TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond

Kaur Matthias Raavel 193301

LOINC terminoloogia kasutusele võtmine arhetüüpmustritel põhineval ABC4HEDA alusmudelil

Bakalaureusetöö

Juhendajad: Gunnar Piho, PhD

Kristian Juha Ismo Kankainen, MSc

Tallinn 2022

Autorideklaratsioon

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

Autor: Kaur Matthias Raavel

17.05.2022

Annotatsioon

Selleks, et meditsiiniandmed oleksid koostalitlusvõimelised ning neid oleks võimalik teiseselt kasutada, peavad meditsiiniandmed olema formaliseeritud ning neil peavad olema küljes korrektsed koodid, mis näitavad, millega on tegemist. Käesoleva töö eesmärgiks on võtta ABC4HEDA alusmudelis kasutusele LOINC terminoloogia selliselt, et sarnase lähenemise abil oleks hiljem võimalik kasutusele võtta ka teisi terminoloogiaid, nagu on näiteks SNOMED CT ja ICD. Terminite filtreerimise ning kategoriseerimise loogika lahendatakse peamiselt *Arlow ja Neustadt* Reegli arhetüüpmustriga, mille abil on võimalik Terminoloogia Repositooriumilt küsida vastavaid termineid. Terminite kasutamiseks on võimalik deklaratiivselt luua erinevaid tingimusi, millest Reegli arhetüüpmuster paneb kokku tingimuste avaldise, mida hiljem Terminoloogia Repositoorium lugeda oskab. Autori hinnangul on saavutatud tulemuste abil võimalik võtta kasutusele ka teisi terminoloogiaid. Lisaks oleks võimalik tulemusi kasutada tarkvaraarenduses ka teistel eesmärkidel.

Lõputöö on kirjutatud eesti keeles ning sisaldab teksti 28 leheküljel, 5 peatükki, 22 joonist.

Abstract

Introduction of LOINC terminology to an archetype patterns based ABC4HEDA base model

For medical data to be interoperable and for medical data to be suitable for secondary use, they must be formalized. In addition, they must also include appropriate codes that refer to some corresponding terminology terms. For that to be the case, the formalizing must be done as easily as possible, but as the number of different terms in a terminology can be very large, the task may be difficult. The objective of this thesis is to introduce LOINC terminology into the ABC4HEDA base model so that other terminology, such as SNOMED CT and ICD, can later be introduced by using a similar approach. This thesis is also part of a bigger objective, which is to create an intermediary, that is suitable for describing any medical information and that can be used to improve the interoperability between different medical information systems. In this thesis, the logic of filtering and categorizing terms is solved mainly by the usage of $Arlow\ and\ Neustadt's$ Rule archetype pattern, which makes it possible to query the Terminology Repository for corresponding terms. It is possible to define different rules for the use of terms declaratively. From these declaratively defined rules, the Rule archetype pattern compiles an expression, that the Terminology Repository can read and process. Rules can be added and modified during the runtime of the system, which means that the system does not have to be restarted and no source code needs to be modified. According to the author, the achieved results solve the thesis' objective, and they are a step closer for medical data to be correctly formalized as soon as they are created. According to the author, other terminologies can be introduced into the ABC4HEDA base model with the help of the achieved results. In addition, the results could be used in software development for other purposes as well, for example to create a search engine. From business perspective, the achieved results are valuable, because they can help make business information systems more reliable and they can also reduce the costs of developing and maintaining software.

The thesis is in Estonian and contains 28 pages of text, 5 chapters, 22 figures.

Sisukord

1 Sissejuhatus

1.1 Üldine

Töö autor on alates 2020. aasta oktoobrist osalenud koos Gunnar Piho uurimisgrupiga ABC4HEDA arhetüüpidel põhineva semantilise alusmudeli arendamises, panustades erinevate arhetüüpmustrite koodis realiseerimisse ning nende ühik– ja vastuvõtutestimisse. Käesolev dokument annab ülevaate sellest, kuidas on nimetatud alusmudelis võimalik kasutusele võtta esialgu LOINC terminoloogia ning edaspidi võtta samal meetodil kasutusele ka teisi sarnaseid terminoloogiaid.

1.2 Probleem

Töö ülesandepüstitus seisneb meditsiiniinformaatika andmete koostalitluses ning teisese kasutamise võimaldamises. Selleks, et andmed oleksid koostalitlusvõimelised ning et neid oleks võimalik teiseselt kasutada, peavad andmed olema formaliseeritud ning neil peavad olema küljes korrektsed koodid, mis näitavad, millega on tegemist. Selle jaoks, et see oleks võimalik, peab formaliseerimine olema tehtud võimalikult lihtsaks, kuid probleem on selles, et kuna erinevaid terminoloogiaid ning nendes sisalduvaid termineid on väga palju, võib ülesanne osutuda keeruliseks.

1.3 Eesmärk

Käesoleva töö eesmärgiks on ABC4HEDA alusmudelis LOINC terminoloogia kasutusele võtmine selliselt, et sarnase lähenemise abil oleks võimalik hiljem kasutusele võtta ka teisi terminoloogiaid ning jõuda seeläbi lähemale sellele, et meditsiiniandmed oleksid juba nende tekkimise hetkel formaliseeritud. Käesolev töö on ühtlasi osa ka suuremast eesmärgist, milleks on luua selline vahemees, mis sobiks mis iganes meditsiiniinfo kirjeldamiseks ning mille abil saaks parendada erinevate meditsiiniinformaatika süsteemide koostalitlusvõimet.

1.4 Töö struktuur

Töö esimene osa koosneb metoodika kirjeldamisest, kus tutvustatakse uurimisobjekte ning tööriistu, mida kasutati selleks, et jõuda eesmärgini. Järgmises osas kirjeldatakse tulemusi, milleni kirjeldatud metoodika abil jõuti ning töö viimases osas analüüsitakse saadud tulemusi ja tehakse nende põhjal järeldusi. Sellele järgnevalt kirjeldatakse edasiste tööde võimalikke suundi.

2 Metoodika

2.1 LOINC

LOINC (*Logical Observation Identifiers Names and Codes*) [1] on USA mittetulundusliku meditsiiniuuringute organisatsiooni *Regenstrief Institute* poolt välja töötatud rahvusvaheline standard. LOINC pakub mitmeid universaalseid nimesid ja koode, pakkudes võimalust identifitseerida erinevaid kliinilisi ning laboratoorseid vaatlusi. LOINC 2.72 versioon sisaldab kokku 98268 erinevat terminit, mille hulgas on 26056 kliinilist terminit, mille kasutamisele käesolev töö ka peamiselt keskendub. Näiteid mõnest LOINC terminist on näha joonisel 1. Lihtsustamise mõttes keskendutakse käesolevas töös järgmistele LOINC termini osadele:

- LOINC_NUM – LOINC termini unikaalne identifikaator;
- COMPONENT – aine või üksus mida mõõdetakse;
- SCALE_TYP – kuidas vaatlust väljendatakse;
- SYSTEM – mille põhjal vaatlus tehti (nt. patsient, süda, veri, seerum, uriin);
- CLASS – LOINC termini kategooria, näiteks kehapikkused on kategoorias ’BDYHGT.ATOM’;
- TIME_ASPECT – ajaintervall, üle mille vaatlus tehti;
- LONG_COMMON_NAME – termini pikk detailne inimloetav nimetus;
- EXAMPLE_UCUM_UNITS – rahvusvaheliste ühikute süsteemi UCUM (*Unified Codes for Units of Measure*) [2] ühikute näidised, millega on võimalik vaatluste mõõtmisi kirjeldada;

LOINC termini välja nimetused on andmebaasis suurte tähtedega ning sõnade eraldajateks on alakriipsud (*MACRO_CASE*). Lähtekoodis on nimetused suurte algustähtedega, uut sõna eraldab nimetuse sisene suurtäht (*PascalCase*). Lisaks on lähtekoodis ka mõnda nime lihtsustatud (nt. $SCALE\_TYP = Scale$)

	LOINC_NUM	COMPONENT	SCALE_TYP	SYSTEM	CLASS	TIME_ASPCT	LONG_COMMON_NAME	EXAMPLE_UCUM_UNITS
1	3137-7	Body height	Qn	^Patient	BDYHGT.ATOM	Pt	Body height Measured	[in_us];cm;m
2	3138-5	Body height	Qn	^Patient	BDYHGT.ATOM	Pt	Body height Stated	[in_us];cm;m
3	56066-4	Body height^at age 25	Qn	^Patient	PHENX	Pt	Body height [Stated] --at age 25	[in_us];cm;m
4	8301-4	Body height	Qn	^Patient	BDYHGT.ATOM	Pt	Body height Estimated	[in_us];cm;m
5	8302-2	Body height	Qn	^Patient	BDYHGT.ATOM	Pt	Body height	[in_us];cm;m
6	8303-0	Body height	Qn	^Patient	BDYHGT.MOLEC	Pt	Body height [Percentile]	%
7	8305-5	Body height^post partum	Qn	^Patient	BDYHGT.MOLEC	Pt	Body height --post partum	[in_us];cm;m
8	8306-3	Body height^lying	Qn	^Patient	BDYHGT.MOLEC	Pt	Body height --lying	[in_us];cm;m
9	8307-1	Body height^preoperative	Qn	^Patient	BDYHGT.MOLEC	Pt	Body height --preoperative	[in_us];cm;m
10	8308-9	Body height^standing	Qn	^Patient	BDYHGT.MOLEC	Pt	Body height --standing	[in_us];cm;m
11	83845-8	Body height	Qn	^Father	BDYHGT.ATOM	Pt	Body height Father	[in_us];cm;m
12	83846-6	Body height	Qn	^Mother	BDYHGT.ATOM	Pt	Body height Mother	[in_us];cm;m
13	89269-5	Body height^at birth	Qn	^Patient	CLIN	Pt	Body height Measured --at birth	[in_i];cm
14	91370-7	Body height^used for drug calculation	Qn	^Patient	CLIN	Pt	Body height --used for drug calculation	[in_us];cm;m

Joonis 1. Näide mõnest LOINC terminist

2.2 ABC4HEDA

ABC4HEDA on andmemudel ja kood (semantiline alusmudel), mille eesmärgiks on toetada meditsiiniinformaatikas kasutuses olevate standardite ja terminoloogiate deklaratiivset esitamist ning seeläbi parendada tervishoius kasutuses olevate infosüsteemide koostalitlusvõimet [3, lk 5]. ABC4HEDA mudel põhineb *Arlow ja Neustadt* arhetüüpmustritel [4] ja võimaldab terviseandmeid esitada vastavalt *Zachmani* raamistikule [5]. *Arlow ja Neustadt* arhetüübid ja arhetüüpmustrid on Gunnar Piho doktoritöös ümber töötatud meditsiinilaboratooriumite vajadustest lähtuvalt [6]. Joonisel 2 on näha millised arhetüüpmustrid vastavad millistele *Zachmani* raamistiku küsimustele.

