TARTU ÜLIKOOL
MATEMAATIKA-INFORMAATIKATEADUSKOND
Arvutiteaduse instituut
Tarkvaratehnika õppetool
Informaatika eriala

Janno Veldemann

Ontoloogiate koostamise põhimõtted

Bakalaureusetöö (4 AP)

Juhendaja: Jaak Vilo, PhD

Autor: ..... "....." mai 2005
Juhendaja: ..... "....." mai 2005
Õppetooli juhataja: ..... "....." 2005

TARTU 2005

Sisukord

Sissejuhatus

Nüüdisaegsetes organisatsioonides on palju sellist informatsiooni ja teadmisi, mida tuleb süsteemselt ja korrektselt hallata. Erinevate organisatsioonide vaheline suhtlus vajab ühtset keelt, et möödarääkimiste osakaal oleks võimalikult väike. Üheks viisiks korrastada ja saavutada ühiseid arusaamu asjadest on kasutada selgelt defineeritud mõistete süsteeme e. ontoloogiaid. Lisaks inimestevahelisele suhtlusele organiseerivad ja võimaldavad ontoloogiad ka tarkvarasüsteemide vahelist suhtlust. Ontoloogiatena kirja pandud informatsiooni saavad mõlemad osapooled taaskasutada ning selle peale saab ehitada suhtluse nii organisatsioonisiseselt kui väliselt.

Käesoleva töö peamiseks eesmärgiks on uurida täpsemalt, mis on ontoloogiad ning kuidas neid kasutada saab. Vastuse peaks leidma küsimustele nagu: Kus on võimalik ontoloogiaid kasutada? Milliseid eeliseid annab nende kasutamine? Millest nad koosnevad ja millist kuju võivad omada? Millised on eeldused ja vahendid ontoloogite väljatöötamiseks?

Motivatsioon

Töö autor on puutunud kokku meditsiini valdkonna andmete kogumise tarkvara väljatöötamisega AS-is EGeen. Selle käigus on tekkinud vajadus paremini aru saada erinevate valdkondade võimalikest vajadustest. Praegu võimaldab AS EGeen välja arendatud tarkvaras kasutada suuri hierarhilisi mõistete süsteeme nagu ICD-9 ja ICD-10. Edaspidises arenduses aga vaja hakata arvesse võtma keerukamaid graafi-kujulisi ontoloogiaid ja muid meetode, kuidas integreerida mõistete süsteeme kasutajasõbralikku tarkvarakeskkonda, mida kasutab arvutikauge tavakasutaja, näiteks arst.

Töö ülesehitus

Esimeses peatükis selgitatakse ja kirjeldatakse mis on ontoloogia ning millest tekib vajadus selle järele. Teine peatükk vaatleb võimalikke ontoloogia kasutuselevõtu motivaatoreid ja kasutusviise lähemalt. Kolmandas peatükis jaotatakse ontoloogiad kahte

gruppi, vaadeldakse milliseid struktuure nad võivad omada ning milliste seoste abil on mõisted kirjeldatud. Neljas peatükk annab ülevaate, mil viisil on võimalik ontoloogiaid esitada, kuidas valida sobilik esituskeel ja arenduskeskkond. Lisaks kirjeldatakse lühidalt kolme arendusvahendit vastavalt arenduskeskkonna valikukriteeriumidele. Viies peatükk kirjeldab ontoloogia väljatöötamist kui protsessi e. millised sammud tuleb läbida ontoloogia väljaarendamisel.

I PEATÜKK: ONTOLOOGIA MÕISTE

1.1 Probleemid

Inimesed, organisatsioonid, tarkvarasüsteemid peavad suhtlema omavahel ja teineteisega. Erinevate vajaduste ja tausta kontekstide tõttu võivad vaatenurgad ja oletused käsitletavale teemale olla väga erinevad isegi kui käsitletakse sama teemat. Esineb möödarääkimisi, kuna kasutatakse erinevaid, ülekaetud ja/või omavahel sobimatuid mõisteid, struktuure ja meetodeid. Ühiste arusaamade puudumise tagajärjeks on **nõrk kommunikatsioon** inimeste, organisatsioonide ning tarkvarasüsteemide vahel. Tarkvarasüsteemi ehitamisel aga tekitab ebaefektiivne suhtlus raskusi nõuete tuvastamisel ning seega ka süsteemi **spetsifikatsiooni** väljatöötamisel. Ühitamatud modelleerimismeetodid, paradigmad, keeled ja tarkvaravahendid piiravad tõsiselt **koostalitluse** ning **taaskasutuse** võimet. Siit omakorda jõutakse **ratta taasleiutamiseni** suurte raisatud jõupingutustega [UG96].

1.2 Lahendused

Eelpool kirjeldatud probleemide lahendamiseks tuleb vähendada või elimineerida kontseptuaalne ja terminoloogiline arusaamatus ning jõuda ühisele jagatud arusaamisele. Selline arusaamine toimib kui unifitseeriv raamistik erinevate vaatenurkade vahel ning on alus järgnevale:

- **Kommunikatsioon** erinevate vajaduste ja vaadetega inimeste vahel tulenevalt esitatava sõnumi erinevatest tähendustest;
- Süsteemide vahel saavutatud **koostalitlusvõime** erinevate modelleerimis-meetodite, paradigmade, keelte ja tarkvara tööriistade vahelise tõlkimise teel;
- **Süsteemitehnika kasu** e. täpsemalt

  Korduvkasutus (ingl. k. *reusability*) - jagatud arusaam on aluseks tähtsate olemite, atribuutide, protsesside ja nende omavaheliste suhete formaalseks kodeerimiseks huvipakkuvas valdkonnas. Selline formaalne esitus võib olla (või saada automaatse tõlkimise teel) taaskasutatav ja/või jagatud komponent tarkvarasüsteemis.

U*saldusväärsus* (ingl. k. r*eliability*) - formaalne esitus teeb võimalikuks automaatse terviklikkuse kontrollimise, mis viib tulemusena usaldusväärsema tarkvarani.

S*petsifikatsioon* (ingl. k. s*pecification*) - jagatud arusaam võib abistada infotehnoloogilise (IT) süsteemi nõuete tuvastamist ja detailse kirjelduse defineerimist. Eriti mängib see rolli, kui nõuete täpsustamisel lööb kaasa mitu erinevat gruppi, kes kasutavad erinevat terminoloogiat samas või mitmes valdkonnas [UG96].

Eri kultuurirühma inimestel on erinevad uskumused, veendumused, harjumused ja isegi kasutatav keel. Mitme sellise inimese omavahelisel suhtlemisel on tähtis, et koostegutsemise eesmärk, tegutsemisviisid ja reeglid oleksid paigas ning kõigile üheselt arusaadavad. Vastasel juhul koostöö ei toimi ja siit arenevad edasi tõsisemad probleemid. Sama lugu võib tekkida ka tarkvaratehnikas, kus üks masin ei mõista teist, kuna tarkvara on arendatud erinevate inimeste poolt erinevas kohas erinevaid meetodeid kasutades. Ühise keele kasutamine võimaldaks süsteeme hiljem ka ühendada või panna koostööd tegema.

Süsteemitehnikas kergendavad ühised arusaamad korduvkasutuse, usaldusväärsuse saavutamist ning spetsifikatsioonist tulenevate dokumentatsioonide produtseerimist ning hilisemate ebakõlade vähendamist dokumentatsioonis. Need kolm omadust on ihaldusväärsed tõenäoliselt igale enesest lugupidavale ettevõttele, kes soovib karmis konkurentsis ellu jääda.

Üks viis eelpool kirjeldatud lahenduste saavutamiseks on ontoloogiate kasutuselevõtt ja nende arendamine. Järgnevalt vaatlemegi, mis peitub ontoloogia termini taga, kuidas on seda erinevad inimesed ja grupid defineerinud.

2.3 Mis on ontoloogia?

Levinuimad definitsioonid:

- Ontoloogia on termin filosoofiast ja tähendab „olemise teooriat" [PHILO].
- Ontoloogia on kontseptualisatsiooni ilmutatud spetsifikatsioon [Gru93a].
- Ontoloogia on sõnavara või mõistete teooria tehissüsteemide ehitamiseks [UG96].
- Ontoloogia on terminite ja nende vaheliste seoste ilmne esitus [OWL].

Esimene definitsioon on pärit Kreekast ja tähendabki tõlkes olemise teooriat või õpetust. Täpsemalt on see aga metafüüsika haru, mis käsitab olemise kõige üldisemaid printsiipe, seaduspärasusi ja struktuure. Siia kuuluvad kõik olemist puudutavad küsimused nagu „Mis eksisteerib?", „Mis on?", „Mis ma olen?". Aristotelese pärandiks on kümme kategooriat: olemus, kvantiteet, kvaliteet, suhe (relatsioon), koht (ruum), aeg, asend, omamine, toime (põhjus), talumine (tagajärg). Kategooriate mõte on taandada maailmas valitsev omaduste mitmekesisus üldistele määratlustele, mida on lõplik arv. [PHILO].

