TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond

Ain-Martin Kink 164346

ÄRIPROTSESSIDE MUDELIPÕHINE ARENDAMINE ECLIPSE SIRIUS PLATVORMIL

Bakalaureusetöö

Juhendaja: Mart Roost
Magistrikraad

Tallinn 2019

Autorideklaratsioon

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

Autor: Ain-Martin Kink
20.05.2019

Annotatsioon

Lõputöö „Äriprotsesside mudelipõhine arendamine Eclipse Sirius platvormil“ eesmärk on uurida, kas Eclipse Sirius on sobilik vahend äriprotsesside modelleerimise prototüübi loomiseks. Töö käigus kasutatakse mudelipõhise arhitektuuri põhimõtteid.

Esmalt tutvustatakse äriprotsesside arendamist ning simuleerimist. Siis järgneb mudelipõhise arhitektuuri kirjeldamine, mille käigus uuritakse selle põhimõtteid, eeliseid ja puuduseid. Lisaks tuuakse välja ka valdkonnaspetsiifilised keeled ning nendega seotud mõisted. Töö praktiline osa keskendub arendatud prototüübi tutvustamisele ning viiakse läbi üks praktiline ülesanne.

Töö tulemusena valmis prototüüp, millega saab äriprotsesse kirjeldada. Valminud lahendust võrreldakse Bizagi Modeler’iga, mis samuti on tööriist äriprotsesside kirjeldamiseks ning arendamiseks. Võrdlemisega saab teadmist loodud prototüübi asjakohasusest.

Lõputöö on kirjutatud eesti keeles ning sisaldab teksti 31 leheküljel, 6 peatükki ning 13 joonist, 0 tabelit.

Abstract

Model-Driven Development of Business Processes Using Eclipse Sirius

The bachelor thesis' „Model-Driven Development of Business Processes Using Eclipse Sirius" goal is to examine wheter Eclipse Sirius is a suitable tool to develop a prototype of business process modeling tool. Development relies on principles of model-driven architecture.

First part is introducing business process development and simulation. Then follows the description of model-driven architectur and its advantages and disadvantages. The second part of the thesis introduces and describes the prototype that has been created and with that the development of certain process is carried out.

As a result of the thesis, the prototype has been created with which a processes can be described. The prototype will be compared to Bizagi Modeler which also is a tool for business process development. With the comparison the relevance of the prototype can be identified.

Thesis is in Estonian and contains 31 pages of text, 6 chapters and 13 figures, 0 tables.

Lühendite ja mõistete sõnastik

Eclipse Foundation	***Eclipse Foundation*** Vabavaralise tarkvara Eclipse arendajate kommuuni haldaja
Eclipse Sirius	***Eclipse Sirius*** Mudeli põhist arendust toetav tarkvara
UML	***Unified modeling Language*** Ühtne modelleerimise keel
BPMN	***Business process modeling notation*** Äriprotsessi modeleeerimise notatsioon
MDA	***Model driven architecture*** Mudelipõhine arhitektuur
EMF	***Eclipse Modeling Framework*** Eclipse modeleerimise raamistik
Ecore	***Ecore*** EMF mudel
Metamudel	***Metamudel*** Mudelit kirjeldav mudel
Data-flow diagram	***Data-flow diagram*** Andmeliikumise diagramm
Flowchart	***Flowchart*** Tegevusvoo diagramm
AS-IS	***AS-IS*** Hetkel kehtiv protsess
TO-BE	***TO-BE*** Protsess optimeeritud kujul
Bizagi Modeler	***Bizagi Modeler*** Äriprotsesside modelleerimise tarkvara
OMG	***Object Management Group*** Tarkvaraarenduse standardite arendaja
.NET	***.NET*** Tarkvaraarenduse platvorm
SOAP	***Simple Object Access Protocol*** Andmeedastus protokoll erinevate rakenduste vahel

EJB	***Enterprise Java Beans*** Java äritarkvara arendamist toetav liides
CASE	***Computer aided software engineering*** Tarkvara tööriistade domeen rakenduste disainimiseks ja liidestamiseks
DSL	***Domain specific language*** Valdkonnaspetsiifiline keel
DSML	***Domain specific modeling language*** Valdkonnaspetsiifiline modelleerimise keel
SQL	***Structured query language*** Struktureeritud päringu keel
HTML	***Hypertext markup language*** Hüperteksti märkimiskeel
XML	***Extensible markup language*** Laiendatav märkimiskeel
GPML	***General-purpose modeling language*** Üldine modelleerimise keel
IDE	***Integrated development environment*** Integreeritud arendamise keskkond
GMF	***Graphical modeling framework*** Graafilise modelleerimise raamistik
Java	***Java*** Objekt-orienteeritud programmeerimise keel
RTE	***Runtime environment*** Operatsiooni poolt pakutav keskkond rakenduse käivitamiseks
Event	***Event*** BPMN sündmuse element
Gateway	***Gateway*** BPMN lüüsi element
Task	***Task*** BPMN tegevuse element
Sequence flow	***Sequence flow*** BPMN tegevusvoo element

Association	***Association*** BPMN seotuse element
Java Service	***Java Service*** Java programmeerimise keeles kirjutatud teenus

Jooniste nimekiri

Sisukord

Sissejuhatus

Käesolev bakalaureuse töö on kirjutatud teemal “ Äriprotsesside mudelipõhine arendamine Eclipse Sirius platvormil”. Antud teema valiti, et uurida, kas äriprotsesside arendamist on võimalik edendada, kui selleks kasutada värskemaid modelleerimise põhimõtteid, suurendades erinevate arendustöös vajalike ja koostatud mudelite osatähtsust.

1.1 Taust ja probleem

Äriprotsesside kirjeldamisel kasutatase üldiselt UML ja BPMN skeeme, simulatsioone või jooniseid. Lisaks joonistele koostatakse ka neid kirjeldav dokumentatsioon. Äriprotsessi analüüsi kvaliteedi parendamiseks võiks olla palju abi MDA põhimõtetest, mille alusel saab endale ise, vastavalt vajadustele, oma töövahend luua. Lähtudes MDA põhimõtetest on võimalik ärianalüüsile põhjalikumalt läheneda, sest lisaks visualiseerimisele läheneb mudelipõhine arhitektuur ka rohkem tehnilisest vaatenurgast.

Antud töö võib huvi pakkuda inimesele, kes tegeleb süsteemi- või ärianalüüsiga, kas siis kooli- või ametialases töös. Lõputöö käsitleb mudelipõhise arendamise põhimõtteid ja selle rakendamist, seega üldiselt võib see huvi pakkuda kõikidele erialal tegutsevatele inimestele.

1.2 Ülesande püstitus

Eesmärk on uurida Eclipse Sirius platvormi abil ühe äriprotsesside arendamist. Ootus on teada saada, kas ja kuidas on võimalik äriprotsesside kirjeldamist ja simuleerimist parendada, kasutades MDA põhimõtteid, mida toetab EMF. Eclipse Sirius platvormi uurimisel valitakse üks äriprotsess, mille koostamiseks ja simuleerimiseks soovitakse luua töövahendi protoüüp. Töövahendi loomise ja valitud äriprotsessi arendamise tulemusena peab saama ülesande püstitust kontrollida, valideerida ning seejärel teha vajalike järeldusi.

1.3 Metoodika

Töö tegemisel kasutatakse Eclipse modelleerimise tehnoloogia vahendeid, need võimaldavad erinevat liiki mudelite koostamist, kirjeldamist ja genereerimist. Töös püstitatud eesmärkide saavutamiseks luuakse valdkonnamudel, mis abstrakselt kirjeldab äriprotsessi olemust. Valdkonna mudel on alus, mille abil on võimalik äriprotsessi kirjeldamiseks vajalik töövahend ehitada. Loodud töövahendiga koostatakse terviklik äriprotsess.

1.4 Ülevaade kirjutatud tööst

Käesolev töö koosneb kuuest peatükist. Esimene peatükk on sissejuhatus, mille eesmärk on tutvustada töö tausta, ülesande püstitust ja metoodikat.

Teine peatükk tutvustab töö teoreetilisi aluseid, mis omakorda on aluseks praktilise poole realiseerimisele. Tutvustatakse äriprotsesside mõistet ning mudelipõhist arendust, selle eeliseid ja puuduseid. Viimaseks kirjeldatakse valdkonna spetsiifilisi keeli ja tuuakse nende kohta näiteid.

Kolmas peatükk pühendub töös kasutatavale tarkvarale, milleks on Eclipse Sirius. Lisaks tarkvarale mainitakse üldisemalt ka organisatsiooni Eclipse, kes tegeleb Siriuse arendamisega. Tutvustatakse ka EMF'i, tegu on raamistikuga, mida Sirius kasutab

Neljas peatükk kirjeldab tehtud tööd ehk lahendust püstitatud probleemile. Tutvustatakse loodud ecore mudelit, mille alusel genereeriti ja täpsustati äriprotsessi kirjeldamise prototüüp. Näidatakse elemente, mida on võimalik prototüübiga äriprotssesi koostades kasutada. Lisaks muule kirjeldatakse ka reaalne protsess, katsetamaks loodud prototüübi võimekust.

Viies peatükk analüüsib ja võrdleb arendatud lahendust sama eesmärki täitva tarkvaraga. Võrlduses olev tarkvara on Bizagi Modeler, mis on väga levinud töövahend äriprotsesside arendamiseks. Peatükis pakutakse välja ka edasised suunad, mida järgides võiks tehtud tööd jätkata.

Kuues peatükk võtab kirjutatud töö lühidalt kokku.

2 Äriprotsessid ja MDA

Järgnevas peatükis kirjeldatakse tööga seotud teoreetilisi aluseid, mis toetavad praktilist uurimist ja analüüsi. Peatükk kirjeldab üldisemalt äriprotsessi mõistet, sellega seotud üksikasju, modelleerimist ja simuleerimist. Äriprotsessidele lisaks toob välja spetsiifilisemalt MDA mõiste ning kirjeldab sellega seonduvaid eeliseid ja puuduseid.

