TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond

Getter Keerd 177981IABM

SOBIVA SKALEERITAVA PAINDMETOODIKA VALIMINE SUURELE ORGANISATSIOONILE ETTEVÕTTE X NÄITEL

Magistritöö

Juhendaja: Erki Eessaar
PhD

Tallinn 2020

Autorideklaratsioon

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

Autor: Getter Keerd

11.05.2020

Annotatsioon

Skaleeritav paindmetoodika on mõeldud kasutamiseks suurtes organisatsioonides ja projektides. Käesoleva magistritöö peamiseks eesmärgiks on parima skaleeritava paindmetoodika valimine konkreetsele ettevõttele (ettevõte X). Lõputöö teiseks eesmärgiks on kirjanduse uurimise käigus luua mõnedele skaleeritavatele paindmetoodikatele metamudelid, mis kirjeldavad ära vastava metoodika põhimõisted ja nendevahelised seosed. Metamudelite koostamise ja nende võrdlemise käigus kogutud teadmised aitavad teha esimeseks eesmärgiks olevat valikut.

Parima paindmetoodika valimiseks kasutatakse Saaty analüütiliste hierarhiate meetodit, mille käigus luuakse otsustusmudel arvestades ettevõtte konteksti. Otsustusmudeli alusel tehtud otsusele tehakse tundlikkuse analüüs, mille sisuks on tekitada mudeli struktuuris põhjendatud muudatusi ning vaadelda uut tekkinud lõpptulemust. Metamudelite loomiseks kirjeldab autor modelleerimise põhimõtted. Mudelid esitatakse UML klassidiagrammidena ning nende sõnaliste kirjeldustena. Koostatud metamudelid on sisendiks Saaty otsustusmudelisse alternatiivide valimisele.

Magistritöö raames uuriti lähemalt viite skaleeritavat paindmetoodikat, mille kohta loodi metamudelid. Metamudelite loomise protsessi üks osa oli mudelite valideerimine erinevate osapooltega. Lisaks võrreldi metamudeleid omavahel ontoloogilise analüüsi meetodil. Modelleerimise tulemusel välistas autor otsustusmudelisse alternatiivide valikul kaks metoodikat. Seega sobivaid alternatiive oli kolm – *Large-Scale Scrum Huge*, *Scaled Agile Framework* ja *Disciplined Agile Delivery*. Otsustusmudeli alusel osutus võitjaks *Scaled Agile Framework.* Otsustusmudelile tehtud tundlikkuse analüüs lõpptulemust ei muutnud. Loodud otsustusmudelit saavad (küllap kontekstist tulenevate kohandustega) kasutada ka teised suured organisatsioonid, mis peavad sellist valikut tegema.

Lõputöö on kirjutatud eesti keeles ning sisaldab teksti 93 leheküljel, 8 peatükki, 58 joonist, 7 tabelit.

Abstract

Choosing a Suitable Scaled Agile Methodology for a Large Organization in the Example of Company X

The main aim of this Master's thesis is to choose the best scalable agile methodology for a specific company (company X). The second goal of this thesis is to develop metamodels of some scaled agile methodologies describing the most important elements of each methodology and the relations between the elements. The knowledge that is acquired in the metamodelling process helps us to make the choice of the first goal.

Agility is an important topic in software engineering. If done right it can be very beneficial to organizations in achieving their goals. However, using agile methodologies in large organizations can be complicated due to the significant number of people involved and teams who are often geographically dispersed. Triggered by the need to use agile in large organizations, more and more scaled agile methodologies have been created that mostly differ in scale and number of teams. [1] These methodologies foresee additional artifacts, ceremonies, and roles to facilitate cooperation and coordination between teams. The methodologies have additional levels of management and therefore also additional level of complexity. In this thesis a selection between different scaled agile methodologies in case of a specific context (company X) is conducted by using the Saaty's Analytical Hierarchy Process (AHP). The process is a structured method for making decisions. The process helps decision makers to make objective decisions instead of subjective ones. The outcome of applying this method is a decision model based on the specific context (in case of this thesis company X). In addition, a sensitivity analysis to this decision model is done. It means adjusting the structure of the model and observing the new outcome. [2]

For the creation of metamodels, the author describes the principles of modeling. Metamodels will be presented as UML class diagrams that are accompanied with textual explanations. These metamodels are an input for choosing alternatives to Saaty's decision making process.

During writing the master's thesis, five scaled agile methodologies were studied and metamodels were created for all of these methodologies – *Scaled Agile Framework* (SAFe), *Nexus*, *Large-Scale Scrum* (LeSS), *Large-Scale Scrum Huge* (LeSS *Huge*), and *Disciplined Agile Delivery* (DAD). The author describes the metamodelling approach based on the principles for the creation of metamodels and identifies four views that must be created for each methodology. 17 different diagrams were created as a result of the modeling. The first three views of LeSS methodology also apply to LeSS *Huge*, only one additional view was created specifically for LeSS *Huge*. Validating the metamodels with different parties is a part of the modeling process to harmonize/adjust the interpretation. It was done in the process. Three metamodels (of the methodologies that according to the initial assessment are the best suitable for the company X) were also compared with each other by using an ontological evaluation method [3] to find the main differences and similarities of their elements.

As a result of the modeling, the author chose for the decision model three scaled agile methodologies as alternatives. These three were SAFe, LeSS *Huge*, and DAD. The criteria of the decision model were chosen based on their relevance in the choice of methodology and the context of the specific company (company X). The result of the decision-making process was choosing SAFe. Sensitivity analysis was made for all the criteria and the result remained the same.

This decision model can be used (with contextual adjustments) by other large organizations that have to make a similar choice. Provided metamodels may be interesting to people who want to learn more about scaled agile methodologies.

The thesis is in Estonian and contains 93 pages of text, 8 chapters, 58 figures, 7 tables.

Lühendite ja mõistete sõnastik

AHP	*Analytic hierarchy process* Analüütilise hierarhiate protsess on meetod tegemaks otsuseid võimalikult objektiivselt. Meetodi väljatöötaja nime järgi tuntakse seda ka Saaty meetodina [2].
ART	*Agile Release Train* Agiilne väljalaske rong* on SAFe metoodika programmi taseme peamine organisatoorne üksus, mis seob inimesed ja nende töö ühisesse programmi visiooni, missiooni ja soovilogisse [4].
Backlog	Soovilogi ehk tegemata tööde (nõuete) prioriteediloend [5] [6].
CM	*consistency measure* Kooskõla määr [7]
DAD	*Disciplined Agile Delivery* Skaleeritav paindmetoodika
DevOps	Tarkvaraarenduse kultuur, mis ühendab omavahel tarvaraarenduse ja -operatsioonid [8].
DoD	*Definition of Done* „Valmis“ definitsioon* on loetelu kriteeriumitest, mis peavad olema täidetud enne kui realiseeritud nõue loetakse tehtuks [9].
DSDM	*Dynamic Software Development Method*
FDD	*Feature-Driven Development*
Feature	Erisus
Feature team	Erisuse tiim, millel on vajalikud teadmised ja oskused tegemaks algusest lõpuni kliendikeskset uut funktsionaalsust (*Feature*), arendades selleks vajalikke komponente [10].
ISO	*International Organization for Standardization*
Kanban	Arendusmetoodika *Kanban* on meetod, mis võimaldab tiimil visualiseerida töövoogu ja kehtestada tööde teatud staatustes oleku ehk WIP (*Work in Progress*) piirid, mõõtes läbilaskevõimet ja parendades pidevalt arendusprotsesse [4].
LeSS	*Large-Scale Scrum*

	Skaleeritav paindmetoodika
MOF	Meta-Object Facility Neljatasemelise arhitektuur, mille struktuur, tähendus ja omavahelised seosed on defineeritud [11].
Nexus	Skaleeritav paindmetoodika
NIT	*Nexus Integration Team* *Nexuse* integreeritud tiim, mille eesmärk on edendada ja juhendada *Nexuse* rakendamist nii Scrum tiimides kui kogu organisatsioonis sisemiselt ning tagada võimalikud eeldused, et tiimid suudaksid ise integreeritud juurdekasvu anda [12].
OCL	*Object Constraint Language* Objektikitsenduskeel
OMG	*Object Management Group* Infotehnoloogia valdkonna standardite koostamisega tegelev organisatsioon.
OWL	*Web Ontology Language*
PO	*Product Owner* Tooteomanik
RAGE	*Recipes for Agile Governance in the Enterprise* Skaleeritav paindmetoodika
SAFe	*Scaled Agile Framework* Skaleeritav paindmetoodika
Scrum	Paindmetoodika
Spotify	Skaleeritav paindmetoodika
UML	*Unified Modeling Language* Unifitseeritud modelleerimiskeel. Standardiseeritud üldotstarbeline visuaalne modelleerimiskeel [13].
UX	*User Experience* Kasutatavus
WIP	*Work in Progress*
XP	*Extreme Programming* Ekstreemprogrammeerimine. Paindmetoodika.

Sisukord

Jooniste loetelu

Tabelite loetelu

1 Sissejuhatus

Käesolev lõputöö keskendub erinevate skaleeritavate paindmetoodikate uurimisele, mis sobivad kasutamiseks suurtele organisatsioonidele. Agiilsus on töö kirjutamise ajal (2020. aasta kevad) populaarne teema. Selle põhimõtete järgimine tõotab mitmeid eeliseid nagu näiteks muutustele kiiresti reageerimine, suuremat tiimi tootlikkust ja arenduse paremat läbipaistvust. Vajadus agiilsete metoodikate laiendamiseks suuremale tootearendusele on viinud mitmete skaleeritavate paindmetoodikate tekkeni, millest mõni on leidnud laialdast kasutust, osa neist on aga üsna tundmatud. [1] Skaleeritav paindmetoodika on raamistik, mis on mõeldud suure organisatsiooni suure arvu tiimide abil tootearenduse organiseerimiseks [14]. Suures organisatsioonis töötab IT arenduses üle viiekümne töötaja ning arendustiime on vähemalt kaheksa. Järgnevalt annab autor ülevaate töö taustast ja probleemidest, tuues lisaks välja töö peamised eesmärgid ja nende saavutamiseks kasutatav metoodika. Samuti antakse ülevaade lõputöö ülesehitusest.

1.1 Taust ja probleem

Tänapäeva IT arendusega tegelevates organisatsioonides on sõna „agiilne“ üks võtmesõnadest – kõik tahavad olla paindlikud ja kasvada ning soovivad, et kasutatav arendusmetoodika neid eesmärke toetaks [15]. Mõningad populaarsemad näited suurtele organisatsioonidele mõeldud paindmetoodikate kohta on *Scaled Agile Framework* (SAFe) ja *Nexus*, kuid neid on veel mitmeid [16]. Need metoodikad on raamistikud, sest annavad kasutajale vabaduse metoodikat oma vajadustest lähtuvalt kohandada. Kõik need raamistikud erinevad teineteisest rohkemal või vähemal määral, kuid kuidas valida sobivaim paindmetoodika, kui ettevõte on sellise valiku ees? Millised on need kriteeriumid, mida tuleb metoodika valikul arvestada ja millised on nende raamistike peamised erinevused? Käesolev lõputöö eesmärk on leida nendele küsimustele vastused. Antud lõputöö teema valikut inspireeris see, et autori töökohal viiakse läbi SAFe raamistiku juurutusprotsess ja autoril tekkis huvi, kas antud raamistiku valik on antud ettevõttele parim või oleks mõni alternatiiv sobivam. Töö autor on ettevõttes ise tooteomaniku (*Product Owner*) rollis.

1.2 Ülesandepüstitus ja eesmärgid

Käesolevas lõputöös on autor püstitanud kaks suuremat eesmärki. Töö peamiseks eesmärgiks on parima skaleeritava paindmetoodika valimine konkreetse ettevõtte (ettevõtte X) konteksti arvestades. Kontekstiga, antud juhul ettevõttega, arvestamine on IT valdkonna rakenduslikes uuringutes väga oluline [17]. Lisaks sobiva metoodika valikule konkreetse ettevõtte konteksti arvestades on töö tulemiks ka otsustusmudel, mida võib olla võimalik (küllap kontekstist tulenevate kohandustega) kasutada ka teistel suurtel organisatsioonidel, mis peavad sellist valikut tegema. Käesolevas lõputöös ei avalikustata konfidentsiaalsuse eesmärgil ettevõtte nime, mille näitel koostatud otsustusmudelit kasutatakse. Samas ollakse konteksti kirjeldamisel piisavalt detailne, mis puhul organisatsiooni nime välja ütlemine siinkohal ei olegi vajalik. Esmase kirjanduse uurimise käigus tuvastati, et ka varem on ilmunud mitmeid teadusartikleid, kus mõne meetodi abil on valitud sobivat arendusmetoodikat (alternatiivideks Scrum, ekstreemprogrammeerimine jt) [18] [19] [20]. Samas aga ei leidu sellist uuringut, kus oleks tehtud ja kasutatud otsustusmudelit ettevõttele sobiva skaleeritava paindmetoodika valikuks. Loodud otsustusmudeli alusel tehtud otsusele tehakse tundlikkuse analüüs, mille sisuks on tekitada mudeli struktuuris põhjendatud muudatusi ning vaadelda uut tekkinud lõpptulemust [21].

Lõputöö teiseks eesmärgiks on kirjanduse uurimise käigus luua erinevatele paindmetoodikatele metamudelid, mis kirjeldavad ära vastava metoodika mõisted ja nendevahelised seosed. Metamudelid esitatakse UML klassidiagrammidena ja nende sõnaliste selgitustena. Seda tehakse töö esimeses osas, sest koostatud metamudelid on sisendiks otsustusmudelisse alternatiivide ja kriteeriumite valimisel ning nende koostamisel saadud teadmised aitavad metoodikaid ka hiljem võrrelda. Kui kirjanduse uurimise ja analüüsimise käigus selgub, et mõni paindmetoodika oma eripäradega ei sobi antud ettevõtte konteksti arvestades otsustusmudeli alternatiivide valikusse, siis selle metoodika kohta metamudelit looma ei hakata, sest selle loomise peamine eesmärk on võimalikest alternatiividest parema arusaama tekitamine. Autor on leidnud metamudeleid metoodikate kohta (nt Scrum, ekstreemprogrammeerimine), kuid mitte skaleeritavate paindmetoodikate kohta [22] [23] [24].

1.3 Metoodika

Lõputöö tegemine põhineb disainiteaduse (*design science)* meetodil, mille idee on arendada artefakti ehk tehist ja seejärel seda valideerida. Töö tulemusena luuakse uut teadmist, mitte ei kasutata ainult olemasolevaid teadmisi mingi tehise valmis tegemiseks nagu näeb ette rutiinne kavandamine (*routine design*). [25] Kõigepealt otsib autor infot skaleeritavate paindmetoodikate kohta ning hindab iga leitu korral, kas sellele on mingeid omadusi, mis juba eos välistavad selle ettevõttes kasutamise. Selle kontrolli läbinud metoodikatele loob autor metamudelid, võrdleb neid omavahel ning loob otsustusmudeli, mille alternatiivide ja kriteeriumite valikul arvestatakse sisendina metamudelite võrdlemisel leitud erinevusi ja ettevõtte kontekstist tulenevaid piiranguid. Koostatud otsustusmudeliga leitakse konkreetse ettevõtte näitel sellele sobiv metoodika.

Paindmetoodikate metamudelid esitatakse UML (*Unified Modeling Language)* klassidiagrammidena ja nende sõnaliste selgitustena. UML on standardiseeritud üldotstarbeline visuaalne modelleerimiskeel, mida võib muuhulgas kasutada suurte ja keerukate süsteemide modelleerimisel. [13] Üldotstarbelisus viitab keele erinevate kasutusvaldkondade rohkusele ja visuaalsus tähendab, et mudelid esitatakse visuaalselt erinevat tüüpi diagramme ehk skeeme kasutades.

Loodavates metamudelites kirjeldatakse klassidena ära metoodikate poolt ettenähtavad tegutsejad, tehised ja tegevused ning klasside seostena nende omavahelised seosed. Sellele lisanduvad klasside ja seoste tekstilised kirjeldused [26]. „Meta“ tähendab tase kõrgemat, üldisemat vaatepunkti [27] ning antud olukorras tähendab see, et ei kirjeldata mitte konkreetse metoodika rakendamist konkreetses olukorras, vaid selle metoodika üldiseid ehitusplokke, nende omavahelisi seoseid ja metoodika elementidele kehtivaid piiranguid (näiteks nagu UMLi metamudel kirjeldab UML keele ehitusplokid ja põhimõtted). Formaalsemalt väljendudes on modelleerimise tulemiks metamudelid, mis vastavad MOF (*Meta-Object Facility*) arhitektuuris tasemele M2. MOF on disainitud neljatasemelise arhitektuurina, mille struktuur, tähendus ja omavahelised seosed on defineeritud. Tasemel M2 kirjeldatakse ära keel (näiteks UML), mille abil saab esitada M1 kihi elemendid (näiteks UML abil koostatud mudelite elemendid). [11]

Autor lähtub modelleerimisel headest praktikatest (sh mustrid [28]) ja kontseptuaalsete andmemudelite loomise põhimõtetest, üritades samal ajal vältida halbu praktikaid ning

antimustreid ehk nurimustreid [29]. Paralleelselt otsitakse ja kirjeldatakse olemitüüpe, seosetüüpe ja atribuute [30] [31] – antud töö kontekstis võib neid ka nimetada metaklassideks, nendevahelisteks seosteks ja metaklasside atribuutideks. Reeglid, millest autor modelleerimisel lähtub, on detailsemalt kirjeldatud töö põhiosas (vt peatükk 4). Neid põhimõtteid järgides tagab töö autor mudelite kvaliteedi ja arusaadavuse kõikidele üldiseid modelleerimise põhimõtteid tundvatele osapooltele. Lisaks viib autor läbi mudelite ühise ülevaatuse nimetatud paindmetoodikaid tundvate isikutega, kes metamudeli vaatlemisel ja analüüsimisel saavad anda hinnangu mudeli korrektsusele ja tõlgendatavusele. Lisaks tehakse ülevaatus ka IT-valdkonna inimestega, kes antud metoodikaid ei tunne. See annab objektiivsema tagasiside, kui hästi mudelid metoodikat kirjeldavad. Ülevaatuse õnnestumise üheks eelduseks on mudelite hea ja arusaadav esitus.