Gruber'i [Gru93a] poolt kasutusele võetud teine definitsioon on küllaltki levinud oma lühiduse, kuid tiheda sisu tõttu. Kontseptualisatsioon (ingl. k *conceptualization*) on abstraktne lihtsustatud vaade maailmale, mida soovime esitada mingil põhjusel. Maailmavaade on tihtipeale kujutatud mõistete hulga (näiteks: olemid, atribuudid, protsessid), nende definitsioonide ning nende omavaheliste suhetena. Kusjuures need mõisted, seosed peavad eksisteerima kaudsel või siis ilmutatud kujul.

Kontseptualisatsioon võib olla varjatud eksisteerides vaid kellegi peas või väljendatuna tarkvaratükis. Siiski, enamikes käsitlustes vaadeldakse ontoloogiat ilmutatud seletuse või esitusena kontseptualisatsioonist.

Uscholdi [UG96] käsitluse järgi on ontoloogia termin, mis väljendab ühist arusaamist huvipakkuvas valdkonnas, mida võib kasutada unifitseeritud raamistikuna probleemide lahendamiseks.

Ontoloogia võib võtta erinevaid vorme, aga tingimata sisaldab ta **terminite sõnavara** (ingl. k. *vocabulary of terms*) ja nende **tähenduse täpsustust** (definitsioonid). Formaalsuse aste, kuidas sõnavara tekitatakse ning tähendusi kirjeldatakse, võib varieeruda märgatavalt sõltudes valdkonnast ja vajadusest alates loomuliku keele tasandist lõpetades põhjalikult defineeritud terminite, teoreemide ja tõestustega täielikkusest ja mittevasturääkivustest.

Neljas definitsioon on OWL (*Web Ontology Language*) poolt tõlgendus ontoloogiale. OWL ise kasutab ontoloogiaid sõnastikes terminite tähenduse ja seoste esitamiseks [OWL].

Kokkuvõtvalt täielik üksmeel ontoloogia definitsiooni koha pealt puudub, erinevad grupid võtavad aluseks erinevad definitsioonid vastavalt oma valdkonna juhtivatele ideedele ja küsimustele. Pooleldi üksmeel on vaid eesmärkide suhtes, mis peaks võimaldama paremat maailma defineerimist, efektiivsemat suhtlust nii inimeste kui tarkvarasüsteemide poolt, ressursside taaskasutust. Viimased on eriti levinud just tarkvaratehnikas.

II PEATÜKK: KASUTUSKATEGOORIAD

Antud peatükk kirjeldab põhilisi motivaatoreid ontoloogia kasutamiseks, milledeks on kommunikatsioon (ingl. k. *communication*), koostalitlusvõime (ingl. k. *iteroperabilty*) ja süsteemitehnika (ingl. k. *system engineering*). Need on põhilised valdkonnad, kus ontoloogiaid kasutada ning millede protsesse võiks ontoloogiad parendada.

2.1 Kommunikatsioon

Ontoloogiad vähendavad kontseptuaalset ja terminoloogilist arusaamatust pakkudes organisatsioonis unifitseeritud raamistikku kommunikatsiooniks. Sel viisil võimaldavad ontoloogiad jagatud arusaamist ja kommunikatsiooni erinevate vajaduste ja nägemustega inimeste vahel. Inimestevahelise kommunikatsiooni jaoks on piisav kui luua ühemõtteline kuid mitteformaalne ontoloogia. Järgnevalt hindame ontoloogia kasutuse erinevaid külgi, mis hõlbustavad inimestevahelist suhtlemist organisatsioonis.

Normatiivmudelid. Integreeritud laiaulatuslikus tarkvarasüsteemis peavad erinevad inimesed omama ühist arusaamist süsteemist ja selle eesmärke. Kasutades ontoloogiat võime koostada süsteemi normatiivse mudeli. See annab süsteemile semantika ja laiendatava mudeli, mida on võimalik hiljem viimistleda, ning mis võimaldab semantilisi transformatsioone erinevate kontekstide vahel.

Suhetevõrk. Suhetevõrkude loomisega ontoloogiate abil saame end hoida kursis mis on omavahel seotud ning uurida ja navigeerida selles võrgustikus. Näiteks, inimestel on erinevad arusaamised ja oletused vaadeldavale süsteemile ning kuidas osad on nende vahel seotud. Ontoloogia tuvastab loogilised ühendused elementide vahel üle süsteemi mudelite.

Terviklikkus ja mitmetähenduslikkuse puudus. Ontoloogia pakub ühemõttelisi definitsioone tarkvarasüsteemis kasutatavatele terminitele.

Erinevate vaatenurkade ühendamine. Kui meil on mitme tegijaga ühenduses olev süsteem, siis integreerimine läbi ühise arusaamise saab eluliselt tähtsaks. Näiteks,

erinevatel positsioonidel olevad inimesed omavad erinevat vaadet, mida organisatsioon teeb, milliseid eesmärke püüab saavutada ning kuidas need eesmärgid on saavutatavad. Kasutades ontoloogiat süsteemi normatiivmudeli pakkumiseks, on ühendamine saavutatav abistades osalejaid omavahel suhtlemises ja ühisarusaamisele saamisel.

Erinevate vaatenurkade ühendamine on aluseks standardite väljatöötamisele kommuunis. Võttes omaks ühise ontoloogia, kasutavad kõik osalejad standardiseeritud terminoloogiat kõikide objektide ja suhete kohta oma valdkondades. [UG96]

Ühtse arusaamise jagamiseks informatsiooni struktuurist inimeste ja tarkvara agentide vahel on üks põhilisi eesmärke ontoloogiate välja töötamisel. Näiteks, oletame, et mitu erinevad veebilehte sisaldavad meditsiinilist informatsiooni. Kui need lehed jagavad ja avaldavad sama terminite ontoloogiat, mida kasutavad alusena, siis arvuti agendid saavad nendelt erinevatelt lehtedelt koguda ja ühendada informatsiooni. Agendid võivad kasutada ühendatud informatsiooni kasutaja küsimustele vastamiseks või sisendandmeteks teistele rakendusprogrammidele.

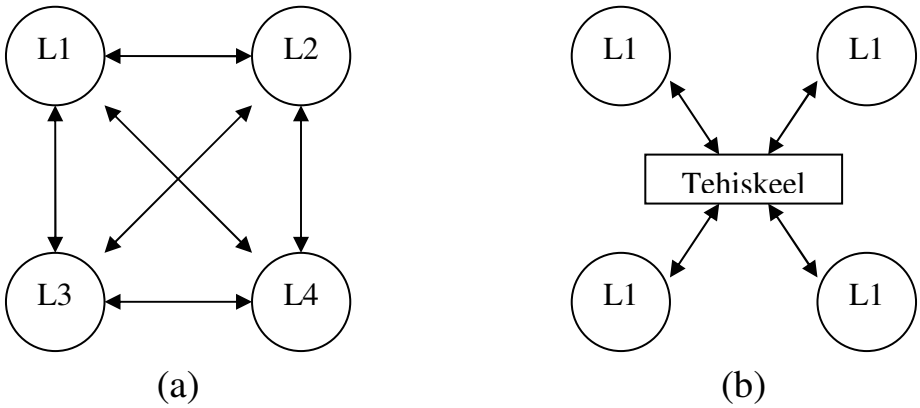
2.2 Koostalitlusvõime

Sama organisatsiooni eraldiseisvad üksused kalduvad arendama isiklikke keeli sõltudes üksuse kultuurist, kogemustest ja eesmärkidest. Erinevatel kasutajatel on vajadus vahetada andmeid ning selleks võidakse kasutada erinevaid tarkvara vahendeid. Sellest tulenevalt on tähtis ontoloogiarakenduste koostalitlusvõime – suhtlus teiste süsteemidega. Siin kasutatakse ontoloogiat kui vahetusformaati.

2.2.1 Ontoloogia kui tehiskeel

Iga äriprotsessi ümberkorraldamise IT keskkond või mitmetegija süsteemid peaksid kasutama ühendatud ärimudeleid, kus on ühendatud tegevused, ressursid, organisatsioon, eesmärgid, tooted ja teenused. Need ühendatud ettevõtte mudelid on kui ühine hoidla (ingl. k. *repository*), mis on ligipääsetav mitmetele tööriistadele.

Lisaks võib tehiskeel ühendada olemasolevate andmete hoidlaid, kas siis standardiseerides terminoloogiat hoidlate erinevate kasutajate seas või pakkudes semantilise baasi translaatoritele erinevate kasutajate seas.



Joonis 2.1. Ontoloogia kui tehiskeel

Ontoloogiaid on võimalik kasutada toetamaks tõlkimist erinevate keelte ja esituste vahel. Üks võimalus on kavandada unikaalsed translaatorid iga kahe osapoole vahel. Aga see nõuaks $O(n^2)$ translaatorit n erineva ontoloogia kohta (Joonis 2.1a). Kasutades ontoloogiat tehiskeelena tõlketoetuseks, võime redutseerida translaatorite arvu $O(n)$ peale n erineva ontoloogia puhul, kuna vajalikud on ainult translaatorid oma ontoloogia ja vahetus ontoloogia vahel (Joonis 2.1b). Sellist lähenemist kasutab *Process Specification Language* [PSL].

