TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond
Informaatikainstituut
Infosüsteemide õppetool

Kasutajaliideste mudelipõhine arendamine EMF Forms abil

Bakalaureusetöö

Üliõpilane: Oliver Oidekivi
Üliõpilaskood: 134285 IABB
Juhendaja: Mart Roost

Tallinn
2016

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*)

Annotatsioon

Käesolev bakalaureuse töö käsitleb süsteemianalüüsi projektide kvaliteedi tõstmise võimalusi läbi mudelipõhise arenduse, kasutades selle jaoks EMF Forms raamistikku. Töö käigus uuritakse MDA-põhise meetodi kasutamise võimalikkust ning selle erinevusi tavapärase süsteemianalüüsi protsessist.

Töö esimeses pooles kirjeldatakse üldiseid mudelipõhise arenduse ja lõppkasutaja arendusse kaasamise teooriaid. Lisaks, käsitletakse selliseid Eclipse raamistikke nagu EMF ja EMF Forms. Töö teises pooles analüüsitakse õppeinfosüsteemi ning selle põhjal luuakse realisatsioon kasutades eelnevalt kirjeldatud Eclipse raamistikke.

Töö tulemusena on loodud õppeinfosüsteemi kirjeldav EMF mudel ja sellest genereeritud EMF Forms vaatemudelite prototüüp. Lisaks, on kirjeldatud mõlema meetodi erinevused ja uue protsessi eelised ja puudused.

Lõputöö on kirjutatud eesti keeles ning sisaldab teksti 58 leheküljel, 4 peatükki, 24 joonist, 1 tabel.

Abstract

The bachelor's thesis is about improving the quality of system analysis projects through model driven architecture using EMF Forms framework. This work examines the possibility of using an MDA-based approach and analyses its differences with conventional system analysis methods.

The first part of this work describes the theories behind MDA and EUD. Additionally, the main Eclipse frameworks used in this work, EMF and EMF Forms, are described. In the second part of this work an analysis is conducted on the study information system and a software prototype is created using previously described Eclipse frameworks.

As a result of this work an EMF model describing the study information system is created and a software prototype using EMF Forms viewmodels is generated. Further, the differences between both methods and the advantages and disadvantages of the new method are described.

The thesis is in Estonian and contains 58 pages of text, 4 chapters, 24 figures, 1 table.

Lühendite ja mõistete sõnastik

EMF	***Eclipse Modeling Framework*** Eclipse raamistik, mis võimaldab teostada mudelipõhist arendust.
EMF Forms	***EMF Forms*** Eclipse raamistiks kasutajaliideste genereerimiseks EMF mudeli põhjal.
Ecore	***Ecore*** EMF keskne mudel.
OMG	***Object Management Group*** Rahvusvaheline mittetulunduslik standardiseerimise konsortsium, mis tegeleb muuhulgas modelleerimise ja mudelipõhiste standardite loomisega.
Vaatemudel	***Viewmodel*** Liides, mis esitab andmeid, mida soovitakse näidata.
ÕIS	***Study Information System*** Õppeinfosüsteem, haldab õppetöö jaoks vajalikku informatsiooni.
UML	***Unified Modeling Language*** OMG standardi põhine üldotstarbeline noteeringukeel üldiselt objektipõhise keerulise tarkvara spetsifitseerimiseks ja visualiseerimiseks.
MDA	***Model Driven Architecture*** Tarkvaraarenduse meetod.
TTÜ	***Tallinn University of Technology*** Tallinna Tehnikaülikool, õppeasutus Eestis.
Metamudel	***Metamodel*** Mudeli mudel.

IDE	***Integrated Development Environment*** Tarkvara arenduse keskkond.
XML	***Extensible Markup Language*** Märgistuskeel
XMI	***XML Metadata Interchange*** Standard, mis kirjeldab metadandmete vahetamist XML abil.
EUD	***End-User Development*** Lõppkasutajapoolne arendamine.
REST	***Representational State Transfer*** Rakenduste vahelise suhtluse protokol.
JSON Forms	***JSON Forms*** Kasutajaliideste loomise raamistik.
Generaator	***Generator*** Tekitab uue koodi sisendi põhjal.
CORBA	***Common Object Request Broker Architecture*** OMG standard.
SOAP	***Simple Object Access Protocol*** Rakenduste vahelise suhtluse protokol. Sõnumite saatmise ja saamise formaat.
CASE	***Computer-Aided Software Engineering*** Tarkvara disainimiseks ja realiseerimiseks mõeldud tööriistad.
JBoss	***Jboss Application Server*** Jboss Java-põhine rakendusserver.
J2EE	***Java 2 Enterprise Edition*** Platvormist sõltumatu Java-põhine keskkond, milles saab arendada, konstrueerida ja juurutada veebipõhiseid ettevõtete rakendusi läbi veebi.

EclipseUML	***EclipseUML*** Tööriist, mille abil saab Ecore mudeleid luua.
EMF Client Platform	***EMF Client Platform*** Eclipse raamistik EMF-põhiste kliendirakenduste loomiseks.
WYSIWYG	***What You See Is What You Get Editor*** Teksti ja graafika redigeerimise käigus kuvatakse samal ajal teisel vormil hetke tulemust, mis on võimalikult sarnane lõpptulemusele
BPM	***Business Process Management*** Operatsioonijuhtimise ala, mis keskendub ettevõtete äriprotsesside haldamisele ja optimeerimisele.
BPMN	***Business Process Modeling Notation*** Äriprotsesside modelleerimise keel.

Jooniste nimekiri

Joonis 1. MDA arhitektuur 17
Joonis 2. EMF mudeli komponendid 24
Joonis 3. EMF mudel näide 25
Joonis 4. EMF Java mudel näide 26
Joonis 5. EMF XML mudel näide 26
Joonis 6. Lihtsustaud Ecore seoseid väljendav metamudel 27
Joonis 7. EMF Client Platform 30
Joonis 8. Väärtusvahetuse mudel 33
Joonis 9. Ärikasutusjuhtude kontekstdiagramm. 33
Joonis 10. Esialgne kontseptuaalne klassidiagramm 34
Joonis 11. Deklaratsioonide haldamise kasutusjuhtude diagramm 35
Joonis 12. Täpsustatud kontseptuaalne klassidiagramm 38
Joonis 13. Deklaratsiooni alustamise vaade 39
Joonis 14. Deklaratsiooni esitamise vaade 40
Joonis 15. Deklaratsiooni muutmise vaade 40
Joonis 16. Deklaratsiooni tagasi võtmise vaade 40
Joonis 17. Ecore valdkonnamudel 43
Joonis 18. Deklaratsiooni vaatemudel ja kasutajaliidese eelvaade. 45
Joonis 19. Texo kliendi ja serveri vaheline suhtlus 46
Joonis 20. Bizagi andmemudel näide 48
Joonis 21. MDA mudeli teisendamine 56
Joonis 22. Deklaratsiooni aine vaate näide 56
Joonis 23. Bizagi BPM Suite ülesehitus 57
Joonis 24. Bizagi Forms näidis 58

Tabelite nimekiri

Tabel 1. Ecore klassid.....27

Sisukord

1. Sissejuhatus 12
1.1 Taust ja probleem 12
1.2 Ülesande püstitus 12
1.3 Metoodika 13
1.4 Ülevaade tööst 13
2. Töö teoreetilised alused ja hüpoteesid 15
2.1 Model Driven Architecture 15
2.1.1 Põhjused 15
2.1.2 MDA arhitektuur 16
2.1.3 Eelised 19
2.1.4 Puudused 20
2.2 Lõppkasutaja kaasamine arendusse 21
2.2.1 End User Development 21
3. Töös kasutatavad tehnoloogiad ning analüüs 23
3.1 Töös kasutatavad tehnoloogiad 23
3.1.1 Eclipse 23
3.1.2 Eclipse Modeling Framework 24
3.1.3 EMF Forms 29
3.1.4 Veebiliides 31
3.2 Süsteemianalüüs 31
3.2.1 Eesmärgid ja nõuded 31
3.3 Uuritav süsteem 32
3.3.1 Skoop ja kirjeldus 32
3.3.2 Tavapärane meetod 37
3.3.3 Uus meetod 41
4. Realisatsiooni teostamine 42
4.1 Mudeli koostamine 42
4.2 Mudel 43
4.3 Mudelist kasutajaliidese genereerimine 44
4.4 Integratsioon olemasoleva süsteemiga 46
4.5 Lõppkasutaja kaasamine 46
4.6 Alternatiivsed tehnoloogiad 47

4.7 Realisatsiooni järeldused. 49
4.8 Edasiarendamise võimalused 50
Kokkuvõte 52
Summary 53
Lisa 1 MDA mudeli teisendamise muster 56
Lisa 2 EMF Forms prototüübi vaateid 56
Lisa 3 Bizagi Studio tarkvara 57

1. Sissejuhatus

Bakalaureusetöö on kirjutatud teemal „Kasutajaliideste mudelipõhine arendamine EMF Forms abil“. Teema valiti kuna autor soovib uurida kas tavapärast süsteemianalüüsi protsessi on võimalik täiustada kasutades uudsemaid mudelipõhise arenduse ja lõppkasutaja kaasamise põhimõtteid.

Antud peatükis tutvustatakse käesoleva töö tausta ning ülesande püstitust. Selgitatakse miks antud teema on oluline ning kirjeldatakse millise metoodikaga jõutakse lõpptulemuseni. Lõpuks tehakse lühiülevaade tööst.

1.1 Taust ja probleem

Modelleerimise ja Süsteemianalüüsi õppeainetes, TTÜ erialadel, on tudengiprojektide tulemiks tavaliselt olnud UML mudelid, tekstidokumendid ja kasutajaliideste pildid. Analüüsitöö kvaliteedi tõstmisel nii tudengiprojektides kui ka ettevõtete töödes oleks palju abi MDA kontseptsiooni järgi kiiresti mudelist genereeritavatest töötavatest tarkvaraprototüüpidest kuna mudelipõhine arendus lubab tõsta arendajate produktiivsust samal ajal parandades loodava tarkvara kvaliteeti ja kasutamise võimalusi.

Töö võiks esmalt pakkuda huvi nii tudengitele kes tegelevad süsteemianalüüsi õppimisega, kui ka antud ainete õppejõududele, kes saavad selle töö ideid õppeaine õpetamisel kasutada. Teisest küljest võiks see pakkuda huvi ka tarkvaraarenduse ja analüüsimisega tegelevatele inimestele, kes saaksid selle pidevalt areneva tehnoloogia enda töödes kasutusele võtta.

1.2 Ülesande püstitus

Bakalaureuse töö peamiseks eesmärgiks on uurida mudelipõhise arenduse üldiseid tööpõhimõtteid ning sellele tuginedes konkreetsemalt Eclipse Modeling Framework tehnoloogia baaskomponentide ning EMF Forms raamistiku kasutusvõimalusi modelleerimise ning süsteemianalüüsiga seotud tarkvaraprojektides. Samuti uuritakse lõppkasutaja kaasamise võimalikkust ja võimalusi mudelipõhisesse tarkvaraarendusse *End-User Development* kontseptsiooni järgides. Lõpuks viiakse läbi näidisprojekt

Tallinna Tehnikaülikooli õppeinfosüsteemi (ois.ttu.ee) näitel. Näide koostatakse tudengi vaates õppeainete deklareerimise põhjal.

1.3 Metoodika

Eclipse modelleerimise tehnoloogia tööriistad (EMF Technology tools) pakuvad modelleerimise, mudeliteisenduse ja genereerimise võimalusi erinevat tüüpi mudelite jaoks. Valdkonnamudelist läbi nõuete püstitamise tavapäraste ekraanivormidega tarkvarani jõudmiseks läheb vaja minimaalselt kahte sellist tööriista: Eclipse Modeling Framework ning EMF Forms.

Eesmärkide saavutamiseks luuakse näidisvaldkonna EMF mudel ehk valdkonnamudel, mis on metamudel, sellest genereeritakse EMF Forms põhised ekraanivormide mudelid ning Java ja XML kood, mille põhjal moodustub töötav prototüüp/rakendus.

1.4 Ülevaade tööst

Antud töö koosneb 5-st peatükist. Esimene peatükk on sissejuhatus, mis annab ülevaate antud tööst ja oodatavatest eesmärkidest ning kasutatud metoodikast.

Teises peatükis kirjeldatakse antud töö teoreetilisi aluseid ja hüpoteese. Kirjeldatakse mudelipõhist arendust, selle loomise põhjuseid ning eeliseid ja puuduseid. Lisaks uuritakse lõppkasutaja arendusse kaasamise võimalusi.

Kolmandas peatükis kirjeldatakse antud töö realisatsiooniks kasutatud tehnoloogiaid. Kirjeldatakse Eclipse kui ettevõtet ning uuritakse nende poolt loodud raamistikke. Järgmisena kirjeldatakse tavapärast süsteemianalüüsi õppeainet TTÜ-s ning millised nõuded sealsetele projektidele on. Viimaks viiakse läbi näiteprojekt praeguse süsteemianalüüsi projektide nõuete järgi. Teostatakse analüüs õppeinfosüsteemi osale ning luuakse vajalikud mudelid. Lõpuks selgitatakse, mida peab analüüsi mudelite EMF realisatsiooni teostamisel arvestama.

Neljandas peatükis teostatakse analüüsi mudelite EMF realisatsioon eelneva, kolmanda peatüki, näiteprojekti alusel. Kirjeldatakse UML klassidiagrammi ning Ecore metamudeli erinevusi. Luuakse Ecore metamudel ning sellest genereeritakse EMF Forms kasutajaliidesed. Uuritakse võimalusi ühendada loodud tarkvara prototüüp hetkel

kasutatava süsteemiga. Selgitatakse, kas on võimalik kaasata lõppkasutaja arendusse ning kirjeldatakse alternatiivseid tehnoloogiaid, mida oleks saanud valitud tehnoloogiate asemel kasutada. Viimaks esitatakse autori järeldused realisatsiooni teostuse kohta.

Lõpus tehakse kokkuvõte antud töös saavutatud tulemuste kohta.

2. Töö teoreetilised alused ja hüpoteesid

Järgnevas peatükis kirjeldatakse üldiseid teoreetiliseid aluseid, mis antud töös kasutatakse. Uuritakse mudelipõhist arendust, selle loomise põhjuseid, arhitektuuri, positiivseid ja negatiivseid külgi. Lõpuks kirjeldatakse lõppkasutaja arendusse kaasamise põhimõtteid.

2.1 Model Driven Architecture

Järgnevas alapeatükis uuritakse lähemalt üldisemat MDA teooriat ning selle arendamise ja kasutuselevõtu soovi põhjuseid. Samuti uuritakse sellest saadavaid eeliseid ning ka kaasnevaid puuduseid. Viimaks hinnatakse, milline võiks olla selle kasu süsteemianalüüsile.

2.1.1 Põhjused

Model Driven Architecture on Object Management Group standardiseerimise organisatsiooni poolt arendatav tarkvara arhitektuuri raamistik. OMG sihiks on tagada, et kõik varem loodud, praegune ning ka tulevikus arendatav tarkvara oleks võimalik omavahel integreerida. 2000.ndate alguses põhinesid peaaegu kõik rakendusserveri põhised tooted OMG standarditel.