Metamudelite loomiseks kasutatakse *Enterprise Architect CASE* modelleerimisvahendit, mis on mõeldud erinevate süsteemide, tarkvara, protsesside ja arhitektuuride modelleerimiseks ja kus muuhulgas saab luua UML mudeleid [32]. Kuna lõputöö autoril on varasem kogemus antud tarkvaraga modelleerimisel, siis mudelite tegemiseks on autor teinud oma valiku just selle tarkvara kasuks.

Töö ühe tulemusena luuakse metamudelid. See mõiste on kasutusel tarkvarakeelte arendamisel (vt nt UML metamudel) [33] ning metamudeleid kasutatakse seal, et kirjeldada keele abstraktset süntaksi ehk lihtsamalt öeldes keele ehitusplokke ja nendele kehtivaid reegleid. Sellele mõistele lähedasteks mõisteteks on valdkonnamudelid ja ontoloogiad. Võib öelda nii, et metamudelitel, ontoloogiatel ja valdkonnamudelitel on erinev otsese kasutamise eesmärk, neil võib olla erinev täpsusaste, nende loomiseks võidakse kasutada erinevaid keeli (metamudelite ja valdkonnamudelite loomiseks näiteks UMLi ja ontoloogiate loomiseks OWL keelt [34]) ja nendest räägivad erinevad teadlaste kogukonnad. Samas on neil kõigil ühine sügavam eesmärk – aidata laiemal inimeste ringil kokku leppida mõisted, millest mida mingi „asja“ kirjeldamiseks kasutatakse, kirjeldada seda „asja“ võimalikult täpselt ja arusaadavalt ning aidata sellega laiemal inimeste ringil kirjeldatavast „asjast“ paremini ja sarnasemalt aru saada. [35] Teiste sõnadega esitavad need kõik objektikeskset esitusviisi kasutades teadmisi modelleeritava valdkonna kohta. UMLi võib kasutada ontoloogiate esitamiseks [35] ja seega poleks väga vale nimetada töös loodavaid mudeleid ka „ontoloogiaks“ või „valdkonnamudeliks“ kui järjekindluse huvides kasutatakse sõna „metamudel“. Mingis mõttes on ka arendusmetoodika ühine „keel“ (näiteks kui öelda „sprint“, siis kõik arenduse osalised võiksid ühtemoodi aru

saada, mida see tähendab ja nendelt nõuab), mida organisatsiooni huvitatud osapooled peavad mõistma ja järgima ning metamudel kirjeldab selle keele abstraktse süntaksi.

Paindmetoodikate erinevuste leidmiseks koostatud metamudelite abil võrreldakse mudeleid teineteisega kasutades ontoloogilise analüüsi meetodi põhimõtteid [30] [3]. Analüüsimaks loodud metamudeleid valitakse "ontoloogiaks" üks metoodika ja selle metamudel, ning võrreldakse sellega teisi metamodelleeritud metoodikaid.

Ettevõttele sobiva paindmetoodika valiku tegemiseks kasutab autor Saaty analüütiliste hierarhiate meetodit (AHP). Selleks luuakse otsustusmudel ja konkreetse ettevõtte konteksti arvestades rakendatakse loodud otsustusmudelit. [2] Saaty meetodi looja on USA matemaatik Thomas L. Saaty ja meetod töötati välja 1970ndatel aastatel. „Saaty meetod püüab otsustussegaduses ja suures subjektivismis korda luua, nii et pärast otsustuse tegemist saaks olla veendunud nii oma otsustuse õigsuses kui ka osata veenda teisi, miks otsustati just nii ja mitte teisti.“ [36] Antud meetod valiti, sest see on laialdaselt kasutusel, selle kohta on palju materjale, seda õpetatakse ülikoolis ja autor on sellega tuttav. Saaty meetodi kasutamisel tuleb kõigepealt defineerida probleem ja eesmärgid. Seejärel tuleb tuvastada võimalikud kriteeriumid ehk mõjutegurid ning alternatiivid ehk valikud, mida tuleb paarikaupa iga kriteeriumi suhtes võrrelda. Samuti tuleb üksteisega paarikaupa võrrelda ka kriteeriumeid endid. Kriteeriumid ja alternatiivid tuleb korrastada mitmetasemelisse hierarhilisse struktuuri. Puu juureks on eesmärgid, mille all on mõjutegurid ja alammõjutegurid. Puu lehtedeks on alternatiivid. Otsustuse tulemuse leidmisele järgneb tulemuste analüüs ja tulemuste tundlikkuse analüüs. [36] Hierarhilise struktuuri koostamiseks ehk otsustusmudeli loomiseks ja arvutuste automatiseerimiseks kasutatakse töös tasuta veebipõhist vabavara *Web-Hipre* [37]. Saaty otsustusmudelisse kriteeriumite valimiseks uuritakse valdkonna kirjandust, samuti arvestatakse kriteeriumite valikul ettevõtte kontekstist tulenevate võimalike piirangutega [38]. Saaty meetodist on käesolevas lõputöös lühiülevaade töö põhiosas (vt jaotis 6.1).

1.4 Ülevaade tööst

Käesolev lõputöö koosneb kaheksast peatükist. Töö esimeses pooles tehakse ülevaade varasematest lõputöö teemaga haakuvatest uuringutest (vt peatükk 2). Seejärel kirjeldatakse ettevõtet, mille vajadustest lähtuvalt võimalikke sobivaid paindmetoodikaid uuritakse ja otsustusmudel luuakse (vt peatükk 3). Metamudelite loomiseks defineeritakse

kvaliteedi põhimõtted, mida autor plaanib mudelite koostamisel järgida (vt peatükk 4). Seejärel kirjeldatakse erinevaid skaleeritavaid paindmetoodikaid, mille käigus koostab autor neid metoodikaid kirjeldavad metamudelid, mis kirjeldavad ära paindmetoodikas esinevad rollid, tegevused ja tehised (vt peatükk 5). Mõningaid loodud metamudeleid võrreldakse omavahel, et leida nende abil esitatud metoodikate peamised erinevused (vt jaotis 5.6). Loodud mudelite põhjal valitakse Saaty otsusmudeli alternatiive. Töö teises pooles luuakse otsustusmudel sobiva paindmetoodika valikuks ettevõttele X ja kasutatakse seda (vt peatükk 6). Töö lõpus analüüsitakse tulemusi ning tehakse järeldused ja esitatakse täiendavad ettepanekud (vt peatükk 7).

2 Varasemad uuringud antud valdkonnas

Käesoleva lõputöö teema valimisel otsis autor antud valdkonnas juba tehtud teaduslikke uuringuid. Autor eeldas, et erinevaid arendusmetoodikaid ja nende valikut on varasemalt uuritud, kuid millises mahus ja kontekstis, seda autor ei teadnud ning otsingu eesmärk oligi see välja selgitada. Tõestamaks põhjaliku otsingu läbiviimist toob lõputöö autor antud peatükis välja antud teemal leitud varasemad uuringud.

Autor viis uuringu läbi kahes etapis. Kõigepealt katsus ta leida erinevate arendusmetoodikate metamudeleid ning nende metoodikate metamudelitel põhinevat võrdlust, eelkõige otsides skaleeritavaid paindmetoodikaid. Teiseks otsingueesmärgiks oli leida Saaty meetodi abil tehtud paindmetoodikate valikuid. Uuringu käigus kasutati selliseid otsingumootoreid nagu *Google Scholar*, e-ressursside portaal *Primo*, *ResearchGate*, TalTech raamatukogu digikogu, *Google* ja varasemalt läbitud loengute õppematerjalid.

Modelleeritud arendusmetoodikate leidmiseks kasutati omavahel kombineerides esimesest ja teisest loetelust järgmisi otsingusõnu:

1. *Scrum, agile method, agile development methodology, scaled agile, Scaled Agile Framework (SAFe), Spotify, Disciplined Agile Delivery (DAD), Large Scale Scrum (LeSS), Nexus.*
2. *Metamodel, class diagram, conceptual diagram, data model.*

Autor leidis järgnevad varasemad uuringud arendusmetoodikate modelleerimise kohta.

- D. Leffingwell, Agile Software Requirements: Lean Requirements Practices for Teams, Programs, and the Enterprise, Addison-Wesley Professional, 2010.

  Antud raamatus on esitatud agiilse ettevõtte soovilogi (tööde nimekirja) metamudel (*Enterprise backlog model*), mille autoriks on SAFe metoodika looja Dean Leffingwell. Joonis kajastab erineva taseme nõuete tüüpe ja nendevahelisi seoseid. Tervikmudel on oma elementide rohkuse poolest üsna keerukas kuna

kaasaegne tarkvaraarendus ongi keerukas ja seda isegi paindmetoodikaid kasutades. [39]

- „SAFe. SAFe Requirements Model," Scaled Agile, (02.10.2018). [WWW] https://www.scaledagileframework.com/safe-requirements-model/. [06.02.2020].

  Ülalmainitud allikas on väljatoodud SAFe raamistiku nõuete mudel (*SAFe requirements model*), mis esitab soovilogis (*backlog*) kajastuvad erinevate tasemete nõuete tüübid. [40] Kui eelnevas allikas väljatoodud metamudel põhineb paindmetoodikatel üldiselt, siis antud mudel käsitleb just SAFe raamistiku nõudeid. Mõlemad mudelid iseloomustavad vaid ühte osa agiilsest lähenemisest süsteemiarendusele, mitte tervikut.

- Eessaar, E. „Süsteemiarendus ja andmebaasi disaini koht selles.," [WWW] https://maurus.ttu.ee/download.php?aine=346&document=32454&tyyp=do (29.11.2019)

  Antud dokumendis on esitatud kaks valdkonnamudelit – *Scrum* metoodika ja ekstreemprogrammeerimise (XP) kohta, mis kirjeldavad neile omaseid põhimõisteid ja nende mõistete omavahelisi seoseid. [23]

- H. Ayed, B. Vanderose ja N. Habra, „A metamodel-based approach for customizing and assessing agile methods.," *Eighth International Conference on the Quality of Information and Communications Technology.*, Lisbon, Portugal, IEEE, 2012.

  Arendusmetoodika modelleerimist on tehtud ka antud uuringu käigus, kus on aluseks võetud samuti *Scrum* ja ekstreemprogrammeerimise metoodikad. Antud metamudelid on loodud toetamaks agiilsete metoodikate kasutuselevõttu. [24]

Uuringute puhul, mis käsitlevad Saaty meetodi rakendamist sobiva paindmetoodikate valikul, kasutati omavahel kombineerides järgimisi otsingusõnu: *Saaty decision model*, *Saaty decision making*, *agile development methods*, *decision making*, *AHP*, *scaled agile methods*, *choosing appropriate scaled agile method*

Autor leidis järgnevad varasemad uuringud Saaty meetodi abil paindmetoodikate hulgast valiku tegemise kohta.

- Y. Harab, C. Noteboom ja S. Sarnikar, „Evaluating Project Characteristics for Selecting the Best-fit Agile Software Development Methodology: A Teaching Case," *Journal of the Midwest Association for information Systems, 1(1),* pp. 33-52, 2015.

  Antud uuringus kirjeldati nelja erinevat paindmetoodikat ning uuringu eesmärk oli mitmete erinevate kriteeriumite alusel nende hulgast valida kõige sobivam. Alternatiividekst olid XP (ekstreemprogrammeerimine), *Scrum*, *Feature-Driven Development* ja *Crystal* ning töö tulemiks oli mitme kriteeriumiga otsustusmudel ja selle alusel tehtud valik. Valitud metoodikaks osutus XP. [18]

- A. Sharma ja R. K. Bawa, „A multilevel hybrid approach for selection of agile development method using AHP, promethee and fuzzy logic," Punjab, India, 2017.

  Nimetatud uuringu eesmärgiks oli valida sobiv arendusmetoodika nelja erineva meetodi abil – AHP (Saaty meetod), Fuzzy AHP, PROMETHEE, Fuzzy PROMETHEE. Alternatiividekst on kuus metoodikat, neist neli olid esindatud ka eelmises väljatoodud uuringus, lisaks neile oli alternatiividekst veel *Dynamic Software Development Method* (DSDM) ja *Lean Development*. Tulemuseks oli nelja meetodi kasutamine sobivaima paindmetoodika valikuks. Erinevad meetodid andsid erineva tulemuse, AHP ja Fuzzy AHP puhul osutus võitjaks *Crystal*, PROMETHEE ja Fuzzy PROMETHEE meetodi puhul *Scrum.* [20]

- M. Zukova, „Enterprise Agile raamistiku valik Microsoft Dynamics toodete juurutamisel," Magistritöö. Tallinna Tehnikaülikool, Tallinn, 2016.

  Ülaltoodud lõputöö eesmärgiks oli valida välja välja sobiv skaleeritav agiilne raamistik *Microsoft Dynamics* toodete kasutuselevõtuks. Raamistiku valik tehti Saaty meetodit kasutades ning loodud otsustusmudeli põhjal osutus valikuks SAFe. [41] Kuigi antud lõputöös on loodud otsustusmudel sobiva paindmetoodika valikuks, siis on seda tehtud ühe programmi kasutuselevõtu jaoks, mitte terve ettevõtte jaoks (nagu plaanitakse teha käesolevas lõputöös).

Kokkuvõtteks võib järeldada, et sellisel tasemel ja skoobiga nagu käesoleva lõputöö autor plaanib antud valdkonda uurida, ei ole varemalt ühtegi uuringut läbi viidud.

3 Ettevõtte X kirjeldus

Antud lõputöö peamiseks eesmärgiks on luua otsustusmudel sobiva skaleeritava paindmetoodika valikuks ja seda rakendada konkreetse ettevõtte näitel. Ettevõtte nime konfidentsiaalsuse eesmärgil ei avaldata (viitamiseks kasutatakse fraasi „ettevõte X“), kuid käesolevas peatükis kirjeldatakse antud ettevõtet ja IT arenduse struktuuri lähemalt, nii nagu oli see enne SAFe raamistiku juurutamise alustamist.

Organisatsiooni tüübilt on tegemist suurettevõttega. Ettevõttes on üheksa üksust – personal, finants- ja tuginteenused, juriidika, kommunikatsioon, äriklient, eraklient, müük ja teenindus, taristu ning digikanalid- ja analüütika. Ettevõtte taristu üksuse üks osa on IT arendus, kus on töötajaid kokku umbes 130. Suur osa ettevõtte arendusressursist on partnerite näol sisseostetud. Ettevõttel on umbes 30 tarkvaraarenduses osalevat partnerit, kes peamiselt kannavad arendaja rolli. Seega kokku töötab ettevõtte IT arenduses umbes 160 inimest. Ettevõttes on erinevate teenusejuhtide näol aktiivne äripool. Neil on ärivisioon ja nad teevad IT arendusele pidevalt tellimusi. Ettevõte arendab lahendusi ainult sisemiseks kasutamiseks, väljaspoole ettevõtet arendusteenust ei pakuta.

Ettevõttes on siiani kasutatud projektipõhist arendamist. Projekti alustades oli projektijuhi ülesanne koostöös tiimiga valida sobiv arendusmetoodika, mille järgi tööd korraldada. Enamasti on ettevõttes kasutatud Scrum’i ja kose mudelil põhinevat arendust, aga ka Kanban’i. Tavapärane on ka mitme arendusmetoodika segunemine ehk nö hübriidmetoodika rakendamine. Projektipõhisel arendamisel nähti ettevõttes mitmeid kitsaskohti. Tihti olid projektid väga suured ja ajamahukad, paljuski rakendati kose mudelit ja väärtust suudeti luua alles mitme kuu pärast, mis ärilisi tellijaid ei rahuldanud. Suurte projektide puhul oli tavapärane, et tähtaegadest ei suudetud kinni pidada. Suureks probleemiks oli sõltuvuste kindlakstegemine ja nende lahendamine jäi tihtipeale ainult projektijuhi kanda. Eesmärk oli eelkõige avada projektile eelarve ja sageli tekkis olukord, kus sõltuvuste määratlemine jäi seetõttu tagaplaanile. Lisaks puudus ettevõttes tooteportfelli juhtimine. Ettevõte omas ainult projektipõhist vaadet ja suurt pilti keskselt ei vaadatud. Samuti tekitas projektipõhine arendus ressursside jaotamise probleeme, kuna

samal ajal võis olla tähelepanu keskmes mitme suure projekti tegemine, mis võis viia selleks vajalike vahendite kattuvuseni.

Ettevõtte IT arenduses on kaetud järgmised rollid ja nende vastutused.