<table>
<tr><th colspan="7">Business requirements</th></tr>
<tr><th>What</th><th>How</th><th>Where</th><th>Who</th><th colspan="2">When</th><th>Why</th></tr>
<tr><th>Things</th><th>Processes</th><th>Locations</th><th>Persons</th><th colspan="2">Events</th><th>Strategies</th></tr>
<tr><td>Products and services</td><td>Reporting (feedback)</td><td>Organization and organization structure</td><td>Persons</td><td colspan="2">Business events</td><td>Business rules</td></tr>
<tr><td></td><td></td><td colspan="3">Party AP</td><td></td></tr>
<tr><td>Product AP</td><td colspan="3">Party relationship AP</td><td>Order AP</td><td>Inventory AP</td><td></td></tr>
<tr><td colspan="7">Rule AP</td></tr>
<tr><td colspan="7">Quantity and money AP</td></tr>
<tr><td colspan="7">Common infrastructure</td></tr>
</table>

Joonis 2. *Zachmani* raamistiku ja arhetüüpmustrite vastavuse tabeli pilt G. Piho doktoritööst, [6, lk 19]

2.3 Arhetüüp ja arhetüüpmuster

Sõna arhetüüp pärineb kreeka keelest ning see tähendab originaalmustrit. Arhetüübi puhul on tegemist mingi asja või asjaoluga, mis järjekindlalt kordub ning mida peetakse universaalseks mõisteks või olukorraks. Arhetüüpmuster on selline muster, mis koosneb mitmest arhetüübist. $Arlow\ ja\ Neustadt$ on oma raamatus kirjeldanud ära sellised arhetüübid ja arhetüüpmustrid, mida oleks võimalik kasutada tarkvaraarenduses. Arhetüübid ja arhetüüpmustrid on omavahel tihedalt seotud ning need on välja kujundatud sellel eesmärgil, et nendega saaks võimalikult täpselt kirjeldada reaalseid ärivaldkondi [4].

2.3.1 Osapoole arhetüüpmuster

Osapoole ($Party$) arhetüüpmuster on töötatud välja selleks, et oleks võimalik kirjeldada ning salvestada äri jaoks vajalikku informatsiooni Persoonist ($Person$) või Organisatsioonist ($Organization$). Persoon on füüsiline isik ja Organisatsioon on inimeste grupp (meeskond, osakond, ettevõte). Käesolevas töös keskendutakse Persoonile ning tema Kehameetrika ($Body\ Metric$) osale, mis on mõeldud kirjeldamaks just seda informatsiooni, mis on seotud Persooni füüsilise kehaga ja mida saab kasutada inimese tervisliku seisundi kohta käiva informatsiooni kirjeldamiseks [3, lk 13]. $Arlow\ ja\ Neustadt$ raamat mainib Kehameetrika arhetüüpi vaid väga põgusalt ning detailselt seda lahti ei seletata, kuna nimetatud autorite hinnangul on see väga lai mõiste ning nende sõnul on seda arhetüüpi tarvis vaid väga spetsiifilistes infosüsteemides [4]. Kuna käesolev töö keskendub meditsiiniinformaatikale, siis on Kehameetrika arhetüüp kahtlemata vajalik ning sellele keskendutakse rohkem. Kehameetrika arhetüüpi on laiendatud veel Kehameetrika tüübiga ($Body\ Metric\ Type$), mis võimaldab ära määrata, millise Kehameetrikaga on täpsemalt tegemist. Seega kasutatakse Kehameetrika arhetüübi (nagu ka kõikide teiste arhetüüpide) juures Peter Coadi $Item\ Description$ mustrit, mis võimaldab teadmisi programmeerimise asemel spetsifitseerida [7, lk 153].

2.3.2 Kvantiteedi arhetüüpmuster

Kvantiteedi ($Quantity$) arhetüüpmuster on mõeldud erinevate asjade koguste või mõõtude kirjeldamiseks, kuid see hõlmab endas ka loogikat, nagu näiteks mõõtühikute omavaheline teisendamine või ümardamine. Kvantiteedi arhetüüpmustrit läheb vaja selleks, et oleks võimalik teha erinevaid Kehameetrika mõõtmisi, näiteks mitu

sentimeetrit oli inimese pikkus, mitu kilogrammi oli tema kaal või mitu millimeetrit elavhõbedasammast oli tema vererõhk [3, lk 21, 23], [4].

2.3.3 Reegli arhetüüpmuster

Reegli (*Rule*) arhetüüpmuster on vajalik, et saaks kirjeldada ja rakendada erinevaid äri ja ärisüsteemi reegleid. Reegli arhetüüpi saab kasutada näiteks süsteemi mingile osale vanusepiirangu sätestamiseks või näiteks ühe osapoole esindusõiguse kontrollimiseks mõnele teisele osapoolele, kuid sellele ei ole ka samas mingeid piiranguid seatud ning seda saab vastavalt äri ja süsteemi nõuetele täiendada ja kohandada. Reegli arhetüüpmustri osad on Reeglite komplekt (*Rule set*), mis sisaldab endas läbi Reeglite kasutamiste (*Rule usage*) konkreetseid Reegleid. Reeglid koosnevad omakorda Reegli elementidest (*Rule element*), millega sätestatakse konkreetse Reegli kindlad tingimused. Reeglite komplektiga on võimalik kokku panna mitmeid erinevaid Reegleid. Eesmärk on seeläbi võimaldada mitmele erinevale Reeglile vastavuse kontrollimine samaaegselt, kusjuures on seeläbi võimaldatud ka Reeglite korduvkasutamine [3, lk 22–23], [4].

2.4 Repositooriumi muster

Repositooriumi (*Repository*) muster on vahemees andmete kihi ning äriloogika kihi vahel ning aitab neid üksteisest eraldatuna hoida. Repositooriumi kaudu saab andmeid leida, lisada, muuta ning kustutada nagu tavaliselt andmekogust. Repositoorium teeb taustal kõik selleks vajalikud operatsioonid. See aitab vähendada korduvat päringute loogikat, mis tähendab seda, et ei ole vaja iga andmetüübi jaoks eraldi välja kirjutada, kuidas vajalikke operatsioone andmebaasis teha tuleks [8].

2.5 Tööriistade kirjeldus

Tööriistadena kasutati keelt C# [9], .NET 6.0 raamistikku [10], ASP.NET Core raamistikku [11], Blazor raamistikku [12], Entity Framework Core raamistikku [13], Visual Studio 2022 Enterprise [14] arenduskeskkonda, Microsoft SQL [15] andmebaasi ja Microsoft SQL Management Studio 18 [16] töövahendit. Diagrammide loomiseks kasutati diagrams.net veebikeskkonda [17].

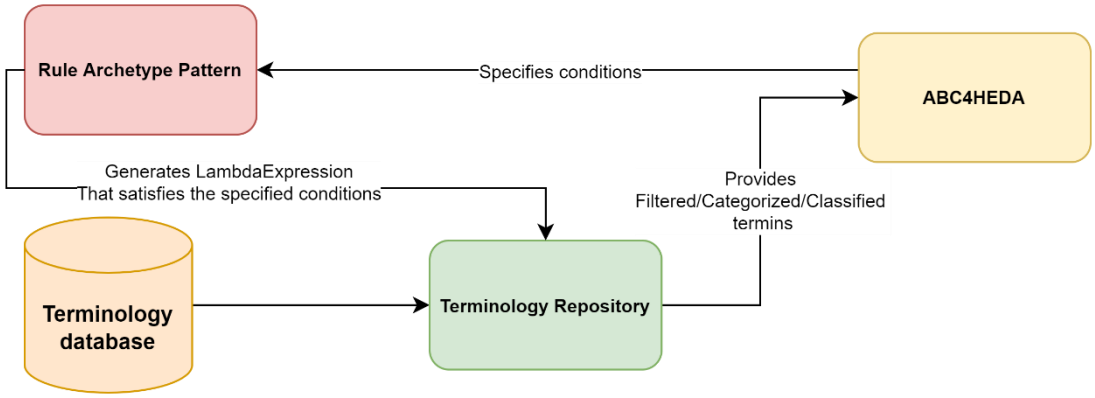
2.6 Täiendav info töö vormistuse kohta

Kui järgneva töö teksti sees on mõni eestikeelne sõna suure algustähega, siis räägitakse mõnest tarkvara klassi parameetrist või mõnest konkreetsest tarkvara tehisest. Kui jooniste juures ei ole öeldud teisiti, siis on tegemist autori koostatud joonistega. Kõik joonised on koostatud ingliskeelsena, et neid oleks võimalik esitleda ka rahvusvahelisele kogukonnale.

