TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond
Informaatikainstituut
Infosüsteemide õppetool

CASE vahendite Enterprise Architect ja DB-MAIN võrdlemine tervishoiuasutuse infosüsteemi vastuvõtuaegade allsüsteemi projekteerimise näitel

Bakalaureusetöö

Üliõpilane: Kai Kabin
Üliõpilaskood: 091291IABB
Juhendaja: dotsent Erki Eessaar

Tallinn
2015

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.

……. …….

(kuupäev) *(allkiri)*

CASE vahendite Enterprise Architect ja DB-MAIN võrdlemine tervishoiuasutuse infosüsteemi vastuvõtuaegade allsüsteemi projekteerimise näitel

Annotatsioon

Bakalaureusetöö teemaks on tervishoiuasutuse infosüsteemi vastuvõtuaegade funktsionaalse allsüsteemi ja selle tööks vajaliku andmebaasi alamosa projekteerimine kasutades Enterprise Architect ja DB-MAIN CASE vahendeid. Antud töö eesmärgiks on kasutatud vahendite võrdlus registripõhiste infosüsteemide projekteerimise seisukohast – kumba vahendit on parem kasutada niisuguste infosüsteemide projekteerimiseks ja ka selle õpetamiseks.

Töö esimeses osas tutvustatakse erinevate juhendmaterjalide põhjal Enterprise Architect ja DB-MAIN vahendeid ning nende poolt pakutavaid võimalusi. Teises osas projekteeritakse infosüsteemi alamosa. Tegemist ei ole reaalselt kasutatava või kasutama hakatava infosüsteemiga, vaid autori nägemusega sellest. Süsteemi projekteerimine on jagatud omakorda kolmeks suureks etapiks: strateegia, detailanalüüsi ja disaini etapp. Viimases osas võrreldakse Enterprise Architect ja DB-MAIN vahendeid kõigi kolme projekteerimise etapi seisukohast.

Bakalaureusetöö tulemuseks on tervishoiuasutuse infosüsteemi vastuvõtuaegade allsüsteemi projekt ning Enterprise Architect ja DB-MAIN võrdlus sellise projekti loomise seisukohast. Autor järeldas, et registripõhiste infosüsteemide projekteerimiseks ja selle õpetamiseks on parem kasutada Enterprise Architecti.

Lõputöö on kirjutatud eesti keeles ning sisaldab teksti 67 leheküljel, 5 peatükki, 16 joonist, 4 tabelit.

Comparing CASE Tools Enterprise Architect and DB-MAIN in the Example of Designing Scheduling Subsystem of a Healthcare Institution's Information System

Abstract

The topic of the bachelor’s thesis is the design of the visit scheduling functional subsystem of a healthcare institution’s information system, as well as the database section required for its operation, using the CASE tools Enterprise Architect and DB-MAIN. The purpose of this work was to compare the tools used from the viewpoint of designing register-based information systems – which tool is better to use in the design of such information systems, as well as in the teaching of such skills.

The first part of the work introduces the Enterprise Architect and DB-MAIN tools and the features they offer using various guide materials. In the second part, the information system subsection is designed. This is not an information system that is actually being used or is intended for production use, but rather the author’s vision of one. The system designing process is, in turn, split into three major phases: strategy, detailed analysis, and design. In the last part, the tools Enterprise Architect and DB-MAIN are compared from the viewpoint of all three phases.

The outcome of the bachelor’s thesis is a design of a visit scheduling subsystem of a healthcare institution’s information system, as well as a comparison of the Enterprise Architect and DB-MAIN tools from the viewpoint of creating such a project. The author has concluded that Enterprise Architect is preferable from the viewpoint of designing and teaching the design of register-based information systems.

The final thesis is written in the Estonian language, contains text on 67 pages, 5 chapters, 16 drawings, and 4 tables.

Lühendite ja mõistete sõnastik

Agregatsioon ehk osa-terviku seos esitab osa-terviku seost olemitüüpide vahel, kus üks olemitüüpidest esitab osa ja teine tervikut ning osaliste vahel pole eksistentsiaalset sõltuvust (Eessaar, 2015a).

„**CASE vahend** on tarkvarasüsteem, mis aitab tarkvara arendajat ühe või rohkema tarkvara arendustsükli etapi jooksul“ (Eessaar, 2014).

„**Füüsiline disain** optimeerib / häälestab loogilise disaini lahendusi konkreetsete „füüsiliste“ keskkondade jaoks, st konkreetsete riist- ja tarkvara produktide jaoks“ (Eessaar, 2012a).

Kontseptuaalne andmemudel on mittetehniline mudel, mis kirjeldab süsteemi baasandmeid pööramata tähelepanu andmebaasi realiseerimisega seotud küsimustele ega realiseerimise platvormist tulenevatele võimalustele ja piirangutele (Eessaar, 2012b).

„**Loogiline disain** tegeleb konkreetsest realisatsiooni- ja rakenduskeskkonnast sõltumatute, järelikult nende keskkondade jaoks spetsiaalselt optimeerimata lahenduste loomisega“ (Eessaar, 2012a).

Mitmesus näitab, kui palju objekte võib osaleda antud seoses (Lang, 2003).

SQL-andmebaas – andmebaas, mille haldamiseks kasutatakse SQL keelt.

UML on üldotstarbeline modelleerimiskeel keerulise (peamiselt suurte objektorienteeritud) tarkvaraprojektide spetsifitseerimiseks ja visualiseerimiseks (Vallaste, 2000).

Üldistusseos – seos, mis ühendab omavahel ülatüübi ja tema alamtüübi (Eessaar, 2012b).

Jooniste nimekiri

Tabelite nimekiri

Sisukord

Sissejuhatus

Antud töö teemaks on tervishoiuasutuse infosüsteemi vastuvõtuaegade funktsionaalse allsüsteemi ja selle tööks vajaliku andmebaasi alamosa projekteerimine kasutades Enterprise Architect (versiooni 11) ja DB-MAIN (versiooni 9.2.0) CASE (*Computer Aided System Engineering*) vahendeid. Käesoleva töö eesmärgiks on kasutatavate vahendite võrdlus registripõhiste infosüsteemide projekteerimise seisukohast. Registripõhise infosüsteemi tarkvara pakub kasutajatele andmete haldamise teenuseid ning andmebaasil on sellises infosüsteemis keskne koht. Autor soovib välja selgitada, kumba vahendit on parem kasutada selliste infosüsteemide projekteerimiseks ja selle õpetamiseks. Võrreldavad programmid on valitud põhjusel, et Enterprise Architect on kasutusel Tallinna Tehnikaülikoolis andmebaaside projekteerimise õpetamiseks ning mõlemad on kaasaegsed CASE vahendid, mida soovitatakse üliõpilastele andmebaaside kursustes.

Töö algab Enterprise Architect ja DB-MAIN tutvustusega. Selleks, et püstitatud eesmärki saavutada, on töö järgmises osas projekteeritud tervishoiuasutuse infosüsteemi vastuvõtuaegade funktsionaalne allsüsteem ning selle kasutatav rakenduse alamosa. Töös arendatakse edasi autori poolt õppeaines „Andmebaasid I“ loodud aineprojekti (Kabin, 2014). Töö viimases osas on toodud modelleerimisvahendite võrdlus, mis põhineb modelleerimisülesande lahendamisega saadud teadmistel.

Projekti teema on valitud lähtuvalt sellest, et hetkel valitseb olukord, kus erinevate meditsiiniasutuste digitaalse registreerimise süsteemid pole üksteisega ühilduvad. Seoses sellega puudub ka üks keskne vahend, mille abil patsient saaks näha korraga kõigi Eestis praktiseerivate meditsiinitöötajate vastuvõtuaegu ja järjekordi. (Eesti E-tervise Sihtasutus, 2015)

Antud töös ei ole tegemist reaalselt kasutatava süsteemiga. Autor on arendanud välja oma versiooni vastuvõtuaegade infosüsteemist spetsiaalselt selle töö jaoks. Infosüsteemi projekteerimine on jaotatud kolmeks suureks etapiks: strateegia etapp, detailanalüüsi etapp ja disaini etapp. Disaini etapp on omakorda jagatud loogiliseks ja füüsiliseks disainiks. Infosüsteemi projekteerimine viiakse läbi lihtsustatud viisil, et keskenduda valitud

modelleerimisvahendite võimaluste uurimisele. Töös luuakse mudeleid, mida üliõpilased peavad looma ka Tallinna Tehnikaülikooli andmebaaside ainetöö käigus.

Antud töö sihtrühmaks on Enterprise Architect ja DB-MAIN CASE vahendite võimalikud kasutajad. Nende hulka kuuluvad erinevate õppeasutuste töötajad, kellel on vaja otsustada, millist programmi õpetamisel kasutada.

1. Kasutatavate vahendite tutvustus

Selles peatükis esitatakse kasutatavate CASE vahendite esialgne teoreetiline tutvustus, mis põhineb mitmesugustel juhendmaterjalidel.

1.1. Enterprise Architect

Käesoleva töö tegemisel kasutab autor Sparx Systems poolt pakutavat modelleerimisvahendit Enterprise Architect versiooni 11.

Enterprise Architect on UML tööriist info- ja tarkvarasüsteemide analüüsiks, disainiks ja rakendamiseks (Sparx Systems – Enterprise Architect, 2015a). See võimaldab muuhulgas:

- Tarkvara- ja infosüsteemide projekteerimist ning mudelitest koodi genereerimist (st programmeerimise osalist automatiseerimist);
- ärianalüüsi, äriprotsesside modelleerimist, nõuete haldust;
- süsteemide modelleerimist, süsteemide arhitektuuri modelleerimist, komponentide projekteerimist, simuleerimist;
- mitmesuguste süsteemide, protsesside, andmete, tegevuste ja struktuuride visualiseerimist. (Sparx Systems – Enterprise Architect, 2015b).

Infosüsteemi projekteerimiseks kasutatakse antud töös UML diagramme, mis on jagatud kahte suurde gruppi. Järgnevalt nimetatakse vaid selliseid diagramme, mida käesolevas töös läbiviidud modelleerimisülesande lahendamiseks vaja läks.

- Struktuuridiagrammid (*Structural Diagrams*) kajastavad süsteemi struktuurilisi koostisosi, staatilisi suhteid või arhitektuuri. (Sparx Systems – Enterprise Architect, 2015c).
    - Klassidiagrammid (*Class Diagrams*) võimaldavad kirjeldada süsteemi staatilist struktuuri klassidena ja klasside vaheliste seostena ning lisada kirjeldusse juurde ka käitumusliku komponendi kirjeldades klassidega seotud operatsioone. Klassid on ühesuguste omaduste ja käitumisega objektide ehk olemite

üldistused. Andmebaasi struktuuri kirjeldamine kontseptuaalsel, loogilisel ja füüsilisel tasemel toimub UMLi kasutamise korral just klassidiagrammide abil, kusjuures loogilise ja füüsilise andmemudeli korral kirjeldatakse operatsioonide abil tabelitega seotud kitsendusi. (Sparx Systems – Enterprise Architect, 2015d)

- Käitumisdiagrammid (*Behavioral Diagrams*), esitavad süsteemi kohta dünaamilise vaate (mis sündmused alustavad protsesse, kes või mis teeb mida ja millises järjekorras). (Sparx Systems – Enterprise Architect, 2015c).
    - Tegevusdiagrammide (*Activity Diagrams*) abil kirjeldatakse protseduuride loogikat, süsteemide talitusprotsesse ning nendes toimuvaid töövooge (Fowler, 2007). Tegevusdiagrammide abil saab kirjeldada näiteks kasutusjuhtude stsenaariume või süsteemi üldist töövoogu üle erinevate kasutusjuhtude.
    - Kasutusjuhtude diagrammid (*Use Case Diagrams*) esitavad kasutusjuhte, tegutsejaid ning nendevahelisi seoseid. Iga kasutusjuht kirjeldab elementaarset äriprotsessi, mis saab alguse mingist süsteemile olulisest sündmusest, toimub ühel ajal ja kohas, enamasti ühe tegutseja poolt ja aitab sellel tegutsejal talle seatud eesmärke täita. Kasutusjuhtude abil saab kirjeldada süsteemile esitatavaid funktsionaalseid nõudeid. Kasutusjuhtude diagramm on vaid osa kasutusjuhtude mudelist. Lisaks diagrammidele, mis on kui raamatu sisukord, tuleb ka esitada kasutusjuhtude tekstilised ja struktureeritud kirjeldused, mis on kui raamatu sisu.
    - Seisundidiagrammid (*State Machine Diagrams*) võimaldavad kirjeldada süsteemi jaoks huvipakkuvate objektide elutsükleid (ajast mil nad muutuvad süsteemi jaoks oluliseks ehk sünnivad kuni ajani, mil nad lõpetavad süsteemi silmis oma eksistentsi ehk surmani), näidates sündmuseid, mis viivad neid objekte ühest seisundist teise, tingimusi, mis peavad olema selleks täidetud ning süsteemi realisatsiooni sellistele sündmustele. (Eessaar, 2015a)

Antud töös on loodud tegevusdiagramm, kasutusjuhtude diagramm, vastuvõtuaja seisundidiagramm ning kontseptuaalne eskiismudel, olemi-suhte diagramm ja andmebaasi füüsiline andmemudel (SQL-andmebaasi jaoks) klassidiagrammi baasil. Enterprise Architect võimaldab luua rohkem erinevat tüüpi diagramme, kuid selle töö jaoks neid vaja ei läinud.

Mudelielemente (näiteks olemitüübid, kasutusjuhud) saab jagada ka pakettidesse. Kui viidata elemendile, mis ei kuulu samasse paketti, kuhu diagramm kuulub, on enne elemendi nime märgitud paketi nimi, kuhu see element kuulub. Näidet selle kohta saab näha jooniselt 7. Pakettide kasutamisel on samasugune eesmärk nagu failisüsteemis kataloogide kasutamisel –

vältida elementide nimekonflikte ja hõlbustada nende ülesleidmist. Pakettide abil saab edukalt esitada infosüsteemide allsüsteeme.

Enterprise Architect vahendis on võimalik teha teisendust kontseptuaalsest andmemudelist füüsilisele andmemudelile, kuid selleks peab kasutaja läbima üsna pika protsessi. Mudelite kooskõla on samuti võimalik kontrollida; vaadatakse näiteks, kas kõik mudelis kasutatavad ja viidatud elemendid, on ka reaalselt viidatavas asukohas olemas. Näiteks, kas füüsilisel andmemudelil kajastatav andmebaas on ka reaalselt komponentide vaates olemas.

1.2. DB-MAIN

Antud töö tegemisel kasutab autor REVER S.A. poolt pakutavat modelleerimisvahendit DB-MAIN versiooni 9.2.0. See võimaldab luua:

- analüüsimudeleid (nt olemi-suhte diagramm, UML klassidiagramm, UML tegevusdiagramm, UML kasutusjuhtude eskiismudel);
- loogilise disaini mudeleid (nt relatsiooniline);
- füüsilise disaini mudeleid (REVER S.A. – DB-MAIN, 2015a).

Arendusprotsessi sammud, mille läbiviimist DB-MAIN toetab: nõuete analüüs sh andmebaasile esitatavate nõudmiste kirjeldamine kontseptuaalse andmemudeli abil, skeemi integratsioon, loogiline disain, füüsilise disain, skeemi optimeerimine, koodi genereerimine (REVER S.A. – DB-MAIN, 2015b).

Antud töös luuakse analüüsimudeleid ning loogiline ja füüsiline andmemudel. Autor loob tegevusdiagrammi, kasutusjuhtude diagrammi, vastuvõtuaegade registri kontseptuaalse andmemudeli olemi-suhte diagrammi, andmebaasi loogilise ja füüsilise andmemudeli (SQL-andmebaasi jaoks).

Loogilises ja füüsilises andmemudelis kasutatakse vaikimisi järgmist notatsiooni: tabelite vahelisi viiteid kujutatakse noolega, suunaga tabelile, kus asub seoses osalev primaarvõti. Näidet nende kohta saab näha vastavalt jooniselt 12 ja 14.

DB-MAIN võimaldab kontrollida mudelite kooskõla. Näiteks saab kontrollida, et iga mudeli element oleks unikaalse nime ja koodiga ning et need ei oleks pikemad kui mudelis maksimaalselt lubatud. Loogilisel ja füüsilisel andmemudelil saab kontrollida, et igal tabelil oleks määratud veerge ja igale veerule oleks määratud andmetüüp. Sama kontrolli tehakse automaatselt ka enne mudelite teisendamist ja skriptide loomist.

Kontseptuaalses andmemudelis saab seoste mitmesust kuvada numbriga Nimetatud andmemudeli saab DB-MAIN vahendis teha klassidiagrammi baasil. Näidet selle kohta saab näha jooniselt 8.

2. Strateegia etapp

Infosüsteemi projekteerimine on jaotatud kolme etappi: strateegia etapp, detailanalüüsi etapp, disaini etapp. Käesolev peatükk käsitleb strateegia etappi. Töö järgib andmebaaside õppeainetes kasutatavat projekti struktuuri.

2.1. Terviksüsteemi üldvaade

Töös käsitletakse interneti kaudu arstile vastuvõtuaja broneerimisega seonduvat: registraatori poolt vastuvõtuaja väljapakkumist ja patsiendi poolt vastuvõtuaja broneerimist. Järgnevalt on autor sõnastanud tervishoiuasutuse eesmärgid ja tervishoiuasutuse infosüsteemi eesmärgid, loetlenud lausendid, põhiobjektid, põhilised sündmused, tegutsejad ja tegutsemiskohad ning jaotanud terviksüsteemi allsüsteemideks. Antud töös ei vaadelda infosüsteemi suhtlust e-tervise infosüsteemidega, sest see pole selle töö eesmärkide saavutamise jaoks vajalik. Nii nagu kogu selle projekti puhul, on ka üldvaate puhul tegemist lihtsustusega, sest töö eesmärgiks on võrrelda CASE vahendeid ja demonstreerida nende kasutusvõimalusi, mitte luua reaalset e-tervise süsteemi osa.