2.2.2 Koostalitlusvõime mõõtmed

Lisaks tööriistadele ja hoidlatele, on võimalik välja tuua mitmed tunnusjooned seoses koostalitlusvõimega. Esiteks, on meil vaja võtta arvesse kasutajate vaheliste seoste loomust, kes jagavad tööriistu ja andmeid. On eluliselt tähtis, et erinevate tegijate ja organisatsioonide poolt kasutatavad ontoloogiad ja tööriistad sama ettevõtte sees oleksid ühiselt tarvitatavad ja taaskasutatavad läbi mitme organisatsiooni. Järgnevalt on kirjeldatud koostalitlusvõime eristatavad mõõtmed.

Seesmine koostalitlusvõime. Kõik süsteemid, mis nõuavad omavahel opereerimist on mingi organisatsioonilise üksuse vahetu kontrolli all. Erinevused eksisteerivad

ajaloolistel põhjustel ning pärandsüsteemid (ingl. k. *legacy system*), mida rohkem ei muudeta, tuleb integreerida.

Väline koostalitlusvõime. Organisatsiooniline üksus soovib eraldada ennast väljastpoolt tulevatest muudatustest. Väline võib siinkohal tähendada teist osakonda samas organisatsioonis.

Ühendatud ontoloogiad valdkondade vahel. Erinevate valdkondade ontoloogiate ühendamine mingi töö toetamiseks võib olla vajalik. Näiteks ontoloogia mis toetab töövoo juhtimissüsteeme peab ühendama protsesside, ressursside, toodete, teenuste ja organisatsiooni ontoloogiad. Töövoo vahendite hulk kasutab siis seda hulka integreeritud ontoloogiatest.

Tööriistade vahel ontoloogiate ühendamine. Sama valdkonna erinevate ontoloogiate ühendamine pärandsüsteemides võib olla vajalik. Näiteks, erinevad tööriistad võivad kasutada erinevaid protsessi ontoloogiaid. Koostalitlusvõime saavutamiseks on vajalik kasutada ühist ontoloogiat, mida mõlemad tööriistade komplektid oskavad kasutada [UG96].

2.3 Süsteemitehnika

Koostalitlusvõime ja kommunikatsioon mängivad põhilist rolli süsteemis opereerimisel. Nüüd käsitleme motivaatoreid, mis toetavad tarkvara süsteemi kavandamises ja arenduses endas.

2.3.1 Korduvkasutus

Ontoloogiad peavad olema taaskasutatavad, et olla efektiivsed, nii et oleks võimalik importida ja eksportida mooduleid erinevate tarkvarasüsteemide vahel. Probleem ilmneb kui tarkvara vahendeid rakendatakse uutele valdkondadele, kus nad ei pruugi toimida soovitult, kuna nad tuginesid algupärase rakenduse eeldustele mitte uutele. Valdkonna klasside ja tööde kirjeldamisega nendes valdkondades pakuvad ontoloogiad raamistikku

otsustamaks millised ontoloogia küljed on taaskasutatavad erinevate valdkondade ja tööde vahel.

Et ontoloogiatest oleks kasu peavad nad olema veel vajadustele kohaldatavad. Lisaks peaksid olema ontoloogiad laiendatavad, et ühendada endasse veel uusi piiranguklasse ja mõistete, piirangute spetsialiseerumist konkreetsele probleemile.

Kui keegi on töötanud välja detailse ontoloogia, siis saavad teised lihtsalt kasutada seda samas valdkonnas või laiendada seda veidi vastavalt oma vajadustele. Kui on vaja arendada mahukat ontoloogiat, on võimalik integreerida mitu olemasolevat ontoloogiat kirjeldades ära suurema valdkonna jaotused.

2.3.2 Usaldusväärsus

Mitteformaalsed ontoloogiad võivad parandada tarkvarasüsteemide usaldusväärsust olles baasiks käsitsikontrollimiseks spetsifikatsiooni vastavuses disainile. Formaalsete ontoloogiate kasutamine võimaldab pool või täis automaatset tarkvarasüsteemi terviklikkuse kontrolli vastavalt deklaratiivsele spetsifikatsioonile.

2.3.3 Spetsifikatsioon

Mitteformaalselt lähenedes aitavad ontoloogiad kaasa süsteemi nõuete tuvastamise protsessile ja mõista süsteemi komponentide vahelisi seoseid. See on iseäranis tähtis süsteemides, kus on kaasatud laiali jagatud projekteerijate meeskonnad erinevatel aladel.

Formaalselt lähenedes pakuvad ontoloogiad tarkvara süsteemi deklaratiivset spetsifikatsiooni, mis võimaldab mõista, milleks süsteem on kavandatud, selle asemel et näidata, kuidas süsteem toetab funktsionaalsust [UG96].

Terminite deklaratiivse spetsifikatsiooni alusel on võimalik ontoloogia abil valdkonna teadmiste analüüs. Formaalne terminite analüüs on äärmiselt hinnaline püüdes taaskasutada olemasolevaid ontoloogiaid ning ontoloogiate laiendamisel [M02].

Tehes ilmseks valdkonna eeldused implementatsiooni alusena, võimaldab see neid eeldusi kergelt muuta, kui teadmised valdkonnast muutuvad. Püsiprogrammeeritud (ingl. k. *hard-coded*) eeldused maailmast programmeerimiskeele koodis muudavad raskesti leitavaks, mõistetavaks ning ka raskesti muudetavaks need eeldused. Sõnaselge spetsifikatsioon valdkonna teadmistest on kasulik ka uutele kasutajatele, kes peavad õppima valdkonna terminite tähendusi.

2.3.4 Muud kasud

Lisaks kolmele eelmisel süsteemitehnika kasulikkuse teguritele kuuluvad siia veel **otsing**, **hooldus** ja **teadmuse omandamine**. Otsingu seisukohalt saab ontoloogiat kasutada kui metaandmeid, mis on kui indeksid informatsioonihoidlale. Süsteemiarenduses või lõpprakenduse osana võib ontoloogiate kasutamine lihtsustada hooldust mitmel viisil. Süsteemid, mis on ehitatud kasutades ilmutatud ontoloogiaid (mitte kellegi peas või mõtteis paiknevad), parandavad tarkvara dokumentatsiooni, mis omakorda vähendab hoolduskulusid. Lisaks on võimalik teda hooldada ühest kohast, kui ontoloogiat kasutatakse neutraalse autoriseeriva keelena, millel on mitu sihtkeelt (keelt kuhu tõlkida). Kiirust ja usaldusväärsust on võimalik parendada kasutades olemasolevat ontoloogiat lähtepunktina teadmuse omandamise juhtimiseks teadmuspõhiste süsteemide ehitamisel.

Kokkuvõtvalt ontoloogiate kasutuselevõtt aitab kaasa nii inimeste vahelise, masinate vahelise kui ka inimeste ja masinate vahelisele suhtlusele tagades ühemõttelisuse, täpsustatud sõnavara ning suhtlemisstandardid. Nendest tulenevalt on lihtsam ka iga järgmine protsess, sest eeltöö on tehtud. Tarkvarasüsteemide abil on võimalik kergemini infot leida huvipakkuva valdkonna sõnavaraga tutvumiseks ja täpsustamiseks, lisaks uute terminite ja definitsioonide puhul lisada neid sinna. Uute süsteemide lisandumisel organisatsiooni saab taaskasutada olemasolevat ontoloogiat uute süsteemide sobitamisel organisatsiooni.

III PEATÜKK: KONTSEPTSIOONID

Käesolev peatükk jaotab ontoloogiad kaheks, kirjeldab seoseid mõistete vahel ning annab ülevaate levinumatest struktuuridest.

3.1 Üldine jaotus

Ontoloogiaid saab jagada üldiselt kaheks: üldised ja valdkonnapõhised. Üldised ontoloogiad on ühised mitme valdkonna vahel. Neid võib jagada veel kõrgtaseme ontoloogiateks, ühikuteks, dimensioonideks. Valdkonnapõhised on loodud aga pigem kindla valdkonna jaoks nagu meditsiin, kunstiajalugu jne.

3.1.1 Üldised ontoloogiad

Üldised ontoloogiad (nimetatakse ka kõrgtaseme ontoloogiateks) on baasiks, millele ehitatakse valdkonna ontoloogiad. Kirjeldavad väga üldisi mõisteid nagu ruum, aeg, põhjus, objekt, sündmus, tegevus jne. Nad on sõltumatud konkreetsest probleemist või valdkonnast. Üldisteks ontoloogiateks nimetatakse ka igasuguseid ühikuid ja dimensioone. Nende potentsiaal korduvkasutuses on kõige suurem, kuna ideeliselt ei tohiks neid peale välja arendamist ja kasutuselevõttu muuta. Levinumatest üldistest ontoloogiatest võib välja tuua järgmised: CYC, SUMO, Microkosmos, SENSUS, GUM, DOLCE.

3.1.2 Valdkonna ontoloogiad

Valdkonna ontoloogiad kirjeldavad mingi üldise valdkonnaga seotud sõnavara kohandades kõrgtaseme ontoloogia terminid endale sobivaks. Näiteks:

- Meditsiin: UMLS, SNOMED, Galen
- Kunstiajalugu: AAT, ULAN

Valdkonna ontoloogiad on rohkem dünaamilisemad kui üldised ontoloogiad. Nad võivad omada tähendust ainult teatud valdkonnas, aga võib siduda ka teistega.