2.1 Äriprotsess

Äriprotsess on kombinatsioon tegevustest ettevõttes või organisatsioonis, kombinatsiooni struktuur kirjeldab protsessi loogilise järjekorra ning sõltuvused, mille eesmärk on jõuda soovitud tulemuseni. Protsessi läbiviijad, ehk tegutsejad, teevad tegevusi, et saavutada konkreetset püstitatud eesmärki või eesmärke, mis toodavad protsessi kasutajale väärtust. Kirjeldatud sammud on korduvad ja neid võib teostada mitu erinevat tegutsejat korraga, eelistatult standardiseeritud ning parimal võimalikul viisil, et võimalikult väikse kuluga saavutada võimalikult palju kasu. Üldiselt äriprotsessid jagunevad kaheks – automatiseeritud ja käsitsi tehtavad. Kui tegu on käsitsi täidetava protsessiga, siis tehnoloogilisi vahendeid tegevustesse ei kaasata. Kui tegu on automaatse protsessiga, siis protsessi läbiviimiseks kasutatakse tehnoloogilisi lahendusi, mis aitavad protsessi läbi viia veelgi täpsemal, standardiseeritud ja optimiseeritud viisil. [1], [2]

2.2 Äriprotsessi modelleerimine ja simuleerimine

Alapeatükis kirjeldatakse täpsemalt äriprotsesside modelleerimist ja simuleerimist. Lisaks üldistustele tuuakse välja modelleerimise ja simuleerimisega seotud üksikasju. Tutvustatakse ka äriprotsessi simuleerimise tarkvara.

2.2.1 Äriprotsessi modelleerimine

Äriprotsessile toetudes on võimalik ettevõtet analüüsida ja realiseerida. Seetõttu on äriprotsessi modelleerimine väga oluline eeldus eduka ettevõtte loomiseks, mis kasutavad äriliste eesmärkide saavutamiseks mahukaid tegevusi. Ühtlasi on süsteemide arendamisel vaja aru saada, kuidas organisatsiooni siseselt erinevad äriprotsessid toimuvad. Modelleerimine on tõhus viis äriprotsesse ning selles sisalduvaid sündmusi, tegevusi ja

erandeid kirjeldada. Läbi modelleerimise on võimalik äriprotsessi üheselt mõistetavalt kujutada. [2]

Äriprotsessi modelleerimine on organisatsiooni äriprotsesside graafiline visualiseerimine, peamiselt kasutatakse selleks BPMN'i – äriprotsesside modelleerimise notatsiooni. Modelleerimise eesmärk on tuvastada potentsiaalsed arendamist vajavad kohad, mis takistavad sujuvat protsessitegevuste kulgu. Protsesside visualiseerimiseks kasutatakse peamiselt (*data-flow diagram*)andme- ja (*flowchart diagram*)tegevusvoodiagrammi. [3]

Äriprotsessi modelleerimise eesmärk on kas luua täiesti uus äriprotsess toetamaks organisatsiooni tegevust või täiustada juba olemasolevat äriprotsessi. Olemasoleva äriprotsessi arendamiseks on vajalik kirjeldada kaks erinevat varianti arendatavast äriprotsessist. Nendeks on AS-IS, ehk äriprotsess praegusel hetkel, ning TO-BE, ehk äriprotsess arendatud kujul ja tulevikus, variandid ärprotsessidest. Kirjeldatud teguviis annab eelduse kinnitamaks või ümber lükkamaks, kas plaanitud äriprotsessi kasutusele võtmine on organisatsioonile kasumlik või mitte. Arendades täiesti uut äriprotsessi, siis on mõistlik luua erinevaid variante ja proovida erinevaid stsenaariumeid, et leida optimaalsem viis eesmärgi saavutamiseks. [2], [3]

2.2.2 Äriprotsessi simuleerimine

Modelleeritud äriprotsessi on vajalik ka valideerida, et oleks alust arendatud äriprotsessiga kaasnenud muudatusi organisatsioonis sisse viia. Selleks, et äriprotsessi valideerida on vajalik kõnealuse protsessi AS-IS ja TO-BE variandid kirjeldada ning neid simuleerida. Simuleerimise tulemust võrreldes, saab teha kindlaks kumb variant kirjeldatud protsessidest on tulemuslikum.

Simulatsioon on arvuti-põhine süsteemi esitus, mis jäljendab süsteemi dünaamilist käitumist. Sama loogikat kasutades saab simuleerida ka äriprotsessi, kui see on piisava tasemeni kirjeldatud. Äriprotsessi simuleerimist saab vaadelda kui kogumit meetoditest, tehnikatest ning töövahenditest, mis toetavad äriprotsessi analüüsi ning arendamist. [4]

Praegusel hetkel populaarne tarkvara kirjeldamaks äriprotsesse ja nende simulatsioone on Bizagi Modeler. Tarkvara abil saab peale äriprotsessi kirjeldamist täpsustada protsessi stsenaariumit, mis on justkui simulatsiooni sisendiks. Mida täpsemalt on sisend

kirjeldatud, seda täpsemat tulemust on võimalik simulatsiooniga saavutada. Kui simulatsiooni tulemus on täpne, siis on alust äriprotsessi arendamise tulemusi realses elus implementeerida. [5]

2.3 Mudelipõhine arhitektuur

Alapeatükis tutvustatakse MDA ehk mudelipõhise arhitektuuri arendusmetoodikat. Selgitatakse välja põhjused, miks soovitakase MDA metoodikat kasutusele võtta. Lisaks kirjeldatakse nii MDA eeliseid kui ka puuduseid, mis selle kasutuselevõtuga kaasneda võiksid.

2.3.1 MDA - Model driven architecture

Mudelipõhine arhitektuuri raamistik on välja töötatud OMG ehk Object Management Groupi poolt. OMG eesmärk organisatsioonina on kirjeldada standardeid, millele tuginedes on uute loodavate tarkvara süsteemide integreerimine võimalikult sujuv ja kuluefektiivne. Eesmärkide saavutamiseks töödeldigi välja MDA raamistik, mis toetub mitmetel OMG poolt kirjeldatud standarditel.

MDA põhiline argument on katta tervet tarkvara arendustsüklit – hõlmates analüüsi, disaini, programmeerimist, testimist, komponentide koostoimimist. Lisaks ka tarkvara kasutusele võtmist kui ka hooldamist. Neid kõiki tegevusi on vajalik katta, seejuures tagada ka ülevaatlikkus arendustsüklist. [6]

Mudelipõhist arhitektuuri iseloomustavad mitmed mõisted, millele MDA arendus toetub:

- **Süsteem** – MDA kontekstis on tegu juba eksisteeriva või arenduses oleva tarkvarasüsteemiga;
- **Mudel** – funktsiooni, struktuuri ja süsteemi käitumise spetsifikatsioon antud kontekstis ja kindlast vaatepunktist vaadeldes. Mudel on tihti kirjeldatud kombineerides joonised ja teksti, üldiselt kasutades UML diagrammi ja seda kirjeldavat teksti;
- **Mudelipõhine ideoloogia –** kirjeldab lähenemist tarkvaraarendusse, milles mudeleid kasutatakse peamise vahendina tarkvara dokumeteerimises, analüüsimises, disainimises, arendamises, rakendamises ning hooldamises;

- **Arhitektuur –** tarkvarasüsteemi arhitektuur hõlmab endas süsteemi kuuluvaid komponente, nendevahel olevaid ühendusi ja suhtluse reegleid;

- **Vaatepunkt ja MDA vaatepunkt –** vaatepunkt abstraktne vaade tarkvarasüsteemile, mis pühendub vaid ühele konkreetsele süsteemile ignoreerides ebaolulisi detaile.

  MDA kasutab kolme erinevat vaatepunkti liike:

  - Struktuurist sõltumatu – vaatepunkt keskendub kontekstile ja tarkvara nõuetele, kuid mitte tehnilisele struktuurile;

  - Platvormist sõltumatu – vaatepunkt keskendub kontekstist väljas oleva süsteemi operatsioonidele ja võimetele;

  - Platvormist sõltuv – vaatepunkt keskendub vaid süsteemi platvormi detailidele, mis on kontekstiks võetud. N-ö täiendab platvormist sõltumatud vaatepunkti.

- **Platvorm –** platvorm on kogum allsüsteemidest ja tehnoloogiatest, mis koos tagavad funktsionaalsuses läbi liideste ja kasutusmustrite;

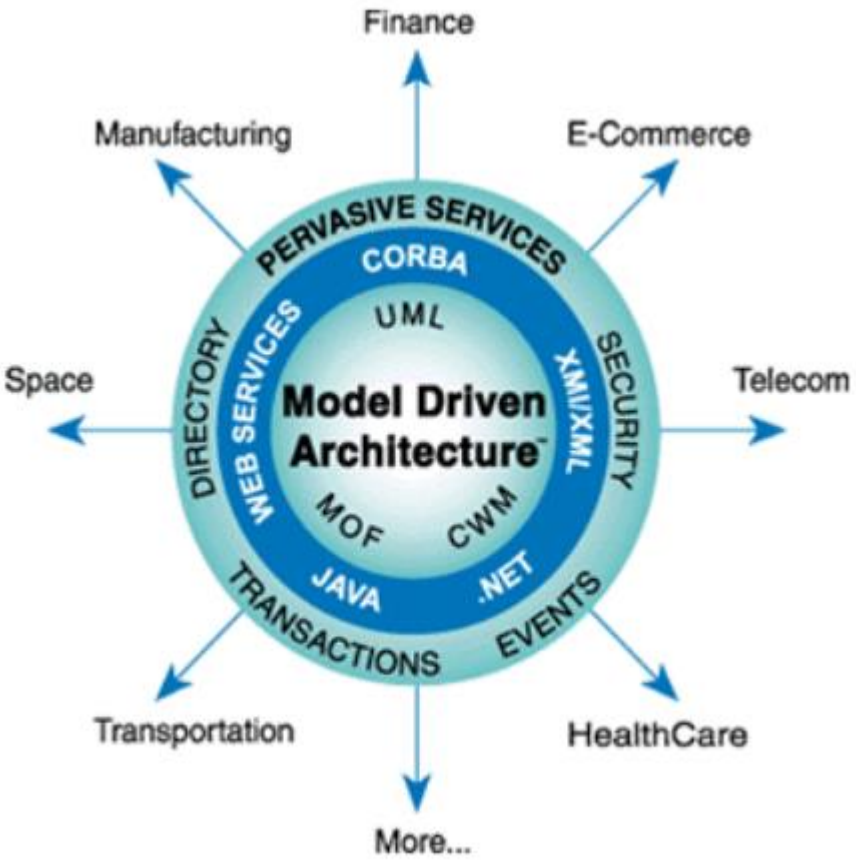
- **Platvormi sõltumatus** – platvormi sõltumatus on omadus, mis võib eksisteerida ja toimida sõltumatult teiste platvormide funktsionaalsusest ja olemasolust;

