TALLINNA TEHNIKAÜLIKOOL

Infotehnoloogia teaduskond

Kevin Laanemägi, 176972 IAPM

SEMANTILINE KOOSVÕIME ANDMETE ÜHEKORDSE KÜSIMISE KESKKONNAS PROTOTÜÜBI NÄITEL

Magistritöö

Juhendaja: Jaak Tepandi, PhD

Tallinn 2020

Autorideklaratsioon

Kinnitan, et olen koostanud lõputöö iseseisvalt ning seda ei ole varem kellegi teise poolt kasutatud kaitsmisel. Kõik töö koostamisel kasutatud teiste autorite tööd, olulised seisukohad, kirjandusallikatest ja mujalt pärinevad andmed on töös viidatud.

Autor: Kevin Laanemägi

11.05.2020

Annotatsioon

Käesoleva magistritöö eesmärgiks oli uurida, kuidas toimib semantiline koosvõime andmete ühekordse küsimise keskkonnas kahe riigi vahel, ning uuringu tulemuste põhjal luua semantika vahend, mis võimaldaks läbi viia kolme kodanikele suunatud riikide vahelist toimingut kasutades TOOP-i [1] nõudeid.

Töö tulemiks on eksperimentaalne prototüüp, milles on rakendatud semantilist koosvõimet kolme ülesande näitel. Töös kasutatakse ontoloogiaid ülesannete jaoks vajaminevate andmete kirjeldamiseks ning ontoloogiate jooksutamiseks kasutatakse Apache Jena raamistikku.

Prototüüp on arendatud kasutades Java programmeerimiskeelt ja selle raamistikke. Valminud prototüübi loomisel kasutatud võimalusi on võimalik kasutada andmete ühekordse küsimise printsiibi projektis.

Lõputöö on kirjutatud eesti keeles ning sisaldab teksti 54 leheküljel, 4 peatükki, 16 joonist ja 4 tabelit.

Semantic Interoperability in the Once-Only Principle Environment on the Example of Prototype

Abstract

Semantic web is new area of research in e-government applications and in legal systems [2]. Semantics as an interaction of linguistics has been studied since the time of Aristotle. With the IT revolution, this topic has become relevant in various fields like Information Science and Artificial Intelligence [3]. Different e-governments applications are describing data differently: semantics differ in structure, terminology and linguistics. That makes processing cross-border applications more difficult. To solve this issue, different approaches are being developed, which aim to create common semantics.

The purpose of the master's thesis is to research how semantic interoperability functions between two countries and based on the research results semantic tool is developed that uses semantic interoperability in the Once-Only Principle environment. Semantic tool must allow three procedures, which are aimed for citizens: applying for allowances, applying for social benefits and applying for pension. Semantic tool uses TOOP [1] requirements. As an example, a prototype is developed, which must meet the aforementioned requirements and also the extra requirements created in the thesis.

Firstly, requirements for prototype are designed. Secondly, three different ontologies are created, each for their own purpose. Two are made for storing their respective country's data and one ontology is for mapping countries' ontology to common ontology. For developing semantic tool, architecture is designed and tools as well as frameworks are chosen. Semantic tool is developed using Java progamming language and Java based frameworks.

The result of the master's thesis is an experimental prototype, that uses semantic interoperability in the Once-Only Principle environment. Additionally, this thesis

provides an experience in ensuring semantic interoperability in OOP environment. The prototype enables user to apply for three required procedures and successfully processes the applications. Prototype meets the requirements.

The thesis is in Estonian and contains 54 pages of text, 4 chapters, 16 figures, 4 tables.

Mõisted ja lühendid

Semantika - Semantika on keeleteaduse haru, mis uurib keeleüksuste tähendusi ning nende tähenduste muutumist, keele ja reaalsete objektide ning mõtlemise suhteid [4].

TOOP (*The only once principle*) - Ühekordse andmete küsimise printsiip [1].

SDGR (*Single Digital Gateway Regulation*) - Ühtse digivärava regulatsioon [5].

Ontoloogia - Ontoloogia defineerib terminid, mida kasutatakse teadmiste kirjeldamiseks ja esitlemiseks [3].

SDG (*Single Digital Gateway*) - Ühtne digivärav [5].

Protégé - Tarkvara ontoloogia loomiseks [6].

Apache Jena - Vabavaraline, avatud lähtekoodiga Java raamistik semantilise veebi ja seotud andmetega [7] rakenduste ehitamiseks [8].

OCNL - On kontrollitud naturaalne keel ontoloogiate kirjutamiseks, mis ühildub OWL2 ja SWRL standardiga [9].

Generic programming - koodi kirjutamise ja disaini viis, kus algoritmid on kirjutatud parameetriliste tüüpide terminitega, et suurenda taaskasutust [10].

Sisukord

Joonised

Tabelid

Peatükk 1

Sissejuhatus

Antud magistritöö eesmärk on uurida, kuidas toimib semantiline koosvõime andmete ühekordse küsimise keskkonnas kahe riigi vahel ning uuringu tulemuste põhjal luua prototüüp, mis teostab kolme ülesannet kasutades TOOP-i[2] nõudeid. Töö kajastab erinevaid võimalusi ontoloogiate vastavusse seadmiseks ja annab ülevaate levinumatest programmidest ontoloogiate loomiseks. Töös räägitakse, kuidas sellised süsteemid töötavad, tuuakse välja probleemid, lahendused ja nõuded. Töö käigus uuritakse lähemalt menetlusi ja kahe riigi vahelist andmevahetust kasutades ontoloogiaid. Magistritöös loodud prototüüp on kirjutatud Java programmeerimiskeeles.

Semantiline veeb on nüüdisajal uus uurimisvaldkond e-riigi rakenduste ja tsiviilõiguslike rakenduste arendamisel [2]. Semantikat kui keeleteaduse koosvõimet on uuritud juba Aristotelese ajast. IT-revolutsiooniga on see teema tulnud päevakorda erinevates valdkondades, milleks on tehisintellektid, informatsiooniteadus ja teadmiste haldamine [3]. Nendes kõikides valdkondades on alamvaldkonnad, mille alla kuulub ka käesolev magistritöö. Praegusel ajal on loodud erinevaid arenduskeeli ontoloogia arendamiseks veebirakenduste jaoks, et hõlbustada veebiressursside teadmiste esitamist [4]. Ontoloogiaid kasutatakse teadmiste ja semantiliste andmete esitamiseks [15]. Seda võib nimetada konkreetse domeeni ametlikuks esitamiseks [16], mis koosneb kahest üldisemast osast: kinnituselemendist ja terminoloogilisest komponendist [17]. Enamik ontoloogia tehnika uuringuid on seotud tugevate, arukate, teadmistepõhiste süsteemide arendamisega erinevates valdkondades [18]. On-

toloogiaid on üha enam hakatud kasutama ka elektroonilise informatsiooni struktureerimiseks [19] ja selle taustal on valitsused üle maailma hakanud välja arendama erinevaid e-riigi [20] teadmiste halduse projekte, mille jaoks on loodud erinevaid ontoloogiaid andmete edastamiseks läbi teadmiste esitamise [21].

Informatsiooniteaduses on San Segundo and Beltrán defineerinud ontoloogia kui "Instrumendid või struktuurid koos tehnoloogilise tõlkega, mida saab kasutada teadmiste esitamiseks veebikataloogides, andmebaasides, sõnastikes ja muudes sõnavara kontrollivates seadmetes" [22]. Põhieesmärk ontoloogial on täiustada informatsiooni esindamist ja kätte saamist [3]. Ontoloogia ise kirjeldab domeeni teadmisi läbi nelja põhi osa [3]:

- klasside alamklasside
- rollide või omaduste
- aspektide (rollide piirangud)
- klassi objektide

Erinevad ontoloogilised märgistuskeeled:

- DAML+OIL - Loodi kahe meeskonna koostöös. Üheks meeskonnaks oli eurooplaste OIL meeskond ja teiseks oli USA DARPA, kes oli varem loonud DAML keele [23] (*DARPA Agent märgistuskeel*). Uus loodud keel oli tehniliselt palju võimekam.
- OIL - (*Ontology Inference Layer*) - See on esimene ontoloogiat esindav keel, mis baseerub W3C standarditele ja see raamistik on välja arendatud kahe uurimusprojekti *On-To-Knowledge* [24] ja *IBROW* tulemusel [25].
- OWL (*Web Ontology Language*) publitseeriti esmakordselt aastal 2002, baseerub XML tehnoloogial ja suudab ühilduda RDF - iga [4].

Semantilise veebi tehnoloogiad genereerivad uusi võimalusi informatsiooni koosvõimeks. Selle abil saab luua automatiseeritud arusaamisi veebiressurssidest ning

selleks on loodud ka hulgaliselt pilootprojekte ja riiklikke projekte e-riigi arendamiseks [3]. Üks kord küsimise printsiip on välja töötatud Euroopa Liidu arenduskavasse, [1] et muuta riikidevahelised toimingud lihtsamaks ärilistel, kaubanduslikel ja riigiteenuste kasutamise eesmärkidel. Tehtud pilootprojekt on e-teenus erinevate stsenaariumite jaoks, nagu näiteks piiriülene äriajamine ja osalemine riigihankemenetlustel, kasutades selleks üks kord küsimise printsiipi. See põhineb eeldusel, et erinevate riikide administratsioonid võimaldavad e-teenuseid, mis on suunatud erinevate riikide ettevõtjatele [26].

Teise pilootprojektina oli mõeldud e-teenus, mis käsitleb lihtsustamise vajadust laeva- ja meeskonna tunnistuste valdkonnas, mis on hetkel paberformaadis ja asub rahvusvahelise mereõiguste administratsioonis. Pilootprojektis kasutati andmetele ligi pääsemiseks e-teenust ning seal saadi kõik vajalikud toimingud tunnistuse saamiseks ära teha [27].

Kõige suurem demo ühekordselt andmete küsimiseks kahe riigi näitel on Elonia ja Freedonia vahel tehtud pilootprojekt. Mõlemad projektis osalenud riigid suutsid olla nii andmete saatja kui ka vastuvõtja rollis, mille tulemusel sai selle lugeda õnnestunuks [1].

Ka Araabias on loodud riiklikke ontoloogiaid andmete e-riigi arendamiseks. [28] Enamik nendest on seotud Araabia ontoloogiate käsitlemist NLP tehnikates informatsiooni otsinguks [29], teksti märkimiseks [30], tekstide kokkuvõteteks [31] ja küsimustele vastamiseks [32].

Antud magistritöö uudsus peitub semantilise koostöö arendamises üks kord küsimise printsiibis. Töö tegemisel on õpitud varasemate tööde probleemkohtadest ja nendest hoidutud. Magistritöös on kasutatud ajakohast informatsiooni ja töö tegemisel on vaadatud, et tööriistad, mida on kasutatud praktilise töö loomisel, oleksid ajakohased ning saaksid jätkuvalt uuendusi.

1.1 Probleemipüstitus

Ühtne digivärav ehk SDG on protsess, mis aitab läbi viia e-riigi menetlusi veebis teises liikmesriigis ning tagab projektiga seotud liikmesriigi kodanikele ja ettevõtjatele keskse juurdepääsu kogu vajalikule teabele menetluste kasutamise jaoks. Magistritöös kasutatakse esialgu kolme menetlust, mis on välja toodud SDGR artikkel 5 ja lisa 2 dokumendis [5]. Kõigis projektis osalevatel liikmesriikidel kirjeldatakse andmeid erinevalt ning nende töötlemiseks on vaja luua ühtne semantika, mis tõlgendaks kirjeldusi ühtemoodi. Töö eesmärk on realiseerida semantilist koosvõimet prototüübi näitel järgnevate ülesannete jaoks, kasutades TOOP-i [1] nõudeid:

1. Avaliku sektori asutuse õppetoetuse taotlemine,
2. Sotsiaaltoetuse taotlemine,
3. Pensioni taotlemine riiklikest skeemidest [33]

ning selle käigus analüüsides leida sobivad semantilise koosvõime võimalused ja ka probleemid selle juures. Prototüüp näitab, kuidas töötab semantiline koosvõime andmete ühekordses küsimises kahe riigi vahel, Moraania (Holland) ja Nagooria (Eesti) näitel. Magistritöö on sõltumatu riikidest, viide riikidele on ainult illustreeriv. Uurimisküsimused, millele leiab vastuse käesolevast magistritööst:

1. Kuidas toimib semantiline koosvõime andmete ühekordse küsimise keskkonnas, valitud ülesannete näidetel, valitud kahe riigi vahel?
2. Millised on semantilise koosvõime võimalused ja probleemid?
3. Kuidas andmete ühekordse küsimise keskkonnas semantilist koostöövõimet paremaks muuta?

1.2 Metoodika

Magistritöös kasutatakse disainiteaduse metoodikat [34] probleemide lahendamisel ja töö struktuuri loomisel. Antud töö teemal on puudulikud uuringud, mis näitak-

sid, kuidas toimib semantiline koosvõime andmete ühekordse küsimise keskkonnas kahe riigi vahel, milles on arendatud ka mitmekeelne ontoloogia ning selle vastavusse seadmine. Semantika koosvõime uurimuses uuritakse erinevaid semantika arendamise võimalusi, keskkondasid, töötamise põhimõtteid ja loomist ning kasutamise võimalusi. Millised sobilikud raamistikud aitavad kaasa antud magistritöö praktilise väljundi loomisele ning kuidas ja kas need töötavad antud kontekstis. Milliseid osasid on vaja luua, et praktiliseks pooleks vajaminev semantiline koosvõime töötaks ja kuidas seda rakendada üks kord küsimise printsiibi abil.

Eelneva uuringu käigus valitakse välja parimad lahendused, mida kasutades disainitakse semantiline vahend kahe riigi vaheliseks andmete vahetuseks, mis realiseeritakse prototüübiks. Eelnevalt uuritakse, kuidas luua ontoloogiat ning seda arenduse käigus mitmekeelseks ontoloogiaks muuta, mida saab kasutada andmete töötlemisel e-toimingute tegemiseks. Valitakse välja kolm rahvusvahelist e-toimingut, mida Euroopa kodanik võib teha teises riigis enda koduriigis olevate andmetega. Arendusel kasutatakse agiilse tarkvaraarenduse metoodikat.

Lõputöö väljundiks on eelneva analüüsi põhjal loodud prototüüp, milles toimivad valitud e-toimingud, andes väljundiks valitud riigi toiminguks vajaminevate andmete põhjal toimingule vastava väljundfaili. Loodud semantilist koosvõimet testitakse kolme väljavalitud e-toiminguga ning vaadatakse, kas väljundiks tulnud fail sisaldab ontoloogias esinenud andmeid või mitte ja kas e-toiminguid valideeritakse vastavalt e-toiminguga seotud nõuetele. E-toimingute nõuded on välja toodud teises peatükis. Selle põhjal järeldatakse, milline oli lõputöö projekti lõplik tulemus.

Valideerimiseks analüüsitakse mitmekeelset ontoloogiat [35] kasutatava vahendi nõuetele vastavust ning selle edasi arendamise ja paremaks muutmise võimalusi. Vaadeldakse, kas genereeritud väljundfailis olevad andmed vastavad toiminguks vajaminevatele andmetele riigis.

Teiseks metoodikaks, mille järgi saaks antud magistritöö probleemipüstitust lahendada, oleks tegevusuuringu sümbioos disainiteadusega [36]. Seda metoodikat saaks kasutada, kui oleks ka lõppkasutaja, kelle peal seda kohe rakendada saaks, ning kellele oma uurimustulemusi edasi anda. Selle metoodika tugevusteks võrreldes

magistritöös kasutusel oleva metoodikaga on kohene lõppkasutaja peal rakendamine ja vaatlemine ning hiljem vaatluse põhjal uute versioonide arendus. Magistritöös valitud metoodika nõrkuseks on, et prototüüpi ei rakendata reaalses olukorras.

Peatükk 2

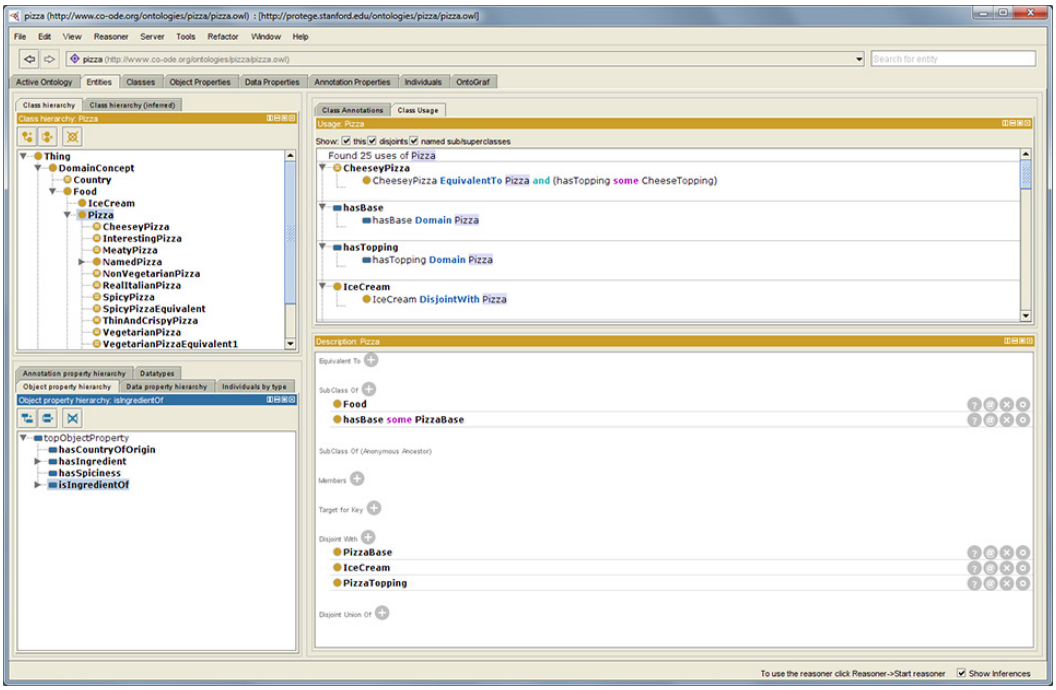
Ülevaade

Ülevaate peatükk tutvustab erinevaid ontoloogiate ehitamise programme ning annab ülevaate, mis on ontoloogia ja kuidas ontoloogiaid vastavusse saab viia.