- **IT analüütik**, kes suhtleb äritellijaga ja vastutab nõuete kogumise eest. IT analüütik mõtleb ka välja lahenduse ning tihti on selle rolli kandja lisaks testija rollis, et loodud lahendust valideerida.
- **Arendaja**, kes teostab töötava lahenduse (vastavalt analüütiku ettekirjutustele).
- **Testija**, kes valideerib lahendust, mille arendaja valmis tegi. Ettevõttes ei ole igas tiimis sisemiselt eraldi testija rolli, vaid on testija, kas vastutab mitme arendustiimi lahenduste testimise eest. Enamasti on arendaja ka ise testija rollis.
- **Arhitekt**, kes kavandab tehnilise lahenduse ja süsteemi arhitektuuri.
- **Projektijuht** koostab projektiplaani ning vastutab projekti juhtimise eest. Projektijuht juhib ühe valdkonna projekte üle erinevate tiimide.
- **Rakenduse administraator**, kes tagab rakenduse haldamise ja töökindla toimimise.
- **UX disainer**, kes vastutab uute loodavate funktsionaalsuste kasutajamugavuse eest.

Arendusmeeskondi on ettevõttes kokku 18. Ettevõttes on komponent-tiimid, mis tähendab, et tiimid on struktureeritud arhitektuuriliste komponentide järgi. Seega iga tiimi vastutuses on üks või enam komponenti, mille arendamise võimekus on tiimiliikmetel olemas. Mitmed tiimid on ristfunktsionaalsed, kus kõik liikmed suudavad olla mitmes rollis korraga (nt arendaja, testija ja analüütik) [42]. See tähendab, et ühe projekti jaoks (kliendile väärtuse loomiseks) on enamasti läinud vaja mitut tiimi, sest arendusi on vaja teha erinevates komponentides. Lisaks on ettevõttes juurutatud DevOps tarkvaraarenduse kultuur.

Ettevõtte tulevikueesmärk on paremaks sõltuvuste haldamiseks koordineerida kõikide tiimide tööd korraga ja rakendada agiilseid tarkvaraarenduse põhimõtteid, et luua ärile pidevalt väärtust.

4 Metamudelite koostamise põhimõtted

Antud modelleerimise täpsemateks tulemiks on metamudelid, mis vastavad MOF (*Meta-Object Facility*) arhitektuuris tasemele M2. Metaandmed on andmed andmete kohta ja metamudel on mudel, mis kirjeldab andmeid ja nende andmete omavahelisi seoseid. MOF on disainitud neljatasemelise arhitektuurina, mille struktuur, tähendus ja omavahelised seosed on defineeritud. MOF arhitektuuri elemendid igal tasemel on sõltuvad ülemise taseme elementidest. Kõige ülemine kiht M3 on kõige üldisem, mida allapoole, seda täpsemaks antud kihis mudelielemendid lähevad. Tasemele M0 vastavad modelleeritava maailma objektid e eksemplarid, tasemele M1 nende objektide kirjeldus mudelitena mingis modelleerimiskeeles, tasemele M2 selle modelleerimiskeele kirjeldus ja tasemele M3 selle keele kirjeldus, mille abil saab erinevaid modelleerimiskeeli kirjeldada (metamodelleerida). Seda arhitektuuri kasutades kirjeldatakse näiteks UML (*Unified Modeling Language)*. UMLi metamudel vastab tasemele M2. UMLi kirjeldamiseks kasutatakse MOF metamodelleerimise keelt, mis põhineb UMLi alamhulgal (täpsemalt klassimudelitel). [11]

UML valiti antud töös modelleerimiseks, sest vaatamata erinevatele väidetavatele probleemidele (näiteks liigne keerukus, ülepaisutatus ühes kohas hägusus teises kohas) [43] on see keel maailmas laialdaselt kasutusel tarkvara, infosüsteemide ja äri modelleerimisel. Selle keele kohta on olemas standard [44] (OMG ja ISO poolt heakskiidetud), selles keeles loodud mudeleid oskavad paljud inimesed lugeda ja selle keele jaoks on palju modelleerimisvahendeid.

Mudelite loomisel kasutatakse *Enterprise Architect* modelleerimistarkvara (CASE vahendit). Seal saab muuhulgas luua UML keelseid mudeleid [32]. Kuna lõputöö autoril on varasem kogemus antud tarkvaraga modelleerimisel, siis mudelite tegemiseks valis autor just selle tarkvara.

Eelnevalt kirjeldatud modelleerimistarkvara *Enterprise Architect* ja standardse modelleermiskeele UML abil esitatakse kirjanduse lugemise käigus kogutud informatsiooni põhjal paindmetoodikate metamudelid klassidiagrammidena, millega

kirjeldatakse ära metoodika elemendid, sh rollid, tegevused, tehised, ning nende omavahelised seosed [26]. Siinkohal lähtub autor ainult kirjanduses esitatud metoodika kirjeldusest. Tegelikkuses elus ei pruugi omavahelised seosed olla nii rangelt määratletud ja neid kohaldatakse vastavalt kontekstile vajaduse tekkimisel. Klassidiagrammide juures esitatakse nende sõnalised selgitused.

4.1 Kvaliteedi tagamine

Metamudelite koostamisel on väga oluline järgida erinevaid mudelite kvaliteedi nõudeid ja tagada seeläbi mudelite kvaliteet. Mudeleid kasutatakse tarkvaraarenduses näiteks nõuete kirjeldamiseks ja kui mudel ei ole arendajale mõistetav või on valesti tehtud, siis võib ka loodav süsteem olla vale või nõuab süsteemi loomine rohkem pingutust kui vaja. Selleks, et autori poolt loodavad mudelid oleksid võimalikult täpsed ja hästi arusaadavad, on vaja nende koostamisel järgida kolme kvaliteedi aspekti.

1. **Süntaks**. Mudel peab olema süntaktiliselt korrektne, st vastama keelereeglitele. Süntaks võib olla ebakorrektne kahel juhul – morfoloogilised vead, kus mudel esitab sümboleid, mida ei ole antud keeles defineeritud, või süntaksi ebatäielikkus, milles mudelil esitatav teave ei järgi keelelist grammatikat. Korrektse süntaksi kasutamise tagab CASE vahend, milleks antud töö puhul on *Enterprise Architect*.
2. **Semantika**. Mudelis esitatud väited modelleeritava maailma/valdkonna kohta peavad olema tõesed (valiidsus ehk paikapidavus) ning mudelis peavad olema kõik huvipakkuvad/vajalikud tõesed väited modelleeritava maailma kohta (täielikkus). Mida sarnasemad on mudel ja modelleeritav valdkond, seda parem on mudeli semantiline kvaliteet. Mudel on maailma lihtsustus ja täpse vastavuse saavutamine ei ole otstarbekas. Seega peab mudeli semantiline kvaliteet olema *piisavalt hea* modelleerimise abil lahendatava ülesande jaoks. Semantilisi kvaliteedi eesmärke on kaks.
    - **Paikapidavus** (*validity*) ehk kas kõik mudelis esitatud väited metoodika kohta on asjakohased ja korrektsed vastavalt probleemile.
    - **Täielikkus** (*completeness*) ehk kas mudel sisaldab kõiki vajalikke asjakohaseid ja korrektseid väiteid metoodika kohta.

3. **Pragmaatika.** Mudel peab olema hästi kasutatav (arusaadav, loetav). Tuleb valida kasutajaskonnale sobiv esitusviis ning seda järjekindlalt kasutada. [45]

Nendest kolmest kvaliteedi aspektist lähtuvalt on autor defineerinud mudelite koostamise põhimõtted (Tabel 1), mis aitavad tagada mudelite kvaliteedi. Nende reeglite järgimise tulemusel tekivad ühtsed mudelid ning välditakse modelleerimises esineda võivaid halbu praktikaid [29] [31].

Tabel 1. Modelleerimise põhimõtted [30] [31].

ID	**Põhimõte**	**Põhimõtte sisu**
1	Elementide arv	Optimaalne elementide arv ühel diagrammil (vaade, mille kaudu visualiseeritakse mudelielemente) on 7+/-2 põhielementi, mis tagab ka optimaalse diagrammi suuruse. Küll aga ei sea autor seda põhimõtet piiravaks juhul, kui peaks parema ülevaate saamise eesmärgil tekkima ühel diagrammil vajadus rohkemate elementide näitamiseks. Viimasel juhul on kindlasti eriti oluline peita nende elementide mitte-huvipakkuvad detailid.
2	Elementide terminoloogia	Mudelis esitatavad mõisted leitakse valdkonna kirjanduse lugemise käigus erinevatest teadusartiklitest ja paindmetoodikaid kirjeldavatest raamatutest. Kasutatakse läbivalt ühesugust terminoloogiat, et lihtsustada mudelitest arusaamist.
3	Keelelisus	Mudeli koostamisel kasutatakse eesti keelt. Kuigi enamik mõisted on tuntud inglise keelses, siis töö autor üritab leida neile sobivad tõlked. Mudeli tekstilises kirjelduses esitatakse kõigepealt eestikeelne tõlge ja sulgudes ingliskeelne tõlge. Mõisted jäetakse tõlkimata, kui need on inglise keelest ülevõetud eestikeelsesse kasutusse. Tõlkimisel kasutatakse erinevaid IT eriala sõnastikke nagu EKI inglise-eesti sõnastik [46], IT terministandardi sõnastik [47], Standardipõhine tarkvaratehnika sõnastik [6] ja Arendussõnastik [5]. Teatud terminitel ei eksisteeri eestikeelseid tõlkeid, mistõttu proovib autor nendele mõistetele sobivad tõlked ise välja pakkuda. Ise väljamõeldud tõlked tähistatakse tärniga (*). Neid tõlkeid nimetatakse ka jaotises 7.3. Kirjandusest leitud tõlke puhul viidatakse tekstis allikale.
4	Klasside nimed	Kõik klasside (metaklasside, olemitüüpide) nimed esitatakse parema loetavuse huvides ainsuses.
5	Suur- ja väiketähtede kasutus	Pakettide ja klasside nimedes on esimene täht suurtäht, edasi väiketähed. Atribuutide nimed on läbivalt väiketähtedega. Erandiks on akronüümid, mille puhul kasutatakse läbivalt suurtähti.

ID	Põhimõte	Põhimõtte sisu
6	Pakettide struktuur	Mudelielementide organiseerimiseks kasutatakse pakette. Kõikide paindmetoodikate puhul kasutatakse ühesugust pakettide struktuuri. Iga metoodika kirjeldab ära selle tegevused, artefaktid ehk tehised ja tegutsejad. Metoodikast arusaamise hõlbustamiseks kategoriseeritakse mudelielemendid nelja gruppi ehk paketti – tehis, tegutseja, tegevus ja varia. Varia alla kuuluvad elemendid, mida ei saa lahterdada esimesse kolme paketti. Näiteks on seal klass „metoodika“, kuna metoodika on nii tegevused, tehised ja rollid kokku. [23] Diagrammid luuakse nii, et iga diagrammile paigutatud metaklassi puhul on näha ka selle pakett.
7	Seosetüüpide võimsustikud	Seosetüüpidele on võimsustike määramine kohustuslik, et kirjeldada metoodikat võimalikult detailselt.
8	Seosetüüpide kirjeldused	Seosetüüpide kohta lisatakse diagrammidest eraldiseisvad tekstikirjeldused seal, kus see on mudeli paremaks mõistmiseks vajalik.
9	Elementide värvimine	Näitamaks metoodika nüansse (nt erinevaid tasemeid) kasutatakse diagrammidel mudelielementide värvimist. Selle eelduseks on värvide tähenduse dokumenteerimine. Värve kasutatakse järjekindlalt.
10	Atribuudid	Vajadusel esitatakse klasside atribuudid (mis väljendavad olemite nimelisi ja mudeli kontekstis huvipakkuvaid omadusi). Kui atribuutidega klassi on vaja näidata on mitmel diagrammil, siis näidatakse selle atribuute ainult põhidiagrammil.
11	Rollid	Roll kirjeldab klassi tähendust seosetüübi kontekstis, milles see osaleb, ning need lisatakse seosetüüpidele parema loetavuse huvides sinna, kus autor näeb, et selleks on vajadust.
12	Kitsenduste kasutamine	Täiendavate kitsenduste (lisaks võimsustikele) mudelis esitamiseks võib need diagrammile kirja panna või diagrammi juures eraldi dokumenteerida. *Object Constraint Language* (OCL) on üks võimalik formaalne keel, milles kitsendusi väljendada [48]. Autor otsustas esitada mudelites kitsendused inimkeelsete lausetena, sest leiab, et nii on erinevatele osapooltele mudelitest lihtsam arusaada. Erandiks on üldistuste hulkadega (*generalization set*) seotud kitsendused: ▪ {Mandatory} – iga ülatüüpi olem (ülaklassi objekt) peab kuuluma ka mõnda alamtüüpi (alamklassi). ▪ {Optional} – võib leiduda ülatüüpi olemeid, mis ei kuulu ühtegi alamtüüpi. ▪ {And} – ülatüüpi olem võib korraga kuuluda mitmesse alamtüüpi.

ID	Põhimõte	Põhimõtte sisu
		▪ {Or} – ülatüüpi olem võib korraga kuuluda ainult ühte alamtüüpi.

Kontrollimaks metamudelite kvaliteeti valideerib autor neid mudeleid valdkonna ekspertide abil. Valideerimine toimub mudelite ülevaatuse käigus, kaasates protsessi eksperdid, kes antud paindmetoodikaid tunnevad kas teoreetiliselt või on neid praktiseerinud. Mudelite ülevaatajad on autori kolleegid, kes ametitelt on agiilne juht (*Agile Coach*) ja arendusjuht. Lisaks vaadatakse mudelid üle koos IT valdkonnas töötavate isikutega, kes metoodikaid ei tunne. See annab konstruktiivse tagasiside selle kohta, kas loodud metamudelid kirjeldavad metoodika sisu ja selle elemente piisavalt täpselt ja arusaadavalt. Ülevaatused viiakse läbi siis kui kõigi mudelite esimene versioon on valminud ning mudeleid vaatab autor iga eksperdiga üheskoos eraldi läbi. Selle protsessi eesmärk on saada võimalikult objektiivne tagasiside, et võimalikult paljud mudelites esinevad tõlgendamise vead saaksid parandatud. Ülevaatuse tulemuste alusel tehakse mudelitesse vajadusel parandusi. Mudelite ülevaatamisel peetakse silmas kolme kvaliteedi aspekti (süntaks, semantika ja pragmaatika) ning eelnevalt kirjeldatud modelleerimise põhimõtteid. Protsessi ja tulemusi kommenteeritakse põgusalt paindmetoodikate peatükis (vt jaotis 5.5).

4.2 Metamudelite vaated

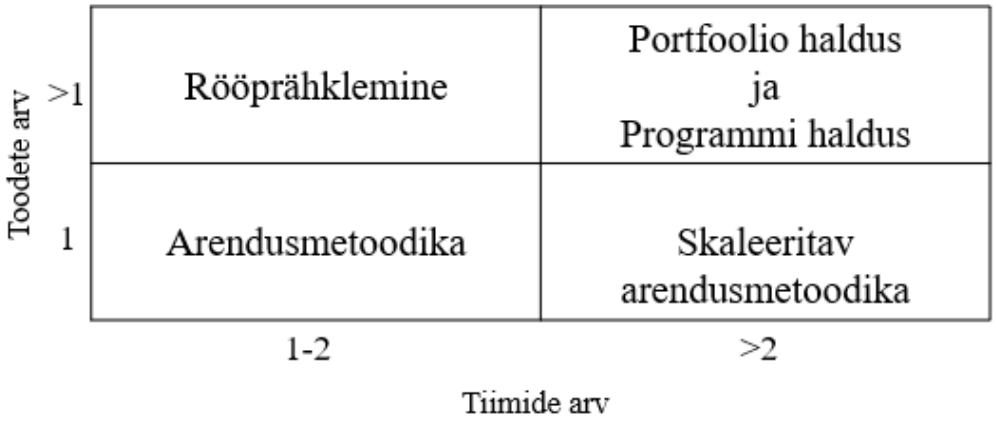
Paindmetoodikad defineerivad väga mitmeid erinevaid rolle, artefakte ehk tehiseid ja tegevusi. Mudelielementide suure arvu tõttu ei saa neid kõiki ühel diagrammil esitada, sest tulemus oleks halvasti loetav. Samuti läheks see vastuollu Tabel 1 esimese põhimõttega. Seetõttu tuleb autoril leida viis, kuidas mudelielemente diagrammidele jaotada. Siinkohal tuleb eelkõige arvesse võtta seda, et kõige rohkem omavahelisi seoseid omavad elemendid oleksid koondatud ühele diagrammile. Üks ja sama mudelielement võib olla erinevatel diagrammidel ja ühel diagrammil võib olla mitu elementi. Ühel ja samal teemal koostatud diagrammid (üks või mitu) moodustavad vaate. Järgnevalt esitatakse vaated ja nende sisu, mida lõputöö autor peab iga metoodika kohta nende abil mõistlikuks esitada.