3 Peamised tulemused

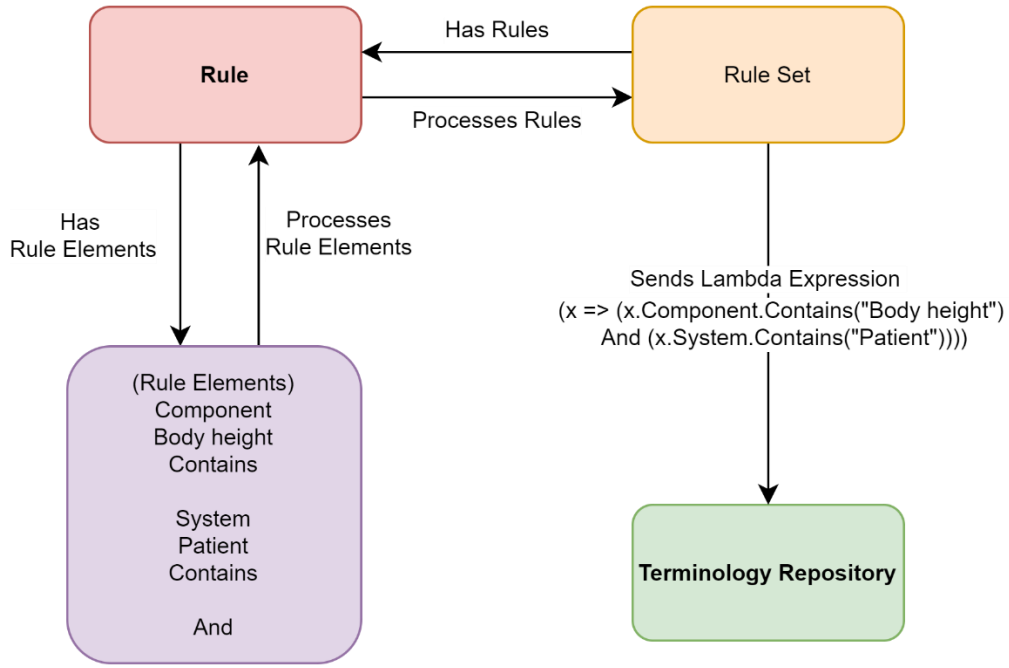
3.1 Ülevaade probleemi lahendusest

Järgnevalt on kirjeldatud probleemi lahenduse peamist tulemust. Terminoloogia andmebaas sisaldab informatsiooni meditsiiniterminite kohta. ABC4HEDA-sse on liidestatud Terminoloogia Repositoorium, mis on ühenduses Terminoloogia andmebaasiga. Reegli arhetüüp on üks osa ABC4HEDA-st, kus on võimalik täpsustada tingimusi, millele peab Terminoloogia Repositooriumilt küsitud tulemus vastama. Reegli arhetüüp paneb täpsustatud tingimused kokku ning saadab need *Lambda Expression* [18] kujul Terminoloogia Repositooriumile ning järgnevalt Terminoloogia Repositoorium otsib ning saadab vastavad terminid ABC4HEDA-sse, kus saab neid termineid kasutada Persoonide Kehameetrikate kirjeldamiseks. Kirjeldatud protsessi illustreerib joonis 3.



Joonis 3. Ülevaade probleemi lahendusest

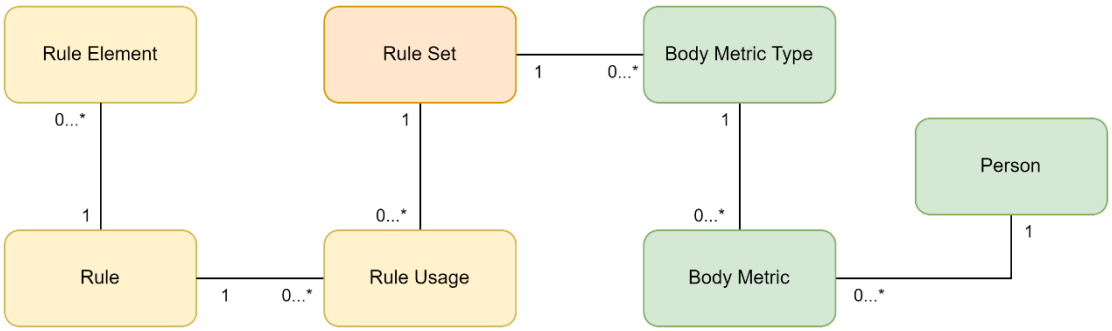
Reeglisse on Reegli elementide abil kirjeldatud tingimus või tingimused. Reegel töötleb Reegli elemente ning paneb selle abil kokku avaldise, mida Reeglite komplekt küsib igalt tema alla kuuluvalt Reeglilt. Reeglite komplekt paneb kõik Reeglite avaldised kokku ning saadab tulemuse Terminoloogia Repositooriumile, mille ülesandeks on selle põhjal otsida välja vastavad terminid. Kirjeldatud protsessi illustreerib joonis 4.



Joonis 4. Reegli arhetüüpmustri ülesanne probleemi lahenduses

3.2 Probleemi lahenduse detailsed etapid

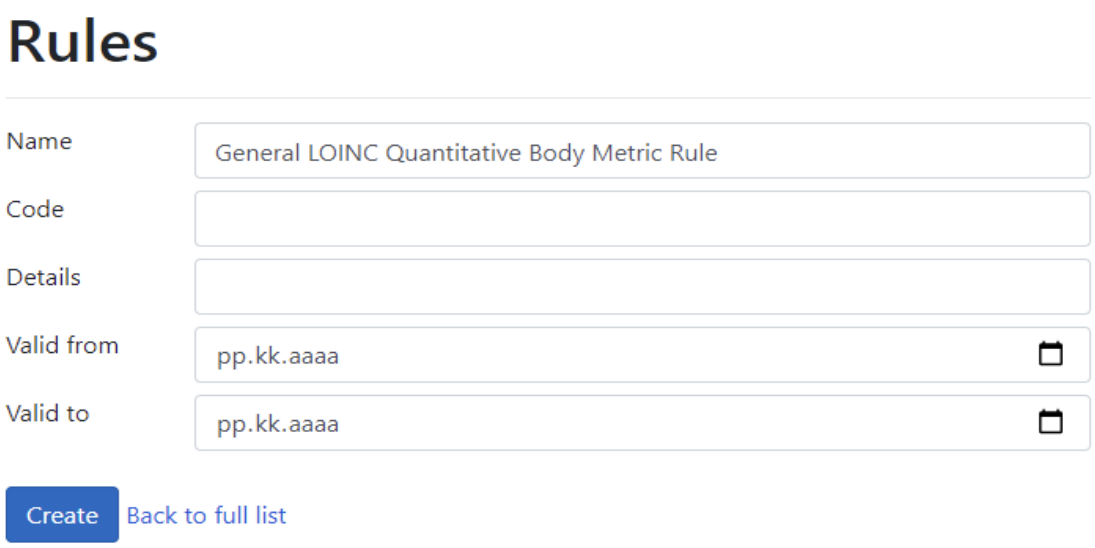
Selles alapeatükis kirjeldatakse detailselt lahti, kuidas läbi kasutajaliidese lisatakse Reeglit ning Kehameetrika tüüpi ja kuidas näeb välja Kehameetrika lisamine Persoonile. Reegli abil Persoonile Kehameetrika lisamiseks vajalike klasside struktuuri illustreerib joonis 5. Igal Reeglil võib olla mitu Reegli elementi. Reegli ja Reeglite komplekti vahel on mitu-mitmele seos, mis tähendab, et ühte Reeglit saab kasutada mitmes Reeglite komplektis. Kehameetrika tüübil on üks Reeglite komplekt, kuid ühte Reeglite komplekti on võimalik kasutada mitme Kehameetrika tüübi jaoks. Kehameetrika kasutab alati ühte Kehameetrika tüüpi, kuid ühte Kehameetrika tüüpi saab kasutada mitmel Kehameetrikal. Persoonil võib olla mitu Kehameetrikat, kuid Kehameetrikal on alati üks vastav Persoon.



Joonis 5. Reegli, Kehameetrika ja Persooniga kaasnevate klasside struktuur

3.2.1 Reegli loomine

Selleks, et luua Reeglit, tuleb navigeerida esmalt Reeglite loomise lehele. Reeglile peab andma Nimetuse (*Name*), vajadusel on võimalik täpsustada ka Kood (*Code*) ning Detailid (*Details*). Kui andmed on sisestatud, saab luua Reegli (Joonis 6).



Joonis 6. Reegli loomise vorm ABC4HEDA kasutajaliideses

Järgnevalt tuleb minna loodud Reegli detailide alla, mis viib Reegli elementide juurde. Järgnevalt määratakse Reeglile kaks LOINC terminiga seotud tingimust, mis peavad mõlemad korraga kehtima:

- $Scale$ sisaldab "Qn"
- $System$ sisaldab "$Patient$"

Selleks, et eelpool kirjeldatud tingimusi kirjeldada, tuleb lisada Reeglile seitse Reegli elementi (Joonis 7). Seda, mis on oluline Reegli elementide sisestamisel ning milline on selle taga seisnev loogika, on täpsemalt lahti seletatud peatükis 3.3.

Index	Element Type	Name	Value
1	String	LoincElement	Scale
2	String	LoincElement	Qn
3	Operator	Contains	
4	String	LoincElement	System
5	String	LoincElement	Patient
6	Operator	Contains	
7	Operator	And	

Joonis 7. Loetelu Reegli elementidest ABC4HEDA kasutajaliideses

Täpselt samal viisil nagu eelnevalt näidatud, luuakse veel üks Reegel nimetusega "$LOINC\ Body\ Height\ Rule$" mille elemendid sätestavad tingimuse, et "$Component$" peab endas sisaldama väärtust "$Body\ height$".

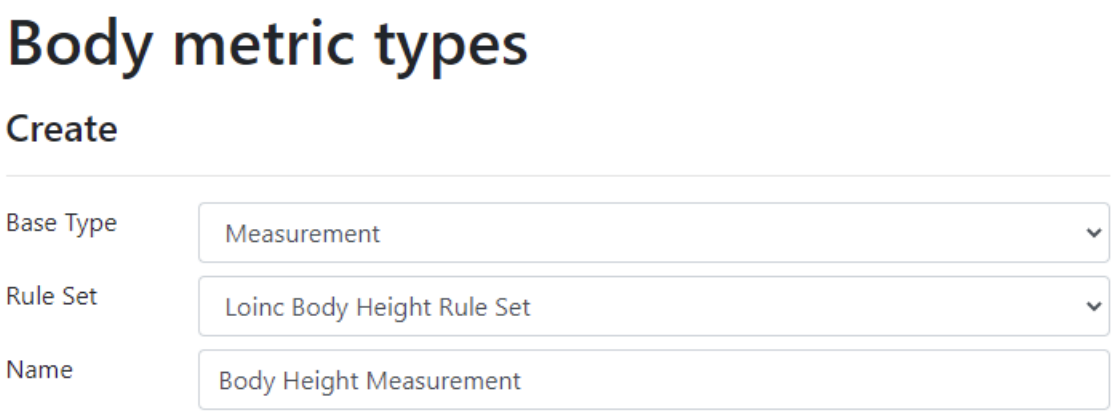
Seejärel luuakse uus Reeglite komplekt ning sellele lisatakse kaks Reeglite kasutamist, mis määravad, et loodud Reeglite komplekt kasutab eelnevalt loodud Reegleid "$General\ LOINC\ Quantitative\ Body\ Metric\ Rule$" ja "$LOINC\ Body\ Height\ Rule$" (Joonis 8).

