TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond

Ain-Martin Kink 204003

VEEBIPÕHISE MODELLEERIMISTARKVARA LOOMINE VALDKONNASPETSIIFILISTE MODELLEERIMISKEELTE PLATVORMIL

Magistritöö

Juhendaja: Mart Roost
magistrikraad

Tallinn 2023

Autori deklaratsioon

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

Kuupäev: 03.01.2023

Annotatsioon

Käesoleva magistritöö eesmärk on uurida mudelipõhisel arhitektuuril ja *no-code* arendamisel põhinevat tarkvaraarendamise metoodikat ning selle alusel arendada veebipõhiselt kasutatav modelleerimise tarkvara. Sellist arendusmetoodikat lubab Eclipse Sirius Web platvorm, mis on ka töö uurimusobjektiks. Modelleerimistarkvara arendamisel lähtutakse ökonoomsest ja komponendipõhisest arendusmetoodikast.

Töös valiti BPMN valdkond, mille põhjal defineeriti valdkonna nõuetele vastav valdkonnaspetsiifiline mudel ehk valdkonnamudel. Valdkonnamudeli põhjal arendati ka vaade ehk tarkvara realisatsioon äriprotsesside kaardistamiseks. Tarkvaraga koostatakse ka üks protsess, valideerimaks arendatud funktsionaalsust.

Töö tulemuseks on mudelipõhiselt arendatud veebipõhine modelleerimise töövahend. Vastavalt tulemusele hinnatakse ja analüüsitakse valitud arendusmetoodikat, Eclipse Sirius Web platvormi ja realiseeritud tarkvara. Platvormi võrreldakse Eclipse Sirius Desktop platvormiga ning realiseeritud tarkvara Eclipse Sirius Desktopis arendatud lahendusega ning Bizagi Modeler tarkvaraga. Arendusmetoodika ja platvormi analüüs ning võrdlemine teiste sarnaste lahendustega annab aluse arendusmetoodika ja platvormi valideerimiseks.

Lõputöö on kirjutatud eesti keeles ning sisaldab teksti 59 leheküljel, 7 peatükki, 30 joonist, 2 tabelit.

Abstract

Development of Web-Based Modelling Software Using Domain-Specific Language Modelling Platform

The Master thesis' goal is to examine the development of web-based modelling software which relies on model driven architecture and no-code development. This approach is enabled by Eclipse Sirius Web platform which also is the main object of this research. Additionally the development is based on economic and component-based methodology.

The domain of business process modelling notation was selected from which the domain model was defined according to the BPMN requirements. Based on the domain model a view, in other words a software, was developed which allows creating the business process diagrams. To validate the functionality of developed software a business process will be mapped with it.

Main result of the thesis is web-based modeling software which was developed using model driven approach. According to the result the methodology, Eclipse Sirius Web platform and developed software are analyzed and evaluated. The platform will be compared to Eclipse Sirius Desktop and developed modelling software will be compared with Bizagi Modeler and a software that was developed using Eclipse Sirius Desktop. Comparison and analysis provides a basis for validation of development methodology and platform.

Thesis is written in Estonian and contains 59 pages of text, 7 chapters, 30 figures, 2 tables.

Lühendite ja mõistete sõnastik

Eclipse Sirius Desktop	***Eclipse Sirius Desktop*** Eclipe Sirius töölaua tarkvara
Eclipse Sirius Web	***Eclipse Sirius Web*** Eclipse Sirius veebipõhine tarkvara
EMF	***Eclipse Modeling Framework*** Eclipse modelleerimise raamistik
Domeenimudel	***Domeenimudel*** Valdkonda kirjeldav mudel
Metamudel	***Metamudel*** Mudelit kirjeldav mudel
Ecore	***Ecore*** EMF mudel
Obeo Designer	***Obeo Designer*** Töölaua rakendus, mis toetub Eclipse modelleerimise raamistikule
Obeo Cloud Platform	***Obeo Cloud Platvorm*** Veebirakendus, mis toetub Eclipse modelleerimise raamistikule
BPMN	***Business process modeling notation*** Äriprotsesside modelleerimise notatsioon

DSML	***Domain-specific modeling language*** Valdkonnaspetsiifiline modelleerimise keel
GPML	***General purpose modeling language*** Üldise eesmärgiga modelleerimise keel
MDA	***Model-driven architecture*** Mudelipõhine arhitektuur
Representation	***Representatsioon*** Domeenimudeli esituse kirjeldus
API	***Application programming interface*** Rakenduse programmiliides
AS-IS	***AS-IS*** Hetkel kehtiv protsess
TO-BE	***TO-BE*** Protsess optimeeritud kujul
UML	***Unified modeling language*** Ühtne modelleerimise keel
Edge	***Edge*** Ühendusjoon
Node	***Node*** Punkt

No-code	***No-code*** Tarkvaraarendamise viis ilma programmikoodi kirjutamata
SaaS	***Software as a Service*** Tarkvara teenusena
PaaS	***Platform as a service*** Platvorm teenusena
IaaS	***Infrastructure as a service*** Infrastruktuur teenusena
SLA	***Service level agreement*** Teenustaseme kokkulepe
Runtime	***Runtime*** Rakenduse käivitamise keskkond
Legacy	***Legacy*** Aegunud tehnoloogiga käigus olev tarkvara
MOF	***Meta model facility*** Mudelipõhise arenduse standard
EDOC	***Enterprise distributed object computing*** Mõiste kirjeldus
XMI	***XML Metadata Interchange*** Metaandmete andmevahetuse standard

Spring	***Spring*** Java platvormi rakenduse raamistik
React	***React*** JavaScript kasutajaliidese teek
PostrgreSQL	***PostgreSQL*** Relatsioonilise andmebaasi haldamise süsteem
GraphQL	***GraphQL*** Andmete pärimise keel

Jooniste loetelu

Tabelite loetelu

Sisukord

Sissejuhatus

Magistritöö on kirjutatud teemal „Veebipõhise modelleerimistarkvara loomine valdkonnaspetsiifiliste modelleerimiskeelte platvormil“. Töö teema on valitud, et täiendavalt uurida minu kirjutatud bakalaureuse töös saavutatud tulemusi, mille pealkirjaks on „Äriprotsesside mudelipõhine arendamine Eclipse Sirius platvormil“. [1] Bakalaureuse töö keskendus mudelipõhise arhitektuuri uurimisele, töö tulemusena arendati mudelipõhisel ja *no-code* arendusmetoodikal põhinev lahendus, millega on võimalik kirjeldada BPMN standardile vastavaid äriprotsesse. Töö eesmärk oli luua mudelipõhiselt arendatud ja laiendatav tarkvara, töö tulemuse põhjal on ka järgnevatel aastatel tudengid oma lõputöö raames sama temaatikat edasi uurinud. Käesolev magistritöö keskendub samuti mudelipõhise arenduse metoodika täiendavale uurimisele ja praktiseerimisele, aga luues seeläbi veebipõhiselt kasutatava modelleerimistarkvara, tuginedes praegu kättesaadavale mudelipõhise arenduskeskkonna tehnoloogiale.

1.1 Taust ja probleem

Tarkvara loomisel on oluline, et valmis lahendus oleks lõppkasutajale piisavalt mugavalt kasutatav ja võimalikult lühikese arendustsükliga kättesaadav. Lisaks on oluline, et tarkvara lahendus on lihtsasti täiendatav ja uuenduste evitamine on võimalikult sujuv protsess. Sellest tulenevalt on töö eesmärgiks uurida, kuidas bakalaureuse töös arendatud tarkvara viia üle mugavamasse veebipõhisesse keskkonda, seejuures järgides mudelipõhise arhitektuuri ja *no-code* arendusmetoodika põhimõtteid, mis tähendab, et arendatav tarkvara on loodud toetudes mudelitele ja ilma programmikoodi kirjutamiseta. Bakalaureuse töös valmis tarkvara, millega on võimalik edukalt koostada BPMN standardile vastavaid äriprotsesside skeeme. Töö tegemisel aga selgus, et tarkvara juures erinevate muudatuste tegemine ning nende realiseerimine on üsna tülikas, ajamahukas ja tehniliselt veaohtlik protsess. Magistritöös samuti kavatsetakse *no-code* tarkvaraarendusega luua modelleerimiskeel ning selle põhjal ehitada modelleerimistarkvara. Muuhulgas soovitakse saavutada olukord, kus arendatud tarkvara on veebipõhiselt hallatav ja lõppkasutajale kasutamiseks kättesaadav. Teisisõnu luuakse tarkvara teenusena (SaaS – software as a service) tüüpi rakendus. See võib tarkvara loomise ja täiendamise potentsiaalsele lihtsustamisele kaasa aidata. Käesolev uurimus

võib esmajoones huvi pakkuda lugejale, kes on kokku puutunud Eclipse Sirius modelleerimistarkvaraga ja kes on varasemalt mudelipõhisel arhitektuuril põhineva tarkvaraarendusega tegelenud. Lõputöö vaatleb veebipõhise modelleerimistarkvara arendamist toetudes mudelipõhise arenduse metoodikale ja arhitektuurile, seega võib töö pakkuda huvi ka inimesele, kes alles soovib ennast mudelipõhisel arhitektuuril põhineva arendusega kurssi viia.

1.2 Ülesande püstitus

Varasemalt on nii minu kui ka mitme teise tudengi bakalaureuse töös uuritud Eclipse Sirius Desktop arenduskeskkonda, Yana Sirotkina lõputöö „Ärianalüüsi toetava valdkonnaspetsiifilise modelleerimiskeele prototüübi loomine“, Aleksandr Bakalkini lõputöö „Äriobjektide modelleerimist toetava valdkonnaspetsiifilise modelleerimiskeele prototüübi loomine“. [2], [3] Praeguseks hetkeks on Eclipse Sirius Desktopist loodud veebipõhine versioon - Eclipse Sirius Web - mis lubab lõppkasutajale hõlpsamat tarkvara arendamist ning arendatud tarkvarale paremat ligipääsu. [4] Lõputöö eesmärk on uurida Sirius Web raamistiku ja Obeo Cloud platvormi tehnoloogilisi võimalusi tarkvara arendamiseks. Ootus teada saada, kas nimetatud pilvepõhist arendusplatvormi kasutades on võimalik saavutada mudelipõhisel ja *no-code* põhimõtetel toetuva tarkvaraarendusega samad eesmärgid, mis Sirius Desktop rakenduses ehk lokaalses arenduskeskkonnas. Töös tuvastatakse ja võrreldakse Sirius Web ja Sirius Desktop omadusi, mida arvestatakse modelleerimistarkvara loomise käigus. Lisaks on töö eesmärk võrrelda rakenduse loomise keerukust veebipõhise ja lokaalse arenduskeskkonna vahel. Veebipõhise töövahendi loomisel on võimalik saadud tulemust võrrelda Sirius Desktop lahendusega. Sirius Desktopi lahendus on bakalaureuse töös valideeritud Bizagi Modeler tarkvara vastu ning see annab piisava aluse Sirius Web vahendit valideerida Sirius Desktopi töövahendi vastu. Praktilise töö tulemus võimaldab lõputöö tulemust valideerida ja teha vajalike järeldusi ning anda hinnang Sirius Web platvormile. Lõputöö eesmärkide täitmisel soovitakse leida vastus järgmistele küsimustele:

- Kas Sirius Web veebikeskkonnas on võimalik täiesti algusest arendada nõuetele vastav äriprotsesside modelleerimistarkvara, kasutades *no-code* ja mudelipõhist arendusmetoodikat?

- Millisel määral on võimalik Sirius Desktop arendatud lahendust taaskasutada veebipõhises arenduskeskkonnas ehk Sirius Web'is?
- Milline erinevus on kahe tehnoloogia kasutamise keerukuses?

Lisaks akadeemilisele väärtusele annab lõputöö tulemus aluse anda konstruktiivset tagasisidet ka Eclipse ettevõttele, kes on Sirius Web projekti edasiviija.

1.3 Eesmärk

Magistritöö peamine eesmärk on kavandada eelmises alapeatükis nimetatud bakalaureuse töödes loodud metamudelitest üks ühine metamudel ja realiseerida selle alusel modelleerimise töövahend. Metamudelisse kaasatakse äriprotsesside modelleerimise komponent, väärtusvahetuste ja äritransaktsioonide komponent ning viimaseks äriobjektide modelleerimise komponent. Teisisõnu, eesmärk on Eclipse Sirius Desktopis loodud metamudelid üle viia Sirius Web platvormile. Kolmest komponendist koosnevast metamudelist tulenevalt on üheks alameesmärgiks äriprotsesside modelleerimise komponendi täielik üleviimine Sirius Web platvormile, mis tähendab, et metamudelile realiseeritakse ka vastavalt toimiv äriprotsesside modelleerimise töövahend. Äriprotsesside komponendi täielik üleviimine uuele platvormile annab aluse võrrelda omavahel Sirius Desktop ning Sirius Web tehnoloogiaid. Lisaks saab kaardistada veebipõhise tarkvara loomise metoodikat lähtudes Sirius Web platvormi praegu kehtiva versiooni võimalustest. Kuna metamudelisse on kaasatud lisaks äriprotsessidele ka äriobjektide ja äritransaktsioonide komponendid, siis tulevikku vaadates on eesmärk tagada läbi metamudeli ka nende kahe komponendi funktsionaalsuse loomise eeldus.

1.4 Ülevaade tööst

Magistritöö koosneb kuuest peatükist. Esimene peatükk juhatab töö sisse, tutvustab, probleemi ning selle tausta, püstitatud eesmärki ning kuidas läbi probleemi lahendamise kavatsetakse eesmärk saavutada.

Teine peatükk kirjeldab töö teoreetilist osa, mis toetab töö praktilist poolt ning selle teostamist. Esmalt kirjeldatakse äriprotsesside ning valdkonnaspetsiifiliste

modelleerimiskeelte olemust. Seejärel tutvustatakse mudelipõhise arhitektuuri põhimõtteid ning ökonoomse valdkonnaspetsiifilise keele loomise põhiideid.

Kolmas peatükk kirjeldab töös kasutatava tarkvara teenuse liiki ning no-code tarkvaraarendamise metoodikat. Mõlemad viitavad töös kasutatavale tehnoloogiale ja metoodikale.

Neljas peatükk kirjeldab töös kasutatud tarkvara ning võrdleb omavahel Eclipse Sirius Desktop ja Eclipse Sirius Web tehnoloogiaid. Tehnoloogiate juures tutvustatakse ka organisatsiooni Eclipse Foundation ning EMF ehk Eclipse modelleerimise raamistikku.

Viies peatükk kirjeldab töös teostatud praktilist poolt. Tutvustatakse loodud valdkonnamudelit, millest lähtuvalt kirjeldati modelleerimise vahend äriprotsesside kaardistamiseks. Kirjeldatakse ka kõikide elementide loomist, mida on modelleerimise töövahendi realiseerimiseks tarvis. Lisaks tehakse läbi ka protsessi kirjeldamine, selle abil saab loodud lahenduse võimekust katsetada.

Kuues peatükk analüüsib mudelipõhisel arhitektuuril ja *no-code* arendusmetoodikal põhinevat lahendust. Lisaks arendatud lahendusele analüüsitakse arendusmetoodikat ning Sirius Web platvormi. Tuuakse välja ka aspektid, mida oleks saanud töökäigus paremini teha. Pakutakse välja ka suunad, mida töö tulemustest lähtudes edasi uurida.

Lõpetuseks kokkuvõte, mis kirjeldab lühidalt kirjutatud tööd ning selle tulemust.

2 Modelleerimine ja mudelipõhine arhitektuur

Järgnev peatükk kirjeldab äriprotsesside modelleerimist, selle olemust ja kasulikkust. Seejärel tutvustatakse mudelipõhist arhitektuuri ja sellega seotud mõisteid ning kuidas nendele tuginedes on võimalik luua modelleerimist võimaldav tarkvara.

2.1 Äriprotsessid ja valdkonnaspetsiifilised keeled

Käesolev alapeatükk tutvustab põgusalt bakalaureuse töös käsitletud teooriat, mis juhatab sisse magistritöö teoreetilise sisu. Esmalt tuuakse esile äriprotsesside mõiste, lisaks tutvustatakse mudelipõhise arhitektuuri põhitõdesid ning selle baasil kirjeldatakse ka valdkonnaspetsiifilise modelleerimise keeli ja nende loomist.

2.1.1 Äriprotsessi modelleerimine

Äriprotsess sisaldab endas sündmuseid, tegevusi ja tegutsejaid, mis eksisteerivad ettevõttes või asutustes. Protsessi struktuur kirjeldab sündmuste ja tegevuste loogilise järjekorra ning nendevahelised sõltuvused, mille eesmärk on saavutada püstitatud eesmärk. Ettevõttele on protsess justkui standard, mille alusel oma tegevusi parimal ja efektiivseimal viisil sooritada, mis peab tootma selle käitajale ja seotud osapooltele väärtust. [1], [5]

Äriprotsesside modelleerimine on organisatsioonile vajalik, sest läbi modelleerimise on võimalik erinevate osapoolte jaoks protsess graafiliselt esitada. Protsessi visualiseerimine annab hea aluse ettevõtet ja tema tegevusi analüüsida ning realiseerida. Modelleerimiseks on välja mõeldud standard - äriprotsesside modelleerimise notatsioon ehk BPMN. Modelleerimise üks eesmärk on kaardistada organisatsioonis hetkel kehtivad ja toimivad protsessid ehk AS-IS protsessid. See annab aluse protsesside kitsaskohtade avastamiseks, analüüsimiseks ning läbi selle protsessi enda optimeerimiseks. Analüüsitud ja arendatud äriprotsessi kuju nimetatakse TO-BE protsessiks. Kui protsessi AS-IS kujule on kõrvutatud TO-BE kuju, siis see lubab kahte varianti omavahel võrrelda ning loob piisava teabe, et otsustada, kas arendatud TO-BE protsess on piisavalt efektiivne, et seda realiseerima hakata. Hea tulemus on kui protsessi on võimalik mõistlike ressurssidega efektiivsemaks muuta. [1], [6]

2.1.2 DSML ehk valdkonnaspetsiifiline modelleerimise keel

Modelleerimise keel on üks standardite kogum, mis määrab ära, kuidas on võimalik kindlaid asju modelleerida. Modelleerimise keele defineerib selle süntaks ning semantika - näiteks UML ehk ühtne modelleerimise keel võimaldab täpselt ja standardsel kujul visualiseerida arendatava infosüsteemi disaini.

UML on üldise eesmärgiga modelleerimise keel ehk GPML, mis tähendab, et seda on võimalik kasutada sõltumata valdkonnast. GPML-i eesmärk on täita mitme valdkonna vajadusi korraga. Üldise eesmärgiga keel on üldlevinud standarditega ning ei sisalda endas valdkonnaspetsiifilisi aspekte.

GPML-i vastand on DSML ehk valdkonnaspetsiifiline modelleerimise keel. DSML-i arendamiseks või kasutamiseks tekib vajadus siis, kui on tarvis üldisemaid modelleerimise kontsepte täiendada valdkonnaspetsiifiliste aspektidega. Täiendavad asjaolud lisatakse modelleerimiskeelele juurde vastavalt vaadeldava valdkonna nõuetele. Loodud valdkonnaspetsiifiline modelleerimise keel võimaldab modelleerijal kasutada domeeni ehk valdkonna kontsepte, ilma et peaks üldisema ja laiema kasutusalaga üldise keele kontsepte kasutama. DSML kasutamine optimeerib modelleerija tööd, sest saab keskenduda vaid oma valdkonna asjaoludele, ilma et peaks kulutama aega üldise modelleerimiskeele standardi sobitamisega enda valdkonnaspetsiifikaga. Näiteks käesolevas töös BPMN standardile vastav keel on GPML ehk üldise eesmärgiga modelleerimise keel, aga kui standardist võetakse vaid vajaminev ning kaardistatakse kitsamalt, siis see muutub valdkonnaspetsiifiliseks keeleks ehk DSML-iks. [7], [8]

2.2 MDA ehk mudelipõhine arhitektuur

Mudelipõhine arhitektuur ehk MDA on standard, mille on välja töötanud Object Management Group (OMG). Alapeatükk tutvustab põhjendusi, miks MDA arendusmetoodika võib arendajale huvi pakkuda. Seejuures vaadeldakse MDA eeliseid ja puuduseid, mida seda metoodikat järgides tuleks arvestada.