2.1.1. Organisatsiooni eesmärgid

Tervishoiuasutuse kui organisatsiooni eesmärgid on järgmised.

- Kõrgetasemeliste tervishoiu-, sotsiaal- ja rehabilitatsiooniteenuste ning nendega seotud muude teenuste osutamine Eesti Vabariigi kodanikele ja teistele Eesti Vabariigis viibivatele isikutele seadustes ja lepingutes ettenähtud tingimustes ja ulatuses („Sihtasutus Jõgeva Haigla põhikiri“).

2.1.2. Infosüsteemi eesmärgid

Tervishoiuasutuse infosüsteemi eesmärgid on järgmised.

- Saada ülevaade tervishoiuasutusega seotud isikutest (patsiendid ja töötajad).
- Saada ülevaade tervishoiuasutuse poolt pakutavatest teenustest.
- Võimaldada arstide vastuvõttudele elektrooniliselt registreerida.
- Saada ülevaade patsientide haigusjuhtudest.
- Saada ülevaade patsientidele tehtud uuringutest ning selleks võetud proovidest.
- Võimaldada haigusjuhtusid elektrooniliselt koostada.
- Saada ülevaade töötajate töökoormusest ja töö tulemustest.
- Saada ülevaade tervishoiuasutuse struktuurist.
- Saada ülevaade tervishoiuasutuse poolt väljastatud arvetest ning tervishoiuasutusele saadetud arvetest.
- Saada ülevaade tervishoiuasutuse käsutuses olevatest varadest.
- Saada ülevaade tervishoiuasutuse struktuuriüksuste väljastatud ostutellimustest.
- Saada ülevaade tervishoiuasutuse käsutuses olevatest ruumidest.
- Võimaldada elektroonilist dokumendihaldust.
- Tagada infosüsteemi toimimist, võttes kasutusele ja hallates selleks vajalikke klassifikaatoreid.

2.1.3. Lausendid

Antud infosüsteemi tausta struktureeritud kirjeldus lausenditena.

- Uudistaja on registreerimata kasutaja.
- Patsient on registreeritud kasutaja.
- Patsient on isik.
- Patsiendile osutatakse tervishoiuasutuses teenust.
- Töötaja on isik.
- Arstil on eriala.
- Erialal on kood.
- Erialal on nimetus.

- Registraator on tervishoiuasutuse töötaja.
- Arst on tervishoiuasutuse töötaja.
- Raamatupidaja on tervishoiuasutuse töötaja.
- Personalitöötaja on tervishoiuasutuse töötaja.
- Tervishoiuasutuse juhataja on tervishoiuasutuse töötaja.
- Tervishoiuasutuse omanik on kohalik omavalitsus.
- Registraator haldab vastuvõtu andmeid.
- Raamatupidaja haldab ettevõtte finantsasju.
- Personalitöötaja haldab töötajate andmeid.
- Arst on tervishoiutöötaja.
- Arst võtab patsiente vastu.
- Arst suunab patsiendi uuringule.
- Õde võtab patsiendilt uuringu jaoks proove.
- Õde on tervishoiutöötaja.
- Laboritöötaja töötleb patsiendilt uurimiseks võetud materjali.
- Laboritöötaja on tervishoiutöötaja.
- Arst koostab patsiendi haigusjuhu.
- Arstil on vastuvõtuaeg.
- Vastuvõtt toimub ruumis.
- Patsient registreerub vastuvõtuajale.
- Tervishoiuasutuse töötaja töötab struktuuriüksuses.
- Osakond on struktuuriüksus.
- Struktuuriüksusele esitatakse arve.
- Struktuuriüksus tasub arve.
- Struktuuriüksus väljastab arve.
- Tervishoiuasutus omab põhivara.
- Struktuuriüksus väljastab ostutellimuse.
- Tervishoiuasutuse valduses on hoone.
- Hoone sisaldab ruume.
- Tekib uus dokument.
- Haigusjuhu kirjeldus on dokument.
- Organisatsioon on lepinguliselt seotud tervishoiuasutusega.

- Osapool on isik, isikute grupp või organisatsioon.
- Osapool on seotud tervishoiuasutusega.
- Võetakse kasutusele uus klassifikaator.

2.1.4. Põhiobjektid

Tervishoiuasutuse tegevusega seotud põhiobjektid on järgmised.

- Töötaja
- Patsient
- Vastuvõtuaeg
- Haigusjuht
- Proov
- Uuring
- Struktuuriüksus
- Arve
- Põhivara
- Ostutellimus
- Ruum
- Dokument
- Isik
- Organisatsioon
- Osapool
- Klassifikaator

2.1.5. Põhilised sündmused

Põhilised sündmused tervishoiuasutuse tegevuses on järgmised.

- Tervishoiuasutus saab teada isikust, kes hakkab seal töötama või kellele hakatakse teenust osutama.
- Patsient pöördub terviseprobleemiga või selle kahtlusega arsti poole.
- Arst kahtlustab patsiendil terviseprobleemi.
- Laboritöötaja saab info uuringu vajadusest.

- Arstile saabuvad uuringu vastused.
- Arst esitab arve tervishoiuasutuse raamatupidamisele.
- Kuulutatakse välja konkurss vabadele ametikohtadele.
- Tervishoiuasutuse nõukogule esitatakse taotlus uue struktuuriüksuse loomiseks.
- Tervishoiuasutuse nõukogule esitatakse taotlus struktuuriüksuse töö lõpetada.
- Struktuuriüksus osutab patsiendile teenust.
- Struktuuriüksus on ostnud tervishoiuasutuse tööks kaupu või teenuseid.
- Tervishoiuasutuse käsutusse saabub uus vara.
- Tervishoiuasutuse käsutuses olevat vara enam ei eksisteeri, st vara on hävitatud või hävinud, kadunud jms.
- Struktuuriüksus vajab oma tööks mingit vara.
- Valminud on tervishoiuasutuses uus osakond.
- Lõppenud on tervishoiuasutuse osakonna või selle osa renoveerimine.
- Tervishoiuasutusele saadetakse või tervishoiuasutuse siseselt tekib uus dokument.
- Võetakse kasutusele uus klassifikaatori väärtus.

2.1.6. Tegutsejad

Tervishoiuasutuse infosüsteemiga tegelevad järgmised tegutsejad.

- Uudistaja (registreerimata kasutaja)
- Patsient (registreeritud kasutaja)
- Registraator
- Arst
- Õde
- Laboritöötaja
- Raamatupidaja
- Personalitöötaja
- Tervishoiuasutuse juhataja
- Omanik

2.1.7. Asukohad

Tegutsejad teostavad oma operatsioone järgmistes asukohtades.

- Patsiendid ja uudistajad kasutavad isiklikku ja tööarvutit ning avalikke internetipunkte.
- Registraatorid töötavad tervishoiuasutuses eraldatud ruumides. Ühes ruumis võib olla mitu registraatorit. Igaühel neist on eraldi tööarvuti, seega oma töökoht.
- Arstid töötavad tervishoiuasutuses. Igaühel neist on eraldi kabinet ja tööarvuti.
- Õed töötavad tervishoiuasutuses. Neil on mitme peale üks tööarvuti.
- Laboritöötajad töötavad laboris oma tööarvutiga.
- Raamatupidaja, personalitöötaja ja tervishoiuasutuse juhataja töötavad tervishoiuasutuses spetsiaalselt ettenähtud ruumides. Igaühele neist on oma tööarvuti.
- Omanik töötab kohaliku omavalitsuse ruumides, kasutatakse tööarvuteid.

2.1.8. Terviksüsteemi tükeldus allsüsteemideks

Järgnevates alajaotustes on kirjeldatud tervishoiuasutuse infosüsteemi allsüsteeme.

2.1.8.1. Pädevusalad

Organisatsiooni sisesed pädevusalad.

- Registraatori pädevusala
- Arsti pädevusala
- Õe pädevusala
- Laboritöötaja pädevusala
- Raamatupidaja pädevusala
- Personalitöötaja pädevusala
- Tervishoiuasutuse juhataja pädevusala
- Omaniku pädevusala

Organisatsiooni välised pädevusalad.

- Uudistaja pädevusala
- Patsiendi pädevusala (patsient kui klient)

2.1.8.2. Funktsionaalsed allsüsteemid

Sisulised allsüsteemid (seotud organisatsiooni põhitegevusega).

- Patsientide funktsionaalne allsüsteem
- Vastuvõtuaegade funktsionaalne allsüsteem
- Haigusjuhtude funktsionaalne allsüsteem
- Proovide funktsionaalne allsüsteem
- Uuringute funktsionaalne allsüsteem

Administratiivsed allsüsteemid (võivad olla kasutusel paljudes erinevates organisatsioonides, mille tegevusalad ja eesmärgid on erinevad).

- Osapoolte funktsionaalne allsüsteem
- Isikute funktsionaalne allsüsteem
- Organisatsioonide funktsionaalne allsüsteem
- Töötajate funktsionaalne allsüsteem
- Struktuuriüksuste funktsionaalne allsüsteem
- Arvete funktsionaalne allsüsteem
- Põhivara arvestuse funktsionaalne allsüsteem
- Ostutellimuste funktsionaalne allsüsteem
- Ruumide funktsionaalne allsüsteem
- Dokumentide funktsionaalne allsüsteem
- Klassifikaatorite funktsionaalne allsüsteem

2.1.8.3. Registrid

Sisulised registrid (seotud organisatsiooni põhitegevusega).

- Patsientide register
- Vastuvõtuaegade register
- Haigusjuhtude register
- Proovide register
- Uuringute register

Administratiivsed registrid (võivad olla kasutusel paljudes erinevates organisatsioonides, mille tegevusalad ja eesmärgid on erinevad).

- Osapoolte register
- Isikute register
- Organisatsioonide register
- Töötajate register
- Struktuuriüksuste register
- Arvete register
- Põhivara register
- Ostutellimuste register
- Ruumide register
- Dokumentide register
- Klassifikaatorite register

2.2. Vastuvõtuaegade allsüsteem

Järgnevalt on täpsemalt uuritud vastuvõtuaegade allsüsteemi patsiendi ja registraatori pädevusala piires. Teisi pädevusalasid ei ole uuritud, sest infosüsteemi projekteerimine on toodud antud töös lihtsustatud kujul ja modelleerimisvahendite võrdlust see ei muudaks. Välja on toodud vastuvõtuaegade allsüsteemi eesmärgid, kasutavad pädevusalad ja registrid. Diagrammidena on toodud vastuvõtuaegade allsüsteemi põhiprotsessi tegevusdiagramm ja kasutusjuhtude diagramm. Lisaks on kirjeldatud kasutusjuhtusid kõrgtaseme formaadis.

2.2.1. Eesmärgid

Vastuvõtuaegade allsüsteemi eesmärgid on järgmised.

- Võimaldada registraatoritel elektrooniliselt registreerida vastuvõtuaegu.
- Võimaldada registraatoritel saada ülevaade arstide vastuvõtuaegadest ning nendele registreerinud patsientidest.
- Võimaldada registraatoritel tühistada vastuvõtuaegu.
- Võimaldada patsiendil elektrooniliselt registreeruda konkreetsetele vastuvõtuaegadele.

- Võimaldada patsiendil saada ülevaade vastuvõtuaegadest, kuhu ta on registreerunud.
- Vähendada 50% aega, mille jooksul peavad patsiendid vastuvõtuajal vastuvõtu kabineti ukse taga ootama.

2.2.2. Allsüsteemi kasutavad pädevusalad

Vastuvõtuaegade allsüsteemi kasutavad järgmised pädevusalad.

- Registraatori pädevusala
- Patsiendi pädevusala

2.2.3. Allsüsteemi poolt kasutatavad registrid

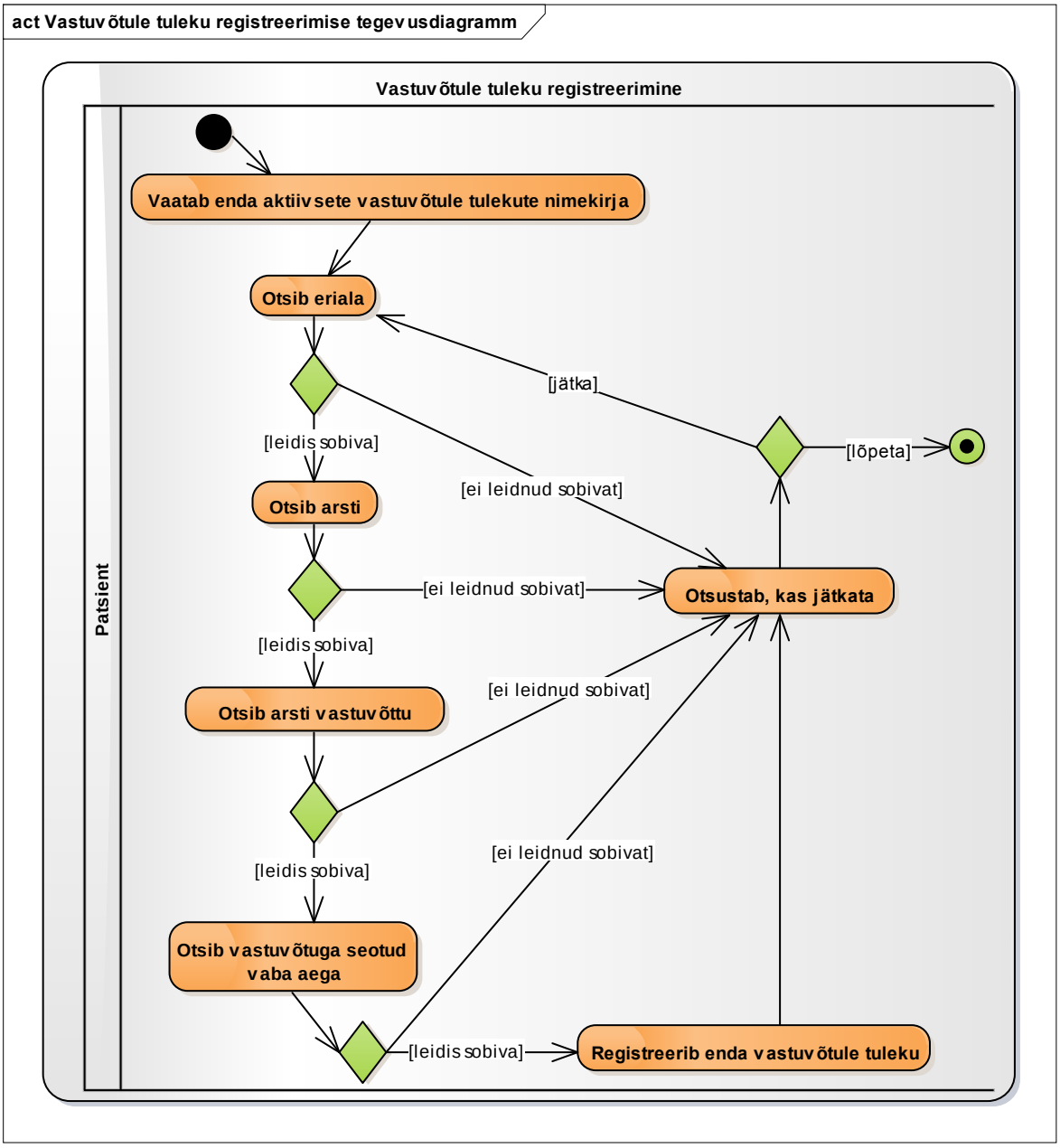
Allsüsteem teenindab vastuvõtuaegade registrit.

Allsüsteem kasutab järgmiseid registreid.

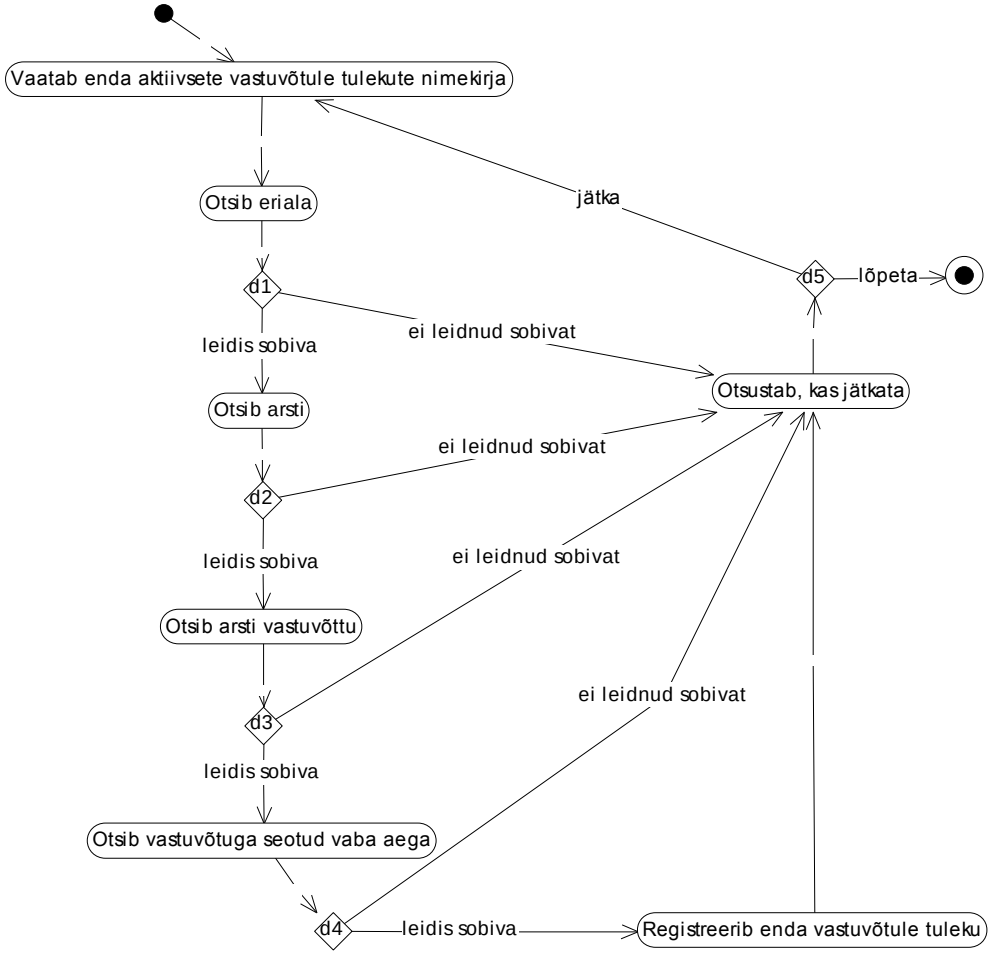
- Isikute register
- Patsientide register
- Töötajate register
- Klassifikaatorite register
- Ruumide register

2.2.4. Allsüsteemi põhiprotsessi tegevusdiagramm

Tegevusdiagrammi on põhimõtteliselt võimalik koostada nii Enterprise Architect kui ka DB-MAIN abil. Joonisel 1 on vastuvõtuaegade allsüsteemi ühe põhiprotsessi tegevusdiagramm koostatud Enterprise Architect abil. Joonisel 2 on sama diagramm koostatud DB-MAIN abil.