1. **Tiimi vaade**, mis kirjeldab ära erinevad tiimi liikmete rollid ja muud seotud rollid. Lisaks annab tiimi vaade infot, millised arendusmetoodikad on antud raamistiku puhul tiimidel võimalik kasutada.
2. **Tiimi sprindi tseremooniate vaade**, mis annab ülevaate sprindi (iteratsiooni) jooksul toimuvatest sündmustest ehk tseremooniatest. Lisaks kirjeldab see, milliseid nõuete nimekirju (soovilogi) iga sündmuse puhul käsitletakse ja kes neid tseremooniaid läbi viivad.
3. **Tiimiüleste tseremooniate vaade** kirjeldab ära sündmused, mis toimuvad sprindis või mõnes muus ajaraamis, et hallata korraga kõikide tiimide poolt tehtavat tootearendust. Kui eelmine vaade kirjeldas ära ainult tiimikohased sündmused, siis antud vaates on need tseremooniad, kus osalevad erinevad tiimid või nende esindajad.
4. **Tootearenduse vaade** annab ülevaate, kuidas toimub vaadeldava metoodika korral tootearendus. Lisaks esitatakse mudelis olulisemad organisatsiooni tootearendust puudutavad funktsioonid ehk kirjeldatakse, kuidas toimub tootearendusele kaasaaitamine organisatsiooni erinevatel tasemetel. Antud vaade kirjeldab, millised on toote arendamisel olulised rollid, tehised ja tegevused.

Ühe ja sama vaate puhul proovitakse erinevate metoodikate puhul samaväärsed või sarnase sisuga elemendid paigutada vaates visuaalselt samadesse kohtadesse, et mudelite võrdlemisel oleks parem erinevusi märgata.

5 Skaleeritavad paindmetoodikad

Agiilne arendus on tarkvararenduse lähenemisviis, mis kasutab iteratiivset kavandamise, planeerimise, arendamise ja juurutamise protsessi, et rahuldada kliendi muutuvaid vajadusi, tagades samal ajal pideva arendustulemuste üleandmise ehk tarnimise väiksemate tükkide kaupa. Agiilse tarkvaraarenduse korral arenevad nõuded ja lahendused iseorganiseeruvate ristfunktsionaalsete tiimide vahelise koostöö kaudu [15]. Agiilse arenduse jätkusuutliku praktiseerimise märksõnadeks on iseorganiseeruvad tiimid, kiire ja pidev arendustulemuste tarne – enamasti iga kahe nädala tagant, nõuete muutumisega nõustumine ning äri- ja arendustiimi tugev koostöö. Tuntuimate paindmetoodikate (agiilsete metoodikate) seas on näiteks Scrum, mida 2019. aastal kasutati paindmetoodikate kasutamise aruande (*13th State of Agile Report*) järgi 54% juhtudest [49]. Paindmetoodikaid võib nimetada raamistikeks, sest neid saab projekti vajadustest lähtuvalt kohandada. Väiksema tootearenduse juhtimiseks on kõige efektiivsemad Scrum ja Kanban. Agiilne skaleerimine pole midagi muud kui nende paindmetoodikate kohaldamine ja kombineerimine mitmeid tiime hõlmavate tootearenduste läbiviimiseks. [15] Scrum ise on üsna lihtne, see on aga suur eelis, sest keerukus on skaleerimisel suur puudus [12]. Suurte organisatsioonide jaoks, kus omavaheline suhtlus on väljakutseid pakkuv, pakuvad agiilsed skaleeritavad raamistikud hästi dokumenteeritud hüppelaua, mis aitavad tagada kommunikatsiooni ja annavad juhised tootearenduse skaleerimiseks [50].



Joonis 1. Arendusmetoodika skaleerimine [51].

Toode (*Product*) on terviklik kliendikeskne lahendus, mida reaalsed kliendid kasutavad. Toode ei ole üksik tarkvara komponent, platvorm ega kiht. [52]

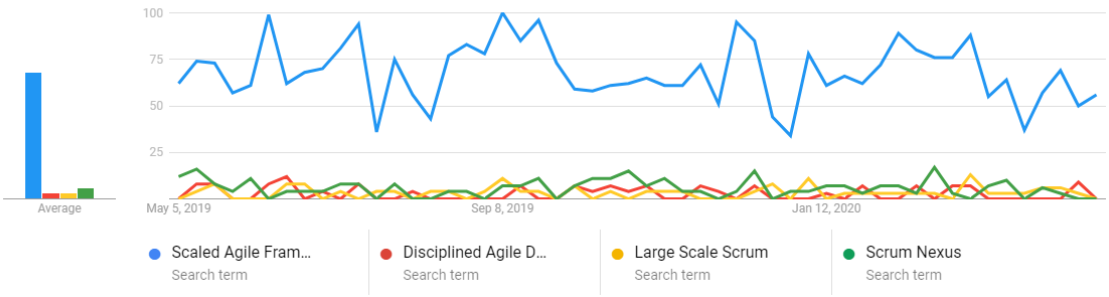
Joonis 1 on maatriks, kuidas on omavahel skaleeritava paindmetoodika tekkimisel suhtes seotud toodete ja tiimide arv. Kui tegemist on ühe kuni kahe tiimiga ja ühe toote arendusega, siis tähendab see lihtsalt arendusmetoodika rakendamist. Kui tiimide arv jääb samaks, kuid tooteid on rohkem kui üks, siis võib nimetada seda rööprähklemiseks (*multitasking*), kus üks tiim arendab mitut toodet korraga või näiteks jagatakse ära iteratsioonide lõikes, millise toote arendusega parasjagu tegeletakse. Kui arendatavaid tooteid on üks, kuid tiime, kes selle kallal töötavad, on üle kahe, on vaja skaleeritavat metoodikat. Kui aga tooteid on mitu ja ka tiime on üle kahe, siis tuleb tegeleda juba väga mastaapselt ka portfelli haldusega. [51]

Käesoleva lõputöö autor leidis järgmised skaleeritavad paindmetoodikad (raamistikud).

- *Scaled Agile Framework* (SAFe),
- *Spotify*,
- *Disciplined Agile Delivery* (DAD),
- *Large-Scale Scrum* (LeSS),
- *Large-Scale Scrum Huge* (LeSS *Huge*),
- *Nexus*,
- *Recipes for Agile Governance in the Enterprise* (RAGE). [53]

Autor koostab metamudelid SAFe, DAD, *Nexus*, LeSS ja LeSS *Huge* metoodikate kohta, kuna esmase kirjanduse uurimise tulemusena selgus, et nende hulgast võiks leida sobivad alternatiivid otsustusmudelisse. Seda, kas see ka nii on, selgub metamudelite loomisel, mis võimaldavad metoodikatest paremini aru saada ja selle põhjal otsustada. Metamudelid koostatakse kõige uutemate versioonide järgi (2020. aasta kevade seisuga). Versiooninumbri esinemisel mainitakse seda vastavat metoodikat käsitlevas jaotises.

Autor välistas *Spotify* raamistiku, sest see ei täpsusta, kuidas tiimi tasemel peaks töö käima. Lisaks on vähe juhiseid, kuidas seda raamistikku kasutusele võtta. Samuti ei ole Spotify puhul määratletud, kuidas juhtida ja hallata portfelli. [54] Lisaks ei uurita lähemalt ka RAGE metoodikat, kuna tegemist on üsna uue metoodikaga ja selle rakendamise kohta puuduvad autorile teadaolevad edulood. Samuti ei ole selle kasutuselevõtuks piisavalt informatsiooni. [53]



Joonis 2. Skaleeritavate paindmetoodikate otsinguhuvi näitajad [55].

Uuringusse metoodikate valimisel lähtuti eelkõige nende populaarsusest, kui palju on neid varasemalt praktiseeritud ja kui palju on varasemaid edulugusid ning kui lihtne on nende olemasoleva teabe põhjal kasutuselevõtmine. Joonis 2 esitab *Google Trends* abil koostatud graafiku autori poolt uurimiseks valitud skaleeritavate paindmetoodikate viimase aasta otsinguhuvi kohta 2020. aasta kevade seisuga. Sellest järeldub, et enim on huvi tuntud SAFe metoodika vastu. Teiste võrreldavate paindmetoodikate vastu on huvi kordades väiksem ning nende huvitase on kõigil kolmel sarnase suurusega. Seega võib järeldada, et eksisteerib hulk skaleeritavaid paindmetoodikaid, mis kirjeldavad juhised ja põhimõtted, kuidas skaleerida agiilseid arenduspraktikaid. Siiski ainult osasid neist kasutatakse laialdasemalt, sest probleemiks on ebapiisavad juhised praktikutele sealhulgas kuidas valida sobivaim raamistik vastavalt ettevõttele ja kontekstile. [56]

Käesolevas peatükis kirjeldatakse ülalnimetatud agiilseid skaleeritavaid paindmetoodikaid, andes lühiülevaate peamistest metoodika põhimõtetest. Uuritud kirjanduse põhjal luuakse paindmetoodikate metamudelid, mis esitatakse klassidiagrammidena, millele lisanduvad sõnalised kirjeldused. Nende loomise eesmärk on tuua välja iga metoodika korral selle tegutsejaid, tegevusi ja tehiseid ning nende omavahelisi seoseid. Loodud metamudelid peaksid andma visuaalse ja kompaktse ülevaate metoodikate mõistetest ja seostest. Peatüki lõpus võrreldakse omavahel metoodikaid ontoloogilise analüüsi põhimõtetest lähtuvalt. Selle tulemusena tuuakse välja metoodikate erinevused ja ka sarnasused.

5.1 Scaled Agile Framework metoodika

Scaled Agile Framework (SAFe) on skaleeritav ja kohaldatav raamistik ettevõtetele, mis aitab juhtida keerukate süsteemide tarkvaraarendust kõikides organisatsiooni tasemetes

ning luua klientidele maksimaalset väärtust võimalikult jätkusuutliku lühikese ajaga, tagades võimalikult kõrget kvaliteeti ja väärtust. SAFe raamistiku kasutamine võimaldab tagada mitme agiilse tiimi koostöö ja sama tootega samadel põhimõtetel töötamise. See ühendab agiilse *Lean* tootearenduse süsteemse mõtlemisega ning selle aluseks olevad laiaulatuslikud teadmised baseeruvad *Lean-Agile* põhimõtetel ja väärtustel. Metoodika raames kirjeldatakse ära rollid, vastutusalad, tehised ja tegevused, mida on vaja paremate äritulemuste saavutamiseks. SAFe raamistiku töötas välja Dean Leffingwell ning aja jooksul on metoodikat pidevalt täiendatud. [57]

SAFe metoodika jaotatakse varasema versiooni 4.6 järgi neljaks tasemeks – tiimi tase, programmi tase, lahenduse tase ja portfelli tase. Igale tasemele on sellele omased rollid, tegevused ja tehised, mida modelleeritakse metamudelis ning selgitatakse mudeli sõnalises kirjelduses täpsemalt. Järgnevalt teeb autor ülevaate SAFe metoodika tasemete sisust. [58]

Portfelli tase (*Portfolio level*) sisaldab põhimõtteid, tavasid ja rolle, mis on vajalikud väärtusahelate arenduste algatamiseks ja haldamiseks. SAFe portfell sisaldab ühte või rohkemat arenduse väärtuseahelat, igaüks neist on pühendunud toodete väljatöötamisele, mis aitab ellu viia ettevõtte ärilist strateegiat [58]. Siin tasemes määratletakse väärtusahelate ja nende lahenduste strateegia ja investeeringute rahastamine. Portfelli tase on SAFe raamistiku struktuuris kõige kõrgem tase, mis kujundab kogu organisatsiooni struktuuri arvestades erinevate väärtusahelate eesmärke. Selleks, et väärtusahelad saaksid täita ettevõtte ärilisi eesmärke, on vajalik neile eraldada vastav eelarve ja teha investeeringuid, mille tegemise eest vastutabki antud taseme rollid. [59]

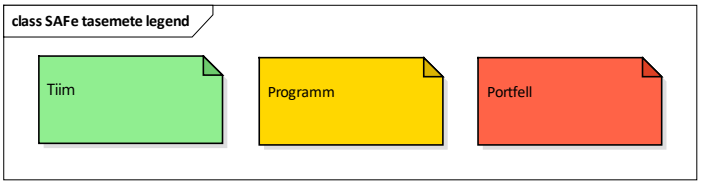
Lahenduse tase (*Large Solution level*) sisaldab rolle, tehiseid ja protsesse, mis on vajalikud suurte ja keerukate lahenduste loomiseks. See on keskendunud lahenduste väljatöötamise nõuete püstitamisele, mitme ART'i (*Agile Release Train* ehk Agiilne väljalaske rong) ja teiste osapooltega koostööle ning vajadusele tagada vastavus määrustele ja standarditele. [59]

Programmi tase (*Program level*) sisaldab rolle ja tegevusi, mis on vajalikud lahenduste pideva tarne jaoks ARTis. Programmi tase on koht, kus arendustiimide, huvirühmade ja muude osapoolte missiooniks on pidevalt tegeleda lahenduse väljatöötamisega. ART kirjeldab programmi tasemel tiime, rolle ja tegevusi, mis pakuvad katkematut väärtuse

loomist. ARTid on virtuaalsed organisatsioonid, mis on loodud funktsionaalsete piiride ületamiseks, mittevajalike takistuste ja sammude eemaldamiseks ning ärilise väärtuse kiiremaks saavutamiseks rakendades SAFe *Lean-Agile* põhimõtteid ja tavasid. ART on programmi taseme peamine organisatoorne üksus, mis seob inimesed ja nende töö ühisesse programmi visiooni, missiooni ja soovilogisse [59]

Tiimi tase (*Team level*) sisaldab rolle, tegevusi, sündmuseid ja protsesse, mille arendamisega loob tiim väärtust ARTi kontekstis. Kuigi tiimi taset on kujutatud eraldi tasemena, siis tegelikkuses on see programmi taseme oluline osa. Erinevad ART rollid toetavad arendustiime, tänu millele on tiimid täielikult võimelised kirjeldama, arendama, testima ja tarnima igas iteratsioonis valminud ärilist väärtust loovat süsteemi osa. [59]

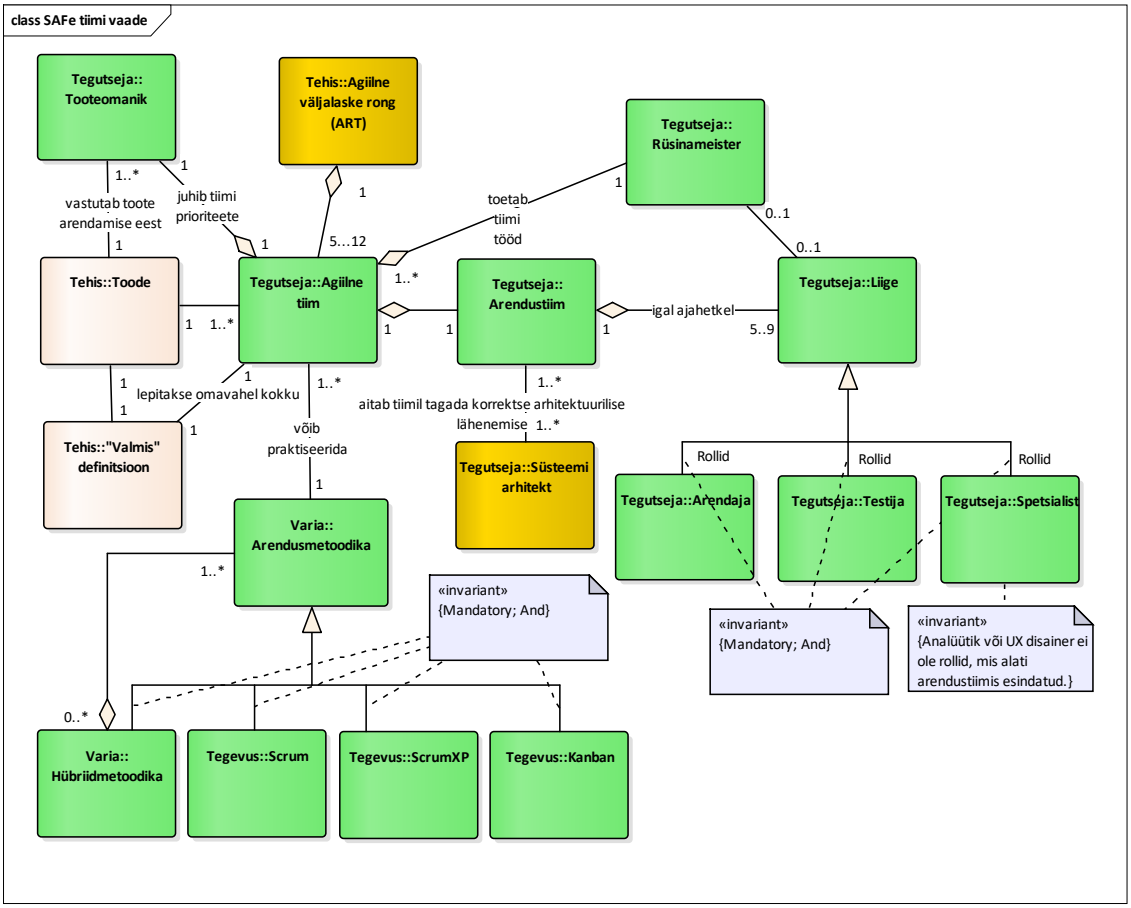
Kõige uuem SAFe 5.0 versioon ilmus 2019. aastal ja see kirjeldab neli erinevat metoodika versiooni, mis tulenevad erinevaid tasemeid omavahel kombineerides. Versiooni valik tuleb teha ettevõtte kontekstit lähtuvalt. Terviklikus SAFe (*Full* SAFe) vaates on kolm taset – portfelli, lahenduse ning tiimi ja põhialuse tase (*Essential*). Põhialuse tase on saadud ühendades omavahel tiimi ja programmi tasemed. Autor leiab, et erinevatest SAFe versioonidest oleks antud ettevõttele sobivaim portfelli SAFe (*Portfolio SAFe*), sest siin puudub lahenduse tase, mis on vajalik veelgi suuremahulisema tootearenduse korral, kus inimesi on mitu sada või rohkem. Seetõttu lähtutakse modelleerimisel portfelli SAFe'st milles on põhialus ja portfelli tase. Selleks, et eristada millised rollid, tegevused ja tehised kuuluvad loodavas metamudelis ühe või teise taseme juurde, kasutatakse klasside värvimist Joonis 3 esitatud legendi alusel. [58]



Joonis 3. SAFe metamudelite koostamise legend.