Rule Set	Rule
Loinc Body Height Rule Set	General LOINC Quantitative Body Metric Rule
Loinc Body Height Rule Set	LOINC Body Height Rule

Joonis 8. Reeglite komplekti ja Reegli vahelised Reegli kasutamised ABC4HEDA kasutajaliideses

3.2.2 Kehameetrika tüübi loomine

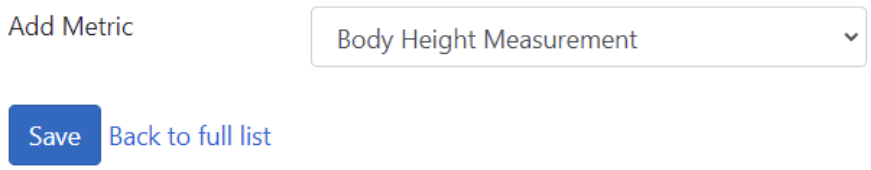
Kui Reeglite komplekt koos Reeglite kasutamiste, Reeglite ning Reegli elementidega on loodud, saab järgnevalt luua uue Kehameetrika tüübi ja määrata, et see kasutab eelnevalt loodud Reeglite komplekti. Kehameetrika tüübile tuleb anda nimi, mis annab selgituse, mis mõõtmist selle abil täpsemalt kirjeldada saab. Vastavalt eelnevalt loodud Reeglitele, saab selle nimeks "*Body Height Measurement*" (Joonis 9).



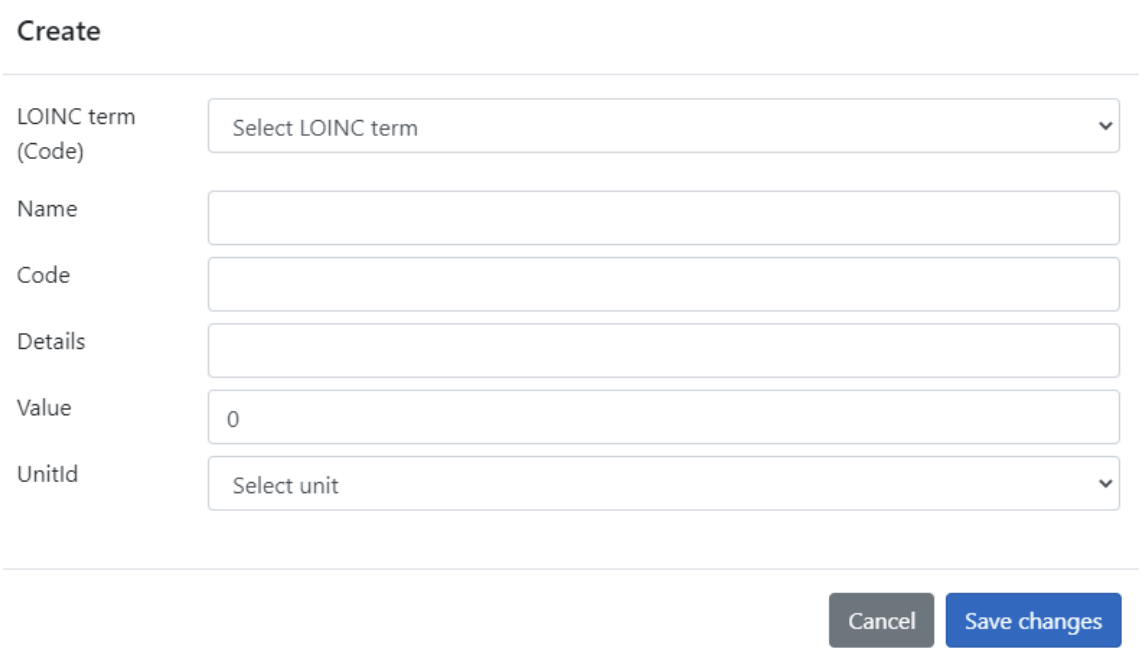
Joonis 9. Kehameetrika tüübi loomise vorm ABC4HEDA kasutajaliideses

3.2.3 Kehameetrika lisamine Persoonile

Kui Kehameetrika tüüp on loodud ning sellele on ka määratud Reeglite komplekt, on võimalik minna järgnevalt Persooni loomise või muutmise lehele ning Kehameetrikate lisamise ning muutmise sektsioonist saab valida eelnevalt loodud Kehameetrika tüübi (Joonis 10) ning peale seda avaneb Kehameetrika loomise vorm (Joonis 11).

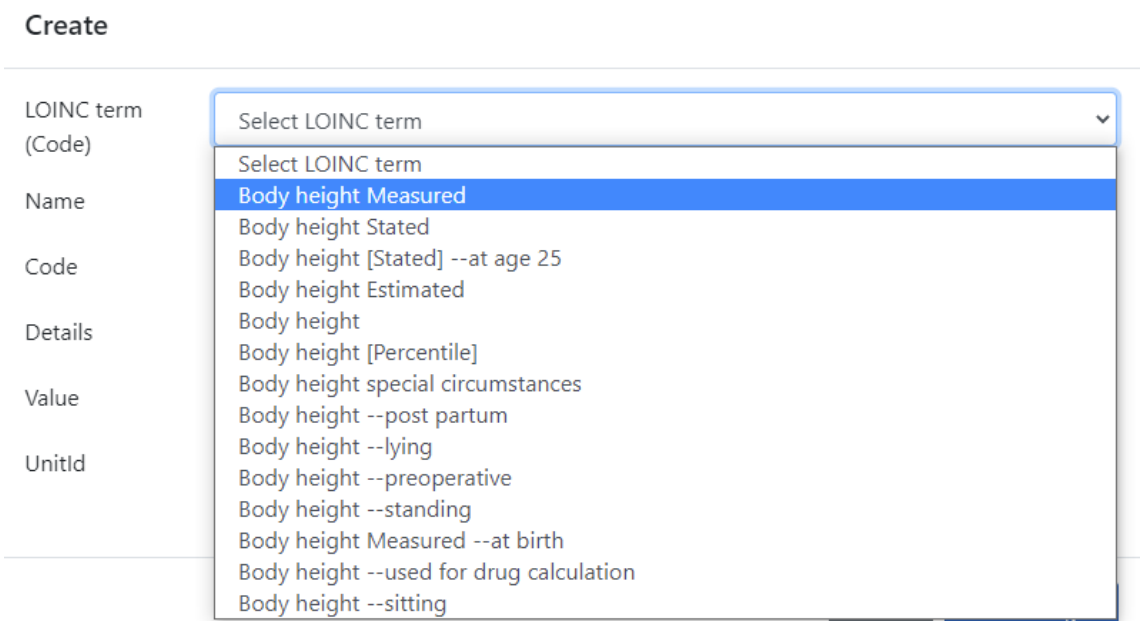


Joonis 10 Kehameetrika tüübi valimine Persooni muutmisel ABC4HEDA kasutajaliideses



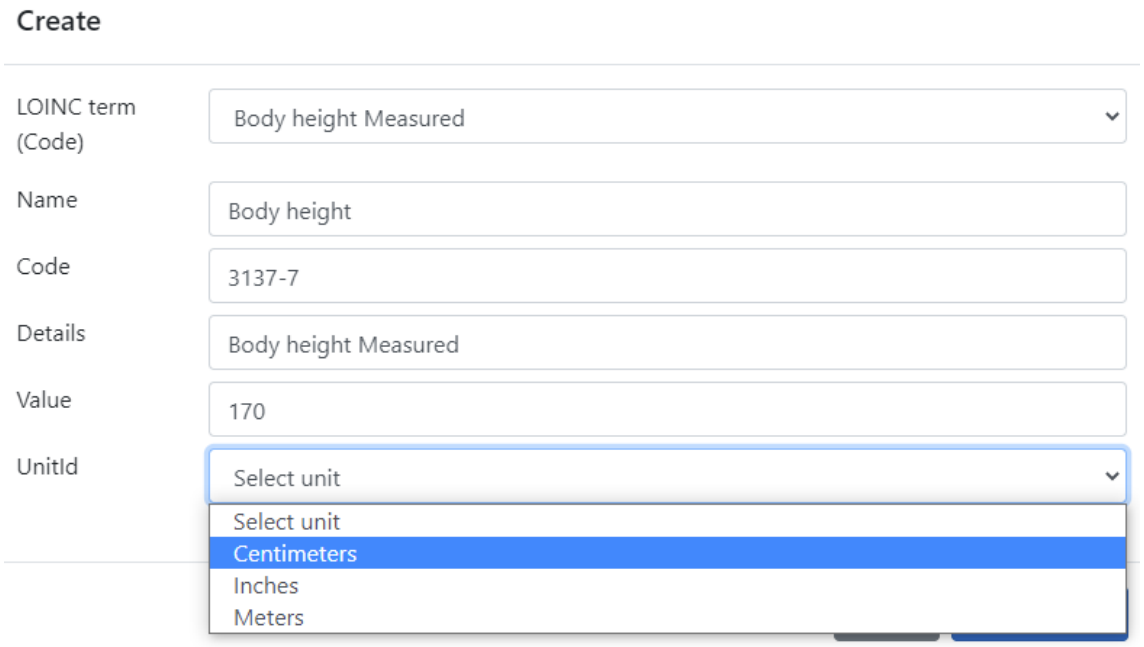
Joonis 11. Kehameetrika loomise vorm ABC4HEDA kasutajaliideses

Esimene valikloend sisaldab Reegli tingimustele vastavalt välja filtreeritud termineid (Joonis 12).



Joonis 12. LOINC termini valimine Kehameetrika loomise vormil ABC4HEDA kasutajaliideses

Kui teha valik, siis eeltäidetakse Nimi, Kood ja Detailid ning filtreeritakse vastavalt valitud terminis sisalduvate UCUM koodide järgi välja sobivad ühikud (näiteks ei pakuta pikkusega mitteseotud ühikuid nagu ruumalad või kiirused), mida saab valida kõige alumisest valikloendist (Joonis 13). Järgnevalt tuleb sisestada veel mõõtmise väärtus.



Joonis 13. Ühiku valimine Kehameetrika loomise vormil ABC4HEDA kasutajaliideses

Joonisel 14 on näha lisatud Kehameetrika kirje lõpptulemust.