Joonis 1. Enterprise Architect abil koostatud vastuvõtule tuleku registreerimise tegevusdiagramm

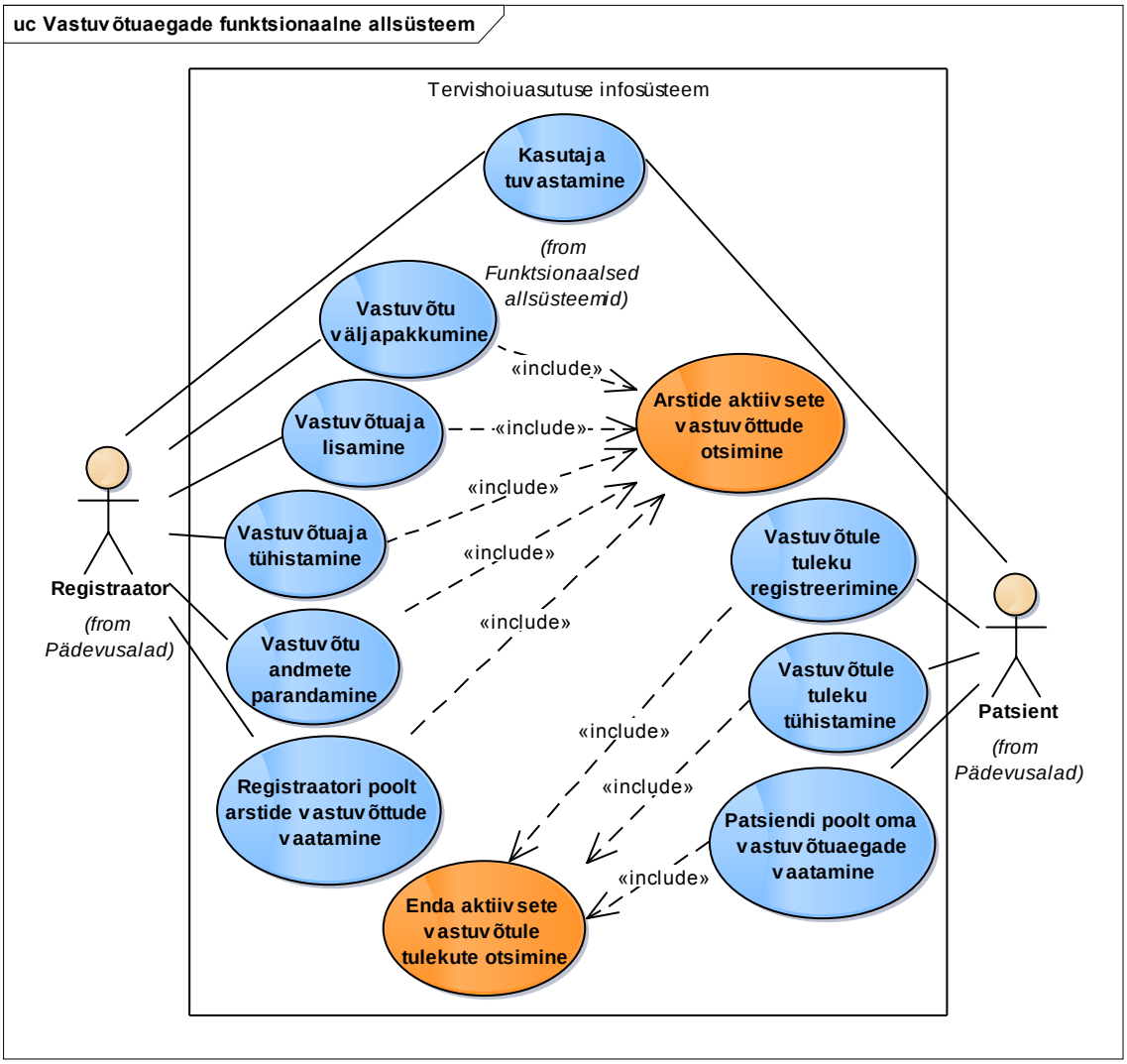


Joonis 2. DB-MAIN abil koostatud vastuvõtule tuleku registreerimise tegevusdiagramm

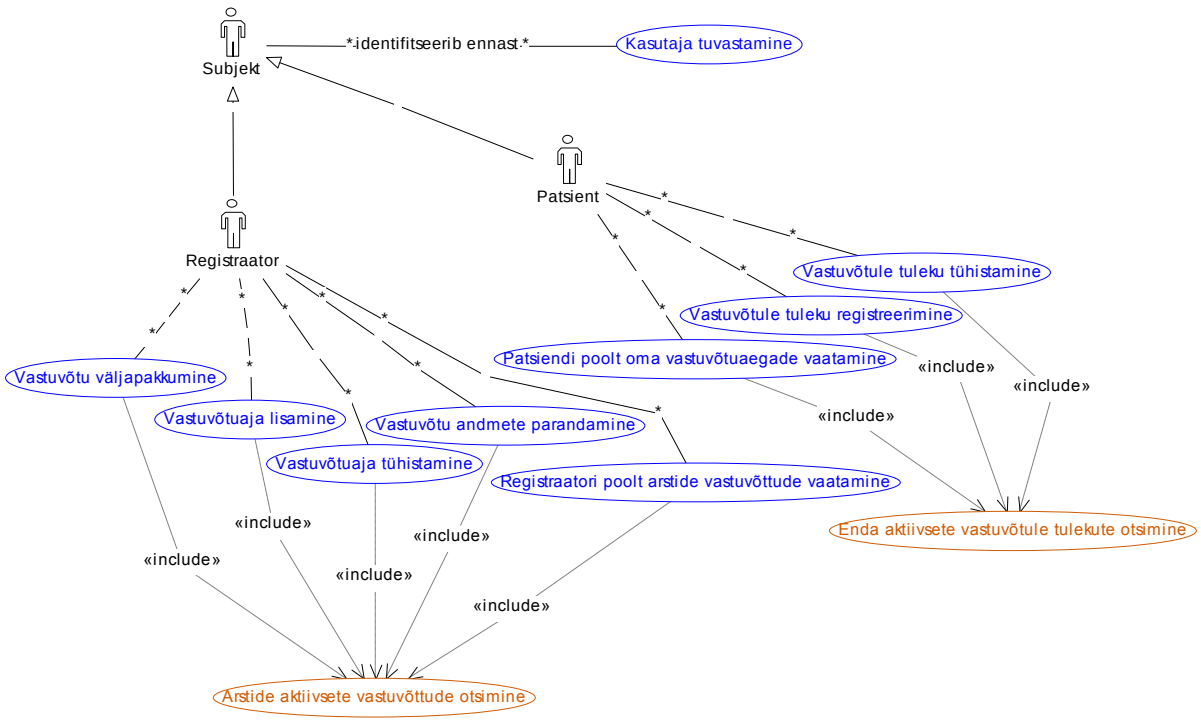
2.2.5. Allsüsteemi kasutusjuhtude eskiismudel

Kasutusjuhtude eskiismudelit on põhimõtteliselt võimalik koostada nii Enterprise Architect kui ka DB-MAIN abil. Joonisel 3 on vastuvõtuaegade allsüsteemi kasutusjuhtude diagramm koostatud Enterprise Architect abil. Joonisel 4 on sama diagramm koostatud DB-MAIN abil.

Sinisega on tähistatud põhikasutusjuhud. Oranžiga on tähistatud abistavad kasutusjuhud (sisuliselt kasutusjuhu fragmendid), mis on kirja pandud selleks, et mitte kirjeldada mitmekordselt erinevates kasutusjuhtudes esinevat ühesugust funktsionaalsust.



Joonis 3. Enterprise Architect abil koostatud vastuvõtuaegade allsüsteemi kasutusjuhtude diagramm



Joonis 4. DB-MAIN abil koostatud vastuvõtuaegade allsüsteemi kasutusjuhtude diagramm

Järgnevalt esitatakse kõrgtaseme formaadis kasutusjuhtude tekstikirjeldused. Selliseid kirjeldusi saaks mõlemas CASE vahendis mudelisse kirja panna kasutusjuhu mudelielemendiga seotud kommentaarina.

<u>Kasutusjuht:</u> <u>Kasutaja tuvastamine</u>

Tegutsejad: Patsient, registraator

Kirjeldus: Tegutseja identifitseerib ennast, sisestades kasutajanime, parooli ja oma rolli süsteemis. Süsteem autendib tegutseja, st kontrollib tegutseja väidetavat identiteeti. Kui tegutseja on identifitseeritud, siis lubatakse tegutseja süsteemi siseneda, vastasel juhul mitte.

Märkus. Kasutusjuhtu „Kasutaja tuvastamine“ kasutatakse kõikides funktsionaalsetes allsüsteemides. See kirjeldab süsteemi funktsionaalsuse allsüsteemide ülest (turvalisuse) aspekti.

Kasutusjuht: Arstide aktiivsete vastuvõttude otsimine

Tegutsejad: Registraator

Kirjeldus: Registraator otsib arstidega seotud aktiivseid vastuvõtte. Need on vastuvõtud, mis toimuvad vastuvõtu lisamise ajaga samal kuupäeval või tulevikus.

Kasutusjuht: Vastuvõtu väljapakkumine

Tegutsejad: Registraator

Kirjeldus: Registraator pakub välja kuupäevad, millal arst saab patsiente vastu võtta ning anda konsultatsiooni tervisega seotud küsimustes. Iga vastuvõtuga koos võib ta registreerida ka info, kas konkreetsele vastuvõtule on vajalik saatekiri, kas vastuvõtt on tasuline.

Kasutusjuht: Vastuvõtuaja lisamine

Tegutsejad: Registraator

Kirjeldus: Registraator pakub välja vastuvõtu kuupäeva konkreetsed kellaajad, millal arstid saavad patsiente vastu võtta. Igale kellaajale saab konsultatsiooniks registreeruda maksimaalselt üks patsient.

Kasutusjuht: Vastuvõtuaja tühistamine

Tegutsejad: Registraator

Kirjeldus: Registraator kustutab vastuvõtuaja, sest arst ei saa mingil tööalasel või isiklikul põhjusel sellel ajal patsiente vastu võtta. Kui patsient on juba ennast sellele ajale registreerinud, siis saadab süsteem patsiendile vastava teate meilile.

Kasutusjuht: Vastuvõtu andmete parandamine

Tegutsejad: Registraator

Kirjeldus: Registraator parandab vastuvõtuga seotud infot saatekirja vajalikkuse kohta, arsti või ruumi, kus vastuvõtt toimub.

Kasutusjuht: Registraatori poolt arstide vastuvõttude vaatamine

Tegutsejad: Registraator

Kirjeldus: Registraator vaatab kõiki arstidega seotud vastuvõtte. Süsteem esitab erinevalt vastuvõtud, mis on veel toimumata ja vastuvõtud, mis on juba toimunud. Registraator valib konkreetse vastuvõtu ning vaatab sellega seotud vastuvõtuaegu, nendele registreerunud patsiente.

Kasutusjuht: Enda aktiivsete vastuvõtule tulekute otsimine

Tegutsejad: Patsient

Kirjeldus: Patsient otsib endaga seotud aktiivseid vastuvõtule tulekuid. Need tulekud on seotud vastuvõtuaegadega, mille algus on tulevikus või mis algavad vaatamise hetkel.

Kasutusjuht: Vastuvõtule tuleku registreerimine

Tegutsejad: Patsient

Kirjeldus: Patsient valib arsti, kelle vastuvõtule ta soovib tulla. Kui arstil on mõni tulevikus toimuv vastuvõtt, siis saab patsient valida selle vastuvõtu ning näha selle vastuvõtuga seotud vabu vastuvõtuaegu. Patsient saab ennast registreerida vabale vastuvõtuajale.

Kasutusjuht: Vastuvõtule tuleku tühistamine

Tegutsejad: Patsient

Kirjeldus: Patsient saab kustutada oma registreerumise vastuvõtuajale.

Kasutusjuht: Patsiendi poolt oma vastuvõtuaegade vaatamine

Tegutsejad: Patsient

Kirjeldus: Patsient vaatab kõiki vastuvõtuaegu, millele ta on ennast registreerinud. Süsteem esitab erinevalt vastuvõtuajad, mis algavad tulevikus ja vastuvõtuajad, mis on juba möödunud.

2.3. Vastuvõtuaegade register

Järgnevalt on uuritud vastuvõtuaegade allsüsteemi toimimiseks vajalikke registreid (eeskätt vastuvõtuaegade registrit). Vastuvõtuaegade registri kohta on loetletud selle eesmärgid, seda kasutavad pädevusalad ja seda teenindavad funktsionaalsed allsüsteemid ning infovajadused, mida selles registris olevad andmed aitavad lahendada. Paljud infovajadused on sellised, mille rahuldamiseks läheb tegelikult vaja andmeid mitmest registrist. Lisaks on vastuvõtuaegade registri seosed teiste registritega ja ärireeglid, millega süsteemi (sh andmebaasi arendamisel) arvestada tuleb. Diagrammina on toodud vastuvõtuaegade registri kontseptuaalne eskiismudel.

2.3.1. Eesmärgid

Vastuvõtuaegade registri eesmärk on säilitada informatsiooni vastuvõtuaegade kohta sellises mahus, et oleks tagatud vastuvõtuaegade funktsionaalses allsüsteemis defineeritud eesmärkide täitmine.

2.3.2. Registrit kasutavad pädevusalad

Vastuvõtuaegade registrist loevad andmeid ja muudavad andmeid patsiendid ja registraatorid.

2.3.3. Registrit teenindavad funktsionaalsed allsüsteemid

Vastuvõtuaegade registri andmeid kasutab ja uuendab vastuvõtuaegade funktsionaalne allsüsteem.

2.3.4. Infovajadused

Kasutajatel on vaja järgmiseid andmeid.

- Nimekiri konkreetse arsti vastuvõttudest.
- Nimekiri konkreetse vastuvõtuga seotud vastuvõtuaegadest.
- Nimekiri konkreetse arsti vabadest vastuvõtuaegadest.
- Nimekiri vastuvõtuaegadest, millele konkreetne patsient on registreerunud.

2.3.5. Seosed teiste registritega

Vastuvõtuaegade register on seotud teiste registritega järgmiselt.

- **Töötajate register** – Töötajate registriga on vastuvõtuaeg seotud olemitüüpide **Töötaja** ja **Arst** kaudu. Registraator haldab vastuvõtu andmeid. Vastuvõtuajal annab arsti patsiendile konsultatsiooni.
- **Patsientide register** – Patsientide registriga on vastuvõtuaeg seotud olemitüübi **Patsient** kaudu. Patsient registreerib ennast vastuvõtuajale.

- **Ruumide register** – Ruumide registriga on vastuvõtuaeg seotud olemitüübi **Ruum** kaudu. Vastuvõtt toimub kindlas ruumis.
- **Klassifikaatorite register** – Igal vastuvõtul on tüüp, mis on klassifikaator.

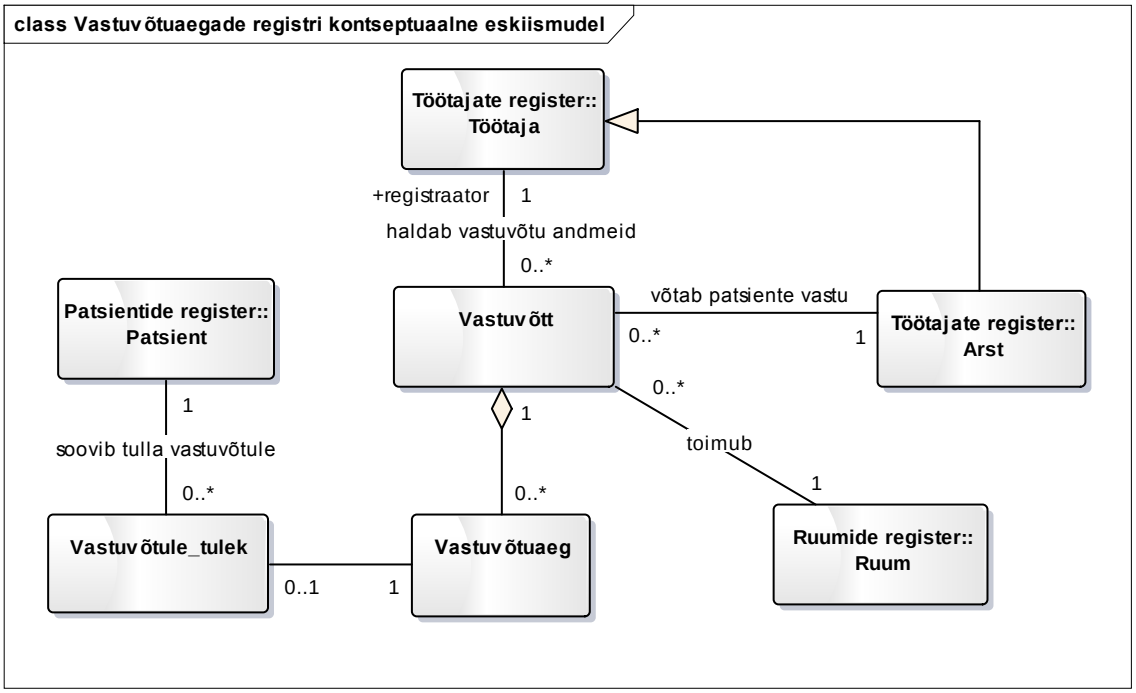
2.3.6. Ärireeglid

Ärireeglid, millega tuleb arvestada, on järgmised.

