v XML, RDF(S), OIL, DAML+OIL, OWL, Prolog, UML in X-CARIN sisteme, uvoz pa iz XML, RDF(S), DAML+OIL, OWL in UML sistemov.
Za urejanje ontologij v orodju WebODE so omogočene naslednje funkcionalnosti:

- grafični uporabniški vmesnik in vmesnik na podlagi obrazcev,
- upravljalec različnih pogledov,
- preverjanje konsistentnosti (omejitev tipov, omejitev številnosti, pregled konsistentnosti taksonomij …),
- mehanizem sklepanja na podlagi jezika Prolog,
- kreiranje aksiomov,
- dokumentiranje,
- spajanje ontologij.

Poleg tega vključuje WebODE kreiranje konceptualnih pogledov iz istega konceptualnega modela, kar omogoča kreiranje in shranjevanje različnih delov ontologij in prirejanje pogledov na ontologije za določeno uporabo.
Za potrebe sodelovanja pri razvoju ontologij je poskrbljeno z mehanizmom, ki omogoča kreiranje skupine uporabnikov z dostopom do ontologij. Poleg tega obstajajo tudi mehanizmi sinhronizacije, ki omogočajo hkratno urejanje ontologije več uporabnikom [18, 23].



Slika 35: Orodje WebODE

9.3.6 Protege

Protege je bil razvit na univerzi Stanford University. Je prosto dostopna in odprtokodna platforma, ki ponuja uporabnikom vrsto orodij za konstruiranje domenskih modelov in ontologij za aplikacije, osnovane na znanju. Protege v osnovi omogoča implementiranje različnih struktur in aktivnosti modeliranja znanja, ki podpirajo kreiranje, vizualizacijo in manipuliranje z ontologijami v različnih predstavitvenih načinih. Protege je možno prilagajati glede na različne zahteve za doseganje uporabnikom prijaznejše podpore kreiranju modelov znanja in vnašanju podatkov.
Arhitektura orodja Protege je zgrajena modularno na osnovi programskega jezika Java in omogoča razvijalcem preprosto razširljivost ter dodajanje novih modulov oziroma funkcionalnosti. V knjižnici dodatnih modulov Protege Plugin Library je na voljo mnogo dodatkov različnih razvijalcev.
Platforma Protege omogoča dva različna načina modeliranja ontologij, z urejevalnikom Protege-Frames ali urejevalnikom Protege-OWL [18, 22].

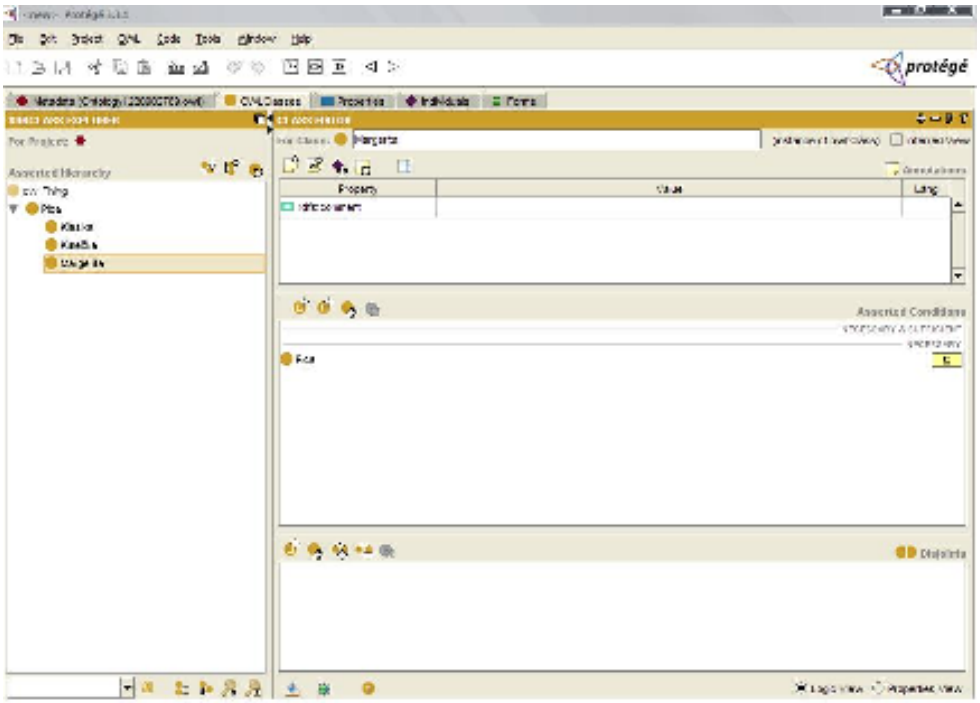
Protege-Frames
Urejevalnik Protege-Frames uporabnikom zagotavlja uporabniški vmesnik in strežnik znanja za konstruiranje in shranjevanje strukturiranih domenskih ontologij, prilagodljivih obrazcev za vnašanje podatkov in vnašanje podatkov o primerkih. Model znanja v Protege-Frames je skladen z jezikom OKBC. Ta model je sestavljen iz zbirke razredov, urejenih v hierarhijo, ki predstavljajo pomembne koncepte domene, zbirke sledi, povezanih z razredi, za opis njihovih lastnosti in relacij in zbirke primerkov teh razredov.
Značilnosti urejevalnika Protege-Frames so:

- široka paleta prilagodljivih elementov uporabniškega vmesnika,
- razširljiva arhitektura, kateri je mogoče dodati elemente, kot so grafične komponente (grafi, tabele), multimedijski elementi (zvoki, slike, video), različni formati shranjevanja (RDF(S), XML, OIL, DAML+OIL) in dodatna orodja za vizualizacijo, upravljanje ontologij,
- API vmesnik, ki omogoča drugim aplikacijam dostop, prikazovanje in uporabo ontologij, kreiranih z urejevalnikom Protege-Frames [22].

Protege-OWL
Urejevalnik Protege-OWL je razširitev orodja Protege za podporo jeziku OWL in viziji semantičnega spleta. OWL ontologije lahko vključujejo opise razredov, lastnosti in njihovih primerkov. Semantika jezika OWL omogoča izpeljavo logičnih posledic ontologije, to je dejstva, ki niso dobesedno prisotna v ontologiji, a so razvidna iz semantike.
Značilnosti urejevalnika Protege-OWL so:

- nalaganje in shranjevanje OWL in RDF ontologij,
- urejanje in vizualizacija razredov, lastnosti in SWRL pravil,
- definiranje logičnih karakteristik razredov z OWL izrazi,
- izvrševanje sklepov, kot so klasifikatorji opisne logike,
- urejanje OWL primerkov za namene semantičnega spleta.

Tudi za arhitekturo Protege-OWL je značilno preprosto konfiguriranje in razširjanje z dodatnimi moduli [22].



Slika 36: Orodje Protege

9.4 Primerjava orodij

Primerjava orodij za razvoj ontologij je bila narejena podobno kot primerjava metodologij, s pomočjo programa DEXi. V drevo kriterij za ocenjevanje so vključeni kriteriji, ki so opisani v poglavju Ogrodje za primerjavo in ocenjevanje orodij. V primerjavo je bilo vključenih vseh šest opisanih orodij.

9.4.1 Predstavitev in razlaga uteži kriterijev

Kriteriji ocenjevanja metodologij so razvrščeni v drevesno strukturo, vsak kriterij pa ima določeno utež, ki predstavlja vpliv tega kriterija na skupno oceno razvitosti metodologije. Lokalna vrednost uteži pomeni vpliv kriterija v sklopu ene veje drevesa kriterijev, globalna vrednost uteži pa pomeni vpliv kriterija na celotno oceno metodologije. Kriteriji in uteži so predstavljeni na sliki 37.

Povprecne uteži

Kriterij	Lokalne	Globalne
Razvitost orodja		
Prijaznost do uporabnika	36	36
Posodabljanje orodja	36	13
Uporabniški vmesnik	30	11
Dostopnost	19	7
Podpora metodologiji	15	5
Arhitektura in razširljivost orodja	31	31
Arhitektura programske opreme	22	7
Razširljivost	39	12
Shranjevanje ontologij	20	6
Združljivost z drugimi orodji	20	6
Predstavitev znanja	14	14
Predstavitev modela znanja	12	2
Jezik za kreiranje aksiomov	28	4
Uvažanje iz jezikov	30	4
Izvažanje v jezike	30	4
Vključene storitve sklepanja	10	10
Vgrajen mehanizem sklepanja	56	5
Preverjanje konsistentnosti in omejitev	33	3
Avtomatska klasifikacija	11	1
Uporabnost orodja	10	10
Grafični prikaz taksonomij	33	3
Omogočeno skupinsko delo	33	3
Knjižnica ontologij	33	3

Slika 37: Kriteriji za ocenjevanje razvitosti orodij

Skupna ocena razvitosti orodja je sestavljena iz ocen sklopov prijaznost do uporabnika, arhitektura in razširljivost orodja, predstavitev znanja, vključene storitve sklepanja in uporabnost orodja. Največji delež k skupni oceni prispevata sklopa prijaznost do uporabnika, kjer so opisani kriteriji, ki vplivajo na preprostejše in udobnejše delo z orodjem s stališča uporabnika, ter arhitektura in razširljivost orodja, ki vključuje informacije o prilagodljivosti orodja drugim orodjem in funkcionalnostim.