- **Platvormi mudel** - platvormi mudel kirjeldab platvormi tehnilisi omadusi, elemente ja teenuseid, mida platvorm pakub;

- **Mudeli muutmine** – mudeli muutmine on protsess, mille käigus ühte mudelit muudetakse ühe süsteemi raames. Muudatus hõlmab endas uue info lisamist platvormist sõltumatule mudelile, luues platvormist sõltuva mudeli;

- **Implementatsioon –** implementatsioon hõlmab endas kogu informatsiooni ja spetsiifikat, mis on vajalik süsteemi konstrueerimiseks ning käivitamiseks. Info on vajalik, et luua objekt ning lubada objektil pakkuda vajalikke teenuseid süsteemi terviku osana. [7], [8]

Mudelipõhise arhitektuuri kirjelduse visualiseerimiseks on välja toodu järgnev joonis:



Joonis 1. Mudelipõhise arhitektuuri kirjeldus

Allikas: Frank Truyen, Fast Guide to Model Driven Architecture

Olenemata soovitud eesmärgist, esimene samm MDA'l põhineval rakenduse arendamisel on alati luua platvormist sõltumatu mudel, mille kirjeldamiseks kasutatakse UML notatsiooni. Üldistatud mudelit on võimalik kohandada spetsiifilisemale platvormile, näiteks .NET, SOAP, EJB või muule, mida soovitakse kasutada. [8]

2.3.2 Lubatud eelised

Uute tehnoloogiate kasutusulevõtt eeldab, et see toob selle kasutajale ka mingit kasu. MDA kasutuselevõtuks on lubatud erinevaid eeliseid, peamiselt on need seotud mudelite loomisega, mis on masin-loetavad ning pakuvad pikaajalist paindlikkust erinevates aspektides:

- **Kasutatavate tehnoloogiate vananemine** – uusi tarkvaralahendusi on võimalik lihtsamini integreerida juba olemasolevale süsteemile;

- **Ühildatavus** – arendatud funktsionaalsust on võimalik kiiresti migreerida uutesse keskondadesse ja platvormidele, vastavalt ärivajadustele;

- **Produktiivus** – automeerides palju tülikaid ülesandeid, saavad arendajad keskenduda süsteemi põhiloogikale;

- **Kvaliteet** – loodud ja muudetud mudelite ja funktsioonide järjepidevus ning usaldusväärsus tagavad üleüldise süsteemi kvaliteedi;

- **Integratsioon** – integreerimine väliste süsteemidega on suuresti lihtsustatud;

- **Tarkvara hooldamne** – disaini kättesaadavus masin-loetavas vormingus annab analüütikutele, arendajatele ja testijatele otsese ligipääsu süsteemispetsifikatsioonile, lihtsustades tarkvara hooldustöid;

- **Testimine ja simuleerimine** – kirjeldatud mudeleid saab otseselt testida erinevate nõuetega ning erinevates struktuurides. Mudeleid saab kasutada ka simulatsioonides, näiteks testida süsteemide käitumist, mis on veel disainifaasis.

- **Kasumlikkus** – Ettevõtted suudavad investeeringud arendusprotsessi paremini enda kasuks tööle panna. Suur eeltöö lubab tulevikus hõlpsamini olemasolevale tarkvarale uut funktsionaalsust lisada.

Kirjeldatud eelised on piisavalt kasumlikud, et rakendada tarkvara arendamise protsessi mudelipõhist arhitektuuri. [7], [8]

2.3.3 Teadaolevad puudused

MDA kasutuselevõtt lubab endaga tuua palju tegureid, mis mõjutavad tarkvaraarendust postiivsel viisil. Eelmises peatükis on välja toodud eesmärgid, mida MDA-põhine arendus lubab. Neid lubadusi on tarvilik uurida, et kontrollida, kas need ka reaalses tarkvaraarenduses välja tulevad.

MDA’d tutvustavad allikad ning mõned MDA-põhiste tööriistade arendajad on väitnud, et antud raamisitku kasutuselevõtuga arendusele kuluv aeg väheneb. Põhjenduseks on toodud, et arendajad ei pea keskenduma sihtplatvormi ja infrastruktuuri õppimisele, vaid saavad oma aja suunata programmiloogika loomisele. Sellise väite reaalsus peaks võimaldama kiiresti uusi rakendusi luua ning neid olemasoleva edukalt siduda, seda ka

juhul kui sihtplatvormid muutuvad. Kui saab arendada sihtplatvormist sõltumatult, siis MDA'd kasutates peaksid loodavad rakendused valmima kiiresti ning rakendused peaksid omavahel suuremate probleemideta töötama. Nende väidete uurimiseks tehti Carnegie Mellon Ülikoolis, Pittspurgh'i linnas, uuring, mille eesmärk oli antud lubadusi tõestada või ümber lükata.

Esmalt võeti vaatluse alla, kas MDA kasutamine tarkvaraarenduses vähendab arendamisel kuluvat aega. See väide lükati osaliselt ümber, kuna tuvastati, et kui esimese rakenduse arendamisel MDA tööriista kasutati, siis kuluv aeg ei olnud väiksem. Probleem oli see, et esmakordselt pidi platvormi spetsiifiliselt tööriistade seadeid muutma, mis võtavad üsna suure aja. Küll aga kui arendada samale platvormile uus rakendus, siis enam tööriistade seadeid ei pea seadistama ning saabki vaid tarkvaraarendusega tegeleda ja säästetakse aega – see kehtib vaid siis kui uus rakendus teha samale platvormile. Juhul kui rakendus tuleb luua uuele platvormile, siis tööriista ümberseadistamisele kuluv aeg vähendab MDA arendusviisiga säästetud aega. Esmakordselt kulub arendajatel aega ka üsnagi keeruliste MDA-põhiste tööriistade keskkondade õppimisele.

Teiseks uuriti väidet, mis lubas arendajal vaid programmiloogikale keskenduda, et arendaja ei pea teadma platvormi ja infrastruktuuri detaile, millele uut tarkvara luuakse. Kui eelmine väide lükati osaliselt ümber, siis antud väide lükati täielikult. Eeldades, et arendus algab hetkest, kui disain on arendajale edastatud, siis MDA tööriist ja mudelid on vaja esmalt seadistada enne kui üleüldse mingit koodi on võimalik genereerida. Seadistamine nõuab arendajalt väga põhjalikke teadmisi sihtplatvormist. MDA-põhine arendus vabastab arendajat infrastrutkuuri koodikirjutamisest, kuid mitte tööriista seadistamisest, mis eeldavad laiapõhjalisi teadmisi sihtplatvormist ja sellel all olevast infrastruktuurist.

Uuring järeldas, et hetkeseisus olevad MDA tööriistad pole veel piisava tasemeni arendatud, et need suudaksid MDA-põhise arenduse lubadusi täita, suudab vaid tingimuslikult ja pigem kesiselt. Juhul kui suudetakse tagada platvormist sõltumatus, siis mudelipõhine arendus võib-olla järgmine arenguetapp CASE-vahendite ja arenduskeskkondade jaoks. [7], [9]

2.4 Valdkonnaspetsiifilised keeled

Käesolev alapeatükk kirjeldab valdkonnaspetsiifilisi keeli ja nendega seonduvat. Valdkonnaspetsiifilised- ehk domeeni spetsiifilised keeled(DSL) omavad ühte kindlat ülesannet, mida keel täitma peab. Domeeni spetsiifilist keelt arendatakse silmas pidades kindlat või kindlaid ülesandeid, mida täitma peab. Arendatud valdkonna spetsiifiline keel on justkui abstraktne kirjeldus, mille abil on võimalik midagi konkreetsemat kirjeldada. Lisaks domeeni spetsiifiliste keeltele on ka domeeni spetsiifilised modelleerimse keeled(DSML), mis omab suurt rolli MDA arenduses. Kasutates MDA põhimõtteid tarkvara arendamsiel, defineeritakse üsna tihti domeenile vastav modelleerimise keel, selle eesmärk on suurendada semantilist täpsust. [10], [11]

2.4.1 DSL – valdkonna spetsiifiline keel

Domeeni spetsiifilised keeled on hakanud saama laiemat kasutust, kuna populaarsust on kogunud domeeni spetsiifiline modelleerimine. Domeeni ehk valdkonna spetsiifiline modelleerimine keskendub ühele valdkonnale. Nii domeeni spetsiifilist modelleerimist kui ka keelt iseloomustab asjaolu, et seda kasutatakse kindlalt määratud ülesande või ülesannete täitmiseks. Ülesandest püstitusest tulenevalt arendatakse vajalik valdkonna spetsiifiline keel, mille abil on võimalik ülesannet lahendada. [12]

Arendatud domeeni spetsiifiline keel pakub intiutiivse liidese domeeni arendajale, kes domeeniga ehk valdkonnaga tegeleb. DSL omab süntaksit ja semantikat, mida on võimalik esitada ja käivitada – seetõttu on võimalik keelt ka kompileerida. Teadmist, mida keel omab esitatakse domeenikoodis. Domeenikood on kood, mis keskendubki ühele kindlale eesmärgile.

DSL'id on tõestanud, et need on väga võimekad ja edukad. Näiteks üldlevinud valdkonna spetsiifilised keeled on:

- SQL – kasutatakse andmebaaside arendamisel;
- HTML – kasutatakse veebiarenduses;
- XML – kasutatakse andmete kodeerimiseks.

Nimetatud keeled on arendatud silmas pidades ühte kindlat eesmärki ning neid kasutatakse sihipäraselt. [13]

2.4.2 DSML – valdkonna spetsiifilised modelleerimise keeled

Modelleerimise keel kannab eesmärki luua kontseptuaalse mudelite klass. Eesmärgi täitmiseks pakub keel kontseptide kogumi. Kontseptuaalne modelleerimise keel on kirjeldatud süntaksi ja semantika abil. Modelleerimise keeled liigituvad kaheks GPML ehk üldise eesmärgiga modelleerimise keeled ning DSML ehk valdkonna spetsiifilised modeleermise keeled.