2.2.1 Mudelipõhine arhitektuur

OMG ehk Object Management Group on standardite organisatsioon, mis töötab välja standardeid, mille eesmärk on vähendada tarkvaraarenduse keerukust, kulusid ning teha uute tarkvaraarenduste juurutamist. Üks organisatsiooni initsatiive on on ka MDA ehk mudelipõhine arhitektuur, tegu on tarkvaraarendamise arhitektuurne raamistik. MDA toetub mitmetele teistele OMG poolt välja töödeldud standarditele, mis on saanud tarkvaraarendusvaldkonnas laia kasutust. Mõned muud OMG standardid on näiteks UML – *unified modeling language*, MOF – *meta model facility* ja EDOC – *enterprise distributed object computing*. [1], [9]

Mudelipõhise arhitektuuri põhiidee on katta kogu tarkvaraarenduse elutsükkel, alustades süsteemianalüüsist ja disainimisest, millele järgneb programmeerimine ja testimine ning lõpetuseks tarkvara komponentide kompileerimine, kasutuselevõtmine ning hooldamine. MDA iseenesest pole uus standardi spetsifikatsioon, pigem on tegu lähenemisega tarkvaraarendusele, mis kasutab praegu eksisteerivaid OMG standardeid ära. Teisisõnu on tegu standardiga, mis implementeerib juba olemasolevaid standardeid, et efektiivselt arendada ja jälgida arendusprotsessi. [1], [10]

2.2.2 MDA põhimõisted

MDA põhimõisted, millele raamistik tugineb on mitmeid ning nende lühikirjeldused on järgnevad:

- **Süsteem** – infosüsteem, kas juba olemasolev või loomisel süsteem, mida vaadeldakse.
- **Mudel** – formaalne spetsifikatsioon, mis kirjeldab süsteemi funktsionaalsust, struktuuri või käitumist kindlas kontekstis ja kindlast vaatepunktist vaadatuna.
- **Mudelipõhine metoodika** – tarkvaraarendamise viis, mille dokumentatsiooni, analüüsi, disaini, ehitamist, evitamist ja hooldamist on kirjeldatud mudelite abil.
- **Arhitektuur** – kirjeldab süsteemi komponente, nendevahelisi seosed ning seoste reegleid. MDA kontekstis kirjeldatakse tarkvara arhitektuuri läbi omavahel seotud mudelite.

- **Vaatepunkt** – abstraheerimise tehnika, mille käigus keskendutakse vaid kindlale süsteemi osale, seejuures ei vaadelda väheolulisi detaile. Vaatepuntki kirjeldatakse läbi ühe mudeli, mis kirjeldab süsteemi vaadeldavat osa abstraktselt.
- **Platvorm** – kogum alamsüsteemidest ja tehnoloogiatest, see pakub ühtse ülevaate funktsionaalsusest läbi liidesta ja kasutusmustrite. Platvormi vaadeldes saab kasuliku teadmise, ilma et peaks keskenduma selle implementeerimisele. Platvorm võib olla operatsioonisüsteem, programmeerimiskeel, andmebaas, või ka kasutajaliides.
- **Platvormist sõltumatus** – sõltumatus on omadus, mida mudel omab, et selle omadusi saab kirjeldada, ilma et peaks platvormi aspektidest sõltuma. Sõltumatus on ühtlasi ka suhteline indikaator, mis määrab kahe platvormivahelist abstraktsuse taset.
- **Platvormi mudel** – platvormi mudel kirjeldab tehniliste kontseptide kogumit, selles sisalduvaid elemente ja teenuseid. Lisaks kirjeldab mudel ka elementide ja teenuste kitsendusi, millega peavad teised süsteemi osad nende kasutamisel arvestama.
- **Mudeli transformatsioon** – transformatsioon on ühe mudeli teisendamine teiseks ühe süsteemi piires. Transformatsiooni eesmärk on platvormist sõltumatu mudeli teisendamine platvormi spetsiifiliseks mudeliks. Tegu on sammuga, mis tuleb enne mudeli implementatsiooni ehk realiseerimist teha.
- **Mudeli implementatsioon** – implementatsioon on tehniline kirjeldus, mis sisaldab endas kogu informatsiooni, mis on tarvis, et mudelitega kirjeldatud süsteemi ehitada ja käivitada. See peab andma kogu info, et luua mudelist objekt ja tagada sellele kogum teenuseid, millega objekt hakkab süsteemi teiste osadega suhtlema. [1], [9]

Mudelipõhisel arhitektuuril põhineval tarkvaraarenduse rakendamisel on vajalik luua platvormist sõltumatu mudel, mille kirjeldamiseks kasutatakse UML modelleerimise standardit. Seejärel on võimalik koostatud platvormist sõltumatut mudelit kohandada spetsiifilisemale platvormile. Sellest on ka see eelis, et MDA ei seo tarkvara analüüsi tasemel arendajat ühegi platvormiga, vaid need otsused saab hilisemas tarkvaraarenduse faasis teha. [9], [11]

2.2.3 MDA eelised

Mudelipõhine arhitektuur annab lubaduse hõlbustada masinloetavate mudelite loomist eesmärgiga tagada pikaajaline paindlikkus tarkvara elutsükli jooksul. Täpsemalt lubatud aspektid lühikirjeldustega on järgmised:

- **Kaitse tehnoloogia vananemise korral** – uusi arendatud tarkvaralahendusi on võimalik lihtsamalt ja mugavamalt olemasolevale süsteemiga implementeerida.
- **Ühilduvus** – vastavalt ärilisele vajadusele on olemasolevat süsteemi funktsionaalsust võimalik kiiremini migreerida uutesse keskkondadesse ja platvormidele.
- **Produktiivsus ja kiire evitamine** – läbi mudelite on võimalik mitmeid tülikaid tarkvara uuendamise protsesse automatiseerida, siis läbi selle kasvab ka produktiivsus, kui arendajad saavad rohkem keskenduda süsteemi põhiloogika arendamisele.
- **Kvaliteet** – läbi mudelite ühtselt loodud tarkvara funktsioonid ja arhitektuur tagavad süsteemi kvaliteedi ja üheseltmõistetavuse.
- Integreerimine – uute lahenduste integreerimine *legacy* või väliste süsteemidega on suurelt toetatud.
- **Hooldamine** – infosüsteemi masinloetav disain ja spetsifikatsioon on kättesaadav ja arusaadav nii analüütikutele, arendajatele kui ka testijatele, mis hajutab tarkvara hoolduse tööd kõikide rollide vahel ära. Tekitab soodsa olukorra, kus tarkvara lahendust ja dokumentatsiooni on lihtsam kaasaegsena hoida.
- **Testimine** – loodud mudeleid saab otse valideerida süsteeminõuete vastu. Mudeleid saab testida erinevate komponentide vastu ehk saab teha integratsiooni testimist. Lisaks saab mudeleid kasutades testida süsteemi oletatavat toimimist ka juba selle disainimise faasis.
- **Parem investeeringu tasuvus** - mudelipõhise arhitektuuri eelised tagavad efektiivsema tarkvaraarenduse protsessi, mis omakorda suurendab arendusse paigutatud investeeringu väärtust.

Esile toodud lubatud eelised tekitavad piisava motivatsiooni, et rakendada tarkvaraarenduses mudelipõhist arhitektuuri raamistikku. Mitmed ettevõtted on selle

kasumlikkust täheldanud, kuid ärisaladuse tõttu pole selle implementatsioonist väga detailselt avalikult kõnelenud. [1], [9]

2.3 Ökonoomne valdkonnaspetsiifiliste keelte loomine

Käesoleva peatükk kirjedab meetodinseneeriat ja selle rakendamise põhimõtteid. Tutvustatakse ka komponendipõhise keele ehitamise tausta ning kuidas seda realiseeritakse meetodinseneeria põhimõtetele toetudes.

2.3.1 Meetodinseneeria põhideed

Meetodinseneeria on valdkond, mille esmane eesmärk on süstemaatiliste meetodite koostamine tarkvaraartefaktide väljatöötamiseks. Tarkvaraartefaktide alla kuuluvad näiteks nõuded, disainidokumendid ning kasutusjuhud, klassidiagrammid ja lisaks muud UML mudelid. Kuid ajajooksul on meetodinseneeria potentsiaalsed rakendusviisid laienenud. Lähenemisviisina saab meetodinseneeriat rakendada ka muudes valdkondades, mis nõuavad keerulise lahenduse loomist – näiteks infosüsteemide haldamisega seotud meetodite ja reeglite loomine. Situatsiooniline meetodinseneeria rõhutab meetodinseneeria kasulikkust sellega, et lisaks üldisele disainile toetab ka mehhanismi, mis üldise disaini täiendamisel arvestab ka situatsioonist ehk olukorrast lähtuvat konteksti. Üldise disaini omadusi täiendatakse või konfigureeritakse selliselt, et loodav lahendus on sihtprobleemile keskendunud, rõhudes situatsiooni põhisele kontekstile ja eesmärkidele. Situatsiooniline lähenemise sammud probleemi lahendamiseks on järgmised kuus:

1. Tuletatakse asjakohased disainitegurid teadaolevatest praktikatest;
2. Võttes arvesse eelmises punktis tuletatud tegureid, täpsustatakse olemasolevate lahenduste ehk arhetüüpide rühmitamise teel lahendusmustrid;
3. Vastavalt disainieesmärkidele võrreldakse olemasolevaid (*as-is*) lahendusi vajaminevate (*to-be*) lahendustega. Võrdluseks tuletatakse metoodika arendussuund.
4. Võrreldes arendussuundi ja praeguste arhetüüpide võimekuse või tegevuse komponente, tuvastatakse süstemaatiliselt erinevused, mis omakorda moodustavad uue disainimustri.

5. Asjakohaste disainimustrite jaoks määratakse ühised võimekuse või tegevuse komponendid, mis omakorda moodustavad meetodimooduleid.
6. Arvestades viiendas punktis tuvastatud meetodimooduleid saab igat vajalikku täiendust esitada kokkuvõtva kompositsioonina. Kompositsioonireeglite kogum moodustab mehhanismi, kus meetod kohaneb situatsiooniga. [12]

Sellist viisi järgides on võimalik ka modelleerimiskeel luua, vaadeldakse situatsiooni ehk ülesannet ja selle täitmiseks kombineeritakse juba teadaolevatest keelte fragmentidest uus keel selliselt, et midagi vajalikku pole puudu ega midagi mitte-vajalikku ülearu.

2.3.2 Ökonoomse ja komponendipõhise keele ehitamine

Komponendipõhine lähenemine näeb kontseptuaalsete mudelite loomisel ette, et mudelid lähtuvad ettevõtte transformatsiooni kontekstipõhistest omadustest. Selle vastandid on „üks-suurus-sobib-kõigile“ keeled, näiteks ettevõttearhitektuuri modelleerimise keel ArchiMate. Piirangud mida sellised üldise funktsionaalsusega keeled endast kujutavad on peamiselt seotud valdkonnaspetsiifiliste aspektide kaardistamisega. Näitena tuues, siis *ArchiMate*’ga pole võimalik ettevõtte transformeerimise modelleerimisel väljendada majanduslike põhjenduste sidumist arhitektuurse disainiga, ettevõtteüleseid turvariskide analüüsi või ärimudeliga seotud murekohtasid. Lühidalt ArchiMate kui „üks-suurus-sobib-kõigile“ keelel puudub ettevõtte transformeerimise modelleerimisel väljendusviis konkreetsest ettevõttest sõltuvaid võimalike kontekstipõhiste asjaolude kaardistamiseks. [13]

Selle puudujäägi lahendamiseks on välja pakutud ArchiMate metamudeli laiendamist valdkonnapõhiste aspektidega. Eraldi vaadelduna on iga valdkonnapõhine aspekt ArchiMate metamudeli laiendus, mis esmalt peab ühilduma nii esialgse metamudeliga kui ka teiste juba loodud laiendustega. Selliselt võib tekkida olukord, kus ArchiMate kontseptuaalne säästlikkus on rikutud. Kontseptuaalne säästlikkus on ArchiMate peamine disainiprintsiip, et kontseptide modelleerimise keel peab disainilt olema ökonoomne. Kontseptuaalne ökonoomsus on probleemiks näiteks UML ehk ühtsel modelleerimise keelel, millel on väga palju erinevaid reegleid ja kasutuspõhimõtteid. Seni kaua kuni kasutatakse „üks-suurus-sobib-kõigile“ filosoofiat, siis selle laiendamine

valdkonnapõhiste kontseptidega viib tahes-tahtmata ökonoomse keele disaini probleemideni. [13]

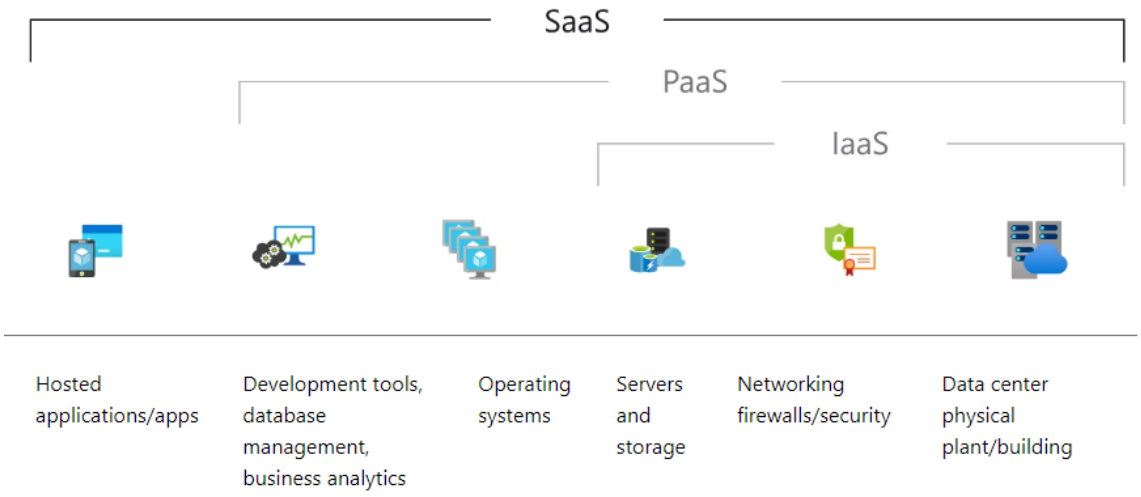
Ökonoomse keele disainimisel tuleb esmalt kaardistada funktsioon, mida see peab kandma – teisisõnu tuleb määrata valdkond ja selle spetsiifika, millega disainitav keel peab kohanema. Keel disaini ei pea nullist leiutama, vaid saab ära kasutada olemasolevaid keeli ja nende funktsioone. Olemasolevate keelte taaskasutamisel saavutatakse ökonoomsus vaid selliselt, kui valitakse funktsioonid, mida on konkreetses valdkonnas täpselt vaja. Näiteks kui BPMN standardis sisaldub äriprotsessi lüüs, mille alamliik on välistav otsuse lüüs (*XOR gateway*) aga konkreetses valdkonnas seda pole vaja kasutada, siis uue keele loomisel seda funktsionaalsust ei kaasata. Teisalt saab loodavasse keelde kaasata komponente, mida üldine standard ette ei näe, kuid valdkonna erisuse tõttu on see oluline lisafunktsionaalsus. Selliselt laiendatakse keelt ja selle kasutamist vastavalt valdkonna nõuetele. Juhul kui on loodavasse modelleerimiskeelde on kaasatud omadusi, mida valdkonnast lähtuvalt pole tarvis, siis pole tegu enam ökonoomse ja komponendipõhise keele arendamisega. [14]

3 Tarkvara teenusena ja pilvepõhine arendamine

Peatükk tutvustab lugejale tarkvara teenusena rakenduse tüübi olemust ja sellega kaasnevaid potentsiaalseid kasutegureid ning puuduseid. Lisaks SaaS *(software-as-a-service*) tüüpi rakendustele tutvustatakse ka *no-code* tarkvaraarenduse metoodikat.

3.1 Tarkvara teenusena

SaaS on tarkvara, mida majutatakse pilvepõhises keskkonnas ning selle kasutamiseks piisab üldjuhul veebibrauserist, mobiilirakendusest või väga väikese funktsionaalsusega kliendipoolsest tarkvarast. Tuntumad näited SaaS tarkvarast on meilinduse tarkvara Outlook ja Microsoft Office veebipõhised kontoritarkvarad, mis on koondatud Microsoft Office 365 nime alla. Üks tuntumaid SaaS tarkvara on ka ettevõtte ressursiplaneerimise programm SAP ja selle erinevad moodulid.



Joonis 1. Erinevused SaaS, PaaS ja IaaS tarkvarade vahel [15]

SaaS eesmärk on kliendile pakkuda terviklikku lahendust, mis tagab temale kõik vajaliku tarkvara kasutamiseks, see tähendab, et SaaS tarkvara pakkuja vastutab tarkvara funktsionaalsuse, hoolduse ja infrastruktuuri eest. Seejuures hoiab kliendil teenusega seotud kulud madalate või ette ennustatavatena. SaaS teenuse kasutamiseks tuleb kliendil endale vaid kasutaja, vajadusel tasuda pakkuja poolt määratud tasu ning seejärel saab tööle asuda. [16]

SaaS teenuse rakendused nõuavad minimaalselt või mitte mingit kliendipoolset tarkvara haldamist või hooldamist. *SaaS* teenuse pakkuja vastutab kliendi ees järgneva eest:

- Serverite, võrguseadmete, andmetalletus riistvara ja operatsioonisüsteemi haldamine ning hooldamine.
- Tarkvara funktsionaalsuse toimimine ja turvalisuse täiendamine;
- Tarkvara poolt tekitava koormuse vastuvõtmise võimekuse, infrastruktuuri jaotamise, andmete varundamise ja SLA nõuetele vastamine.

SaaS teenus kasutab ära eeliseid, mida pakub pilvepõhine andmetöötluse infrastruktuur. Pilvepõhine infrastruktuur on paindlik lahendus kliendile pakkuda kasutajale võimalikult lihtne ligipääs tarkvarale. SaaS rakendused on loodud selliselt, et neid majutatakse vaid pilvepõhisel riistvaral. SaaS tarkvara pakkuja võib hoida tarkvara oma enda poolt ehitatud infrastruktuuril või kasutada pilvepõhise teenuse pakkujat – näiteks Amazon Web Services, Google Cloud, IBM Cloud või Microsoft Azure. Tuntud ja globaalse pilvepõhise teenusepakkuja kasutamine loob eelduse, et tarkvara pakkujal on võimalus oma rakendust ka globaalselt turundada. [15], [16]

SaaS rakendust pakutakse klientidele selliselt, et igale kliendile on rakendusest oma instants. See tähendab, et ühe kliendi rakenduse, kasutajate ja süsteemi andmed ning süsteemi konfiguratsioonid on teiste klientide andmetest eraldatud. Taoline lähenemine maandavad rakenduste mitte-toimimise ja andmete lekkimise riske. Näiteks kui ühel kliendil on pahatahtlik kasutaja, kes rikub mõningaid andmeid või kasutab ära potentisaalset turvaauku, siis see probleem on isoleeritud kujul vaid sellel ühel kliendil, mitte kõikidel teistel, kes samuti kasutavad kõnealust teenust. [16]

SaaS teenuse poolt pakutavad peamised kasutegurid:

- **SaaS võimaldab kohest tarkvara kasutusele võtmist** – klient saab peale teenuse tellimist sisuliselt kohe tarkvara kasutusele võtta ja tööga seotud eesmärki täita. Puudub vajadus tarkvara installimiseks lõpp-kasutaja töövahendisse.
- **SaaS tagab ligipääsu uutele arendatud lahendustele nii pea kui need on kättesaadavaks tehtud** – rakenduse uuendused saabuvad kasutajateni ilma, et

nad nende kasutamiseks midagi tegema peavad. Kõik tehtavad uuendused, olgu need nii suured või väikesed kui tahes realiseeritakse ilma kasutaja tööd häirimata.