3.2 Levinuimad seosed

Ontoloogias olevate mõistete vahel võivad olla seosed, mis üldiselt jagunevad kaheks: taksonoomiad, assotsiatiivsed seosed. Taksonoomiad organiseerivad mõisted alam- ja ülemmõistete puu struktuuridesse. Üks harilikum vorm on „on" (ingl. k. *is-a*) seos (spetsialiseerimine). Näiteks elevant on imetaja, kus imetaja on üldisem mõiste ja elevant on üks imetajatest. Teine levinud vorm on „osa" (ingl. k. *part-of*) seos (osastav). Näiteks lont on osa elevandist, kus lont on üks osa elevandist. Lihtsates ontoloogiates on levinud just taksonoomiad ning enamasti ka üksikult nii, et samas ontoloogias ei ole nii „on" kui „osa" seost. Ühe kindla seose kasutamine üle puu lihtsustab ontoloogiat kasutaval süsteemil selle peal operatsioone teha, samas kui iga erineva seose lisamine muudab seda keerulisemaks.

Assotsiatiivsed seosed on seosed, mis ühendavad mõisteid üle kogu puu struktuuri, seoses ei ole ainult ülem ja alluv. Nendeks võivad olla näiteks nimetav ja lokatiivne, kus nimetav kirjeldab mõiste nimesid ning lokatiivne kirjeldab ühe mõiste asetust teise suhtes. Lisaks võivad seosed esindada funktsioone või protsesse, mida mõiste omab või millesse ta on kaasatud.

3.3 Levinuimad struktuurid

Teine viis jagada ontoloogiaid on nende struktuuri järgi. Levinuimad struktuurid on puukujuline ja suunatud atsükliline graaf (ingl k. *directed acyclic graph*). Siinkohal kirjeldan puustruktuuriga RHK-10 ning suunatud atsüklilise graafiga geeniontoloogiat.

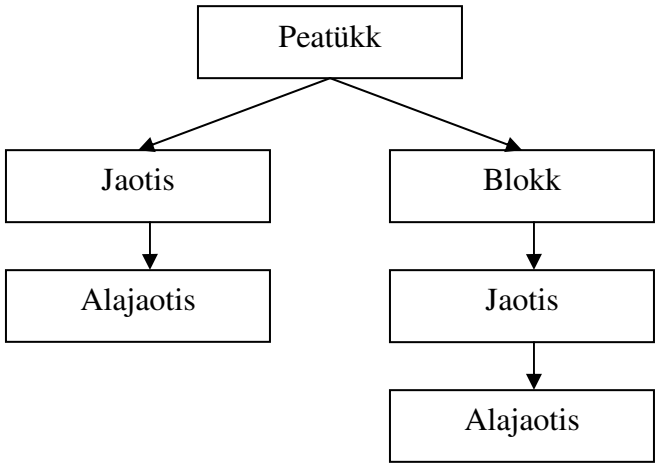
3.3.1 RHK-10

RHK-10 e. ICD-10 (ingl. k. *International Classification of Diseases*) on rahvusvaheline haiguste ja nendega seotud terviseprobleemide statistiline klassifikatsioon. Selle eesmärgiks on võimaldada eri riikides eri aegadel kogutud suremuse ja haigestumuse andmete süstemaatilist registreerimist, analüüsimist, interpreteerimist ja võrdlemist. Praktikas on RHK kujunenud rahvusvaheliseks standardseks diagnostiliseks klassifikatsiooniks kõikidel üldepidemioloogilistel ja mitmel tervishoiu juhtimisalasel

otstarbel. Haiguste diagnooside ja muude terviseprobleemide ülekandmiseks sõnadest kasutatakse tärkkoodi, mis võimaldab andmete hõlpsat säilitamist, otsingut ja analüüsi. RHK-d on võimalik kasutada andmete klassifitseerimiseks, mis mitmesugustel tervist kajastavatel dokumentidel leiduvad selliste pealdiste all nagu „diagnoos", „suunamispõhjus", „ravitud seisundid" ja „konsultatsiooni põhjus" ning mis on statistika ja muu tervislikku olukorda puudutava teabe aluseks. Põhi-RHK on kolmekohaliste koodidega üksikjaotiste loetelu, millest iga jaotist võib edasi jaotada kuni kümnesse neljakohalise koodiga alajaotisesse.

Struktuur

RHK-10 on selgelt kujutatav puukujulise hierarhiana. Järgneval joonisel on näidatud ontoloogia võimalik puukujuline klasside struktuur.



Joonis 1. Põhi-RHK-10 struktuur

Igal klassil on teatud kodeerimisjuhend ning reeglid. Jaotiste defineerimiseks on kolmekohaline kood, kus esimene sümbol kuulub ladina tähestikku vahemikus A..Z (välja arvatud täht 'U', mis on jäetud tuleviku laiendusteks), millele järgneb 2 numbrit (näiteks: F31). Enamik jaotisi jagatakse edasi alajaotisteks (täpsustavad haigust või seisundit). Neid identifitseeritakse lisades jaotise koodile juurde punkti ja veel ühe numbri (näiteks: F31.5). Peatükk sisaldab jaotisi või blokke, mis omakorda sisaldavad jaotisi. See tähendab seda, et peatükil võib olla üksikuid jaotisi, kuid mõnikord

koondatakse jaotised blokiks (näiteks: peatüki kood A00-B99; bloki kood A00-A09). Klasside omadustena võib ära nimetada eestikeelset ja ladinakeelset nimetust [Kün96].

Järgnevalt tuuakse näide RHK-10 klassifikatsiooni hierarhilisest asetusest. Hierarhia on kujutatud taanetega s.t. suurema taandega objektid alluvad esimesele eelnevale väiksema taandega objektile.

Näide 1.

Peatükk II: C00-D48, Nimetus: Kasvajad
    Blokk: C45-C49, Nimetus: Mesoteelkoe ja pehmete kudede pahaloomulised kasvajad
    Jaotis: C50, Nimetus: Rinna pahaloomuline kasvaja
    Blokk: C51-C58, Nimetus: Naissuguelundite pahaloomulised kasvajad
Peatükk V: F00-F99, Nimetus: Psüühika- ja käitumishäired
    Blokk: F20-F29, Nimetus: Skisofreenia, skisotüüpsed ja luululised häired
    Blokk: F30-F39, Nimetus: Meeleoluhäired
        Jaotis: F30, Nimetus: Maniakaalne episood e. mania
            Alamjaotis: F30.0, Nimetus: Hüpomania

RHK-10 näol on tegemist range alamklassi hierarhiaga. See tähendab, et kui A on B ülemklass ning meil on klassi B isendist objekt, siis sellest tingimata järeldub, et objekt on ka klassi A isend.

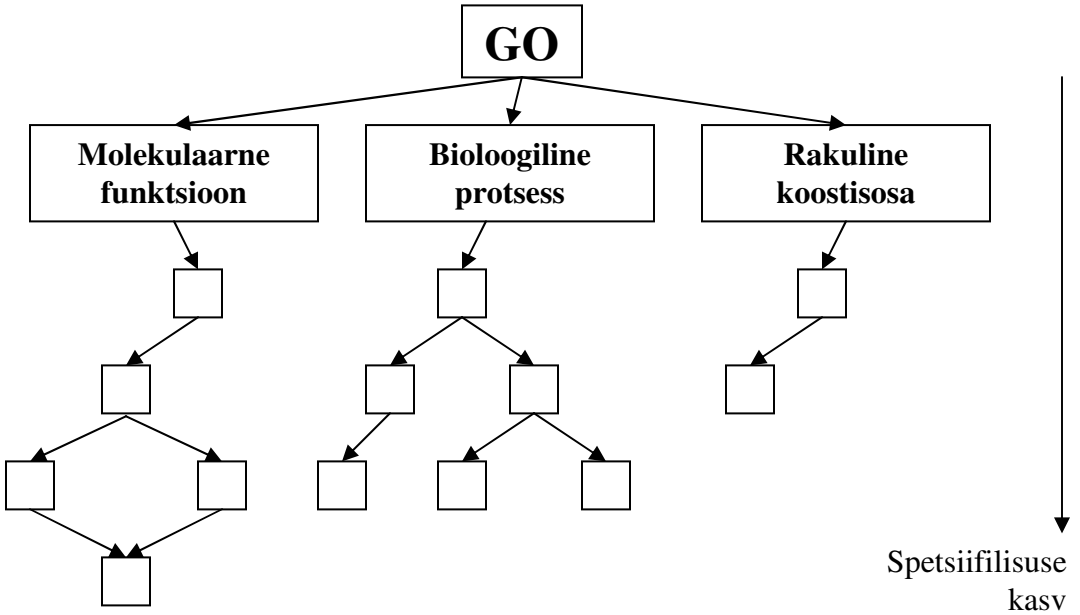
3.3.2 Geeni Ontoloogia

Geeni Ontoloogia (GO) projekti eesmärk on toota struktureeritud, täpselt defineeritud, üldine ja dünaamiliselt juhitav sõnavara geeni produktide (valgud, RNA jne.) rollide kirjeldamiseks kõigis organismides. GO pakub kolm defineeritud terminite struktureeritud võrgustikku kirjeldamaks geeniproduktide tunnuseid. Organiseerimise põhimõtted on: 1) molekulaarne funktsioon (ingl. k. *molecular function*), 2) bioloogiline protsess (ingl.k. *biological process*) ja 3) rakuline koostisosa (ingl. k. *cellular component*). Bioloogilise protsessi ontoloogia tegeleb bioloogiliste eesmärkidega millele geen või geeniprodukt kaasa aitab. Protsess teostatakse läbi ühe või mitme järjestatud molekulaarse funktsiooni. Molekulaarne funktsioon tegeleb geeniprodukti biokeemiliste tegevustega. See kirjeldab mis on tehtud ilma täpsustamata kus või kuna sündmus aset leidis. Rakulise koostisosa ontoloogia kirjeldab kohad, kus geeniprodukt võib olla

aktiivne. Geeniproduktil on üks või mitu molekulaarset funktsiooni, seega võib ta osaleda ühes või rohkemas bioloogilises protsessis ning võib paikneda ühes või mitmes rakulises komponendis.