Znotraj sklopa prijaznost do uporabnika je največji poudarek na kriterijih posodabljanja orodja in uporabniškega vmesnika, ki imata tudi na globalnem nivoju, ob kriteriju razširljivosti, največji vpliv na skupno oceno.

Sklopi predstavitev znanja, vključene storitve sklepanja in uporabnost orodja imajo malo manjši vpliv na oceno, ker opisujejo bolj specifične funkcionalnosti orodja.

Izmed kriterijev imata najmanjši vpliv na skupno oceno kriterija avtomatska klasifikacija in predstavitev modela znanja, ker je njuna vrednost enaka pri vseh orodjih in tako ne prispevata veliko k razlikovanju razvitosti orodij.

9.4.2 Podatki o orodjih

Zbrani podatki o orodjih se nahajajo v tabeli 5.

Lastnost		Ontolingua	WebOnto	OntoEdit	OntoStudio	WebODE	Protege
Osnovni podatki	Razvijalci	Knowledge System Laboratory (Stanford University)	Knowldege Media Institute (Open University)	AIFB (University of Karlsruhe)	OntoPrise	Artificial Intelligence Lab (Technical University of Madrid)	Stanford Medical Informatics (Stanford University)
	Verzija	1.0650 (oktober 2002)	2.3 (maj 2001)	3.0 (avgust 2002)	2.1 (maj 2008)	2.0.9 (november 2003)	3.3.1
Prijaznost do uporabnika	Dostopnost	Prosto dostopno	Prosto dostopno	Ukinjeno	Plačljivo	Prosto dostopno	Prosto dostopno
	Posodabljanje orodja	Ni posodobitev	Ni posodobitev	Ni posodobitev	Redno	Ni posodobitev	Redno
	Uporabniški vmesnik	Zastarel	Zastarel	Primeren	Moderen	Primeren	Moderen
	Podpora metodologiji	Ne	Ne	Da (On-To-Knowledge)	Ne	Da (METHONTOLOGY)	Ne
Arhitektura in razširljivost orodja	Arhitektura programske opreme	Odjemalec/strežnik	Odjemalec/strežnik	Samostojno orodje	Samostojno orodje	3 nivojska arhitektura	Samostojno orodje
	Razširljivost	Ne	Ne	Da	Da	Da	Da
	Shranjevanje ontologij	Datoteke	Datoteke	Podatkovna baza	Podatkovna baza	Podatkovna baza	Podatkovna baza
	Z drugimi orodji	Da	Da	Da	Da	Da	Da
Predstavitev znanja	Predstavitev modela znanja	Struktura	Struktura	Struktura	Struktura	Struktura	Struktura
	Jezik za kreiranje aksiomov	KIF	OCML	FLogic	FLogic	WAB	PAL
	Uvažanje iz jezikov	Ontolingua, KIF, OKBC, CML	OCML	XML, RDF(S), FLogic, DAML+OIL	XML, RDF(S), FLogic, OWL, UML,	XML, RDF(S), DAML+OIL, OWL, UML	XML, RDF(S), OWL
	Izvažanje v jezike	KIF, CLIPS, OKBC, LOOM	OCML, Ontolingua, RDF(S), OIL	XML, RDF(S), FLogic, DAML+OIL	XML, RDF(S), FLogic, OWL, UML,	XML, RDF(S), OIL, DAML+OIL, OWL, Prolog, UML, Java/Jess	RDF(S), OWL, XML
Vključene storitve sklepanja	Vgrajen mehanizem sklepanja	Ne	Da	Da	Da	Da	Da
	Preverjanje konsistentnosti in omejitev	Ne	Da	Da	Da	Da	Da
	Avtomatska klasifikacija	Ne	Ne	Ne	Ne	Ne	Ne
Uporabnost orodja	Grafični prikaz taksonomij	Da	Da	Ne	Ne	Da	Da
	Omogočeno skupinsko delo	Da	Da	Da	Da	Da	Da
	Knjižnica ontologij	Da	Da	Da	Da	Ne	Da

Tabela 5: Zbrani podatki o orodjih po določenih kriterijih [5, 18]

9.4.3 Rezultati primerjave orodij in ocena razvitosti

V tabeli 6 so predstavljeni rezultati primerjave razvitosti orodij glede na opisane kriterije in zbrane podatke iz tabele 5. Podrobni rezultati in podatki primerjave so prikazani v prilogi 2 (poročilo primerjave iz programa DEXi).

Orodje	Ontolingua	WebOnto	OntoEdit	OntoStudio	WebODE	Protege
Razvitost	Slabo razvito	Srednje razvito	Srednje razvito	Dobro razvito	Dobro razvito	Dobro razvito

Tabela 6: Rezultati primerjave orodji

Po tej primerjavi je dobilo orodje Ontolingua oceno slabo razvito, orodji WebOnto in OntoEdit oceno srednje razvito, orodja OntoStudio, WebODE ter Protege pa oceno dobro razvito.

Orodje Ontolingua Server je edino, ki nima vključenega mehanizma sklepanja in možnosti preverjanja konsistentnosti in omejitev. Poleg tega nima možnosti razširljivosti in združljivosti z drugimi orodji, ontologije lahko shranjuje samo v obliki datotek, manjka pa tudi podpora modernejšim jezikom za razvoj ontologij. Tudi za orodje WebOnto velja, da ne nudi možnosti razširljivosti in povezave s podatkovnimi bazami.

Orodje OntoEdit je bilo vključeno v primerjavo predvsem zaradi njegove povezanosti in podpore metodologiji On-To-Knowledge. OntoEdit namreč ni več dostopen, saj ga je nadomestilo modernejše in komercialno orodje OntoStudio.

Drugo orodje, ki nudi podporo metodologiji, je spletno orodje WebODE, ki podpira razvojni proces metodologije METHONTOLOGY. WebODE ima tudi dobro podporo različnim jezikom in dobro zasnovano trinivojsko arhitekturo. Njegovi slabosti sta odsotnost knjižnice ontologij in neažurno posodabljanje oziroma razvoj orodja.

Najsodobnejši orodji za razvoj ontologij, ki imata tudi najmodernejši vmesnik, sta orodji OntoStudio in Protege. Obe sta samostojni orodji, z možnostjo razširljivosti povezave do podatkovnih baz in združljivosti z drugimi orodji. Njuna slabost je predvsem odsotnost podpore metodologijam. OntoStudio trenutno še ne omogoča razvoja ontologij v jeziku OWL, vendar naj bi bil dodatek te funkcionalnosti orodju že kmalu na voljo.

Če na splošno pogledamo na orodja za razvoj ontologij, lahko najprej opazimo, da obstaja s stališča posodabljanja velika razlika med orodji. S stališča uporabnika je namreč zelo pomembno, da je delo z orodjem čim bolj preprosto in učinkovito. Na eni strani sta OntoStudio in Protege sodobni orodji z modernima uporabniškima vmesnikoma, na drugi strani pa vsa ostala orodja, katerih zadnje verzije so stare že nekaj let. Temu primerno so zastareli tudi njihovi uporabniški vmesniki. OntoStudio in Protege imata tako z vidika prijaznosti orodja do uporabnika veliko prednost pred ostalimi orodji.

Za preprostejši razvoj ontologij manjka orodjem predvsem boljša podpora metodologijam. OntoEdit in WebODE sta namreč edini orodji, ki sta dobro povezani z metodologijama. Vendar pa tudi tu nobeno od orodij ne podpira faze upravljanja projekta, slabo pa je pokrito tudi vzdrževanje ontologij.

Glede arhitekture orodij lahko na splošno rečemo, da se pozna napredek pri stopnji razširljivosti in združljivosti orodij z drugimi orodji, za shranjevanje ontologij pa se uporabljajo podatkovne baze. S tem se zelo poveča prilagodljivost orodja, saj je mogoče orodjem dodajati nove funkcionalnosti glede na specifične zahteve razvoja. Orodja so večinoma osnovana na programskem jeziku Java. Glede na arhitekturo lahko orodja razdelimo v skupino spletnih in skupino samostojnih orodij. Spletna orodja nudijo sicer lažjo podporo sodelovanju pri razvoju ontologij, vendar so njihovi uporabniški vmesniki po navadi

bolj togi. Ker sta OntoStudio in Protege samostojni orodji, lahko sklepamo, da se razvoj nagiba čedalje bolj k tej vrsti arhitekture.
Za predstavitev znanja vidimo, da vsa orodja uporabljajo pretežno strukturne jezike. Vsa orodja vključujejo jezike za kreiranje aksiomov, dobro pa je pokrita tudi združljivost z različnimi zapisi pri uvažanju in izvažanju ontologij.
Razvoj ontologij je lahko zelo kompleksen projekt, zato je podpora sodelovanju več skupin strokovnjakov pri razvoju zelo pomembna. Vsa orodja, vključena v primerjavo, omogočajo sodelovanje in sočasno delo pri razvoju ontologij. Z vidika ponovne uporabe znanja oziroma ontologij pa je pomembna prisotnost knjižnice ontologij, ki omogoča vključevanje znanja obstoječih ontologij v razvoj novih. Vsa orodja, razen WebODE, vključujejo knjižnico ontologij.
Orodja za razvoj ontologij so se do sedaj še vedno večinoma razvijala in uporabljala na znanstvenem področju. OntoStudio kot komercialno naravnano orodje tako predstavlja nekakšen korak naprej pri vključevanju razvoja ontologij v podjetja in izdelave večjega števila aplikacij, osnovanih na ontologijah. Takšna usmerjenost bo verjetno pospešila tudi razvoj orodij za kreiranje ontologij.