Kuigi versiooni 5.0 järgi on tiimi ja programmi tase ühine, näidatakse neid iseloomustavaid elemente eraldi värvidega. Roheline värv kirjeldab tiimi taseme elemente, kollane värv programmiga seotud elemente ning portfelli taset iseloomustavad punase värviga elemendid. Seega rohelised ja kollased elemendid kuuluvad mõlemad

põhialuse tasemesse. Beežiga esitatud klassid ei ole otseselt seotud ühegi konkreetse tasemega.

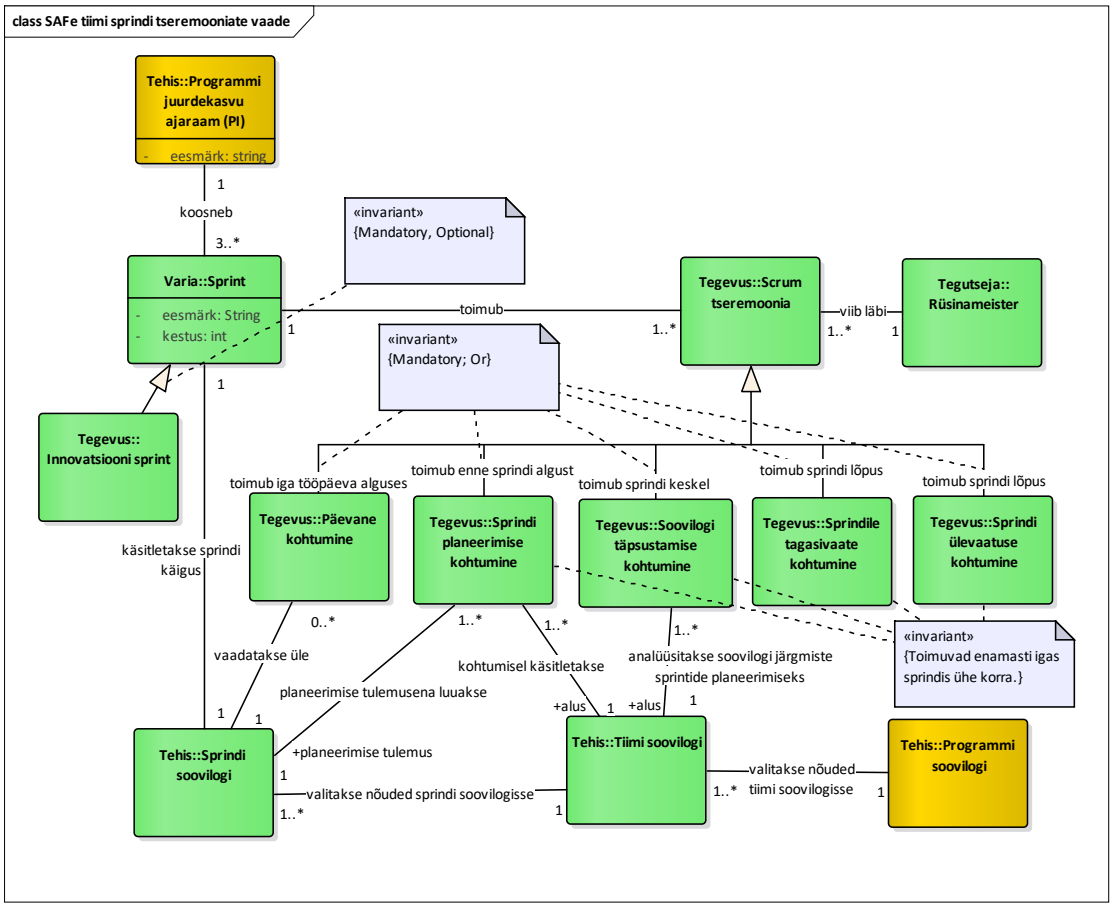


Joonis 4. SAFe tiimi vaade.

Joonis 4 on diagramm, mis kirjeldab SAFe tiimi ja selle toimimist. Arendustiim (*Development team*) on väike ristfunktsionaalne (*cross-functional*) arendajatest (*Developer*) ja testijatest (*Tester*) koosnev rühm, kuhu kuulub enamasti viis kuni üheksa liiget ja kes kõik töötavad ühiselt toote funktsionaalsuse loomise eesmärgil. Enamjaolt moodustavad arendustiimi arendajad, kuid tihtipeale on igal tiimil ka oma testija (*Tester*). Samas ei pruugi see nii olla juhul kui arendajad ise oma töid ka testivad. Samuti võib arendustiimis olla veel spetsialiste, kes on vajalikud funktsionaalsuse tootmiseks, näiteks analüütik või kasutatavuse (UX) disainer. Üks liige võib olla mitmes rollis korraga. Arendustiim on agiilse tiimi (*Agile team*) osa. Igas ARTis on viis kuni kaksteist agiilset tiimi, mis hõlmavad vajalikke rolle ja kasutavad vajalikku taristut, et tarnida täielikult nõuetele vastav ja testitud tarkvara. Agiilne tiim on ristfunktsionaalne ScrumXP'd või Scrum Kanban'i viljelev rühm, kuhu lisaks arendustiimile kuuluvad ka tooteomanik

(*Product Owner*) ja rüsinameister (*Scrum Master*). Igal agiilsel tiimil on oma tooteomanik ja iga tooteomanik töötab korraga ainult ühe agiilse tiimiga. [59]

Agiilsel tiimil on võimekus ja volitus tegeleda iteratsioonide jooksul toote nõuete püstitamise, arenduse ja testimisega. Tooteomanik tegeleb agiilse tiimi tööde prioriteetide määramisega, rüsinameister aga on kui tiimi *coach* (treener, teenindav juht), kes aitab tiimil tekkinud takistusi eemaldada, tegeleb Scrum tseremooniate läbiviimisega ning toetab tiimi, et nende töö oleks võimalikult tulemuslik. Rüsinameister võib olla ühine mitmele tiimile ja töötada osalise koormusega. Ideaalis on ta ikkagi täiskohaga pühendunud ühele tiimile. Agiilseid tiime nõustab süsteemi arhitekt (*System Architect*), kes püstitab süsteemile arhitektuurilised nõuded ja kontrollib, et arendused on tehtud vastavalt arhitektuurilistele nõuetele. ScrumXP on arendusprotsess, kus tiimi töö haldamiseks kasutatakse Scrum põhimõtteid ning ekstreemprogrammeerimise (XP, *Extreme Programming*) tarkvaraarenduse praktikaid. SAFe tiimid saavad valida, kas nad tegutsevad Scrum, Kanban või ScrumXP tiimidena või hoopiski hübriidmudeli alusel. [59] [4]



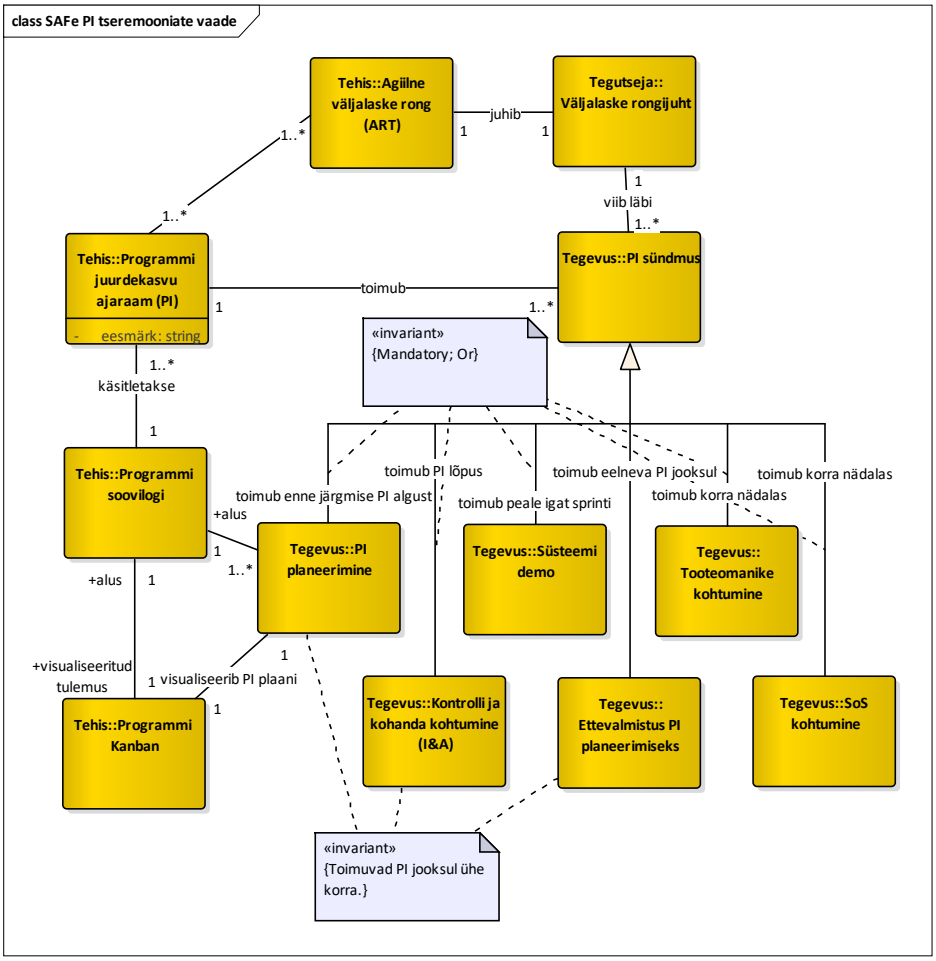
Joonis 5. SAFe tiimi sprindi tseremooniate vaade.

Joonis 5 esitab sprindi tegevused ja nende seosed. Iteratsioon ehk sprint (*Iteration*) on fikseeritud pikkusega arenduse pisitsükkel, mille eesmärk on luua väärtust, tarnides (andes kliendile üle) uut arendatud ja testitud funktsionaalsust (toote osa). Sprindi kestus on üks kuni neli nädalat, kuid soovitatav optimaalne kestus on kaks nädalat. Igale sprindile on iseloomulikud viis Scrum tseremooniat, mida viib läbi rüsinameister ja kus osaleb kogu agiilne tiim. Kõik tseremooniad, välja arvatud päevane kohtumine, toimuvad sprindi jooksul enamasti üks kord. Programmi juurdekasvu ajaraam* (*Program Increment* ehk PI) on ajaraam, mille jooksul ART tarnib toote juurdekasvu ja see koosneb kolmest või rohkemast sprindist. Innovatsiooni sprint (*Innovation and Planning Iteration* (IP)) on PI viimane sprint, mis annab agiilsele tiimile aja innoveerida nii toodet kui ka protsessi ja tegeleda eesoleva PI planeerimise ettevalmistustega (soovilogi tööde analüüsimisega). Scrum tseremooniad on järgmised.

- Päevane kohtumine (*Stand Up*) [23] on igapäevane tiimi infovahetuse koosolek, mille kestus peaks jääma 15 minuti piiresse ja kus osalevad kõik arendustiimi liikmed ning rüsinameister. Tooteomaniku osalemine ei ole kohustuslik. Päevane kohtumine toimub püstijalu ja iga tiimi liige saab sõna. Ta peab rääkima, millega ta tegeles eile, mida ta plaanib teha täna ning kas ja millised on võimalikud takistused. Selline lühike kuid ülevaatlik formaat tagab, et kogu tiim on kursis, millega keegi parasjagu tegeleb ning vajadusel saab rüsinameister sisendi selle kohta, millega on vaja tiimi toetada. Kohtumise käigus käsitletakse sprindi soovilogi.
- Sprindi planeerimise kohtumisel (*Iteration Planning*) [23] määratletakse tulevase sprindi eesmärk ja valitakse sprinti tiimi soovilogist (*Team Backlog*) eesolevaid arendustöid. Tööde planeerimisel lähtutakse tiimi töötundide mahust ning vastavalt sellele annab tiim lubaduse töödele, mida plaanib antud sprindi raames arendada. Kohtumise käigus käsitletakse tiimi soovilogi ja planeerimise tulemusena luuakse sprindi soovilogi.
- Sprindi ülevaatuse kohtumine (*Iteration Review*) [23] on sprindi lõpus toimuv koosolek, kus vaadatakse üle kogu iteratsiooni käigus tehtud töö, hinnatakse kogu iteratsiooni jooksul tehtut, sh antakse hinnang edusammudele ja kohandatakse soovilogi järgmise iteratsiooni jaoks.
- Sprindile tagasivaate kohtumine (*Iteration Retrospective*) [23] on sprindi lõpus toimuv koosolek, mille käigus analüüsitakse, mis läks iteratsiooni jooksul hästi, mida saaks teha paremini ja millised olid peamised takistused. Eesmärgiks on

selgitada välja edaspidiste süsteemi parenduste võimalused. Retrospektiiv põhineb iteratsiooni ülevaate käigus esitatud kvalitatiivsel ja kvantitatiivsel teabel.

- Soovilogi täpsustamise kohtumise* (*Backlog Refinement*) raames vaadatakse üle tiimi soovilogis ootel olevaid töid, et prioriteetsemaid töid täpsustada, detailsemalt kirjeldada ning hinnata töömahtu. [59] [4]

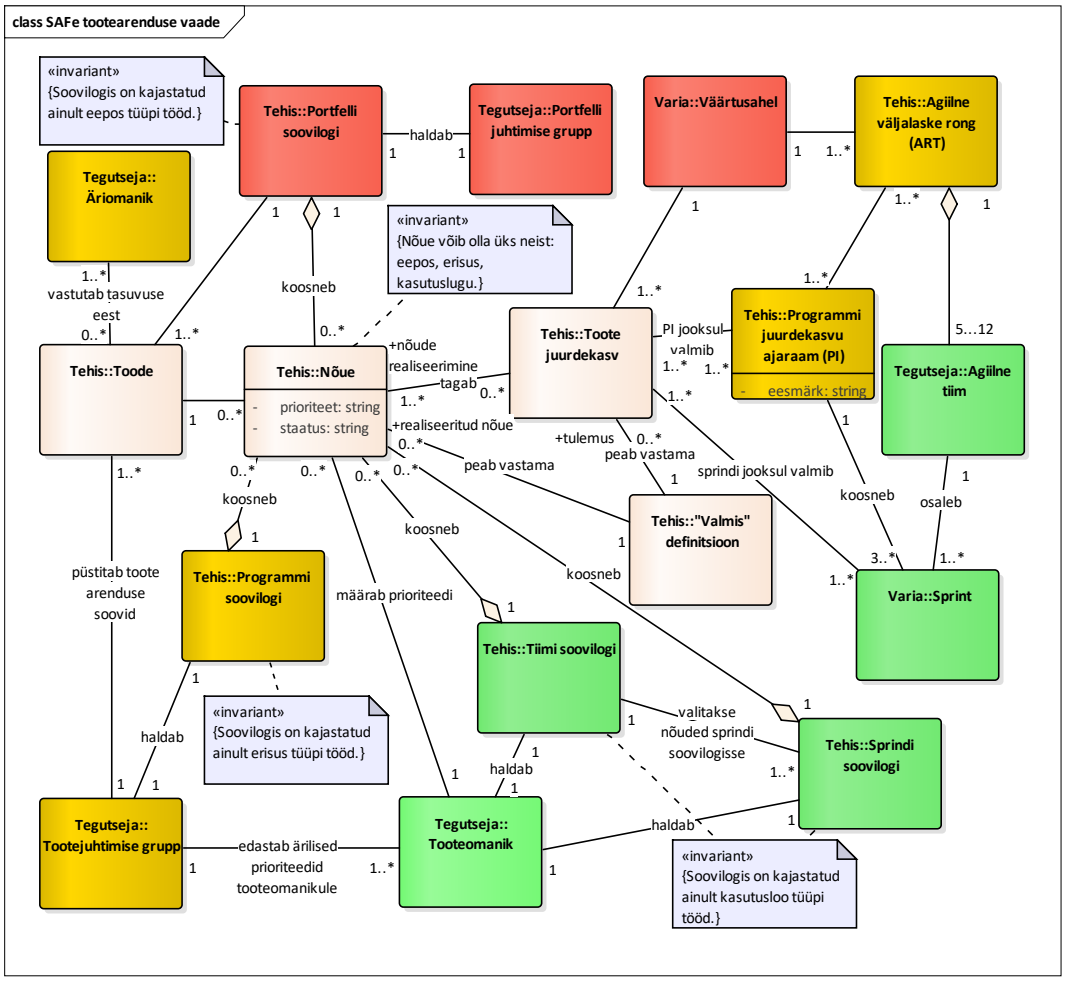


Joonis 6. SAFe PI tseremooniate vaade.