2.1 Ülevaade ontoloogia ehitamise programmidest

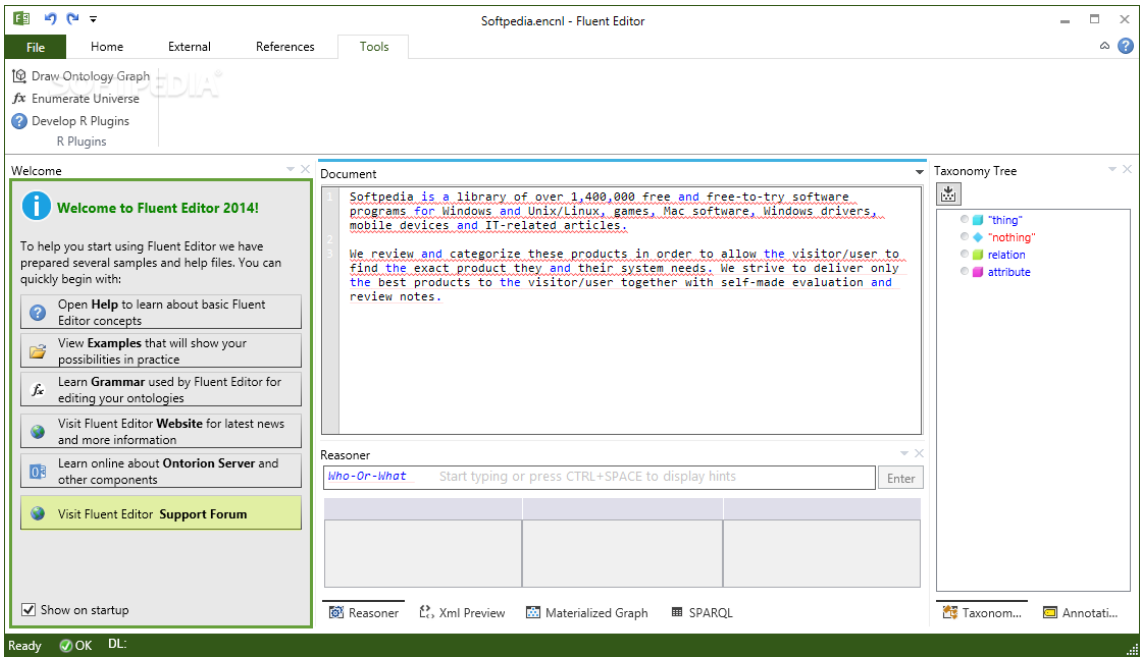
Siin alapeatükis antakse lühike ülevaade neljast programmist, millega saab luua ontoloogiaid. Programmid on valitud W3 veebilehe ontoloogia tööriistade listist [37]. Valiku tegemisel toetuti veebis soovitatud programmidele. Kõige esimese asjana töö käigus tuleb välja valida sobiv ontoloogia loomise tööriist, mis sobiks prototüübi ontoloogiate loomiseks. Ontoloogia loomiseks kasutatakse Protégé programmi. Protégé programm valiti, sest sellel on väga suur kasutajate tugi.

Protégé on tasuta, avatud lähtekoodiga ontoloogia redaktor ja teadmistel põhinev raamistik. Platvorm pakub graafilist kasutajaliidest ontoloogiate defineerimiseks ja toetab pistikprogramme, mis teeb selle paindlikuks rakenduste arendamisel. Rakendus on kirjutatud Java programmeerimiskeeles. Kaks põhiviisi ontoloogiate modelleerimiseks on Protégé-Frames redaktor ja Protégé-OWL redaktor. See on arendatud Stanfordi ülikoolis ja on tugevalt toetatud suures kogukonnas [4]. Joonisel 2.1 on kuvatõmmis programmist Protégé.



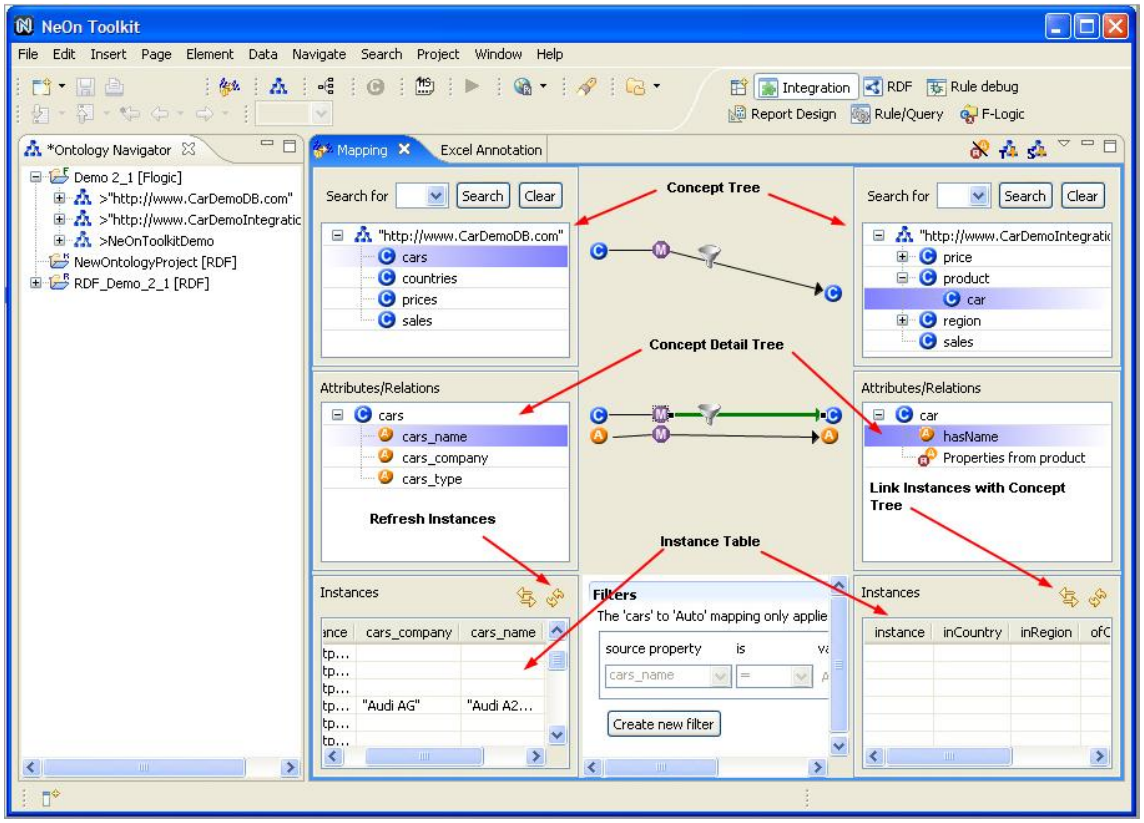
Joonis 2.1: Kuvatõmmis programmist Protégé [6]

Fluent Editor on tööriist millega luuakse, muudetakse, manipuleeritakse ja tehakse päringuid keerulistele ontoloogiatele, mis on kirjutatud OWL-, RDF- või SWRLis. Seda võib kasutada kõikide W3C semantilise veebi standarditega, sellel on intuitiivne kasutajaliides, mis kasutab OCNLi ning see on kasutajasõbralik alternatiiv XML ontoloogia keelele nagu OWL või RDF, aga on samaaegselt ühilduv OWL2ga , RDFi ja SWRLiga. OCNLi saab ka kasutada pärigu keelena, mis on ühilduv SPARQLiga. Tööriist aitab hallata keerulisi ontoloogiaid järgmiste vahenditega : OWL-DL Reasoner aken, mis küsib ontoloogialt päringuid, SPARQL aken, milles saab SQARQL päringuid jooksutada, XML aken, et vaadata kuidas näeb välja kirjutatud OCNL käsk OWLis, taksonoomia puu vaade ja annotatsioonide vaade. Tööriist pakub tuge läbi pistikprogrammi Protégé programmi jaoks eksportimise ja importimisega. Tööriista kasutamine ei ole maksustatud, kui seda kasutada mitteärilistel eesmärkidel [4]. Joonisel 2.2 on kuvatõmmis programmist Fluent Editor.



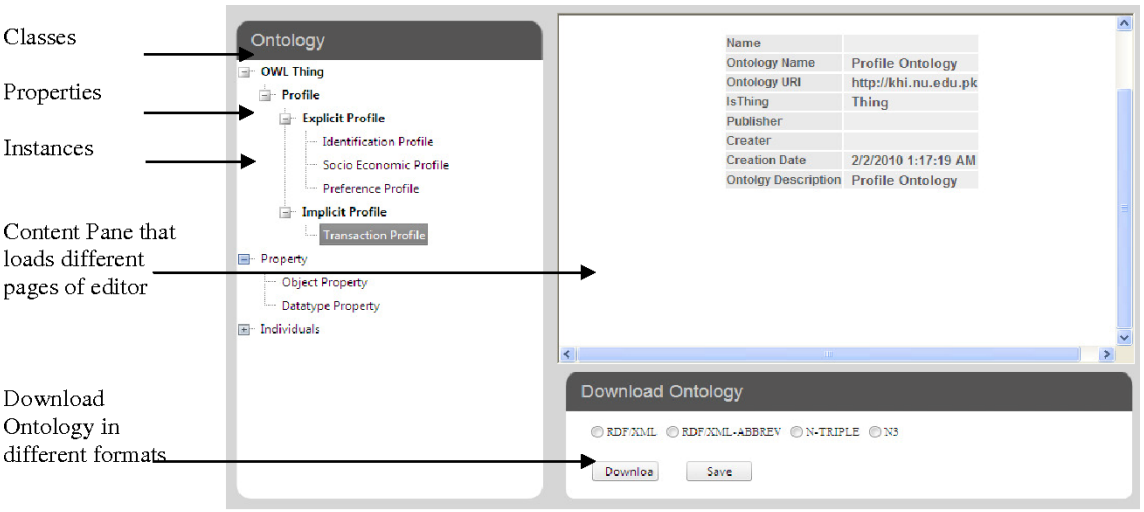
Joonis 2.2: Kuvatõmmis programmist Fluent Editor [11]

Neon-Toolkit on ontoloogia projekteerimise keskkond, mis algselt arendati NeOn projekti osana. See on ehitatud moodularhitektuuril ning on mõeldud semantiliste rakenduste loomiseks. See on avatud lähtekoodiga ja põhineb Eclipse platvormil. NeOn Toolkit on toetatud ulatuslikult pistikprogrammidega, mis muudavad ontoloogia projekteerimise lihtsamaks. Keskkond toetab ontoloogiate loomist F-Logic ja OWL keeles. Kõige uuem versioon on 2.5.2 [12]. Selle keskkonna sihtrühmaks on semantilise veebi ja teadmiste haldamiste uurijad ning professionaalsed kasutajad kaubandusliku taustaga. Selle projekteerimise protsessi aluseks on elutsükli aspektid, mida pakutakse läbi pistikprogrammide. Erinevate pistikprogrammide tugi muudab selle keskkonna väga paindlikuks ja lubab igal arendajal puuduva osa, mida läheb vaja ontoloogia loomise juurde, luua koheselt, ja kasutada seda keskkonnas. Kasutades avatud ja koostöö valmiduse lähenemist funktsionaalsuste arendamiseks on tegu väga hea keskkonnaga, mis baseerub laiendataval arhitektuuril [38]. Joonisel 2.3 on kuvatõmmis programmist Neon-Toolkit.



Joonis 2.3: Kuvatõmmis programmist Neon-Toolkit [12]

Tool for Ontology Development and Editing (TODE) on .NET-raamistikul baseeruv ontoloogia loomise ja töötlemise tööriist. Selle programmi disain baseerub MVC arhitektuuril, mis on sobiv edasisteks laiendusteks teistele klientidele, et hoiduda selle juures suuremate äriloogika muudatustest. Tode pakub väga kasutussõbralikku liidest domeeni teadmiste modelleerimiseks. Loodud ontoloogiat saab eksportida W3C standardkeeltesse. Veebipõhise kasutajaliidese loomise juures on jälgitud kõiki W3C HCI nõudeid, et luua kasutajasõbralik veebileht [39]. Joonisel 2.4 on kuvatõmmis programmist TODE.

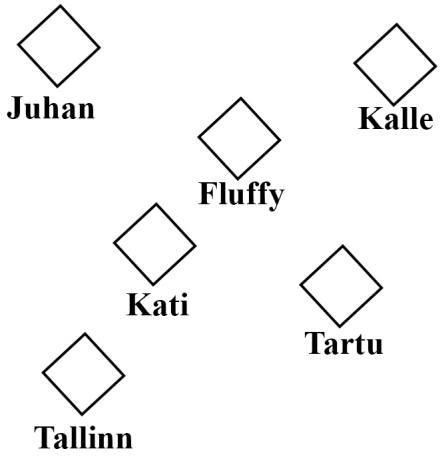


Joonis 2.4: Kuvatõmmis Tode programmist [13]

2.2 Ülevaade ontoloogiast

Siin alapeatükis antakse ülevaade ontoloogiast. Ontoloogiat kasutatakse teadmiste esitamiseks mingi domeeni kohta. See kirjeldab domeeni mõisteid ja nende mõistete vahelist suhet. Erinevad ontoloogiakeeled pakuvad erinevaid lahendeid. Üks ontoloogia ehitamise keeltest on OWL, mis on loodud W3C poolt. OWL ontoloogiatel on sarnased osad nagu Protégé raamil baseeruvatel ontoloogiatel, aga terminoloogia on natukene erinev. OWL ontoloogia koosneb klassi eksemplaridest, omadustest ja klassidest [40]. Klassi eksemplarid on esindatud objektid domeenil, millest me oleme huvitatud. Näide on toodud joonisel 2.5, kus iga teemant näitab ühte eksemplari [41].

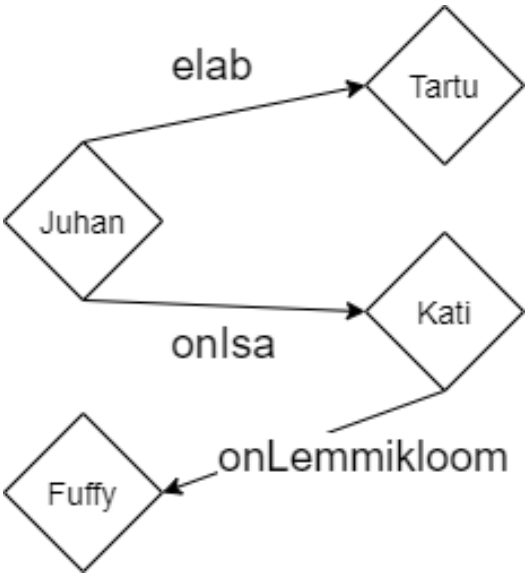
Eksemplar ehk isend, representeerib objekti domeenis. Eksemplar representeerib ontoloogia kõige alumist taset [41]. See on asi, mida ontoloogia kirjeldab või potentsiaalselt võib kirjeldada [42]. Need on formaalne ontoloogia osa ja on üks võimalus kirjeldada huvide üksuseid. Ontoloogia võib ka olla ilma eksemplarideta [43]. Eksemplarideks joonisel 2.5 on Juhan, Fluffy, Kalle, Kati, Tartu ja Tallinn.