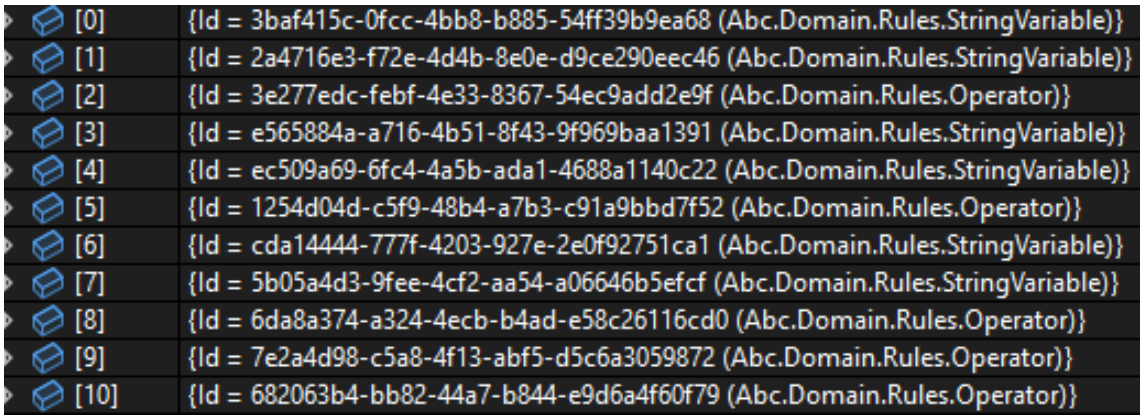
Metric type	Type	Name	Code	Details	Signed by	Value
Quantity	Body Height Measurement	Body height	3137-7	Body height Measured		170 Centimeters

Joonis 14. Kehameetrika kirje ABC4HEDA kasutajaliideses

3.3 Filtreerimise loogika täpne kirjeldus

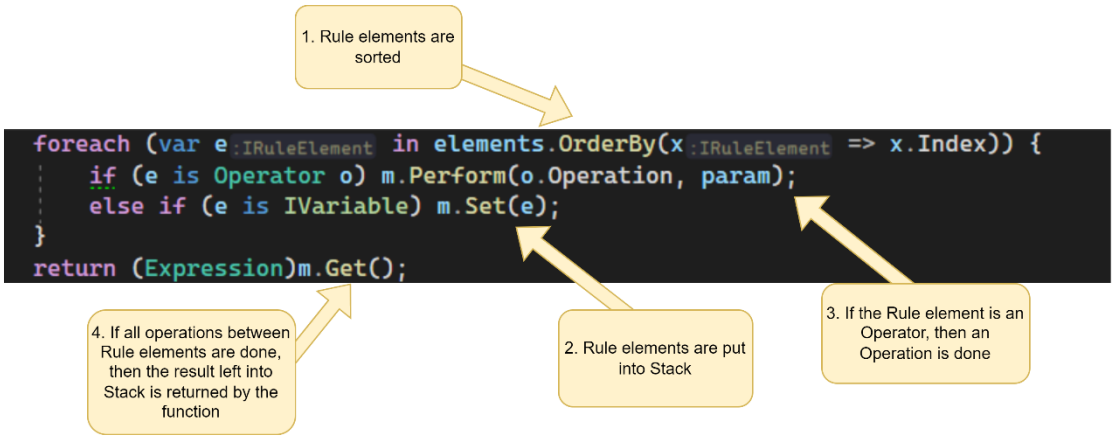
Eelmises peatükis kirjeldati esmalt Reegli loomise protsessi. Reegli loomise oluline osa on see, et Reegli elemendid peavad olema sisestatud Tagurpidi Poola notatsioonis (*Reverse Polish Notation*) [19]. Kõigepealt tuleb defineerida Reegli element, mis sisaldab välja nimetust (nt. "*Scale*"), sellele järgnevalt tuleb defineerida Reegli element, millega soovitakse teha operatsioon välja sisaldavale Reegli elemendile (nt. "*Qn*") ning siis sellele järgnevalt tuleb defineerida veel Operaator, milleks praegusel juhul oli *Contains* (sisaldab). Ilmtingimata ei ole vajalik neid alati just selles järjekorras sisestada, vajadusel saab järjekorda hiljem *Indexit* muutes sättida. See tingimuste avaldis kokku ütleb, et esimesena sisestatud väli peab sisaldama teisena sisestatud väärtust. Kui soovida lisada veel mõnda tingimust, tuleb sisestada samamoodi Reegli elemendid ning Operaator ja siis veel lisaks üks Operaator, mis paneb kokku kaks tingimust, eelnevalt toodud näites oli selleks AND (ja - kõik tingimused peavad kehtima). Toodud näites on demonstreeritud vaid ühte varianti sellest, milliseid loogikaavaldisi on võimalik defineerida. Lisaks AND operatsioonile on võimalik kasutada ka näiteks OR (või - üks tingimustest peab kehtima) või XOR (eksklusiivne või - vaid üks tingimustest tohib kehtida) operatsiooni ning lisaks *Contains* operatsioonile veel näiteks *Equals* (võrdne), *LessThan* (väiksem kui), *GreaterThan* (suurem kui), *StartsWith* (algab), *EndsWith* (lõppeb) ja ka muid.

Kui kogu Reegli komplekt on defineeritud, siis küsib Reegli komplekt iga sellele määratud Reegli kohta selles sisalduvate tingimuste avaldist. Kui Reegel hakkab panema kokku avaldist, siis esmalt Reegel otsib välja kõik oma Reegli elemendid ning sorteerib need *Indexite* järgi kasvavasse järjekorda (Joonis 15).



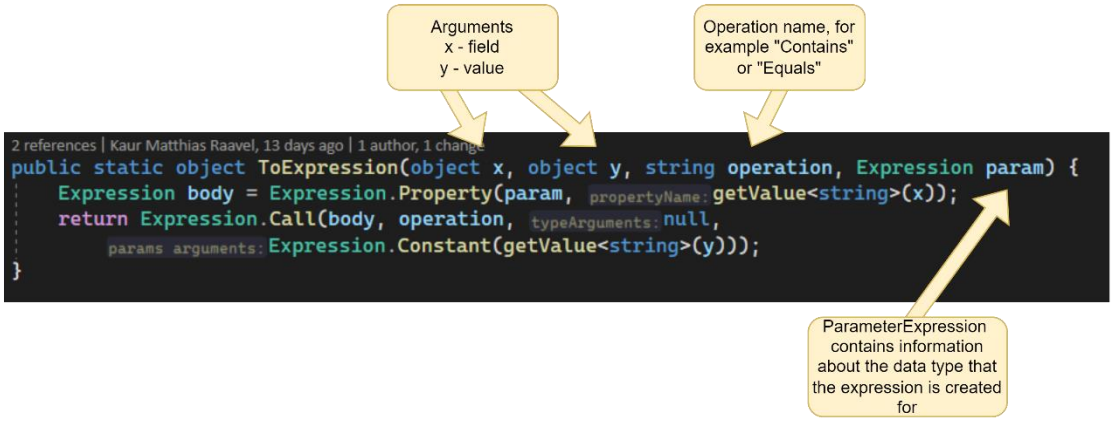
Joonis 15. Visual Studio *debug* vaade *Indexi* järgi sorteeritud Reegli elementidest

Reegel itereerib üle oma elementide ning paneb muutuja tüüpi Reegli elemente (argumente) *Stacki* (pinusse) seni, kuni jõutakse Operaatorini. Siis võetakse *Stackist* esimesed Reegli elemendid ning nende vahel tehakse Operaatorile vastav operatsioon. Tegevust jätkatakse seni, kuni kõik Reegli elemendid on läbi käidud ning *Stacki* on jäänud vaid tegevuse tulemusel loodud avaldis, mis saadetakse Reeglite komplektile. Joonisel 16 on näha Reegli elementide sorteerimise ning hindamisega tegelevat koodi.



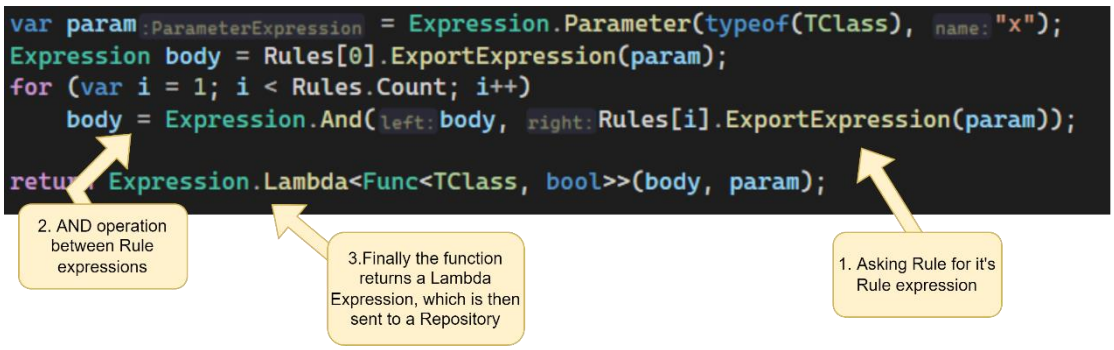
Joonis 16. Reegli elementide sorteerimise ning hindamisega tegelev kood

Argumentide vaheliste operatsioonide tegemiseks on kasutusel .NET 6.0 sisse ehitatud *Expression* klassi meetodid [20]. Joonisel 17 on näha näidet, kuidas kirjutatud koodi abil tehakse näiteks *Contains* või *Equals* operatsiooni.



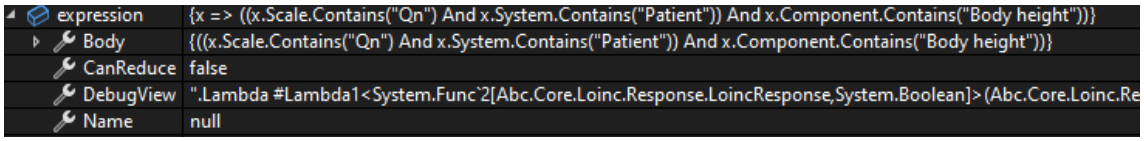
Joonis 17. Avaldise koostamisega tegelev kood

Reeglite komplekt paneb kokku igalt Reeglilt saadud avaldise, tehes nende vahel omakorda AND operatsiooni (Joonis 18).



Joonis 18. Reeglite omavahel kombineerimisega tegelev kood

Reeglite komplekti poolt saadud *Lambda Expression* läheb edasi *LoincRepository* vastavale meetodile, mis tagastab esitatud tingimuste põhjal välja otsitud terminid. Joonisel 19 on näha *Lambda Expression* visuaalset esitust ning joonisel 20 on näha *LoincRepository* meetodit, millelt küsitakse eelvalmistatud tingimuste avaldisele vastavaid termineid.