GPML ehk üldise eesmärgiga modelleerimise keel on sõltumatu valdkonnast. Selle eesmärk on hoopis katta suure hulga domeene korraga, seetõttu on see ka üldisem. Keel koosneb üldistest modelleerimise kontseptidest, milleks on näiteks olemitüüp, klass, atribuut ja muud taolis. Üldise eesmärgiga modelleerimise keel ei sisalda endas valdkonnaspetsiifilise keele aspekte.

DSML ehk domeeni spetsiifiline modelleerimise keel on GPML'i vastand. DSML on mõeldud kasutama ühe kindla valdkonnaspetsiifiliselt. DSML täiendab üldiseid modelleerimise kontsepte spetsiifilisemate asjaoludega, mis kirjeldatakse vastavalt käsitletava valdkonna tehniliste nõudmiste alusel. Valdkonnaspetsiifiline modelleerimise keel lubab modelleerijal luua domeenitasandi kontsepte, ilma et peaks etteantud ja primitiivseid kontsepte kasutama. Selle tulemusel muutub modelleerija töö efektiivsemaks, kuna saab keskenduda vaid ühele kindlale valdkonnale, ilma et peaks kulutama aega üldiste modelleerimise kontseptide sobitamisele enda valdkonna spetsiifikaga. [14]

3 Töös kastatavad tehnoloogiad

Käesolev peatükk kirjeldab tehnoloogiaid, mida töö kirjutamisel kasutati. Praktilise poole teostamiseks kasutati Eclipse programme ja raamistikke, mis toetavad mudelipõhist arendust. Vastavalt on nendeks Eclipse Sirius ja Eclipse Modeling Framework(EMF).

3.1 Eclipse

Alapeatükk tutvustab Eclipse poolt loodud organisatsiooni ning tarkvara, mida töös kasutatakse.

3.1.1 Eclipse Project ja Eclipse Foundation

Eclipse organisatsioon on loodud 2001. aastal tolleaegsete tarkvaraarenduse turuliidrite poolt, nimeks sai Eclipse Project. Organisatsiooni loomises osales esialgu IBM, kuid aegamisi liitus veel mitmed tarkvaraarendus ettevõtted. Lisaks Eclipse Project organisatsioonile loodi ka Eclipse Foundation organisatsioon aastal 2004. Eclipse Foundation on mittetulundusühing, mille eesmärk on hallata Eclipse kasutajate kommuuni ning kontrollida, et kommuun oleks neutraalne, avatud ning läbipaistev. Eclipse pakub üldiselt vabavaralist tarkvara, mida arendavad vabatahtlikud arendajad erinevatest firmadest. Vabavaralise tarkvara pakkumiseks pakub Eclipse litsentsi 'Eclipse Public License' – litsents lubab tarkvara kasutada, muuta, jagada ning seda kõike tasuta. Kõige tuntum ja laialdasemalt kasutatav tarkvara, mida Eclipse pakub on Eclipse IDE, ehk Eclipse integreeritud arenduskeskkond. Lisaks IDE'le pakub ka sellele erinevaid laiendusi. [15]

3.1.2 Eclipse Sirius

Sirius Eclipse on Eclipse poolt arendatud töövahend, see lubab kasutajal luua vastavalt enda vajadustele graafilise modelleerimise töölaua. Kasutajale on antud kasutatada üldised tööriistad MDA arenduseks, kuid vastavalt oma eesmärkide täitmiseks on võimalik sellele teha täiendusi. Sirius kasutab Eclipse Modeling tehnoloogiaid, milleks on EMF ja GMF, vastavalt *Eclipse Modeling Framework* ja *Graphical Modeling Framework*. Siriuse arendamist toetavad firmad OBEO ning Thales. [16], [17]

Eclipse poolt pakutavat tarkvara Eclipse Sirius kasutatakse bakalaureusetöö kirjutamiseks, sest tegu on vabavaralise tarkvaraga, mida saab vastavalt vajadustele

laiendada. Seeläbi on võimalik uurida mudelipõhise arhitektuuri ideid ning teostada äriprotsessi arendamist.

3.2 Eclipse Modeling Framework

Alapeatükk tutvustab Eclipse poolt loodud raamistikku – Eclipse Modeling Framework. EMF on Eclipse Modeling Project alla kuuluv lahendus. EMF on modellerimise raamistik ja koodigenereerimise vahend, mille abil on võimalik luua tööriistu ja rakendusi, mis põhinevad stuktureeritud andmemudelil. Mudeli spetsiifika kirjeldamisest, kasutades XMI'd, pakub EMF tööriistu ja *runtime* keskkonna toetust luues Java klassid kirjeldatud mudeli tarvis. Koos sellele loob ka ühildamise jaoks vajalikud klassid, mis lubavad vaadata ja muuta koostatud mudelit tagantjärgi. EMF eesmärk on leida tasakaal programmeerimise ja modelleerimise vahel ning seejuures tutvustada üldisemalt MDA põhimõtteid. [15]

Eclipse Modeling Framework koosneb kolmest põhiosast:

- EMF – tuum-EMF raamistik koosneb metamudelist, nimetus sellel on Ecore. Ecore kirjeldab mudeleid ja toetab nende kasutamist *runtime* keskkonnas. Metamudelit on võimalik jooksvalt muuta ja pakub muudatuste sujuvat sisseviimist.

- EMF.Edit – EMF.Edit raamistikusse kuuluvad üldistatavad ja taaskasutatavad klassid, mis lubavad luua EMF mudelite redaktoreid.

- EMF.Codegen – EMF.Codegen raamistiku eesmärk on genereerida kõik vajalik loomaks EMF mudeliredaktorit. See sisaldab endas graafilist liidest, mille abil on võimalik kirjeldada redaktorit ning seeläbi see ka luua. EMF.Codegen raamistik kasutab Eclipse JDT komponenti.

Koodi genereerimine kolmetasandiline:

- Mudel – pakub kõikidele mudeliklassidele Java liidsed ning implementeerimise klassid. Lisaks pakub ka metaandmete implementeerimise klassi.

- Adapterid – genereerivad implementeerimise klassid, mis on ühildatud mudeliklassidega. Nende läbi on võimalik mudeli klassi muuta ja kuvada.

- Redaktor – loob struktureeritud redaktori, mis vastab Eclipse EMF mudeli redaktori soovitatud nõudmistele. Redaktor on alguspunkt, millest alates on kasutajal võimalik loodud mudelit muutma hakata.

Kõiki osasid on võimalik uuesti genereerida, kui on teada, et on vajalik metamudelis mingeid muudatusi teha. Uuesti genereerimine jätab alles kasutaja poolt tehtud muudatused *runtime* keskkonnas. See ongi üks eelis MDA arenduses, et kui on vaja algtasemel väiksemaid muudatusi teha, siis seni tehtud töö ei lähe kaduma. [18]

EMF loodi võimaldamaks lihtsat mudelipõhist arendust, genereerides suurema osa koodist, see aga võimaldab uusi süsteeme ja töövahendeid sujuvamat kasutusele võtta. Genereeritakse üldisem osa koodist, milles kasutaja peab mõningaid muudatusi tegema, et see eesmärke paremini täidaks. EMF abil saab läbi mudeli genereerida reaalset koodi, seega EMF eesmärk on näidata, et graafiliste mudelite koostamine on vajalik enamaks kui vaid arendatava tarkvara dokumentatsiooni jaoks. [19]

4 Äriprotsessi arendamise realisatsioon

Käesolev peatükk kirjeldab äriprotsessi kirjeldamist ja arendamist kasutades mudelipõhise arhitektuuri põhimõtteid. Realisatsiooni teostamiseks kasutatakse Sirius Eclipse tarkvara võimalusi. Võimaluste abil luuakse uus valdkonnamudel, mille abil on võimalik püstitatud eesmärgi poole liikuda. Eesmärk on äriprotsesside kirjeldamine ja arendamine, siis loodav valdkonnamudel peab seda toetama. Koostatud mudeli abil kirjeldatakse BPMN elemendid, millega on võimalik juba konkreetsemat äriprotsessi koostada ja simuleerida. Realisatsiooni käigus on vajalik hinnata ja välja selgitada, kas käesolev metoodika suudab vajalikke eesmärke täita. Juhul kui mõnda eesmärki täita ei suudeta või pole muul põhjusel piisavalt efektiivne, siis tuuakse vastavad põhjused välja.

4.1 Valdkonnamudeli koostamine

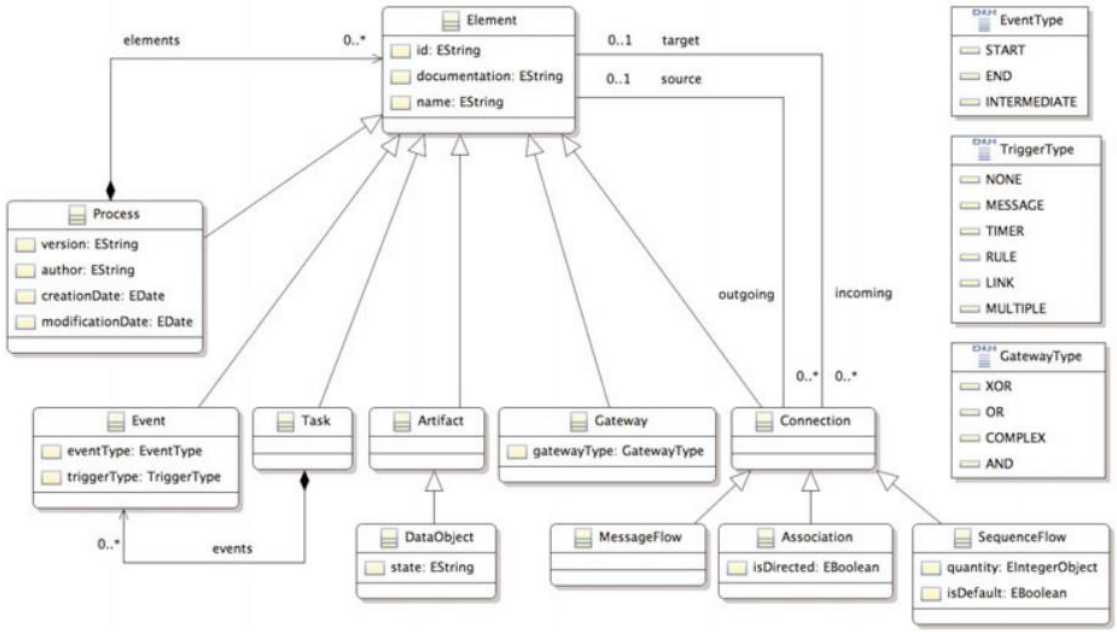
Uurimise all on äriprotsesside arendamine, mille käigus viiakse läbi Tallinna Tehnikaülikooli õppeinfosüsteemi ainete deklareerimise protsess. Eesmärk on uurida, kas seda protsessi on võimalik kirjeldada ja ka simuleerida kasutades mudelipõhise arhitektuuri põhimõtteid. Äriprotsessi kirjeldamise standard on BPMN ehk äriprotsesside modelleerimise notatsioon. Selleks, et BPMN elemente kirjeldada ja kasutada, on vaja koostada vastav ecore mudel.