Joonis 2.5: Eksemplaride näide

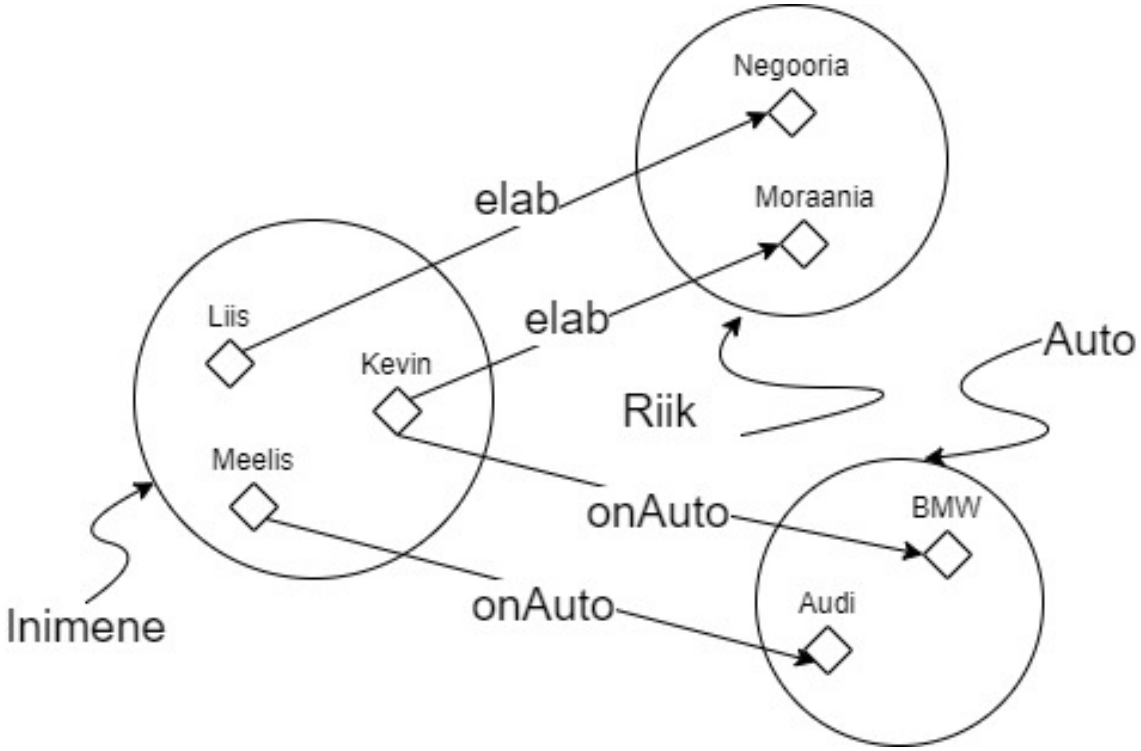
Omadused on seos kahe asja vahel eksemplaris ehk siis lüli, mis ühendab kahte eksemplari omavahel. Omaduste näide on joonisel 2.6, milles on näha, et omadus önLaps"ühendab eksemplari Juhan eksemplariga Kati ja omadus "elab"ühendab Juhanit eksemplariga Tartu [41]. OWLis on kolme tüüpi omadusi:

- Objekti omadused
- Andmetüübi omadused
- Annotatsiooni omadused



Joonis 2.6: Omaduse näide

Klassid on enamike ontoloogiate keskmes. Klassid on konkreetsed representatsioonid mõistetest. Klassid kirjeldavad domeeni mõisteid. Joonisel 2.7 on välja toodud kolm klassi. Inimese, auto ja riigi klassid. Inimese klass sisaldab inimesi. Riigi klassis on välja toodud riigid, kus inimesed elavad. Auto klassis on välja toodud automargid. Neid kõiki klasse seovad omadused. Näiteks eksemplar Liis elab Negoorias ning Meelis omab autot Audi. Klassid on OWLis tõlgendatud eksemplaride hulkadeks. Neid võib kirjeldada kasutades formaalselt kirjeldust, mis näitab täpseid nõudeid klassi liikmeks olemiseks [42]. Klassihierarhiat kutsutakse ka taksonoomiaks. Ontoloogia koos individuaalse klassiisendite komplektiga moodustab teadmiste baasi. Ontoloogia ehitamisel peame loodavad klassid üksteisest eraldama, et ühe klassi eksemplar ei saaks olla teise klassi liige grupis [41].



Joonis 2.7: Klasside näide

2.3 Ontoloogiate vastavusse seadmine

Järgnevas alampeatükis seletatakse erinevaid ontoloogiate vastavusse viimise viise. Kõige keerulisem osa antud magistritöö juures on andmetest ühtselt arusaamine kui ontoloogiad on erinevad keele ning struktuuri poolest. Järgnevalt vaadeldakse, kuidas ja mida on vaja selleks teha.

Võimalused ontoloogiate vastavusse viimiseks on erinevad. Laialdaselt on kasutusel masinõppe lähenemine ontoloogiate vastavusse viimiseks. Masinõppe lähenemisel kasutatakse erinevaid meetodeid ontoloogia vastavusse viimiseks. Masinõppel baseeruval ontoloogia vastavusse viimise süsteemi aluseks on klassifikaator. Hetkel on loodud mitmeid klassifikaatoreid ja neid rakendatud masinõppel baseeruvate otsuste tegemise probleemidel. Ontoloogia joondamine on selles mõeldud kui tõenäosustiheduse funktsioon modelleerimisel. Sellisel viisil on kasutatud parameetrilist lähenemist, milles tehakse selgesõnalised eeldused aluseks oleva mudel põhjal. Masinõppe lähenemine on jagatud kahte faasi. Esimeses faasis viiakse läbi treeningud või õpitakse tundma andmeid. Teises faasis kasutatakse õpitut uute ontoloogiate sobitamiseks. Selleks kasutatakse otsustuspuid, närvivõrke ja tugivektoreid. Meetodi tõhusus, kasutades objektidel põhinevat lähenemist, seisneb selles, et mida rohkem on olemasolevaid objekte ontoloogias, seda paremini see lähenemine töötab. Veel mõned meetodi nimed, mida kasutatakse masinõppe lähenemisel ontoloogia vastavusse viimiseks [44]:

- GLUE
- LSD

Reeglitepõhine lähenemine seisneb selles, et reaalsele aplikatsioonile luuakse sobitusreeglite komplekt. Loodud reegleid saab kasutada konkreetsete objektide sobitamiseks. Skeemi seostamiseks on olemas ainult piiratud kogus semantilist informatsiooni, saadud tulemus pole üldiselt perfektne, seal esineb alati määramatust täpse seostamise vahel objekti tasandil. Seda lähenemist kasutatakse, et arvutamise reegleid oleks alati lihtne realiseerida ja need oleksid lihtsalt arusaadavad kõigile. Erinevad algoritmid reeglitepõhise lähenemise kasutamiseks [45]:

- Prompt
- Chimaera
- Smart
- QOM
- OntoMorph

Semantilise põhjenduse meetodi mõte seisneb algoritmi kasutamises ontoloogia vastavusse viimiseks. Semantilised algoritmid käsitlevad sisendit vastavalt semantilisele tõlgendusele. Intuitsioon on see, kui kaks entiteeti on samad, siis nad jagavad sama tõlgendust. Ontoloogiates on antud tavaliselt struktuur ja sellest tulenevalt ei ole selles elemendi tasandil semantilisi tehnikaid. Semantilised relatsioonid on kogumikes nagu lahknemises, samaväärsuses, tähtsam ja vähemtähtsam. Algoritmid, mida kasutatakse sellise lähenemise puhul [44]:

- CTXMATCH
- S-match

Peatükk 3

Prototüübi loomine semantilisele koosmõjule

3.1 Semantilise vahendi loomise nõuded

Siin alapeatükis tuuakse välja nõuded, mida peab loodav semantiline vahend sisaldama. Vahend peab teostama kolme eelnevalt välja toodud e-toimingut. Järgnevalt tuuakse välja nõuded, mida peab semantiline vahend järgima.

Semantilise vahendi jaoks luuakse kolm ontoloogiat. Igal riigil, keda kasutatakse realiseerimises, peab olema oma ontoloogia. Riik hoiab kodanike andmeid enda ontoloogias. Vahendi jaoks luuakse ühine ontoloogia, mida hiljem kasutatakse ontoloogiate vastavusse seadmisel. Kaks ontoloogiat on riikide ontoloogiad ja üks on ühine ontoloogia. Vahend viib vastavusse korraga ühe riigi ontoloogia. Ontoloogiad peavad sisaldama menetluse jaoks vaja minevat infot. Päringu tüüpe ei arvestata ontoloogiate vastavusse seadmisel. Ontoloogiad ei viida vastavusse igal päringul. Kõigis vastavusse viidud ontoloogiates kasutatakse päringuid samas keeles. Ontoloogiad on vaja vastavusse viia, et semantiline vahend saaks teha neile päringuid teha sama struktuuri järgi. Kõik vastavuse viidud ontoloogiad, millele tehakse päringuid peavad olema samas keeles ja ühesuguse struktuuriga. Vahend peab suutma teha päringuid ontoloogiatele, mida hoitakse serveris. Taotluste päringute tegemine seman-

tika vahendisse tahab saada kaasa JSON objekti. Objekt sisaldab taotlust sooritava kodaniku andmeid, kes soovib taotlust sooritada läbi semantika vahendi. Semantilises vahendis ei teostata veatöötlust. Semantiline vahend peab vastama päringutele vastustega. Taotluse tegija saab semantika vahendilt vastuse taotluse tulemusest. Vahend peab toetama tavapensioni, vajaduspõhise õppetoetuse ja sotsiaaltoetuse taotlemist. Taotlusi on vaja valideerida. Iga taotluse kohta on kaks kodaniku, kes saavad taotlust esitada. Kodanikud on pärit erinevatest riikidest. Valideerimiseks luuakse RESTful veebirakendus, mis vastab semantilisele vahendile valideerimise tulemusega. Semantika vahend peab olema RESTful veebirakendus.

Ontoloogiad on vaja viia vastavusse ainult ühe korra, sest riigil on sellisel juhul see olemas. Keskses süsteemis ei hoita andmeid. Iga riik vastutab kohalike andmete hoiustamise ja kättesaadavuse eest ise. Andmetele pääsetakse ligi üle urlide.

3.2 Menetlused

Selles alapeatükis kirjeldatakse kolme automatiseeritud menetlust. Iga menetlus vastutab ühte tüüpi taotluse eest. Vajalike andmete küsimine erinevate ontoloogiate vahel on keerukas. Menetluste läbi viimisel esineb ühiseid probleeme. Erinevate riikide ontoloogiatel esinevad keelelised, struktuurilised ja terminoloogilised erinevused. Need probleemid muudavad keeruliseks andmete vahetuse kahe riigi vahel. Erineda võib ka menetlus riikides. Taotluste tegemiseks ei jaga riigid iseseisvalt kodanike andmeid. Probleeme aitab lahendada loodud semantika vahend.

3.2.1 Õppetoetuse taotlemine

Õppetoetuse taotlemiseks on loodud automatiseeritud menetlus, mis on suunatud vajaduspõhise õppetoetuse jaoks. Järgnevalt tuuakse välja prototüübis realiseeritud näited, olukorra kirjeldus, probleem ja lahendus vajaduspõhise õppetoetuse saamiseks.

Isik Kevin Laanemagi tahab taotleda õppetoetust, õppides Negoorias. Ta ei ole

Negooria kodanik ja ta ei saa sealt õppetoetust taotleda. Kevin Laanemagi on Moraania kodanik ja ta saab sealt riigist õppetoetust taotleda. Ta õpib Negooria ülikoolis nimega TTU. Ta ei ole õppepuhkusel ning õpib nominaalajal. Tal on kokku saadud 65 EAP-d ja ta õpib MsC astmes. Tal on isa ja ema, kes elavad Moraanias. Pangakonto asub tal samuti Moraanias Swedbanki pangas. Tema sissetulek on 435 eurot kuus. Ta on 22-aastane.

Isik Liis Maasikas tahab taotleda õppetoetust, õppides Moraania ülikoolis vahetusüliõpilasena. Ta pole Moraania kodanik. Ta on Negooria kodanik ning saab sealt riigist taotleda õppetoetust. Liis õpib TLU ülikoolis, ta pole õppepuhkusel ja õpib nominaalajal. Tal on käimas hetkel teine semester ning Liis õpib magistris. Ülikoolis on tal hetkel täidetud 30 EAPd. Liisil on Negoorias pangakonto Strumbanki pangas. Tema sissetulek on 500 eurot kuus. Tal on ema ja isa, kes on samuti Negooria kodanikud. Liis on 20 aastane.

Taotleja õpib riigis, kus ta ei ole kodanik. Tal puudub võimalus taotleda kohalikku õppetoetust. Õppetoetuse taotlemiseks peab ta taotluse esitama riigis, kus ta on kodanik. Õppetoetust saab taotleja taotleda riigist, kus ta on kodanik. Taotluse tegemiseks on vaja andmeid õppimise kohta riigist, kus taotleja õpib. Igal riigil on oma nõuded taotluse tegemiseks.

Järgnevalt tuuakse välja nõuded, mis on vaja täita, et saada vajaduspõhist õppetoetust. Esimesena tuuakse välja nõuded Moraania riigis taotluse tegemiseks. Taotluse nõuete tegemisel on eeskujuks võetud Eesti riigi vajaduspõhise õppetoetuse süsteem. Tudeng peab õppima täiskoormusel ja täidab õppekava nõudeid 75 protsendi ulatuses. Esimesel semestril taotledes peab õpilane õppima täiskoormusel. Tudeng peab olema Moraania kodanik. Tudengi perekonnaliikmete sissetulek perekonnaliikme kohta ei tohi ületada 523 eurot kuus. Kui tudengi vanus on kuni 24 eluaastat, siis arvestatakse tema taotlusesse ka tema pereliikmete sissetulekud. Üle 25 aastaste tudengite sissetulekuks arvestatakse ainult tudengi enda sissetulek või kui ta on abielus, siis ka tema kaaslase sissetulek. Tudeng ei tohi olla akadeemilisel õppepuhkusel. Taotlust on võimalik sooritada üks kord õppesemestri jooksul [14]. Vajalikud andmed, mida on vaja taotluse esitamiseks: taotlust esitava isiku andmed, pereliikmete andmed, õppimise andmed.

Näide Moraania õppetoetuse summadest 3.1 [14]:

Üliõpilase keskmine sissetulek pereliikme kohta	kuni 130,75 eurot	130,76 - 261,50 eurot	261,51 - 523 eurot
Õppetoetuse suurus	220 eurot	135 eurot	75 eurot

Tabel 3.1: Õppetoetuse summad [14]

Negooria riigi õppetoetuse taotluse tegemine. Taotluse nõuete tegemiseks on võetud eeskujuks Hollandi riigi õppetoetuse süsteem. Tudeng peab õppima täiskoormusel ja läbima vähemalt esimese semestri. Kõigi pereliikmete sissetulek ei tohi ületada 13000 eurot. Pereliikmeteks loetakse kõiki pereliikmeid. Tudengi vanus ei tohi olla üle 30 eluaasta. Tudeng peab olema Negooria rahvusest, või ta vanemad peavad olema Negooria riigi kodanikud. Toetussumma on 173.58 eurot [46]. Vajalikud andmed, mida on vaja taotluse esitamiseks. Isiku andmed, kes taotlust esitab, pereliikmete andmed, õppimise andmed. Õppetoetuse taotleja kasutab loodud prototüüpi, et esitada taotlus vajaduspõhise õppetoetuse saamiseks. Taotluse esitamisel annab taotleja kaasa andmed, mida prototüüp vajab menetluse läbiviimiseks. Prototüüp vastab taotlejale vajaduspõhise õppetoetuse saamise kohta.

Riikide vahel erinevad taotlused ja taotluse jaoks vaja minevad andmed. Näiteks Negooria riigis ei arvestata vanust taotluse esitamisel. Moraanias antakse toetus, kui sissetulek ühe pereliikme kohta ei ületa 523 eurot. Riikides kasutatakse erinevaid keeli ja õppimise andmete küsimiseks Negooria riigist oleks vaja teada, kuidas selles keeles on andmete nimed, mida menetlus vajab. Ontoloogiate erinevus struktuurselt raskendab õppimise kohta käivate andmete küsimise päringu loomist. Õppetoetuse taotlemine käib mõlemas riigis erineva menetlusega ja nõuetega.

Semantiline vahend annab sellele probleemile lahenduse. Selle kaudu saab ontoloogiad ühise struktuuri, keele ja terminoloogia peale. Semantilise vahendi sees on olemas päringud, mis aitavad saada kätte Negooria või Moraania riigist õppimise andmed. Semantiline vahend oskab ise kokku panna andmed, mida menetlus vajab ning valideerib õppetoetuse taotluse vastavalt valitud riigi nõuetele.

3.2.2 Pensioni taotlemine

Pensioni taotlemiseks on loodud automatiseeritud menetlus, mis on suunatud riikliku pensioni taotlemise jaoks. Järgnevalt tuuakse välja prototüübis realiseeritud näited, olukorra kirjeldus, probleem, ja lahendus pensioni saamiseks.

Isik Peeter Leevin on 64-aastane ja ta hakkab pensionile jääma. Ta kolis Negooriasse. Olles Moraania kodanik, taotleb ta pensionit Moraaniast. Ta on suurema aja elust töötanud Moraanias. Ta on naisest lahus, aga tal on üks laps, keda nad naisega koos kasvatasid 35 aastat. Lapse nimi on Kevin Laanemagi. Tema tööd on alati olnud seotud projektidega. Tema töökohtadeks on olnud CBV, CGH ja paljud teised. Viimane ametlik töökoht jäi tal aastasse 2018. Tema pangakonto asub Moraania pangas SEB.

Isik on Meelis Tamm. Ta on 67-aastane ning sündis aastal 1957. Ta on tunnustatud doktor. Ta on terve oma elu töötanud haiglates arstina. Esimest korda läks ta haiglasse tööle aastal 1976. Ta on töötanud Negoorias, aga nüüd kolis elama Moraaniasse. Olles Negooria kodanik taotleb ta AOW baaspensionit. Ta elab üksinda ja ta ei võtnud oma laste kasvatamisest osa. Ta käis sõjaväes olles 18-aastane ning peale seda läks õppima arstiteadust. Tema pangakonto asub Moraania pangas NIBC Bankis.

Taotleja elab riigis, kus ta ei ole kodanik. Tal puudub võimalus taotleda riiklikku pensioni sellest riigist. Ta on saanud alles pensioniealiseks ning peab tegema pensioni saamiseks taotluse. Pensionit saab ta taotleda riigist, kus ta on kodanik. Taotleja pangakonto on riigis, kus ta hetkel elab. Igal riigil on oma nõuded taotluse tegemiseks.