Struktuur

GO terminid on organiseeritud struktuuridesse, mida nimetatakse suunatud atsüklilisteks graafideks. Erinevus hierarhiatest seisneb selles, et alam võib omada mitut ülemat. Näiteks bioloogilise protsessi termin "heksoosi biosüntees" omab kahte ülemat: "heksoosi metabolism" ja "monosahhariidi biosüntees", kuna biosüntees on metabolismi alamtüüp ning heksoos monosahhariidi tüüp. Kui mistahes heksoosi biosünteesis kaasa lööv geen on termini lühikirjelduses, on ta ka automaatselt kirjeldatud ka nii heksoosi metabolismis kui ka monosahhariidi biosünteesis. Iga GO termin peab lisaks alluma "tõese tee reeglile": kui laps termin kirjeldab geeni produkti, siis kõik tema vanemterminid peavad samuti rakenduma sellele geeniproduktile. Järgmisel joonisel on visuaalsena kujutatud GO terminite struktuur [GO].



Joonis 2. GO struktuur

Joonisel 2 on näha GO jagunemine kolmeks. Kastikesed põhimõistete all on esitatud juhuslikult näitlikustamaks võimalikku terminite paiknemist ning seoseid. Iga termin sisaldab endas termini nime ning 7 kohalist unikaalset numberidentifikaatorit. Seda kasutatakse vaid andmebaasi ristviitena andmebaaside koostööks. Lisaks on võimalik defineerida termini juurde veel selle täpne definitsioon, kommentaar, sünonüüm ning andmebaasi ristviide.

Kokkuvõtvalt võib jagada ontoloogiad üldisteks ontoloogiateks ja valdkonnapõhisteks ontoloogiateks. Struktuurilt kasutatakse tavalist hierarhilist puu struktuuri ja suunatud atsüklilist graafi struktuuri. Seostest kasutatakse taksonoomiaid ja assotsiatiivseid seoseid.

IV PEATÜKK: KEEL JA KESKKOND

Varasemalt kirjeldasime mis on ontoloogiad, milleks neid kasutada ning millised nad välja näha võiksid. Selleks, et ontoloogiaid teha, on meil vaja keelt ja lihtsustamise mõttes ka arendamiskeskkonda. Ontoloogiate väljatöötamiseks on loodud palju tööriistu, mille abil on võimalik läbida enamik ontoloogia arenduseks vajalikud etapid (kirjeldatud järgmises peatükis). Tööriistad võivad aidata lisaks võrrelda ja linkida ontoloogiaid, võrrelda neid, valideerida, ühendada ning muuta teistsugusele kujule. Kuna päris universaalset tööriista pole olemas siis võib tihtipeale vaja minna mitme vahendi kasutamist.

4.1 Esituskeel

Ontoloogia peab olema kodeeritud mingis keeles. Mida lihtsam on ontoloogia, seda lihtsamalt võib teda ka esitada ning keele valikuga seonduvaid küsimusi tekib vähem. Mida keerulisemat ontoloogiat soovime luua, seda rohkem tähelepanu tuleb pöörata väljendusrikkuse võimsusele ning arutluskeele valikule. Selge on, et keel peab olema võimeline väljendama valitud valdkonna mõisteid. Keele valikut võib raskendada nende rohkus, nii eraotstarbel kui ka standarditel põhinevaid. Ontoloogia keel on formaalne keel, mille läbi ehitatakse ontoloogiaid [GRO]. Järgnevalt on toodud ära levinumate keelte üldised kirjeldused:

Knowledge Interchange Format (KIF)

Kavandatud kasutamiseks ühitamatute arvutiprogrammide vahel teadmuse vahetamiseks, mis võivad olla tehtud erinevate programmeerijate poolt, erineval ajal, erinevas keeles jne. Järgmisel kategoorilised tunnused on omased KIF kavandusele:

- Keelel on deklaratiivne semantika. Esituse avaldiste tähendust on võimalik mõista ilma, et peaks abi paluma interpretaatorilt nendega töötlemiseks.
- Keel on loogiliselt kõikehõlmav. Hoolitseb suvaliste loogiliste lausete avaldiste eest.
- Keel hoolitseb teadmuse teadmusesituse eest. See võimaldab kasutajal teadmusesituse otsuseid ilmselt sõnaselgelt ja võimaldab kasutajal sisestada uusi teadmusesituse konstruktsioone keelt muutmata.[KIF]

Web Ontology Language (OWL)

OWL on W3C poolt üheks semantilise veebi standardiks kuulutatud keel struktureeritud veebipõhiste ontoloogiate loomiseks. OWL kasutab mõlemaid, URI'sid (*Uniform Resource Identifier*) nimetamiseks ja kirjeldusraamistikku veebi jaoks RDF abil, ontoloogiatele järgmiste võimete lisamiseks:

- võime olla jagatud mitme süsteemi vahel,
- koormustaluvus veebi nõuetele,
- juurdepääsu ja rahvusvahelisuse ühilduvuse tagamine veebi standardite abil,
- avatus ja laiendatavus.

OWL ehitub RDF ja RDF skeemile ja lisab rohkem sõnavara klasside ja omaduste kirjeldamiseks, muuhulgas veel klassidevahelised seosed, kardinaalsus (näiteks: täpselt üks), võrdsus, rikkam omaduste kirjeldus, omaduste tunnused (näiteks: sümmeetria).

OWL on ajalooliselt seotud selliste keeltega nagu OIL (*Ontology Inference Layer*) ja DAML (*DARPA Agent Markup Language,* vahel ka DAML-ONT). OIL oli arendatav euroopa teadlaste poolt ning DAML-ONT ameerika teadlaste poolt. Jõupingutuste ühendamisel tekkis DAML+OIL keel, mida sooviti esitada standardina. Selle tulemusena moodustati uus uurimisrühm, mis pidi välja töötama W3C (*World Wide Web Consortium*) standardi DAML+OIL põhjal. 10. veebruaril 2004 kuulutaski W3C OWL'i soovituslikuks standardiks [OWL].

Keele valik peaks sõltuma just vajadustest ja nõuetest, mida ontoloogia peab suutma väljendada. Pole mõtet valida sellist keelt, millega saab enamikke maailma asju väljendada, aga meie konkreetse lahenduse jaoks kasutame vaid väikest osa nendest võimalustest. Selline valik on muidugi õigustatud, kui me näeme tõesti, et tulevikus võib vajadus tekkida.

4.2 Keskkonna valikukriteeriumid

Enne ontoloogia väljatöötamist peab kindlasti kaaluma erinevate arendusvahendite kasutamise vahel vastavalt käesolevatele ning ka tulevikus tekkivatele vajadustele. Tihtipeale on ühe ontoloogia väljatöötajaid mitu, seetõttu on neil vaja näha üksteise töökeskkonda. See on saavutatav sessioonidega. Koostöö nõuab kokkulangevuskontrolli, lukustust, versioonimist ja juurdepääsu süsteeme.

Erinevate rakenduste ühendamisel keerulisemate platvormidega muutub üha tähtsamaks, et erinevad keskkonnad võimaldaksid lugeda ja kirjutada ühilduvaid formaate ning ka ühildada mitme teise riistvara/tarkvara keskkonnaga. Üheks lahenduseks võib pidada Java-keele põhiseid rakendusi. Teised süsteemid aga, mis toetavad erinevaid sisend ja väljund formaate, saavad aru üldistest standarditest ning pakuvad tõlke ja vastendamist, võivad aidata.