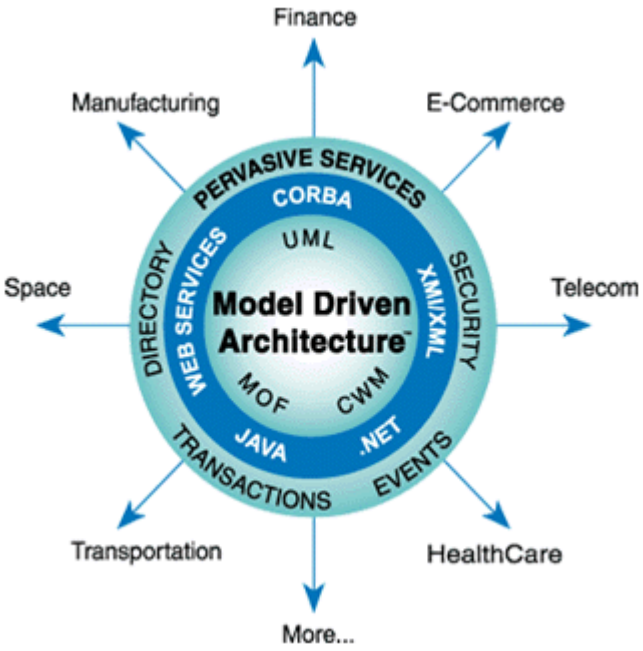
Nende esimeseks suureks saavutuseks oli CORBA arhitektuur, mis ühendas endas kõik tuntumad 20.sajandi lõpu tehnoloogiad nagu Internet, Java, SQL, Windows ja tuhandeid veel ning mille eesmärk oli tagada, et kõik rakendused suudaksid omavahel ühenduda olenemata arvutist, operatsioonisüsteemist, programmeerimisekeelest või võrgust. Lisaks sellele on OMG loonud UML modelleerimise standardi, mille eesmärk on võimaldada hinnata ning parandada loodava rakenduse arhitektuuri enne kui seda arendama hakatakse kuna muudatuste tegemine on varasemas staadiumis odavam. Samuti on nende loodud mudelite vaheline XMI standard, mis põhineb XML ja MOF tehnoloogiatel ning CWM, mille eesmärk on normeerida andmebaaside skeeme, nende skeemide muutmise ning OLAP mudelite jms väljendamist. Antud standardid toetavad CORBA arhitektuuri kuid ei ole sellega põimunud ning neid saab ja suuresti kasutatakse ka eraldi tarkvaraarenduse keskkondades nagu JAVA, .NET, XML/SOAP jms.

Probleem millega ettevõtted kokku puutusid oli vahetarkvara kasutamise vajadus, mis võis põhineda ühisel, patenditud või vahepealsel standardil. Vahetarkvara eesmärk oli tagada erinevate firmade ning nende keskkondade vahelist koostalitusvõimet, näiteks pangatransaktsioonid, sõnumite saatmine jne. Suurtel ettevõtetel oli peaaegu võimatu standardiseeruda ainult ühe sellise lahenduse peale kuna erinevatel osakondadel olid erinevad nõudmised ja ootused ning isegi kui suudeti seda saavutada, siis teiste organisatsioonide ja näiteks B2B veebipoodidega suhtlemiseks oli siiski vaja lisalahendusi. Peale selle, arendati valdkonnas pidevalt uusi vahetarkvara lahendusi juurde, mis lubasid eeliseid enda kasutuselevõtuga, kuid vana asendamine uuega tõi ettevõtetele kaasa lisakulutusi ning oli töövoolu segav. Algselt oli lootus, et lõpuks selgub kindel võitja millele langeb enamuse valik ning mis stabiliseerib ning ühtlustab vahetarkvarade kasutamise valiku, kuid OMG aktsepteeris, et selleni suure tõenäosusega ei jõuta.

Lahendusena nähti mudelipõhise arenduse kasutuselevõttu. MDA idee põhines OMG modelleerimise standarditel ning selle eesmärgiks oli pakkuda ettevõtetele paindlikkust läbi võimaluse pidevalt stabiilsest mudelist koodi tuletada kuna see tarkvara arhitektuuri aluseks olev infrastruktuur muutub ajaga. Lisaks lubati ettevõtetele suuremat investeeringutasuvust kuna rakendusi ja domeenimudeleid on võimalik taaskasutada kogu tarkvara elutsükli vältel. [1]

2.1.2 MDA arhitektuur

MDA põhiline aspekt on selle võime katta tervet tarkvara arenduse elutsüklit, hõlmates analüüsi, disaini, programmeerimist, testimist, komponentide kokkupanemist ja ka kasutuselevõttu ning hooldust. Eesmärgiks on hõlmata suurt osa levinud teenuseid mis rakendustega kaasnevad.



Joonis 1. MDA arhitektuur

Allikas: Frank Truyen, *The Fast Guide to Model Driven Architecture*

Peamised MDA mõisted, millele see tugineb:

- **Süsteem** – Olemasolev või arendatav tarkvarasüsteem
- **Mudel** – Vormikohane süsteemi funktsioonide, struktuuri ja käitumise kirjeldus vaadeldavas kontekstis ja kindla kasutaja vaates. Tihti esitatakse läbi joonistuste, modelleerimiskeele abil, ja dokumentatsiooni teksti.
- **Arhitektuur** – Komponentide ning nende vaheliste seoste kirjeldus ning reeglid kuidas omavaheline suhtlus toimib. Arhitektuur esitatakse läbi omavahel seostatud mudelite.
- **Vaatepunkt ja MDA vaatepunktid** – Vaatepunkt on abstraktsiooni tehnika, mis võimaldab kindlatele süsteemi probleemidele keskenduda, eemaldades ebaolulise. Esitatakse läbi ühe või mitme mudeli. MDA kirjeldab vaikimisi kolm vaatepunkti:
    - Struktuurist ja arhitektuurist sõltumatu vaatepunkt (*computation independent viewpoint*) – Keskendub kontekstile ja tarkvara nõudmistele, kuid mitte struktuurile või töötlemisele

    - Platvormist sõltumatu vaatepunkt – Keskendub süsteemi võimetele väljaspool kindalt platvormi
    - Platvormist sõltuv vaatepunkt – Lisab platvormist sõltumatule vaatepunktile detailid kindla süsteemi kohta.

- **Platvorm**– Platvorm on hulk alamsüsteeme ja tehnoloogiaid, mis pakuvad sidusat funktsionaalsust läbi liideste ja kasutusmustrite. Näiteks operatsioonisüsteemid.
- **Mudeli muutmine** – Ühte mudelit on võimalik teiseks muundada, kui need asuvad samas süsteemis. Kombineerib platvormist sõltumatu mudeli lisainformatsiooniga, et luua platvormispetsiifiline mudel.
- **Realiseerimine** – Kirjeldus, mis esitab kõik vajaliku informatsiooni, et luua süsteem ning panna see tööle. Esitab objekti loomiseks vajaliku informatsiooni ning tagab, et sellele on kättesaadavad vajalikud teenused.

MDA kirjeldab kolm eelnevalt mainitud vaatepunktile vastavat mudelit: Arhitektuurist ja platvormist sõltumatud mudelid ja platvormist sõltuv mudel. Iga nende mudeli sees on võimalik kirjeldada omakorda hulka mudeleid, mis keskenduvad aina kitsamale vaatepunktile, näiteks kasutajaliides, informatsioon, arhitektuur jms.

MDA-põhise rakenduse loomise esimeseks sammuks on alati platvormistsõltumatu mudeli loomine UML keele abil, mida saab hiljem teisendada spetsiifilisema tehnoloogia keelde. Keerukamate süsteemide puhul võib luua rohkem, üksteisest sõltuvaid, mudeleid. Lisaks tavapärasele kihtide vahelisele vertikaalsele mudelite teisendamisele saab sama taseme mudeleid ka horisontaalset teisendada, kasutades selleks ühte kolmest MDA tüüpmudelitest või rakendades kõikides vaatepunktides ühtset arhitektuuri. Lisaks algsele mudelile peab olema tagatud lisainformatsioon mudeli teisendamise jaoks. Üldine muster on: algse mudeli ja teisendamise reeglite põhjal genereeritakse sihtmärgiks olev mudel (vt. Lisa 1). [2]

2.1.3 Eelised

MDA raamistikku on rakendatud nii suurtes kui väikestes organisatsioonides. Selle lubadus on toimimise korral aidata masinloetavate mudelite loomist, mis tagaks pikaajalise paindlikkuse.

Oodatav kasu esineks erinevates aspektides:

- **Tehnoloogia iganemine** – Loodavaid rakendusi saaks kergemini ühildada ja toetada olemasolevate süsteemidega.
- **Porditavus** – Olemasolevat funktsionaalsust saab lihtsamini uutesse keskkondadesse üle kanda vastavalt ärivajadustele.
- **Produktiivsus** – Automatiseerimisega kulutavad arendajad vähem aega programmeerimise koodi kirjutamisele ning saavad keskenduda süsteemi loogika arendamisele.
- **Kvaliteet –** Paraneb süsteemi üldine kvaliteet kuna suureneb genereeritavate komponentide töökindlus ning üldine kooskõlalisus.
- **Integratsioon –** Lihtsustub pärand- ja välissüsteemide vaheliste ühildussildade loomine.
- **Hooldamine –** Projekti olemasolu masinloetavas vormis annab analüütikutele, arendajatele ja testijatele ligipääsu süsteemi tehnilistele kirjeldustele, mis hõlbustab nende hooldustöö tegemist.
- **Testimine ja simulatsioon –** Mudeleid saab valideerida nõuete põhjal ning neid saab kasutada loodava süsteemi käitumise simuleerimiseks.

[2]

2.1.4 Puudused

MDA arhitektuuri kasutuselevõtu eesmärgiks on kaasa tuua mitmeid positiivseid muudatusi tarkvaraarenduses, kuid on mõningad lubadused mida on tarvilik lähemalt uurida ning mis selgub, et on tingimuslikud.

MDA-põhine kirjandus ning mõningad selle põhiste tööriistade tootjad väidavad, et antud raamistiku kasutuselevõtuga väheneb loodava tarkvara arenduse aeg ning arendajad saavad keskenduda programmiloogikale kuna ei pea õppima detailselt sihtplatvormi ja infrastruktuuri kohta. See peaks võimaldama kiiresti ja lihtsalt rakenduste kasutuselevõttu samal ajal kui siht-platvormid muutuvad, mis omakorda peaks tugevasti kasu tooma erinevate tarkvarade koostalitusvõimele. Selle tõestamiseks tehti Ameerika Ühendriikide Kaitseministeeriumi tellimusel Carnegie Mellon Ülikoolis uuring, mis neid hüpoteese tõestada üritas.

Esimesena vaadati kas MDA vähendab arendamiseks kulunud aega. Selle jaoks võrreldi kui palju aega läks Java 2 Enterprise Edition rakenduse arendamiseks mudelipõhise lähenemisega ning tavapärase IDE-ga. Antud hüpotees lükati mudeli probleemi meetodi kasutades osaliselt ümber. Mudelipõhiselt arendamiseks kulunud aeg ei vähene esimese loodava rakenduse puhul ning võib isegi kasvada, kuid järgnevate identsele platvormile loodavate rakenduste loomise aeg võib väheneda suuresti. Kui platvorm millele hakatakse uut rakendust arendama ei ole identne algsega, siis on vajalik tööriistade seadeid muuta, mis vähendab mudelipõhise arendamisega säästetud aega. Lisaks kulub aega erinevate ning küllaltki keeruliste MDA-põhiste keskkondade õpetamisele arendajatele.

Teiseks uuriti arendaja võimalust keskenduda ainult programmiloogikale ehk väidet, et ta ei pea teadma detaile platvormist ega selle infrastruktuurist, millele tarkvara arendatakse. Hüpoteesi tõestamise jaoks tehti katse, kus arendaja pidi mõistma J2EE tehnoloogiat kontseptuaalsel tasemel, kuid ei pidanud sellest ega JBoss serverist detailselt aru saama, samal ajal kui ta arendas J2EE rakendust JBoss jaoks kasutades MDA-põhist vahendit. See hüpotees lükati täielikult ümber. Selgus, et enne koodi genereerimist on vaja seadistada tööriistad ja teisendamisereeglid ning luua mudel, mis kõik nõuab üldiselt põhjalikke teadmisi platvormist millele arendust tehakse. Arendaja ei pea ainult infrastruktuuri koodi ise kirjutama, kuid tal peavad teadmised eksisteerima.

Antud uuringu üheks järelduseks oli, et saadaolevad vahendid ei ole veel piisavalt arenenud, et MDA tehnoloogia potentsiaali täielikult ära kasutada, kuid kui suudetakse saavutada täielikult platvormist sõltumatus, siis on mudelipõhine arendus järgmine samm CASE vahendite ja arenduskeskkondade jaoks. [3]

2.2 Lõppkasutaja kaasamine arendusse

Üks moodus kuidas mudelipõhisest arendusest rohkem kasu saada ning kasutajate muutuvaid vajadusi rahuldada on kasutajate endi süsteemiarendusse kaasamine, rakendades selle jaoks kasutajapoolse arenduse ehk *End-User Development* põhimõtteid. Järgnevas alapeatükis uuritakse üldiseid kasutajapoolse arenduse põhimõtteid ning selle positiivseid ja negatiivseid külgi ning võimalusi mudelipõhises arenduses rakendamiseks.

2.2.1 End User Development

End-User Development on kogum meetodeid, tehnikaid ja tööriistu, mis võimaldavad lõppkasutajal teatud kindlas arengujärgus luua, muuta või laiendada mingit tarkvara osa, seejuures omamata detailseid teadmisi ja oskuseid tarkvara arendusest. USA tööjõustatistika büroo ennustuste kohaselt oli 2012. aastaks Ameerikas 3 miljonit professionaalset programmeerijat, kuid üle 55. miljoni inimese, kes kasutasid enda töös arvutustabeleid, andmebaase või kirjutasid valemeid ja käsklusi. Teise uuringu kohaselt ennustati, et 2014. aastaks on olukord, kus üle 25% äritarkvarast luuakse mitte-professionaalsete inimeste poolt.

Lõppkasutaja kaasamiseks on mitmeid põhjuseid. Arendajatel puuduvad vajalikud teadmised valdkonnast ning lõppkasutajad ei suuda seda analüüsi käigus piisavalt hästi edasi anda. Tarkvaraarendustsüklid on liiga pikad ning kasutajate soovid ja nõudmised muutuvad liiga kiiresti. Lõppkasutajatel puuduvad professionaalse arendaja oskused ning seetõttu ei saa EUD puhul kasutada traditsioonilisi arendusmetoodikaid.

Peamised EUD tööriistade eesmärgid on olla lihtsasti õpitavad ja kasutatavad lõppkasutaja jaoks, kuid peaksid samal ajal võimaldama ka ekspertidel töötada keerukamate projektide arendamisega. See tähendab, et nendel tööriistadel peavad olema lihtsad ja arusaadavad kasutajaliidesed, mis võimaldavad luua detailseid ja täielikke projekte. EUD eesmärk on vähendada õppimiskõverat keerukamate projektide loomise

jaoks mõeldud tööriistade puhul. Tihti EUD laseb kasutajatel muuta ja modifitseerida varasemalt programmeerijate poolt loodud elemente. [4]

Kasutajapoolsel arendusel ning mudelipõhisel arendamisel on erinevad eesmärgid, kuid 2013. aastal *Springer'i* lõppkasutajapoolse arendamise artiklite kogumis avaldatud mobiiliarenduse põhisest uuringust selgub, et neid kahte ning täpsemalt EUD põhimõtteid ja Eclipse Modeling Framework'i, mida uuritakse ja selgitatakse järgnevates peatükkides täpsemalt, on võimalik koos kasutada, saavutades seejuures suuremat kasu.

Peamised positiivsed küljed kuidas need kaks tehnoloogiat teineteist mõjutasid:

- Metamudelid olid kasulikud valitud lõppkasutajapoolse arenduse notatsiooni realisatsiooni kirjeldamisel kuna metamudelite loomine nõuab täpset notatsiooni mõistete tundmist. Lisaks sellele on metamudel tööriist, mis otsib võimalusi mudelite lihtsustamiseks.

- Mudelipõhine arendus lihtsustab tarkvarakeskkondade arendajate tööd, sealhulgas lõppkasutaja arenduse teostuse raamistike loomist.

- Metamudelite kasutamine terve arendustsükli vältel toob kaasa terviklikkuse tagamise, mis aitab kaasa tarkvara disainimis tsüklile.