- **SaaS tagab kuluefektiivse skaleeritavuse** – vastavalt vajadusele saavad teenuse kliendid kerge vaevaga tarkvara võimekust lisada või vähendada. Kui tekib ülekoormus, siis saab juurde tellida riistvaralist võimekust ning kui tarkvara kasutajate hulk väheneb, siis saab vastupidiselt võimekust liiasuse võrra väheneda, vastavalt sellele muutub ka teenuse hind.
- **SaaS tagab ennustatavad püsikulud** – SaaS teenuse puhul ei pea arvestama infrastruktuuri kuludega ega ka n-ö majasisese IT meeskonnaga, kes paigaldaks, uuendaks ja hooldaks kasutatavat tarkvara. Kõik infrastruktuuri ja tarkvara arendusega seotud kulud on üks-ühele seoses tarkvara ehk teenuse kasutamisega.

SaaS-ist kõneldes ei saa mööda vaadata teistest pilvepõhise teenuse pakkumise mudelitest, lisaks SaaS on olemas ka PaaS ja IaaS, mis vastavalt vähendavad teenuse pakkujalt vastutust ja suunavad selle kliendile. [15], [16]

PaaS (*platform-as-a-service*) on platvorm teenusena, mis tähendab, et kliendile pakutakse täielikku pilvepõhist platvormi. Platvormi eest vastutab täielikult teenuse pakkuja, kes peab pilvepõhiselt tagama riistvara, tarkvara, arendustööriistad ja infrastruktuuri, mis pakub kliendile võimaluse tarkvaraarendamiseks ja arendatud tarkvara kasutamiseks. PaaS tagab tarkvaraarendus tiimile keskkonna, kus saab ehitada, testida, avalikustada, uuendada ja skaleerida rakendusi kiiremini ja odavamalt kui tarkvaraarendus tiim peaks ise teenuse poolt pakutud platvormi üleval hoidma. [16]

IaaS (*infrastructure-as-a-service*) on infrastruktuur teenusena, see tagab kliendile teenuse pilvepõhise võrgu ja andmetöötlus- ja salvestamise ressursi. IaaS teenus on sobilik kliendile, kes soovib kontrolli oma rakendustele ja platvormile, kuid soovivad kuluefektiivselt infrastruktuuri ressursse vastavalt vajadusele suurendada või vähendada. Selle asemel, et ehitada ja hallata enda kohapeal olevat n-ö andmekeskust kasutab klient hoopis teenusena pakutavat infrastruktuuri. Selline teenus hoiab kliendi kulud kontrolli all ja olukordades, kus klient peab riistvaralist võimekust suurendama või vähendama, on oluliselt kuluefektiivsemad, kui omal kohapeal infrastruktuuri võimekust muuta. [16]

3.2 No-code tarkvaraarendamise metoodika

Kindlatele nõuetele vastava tarkvara arendamine eeldab tarkvaraarendajalt häid tehnilisi oskuseid ning tugevat kogemuste pagasit. Lisaks nõuab tarkvaraarendus üldjuhul ka arenduskeskkonna konfigureerimist ja seejuures erinevate tarkvara elementide ja komponentide paigaldamist arendaja arvutisse. Teisisõnu on vaja teha terve rida tegevusi enne kui arendajal on võimalik tarkvaraarendusega alustada. [17]

Selleks, et lahendada olukord, kus tarkvaraarenduseks on vaja teha palju eeltööd ja keerukaid tegevusi läbiviia enne kui sisulise tegevuseni jõuaks on välja mõeldud mitmeid platvorme, mille abil saab *no-code* metoodikat kasutades tarkvara arendada. *No-code* arendusplatvormid on üldjuhul SaaS tüüpi teenused, kus kliendile on pilvepõhiselt tagatud kõik vajalikud tugipunktid tarkvara loomiseks, jooksutamiseks ja lõppkasutajale väljaandmiseks. *No-code* tarkvaraarendamise metoodika näeb ette, et tarkvara loomisel ei pea koodi kirjutama, vaid seda saab teha läbi kasutajasõbraliku graafilise kasutajaliidese. Selliselt on loodud soodsad võimalused tarkvara arendamiseks ka nendele, kes pole programmeerimise kui koodikirjutamisega kokku puutunud. [18]

No-code arendamine sai alguse veebilehtede vormide koostamisest, kuid tänasel päeval on selle kasutamine oluliselt laiema kasutatavusega. Erinevad *no-code* arendusplatvormid lubavad juba luua ka mobiili-, veebi- ja häälkäskluse rakendusi, lisaks on arendatud *no-code* abil ka integratsioone ja tegevuste automatiseerimisi. Oskamata kirjutada ainsatki koodirida on võimalik luua näiteks Voiceflow platvormiga kõneroboteid, Shopify platvormiga e-poode ning Zapier abil ehitada automatiseeritud tööprotsesse. Need on vaid vähesed näited *no-code* plavormidest, millega on võimalik toimivaid lahendusi luua. [19], [20], [21]

Kuna *no-code* arendusmetoodika ei eelda suurt tehnilist tausta, siis on see rohkematele inimestele atratkiivne võimalus tarkvara arendamiseks. Isegi mitte-programmeerijad saavad täita programmeerija ülesandeid. Selliselt luuakse olukord, et näiteks ka ärianalüütik saab anda tarkvara funktsionaalsuse realiseerimise sisendi ilma, et peaks keerukaid vanemarendaja ülesandeid täitma. Selle asemel saab koheselt proovida ja katsetada tarkvara toimimist. Mida varem saavutatakse arusaam, et arendatud tarkvara

toimib vastavalt nõuetele, seda varem on võimalik see lõppkasutajale kättesaadavaks teha. [18], [22]

No-code arendusmetoodikal on palju positiivseid omadusi, kuid sellegipoolest ei puudu sellel negatiivsed küljed. *No-code* arendusmetoodikat praktiseerides peab kasutaja valima endale kindla platvormi, mis sellist metoodikat toetab. Platvorm, mis lubab *no-code* metoodikaga tarkvara arendada, on tihti peale ühe valdkonnaspetsiifiline, mis viitab, et arendatava rakenduse funktsionaalsus on piiratud arendusplatvormi poolt pakutud võimalustega. Kui kasutaja soovib midagi enamat arendada kui platvorm lubab, siis tuleb tal see funktsionaalsus ise kirjutada või vahetada platvormi, mis soovitud funktsionaalsuse arendamist toetab. *No-code* eesmärk arendada tarkvara lahendusi läbi graafiliste elementide on küll kasutajasõbralik, samas limiteerib kontrolli enda arendatud tarkvara üle. Koodi kirjutamisel on arendaja täpselt teadlik oma programmist, sest tema on selle kirjutanud ja kompileerinud. Kasutajal on *no-code* platvormil arendatu kohta vaid seda informatsiooni, mida talle läbi graafilise kasutajaliidese pakutakse. Piiratud kontrolli tõttu võib tekkida olukord, kus programmi funktsionaaluses tekib viga, mille tuvastamine ja parandamine pole võimalik. Arendaja piiratud kontroll oma arendatud rakenduse üle avab soodsa võimaluse pahatahtlikel inimestel arendusplatvorm katki teha ja seeläbi lõhkuda ka selle abil arendatud rakendused. Võrreldes tavalise rakendusega on see negatiivne külg, sest *no-code* rakenduse puhul ligipääs detailsele programmikoodile puudub ning seetõttu ei saa arendaja lähtekoodist sõltuvaid potentsiaalseid turvariske maandada. [22]

4 Töös kasutatavad tehnoloogilised lahendused

Käesolev peatükk kirjeldab tehnoloogilisi lahendusi, mida antud töös uuritakse ja seatud eesmärkide saavutamiseks kasutatakse. Praktilise teostuse ja analüüsi tegemiseks kasutati Eclipse Foundation'i poolt arendatud tarkvarasid ja raamistikke, mis keskenduvad graafilise valdkonnaspetsiifiliste modelleerimiskeelte defineerimisele ja nende põhjal tööriistade loomisele.

4.1 Eclipse Foundation

Eclipse Project on 2001 aastal IBM poolt algatatud organisatsioon mida toetasid tarkvaramüügi suurettevõtted. Aastal 2004 kasvas Eclipse Projectist välja Eclipse Foundation, mille eesmärk on hallata Eclipse tarkvara kasutajate ja arendajate kommuuni. Eclipse Foundation on Euroopas baseeruv rahvusvaheline mittetulunduslik organisatsioon, millesse kuulub üle kolmesaja liikme. Liikmete hulka kuuluvad infotehnoloogia haru liidrid, kelle jaoks on avatud lähtekoodiga lahendused peamised äristrateegia võimaldajad. Suure liikmete arvu tõttu on Eclipse Foundationi eesmärk hoida Eclipse kommuuni neutraalsust, avatust ning läbipaistvust. See on tagatud sellega, et Ecipse pakub vabavaralist tarkvara, mida arendavad vabatahtlikkuse korras erinevad arendajad erinevatest ettevõtetest. Vabavaralise tarkvara lähtekoodi ligipääsemiseks pakub Eclipse arendajatele litsentsi 'Eclipse Public License' – litsentsi on tarvis, et arendaja saab tarkvara kasutada, arendada ja arendatud osa ka teiste kommuni liikmetele jagada. Eelduseks ongi see, et arendatud tarkvara ei tohi enda teada jätta, vaid tuleb see ka teistele kättesaadavaks teha. Üks tuntumaid Eclipse projekte on Eclipse IDE ehk Eclipse integreeritud arenduskeskkond, mis on peaasjalikult tuntud kui Java arenduskeele keskkond, kuid lisaks sellele pakutakse ka muude programmeerimise keelte tarvis arenduskeskkondasid. Käesolevas töös on kasutusel kaks Eclipse tehnoloogiat – Eclipse Sirius ja see toetub EMF ehk Eclipse modelleerimise raamistikule. [23], [24]

4.2 EMF ehk Eclipse modelleerimise raamistik

Eclipse Modeling Framework ehk Eclipse modelleerimise raamistik on Eclipse Foundationi poolt loodud Eclipse Modeling Projecti alla kuuluv lahendus. Tegu on koodi

genereerimise raamistikuga, mis võimaldab struktureeritud andmemudelil põhinevaid rakenduse ja töövahendite ehitamist. Koostatud mudel on masinloetav XMI standardis, mille abil EMF tööriistad ja runtime keskkonna, et luua mudelist lähtuvalt Java klassid ja klassidele adapterite klassid. Adapterklassid võimaldavad vaadelda mudelit ning teha selles vajalikke täiendusi. EMF on üldlevinud andmemudelite standard, sellel põhinevad mitmed tehnoloogiad ja raamistikud. [25]

EMF koosneb koolmest põhiosast:

- **EMF tuum** – EMF-i tuum hõmab endas metamudelit, mille nimetus on Ecore. Ecore abil kirjeldatakse mudeleid, mille kasutamine on teotatud *runtime* keskkonnas. Metamudeli jooksev efektiivne muutmine ja täiendamine on toetatud.
- **EMF.Edit** – EMF.Edit on raamistik, mis sisaldab endas üldiseid ja taaskasutatavaid klasse, klasside abil on võimalik ehitada EMF mudelite redaktoreid.
- **EMF.Codegen** – EMF.Codegen lahenduse võimekus on genereerida kõik vajalik, mida on tarvis EMF mudeli redaktori loomiseks. Genereerimisel luuakse ka graafiline kasutajaliides, mida saab arendaja vastavalt avajadusele muuta. Genereerimise lahendus kasutab JDT (*Java Development Tooling*) komponenti, mis on üks Eclipse Project poolt arendatud ja hallatav töövahend. [1], [23], [25]
- Mudelitel põhineva koodigenereerimine on kolmetasandiline:
- **Mudeli tase** – tagab metamudelis kirjeldatud mudelite Java liidesed ja implementeerimise klassid.
- **Adapteri tase** – genereerib implementeerimise klassid, mis kohandavad metamudeli klassidega. Implementeerimise klassidega on võimalik metamudelit kuvada ja muuta.
- **Redaktori tase** – genereerib struktureeritud redaktori, mis vastab koostatud EMF mudelile ehk metamudelile. Antud tase on n-ö alguspunkti, millest alates saab metamudeli peale hakata looma nõuetele vastavat funktsionaalsust. [1], [25]

Eelnevalt kirjeldatud tasemetes on võimalik muudatusi teha, uuesti koodi genereerida ja seejuures säilitada varasemalt tehtud lahenduse funktsionaalsuse toimimise. Peamiselt tuleneb regenereerimise vajadus EMF tuumas tehtud muudatustest. Ühtlasi viitab see ka

MDA ehk mudelipõhise arhitektuuri eelisele, et kui on tarvis algtasemel muudatusi läbi viia, siis selle ühildamine senitehtud lahendusega on sujuv ja efektiivne. [9], [25]

4.3 Sirius

Sirius on Eclipse Foundation'i poolt hallatav projekt, mille abil on kasutajal võimalik oma nõuetele vastavat graafilise modelleerimise töölauda. Töölaua arendamisel kasutatakse Eclipse modelleerimise tehnoloogiaid, muu hulgas ka EMF ehk Eclipse modelleerimise raamistikku ja GMF ehk graafilise modelleerimise raamistikku. Siriuse projekti loojad ja peamised arendajad on ettevõtted Obeo ja Thales, kes on Siriuse näol loonud mudelipõhisel arhitektuuril põhineva töölaua, mida on võimalik kergesti arendades kohandada kasutaja nõuetele vastavaks. [26]

Modelleerimise töölaud, mida on arendatud kasutades Sirius koosneb mitmetest Eclipse redaktoritest (diagrammid, tabelid ja graafipuud) – need lubavad kasutajale luua, muuta ning visauliseeride EMF baseeruvaid mudeleid. Eclipse redaktor – näiteks diagramm – on defineeritud ühe mudeli abil, mis omakorda defineerib loodava modelleerimise töölaua struktuuri, seal hulgas funktsionaalsuse, muutmise ja navigeerimise tööriistad. Tehniliselt on Eclipse abil mudelipõhiselt arendatud tarkvara struktuur interpreteeritud Eclipse IDE *runtime* poolt. [26]

Järgnevalt on toodud näited ja kirjeldused modelleerimise töövahenditest, mis on Siriuse abil arendatud

- Capella on Thalese poolt arendatud modelleerimise tööriist, mis on mõeldud tarkvara ja riistvara arhitektuuride kirjeldamiseks. Funktsionaalsuse toetamiseks on loodud DSML ehk domeenispetsiifiline keel, mis rikastab UML ja SysML üldise eesmärgiga modelleerimiskeelte ehk GPML standardeid. [27]
- PLAT4MC on Robert Bosch'i poolt arendatud modelleerimise platvorm, mis toetab mitmetuumalise riistvara süsteemide disainimist. Platvorm lubab kasutajal visualiseerida kavandatava riistvara ülesehitust. [27]
- Mindstorms Designer on Obeo poolt arendatud modelleerimise vahend, mis lubab kasutajal kirjeldada Lego roboti tegevustejada diagrammi abil. Diagrammile

toodud elemendid kannavad taustal Java koodi, mis vastab roboti poolt toetatud funktsionaalsusele. Diagrammile kirjeldatud tegevuste järjekorra saab robotile täitmiseks laadida. [27]

Eelnevad näited annavad piisava kindluse, et Siriusega on võimalik mudelipõhiselt võimekaid tarkvarasid realiseerida.

4.3.1 Eclipse Sirius Desktop

Obeo Designer on Obeo poolt loodud tarkvara, mille on eesmärk on tagada lihtne ja kättesaadav viis valdkonnaspetsiifiliste graafiliste modelleerimistööriistade arendamiseks. Eclipse Sirius on Obeo Designer'i nii-öelda mootoriks, Designer'iga on integreeritud Eclipse Siriuse redaktorid. Redaktorite kogum on lahendus, mille abil saab luua vajaduspõhiselt kohandatud modelleerimise töövahendeid. Obeo Designer on Eclipse Siriusel põhinev töölaua ehk desktop rakendus ning sellega loodud modelleerimise vahendid on samuti töölaua põhised, siis sellest on ka nimetus Eclipse Sirius Desktop. [28], [29]
Obeo Designer kasutab eelnevates peatükkidest tuttavaid Eclipse Modeling Project'i komponente:

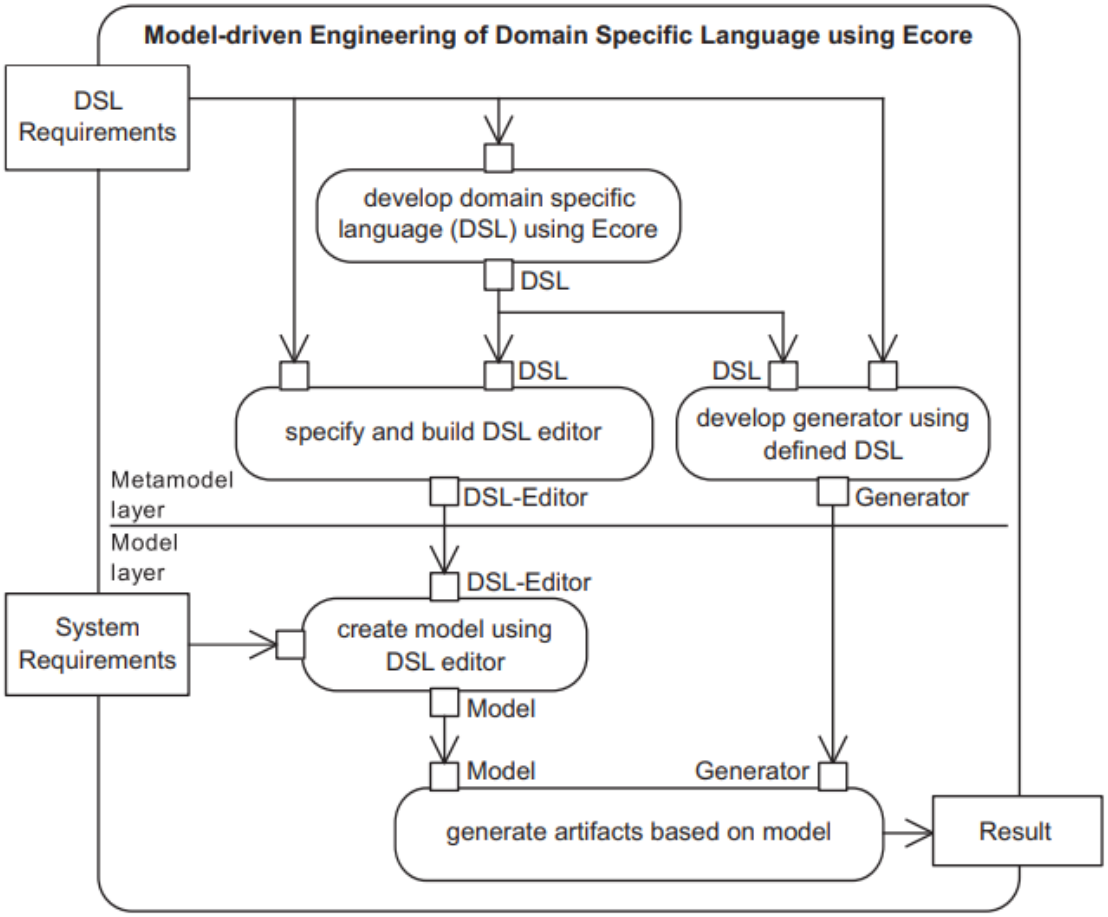
- **EMF** – struktureeritud andmemudelil baseeruv tarkvara ehitamise tööriist;
- **Sirius** – graafiline modelleerimise raamistik graafiliste redaktorite loomiseks;
- **Ecore Tools** – vahendid, millega defineerida domeenipõhist keelt (DSL – domain specific language). [29]

Lisaks on implementeeritud ka neid Eclipse Modeling Project'i komponente:

- **EMF Compare** – vahend EMF mudelite omavaheliseks võrdlemiseks ja tehtud muudatuste kokkuviimiseks.
- **Acceleo** – mallipõhise koodigenereerimise keel ja töövahendite komplekt;
- **EEF** – vahend tegemaks vormipõhiseid EMF mudelite töövahendeid;
- Xtext – EMF mudelite tekstilise muutmise töövahendite loomise lahendus;
- **CDO** – andmebaas, mis hoiab endas mudelite andmeid. Selle abil hallatakse ka mitme kasutaja koostöös tehtud mudeli muudatusi.