McGuinness [M02] on välja toonud põhilised argumendid-tunnused, mida peaks kaaluma ontoloogia arenduskeskkonna valikul:

- **Koormustaluvus**. Selgeks tuleb teha, kui suuri ontoloogiaid luua ja hallata soovitakse ja ka samaaegsete kasutajate arv. Koormuseks siinkohal ongi ontoloogia suurus ja kasutajate arv.
- **Versioonimine**. Rakendused võivad paikneda erinevates keskkondades, ka erinevatel kontinentidel, seetõttu tuleb kuidagi hallata ontoloogia versioone. Tüüpiliselt on tarkvara arenduskeskkondades kasutusel näiteks eraldi versioonikontrolli süsteem nagu CVS või Subversion.
- **Turvalisus**. Erinevad rakendused vajavad erinevaid õigusi ontoloogiatele ligipääsuks. Seetõttu on vajalik keskkond, mis muudab mingid ontoloogia osad nähtavaks mingi turvamudeli alusel.
- **Analüüs.** Ontoloogia ei ole kohe alguses valmis, mingil hetkel ei ole ta veel terviklik ning võib sisaldada vigasid. Siis on hea kui analüüsivahendid suunavad tähelepanu teatud kohtadele, mis vajavad muudatusi. Erinevad diagnostika testid võivad aidata kasutajatel leida probleeme ja vigu.
- **Elutsükli küsimused**. Aja jooksul kasvavad ontoloogiad suuremaks ning ka siis, mitme aasta pärast, tuleb hallata neid. Süsteemi ühendatakse juurde uusi ontoloogiaid. Tuleks kaaluda selliste võimaluste peale nagu terminite ühendamine, termini jagamine mitmeks, mitmesed nimeruumid, lähtekoodi kontrolli süsteemid jne.
- **Kasutusmugavus**. Isegi kui keskkonnal on olemas kogu funktsionaalsus mida vaja, võib see jääda kasutamata, kui kasutaja ei oska süsteemi erinevaid osi kasutada.

Seetõttu peaksid igasugused kasutusjuhendid, käsiraamatud, graafilised sirvimisvahendid olemas olema.

- **Mitmekülgne kasutajatugi**. Mõned keskkonnad on mõeldud kogenud, teised kogenematutele kasutajatele ning mõned süsteemid võimaldavad kasutajatel keskkonda kujundada vastavalt kasutajatüübile. Vaja on teha kindlaks, kas keskkond on kasutatav kõigile ettenähtud tüüpi kasutajatele.
- **Esitlusstiil**. Tihedalt seotud eelmise punktiga. Mõned kasutajad vajavad näha ulatuslikul määral detaile, mõned kärbitud informatsiooni ja mõned üldse abstraktsioone. Informatsiooni esitus võib olla tekstiline, graafiline või muu tajutav viis. Kuna ükski keskkond ei pea toetama kõiki esitlusstiile, siis on tähtis, et oleks võimalik keskkonda laiendada lisades uusi esitlusmeetodeid vajadusel.
- **Laiendatavus**. Võimatu on ette näha rakenduse kõiki nõudmisi, mis võivad teatud aja pärast tekkida. Seetõttu on tähtis, et oleks võimalik kohandada keskkonda vastavalt kasutajate ning projektivajadustele. Üheks võimaluseks on pistikprogrammide (ingl. k. *plug-in*) tugi e. võimalus lisada põhiprogrammile lisaväärtust andvaid komponente.

4.3 Arenduskeskkonnad

Järgnevalt kirjeldatakse ontoloogiate väljatöötamise vahendeid Protégé, DAG-Edit ning Chimaera. Erinevaid ontoloogiaredaktoreid on siiski tekkinud väga suurel hulgal (üle 90'e) ning nendest üldise tabelkujul ülevaate leiab viidete [D02] ja [D04] alt. Siinsel käsitlusel vaadeldakse tööriistade omadusi vastavalt selle peatüki 4.1 alampeatükile.

4.3.1 Protégé

Tegemist vaba avatud lähtekoodiga programmeerimiskeeles Java tehtud ontoloogia redaktoriga ja teadmusbaasi raamistikuga [Protégé]. Toetatud on ta tugeva arendajate ning akadeemilise, valitsus ja korporatiivse kommuuni poolt, kes kasutavad Protégé'i mitmesugustel aladel nagu biomeditsiin, korporatiivseks modelleerimiseks, teadmiste kogumiseks.

Tarkvara on mõeldud ontoloogiate loomiseks, muutmiseks ja sirvimiseks. Olemas on palju pistikprogramme põhiprogrammi funktsionaalsuse laiendamiseks. Teadmuse mudel on freimi põhine. Freimi mõiste on pärit intellektitehnikast ning tähendab andmekeskset teadmuse esitust, mis seostab objekti mingi tunnusomaduste kogumiga, kusjuures iga selline omadus salvestatakse teatud organiseeritud lahtris. Ontoloogia koosneb klassidest, slottidest, fassettidest ja aksioomidest. Klassid on valdkonna mõisted. Slotid kirjeldavad klassi omadusi ja tunnusi. Fassettid kirjeldavad sloti omadusi ning aksioomid defineerivad lisakitsendused.

Funktsionaalsuse poolelt on võimalik klasse, tunnuseid ja isendeid luua, lisada, vaadata ning otsida. Superklasse saab lisada, kustutada või asendada. Võimalik on ontoloogial päringuid teostada. Tööriistal on väga hea tugi olemas abide näol (kahjuks dokumentidena ainult veebis). Protégé funktsionaalsust saab laiendada pistikprogrammide abil, mida on juba hulganisti ka välja töötatud. Üks häid külgi on veel eri formaadis ontoloogiate import ning ka eksport. Näiteks on võimalik importida teksti failidest, andmebaasi tabelitest ning ka standardiks saanud RDF failidest. Salvestada saab teksti failidesse, JDBC andmebaasi ning RDF formaati. Pistikprogrammide abil saab nii import kui eksport võimalusi laiendada.

Protégé eelisteks oma konkurentide ees on intuitiivne ja lihtsalt kasutatav graafiline kasutajaliides, skaleeritavus (andmebaas laeb freimid nõudmisel ning kasutab vahemällusalvestust mälu vabastamiseks vajadusel) ja laiendatav pistikuline arhitektuur (pistikprogrammide lisamine põhiprogrammi koosseisu).

4.3.2 DAG-Edit/OBO-Edit

Tegemist on vabalt laetava lahtise lähtekoodiga Javas tehtud tarkvaraga [OBOEDIT]. OBO-Edit (*Open Biomedical Ontologies*) on uus versioon DAG-Edit'st (*Directed Acyclic Graph*) ning on hetkel beeta seisuses (edaspidi nimetame tööriista OBO-Edit nimega). Tööriist pakub graafilist kasutajaliidest GO (geeniontoloogia) või teiste suunatud atsükliliste ontoloogia failide sirvimiseks, otsinguks ja muutmiseks. Põhirõhk ongi GO

sirvimisel ja toimetamisel. Programmi kontseptuaalne mudel on suunatud atsükliline graaf.

Funktsionaalsuselt on võimalik mõistete tekitamine ja kustutamine, sünonüümide lisamine, andmebaasi viidete lisamine ning mõistete ühendamine. Abi on olemas väga kesisel moel, siiski mõnel määral võib leida materjale veebist. Lisaks on kogu OBO-Edit programmiga kaasasolev dokumentatsioon installatsioonipaketis esitatud veel DAG-Edit nime all. Kuna OBO-Edit on mõeldud eelkõige GO ontoloogiatega töötamiseks, siis on võimalik laadida ontoloogiaid GO Flat failist, GO MySQL ja Postgres andmebaasist, GO RDF Flat ning OBO formaat failist. Samadesse formaatidesse saab ka salvestada. Hea omadusena võib nimetada veel kasutaja defineeritud pistikprogrammide toetust, millega on laiendada lugemise, salvestamise, importimise ja eksportimise võimalusi.

4.3.3 Chimaera

Tegemist on tarkvarasüsteemiga, millega on võimalik luua ja hooldada veebis hajutatult asetsevaid ontoloogiaid [Chimaera]. Kaks peamist toetatud funktsiooni on mitme ontoloogia ühendamine ning ühe või mitme ontoloogia hindamine. Efektiivseks töötamiseks nõuab keskkond siiski suhteliselt kiiret võrguühendust. Lehti kuvatakse HTML formaadis, erinevate operatsioonide ning mugavuste pakkumiseks kasutatakse JavaScripti.

Teadmuse mudel on freimipõhine. Ontoloogia koosneb siin klassidest ja slottidest, fassetteid ei kuvata. Kasutajaliideses pakutakse umbes 70 käsku, mille abil on võimalik erinevaid operatsioone täide viia. Mõistete ja omaduste eristamisena kasutatakse värve. Süsteem toetab väga paljude eri formaatide lugemist, sealhulgas ka Protégé faile. Hea abiinfo süsteem on olemas nii põhisüsteemi kasutades kui ka eraldi.

Võrreldes eelpool vaadeldud tööriistadega pakub Chimaera mitmekasutaja toetust. Kasutajad saavad ühenduda gruppideks seansiks. Grupiliikmeid teavitatakse muudatustest, kui keegi midagi ontoloogias muudab.

Chimaera põhilisteks tugevusteks on tema funktsionaalsus, sealhulgas ühendamine ja diagnoosimine, paljude erinevate formaatide import ja eksport, väga hea abi ning mitmekasutaja töötamisvõimalus ontoloogia kallal.