Samas olid ka teatud probleemid. Metamudelite koostamine ei olnud piisavalt tõhus alternatiivsete notatsioonide võimaluste otsimisel. Paber ning tarkvara abil visuaalsed prototüübid olid selleks tööks eelistatud kuna nõudmisi tekkis pidevalt juurde ja need muutusid keerukamaks ning metamudelite koostamine ei olnud sellistes tingimustes sobilikuim vahend. [5]

Teises *Springer'i* artiklis on väidetud, et EUD ei ole kõigest koodi muutmine või kohandamine vaid metamudelite algväärtustamine, mis ühendab selle MDA teooriaga ning on kooskõlas antud bakalaureusetöö eesmärkidega. [6]

3. Töös kasutatavad tehnoloogiad ning analüüs

Järgnevas peatükis antakse ülevaade töös kasutatud tehnoloogiatest, millel on potentsiaal aidata analüüsitöö kvaliteeti tõsta ning mida kasutatakse ka realisatsiooni teostamiseks. Lisaks analüüsitakse osa praegusest õppeinfosüsteemist praeguse analüüsimetoodika järgi ning projekteeritakse nõuded ja mudelid EMF Forms prototüübi realiseerimiseks. Viimaks, kirjeldatakse tavapärast ning loodavat süsteemianalüüsi protsessi.

3.1 Töös kasutatavad tehnoloogiad

Alapeatükis kirjeldatakse realisatsiooni teostamiseks kasutatavaid tehnoloogiaid ning nende eesmärke.

3.1.1 Eclipse

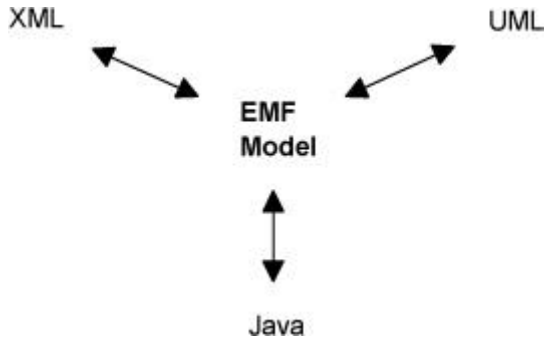
Eclipse.org asutati 2001. aastal üheksa tarkvaraarenduse turuliidri poolt, kelle hulka kuulusid sellised ettevõtted nagu IBM, Rational Software, Red Hat jt. 2003-nda aasta lõpuks kuulus nende liitu juba üle 80. ettevõtte. 2004-nda aasta algul restruktureeris Eclipse juhatus ettevõtte toimimise ning neist sai mittetulundusühing, mis toimib tänu oma liikmetasudele. Eclipse kommuuni juhtivaks organisatsiooniks sai kuu varem loodud Eclipse Foundation. MTÜ alustamise peamiseks põhjuseks oli soov luua tootjatest sõltumatu, vaba ning läbipaistev kogukond Eclipse ümber. Eclipse pakub vabavaralist tarkvara läbi oma ’Eclipse Public License’ litsentsi, mis tähendab, et seda tarkvara võib kasutada, muuta ja jagada tasuta. Lisaks, pakuvad nad professionaalseid teenuseid oma kasutajatele, kuid nad ei palka arendajaid, nendeks on üldiselt erinevate organisatsioonide poolt palgatud või iseseisvad vabatahtlikult töötavad inimesed. Peamine tarkvara mida Eclipse pakub on Eclipse IDE ning selle laiendused. [7]

Põhjus miks bakalaureusetöö tegemise jaoks Eclipse tarkvara valiti on nende loodud vabavara, mis on piisavalt laiendatav, et koostada kõik vajalikud mudelid, uurida EUD ja MDA kontseptsioone ning teostada projekti realisatsioon. Samuti, on nad toetatud paljude suurimate tarkvaraarenduse ettevõtete poolt.

3.1.2 Eclipse Modeling Framework

Üks Eclipse poolt pakutavatest ning antud töös kasutatavatest lahendustest on Eclipse Modeling Project alla kuuluv Eclipse Modeling Framework(Core).

EMF on Java-põhine modelleerimise raamistik ja koodigeneraator Eclipse IDE jaoks, mis laseb mudelipõhiselt ehitada tööriistu ja teisi aplikatsioone. Selle keskseks osaks on klassidiagrammi tüüpi Ecore mudel, mis on sarnane, kuid väiksema abstraktsusega, kui UML standardi põhjal modelleeritud diagramm. EMF ühendab endas Java, XML ja UML tehnoloogiad, mille eesmärgiks on võimaldada genereerida ühest neist vormidest ka teised ning vastavad vajalikud implementatsiooniklassid. EMF keskendub ainult klasside modelleerimisele, samas kui UML standardi põhimõte on luua mitmeid erinevat tüüpi diagramme, et keerukamaid süsteeme kirjeldada.



Joonis 2. EMF mudeli komponendid

Allikas: *Eclipse Modeling Framework - A Developer's Guide*

EMF üritab saavutada tasakaalu programmeerimise ning modelleerimise vahel ning tutvustada lihtsamaid MDA põhimõtteid. Tasakaalu all mõeldakse seda, et Java programmeerimise koodi ja mudelite vahelised seosed oleksid piisavalt lihtsad, et need oleksid kergesti mõistetavad nii programmeerijatele kui modelleerijatele, kuid, et see oleks jätkuvalt piisav, et tagada vajalikud rakenduste vahelised seosed. Lisaks, on EMF eesmärgiks suurendada produktiivsuse kasvu vähendades vajadust käsitsi programmeerimiskoodi kirjutada. [7]

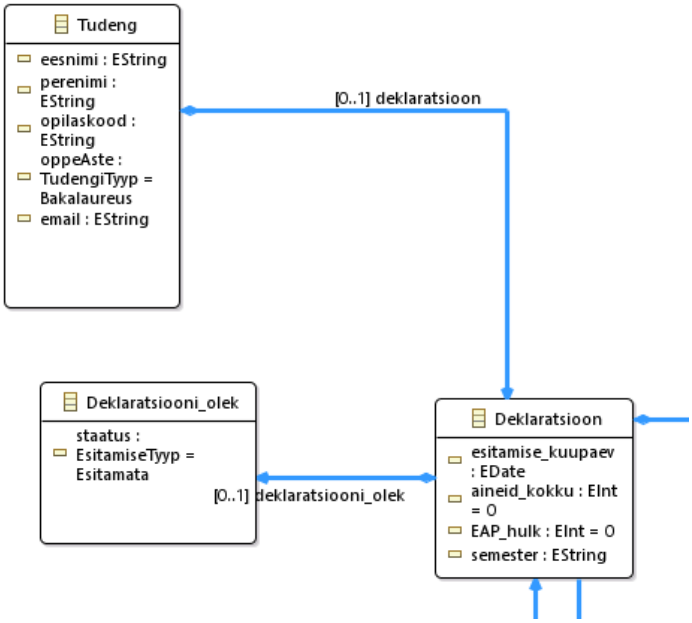
Kui üldise MDA arhitektuuri peamised kahtlused olid liiga suur ambitsioon, mudelite liigne või vähene keerukus ning kasutatavuse võimalused, siis EMF loomise eesmärgiks oli lihtsalt ja pragmaatiliselt võimaldada mudelipõhist arendust genereerides teatud osa üldisemast koodist, võimaldades seeläbi keerukamaid süsteeme ja töövahendeid

kasutusele võtta. EMF eesmärgiks on ka näidata, et modelleerimine on kasulik rohkem kui ainult dokumentatsiooni jaoks. [8]

Põhjus miks Eclipse IDE ja EMF raamistik valiti antud töö teostamiseks on nende vabavaralisus, laiendatavus ning küpsus. Mitmed UML põhised tööriistad on loodud EMF baasil, sealhulgas IBM'i *Rational Software Engineer*. Lisaks sellele on Eclipse IDE väga multifunktsionaalne ning levinud tarkvaraarenduses. *Zeroturnaround* ettevõtte poolt 2012. aastal tehtud harjumuste küsitlusest selgus, et kaks kolmandikku kõigist vastajatest, kes tegelesid tarkvaraarendusega, kasutasid enda töös Eclipse tehnoloogiaid. [9] [11]

3.1.2.1 EMF mudeli defineerimine

EMF mudel on UML klassidiagrammi alamhulk, mis kirjeldab objekti atribuute, nendevahelisi seoseid, võimalikke operatsioone ning kitsendusi. EMF kontseptuaalset mudelit saab defineerida Java liidese, UML klassidiagrammi või XML skeemi kaudu, mis võimaldab erinevate oskustega inimestel kirjeldada üks nendest vormidest ning edasi genereerida teised. [8]



Joonis 3. EMF mudel näide

UML klassidiagramm võib koosneda klassidest, nende atribuutidest ja operatsioonidest ning klasside vahelistest seostest. Klassinimi on vastavuses Java klassi definitsiooniga ja XML skeemi komplekstüübi definitsiooniga. Atribuudid on vastavuses Java get()/set() meetodite paaridega ning XML kihiliste elementide deklaratsioonidega. Seosed on

vastavuses Java get() meetodiga ja XML kihilise elemendi deklaratsiooni või komplekstüübiga.

```
public interface Tudeng extends EObject {
    /**
     * Returns the value of the '<em><b>Deklaratsioon</b></em>' containment reference.
     * <!-- begin-user-doc -->
     * <p>
     * If the meaning of the '<em>Deklaratsioon</em>' containment reference isn't clea
     * there really should be more of a description here...
     * </p>
     * <!-- end-user-doc -->
     * @return the value of the '<em>Deklaratsioon</em>' containment reference.
     * @see #setDeklaratsioon(Deklaratsioon)
     * @see oppeinfosysteem.OppeinfosysteemPackage#getTudeng_Deklaratsioon()
     * @model containment="true"
     * @generated
     */
    Deklaratsioon getDeklaratsioon();
```

Joonis 4. EMF Java mudel näide

EMF mudelite kirjeldamiseks kasutatakse kõrgema abstraktsuse tasemega kontseptsioone kui lihtsamate klasside ja meetodite jaoks. Näiteks, atribuudid kirjeldavad meetodite paare ning nad võimaldavad vaatlejate teavitamist(UI vaated), salvestamist, laadimist ja püsivat andmete hoidmist. Klasside vahelised seosed võivad olla kahesuunalised, säilitades seejuures viite terviklikkuse.

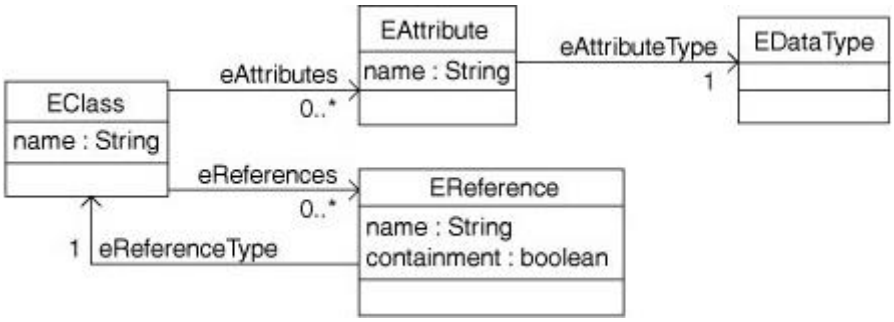
```
<?xml version="1.0" encoding="UTF-8"?>
<oppeinfosysteem:Tudeng xmi:version="2.0" xmlns:xmi="http://www.omg.org/XMI"
xmlns:oppeinfosysteem="http://www.example.org/oppeinfosysteem"
 eesnimi="Nimi" perenimi="Perenimi" opilaskood="111111IABB" email="nimi@ttu.ee">
  <deklaratsioon esitamise_kuupaev="2016-01-30T00:00:00.000+0200" aineid_kokku="
   EAP_hulk="30" semester="2015/16 Kevad">
    <deklareerimisperiood href="Dekanaat.oppeinfosysteem#//@deklareerimisperiood
  </deklaratsioon>
</oppeinfosysteem:Tudeng>
```

Joonis 5. EMF XML mudel näide

Ühtse mudeli defineerimiseks nendest kolmest „mudeli osast“ on vaja ühist terminoloogiat, mis neid kirjeldaks. Lisaks, on EMF tööriistade ja generaatori rakendamise jaoks vaja mudelit informatsiooni jaoks. Selle saavutamiseks kasutatakse metamudelit. [7]

3.1.2.2 EMF mudel

EMF mudeleid kujutatakse läbi Ecore mudeli. Ecore on samuti EMF mudel ning seega iseenda metamudel. Metamudel on mudeli mudel.



Joonis 6. Lihtsustaud Ecore seoseid väljendav metamudel

Allikas: *Eclipse Modeling Framework - A Developer's Guide*

Mõned Ecore mudeli väljendamise jaoks vajalikud Ecore klassid:

Tabel 1. Ecore klassid

EClass	Väljendab modelleeritud klassi. Sellel on nimi, null või enam attribuuti ja null või enam seost.
EAttribute	Väljendab modelleeritud atribuuti. Sellel on kohustuslik nimi ja tüüp.
EReference	Väljendab klasside vahelise seose ühte poolt. Sellel on nimi, boolean tüüpi lipp(signaal), mis näitab kas see väljendab tõkestatust ning viite sihtmärk, mis on omakorda klass.
EDataType	Väljendab atribuudi tüüpi. Võib olla primitiiv- või objekttüüp.

Loodava rakenduse mudeli klassistruktuuri kirjeldamise jaoks algväärtustatakse Ecore defineeritud klassid ehk luuakse uus mudel, kus on määratud kõik vajalikud andmed. Ecore klasside algväärtustamistel luuakse *core* mudel. Metamudelit on võimalik kirjeldada läbi Java liidese, XML skeemi, UML impordi/ekspordi või luues uue mudeli kasutades EMF Ecore redaktorit ja EclipseUML tööriistu. Andmete püsivaks salvestamiseks ning taasloomise võimaldamiseks kasutatakse XMI(XML Metadata

Interchange) vormi, mis on standard metaandmete serialiseerimiseks ehk objektide baitideks teisendamiseks eesmärgil andmed mällu talletada. EMF toetab ka relatsioonilise andmebaasi RDB skeemi järgi mudeli koostamist ja andmete formaadi täpsustamist. [7]

Bakalaureusetöö käigus luuakse metamudel kasutades EMF Ecore ja EclipseUML graafilist redaktorit.

3.1.2.3 EMF klasside genereerimine

Üks EMF suurimatest eelistest on produktiivsuse kasv tänu võimalusele genereerida Java programmeerimiskoodi. Alapeatükis vaadatakse kuidas Ecore diagrammist Java koodi genereerimine toimub ning millised on nende vahelised seosed.

Ecore klass(EClass) on vastavuses kahe Java klassiga: liidese ja selle liidese rakendamise klassidega. Iga genereeritud Java liidese klass laiendab(extends) EObject klassi, mis on EMF tehnoloogias iga modelleeritud klassi aluseks. EObject laiendab ise Notifier liidest, mis tähendab, et see saadab teavituse igakord kui objekti atribuute või objektide vahelisi seoseid on muudetud. See on oluline osa EMF tehnoloogiast kuna see võimaldab vaateid ja teisi sõltuvaid objekte uuendada.

Teistsuguste seoste korral võidakse luua ka keerukamad mustrid, näiteks kahesuunaliste seoste puhul, kuid genereeritud koodi üldine eesmärk on olla võimalikult efektiivne, arusaadav ja lihtne. Genereeritud klasside eesmärk ei ole täielikult kõike vajalikku koodi valmis luua vaid on mõeldud, et kasutaja lisab käsitsi enda jaoks vajalikke klasse ja meetodeid sellele koodile juurde. EMF võimaldab mitmekordset genereerimist ilma kasutaja tehtud muudatuste ära kustutamiseta. Selle jaoks lisab generaator meetodide juurde Javadoc'i dokumentatsiooni @generated tähise ning kontrollib igakord selle olemasolu ehk kui kasutaja eemaldab antud märgise dokumentatsioonist, siis vastavat meetodit või klassi üle ei kirjutata. Konfliktide korral säilitatakse kasutaja kirjutatud kood.

Lisaks Java klassidele genereerib EMF mudelist ka manifest faili, mis võimaldab mudelit Eclipse pluginana kasutada ning XML skeemi faili mudeli jaoks.