Kokkuvõtvalt on Obeo Designer vahend, mis on integreeritud omavahel seoses olevaid Eclipse Sirius komponente ja pakub nende komponentide efektiivseks kasutamiseks kasutajasõbralikku kasutajaliidest.

Joonisel 2 on välja toodud kõik arendamise sammud, mida tuleb Sirius Desktop'is teha, et EMF .ecore võimalustele tuginedes valmiks domeenispetsiifiline keel. [30]



Joonis 2. Valdkonnaspetsiifilise keele mudelipõhiselt arendamise sammud [31]

Eelneval joonisel kirjeldatud töövoog on läbi tehtud bakalaureuse töös, mille käigus loodi valdkonnaspetsiifiline keele metamudel, seejärel loodi metamudelil põhinev redaktor-Redaktor on sisuliselt tulem, ehk modelleerimisevahend, mis bakalaureuse töö kontekstis võimaldas BPMN notatsioonis äriprotsesside kaardistamist. [1] Järgmine peatükk kirjeldab aga Eclipse Sirius Web lahendust ning millised on selle vahendi omapärad ja modelleerimise vahendi arendamise tegevussammud.

4.3.2 Eclipse Sirius Web

Obeo Studio on Obeo poolt arendatud tehnoloogia, mis võimaldab luua veebipõhiseid modelleerimistöövahendeid. Tehnoloogia võimaldab töövahendeid luua vastavalt nõuetele vastavale valdkonnale – olgu see tarkvaraarenduse, süsteemitehnika, ettevõtte arhitektuuri, robootika või elektroonika valdkond. Seejuures on arendajal võimalik täiesti oma spetsiifikaga valdkond luua ja selle alusel töövahend arendada. Antud töö kontekstis nimetatakse seda tehnoloogiat ka Eclipse Sirius Web-ks ning nimetus sellest, et Obeo Studio toetub Eclipse Sirius Project'ile (EPL 2.0 litsents) ja Sirius Web komponendile. Sirius Web on veebipõhine versioon Siriusest, mida juurutatakse pilvepõhisel platvormil ja esitatakse veebibrauseris. Teisisõnu toetub samadele põhiideedele nagu Eclipse Sirius Desktop, et on võimalus läbi mudelipõhise arendusmetoodika arendada valdkonnaspetsiifiline graafiline modelleerimise keel ning selle alusel luua toimiv modelleerimise töövahend. Erinevus on selles, et Obeo Studio on veebipõhine rakendus ning seda majutatakse Obeo Cloud pilveteenuses. [4], [32]
Sirius Web põhiomadused:

- Sirius Web'iga defineeritud modelleerimise stuudio ehk töövahend lubab valdkonnaspetsiifilises keeles muuta visuaalset notatsiooni kui ka metamudeli olemeid.
- Loodud modelleerimise töövahendit pakub moodsat kasutajakogemust, mida juurutatakse veebiserveris. Veebiserver kasutab järgnevaid tipptasemel tehnoloogiaid – Spring, React, PostrgreSQL ja GraphQL.
- Modelleerimise töövahendit majutatakse Obeo Cloud pilvepõhises serveris, siis töövahendi kasutamiseks puudub selle lõppkasutajal vajadus töövahendi töölauale installeerimiseks. Arendatud töövahend on ligipääsetav läbi veebibrauseri. Ühtlasi on tegu SaaS ehk tarkvara teenusena tüüpi lahendus.
- Koosmodelleerimise võimekus, veebipõhine tarkvara majutus loob soodsa olukorra modelleerimisprojekti üheaegseks ligipääsuks ja redigeerimiseks. [33], [34]

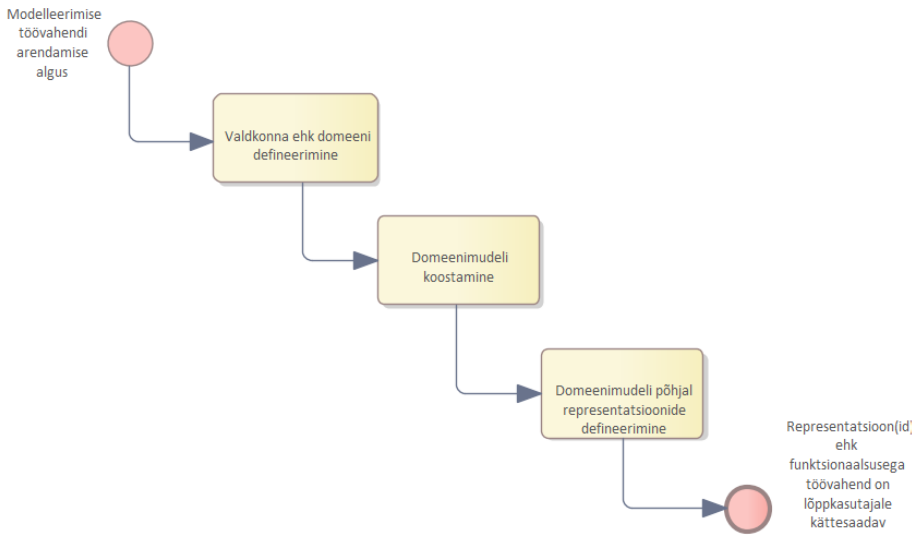
Eelnevalt nimetatud põhiomadused muudavad modelleerimise töövahendi arendamise ja lõpptoodangu kasutamise oluliselt kiiremaks ja intuitiivsemaks. [35] Obeo Cloud

Platvormil toimiv Obeo Studio veebirakendust on võimalik arendada kolmel erineval viisil:

- Täiesti algusest domeenimudeli ja representatsiooni koostamisel teisisõnu *Web Studio Definition* loomine;
- Kasutades Sirius Desktopi rakenduses varasemalt loodud EMF mudelit ja representatsiooni;
- Täiesti algusest koodi kirjutades programmeerides.

Nendest kolmest lõputöö keskendub *Web Studio Definition* ehk veebipõhise modelleerimise töövahendi defineerimisele, sest see on nendest ainuke viis, milles saab modelleerimise töölaua loomisel *no-code* ja mudelipõhist arendusmetoodikat rakendada. [4], [36] Välistati Sirius Desktop EMF mudeli taaskasutamine, sest Sirius Web lahendus ei toeta andmetüüpe, mida Sirius Desktopi modelleerimise töövahendi arendamisel kasutati. Lisaks eeldab EMF taaskasutamine ka oma arvutis arenduskeskkonna loomist ning koodi kirjutamise. Ning kolmas variant on täiesti algusest EMF mudelite arendamine, genereerimine ja veebikeskkonda evitamine. Viimased kaks varianti välistati, sest nendel puhkudel ei saa tugineda *no-code* arendamise metoodikale.

Sirius Web’is modelleerimise töövahendi arendamine ja kasutajatele kättesaadavaks tegemine eeldab palju vähem tegevusi kui võrrelda Sirius Desktop keskkonnaga. Arendaja ei pea arenduskeskkonda oma arvutisse paigaldama, arendamisel pole tarvis mudelist klasse ja klasside redaktoreid genereerida. Lisaks ei tule arendajal mõelda lahendusele, kuidas loodud töövahend lõppkasutajale kättesaadavaks teha – seda saab jagada veebilingina ja veebibrauseriga ligi pääs on tagatud. Järgnevalt on välja toodud peamised sammud alates töölaua arendamise algusest kuni lõppkasutajale kättesaadavaks tegemiseni.



Joonis 3. Veebipõhiselt arendatava valdkonnapõhise keele tegevusvoog

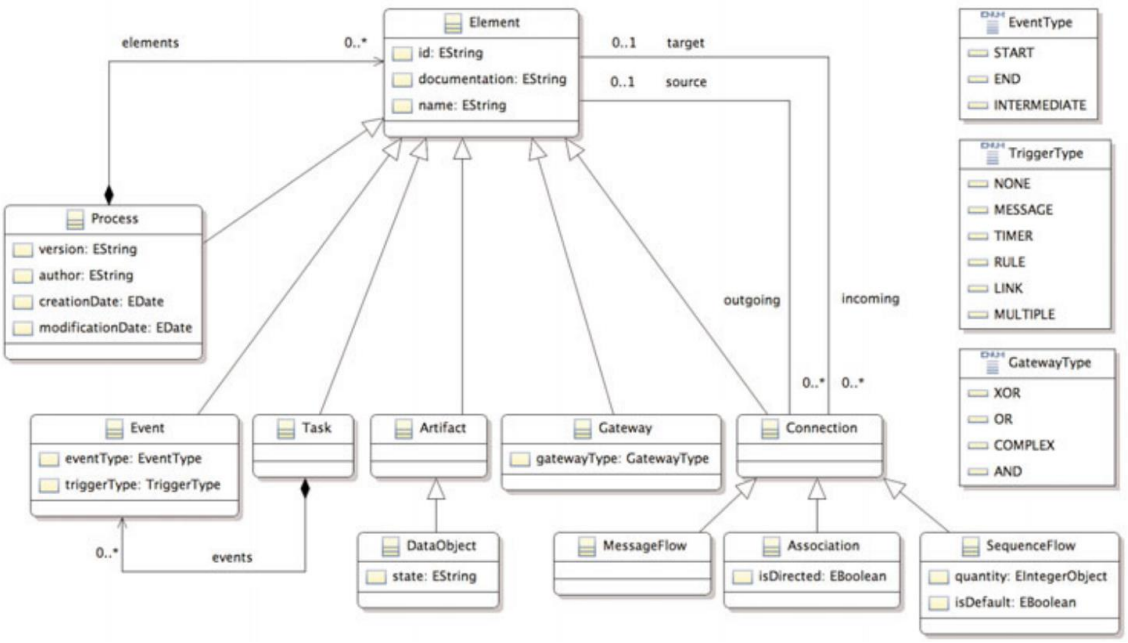
Joonisel 3 on kirjeldatud tegevusvoog, mida peab kasutaja oma modelleerimisstuudio defineerimiseks ja täiendavaks arendamiseks tegema. Kui kõrvutada seda voogu joonisel 2 kujutatud vooga, siis on näha, et tehtavaid tegevusi on vähem, sellest tulenevalt on ka uuenduste implementeerimine oluliselt kiirem protsess.

5 Mudelipõhiselt arendatud tarkvara realisatsioon

Peatükk tutvustab mudelipõhisel arhitektuuril põhineval metoodikal arendatud prototüüpi, millega on võimalik kirjeldada äriprotsesse. Prototüübi realiseerimiseks kasutati Obeo Cloud Studio veebipõhist tarkvara, mis omakorda toetub Eclipse Sirius raamistikule. Funktsionaalne eesmärk on toetada äriprotsesside modelleerimist vastavalt BPMN standardile.

5.1 Nõuded valdkonnamudelile

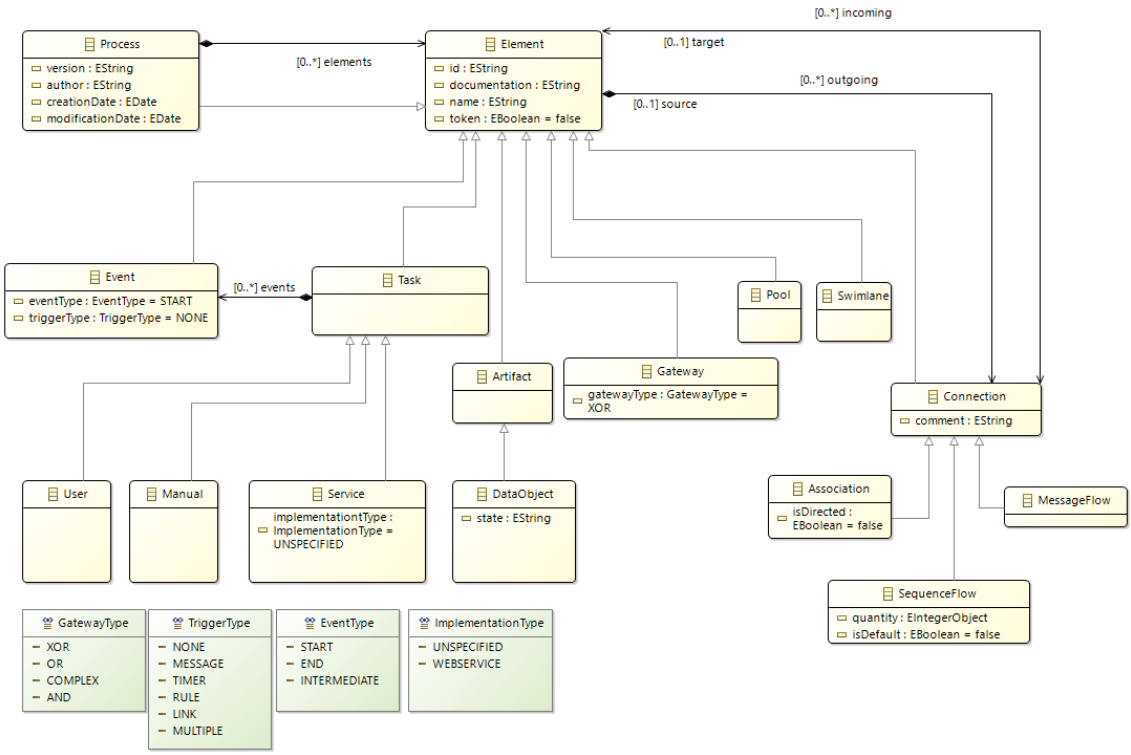
Selleks, et oleks võimalik arendada valdkonnaspetsiifiline modelleerimistarkvara on esmalt vaja luua seda toetav valdkonnamudel, mille abil on võimalik tarkvara kirjeldada. Antud töö kontekstis on valdkonnamudel ühtlasi ka metamudel, sest selle abil on võimalik kirjeldada erinevaid elemente ja tarkvara käitumist. Äriprotsesside kirjeldamise valdkonnamudel on tuletatud autori Richard C. Gronback raamatust *Eclipse Modeling Project: A Domain Specific Language Kit*. [37] Raamatus esile toodud mudel toetab töös püstitatud eesmärke, sest tegemist on BPMN spetsiifilise mudeliga.



Joonis 4. Äriprotsessi koostamise valdkonnamudel

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

Joonisel kirjeldatud valdkonnamudel toetab äriprotsesside modelleerimist, seega võeti see esmalt bakalaureuse töös aluseks. Eclipse Sirius Desktop'is kirjeldati see mudel, et selle abil mudelipõhisele arendusmetoodikale toetudes tarkvara arendada. Praktikas ei saanud allikas esile toodud mudelit kohe kasutusse võtta, mudelit kirjeldades pidi tegema täiendusi, et see ka tehniliselt toimima hakkaks. Peamised muudatused on seotud *Connection* elemendiga, et oleks võimalik erinevate objektide vahel korrektselt seoseid tekitada ja ka näidata. Järgneval joonisel on Sirius Desktop'is koostatud valdkonnamudel, mis on äriprotsesside kaardistamise rakenduse toimimise aluseks. [1], [37]

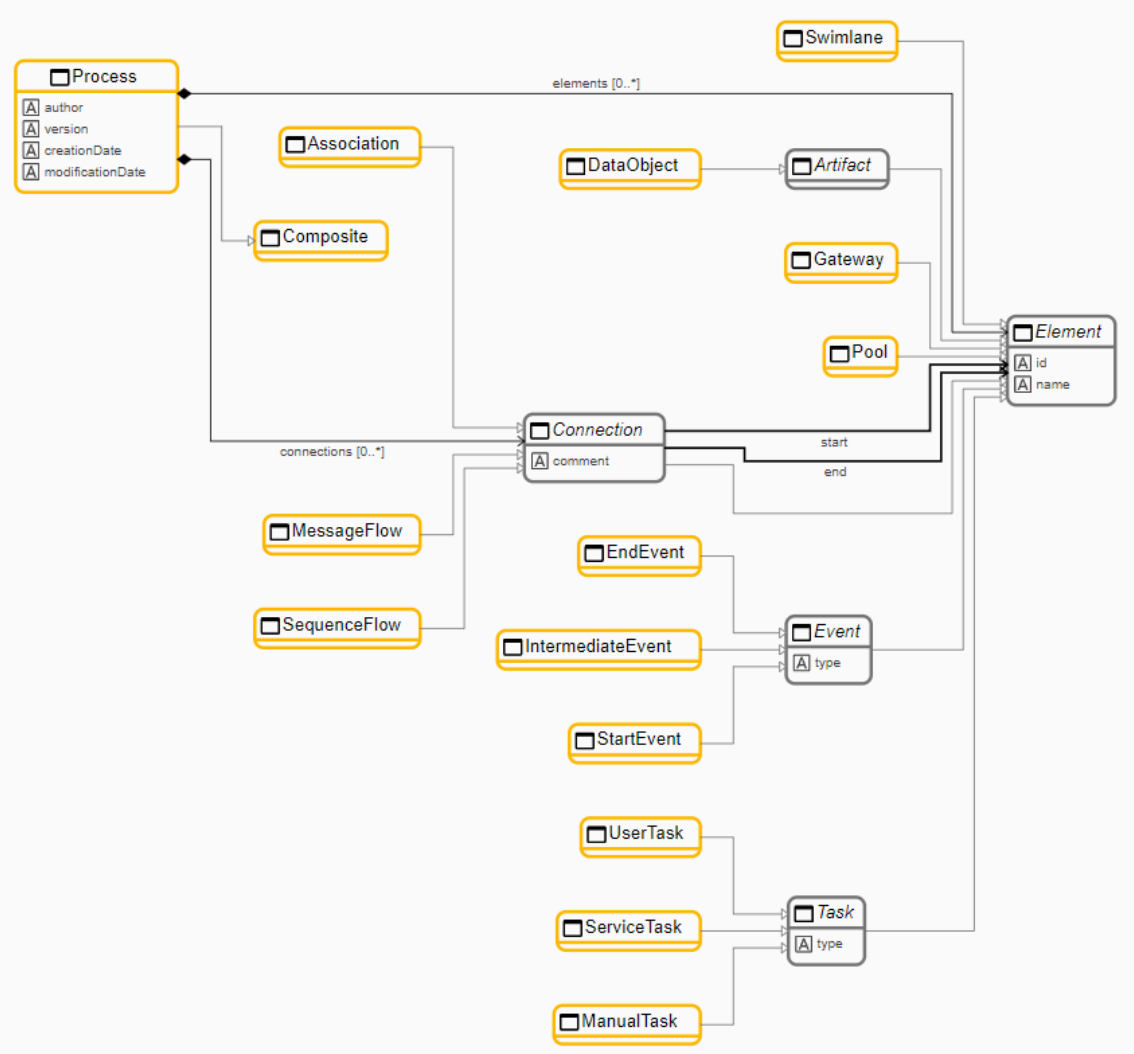


Joonis 5. Äriprotsessi koostamise valdkonnamudel Eclipse Sirius Desktop platvormil Allikas: Äriprotsesside mudelipõhine arendamine Eclipse Sirius platvormil

Eclipse Sirius Desktop ja *Eclipse Sirius Web* toetuvad mõlemad *Eclipse Modeling Framework*'ile, siis tehti eeldus, et joonisel 5. koostatud mudeli saab veebipõhisele Sirius Web platvormile üle viia.

5.2 Koostatud valdkonnamudel

Käesolev peatükk kirjeldab töös koostatud domeeni- ehk valdkonnamudelit, mille alusel defineeritakse modelleerimise töövahendi elemendid ja kasutamisreeglid. Eelnevas peatükis tehti eeldus, et Sirius Desktopis koostatud domeenimudel toimib ja Sirius Webis, siis tugineti Sirius Desktop'is koostatud domeenimudelile ja prooviti see samal kujul kirjeldada ja tööle saada.