Kokkuvõtvalt tuleks keel ja arenduskeskkond valida vastavalt vajadustele ja võimalustele. Silmas peaks pidama ka konkreetseid standardeid, mis ametlikult või siis mitteametlikult on erinevates arendusgruppides kasutusele võetud. Keele valikul peab eelkõige kindlaks tegema, kas keel võimaldab väljendada kõiki objekte, nende omadusi ja seoseid ning lisaks kas keel on toetatav samas arendatava tarkvara kõrval, mille jaoks ontoloogiat luuakse. Antud hetkel on semantiliste veebide ontoloogiate loomisel standardkeeleks OWL. Arenduskeskkonnad ametlikult standardiseeritud ei ole, kuid head väljavaated on Protégé'il ja OBO-Edit'il.

V PEATÜKK: METODOLOOGIAD

Varasemast käsitlusest on teada ontoloogiate olemus, milleks neid vaja läheb, millist tüüpi ontoloogiaid on olemas ning kuidas ontoloogiaid väljendada ning mille abil neid genereerida. Ainuüksi eelpool nimetatud faktidest on aga ontoloogia väljatöötamiseks vähe – on vaja juhtnööre, protsessijuhendit. Ehk siis teiste sõnadega metodoloogiat, kuidas töötada välja ontoloogia vajaduse tekkimisel.

Vähesed teadusgrupid pakuvad ontoloogiate ehitamiseks täpseid juhiseid ja metodoloogiaid. Kuna ontoloogiline arendus on veel suhteliselt toores, kasutab iga grupp ka oma metodoloogiat. Tunnustatumate metodoloogiatena on tuntud:

- Metodoloogia Uschold ja King poolt
- Metodoloogia Grüninger ja Fox poolt
- METHONTOLOGY
- SENSUS'el põhinev metodoloogia

Kõigi loetletud metodoloogiate lühiülevaade on toodud ära uurimuses [Lop99], kus võrreldakse neid tarkvara arendusprotsessi standardiga IEEE 1074-1995.

5.1 Fundamentaalsed reeglid ontoloogia kavandamiseks

Sõltumata ontoloogia kavandamise viisist ühendavad kõiki neid teatud reeglid. Need on nagu õpetuslaused, mida peaks alati meeles pidama.

- Pole olemas ühte kindlat teed valdkonna modelleerimiseks - alati on olemas alternatiive. Parim lahendus sõltub peaaegu alati rakendusest mis mõtteis ning ette nähtavatest laiendustest.
- Ontoloogia arendus on tingimata iteratiivne protsess, see tähendab et arendus koosneb mitmest iteratsioonist. Iteratsioonid omakorda koosnevad vabas järjekorras erinevates proportsioonides nõuete, analüüsi, teostamise ja teiste vajalike tegevuste hulgast, kus proportsioonid sõltuvad sellest, kus vastav iteratsioon arendustsüklis asub.

- Ontoloogia mõisted peavad olema objektilähedased (füüsiline ja loogiline) ja relatsioonid huvivaldkonna lähedased. Need on kõige tõenäolisemalt lausetes tuntud nimisõnad (objektid) ja tegusõnad (seosed), mis kirjeldavad valdkonda. [NM01]

Esimene punkt viitab sellele, et pole olemas ühte teed, kuidas kavandada ontoloogiat. Kõik sõltub konkreetsetest vajadustest, situatsioonist ja võimalustest. Teise punkti järgi on arendus iteratiivne protsess e. siis kirjeldatud etappe läbitakse ühel või teisel moel mitu korda läbi tsüklite (iteratsioonide). Kolmas punkt nõuab, et kasutatavad terminid eksisteeriksid mingil moel ning neid on võimalik kirjeldada. Viimane punkt haakub tihedalt ka ontoloogia filosoofilise käsitlusega.

Gruber'i käsitluse järgi on 5 põhilist kavanduskriteeriumit: selgus, sidusus, laiendatavus, minimaalne kodeerimismõjutus, minimaalne ontoloogiline kohustus. Ontoloogias spetsifitseeritud mõisted ja seosed peavad olema selged nii ainevalla spetsialistilile kui ka teadmusinsenerile. Piiritlevad aksioomid peavad olema loogiliselt kooskõlas ja kinnitama järeldused, mis on kooskõlas definitsioonidega. Ontoloogia peab olema lihtsalt laiendatav ja hallatav, ilma et peaks muutma suuri osi ontoloogias. Kodeerimismõjutus ilmneb, kui esitusvalik on tehtud notatsiooni mugavusest lähtudes. Kontseptualisatsioon peab olema spetsifitseeritud teadmustasemel sõltumata konkreetsest sümboltaseme kodeeringust. Ontoloogia peaks esitama võimalikult vähe nõudlusi modelleeritava maailma kohta, võimaldades seotud osapooltel spetsifitseerida ja konkretiseerida ontoloogiat vastavalt vajadusele.

5.2 Ontoloogia kavandamise sammud (Noy ja McGuinness)

Noy ja McGuinness [NM01] on pannud enda kogemuste põhjal kirja ontoloogia iteratiivse kavandamise sammud. Need on tugevaid mõjutusi saanud Uscholdi ja Kingi [UK95] metodoloogiast.

Samm 1: Määra ontoloogia valdkond ja skoop

Esimesel sammul peaks enda jaoks leidma vastuse järgnevatele küsimustele. Küsimused võivad protsessi jooksul muutuda, aga nad aitavad piirata mudeli skoopi.

Millist valdkonda ontoloogia katab?

Milleks me ontoloogiat kasutama hakkame?

Millistele küsimustele peaks informatsioon ontoloogias vastust andma?

Kes hakkab kasutama ja hooldama ontoloogiat?

Samm 2: Kaalu olemasolevate ontoloogiate taaskasutust

Alati tasub mõelda selle peale mida teised on teinud ja kontrollida kas me saame kasutada olemasolevaid allikaid oma valdkonna või töö jaoks. Olemasolevate ontoloogiate taaskasutamine võib olla ka nõue, kui meie süsteem peab suhtlema teiste rakendustega, millele on juba kehtestatud konkreetsed ontoloogiad. Formalism, milles ontoloogiaid esitatakse ei oma tähtsust, kuna teadmusesituse süsteemid suudavad importida ja eksportida ontoloogiaid. Isegi kui süsteem ei oska otseselt töötada konkreetse formalismiga, siis ühest formalismist teise tõlkimine ei ole raske töö.

Samm 3: Loetle ontoloogia tähtsad terminid

Siin osutub kasulikuks kõikide terminite üleskirjutamine, millega me soovime lauseid koostada või soovime kasutajale seletada. Järgmistele küsimustele tuleks leida vastused:

Millised on terminid, mille vahendusel me soovime suhelda?

Millised on omadused nendel terminitel?

Mida me nende terminitega öelda soovime?

Samm 4: Defineeri klassid ja klassihierarhia

Meetodid selleks on ülalt-alla, alt-üles ja nende kombinatsioon ning tulemuseks mõistete hierarhiline asetus (võib olla puu, suunatud atsükliline graaf jne). Ülalt alla lähenemise puhul alustatakse kõige üldisemate mõistete defineerimisega ning jätkatakse nende mõistete spetsialiseerimisega. Alt üles lähenemine on vastupidine, kus defineeritakse kõige spetsiifilisemad klassid, hierarhia lehed ning seejärel jätkatakse grupeerides neid järjest üldisematesse klassidesse. Kombinatsioon on kahe eelmise lähenemise vahepealne lähenemisviis. Siin defineeritakse kõige esilekerkivamad mõisted ning seejärel üldistatakse või spetsialiseeritakse neid vastavalt.

Samm 5: Defineeri klasside omadused – slotid

Klass ise ei oma piisavalt informatsiooni, seetõttu tuleb kirjeldada ka mõiste sisemine struktuur. Erinevateks omadusteks on loomuomased omadused (maitse, aroom), välised

omadused (nimi, päritolu), osad (nii füüsilised kui abstraktsed), suhestumine teiste indiviididega. Kõik klassi alamklassid pärivad selle klassi slotid.

Samm 6: Defineeri slottide fassetid

Fassetid kirjeldavad sloti omadusi. Need võivad olla väärtustüüp (näiteks: String, Number, Boolean), lubatud väärtused, lubatud väärtuste arv.

Samm 7: Loo isendid

Individuaalse klassiisendi defineerimiseks tuleb valida klass, luua individuaalne isend sellest klassist ning täita slottide väärtused.

Originaalvariandis Uscholdi ja Kingi [UK95] poolt polnud protsess iteratiivne ning ei kirjeldanud täpselt ka tehnikaid ja tegevusi. Noy ja McGuinness'i [NM01] käsitlus toob näidete varal rohkem selgust ning lisaks annab ka juhiseid millises vormis erinevaid mõisteid kirja panna. Sammude sees pole mainitud kontrollimist ja testimist, kuid see käib kogu protsessi jooksul sammudega kaasas. Kui ontoloogiat ei koostanud oma ala ekspert, siis oleks soovitav ka enne kasutuselevõttu lasta üle kontrollida see eksperdi poolt.

5.3 Methontology

Uurimuse [GFC01] järgi kõige küpsem metodoloogia on Methontology. Ontoloogia arendusprotsess kirjeldab ära tegevused, mis viiakse läbi ontoloogiate ehitamisel. Üldiselt jagatakse tegevused kolme kategooriasse.