10 PRIMER

V tem poglavju je predstavljen primer preproste ontologije, kreirane v orodjih OntoStudio in Protege. Ontologija v orodju OntoStudio je kreirana v jeziku RDF, ontologija v orodju Protege pa v jeziku OWL.

10.1 Predstavitev domenskega področja

V praktičnem primeru ontologije je zajet del domenskega področja izdelovanja pic. Primer vsebuje razrede:

- Pica, ki se deli na podrazrede Klasika in Domača,
- Testo, ki se deli na podrazreda Tanko_testo in Debelo_testo,
- Dodatek, ki se deli na podrazrede Sir, Šunka in Gobe.

Definirane lastnosti v primeru so naslednje:

- Vsebuje_testo,
- Vsebuje_dodatek,
- Čas_peke.

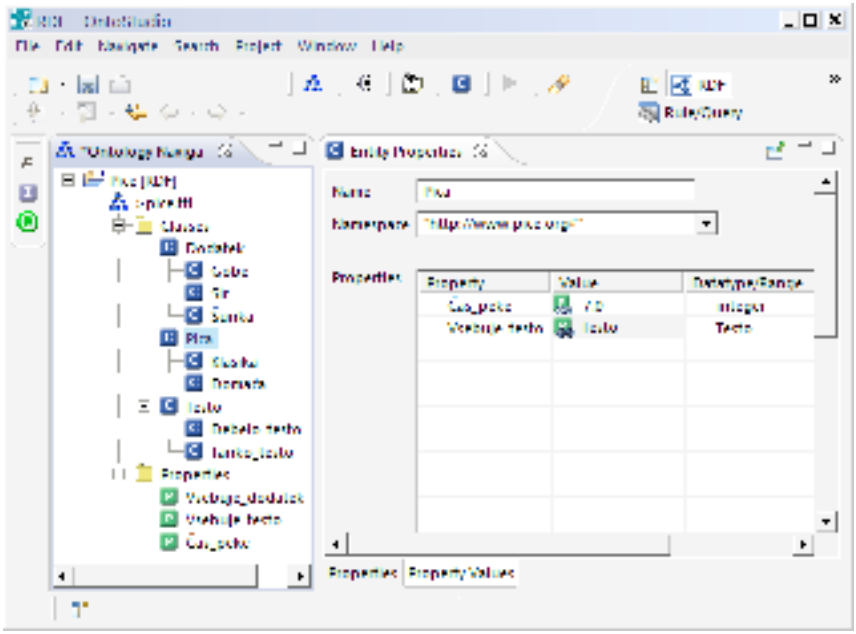
Lastnost Vsebuje_testo, ki predstavlja dejstvo, da je sestavni del pice testo, ima svojo domeno med primerki razreda Pica in svoj domet med primerki razreda Testo. Lastnost Vsebuje_dodatek predstavlja dejstvo, da pica vsebuje dodatek in ima domeno med primerki razreda Pica ter domet med primerki razreda Dodatek. Lastnost Čas_peke pa predstavlja čas peke pice ter ima domeno med primerki razreda Pica in domet med celimi števili.

Glavna dejstva, ki jih prikazuje ontologija, so:

- Primerek razreda Pica vsebuje primerek iz razreda Testo,
- Čas peke primerkov razreda Pica je sedem minut,
- Primerek pice podrazreda Klasika vsebuje primerek dodatka iz podrazredov Sir, Šunka in Gobe.

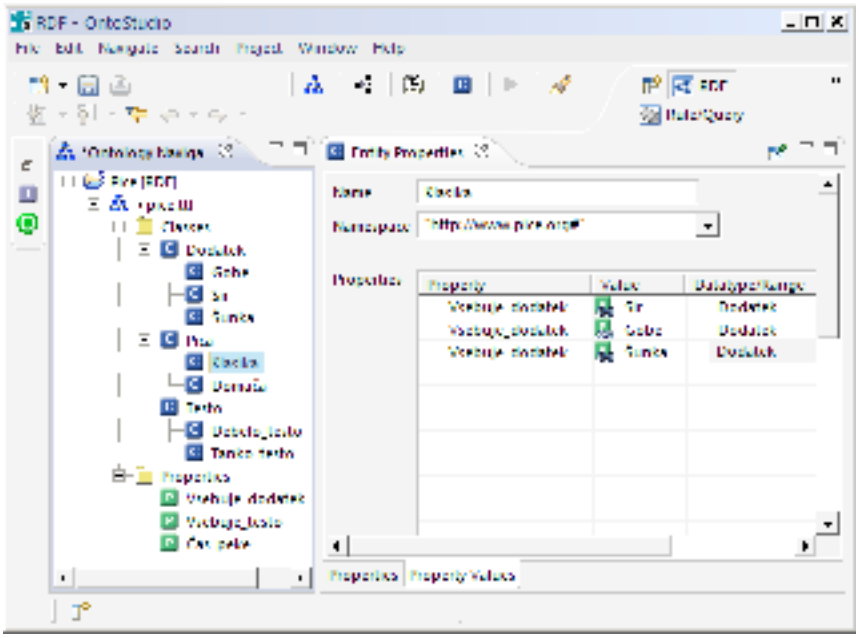
10.2 Primer v orodju OntoStudio

Na sliki 38 je prikazan pogled na orodje OntoStudio. V levem oknu orodja je prikazano drevo razredov in seznam lastnosti ontologije. V desnem oknu pa so predstavljene lastnosti razreda Pica. Pica vsebuje testo, čas pečenja pa je 7 minut.



Slika 38: Razred Pica in njegove lastnosti v orodju OntoStudio

Na sliki 39 so v desnem oknu orodja predstavljene lastnosti razreda Klasika, ki je podrazred razreda Pica. Pica klasika vsebuje dodatke: sir, šunko in gobe.

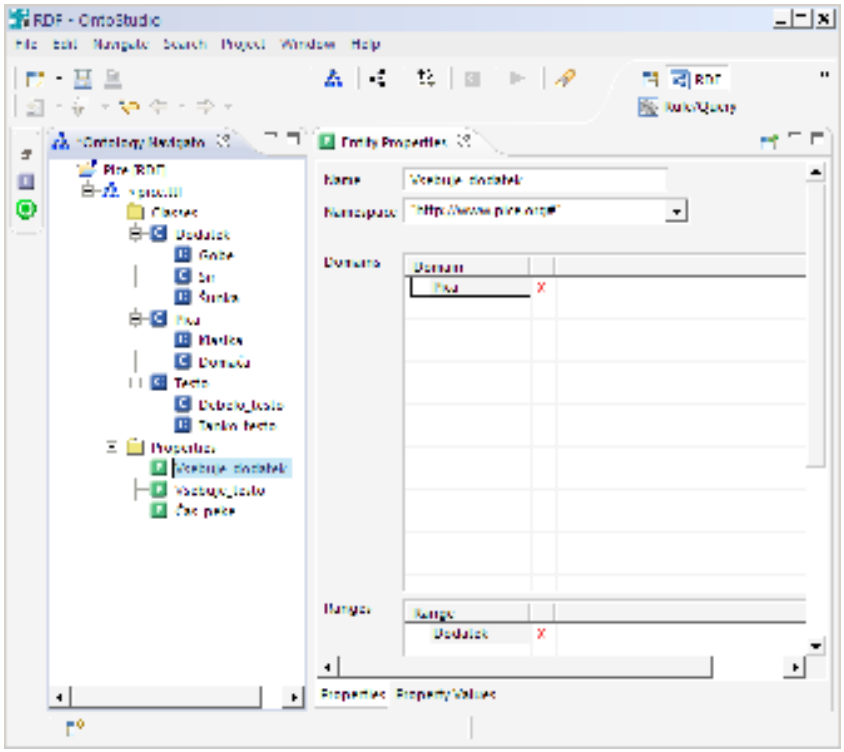


Slika 39: Razred Klasika in njegove lastnosti v orodju OntoStudio

Naslednji izsek iz RDF zapisa ontologije predstavlja dejstvo, da je razred Klasika podrazred razreda Pica.

```
<rdf:Description rdf:about="http://www.pice.org#Klasika">
      <rdfs:subClassOf rdf:resource="http://www.pice.org#Pica"/>
</rdf:Description>
```

Na sliki 40 je prikazana določenost domene in dometa lastnosti Vsebuje_dodatek. Domena lastnosti so primerki razreda Pica, domet pa primerki razreda Dodatek.