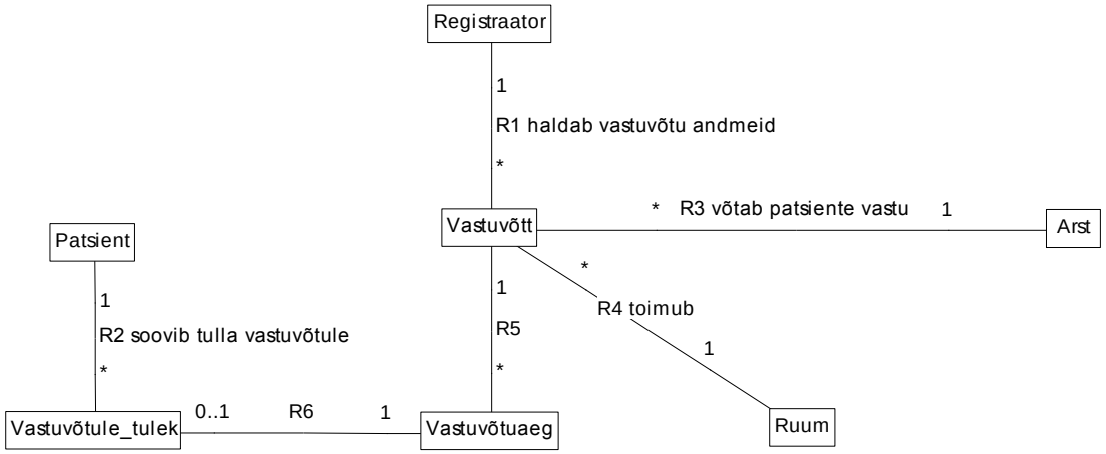
- Kõik arstid ja patsiendid peavad olema Eesti Vabariigi residendid, kellel on seega Eesti isikukood.
- Igal arstil on null või rohkem vastuvõttu.
- Iga vastuvõtuga on seotud null või rohkem vastuvõtuaega.
- Ühelgi vastuvõtul ei tohi olla kaks või rohkem samasugust vastuvõtuaega.
- Igale vastuvõtuajale saab ennast registreerida maksimaalselt üks patsient.
- Ei ole piiranguid, kui mitmele vastuvõtuajale võib patsient ennast samaaegselt registreerida.
- Ei saa lisada vastuvõttu, mis toimub varasemal kuupäeval, kui on vastuvõtu lisamise aeg.
- Ei saa lisada vastuvõtuaega, mis toimub samal kuupäeval kui vastuvõtuaja lisamise kuupäev, kuid varasemal kellaajal, kui vastuvõtuaja lisamise kellaaeg.
- Ei saa tühistada vastuvõttu, mis toimub varasemal kuupäeval, kui on vastuvõtu tühistamise aeg.
- Ei saa tühistada vastuvõtuaega, mis toimub varasemal kuupäeval, kui on vastuvõtuaja tühistamise aeg.
- Ei saa ennast registreerida vastuvõtuajale, mis algab varasemal ajal, kui on vastuvõtule tuleku registreerimise aeg.
- Ei saa tühistada enda vastuvõtule tulekut, mis algab varasemal ajal, kui on vastuvõtule tuleku tühistamise aeg.
- Ühes ja samas ruumis ei saa samal ajaperioodil toimuda osaliselt või täielikult kattuvaid vastuvõtte.
- Arstil ei saa olla erinevates ruumides vastuvõtte, mille ajaperiood osaliselt või täielikult kattub.
- Patsiendil ei saa olla kahte vastuvõttu, mille ajaperiood osaliselt või täielikult kattub.

2.3.7. Registri kontseptuaalne eskiismudel

Kontseptuaalset eskiismudelit on põhimõtteliselt võimalik koostada nii Enterprise Architect kui ka DB-MAIN abil. Joonisel 5 on Enterprise Architect abil koostatud vastuvõtuaegade registri kontseptuaalne eskiismudel, mis on esimene ja lihtsustatud versioon kontseptuaalse andmemudeli olemi-suhte diagrammist. Joonisel 6 on sama eskiismudel koostatud DB-MAIN abil. Mõlemas süsteemis saab soovi korral peita klassidiagrammi atribuute.



Joonis 5. Enterprise Architect abil koostatud vastuvõtuaegade registri kontseptuaalne eskiismudel



Joonis 6. DB-MAIN abil koostatud vastuvõtuaegade registri kontseptuaalne eskiismudel

3. Detailanalüüsi etapp

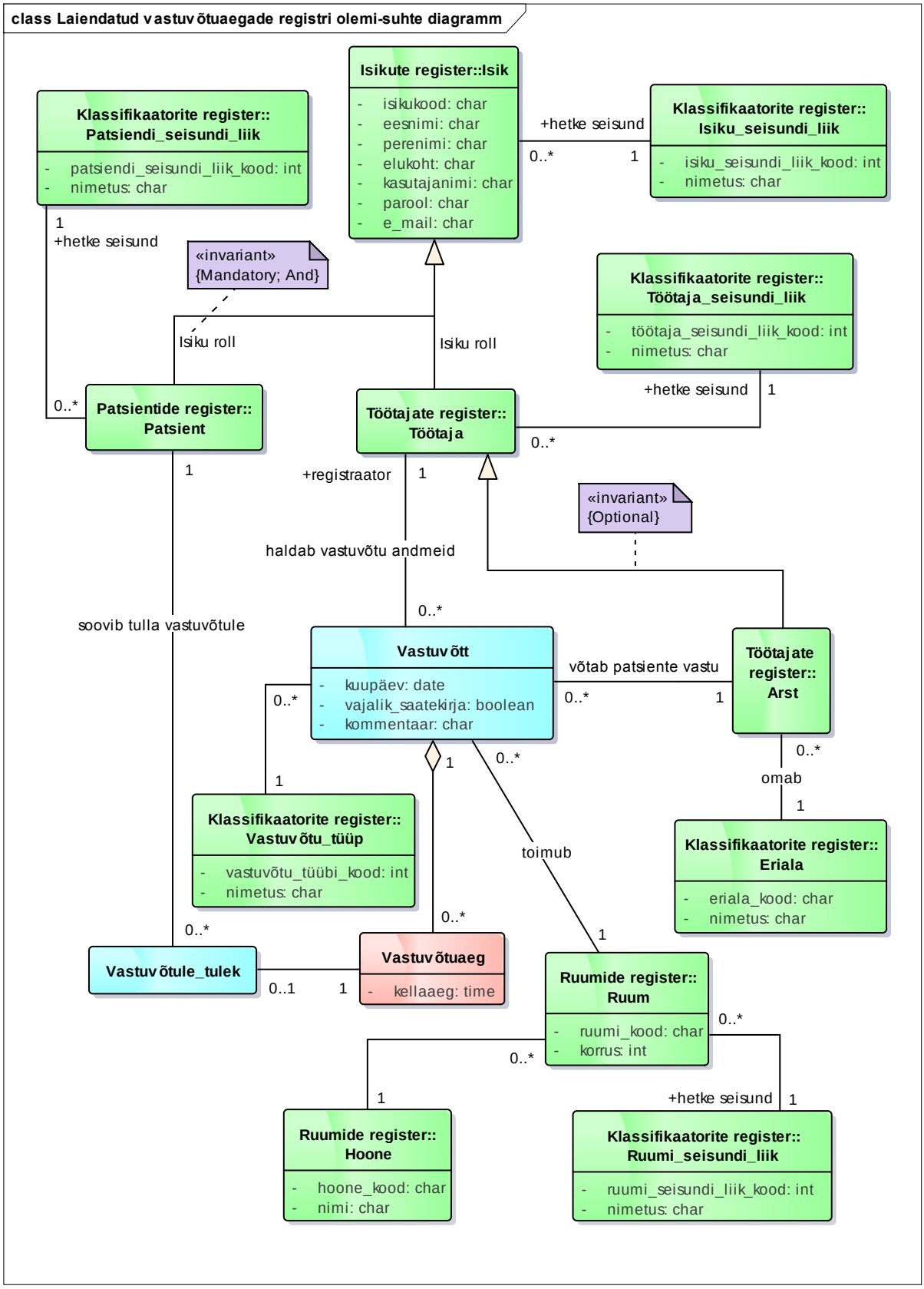
Selles peatükis on käsitletud infosüsteemi projekteerimise teist sammu: detailanalüüsi etappi. Diagrammina on toodud kontseptuaalse andmemudeli olemi-suhte diagramm, vastuvõtuaja seisundidiagramm ja tabelina on välja toodud CRUD maatriks.

3.1. Kontseptuaalne andmemudel

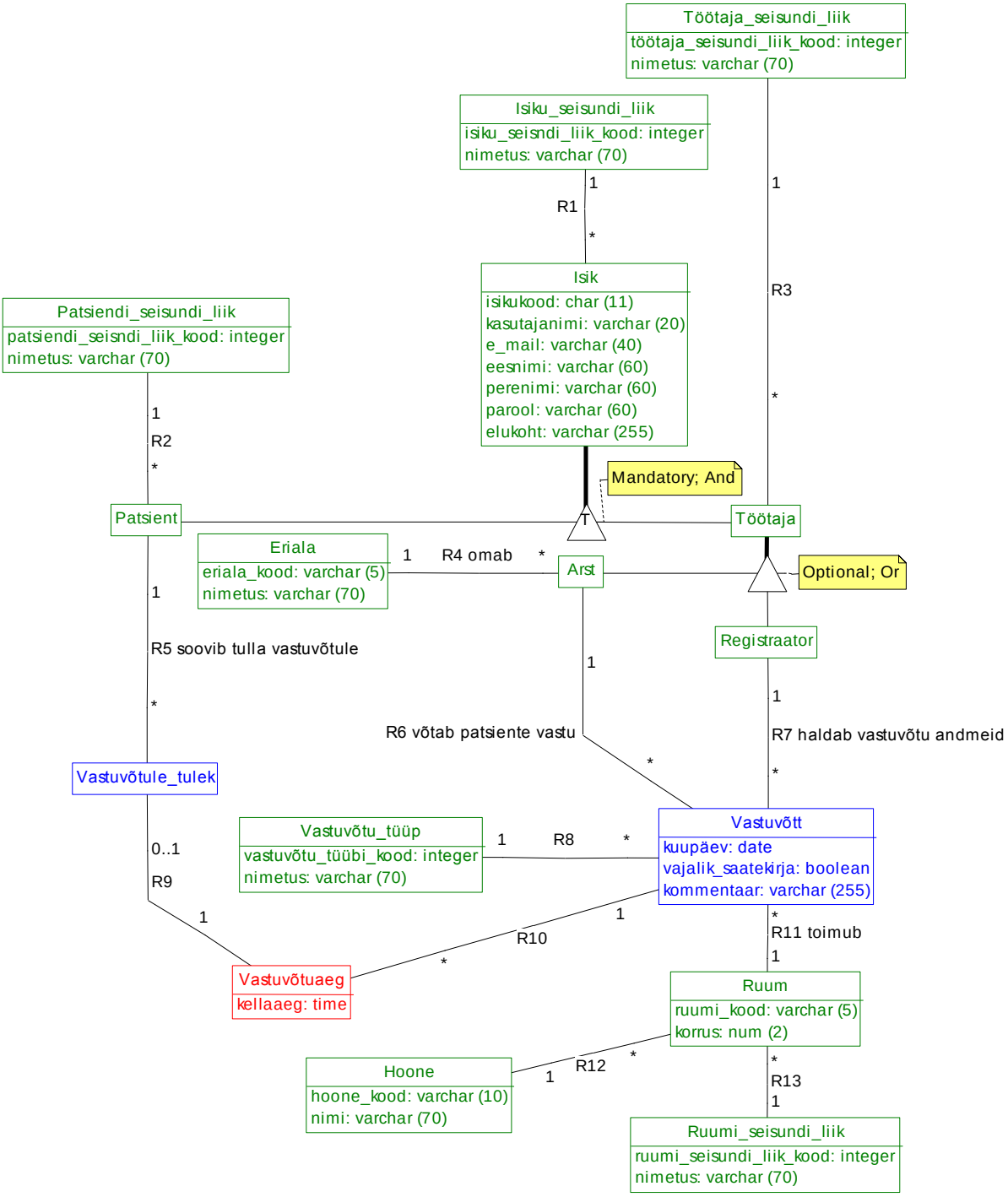
3.1.1. Olemi-suhte diagramm

Järgnevalt on toodud kontseptuaalse andmemudeli olemi-suhte diagramm, mida on põhimõtteliselt võimalik koostada nii Enterprise Architect kui ka DB-MAIN abil. Sellele on kantud ka teistesse registritesse kuuluvaid olemitüüpe, millele vastavaid andmeid läheb vaja, et täita vastuvõtuaegade funktsionaalse allsüsteemi eesmärke. Joonisel 7 on olemi-suhte diagramm koostatud Enterprise Architect abil. Joonisel 8 on sama diagramm koostatud DB-MAIN abil. Punasega on tähistatud registri (antud juhul vastuvõtuaegade registri) põhiobjekt. Sinisega on tähistatud registrisse (antud juhul vastuvõtuaegade registrisse) kuuluvad mitte-põhiobjektid. Rohelisega on tähistatud teistesse registritesse kuuluvad objektid, mida on antud juhul vaja vastuvõtuaegade funktsionaalse allsüsteemi toimimise tagamiseks.

DB-MAIN süsteemis saab andmetüüpi täpsemalt määrata. Mõlemas süsteemis saab diagrammile lisada märkmelehti, et kirjeldada näiteks mudelielementidega seotud kitsendusi.



Joonis 7. Enterprise Architect abil koostatud laiendatud vastuvõtuaegade registri olemi-suhte diagramm



Joonis 8. DB-MAIN abil koostatud laiendatud vastuvõtuaegade registri olemi-suhte diagramm

3.1.2. Olemitüüpide definitsioonid

Tabelis 1 on toodud olemitüüpide kuuluvus registrisse ja nende definitsioonid. Neid definitsioone saab nii Enterprise Architect kui DB-MAIN vaheldis kirjeldada vastava elemendi kommentaarina. Joonisel 9 on näitena toodud *Isik* definitsiooni esitus DB-MAIN vahendi abil loodud mudelis.

Tabel 1. Olemitüüpide definitsioonid

Olemitüübi nimi (aliased)	**Kuuluvus registrisse**	**Definitsioon**
Isik	Isikute register	Mistahes tervishoiuasutusega seotud füüsiline isik. Isik võib olla seotud tervishoiuasutusega töösuhte kaudu, teenuse saajana või lepinguliselt.
Isiku_seisundi_liik	Klassifikaatorite register	Isiku elutsükli võimalikud sammud, mis on määratud isiku ja Eesti Vabariigi vahelise suhte hetkeolukorraga (kodanik, kodakondsusest vabastatud, surnud). Näiteks kui isik ei ole enam EV kodanik, siis kaob tal õigus tervishoiuteenust saada.
Patsient	Patsientide register	Isik, kellele osutatakse tervishoiuasutuses tervishoiuteenust.
Patsiendi_seisundi_liik	Klassifikaatorite register	Patsiendi elutsükli võimalikud sammud, mis on määratud patsiendi ja tervishoiuasutuse vahelise suhte hetkeolukorraga (nt aktiivne, patsiendiks olek peatatud, patsiendiks olek lõpetatud).
Töötaja	Töötajate register	Tervishoiuasutuses töölepinguga töötav isik, kes haldab vastuvõtu andmeid.
Töötaja_seisundi_liik	Klassifikaatorite register	Töötaja elutsükli võimalikud sammud (nt tööl, tööleping ajutiselt peatatud, töölt lahkunud).
Arst	Töötajate register	Tervishoiuasutuses töölepinguga töötav isik, kes võtab patsiente vastu.
Eriala	Klassifikaatorite register	Terviseameti tervishoiutöötajate registris on tervishoiutöötajal kutse (arst, hambaarst, õde, ämmaemand) ja eriala („Tervishoiutöötajate riikliku registri põhimäärus“). Eriala näiteks on neuroloogia, gastroenteroloogia, kardioloogia.
Ruum	Ruumide register	Hoones asuv siseruum, millel on põrand, seinad ja katus ning millesse on inimestel võimalik siseneda.
Ruumi_seisundi_liik	Klassifikaatorite register	Ruumi elutsükli võimalikud sammud (kasutuses, ajutiselt kasutusest maas, lõplikult kasutusest maas).
Hoone	Ruumide register	„Hoone on väliskeskkonnast katuse ja teiste välispiiretega eraldatud siseruumiga ehitis“ („Ehitusseadus“).
Vastuvõtu_tüüp	Klassifikaatorite register	Eesti Haigekassa tervishoiuteenuste loetelust tulenev vastuvõtu tüüp. Näiteks: ambulatoorne vastuvõtt, erakorraline vastuvõtt („Eesti Haigekassa tervishoiuteenuste loetelu“).

Olemitüübi nimi (aliased)	Kuuluvus registrisse	Definitsioon
Vastuvõtt	Vastuvõtuaegade register	Ajaperiood, mille jooksul arst kohtub patsientidega, kellele annab konsultatsiooni tervisega seotud küsimustes.
Vastuvõtule_tulek	Vastuvõtuaegade register	Patsiendi kinnitus, et ta soovib konkreetsel vastuvõtuajal tulla arsti vastuvõtule.
Vastuvõtuaeg	Vastuvõtuaegade register	Ajaperiood, mille jooksul saab arst vastu võtta ühe konkreetse patsiendi.