Generaatoril on lisaks mudelis kirjeldatud andmetele vaja ka muud kasutaja poolt määratavat lisainformatsiooni, näiteks kuhu failid salvestatakse, milliseid eesliiteid kasutada jms. See kõik salvestatakse generaatori mudelisse, mis käitub ümbrisena core

mudeli ümber ehk tegelikult EMF generaator kasutab oma tööks generaatori mudelit, mitte kirjeldatud core mudelit. Selle saavutamiseks loob EMF kaks erinevat faili: .ecore ja .genmodel. Esimene neist on XMI serialisatsioon core mudelist ja viimane on generaatori mudel, mis viitab algsele .ecore failile. Selle eesmärk on hoida kirjeldatud mudel ja ainult koodi genereerimise jaoks vajalikud andmed lahus. [7]

EMF võimaldab genereerida ka .edit ja .editor projekti failid, mille abil saab luua laiendusi mudeli vaatamiseks ja muutmiseks, kuid antud töös kasutatakse selle asemel EMF Forms raamistikku.

3.1.3 EMF Forms

Paljud rakendused on andmepõhised ning võimaldavad kasutajal otsida, vaadata ja muuta selle aluseks oleva andmemudeli andmete väärtusi. See kõik esitatakse enamasti vormipõhise asetustega kas arvuti-, mobiili- või veebirakendusena. Käsitsi sellist tüüpi kasutajaliideste arendamisel on mitmeid puuduseid: nähtavad süsteemikomponendid vajavad pidevat uuendamist vastavalt kasutajate tagasisidele või nende aluseks oleva andmemudelisse muudatuste tegemisel, muudatusi peab läbi viima käsitsi UI koodis, kasutajaliideseid peab hooldama ja töös hoidma, mis on kulukas ja aeganõudev ning tihti on olukord, kus sama rakenduse kõik vormid peavad vastama teatud ühistele nõuetele välimuses ja töös. Tavapärasemad UI arendamise tehnoloogiad nagu näiteks SWT, JavaFX ja Swing võimaldavad neid probleeme lahendada, kuid on tihti liiga keerukad ja mahukad, sest need ei keskendu ainult vormipõhistele liidestele. Selle jaoks on loodud EMF Forms raamistik.

EMF Forms genereerib kohandatavad vaatemudelid vastavalt EMF Ecore mudelile ning nende põhjal saab moodustada kasutajaliidesed. Formsi eesmärgiks on võimalikult lihtsalt ja efektiivselt luua vormipõhiseid liideseid, võimaldada kergesti andmemudeli muudatust ja liideste uuendamist ning vähendada tehniliste oskuste teadmiste vajadust.

EMF Forms abil genereerimise aluseks on kirjeldatud Ecore mudel ja renderdamise mootor, mis kasutab kindlat UI tööriistakomplekti, et tõlgendada mudelit ning moodustada selle põhjal liidesed. Valida saab näiteks Swing, SWT, JavaFX ja Web tehnoloogiate vahel. See tähendab, et valitud renderdajast sõltub ka millised liidesed luuakse. EMF Forms poolt loodud kasutajaliideseid saab integreerida samamoodi nagu käsitsi arendatud liideseid. [12]

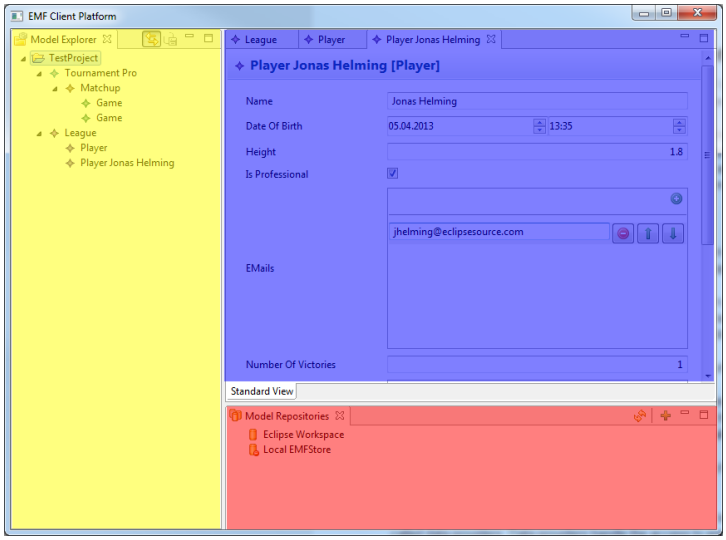
Teoreetiliselt võiks tehniliste oskuste vähene vajadus tähendada, et EMF Forms abil saaks rakendada EUD kontseptsioone kasutajaliideste genereerimisel ehk ka ilma IT-taustata inimesed oleksid võimelised selle arendamisega toime tulema.

3.1.3.1 EMF Client Platform

EMF Forms kuulub EMF Client Platform raamistiku alla ning on üks selle osakomponentidest. Kasutajate jaoks on kolm enimkasutatavat komponenti:

- Explorer(vt. Joonis 7. - kollane) – Võimaldab uute projektide ning elementide loomist ja näitab mudelite hierarhiat ning võimaldab seda muuta.
- Editor(vt. Joonis 7. - sinine) - EMF Forms põhjal loodud vormipõhine kasutajaliides.
- Repository Explorer(vt. Joonis 7. - punane) – Laseb laadida uusi andmeid teistest allikatest.

EMF Formsi saab kasutada ka eraldiseisva rakendusena teistest komponentidest. EMF Forms poolt genereeritud vaatemudelites tehtud muudatused kajastuvad EMF Client Platform rakenduses.



Joonis 7. EMF Client Platform

Allikas: *Getting started with the EMF Client Platform*

Muudatuste tegemisel vaatemudelitesse piisab ainult ECP rakenduse uuesti laadimisest, et muutused avalduksid kasutajale. [13]

3.1.4 Veebiliides

EMF Forms rakenduse veebis kasutamiseks on ühe viisina võimalik kasutada JSON Forms raamistikku. Mõlema eesmärgiks on kergesti luua andmemudelipõhiseid kasutajaliideseid ja mõlemad defineerivad kasutajaliidesed läbi erinevat tüüpi vaatemudelite, mis on omavahel ühilduvad. Eclipse IDE-s on võimalik automaatselt genereerida EMF Forms vaatemudelitest vajalik Javascript kood, mida saab JSON Forms abil veebilehel kasutada. [14]

JSON Forms põhineb Javascriptil ning selle kasutamiseks lisatakse vajalik kood, milleks on JSON tehnoloogia põhine skeem, AngularJS rakendusse. Nimetatud rakenduse saab kasutaja enda arvutisse laadida, mille tulemusel tekib veebilehe fail ning enda koodiga asendatavad Javascript failid. Loodud veebilehe faili saab internetti üleslaadida.

JSON Forms tehnoloogia puuduseks antud hetkel on EMF klasside ja neist genereeritavate vormide vaheliste seoste toe puudumine ehk saab luua ainult ühe lehekülje vorme, mis ei viita teistele.

3.2 Süsteemianalüüs

Järgnev alapeatükk kirjeldab tavapärast Süsteemianalüüsi õppeainet ning selle nõudmisi ja uurib kuidas seda oleks võimalik muuta, et sinna mudelipõhise arenduse põhimõtted sisse viia.

3.2.1 Eesmärgid ja nõuded

Süsteemianalüüsi mõiste tähendab tervikliku (info)süsteemi komponentideks jaotamist eesmärgiga uurida kui hästi need osad omavahel sobituvad saavutamaks neile seatud eesmärki. Seda probleemi lahendamise tehnikat õpetab TTÜ-s ka samanimeline õppeaine, kus tudengite loodava projekti eesmärk on uurida ühte kindlat infosüsteemi ning teostada, kasutades iteratiivset arendusprotsessi metoodikat, selle kohta kolme iteratsiooniline analüüs. Tulemuseks on tavaliselt UML standardipõhised mudelid, tekstipõhine dokumentatsioon ning kasutajaliideste pildid. Mudelite koostamiseks kasutatakse enamasti Enterprise Architect tarkvara. Sarnase ülesehituse ja eesmärkidega, kuid väiksemas mahus, on ka eelneva eelduseks olev modelleerimise aine, milles tutvustatakse üldisemalt ja vähemdetailselt mudelite koostamise põhimõtteid.

Süsteemianalüüsi projekti tehakse üksi või kahekesi ning nõutud on koostada minimaalselt 8-12, aga soovitavalt vähemalt 16-24 erinevat kontsepti. Sinna alla kuuluvad kasutusjuhtude, tegevus-, kontekst-, oleku-, jada- ning erinevad klassidiagrammid. Kõige detailsem klassidiagramm mis luuakse on täpsustatud kontseptuaalne mudel. [15]

Analüüsitöö kvaliteedi tõstmiseks oleks kasu kui loodavast kontseptuaalsest klassidiagrammist ehk valdkonnamudelist saaks genereerida kasutajaliidesed, mis asendaks ka hetkel kasutatava meetodi käigus tehtavaid kasutajaliideste pilte.

3.3 Uuritav süsteem

Antud bakalaureusetöö raames uuritakse osaliselt tavapärase süsteemianalüüsi õppeaine protsessi ning võrreldakse praeguseid ning oodatavaid uusi lõpptulemusi. Uurimuse aluseks on Tallinna Tehnikaülikooli õppeinfosüsteem. Selle jaoks projekteeritakse sarnaselt õppeaine põhimõtetele vajalik dokumentatsioon ning võrreldakse hetkel kasutatava viisi abil saadavat kontseptuaalse klassidiagrammi ja kasutajaliideste piltide tulemit uue mudelipõhise arenduse käigus saadava klassidiagrammi ja reaalsete kasutajaliidestega. Projekt luuakse 2015. aasta sügisel süsteemianalüüsi õppeaines kasutatud näidisprojekti ning 2011. aastal „Kontseptuaalse süsteemianalüüsi“ õppeaines loodud „e-deklaratsioonide haldamise“ näidete põhjal. Keskendutakse ainult klassi- ja kasutusjuhtude diagrammidele. Tegevus-, kontekst-, oleku- ega jadadiagramme ning sellega seonduvat dokumentatsiooni ei looda.

3.3.1 Skoop ja kirjeldus

Missioon ja eesmärgid:

„Teenuse eesmärgiks on lihtsustada tudengitele ainete deklareerimist, anda õppejõududele ülevaade aine deklareerinud tudengitest ja võimaldada õppejõul tudengi ainedeklaratsiooni mitte aktsepteerida, vähendada dekanaadi töökoormust.„ [16]

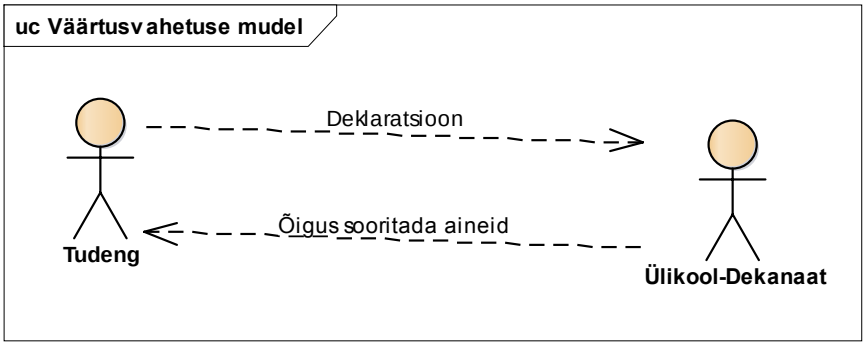
Kasutatavad tööriistad:

Projektis kasutatakse Enterprise Architect CASE vahendit, et jäljendada praegust õppeaine protsessi ning selles loodavaid mudeleid ning Eclipse IDE-t ning EMF ja EMF Forms raamistikku, et luua alternatiivsed mudelid.

Ärisüsteemi määratlus ja lühikirjeldus:

Ärisüsteemina vaatleme tudengi ainete deklareerimise alamvaldkonda ülikooli õppetöö valdkonnas.

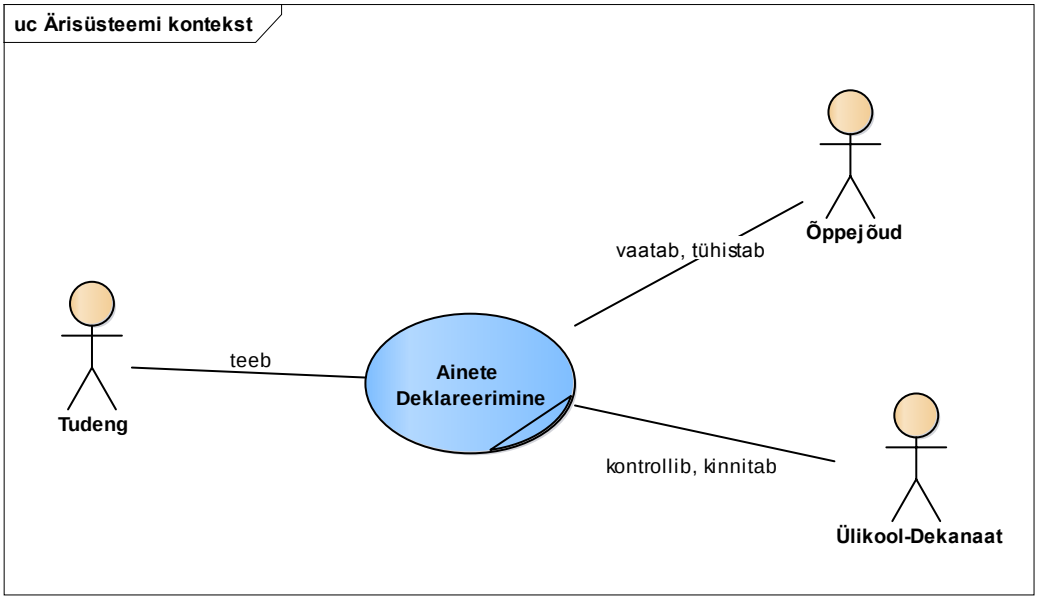
Ainete deklareerimise põhieesmärgiks on õiguse andmine tudengile (õppida ning sooritada konkreetsel semestril konkreetseid aineid).



Joonis 8. Väärtusvahetuse mudel

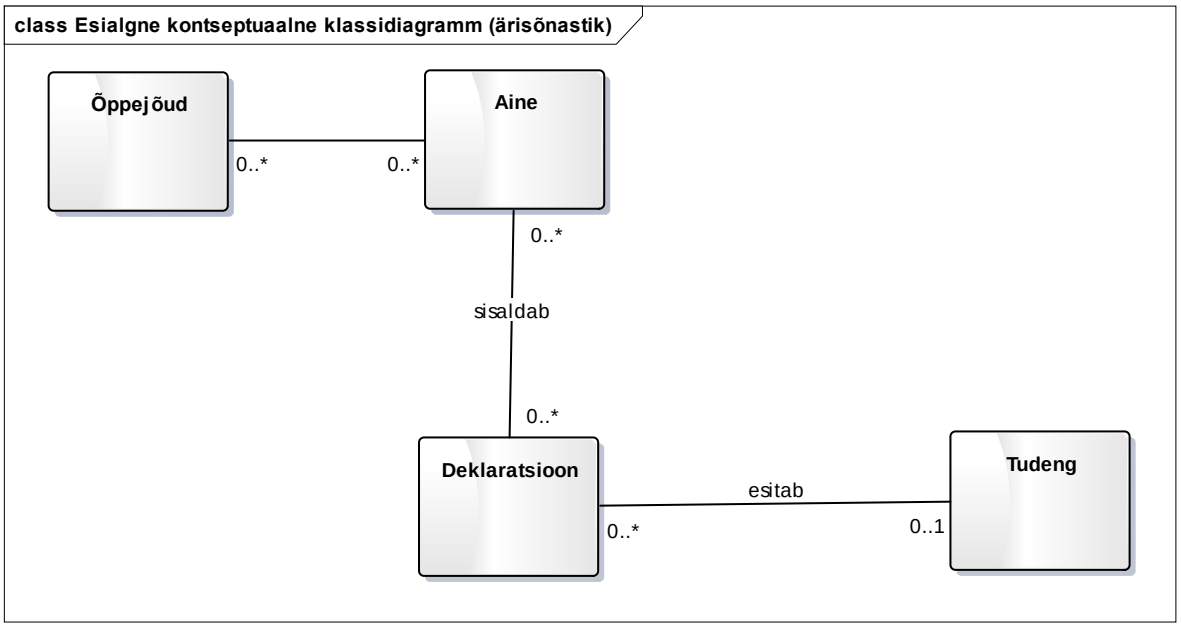
Ainete deklareerimise osapoolteks (põhitegelasteks) on Tudeng, Dekanaat (esindab Ülikooli, täpsemalt selle konkreetset teaduskonda) ning deklareeritavaid aineid õpetavad Õppejõud. Tudeng koostab ning esitab deklaratsiooni, Dekanaat kontrollib ning kinnitab deklaratsioonid, Õppejõud saab vaadata oma ainetele deklareerimisi ning võib tühistada oma aine kuulumisi konkreetsete tudengite deklaratsioonidesse.