Projekti juhtimise tegevused. Siia kuuluvad planeerimine, juhtimine ja kvaliteedikontroll. Planeerimine kirjeldab millised tööd on vaja ära teha, kuidas on nad korraldatud, kui palju aega ja ressursse on vaja tööde lõpetamiseks. Juhtimine tagab, et planeeritud tööd saaksid lõpule viidud. Kvaliteedikontroll tagab iga väljuva produkti (ontoloogia, tarkvara, dokumentatsioon) kvaliteedi headuse.

Arendus-orienteeritud tegevused. Siia kuuluvad spetsifitseerimine, kontseptualisatsioon, formaliseerimine, implementeerimine ja hooldus. Spetsifitseerimine annab teada, miks ontoloogiat ehitatakse, mis on selle kavandatavad kasutused ning kes on lõppkasutajad. Kontseptualisatsioon struktureerib valdkonna teadmuse kui tähendusrikkad mudelid teadmuse tasemel. Formaliseerimine muundab kontseptuaalse

mudeli formaalsele mudelile. Implementeerimine ehitab arvutatavad mudelid mingis arvutatavas keeles. Hooldus uuendab ja parandab ontoloogiat.

Tugitegevused. Siia kuulub rida tegevusi, mida tehakse eelmise punkti tegevustega samal ajal, kuna ilma nendeta ei ole võimalik ontoloogiat ehitada. Siia kuuluvad teadmiste omandamine, hindamine, integreerimine, dokumenteerimine ja konfiguratsiooni juhtimine. Teadmiste omandamine aitab valdkonna teadmusel vormi võtta. Hindamise kaudu võetakse vastu tehnilisi otsusi ontoloogiate, nendega seotud tarkvara keskkondade ja dokumentatsiooni kohta igas faasis ja nende elutsükli faaside vahel. Integreerimine on vajalik kui soovime luua uut ontoloogiat taaskasutades teisi ontoloogiaid, mis juba olemas on. Dokumentatsioon detailiseerib selgelt ja ammendavalt iga faasi tulemusi. Konfiguratsiooni juhtimise abil salvestatakse kõik dokumentatsiooni, tarkvara ja ontoloogia koodid muudatuste jälgimiseks.

Methontology eelisteks teiste metodoloogiate ees on tegevuste suurem detailiseeritus. Lisaks on rakenduste ehitamine rakendusest sõltumatu ning kasutatakse mõistete identifitseerimisena kombinatsiooni strateegiat (kõigepealt tuvastatakse kõige esilekerkivamad mõisted) [GFC01].

Kokkuvõtvalt metodoloogia valimisel tuleks tutvuda enne lõpliku valiku tegemist ka alternatiividega ning lugeda teiste kogemusi ja kommentaare. Kuna metodoloogiad ontoloogiate väljatöötamise vallas ei ole veel päris küpsed, siis tuleks neisse suhtuda suure ettevaatusega ning vaadata neile ka peale veidi kriitilisema pilguga. Kriitiline pilk aitab kainemalt mõelda ning otsusi teha, sest protsess mis sobis ühes keskkonnas ei pruugi sobida teises kohas.

Kokkuvõte

Infomahtude kasvamine, inimeste maailmavaadete erinevused, väljatöötatavate masinate ja tarkvara koostöö saavutamine, vajadus infot struktureerida ning korrastada tingib vajaduse mingi ühtse raamistiku loomiseks. Raamistiku abil peavad erinevad instantsid saama efektiivsemalt informatsiooni vahetada ning kasutada.

Ontoloogiad võimaldavat olemasolevat või kavandatavat informatsiooni paremini korrastada, esitada ning kasutada. Nii inimeste kui masinatevahelise suhtluse alusena saab ontoloogiaid kasutada ühiste arusaamade saavutamiseks. Lisaks saavad mõlemad osapooled hea ülevaate valdkonnast. Põhilisteks motivaatoriteks ontoloogiate kasutuselevõtuks on vajadus parendada kommunikatsiooni ja koostalitlusvõimet. Süsteemitehnika poole pealt on olulised korduvkasutus, usaldusväärsus ja spetsifikatsioon.

Ontoloogiad jagatakse kahte suuremasse gruppi: üldised ja valdkonna ontoloogiad. Üldised on aluseks valdkonna ontoloogiatele ning valdkonna ontoloogiad spetsifitseerivad mingi valdkonna sõnavara koos seostega.

Ontoloogiate loomiseks on vajalik keele, arenduskeskkonna ning metodoloogia valik. Valik tuleb teha vastavalt konkreetsetele vajadustele ja nõudmistele ning vajaduse korral valitut ümber kohandada.

Principles for designing ontologies

Bachelor thesis

Janno Veldemann

Abstract

The increasing volumes of information, different views of people, achieving interoperability between machines, software and people, the need to structure and arrange information create requirement for using a unified framework. With in that framework different instances should have a more effective way to use and exchange information. The usage of commonly designed and accepted ontologies is one way to achieve those goals.

Ontologies can be the basis of achieving common understanding in communicating between humans and machines. Ontologies allow to arrange, represent and use existing or upcoming information better. In addition both sides have a better overview of the domain of interest. The main motivators to implement ontologies are the need to enhance communication, interoperability and reusability, reliability, specification from systems engineering side.

Ontologies can be divided into two main groups: top-level and domain-specific ontologies. Top-level ontologies are the basis of domain ontologies and domain ontologies specify the vocabulary and relations of some more specific domain of interest.

It is important to choose wisely the language, environment and methodology to develop ontologies. The choice has to be done according to particular needs and demands. When necessary the chosen technique should be revised.

Kasutatud kirjandus

[Den02] **Denny, M.** Ontology Building: A Survey of Editing Tools. November, 2002. http://www.xml.com/pub/a/2002/11/06/ontologies.html

[Den04] **Denny, M.** Ontology Tools Survey, Revisited. Juuli, 2004 http://www.xml.com/pub/a/2004/07/14/onto.html

[GFC01] **Gómez-Pérez, A., Fernández-López, M., Corcho, O.** Methodologies, Tools and Languages. Where is the Meeting Point? 2001 http://www.schin.ncl.ac.uk/protege2001/ presentations/GOMEZ-~1.PDF

[Gru93a] **Gruber, T. R.** A translation approach to portable ontologies. – *Knowledge Acquisition*, 1993, 5(2):199-220,.

[Gru93b] **Gruber, T. R.** Toward principles for the design of ontologies used for knowledge sharing. – *Technical report KSL 93-04*, Knowledge Systems Laboratory, Stanford, 1993.

[GF95] **Grüninger, M., Fox, M.S.** Methodology for the design and evaluation of ontologies. *Workshop on Basic Ontological Issues in Knowledge Sharing*, Montreal, 1995.

[Kün96] **Küng, A.** Rahvusvaheline haiguste ja nendega seotud terviseprobleemide statistiline klassifikatsioon. 10 väljaanne, 2. köide, 1996.

[Lop99] **López, F. M.** Overview of methodologies for building ontologies. – *Proceedings of IJCAAI-99 Workshop on Ontologies and Problem Solving Methods*, Stockholm, Sweden August 2, 1999

[M02] **McGuinness, D. L.** Ontologies Come of Age. – *Spinning the Semantic Web: Bringing the World Wide Web to Its Full Potential*. MIT Press, 2002.

[Mus00] **Musen, M.A.** Ontology-Oriented Design and Programming. *SMI Technical Report 2000-0833*. 2000.

[NM01] **Noy, N. F., McGuinness, D. L.** Ontology Development 101: A Guide to Creating Your First Ontology. – *Stanford Medical Informatics Technical Report SMI-2001-0880*. Märts 2001.

[UK95] **Uschold, M. King, M.** Towards a Methodology for Building Ontologies. Workshop on Basic Ontological Issues in Knowledge Sharing. 1995.

[UG96] **Uschold, M., Gruninger, M.** ONTOLOGIES: Principles, Methods and Applications. – *Knowledge Engineering Review*, 1996, vol. 11, no. 2, lk. 93-136.

[UJ99] **Uschold, M., Jasper, R.** A Framework for Understanding and Classifying Ontology Applications – *Proceedings of the 12th Banff Knowledge Acquisition for Knowledge Based Systems Workshop*, University of Galgary/Stanford University, 1999.

[Chimaera] Chimaera
http://www.ksl.stanford.edu/software/chimaera/

[GO] Gene Ontology
http://www.geneontology.org/

[GRO] Dr. John Grohol's Psych Central
http://psychcentral.com/psypsych/Ontology_(computer_science)#Available_ontologies

[KIF] Knowledge Interchange Format, 1998
http://logic.stanford.edu/kif/dpans.html

[PHILO] Raul Corazzon. Ontology. A resource guide for philosophers.
http://www.formalontology.it/

[PSL] Process Specification Language
http://www.mel.nist.gov/psl/index.html

[TOVE] TOVE Ontology Project
http://www.eil.utoronto.ca/enterprise-modelling/tove/

[OBO] Open Biomedical Ontologies
http://obo.sourceforge.net/main.html

[OBOEDIT] OBO-Edit
http://sourceforge.net/project/showfiles.php?group_id=36855

[OWL] OWL Web Ontology Language Overview, 2004
http://www.w3.org/TR/2004/REC-owl-features-20040210/

[Protégé] Protégé
http://protege.stanford.edu/

Veebiviited on viimati kontrollitud: 25.05.2005