Joonis 19. Tingimuste avaldise visuaalne esitus

```
2 references | Kaur Matthias Raavel, 13 days ago | 1 author, 1 change
public async Task<List<LoincResponse>> FilterByExpression(Expression expression)
    => await db.LoincSet.Where((Expression<Func<LoincResponse, bool>>)expression).ToListAsync();
```

Joonis 20. *LoincRepository* meetod eelvalmistatud tingimuste avaldisega terminite küsimiseks

4 Analüüs ja järeldused

4.1 Eesmärgi põhjendus

Käesoleva töö eesmärk oli muuta LOINC terminid ABC4HEDA-s kasutatavaks ning see eesmärk saavutati. Töö tulemusena on võimalik panna paika tingimusi, millele terminid peavad vastama ja neid saab ka nende tingimuste järgi välja otsida. Tingimuste järgi välja otsitud termineid on võimalik kasutada Persoonide Kehameetrika salvestamiseks. Üheks töö eesmärgiks oli ka see, et sarnase lähenemise abil, nagu võeti kasutusele LOINC, oleks võimalik kasutusele võtta ka teisi terminoloogiaid – seda kirjeldatakse täpsemalt edasiste tööde peatüki 4.4 teises lõigus. Saavutatud tulemused viivad lähemale ka sellele, et meditsiiniandmed muutuksid formaalseteks kohe nende tekkimise ajal, ilma et selle jaoks oleks vaja spekuleerida ning teha erinevaid algoritme, ennustamaks, mida vabatekstina kirjutatud andmed tähendavad.

Tegemist on ka sammuga suurema eesmärgi täitmiseks. Suuremaks eesmärgiks on luua selline vahemees, mis sobiks mis iganes meditsiiniinfo kirjeldamiseks, et puuduks vajadus kokku leppida, et tuleb kasutada seda või teist suhtluskeelt. Eesmärk on ehitada selline süsteem, kus süsteemide omavaheline suhtlus põhineb födereeritud ($federated$) lähenemisel, mis tähendab seda, et standardites ja suhtluskeeltes saab kokku leppida käitusajal ($runtime$) [21, lk 648]. Eesmärgiks ei ole luua sellist süsteemi, kus süsteemide integratsioon põhineb ühtsel ($unified$) lähenemisel. Ühtne lähenemine tähendab, et lepitakse varasemalt kokku nendes standardites, milles suhtlus käib [21, lk 648]. Niisugune lähenemine ei ole hea, sest ei saa eeldada, et kõik maailma meditsiiniasutused hakkavad kasutama täpselt samu standardeid. Kui mõni meditsiiniasutus on harjunud mingisugust standardit kasutama, siis suure tõenäosusega ta ei ole huvitatud selle täielikust muutusest või vahetusest. Sellest tulenevalt on ka reaalses tänapäeva maailmas olukord, kus puudub meditsiiniasutuste vahel ühiselt kokku lepitud terminoloogia ning see on koostalitluse juures suureks kitsaskohaks. Eesmärk on luua vahemees, mille abil saab süsteemide omavahelises suhtluses kasutatavaid standardeid, mõisteid ja kasutatavaid koode spetsifitseerida ka käitlusajal. Üks meditsiiniasutus annab informatsiooni temale sobivas formaadis ning ABC4HEDA võtab selle informatsiooni

vastu, töötleb seda vastavalt ning edastab selle teisele meditsiiniasutusele temale arusaadavas formaadis. ABC4HEDA-l on olemas ka muid ärilisi omadusi ning süsteemide omavaheline koostalitlus ei piirdu vaid meditsiiniandmete vahetamisega, võimalik on ka näiteks mõne analüüsi tellimine teiselt asutuselt ja ka sellega kaasnev arveldamine. Nõudeid saab deklaratiivselt spetsifitseerida [6, lk 13]. ABC4HEDA on kavandatud erinema teistest sarnastest rakendustest (näiteks Mirth Connect [22] ja BizTalk [23]) selle poolest, et pakub üks-ühega andmevahetuse asemel üks-mitmega andmevahetuse võimalust.

4.2 Metoodika põhjendus

Tarkvaraarenduses on aastate jooksul kujunenud välja erinevaid disainimustreid. Disainimustreid on mõistlik kasutada, need on loodud selleks, et tarkvara oleks võimalik luua efektiivsemalt ning et lõpptulemusena loodud tarkvara oleks kvaliteetsem. Järgnevalt on kirjeldatud, kuidas on mõningate disainimustrite põhimõtteid rakendatud. Üheks mustriks, mida kasutati, on *Single Responsibility Principle* (SRP). Selle disainimustri põhimõte seisneb selles, et iga moodul, klass või funktsioon arvutiprogrammis peaks omama ühte vastutust ning see vastutus peaks olema kapseldatud selle sisse [24]. Põhimõtte järgimist on illustreeritud joonisel 3. Igal süsteemi osal on oma ülesanne ning mitte midagi rohkemat. Reegli ülesanne on tegeleda sellega, et oleks võimalik panna paika täpsed tingimused, millele termin peab vastama, seejuures Repositooriumi ülesanne on tegeleda andmete ligipääsuga, kuid repositoorium ei pea midagi teadma sellest, kuidas pannakse paika need tingimused, millele andmed peavad vastama. Repositoorium peab vaid võtma eelnevalt valmistatud tingimuse ning tagastama selle põhjal andmed – ei midagi rohkemat. Selle tulemusena on loodud kood puhtam, arusaadavam ja töökindlam. Lisaks sellele, et tänu erinevate ülesannete eraldi hoidmisele on kood puhtam, saab koodi ka taaskasutada. Kui eelnevalt kirjeldatud andmete filtreerimise loogika oleks sisse kirjutatud ühte kindlasse Repositooriumisse, siis oleks see kasutatav vaid ühe kindla andmetüübi peal. Kuid kuna see loogika on kirjeldatud hoopis Reeglis, siis on võimalik seda kasutada ka kõikides teistes andmete ligipääsuga tegelevates Repositooriumites.

Teine kasutatud disainimuster on *Open – Closed Princile* (OCP). Selle põhimõte seisneb asjaolus, et kood peab olema avatud lisadele, kuid suletud muudatustele [25].

Seda põhimõtet rakendati, kui Reeglit arendati selliselt, et seda oleks võimalik kasutada eelnevalt kirjeldatud tulemustes. Algselt oli Reegel mõeldud selleks, et oleks võimalik kontrollida ärireeglitele vastavust, kasvõi näiteks seda, et kas ühel või teisel isikul on mingisugune esindusõigus või kas tal on üldse kompetents, et saada omale vastav esindusõigus. Reeglisse oli võimalik samamoodi sisestada Reeglite komplekte, Reegleid, Reegli elemente ja Reeglite kasutamisi, kuid see pidi lõpuks peale tingimustele vastavuse kontrolli tagastama vaid tõeväärtuse $True$ või $False$ (tõene või väär). Mõnede täiendustega oli võimalik teha Reeglile selline juurdearendus, et kogu sisemine Reeglite ja Reegli elementide töötlus jäi kasutama samu klasse ning meetodeid, kuid selle asemel, et tagastada vaid tõeväärtus, on võimalik nüüd tagastada ka $Lambda\ Expression$ [18], mille abil on võimalik tõeväärtust hiljem arvutada. Juurdearenduse tegemisega kirjutati uut koodi juurde, kuid juba olemasoleva koodi peal muudatusi ei tehtud, mistõttu ei olnud ohtu, et vana kood enam ei tööta. Sellele, et vana kood siiski töötab, andsid lõpliku kinnituse ka ühiktestid. Edasiarendustes tuleks OCP printsiipi järgida selliselt, et oleks kaks ühe ja sama liidesega ($interface$) klassi, mis vastavad meetodid omakorda üle kirjutavad ($override$).

Põhjus, miks siinse töö kontekstis osutus valituks just LOINC, on põhjendatud järgnevalt. Kui võtta võrdluseks kõrvale näiteks SNOMED CT [26], siis nagu ka varasemalt mainitud, on LOINC 2.72 versioonis kokku 98 268 terminit, samal ajal kui SNOMED CT rahvusvahelise versiooni 2020. aasta 31. jaanuari väljalaskes oli 352 567 terminit, ehk enam kui kolm korda rohkem. Lisaks on SNOMED CT laiema haardega ning seda saab võtta pigem kui üleüldist meditsiiniterminite standardit, samal ajal kui LOINC on labori- ja kliiniliste tulemuste terminoloogia, kus puudub informatsioon näiteks diagnooside ja ravi kohta. LOINC koodidega kaasnev informatsioon vaatluse keskkonna ja mõõteriista iseloomu kohta on mõeldud küll käima LOINC terminiga kaasas, kuid terminis endas seesugune informatsioon puudub [27, lk 440]. Kusjuures SNOMED International ning Regenstrief Instituudi vahel on sõlmitud koostöölepe, mis sätestab selle, et SNOMED CT ja LOINC ei dubleeri üksteist [28]. Eelnevast tulenevalt paistis LOINC termin vähem keeruline ning tundus ratsionaalsem alustada esialgu LOINC kasutusele võtmisest ning alles hiljem võtta saadud teadmiste põhjal kasutusele SNOMED CT. Lisaks, kuna LOINC ei ole mõeldud eraldiseisva standardina kasutamiseks, on võimalik edasiste töödena veel valideerida seda, kas ABC4HEDA on piisav selleks, et see saaks toetada LOINC kasutamist.

4.3 Süsteeminõuete täpsustamine rakenduse käimise ajal

Eelnevalt esitletud tulemused näitavad, kuidas on võimalik kirjeldada ärireegleid rakenduse käimise ajal. Kui kujutada näiteks ette sellist olukorda, et on mingisugune meditsiinilabor, kus tehakse teatud kindlaid analüüse, mida olemasolevad masinad võimaldavad teha. Nüüd aga tuleb sinna uus masin või mõnel muul viisil tekib laborile uus kompetents ning edaspidi saavad nad teha ka teistsuguseid analüüse. Sellest tulenevalt on vaja muuta ka nõudeid infosüsteemis. Selle asemel, et muuta lähtekoodi ning rakendust taaskäivitada, on võimalik täiendada Reegleid ning Kehameetrika tüüpe selliselt, et oleks võimalik sisestada ka uute analüüside andmeid. Kuna nõudeid on võimalik täpsustada samal ajal, kui rakendus käib, tähendab see, et muudatuste tegemine on turvalisem, ei ole vaja uuendada lähtekoodi ning pole vaja rakendust uuesti käivitada. Selle tulemusena saavad näiteks teised labori töötajad jätkata samal ajal oma tööga.