Valdkonnamudel koostatakse ja esitatakse diagrammina. Mudeli kirjeldamiseks kasutatakse EclipseUML tööriista. Tulemus on terviklik klassidiagramm, mis kirjeldab äriprotsessi ning selle koostamiseks vajalikke olemeid.

4.2 Koostatud metamudel

Äriprotsesside kirjeldamiseks loodud valdkonnamudel pärineb Richard C. Gronback *Eclipse Modeling Project: A Domain Specific Language Kit* raamatust. [10] Allikas olevale mudelile pidi tegema mõned täiendused, et seda saaks kasutada.

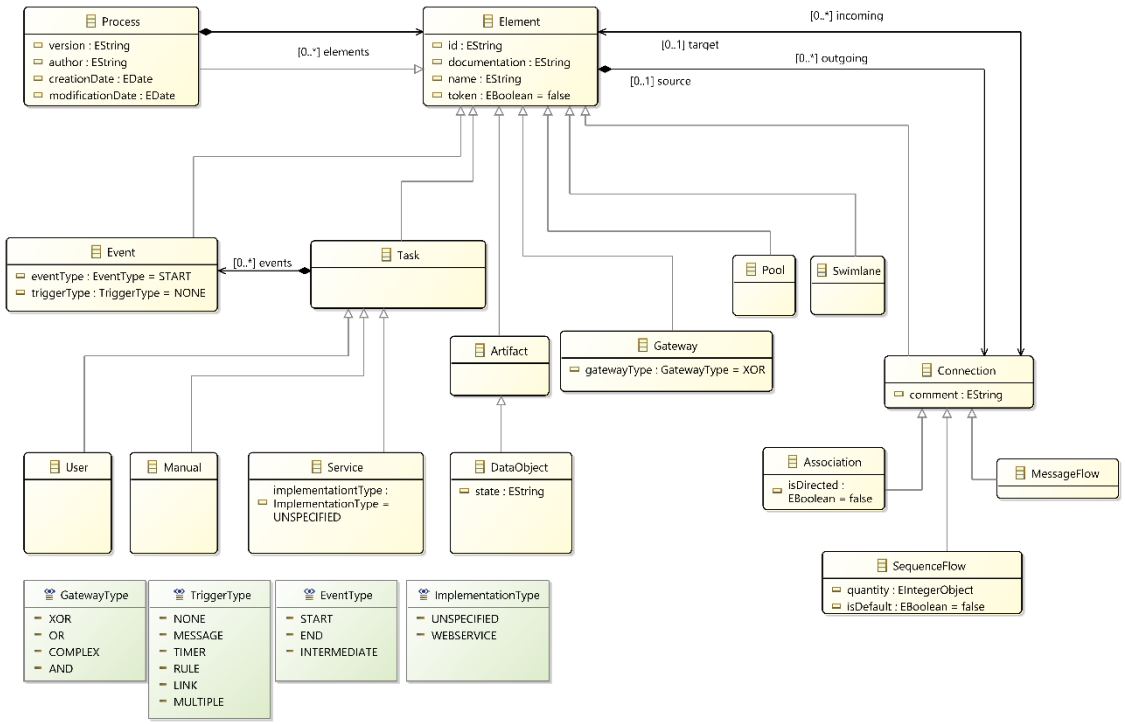
Järgnevalt on esitatud kaks joonist, esimene nendest kujutab allikas kirjeldatud valdkonnamudelit ning teine aga töö käigus koostatud valdkonnamudelit.



Joonis 2. Üldine äriprotsessi koostamise valdkonnamudel

Allikas: *Eclipse Modeling Project: A Domain Specific Language Kit*

Joonis kirjeldab valdkonnaspetsiifilist mudelit, mille abil on võimalik koostada äriprotsess. Küll aga alllikast saadud mudel ei kata antud töös püstitatud eesmärke, seetõttu ei saanud seda ilma täiendusteta kasutada. Järgneval joonisel on allikast saadud mudeli edasiarendatud kuju. Muudatused tehti eesmärgiga, et äriprotsessi koostamise prototüüp omaks rohkem võimalusi ning seeläbi oleks võimekam. Peamiselt hõlmavad muudatused soovitud elementide kasutamise võimaldamist ning simulatsiooni toetamist. Millest tuleb välja ka mudelipõhise arenduse eelis, et kui arendatavale lahendusele seatakse lisanõudeid, siis on võimalik vastavaid muudatusi läbi mudeli üsna kergesti implementeerida.



Joonis 3. Koostatud valdkonnamudel

Täiendatud valdkonnamudelil on võrreldes allika mudeliga üsna palju muudatusi tehtud. *Task* ehk tegevuse olem on täpsemaks muudetud. Tegevust iseloomustab tegevusetüüp, kuid iga tegevusetüüp on ka eraldi olemina kirjeldatud, mis pärinevad *Task* olemist. Selle põhjus on see, et eriliiki tegevustel võivad olla erinevad atribuudid, mida kasutada. Teiseks on *Element* olemile juurde tehtud token atribuut, see on aluseks kontrollile, mille abil saab kontrollida, millises positsioonis käivitatud simulatsioon parasjagu on. Kolmandaks on *Connection* elemendil atribuut *Comment*, mille abil saab näiteks hargnemise tingimust seada ja kuvada. Lisatud on ka enumeratsioon *ImplementationType*. *ImplementationType* määrab, mis viisil *Service Task* tüüpi tegevust realiseeritakse. Uued olemid on ka *Pool* ning *Swimlane*, nende abil on võimalik näidata tegutsejat, kes tegevusi läbi viib.

4.3 Metamudeli alusel BPMN elementide kirjeldamine

Äriprotsess koosneb elementidest, mis võivad olla sündmused, ühendused, ülesanded ja lüüsid ning nende erivariandid. Elementide kasutamiseks on esmalt vaja neid kirjeldada ning nende kirjeldamist toetab eelmises peatükis välja toodud ecore metamudel, mis tegelikult on valdkonnamudel.

Elementide kirjeldamist teostatakse Sirius Eclipse runtime keskkonnas, milles on selleks kasutada Sirius Specification Editor. Tööriista abil muudetakse projektis .odesign faili, mis sisaldab endas elementide kirjeldust. Sirius Specification Editor'ga saab luua eesmärgi täitmiseks vajalikke elemente, mida kasutatakse hiljem diagrammi koostamisel. Lisaks elementidele on vajalik luua ka vastavad tööriistad, millega elemente diagrammile tekitada. Iga erineva elemendi loomiseks on vaja kirjeldada oma tööriist, et element diagrammile tekitada.

Järgnevalt esitatakse, kuidas on äriprotsessi koostamiseks vajalikud BPMN elemendid kirjeldatud. Kirjeldused toetuvad valdkonnamudelile ning on koostatud Sirius Specification Editor tööriista kasutades.

4.3.1 Sündmuse ehk Event element

Sündmus on üldjuhul mingi asjaolu, mis kas alustab või lõpetab äriprotsessi. Kuid sündmus võib ka äriprotsessi keskel asuda, mis mõjutab protsessi kulgu. Sündmuse elemendi koostamisel pidi seda arvesse võtma.

- *Start event* ehk algatav sündmus algatab äroprotsessi ning sellest võib ühenduse teha ainult väljuval suunal. Teisisõnu on see kõige esimene element äroprotsessi jadas ning seda liiki võib vaid üks olla. [20]

- *Intermediate event* ehk sündmus, mis võib protsessi kestel toimuda, mis omakorda mõjutab äriprotsessi ennast. Tegevus võib selle suunas liikuda kui ka sellest välja, ehk siis liikumise suund piiratud pole. [20]

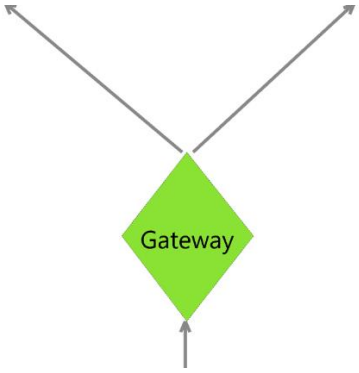
- *End event* ehk protsessi lõpetav sündmus on kõige viimane sündmus ning sellesse saab tegevus vaid liikuda. Peale seda ei tohi enam ühtegi tegevust ega sündmust toimuda. [20]

Joonis 4. Eriliiki sündmuste kujutamine diagrammi koostamise prototüübis

BPMN näeb, ette et kirjeldatud kolmel sündmuse elemendil on omakorda eriliigid, kuid antud töös pole neid nii suure täpsusega kirjeldatud. Neid pole nii täpselt kirjeldatud, kuna on teada, et kirjeldatavas äriprotsessis pole vaja neid nii täpselt kujutada.

4.3.2 Lüüsi ehk Gateway element

Lüüs ehk *gateway* element väljendab protsessis hargnemist. Tegu on otsustuskohaga, milles kontrollitakse, kas mingi tingimus on täidetud ja vastavalt tingimuse täidetusele liigutakse tegevusega vastavas suunas. Tegu on elemendiga, milles tegevusesuund pole piiratud, ehk protsessi suund võib nii selleni kui sellest välja liikuda. [20]



Joonis 5. Lüüsi kujutamine protsessi koostamise prototüübis

Lüüsi kujutatakse protsessis sarnaselt BPMN standardile ehk siis rohelise rombina. Antud töös pole kirjeldatud rombi erivariante kuna vajadus selleks puudub. Mudelipõhise arendusega on see aga lihtsasti tehtav.