Pensioni taotlemise nõuded Moraania riigis. Pensioni taotleja peab olema Moraania riigi kodanik. Tema vanus ei tohi olla vähem kui 64 eluaastat. Taotlejal peab olema vähemalt 15 aastat pensionistaaži. Taotleja võib esitada kõikides riikides töötamise andmeid. Taotlejal peab olema pangakonto. Pensionistaaži sisse arvestatakse kõik esitatud ja tõendatud töötamise andmed. Andmed, mida on vaja pensioni taotlemiseks: Isiku andmed, panga andmed, tööstaaži andmed. Pensioni suuruse ar-

vutamiseks on valem 215,5148 (baasosa) + x (staaž) x 7,104. Kõige esimene number on baasosa ehk riiklikult kinnitatud summa. Sellele summale lisatakse juurde staaž * koefitsient. Selle tulemusel saadakse pensionisumma. Menetluse kirjelduses võeti eeskuju Eesti pensionisüsteemist [47]. Pensioni taotlemise nõuded Negooria riigis. Pensioni taotleja peab olema Negooria riigi kodanik. Tema vanus ei tohi olla vähem kui 67 eluaastat. Taotlejal peab olema maksimaalselt 50 aastat töökogemust taotletavas riigis, sinna alla läheb ka kooliaeg ja sõjaväe kohustus. Andmed mida on vaja pensioni taotlemiseks: isikuandmed, pangaandmed, tööstaaži andmed. Menetluse kirjelduses võeti eeskuju Hollandi pensionisüsteemist [48]. Maksimaalne pensionisumma on 1162 eurot. Valem pensioni arvutamiseks. Kui sul on 50 aastat tööstaaži siis on sinu summa 1162 eurot. Iga aasta eest, mis sul on puudu lahutatakse summast 2 protsenti. Lõplik summa arvutatakse tööstaaž * 2 protsenti. Pensioni taotleja kasutab loodud prototüüpi, et esitada taotlus riikliku pensioni saamiseks. Taotluse esitamisel annab taotleja kaasa andmed, mida prototüüp vajab menetluse läbiviimiseks. Prototüüp vastab taotlejale pensioni saamise kohta.

Pensioni saamiseks erineb menetlus mõlemas riigis. Nõuded pensioni taotluse esitamiseks erinevad mõlemas riigis. Moraanias peab vanaduspensioni saaja olema vähemalt 64-aastane. Negoorias peab vanaduspensioni saaja olema vähemalt 67 aastane.Moraania riigis võetakse arvesse kõiki tõendatud tööaastaid pensioni summa arvutamisel, aga Negoorias arvestatakse ainult neid tööaastaid, mille jooksul oled töötanud seal riigis. Riigiti arvutatakse pensionit erinevalt.

Semantiline vahend annab sellistele probleemidele lahenduse. Selle sees on olemas päringud, mis aitavad saada kätte Negooria või Moraania riigist pensioni taotluseks vajaminevad andmed. Semantiline vahend oskab ise kokku panna andmed, mida menetlus vajab ning valideerib pensioni taotluse vastavalt valitud riigi nõuetele. Pensioni summad arvutatakse vastavalt riiklikele nõuetele.

3.2.3 Sotsiaaltoetuse taotlemine

Sotsiaaltoetuse taotlemiseks on loodud automatiseeritud menetlus, mis on suunatud puudega tööealise inimese sotsiaaltoetuse taotlemise jaoks. Järgnevalt tuuakse välja

prototüübis realiseeritud näited, olukorra kirjeldus, probleem ja lahendus sotsiaaltoetuse saamiseks.

Isik Maali Vutt on Moraania kodanik. Talle meeldib ekstreemsport ja mägironimine. Ta töötas mägironimise instruktorina. Ta on 45-aastane. Ta sai hea pakkumise 4 aastat tagasi mägironimise instruktorina Negooriasse. Pool aastat tagasi juhtus temaga õnnetus, ning ta on nüüd nägemispuudega inimene, ehk ta nägemine on alanenud sel määral, et see piirab tema töötegemist. Maalile tehti puuderaskusastme määramine Negoorias doktor Kalmer Rebase poolt. Nüüd, kui ta on Negoorias, tahab ta taotleda sotsiaaltoetust puudega inimesele Moraania kodanikuna. Tema puudeaste määrati keskmiseks ehk töö jätkamine pole võimalik. Tema pangakonto asub Negoorias ning ta jätkab elu seal riigis.

Isik Mari Kask on Negooria kodanik. Ta on õmbleja, disainer ja modell. Ta on 66-aastane. Olles disainer reisib ta palju ning külastab maailma eri paiku. Ühel moeetendusel osales ta modellina Moraanias. Olles ühel moeetendusel modelliks, kukkus ta lavalt alla ning sai kuulmiskahjustuse. Pärast seda, kui ta avastas, et ei kuule enam kõrgeid helisid, lasi ta endale teha puuderaskusastme analüüsi ning talle määrati konduktiivne kuulmislangus. Jäädes nüüd pikemaks ajaks Moraaniasse, taotleb ta sotsiaaltoetust puudega inimesele Negooria riigist. Tal on pangakonto Negooria riigis NIBC BANK pangas.

Taotleja elab riigis, kus ta ei ole kodanik. Tal puudub võimalus taotleda sotsiaaltoetust puudega tööealisele kodanikule selles riigis. Taotleja on tööealine kodanik. Taotleja on teostanud puudeastmetuvastuse teises riigis. Igal riigil on oma nõuded taotluse tegemiseks. Sotsiaaltoetuse taotlus on suunatud puudega tööealistele. Nõuded Moraania riigi sotsiaaltoetuse taotluse tegemiseks. Taotleja peab olema Moraania riigi kodanik ja peab olema tööealine. Taotlejal peab olema registreeritud puue. Taotlus tuleb esitada uuesti, kui puude raskuste on korduval tuvastamisel. Taotleja peab olema tööealine. Taotlejal peab olema pangakonto. Vajalikud andmed, mida on vaja taotluse esitamiseks. Isiku andmed, kes taotlust esitab, puude raskuse andmed, panga andmed. Puudega tööealise inimese toetust makstakse iga kuu ja summas 25.57. Menetluse kirjelduses võeti eeskuju Eesti sotsiaaltoetuse süsteemist [49]. Nõuded Negooria riigi sotsiaaltoetuse taotluse tegemiseks. Taotleja peab

olema Negooria riigi kodanik. Taotleja peab olema tööealine. Taotleja peab olema registreeritud puue. Taotlejal peab olema pangakonto. Vajalikud andmed, mida on vaja taotluse esitamiseks. Isiku andmed, kes taotlust esitab, puude raskuse andmed, panga andmed. Puudega tööealise inimese toetust makstakse iga kuu ja summas 168.24 Menetluse kirjelduses võeti eeskuju Hollandi sotsiaaltoetuse süsteemist[50]. Sotsiaaltoetuse taotleja kasutab loodud prototüüpi, et esitada taotlus puudega tööealise inimese sotsiaaltoetuse saamiseks. Taotluse esitamisel annab taotleja kaasa andmed, mida prototüüp vajab menetluse läbiviimiseks. Prototüüp vastab taotlejale sotsiaaltoetuse saamise kohta.

Sotsiaaltoetuse saamiseks erineb menetlus mõlemas riigis. Taotleja on teostanud puude astme tuvastuse teises riigis. Sellega kaasneb struktuuriline erinevus, kuidas andmeid esitatakse ja millise malli järgi pannakse kirja puude astmeid. Astmete tähendused võivad erineda kahe riigi vahel. Andmete märkimise terminid ja struktuur erineb riikide vahel.

Semantiline vahend lahendab need probleemid. Selle kaudu töödeldakse andmed vastavaks iga riigi jaoks ning struktuur ja terminid ühtlustatakse. Andmed ühtlustatakse ja samastatakse kahe riigi vahel. Semantiline vahend oskab ise kokku panna andmed, mida menetlus vajab ning valideerib sotsiaaltoetuse taotluse vastavalt valitud riigi nõuetele.

3.3 Arhitektuur

Siin alapeatükis räägitakse arhitektuurist. Arhitektuur võetakse tükkideks ning seletatakse lahti, mille eest mingi osa vastutab ja kuidas käib suhtlus rakenduste vahel. Kokku on loodud kaks arhitektuuri joonist. Joonis 3.8 kirjeldab ontoloogiate vastavusse seadmise arhitektuuri. Joonis 3.1 kirjeldab üldist arhitektuuri, kuidas menetlused süsteemis läbi viiakse.

Prototüüp on ehitatud, kasutades Spring Booti raamistikku Java programmeerimiskeeles. Kokku kasutatakse kahte erinevat REST rakendust. Üks neist on prototüüp ontoloogiate vastavusse seadmiseks ja menetluste tegemiseks, teist kasutatakse

menetluste valideerimiseks, vastavalt riiklikult seatud välja toodud nõuetele. Valideerimise osa jäljendab erinevaid institutsioone menetluste toiminguteks.

Prototüübi arhitektuur koosneb mitmest osast. Menetluste päringute meetodid peavad kaasa saama kasutaja andmetega JSON failid. Neid andmeid kasutatakse, et tuvastada menetlust tegeva inimese andmed ja leida neile vasted. Joonisel 3.2 on tabeli kujul kirjeldatud näidisobjekt kasutajapoolsete andmete sisestamiseks.

Andmeväli	Väärtus
fName	Peeter
lName	Leevis
citizenship	Moraanian
idCode	3860805815
age	64
schoolCountry	
schoolName	Peeter
bankAccountCountry	est
homeCountry	est
localCountry	ned
workCountry	
wifeName	
childName	Kevin Laanemagi

Tabel 3.2: Kasutaja sisestatavad andmed

Kõik algab kontrolleri osast. Kontroller vastutab lihtsate päringute ning neile vastamise eest, kasutades selleks staatuse koode. Sinna on üles seatud päringud erinevate urlidega. Kontroller sisaldab POST ja GET toiminguid. Järgnevas tabelis 3.3 on välja toodud kontrolleris olevate päringute urlid, meetodid ja millised on menetluste päringute urlid.

Väärtus	Meetod	Menetlus
/	GET	
/socialAssistance	POST	Sotsiaaltoetuse taotlus
/pension	POST	Pensioni taotlus
/scholarship	POST	Õppetoetuse taotlus
/upload	POST	
/ontologycreator	GET	

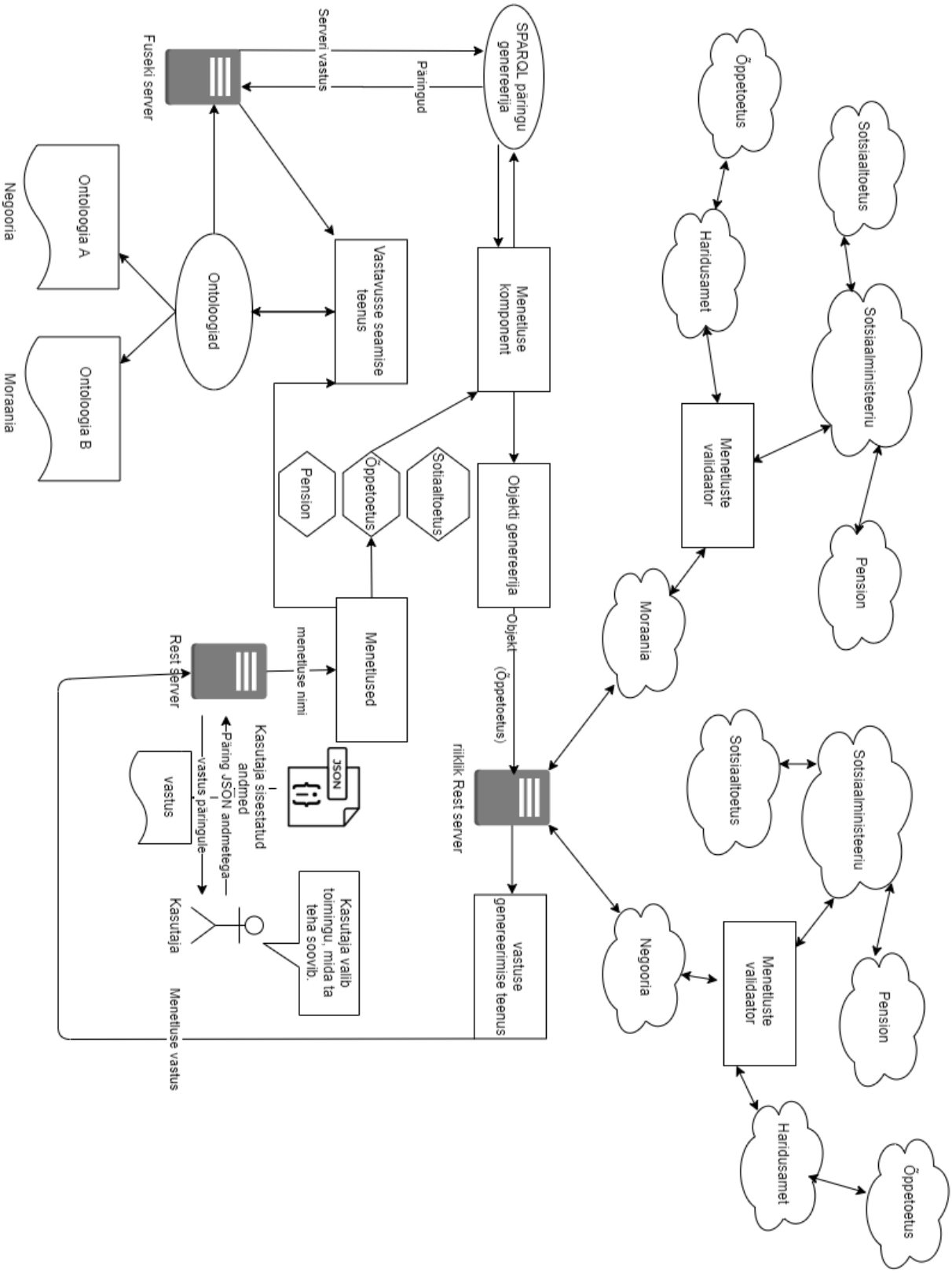
Tabel 3.3: Kontrolleri päringud

Järgmiseks lüliks arhitektuuris on vastavusse viimise teenus. Vastavusse viimise teenus vastutab eelkõige selle eest, et ontoloogiad saaksid vastavusse viidud. Kogu vastavusse viimine ja joondamine käib läbi selle teenuse. Teenus vastutab ka ontoloogiate Fuseki serverisse laadimise eest. Prototüüp kasutab Negooria ja Moraania ontoloogiaid. Ontoloogiad on vaja Fuseki serverisse laadida selleks, et kasutades päringuid, saaks neilt andmeid küsida. Teenuse sees genereeritakse ka SPARQL vastavusse viidud päringud. Fuseki serverisse andmete ülesselaadimiseks kasutatakse Apache Jena raamistikku. Fuseki serveriga suhtlus on selle raamistiku üks osa. Selle raamistiku abil viiakse vastavusse ja joondatakse ontoloogiad. Fuseki server tegeleb ontoloogiate hoiustamisega, töötlemisega ja päringutele vastamisega. Iga riigi ontoloogia hoitakse erineva urli peal, et jäljendada erinevaid riike, kus selle riigi ontoloogiale ligi pääseb. Fuseki serveril on ka kasutajaliidese pool, kus saab teha päringuid ontoloogiatest. Edasi tuleb menetluse komponent. Komponent vastutab selle eest, et kogutakse õiged andmed kokku, mida on vaja menetluse töötlemise jaoks. Igal menetlusel on enda komponent. Iga menetluse jaoks kogutakse erinevaid andmeid ontoloogiatest. Komponent teeb päringud Fuseki serverisse ja saadud vastused lisatakse objekti. Objekt saadetakse üle HTTP kliendi teise menetluste valideerimise eest vastutavasse REST rakendusse. Igal menetluse valideerimisel on erinev päringuaadress.

Semantiline prototüüp aitab seda sama protsessi kiirendada ning paberimajandust vähendada. Selleks, et Kevin saaks taotleda õppetoetust peab ta sisestama oma andmed ning kooli nime, kus ta õpib, kõik ülejäänud teeb tema eest juba keskkond

ise kasutades semantilisi teisendusi andmete kättesaamiseks.

Järgnevalt kirjeldatakse prototüübi tööd koos menetlusega. Kevin esitab oma andmetega päringu õppetoetuse menetluseks. Json fail saadetakse rest teenusele. Edasi kogutakse vajalikud andmed õppetoetuse esitamiseks haridusministeeriumile. Kõigepealt vaadatakse, millise riigi kodanik menetluse esitaja on. Sellele järgneb kodaniku riigi ontoloogiast vaja minevate andmete küsimine. Kui taotleja õpib teise riigi ülikoolis, siis tehakse andmete päringud selle riigi ontoloogiale ning uuritakse, kas selline õpilane õpib seal riigis. Kui õpib, kogutakse andmed taotletava riigi haridusministeeriumi jaoks, et esitada taotlus. Kui prototüüp on kõik andmed kokku kogunud, mida on taotluseks vaja, pöördutakse teise REST rakenduse poole, mis täidab menetluse valideerimise ülesannet. Kaasa saadetakse andmed, mida on vaja menetluse läbiviimiseks. Ette antud validatsiooninõuetele tuginedes valideeritakse andmed, ja antakse vastus, kas taotluse esitaja saab õppetoetust, või ta ei täida nõudeid, ning ta ei saa midagi. Lõpuks saadab menetluse valideerimise rakendus prototüübile andmed tagasi ning taotluse eitaja saab vastuse.