Eesmärkmudel ja ärikasutusjuhud, äriprotsesside struktuur



Joonis 9. Ärikasutusjuhtude kontekstdiagramm.

Järgnevalt esitatakse esialgne kontseptuaalne klassidiagramm, kus on toodud põhilised äriolemid ja nende vahelised seosed. Täispikas projektis jaotatakse see esialgseteks registriteks.

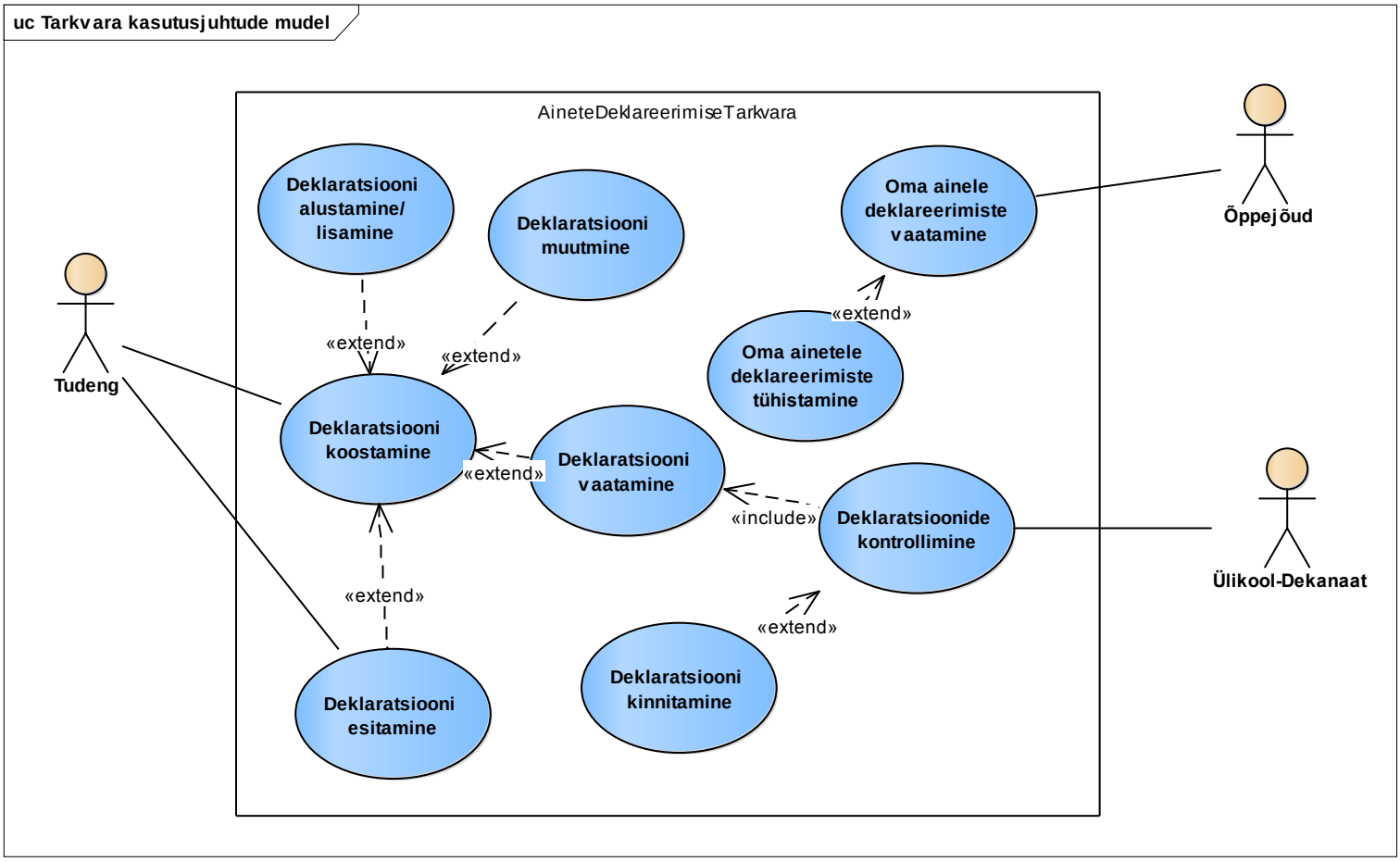


Joonis 10. Esialgne kontseptuaalne klassidiagramm

Tarkvara määratlus ja lühikirjeldus:

Näites analüüsitavaks tarkvarasüsteemiks on õppeinfosüsteemi osa, mis võimaldab tudengil deklaratsiooni koostada ning esitada, dekanaadi töötajal deklaratsioone kontrollida ja (mitte)kinnitada, õppejõul oma ainetele deklareerimisi vaadata ja (mitte)tühistada.

Järgnevalt on toodud primaarsete kasutusjuhtude diagramm



Joonis 11. Deklaratsioonide haldamise kasutusjuhtude diagramm

Käesolevas töös keskendutakse otseselt tudengi poolt selle tarkvara abil läbi viidavatele tegevustele – deklaratsiooni koostamisele ja esitamisele

Diagrammil olevate kasutusjuhtude lühikirjeldused on järgmised:

- **Deklaratsiooni alustamine/lisamine**

Tudeng soovib semestri alguses deklaratsiooni esitamise perioodil lisada enda õppimiste jaoks uues deklaratsiooni. Tudeng tuvastab ennast ja siseneb õppeinfosüsteemi. Tudeng loob uue deklaratsiooni. Tudengil on võimalik enda deklaratsiooni koostada.

- **Deklaratsiooni koostamine**

Tudeng on lisanud uue deklaratsiooni. Tudeng saab otsida enda deklaratsiooni jaoks vajalikke ja soovitud aineid. Süsteem tagastab otsitava aine. Tudeng valib aine ja määrab sellele õppejõu. Süsteem salvestab selle tema deklaratsiooni. Deklaratsiooni saab jätta ka tühjaks. Deklaratsiooni saab salvestada ilma esitamata ning seda on võimalik muuta. Tudengil on koostatud deklaratsioon.

- **Deklaratsiooni muutmine**

Tudeng on salvestanud kuid mitte veel esitanud deklaratsiooni ja soovib seda muuta. Tudeng avab deklaratsiooni. Süsteem tagastab tudengi salvestatud deklaratsiooni. Tudeng saab lisada või kustutada õppeaineid ja muuta õppeainete õppejõudusid. Tudeng salvestab muudetud deklaratsiooni. Süsteem salvestab muudetud deklaratsiooni.

- **Deklaratsiooni esitamine**

Tudeng on koostanud deklaratsiooni ja lisanud sinna õppeained või jätnud selle tühjaks. Tudeng salvestab ja esitab deklaratsiooni. Süsteem salvestab tudengi deklaratsiooni ja selle olekuks saab „esitatud“. Deklaratsiooni kehtivusajaks määratakse süsteemi poolt deklareerimise semestri pikkus.

- **Deklaratsiooni vaatamine**

Tudeng on ainult salvestanud või salvestanud ja esitanud enda deklaratsiooni. Tudeng soovib deklaratsiooni vaadata. Tudeng avab deklaratsiooni. Süsteem tagastab tudengi koostatud deklaratsiooni. Deklaratsioonis kuvatakse õppeained ja nendega seotud vajalikud andmed. Ülikool-Dekanaat näeb tudengi deklaratsiooni ja selle andmeid.

- **Deklaratsiooni kontrollimine**

Ülikool-Dekanaat saab vaadata tudengite esitatud deklaratsioone. Ülikool-Dekanaat saab pärast esitamist ja teatud ajal pärast deklaratsiooni esitamise tähtaega vajadusel ise esitada tudengi deklaratsiooni tema nõusolekul või teha sinna tudengi soovil muudatusi. Süsteem salvestab tehtud muudatused.

- **Deklaratsiooni kinnitamine**

Tudeng on esitanud deklaratsiooni. Süsteem automaatselt kinnitab deklaratsiooni. Kui tudeng on koostanud deklaratsiooni, aga ei esitanud ja deklaratsiooni esitamise tähtaeg läheb mööda, siis Ülikool-Dekanaat saab deklaratsiooni ise manuaalselt kinnitada.

- **Oma ainele deklareerimiste vaatamine**

Tudeng on koostanud deklaratsiooni ning määranud selle aines õppejõu. Õppejõud näeb, et tudeng on registreerinud tema ainesse. Õppejõul on võimalus kinnitada või tagasi lükata tudengi deklaratsioon. Õppejõud võib ka midagi teha, mis juhul loetakse tudengi deklaratsioon aktsepteerituks.

- **Oma ainele deklareerimiste tühistamine**

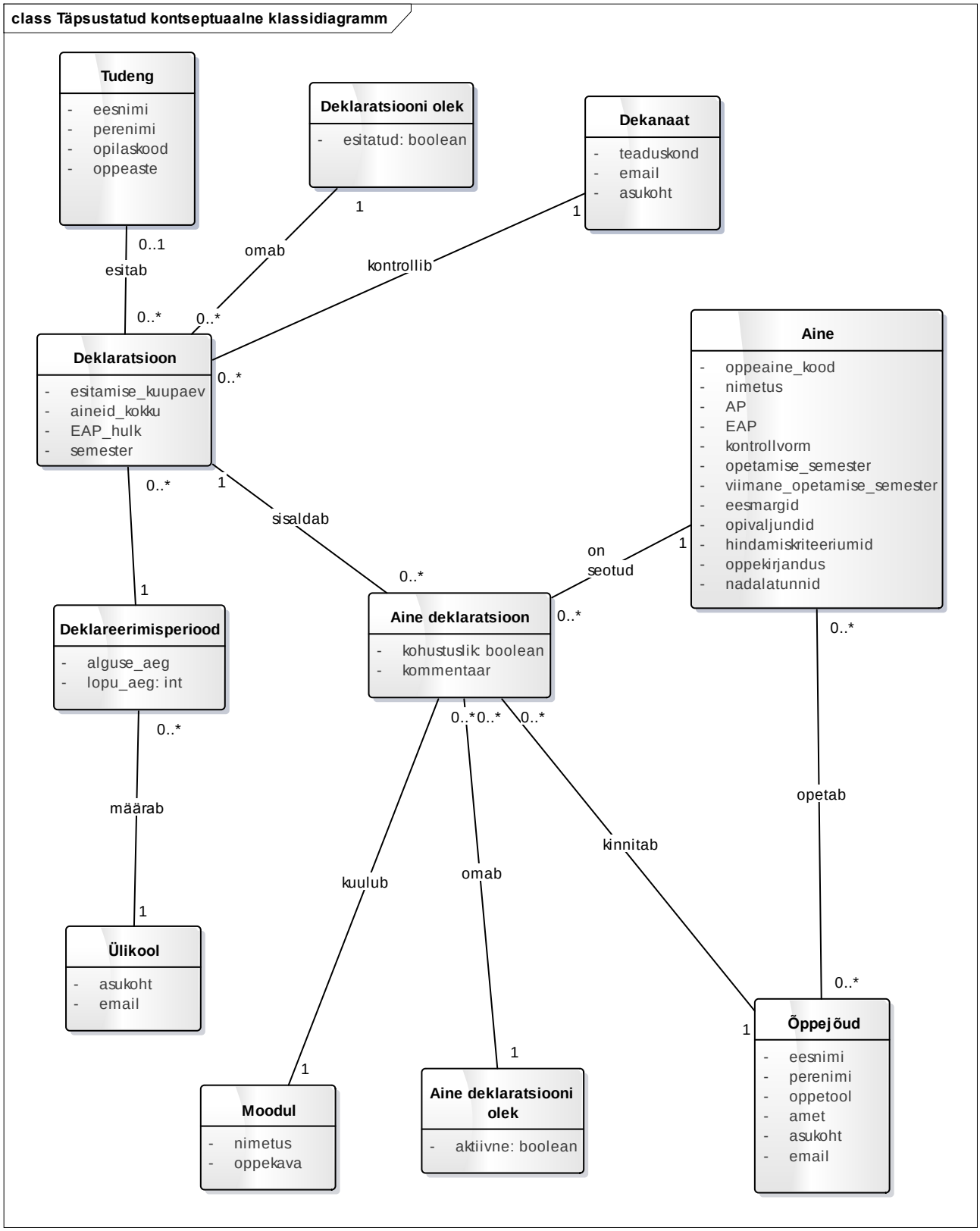
Tudeng on koostanud deklaratsiooni ning määranud selle aines õppejõu. Õppejõud näeb, et tudeng ei ole täitnud eelduseid, ainele on juba liiga palju õpilasi registreerinud või on mõni muu põhjus mis juhul õppejõud soovib tudengi deklaratsiooni tagasi lükata. Õppejõud märgib, et tudengi deklaratsioon ei ole aktsepteeritud. Tudengile kuvatakse deklaratsioonis, et teda ei ole soovitud ainele aktsepteeritud ja teda ei registreerita klassi.

3.3.2 Tavapärane meetod

3.3.2.1 Täpsustatud kontseptuaalne klassidiagramm

Täpsustatud kontseptuaalsed klassidiagrammid esitatakse üldiselt süsteemianalüüsi projektis mitmes detailimisfaasi iteratsioonis. Need iteratsioonid keskenduvad olulisematele tarkvara kasutusjuhtudele, mis peegeldavad kasutusjuhtude prioriteete. Antud töös tehakse projekt ühe iteratsioonina ning seetõttu luuakse ka üks kontseptuaalne klassidiagramm.

Täpsustatud kontseptuaalne klassidiagramm



Joonis 12. Täpsustatud kontseptuaalne klassidiagramm

Mudeli keskseks objektiks on „Deklaratsioon“. Deklaratsioon on seotud konkreetse tudengiga. Deklaratsioonil on olemas deklaratsiooni olek. Deklaratsioonid on seotud dekanaadiga, kes saab kõiki deklaratsioone kontrollida.

Deklaratsioonil on deklareerimiseperiood, mille jooksul uut deklaratsiooni on võimalik esitada. Deklareerimisperiood määratakse ülikooli poolt. Ülikool võib määrata mitu deklareerimise perioodi.

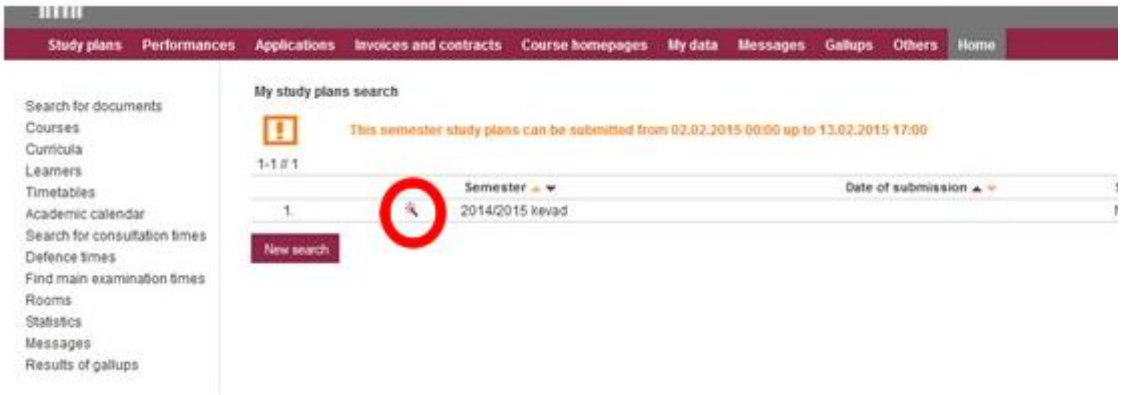
Deklaratsioon on seotud mitme ainega ja ainel võib olla mitu deklaratsiooni. Ainel on õppejõud, kes seda õpetavad. Ühel õppejõul võib olla mitu õpetatavat ainet.