Slika 40: Domena in domet lastnosti Vsebuje_dodatek v orodju OntoStudio

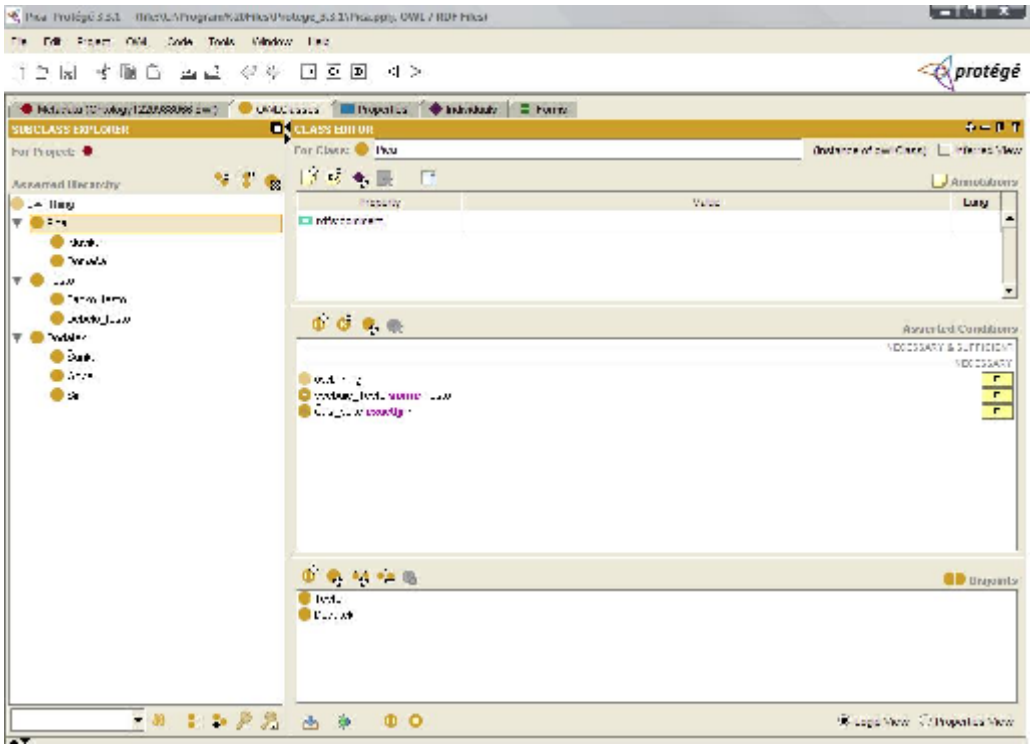
Naslednji izsek iz RDF zapisa ontologije predstavlja definiranje domene in dometa lastnosti Vsebuje_dodatek.

```
<rdf:Description rdf:about="http://www.pice.org#Vsebuje_dodatek">
      <rdfs:domain rdf:resource="http://www.pice.org#Pica"/>
</rdf:Description>

<rdf:Description rdf:about="http://www.pice.org#Vsebuje_dodatek">
      <rdfs:range rdf:resource="http://www.pice.org#Dodatek"/>
</rdf:Description>
```

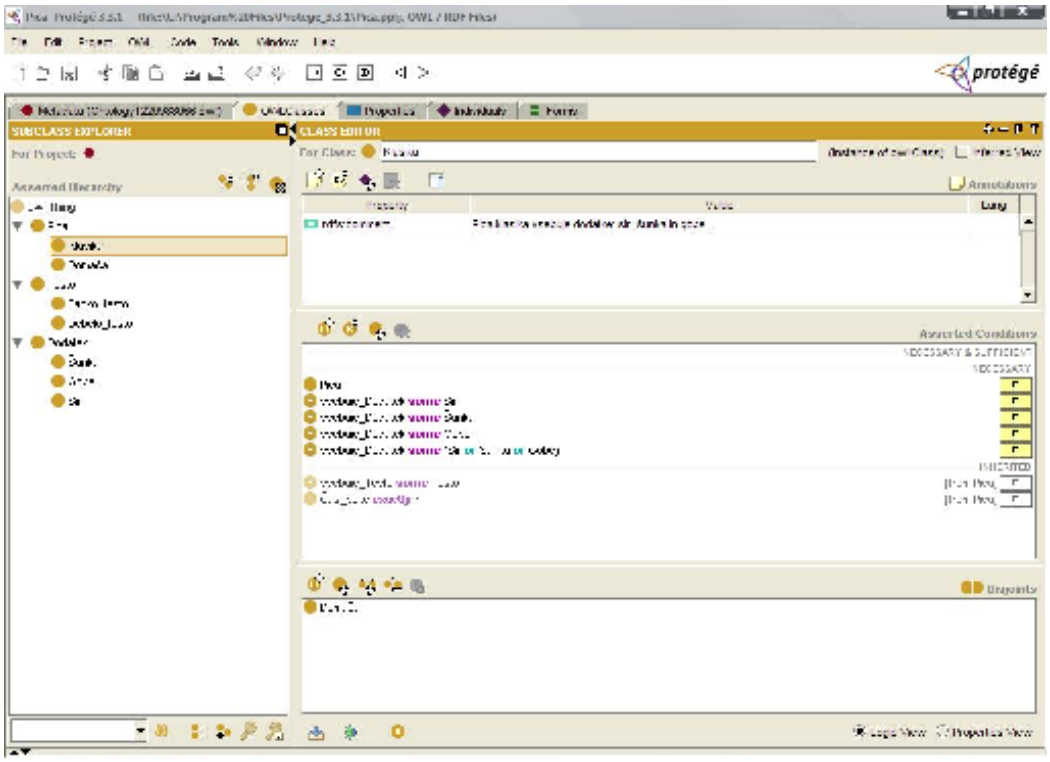
10.3 Primer v orodju Protege

Na sliki 41 je prikazan pogled na orodje Protege. V levem oknu orodja je prikazano drevo razredov ontologije. Na desni strani so v srednjem oknu predstavljena dejstva, da pica vsebuje testo in da je čas pečenja 7 minut. V spodnjem oknu je predstavljeno dejstvo, da sta razreda Dodatek in Testo nepovezana z razredom Pica. To pomeni, da primerki enega izmed teh razredov ne morejo biti hkrati primerki katerega izmed drugih dveh razredov.



Slika 41: Razred Pica in njegove lastnosti v orodju Protege

Na sliki 42 so prikazane lastnosti razreda Klasika. V zgornjem oknu na desni strani je opis razreda Klasika. V srednjem oknu so predstavljena dejstva, da vsebuje pica klasika dodatke: sir, šunko in gobe. Poleg tega je določeno, da so lahko ti trije dodatki tudi edini dodatki pice klasika. Od nadrejenega razreda Pica podeduje podrazred Klasika določitve, da se pica peče 7 minut in da vsebuje testo.

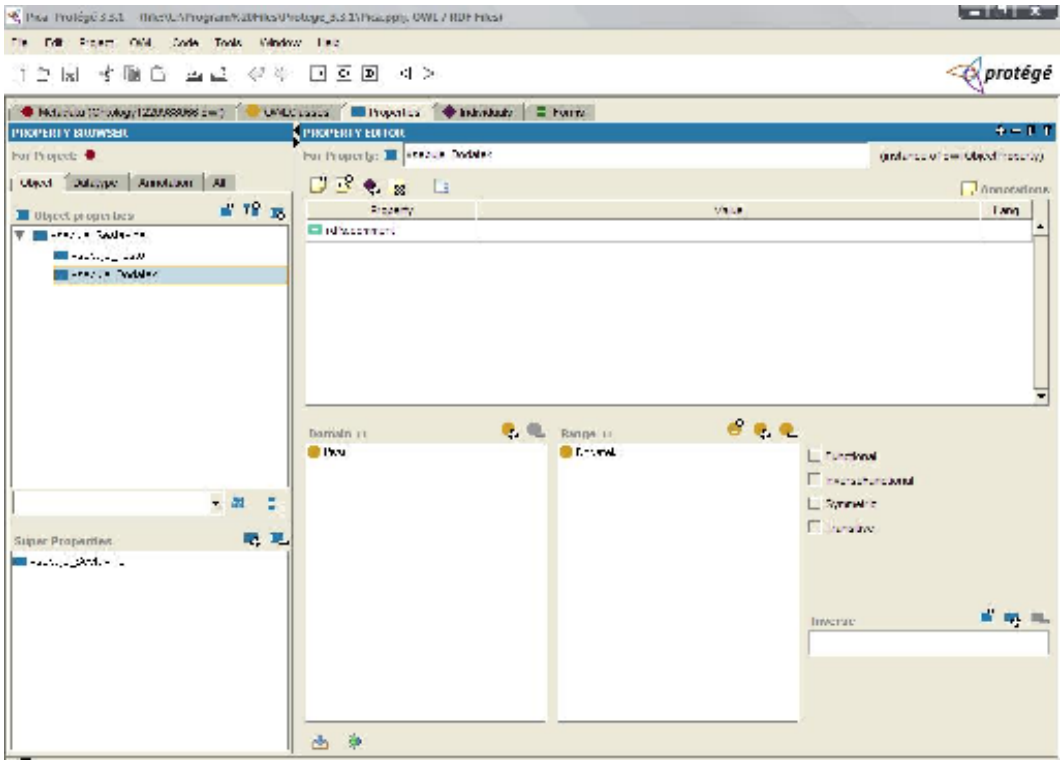


Slika 42: Razred Klasika in njegove lastnosti v orodju Protege

Spodnji izsek OWL zapisa ontologije iz orodja Protege prikazuje definiranje razreda pice Domača, ki je podrazred razreda Pica in je hkrati nepovezan z razredom Klasika.

```
<owl:Class rdf:ID="Domaca">
    <owl:disjointWith>
      <owl:Class rdf:ID="Klasika"/>
    </owl:disjointWith>
    <rdfs:subClassOf>
      <owl:Class rdf:ID="Pica"/>
    </rdfs:subClassOf>
  </owl:Class>
```

Na sliki 43 je v levem zgornjem oknu orodja prikazana struktura lastnosti. V tem primeru sta lastnosti vsebuje_Testo in vsebuje_Dodatek predstavljeni kot bolj specifični lastnosti splošne lastnosti vsebuje_Sestavine. V levem spodnjem oknu je za lastnost vsebuje_Dodatek prikazana njena nadrejena lastnost vsebuje_sestavine. V desnih spodnjih oknih orodja je za lastnost vsebuje_Dodatek določena domena iz razreda Pica in domet iz razreda Dodatek.