Joonis 9. Isik definitsiooni esitus DB-MAIN vahendi abil loodud mudelis

3.1.3. Atribuutide definitsioonid

Tabelis 2 on toodud atribuutide definitsioonid ja näiteväärtused. Neid definitsioone saab nii Enterprise Architect kui DB-MAIN vaheldis kirjeldada vastava elemendi kommentaarina. Joonisel 10 on näitena toodud *isikukood* definitsiooni esitus Enterprise Architect vahendi abil loodud mudelis.

Tabel 2. Atribuutide definitsioonid

Olemitüübi nimi	Atribuudi nimi	Atribuudi definitsioon	Näiteväärtus
Isik	isikukood	Eesti Vabariigi isikukood. „Isikukood on isiku soo ja sünniaja alusel moodustatud isiku üheselt kindlaks määramist võimaldav arv.“	36611014244

Olemitüübi nimi	Atribuudi nimi	Atribuudi definitsioon	Näiteväärtus
		(„Rahvastikuregistri seadus“) „Oluline on teada, et isikukoodid ei kuulu delikaatsete isikuandmete alla. Isikukoodid on tavalised isikuandmed ja nende kasutamisele ei ole seatud rohkem piiranguid kui näiteks inimese nime või sünniaja kasutamisele.“ („Andmekaitse inspektsiooni juhis. Isikukoodi kasutamine“) {Isiku unikaalne identifikaator. Registreerimine on kohustuslik. Isikukood koosneb üheteistkümnest numbrimärgist. Esimene märk on numbrimärk vahemikus 3 kuni 6. See on soo tähis. Neljas märk on number 0 või 1 See on sünnikuu esimene number. Kuues märk on number vahemikus 0 kuni 3. See on sünni päeva esimene number.}	
Isik	eesnimi	„Lapsele pärast sündi (registreerimisel) pandav nimi, osa isikunimest. Eesnimi asetseb harilikult perekonnanime ees, harva järel (nt Ungari pruugis).“[1] {Registreerimine on kohustuslik. Eesnimi ei tohi olla tühi string või ainult tühikutest koosnev string. Eesnimi võib sisaldada ainult tähti, tühikuid või kriipse. Eesnimi algab suure tähega. Ülejäänud tähed on väikesed. Kui inimesel on mitu eesnime, siis need on eraldatud ühe tühiku või kriipsuga. Ka iga järgnev eesnimi algab suure tähega.}	Toomas

[1] http://mt.legaltext.ee/esterm/concept.asp?conceptID=196&term=eesnimi

Olemitüübi nimi	Atribuudi nimi	Atribuudi definitsioon	Näiteväärtus
Isik	perenimi	„Nimi, mis on isikul ühine teiste tema perekonna liikmetega.“[2] {Registreerimine on kohustuslik. Perenimi ei tohi olla tühi string või ainult tühikutest koosnev string. Perenimi võib sisaldada ainult tähti, tühikuid või kriipse. Perenimi algab suure tähega. Ülejäänud tähed on väikesed. Kui inimesel on mitu perenime, siis need on eraldatud ühe tühiku või kriipsuga. Ka iga järgnev perenimi algab suure tähega.}	Tamm
Isik	elukoht	Isiku alalise elukoha aadress. „Koha-aadress on territooriumi haldusjaotuse hierarhiast ja ametlikest kohanimedest lähtuv aadressobjekti tekstilis-numbriline kirje või tunnus. Ühele objektile võib määrata mitu koha-aadressi. Ühele objektile määratud koha-aadressid on paralleelaadressid.“ („Aadressandmete süsteemi kehtestamine“) Näide: Tallinn, 34124, Ehitajate tee 62-12. Harjumaa, Viimsi vald, Kaku küla, Laane talu. {Elukoht ei tohi olla tühi string, ainult tühikutest koosnev string või ainult numbritest koosnev string.}	Tallinn, Käo 54
Isik	kasutajanimi	Isikule antud unikaalne nimi, mida kasutatakse tema infosüsteemi poolseks autentimiseks. Töötaja kasutajanimi moodustatakse tema eesnimest ja perenimest	toomas.tamm

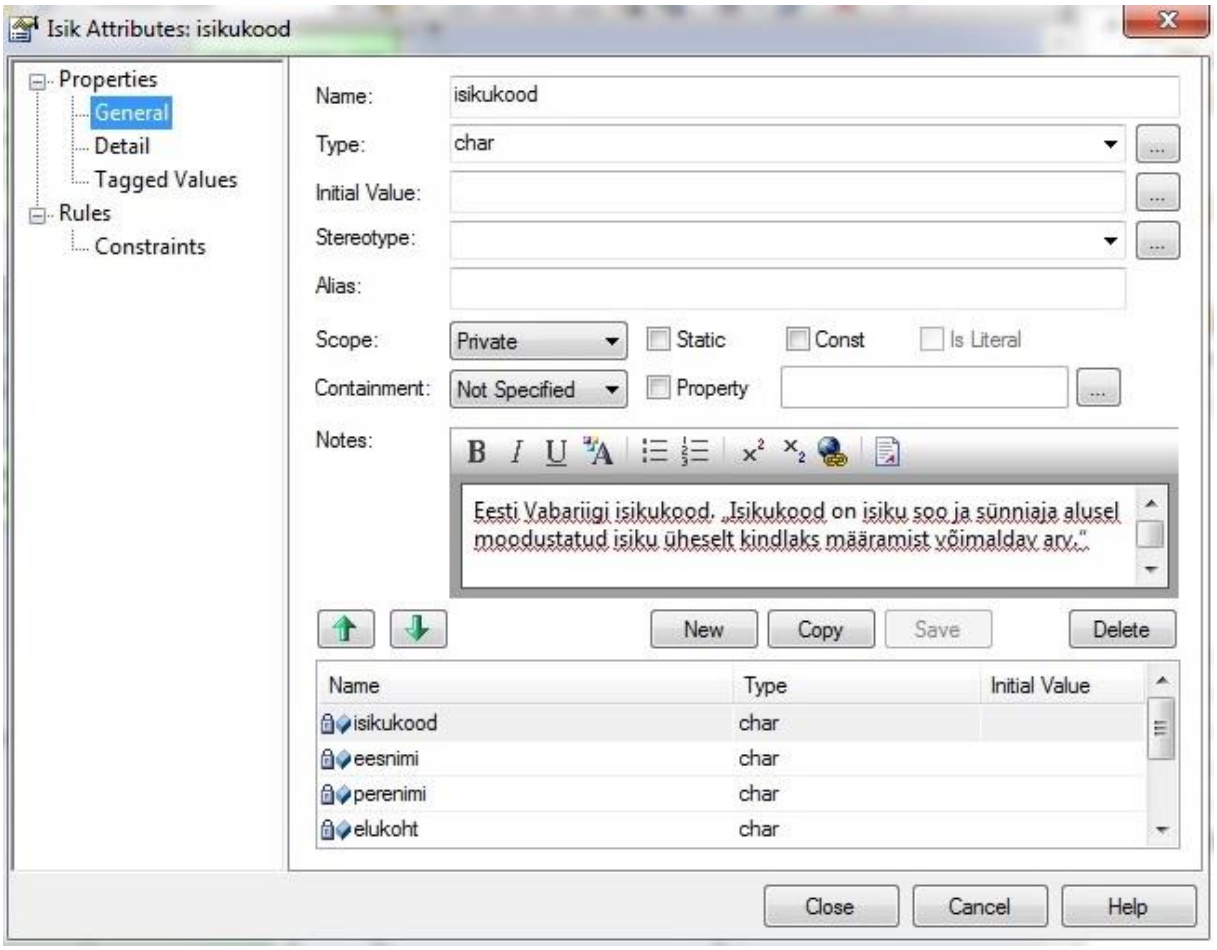
[2] http://mt.legaltext.ee/esterm/concept.asp?conceptID=997&term=perekonnanimi

Olemitüübi nimi	Atribuudi nimi	Atribuudi definitsioon	Näiteväärtus
		väikeste tähtedega, mille vahel on „.“. {Isiku unikaalne identifikaator. Registreerimine on kohustuslik. Kasutajanimi ei tohi olla lühem kui neli märki. Kasutajanimi ei tohi sisaldada tühikuid.}	
Isik	parool	Isiku identsust tõendav teadmuslik (miski, mida isik teab) volitustõend. {Registreerimine on kohustuslik.}	G45rT3v
Isik	e_mail	Aadress, millele saab üle võrgu (ühest arvutist või tööjaamast teise) saata isikule mõeldud kirjalikke sõnumeid. {Isiku unikaalne identifikaator. Registreerimine on kohustuslik. e_mail peab sisaldama ’’@’’ märki.}	tamm@mail.ee
Eriala	eriala_kood	Terviseameti tervishoiutöötajate registris oleva tervishoiutöötaja eriala kood. {Registreerimine on kohustuslik.}	E230
Eriala	nimetus	Terviseameti tervishoiutöötajate registris oleva tervishoiutöötaja eriala nimetus. {Registreerimine on kohustuslik.}	neuroloogia
Ruum	ruumi_kood	Ruumi hoone piires unikaalselt identifitseeriv kood, mis on ka kirjutatud ruumi sisenemiseks mõeldud ukse juurde. Seda koodi kasutatakse ruumile viitamiseks ka väljapool andmebaasi. {Registreerimine on kohustuslik. Ruumi kood võib sisaldada tähti, numbreid, kriipse, tühikuid.	205

Olemitüübi nimi	Atribuudi nimi	Atribuudi definitsioon	Näiteväärtus
		Ruumi kood ei tohi olla tühi string või ainult tühikutest koosnev string. Ühes hoones ei tohi olla kahte või rohkem samasuguse koodiga ruumi.}	
Ruum	korrus	Struktuur, mis koosneb hulgast ruumidest, mis asuvad vertikaalteljel ühisel tasapinnal. {Registreerimine on kohustuslik. Korrus võib olla vahemikus -5 kuni 10 (otspunktid kaasa arvatud).}	2
Hoone	hoone_kood	Hoonet unikaalselt identifitseeriv kood, mida kasutatakse hoonele viitamiseks ka väljapool andmebaasi. {Hoone unikaalne identifikaator. Registreerimine on kohustuslik. Hoone kood võib sisaldada tähti, numbreid, kriipse, alakriipse, tühikuid. Hoone kood ei tohi olla tühi string või ainult tühikutest koosnev string.}	PERH
Hoone	nimi	Hoonet unikaalselt identifitseeriv nimi, mida kasutatakse hoonele viitamiseks ka väljapool andmebaasi. {Hoone unikaalne identifikaator. Registreerimine on kohustuslik. Nimi ei tohi olla tühi string või ainult tühikutest koosnev string.}	Põhja-Eesti Regionaalhaigla
Vastuvõtt	kuupäev	Kuupäev, millal patsiendil on võimalik tulla vastuvõtule. {Registreerimine on kohustuslik. Kuupäev peab olema vahemikus 1. jaanuar 2010 ja 1. jaanuar 2100 (otspunktid kaasa arvatud).}	10.03.2011

Olemitüübi nimi	Atribuudi nimi	Atribuudi definitsioon	Näiteväärtus
Vastuvõtt	vajalik_saatekiri	Märge, kas konkreetsele vastuvõtule tulekuks on vajalik saatekiri. {Registreerimine on kohustuslik.}	Tõene
Vastuvõtt	kommentaar	Kommentaar, mis selgitab, kas konkreetne vastuvõtt on tasuline. {Kommentaar ei tohi olla tühi string või ainult tühikutest koosnev string.}	Tasuline vastuvõtt
Vastuvõtuaeg	kellaaeg	Kellaaeg vastuvõtu kuupäeval, millal patsiendil on võimalik tulla vastuvõtule. {Registreerimine on kohustuslik. Ühelgi vastuvõtul ei tohi olla kaks või rohkem samasugust vastuvõtuaega.}	11:00:00
Isiku_seisundi_liik	isiku_seisundi_liik_kood	Isiku elutsükli võimaliku sammu kood. {Registreerimine on kohustuslik.}	1
Isiku_seisundi_liik	nimetus	Isiku elutsükli võimaliku sammu nimetus. {Registreerimine on kohustuslik.}	Kodanik
Patsiendi_seisundi_liik	patsiendi_seisundi_liik_kood	Patsiendi elutsükli võimaliku sammu kood. {Registreerimine on kohustuslik.}	1
Patsiendi_seisundi_liik	nimetus	Patsiendi elutsükli võimaliku sammu nimetus. {Registreerimine on kohustuslik.}	Aktiivne
Töötaja_seisundi_liik	tööaja_seisundi_liik_kood	Töötaja elutsükli võimaliku sammu kood. {Registreerimine on kohustuslik.}	1
Tööaja_seisundi_liik	nimetus	Töötaja elutsükli võimaliku sammu nimetus. {Registreerimine on kohustuslik.}	Tööl

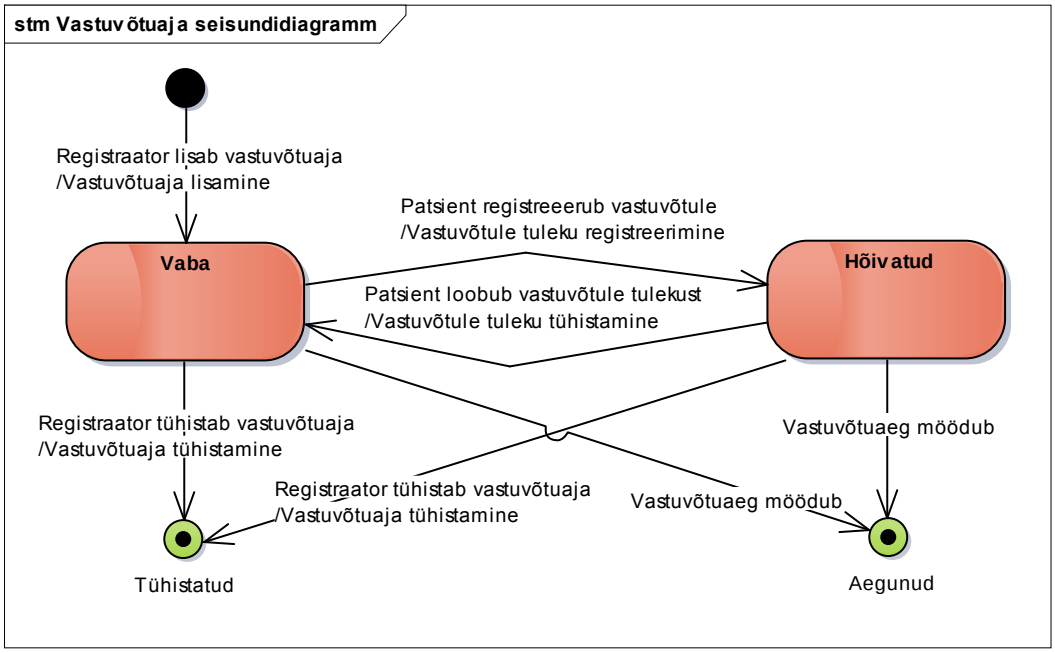
Olemitüübi nimi	Atribuudi nimi	Atribuudi definitsioon	Näiteväärtus
Ruumi_seisundi_liik	ruumi_seisundi_liik_kood	Ruumi elutsükli võimaliku sammu kood. {Registreerimine on kohustuslik.}	1
Ruumi_seisundi_liik	nimetus	Ruumi elutsükli võimaliku sammu nimetus. {Registreerimine on kohustuslik.}	Kasutuses
Vastuvõtu_tüüp	vastuvõtu_tüübi_kood	Eesti Haigekassa tervishoiuteenuste loetelust tulenev vastuvõtu tüübi kood. {Registreerimine on kohustuslik.}	1
Vastuvõtu_tüüp	nimetus	Eesti Haigekassa tervishoiuteenuste loetelust tulenev vastuvõtu tüübi nimetus. {Registreerimine on kohustuslik.}	Ambulatoorne vastuvõtt



Joonis 10. Isikukood definitsiooni esitus Enterprise Architect vahendi abil loodud mudelis

3.2. Registri põhiobjekti seisundidiagramm

Seisundidiagrammi on võimalik koostada ainult Enterprise Architect abil. Vastuvõtuaegade registri põhiobjekt on „Vastuvõtuaeg“. Joonisel 11 on toodud Enterprise Architect abil koostatud vastuvõtuaja seisundidiagramm.



Joonis 11. Enterprise Architect abil koostatud vastuvõtuaja seisundidiagramm

3.3. CRUD maatriks

Järgnevalt on tabelis 3 toodud CRUD maatriks olemitüüpide ja kasutusjuhtude täpsusega. Veergudeks on kasutusjuhud ja ridadeks olemitüübid. Maatriksi veergudele vastavad kasutusjuhud ja ridadele olemitüübid. Oranžil taustal on esitatud olemitüübid, mis kuuluvad vaadeldava allsüsteemi teenindatavasse registrisse. Enterprise Architect põhimõtteliselt võimaldab CRUD maatriksi koostamist (vt joonis 16).

Tabel 3. CRUD maatriks

Kasutusjuhud / Olemitüübid	1	2	3	4	5	6	7	8	9	**Kokku**
Isik	R					R	R	R	R	R
Isiku_seisundi_liik	R									R
Patsient	R					R				R
Patsiendi_seisundi_liik	R									R
Registraator	R						R	R	R	R
Registraatori_seisundi_liik	R									R
Arst	R	R	R	R	R	R	R	R	R	R
Eriala	R						R			R
Ruum		R	R	R	R	R	R	R	R	R
Ruumi_seisundi_liik		R	R	R	R	R	R	R	R	R
Hoone		R	R	R	R	R	R	R	R	R
Vastuvõtu_tüüp		R	R	R	R	R	R	R	R	R
Vastuvõtt		CR	R	RD	RU	R	R	R	R	CRUD
Vastuvõtuaeg			CR	RD		R	R	R	R	CRD
Vastuvõtule_tulek				RD		R	CR	RD	R	CRD

Kasutusjuhtude nimekiri.