Deklaratsioon on seotud konkreetse aine deklaratsiooniga, mis on seotud ühe kindla ainega. Õppejõud saab vaadata ainult aine deklaratsiooni, mille kaudu ta saab temale deklareeritud aineid kinnitada ja kommentaare lisada. Aine deklaratsioon on seotud mooduliga, mis määrab kuhu valitud aine õppekavas kuulub. Aine deklaratsioonil on olek.

3.3.2.2 Kasutajaliideste eskiisid

Õppeinfosüsteemi deklaratsiooni loomise ja muutmise kasutajaliideste eskiisid on järgmised:

Uue deklaratsiooni loomise alustamise vaade



Joonis 13. Deklaratsiooni alustamise vaade

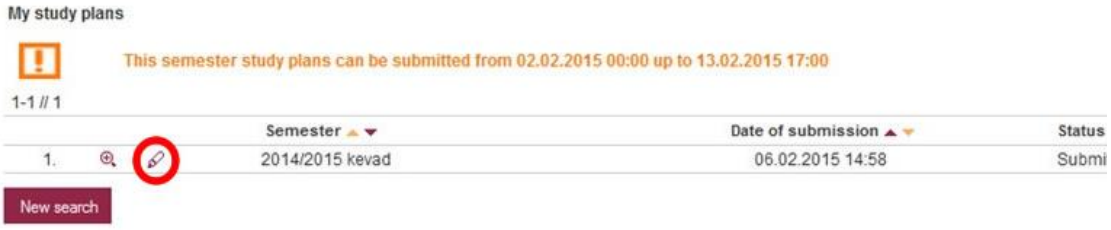
Allikas: TTÜ ÕISi juhend üliõpilastele

Deklaratsiooni esitamise vaade

Joonis 14. Deklaratsiooni esitamise vaade

Allikas: TTÜ ÕISi juhend üliõpilastele

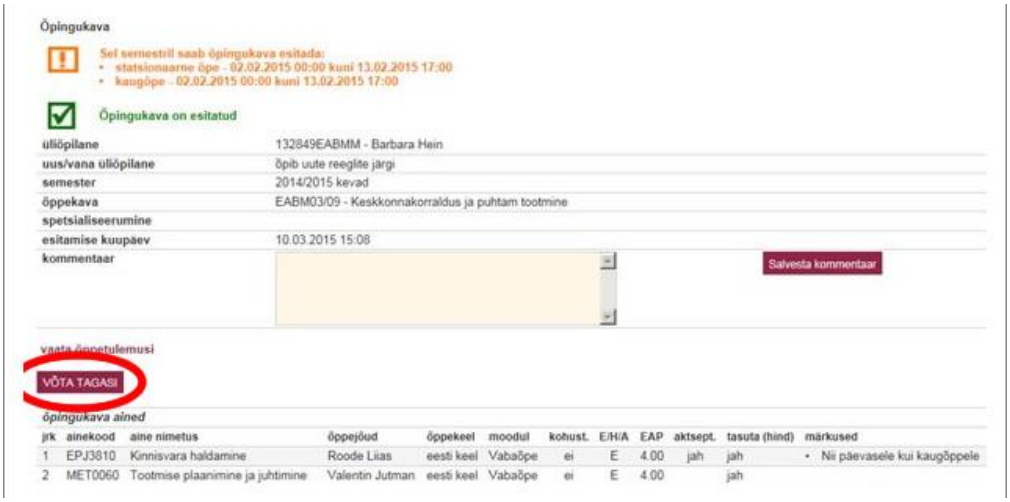
Deklaratsiooni muutmise alustamise vaade



Joonis 15. Deklaratsiooni muutmise vaade

Allikas: TTÜ ÕISi juhend üliõpilastele

Deklaratsiooni tagasi võtmise vaade:



Joonis 16. Deklaratsiooni tagasi võtmise vaade

Allikas: TTÜ ÕISi juhend üliõpilastele

Kasutajaliides on vaade domeenimudelisse, s.t. enamus kasutajaliidese andmeelementidest (väljadest) suhestatakse otseselt kontseptuaalse klassidiagrammi vastavate elementidega (atribuutidega). Selline vastavus on jälgitav ka Õppeinfosüsteemi kasutajaliideste puhul. Kasutajaliideste eesmärk süsteemianalüüsi kontekstis on lihtsustada süsteemist arusaamist ning see võimaldab täpsustada nõudeid

3.3.3 Uus meetod

UML standard eristab mudeli ja diagrammi mõisteid (diagramm on vaade mudelisse; ühte mudelit esitatakse reeglina paljude diagrammidega (nii palju kui vaja)). EMF Ecore mudelis eeldatakse aga ka täpselt ühte Ecore diagrammi. Siin tekib justkui põhimõtteline vastuolu vana ja uue meetodi vahel. Süsteemianalüüsi aines on seni tehtud üle erineva iteratsioonide mitmeid kontseptuaalseid klassidiagramme, EMF Forms tehnoloogia kasutamine toob kaasa ühe (meta)mudeli kohta ühe diagrammi piirangu. Seepärast on oluline tulevikus uurida võimalusi jagada MDA-põhiseid projekte ja selles loodavaid (meta)mudeleid mitmeks põhuobjektile vastavaks osaks jagades need nii Eclipse modelleerimise väiksemateks projektideks.

4. Realisatsiooni teostamine

Järgnevas peatükis kirjeldatakse uue meetodi rakendamist süsteemianalüüsi projektides. Selle jaoks vaadatakse uue tehnoloogilise lahenduse detailsemaid erinevusi tavapärasest ning luuakse tuginedes sellele EMF-põhine uus valdkonnamudel. Antud mudelist genereeritakse kasutajaliidesed, mida modifitseeritakse, et täita vajalikud funktsioonid. Lisaks, uuritakse võimalust kasutada praeguse süsteemi andmete kasutamise võimalusi uues süsteemis. Hinnatakse kas uus meetod suudab täita vajalikke nõudmisi ning kirjeldatakse funktsioone mida ei suudetud täita.

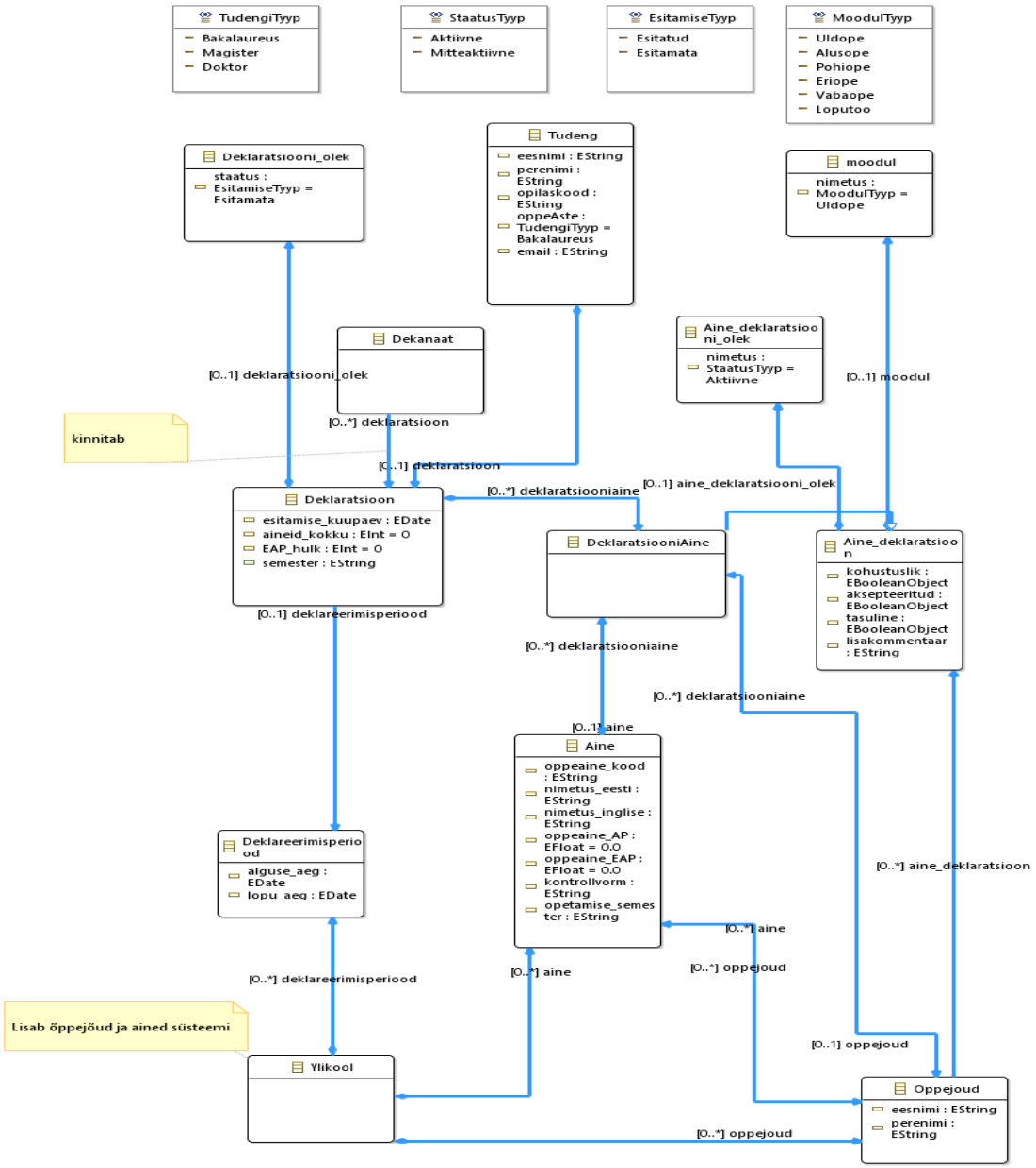
4.1 Mudeli koostamine

Uurimisaluseks on sama infosüsteem mis eelnevas peatükis kirjeldatud süsteemianalüüsi projektis ehk õppeinfosüsteem. Eesmärk on uurida võimalusi kuidas, kasutades MDA põhimõtteid, oleks võimalik luua uus meetod süsteemianalüüsi projektide koostamiseks. Nõudmised ja tehtud analüüs süsteemile püsivad ning selle põhjal koostatakse üks metamudel ehk Ecore diagramm.

Uus mudel koostatakse Ecore diagrammina, EclipseUML tööriista kasutades. Loodav mudel on UML klassidiagrammi alamklass ehk UML põhise klassidiagrammi koostamise oskuseid on võimalik rakendada ka mudelipõhise arenduse metamudeli koostamiseks.

4.2 Mudel

Järgmiseks esitatakse EclipseUML abil koostatud metamudel.



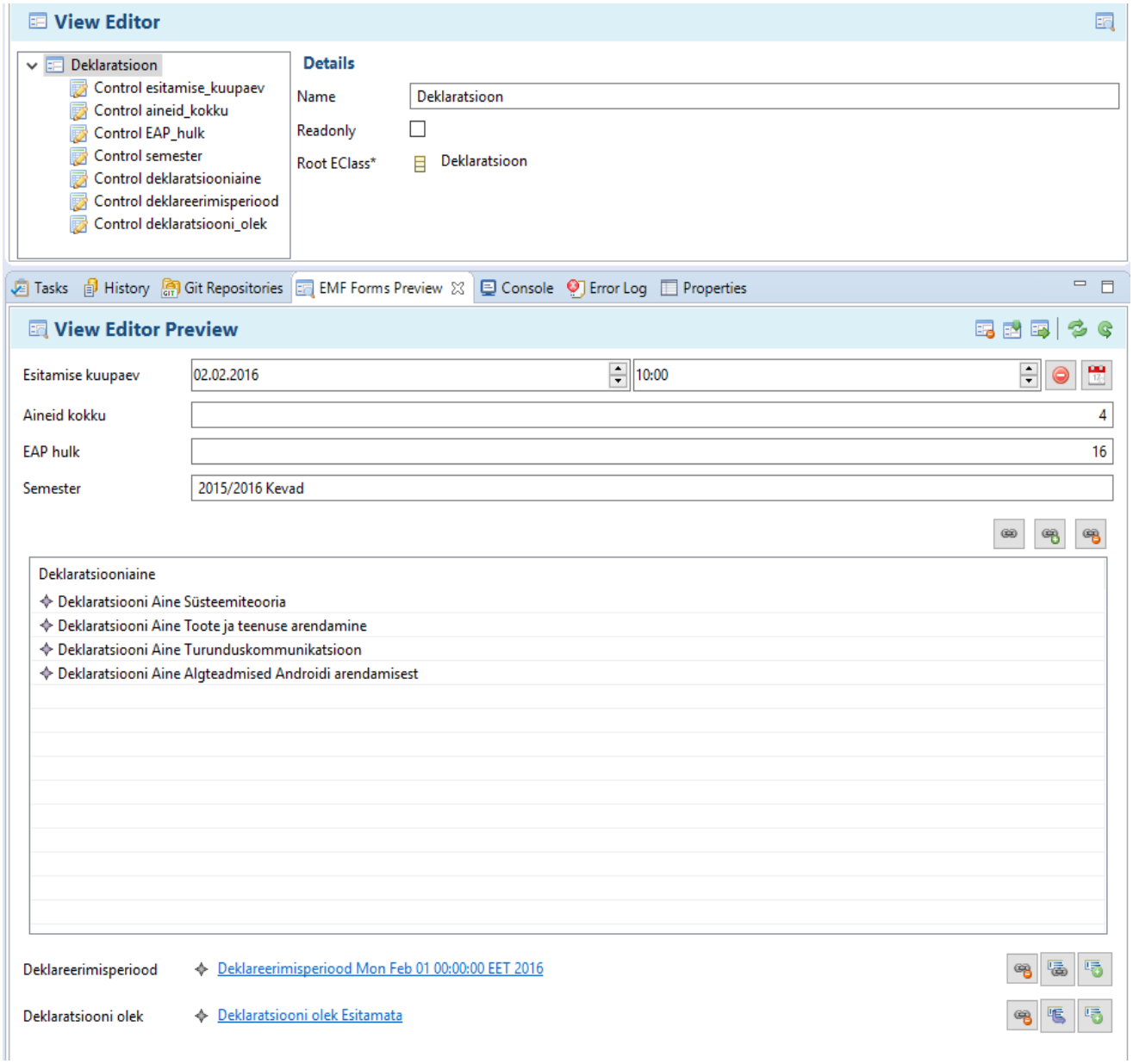
Joonis 17. Ecore valdkonnamudel

Diagrammi selgitus on sarnane eelnevate UML kontseptuaalsete mudelite selgitustele ning objektid ja seosed püsivad. Kuna õppeinfosüsteemi kasutamisel on mõnel atribuudil alati kindlad väärtused mida need saavad omandada, siis kasutati selle mudeli loomisel listide lisamise võimalust, mis on samuti toetatud ka UML standardis. Listi olemasolul ning atribuudiga seostamisel, saab seotud välja väärtuseks valida ainult listis täpsustatud literaale. Ecore mudel võimaldab lisada klassidele ka operatsioone, mis genereerivad vajalikud Java klassid, milles on võimalik kirjeldada soovitud funktsioone.