Slika 43: Lastnost vsebuje_Dodatek z definirano domeno in dometom v orodju Protege

Izsek iz OWL zapisa ontologije predstavlja definiranje domene in dometa za lastnost vsebuje_Dodatek.

```
<owl:ObjectProperty rdf:ID="vsebuje_Sestavine"/>
<owl:ObjectProperty rdf:about="#vsebuje_Dodatek">
    <rdfs:range rdf:resource="#Dodatek"/>
    <rdfs:domain rdf:resource="#Pica"/>
    <rdfs:subPropertyOf rdf:resource="#vsebuje_Sestavine"/>
</owl:ObjectProperty>
```

Primer ontologije je bil predstavljen v orodjih, ki sta bili najbolje ocenjeni v primerjavi v prejšnjem poglavju. Pri tem je možno videti osnovne pristope obeh orodij h gradnji ontologij in primerjati zapise jezikov RDF in OWL. Prednost orodja Protege v primerjavi z orodjem OntoStudio je predvsem v podpori jeziku OWL, ki je nekoliko bolj izrazno od jezika RDF. Tudi uporabniški vmesnik orodja Protege je bolj intuitiven in pregleden.

11 ZAKLJUČEK

Pri pregledu in primerjavi metodologij za razvoj ontologij sem ugotovil, da je nastalo veliko število metodologij, ki so si med seboj dokaj različne glede na pristop in korake pri gradnji ontologij. Za osnovne kriterije primerjave sem uporabil faze iz standarda za proces razvoja programske opreme (IEEE 1074-1995). V primerjavi sta izstopali predvsem metodologiji METHONTOLOGY in On-To-Knowledge, ki pokrivata največ korakov razvojnega procesa, uporabljata se na več področjih, imata pa tudi edini dobro podporo orodij. Metodologija METHONTOLOGY je podprta z orodjem WebODE, On-To-Knowledge pa z orodjem OntoEdit in delno tudi z njegovim naslednikom OntoStudio. Pri ostalih metodologijah je definiranost korakov razvojnega procesa slabša, pri čemer so določeni koraki razvoja premalo podrobno opisani ali pa sploh niso vključeni v metodologijo. Glavni problem teh metodologij pa je predvsem pomanjkanje podpore orodij, ki bi poenostavile in spodbujale gradnjo ontologij po definiranih korakih. Tako lahko ugotovimo, da ne obstaja metodologija, ki bi pokrivala vse korake razvojnega procesa. Na splošno manjka predvsem podpora procesom upravljanja projekta in procesom po razvoju ontologije (delovanje, podpora, vzdrževanje, upokojitev).

Pri primerjavi orodij za razvoj ontologij je do izraza prišla predvsem razlika med spletnimi orodji, ki so vsa starejša in že nekaj let niso bila deležna posodobitev, in novejšima samostojnima orodjema OntoStudio in Protege. Ti orodji sta osnovani na arhitekturi, ki omogoča preprosto razširljivost z dodajanjem novih modulov in združljivost z drugimi orodji. OntoStudio je plačljivo orodje, ki ga uporablja že kar nekaj razvitih podjetij, njegova slabost pa je, da trenutno še ne podpira razvoja ontologij v jeziku OWL. Protege pa je prosto dostopno orodje z zelo preglednim in intuitivnim uporabniškim vmesnikom, ki podpira tudi razvoj ontologij v jeziku OWL. Prednost spletnih orodij je dobra podpora sodelovanju pri razvoju ontologij.

Iz teh primerjav lahko povzamemo, da bi morali razvijalci metodologij težiti k vzpostavitvi enotne metodologije, ki bi povezovala več pristopov razvoja ontologij in hkrati pokrivala vse faze procesnega razvoja. Podobno pa je tudi pri orodjih prisotno pomanjkanje podpore metodologijam in vsem fazam življenjskega cikla ontologije. Kombinacija takšne enotne metodologije in razvojnega orodja s podporo vsem fazam življenjskega cikla ontologije bi morala biti cilj razvijalcev, saj bi pripomogla k preprostejšemu, bolj sistematičnemu, enotnejšemu in hitrejšemu razvoju ontologij.

Glede razvitosti področja ontologij lahko rečemo, da so ontologije še vedno v fazi prehoda z znanstvenega področja na področja širše uporabe v okviru semantičnega spleta, poslovnih aplikacij, inteligentnih programskih agentov itd. Sicer se ontologije že uporabljajo na mnogih področjih, vendar lahko pričakujemo še bolj množično uporabo. Če vzamemo za primerjavo razvitost metodologij in orodij za razvoj programske opreme, ugotovimo, da je na področju ontologij še dovolj prostora za razvoj. Tako pri ontologijah, na primer, še ni mogoče zaslediti hitrih pristopov (Rapid Application Development - RAD) prototipiranja k razvoju in uporabi ontologij v poslovnih aplikacijah in pristopov k analognim agilnim metodologijam pri razvoju programske opreme.

Ker se ontologije uporabljajo za zajem in skupno rabo znanja, ki postaja v današnji družbi vse bolj pomembno, lahko v prihodnje pričakujemo nadaljnjo rast uporabe ontologij in posledično tudi razvoj učinkovitejših metodologij in orodij.

12 PRILOGE

12.1 Priloga 1: Poročilo primerjave metodologij za razvoj ontologij iz programa DEXi

Drevo kriterijev

Kriterij	Opis
Razvitost metodologije	
├**Strategija gradnje**	
├Plan življenjskega cikla	
├Odvisnost razvoja od aplikacije	
├Strategija identificiranja konceptov	
└Uporaba jedrne ontologije	
├**Definiranost procesa razvoja ontologije**	
├**Upravljanje projekta**	
├Zasnova projekta	
├Nadzorovanje	
└Upravljanje kakovosti ontologije	
├**Razvojno orientirani procesi**	
├**Procesi pred razvojem**	
├Proučevanje okolja	
└Proučevanje izvedljivosti	
├**Procesi razvoja**	
├Obdelava zahtev	
├Oblikovanje	
└Implementacija	
└**Procesi po razvoju**	
├Namestitev	
├Delovanje	
├Podpora	
├Vzdrževanje	
└Upokojitev	
└**Spremljevalni procesi**	
├Zajem znanja	
├Vrednotenje	
├Upravljanje konfiguracij	
├Dokumentiranje	
└Usposabljanje	
├Uporaba metodologije	
└Podpora orodja	

Zaloge vrednosti

Kriterij	Zaloga vrednosti
Razvitost metodologije	**Slabo razvita**; Srednje razvita; ***Dobro razvita***
Strategija gradnje	**Slabo definirana**; ***Dobro definirana***
Plan življenjskega cikla	**Ni predvideno**; ***Iterativno***; ***Inkrementalno***; ***Razvijanje prototipov***
Odvisnost razvoja od aplikacije	Ni odvisen; Delno odvisen; Odvisen
Strategija identificiranja konceptov	Od zgoraj navzdol; Od spodaj navzgor; Od sredine navzven
Uporaba jedrne ontologije	Da; Ne
Definiranost procesa razvoja ontologije	**Slaba**; Srednja; ***Dobra***
Upravljanje projekta	**Slabo definirano**; Delno definirano; ***Dobro definirano***
Zasnova projekta	**Ni predvideno**; Predvideno; ***Opisano***
Nadzorovanje	**Ni predvideno**; Predvideno; ***Opisano***
Upravljanje kakovosti ontologije	**Ni predvideno**; Predvideno; ***Opisano***
Razvojno orientirani procesi	**Slabo definirano**; Delno definirano; ***Dobro definirano***
Procesi pred razvojem	**Slabo definirano**; Delno definirano; ***Dobro definirano***
Proučevanje okolja	**Ni predvideno**; Predvideno; ***Opisano***
Proučevanje izvedljivosti	**Ni predvideno**; Predvideno; ***Opisano***
Procesi razvoja	**Slabo definirano**; Delno definirano; ***Dobro definirano***
Obdelava zahtev	**Ni predvideno**; Predvideno; ***Opisano***
Oblikovanje	**Ni predvideno**; Predvideno; ***Opisano***
Implementacija	**Ni predvideno**; Predvideno; ***Opisano***
Procesi po razvoju	**Slabo definirano**; Delno definirano; ***Dobro definirano***
Namestitev	**Ni predvideno**; Predvideno; ***Opisano***
Delovanje	**Ni predvideno**; Predvideno; ***Opisano***
Podpora	**Ni predvideno**; Predvideno; ***Opisano***
Vzdrževanje	**Ni predvideno**; Predvideno; ***Opisano***
Upokojitev	**Ni predvideno**; Predvideno; ***Opisano***
Spremljevalni procesi	**Slabo definirano**; Delno definirano; ***Dobro definirano***
Zajem znanja	**Ni predvideno**; Predvideno; ***Opisano***
Vrednotenje	**Ni predvideno**; Predvideno; ***Opisano***
Upravljanje konfiguracij	**Ni predvideno**; Predvideno; ***Opisano***
Dokumentiranje	**Ni predvideno**; Predvideno; ***Opisano***
Usposabljanje	**Ni predvideno**; Predvideno; ***Opisano***
Uporaba metodologije	**Na ozjem podrocju**; Delno razsirjena; ***Vec podrocij***
Podpora orodja	**Ni orodij**; Delno podprto; ***V celoti podprto***