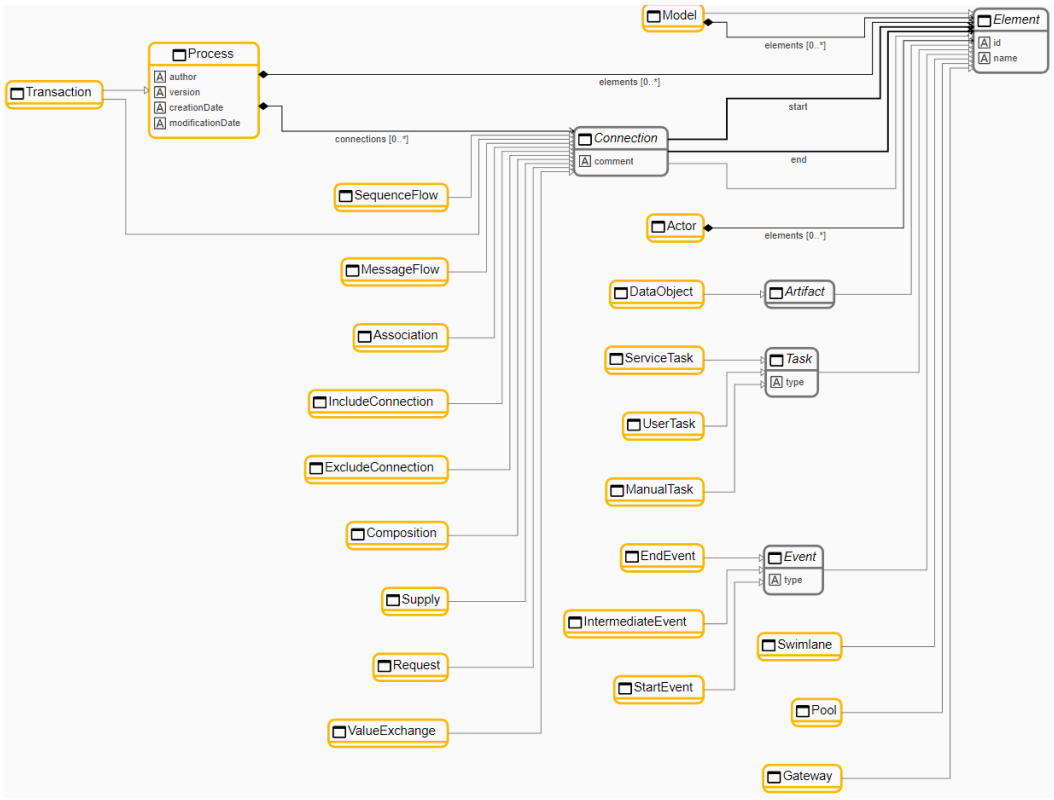


Joonis 6. Äriprotsessi koostamise valdkonnamudel Eclipse Sirius Web platvormil

Eeldus, et Sirius Desktopi valdkonnamudeli saab täiesti üle tuua ja kasutusele võtta ei pidanud paika. Joonisel 5 ja joonisel 6 on mõnevõrra erinevad olemid ja ühendused, seejuures ka muud andmetüübid kasutusel. Võrreldes Sirius Desktop valdkonnamudeliga, siis Sirius Webis tuli mõningaid muudatusi teha, peamiselt seetõttu, et Sirius Web toetab

mudeli koostamisel vähem funktsioone ja andmetüüpe. Esmalt eemaldati enumeratsioonid GatewayType, TriggerType, EventType, ImplementationType. Põhjus tuleneb sellest, et Sirius Web toetab vaid kolme andmetüüpi. Andmetüübid, mida Sirius Web toetab, on *string*, *integer* ja *boolean*, seetõttu kõik muud andmetüübid, enumeratsioonid kaasaarvatud, pole toetatud ja neid kasutada ei saa. Teine suurem erinevus on, et Event olemi erinevad tüübid pidi eraldama ning nendest eraldi olemid tegema. Sirius Desktop'is olid need kõik koondatud Event olemi alla aga Sirius Web'is tehti abstraktne Event olem, millega seoti kolme olemiga kolm erinevat Event tüüpi – StartEvent, IntermediateEvent ja EndEvent. Nüüd on need sarnaselt esitatud nagu Task olem koos alamtüüpidega, kusjuures Task on mõlemas domeenimudelis samamoodi kirjeldatud. Viimaseks jäeti Sirius Web domeenimudelis ära sisalduvusseos Event ja Task olemite vahel, sest Sirius Web praegust funktsionaalsust arvestades pole taolise seosega opereerimine vajalik.

Tulevikku vaadates on soov saada tulemus, kus Sirius Web projektiga oleks võimalik ka äritransaktsioone kirjeldada, siis sellest lähtuvalt on tarvis sisse viia ka vastavad valdkonnamudeli täiendused. Täiendustest on lähtutud Yana Sirotkina bakalaureuse töös, „Ärianalüüsi toetava valdkonnaspetsiifilise modelleerimiskeele prototüübi loomine", koostatud valdkonnamudelist. Töös teostati äritransaktsioonide kaardistamise lahendus Sirius Desktop vahendiga. [2]



Joonis 7. Äritransaktsioonide spetsiifikaga täiendatud valdkonnamudel

Lisandusid olemid:

- Model;
- Transaction;
- IncludeConnection;
- ExcludeConnection;
- Composition;
- Supply;
- Request;
- ValueExchange;
- Actor.

Model on üldine element, mis koondab kogu mudeli tervikuna kokku. Ülejäänud lisandunud elemendid esindavad äritransaktsioonide ehk väärtusvahetuste kirjeldamise valdkonda.

Lisaks Yana Sirotkina tööle vaadeldakse ka Aleksandr Bakalkini bakalaureuse töös, „Äriobjektide modelleerimist toetava valdkonnaspetsiifilise modelleerimiskeele prototüübi loomine“, koostatud valdkonnamudelit. Töös kajastati äriobjektide kirjeldamise valdkonda ning realiseeriti nende kaardistamine Sirius Desktop vahendiga. [3] Valdkonnamudeli täiendus suurendas mudeli suurust sellises mahus, et seda kirjeldav diagramm on välja toodud lisades (Lisa 2). Arvestades, et käesoleva magistritöö keskendub äriprotsesside kaardistamisele, siis praegusele hetkel ainult laiendatakse valdkonnamudelit selliselt, et tekib eeldus äritransaktsioonide ja äriobjektide kirjeldamise lahenduse arendamiseks. See tähendab, et praeguses lahenduses valdkonnamudelile täiendava funktsionaalsuse jaoks vaateid ehk representatsioone ei arendata.

Bakalkini ja Sirotkina lõputöödes käsitletud väärtusvahetuste ja äriobjektide valdkonnamudelite olemite ületoomine Sirius Web keskkonda annab aluse tulevikus ka täiendustele vastavad representatsioonid ehk vaated arendada ja neid veebipõhiselt kasutada. Laiendatud valdkonnamudel on piisavalt võimekas, mille peale on võimalik arendada äriprotsesside, äriobjektide ja äritransaktsioonide modelleerimise võimekus. [2], [3]

5.3 Äriprotsesside modelleerimisstuudio realisatsioon

Peatükis 4.2 kirjeldatud domeenimudeli olemid annavad aluse BPMN standarditele vastavate elementide kirjeldamiseks ning nendega seotud reeglite defineerimiseks. Läbi reeglite defineerimise sisuliselt kirjeldatakse ka loodava tarkvara toimimine. Definitsioon on Sirius View ehk representatsioon ehk moodus domeenimudeli esitamiseks – ühtlasi ka mudelipõhiselt arendatud modelleerimistarkvara realisatsioon.

Sirius Web'is on võimalik defineerida võimalik Node ja Edge elemente. Node on punkt ning Edge on ühendus, mis seob omavahel punkte. Järgnevalt on välja toodud täpsed Node ja Edge kirjeldused, mis on äriprotsesside modelleerimise tarkvara kasutamiseks vajalikud. Järgnevad peatükid toovad iga vajaliku Node ja Edge täpsed kirjeldused välja, neid kirjeldusi võib interpreteerida süsteemi tehnilise dokumentatsioonina. Sirius Web võimaldab väga paljusid atribuute defineerida, siis välja tuuakse vaid oluline, mis on käesoleva töö praktilise tulemuse saavutamiseks vajalik.

Järgnevad peatükid kirjeldavad ära elemendid, mida peab saama äriprotsessi koostades kasutada. Esmalt tuuakse esile nende kirjeldus domeenimudelis ning seejärel defineeritud nende välimus diagrammil ning funktsioonid, mille abil saab neid diagrammile tekitada. Seejärel on välja toodud, millisena elementi diagrammil kuvatakse ning mis on konkreetse elemendi roll BPMN äriprotsessi diagrammil. Ühenduselementide puhul on kirjeldatud reeglid, milliste ühendusliikide puhul saab milliseid BPMN elemente omavahel siduda.
Node ehk punkti kirjeldamisel lähtutakse järgnevast kujust:
Node kirjeldus – punkti valdkonnamudelist sõltuvad parameetrid:

- **Name –** node ehk puntki nimetus;
- **Domain Type –** viide domeenimudelis kirjeldatud olemile, millele node ehk punkt põhineb.

NodeStyle kirjeldus – punkti välimuse määravad parameetrid:

- **Color** – diagrammile loodava objekti põhivärv;
- **Border Color –** diagrammile loodava objekti piirvärv.

CreateNode kirjeldus – punkti loomise funktsionaalsuse defineerimine:

- **Name** – töövahendi kasutajale kuvatav funktsiooni nimetus, millega luuakse kindle objekt diagrammile.

CreateInstance kirjeldus – punkti instantsi diagrammile loomise funktsiooni kirjeldus:

- **Type Name** viide valdkonnamudeli olemile, mille abil on punkti objekt kirjeldatud;
- **Reference Name** – viite nimi, mis seob kahte valdkonnamudeli olemit;
- **Variable Name** – nimi, mis määratakse loodud objekti instantsile.

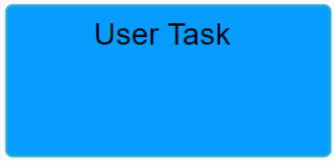
Edge ehk ühendusjoone kirjeldamisel lähtutakse järgnevast kujust:

- Edge kirjeldus –ühendusjoone valdkonnamudelist sõltuvad parameetrid:
    - **Name** – kasutajale kuvatav joone nimetus, mis näitab, millist joont saab luua;
    - **DomainType** – viide valdkonnamudeli olemile, mille abil on ühendusjoone objekt kirjeldatud.
    - **Source Node Descriptions –** kirjeldab, millistest punktidest ehk objektidest on võimalik ühendusjoont alustada;
    - **Target Node Descriptions –** kirjeldab, millistes punktides ehk objektides on võimalik ühendusjoont lõpetada;
    - **Source Nodes Expression** - ühendusjoone alguse definitsiooni väljendus AQL süntaksis;
    - **Target Nodes Expression** – ühendusjoone lõpu definitsiooni väljendus AQL süntaksis.
- EdgeStyle kirjeldus – ühendusjoone välimuse määravad parameetrid:
    - **Color –** diagrammil ühendusjoone värvus;
    - **Italic –** määrab, kas ühendusjoonele kirjutatud tekst on kaldjoones;
    - **Bold** – määrab, kas ühendusjoonele kirjutatud tekst on paksus kirjas;
    - **Line Style –** määrab ühendusjoone stiili (nt pidevjoon või katkendjoon).

5.3.1 Kasutaja tegevus ehk User Task kirjeldus

Kasutaja tegevuse objekti kasutamiseks tuleb kirjeldada Node, NodeStyle elemendid ning defineerida CreateNode ja CreateInstance funktsioonid:

- Node kirjeldus:
    - **Name** User Task;
    - **Domain Type**: UserTask.
- NodeStyle kirjeldus:
    - **Color** #069DFF;
    - **Border Color** #33B0C3.
- CreateNode kirjeldus:
    - **Name** Create User Task.
- CreateInstance kirjeldus:
    - **Type Name** BusinessProcess::UserTask;
    - **Reference Name** elements;
    - **Variable Name** newUserTask.



Joonis 8. User Task diagrammil

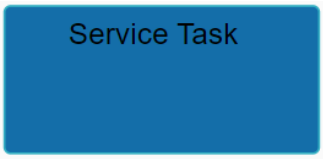
User Task on kasutaja tegevuse andmevoo element kirjeldab inimese poolt tehtavat tegevust, mida tehes toetutakse tavaliselt mõnele tehnoloogilisele lahendusele. Tegevus teostatakse kasutaja ja tehnoloogia kaasabil. BPMN standardi järgi kujutatakse seda ristkülikuna, mille vasakul üleval nurgas on isiku sümbol. Sirius Web funktsionaalsuse piirangute tõttu kuvatakse seda sinise ristkülikuna. [38], [39]

5.3.2 Teenuse poolt tehtav tegevus ehk Service Task kirjeldus

Teenuse poolt tehtava tegevuse objekti kasutamiseks tuleb kirjeldada Node, NodeStyle elemendid ning defineerida CreateNode ja CreateInstance funktsioonid:

- Node kirjeldus:
    - **Name** Service Task;
    - **Domain Type** ServiceTask.
- NodeStyle kirjeldus:
    - **Color** #146EA9;

    - **Border Color** #33B0C3.
- CreateNode kirjeldus:
    - **Name** Create Service Task.
- CreateInstance kirjeldus:
    - **Type Name** BusinessProcess::ServiceTask;
    - **Reference Name** elements;
    - **Variable Name** newServiceTask.



Joonis 9. Service Task diagrammil

Service Task on tegevus, mis täidetakse täies mahus automaatika poolt, näiteks programmikoodi abil automatiseeritud rakendus või veebiteenus. BPMN standardi järgi kujutatakse seda ristkülikuna, mille vasakul üleval nurgas on hammasratta sümbol. Sirius Web funktsionaalsuse piirangute tõttu on lahendatud selle kuvamine tumesinise ristkülikuga. [38], [39]

5.3.3 Manuaalselt tehtav tegevus ehk Manual Task kirjeldus

Manuaalselt tehtava tegevuse objekti kasutamiseks tuleb kirjeldada Node, NodeStyle elemendid ning defineerida CreateNode ja CreateInstance funktsioonid:
Node kirjeldus:

- **Name** Manual Task;
- **Domain Type** ManualTask.

NodeStyle kirjeldus:

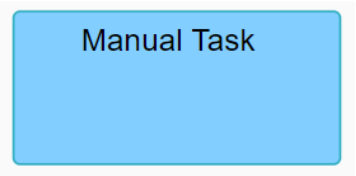
- **Color** #82CEFF;
- **Border Color** #33B0C3.

CreateNode kirjeldus:

- **Name** Create Manual Task.

CreateInstance kirjeldus:

- **Type Name** BusinessProcess::ManualTask;
- **Reference Name** elements:
- **Variable Name** newManualTask.



Joonis 10. Manual Task diagrammil

Manual Task ehk manuaalselt tehtavat tegevust tehakse inimese poolt, ilma igasuguse tehnoloogilise kaasabita. Oma olemuselt on see vastupidine Service Task ehk teenuse poolt tehtava tegevusega. BPMN standardi järgi kujutatakse seda ristkülikuna, mille vasakul üleval nurgas on kätt kujutav sümbol. Sirius Web funktsionaalsuse piirangute tõttu on lahendatud selle kuvamine helesinise ristkülikuga. [38], [39]

5.3.4 Protsessi algatav sündmus ehk Start Event kirjeldus

Protsessi algatava sündmuse objekti kasutamiseks tuleb kirjeldada Node, NodeStyle elemendid ning defineerida CreateNode ja CreateInstance funktsioonid:

- Node kirjeldus:
  - **Name** Start Event;
  - **Domain Type** StartEvent.
- NodeStyle kirjeldus:
  - **Color** #B9FFB5;
  - **Border Color** #00E855;
  - **Border Radius** 100.
- CreateNode kirjeldus:
  - **Name** Create Start Event.
- CreateInstance kirjeldus:
  - **Type Name** BusinessProcess::StartEvent;
  - **Reference Name** elements;
  - **Variable Name** newStartEvent.

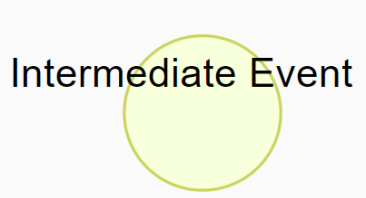
Joonis 11. Start Event diagrammil

Start Event on protsessi algatav sündmus, seda saab ühe protsessi käigus vaid üks olla. BPMN standardi järgi kujutatakse seda rohelist värvi ringikujulise sümbolina. [33], [38], [39]

5.3.5 Protsessi kestel toimuv sündums ehk Intermediate Event kirjeldus

Protsessi kestel toimuva sündmuse objekti kasutamiseks tuleb kirjeldada Node, NodeStyle elemendid ning defineerida CreateNode ja CreateInstance funktsioonid:

- Node kirjeldus:
  - **Name** Intermediate Event;
  - **Domain Type** IntermediateEvent.
- NodeStyle kirjeldus:
  - **Color** #F7FFDB;
  - **Border Color** #CCD45D;
  - **Border Radius** 100.
- CreateNode kirjeldus:
  - **Name** Create Intermediate Event.
- CreateInstance kirjeldus:
  - **Type Name** BusinessProcess::IntermediateEvent;
  - **Reference Name** elements;
  - **Variable Name** newIntermediateEvent.



Joonis 12. Intermediate Event diagrammil

Intermediate event on protsessi vältel toimuv sündmus, mis käivitab protsessi jätkamise. Ühe protsessi kirjeldamisel juures võib neid mitu olla. BPMN standardi järgi kujutatakse seda kollast värvi ringikujulise sümbolina. [33], [38], [39]

5.3.6 Protsessi lõpetav sündmus ehk End Event kirjeldus

Protsessi lõpetava sündmuse objekti kasutamiseks tuleb kirjeldada Node, NodeStyle elemendid ning defineerida CreateNode ja CreateInstance funktsioonid:

- Node kirjeldus:
  - **Name** End Event;
  - **Domain Type** EndEvent.
- NodeStyle kirjeldus:
  - **Color** #FAAEAA;
  - **Border Color** #FF1B0F;
  - **Border Radius** 100.
- CreateNode kirjeldus:
  - **Name Create** End Event.
- CreateInstance kirjeldus:
  - **Type Name** BusinessProcess::EndEvent;
  - **Reference Name** elements;
  - **Variable Name** newEndEvent.

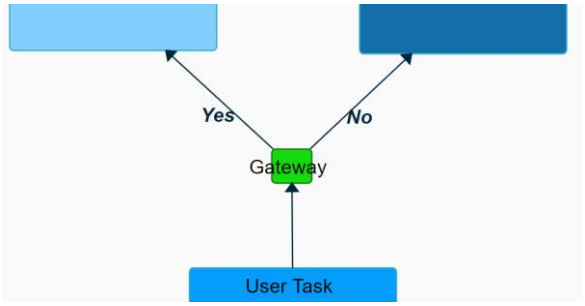
Joonis 13. End Event diagrammil

End Event on protsessi lõpetav sündmus, üldjuhul eelneb sellele Task ehk tegevus tüüpi element. Protsessi kirjeldamisel võib lõpu sündmuseid mitu olla, protsessil võib mitu erinevat lõpptulemust olla. BPMN standardi järgi kujutatakse seda punase ringikujulise sümbolina. [33], [38], [39]

5.3.7 Lüüs ehk Gateway kirjeldus

Lüüs objekti kasutamiseks tuleb kirjeldada Node, NodeStyle elemendid ning defineerida CreateNode ja CreateInstance funktsioonid:

- Node kirjeldus:
  - **Name** Gateway;
  - **Domain Type** Gateway.
- NodeStyle kirjeldus:
  - **Color** #16DB0F;
  - **Border Color** #0E750B.
- CreateNode kirjeldus:
  - **Name** Create Gateway.
- CreateInstance kirjeldus:
  - **Type Name** BusinessProcess::Gateway;
  - **Reference Name** elements;
  - **Variable Name** newGateway.