4.3 Mudelist kasutajaliidese genereerimine

Mudelile rakendatakse EMF Forms generaatorit, mille põhjal luuakse uude projekti soovitud klasside vaatemudelid. Vajadusel muudetakse, lisatakse või eemaldatakse loodud vaatemudelite välju.

Deklaratsiooni vaatemudel ja kasutajaliidese eelvaade.



Joonis 18. Deklaratsiooni vaatemudel ja kasutajaliidese eelvaade.

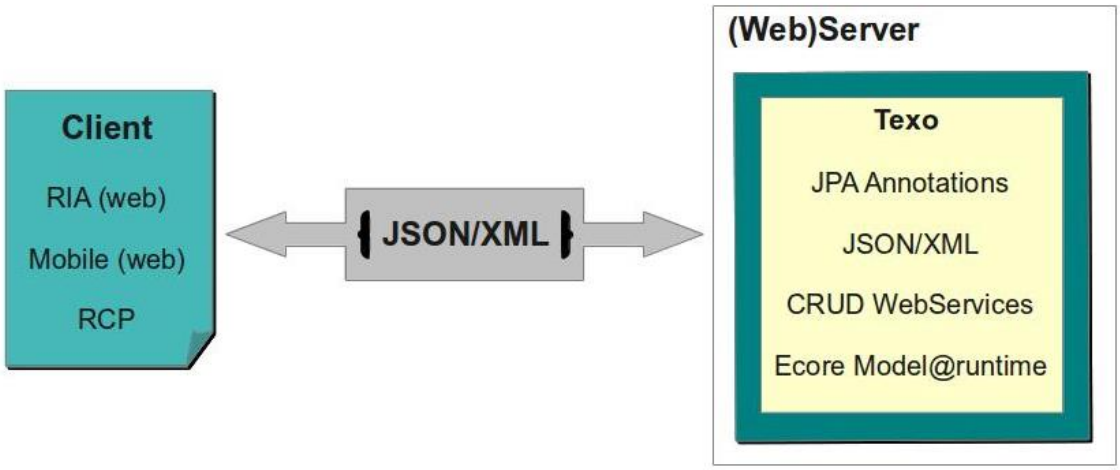
Joonis 18. on esitatud deklaratsiooni vaatemudel ja kasutajaliidese eelvaade, mis on õpilase deklaratsiooni esitamise üheks keskseks osaks. Eesmärgiks oleks saavutada olukord, kus süsteem salvestab automaatselt esitamise kuupäeva ning arvutab ained kokku ja EAP hulga ning esitab need vormil. Semester valitakse valdkonnamudelis kirjeldatud loetelust. Deklareerimisperiood määratakse ülikooli poolt ning see seotakse deklaratsiooniga. Deklaratsiooni olek on loomisel „esitamata“ ja pärast esitamist muutub „esitatud“-ks. Deklaratsiooniained salvestatakse süsteemi ülikooli poolt ning neid on võimalik valida läbi deklaratsiooni vormi. Deklaratsiooniaine kaudu saadavad andmed on

mida tuleks kasutada ainete ja EAP hulga arvutamiseks. Deklaratsiooniaine, Deklareerimisperiood ja Deklaratsiooni olek saavad andmed teistelt vormidelt ning esitavad need deklaratsiooni vormil.

Vaatemudeleid on võimalik luua iga Ecore mudelis kirjeldatud klassi kohta.

4.4 Integratsioon olemasoleva süsteemiga

Loodud Ecore mudelit on teoreetiliselt võimalik siduda olemasolevate süsteemidega ehk on võimalik küsida olemasolevaid andmeid läbi veebiteenuste. Selle jaoks on loodud erinevaid laiendusi, näiteks EMF-REST ja Eclipse Texo, mis võimaldavad REST teenuste abil JSON või XML formaadis edastada ja vastu võtta soovitud andmeid.



Joonis 19. Texo kliendi ja serveri vaheline suhtlus

Allikas: http://wiki.eclipse.org/Texo

4.5 Lõppkasutaja kaasamine

Kuna näiteprojektis kirjeldatud õppeinfosüsteemi abil ainete deklareerimise protsess on kõikidel teaduskondadel ühesugune ning süsteem peaks käituma igakord sarnaselt, siis ei leitud sobilikku võimalust kuidas lõppkasutajapoolse arendamise põhimõtteid oleks antud näite puhul võimalik mudelipõhise arendusega täielikult siduda.

Peamine viis kuidas lõppkasutaja saaks sellises projektis osaleda oleks kasutades oma teadmisi valdkonnast aidates arendajal kirjeldada valdkonnamudelit ehk antud juhul Eclipse Ecore mudelit. Praegusel juhul oleks lõppkasutaja rollis ülikooli dekanaat, kellel on detailne ülevaade tudengite rollidest ja õppeinfosüsteemi toimimisest ning kes peaks

suutma tagada, et loodav süsteem oleks lõpuks parem kasutada kui see mille metamudeli loomisel osalesid ainult arendajad. EMF Forms sissetoomine analüüsitöösse kindlasti suurendaks dekanaadi töötaja võimaluse valdkonnamudeli koostamisel vahetult ja aktiivselt osaleda.

Teise võimalusena oleks lõppkasutaja rollis tudeng kes õpib ülikoolis ning soovib deklaratsiooni esitada. Deklaratsiooni esitamisel ta valib ained ja õppejõud ning seeläbi kirjeldab kuidas süsteem tema jaoks käitub ehk millised õppejõud saavad talle hindeid määrata, millistest ainetest ta ainepunkte saab, kuidas ta keskmine hinne selle põhjal kujuneb jne. Ehk tudeng kasutab üldist malli, milleks on deklaratsioon, teeb selles valikud, ehk väärtustab mudeli klasside väljad, ning selle põhjal saab luua enda jaoks unikaalset ja tema andmetega seotud keskkonda.

4.6 Alternatiivsed tehnoloogiad

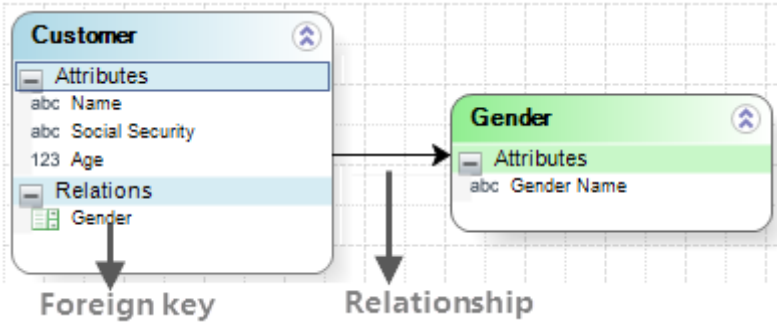
Eclipse EMF ja EMF Forms raamistikud on üks võimalus kuidas valdkonda kirjeldada ja kasutajaliideseid genereerida. Peamiseid põhjuseid miks need tehnoloogiad töö tegemiseks valiti kirjeldati juba varasemalt, kuid lisaks sellele on võimalik sarnaseid tulemusi saavutada ka alternatiivsete meetodite ja tööriistade abil.

Üheks selliseks viisiks oleks äriprotsesside modelleerimise notatsiooni ehk BPMN põhist Bizagi tarkvara kasutada. Kui antud töös kirjeldatud ja uuritud Eclipse tehnoloogiad olid objektipõhised, siis Bizagi tehnoloogia oleks protsessipõhine.

Bizagi BPM Suite on Bizagi poolt loodud tasuline tarkvara lahenduste pakett, mis ühendab endas nende poolt loodud kolme erinevat rakendust: vabavaraline Modeler ning tasulised Studio ja Engine. Selle eesmärgiks on toetada kogu äriprotsesside elutsüklit. BPM Suite on kihilise ülesehitusega, mis võimaldab kergemini olemasolevaid andmeid Bizagiga siduda. See koosneb kolmest kihist: äripoolele kuuluv protsesside kiht ja IT poolt hallatavad andme- ja ettevõtte kaardistamise kihid. Läbi viimase toimub suhtlus erinevate Bizagi Suite-st väljaspool olevate andmebaaside ja pärandsüsteemidega jms. (vt. Lisa 2).

Bizagi näeb Suite kasutamisel soovituslikult ette, et esmalt koostatakse vabavaralises Bizagi Modeler tarkvaras äriprotsesside mudel, mis kirjeldab ühe kindla äriprotsessi töövoolu.

Järgnevalt tuleb määrata atribuudid mudeli reaalsetele või abstraktsetele kirjeldatavatele tegevustele või tegutsejatele.



Joonis 20. Bizagi andmemudel näide

Allikas: Bizagi BPM Suite V10.7 - User guide

Andmemudel kirjeldab kuidas andmetele ligipääsetakse ning kuidas neid salvestatakse süsteemis. See mudel on pigem andmebaasi mudeli kui klassidiagrammi sarnane - loomisel on võimalik kasutada nelja erinevat tüüpi klassi ja nelja erinevat nendevahelist seost, samuti on võimalik mudeli seoste puhul kasutada võtmeid(surrogaat-, välis- ja primaarvõti), mis määravad kuidas need klassid omavahel seotud on. Tänu kihilisele ülesehitusele on läbi andmemudeli võimalik väliste süsteemidega ühenduda. Selle jaoks on Bizagil replikatsiooni ja virtualisatsiooni tehnoloogiad.

Peamine erinevus Ecore valdkonna mudeli ja Bizagi andmemudeli vahel on see, et esimese eesmärk on kirjeldada kogu valdkonna (n. Õppetöö) objekte ja informatsiooni mõistelisel (kontseptuaalsel) tasemel, aga viimase eesmärgiks on kirjeldada konkreetse (äri)protsessi tegevuste (s.h. vormide) all olevaid andmeid. Esimene rakendab objektorienteeritud modelleerimist (mistõttu sobib Süsteemianalüüsi ainesse paremini), teine mitte (sobib hästi Äriprotsesside ainesse).

Pärast andmemudeli kirjeldamist on selle abil võimalik genereerida kasutajaliideste vormid. Vormidele pääseb lõppkasutaja ligi läbi veebibrauserite ning seal esitatavad vormid võivad hallata kõiki inimtegevusega seotud toiminguid. Vormide loomisel kasutatakse adekvaatkuva(WYSIWYG) metoodikat ehk vormi loomisel kuvatakse seda samamoodi nagu hiljem lõppkasutajalegi ning toimub sarnaselt EMF Formsile, kus on võimalik määrata milliseid välju ja nuppe kuvatakse. Samuti on võimalik määrata erinevaid valideerimise ning tegevuste meetodeid vormile.

Lisaks eelnevatele tegevustele ning pärast vormide genereerimist võimaldab Bizagi Suite veel kirjeldada ärireeglid, teavitused, dokumentide mallid, rakenduse integratsiooni, tööjaotuse, turvalisuse definitsioonid jms. Lõpuks, Bizagi Engine võimaldab kõik loodud ja automatiseeritud protsessid käivitada iga kasutaja arvutites või mobiilides. [16]

4.7 Realisatsiooni järeldused.

Realisatsiooni teostamiseks loodi eelnevas peatükis kirjeldatud õppeinfosüsteemi analüüsi põhjal uus valdkonna metamudel, mille eesmärgiks oli toetada mudelipõhist arendust ning võimaldada sellest kiiresti kasutajaliideseid genereerida ning tarkvara prototüüpe luua.

EclipseUML tööriista abil EMF mudeli koostamisel oli võimalik üle kanda ning kasutada eelnevaid teadmisi UML standardist ning õppimiskõver oli seeläbi väiksem kui see oleks olnud ilma eelnevate modelleerimise teadmisteta. Loodud mudel sarnaneb UML-põhisele kontseptuaalsele klassidiagrammile. Antud töös ei loodud projekti iteratsioonidena, seetõttu ei uuritud kas ja kuidas on võimalik luua mitut metamudelit ja neid ühendada süsteemi kirjeldamiseks, mis on oluline osa süsteemianalüüsi projektides.

Iga Ecore klassi jaoks oli võimalik kiiresti genereerida kasutajaliidesed. Siiski, ei olnud ilma Java koodita võimalik täita kogu soovitud funktsionaalsust ning ei jõutud sellise prototüübini, mis suudaks oma funktsionaalsuselt jäljendada kasutusel olevat õppeinfosüsteemi. Mõned näited, mis funktsionaalsuseni sooviti jõuda, kuid mida ei saavutatud: õppeainete ja ainepunktide loendur tudengi deklaratsioonis, deklaratsiooni esitamine ja kuupäeva automaatne salvestamine.

Realisatsiooni teostamise käigus uuriti võimalusi kuidas praeguse süsteemi andmeid oleks võimalik loodud süsteemiga siduda ning leiti, et selle jaoks on loodud erinevaid veebiteenuste tehnoloogiaid, mis lasevad REST teenuste abil JSON/XML formaadis sõnumeid vahetada ning kaks potentsiaalset lahendust, mida autori arvates võiks edasi uurida on Eclipse Texo ja EMF-Rest.

Probleemseks kohaks jäi JSON Forms alles arendusjärgus olev klasside vaheliste viidete ($ref) toe puudumine, mistõttu ei olnud genereeritud kasutajaliideseid võimalik lihtsasti veebi lisada.

Uuriti ka võimalust kas sellise süsteemianalüüsi projekti puhul oleks võimalik kaasata lõppkasutajat arendusse, kasutades *end-user development* põhimõtteid. Parim viis kuidas lõppkasutaja oleks saanud arendusse panustada, oleks olnud aidates arendajal koostada Ecore metamudelit. Teisest küljest sai lõppkasutaja kirjeldada süsteemi enda jaoks valides erinevaid deklaratsiooni aineid, mis sisuliselt tähendab õppekeskkonna individualiseerimist konkreetse tudengi jaoks.

EMF Ecore mudeli ja UML kontseptuaalse klassidiagrammi koostamise vahel ajakulu muutust ei täheldatud kuna mõlemad kasutasid samal hulgal elemente ja seoseid. EMF Forms see-eest säästab kasutajaliideste genereerimisel tunduvalt palju aega, mis oleks läinud kasutajaliideste prototüüpide loomisele UML-põhises tööriistas. Lisaks, võimaldab EMF Forms kergesti kõik kasutajaliidesed uuesti genereerida juhul kui on vajalik metamudelisse muudatusi teha, samas kui joonistatud prototüübi puhul võtaks muudatuste tegemine rohkem aega.

Viimaseks järelduseks mida peab uue süsteemianalüüsi projektide loomise meetodi koostamise puhul arvestama on see, et Eclipse IDE ei ole suunatud modelleerimisele nagu selle jaoks tihedamini kasutatavad vahendid, näiteks Enterprise Architect või Rational Rose, mistõttu on vaja Eclipse jaoks installeerida UML võimekuse tagavad lisad, et oleks võimalik luua kõiki mudeleid mida süsteemianalüüsi projektides võib vaja minna. Neid lisasid on samuti vaja testida kooliprojektide loomise kontekstis, et olla kindel, et need tagavad samasuguse võimekuse nagu seni kasutatavad vahendid.

4.8 Edasiarendamise võimalused

Bakalaureuse töös käsitleti mudelipõhist arendust ja lõppkasutaja kaasamist sellesse ning üritati teooria põhjal luua realisatsiooni kasutades Eclipse raamistikke. Realisatsiooni järeldusi kirjeldati eelnevas alapeatükis, kuid selles peatükis üritatakse teha järeldusi ka kogu eelneva temaatika kohta ning pakkuda välja võimalikke edasiarendusi tulevikuks, näiteks potentsiaalseks magistritööks.

Peamiseks puudujäägiks ja teemaks milleni antud töös ei jõutud piisavalt oli EUD põhimõtete kasutamine lõppkasutaja kaasamiseks arendusse kuna valitud projekti teema ei toetanud seda piisavalt. Samas on sellekohane teooria aktuaalne ning populaarsust

kogumas ja autori hinnangul oluline üritada leida võimalusi kaasata seda tarkvaraprojektide analüüsimiseks ja arendamiseks.