Povprecne uteži

Kriterij	Lokalne	Globalne
Razvitost metodologije		
Strategija gradnje	6	6
Plan življenjskega cikla	100	6
Odvisnost razvoja od aplikacije	0	0
Strategija identificiranja konceptov	0	0
Uporaba jedrne ontologije	0	0
Definiranost procesa razvoja ontologije	66	66
Upravljanje projekta	10	7
Zasnova projekta	24	2
Nadzorovanje	38	3
Upravljanje kakovosti ontologije	38	3
Razvojno orientirani procesi	70	46
Procesi pred razvojem	17	8
Proučevanje okolja	50	4
Proučevanje izvedljivosti	50	4
Procesi razvoja	58	27
Obdelava zahtev	30	8
Oblikovanje	39	11
Implementacija	30	8
Procesi po razvoju	25	12
Namestitev	25	3
Delovanje	16	2
Podpora	20	2
Vzdrževanje	35	4
Upokojitev	4	1
Spremljevalni procesi	20	13
Zajem znanja	29	4
Vrednotenje	25	3
Upravljanje konfiguracij	11	2
Dokumentiranje	25	3
Usposabljanje	9	1
Uporaba metodologije	9	9
Podpora orodja	18	18

Rezultati vrednotenja

Kriterij	Cyc	Enterprise Ontology	TOVE	KACTUS	METHONTOLOGY
Razvitost metodologije	**Slabo razvita**	**Slabo razvita**	Srednje razvita	Srednje razvita	***Dobro razvita***
├**Strategija gradnje**	***Dobro definirana***	**Slabo definirana**	***Dobro definirana***	***Dobro definirana***	***Dobro definirana***
├Plan življenjskega cikla	***Razvijanje prototipov***	**Ni predvideno**	***Razvijanje prototipov***	***Razvijanje prototipov***	***Razvijanje prototipov***
├Odvisnost razvoja od aplikacije	Ni odvisen	Ni odvisen	Delno odvisen	Odvisen	Ni odvisen
├Strategija identificiranja konceptov	*	Od sredine navzven	Od sredine navzven	Od zgoraj navzdol	Od sredine navzven
└Uporaba jedrne ontologije	Da	Ne	Ne	Ne	Ne
├**Definiranost procesa razvoja ontologije**	**Slaba**	**Slaba**	Srednja	Srednja	Srednja
├**Upravljanje projekta**	**Slabo definirano**	**Slabo definirano**	**Slabo definirano**	**Slabo definirano**	Delno definirano
├Zasnova projekta	**Ni predvideno**	**Ni predvideno**	**Ni predvideno**	**Ni predvideno**	**Ni predvideno**
├Nadzorovanje	**Ni predvideno**	**Ni predvideno**	**Ni predvideno**	**Ni predvideno**	Predvideno
└Upravljanje kakovosti ontologije	**Ni predvideno**	**Ni predvideno**	**Ni predvideno**	**Ni predvideno**	Predvideno
├**Razvojno orientirani procesi**	**Slabo definirano**	**Slabo definirano**	Delno definirano	Delno definirano	Delno definirano
├**Procesi pred razvojem**	**Slabo definirano**	**Slabo definirano**	**Slabo definirano**	**Slabo definirano**	**Slabo definirano**
├Proučevanje okolja	**Ni predvideno**	**Ni predvideno**	**Ni predvideno**	**Ni predvideno**	**Ni predvideno**
└Proučevanje izvedljivosti	**Ni predvideno**	**Ni predvideno**	**Ni predvideno**	**Ni predvideno**	**Ni predvideno**
├**Procesi razvoja**	**Slabo definirano**	Delno definirano	***Dobro definirano***	***Dobro definirano***	***Dobro definirano***
├Obdelava zahtev	**Ni predvideno**	Predvideno	***Opisano***	Predvideno	***Opisano***
├Oblikovanje	**Ni predvideno**	**Ni predvideno**	***Opisano***	***Opisano***	***Opisano***
└Implementacija	Predvideno	Predvideno	***Opisano***	Predvideno	***Opisano***
└**Procesi po razvoju**	**Slabo definirano**	**Slabo definirano**	**Slabo definirano**	**Slabo definirano**	**Slabo definirano**
├Namestitev	**Ni predvideno**	**Ni predvideno**	**Ni predvideno**	**Ni predvideno**	**Ni predvideno**
├Delovanje	**Ni predvideno**	**Ni predvideno**	**Ni predvideno**	**Ni predvideno**	**Ni predvideno**
├Podpora	**Ni predvideno**	**Ni predvideno**	**Ni predvideno**	**Ni predvideno**	**Ni predvideno**
├Vzdrževanje	**Ni predvideno**	**Ni predvideno**	**Ni predvideno**	**Ni predvideno**	Predvideno
└Upokojitev	**Ni predvideno**	**Ni predvideno**	**Ni predvideno**	**Ni predvideno**	**Ni predvideno**
└**Spremljevalni procesi**	Delno definirano	Delno definirano	Delno definirano	Delno definirano	***Dobro definirano***
├Zajem znanja	Predvideno	Predvideno	Predvideno	Predvideno	***Opisano***
├Vrednotenje	**Ni predvideno**	Predvideno	Predvideno	Predvideno	***Opisano***
├Upravljanje konfiguracij	**Ni predvideno**	**Ni predvideno**	**Ni predvideno**	**Ni predvideno**	***Opisano***
├Dokumentiranje	**Ni predvideno**	Predvideno	Predvideno	Predvideno	***Opisano***
└Usposabljanje	**Ni predvideno**	**Ni predvideno**	**Ni predvideno**	**Ni predvideno**	**Ni predvideno**
├Uporaba metodologije	Delno razsirjena	**Na ozjem podrocju**	**Na ozjem podrocju**	Delno razsirjena	***Vec podrocij***
└Podpora orodja	Delno podprto	Delno podprto	**Ni orodij**	**Ni orodij**	***V celoti podprto***

Kriterij	SENSUS	DOGMA	On-To-Knowledge	HCOME	DILIGENT
Razvitost metodologije	**Slabo razvita**	**Slabo razvita**	***Dobro razvita***	Srednje razvita	Srednje razvita
├**Strategija gradnje**	**Slabo definirana**	**Slabo definirana**	***Dobro definirana***	***Dobro definirana***	***Dobro definirana***
│├Plan življenjskega cikla	**Ni predvideno**	**Ni predvideno**	***Razvijanje prototipov***	***Inkrementalno***	***Inkrementalno***
│├Odvisnost razvoja od aplikacije	Delno odvisen	Odvisen	Odvisen	Ni odvisen	Ni odvisen
│├Strategija identificiranja konceptov	Od sredine navzven	Od zgoraj navzdol	Od sredine navzven	Od sredine navzven	Od sredine navzven
│└Uporaba jedrne ontologije	Da	Da	*	Da	Da
├**Definiranost procesa razvoja ontologije**	**Slaba**	**Slaba**	***Dobra***	Srednja	Srednja
│├**Upravljanje projekta**	**Slabo definirano**	**Slabo definirano**	***Dobro definirano***	**Slabo definirano**	**Slabo definirano**
││├Zasnova projekta	**Ni predvideno**	**Ni predvideno**	***Opisano***	**Ni predvideno**	**Ni predvideno**
││├Nadzorovanje	**Ni predvideno**	**Ni predvideno**	***Opisano***	**Ni predvideno**	**Ni predvideno**
││└Upravljanje kakovosti ontologije	**Ni predvideno**	**Ni predvideno**	***Opisano***	**Ni predvideno**	**Ni predvideno**
│├**Razvojno orientirani procesi**	**Slabo definirano**	**Slabo definirano**	***Dobro definirano***	Delno definirano	Delno definirano
││├**Procesi pred razvojem**	**Slabo definirano**	**Slabo definirano**	***Dobro definirano***	**Slabo definirano**	**Slabo definirano**
│││├Proučevanje okolja	**Ni predvideno**	**Ni predvideno**	Predvideno	**Ni predvideno**	**Ni predvideno**
│││└Proučevanje izvedljivosti	**Ni predvideno**	**Ni predvideno**	***Opisano***	**Ni predvideno**	**Ni predvideno**
││├**Procesi razvoja**	Delno definirano	Delno definirano	***Dobro definirano***	***Dobro definirano***	***Dobro definirano***
│││├Obdelava zahtev	Predvideno	Predvideno	***Opisano***	Predvideno	Predvideno
│││├Oblikovanje	**Ni predvideno**	Predvideno	***Opisano***	***Opisano***	***Opisano***
│││└Implementacija	***Opisano***	Predvideno	***Opisano***	Predvideno	Predvideno
││└**Procesi po razvoju**	**Slabo definirano**	**Slabo definirano**	***Dobro definirano***	**Slabo definirano**	**Slabo definirano**
│││├Namestitev	**Ni predvideno**	**Ni predvideno**	Predvideno	**Ni predvideno**	**Ni predvideno**
│││├Delovanje	**Ni predvideno**	**Ni predvideno**	***Opisano***	**Ni predvideno**	**Ni predvideno**
│││├Podpora	**Ni predvideno**	**Ni predvideno**	***Opisano***	**Ni predvideno**	**Ni predvideno**
│││├Vzdrževanje	**Ni predvideno**	**Ni predvideno**	Predvideno	Predvideno	Predvideno
│││└Upokojitev	**Ni predvideno**	**Ni predvideno**	**Ni predvideno**	**Ni predvideno**	**Ni predvideno**
│└**Spremljevalni procesi**	**Slabo definirano**	Delno definirano	***Dobro definirano***	Delno definirano	Delno definirano
││├Zajem znanja	**Ni predvideno**	Predvideno	***Opisano***	Predvideno	Predvideno
││├Vrednotenje	**Ni predvideno**	**Ni predvideno**	Predvideno	Predvideno	Predvideno
││├Upravljanje konfiguracij	**Ni predvideno**	Predvideno	Predvideno	Predvideno	**Ni predvideno**
││├Dokumentiranje	**Ni predvideno**	**Ni predvideno**	***Opisano***	Predvideno	**Ni predvideno**
││└Usposabljanje	**Ni predvideno**	**Ni predvideno**	***Opisano***	**Ni predvideno**	**Ni predvideno**
├Uporaba metodologije	Delno razsirjena	Delno razsirjena	***Vec podrocij***	**Na ozjem podrocju**	Delno razsirjena
└Podpora orodja	Delno podprto	***V celoti podprto***	***V celoti podprto***	***V celoti podprto***	Delno podprto