Juhuks, kui nõuete muutmisel peaks toimuma siiski mingi viga, mille tulemusena tekib olukord, kus hiljem süsteem enam korralikult ei tööta, siis tänu läbimõeldud muudatuste halduse süsteemile on hiljem võimalik kontrollida, milliseid muudatusi tehti, kes neid tegi, miks neid tehti ning vajadusel saab taastada varasemad andmed [29, lk 461–462]. Kuna muudatusi ei tehta mitte lähtekoodis, vaid andmebaasis olevatel kirjetel, on lihtne säilitada tehtud muudatuste kohta ajalugu ning on ka lihtne neid muudatusi hiljem tagasi muuta. Kui need muudatused oleksid tehtud hoopis koodis, nõuaks hilisem muudatuste tagasi võtmine palju suuremat tööd, kaasa arvatud ka süsteemi mitmekordset taaskäivitamist, mis kõik võib võtta palju aega ning omakorda võib häiritud olla ka töö tegemine mujal organisatsioonis. Äri poole pealt vaadates tähendab rohkema töö tegemine suuremaid kulutusi tööjõule. Kui selle tulemusena võib olla häiritud töö tegemine mujal, siis tähendab see omakorda saamata jäänud tulu.

Arlow ja Neustadt on öelnud oma raamatus, et reegleid võib olla kolme tüüpi. On need reeglid, mis asuvad lähtekoodis ning mida on väga keeruline muuta, sest muutmine eeldab ka lähtekoodi modifitseerimist. Siis on reeglid, mis asuvad andmebaasi päästikus (*trigger*) ning siis on veel kolmas variant, kus reeglite haldamisega tegeleb reeglite

mootor. [4] Praegusel juhul on ABC4HEDA-s tegemist kolmanda variandiga ehk reeglite mootoriga. (Joonis 21).

Location	Advantages	Disadvantages
Source code	Applications have good performance	Rules are *very* hard to change, and it is expensive to make the change Rules are generally not reusable across applications Applications are inflexible
Database triggers	Rules are more accessible to change Applications are more flexible	Rules are coupled to data Rules are not reusable across data sets Not all rules can be encoded in this way
Explicit rules managed by a rules engine	Rules are completely accessible to change, and it is easy and inexpensive to make the change Rules are made explicit and so are open to inspection and review Rules are separate from applications and data Applications are very flexible	Applications may not perform as well as they would with hard-coded rules—this depends on the rules engine

Joonis 21. Enterprise Patterns and MDA raamatust pärit tabeli pilt Reegli täpsustamiseks [4]

4.4 Edasised tööd

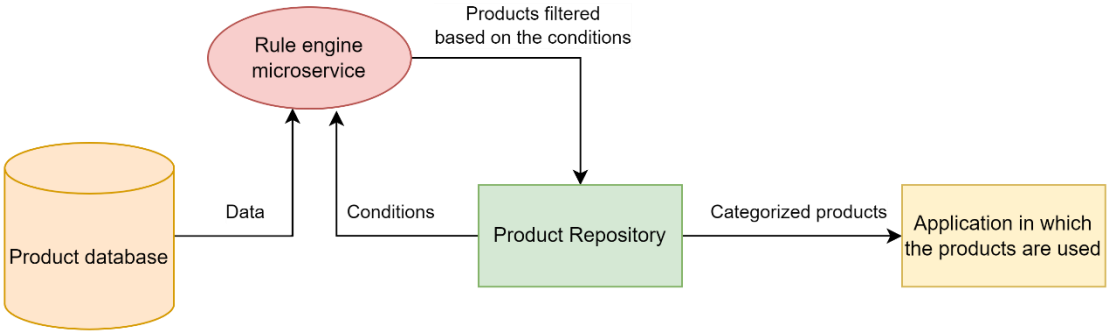
Üheks edasiseks tööks oleks arendada edasi tingimuste kirjeldamist. Kõigepealt peaks olema võimalik valida millisele andmetüübile tingimust luuakse ning sellele järgnevalt saab kasutajaliides ise pakkuda, milliseid väljasid andmetüüp sisaldab. Kui tegemist on näiteks mõne sellise väljaga, kus on mõned üksikud erinevad väärtused, näiteks "$Scale$" puhul võivad nendeks olla muuseas "Qn", "Ord","$OrdQn$","Nom","Nar" siis oleks võimalik genereerida selline valikloend, kust saab kõiki võimalikke väärtusi valida. Võimalike väärtuste leidmine peaks toimuma Reegli arhetüüpmustri abil. See tähendab esiteks seda, et võimalike väärtuste leidmise viis oleks deklaratiivselt spetsifitseeritav ning oleks võimalik muuta tingimusi, millisel juhul pakub liides ise valikloendi ning millisel juhul tuleb väärtus ise sisestada. Lisaks saaks juba kasutajaliidesest mugavalt valida tingimusi, näiteks kas on nõutud kõikide kirjeldatud tingimuste täitmine (AND), või peab olema üks kirjeldatud tingimustest täidetud (OR). Lisaks, kas andmeväli peab sisaldama kirjeldatud väärtust ($Contains$) või olema kirjeldatud väärtusega võrdne ($Equals$) ja nii edasi. Selline kasutajaliides aitaks muuta tingimuste kirjeldamise kiiremaks ja lisaks sellele vähendaks ka ohtu, et süsteemi kasutaja sisestab selliseid

reegleid, mis on vigased, millele ei leidu vastet või mida ei ole võimalik mõnel muul põhjusel täita.

Suures plaanis oleks edasine arendus võtta sarnasel viisil kasutusele ka muud terminoloogiad, mille võimaldamine oli ka käesoleva töö üheks eesmärgiks. Alustuseks võiks nendeks olla näiteks SNOMED CT [26] või ICD [30]. Selle jaoks ei oleks vaja teha suuri muudatusi. Esmalt tuleks luua sarnaselt praegusele $LoincRepositoryle$ juurde veel $SnomedRepository$ ja $IcdRepository$, mille ülesandeks oleks tegeleda vastavate andmebaaside ligipääsuga. Siis näiteks veel üks üldine $TerminologyRepository$, mis võtab kõik terminoloogiate repositooriumid omavahel kokku ning vastavalt Reeglis sätestatud tingimustele otsustab, milliselt konkreetselt $Repository$-lt on vaja andmeid küsida ning siis see konkreetne $Repository$ küsib omakorda vastavast andmebaasist. Kui tekitada juurde ka mainitud üldine $TerminologyRepository$ siis see võimaldaks ka tekitada sellised Kehameetrika tüüpe, kus on sees midagi nii LOINC kui ka ICD terminoloogiast. Kui kujutada ette näiteks olukorda, kus tuleb ravile patsient mingisuguse vigastusega, siis lisaks sellele, et talle määratakse ICD kood tema vigastuse kohta, võib olla vajadus salvestada veel mingisuguseid teisi andmeid tema kohta, mida ICD terminoloogiaga kirjeldada ei saa. Võttes arvesse valdkonna nõudeid, on võimalik tekitada järgnevalt erinevaid kategooriaid, mis hõlmavad osi erinevatest terminoloogiatest. See nõue ei pruugi reaalses maailmas täpselt sellisel kujul eksisteerida, kuid kui see peaks nii olema, või peaks selline nõue kunagi tekkima, siis oleks see võimalik täpsustada Reeglisse deklaratiivselt ning selle jaoks ei oleks vaja teha muudatusi lähtekoodis. Lisaks oleks veel võimalik kirjeldada sellised tingimused, mille puhul vastab ühe terminoloogia termin teise terminoloogia terminile. Näitena võib tuua olukorra, kus mõlema termini nimetuses sisaldub väärtus "kehapikkus" ning mõlema puhul on ette nähtud, et kaasneb ka numbriline väärtus, sellises kontekstis võib olla tegemist mingisuguse ühisosaga.

Kui mõelda veel laiemalt, jättes ABC4HEDA skoobist välja ning mõeldes tarkvaraarenduse peale üldiselt, siis oleks võimalik sarnast lähenemist kasutada ka mõne muu infosüsteemi loomisel näiteks otsingumootori ehitamisel. Otsingumootori idee toimiks väga sarnaselt sellele, kuidas on eelnevalt tulemustes näidatud. Üks variant oleks näiteks kasutada sellist lähenemist toodete kategoriseerimisel. Kui on olemas näiteks andmebaas toodetega, kuid nendele ei ole külge pandud konkreetseid kategooriaid, siis

reegli abil saaks panna paika tingimused, et millisel juhul kuulub toode mingisse kindlasse kategooriasse. Kasutades sarnast lähenemist, nagu kirjeldab kahendotsingu või otsingupuu algoritm [31] (st. et jagada andmeid mitu korda millegi järgi kaheks), leida üles toodete peamised erinevused (nt. uus toode, kasutatud toode, kollane toode, veekindel toode, või midagi muud analoogset), siis saab luua vastavad reeglid ja saab tooted ära kategoriseerida. Üks variant oleks luua eraldi mikroteenus, mis ongi reeglite mootor ja sealt saaks välja eksportida erinevaid tingimusi, mille abil oleks võimalik andmeid filtreerida. Või siis võtta lähenemine, et Reeglite mootorile antakse sisendiks tingimused ning filtreerimata kujul andmed ning Reeglite mootor tagastab filtreeritud andmed. Reegleid oleks võimalik defineerida varasemalt süsteemi administraatori poolt, mida kasutajal on võimalik juba lihtsalt valida (nagu tulemustes Kehameetrika tüüpi), või siis on kasutajal omal liides, kus ta saab valida, et millistele tingimustele peab otsitav toode vastama. (Joonis 22)