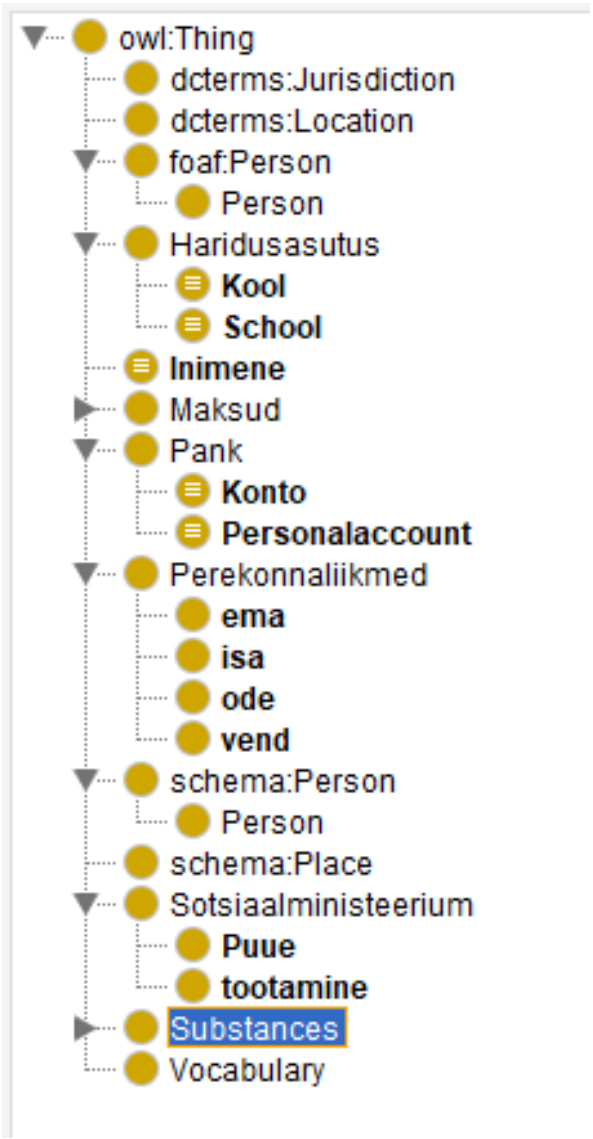
Joonis 3.1: Semantilise vahendi arhitektuur

3.4 Ontoloogiate loomine prototüübi jaoks

Semantika koosvõime uurimise aluseks on ontoloogiad, mida saab töödelda, analüüsida ja vajaminevate andmete kättesaamiseks kasutada. Antud magitritöös tuleb arvestada, et ontoloogiad võivad olla erineva keele, struktuuri ja mõistetavusega nagu reaalne olukord võib välja näha. Järgnevas lõigus kirjeldame kasutusele võetud ontoloogiaid ja nende modifitseerimist ning miks neid vaja läheb.

Ontoloogiate loomiseks kasutati Protégé rakendust. Kasutusele on võetud kolm ontoloogiat. Moraania ja Negooria ontoloogiad erinevad üksteisest struktuuri ja sisu poolest. Moraania ontoloogia on loodud ise, kasutades lihtsat struktuuri ja väljendust. Moraania ontoloogia on eesti keeles. Negooria ontoloogia loomiseks kasutati TOOP pilootprojekti lehel olevat ontoloogiat ning seda täiendati vastavalt magistritöös olevatele nõuetele. Negooria ontoloogia on inglise keeles. Kolmandaks ontoloogiaks loodi keskne ontoloogia, millesse teisaldatakse kõik magistritöös kasutatavad ontoloogiad. Selline valik sai tehtud, et tekitada ontoloogiate vahel mõned erinevused. Ontoloogiate ühtseks tegemise probleeme kirjeldatakse probleemide lõigus. Ontoloogiatele on lisatud eksemplarid, mida hiljem otsitakse ja kasutatakse. Põhilised klassid mudelis on:

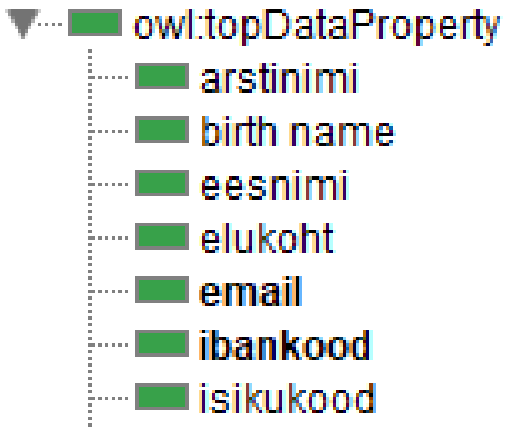
1. Inimene
2. Kool
3. Maksud
4. Pank
5. Perekonnaliikmed
6. Sotsiaalministeerium



Joonis 3.2: Moraania ontoloogia klassistruktuur

Joonisel 3.2 on välja toodud Moraania ontoloogia klassistruktuur, mis sisaldab lihtsaid andmeid.

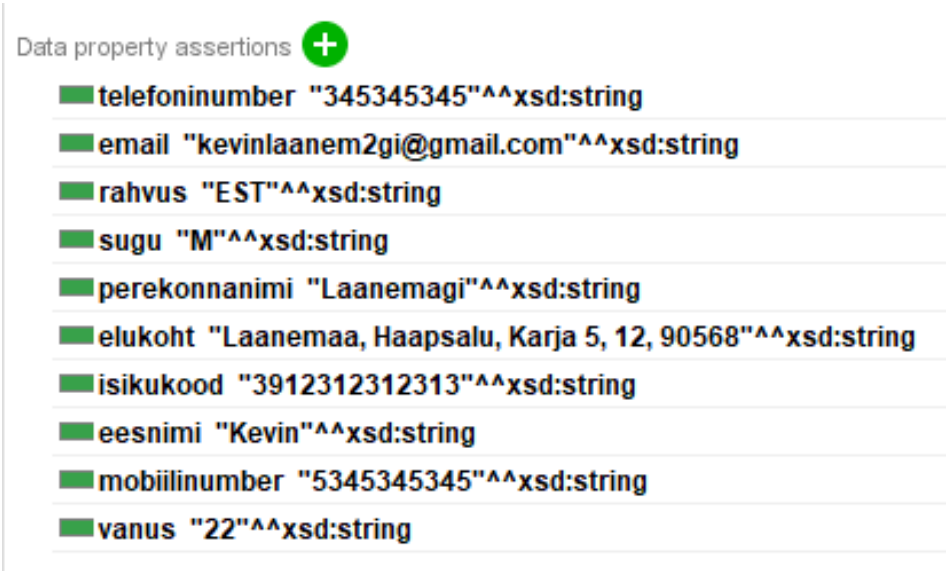
Eksemplaridele on lisatud lisa-atribuudid. Iga klass sisaldab oma eksemplare ning eksemplarides on klassi alla käiv info. Ontoloogiad on RDF/XML formaadis. Joonisel 3.3 on välja toodud Moraania ontoloogias kasutatavad andmete atribuudid.



Joonis 3.3: Eksemplaride omadused

Joonis 3.4: Moraania ontoloogia eksemplarid

Joonisel 3.3 on välja toodud ontoloogias kasutusel olevatest eksemplaride omadustest. Joonisel 3.4 on välja toodud eksemplarid, mida kasutatakse ontoloogias. Joonisel 3.5 on välja toodud inimese eksemplar koos atribuutidega. Kõik näited pärinevad Moraania ontoloogiast.



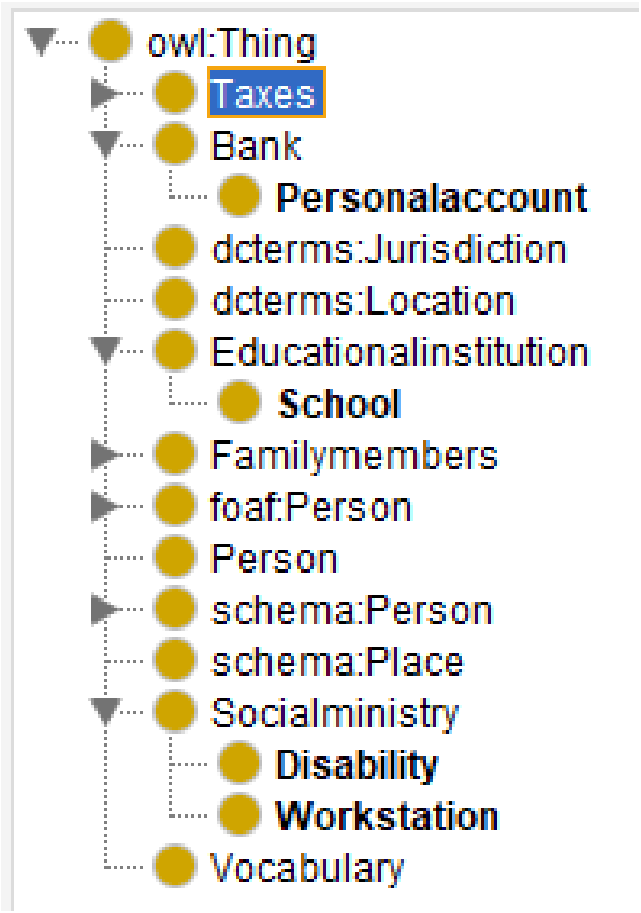
Joonis 3.5: Eksemplar inimene

3.5 Ontoloogiate vastavusse seadmine prototüübis

Ontoloogiad on vaja vastavusse viia, sest prototüüp teeb päringuid ontoloogiatele sama struktuuri järgi. Töös on valitud ühiseks keeleks inglise keel ning kõik ontoloogiad viiakse vastavusse kasutades seda keelt. Vastavusse viimiseks kasutatakse eelnevalt loodud ühist ontoloogia struktuuri. Ontoloogiad viiakse vastavusse, et nendest kätte saada taotluste tegemiseks kodaniku andmed. Ontoloogiate vastavusse viimisel mõeldakse valemile, et kui sõna x ühes keeles on samatähenduslik kui sõna y teises keeles, siis sõna x = y-ga või on väga ligilähedase tähendusega. Kui sõna x = y-ga ja nad jagavad ühist ontoloogia struktuuri, siis x klass sisaldab samu andmeid, mida y klass.

Ontoloogiate ühtseks tegemiseks on vaja need kõigepealt vastavusse viia või luua ühine ontoloogia struktuur, mille peale joondatakse riikide ontoloogiad. Selle tegemisel võivad probleemideks tulla ontoloogiate struktuurilised erinevused, milles kasutatakse samatähenduslikke termineid mõistete lahti seletamiseks või siis keelelised erinevused. Viimase all mõeldakse nii ontoloogia terminoloogia keelt kui ka ontoloogia formaadi keelt. Kui ontoloogia A on kirjutatud inglise keeles ja ontoloogia B eesti keeles, siis võivad eelmainitud probleemid esineda. Töös kirjeldatav meetod kasutab ühise ontoloogia eelnevat loomist ning hiljem sellesse ontoloogiasse erineva-

te andmete lisamist mõlema ontoloogia poolt, vastavalt vajaminevatele andmetele, üleliigseid andmeid sinna juurde ei liideta. Igale riigile ehitatakse uus ontoloogia, et keskne pärimine töötaks kõikide projekti kaasatud riikide vahel. Joonisel 3.6 on toodud pilt ühise ontoloogia klassidest.



Joonis 3.6: Keskse ontoloogia klassistruktuur

Andmete vastavusse viimisel ühtsele ontoloogiale kasutatakse päringuid. Päringute tegemiseks kasutatakse SPARQL päringukeelt. Probleemide vältimiseks on loodud ühtne semantika, mis on koostatud, et kogu kontekst oleks ühtselt arusaadav. Antud prototüübimudelis kasutatakse selleks ontoloogiat, mis on loodud. Järgnevalt on välja toodud näide päringust 3.1.

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX resource: <http://purl.org/vocab/resourcelist/schema#>
PREFIX ns0: <http://www.semanticweb.org/pere/ontologies/2019
/2/untitled-ontology-6#>
prefix ns: <http://creativecommons.org/ns#>
select distinct ?k ?m ?s where {
  ?k rdf:type owl:NamedIndividual.
  ?k rdf:type ?m.
  ?k ns0:eesnimi ?m.
  FILTER (?m = <http://www.semanticweb.org/pere/ontologies/2019/2
  /untitled-ontology-6#Inimene>) .
}
```

Koodinäide 3.1: Päringu tegemise näide

Selle päringu tulemusel saadakse ontoloogiast kätte eesnimi. Eesnimi kuulub inimesele. Päringu lõpus on "FILTER"iga ära määratud, et see peab otsima vastust päringule inimese klassi eksemplaride hulgast. Juhul, kui päring tehakse eestikeelsele ontoloogiale, siis kõigepealt otsitakse ülesse tõlgitud päring, ja seejärel tehakse päring eesnime kohta.

Semantilise integratsiooni ülesanded:

1. Päringute tegemine mitmete ressursside (ontoloogiate) vahel.
2. Andmete teisendamine ühtseks.
3. Ontoloogiast leitud teadmiste esitamine vastavusse viimisega.

Magistritöös kasutatakse struktuuripõhist lähenemist ontoloogiate vastavusse viimiseks. Põhjaks on loodud ühtse ontoloogia struktuur, millel puuduvad objektid. Ühtsel ontoloogial on juba sisse kirjutatud vajaminevad klassid, objektide suhted üksteise suhtes ja omadused. Lõplikule ontoloogiale on lisatud pärast vastavusse viimist eksemplarid, mida hakatakse hiljem kasutama andmete saamiseks. Vastavusse viidud ontoloogiast hakatakse päringute abil andmeid küsima, vastavalt teenustele, mida kasutaja soovib teostada. Antud magistritöö praktilises osas kasutatakse ontoloogiate vastavusse viimiseks ühise ontoloogia loomise meetodit. See põhineb

sisendontoloogia alusstruktuuril. Hüpotees on, et kui kahe mõiste otsene ülikontseptsioon ja alamkontseptsioon on sarnased, siis kaks võrreldavat mõistet on ka võib-olla sarnased [44].

Algselt luuakse ühise ontoloogia struktuur, mis kataks kõiki vajadusi semantiliseks koostööks. Selle loomisel on võetud arvesse erinevaid sõltuvusseoseid klasside objektide vahel. Struktuur on loodud kasutades Protégé tarkvara ning sinna on juurde imporditud klasse, mida kasutatakse TOOP projekti ontoloogiates. Juurde lisatakse veel ühised klasside omadused ja ka andmeomadused, mida hiljem kasutatakse ontoloogiate vastavusse viimisel. Ühise ontoloogia loomiseks kasutatakse Apache Jena Fuseki serverit ja REST teenust. REST teenuses on eraldi teenus, et luua ühine ontoloogia kahe riigi jaoks. Teenusele saadetakse kaasa riigi keele, millist ontoloogiat soovitakse ühiseks muuta. Näites olev url on näidis url Negooria ontoloogia jaoks. Moraania ontoloogia jaoks on muutuja "lang"väärtus "est".
`http://localhost:8080/toop/ontologycreator?lang=NED` vastavusse viimiseks on vaja riiklikud vastavusse viimata ontoloogia ülesse laadida Fuseki serverisse, mis peab eelnevalt töötama arvuti platvormil, millel prototüüpi jooksutatakse. Fuseki server on SPARQL server, mida saab jooksutada operatsiooni süsteemis teenusena, Java veebirakendusena või iseseisva serverina. Sellel on oma kasutajaliides monitoorimiseks ja administreerimiseks. Iseseisev server töötab Jettyi veebiserveri põhjal [8]. Lokaalselt käivitades asub see näites oleval urlil. 3.2 näites on välja toodud koodinäide kuidas laaditakse ülesse ontoloogia.
`http://localhost:3030/manage.html?tab=datasets`

```
private void uploadOntology() throws IOException {
    fusekiService.uploadRDF(
            fileFindComponent.fileFinder("moraaniaOntoloogia.rdf"),
            env.getProperty("service.uri"));
}
```

Koodinäide 3.2: Ontoloogia ülesse laadimine

Ontoloogiate ühiseks saamiseks on vaja vastavusse viia andmed, mida on vaja hiljem ühise ontoloogia loomise jaoks. Selleks luuakse SPARQL päringud. Päringute abil vastavusse viimiseks andmed, mida on vaja kätte saada algselt ontoloo-

gialt. Vastavusse viimiseks küsitakse samanimeliste muutujate alla samad väärtused, kontrollides eelnevalt ontoloogiast, et ontoloogia sisaldaks neid andmeid ja vastava päringuga saaks vastavad andmed kätte. Päringud on loodud Moraania ja Negooria riikide vastavusse seadmata ontoloogiate kohta. Päring on loodud kooli andmete, inimese andmete, töötamise andmete, maksude andmete, isiku puude andmete, pereliikmete andmete ja panga andmete vastavusse viimise jaoks. 3.3 koodinäide on isiku andmete saamiseks mõlema riigi vastavusse seadmata ontoloogitest.