Joonis 6 esitab PI tseremooniate vaate. ART'i juhib väljalaske rongijuht (*Release Train Engineer (RTE)*), kes tagab kogu rongi toimimise ja juhendab rüsinameistreid. RTE korraldab PI tseremooniaid. Agiilne väljalaske rong osaleb PIs, ühes PIs võib osaleda mitu ARTi. PI jooksul käsitletakse programmi soovilogi (*Program Backlog*), kus sisalduvad kogu programmi tiimide nõuded ja mida visualiseeritakse programmi Kanban'il (*Program Kanban*). PIs toimuvad järgmised tseremooniad.

- PI planeerimine (*PI Planning*) on üritus, mille jooksul planeerivad arendustiimid järgmises PIs arendatavaid erisusi ja teevad kindlaks tiimidevahelisi sõltuvusi. Sõltuvalt tiimide arvust kestab PI planeerimine tavaliselt üks kuni kaks päeva. PI planeerimise aluseks on programmi soovilogi.
- Kontrolli ja kohanda kohtumine (*Inspect and Adapt*) on iga programmi lõpus toimuv sündmus, kus demonstreeritakse ja hinnatakse lahenduse hetkeseisu. Seejärel parendavad tiimid oma nõuete soovilogi vastavalt saadud tagasisidele.
- Süsteemi esitlus ehk demo (*System Demo*) on üritus, mille raames tutvustatakse tellijatele ja äri osapooltele (*Business Owner*) äripoole tellitud ja PI jooksul tehtud sprintides valminud funktsionaalsust.
- Ettevalmistus PI planeerimiseks (*Prepare for PI Planning*) on tegevus järgmise PI planeerimise ürituse jaoks, mis hõlmab soovilogide ülevaatust erinevatel tasemetel, organisatsiooni valmisoleku tagamist planeerimiseks ja kogu ürituse korraldamist.
- Tooteomanike kohtumine (*PO sync*) on enamasti iganädalaselt toimuv kokkusaamine. Eesmärgiks on saada aru, kuidas ART'i progress vastab PI eesmärkidele ning käsitelda võimalikke skoobi muudatusi või arendusega seotud probleeme.
- SoS kohtumine (*Scrum of Scrums meeting*) on enamasti iganädalaselt toimuv kohtumine kõikide tiimide rüsinameistrite ja RTE'ga, et vaadata üle eesmärkides püsimine ja rääkida üle tiimidevahelised sõltuvused. [59] [4]

Töö alguses tõi autor välja uuringud, mis on varasemalt antud teemal tehtud (Vt peatükk 2). Kuna autor tuvastas SAFe raamistiku nõuete metamudeli, siis ise ta seda ümberjoonistama ei hakka. Nimetatud metamudel koos autoripoolse lisatud kirjeldusega on kättesaadav lisadest [Lisa 1].



Joonis 7. SAFe tootearenduse vaade.

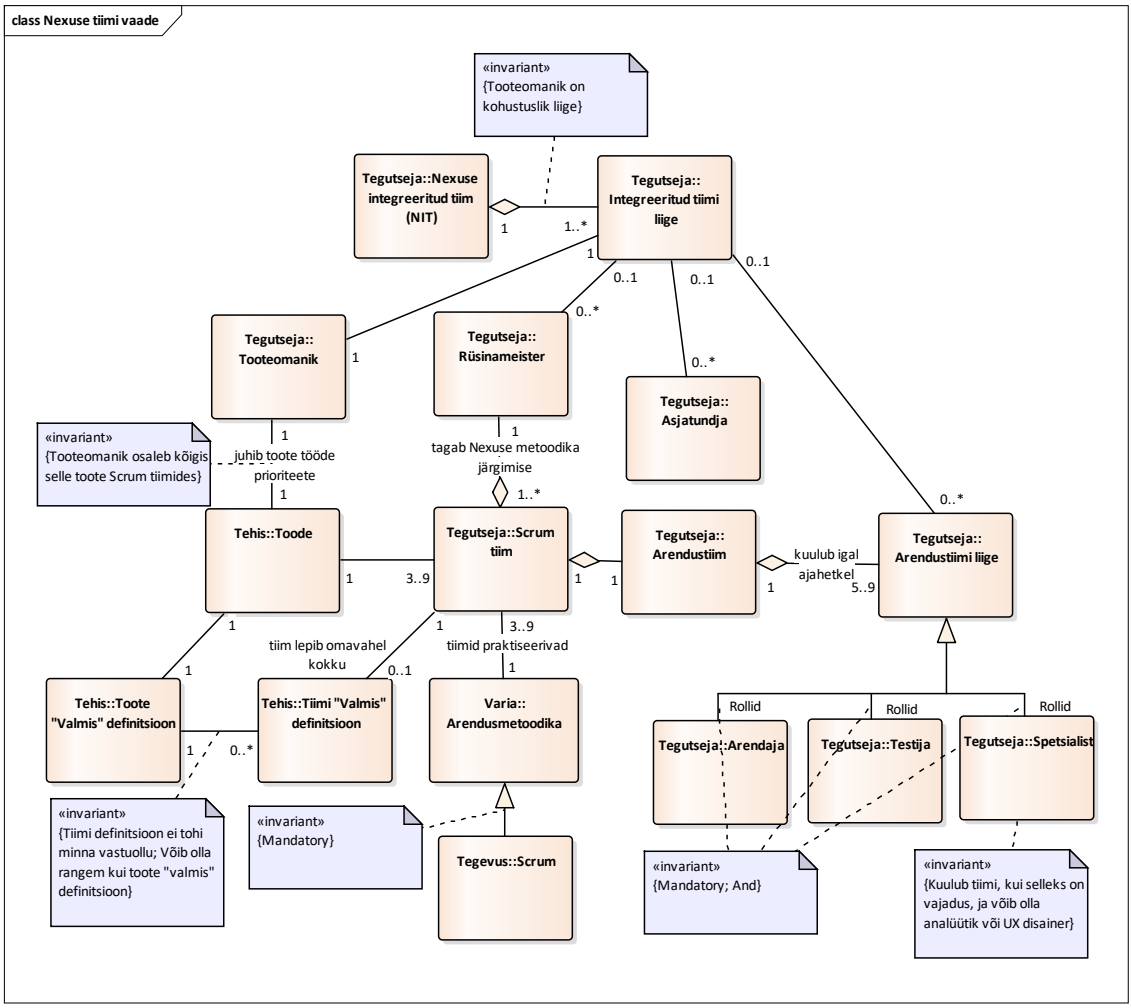
Joonis 7 esitab SAFe metoodika tootearenduse vaate olulisemad elemendid ja nendevahelised seosed. Jooniselt on näha, et nõue võib olla eepose (*Epic*), erisuse (*Feature*) või kasutusloo (*Story*) tüüpi (vt Joonis 58). Igal nõudel on prioriteet ja staatus. Tiimi soovilogi ja sprindi soovilogi koosnevad kasutusloo tüüpi nõuetest, mida haldab ja prioritiseerib tooteomanik vastavalt äriprioriteedile. Tooteomanik vastutab tootega (*Product*) seotud arendustegevuste eest. Programmi soovilogi (*Program Backlog*) koosneb erisuse tüüpi nõuetest, mida haldab ja prioritiseerib tootejuhtimise grupp (*Product Management*). Tootejuhtimise grupp püstitab tootearenduse soovid ja edastab ärilised prioriteedid tooteomanikule. Portfelli soovilogi (*Portfolio Backlog*) koosneb eepose tüüpi nõuetest, mida haldab portfelli juhtimise grupp (*Portfolio Management*). Portfelli juhtimise grupp (*Portfolio Management*) esindab osapooli, kellel on kõige suurem otsustuste tegemise võimalus ja rahaline vastutus. Antud grupp vastutab strateegia kujundamise, investeeringute rahastamise ja agiilse portfelli koordineerimise eest. Üks selle osa on eelarve juhtimine. Portfelli soovilogi (*Portfolio Backlog*) on kõige

kõrgem suuremahuliste tööde tase SAFe raamistikus, mis koosneb nii suurtest ärilistest kui ka tehnilistest töödest. Ühe või mitme nõude realiseerimine sprindi jooksul võib tagada toote juurdekasvu (*Product Increment*), mis peab vastama „valmis“ toote definitsioonile. PI jooksul valmib üks või mitu toote juurdekasvu. ART osaleb ühes või mitmes PIs, PIs osaleb üks või rohkem ARTi. ARTi kuulub viis kuni kaksteist agiilset tiimi, kes osalevad ühistes paralleelsetes sprintides. ART on seotud ühe väärtusahelaga. Väärtusahel (*Value Stream*) peab tagama vajalikud vahendid ehitamaks lahendusi, mis annavad ärile ja kliendile väärtust ehk juurdekasvu. Äriomanik (*Business Owner*) on huvirühm (*stakeholder*), kes on ärilise tellija rollis ja kellel on vastutus äriliste arenduste investeeringutasuvuse (ROI) eest, mis on ARTis toodetud. [59]

5.2 Nexus metoodika

Nexus on lihtne raamistik, mis võimaldab mitmel arendustiimil töötada üheskoos ühe ja sama toote kallal tarnimaks igas iteratsioonis vähemalt ühte täielikult valmis ja integreeritud süsteemi osa (funktsionaalsuse ehk integreeritud juurdekasvu*) (*Integrated Increment*). Mida rohkem tiime ühe toote kallal töötavad, seda rohkem tekib sõltuvuste haldamise näol keerukust. *Nexuse* raamistik aitab neid sõltuvusi leida, ohjata ja vähendada. Kõik *Nexuse* tiimid praktiseerivad Scrum (eesti keeles „rüselus“ [46]) metoodikat. Põhimõtteliselt on *Nexus* laiendustega Scrum metoodika. Nii nagu igal metoodikal on ka *Nexuse* puhul esindatud rollid, tehised ja tegevused. Nexuse autorid on Kurt Bittner, Patricia Kong ja Dave West. [12]

Nexuse eesmärk on, et tiimidel oleks võimalikult vähe teineteisega sõltuvusi. Selleks peavad meeskonnad olema ristfunktsionaalsed ja iseseisvad, arendustööd peavad olema vastavalt tükeldatud ja jagatud ning toote arhitektuur peab olema iseseisvatest komponentidest ülesehitatud, mida iga tiim saab iseseisvalt muuta. [51]

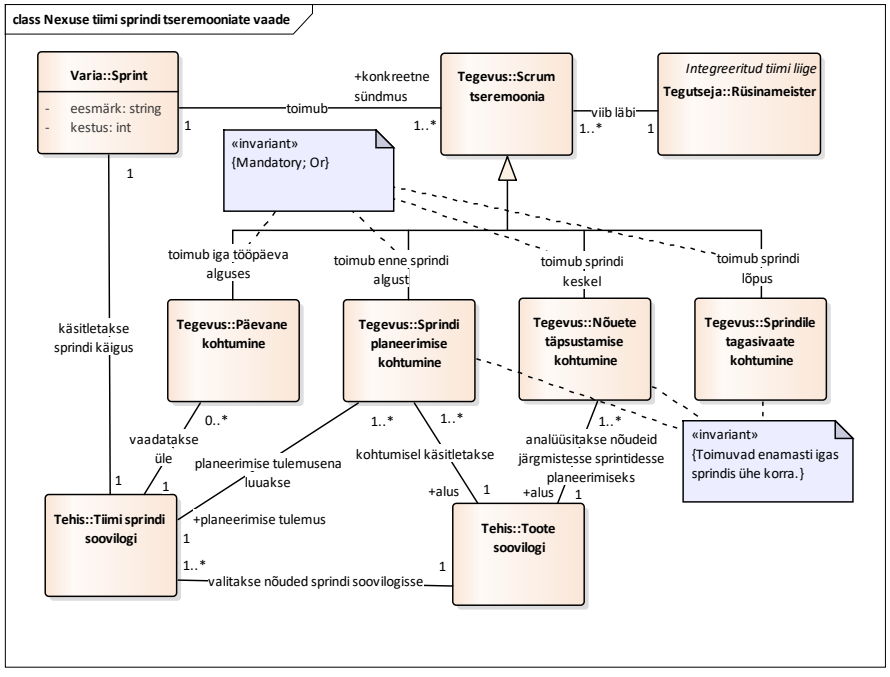


Joonis 8. Nexuse tiimi vaade.

Joonis 8 on näha, et *Nexuse* järgi peetakse optimaalseks tiimide arvuks kolm kuni üheksa Scrum tiimi. Nii nagu Scrum soovitab tiimide liikmete arvu piirata üheksa liikmega, nii soovitab *Nexus* tiimide arvu hoida maksimaalselt üheksa peal, vähendamaks keerukust ja tekitamaks ühtekuuluvustunnet. Arendustiimidesse kuuluvad arendajad ja testijad. Lisaks võivad sinna kuuluda vastavalt vajadusele teised spetsialistid, nagu analüütikud ja kasutatavuse (UX) disainerid. Üks ja sama inimene võib olla samaaegselt mitmes rollis. Arendustiim ja selle liikmed on Scrum tiimi osa. Kuna arendatakse ühist toodet, siis antud tootel on üks tooteomanik (*Product Owner*) ja kõiki *Nexuse* Scrumi tiime juhib üks ja sama tooteomanik. Igal Scrum tiimil võib olla oma rüsinameister (*Scrum Master*), kuid võimalik on ka olukord, kus mitmel arendustiimil võib olla üks ja sama rüsinameister. Tooteomanikku ei loeta ühegi Scrum tiimi liikmeks, kuid rüsinameister kuulub Scrum tiimi hulka. Tooteomanik vastutab toote maksimaalse väärtuse eest, rüsinameister vastutab, et tiimid töötaksid korrektselt *Nexuse* metoodika järgi, andes seeläbi

maksimaalset väärtust. Arendustiim vastutab uue funktsionaalsuse arendamise ja valmis arendustulemuse tarne eest. Kõik *Nexuse* tiimid peavad kasutama ainult Scrum arendusmetoodikat. Toote jaoks on ära defineeritud, mida tähendab, et see toode on „valmis“ * (*Definition of Done* ehk DoD). Iga tiim võib defineerida endale tugevamaid kriteeriumeid ehk omada tiimikohast „valmis“ definitsiooni määratlust, kuid see ei tohi olla kehvem kui kõikidele tiimidele kehtiv ühine toote „valmis“ definitsioon. [12] [53]

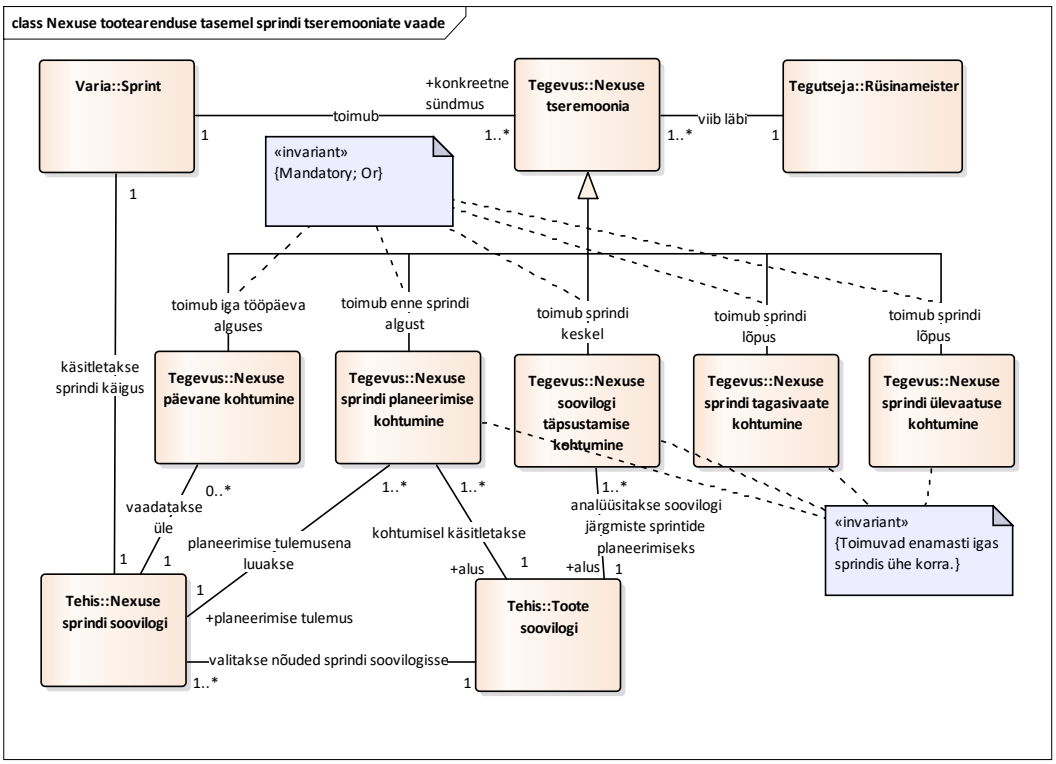
Nexuse raamistikku lisandub võrreldes Scrum’iga täiendav rollide kogum, milleks on *Nexuse* integreeritud tiim (*Nexus Integration Team* ehk NIT), mille eesmärk on edendada ja juhendada *Nexuse* rakendamist nii Scrum tiimides kui kogu organisatsioonis sisemiselt ning tagada eeldused, et tiimid suudaksid ise integreeritud juurdekasvu (uut ja tervikuks kokkupandud funktsionaalsust) toota. NIT vastutab integreeritud toote maksimaalse võimaliku väärtuse saavutamise eest ja tagab, et iga iteratsioon annab tulemuseks vähemalt ühe juurdekasvu. NIT’i koosseisu kuulub alati tooteomanik. Lisaks võivad sinna kuuluda rüsinameister(id) ja osa Scrum tiimide liikmetest, aga ka teised asjatundjad organisatsiooni erinevatest valdkondadest. Selliste asjatundjate näiteks on arhitektid, kes võivad aidata *Nexuse* järgi tarnida integreeritud juurdekasvu ja kes on ajutiselt NIT liikmed kuni selleks on vajadus. NIT liikmetel peab olema „õpetajalik“ mõtteviis, saamaks aru, millised on Scrum tiimide probleemid ning juhendamaks Scrum tiime ise oma probleemidele lahenduste leidmisel. [12] [60]



Joonis 9. Nexuse tiimi sprindi tseremooniate vaade.