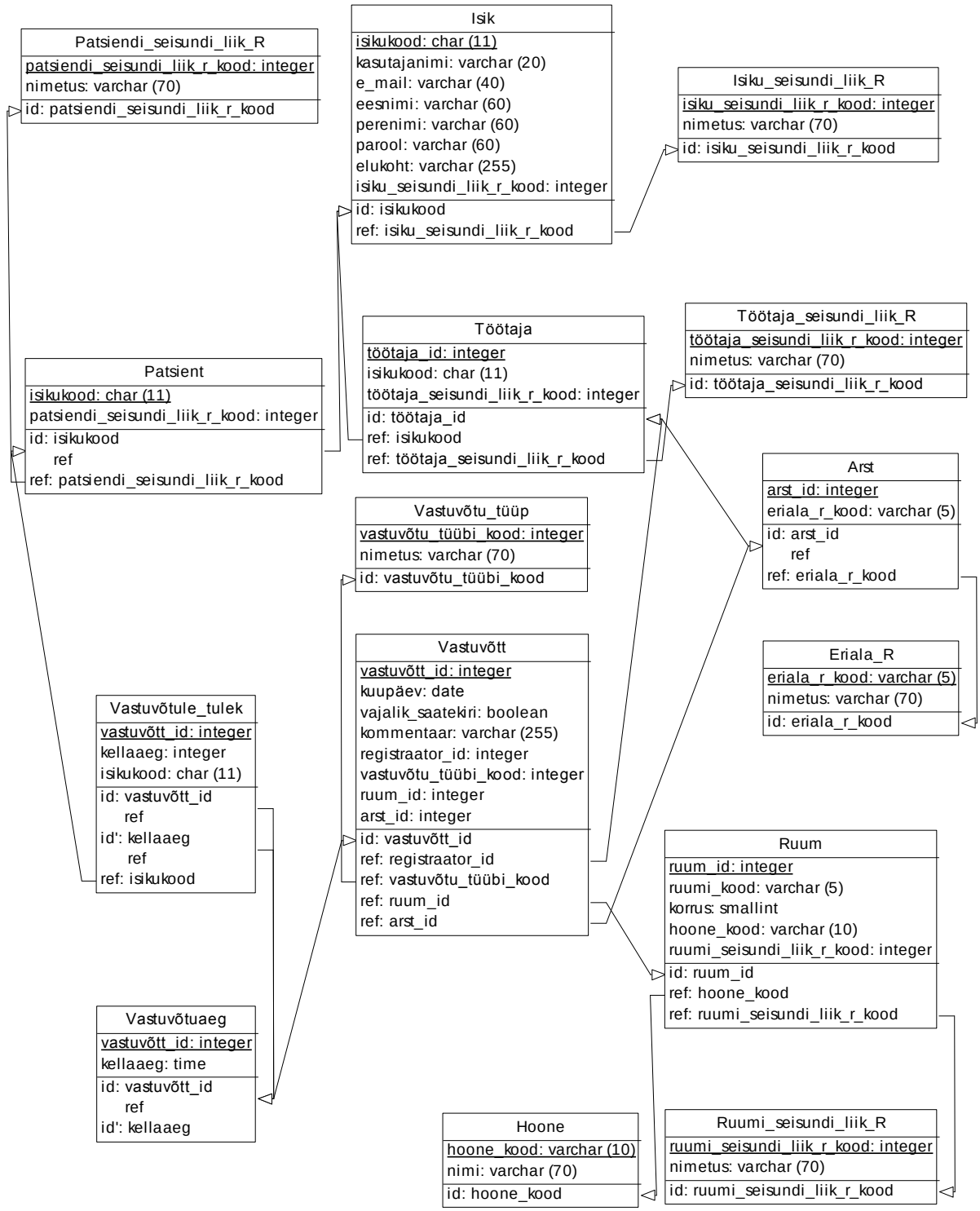
1. Kasutaja tuvastamine
2. Vastuvõtu väljapakkumine
3. Vastuvõtuaja lisamine
4. Vastuvõtuaja tühistamine
5. Vastuvõtu andmete parandamine
6. Registraatori poolt arstide vastuvõttude vaatamine
7. Vastuvõtule tuleku registreerimine
8. Vastuvõtule tuleku tühistamine
9. Patsiendi poolt oma vastuvõtuaegade vaatamine

4. Disaini etapp

Neljas peatükk annab ülevaate andmebaasi projekteerimise kolmandast sammust: disaini etapist. See samm jaguneb omakorda loogiliseks ja füüsiliseks disainiks. Mõlema sammu kohta on toodud vastav andmemudel.

4.1. Loogiline disain

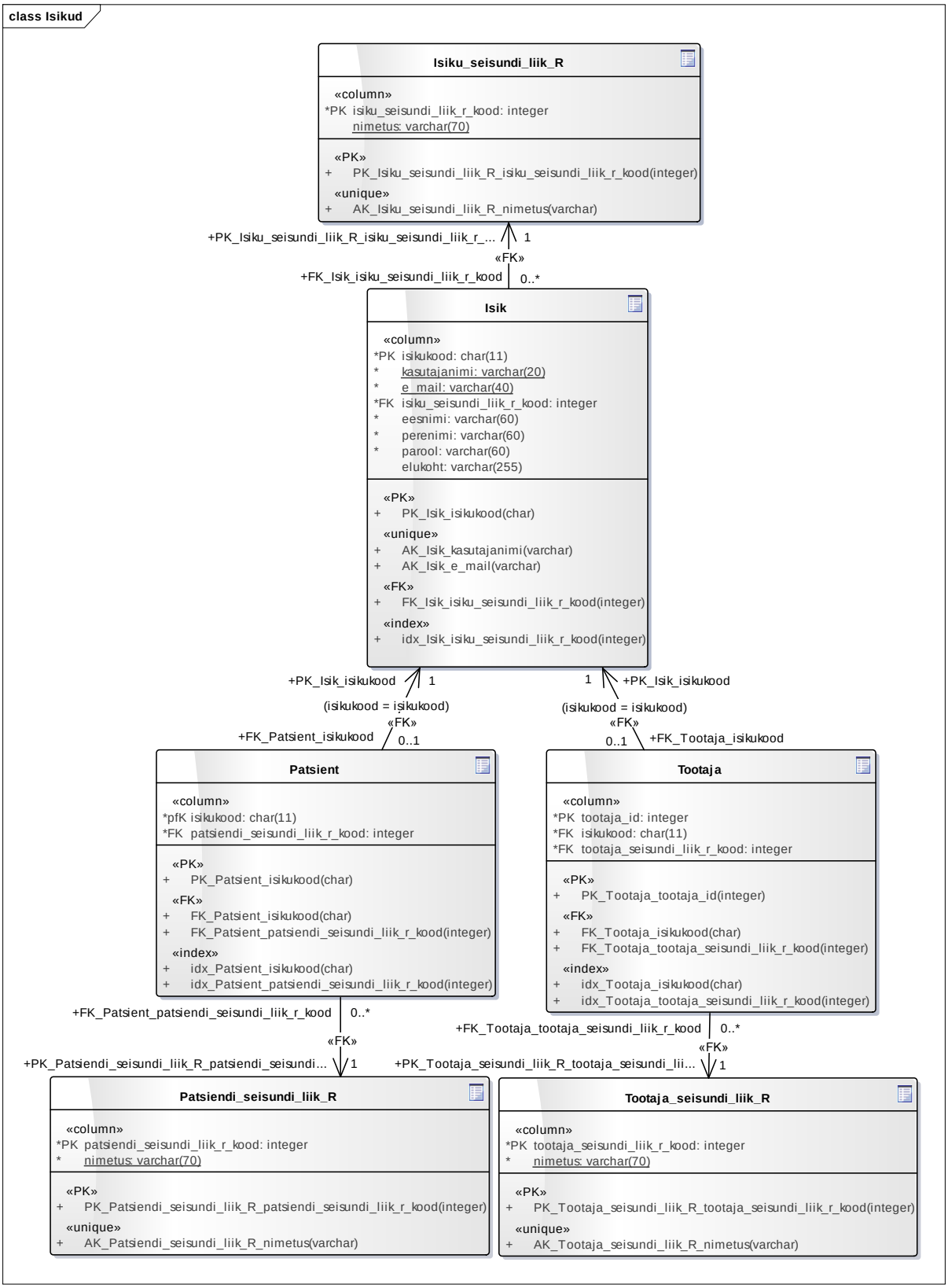
Andmebaasi loogilist andmemudelit on põhimõtteliselt võimalik koostada ainult DB-MAIN abil. Joonisel 12 on toodud DB-MAIN abil koostatud andmebaasi loogiline andmemudel, mis kirjeldab SQL-andmebaasi ülesehitust. Loogilises andmemudelis kasutatakse vaikimisi järgmist notatsiooni: tabelite vahelisi viiteid kujutatakse noolega, suunaga tabelile, kus asub seoses osalev primaarvõti. Primaarvõtmesse kuuluvatele veergudele on diagrammis joon alla tõmmatud, lisaks näidatud tähistusega tabeli veergude all id: veeru nimi. Kui primaarvõtmesse kuulub mitu veergu, siis esimene tähistatakse id: veeru nimi ning teised id': veeru nimi ja joont alla ei tõmmata. Välisvõtmesse kuuluvad veerud tähistatakse diagrammil ref: veeru nimi. (REVER S.A. – DB-MAIN, 2009)

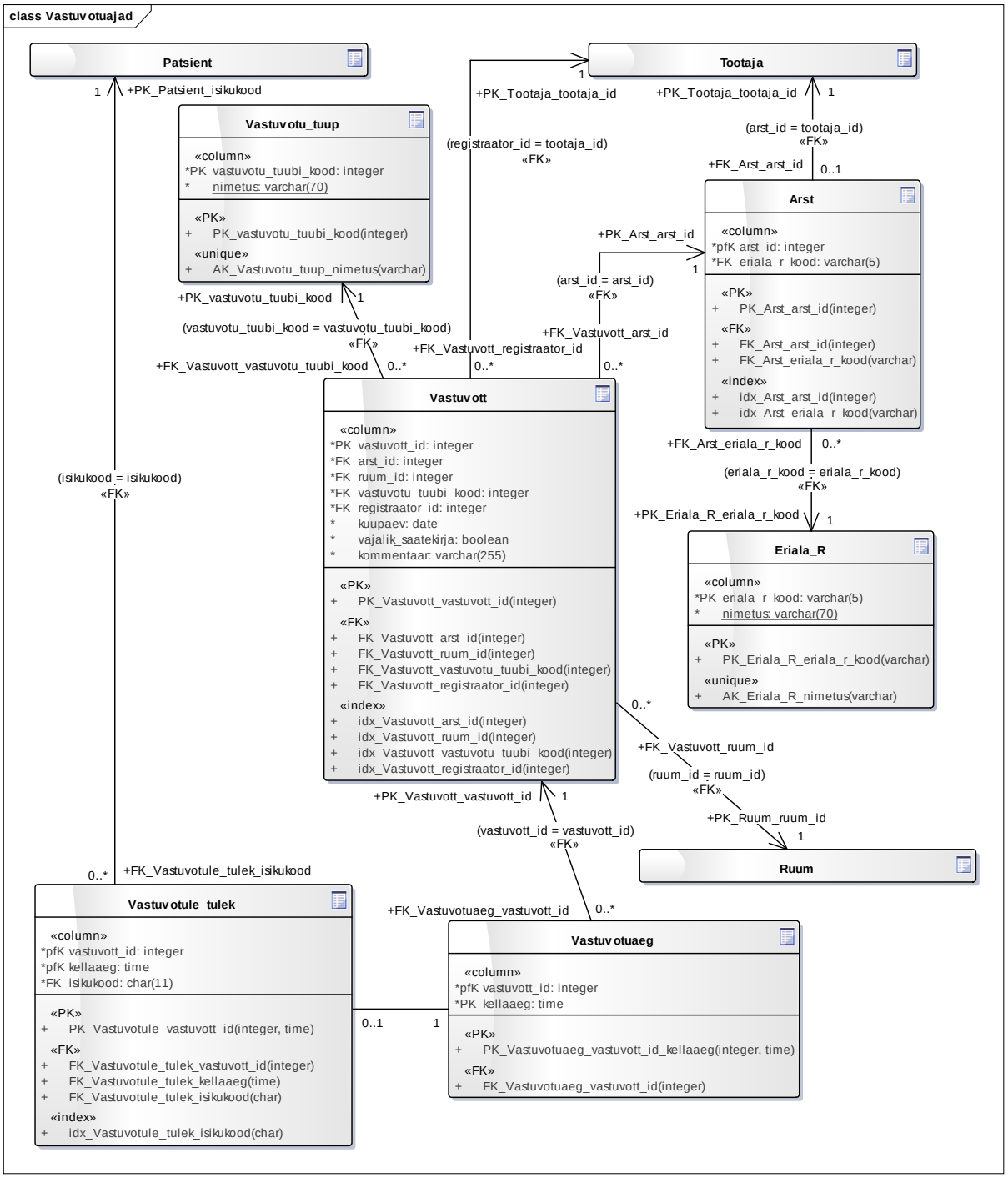


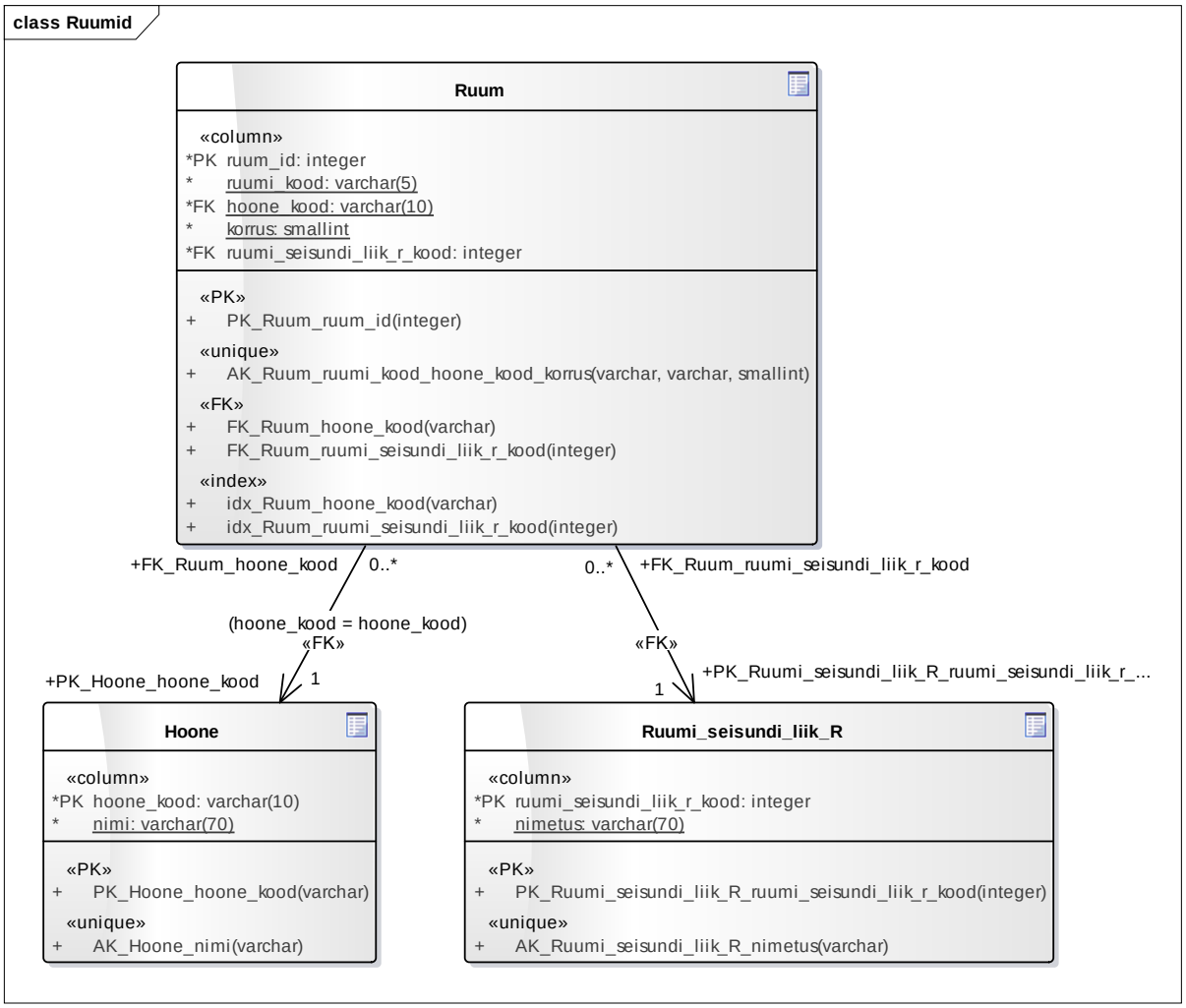
Joonis 12. DB-MAIN abil koostatud andmebaasi loogiline andmemudel