```
public String generatePersonQuery(String language) {
        if (language.equals("en")) {
            return "prefix rdf: <http://www.w3.org/1999/02/22-rdf-
                syntax"
            +"-ns#>
            prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> "
            +"prefix owl: <http://www.w3.org/2002/07/owl#>
            PREFIX foaf: <http://xmlns.com/foaf/0.1/>
            PREFIX ns0: "
            + "<http://www.semanticweb.org/pere/ontologies/2019/"+
            "2/Negooria#> select distinct ?k ?m ?s ?n ?d ?q ?w ?e ?r ?t
                ?l ?c"
                    + " where {\n"
                    + "?k rdf:type owl:NamedIndividual.\n"
                    + " ?k rdf:type ?m.\n"
                    + " ?k ns0:Firstname ?s.\n"
                    + " ?k ns0:Lastname ?d.\n"
                    + " ?k ns0:Age ?n.\n"
                    + " ?k ns0:Genre ?q.\n"
                    + " ?k ns0:Email ?w.\n"
                    + " ?k ns0:Mobilenumber ?e.\n"
                    + " ?k ns0:PersonalIdentificationcode ?r.\n"
                    + " ?k ns0:Nationality ?l.\n"
                    + " ?k ns0:Phonenumber ?t. ?k ns0:Residence ?c.
                        FILTER (?m = <http://www.semanticweb.org/pere/
                        ontologies/2019/2/Negooria#Person>) .}";
        } else {
            return "prefix rdf: <http://www.w3.org/1999/02/22-rdf-
                syntax-ns#>\n"
                    + "prefix owl: <http://www.w3.org/2002/07/owl#>\n"
```

```
                + "PREFIX ns0: <http://www.semanticweb.org/
                    dellalienware/ontologies/2019/4/eesti#> "
                + "select distinct ?k ?m ?s ?n ?d ?q ?w ?e ?r ?t ?l
                    ?c where {\n"
                + " ?k rdf:type owl:NamedIndividual.\n"
                + " ?k rdf:type ?m.\n"
                + " ?k ns0:eenimi"
                + " ?s.\n"
                + " ?k ns0:perekonnanimi"
                + " ?d.\n"
                + " ?k ns0:vanus"
                + " ?n.\n"
                + " ?k ns0:sugu"
                + " ?q.\n"
                + " ?k ns0:email"
                + " ?w.\n"
                + " ?k ns0:mobiilinumber"
                + " ?e.\n"
                + " ?k ns0:Isikukood"
                + " ?r.\n"
                + " ?k ns0:telefoninumber"
                + " ?t. "
                + " ?k ns0:rahvus ?l. "
                + " ?k ns0:elukoht ?c. "
                + "FILTER (?m = <http://www.w3.org/ns/person#
                    Inimene>) .}";
        }
    }
```

Koodinäide 3.3: Isiku andmete päring

Järgmiseks sammuks on päringute loomine eelmainitud andmete kätte saamiseks ontoloogiatest. Koodinäites 3.4 määrame kõigepealt ära, et soovitakse Moraania ontoloogiale päringu suunata kasutades "ee"riigi lühendit ja edasi ontoloogia urli, mille alla ontoloogiad ülesse laeti. Päringus on juba tõlgendatud nimetused. Süsteemile on selgeks tehtud, et millised andmed ontoloogiatest peaksid minema mingi objekti atribuudi väärtuseks. Näiteks muutuja ?l omistab endale väärtuse rahvus andme omaduse eesti keelsest ontoloogiast ja see sõna rahvus on sama tähendusega, mis

sõna nationality inglise keeles ontoloogias. Muutujate nimed on loodud üldistava programmeerimise (generic programming) disaini järgi. Muutujate nimed ?k, ?m, ?s, ?n, ?d, ?q, ?w, ?e, ?r, ?t, ?l, ?c. Iga täheline muutuja omistab andme omaduse väärtust, mille eest ta vastutab. Näiteks muutja ?w vastutab andmeomaduse eest email ja talle omistatakse väärtus, mis tuleb andme omaduselt email. Muutja ?n vastutab vanus tüüpi andme omaduse eest ja talle omistatakse väärtus sellest andme omadusest. Muutuja ?m omistab andme omaduse tüübi ehk siis klassi, mille all andmeid hoitakse. Muutujale ?k omistatakse objekt, mis sisaldab infot, mida päringus küsitakse. Näite puhul omistatakse objekt Inimene. Muutujale ?d omistatakse väärtus perekonnanime andme omadusest. Muutujale ?q omistatakse väärtus sugu andme omadusest. Muutujale ?e omistatakse väärtus mobiilinumber andme omadusest. Muutujale ?r omistatakse väärtus isikukood andme omadusest. Muutujale ?t omistatakse väärtus telefoninumber andme omadusest. Muutujale ?c omistatakse väärtus elukoht andme omadusest. Filtreeringut kasutatakse selleks, et saada päringuga õiget tüüpi objekt, juhul kui peaks olema sarnaste andme omadustega objekte ontoloogias. Sõnade vastavusse viimised päringus on tehtud andes samanimelistele muutjatele väärtused sama tähenduslikust andme omadusest teisest ontoloogiast. Üheks näiteks veel on muutuja ?s, millele omistatakse väärtus andme omadusest "eesnimi"eesti keelsest ontoloogiast ja "Firstname"inglise keelsest ontoloogiast. Andme omaduste vastavusse viimine koodinäite põhjal. Päringute vastavusse viimisel on kasutatud „=“ samaväärsust. Andme omadus Lastname on samaväärne andme omadusega perekonnanimi. Andme omadus Age on samaväärne andme omadusega vanus. Andme omadus Genre on samaväärne andme omadusega sugu. Andme omadus Email on samaväärne andme omadusega email. Andme omadus Mobilenumber on samaväärne andme omadusega mobiilinumber. Andme omadus PersonalIdentificationcode on samaväärne andme omadusega isikukood. Andme omadus Phonenumber on samaväärne andme omadusega telefoninumber. Andme omadus Residence on samaväärne andme omadusega elukoht. Kõik samatähenduslikud andme omadused edastavad oma andmed samanimelistele muutujatele päringus. Päringud vastavusse viimine samatähenduslike andme omaduste järgi. Viimasena antakse kaasa päring, mis küsib andmeid ontoloogiast.

```
service.uri=http://localhost:3030/me
```

```
fusekiService.execSelectAndProcessPerson(
                            "ee",
                            env.getProperty("service.uri.data"),
                            commonOntologyQueries.generatePersonQuery("ee")
                                );
```

Koodinäide 3.4: Fuseki teenuse käivitamise näide

Vastavusse viidud päring läheb FusekiService teenusele. Teenus teeb päringu fuseki serverile, milles asuvad ontoloogiad. Saadud vastused viiakse vastavusse objekti, mida hiljem kasutatakse. Vastavusseviimiseks kasutatakse kahte erinevas keeles olevat ontoloogiat. Üheks neist on inglise keelne ontoloogia ja teiseks on eesti keelne ontoloogia. Selleks, et ontoloogiad vastavusse viia ja joondada need ühiseks on vaja kätte saada objektid ontoloogiast ning muuta need sobivaks uue ontoloogia struktuuriga. Selleks, et saada aru eestikeelsest ontoloogiast on loodud juba tõlgendatud päring. Tõlgendused on õiges keeles loodud mõlema riigi ontoloogiatele. Eritüüpi andmetele on tehtud erinevad päringud. Järgnevalt on välja toodud näide, kuidas tehakse päring, saadakse andmed ja need salvestatakse QueryResult objekti.

```
public List<QueryResult> execSelectAndProcessPerson(
    String language, String serviceURI, String query) {
  List<QueryResult> queryResult = new ArrayList<>();
  QueryExecution q = QueryExecutionFactory.sparqlService(serviceURI,
      query);
  ResultSet results = q.execSelect();
  while (results.hasNext()) {
    QuerySolution soln = results.nextSolution();
    RDFNode k = soln.get("k");
    RDFNode m = soln.get("m");
    RDFNode s = soln.get("s");
    RDFNode n = soln.get("n");
    RDFNode d = soln.get("d");
    RDFNode qa = soln.get("q");
    RDFNode w = soln.get("w");
    RDFNode e = soln.get("e");
    RDFNode r = soln.get("r");
    RDFNode t = soln.get("t");
    RDFNode l = soln.get("l");
```

```
        RDFNode c = soln.get("c");
        queryResult.add(
            new QueryResult(
                language,
                "person",
                String.valueOf(k),
                String.valueOf(m),
                String.valueOf(s),
                String.valueOf(n),
                String.valueOf(d),
                String.valueOf(qa),
                String.valueOf(w),
                String.valueOf(e),
                String.valueOf(r),
                String.valueOf(t),
                String.valueOf(l),
                String.valueOf(c)));
    }
    q.close();
    return queryResult;
}
```

Koodinäide 3.5: Ontoloogiale päringu tegemise näide

Esimese asjana luuakse 3.5 koodinäites järjend (inglise keeles list) andmetüüp, mille sees hoitakse QueryResult tüüpi objekte. Teise tegevusena käivitatakse päring ning see saadetakse Fuseki serverile.. Kaasa antakse päring ning url päringu tegemiseks. Peale seda vastab server vastusega, mida hakatakse töötlema. Vastus võib sisaldada andmeid, või siis olla ka tühi, kui serveril puuduvad andmed päringu sisu kohta. Vastuseid hakatakse ükshaaval üle vaatama while tüüpi tsükli sees. Kõigepealt luuakse muutuja, mis sisaldab küsitud muutujate väärtuseid. Seejärel luuakse RDF sõlme muutuja, kuhu hakatakse omistama vastuses olevaid väärtuseid. Kõige esimesena omistame muutujale k (objekti nimi) väärtuse päringu muutujast ?k. Muutujale s (eesnimi) omistatakse väärtus päringu muutujast ?s. Muutujale n (vanus) omistatakse väärtus päringu muutujast ?n. Muutujale d (perekonnanimi) omistatakse väärtus päringu muutujast ?d. Muutujale qa (sugu) omistatakse väärtus päringu muutujast ?q. Muutujale w (email) omistatakse väärtus päringu muutujast ?w. Muutujale e

(mobiilinumber) omistatakse väärtus päringu muutujast ?e. Muutujale r (isikukood) omistatakse väärtus päringu muutujast r. Muutujale t (telefoninumber) omistatakse väärtus päringu muutujast ?t. Muutujale l (rahvus) omistatakse väärtus päringu muutujast ?l. Muutujale c (elukoht) omistatakse väärtus päringu muutujast ?c. Kõik muutujad on samade nimega, mida kasutatakse päringu tegemiseks, et muutujate kasutus oleks arusaadavam. Kõik muutujad koos väärtustega salvestatakse QueryResult objekti, mis lisatakse seejärel järjendisse. Seda tsüklit sooritatakse kuni päringus pole enam vastuseid. Kõige lõpuks tagastatakse järjend sisuga. QueryResult objekti näide tabeli kujul 3.4:

Objekti atribuudi nimi	Päringu muutuja	Objekti väärtus
lang		est
type		Person
objectName	?k (objekti nimi)	inimene1
objectType	?m	Inimene
objectValue1	?s (eesnimi)	Kevin
objectValue2	?n (vanus)	27
objectValue3	?d (perekonnanimi)	Laanemagi
objectValue4	?q (sugu)	M
objectValue5	?w (email)	kevin.laanemagi@gmail.com
objectValue6	?e (mobiilinumber)	55556566
objectValue7	?r (isikukood)	39110065851
objectValue8	?t (telefoninumber)	+37265891
objectValue9	?l (rahvus)	EST
objectValue10	?c (elukoht)	Eesti, Läänemaa, Haapsalu

Tabel 3.4: QueryResult objekt

Tegemist on objektiga, milles atribuudid kasutavad üldnimesid. Kõik atribuudid on string tüüpi. Objekti sisestatakse näidiskoodis andmed järjekorras: keel, tüüp, objekti nimi, objekti tüüp, muutuja s (eesnimi), muutuja n (vanus), muutuja d (perekonnanimi), muutuja qa (sugu), muutuja w (email), muutuja e (mobiilinumber), muutuja r (isikukood), muutuja t (telefoninumber), muutuja l (rahvus), muutuja c (elukoht). Ühise ontoloogia lugemise meetodi näide 3.6.

```
public void generateCommonOntology(List<QueryResult> queryResultList,
    String lang) {
  try {
    generateRDFs(
        ontologyReaderService.getOntologyLocally("OntologyStructure.
            rdf"), queryResultList, lang);
  } catch (IOException e) {
    e.printStackTrace();
  }
}
```

Koodinäide 3.6: Ühise ontoloogia lugemise meetod

Koodinäites 3.7 loetakse muutujasse ühise ontoloogia mudel. Järgnevalt kirjeldatakse, kuidas loetakse ontoloogia mudelisse. Lugemise meetodile antakse kaasa ühise ontoloogia nimi. Kõigepealt on vaja luua ontoloogia tühi mudel. Mudeli juures määratakse ära, et tegemist oleks täieliku OWL standardile kuuluva mudeliga, mis ehitatakse esialgu arvuti mällu.

```
OntModel model = ModelFactory.createOntologyModel(OntModelSpec.OWL_MEM
    );
```

Koodinäide 3.7: Ontoloogia mudeli lugemine muutujasse

Järgmise sammuna leitakse selle asukoht arvuti kettalt, kasutades selleks faili nime OntologyStructure.rdf 3.8.

```
String path = fileFindComponent.getFilePath("OntologyStructure.rdf");
```

Koodinäide 3.8: Ontoloogia otsimine

Viimase sammuna loetakse tühja mälus paiknevasse mudelisse ontoloogia, mis sai sisse loetud arvuti kettalt ning kirjeldatakse, millist tüüpi ontoloogiaga on tegemist "RDF/XML"3.9.

```
model.read(new FileInputStream(path), null, "RDF/XML");
```

Koodinäide 3.9: Ontoloogia lugemine mudelisse

Nende sammudega on ühine ontoloogiastruktuur sisse loetud ning valmis prototüübis kasutamiseks. Ühise struktuuriga ontoloogiale andmete lisamise koodinäide 3.10:

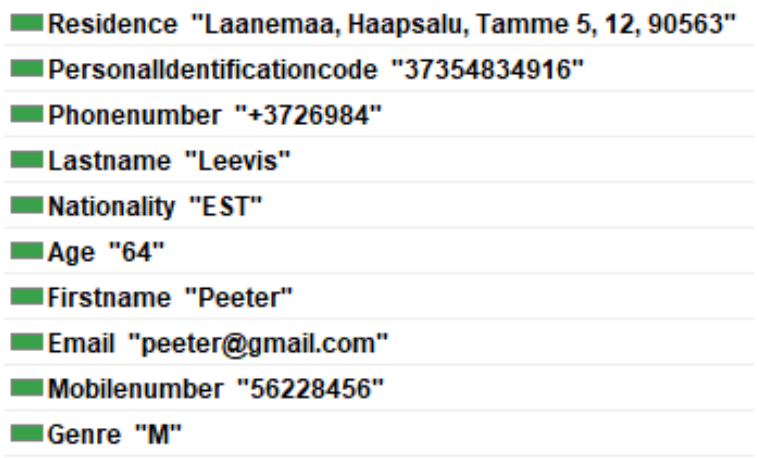
```
ExtendedIterator classes = model.listClasses();
while (classes.hasNext()) {
  OntClass thisClass = (OntClass) classes.next();
  if (thisClass
      .toString()
      .equals("http://www.semanticweb.org/pere/ontologies/2019/2/
          untitled-ontology-6#Person")) {
    Individual ind = thisClass.createIndividual(NS + objectName);
    ind.addRDFType(OWL2.NamedIndividual);
    ind.addProperty(firstName, model.createTypedLiteral(
        objectValue1, XSDDatatype.XSDstring));
    ind.addProperty(lastname, model.createTypedLiteral(objectValue3
        , XSDDatatype.XSDstring));
    ind.addProperty(
        personalIdentificationcode,
        model.createTypedLiteral(objectValue7, XSDDatatype.
            XSDstring));
    ind.addProperty(
        mobilenumber, model.createTypedLiteral(objectValue6,
            XSDDatatype.XSDstring));
    ind.addProperty(email, model.createTypedLiteral(objectValue5,
        XSDDatatype.XSDstring));
    ind.addProperty(genre, model.createTypedLiteral(objectValue4,
        XSDDatatype.XSDstring));
    ind.addProperty(phonenumber, model.createTypedLiteral(
        objectValue8, XSDDatatype.XSDstring));
    ind.addProperty(age, model.createTypedLiteral(objectValue2,
        XSDDatatype.XSDstring));
    ind.addProperty(nationality, model.createTypedLiteral(
```

```
            objectValue9, XSDDatatype.XSDstring));
        ind.addProperty(residence, model.createTypedLiteral(
            objectValue10, XSDDatatype.XSDstring));
    }
}
```

Koodinäide 3.10: Ühise struktuuriga ontoloogiale andmete lisamise koodinäide

Koodinäites täiendatakse inimese klassi osa ja ehitatakse eksemplarid sellele klassile ühises ontoloogias. Kõigepealt otsitakse ontoloogia mudelist model ülesse klassi „Person“. Kui see klass leitakse hakatakse ontoloogiat täiendama. Luuakse eksemplar, mis lisatakse klassile „Person“. Eksemplarile lisatakse järgnevad andme omadused koos väärtustega, mis saadi vastava riigi joondamata ontoloogiast, millest hakati looma ühtset ontoloogiat. Eksemplarile lisatakse järgmised andmeomadused: Firstname, Lastname, Mobilenumber, Email, Genre, Phonenumber, Age, Nationality, Residence, PersonalIdentificationcode. Igale andmeomadusele lisatakse väärtused baasist. Lõplik „Person“ klassi eksemplar on joonisel 3.7. Ühises ontoloogias lisatakse vajalikud andmed järgmistele klassidele:

1. Person (Inimene)
2. School (Kool)
3. Taxes (Maksud)
4. Bank (Pank)
5. Family members (Perekonnaliikmed)
6. Socialministry (Sotsiaalministeerium)



Joonis 3.7: Moraania vastavusseviidud ontoloogia inimese eksemplar

Järgnevas koodinäites 3.11 luuakse andmeomadused, mida kasutatakse ontoloogia ehitamisel. Andmeomaduste loomise näide:

```
DatatypeProperty doctorName = m.createDatatypeProperty(NS + "
    Doctorname");
DatatypeProperty fullName = m.createDatatypeProperty(NS + "Fullname
    ");
DatatypeProperty handicapLevel = m.createDatatypeProperty(NS + "
    Handicaplevel");
```

Koodinäide 3.11: Omaduste loomise näide

Selleks, et viia ontoloogia joondamine lõpuni on vaja vastavusse viidud ontoloogia kirjutada faili. Koodinäites 3.12 otsitakse failinimi, millesse salvestatakse mälus olev ühine vastavusse viidud ontoloogia. Faili asukoha nimesid hoitakse 'application.properties' failis.