Kriterij	UPON
Razvitost metodologije	Srednje razvita
├**Strategija gradnje**	***Dobro definirana***
├Plan življenjskega cikla	***Inkrementalno***
├Odvisnost razvoja od aplikacije	Odvisen
├Strategija identificiranja konceptov	Od sredine navzven
└Uporaba jedrne ontologije	Ne
├**Definiranost procesa razvoja ontologije**	Srednja
├**Upravljanje projekta**	**Slabo definirano**
├Zasnova projekta	**Ni predvideno**
├Nadzorovanje	**Ni predvideno**
└Upravljanje kakovosti ontologije	**Ni predvideno**
├**Razvojno orientirani procesi**	Delno definirano
├**Procesi pred razvojem**	**Slabo definirano**
├Proučevanje okolja	**Ni predvideno**
└Proučevanje izvedljivosti	**Ni predvideno**
├**Procesi razvoja**	***Dobro definirano***
├Obdelava zahtev	***Opisano***
├Oblikovanje	***Opisano***
└Implementacija	***Opisano***
└**Procesi po razvoju**	**Slabo definirano**
├Namestitev	**Ni predvideno**
├Delovanje	**Ni predvideno**
├Podpora	**Ni predvideno**
├Vzdrževanje	**Ni predvideno**
└Upokojitev	**Ni predvideno**
└**Spremljevalni procesi**	***Dobro definirano***
├Zajem znanja	***Opisano***
├Vrednotenje	***Opisano***
├Upravljanje konfiguracij	**Ni predvideno**
├Dokumentiranje	***Opisano***
└Usposabljanje	**Ni predvideno**
├Uporaba metodologije	Delno razsirjena
└Podpora orodja	***V celoti podprto***

12.2 Priloga 2: Poročilo primerjave orodij za razvoj ontologij iz programa DEXi

Drevo kriterijev

Kriterij	Opis
Razvitost orodja	
├**Prijaznost do uporabnika**	
│ ├Posodabljanje orodja	
│ ├Uporabniški vmesnik	
│ ├Dostopnost	
│ └Podpora metodologiji	
├**Arhitektura in razširljivost orodja**	
│ ├Arhitektura programske opreme	
│ ├Razširljivost	
│ ├Shranjevanje ontologij	
│ └Združljivost z drugimi orodji	
├**Predstavitev znanja**	
│ ├Predstavitev modela znanja	
│ ├Jezik za kreiranje aksiomov	
│ ├Uvažanje iz jezikov	
│ └Izvažanje v jezike	
├**Vključene storitve sklepanja**	
│ ├Vgrajen mehanizem sklepanja	
│ ├Preverjanje konsistentnosti in omejitev	
│ └Avtomatska klasifikacija	
└**Uporabnost orodja**	
├Grafični prikaz taksonomij	
├Omogočeno skupinsko delo	
└Knjižnica ontologij	

Zaloge vrednosti

Kriterij	Zaloga vrednosti
Razvitost orodja	**Slabo razvito**; Srednje razvito; ***Dobro razvito***
├**Prijaznost do uporabnika**	**Slaba**; Delna; ***Dobra***
├Posodabljanje orodja	**Ni posodobitev**; ***Redno***
├Uporabniški vmesnik	**Zastarel**; Primeren; ***Moderen***
├Dostopnost	**Ukinjeno**; Placljivo; ***Prosto dostopno***
└Podpora metodologiji	**Ne**; ***Da***
├**Arhitektura in razširljivost orodja**	**Slabše zasnovano**; Srednje zasnovano; ***Dobro zasnovano***
├Arhitektura programske opreme	Odjemalec/strežnik; ***Samostojno orodje***; ***3 nivojska arhitektura***
├Razširljivost	**Ni mogoca**; ***Mogoca***
├Shranjevanje ontologij	Datoteke; ***Podatkovna baza***
└Združljivost z drugimi orodji	**Ne**; ***Da***
├**Predstavitev znanja**	**Slabo pokrito**; Srednje pokrito; ***Dobro pokrito***
├Predstavitev modela znanja	**Opisna logika**; ***Strukturno***
├Jezik za kreiranje aksiomov	**Ne**; ***Da***
├Uvažanje iz jezikov	**Slabo pokrito**; Srednje pokrito; ***Dobro pokrito***
└Izvažanje v jezike	**Slabo pokrito**; Srednje pokrito; ***Dobro pokrito***
├**Vključene storitve sklepanja**	**Slabo pokrito**; Srednje pokrito; ***Dobro pokrito***
├Vgrajen mehanizem sklepanja	**Ne**; ***Da***
├Preverjanje konsistentnosti in omejitev	**Ne**; ***Da***
└Avtomatska klasifikacija	**Ne**; ***Da***
└**Uporabnost orodja**	**Slabo pokrito**; Srednje pokrito; ***Dobro pokrito***
├Grafični prikaz taksonomij	**Ne**; ***Da***
├Omogočeno skupinsko delo	**Ne**; ***Da***
└Knjižnica ontologij	**Ne**; ***Da***

Povprecne uteži

Kriterij	Lokalne	Globalne
Razvitost orodja		
├**Prijaznost do uporabnika**	36	36
│ ├Posodabljanje orodja	36	13
│ ├Uporabniški vmesnik	30	11
│ ├Dostopnost	19	7
│ └Podpora metodologiji	15	5
├**Arhitektura in razširljivost orodja**	31	31
│ ├Arhitektura programske opreme	22	7
│ ├Razširljivost	39	12
│ ├Shranjevanje ontologij	20	6
│ └Združljivost z drugimi orodji	20	6
├**Predstavitev znanja**	14	14
│ ├Predstavitev modela znanja	12	2
│ ├Jezik za kreiranje aksiomov	28	4
│ ├Uvažanje iz jezikov	30	4
│ └Izvažanje v jezike	30	4
├**Vključene storitve sklepanja**	10	10
│ ├Vgrajen mehanizem sklepanja	56	5
│ ├Preverjanje konsistentnosti in omejitev	33	3
│ └Avtomatska klasifikacija	11	1
└**Uporabnost orodja**	10	10
├Grafični prikaz taksonomij	33	3
├Omogočeno skupinsko delo	33	3
└Knjižnica ontologij	33	3

Rezultati vrednotenja

Kriterij	Ontolingua	WebOnto	OntoEdit	OntoStudio	WebODE
Razvitost orodja	**Slabo razvito**	Srednje razvito	Srednje razvito	***Dobro razvito***	***Dobro razvito***
├**Prijaznost do uporabnika**	**Slaba**	**Slaba**	**Slaba**	Delna	Delna
│├Posodabljanje orodja	**Ni posodobitev**	**Ni posodobitev**	**Ni posodobitev**	***Redno***	**Ni posodobitev**
│├Uporabniški vmesnik	**Zastarel**	**Zastarel**	Primeren	***Moderen***	Primeren
│├Dostopnost	***Prosto dostopno***	***Prosto dostopno***	**Ukinjeno**	Placljivo	***Prosto dostopno***
│└Podpora metodologiji	**Ne**	**Ne**	***Da***	**Ne**	***Da***
├**Arhitektura in razširljivost orodja**	**Slabše zasnovano**	**Slabše zasnovano**	***Dobro zasnovano***	***Dobro zasnovano***	***Dobro zasnovano***
│├Arhitektura programske opreme	Odjemalec/strežnik	Odjemalec/strežnik	***Samostojno orodje***	***Samostojno orodje***	***3 nivojska arhitektura***
│├Razširljivost	**Ni mogoca**	**Ni mogoca**	***Mogoca***	***Mogoca***	***Mogoca***
│├Shranjevanje ontologij	Datoteke	Datoteke	***Podatkovna baza***	***Podatkovna baza***	***Podatkovna baza***
│└Združljivost z drugimi orodji	***Da***	***Da***	***Da***	***Da***	***Da***
├**Predstavitev znanja**	Srednje pokrito	Srednje pokrito	***Dobro pokrito***	***Dobro pokrito***	***Dobro pokrito***
│├Predstavitev modela znanja	***Strukturno***	***Strukturno***	***Strukturno***	***Strukturno***	***Strukturno***
│├Jezik za kreiranje aksiomov	***Da***	***Da***	***Da***	***Da***	***Da***
│├Uvažanje iz jezikov	Srednje pokrito	**Slabo pokrito**	***Dobro pokrito***	***Dobro pokrito***	***Dobro pokrito***
│└Izvažanje v jezike	Srednje pokrito	**Slabo pokrito**	***Dobro pokrito***	***Dobro pokrito***	***Dobro pokrito***
├**Vključene storitve sklepanja**	**Slabo pokrito**	***Dobro pokrito***	***Dobro pokrito***	***Dobro pokrito***	***Dobro pokrito***
│├Vgrajen mehanizem sklepanja	**Ne**	***Da***	***Da***	***Da***	***Da***
│├Preverjanje konsistentnosti in omejitev	**Ne**	***Da***	***Da***	***Da***	***Da***
│└Avtomatska klasifikacija	**Ne**	**Ne**	**Ne**	**Ne**	**Ne**
└**Uporabnost orodja**	***Dobro pokrito***	***Dobro pokrito***	Srednje pokrito	Srednje pokrito	Srednje pokrito
├Grafični prikaz taksonomij	***Da***	***Da***	**Ne**	**Ne**	***Da***
├Omogočeno skupinsko delo	***Da***	***Da***	***Da***	***Da***	***Da***
└Knjižnica ontologij	***Da***	***Da***	***Da***	***Da***	**Ne**