4.2. Füüsiline disain

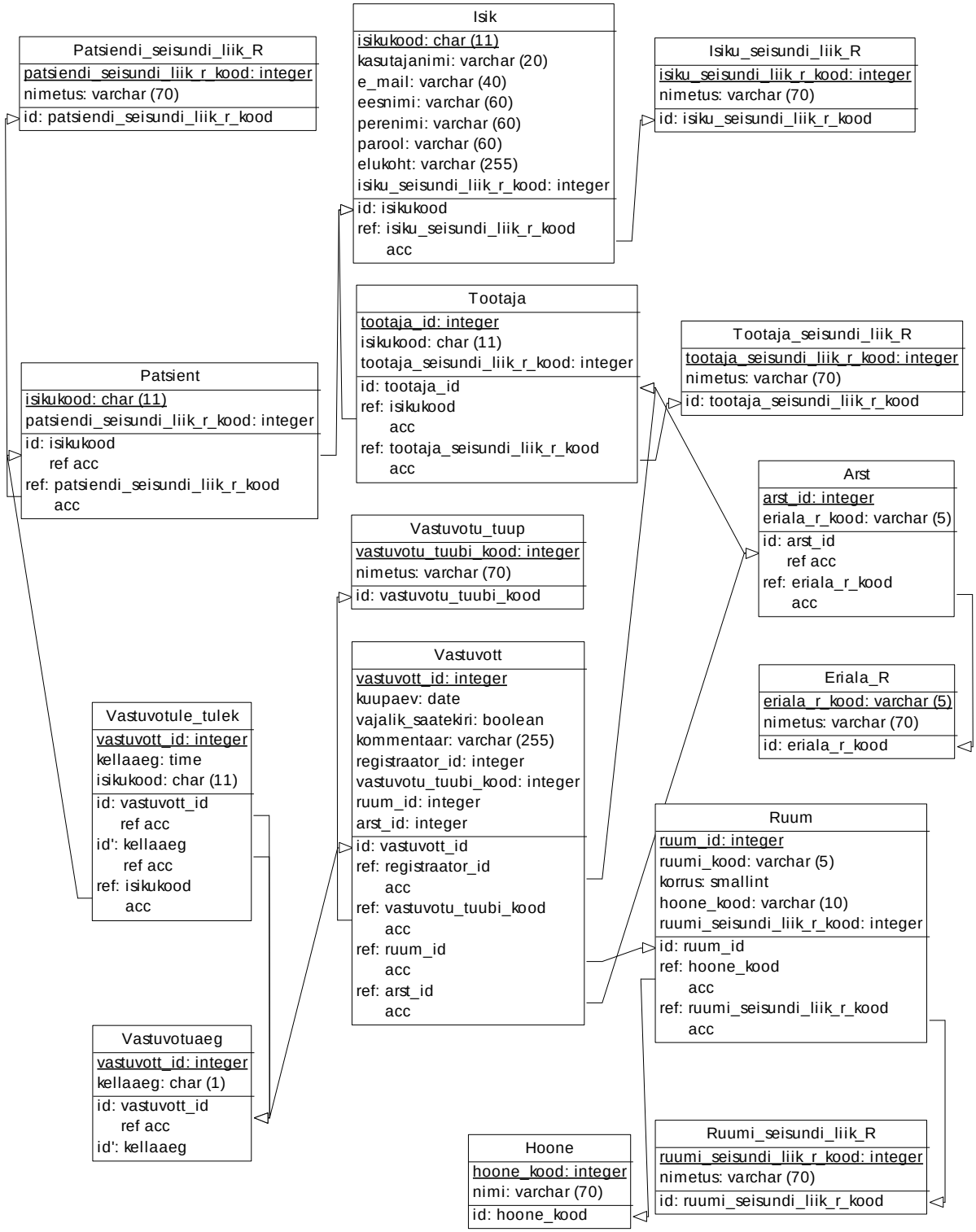
Füüsilist andmemudelit on põhimõtteliselt võimalik koostada nii Enterprise Architect kui ka DB-MAIN abil. Joonistel 13 toodud Enterprise Architect abil koostatud ja joonistel 14 toodud DB-MAIN abil koostatud füüsilised andmemudelid põhinevad PostgreSQL andmebaasisüsteemil. Segadust tekitab, et DB-MAIN notatsioonis tähistab kolmnurk joone otsas 1:M seost (REVER S.A. – DB-MAIN, 2012), kuid UMLis on see üldistuse tähis. Kui DB-MAINis on allajoonitud primaarvõtmesse kuuluvad veerud, siis Enterprise Architect vahendis täiendava unikaalsuse kitsendusega veerud, mida DB-MAIN vahend ei võimalda diagrammil näidata. Indekseid välisvõtme veergudele tähistatakse DB-MAINis lühendiga acc, Enterprise Architect vahendis kasutatakse operatsiooni <<index>> stereotüüpi.


class Isikud
Isiku_seisundi_liik_R
«column»
*PK isiku_seisundi_liik_r_kood: integer
nimetus: varchar(70)
«PK»
+ PK_Isiku_seisundi_liik_R_isiku_seisundi_liik_r_kood(integer)
«unique»
+ AK_Isiku_seisundi_liik_R_nimetus(varchar)
+PK_Isiku_seisundi_liik_R_isiku_seisundi_liik_r_...
1
«FK»
+FK_Isik_isiku_seisundi_liik_r_kood
0..*
Isik
«column»
*PK isikukood: char(11)
* kasutajanimi: varchar(20)
* e_mail: varchar(40)
*FK isiku_seisundi_liik_r_kood: integer
* eesnimi: varchar(60)
* perenimi: varchar(60)
* parool: varchar(60)
elukoht: varchar(255)
«PK»
+ PK_Isik_isikukood(char)
«unique»
+ AK_Isik_kasutajanimi(varchar)
+ AK_Isik_e_mail(varchar)
«FK»
+ FK_Isik_isiku_seisundi_liik_r_kood(integer)
«index»
+ idx_Isik_isiku_seisundi_liik_r_kood(integer)
+PK_Isik_isikukood
1
(isikukood = isikukood)
«FK»
+FK_Patsient_isikukood
0..1
1
+PK_Isik_isikukood
(isikukood = isikukood)
«FK»
0..1
+FK_Tootaja_isikukood
Patsient
«column»
*pfK isikukood: char(11)
*FK patsiendi_seisundi_liik_r_kood: integer
«PK»
+ PK_Patsient_isikukood(char)
«FK»
+ FK_Patsient_isikukood(char)
+ FK_Patsient_patsiendi_seisundi_liik_r_kood(integer)
«index»
+ idx_Patsient_isikukood(char)
+ idx_Patsient_patsiendi_seisundi_liik_r_kood(integer)
Tootaja
«column»
*PK tootaja_id: integer
*FK isikukood: char(11)
*FK tootaja_seisundi_liik_r_kood: integer
«PK»
+ PK_Tootaja_tootaja_id(integer)
«FK»
+ FK_Tootaja_isikukood(char)
+ FK_Tootaja_tootaja_seisundi_liik_r_kood(integer)
«index»
+ idx_Tootaja_isikukood(char)
+ idx_Tootaja_tootaja_seisundi_liik_r_kood(integer)
+FK_Patsient_patsiendi_seisundi_liik_r_kood
0..*
«FK»
+PK_Patsiendi_seisundi_liik_R_patsiendi_seisundi...
1
+FK_Tootaja_tootaja_seisundi_liik_r_kood
0..*
«FK»
+PK_Tootaja_seisundi_liik_R_tootaja_seisundi_lii...
1
Patsiendi_seisundi_liik_R
«column»
*PK patsiendi_seisundi_liik_r_kood: integer
* nimetus: varchar(70)
«PK»
+ PK_Patsiendi_seisundi_liik_R_patsiendi_seisundi_liik_r_kood(integer)
«unique»
+ AK_Patsiendi_seisundi_liik_R_nimetus(varchar)
Tootaja_seisundi_liik_R
«column»
*PK tootaja_seisundi_liik_r_kood: integer
* nimetus: varchar(70)
«PK»
+ PK_Tootaja_seisundi_liik_R_tootaja_seisundi_liik_r_kood(integer)
«unique»
+ AK_Tootaja_seisundi_liik_R_nimetus(varchar)


class Vastuvotuajad
Patsient
Tootaja
Vastuvotu_tuup
«column»
*PK vastuvotu_tuubi_kood: integer
* nimetus: varchar(70)
«PK»
+ PK_vastuvotu_tuubi_kood(integer)
«unique»
+ AK_Vastuvotu_tuup_nimetus(varchar)
Arst
«column»
*pfK arst_id: integer
*FK eriala_r_kood: varchar(5)
«PK»
+ PK_Arst_arst_id(integer)
«FK»
+ FK_Arst_arst_id(integer)
+ FK_Arst_eriala_r_kood(varchar)
«index»
+ idx_Arst_arst_id(integer)
+ idx_Arst_eriala_r_kood(varchar)
Vastuvott
«column»
*PK vastuvott_id: integer
*FK arst_id: integer
*FK ruum_id: integer
*FK vastuvotu_tuubi_kood: integer
*FK registraator_id: integer
* kuupaev: date
* vajalik_saatekirja: boolean
* kommentaar: varchar(255)
«PK»
+ PK_Vastuvott_vastuvott_id(integer)
«FK»
+ FK_Vastuvott_arst_id(integer)
+ FK_Vastuvott_ruum_id(integer)
+ FK_Vastuvott_vastuvotu_tuubi_kood(integer)
+ FK_Vastuvott_registraator_id(integer)
«index»
+ idx_Vastuvott_arst_id(integer)
+ idx_Vastuvott_ruum_id(integer)
+ idx_Vastuvott_vastuvotu_tuubi_kood(integer)
+ idx_Vastuvott_registraator_id(integer)
Eriala_R
«column»
*PK eriala_r_kood: varchar(5)
* nimetus: varchar(70)
«PK»
+ PK_Eriala_R_eriala_r_kood(varchar)
«unique»
+ AK_Eriala_R_nimetus(varchar)
Ruum
Vastuvotule_tulek
«column»
*pfK vastuvott_id: integer
*pfK kellaaeg: time
*FK isikukood: char(11)
«PK»
+ PK_Vastuvotule_vastuvott_id(integer, time)
«FK»
+ FK_Vastuvotule_tulek_vastuvott_id(integer)
+ FK_Vastuvotule_tulek_kellaaeg(time)
+ FK_Vastuvotule_tulek_isikukood(char)
«index»
+ idx_Vastuvotule_tulek_isikukood(char)
Vastuvotuaeg
«column»
*pfK vastuvott_id: integer
*PK kellaaeg: time
«PK»
+ PK_Vastuvotuaeg_vastuvott_id_kellaaeg(integer, time)
«FK»
+ FK_Vastuvotuaeg_vastuvott_id(integer)
+PK_Patsient_isikukood
+PK_Tootaja_tootaja_id
(registraator_id = tootaja_id)
«FK»
(arst_id = tootaja_id)
«FK»
+FK_Arst_arst_id
+PK_Arst_arst_id
(arst_id = arst_id)
«FK»
+FK_Vastuvott_arst_id
+PK_vastuvotu_tuubi_kood
(vastuvotu_tuubi_kood = vastuvotu_tuubi_kood)
«FK»
+FK_Vastuvott_registraator_id
+FK_Vastuvott_vastuvotu_tuubi_kood
(isikukood = isikukood)
«FK»
+FK_Arst_eriala_r_kood
(eriala_r_kood = eriala_r_kood)
«FK»
+PK_Eriala_R_eriala_r_kood
+FK_Vastuvott_ruum_id
(ruum_id = ruum_id)
«FK»
+PK_Ruum_ruum_id
+PK_Vastuvott_vastuvott_id
(vastuvott_id = vastuvott_id)
«FK»
+FK_Vastuvotule_tulek_isikukood
+FK_Vastuvotuaeg_vastuvott_id
1
0..1
0..*



Joonis 13. Enterprise Architect abil koostatud andmebaasi füüsiline andmemudel



Joonis 14. DB-MAIN abil koostatud andmebaasi füüsiline andmemudel

5. Enterprise Architect ja DB-MAIN võrdlus registripõhiste infosüsteemide projekteerimise seisukohast

Järgnevalt on võrreldud Enterprise Architect ja DB-MAIN kasutamist registripõhise infosüsteemi projekteerimise kolmes etapis: strateegia, detailanalüüsi ja disaini etapis. Välja on toodud sarnasused ja erinevused, mida autor leidis vastuvõtuaegade allsüsteemi projekteerides.

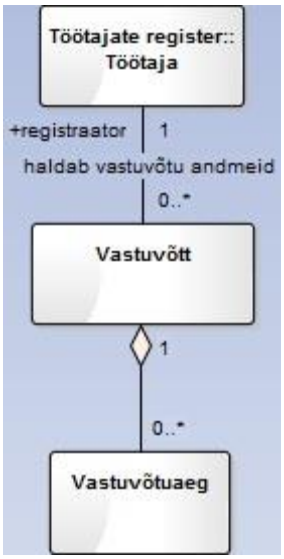
5.1. Strateegia etapp

Strateegia etapis tegi autor kolm diagrammi: tegevusdiagrammi (vt joonised 1 ja 2), kasutusjuhtude diagrammi (vt joonised 3 ja 4) ja andmebaasi kontseptuaalse eskiismudeli (vt joonised 5 ja 6), mis on lihtsustatud versioon kontseptuaalse andmemudeli olemi-suhte diagrammist (Eessaar, 2015b).

Selles etapis ei olnud valitud vahenditel suuri erinevusi: nii Enterprise Architect kui ka DB-MAIN vahendis saab teha kõiki neid diagramme. Kasutusjuhtude diagrammil saab mõlema vahendiga teha erinevaid seoseid nii kasutusjuhtude kui ka tegutsejate vahel. Andmebaasi kontseptuaalne eskiismudel luuakse mõlemas vahendis klassidiagrammi baasil.

Esinesid mõned erinevused valitud vahendite diagrammides.

1. Enterprise Architect võimaldab tegevusdiagrammil igale tegevusele lisada viite selle eest vastutavale rollile (ujumisrajad).
2. Enterprise Architect vahendis saab otsustuspunktile nime panemata jätta, kuid DB-MAINis peab seda tegema.
3. Erinevalt DB-MAIN vahendist on Enterprise Architect klassidiagrammil on võimalik esitada agregatsiooni ja rolle (näiteks *Töötaja* roll *registraator*), mida on kujutatud joonisel 15.



Joonis 15. Agregatsioon ja roll Enterprise Architect abil koostatud vastuvõtuaegade registri kontseptuaalel eskiismudelil

4. Enterprise Architect vahendis saab seoste mitmesuse (võimsustikud) ja nime määramata jätta, kuid DB-MAINis peab seda tegema.

5.2. Detailanalüüsi etapp

Detailanalüüsi etapis tegi autor kontseptuaalse andmemudeli olemi-suhte diagrammi (vt joonised 7 ja 8) ja registri põhiobjekti elutsüklit kirjeldava seisundidiagrammi (vt joonis 11). Kontseptuaalne andmemudel esindab andmete loogilist struktuuri sõltumata tarkvarast ja kasutatavast andmebaasisüsteemist (Connolly ja Begg, 2001).

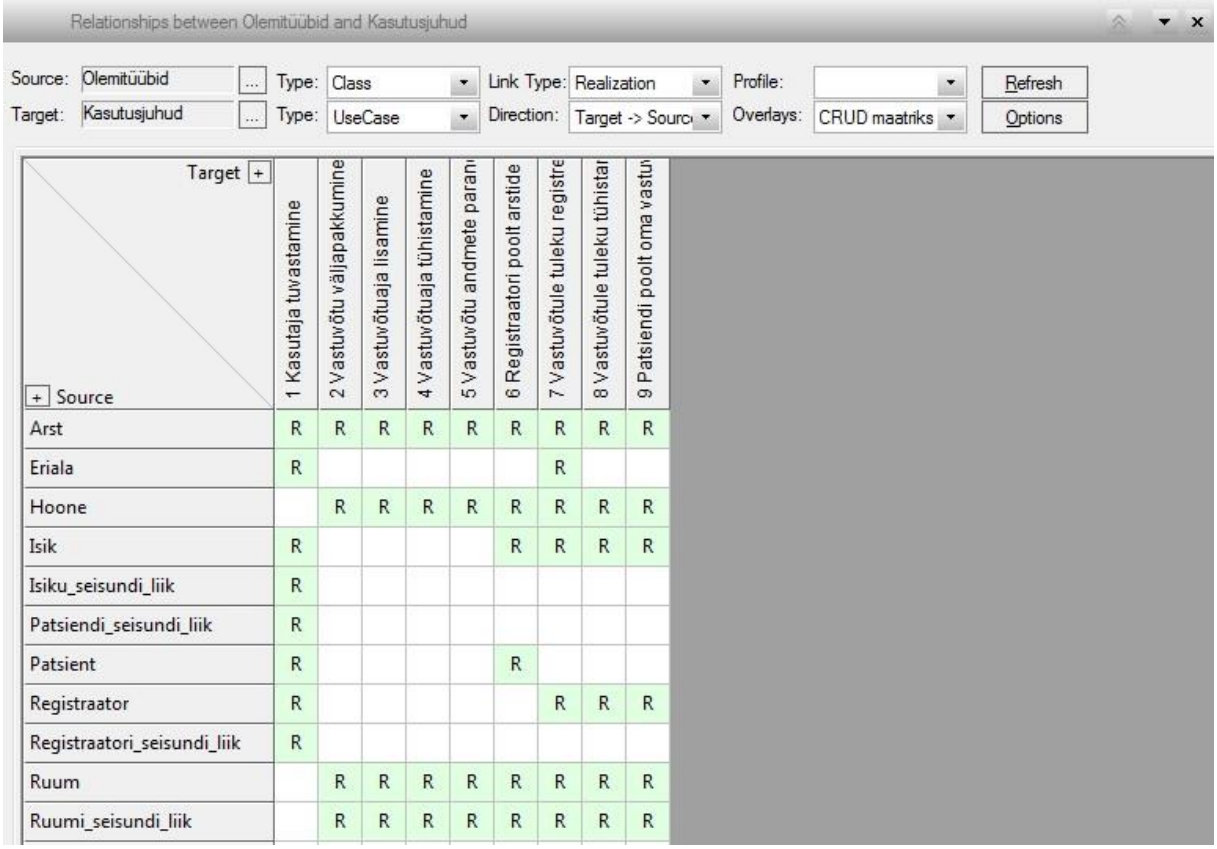
Nii Enterprise Architect kui ka DB-MAIN vahendiga saab teha olemi-suhte diagrammi. Mõlemas vahendis luuakse olemi-suhte diagramm klassidiagrammi baasil. DB-MAINis saab olemi-suhte diagrammil määrata primaarvõtmeid, kuid Enterprise Architect vahendis selline võimalus puudub. Samas on primaarvõti SQL ja relatsioonilise mudeli (st disaini) spetsiifiline mõiste ja seda polekski õige kontseptuaalses mudelis määrata. Kontseptuaalses andmemudelis saab määrata, et üks või mitu atribuuti moodustavad unikaalse identifikaatori ja seda saab määrata Enterprise Architect vahendis, kasutades kas atribuudi <<id>> stereotüüpi või atribuudi *isID* omadust.