Joonis 14. Gateway diagrammil

Gateway ehk lüüs elemendiga kirjeldatakse protsessi jooksul tehtavat otsust. Võib siseneda mitmest tegevusest ning hargneb vähemalt kahe erineva tegevuse suunas, võib ka rohkema, kui lüüsi tingimused seda ette näevad. BPMN standardi järgi kujutatakse lüüsi rohelise rombina, kuid Sirius Web'i elemendi kirjeldamise funktsionaalsus on piiratud, siis antud lahenduse raames kujutatakse seda rohelise ruuduna. [33], [38], [39]

5.3.8 Andmeobjekt ehk Data Object kirjeldus

Andmeobjekti kasutamiseks tuleb kirjeldada Node, NodeStyle elemendid ning defineerida CreateNode ja CreateInstance funktsioonid:

- Node kirjeldus:
    - **Name** Data Object;
    - **Domain Type** DataObject.
- NodeStyle kirjeldus:
    - **Color** #FFFFFF;
    - **Border Color** #919191.
- CreateNode kirjeldus:
    - **Name** Create Data Object;
- CreateInstance kirjeldus:
    - **Type Name** BusinessProcess::DataObject;
    - **Reference Name** elements;
    - **Variable Name** newDataObject.

Joonis 15. Data Object diagrammil

Data Object ehk andmeobjekt kirjedab äriprotsessis andmeid, mida on konkreetse tegevuse tarvis kasutada. Andmeobjekti võib siduda kõikide tegevus tüüpi elementidega, association tüüpi ühendusega. [38], [39], [33]

5.3.9 Bassein ehk Pool kirjeldus

Bassein objekti kasutamiseks tuleb kirjeldada Node, NodeStyle elemendid ning defineerida CreateNode ja CreateInstance funktsioonid:

- Node kirjeldus:
    - **Name** Pool;
    - **Domain Type** Pool.
- NodeStyle kirjeldus:
    - **Color** #FFFFFF;
    - **Border Color** #000000:
    - **Border Radius** 0.
- CreateNode kirjeldus:

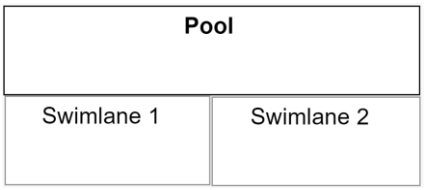
    - **Name** Create Pool.
- CreateInstance kirjeldus:
    - **Type Name** BusinessProcess::Pool;
    - **Reference Name** elements;
    - **Variable Name** newPool.

Pool ehk bassein on element, mis koondab kogu protsessi tegutsejaid ning tegutsejate poolt tehtavaid tegevusi, tegutsejaid mõjutavaid sündmuseid ja tegevuste tegemiseks vajalikke artefakte. [38], [39]

5.3.10 Ujumisrada ehk Swimlane kirjeldus

Ujumisraja objekti kasutamiseks tuleb kirjeldada Node, NodeStyle elemendid ning defineerida CreateNode ja CreateInstance funktsioonid:

- Node kirjeldus:
    - **Name** Swimlane;
    - **Domain Type** Swimlane;
- NodeStyle kirjeldus:
    - **Color** #FFFFFF;
    - **Border Color** #919191;
    - **Border Radius** 0.
- CreateNode kirjeldus:
    - **Name** Create Swimlane.
- CreateInstance kirjeldus:
    - **Type Name** BusinessProcess::Swimlane;
    - **Reference Name** elements;
    - **Variable Name** newSwimlane:



Joonis 16. Pool ja Swimlane elemendid diagrammil

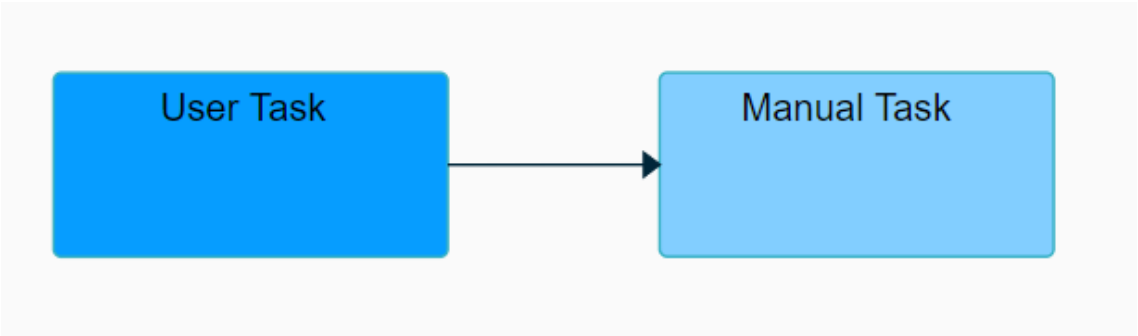
Swimlane ehk ujumisrada kirjeldab protsessis tegutseva osapoole tehtavaid tegevusi, osapooli mõjutavaid sündmuseid ja tegevuste tegemiseks vajalikke artefakte. [38], [39]

5.3.11 Protsessi tegevusvoog ehk Sequence Flow kirjeldus

Protsessi tegevusvoo objekti kasutamiseks tuleb kirjeldada Edge ja EdgeStyle elemendid, ühendusjoonte tarvis pole eraldi funktsioone vaja defineerida:

- Edge kirjeldus:
    - **Name** Sequence Flow;
    - **DomainType** SequenceFlow;
    - **Source Node Descriptions:**
        - User Task;
        - Service Task;
        - Manual Task;
        - Start Event;
        - Intermediate Event;
        - End Event;
        - Gateway;
        - Swimlane.
    - **Target Node Descriptions:**
        - User Task;
        - Service Task;
        - Manual Task;
        - Start Event;
        - Intermediate Event;
        - End Event;
        - Gateway;
        - Swimlane.
    - **Source Nodes Expression** aql:self.start;
    - **Target Nodes Expression** aql:self.end.
- EdgeStyle kirjeldus:
    - **Color** #002639;
    - **Italic** True;

- **Bold** True;
- **Line Style** Solid.



Joonis 17. Sequence Flow diagrammil

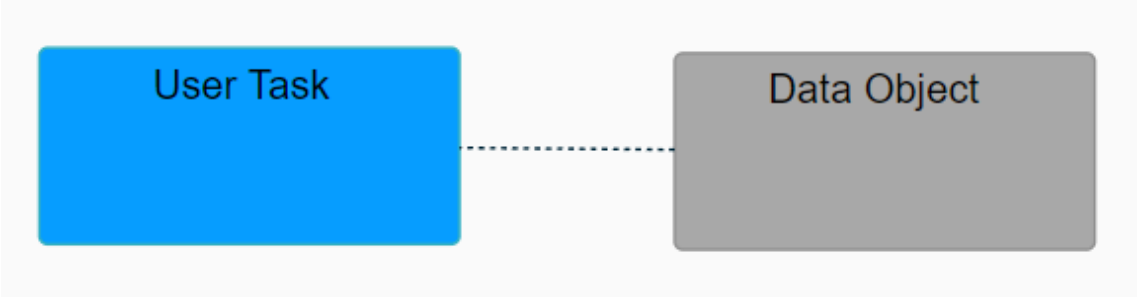
Sequence Flow ehk tegevusvoog näitab protsessi kulgemuse suunda erinevate tegevuste vahel. Saab ühenduda peaagi kõikide BPMN standardi objektidega. BPMN standardi järgi kuvatakse seda suunatud noolega pidevjoonena, selliselt on võimalik seda kujutada ka Sirius Web arendatud tarkvaras. [38], [39]

5.3.12 Protsessi elementidevaheline seos ehk Association element

Protsessi elementidevahelise seose objekti kasutamiseks tuleb kirjeldada Edge ja EdgeStyle elemendid, ühendusjoonte tarvis pole eraldi funktsioone vaja defineerida:

- Edge kirjeldus:
  - **Name** Association;
  - **DomainType** Association;
  - **Source Node Descriptions:**
    - User Task;
    - Service Task
    - Manual Task;
    - Data Object.
  - **Target Node Descriptions:**
    - User Task;
    - Service Task
    - Manual Task;
    - Data Object.
  - **Source Nodes Expression** aql:self.start;

    - **Target Nodes Expression** aql:self.end.
- EdgeStyle kirjeldus:
    - **Color** #002639;
    - **Italic** True;
    - **Bold** True;
    - **Line Style** Solid.



Joonis 18. Association diagrammil

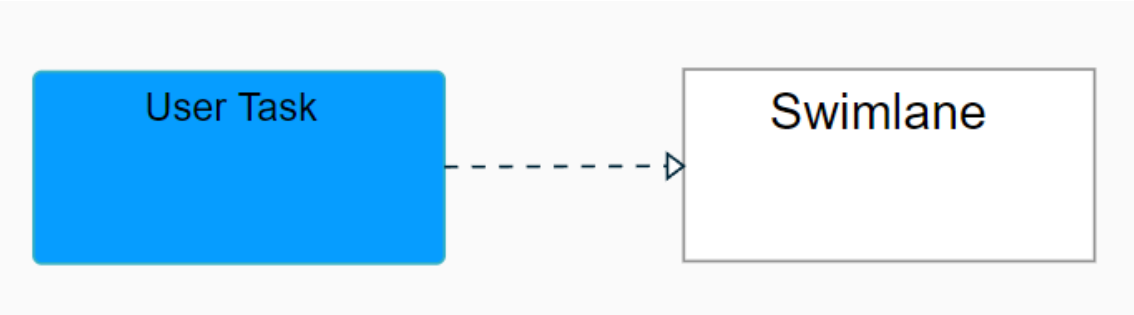
Associaton ehk ühendust kasutatakse äriprotsesside kaardistamisel tegevuste ja artefaktide sidumiseks. Näiteks, et näidata milliseid andmeid on kindla tegevuse läbiviimiseks tarvis. BPMN standardi järgi kuvatakse seda ilma suunata punktiirjoonena, selliselt on võimalik seda kujutada ka Sirius Web arendatud tarkvaras. [38], [39], [33]

5.3.13 Protsessi sõnumivoog ehk Message Flow kirjeldus

Protsessi sõnumivoo objekti kasutamiseks tuleb kirjeldada Edge ja EdgeStyle elemendid, ühendusjoonte tarvis pole eraldi funktsioone vaja defineerida:

- Edge kirjeldus:
    - **Name** Message Flow;
    - **DomainType** MessageFlow;
    - **Source Node Descriptions:**
        - User Task;
        - Service Task;
        - Manual Task;
        - Swimlane;
        - Pool.

    - **Target Node Descriptions:**
        - User Task;
        - Service Task;
        - Manual Task;
        - Swimlane;
        - Pool.
    - **Source Nodes Expression** aql:self.start;
    - **Target Nodes Expression** aql:self.end;
- EdgeStyle kirjeldus:
    - **Color** #002639;
    - **Italic** True;
    - **Bold** True;
    - **Line Style** Solid.

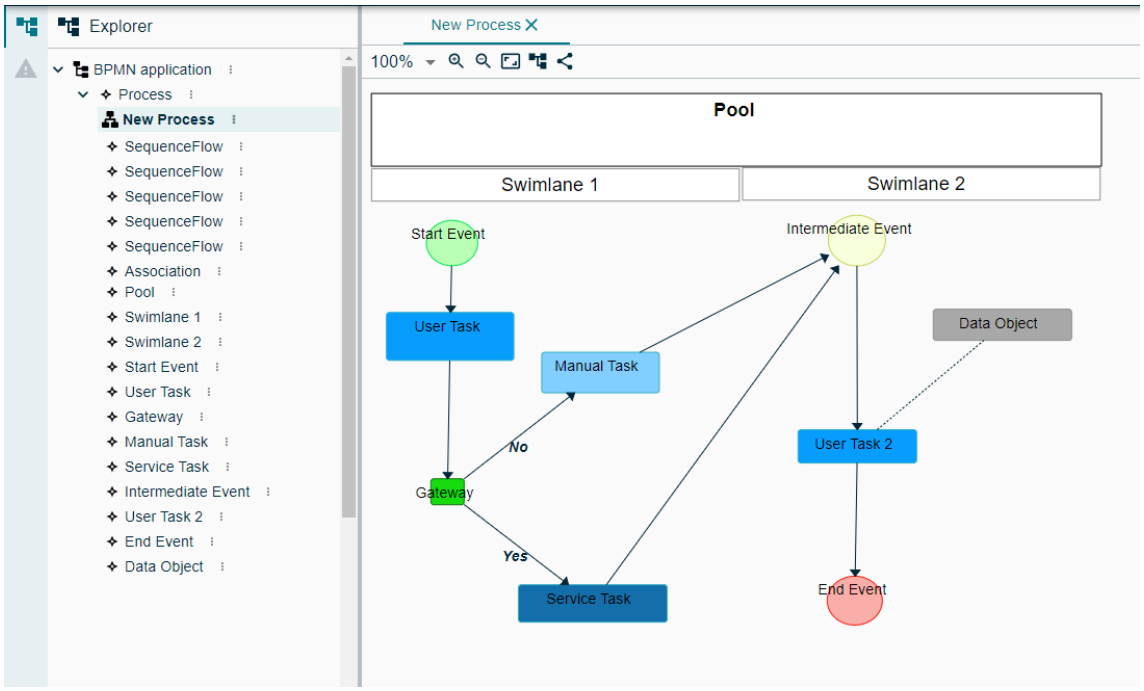


Joonis 19. Message Flow diagrammil

Message Flow on sõnumivoog ning sellega näidatakse sõnumeid või teavet, mis liigub läbi erinevate protsessibasseinide või ujumisradade vahel. Seda ei kasutata ühe basseini sees, ainult siis kui on tarvis ühe protsessi sõltuvust teise n-ö välise protsessiga näidata. BPMN standardi järgi kuvatakse seda suunatud punktiirjoonena, selliselt on võimalik seda kujutada ka Sirius Web arendatud tarkvaras. [38], [39]

5.4 Modelleerimisstuudio töölaud

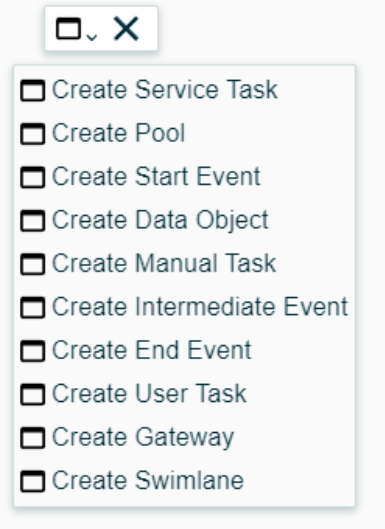
Töölaua peatükis näidatakse lugejale Sirius Web keskkonnas arendatud modelleerimisstuudio ehk töövahendi kasutajaliidest, seejuures tuuakse välja ka võrdlus Sirius Desktop lahendusega. Töölaual on võimalik diagrammile luua kõiki peatükis 5.3 kirjeldatud elemente ning luua nendevahelisi seoseid.



Joonis 20. Kasutajale kuvatav töölaud kõikide võimalike elementidega

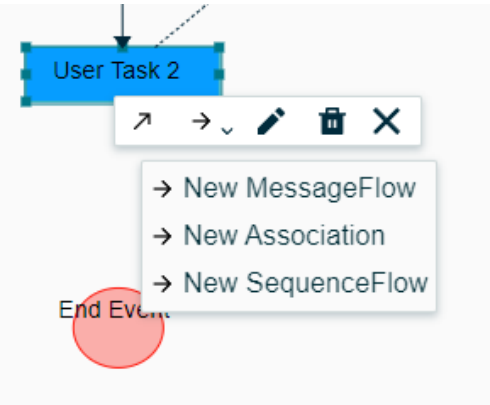
Sirius Web'is käib elementide loomine vaid läbi kursori, mis tähendab, et kõikide elementide loomine käib diagrammi peal. Sirius Desktop sarnast tööriista kasti kasutajale nähtav pole, mis kategoriseeriks elemendid ning millest saaks elemente diagrammile lohistada.

Joonis 21. Kursori vajutusel kuvatavad valikud



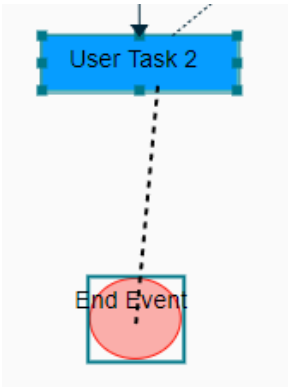
Joonis 22. BPMN elementide loomise nupud

Kursoriga ruudu peale vajutades, esitatakse kasutajale valiku elementidest, mida on võimalik diagrammile luua. Vastava valiku peale klikkides luuakse diagrammile soovitud element.



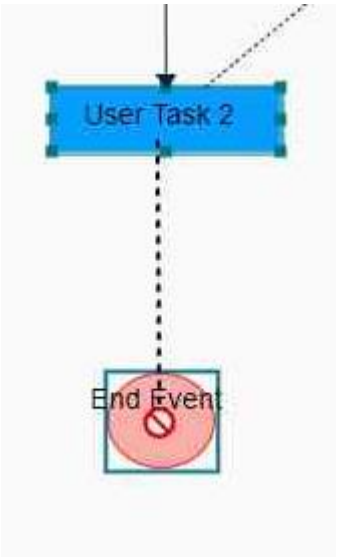
Joonis 23. Punktist algava ühendusjoone valik

BPMN elementide omavaheliseks sidumiseks on tarvis esmalt klikkida elemendi peale, mis saab olema ühenduse üheks otspunktiks. Elemendile vajutades kuvatakse kasutajale valikut, millist liiki ühendust saab konkreetse elemendi liigiga teha.



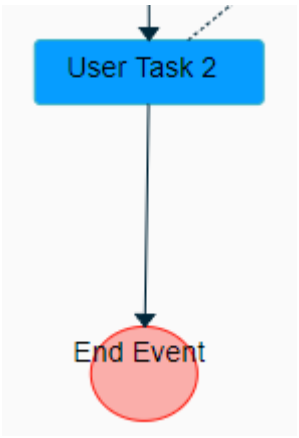
Joonis 24. BPMN elementide vahelise ühenduse loomine

Ühendust saab luua vaid seda tüüpi olemi elementide vahel, mis on peatükkides 5.3.11, 5.3.12 ja 5.3.13 defineeritud. Modelleerimise vahend arvestab reeglitega, mis määrati domeenimudeli ja selle vaate koostamisel.



Joonis 25. Ebasobiva olemi tüübiga elemendiga ühendamine

Juhul kui kasutaja proovib ebasobivat ühendust luua, siis modelleerimise tööriist seda ei luba teha. Ebasobiva ühenduse korral kuvatakse kasutajale keelavat kursorit, millega elemendi peale klikkides ühendust ei looda.