Joonis 22. Reeglite mootori mikroteenuse kasutamine

5 Kokkuvõte

Käesolev töö käsitleb peamiselt meditsiiniandmete formaliseerimise lihtsustamiseks tehtud toimingud. Probleem on selles, et terminoloogiaid on palju ja nende kasutusele võtmine ei pruugi olla niisama lihtne, kuna ühes terminoloogias võib veel omakorda olla termineid kümneid või lausa sadu tuhandeid. Eesmärk oli võtta kasutusele LOINC terminoloogia ning teha seda selliselt, et sarnase metoodika abil oleks hiljem võimalik kasutusele võtta ka muud sarnased terminoloogiad, nagu näiteks SNOMED või ICD. Selleks, et selliste eesmärkideni jõuda, kasutati arhetüüpmustreid ning erinevaid disainimustreid, sealhulgas ka Repositooriumi mustrit. Selleks, et jõuda tulemusteni, modifitseeriti olemasolevas ABC4HEDA alusmudelis olevat Reegli arhetüüpmustrit selliselt, et seda oleks võimalik kasutada lisaks erinevate ärireeglite vastavuse kontrollimiseks ka tingimuste eksportimiseks. Eksporditud tingimusi on hiljem võimalik kasutada terminite otsimiseks. Reeglis töödeldud tingimused saadetakse Terminoloogia Repositooriumile, mille ülesandeks on tingimuste põhjal väljastada tingimustele vastavad terminid. Repositooriumilt saadud termineid on hiljem võimalik kasutada kasutajaliideses, et salvestada Kehameetrikaid. Reeglile tehtud tingimusi on lihtne muuta, selle jaoks ei pea muutma lähtekoodi ning seda saab teha süsteemi töötamise ajal. Tänu sellele saab muudatusi teha kiiresti, kusjuures ei ole vaja programmeerimisoskusi ja see on vähem veaohtlik, sest muudatusi on võimalik sama lihtsasti tagasi võtta. Edasisteks töödeks on esimese asjana arendada edasi Reeglite loomise kasutajaliidest, et Reeglite loomine oleks kiirem ning vähem veaohtlik. Järgnevalt oleks edasisteks töödeks võtta sarnase metoodika abil kasutusele ka teisi terminoloogiaid.

Kasutatud kirjandus

[1] Regenstrief Institute, „LOINC User's guide,“ 2022. [Võrgumaterjal]. Available: https://loinc.org/kb/users-guide/. [Kasutatud märts 2022].

[2] Regenstrief Institute, Inc, „The Unified Code for Units of Measure,“ 2017. [Võrgumaterjal]. Available: https://ucum.org/trac. [Kasutatud märts 2022].

[3] R. Randmaa, T. Sõerd ja K. M. Raavel, „ABC4HEDA dokumentatsioon,“ 2021. [Võrgumaterjal]. Available: https://livettu-my.sharepoint.com/:w:/g/personal/kraave_ttu_ee/EVT_V3URm01NkiDZ3i-Tim8BDK4V4K2wZTuMHNJ2MjKOKA?e=oAmHqU. [Kasutatud mai 2022].

[4] J. Arlow ja I. Neustadt, „Enterprise Patterns and MDA: Building Better Software with Archetype Patterns and UML,“ *Party archetype pattern, Quantity archetype pattern, Rule archetype pattern*, 2003.

[5] J. A. Zachman, „The Zachman Framework,“ 2008. [Võrgumaterjal]. Available: https://www.zachman.com/about-the-zachman-framework. [Kasutatud märts 2022].

[6] G. Piho, „Archetypes Based Techniques for Development of Domains, Requirements and Software : Towards LIMS Software Factory,“ Tallinna Tehnikaülikool, 2011.

[7] P. Coad, „Object-Oriented Patterns,“ *Communications of the ACM,* pp. 152-159, 1992.

[8] M. Fowler, E. Hieatt ja R. Mee, „Patterns of Enterprise Application Architecture,“ *Object-Relational Metadata Mapping Patterns*, Addison-Wesley Professional, 2002.

[9] Microsoft, „C# dokumentatsioon,“ 2022. [Võrgumaterjal]. Available: https://docs.microsoft.com/en-us/dotnet/csharp/. [Kasutatud märts 2022].

[10] Microsoft, „.NET dokumentatsioon,“ 2022. [Võrgumaterjal]. Available: https://docs.microsoft.com/en-us/dotnet/. [Kasutatud märts 2022].

[11] Microsoft, „ASP.NET Core dokumentatsioon,“ 2022. [Võrgumaterjal]. Available: https://docs.microsoft.com/en-us/aspnet/core/?view=aspnetcore-6.0. [Kasutatud märts 2022].

[12] Microsoft, „Blazor dokumentatsioon,“ 2022. [Võrgumaterjal]. Available: https://docs.microsoft.com/en-us/aspnet/core/blazor/?view=aspnetcore-6.0. [Kasutatud märts 2022].

[13] Microsoft, „Entity Framework Core,“ 2021. [Võrgumaterjal]. Available: https://docs.microsoft.com/en-us/ef/core/. [Kasutatud märts 2022].

[14] Microsoft, „Visual Studio,“ 2022. [Võrgumaterjal]. Available: https://visualstudio.microsoft.com/vs/. [Kasutatud märts 2022].

[15] Microsoft, „Microsoft SQL dokumentatsioon,“ 2022. [Võrgumaterjal]. Available: https://docs.microsoft.com/en-us/sql/?view=sql-server-ver15. [Kasutatud märts 2022].

[16] Microsoft, „Microsoft SQL Server Management Studio,“ 2022. [Võrgumaterjal]. Available: https://docs.microsoft.com/en-us/sql/ssms/sql-server-management-studio-ssms?view=sql-server-ver15. [Kasutatud märts 2022].

[17] JGraph Ltd, „diagrams.net,“ 2022. [Võrgumaterjal]. Available: https://www.diagrams.net/. [Kasutatud märts 2022].

[18] Microsoft, „Lambda Expression C#-s,“ 2022. [Võrgumaterjal]. Available: https://docs.microsoft.com/en-us/dotnet/csharp/language-reference/operators/lambda-expressions. [Kasutatud märts 2022].

[19] Wikipedia, „Reverse Polish notation,“ 2022. [Võrgumaterjal]. Available: https://en.wikipedia.org/wiki/Reverse_Polish_notation. [Kasutatud märts 2022].

[20] Microsoft, „Expression Class,“ 2022. [Võrgumaterjal]. Available: https://docs.microsoft.com/en-us/dotnet/api/system.linq.expressions.expression?view=net-6.0. [Kasutatud märts 2022].

[21] D. Chen, G. Doumeingts ja F. Vernadat, „Architectures for enterprise integration and interoperability: Past, present and future,“ *Computers in Industry,* nr 59, pp. 647-659, 2008.

[22] NextGen Healthcare, „Mirth Connect,“ 2022. [Võrgumaterjal]. Available: https://www.nextgen.com/products-and-services/integration-engine. [Kasutatud mai 2022].

[23] Microsoft, „Mircosoft BizTalk dokumentatsioon,“ 2022. [Võrgumaterjal]. Available: https://docs.microsoft.com/en-us/biztalk/. [Kasutatud mai 2022].

[24] M. Martin ja R. C. Martin, „Agile Principles, Patterns, and Practices in C#,“ *The Single-Responsibility Principle (SRP),* 2006.

[25] M. Martin ja R. C. Martin, „Agile Principles, Patterns, and Practices in C#,“ *The Open/Closed Principle (OCP),* 2006.

[26] SNOMED International, „SNOMED 5-Step briefing,“ 2022. [Võrgumaterjal]. Available: https://www.snomed.org/snomed-ct/five-step-briefing. [Kasutatud märts 2022].

[27] C. Drenkhahn ja J. Ingenerf, „The LOINC Content Model and Its Limitations of Usage in the Laboratory Domain,“ *Digital Personalized Health and Medicine,* pp. 437-442, 2020.

[28] Regenstrief Institute, „Regenstrief Institute and SNOMED International collaboration,“ 2013. [Võrgumaterjal]. Available: https://loinc.org/collaboration/snomed-international/. [Kasutatud aprill 2022].

[29] G. Piho, J. Tepandi, D. Thompson, T. Tammer, M. Parman ja V. Puusep, „Archetypes based meta-modeling towards evolutionary, dependable and interoperable healthcare information systems,“ *Procedia Computer Science,* nr 37, pp. 457-464, 2014.

[30] World Health Organization, „International Classification of Diseases 11th Revision,“ 2019. [Võrgumaterjal]. Available: https://icd.who.int/en. [Kasutatud märts 2022].

[31] Wikipedia, „Kahendotsing,“ 2021. [Võrgumaterjal]. Available: https://et.wikipedia.org/wiki/Kahendotsing. [Kasutatud mai 2022].

Lisa 1 – Lihtlitsents lõputöö reprodutseerimiseks ja lõputöö üldsusele kättesaadavaks tegemiseks[1]

Mina, Kaur Matthias Raavel

1. Annan Tallinna Tehnikaülikoolile tasuta loa (lihtlitsentsi) enda loodud teose "LOINC terminoloogia kasutusele võtmine arhetüüpmustritel põhineval ABC4HEDA alusmudelil", mille juhendajad on Kristian Juha Ismo Kankainen ja Gunnar Piho
   1.1. reprodutseerimiseks lõputöö säilitamise ja elektroonse avaldamise eesmärgil, sh Tallinna Tehnikaülikooli raamatukogu digikogusse lisamise eesmärgil kuni autoriõiguse kehtivuse tähtaja lõppemiseni;
   1.2. üldsusele kättesaadavaks tegemiseks Tallinna Tehnikaülikooli veebikeskkonna kaudu, sealhulgas Tallinna Tehnikaülikooli raamatukogu digikogu kaudu kuni autoriõiguse kehtivuse tähtaja lõppemiseni.
2. Olen teadlik, et käesoleva lihtlitsentsi punktis 1 nimetatud õigused jäävad alles ka autorile.
3. Kinnitan, et lihtlitsentsi andmisega ei rikuta teiste isikute intellektuaalomandi ega isikuandmete kaitse seadusest ning muudest õigusaktidest tulenevaid õigusi.

17.05.2022

1 Lihtlitsents ei kehti juurdepääsupiirangu kehtivuse ajal vastavalt üliõpilase taotlusele lõputööle juurdepääsupiirangu kehtestamiseks, mis on allkirjastatud teaduskonna dekaani poolt, välja arvatud ülikooli õigus lõputööd reprodutseerida üksnes säilitamise eesmärgil. Kui lõputöö on loonud kaks või enam isikut oma ühise loomingulise tegevusega ning lõputöö kaas- või ühisautor(id) ei ole andnud lõputööd kaitsvale üliõpilasele kindlaksmääratud tähtajaks nõusolekut lõputöö reprodutseerimiseks ja avalikustamiseks vastavalt lihtlitsentsi punktidele 1.1. ja 1.2, siis lihtlitsents nimetatud tähtaja jooksul ei kehti.