Erinevalt DB-MAIN vahendist saab Enterprise Architectis pakettide loomisega kirjeldada olemitüüpide kuuluvust registritesse ning kasutusjuhtude kuuluvust

funktsionaalsesse allsüsteemi. Mõlema vahendiga saab luua olemitüüpide vahelisi seosetüüpe (sh üldistusseoseid).

Seisundidiagrammi on võimalik luua ainult Enterprise Architect vahendiga. Seisundidiagrammil saab muuhulgas määrata üleminekutega seotud valvurtingimused (*guard condition*). Diagrammis saab ülemineku algatanud sündmuste, valvurtingimuste ning sündmusele järgnenud reaktsiooni esitada visuaalselt mitmel real.

Detailanalüüsi etapis on toodud ka CRUD maatriks. Erinevalt DB-MAIN vahendist on Enterprise Architectiga põhimõtteliselt võimalik CRUD maatriksit koostada. Joonisel 16 on kujutatud osa antud töös koostatud CRUD maatriksist.



Joonis 16. Enterprise Architect abil koostatud CRUD maatriksi osa

5.3. Disaini etapp

Disaini etapp on jaotatud kaheks eraldi osaks: loogiline disain ja füüsiline disain. Järgnevalt on võrreldud Enterprise Architect ja DB-MAIN nendes mõlemas etapis.

5.3.1. Loogiline disain

Loogilise disaini etapis tegi autor kontseptuaalse andmemudeli põhjal loogilise andmemudeli (vt joonis 12). See mudel sõltub andmebaasi loomisel kasutatavast andmemudelist (näiteks relatsiooniline). Töös esitatud loogiline andmemudel erineb kontseptuaalsest andmemudelist näiteks selle poolest, et seal on juba näha primaarvõtmed ja välisvõtmed ning on lahendatud üldistusseosed. (Connolly ja Begg, 2001).

Enterprise Architect vahendiga andmebaasi disainides jääb see etapp vahele, seal teisendatakse kontseptuaalne andmemudel kohe füüsiliseks andmemudeliks, mis lisaks andmebaasi aluseks olevale andmemudelile arvestab ka andmebaasi realiseerimiseks kasutatava andmebaasisüsteemiga. DB-MAINis saab luua loogilise andmemudeli olemasoleva kontseptuaalse andmemudeli põhjal. Siin saab valida, kas tehakse täiesti uus mudel, kuhu lisatakse loodava loogilise andmemudeli elemendid või uuendatakse olemasolevat mudelit.

5.3.2. Füüsiline disain

Füüsilise disaini etapis tegi autor füüsilise andmemudeli (vt joonised 13 ja 14). Füüsiline disain on kohandatud kindlale andmebaasisüsteemile. Eelmises etapis koostatud loogiline andmemudel on antud etapis loodava füüsilise disaini lähtematerjaliks (Connolly ja Begg, 2001).

Mõlema uuritava CASE vahendiga on võimalik teha füüsilist andmemudelit. Kasutaja peab valima andmebaasisüsteemi, millele füüsilist andmemudelit looma hakatakse. Täpitähed eemaldatakse DB-MAINis selle andmemudeli koostamisel automaatselt.

DB-MAIN vahendiga saab teha füüsilist andmemudelit järgmiste andmebaasisüsteemide jaoks:

- Standard SQL (SQL92),

- Access,
- Firebird,
- MySQL,
- PostgreSQL.

Enterprise Architect vahendiga saab teha füüsilist andmemudelit järgmiste andmebaasisüsteemide jaoks:

- DB2,
- Informix,
- Ingres,
- InterBase,
- MSAccess,
- MSAccess 2007,
- MySQL,
- Oracle,
- PostgreSQL,
- SQL Server 2000,
- SQL Server 2005,
- SQL Server 2008,
- SQL Server 2012,
- SQLite,
- SQLSever7,
- Sybase,
- Sybase ASE.

Ärireeglite jõustamiseks vajalikke kitsendusi saab Enterprise Architect vahendis füüsilises andmemudelis kirjeldada. Kasutaja peab vastava CHECK kitsenduse või trigeri ise kirjutama, kuid see salvestatakse mudeli osana.

Kokkuvõtvalt on kõigi kolme etapi võrdlus toodud tabelis 4.

Tabel 4. Enterprise Architect ja DB-MAIN võrdlustabel

	Enterprise Architect	**DB-MAIN**
Strateegia etapp		
Tegevusdiagramm	Jah	Jah
Kasutusjuhtude diagramm	Jah	Jah

	Enterprise Architect	DB-MAIN
Kontseptuaalne eskiismudel	Jah, klassidiagrammi baasil	Jah, klassidiagrammi baasil
Kontseptuaalsel eskiismudelil saab atribuute peita	Jah	Jah
Tegevusdiagrammil saab igale tegevusele lisada viite selle eest vastutavale rollile (ujumisrajad)	Jah	Ei
Otsustuspunktile nimi tegevusdiagrammil	Võib jätta panemata	Jah
Üldistusseos tegutsejate / kasutusjuhtude vahel kasutusjuhtude diagrammil	Jah	Jah
Kasutamisseos tegutsejate / kasutusjuhtude vahel	Jah	Jah
Kontseptuaalsel andmemudelil agregatsioon ja rollid	Jah	Ei
Kontseptuaalsel andmemudelil mitmesuse (võimsustike) määramata jätmise võimalus	Jah	Ei
Olemi-suhte diagramm	Jah, klassidiagrammi baasil	Jah, klassidiagrammi baasil
Olemi-suhte diagrammil unikaalsete identifikaatorite ja primaarvõtmete määramine	Jah, unikaalsete identifikaatorite määramine	Jah, primaarvõtmete määramine
Olemitüüpide registrisse kuuluvuse määramine	Jah, saab luua pakette	Ei
Olemitüüpide vaheliste seosetüüpide määramine	Jah	Jah
Dokumenteerivalt üldistusseose kitsenduste määramine.	Jah	Jah
Seisundidiagramm	Jah	Ei
CRUD maatriksi koostamine	Jah	Ei
Disaini etapp		
Loogiline andmemudel	Ei	Jah
Füüsiline andmemudel	Jah	Jah
Ärireegli kontrolli realisatsioon ja selle põhjal koodi genereerimine	Jah, kasutaja peab ise CHECK kitsenduse või trigeri kirjutama	Ei

5.4. Järeldused

Mõlema vahendiga saab teha strateegia ja detailanalüüsi etapis loodavaid diagramme. Ainult DB-MAIN abil saab luua nii loogilist kui ka füüsilist andmemudelit. Enterprise Architect vahendis saab luua vaid füüsilist andmemudelit.

Vähem erinevusi on kasutatavatel programmidel tegevus- ja kasutusjuhtude diagrammi loomisel. Kontseptuaalse, loogilise ja füüsilise andmemudeli loomisel tuleb erinevus sellest, et Enterprise Architect vahendis pole võimalik luua loogilist andmemudelit. Füüsilise disaini mudeleid saab teha mitmete andmebaasisüsteemide jaoks. Mõlemas vahendis on suur valik

atribuutide andmetüüpe ning samuti on võimalik uusi andmetüüpe mudelisse juurde lisada. Seisundidiagrammi saab luua ainult Enterprise Architect vahendiga.

Autori arvates sobib Enterprise Architect registripõhiste infosüsteemide projekteerimiseks paremini kui DB-MAIN, samuti võiks seda kasutada õpetamisel. Selleks on mitmeid põhjuseid.

1. Enterprise Architecti on mugavam kasutada, diagrammid on loetavamad.
2. DB-MAIN ei võimalda luua seisundidiagramme. Samas on need oluliseks sisendiks kasutusjuhtude leidmisele.
3. Enterprise Architecti abil saab koostada CRUD maatriksit.
4. Enterprise Architect võimaldab kontseptuaalsel andmemudelil näidata agregatsioone ja rolle.
5. Kuigi Enterprise Architect abil kontseptuaalsest andmemudelist füüsilise andmemudeli koostamine võtab aega, on see täiuslikum DB-MAINis loodud mudeliga võrreldes.
6. Enterprise Architect abil on võimalik juba mudeli tasemel kirjeldada ärireeglite kontrollimiseks mõeldud andmebaasiobjekte.

DB-MAIN eeliseks on, et selle abil saab läbida loogilise disaini etapi ning mudelite teisendamine on Enterprise Architectiga võrreldes lihtsam.

Kokkuvõte

Bakalaureusetöö eesmärgiks oli kahe modelleerimisvahendi: Enterprise Architect (versioon 11) ja DB-MAIN (versioon 9.2.0) võrdlus registripõhiste infosüsteemide projekteerimise seisukohast – kumba vahendit on parem kasutada niisuguste infosüsteemide projekteerimiseks ja ka selle õpetamiseks. Selleks projekteeris autor mõlema vahendiga tervishoiuasutuse infosüsteemi vastuvõtuaegade funktsionaalse allsüsteemi ning selle poolt kasutatava andmebaasi alamosa.

Töö alguses tutvus autor Enterprise Architect ja DB-MAIN vahenditega. Järgmisena projekteeris autor töö eesmärgi saavutamiseks tervishoiuasutuse infosüsteemi vastuvõtuaegade funktsionaalse allsüsteemi ja selle poolt kasutatava andmebaasi alamosa. Tegemist on näitega registripõhise infosüsteemi projekteerimisest. See süsteem on kavandatud spetsiaalselt antud modelleerimisvahendite võrdlemiseks. Kuigi projekteerimine on toodud lihtsustatud kujul, on antud töös siiski esitatud üks versioon tervishoiuasutuse infosüsteemi vastuvõtuaegade allsüsteemist. Näiteks on dokumendis toodud ülevaade tervishoiuasutuse ja selle infosüsteemi eesmärkidest; kirjeldatud, millised põhilised sündmused tervishoiuasutuse tegevusele mõju avaldavad ja kuidas terviksüsteemi allsüsteemideks saab jagada. Täpsemalt on kirjeldatud vastuvõtuaegade allsüsteemi ja selle poolt kasutatavaid registreid: eesmärke, allsüsteemi ühte põhiprotsessi ja kasutusjuhtusid. Välja on toodud ärireeglid, millega vastuvõtuaegade registris tuleb arvestada. Projekteeritud on vastuvõtuaegade funktsionaalse allsüsteemi poolt kasutatav andmebaas – toodud on nii andmebaasi loogiline kui ka füüsiline andmemudel.

Töö eesmärk sai täidetud, modelleerimisvahendeid on võrreldud nii strateegia, detailanalüüsi kui ka disaini etappi silmas pidades. Autor jõudis järeldusele, et Enterprise Architect on parem vahend registripõhiste infosüsteemide projekteerimiseks ja selle õpetamiseks kui DB-MAIN. Põhjendusena võib esile tuua, et Enterprise Architecti on mugavam kasutada, diagrammid on loetavamad, füüsiline andmemudel täiuslikum, võimalik kirjeldada ärireeglite kontrolli realisatsiooni, füüsilist andmemudelit saab teha mitmete andmebaasisüsteemide jaoks.

Summary

The purpose of the Bachelor's thesis was the comparison of two modeling tools: Enterprise Architect (version 11) and DB-MAIN (version 9.2.0), from the viewpoint of designing register-based information systems – which tool is better to use in designing and teaching the design of such information systems. For this, the author used both tools to design the functional subsystem for scheduling visits, part of the information system of a healthcare institution, as well as the database section to be used by it.

At the start of the work, the author achieved familiarity with the tools Enterprise Architect and DB-MAIN. Next, the author designed the scheduling subsystem of a healthcare institution's information system and the database section it uses, in order to achieve the purpose. This is an example of designing a register-based information system. The system is intended specifically to compare these modeling tools. Although the design is provided in a simplified manner, nevertheless the work does present a version of a visit scheduling subsystem for a healthcare institution's information system. For example, the document includes an overview of the purposes of the healthcare institution and its information system: a description of the main events that affect the operations of the healthcare institution, and how the whole system can be decomposed into subsystems. There is a specific description of the visit scheduling subsystem, and the registers it used: the purposes, the subsystem's core process, and the use cases. Business rules that must be taken into account in the register of visit times are highlighted. The database to be used by the visit scheduling functional subsystem is designed – both the logical and the physical data models of the database are provided.

The goals of the work were achieved; the modeling tools have been compared within the context of the strategy phase, the detailed analysis phase, and the design phase. The author has reached the conclusion that Enterprise Architect is a better tool for designing register-based information systems and teaching their design than DB-MAIN. Among the justifications for this preference are Enterprise Architect's superior ease of use and diagram readability, the completeness of the physical data model, the ability to describe the implementation of business rule enforcement, and the ability to create a physical data model for several database systems.

Kasutatud kirjandus

1. Aadressandmete süsteemi kehtestamine. Vabariigi Valitsuse 28. juuni 2007. a määrus nr 184. Elektrooniline Riigi Teataja. [WWW] https://www.riigiteataja.ee/ert/act.jsp?id=12848733 (22.02.2015)
2. Andmekaitse Inspektsioon. (2013). Isikukoodi kasutamise juhend. [WWW] http://www.aki.ee/sites/www.aki.ee/files/elfinder/article_files/Isikukoodi%20kasutamise%20juhend.pdf (22.02.2015)
3. Connolly, T. M., Begg, C. E. (2001) Database systems. A Practical Approach to Design, Implementation and Management. Third Edition. Pearson Education. 1236 p
4. Eessaar, E. (2012a) Teema 10. Loogiline disain. CASE. Loengukonspekt õppeaines „Andmebaasid I“
5. Eessaar, E. (2012b) Teema 7. Andmebaaside projekteerimine: strateegiline analüüs ja detailanalüüs. Loengukonspekt õppeaines „Andmebaasid I“
6. Eessaar, E. (2014) Teema 10. Loogiline disain. CASE vahendid. Loengukonspekt õppeaines „Andmebaasid I“
7. Eessaar, E. (2015a) Andmebaaside projekteerimiseks kasutatavad mudelid. Loengukonspekt õppeaines „Andmebaasid I“
8. Eessaar, E. (2015b) Ülikooli infosüsteemi õpingukavade allsüsteem. Näiteprojekt õppeainetes „Andmebaasid I“ ja „Andmebaasid II“
9. Eesti E-tervise Sihtasutus – Digiregistratuur [WWW] http://www.e-tervis.ee/index.php/et/eesti-etervise-sihtasutus/tervise-infosusteem/arendused/digiregistratuur (07.04.2015)
10. Eesti Haigekassa tervishoiuteenuste loetelu. Elektrooniline Riigi Teataja. [WWW] https://www.riigiteataja.ee/akt/129122013057 (22.02.2015)
11. Ehitusseadus. Elektrooniline Riigi Teataja. [WWW] https://www.riigiteataja.ee/akt/728982?leiaKehtiv (22.02.2015)
12. ESTERM [WWW] http://mt.legaltext.ee/esterm/ (22.02.2015)

13. Fowler, M. (2007). UMLi kontsentraat. 3.redaktsioon. Objektmodelleerimise standardkeele UML2.0 lühijuhend. Cybernetica AS
14. Kabin, K. (2014) Tervishoiuasutuse infosüsteemi broneeringute allsüsteem. Aineprojekt õppeaines „Andmebaasid I“
15. Lang, L. (2003) UML-i õppematerjal. Bakalaureusetöö [WWW] http://minitorn.tlu.ee/~jaagup/kool/java/tarkvara/liina_uml.pdf (11.04.2015)
16. Rahvastikuregistri seadus. Elektrooniline Riigi Teataja. [WWW] https://www.riigiteataja.ee/akt/12806791?leiaKehtiv (22.02.2015)
17. REVER S.A. (2009) – DB-MAIN – DB-MAIN 9 Reference Manual [WWW] http://www-db.deis.unibo.it/courses/SI-T/DOCS/DB-MAIN-Reference-Manual.pdf (22.02.2015)
18. REVER S.A. (2012) – DB-MAIN – Tutorial: DB-Main 9.1.1 [WWW] http://each.uspnet.usp.br/sarajane/wp-content/uploads/2014/09/Tutorial-DB-Main.pdf (22.02.2015)
19. REVER S.A. (2015a) – DB-MAIN: a modeling framework [WWW] http://www.db-main.eu/?q=en/node/19 (09.04.2015)
20. REVER S.A. (2015b) – DB-MAIN: a data-architecture tool [WWW] http://www.db-main.be/?q=en (09.04.2015)
21. Sihtasutus Jõgeva Haigla põhikiri. Kinnitatud Jõgeva maavanema 11. augusti 2005 korraldusega nr 594
22. Sparx Systems (2015a) – Enterprise Architect – UML tool for analysis, design and implementation [WWW] http://www.sparxsystems.eu/enterprisearchitect/ea-purchase/ (07.04.2015)
23. Sparx Systems (2015b) – Enterprise Architect – Introduction to Enterprise Architect [WWW] http://www.sparxsystems.com/enterprise_architect_user_guide/9.3/introduction/introduction.html (07.04.2015)
24. Sparx Systems (2015c) – Enterprise Architect – UML Diagrams [WWW] http://www.sparxsystems.com/enterprise_architect_user_guide/8.0/modeling_languages/umldiagrams.html (07.04.2015)
25. Sparx Systems (2015d) – Enterprise Architect – Structural Diagrams [WWW] http://www.sparxsystems.com/enterprise_architect_user_guide/8.0/modeling_languages/structuraldiagrams.html (07.04.2015)

26. Tervishoiutöötajate riikliku registri põhimäärus. Elektrooniline Riigi Teataja. [WWW] https://www.riigiteataja.ee/akt/119062012024 (22.02.2015)
27. Vallaste, H. (2000) E-Teatmik [WWW] http://www.vallaste.ee (19.05.2015)