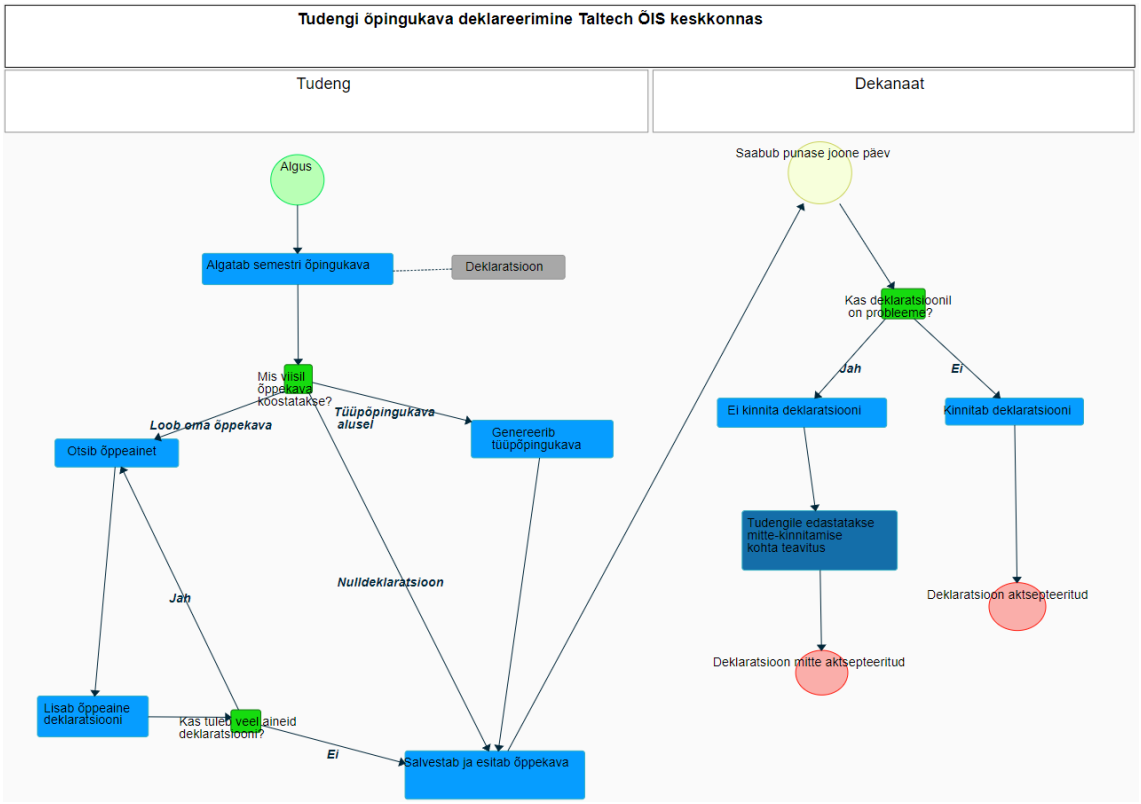


Joonis 26. Sobiv ühendus

Sobiva ühenduse korral tekib ühendus joon kahe elemendi vahel, mida kasutaja soovis omavahel ühendada.

5.4.1 Koostatud protsess

Järgnevalt on esitatud ekraanipilt äriprotsessist, mis on koostatud lõputöös arendatud töövahendiga. Tegu on TalTech õppekava deklareerimise protsess, milles osalevad tudeng ja dekanaat ning mille käigus on kolm otsustuskoha, üks algav sündmus, üks keskne sündmus ning kaks võimalikku tulemust. Tegu on lihtsustatud kujul protsessiga, sest oluline on näidata, et töövahendiga on võimalik protsesse kaardistada. See tähendab, et funktsionaalsuse näitamine ei sõltu kaardistatava protsessi keerukusest.



Joonis 27. Koostatud protsessikirjeldus Sirius Web'is

Protsessi kirjeldamisel kasutatud peaaegu kõik võimalikud elemendid ära, mida eelnevalt koostatud valdkonnamudelis äriprotsesside spetsiifikaga kirjeldati ja mudeli representatsiooniga defineeriti. Ainult element *Manual Task* ehk manuaalselt tehtavat tegevust jäi kasutamata. Järgides komponendipõhist ja ökonoomset tarkvaraarendust, siis antud protsessist lähtuvast on *Manual Task*'i diagrammil kirjeldamise funktsionaalsuse loomine liiasus. Eesmärk oli luua BPMN standardit järgiv modelleerimise töövahend, siis ei lähtutud mitte kirjeldatavast protsessist, vaid BPMN standardi põhiobjektidest.

6 Analüüs ja järeldused

Käesolev peatükk analüüsib töös saadud tulemusi ning teeb analüüsi alusel vajalikud järeldused. Esmalt võrreldakse Eclipse Sirius Web ja Eclipse Sirius Desktop platvorme, tuuakse välja peamised erinevused nii olemuslikult kui ka arendusprotsessis. Seejärel tuuakse käesolevas töös arendatud modelleerimise töövahendi võrdlemiseks teised sarnased vahendid – Sirius Desktop platvormil arendatud lahendus ning Bizagi Modeler. Lisaks antakse hinnang töös kasutatud arendusmetoodikale ning platvormile. Lõpetuseks kirjeldatakse arendatud modelleerimise töövahendit ja pakutakse välja töö tulemuste potentsiaalseid edasi uurimise suundasid.

6.1 Sirius Web ja Sirius Desktop erinevused

Sirius Desktop ja Sirius Web toetuvad mõlemad Eclipse modelleerimise raamistikule, siis sellest hoolimata on tegu siiski kahe erineva tehnoloogiaga, mille erinevusi tuleb vaadelda ja kumbagi kasutades arvestada. Järgnevalt on tabelis väljatoodud peamised mõisted ning kuidas neid Sirius Web või Sirius Desktopis nimetatakse. Välja on toodud peamiselt mõisted, mis väljenduvad töövahendi arendamise kasutajaliideses.

Tabel 1. Mõistete nimetamiste erinevused Sirius Desktop ja Sirius Web vahel

Võrreldav asjaolu	**Sirius Desktop**	**Sirius Web**
Valdkonnamudel	.ecore.	Domain.
Valdkonnamudeli representatsioon	.odesign.	Representation.
Punkt - domeenimudelis olem	Node.	Node.
Serv - domeenimudelis ühendusjoon olemite vahel	Edge.	Edge.
Toetatud andmetüübid	Palju etteantud andmetüüpe, saab ka ise enumeratsioonide näol uusi andmetüüpe juurde luua.	String, integer, boolean.

Tabel kirjeldab kuidas Sirius Desktop'is ja Sirius Web'is sarnaseid mõisteid nimetatakse, millega tuleb arvestada kui liigutakse ühelt lahenduselt teisele. Lisaks mõistete nimetamise erinevusega selgub tabelist peamine põhjus, mis takistab *Sirius Desktop*'is koostatud valdkonnamudeli üleviimist otse *Sirius Web*'i. Juhul kui on loodud valdkonnamudel, mille olemite atribuutide andmetüüpideks on midagi muud kui string, integer või boolean, siis koheselt Sirius Web sellist mudelit ei toeta. Sellest tehti järeldus, et varasemalt koostatud valdkonnamudelit ei saa üks-ühele üle tuua, vaid uus mudel tuleb uuesti käsitsi kaardistada, ning teha seda sellisel kujul nagu Sirius Web platvorm lubab. Mõned ühendusjooned lahendati lihtsamalt ning enumeratsioon andmetüüpide asemel kasutati *string* andmetüüpi.

Järgmiseks on toodud tabel näitamaks erinevusi Sirius Desktop'i ja Sirius Web'i tehtavatest tegevustest, mis on seotud arendusprotsessi läbiviimisega, potentsiaalsete muudatuste tegemise ning muudatuste evitamisega. Tabel illustreerib kummagi tehnoloogia omadusi, seeläbi annab kujutuse, et milline nendest on kasutajale kiiremini õpitav ja kasutusele võetav.

Tabel 2. Arendustegevuse sammude erinevused Sirius Desktop ja Sirius Web vahel

Tegevuse sammu number	**Sirius Desktop**	**Sirius Web**
1. samm	Valdkonna ehk domeeni defineerimine,	Valdkonna ehk domeeni defineerimine,
2. samm	Domeeni klasside genereerimine,	Domeenimudeli loomine,
3. samm	DSL editoride loomine,	Domeenimudeli representatsiooni(de) defineerimine,
4. samm	Vajalike pluginate käivitamine,	Representatsioon(id) ehk funktsionaalsusega tarkvara on kasutajale kättesaadav,
5. samm	Domeenimudeli loomine,	Eelnevate tegevuste iteratsioon,

6. samm	Domeenimudeli representatsiooni(de) defineerimine,	-
7. samm	Loodud rakenduse pakkimine kasutajatele saatmiseks,	-
8. samm	Tarkvara evitamine kasutajatele,	-
9. samm	Representatsioon(id) ehk funktsionaalsusega tarkvara on kasutajale kättesaadav,	-
10. samm	Eelnevate tegevuste iteratsioon,	-

Tabel 2 kirjeldab, et mõned tegevused on mõlema tarkvara puhul kattuvad, kuid selgesti paistab välja, et *Sirus Desktop* puhul tuleb kaks korda rohkem tegevusi teha, alates tarkvara arendamise algusest kuni selle kasutajate kättesaadavaks tegemiseni. Lisaks on *Sirius Desktop*'is tehtavad tegevused võrreldes *Sirius Web*'iga veidi tehnilisemad ja veaohtlikumad, siis tuleb arendajal nendesse tegevustesse rohkem aega panustama. Kuna tabeli eesmärk on peamiselt illustreerida kahe arenduskeskkonna arendustegevuse sammude arvu erinevust, siis ei eeldata, et lugeja nende kõikide sammude sisulist tausta teab.

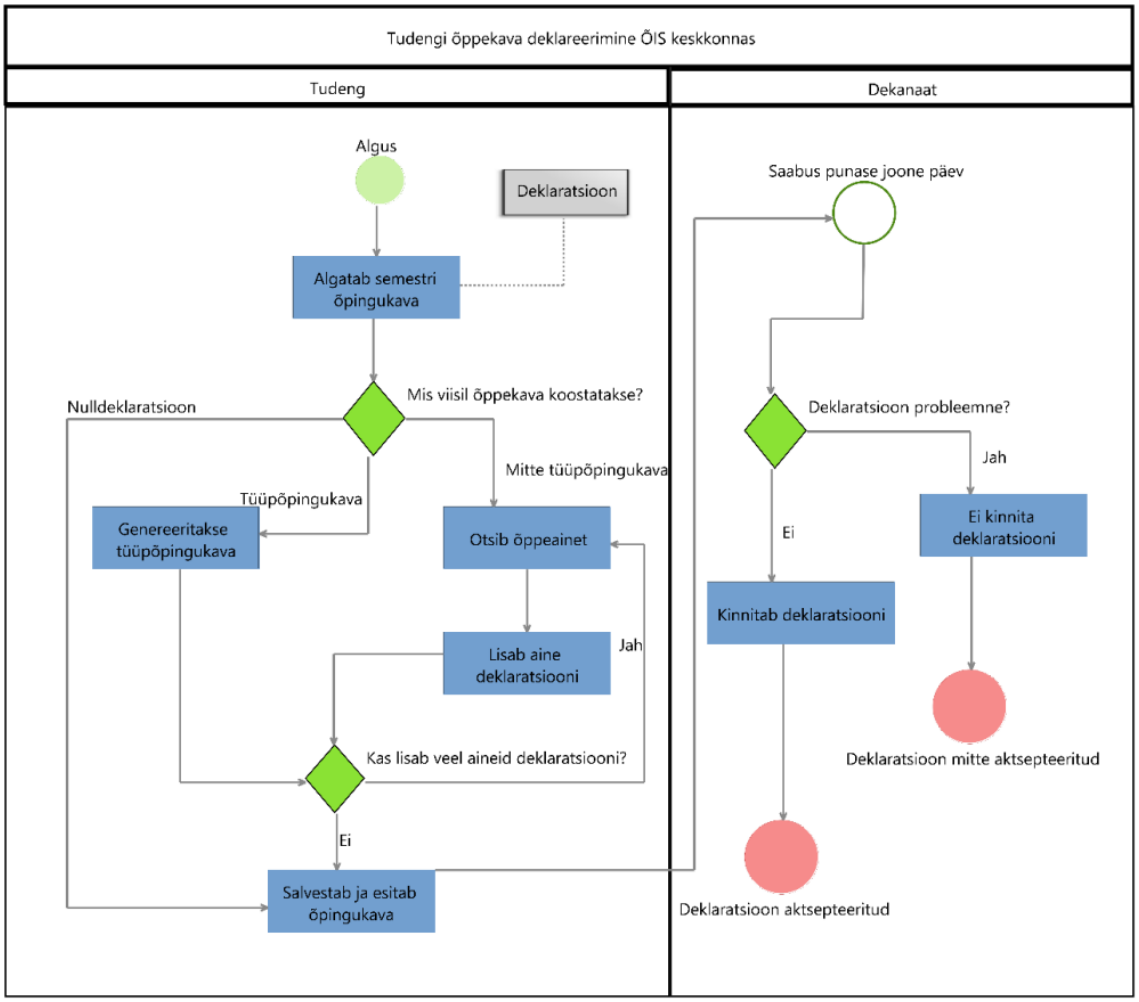
Sirius Web on küll tehniliselt lihtsam kasutusele võtta ning arendustegevuse protsess on oluliselt kiirem, kuid sellel lihtsusel on ka teine külg. Nimelt hetkel domeenimudeli kaardistamiseks on antud vähem vahendeid ning selle kirjeldamisel ei ole võimalik niivõrd süvitsi minna, kui Sirius Desktop puhul. Lisaks pakub Sirius Desktop oluliselt rohkem võimalusi valdkonnamudeli põhjal representatsioonide koostamiseks. See tähendab, et valdkonnamudeli saab võimalikult minimaalsena ja üldisena hoida ning enamus funktsionaalsuse loogikast kandub representatsiooni kirjeldusse.

6.2 Olemasolevad äriprotsesside kaardistamise lahendused

Esmalt tuuakse esile Eclipse Sirius Desktop platvormil arendatud lahenduses koostatud protsess ning seejärel sama Bizagi Modeler'is. Protsessi kaardistamise kogemus annab aluse võrrelda neid lahendusi Sirius Web platvormil arendatud veebipõhise töövahendiga.

6.2.1 Sirius Destkop

Järgnevalt toob autor välja Sirius Desktop platvormil arendatud töövahendiga kirjeldatud protsessi. Töövahend on tudengi bakalaureuse töö tulemus. [1] Tegu on sama protsessiga, TalTech õppekava deklareerimine, mis on punktis 5.4.1 koostatud ka Sirius Web platvormil arendatud lahendusega.



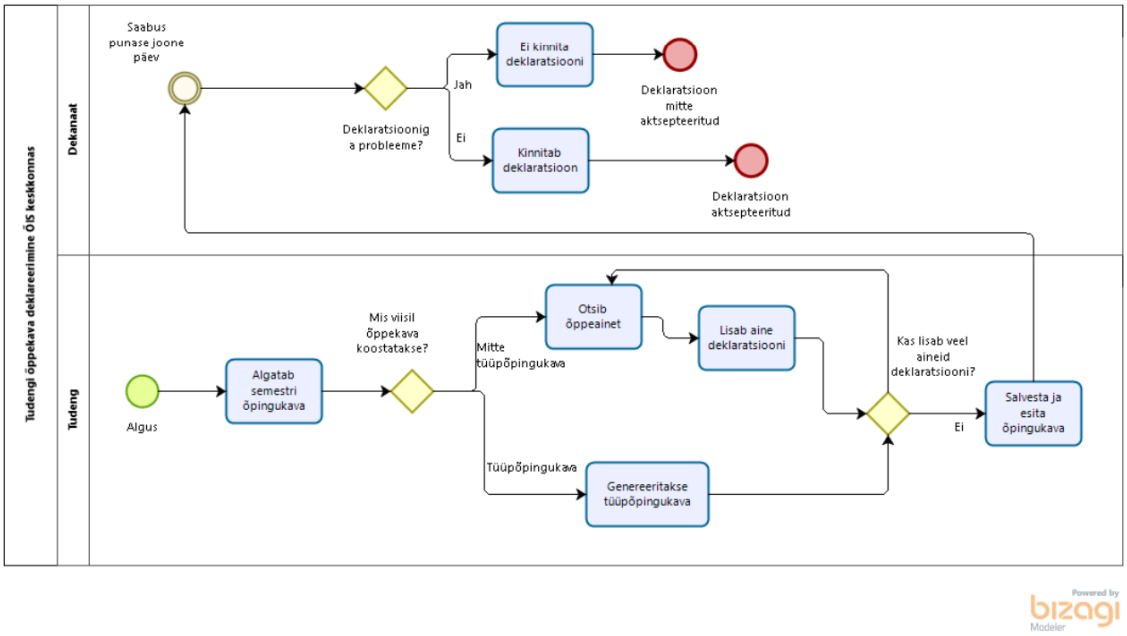
Joonis 28. Koostatud protsessikirjeldus Sirius Desktop'is

Autori bakalaureuse töös valideeriti Sirius Desktopis arendatud tarkvara Bizagi Modeler'ile toetudes. Toodi välja Bizagi Modeler'is kirjeldatud äriprotsess. Seda tehakse

ka nüüd käesolevas töös, et Sirius Web'i lahendust valideerida läbi kahe tarkvara, esmalt Sirius Desktop arendatud modelleerimisvahendi abil, mille funktsionaalsus on omakorda valideeritud Bizagi Modeler äriprotsesside koostamise tarkvara vastu.

6.2.2 Bizagi Modeler

Järgnevalt tuuakse välja Bizagi Modeler'is koostatud TalTech õppekava deklareerimise protsess. Sama protsessi kirjeldus on koostatud nii Bizagi Modeler'is, Sirius Desktop töövahendis kui ka Sirius Web töövahendis.



Joonis 29. Koostatud protsess Bizagi Modeler'is

Bizagi Modeler on Bizagi poolt arendatud üldlevinud äriprotsesside juhtumise tarkvara äriprotsesside modelleerimiseks. Tarkvara peamine eesmärk on võimaldada ettevõtetel dokumenteerida ja arendada äriprotsesse BPMN standardit järgides. Suure populaarsuse ja BPMN standardi funktsionaalsuse olemasolu tõttu valiti Bizagi Modeler valideerimaks käesoleva töös arendatud modelleerimise töövahendi funktsionaalsust. Sarnaselt Sirius Web platvormile lubab ka Bizagi Modeler koosmodelleerimist, kuid see eeldab tarkvara tasulist tellimust. [40]

6.3 Tarkvaraarendamise metoodika

Eclipse Sirius Web võimaldas modelleerimise tarkvara arendamisel tugineda mudelipõhisele arhitektuurile. Nimelt koostati valdkonnaspetsiifiline mudel, esmalt äriprotsesside kaardistamise spetsiifikaga. Valdkonnaspetsiifiline mudel on ühtlasi ka valdkonnaspetsiifilise modelleerimiskeele metamudel. Mudeli põhjal kirjeldati vaade ehk representatsioon. Representatsiooni käivitades on võimalik mudelil põhinevat modelleerimise tarkvara kasutada. Lühidalt sai mudelipõhise arendusmetoodikaga luua valdkonnaspetsiifiline modelleerimise keel ning selle põhjal toimiv tarkvaraline lahendus.

Ökonoomse ja komponendipõhise modelleerimise keele loomise metoodikaga lähtuti põhimõttest, et kasutatakse või taaskasutatakse vaid neid komponente, mida on eesmärgipõhise tarkvara koostamiseks vaja. Seda ka tehti, esmalt loodi äriprotsesside kirjeldamise spetsiifikaga valdkonnamudel, mudelisse kirjeldati minimaalselt vaid need objektid, mida protsessi kirjeldamisel oli tarvis. Ökonoomselt koostatud valdkonnamudeli alusel kirjeldati vajaminevad objektid, mida saab arendatud modelleerimise töövahendiga kasutada. Võib öelda, et ökonoomse modelleerimise keele koostamise vastu eksiti, kui valdkonnamudelis kirjeldati lisaks äriprotsesside kirjeldamise spetsiifikale ka äriobjektide ning äritransaktsioonide ehk väärtusvahetuste spetsiifika. Antud töö praktilise eesmärgi raames on nende kahe valdkonna kaasamine küll liiasus, kuid tuleviku suundasid vaadeldes oli see eelduse tekitamine vastavate funktsionaalsuste realiseerimiseks ettevõtte modelleerimise tarkvaras.

Lisaks eelnevale oli lõputöö üks eesmärkidest luua tarkvara *no-code* metoodikat kasutades. Sellisel viisil tarkvaraarendamist Sirius Web ka võimaldab. Töö käigus loodi tarkvara, mille arendamiseks polnud vaja koodi kirjutamisega programmeerida. Tarkvara programmeeriti hoopis läbi mudelite – kirjeldati valdkonnaspetsiifiline metamudel, millele tuginedes arendati edasine tarkvara loogika. Tarkvara loogika kirjeldati metamudeli olemitest ning nende atribuutidest lähtuvalt. Sirius Web pakub ka terve rida tööriistu, mille abil metamudeli peale on võimalik tarkvara loogikat kirjeldada. Näiteks valdkonnamudeli olemi instantside loomine, instantside muutmine ja kustutamine. Lisaks ka reeglite määramine, et millisel hetkel milliseid olemeid on võimalik luua ning

omavahel ühendada. Kõik funktsionaalsuse kirjeldamine käib läbi veebivormide. Sirius Web lubab ka koodipõhist arendusmetoodikat aga selleks on vaja üles seada lokaalne arenduskeskkond ning läbi selle muudatused veebikeskkonda laadida – oluliselt mahukam ja keerukam viis tarkvara loomiseks. Kuna lõputöö eesmärk oli lähtuda *no-code* arendusmetoodikast, siis lokaalses arenduskeskkonnas arendamine jäi skoobist välja.