Nexuse raamistik lisab klassikalistele Scrum tseremooniatele neli täiendavat sündmust ja asendab ühe Scrum tseremoonia *Nexuse* tasemel tseremooniaga. [12]

Joonis 9 toob välja, et tiimide Scrum tseremooniad on samad nagu Scrum tseremooniad ikka. Ainukene erinevus on selles, et ei toimu individuaalseid tiimide sprindi läbivaatuseid (*Sprint Review*). Sprindi läbivaatus tehakse *Nexuse* tasemel kõikide tiimidega koos, sest arendatav toode on ühine ja mõistlik on kogu sprindis tarnitav funktsionaalsus üheskoos läbi vaadata. Ühes sprindis toimub mitu erinevat Scrum tseremooniat ja iga konkreetne Scrum sündmus toimub ühes konkreetses sprindis. Üks ja sama rüsinameister vastutab kõigi tseremooniate läbiviimise eest. Sprint kestab üks kuni neli nädalat ning igal sprindil on konkreetne eesmärk. Sprindi jooksul käsitletakse konkreetset tiimi sprindi soovilogi (täitmata nõuete nimekiri) (*Scrum Team Sprint Backlog*) [5], milles sisalduvad nõuded valitakse toote soovilogist (*Product Backlog*) [5]. Päevasel kohtumisel vaadatakse üle tiimi sprindi nõuete nimekiri, mis on loodud ühe või mitme sprindi planeerimise tulemusel. Sprindi planeerimisel käsitletakse ühte toote nõuete nimekirja, mis on sprindi planeerimise aluseks. Samuti on see alus nõuete täpsustamise kohtumisel. [12] SAFe raamistiku Scrum tseremooniate kirjeldused kehtivad ka *Nexuse* raamistiku puhul (vt Joonis 5 kirjeldused).



Joonis 10. Nexuse tootearenduse tasemel sprindi tseremooniate vaade.

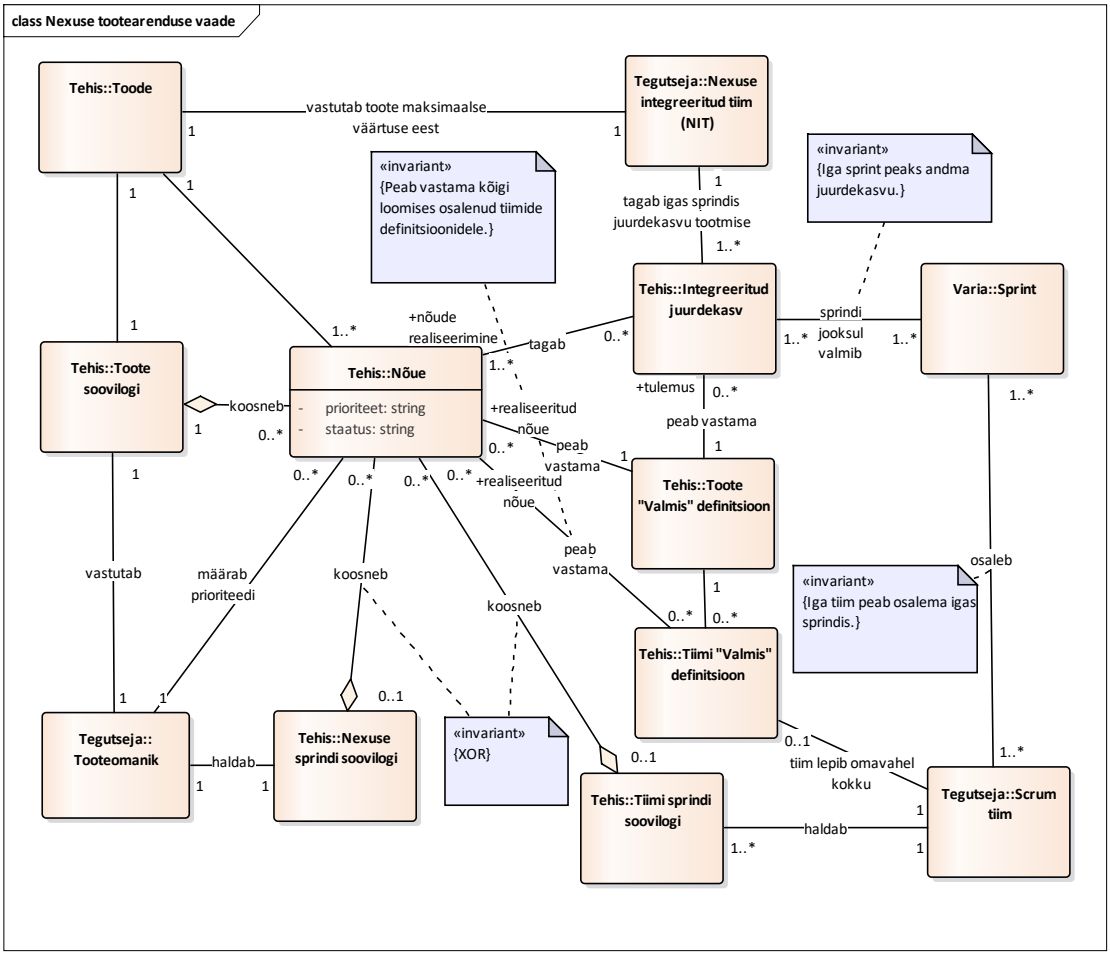
Joonis 10 toob välja viis *Nexuse* Scrum tseremooniat, kus osalevad osapooled erinevatest tiimidest. Iga tseremooniat viib läbi rüsinameister ja need toimuvad sprindi jooksul enamasti ühe korra, väljaarvatud päevane kohtumine, mis toimub iga tööpäeva alguses. Need tegevused laiendavad Scrum metoodikat tagamaks töö jaotamise ja koordineerimise ning kogemuste vahetamise *Nexuse* Scrum tiimide vahel kõige tõhusamal viisil. Sprindi jooksul käsitletakse konkreetset *Nexuse* sprindi soovilogi (*Nexus Sprint Backlog*) [5], milles sisalduvad tööd valitakse toote soovilogist (*Product Backlog*) [5]. *Nexuse* sprindi soovilogi on kogum individuaalsetest sprindi töödest, mis sõltuvad teistest tiimidest. See on justkui sprindi plaan kõikide tiimide lõikes, mis on mõeldud nende omavaheliste sõltuvuste esiletoomiseks ja jälgimiseks. Tiimidevaheliste sõltuvusteta tiimi sprindi töid seal ei kajastata. [12]

Järgnevalt on väljatoodud *Nexuse* tseremooniad ja nende sisu:

- *Nexuse* päevane kohtumine (*Nexus Daily Scrum*) [23] on igapäevane lühike infovahetuse koosolek, mis toob kokku tiimide esindajad, et kontrollida integreeritud juurdekasvu hetkeseisu, tuvastada probleemid ja takistused, käsitleda tiimide tööde omavahelisi sõltuvusi ja jagada üle tiimide infot.
- *Nexuse* sprindi planeerimise kohtumine (*Nexus Sprint Planning*) [23] aitab tiimidel üheskoos panna paika sprindi eesmärgi. Sisendiks on täpsustatud toote nõuete nimekiri ning selle tulemusel luuakse *Nexuse* sprindi nõuete nimekiri. Antud tegevus aitab sünkroniseerida tiimidevahelisi tegevusi ja töid. Kui *Nexuse* sprindi eesmärk on paigas, siis tiimi sprindi planeerimisel (*Scrum team Sprint Planning*) luuakse oma sprindi soovilogi (Joonis 9) ja tegeletakse sõltuvuste vähendamise ja kaotamisega. *Nexuse* sprindi planeerimine on lõppenud kui iga Scrum tiim on lõpetanud oma individuaalse sprindi planeerimise.
- *Nexuse* soovilogi täpsustamise kohtumine* (*Refinement*) on sündmus, mille raames toote soovilogist (*Product Backlog*) [5] olevaid töid kirjeldatakse detailsemalt ning otsitakse tiimidevahelisi sõltuvusi. Kõikidest tiimidest osalevad sellel nii paljud liikmed kui on vaja nimekirjas olevate arendustööde ehk nõuete mõistmiseks ja tükeldamiseks. Selle tseremoonia toimumise sagedus sõltub sõltuvuste hulgast.
- *Nexuse* sprindi tagasivaate kohtumine (*Nexus Sprint Retrospective*) [23] aitab tiimidel jagada kogemusi ja koordineerida ühiste väljakutsete lahendamist. Tiimide esindajad kirjeldavad oma probleeme ja tõstavad esile muud teemad,

mida tuleb käsitleda. Scrum tiim teeb lisaks oma individuaalse tagasivaate sprindile (Joonis 9) ja esindajad kohtuvad veelkord arutamaks tiimide poolt väljatoodud ühiseid probleeme, et nendest õppida ja vajalikke jätkutegevusi kindlaks määrata.

- *Nexuse* sprindi ülevaatuse kohtumine (*Nexus Sprint Review*) [6] viiakse läbi sprindi lõpus eesmärgiga demonstreerida sprindi jooksul valmis saadud funktsionaalsust ning saada äriomanikelt ja tellijatelt integreeritud juurdekasvu kohta tagasisidet. Antud tseremoonia viiakse läbi kõikide tiimidega koos. [12] [51]



Joonis 11. Nexuse tootearenduse vaade.

Joonis 11 järgi on kõikidel tiimidel ühine toote soovilogi (*Product Backlog*) [5], mille eest vastutab tooteomanik. Toote soovilogi koosneb nõuetest tootele (*Product Backlog Item*). Tooteomanik määrab soovilogis oleva nõude prioriteedi. Lisaks iseloomustab nõuet selle staatus. *Nexuse* sprindi soovilogi (*Nexus Sprint Backlog*) ja tiimi sprindi soovilogi (*Team Sprint Backlog*) [5] koosnevad toote soovilogist valitud nõuetest. Ühel

ajahetkel saab üks ja sama nõue olla korraga ainult *Nexuse* sprindi soovilogis või tiimi sprindi soovilogis. Iga tiim peab osalema igas sprindis. Scrum tiimil on ühel ajahetkel üks individuaalne sprindi soovilogi, mida käesoleva sprindi jooksul käsitletakse. Ühe või enama nõude realiseerimine tagab integreeritud juurdekasvu (*Integrated Increment*), mida tarnitakse iga sprindi tulemusel. Sprindi eesmärgiks on tekitada juurdekasvu. NIT vastutab toote maksimaalse väärtuse tagamise eest ja vastutab seeläbi igas sprindis juurdekasvu tarnimise eest. Integreeritud juurdekasv on kõikide tiimide arenduste kokkupandud tulemus, mis on valmis kasutajate töökeskkonda (*live* keskkonda) panemiseks. Selleks peab see vastama toote „valmis“ definitsioonile (*Definition of Done* ehk DoD). Tooteomanik koostöös Scrum tiimidega lepivad selle jaoks omavahel kokku kvaliteedikriteeriumid, millele toote täiendused peavad vastama ja Scrum tiimid peavad sellest kinni pidama. Nõude realiseerimisel tekkinud toote osa peab samuti vastama toote „valmis“ definitsioonile. Tiim võib defineerida endale rangemaid kriteeriumeid, kuid need ei tohi olla kehvemad kui vaikimisi kõikidele tiimidele kehtivas „valmis“ definitsioonis on määratud. Seega tiimi „valmis“ definitisoon peab vastama toote „valmis“ definitsiooni tingimustele. Tiimi poolt nõude realiseerimisel tekkinud toote osa peab vastama tiimi „valmis“ definitsioonile. Nõue loetakse täidetuks, kui realiseeritud funktsionaalsus on edukalt integreeritud juurdekasvule lisatud. [12]

Kuna *Nexuse* tootearendus on ainult ühe toote põhine, siis puudub sellises töökorralduses detailne portfelli juhtimine. Kõik otsused tootearendust puudutavate küsimuste kohta võetakse vastu NIT’is. [12]

5.3 Large-Scale Scrum metoodika

Large-Scale Scrum (LeSS) on raamistik, mis põhineb Scrum põhimõtetel, mida on rakendatud mitmetele tiimidele korraga ühe ja sama toote sünkroonsel arendamisel. LeSS kirjeldab, kuidas Scrum põhimõtteid, eesmärki ja elemente võimalikult hõlpsalt suures arendusprojektis rakendada. LeSS defineerib mitmed printsiibid, kuid üheks peamiseks printsiibiks antud raamistiku puhul on „*More with LeSS*“ („Rohkem LeSS’iga“ ehk vähem on rohkem (sõnamäng)). See tähendab, et LeSS ei pea vajalikuks paljude erinevate rollide olemasolu, sest iga uus roll tähendab tiimi jaoks väiksemat vastutust. LeSS ei soovi kasutada üleliia tehiseid, kuna see raskendab suhtlust tiimide ja kliendi vahel. LeSS ei taha arvukaid tseremooniaid/protsesse, sest mida rohkem protsesse, seda rohkem tuleb

tiimil osata nendega kohaneda. LeSS tahab anda tiimidele väiksema arvu rollide kaudu rohkem vastutust. LeSS tahab rohkem kliendikeskseid tiime (keskendudes kõikide ettevõtte pakutavate teenuste ja toodete kaudu kliendi jaoks positiivsete kogemuste loomisele [4]) kasutades saavutada kasulikke tooteid, luues selleks väiksemat arvu tehiseid. LeSS tahab suuremat tiimi osalust protsessides ja tähendusrikkamat tööd pakkudes vähem protsesse, milles inimesed peavad osalema. LeSS autoriteks on Craig Larman ja Bas Vodde. LeSS raamistikku võib rakendada sadade, aga ka tuhandete osalejatega arendustes. *Large-Scale Scrum* raamistikke on kaks ning need on kirjeldatud tiimide arvu alusel. [52] [61]

1. LeSS (väike): kaks kuni kaheksa tiimi.
2. LeSS Huge (tohutu suur): üle kaheksa tiimi. [52] [61]

Maksimaalselt kaheksat tiimi loetakse väiksema LeSS raamistiku ülempiiriks, sest sellise mahuga suudab tooteomanik oma tegevust veel täpselt suunata ja tootearendust hallata. Kui üks tooteomanik ei suuda kogu tootest enam ülevaadet omada, siis ta ei suuda kliendi ja tiimide vahel tähelepanu tasakaalustada ja toote soovilogi on ühele inimesele haldamiseks liialt suur. Sellisesse olukorda jõudmisel tasub mõelda väiksemalt LeSS raamistikult suuremale üleminekule. [52]

LeSS tiim töötab lähestikku lõppkasutajate ja kliendiga, et selgitada välja soovilogis olevate tööde detailid. LeSS tiime iseloomustab tiimi liikmete pühendumus ühele tiimile, võimekus tiimiliikmete oskuseid kombineerides tarnida süsteemi osi, tiimi liikmete ühises ruumis töötamine ja võimalikult pikaaegne samas koosseisus töötamine. Need neli aspekti tagavad ühiste eesmärkide poole püüdlemise ja vastutuse võtmise. [10]

LeSS ja LeSS *Huge* metoodikatel on järgnevad ühised omadused:

- Üks tooteomanik (*Product Owner*).
- Üks toote soovilogi (*Product Backlog*).
- Ühine sprint üle kõikide tiimide.
- Üks tarnitav toote juurdekasv (*Shippable Product Increment*). [10] [52] [62]

LeSS raamistikus on tooted ja teenused defineeritud võimalikult laialt ja projekti portfelli haldamine on minimaalne. LeSS nõuab edukaks raamistiku juurutamiseks organisatsiooni keerukuse vähendamist ja struktuuri lihtsustamist. LeSS raamistik ütleb,