4.3.3 Tegevus ehk Task element

Tegevus iseloomustab ühte kindlat tegevust, mis on vajalik protsessi läbiviimiseks. Tegevuse käigus selgub ka tingimus, mille alusel otsustatakse võimaliku hargnemise suund. Seega ka tegevuse ehk task element on element, milles tegevusesuund pole piiratud, ehk protsessi suund võib selleni kui ka sellest välja liikuda. [20]

Töö käigus kirjeldati kolm eriliiki tegevust:

- *Manual Task* ehk manuaalselt täidetav tegevus on tegevus, mis on võimalik teha käsitsi, ilma tehnoloogia abita. Diagrammil kujutatakse seda helesinist värvi ristkülikuna. [20]

- *User Task* ehk kasutaja tegevus on tegevus, mida täidab kasutaja ning mis eeldab ka tehnoloogia abi. Kasutaja ja tehnoloogia kaasabil täidetakse tegevus. Diagrammil kujutatakse seda sinist värvi ristkülikuna. [20]

- *Service* ehk teenuse poolt tehtav tegevus on tegevus, mida täidab täielikult arendatud lahendus, näiteks mingi kindel programmikood. Diagrammil kujutatakse seda tumesinist värvi ristkülikuna. [20]

Joonis 6. Tegevuste eriliikide kujutamine protsessi koostamise prototüübis

Antud hetkel on valitud eristamise viisiks erineva varjundiga sinised toonid, mis ei pruugi parim variant olla, kuna võib kasutajale segadust tekitada. Parem variant oleks kasutada logosid, nende abil on võimalik tegevuse tüüpi paremini ära tunda.

4.3.4 Tegevusvoo ühendus ehk Sequence Flow element

Tegevusvoo ühenduse element määrab millises järjekorras protsessielemendid on reastatud ja millises järjekorras neid teostatakse. Visuaalselt kujutatakse seda joonena, millel on nool näitamaks suunda. [20]

Joonis 7. Tegevusvoo ühenduse kujutamine protsessi koostamise prototüübis

Antud ühendusega saab siduda omavahel sündmusi, tegevusi ning lüüse. Nool näitab, millises suunas protsess liigub.

4.3.5 Seotus ehk Association element

Seotus ehk *Association* elementi kasutatakse kui on vaja mingi objekti seotust sündmuse, tegevuse või lüüsiga näidata. Visuaalselt kujutatakse seda diagrammil punktiirjoonena. [20]

Joonis 8. Seotuse kujutamine protsessi koostamise prototüübis

Seotus erineb tegevusvoo ühendusest sellega, et seda saab vaid tegevuse ja artefakti vahelise seose loomiseks kasutada ning sellel puudub suund.

4.3.6 Andmeobjekt ehk data object element

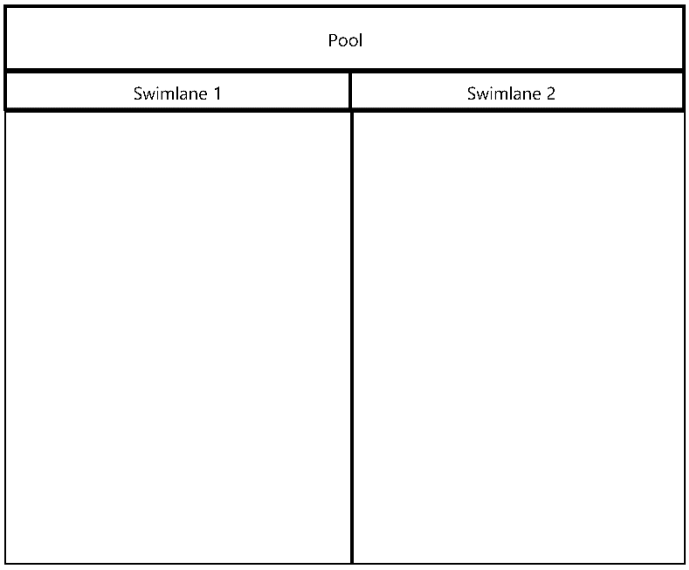
Andmeobjekt näitab mingeid andmeid, mida on mingi tegevuse tegemiseks vajalik teada. Seose andmeobjekti ja tegevuse vahel loob seotuse ehk *association* element. [20]

Joonis 9. Andmeobjekti elemendi kujutamine protsessi koostamise prototüübis

Andmeobjekti kasutatakse, et tuua protsessidiagrammile rohkem detailsust. Detailsuse lisamise täpsem eesmärk on luua programmeerijale parem ettekujutus tegevusega seotud ja vajaminevast infost.

4.3.7 Pool ja Swimlane ehk basseini ja ujumisraja objektid

Basseini eesmärk on näidata üldisemat tegevust ning sellel üldiselt kuvatakse protsessi pealkirja. Basseiniga võidakse kirjeldada ka haru organisatsioonist, milles tegutsejad viivad tegevusi läbi. Ujumisraja eesmärk on kirjeldamisel näidata protsessis osalevat tegutsejat ning tema poolt tehtavat tegevust. Ujumisrada on vajalik näidata iga äriprotsessis figureeriva tegutseja kohta. [20], [21]



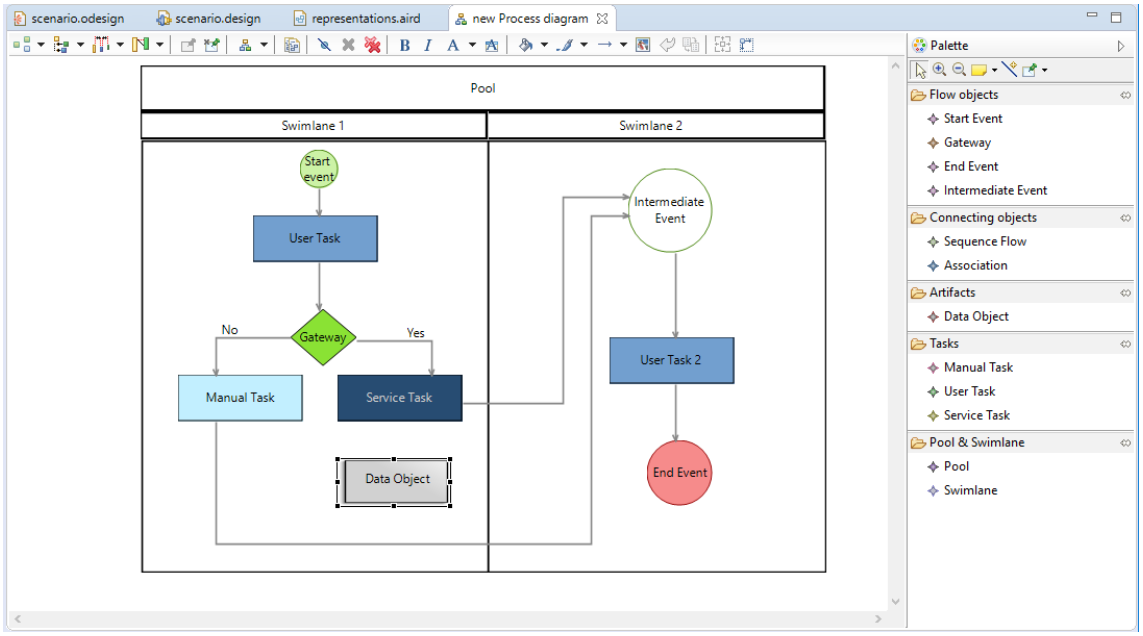
Joonis 10. Ujumisraja ja basseini elemendid protsessi koostamise prototüübis

Eelneval joonisel on kujutatud kombinatsioon kahest elemendist – ujumisrada ning bassein. Sellise pildi saamiseks on diagrammil need elemendid eraldi luua. Hetkel toimivad need vaid visualiseerimise eesmärgil, rakenduslik eesmärk puudub.

4.4 Äriprotsessi kirjeldamise töölaud

Järgnev peatükk annab ülevaate töölaua prototüübist, millega on võimalik äriprotsessi jada kirjeldada. Äriprotsessi kirjeldamiseks kasutatakse eelnevas peatükis mainitud BPMN elemente ja nende esitlemiseks diagrammil loodud tööriistu.

Prototüüp koosneb tööriistadest, millega on võimalik diagrammile BPMN elemente tekitada. Tööriistad on grupeeritult vastavalt millist liiki elementi on võimalik diagrammile luua. Töölaual oleval diagrammil on elemendid kujutatud vaid näite toomiseks, nendel puudub seos reaalse protsessiga.



Joonis 11. Äriprotsessi kirjeldamise prototüübi töölaud

Prototüübis on kasutada erinevaid elemente ning need on grupeeritud järgnevalt.

Flow objects ehk voo elementide grupp sisaldab:

- *Start event* ehk algatavat sündmust;
- *Intermediate event* ehk vahepealset sündmust;
- *End event* ehk lõpetavat sündmust;
- *Gateway* ehk lüüsi.

Connecting objects ehk ühenduse elemendite grupp sisaldab:

- *Sequence flow* ehk tegevusvoo ühenduse element;
- *Association* ehk seotuse element.

Artifacts ehk artefaktide grupp sisaldab:

- *Data Object* ehk andmeobjekt.

Tasks ehk tegevuste elemendid

- *Manual task* ehk käsitsi tehtava tegevuse element;
- *User task* ehk kasutaja poolt tehtava tegevuse element;
- *Service task* ehk masina poolt tehtava tegevuse element.