6.4 Platvormi ja loodud töölaua analüüs ning edasised suunad

Sirius Web veebipõhise valdkonnaspetsiifiliste modelleerimiskeelte arendamise platvormiga arendati töölaud, millega on võimalik kirjeldada äriprotsesside diagramme – täpsem kirjeldus peatükis 5. Funktsionaalsuse testimiseks tehti punktis 5.4.1 praktilise ülesandena läbi ka ühe äriprotsessi diagrammi koostamine, antud hetkel oli protsessiks TalTech õppekava deklareerimine. Funktsionaalsuse kinnitamiseks tehti sama protsessi kirjeldus läbi ka Bizagi Modeler'is ja autori bakalaureuse töö raames Sirius Desktop platvormil arendatud töövahendiga, täpsemalt kirjeldab punkt 6.2. Lähtudes punktis 5.4.1 ja 6.2 koostatud protsessi diagrammidest on näha, et igal tarkvaral on diagramm omamoodi kujuga, kuid sisuliselt on need kõik samad. Ehk teisisõnu nendest kõikidest koostatud diagrammidelt saab lugeja täpselt samasuguse informatsiooni protsessi kohta omandada. See viitab, et funktsionaalne eesmärk on täidetud ning Sirius Web platvormil arendatud modelleerimise töövahendiga on võimalik äriprotsesse mugavdatud BPMN notatsioonis kirjeldada.

Iga tarkvara kasutamisel on väga oluline osa kasutajamugavusel. Sirius Web platvormil diagrammi koostamine ja muutmine on üldiselt mugav protsess, kuid sellel leidub ka ebamugavaid külgi. Diagrammile uute objektide tekitamine on üsna lihtne, kuid suure arvu objektide korral on kasutajale antav valik samuti suur ja kaootiline. Kaootiline seetõttu, et objektide valikut ei saa kuidagi grupeerida, vaid kasutajale antakse valik ühe nimekirjana, milles on iga objekti loomiseks üks funktsioon. See on üks miinus Sirius Desktop platvormi ees, kus sai töölauda paremini muuta ja funktsioone objektide tekitamiseks grupeerida. Lisaks on murekoht ka see, et diagrammil objektide valimisel, tarkvara ei tuvasta alati täpselt, et kas objekt on valitud või mitte – see tekitab palju arusaamatusi näiteks objektide kustutamisel või objektide omavaheliste ühendusjoonte

tekitamisel. Täiesti puuduolev funktsionaalsus on ka ühendusjoonte liigutamine ja ümberpaigutamine, et diagrammil objekte saaks paremini paigutada – sellele viitab ka joonisel 27 kirjeldatud joonis, kus objektid on võimalikult lahus hoitud, et sirgjooned teisi objekte ei ületaks. Hetkel piirab see mugavat protsessijooniste koostamist. Kasutajamugavuse parandamine on hetkel praeguste vahenditega piiratud ja selle parendamiseks on tarvis oodata Sirius Web platvormi arendajatelt tuleviku versioonides vastavaid uuendusi.

Asjaolu, mis ühtib kasutajamugavusega on kaasmodelleerimine aga praeguses kontekstis vaadeldakse seda pigem uue funktsionaalsusena. Kaasmodelleerimine on Sirius Web üks olulisemaid aspekte, sest see lubab ühe modelleerimise projekti instantsi juures mitmel kasutajal diagramme luua ja muuta. See annab aluse sujuvamaks ja efektiivsemaks koostööks. Koos on võimalik luua ja muuta valdkonnamudelit, kirjeldada selle alusel modelleerimise töövahendi vaade ehk representatsioon ja koos on võimalik ka töövahendit kasutada. See on justkui paarisprogrammeerimine aga programmeeritakse läbi mudelite kirjeldamise.

Tuleviku suunad käesoleva töö tulemuse alusel on esmalt äritransaktsioonide ja äriobjektidele vaadete ehk representatsioonide loomine. Nende kohta käivad valdkonnamudeli täiendused on kirjeldatud punktis 5.2. Luues need representatsioonid, siis ühe diagrammi tüübi asemel on võimalik algatada ja koostada kolm diagrammi tüüpi. See võimendab seni tehtud töövahendi funktsionaalsust kolmekordselt.

Sirius Web esimene versioon ilmus 2020, millest alates on välja antud üheksa täiendavat versiooni, esmailmumisest on keskmiselt igas kvartalis uus versioon ilmunud. Sellest tulenevalt tuleks tulemuste edasi arendamisel kindlasti arvestada uute versioonidega lisandunud funktsionaalsustega. Lisandunud funktsionaalsusi tuleks analüüsida ning otstarbekalt kasutada ja vajadusel tagantjärele implementeerida seni koostatud representatsioonides.

Lõputöö käigus selgus, et Sirius Web platvormil on võimalik luua modelleerimistarkvara, millega saab äriprotsesse kaardistada. Seejuures arendati tarkvara *no-code* ja

mudelipõhist arendusmetoodikat kasutades. Võrdlises Sirius Desktop platvormiga selgus, et päris üheselt ei saanud Desktop platvormil arendatud lahendust Sirius Web'i üle kanda, vaid taaskasutati lahendust selliselt, et võeti Desktop platvormil tehtud lahendus mallina ette ning seejärel kirjeldati valdkonnamudel ning selle representatsioon Sirius Web-is täiesti algusest. Lisaks kahe platvormi võrdluses selgus ka erinevus nende kasutamise keerukuses – nimelt Sirius Web on oluliselt intuitiivsem valdkonnaspetsiifiliste modelleerimiskeelte arendamise platvorm. Kuid intuitiivsust varjutab hetkel asjaolu, et Sirius Web platvormil on oluliselt vähem funktsionaalsust domeenimudeli kaardistamisel kui ka domeenimudeli alusel tarkvara lahenduse kirjeldamisel.

Kokkuvõte

Käesoleva töö eesmärk oli uurida Eclipse Sirius Web platvormi ning arendada mudelipõhise ja *no-code* arendusmetoodikaga veebipõhine modelleerimise tarkvara. Tarkvara arendati Eclipse Sirius Web platvormil, mis toetub Eclipse modelleerimise raamistikule. Valdkonnaspetsiifilise keele koostamisel kasutati ära juba olemasolevat komponendi kirjeldust, komponendi taaskasutamisel lähtuti ökonoomsest lähenemisest – kasutati vaid neid aspekte, mida oli tarvis.

Mudelipõhise arendamise lähenemisega sai no-code arendusmetoodikaga arendada veebipõhine valdkonnaspetsiifiline modelleerimistarkvara. Veebipõhisus seisnes selles, et nii arenduskeskkond kui ka lõpp-produkti kasutamine on viidud Obeo Cloud pilveteenusesse – tegu on SaaS ehk tarkvara teenusena, ehk infrastruktuur, arenduskeskkond ja tarkvara käivitamise keskkond on teenusepakkuja poolt tagatud.

Töös realiseeriti veebipõhiselt kasutatav modelleerimise töövahend, millega on võimalik kaardistada äriprotsesse BPMN notatsioonis. Valdkonnamudeli täiendustega tagati funktsionaalsuse laiendatavus äritransaktsioonide ja äriobjektide kaardistamiseks.

Tarkvara realiseerimise kogemuse põhjal analüüsiti ja koostati hinnang arendusmetoodikale, võrreldi Sirius Web ja Sirius Desktop platvorme, seejuures võrreldi koostatud modelleerimise töövahendit teiste sama eesmärki täitvate lahendustega. Analüüsi käigus pakuti välja ka saavutatud tulemuste edasiarendamise eesmärgid.

Lõputöös püstitatud eesmärgid said täidetud. Hinnati Sirius Web platvormi ning kasutatud arendusmetoodika sobivust.

Kasutatud kirjandus

[1] A.-M. Kink, „Äriprotsesside mudelipõhine arendamine Eclipse Sirius platvormil," Tallinna Tehnikaülikool, Tallinn, 2019.

[2] Y. Sirotkina, „Ärianalüüsi toetava valdkonnaspetsiifilise modelleerimiskeele prototüübi loomine," Tallinna Tehnikaülikool, Tallinn, 2020.

[3] A. Bakalkin, „Äriobjektide modelleerimist toetava valdkonnaspetsiifilise modelleerimiskeele prototüübi loomine," Tallinna Tehnikaülikool, Tallinn, 2020.

[4] Eclipse, „Sirius Web," Eclipse, [Võrgumaterjal]. Available: https://www.eclipse.org/sirius/sirius-web.html#. [Kasutatud 28 03 2022].

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

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

[7] F. Ulrich, „Multi-Perspective Enterprise Modelling: Background and Terminological Foundation," Universität Duisburg-Essen, Essen, 2011.

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

[9] 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 20 03 2022].

[10] A. M. M. Dr. Shahid Nazir Bhatti, „Model Driven Architecture," 04 2015. [Võrgumaterjal]. Available: https://www.researchgate.net/publication/274916541_Model_Driven_Architecture. [Kasutatud 29 03 2022].

[11] Techopedia, „Techopedia,“ Techopedia, [Võrgumaterjal]. Available: https://www.techopedia.com/definition/28057/model-driven-architecture-mda. [Kasutatud 20 02 2022].

[12] H. A. Proper, R. Winter ja S. Aier, „Architectural Coordination of Enterprise Transformation,“ Springer International Publishing AG, 2017, p. 278.

[13] H. A. Proper, R. Winter ja S. Aier, „Architectural Coordination of Enterprise Transformation,“ Springer International Publishing AG, 2017, p. 138.

[14] H. A. Proper, R. Winter ja S. Aier, „Architectural Coordination of Enterprise Transformation,“ Springer International Publishing AG, 2017, p. 233.

[15] Azure, „What is SaaS?,“ Microsoft, [Võrgumaterjal]. Available: https://azure.microsoft.com/en-us/resources/cloud-computing-dictionary/what-is-saas/. [Kasutatud 19 11 2022].

[16] IBM, „What is SaaS – software-as-a-service?,“ IBM, [Võrgumaterjal]. Available: https://www.ibm.com/topics/saas. [Kasutatud 19 11 2022].

[17] E. Ozan, „A Novel Browser-based No-code Machine Learning Application Development Tool,“ East Caroline University, Carolina, 2021.

[18] Webflow, „What is no-code development?,“ Webflow, [Võrgumaterjal]. Available: https://webflow.com/no-code. [Kasutatud 15 11 2022].

[19] Voiceflow, „Voiceflow,“ Voiceflow, [Võrgumaterjal]. Available: https://www.voiceflow.com/. [Kasutatud 15 11 2022].

[20] Zapier, „Zapier,“ Zapier, [Võrgumaterjal]. Available: https://zapier.com/. [Kasutatud 15 11 2022].

[21] Shopify, „Shopify,“ Shopify, [Võrgumaterjal]. Available: https://www.shopify.com/. [Kasutatud 15 11 2022].

[22] Fintory, „The pros and cons of no-code software development. The simple way to code?,“ Fintory, [Võrgumaterjal]. Available: https://www.fintory.com/en/blog/no-code-software-development-the-simple-way-to-code. [Kasutatud 15 11 2022].

[23] Eclipse, „About the Eclipse Project,“ Eclipse, [Võrgumaterjal]. Available: https://www.eclipse.org/eclipse/. [Kasutatud 27 11 2022].

[24] Eclipse, „About the Eclipse Foundation,“ Eclipse, [Võrgumaterjal]. Available: https://www.eclipse.org/org/. [Kasutatud 13 04 2022].

[25] Eclipse, „Eclipse Modeling Framework (EMF),“ Eclipse, [Võrgumaterjal]. Available: https://www.eclipse.org/modeling/emf/. [Kasutatud 13 04 2022].

[26] Eclipse, „What is Sirius?,“ Eclipse, [Võrgumaterjal]. Available: https://www.eclipse.org/sirius/overview.html. [Kasutatud 29 11 2022].

[27] Eclipse, „What can you do with Sirius?,“ Eclipse, [Võrgumaterjal]. Available: https://www.eclipse.org/sirius/gallery.html. [Kasutatud 29 11 2022].

[28] Obeo, „The easiest way to define your own modeling tools!,“ Obeo, [Võrgumaterjal]. Available: https://www.obeodesigner.com/en/product. [Kasutatud 18 12 2022].

[29] Obeo, „Key Features,“ Obeo, [Võrgumaterjal]. Available: https://www.obeodesigner.com/en/product/key-features. [Kasutatud 18 12 2022].

[30] Eclipse, „Sirius Desktop,“ Eclipse, [Võrgumaterjal]. Available: https://www.eclipse.org/sirius/. [Kasutatud 28 03 2022].

[31] A. Wichmann, R. Maschotta, F. Bedini ja A. Zimmermann, „Model-Driven Development of UML-Based Domain-Specific Languages for System Architecture Variants,“ Technische Universitat Ilmenau, Ilmenau, 2019.

[32] Obeo, „Obeo Studio Documentation,“ Obeo, [Võrgumaterjal]. Available: https://docs.obeostudio.com/2022.01.0/help_center.html#_user_manual. [Kasutatud 22 10 2022].

[33] A. Lynch, „What is BPMN - Definition, Elements and Purpose,“ Edraw, 18 2 2022. [Võrgumaterjal]. Available: https://www.edrawsoft.com/what-is-bpmn.html. [Kasutatud 12 11 2022].

[34] Eclipse , „GitHub,“ [Võrgumaterjal]. Available: https://github.com/eclipse-sirius/sirius-web. [Kasutatud 01 04 2022].

[35] J. Helming, „Modeling Languages,“ 7 1 2020. [Võrgumaterjal]. Available: https://modeling-languages.com/web-based-modeling-with-eclipse/. [Kasutatud 01 04 2022].

[36] Eclipse, „Sirius Documentation,“ Eclipse, [Võrgumaterjal]. Available: https://www.eclipse.org/sirius/doc/. [Kasutatud 13 04 2022].

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

[38] Lucidshart, „What is BPMN?,“ Lucidchart, [Võrgumaterjal]. Available: https://www.lucidchart.com/pages/bpmn. [Kasutatud 22 10 2022].

[39] Visual Paradigm, „What is BPMN?,“ Visual Paradigm, [Võrgumaterjal]. Available: https://www.visual-paradigm.com/guide/bpmn/what-is-bpmn/. [Kasutatud 22 10 2022].

[40] Bizagi, „Bizagi Modeler,“ [Võrgumaterjal]. Available: https://www.bizagi.com/en/platform/modeler. [Kasutatud 26 12 2022].

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

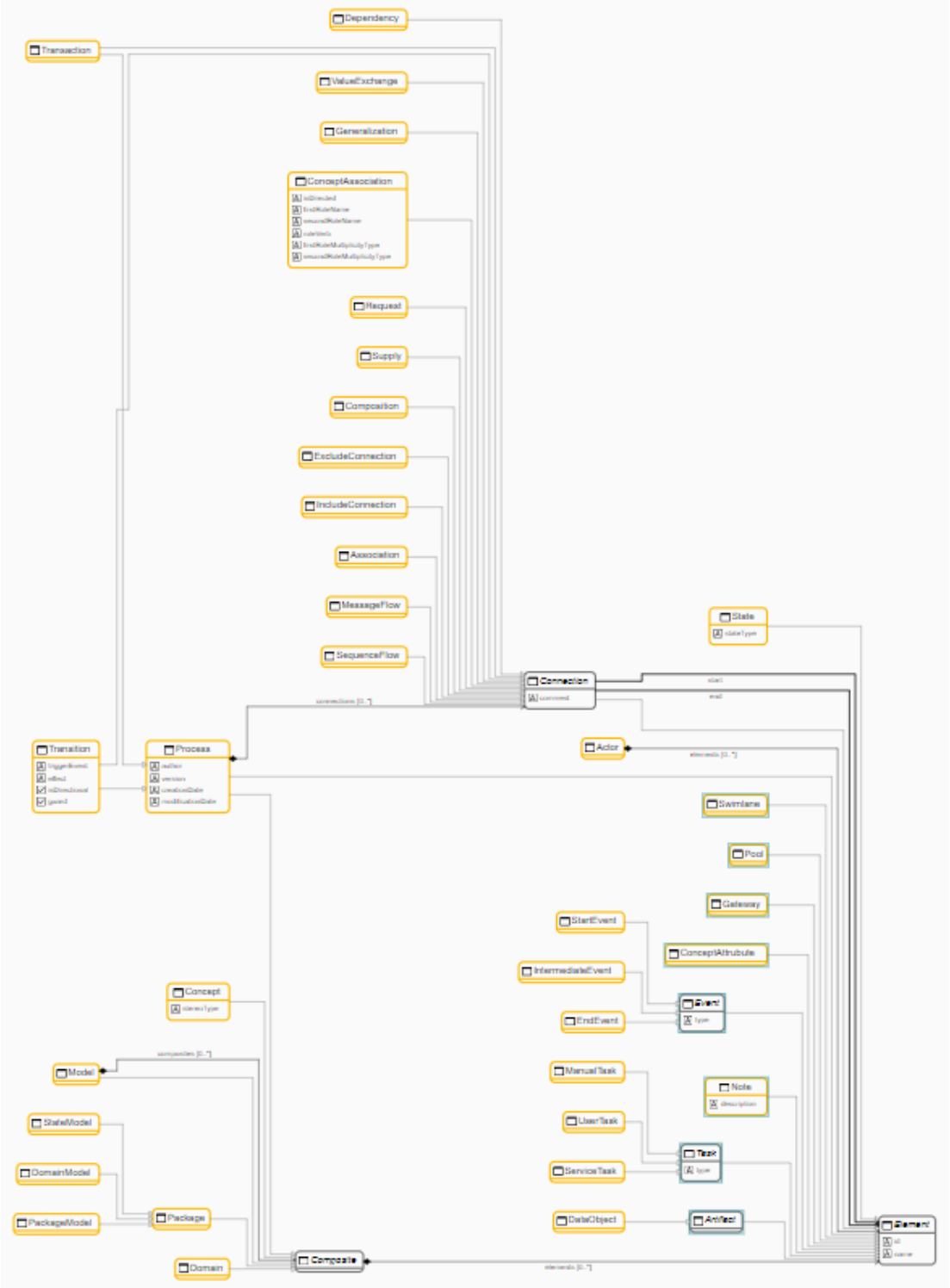
Mina, Ain-Martin Kink

1. Annan Tallinna Tehnikaülikoolile tasuta loa (lihtlitsentsi) enda loodud teose „Veebipõhise modelleerimistarkvara loomine valdkonnaspetsiifiliste modelleerimiskeelte platvormil“, mille juhendaja on Mart Roost
   1.1. reprodutseerimiseks lõputöö säilitamise ja elektroonse avaldamise eesmärgil, sh Tallinna Tehnikaülikooli raamatukogu digikogusse lisamise eesmärgil kuni autoriõiguse kehtivuse tähtaja lõppemiseni;
   1.2. üldsusele kättesaadavaks tegemiseks Tallinna Tehnikaülikooli veebikeskkonna kaudu, sealhulgas Tallinna Tehnikaülikooli raamatukogu digikogu kaudu kuni autoriõiguse kehtivuse tähtaja lõppemiseni.
2. Olen teadlik, et käesoleva lihtlitsentsi punktis 1 nimetatud õigused jäävad alles ka autorile.
3. Kinnitan, et lihtlitsentsi andmisega ei rikuta teiste isikute intellektuaalomandi ega isikuandmete kaitse seadusest ning muudest õigusaktidest tulenevaid õigusi.

31.12.2022

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

Lisa 2 – Täiendatud äriprotsesside, äriobjektide ja väärtusvahetuste valdkonnamudel



Joonis 30. Äriprotsesside, äriobjektide ja väärtusvahetuste valdkonnamudel