Ühe võimalusena kuidas tööd saaks edasi arendada oleks laiendada või üldistada projekti teemat ja õppeinfo aine deklareerimise asemel sooritada töö näiteks õppekava läbimise või õppetegevuse valdkonna kohta üldisemalt. Töös loodavat mudelipõhise arenduse näidet oli raske EUD konteksti panna kuna tudeng ei vaja personaliseeritud tarkvara deklareerimise jaoks, kuigi IT tudengid oskaksid mudelipõhiselt luua enda individualiseeritud EMF vormid ja rakendused. Samas, nagu peatükis 4.6 välja toodi, võiks tudengipoolset deklareerimist lugeda teatud kontekstis EUD kasutamiseks, kui sisestatud andmete põhjal toimuks individuaalse õppekeskkonna mudelipõhine genereerimine nii, et see on metadisaini teooriaga kooskõlas. [18] Ainete deklareerimist saaks ka laiendada näiteks personaalse õppekava deklareerimisele.

Kokkuvõte

Bakalaureuse töö „Kasutajaliideste mudelipõhine arendamine EMF Forms abil“ eesmärgiks oli analüüsitöö kvaliteedi tõstmiseks uurida üldiseid mudelipõhise arenduse ja lõppkasutaja arendusse kaasamise teooriaid ning konkreetseid Eclipse raamistikke, mis võimaldaksid nende aluste põhjal realisatsiooni luua. Teiseks eesmärgiks oli viia läbi infosüsteemi analüüs õppeinfosüsteemi osa näitel ning selle põhjal luua realisatsioon kasutades uuritud EMF ja EMF Forms tehnoloogiaid. Prototüübi põhjal hinnati praeguse süsteemianalüüsi ning uuendatud meetodi erinevusi ja uuest viisist tulenevaid loodetavaid positiivseid külgi.

Bakalaureuse töö käigus selgus, et MDA kontseptsioone on võimalik süsteemianalüüsi projektides kasutada, kuid EUD puhul on see raskendatud ning sõltub projekti teema valikust. Leiti, et mudelipõhine vormide genereerimine vähendab tunduvalt projekti eskiiside loomisele kulunud aega ning laseb luua reaalseid prototüüpe, mida saab kergesti muuta. Valdkonnamudeli loomisel ajakulu võitu ei teki kuna selles kasutatakse sama palju elemente kui varasemalt. Järeldati, et mudelipõhise arenduse abil vormidele funktsioonide loomine on raskendatud kuna nõutud on programmeerimisoskuste omandamine või vajalikud toetavad tehnoloogiad võivad olla arendusjärgus ning mitte toetada soovitud lõpptulemusteni jõudmist.

Tööd oleks vaja edasi arendada EUD valdkonnas, luues projekti laiema teema puhul, mis toetaks individualiseeritud lähenemise vajadust. Lisaks, oleks vajalik uurida vormide funktsioonide lahendamist programmeerimise abil ja uurida võimalusi veebiteenuste kasutamiseks ja vormide veebis kasutamiseks. Viimaseks, tuleks uurida Eclipse IDE kasutamise võimalusi teiste vajalike mudelite modelleerimise jaoks ning võrrelda seda seni kasutatavate vahendite võimekusega.

Töö üldine eesmärk saavutati ning oli võimalik teha järeldusi mudelipõhise arenduse eeliste kohta süsteemianalüüsi projektides.

Summary

The purpose of this document „Model-based development of user interfaces with EMF Forms“ was to explore the possibilities of improving the process of creating system analysis projects using model-driven architecture and end-user development methods together with Eclipse frameworks, specifically Eclipse Modeling Framework and EMF Forms. Using those ideas the goal was to carry out an example project using both conventional and new methods and then analyze the differences between those two to find out if there are any positives to the new approach.

Bachelor's work showed that the MDA concepts can be used for the system analysis projects but it is difficult to also use EUD theories which depend on the subject of the project. It was concluded that generating viewmodels from metamodels significantly reduces the time spent on the creation of project sketches and allows to create realistic prototypes, which can be easily changed. There wasn’t any reduce of time when creating the metamodel because it requires just as many elements as the class diagrams used previously. Some problems were also discovered – creating functions for forms requires prior knowledge of programming and some necessary frameworks might still be in development.

This Bachelor’s thesis still needs to be developed further in the EUD field, using a broader subject for the analysis work that supports an individualized approach. Also, it would be necessary to create the required functions for the forms using programming languages and after that to use other frameworks that help the prototype communicate with other systems and use it in web. Lastly, the usability of Eclipse IDE in terms of creating other UML models needs to be analyzed further and its capabilities should be compared to currently used tools.

The main objectives of the work were achieved and it was possible to make conclusions about the advantages of using the MDA approach in a system analysis project.

Kasutatud kirjandus

[1] Object Management Group, „Model Driven Architecture, White Paper,“ 27 November 2000. [Võrgumaterjal]. Available: http://www.geocities.ws/pravin_suman/Resources/00-11-05.pdf. [Kasutatud 03 04 2016].

[2] F. Truyen, „The Fast Guide to Model Driven Architecture, The Basics of Model Driven Architecture,“ January 2006. [Võrgumaterjal]. Available: http://www.omg.org/mda/mda_files/Cephas_MDA_Fast_Guide.pdf. [Kasutatud 19 05 2016].

[3] G. A. Lewis ja L. Wrage, „Model Problems in Technologies for Interoperability: Model-Driven Architecture,“ Carnegie Mellon University, 2005.

[4] C. Calero, K. Framling, O. Greevy, A. Lastovetsky ja C. Rolland, „End User Development: Survey of an Emerging Field for Empowering People,“ *ISRN Software Engineering*, kd. 2013, p. 11, 2013.

[5] E. Stav, J. Floch, M. Ullah Khan ja R. Saetre, „Using Meta-modelling for Construction of an End-User Development Framework,“ %1 *End-User development, 4th International Symposium*, Springer, 2013.

[6] D. Fogli ja A. Piccinno, „Co-evolution of End-User Developers and Systems in Multi-tiered Proxy Design Problems,“ %1 *End-User Development, 4th International Symposium*, Springer, 2013.

[7] „About the Eclipse Foundation,“ [Võrgumaterjal]. Available: https://eclipse.org/org/. [Kasutatud 25 03 2016].

[8] F. Budinsky, D. Steinberg, E. Merks, R. Ellersick ja T. J. Grose, Eclipse Modeling Framework: A Developer's Guide, Addison Wesley, 2003.

[9] D. Steinberg, „/www.eclipse.org,“ 2005. [Võrgumaterjal]. Available: http://www.eclipse.org/modeling/emf/docs/presentations/EclipseCon/EclipseCon2008_309T_Fundamentals_of_EMF.pdf. [Kasutatud 27 03 2016].

[10] ZeroTurnaround OÜ, „Developer Productivity Report,“ ZeroTurnaround OÜ, 2012.

[11] K. Letkeman, „Comparing and merging UML models in IBM Rational Software Architect: Part 1,“ 12 07 2005. [Võrgumaterjal]. Available: http://www.ibm.com/developerworks/rational/library/05/712_comp/. [Kasutatud 25 04 2016].

[12] M. Koegel ja J. Helming, „eclipsesource.com,“ 26 10 2015. [Võrgumaterjal]. Available: http://eclipsesource.com/blogs/tutorials/getting-started-with-EMF-Forms/. [Kasutatud 29 03 2016].

[13] M. Koegel ja J. Helming, „www.eclipsesource.org,“ 19 10 2015. [Võrgumaterjal]. Available: http://eclipsesource.com/blogs/tutorials/getting-started-with-the-emf-client-platform/. [Kasutatud 30 03 2016].

[14] „EMF Forms Integration Guide,“ EclipseSource, 2 November 2015. [Võrgumaterjal]. Available: https://github.com/eclipsesource/jsonforms/wiki/EMF-Forms-Integration-Guide. [Kasutatud 5 04 2016].

[15] M. Roost, „Tähtajad ja nõutud projekti osad-1.docx,“ 3 12 2015. [Võrgumaterjal]. Available: http://maurus.ttu.ee/aine_index.php?aine=337. [Kasutatud 6 04 2016].

[16] H. Jürgenson, „Projekt aines IDU5360 “Kontseptuaalne süsteemianalüüs” - e-deklaratsioonide haldamine,“ 2011. [Võrgumaterjal]. Available: http://kerf.ee/kontsept/Naidisprojekt_K11.doc. [Kasutatud 6 04 2016].

[17] Bizagi, „Welcome to Bizagi BPM Suite V10.5 - User guide,“ Bizagi, [Võrgumaterjal]. Available: http://help.bizagi.com/bpmsuite/en/10.5/index.html?modeling_the_process.htm. [Kasutatud 21 05 2016].

[18] J. Karat ja J. Vanderdonckt, „Meta-design: A framework for the future of end-user development,“ %1 *End User Development, human-computer interaction series, vol. 9*, Springer, 2006.

Lisa 1 MDA mudeli teisendamise muster

Üldine muster mida jälgitake mudelipõhises arenduses mudeli teisendamiseks

Joonis 21. MDA mudeli teisendamine

Allikas: Frank Truyen, *The Fast Guide to Model Driven Architecture*

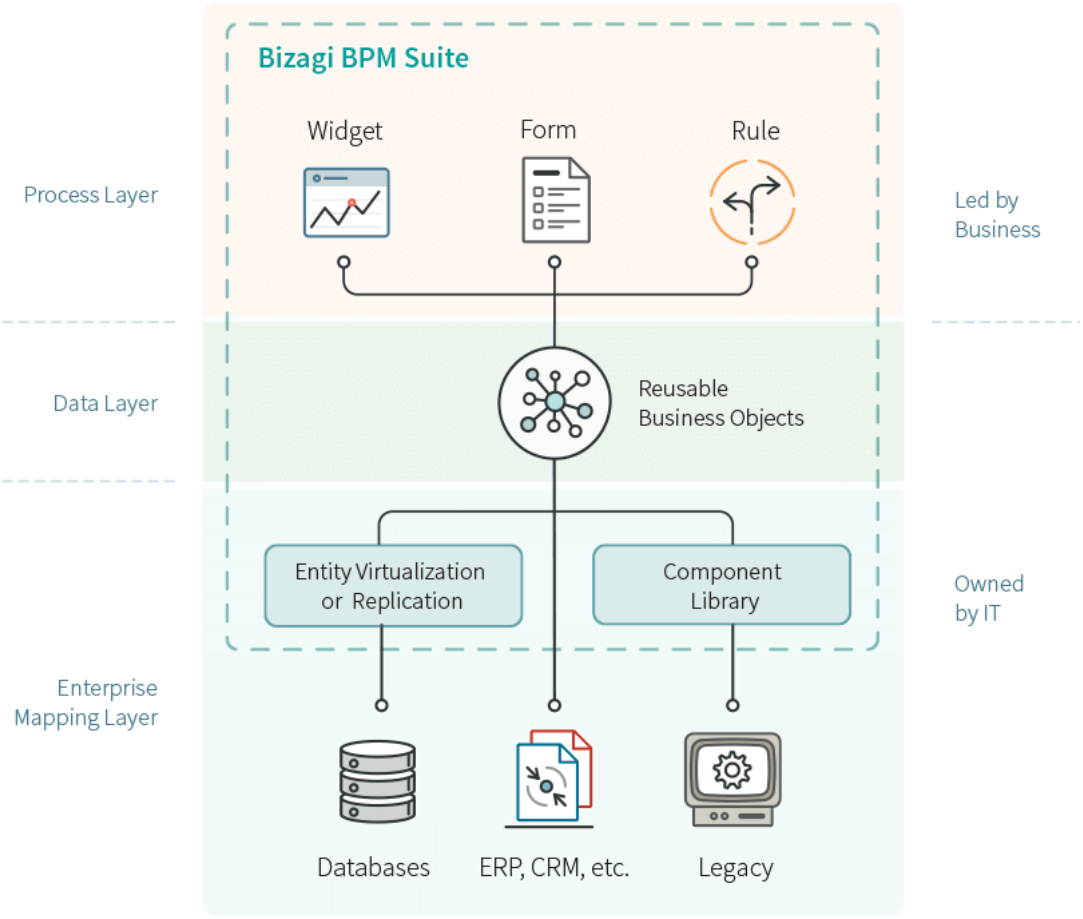
Lisa 2 EMF Forms prototüübi vaateid

Deklaratsiooni aine vaade

Joonis 22. Deklaratsiooni aine vaate näide

Lisa 3 Bizagi Studio tarkvara

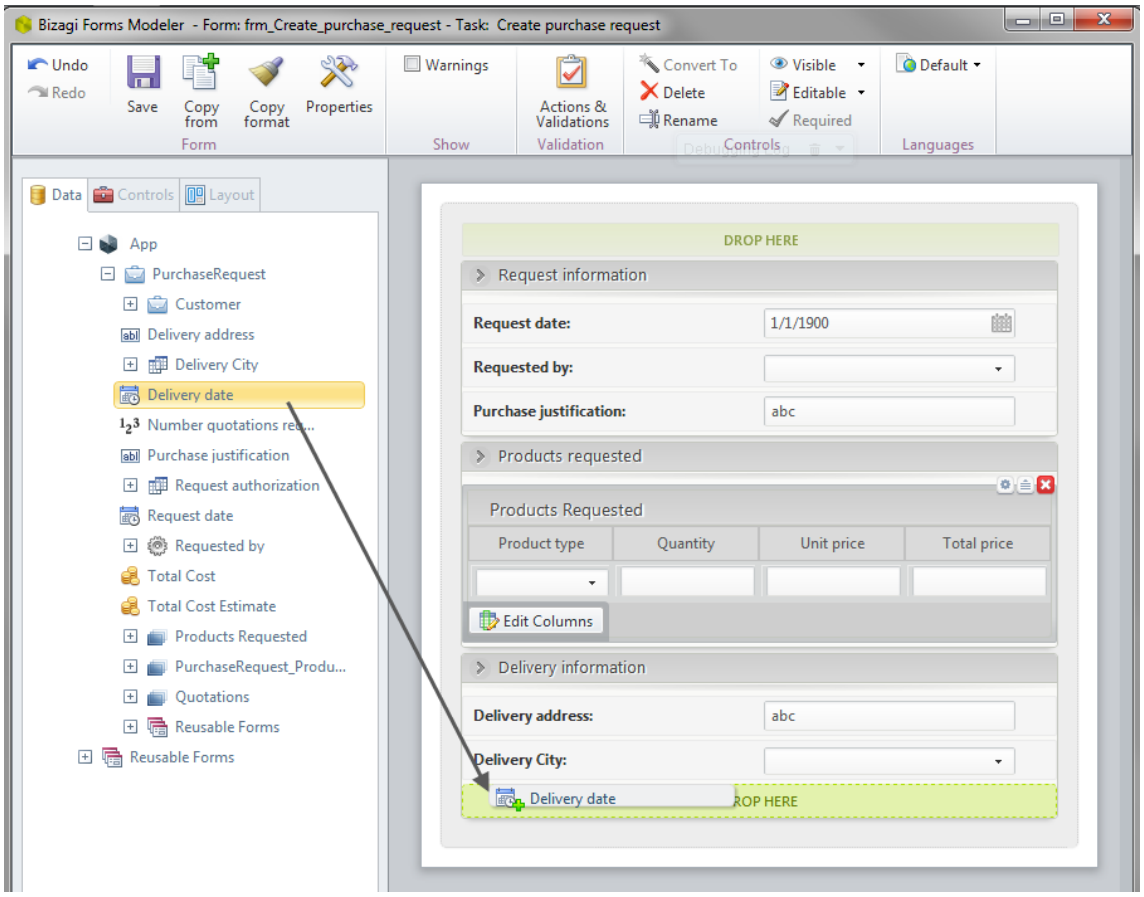
Bizagi BPM Suite kihiline ülesehitus



Joonis 23. Bizagi BPM Suite ülesehitus

Allikas: Bizagi BPM Suite V10.7 - User guide

Bizagi abil loodava vormi näidis



Joonis 24. Bizagi Forms näidis

Allikas: Bizagi BPM Suite V10.7 - User guide