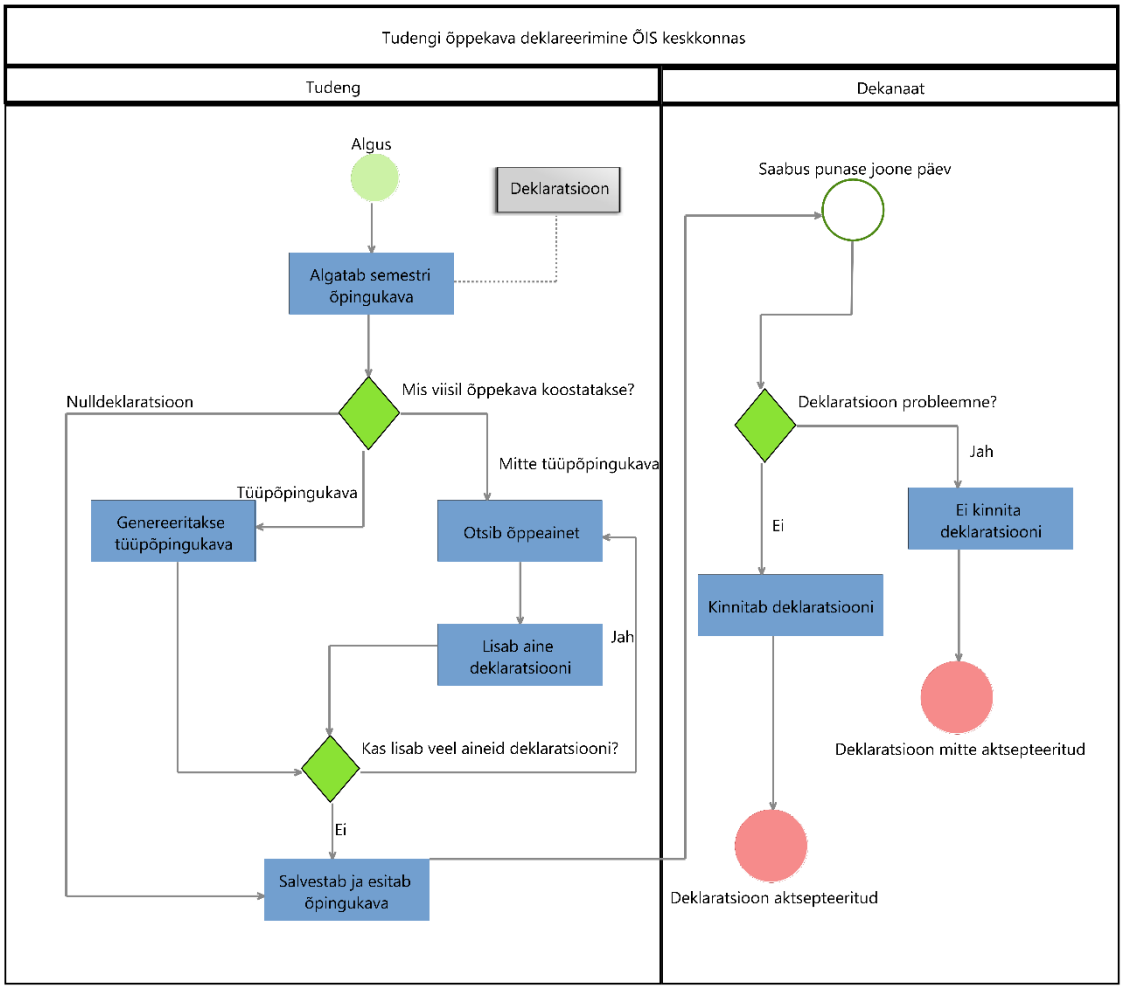
Pool & Swimlane grupp sisaldab:

- *Pool* ehk bassein;
- *Swimlane* ehk ujumisrada.

Koostatud töölaud on piisav, et saaks püstitatud eesmärki täita. Teisisõnu on piisav, et koostada üks äriprotsess ning proovida seda simuleerida.

4.5 Äriprotsessi kirjeldav mudel

Uurimise all olev protsess on Tallinna Tehnikaülikooli õppeinfosüsteemi ainete deklareerimise protsess ning seda täpsemalt tudengi vaatepunktist. Tegu on lihtsustatud variandiga deklareerimise protsessist, kuna töö käigus soovitakse keskenduda pigem tööriista arendamisele ja selle võimaluste katsetamisele, kui väga täpse äriprotsessi kirjeldamisele. Äriprotesess eskendub ainete deklareerimisele, mitte selle käigus tekkinud probleemide lahendamisele. Loodud prototüübi abil koostati kõnealune protsessikirjeldus.



Joonis 12. Prototüübi abil koostatud ÕIS ainete deklareerimise protsess

Protsess hõlmab endas tudengi poolt tehtavaid tegevusi, mida ta teeb kasutades Tallinna Tehnikaülikooli Õppeinfosüsteemi. Peale õpingukava salvestamist ja esitamist jääb see punase joone päeva ootele, mille saabudes kinnitatakse deklaratsioon lõplikult.

4.6 Äriprotsessi simulatsiooni realiseerimine

Järgnev peatükk kirjeldab äriprotsessi simuleerimise realiseerimisest Eclipse Siriuse võimalusi kasutades. Äriprotsessi arendamisel on simuleerimisel tähtis koht, kuna see annab arendajale arusaama arendatud äriprotsessi efektiivsusest.

Eclipse Sirius pakub mitmeid erinevaid tööriistu ja võimalusi diagrammil muudatusi teha. Simuleerimisi realiseerimiseks prooviti töö käigus nendest mõnda, mis esmapilgul tundusid piisavalt võimekad. Esmalt võeti eesmärgiks diagrammil olevatel elementide peal liikumine, et protsess õiges järjekorras n-ö läbi käia. Selleks valiti Sirius Eclipse

tööriistad *Open Dialog* ning *Change Context*. *Open Dialoog* tööriista käivitamisel esitatakse kasutajale dialoogivorm, millele saab infot kuvada ning millega saab otsuse langetada. Eesmärk oli selle abil tuvastada aktuaalne element, kus simulatsioon hetkel asetseb ning kasutajapoolsel kinnitamisel liigub vastav simulatsioon edasi järgneva protsessi elemendi juurde. *Change Context*'i eesmärk antud olukorras oleks olnud värskendada jooksvalt simulatsiooni seisu s.t võtta järgnev element ja teha see aktuaalseks. Selle tööriistaga oleks olnud võimalik ka visuaalselt kuvada protsessi läbikäimist, näiteks muuta elementi erivärviga, et oleks kasutajal võimalik tuvastada ketke positsioon simulatsioonis.

Töö käigus selgus, et otsest viisi selle realiseerimiseks siiski polnud. Eelnevalt nimetatud tööriistad esialgu tundusid, et toetavad realisatsiooni, kuid kahjuks ei vastanud sellele. Üle äriprotsessis olevate elementide liikumine jäi realiseerimata, siis polnud alust ka protsessi simuleerimist edasi uurida. Siin kohal on võimalik pakkuda üks variant, milleks on vastava *Java Service* kirjutamine ja selle ühildamine prototüübiga. *Java Service* tuleb kirjutada Java programmeerimise keeles, siis see otseselt enam Eclipse Siriuse poolt pakutav valmis variant pole.

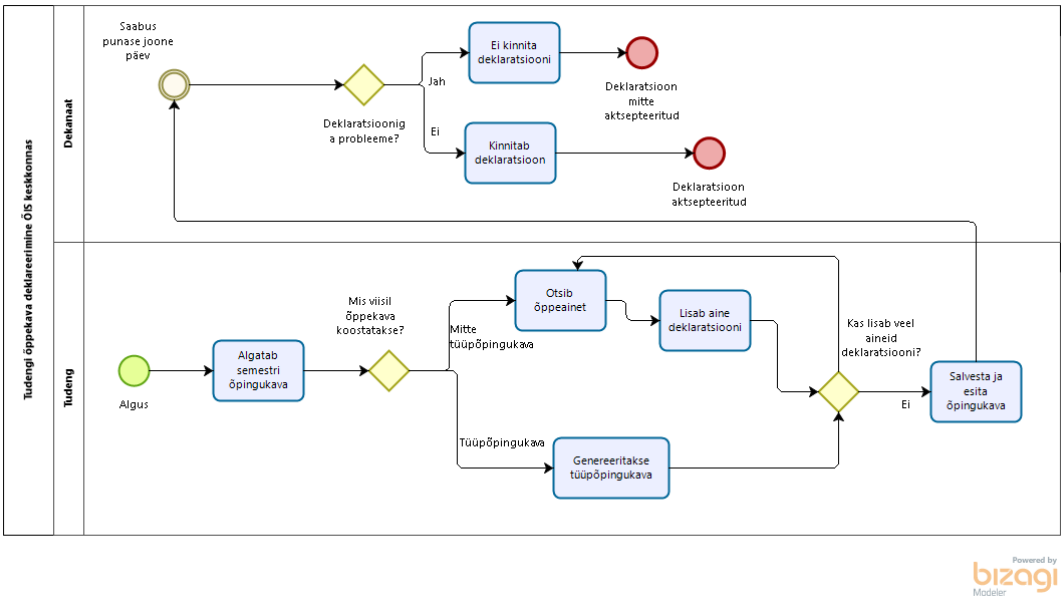
5 Järeldused

Peatükk tutvustab Bizagi Modeler tarkvara, mis keskendub äriprotsesside kaardistamisele. Tehakse läbi sellega protsessi koostamine ning võrreldakse töö käigus koostatud prototüübiga. Tehakse vajalikud järeldused, mille käigus pakutakse välja ka edasised suunad, kuidas praegust töötulemust edasi arendada.

5.1 Bizagi Modeler ja koostatud protsess

Bizagi Modeler on Bizagi poolt arendatud tarkvara, mis keskendub äriprotsesside arendamisele. Tarkvara pakub töölauda, millega on võimalik koostada protsess, kasutades selleks erinevaid BPMN elemente. Üks asi on selle koostamine, teine on aga protsessi läbi testimine. Protsessi testimiseks pakub Bizagi simuleerimise vahendeid. Simuleerimiseks saab kasutaja anda sisendparameetrid ning simulatsioon annab vastuseks tulemuse protsessi efektiivsusest.

Töö käigus arendati prototüüp, millega saab samuti protsesse kirjeldada. Prototüübi asjakohasuse välja selgitamiseks koostati nii prototüübiga kui ka Bizagi Modeler'iga sama protsess. Selle käigus selgusid nende kahe tööriista eelised ja puudused. Järgnevalt kujutatakse, milline näeb välja ÕIS deklareerimise protsess, mis on koostatud Bizagi Modeler modelleerimise tarkvaraga.



Joonis 13. Bizagi abil koostatud ÕIS ainete deklareerimise protsess

Tegu on sama protsessiga, mis peatükis 4.5 koostati prototüübiga. Katsetades kahte erinevat tööriista, et saavutada sama eesmärk annab aluse neid tööriistu omavahel võrrelda.

5.2 Bizagi ja prototüüp

Koostades ühte ja sama äriprotsessi kasutades selleks kahte erinevat töövahendit, Bizagi Modeler ja arendatud prototüüp, saab neid kahte võrrelda. Järgnevalt tuuakse esile nende kahe eelised ja puudused ning viiakse läbi võrdlemine.