```
fileName = env.getProperty("negooria.ontology.file.location");
```

Koodinäide 3.12: Faili nime otsimise näide

Faili asukoha näidis 3.13.

```
moraania.ontology.file.location = C://Users/Dell Alienware/Desktop/toop
    /src/main/resources/MoraaniaOntology.ttl
```

Koodinäide 3.13: Faili asukoha määramise näide

Järgmiseks luuakse faili kirjutaja objekti koos muutujaga 'out' 3.14. Sellele objektile omistatakse fail, mis asub eelnevas näites toodud asukohas.

```
FileWriter out = new FileWriter(fileName);;
```

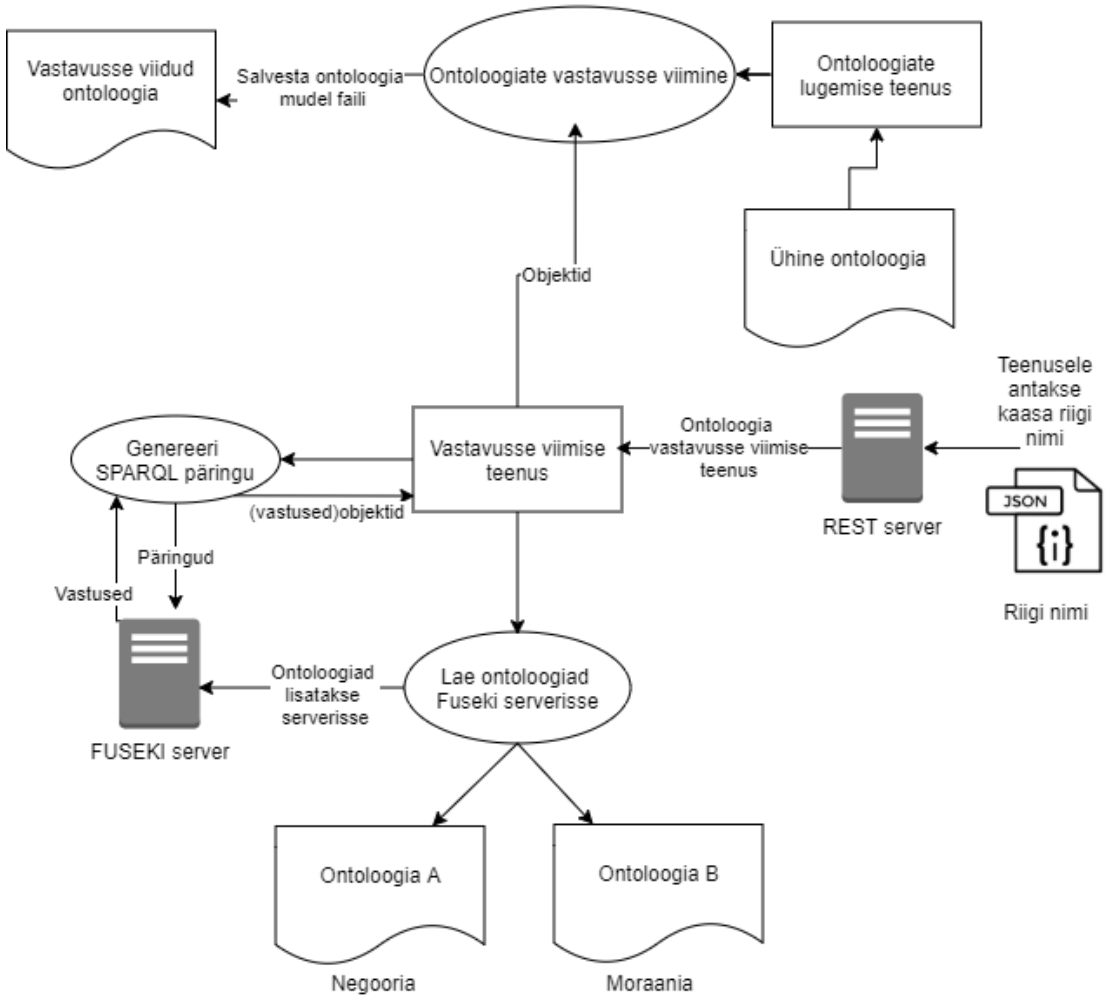
Koodinäide 3.14: Faili kirjutaja objekti loomine

3.15 toodud näites kirjutatakse ontoloogia mudel faili ja fail pannakse pärast seda kinni. Mudeli faili kirjutamisel kirjeldatakse, millisesse faili sisu kirjutatakse ja mis vormingus ontoloogia kirjutatakse. Töös kirjutatakse ontoloogia "TTL"ehk Turtle vormingus. Turtle ehk Terse RDF Triple Language on süntaksi- ja failivorming andmete väljendamiseks RDF andmemudelil. See on levinud andmemudel RDF andmete talletamiseks [51].