Kriterij	Protege
Razvitost orodja	***Dobro razvito***
├**Prijaznost do uporabnika**	***Dobra***
│ ├Posodabljanje orodja	***Redno***
│ ├Uporabniški vmesnik	***Moderen***
│ ├Dostopnost	***Prosto dostopno***
│ └Podpora metodologiji	**Ne**
├**Arhitektura in razširljivost orodja**	***Dobro zasnovano***
│ ├Arhitektura programske opreme	***Samostojno orodje***
│ ├Razširljivost	***Mogoca***
│ ├Shranjevanje ontologij	***Podatkovna baza***
│ └Združljivost z drugimi orodji	***Da***
├**Predstavitev znanja**	***Dobro pokrito***
│ ├Predstavitev modela znanja	***Strukturno***
│ ├Jezik za kreiranje aksiomov	***Da***
│ ├Uvažanje iz jezikov	***Dobro pokrito***
│ └Izvažanje v jezike	***Dobro pokrito***
├**Vključene storitve sklepanja**	***Dobro pokrito***
│ ├Vgrajen mehanizem sklepanja	***Da***
│ ├Preverjanje konsistentnosti in omejitev	***Da***
│ └Avtomatska klasifikacija	**Ne**
└**Uporabnost orodja**	***Dobro pokrito***
├Grafični prikaz taksonomij	***Da***
├Omogočeno skupinsko delo	***Da***
└Knjižnica ontologij	***Da***

VIRI IN LITERATURA

[1] A. Gomez-Perez, O. Corcho, Ontology Languages for the Semantic Web, Universidad Politécnica de Madrid, 2002. Dostopno na: http://www.cs.umbc.edu/771/papers/ieeeIntelligentSystems/webservices/ontologyLanguages.pdf

[2] A. Gomez-Perez, M. Fernandez-Lopez, O. Corcho, Ontology languages. Dostopno na: http://www.cs.man.ac.uk/~ocorcho/InvitedTalks/OntologyLanguages.pdf

[3] B. Matthews., Semantic Web Technologies. Dostopno na: http://www.jisc.ac.uk/uploaded_documents/jisctsw_05_02bpdf.pdf

[4] Lavbič, M. Krisper, »Semantika podatkov in ontologije«, Uporabna informatika, št. 3, letnik XIII, sekcija Razprave, 2005.

[5] Deliverable 1.3: A survey on ontology tools, Vrije Universiteit Amsterdam, 2002. Dostopno na: http://www.aifb.uni-karlsruhe.de/WBS/ysu/publications/OntoWeb_Del_1-3.pdf

[6] Deliverable 1.4: A survey on methodologies for developing, maintaining, evaluating and reengineering ontologies, Vrije Universiteit Amsterdam, 2002, str 41-60. Dostopno na: http://www.aifb.uni-karlsruhe.de/WBS/ysu/publications/OntoWeb_Del_1-4.pdf

[7] E. S. Abou-Zeid, »What can ontologist learn from knowledge management?«, Journal of Computer Information Systems, april, 2003, str. 109-117.

[8] F. Schiappelli, Ontology Building Methodologies and Tools, Rim, 2003. Dostopno na: http://www.dsi.uniroma1.it/~estrinfo/sws/2003_09_19/Ontology%20Building%20Methodologies%20and%20Tools%20-%20Schiappelli.ppt

[9] H. Beck, H. S. Pinto, Overview of Approach, Methodologies, Standards, and Tools for Ontologies. Dostopno na: http://www.fao.org/agris/aos/documents/backgroundpaper.pdf

[10] H. S. Pinto, S. Staab, C. Tempich, DILIGENT: Towards a ne-grained methodology for DIstributed, Loosely-controlled and evolvInG Engineering of oNTologies. Dostopno na: http://www.uni-koblenz.de/~staab/Research/Publications/2004/ECAI04_dilligent_arguments_C0414_final.pdf

[11] J. Davies, R. Studer, P. Warren, Semantic Web Technologies - trends and research in ontology-based systems. 2006, Chichester, England: John Wiley & Sons, pogl. 1, 6, 9.

[12] K. Kotis, G. A. Vouros, J. P. Alonso, »HCOME: A tool-supported methodology for engineering living ontologies«, Semantic Web and Databases. Second International Workshop - SWDB 2004, volume 3372 of LNCS.

[13] M. Fernandez-Lopez, Overview Of Methodologies For Building Ontologies, Universidad Politécnica de Madrid, Spain, 1999. Dostopno na: http://www.lsi.upc.es/~bejar/aia/aia-web/4-fernandez.pdf

[14] M. Krisper, R. Rupnik, M. Bajec, A. Rožanec, A. Zrnec, D. Vavpotič, Enotna metodologija razvoja informacijskih sistemov, Uvod, druga izdaja, december 2003, Univerza v Ljubljani, Fakulteta za Računalništvo in informatiko, pogl. 1.

[15] M. Missikoff, R. Navigli, Applying the Unified Process to Large-scale Ontology Building, Università di Roma "La Sapienza", Roma, Italy, 2005. Dostopno na: http://www.dsi.uniroma1.it/~navigli/pubs/IFAC_2005_Missikoff_Navigli.pdf

[16] N. F. Noy, D. L. McGuinness, Ontology Development 101: A Guide to Creating Your First Ontology, Stanford University. Dostopno na: http://ksl.stanford.edu/people/dlm/papers/ontology-tutorial-noy-mcguinness.pdf

[17] O. Corcho, A. Gomez-Perez, A Roadmap to Ontology Specification Languages, 12th International Conference on Knowledge Engineering and Knowledge Management. Dostopno na: http://www.cs.man.ac.uk/~ocorcho/documents/ekaw00_CorchoGomezPerez.pdf

[18] O. Corcho, M. Fernandez-Lopez, A. Gomez-Perez, »Methodologies, tools and languages for building ontologies. Where is their meeting point?«, v Data & Knowledge Engineering, 2003, Elsevier Science Publishers B. V, Amsterdam.

[19] Ontology (Information science). Dostopno na: http://en.wikipedia.org/wiki/Ontology_(information_science)

[20] Ontology. Dostopno na: http://en.wikipedia.org/wiki/Ontology

[21] OntoStudio 2.1 - professional knowledge engineering. Dostopno na: http://www.ontoprise.de/fileadmin/user_upload/Flyer_EN/Flyer_OntoStudio_en.pdf

[22] Opis orodja Protege. Dostopno na: http://protege.stanford.edu/overview

[23] Orodje WebODE. Dostopno na: http://webode.dia.fi.upm.es/WebODEWeb/index.html

[24] OWL Web Ontology Language. Dostopno na: http://www.w3.org/TR/owl-features

[25] P. Spyns, R. Meersman, M. Jarrar, »Data modeling versus Ontology engineering«, ACM SIGMOD Record, Volume 31, Issue 4, 2002, str. 12-17.

[26] R. Garcia Gonzalez, A Semantic Web approach to Digital Rights Management, Ph.D. Thesis, 2005, Barcelona, pogl. Methodology. Dostopno na: http://rhizomik.net/~roberto/thesis/Thesis.pdf

[27] R. Meersman, Mustafa Jarrar, An Architecture For Practical Ontology Engineering and Deployment: the DOGMA Approach, Vrije Univerisiteit Brussel, Belgija, 2003. Dostopno na: http://lsirwww.epfl.ch/courses/dip/2002ws/doogma.pdf

[28] Resource Description Framework. Dostopno na: http://en.wikipedia.org/wiki/Resource_Description_Framework

[29] S. Bechhofer, Ontology Language Standardisation Efforts, Vrije Universiteit Amsterdam, Nizozemska, 2002. Dostopno na: http://www.ontoweb.org/About/Deliverables/d4.0.pdf

[30] T. Berners-Lee, J. Hendler, O. Lassila, The Semantic Web, Scientific American Magazine, maj, 2001.

[31] T. Gruber, Ontology, 2007. Dostopno na: http://tomgruber.org/writing/ontology-in-encyclopedia-of-dbs.pdf

[32] T. R. Gruber, »Toward principles for the design of ontologies used for knowledge sharing«, International Journal of Human-Computer Studies, Vol. 43, Issues 4-5, November 1995, str. 907-928.

[33] Y. Sure, Methodology, Tools & Case Studies for Ontology based Knowledge Management, v Institute AIFB, 2003, University of Karlsruhe.

IZJAVA

Spodaj podpisani Boštjan Senica izjavljam, da sem diplomsko delo izdelal samostojno pod mentorstvom izr. prof. dr. Marjana Krisperja.

Ljubljana, september 2008

Boštjan Senica