Bizagi Modeler'il võimekus erineva keerukuse ning mahuga protsesse kirjeldada, seetõttu kasutavad seda mitmed suured ettevõtted ja kogenud äriprotsesside arendajad. Seega sellest tulenevalt polnud takistuseks Bizagi Modeler'iga ka eesmärgiks võetud protsessi arendamine. Subjektiivselt võib öelda, et diagrammi koostamine oli kohati ebamugav. Peamine põhjus oli see, et elementide ühendused ei toiminud ega asetanud diagrammile alati nii, kui oli soovitud. Üsna palju aega läks selle peale, et elemendid diagrammil oleksid arusaadavalt esitatud ning diagramm ise oleks kergesti loetav. Bizagi Modeleri kasuks räägib ka asjaolu, et sellega on võimalik elementide erivariante kasutada, olgu selleks siis erinevad *task*, *gateway* ja *event* elemendid. See võimaldab diagrammi elemente täpsemini kujutada.

Prototüübi võimekust hinnates, siis sellega sai samuti eesmärgiks võetud protsessi ära kirjeldada. Kui Bizagi Modeler'is oli elementide vaheliste seoste tegemine kasutajaliideses üsnagi ebamugav, siis subjektiivselt hinnates on prototüübis elementide ühendamine sujuvam ja intuitiivsem. Kirjeldatud protsess on nii prototüübis kui ka Bizagi Modeler'is arendatud kujul sarnaselt arusaadav. Seega võib järeldada, et protsessi kirjeldamine on nendel kahel variandil üsna samal tasemel.

Bizagi Modeleri kindel eelis prototüübi ees on, et Bizagi Modeler'iga on võimalik komplektne protsessi simulatsioon läbi viia, mis on hindamatu väärtusega protsessi arendamisel. Prototüübi arendusega korrektselt toimiva simulatsioonini ei jõutud, seega prototüübiga seda kahjuks teostada ei saa.

Nii mudelipõhiselt arendatud prototüübiga kui ka Bizagi Modeler'iga on võimalik terviklik protsess kirjeldada. Küll aga prototüübi arendusmaht on suure tõenäosusega palju madalam, lõputöö raames arendatud, siis on selles sisalduvat funktsionaalsust oluliselt vähem. Eesmärk, et prototüübiga saaks äriprotsessi koostada on saavutatud, kuid simulatsiooni teostamise eesmärk jäi saavutamata.

5.3 Edasised suunad

Töös kasutati mudelipõhise arenduse põhimõtteid, et koostada toimiv prototüüp, millega on võimalik äriprotsesse koostada. Eelnevas peatükis tehti töötulemuse kohta järeldusi, käesolev peatükk keskendub saavutatud tulemuse edasi arendamisele. Soovitused peavad silmas tulevikku, kui soovitakse samal teemal kirjutada uus lõputöö või üldiselt kõnealust temaatikat edasi arendada.

Kõige suurem vajaka jäämine puudutab äriprotsessi simuleerimist, milleni küll jõuti, kuid toimivat lahendust välja ei arendatud. Toimiv lahendus jäi ilmselt Eclipse Sirius poolt pakutavate töövahendite ning ajapuuduse taha kinni. Eclipse Sirius ei pakkunud koheselt kasutatavaid töövahendeid, mis toimiksid nii kui oleks soovitud. Teisalt ajapuudusest tingitud *Java Service*'i süvenemise puudulikkus ei lubanud ka teenuse kirjutamist, millega oleks ilmselt võimalik olnud protsessi simuleerimine realiseeritud. Sellest tulenevalt tehti ka järeldus, et edasine suund simuleerimise suhtes ongi *Java Service*'i kirjutamine ning rakendamine.

Prototüübis protsessi kirjeldamist oleks võimalik arendada nii, et kasutajal oleks võimalus rohkem eriliike elemente kasutada ning neil oleks välimus, mis teeks diagrammi lugedes kohe selgeks, millise elemendiga on tegu. Hetkel on näiteks *task* tüüpi elementide eritüüpide näitamine lahendatud erivärvidega, mis pole kõige parim variant, kuid siiski annab aluse eristamiseks. Kindlasti leidub võimalus protsessi koostamist diagrammil intiuitiivsemaks teha. Näiteks, et poleks vaja iga ühenduse tekitamisel uuesti töölaualt ühendust valida ja see diagrammile paigutada. Kasutajamugavus on tarkvara kasutamisel väga oluline ning kindlasti ei saa seda tähelepanuta jätta.

Kokkuvõte

Bakalaureuse töö „Äriprotsesside mudelipõhine arendamine Eclipse Sirius platvormil“ eesmärgiks oli uurida mudelipõhise arenduse põhimõtteid. Tuginedes mudelipõhise arenduse põhimõtetele oli eesmärgiks võetud ka prototüübi arendamine, millega on võimalik äriprotsesse arendada. Arendatud prototüübi võimekust hinnati võrreldes tuntud äriprotsesside modelleerimise töövahendiga, Bizagi Modeler. Lisaks mudelipõhise arhitektuurile uuriti ka valdkonnaspetsiifilisi keeli ja modelleerimise keeli.

Tuginedes MDA põhimõtetele, on võimalik arendada äriprotsesside modelleerimiseks mõeldud prototüüp. Prototüübi aluseks on valdkonnamudel, mis kirjeldati. Toetudes koostatud valdkonnamudelile kirjeldati vajalikud BPMN elemendid, nendevahelised võimalikud seosed ning elementide välimus, mida diagrammil kuvatakse. Eelnevad tegevused andsid aluse konkreetse protsessi kirjeldamiseks, milleks oli ÕIS ainete deklareerimine. Protsessi koostamine oli edukas.

Edasised ideed, kuidas tehtud tööd edasi arendada hõlmavad prototüübis koostatud äriprotsessi simuleerimist, kuna praeguse töö käigus toimiva lahenduseni ei jõutud. Simulatsiooni realiseerimiseks oleks vaja uurida Java Service võimalusi ja ühildamist prototüübiga.

Püstitatud töö eesmärk sai saavutatud, sest sai teha järeldusi Eclipse Sirius tarkvara ja mudelipõhise arenduse eeliste ning puuduste kohta prototüüpi arendades.

Kasutatud kirjandus

[1] PNMSoft, „Business Process,“ PNMSoft, [Võrgumaterjal]. Available: http://www.pnmsoft.com/resources/bpm-tutorial/business-process/. [Kasutatud 2019 03 04].

[2] R. S. Aguilar-Saven, „Business process modelling: Review and framework,“ *Elseveir: International journal of production economics,* pp. 129-149, 2003.

[3] N. Greene, „Business Process Modeling: Definition, Benefits and Techniques,“ [Võrgumaterjal]. Available: https://tallyfy.com/business-process-modeling/. [Kasutatud 04 03 2019].

[4] N. Melao, „Research Gate,“ 01 1999. [Võrgumaterjal]. Available: https://www.researchgate.net/publication/281526133_BUSINESS_PROCESS_SIMULATION_AN_OVERVIEW. [Kasutatud 04 03 2019].

[5] Bizagi, „Introducing Modeler Services,“ Bizagi, [Võrgumaterjal]. Available: https://www.bizagi.com/en/products/bpm-suite/modeler. [Kasutatud 04 03 2019].

[6] A. M. M. Dr. Shahid Nazir Bhatti, „Model Driven Architecture,“ 04 2015. [Võrgumaterjal]. Available: https://www.researchgate.net/publication/274916541_Model_Driven_Architecture. [Kasutatud 20 03 2019].

[7] O. Oidekivi, „Kasutajaliideste mudelipõhine arendamine EMF forms abil,“ Tallinna Tehnikaülikool, Tallinn, 2014.

[8] F. Truyen, „The Fast Guide to Model Driven Architecture, The Basics of Model Driven Architecture,“ 2006. [Võrgumaterjal]. Available: https://www.omg.org/mda/mda_files/Cephas_MDA_Fast_Guide.pdf. [Kasutatud 15 03 2019].

[9] L. W. Grace Lewis, „Model Problems in Technologies for Interoperability: Model-Driven Architecture,“ Carnegie Mellon University, Pittsburgh, 2005.

[10] R. C. Gronback, „Eclipse Modeling Project: A Domain-Specific Language,“ Addison-Wesley, 2009.

[11] F. V. O. P. Giovanni Giachetti, „Improving Automatic UML2 Profile Generation for MDA Industrial Development,“ 0302-9743, Valencia, 2008.

[12] M. Iseger, „Domain-specific modeling for generative software development,“ developerFusion, 23 06 2010. [Võrgumaterjal]. Available: https://www.developerfusion.com/article/84844/domainspecific-modeling-for-generative-software-development/. [Kasutatud 08 04 2019].

[13] Inventsoft, „Domain Specific Languages,“ Inventsoft, [Võrgumaterjal]. Available: https://www.intentsoft.com/intentional-technology/domain-specific-languages/. [Kasutatud 11 04 2019].

[14] U. Frank, „Multi-Perspective Enterprise Modelling: Background and Terminological Foundation,“ Universität Duisburg-Essen , Essen, 2011.

[15] Eclipse, „About the Eclipse Foundation,“ Eclipse, [Võrgumaterjal]. Available: https://www.eclipse.org/org/. [Kasutatud 13 04 2019].

[16] Eclipse, „Sirius Documentation,“ Eclipse, [Võrgumaterjal]. Available: https://www.eclipse.org/sirius/doc/. [Kasutatud 13 04 2019].

[17] Eclipse, „Sirius,“ Eclipse, [Võrgumaterjal]. Available: https://www.eclipse.org/sirius/. [Kasutatud 13 04 2019].

[18] Eclipse, „Eclipse Modeling Framework (EMF),“ Eclipse, [Võrgumaterjal]. Available: https://www.eclipse.org/modeling/emf/. [Kasutatud 13 04 2019].

[19] F. Budinsky, D. Steinberg, E. Merks, R. Ellersick ja T. J. Grose, „Eclipse Modeling Framework: A Developer's Guide,“ Addison-Welsey, 2003.

[20] Lucidchart, „What is Business Process Modeling Notation,“ Lucidchart, [Võrgumaterjal]. Available: https://www.lucidchart.com/pages/bpmn. [Kasutatud 30 04 2019].

[21] Visual Paradigm, „Using BPMN Pool and Lane in Business Process Diagram (BPD),“ 05 11 2010. [Võrgumaterjal]. Available: https://knowhow.visual-paradigm.com/business-process-modeling/pool-and-lane/. [Kasutatud 15 05 2019].