```
model.write(out, "TTL");
out.close();
```

Koodinäide 3.15: Mudeli faili kirjutamine

Eelnevalt kirjeldatud fail sisaldab lõpuks ühist vastavusseviidud ontoloogiat. Joonisel 3.8 on välja toodud prototüübi vastavusseviimise arhitektuur.



Joonis 3.8: Ontoloogiate vastavusse seadmise tegevuste kirjeldusega arhitektuur semantika vahendis

Peatükk 4

Valideerimine ja probleemid

4.1 Semantika probleemid ja ettepanekud tulevaseks tööks

4.1.1 Töö analüüs

Töö tulemusena valmis prototüüp, milles saab teostada kolme e-toimingut. Prototüübi jaoks loodi nõuded, mida rakendades ületati takistused, mis esinevad riikide vaheliste ontoloogia abil andme vahetusega seotud süsteemide juures.

Tulemuse saavutamiseks uuriti erinevaid võimalusi, kuidas luua sellist semantikavahendit ning lõpuks tuginedes nõuetele, valmis semantikavahendi arhitektuur, millele tuginedes loodi prototüüp. Töös kasutatud vastavusse viimata ontoloogiad erinevad nii terminoloogia kui ka keele poolest. Prototüübi abil viiakse kaks erinevat semantikat vastavusse. Vastavusseviidud ontoloogiatest olid taotluste jaoks vaja minevad andmed kättesaadavad, kasutades päringuid. Töös sai kasutada sama struktuuri ja terminoloogiaga päringuid pärast vastavusse viimist. Menetluse käigus valideeriti taotlused ja taotleja sai koheselt vastuse taotluse tulemusest. E-toiminguid tehes saab kodanik vastuse koheselt ja ei pea selle toimingu tegemiseks teise riiki sõitma. E-toiminguid sai teha asukoha riigist sõltumatult. Töö vastab nõuetele, mis on välja toodud alapeatükis 3.1. Tööd on võimalik edasi arendada ning kasutada

semantikasüsteemide ehitamise näidisena.

4.1.2 Ettepanekud tulevaseks tööks

Käesolevas töös õnnestus luua semantiline vahend, mis realiseeriti prototüübina, toetudes uuringutulemustele. Töö käigus loodud semantiline vahend töötas edukalt ning teostas menetlusi etteantud nõuete raames, aga selleks, et see töötaks reaalsetel tingimustel, on vaja seda edasi arendada. Järgnevalt on välja toodud mõned ideed, kuidas saaks antud tööd paremaks teha ja edasi arendada. Semantilise vahendi edukus sõltub riikidevahelistest andmete vahetamisest. Selleks, et andmete vahetamine töötaks, on vaja riigil ontoloogiat. Ontoloogiate struktuuriline ja terminoloogiline erinevus lahendatakse semantilises vahendis ontoloogiate vastavusse seadmisega. Selleks, et vastavusse seadmine oleks edukas, oleks vaja tulevikus rakendada masintõlget ontoloogiate tõlkimiseks. Selle abil saaks suurema osa keelelisi probleeme lahendada.

Lisaks tuleks muuta ontoloogiate vastavusse seadmist. Semantikavahendis tuleks vastavusse seadmine viia taotluse päringusse. Igal korral, kui tehakse uus taotlus, oleks vaja luua vastavusse seatud ontoloogia kodanike andmetega ja peale päringu lõppu kustutatakse päringu käigus loodud ontoloogia ära. Selle abil saaks vältida andmete liigset avalikustamist ja hoiustamist.

Semantilist vahendit tuleks reaalsel kasutusel testida, et leida nõrgad kohad ning muuta seda stabiilsemaks ja mugavamaks. Selle tulemusel saaksime näha, kuidas see reaalsetel tingimustel töötaks. Kasutusele võiks võtta ka ühissõnavara raamistiku [52], kus kõik tõlked ja erinevad tähendused sõnade kohta lahti kirjutada, et seda saaks hiljem semantilises vahendis kasutusele võtta.

Antud töös loodud semantilisele vahendile saab juurde lisada menetlusi. Lisaks töös kasutatavatele menetlustele võib selle vahendi abil automatiseerida ka muid menetlusi. Lisada võib ka juurde taotluseid ja olemasolevaid taotluseid täiendada.

4.1.3 Probleemid

Iga asja loomisel on omad kasulikud omadused ja ka kitsendused, mis piiravad loomist ning panevad otsima teisi võimalusi, kuidas neist mööda pääseda. Antud lõik käsitleb töö käigus välja tulnud kitsendusi ja kasulikke omadusi. Alguses tuuakse välja kasutatud lähenemiste üldised kitsendused ja kasulikud omadused, mis on prototüübiks kasutatud tehnoloogiatel ning lõpus on välja toodud samad asjad prototüübi kohta. Üldisemateks suurematekskitsendusteks on ontoloogiaga seotud ning selle töötlemisega seotud kitsendused. Kitsendusi esines ontoloogias, selle kaardistamises, tõlkimises, teisendamises ning kogu arhitektuuri loomisel.

Suuremad kitsendused algavad juba ontoloogiatest ja sellega seotud protsessidest. Kogu prototüübi ulatuses kasutatakse kahte erinevat ontoloogiat. Kuigi me kõik sooviksime, et meil oleks ühtne standard ontoloogiate struktuuri koostamiseks on reaalne tõde see, et iga disainer struktureerib ontoloogiaid erinevat moodi. Sellest tuleb keerukus selle teisaldamisel ühtseks, sest ontoloogiatest arusaamine võib erineda ja selle mõistmiseks tuleb läheneda teisiti. Struktuurilt erinevatel ontoloogiatel võib esineda järgmiseid probleeme:

1. Erinevad terminid kirjeldavad sama mõistet
2. Samad terminid kirjeldavad erinevaid mõisteid.
3. Erinevad modelleerimise paradigmad.
4. Erinevad detailsuse tasemed.
5. Erinev kasutatavus.

Riigiti kasutatavad ontoloogiad võivad erineda nõuete poolest ning olemasolevaid ontoloogiaid tuleb ümbertöödelda, sest need on mõeldud teisiti lähenemiste jaoks ja teiste ülesannete jaoks.

Erinevate ülesannete lahendamiseks on erinevad konventsioonid. Lepingutel võivad olla erinevad formaadid ja nende andmed ja nende andmetüübid võivad üksteisest

erineda. Samuti võivad olla kohalikud ontoloogiad loodud konventsioonidele toetudes, mis lisab andmete kättesaamisele keerukust juurde.

Üheks kõige suuremaks probleemiks rahvusvaheliste süsteemide puhul on see, et erinevad riigid on arendanud ja kasutavad omakeelseid ontoloogiaid, mis ei ole üksteisega sarnased. Sellega tekib mittevastavus keele tasemel. Paljudel keeltel, mida kasutatakse Euroopas, on väljendeid, mida ei ole võimalik ühtselt väljendada. Antud töö kontekstis võib eestikeelses semantikas olla väljendeid, mis ei ole ühtselt mõistetav inglisekeelses semantikas. Samas võib ka juhtuda seda, et erinevates ontoloogiakeeltes ontoloogiad tuleb kõigepealt ühtsesse ontoloogiakeelde tõlkida OWLiks või RDFiks.

Päringute tegemise juures ontoloogiate vahel võib olla keeruline andmeid kätte saada, kui ontoloogiad erinevad struktuurilt ja keelelt. Ehitades suurt süsteemi ei ole kasulik luua iga erineva keele jaoks erinevaid päringuid ning selle vältimiseks peaks looma ühtselt arusaadava ontoloogiastruktuuri, mida kasutades saada vastused päringutele. Prototüübis kasutatakse päringute tegemiseks inglise keelt. Eestikeelsest ontoloogiast vastuste kättesaamiseks kasutatakse tõlgitud päringuid.

4.2 Valideerimine

Siin alapeatükis räägitakse semantilise vahendi valideerimisest. Kas lõplik magistritöös realiseeritud lahendus vastab nõuetele ja eesmärkidele, mis on eelnevalt seatud. Semantiline vahend realiseeriti prototüübina, milles realiseeritakse kolme kodanikkudele suunatud e-toimingut kahe riigi vahel. Semantilise vahendi valideerimiseks peavad kõik eelnevalt seatud nõuded töötama prototüübis.

Uuringute tulemusena disainitud semantika vahend vastab etteseatud nõuetele. Kõik nõuete peatükis väljatoodud nõuded on implementeeritud prototüüpi. Prototüüpi ontoloogiate vastavusse viimiseks on loodud 3 ontoloogiat. Ühise ontoloogia struktuuri ja terminoloogia järgi viiakse ontoloogiad vastavusse. Prototüübis kasutatakse Moraania ja Negooria ontoloogiat. Riiklikus ontoloogias on ainult selle riigi andmed. Riiklikud ontoloogiad erinevad terminite ja keele poolest. Ontoloogiaid ei

viida vastavusse igal päringul. Vastavusse viidud ontoloogiates kasutatakse andmete pärimiseks sama terminoloogiaga päringuid. Ontoloogiaid hoitakse eraldi ning prototüüp suudab neile päringuid teha. Kui kodanik teeb päringu taotluse tegemiseks, annab ta kaasa enda andmed JSON objektina, mida kasutatakse menetluses. Prototüüp vastab taotluse tegemise päringule menetluse tulemuseks tulnud otsusega. Otsuse andmetega tagastatakse fail. Taotlused valideeritakse menetluse käigus. Iga taotluse kohta on prototüübis kasutusele võetud kaks kodanikku, kelle andmete põhjal menetlused läbi viiakse. Semantikavahend on loodud RESTful veebirakendusena.

Prototüüp vastab osadele TOOPi nõuetele. Prototüüp kasutab ühist sõnavara. Prototüübile on selgeks tehtud erinevate sõnade tähendused ning tõlgendused. Prototüüp viib ontoloogiad vastavusse, et need oleks sama struktuuriga ja kasutaks sama terminoloogiat. Iga riik hoiab ontoloogiat riigisiseselt, prototüüp ei salvesta kasutatud andmeid. Andmed küsitakse iga päringu korral uuesti ontoloogiatelt. Andmed kogutakse riigisiseselt kokku ainult ühekordselt ning hoitakse riiklikus ontoloogias [1].

Lõplik magistritöös realiseeritud prototüüp vastab magistritöö käigus seatud nõuetele ning lubab kasutada kolme kodanikele suunatud e-toimingut kahe töös kasutatava riigi vahel.

4.3 Prototüübis kasutatud tööriistad

Magistritöö semantilise vahendi loomiseks valiti välja järgnevad tööriistad. Tööriistad valiti järgnevad, toetudes nõuetele ja kirjandusele.

Spring-raamistik on rakenduseraamistik ja kontrollkonteineri inversioon Java platvormil. Raamistiku tuumikomadusi võib kasutada igas Java rakenduses aga seal on ka olemas laiendusi, mis võimaldavad ehitada veebirakendusi Java EE platvormile. Tegu on avatud lähtekoodiga raamistikuga, mille kõige uuem versioon on 5.1.5. Esimene versioon tuli välja aastal 2002 [53].

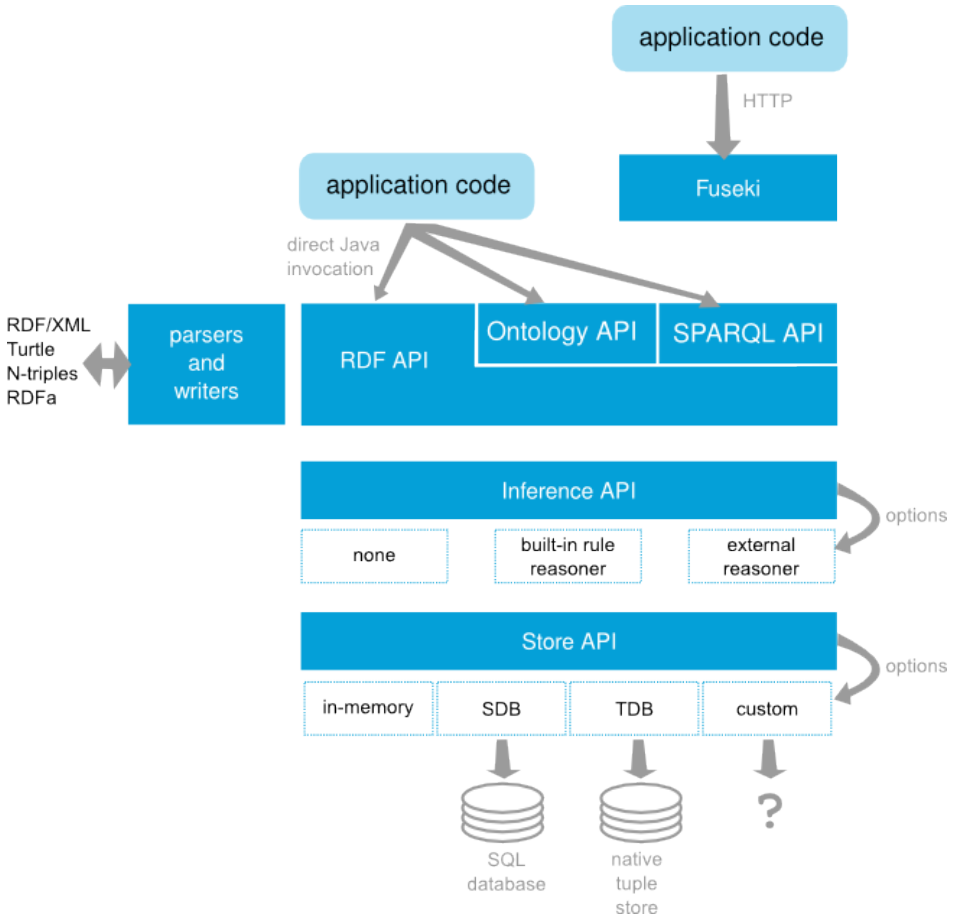
REST API ehk REST rakendusliides on arhitektuuristiil kindlate piiridega veebi-

rakenduste loomiseks. Sellel arhitektuuril ehitatud rakendused tagavad võrgus koostoimimise. Põhiidee seisneb selles, et tehakse päringud REST-arhitektuuriga ülesse seotud URLidele ja seejärel vastab veebirakendus vastustega [54].

Java on programmeerimiskeel ja platvorm, mis on loodud Sun Microsystemsi poolt 1995. aastal [55].

Protégé on avatud lähtekoodiga ontoloogiate töötlemise tarkvara ja intelligentsete süsteemide ehitamise raamistik [6].

Apache Jena on avatud lähtekoodiga Java keelel baseeruv semantilise veebi raamistik ja pakub APId andmete ekstraktimist RDFist ja nende kirjutamist RDFiks-. See tuli välja Hewlett- Packard's Semantic Web Research Lab firmast. Apache Jenal on sisseehitatud tugi ontoloogiakeeltele ning SPARQL päringute tugi. Apache Jena Fuseki on SPARQL server. Joonisel 4.1 on välja toodud arhitektuur [8].



Joonis 4.1: Apache Jena arhitektuur [8]

4.4 Kokkuvõte

Lõputöö eesmärk oli uurida, kuidas toimib semantiline koosvõime andmete ühekordse küsimise keskkonnas kahe riigi vahel ning uuringu tulemuste põhjal luua semantika vahend, mis võimaldaks läbi viia kolme kodanikele suunatud riikide vahelist e-toimingut kasutades TOOP-i[2] nõudeid.

Eesmärkide täitmiseks uuriti erinevaid võimalusi sellise semantika vahendi loomiseks. Kõigepealt uuriti, milliseid sarnaseid süsteeme on varem loodud, seejärel tehti kindlaks nõuded, mida semantikavahend täitma peab. Seejärel uuriti, kuidas menetlusi läbi viia ja loodi semantilise vahendi arhitektuur. Edasi uuriti tehnoloogiaid, mida kasutades saaks luua nõuetele vastava prototüübi.

Töö tulemusena valmis prototüüp riikide vahelise e-toimingute teostamise jaoks. Prototüüp implementeeriti nõuete põhjal ja see ehitati kasutades valitud tehnoloogiaid ja arhitektuuri. Arhitektuuri loomisel mõeldi läbi põhikomponendid, mida peaks semantiline vahend sisaldama, et teostada taotluseid. Arhitektuuris kasutati kahte RESTful veebirakendust. Üks neist oli prototüüp ja teine oli menetluste valideerimise veebirakendus, mille eesmärk oli implementeerida taotluse valideerimist erinevates riikides. Prototüübi kaudu saab teha kolme e-toimingut, kasutades kasutajalugudes olevate kodanike andmeid. Taotluse tegija saab vastuse taotluse tulemusest.

Valideerimise tulemusena leiti, et prototüüp vastab nõuetele. Prototüübis kasutatakse iga riigi jaoks erinevat ontoloogiat, mis viiakse vastavusse, kasutades ühist ontoloogia struktuuri. Prototüübis kasutatakse ka TOOP nõudeid. Prototüüp vastab taotlusele vastusega taotluse tulemusest. Taotluse vastus sisaldab andmeid, mis pärinevad riiklikest ontoloogiatest. Moraania ja Negooria ontoloogiad sisaldavad kumbki ainult enda riigi andmeid. Riiklikud ontoloogiad erinevad keele ja terminoloogia poolest. Prototüübis kasutatakse ühist sõnavara ja selles on selgeks tehtud erinevate sõnade tähendused ning tõlgendused. Andmed küsitakse iga päringu korral uuesti ontoloogiatelt. Taotluse tegija annab kaasa JSON objekti enda andmetega. Prototüüp on ehitatud RESTful veebirakendusena. Lõputööle toetudes on võimalik luua riikidevahelisi andmevahetussüsteeme. Selle tulemusel valminud prototüüpi

peaks hakkama testima kasutajatega ja reaalsete andmetega. Seda saab kasutada kui uurimust, millest leiab informatsiooni, kuidas valmib semantiline vahend.

Kirjandus

[1] Jerry Dimitriou Sander Fieten Carmen Rotuna Jaak Tepandi Ermo Täks Jack Verhoosel DimitriosŽeginis Eric Grandry, Paul Brandt. The once-only principle project generic federated oop architecture (2nd version), 2018. URL `https://toop.eu/sites/default/files/D22_Generic_Federated_OOP_Architecture_Final.pdf`.

[2] Fernando Ortiz-Rodriguez, Melchor Medina, Demian Abrego, and Manuel Zuniga. Smart government an interoperability and multi dialect solution to improve citizens services. In *Proceedings of the International Conference on e-Learning, e-Business, Enterprise Information Systems, and e-Government (EEE)*, pages 81–84. The Steering Committee of The World Congress in Computer Science, Computer . . . , 2018.

[3] Maria Pilar BeltránÕrenes José Angel Martínez Usero. Ontologies in the context of knowledge organization and interoperability in egovernment services. *Conference on Digital Divide, Global Development and the Information Society*, 2005.

[4] Jim Hendler Ian Horrocks Deborah L. McGuinness Peter F. Patel-Schneider Sean Bechhofer, Frank van Harmelen. Owl web ontology language, 2009. URL `https://www.w3.org/TR/owl-ref/`.

[5] EUROPEAN PARLIAMENT. "euroopa komisjon (2017) euroopa parlamendi ja nõukogu määrus millega luuakse ühtne digivärav teabe ja menetluste ning abi- ja probleemilahendamisteenuste pakkumiseks ning millega muudetakse määrust (el) nr 1024/2012", 2017. URL `https://eur-lex.europa.eu/legal-content/ET/TXT/1&from=en`.

[6] Rafael Goncalves Matthew Horridge Samson Tu Tania Tudorache Mark Musen, Csongor Nyulas. Protege, 2018. URL https://protege.stanford.edu/.

[7] Christian Bizer, Tom Heath, and Tim Berners-Lee. Linked data - the story so far. *Int. J. Semantic Web Inf. Syst.*, 5:1–22, 2009.

[8] The ApacheŠoftware Foundation. Apache jena, 2018. URL https://jena.apache.org/.

[9] Paweł Kapłański. Controlled english interface for knowledge bases. *Studia Informatica*, 32(2A):485–494, 2011.

[10] David R Musser and Alexander A Stepanov. Generic programming. In *International Symposium on Symbolic and Algebraic Computation*, pages 13–25. Springer, 1988.

[11] Fluent editor. https://www.softpedia.com/get/Programming/File-Editors/Fluent-Editor-for-OWL.shtml#sgal_0, 2020. Accessed: 2020-01-29.

[12] Prof. Enrico Motta NeOn Foundation. Neon-toolkit wiki, 2014. URL http://neon-toolkit.org/wiki/Main_Page.html.

[13] Noman Islam, MuhammadŠhahab Siddiqui, and Zubair Ahmed Shaikh. Tode : A dot net based tool for ontology development and editing. *2010 2nd International Conference on Computer Engineering and Technology*, 6:V6–229–V6–233, 2010.

[14] Vajaduspõhine õppetoetus. https://www.hm.ee/et/tegevused/korgharidus/oppetoetus, 2020. Accessed: 2020-01-27.

[15] Mokhtaria Hacherouf, Safia Nait-Bahloul, and Christophe Cruz. Transforming xml schemas into owl ontologies using formal concept analysis. *Software & Systems Modeling*, Jan 2018. ISSN 1619-1374. doi: 10.1007/s10270-017-0651-4. URL https://doi.org/10.1007/s10270-017-0651-4.

[16] Thomas R Gruber. A translation approach to portable ontology specifications. *Knowledge acquisition*, 5(2):199–220, 1993.

[17] Pierre Monnin, Amedeo Napoli, and Adrien Coulet. Combining concept annotation and pattern structures for guiding ontology mapping. In *6th International Workshop"What can FCA do for Artificial Intelligence", FCA4AI@ IJCAI2018*, number 2149, 2018.

[18] Abdel-Badeeh M Salem and Silvia Parusheva. Developing a web-based ontology for e-business. *"International Journal of Electronic Commerce Studies"*, 9(2): 119–132, 2018.

[19] Richard J. Daddieco. Retrieving knowledge in e-government: The prospects of ontology for regulatory domain record keeping systems. pages 111–121, 05 2004.

[20] Fathia Bettahar, Claude Moulin, and Jean-Paul Barthès. Towards a semantic interoperability in an e-government application. *Electronic Journal of E-government*, 7(3):209–226, 2009.

[21] Petar Milić, Nataša Veljković, and Leonid Stoimenov. Semantic technologies in e-government: Toward openness and transparency. In *Smart Technologies for Smart Governments*, pages 55–66. Springer, 2018.

[22] Victoria Beltran, Knarig Arabshian, and Henning Schulzrinne. Ontology-based user-defined rules and context-aware service composition system. In *Extended Semantic Web Conference*, pages 139–155. Springer, 2011.

[23] DARPA. The darpa agent markup language homepage, 2006. URL `http://www.daml.org/`.

[24] DA Fensel, FAH van Harmelen, JM Akkermans, M Klein, Jeen Broekstra, Christiaan Fluit, J Meer, H-P Schnurr, RŠtuder, J Davies, et al. Ontology-based tools for knowledge management. 2000.

[25] Dieter Fensel, Frank Van Harmelen, Ian Horrocks, Deborah L McGuinness, and Peter F Patel-Schneider. Oil: An ontology infrastructure for the semantic web. *IEEE intelligent systems*, 16(2):38–45, 2001.

[26] EUROPEAN PARLIAMENT. Cross-border e-services for business mobility, 2017. URL `http://toop.eu/pilot-area1`.

[27] TOOP. Online ship and crew certificates, 2017. URL http://toop.eu/pilot-area3.

[28] Ali Albarghothi, Walaa Saber, and Khaled Shaalan. Automatic construction of e-government services ontology from arabic webpages. *Procedia Computer Science*, 142:104 – 113, 2018. ISSN 1877-0509. doi: https://doi.org/10.1016/j.procs.2018.10.465. URL http://www.sciencedirect.com/science/article/pii/S1877050918321677. Arabic Computational Linguistics.

[29] Aya M. Al-Zoghby and Khaled Shaalan. Conceptual search for arabic web content. In Alexander Gelbukh, editor, *Computational Linguistics and Intelligent Text Processing*, pages 405–416, Cham, 2015. Springer International Publishing.

[30] Maryam Hazman, Samhaa El-Beltagy, and Ahmed Rafea. An ontology based approach for automatically annotating document segments. *IJCSI International Journal of Computer Science Issues*, 1:221–230, 03 2012.

[31] Ibrahim Imam, Nihal Nounou, Dr Hamouda, and Hebat Allah Abdul Khalek. An ontology-based summarization system for arabic documents (ossad). *International Journal of Computer Applications*, 74:38–43, 07 2013. doi: 10.5120/12980-0237.

[32] Hani Al-Chalabi, Santosh Ray, and Khaled Shaalan. Semantic based query expansion for arabic question answering systems. In *2015 First International Conference on Arabic Computational Linguistics (ACLing)*, pages 127–132. IEEE, 2015.

[33] EUROPEAN PARLIAMENT. "regulation (eu) 2018/1724 of the european parliament and of the council", 2018. URL https://eur-lex.europa.eu/legal-content/EN/TXT/?qid=1549716594539\&uri=CELEX:32018R1724.

[34] Guido L. Geerts. A design science research methodology and its application to accounting information systems research. *International Journal of Accounting Information Systems*, 12(2):142 – 151, 2011. ISSN 1467-0895. doi: https://doi.org/10.1016/j.accinf.2011.02.004. URL http://www.sciencedirect.com/science/article/pii/S1467089511000200. Special Issue on Methodologies in AIS Research.

[35] Jacques Guyot, Saïd Radhouani, and Gilles Falquet. Ontology-based multilingual information retrieval. In *CLEF (Working Notes)*, 2005.

[36] Maung Sein, Ola Henfridsson, Sandeep Purao, Matti Rossi, and Rikard Lindgren. Action design research. 2011.

[37] Ontology editors. `https://www.w3.org/wiki/Ontology_editors`, 2020. Accessed: 2020-01-18.

[38] Peter Haase, Holger Lewen, Rudi Studer, and Michael Erdmann. The neon ontology engineering toolkit. 03 2019.

[39] Noman Islam, M.s Siddiqui, and Zubair Shaikh. Tode : A dot net based tool for ontology development and editing. volume 6, pages V6–229, 05 2010. doi: 10.1109/ICCET.2010.5486292.

[40] Alan Rector Robert Stevens Chris Wroe Simon Jupp Georgina Moulton Georgina Moulton Nick Drummond Sebastian Brandt Matthew Horridge, Holger Knublauch. A practical guide to building owl ontologies using protege 4 and co-ode tools edition 1.3. 108:10–13, 03 2011.

[41] N. Noy and Deborah Mcguinness. Ontology development 101: A guide to creating your first ontology. *Knowledge Systems Laboratory*, 32, 01 2001.

[42] Phillip Lord. "components of an ontology", 2010. URL `http://ontogenesis.knowledgeblog.org/514`.

[43] Dr. Roman V Belavkin. Lecture 8: Ontologies. 8:3.

[44] Kaladevi Ramar and G. Gurunathan. Technical review on ontology mapping techniques. 15:676–688, 01 2016. doi: 10.3923/ajit.2016.676.688.

[45] Gunter Saake, Kai-Uwe Sattler, and Stefan Conrad. Rule-based schema matching for ontology-based mediators. *Journal of Applied Logic*, 3 (2):253 – 270, 2005. ISSN 1570-8683. doi: https://doi.org/10.1016/j.jal.2004.07.021. URL `http://www.sciencedirect.com/science/article/pii/S1570868304000606`. Logic-based Methods for Information Integration.

[46] Student finance. https://collegelife.co/nl/guides/student-finance-guide-funding-holland/, 2020. Accessed: 2020-01-27.

[47] Pensioniiga, liigid ja soodustused. https://www.sotsiaalkindlustusamet.ee/et/pension/pension-liigid-ja-soodustused, 2020. Accessed: 2020-01-27.

[48] Aow home. https://www.svb.nl/en/aow-pension, 2020. Accessed: 2020-01-27.

[49] Puudega inimesele. https://www.sotsiaalkindlustusamet.ee/et/puue-ja-hoolekanne/puudega-inimesele, 2020. Accessed: 2020-01-27.

[50] Chapter v: Invalidity. https://ec.europa.eu/employment_social/soc-prot/missoc99/english/05/nl.htm, 2020. Accessed: 2020-01-27.

[51] Rdf 1.1 turtle. https://www.w3.org/TR/turtle/, 2014. Accessed: 2019-09-30.

[52] Core vocabularies. https://ec.europa.eu/isa2/solutions/core-vocabularies_en, 2020. Accessed: 2020-01-18.

[53] Spring boot. https://spring.io/projects/spring-boot, 2020. Accessed: 2019-09-30.

[54] Robert Battle and Edward Benson. Bridging the semantic web and web 2.0 with representational state transfer (rest). *Journal of Web Semantics*, 6(1): 61–69, 2008.

[55] Java. https://www.oracle.com/java/, 2020. Accessed: 2019-09-30.