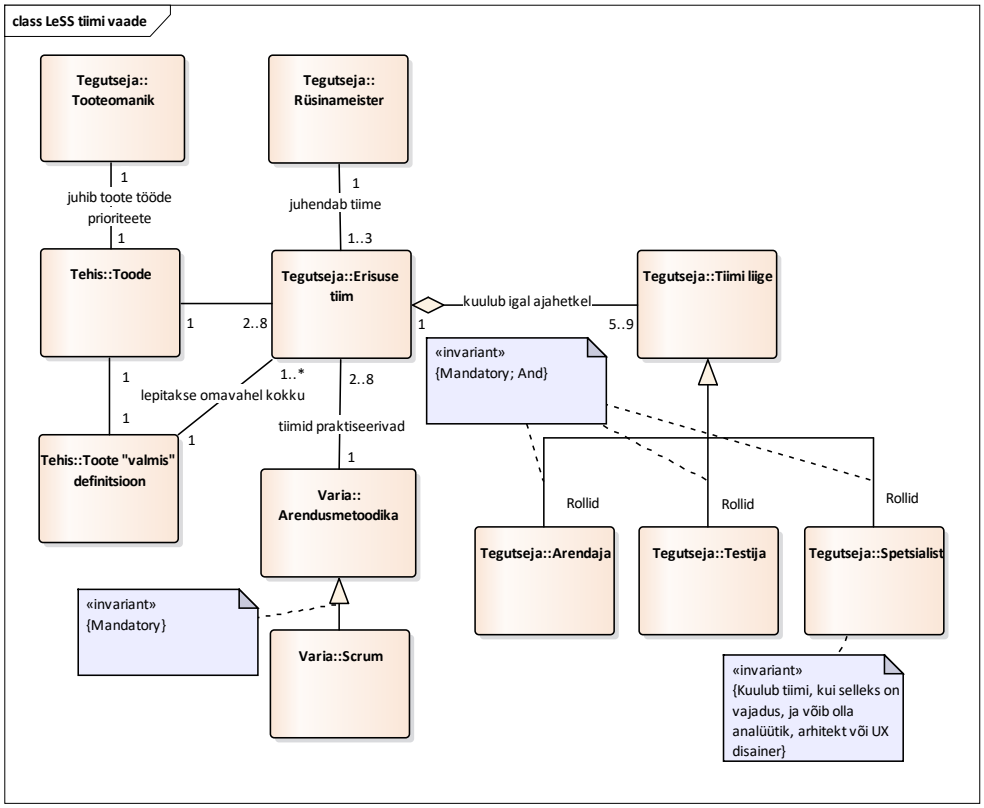
et juhtide (*Manager*) olemasolu on valikuline. Kui nad eksisteerivad, siis muutuvad aga nende vastutusalad. Juhtide peamine vastutus on lahendada organisatsioonilisi ja süsteemseid takistusi ning parendada organisatsiooni käekäiku. Otsus selle kohta, mille kallal tiim töötab, ei ole enam juhtide kontrolli all, vaid seda otsustab tooteomanik. Tooteomanikul on kõige parem ülevaade tootest ja sellest, mis on klientide jaoks kõige väärtuslikum. Juhtidel (*Manager*) pole selles osas mingisugust rolli ning nad ei tohiks tooteomaniku poolt otsustatud tootearenduse detailidele vastu seisa. Keskastme juhtkonna roll on näha tervikut ja tagada organisatsiooni võimekust luua suurepäraseid tooteid. Nad peaksid aitama tiimi ja rüsinameistrit probleemide ja takistuste eemaldamisel. Kõrgema juhtkonna roll nõuab vähem muutmist, kuna nad tegelevad endiselt ettevõtte ja selle tootega seotud strateegiliste otsustega. Kuid ka kõrgema taseme juhtide ülesanne on probleemide lahendamiseks inimeste juhendamine. [10]

Järgnevates jaotistes tuuakse välja autori loodud metamudelid LeSS ja LeSS *Huge* metoodikate kohta ning kirjeldatakse metoodikale omaseid rolle, tegevusi ja tehiseid.

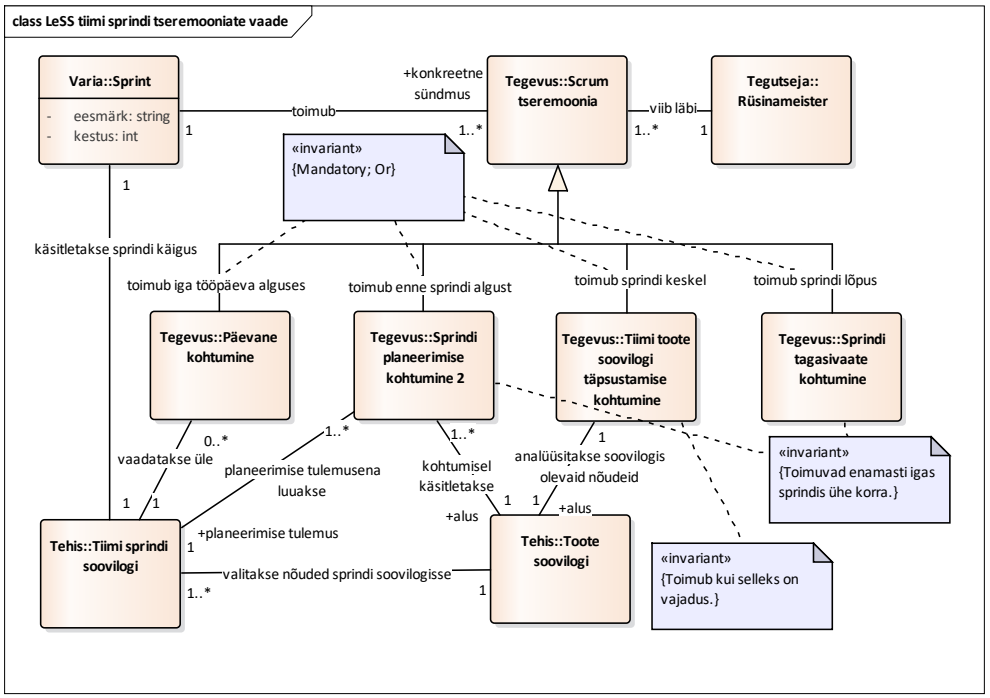
5.3.1 LeSS metoodika

LeSS raamistiku tehised, tegevused ja tegutsejad on peamiselt samad, mis ühel Scrumi praktiseerival tiimil. Joonis 12 on metamudel LeSS tiimi vaate kohta. Antud mudelilt on näha, et LeSS tiime nimetatakse tunnusjoone tiimideks [46] või erisuse tiimideks [63] (*Feature team*). Erisuse tiimiks nimetatakse koosseisu, kellel on vajalikud teadmised ja oskused tegemaks algusest lõpuni kliendikeskset uut funktsionaalsust (*Feature*), arendades selleks vajalikke komponente ja töötades koos ühise koodiga. LeSS metoodika järgi saab olla kaks kuni kaheksa tiimi. Kuna kõik tiimid arendavad ühist toodet, siis on neil ka ühine tooteomanik, kes vastutab toote maksimaalse väärtuse eest ja määrab tiimide tööde tähtsuse järjekorra. Igal tiimil on üks rüsinameister. Üks ja sama rüsinameister võib vastutada ühe kuni kolme tiimi eest ning ta ei ole tiimi liige. Rüsinameistri roll ei ole tiimide vahelise koordineerimise eest vastutamine, vaid juhendada neid ise koordineerima, jälgides, et tiim oleks tootearendusele keskendunud. Rüsinameister juhendab tiime, aidates neil ise jooksvatele probleemidele lahendusi leida ja kõrvaldada takistusi, mis skaleeritava tootearenduse puhul on alati keerulised. Tiimid praktiseerivad ainult Scrum arendusmetoodikat. Tooteomanik koostöös erisuse tiimidega lepivad kokku toote „valmis“ definitsiooni (*Definition of Done*). Erisuse tiimis võib olla viis kuni üheksa

liiget, kes on peamiselt arendajad ja testijad, kuid vajadusel ka analüütikud, arhitektid ja/või kasutatavuse (UX) disainerid. Üks ja sama liige võib olla mitmes rollis. [10] [52]



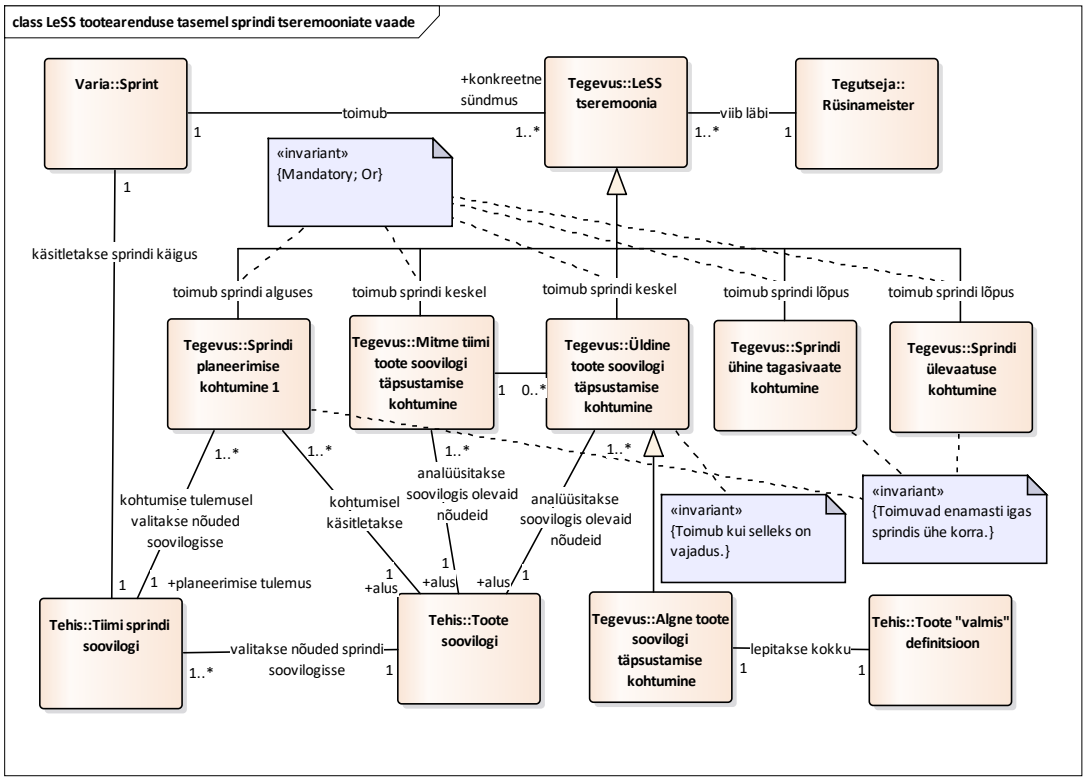
Joonis 12. LeSS tiimi vaade.



Joonis 13. LeSS tiimi sprindi tseremooniate vaade.

Scrum tseremooniaid võib eristada nii tiimi tasemel, kus osalejateks on ainult tiimi liikmed, kui ka üldiselt LeSS tasemel, mis kaasab tiimidest erinevaid osapooli. Joonis 13 esitab LeSS raamistiku tiimi sprindi tseremooniate vaate. SAFe raamistiku Scrum tseremooniate sisu kehtib ka LeSS raamistiku puhul (Joonis 5 kirjeldused). Tiimide tseremooniad ja nende sisu on järgmine.

- Sprindi planeerimise kohtumised toimuvad kahes osas. Sprindi planeerimise kohtumine 1 (*Sprint Planning One*) on LeSS taseme tseremoonia (vt kirjeldust Joonis 14). Sprindi planeerimise kohtumise 2 (*Sprint Planning Two*) raames keskendub iga tiim oma sprindi plaani koostamisele, mis paneb paika arendusstrateegia kuidas ja millal iga planeeritud nõue valmis saada. Kohtumise aluseks on toote soovilogi (*Product Backlog*) ja tulemiks on sprindi soovilogi (*Sprint Backlog*). Tihtipeale arendavad mitu tiimi ühte ja sama funktsionaalsust ja/või komponenti. Sellisel juhul ühinevad need tiimid planeerimise teise osa jaoks, et läbirääkida sõltuvused ja tagada, et tiimide sprindiplaanid ei oleks üksteisega vastuolus. Teise osa lõppemise tulemusel kajastub kogu sprindi plaan sprindi soovilogis.
- Tiimi toote soovilogi täpsustamise kohtumine (*Single-Team Product Backlog Refinement*) on vajadusel korraldatav kohtumine, kui tiimil on vaja individuaalselt mõnda arenduse nõuet täpsustada. Kohtumise aluseks on toote soovilogi. See kohtumine võib olla vajalik mõne suurema arendustöö puhul, mille täpsustamine on eelduseks ka teiste tiimide tööde täpsustamisel. [10] [52]



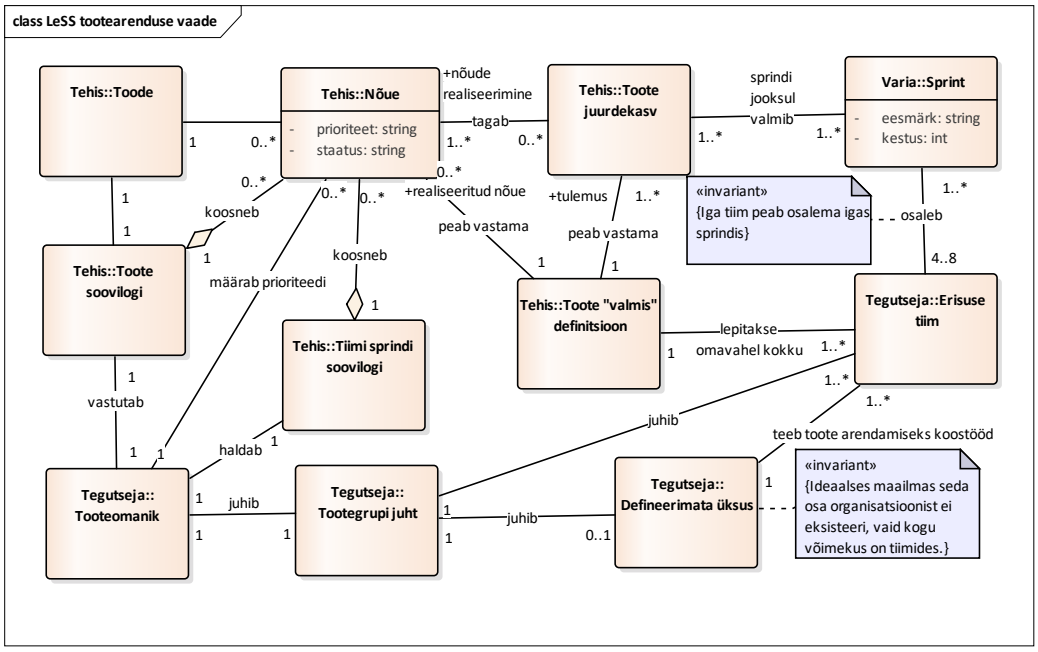
Joonis 14. LeSS tootearenduse tasemel sprindi tseremooniate vaade.

Joonis 14 on toodud ära LeSS tasemel sprindi tseremooniate vaade, kus toimuvad sündmused kaasavad erinevaid tiime või nende esindajaid. Nende sisu on järgmine.

- Päevane kohtumine (*Daily Scrum*) toimub igas tiimis individuaalselt. Eraldi kohtumist, kuhu tuleksid kokku kõik tiimid või nende esindajad ei korraldata. Kui tekib täiendav koordineerimise vajadus, siis lepitakse vastav kohtumine jooksvalt kokku.
- Sprindi planeerimise kohtumine 1 (*Sprint Planning One*) keskendub sprindi eesmärgi määratlemisele ja toote soovilogist järgmise sprindi nõuete valimisele. Lisaks tegeleb see vajadusel nõuete detailide täpsustamisega. Sellel kohtumisel osalevad tooteomanik, kes valib prioriteedi alusel nõuded ja igast tiimist enamasti kaks esindajat. Vastavalt vajadusele võivad osaleda ka tiimid täiskoosseisus [64].
- Mitme tiimi toote soovilogi täpsustamise kohtumine (*Multi-Team Product Backlog Refinement*) on kõige olulisem tööde täpsustamise kohtumine, kuna see hõlmab kõiki tiime. Üldine toote soovilogi täpsustamise kohtumine (*Overall Product Backlog Refinement*) toimub juhul kui on vajadus jagada tiimide vahel suuri toote arendustöid. See on lühike kohtumine esmase tükeldus tegemiseks kui see on üldse vajalik. Enamasti võib selle vahele jätta ning alustada kohe mitme

tiimi ülesest toote soovilogi täpsustamise kohtumisest. Mõlema kohtumise aluseks on toote soovilogi. LeSS defineerib lisaks algse toote soovilogi täpsustamise kohtumise (*Initial Product Backlog Refinement*), mis toimub ühe korra enne esimese sprindi algust ja selle eesmärgiks on leppida kokku toote „valmis“ definitsioon (*Definition of Done*).

- Sprindi ühine tagasivaate kohtumine (*Overall Retrospective*) on täiendav Scrum tseremoonia LeSS raamistikus. Selle eesmärk on arutleda tiimidega organisatoorseid ja muud arendustegevust puudutavaid probleeme. Ühisel sprindi tagasivaate kohtumisel osalevad tooteomanik, rüsinameister ja tiimide esindajad ning see leiab aset peale tiimide individuaalseid sprindile tagasivaate kohtumisi.
- Sprindi ülevaatuse kohtumisel (*Sprint Review*) osalevad enamasti igast tiimist kaks esindajat. Seal vaadatakse äripoolega üle tootesse sprindi tulemusel tehtud täiendused ehk toote juurdekasv* (*Shippable Product Increment*). [10] [52]



Joonis 15. LeSS tootearenduse vaade.

Joonis 15 on toodud ära olulised tootearenduse elemendid. Tootel on üks toote soovilogi (*Product Backlog*), mille eest vastutab tooteomanik. Toote soovilogi koosneb mitmest nõudest (*Product Backlog Item*), millest igaühel on prioriteet ja staatus. Tooteomanik määrab iga nõude prioriteedi. Tiimi sprindi soovilogi koosneb samuti nõuetest. Nõude realiseerimine võib kuid ei pruugi tagada toote juurdekasvu (*Shippable Product Increment*). Toote juurdekasvu tekkimiseks peab alati vähemalt ühe nõude realiseerima.

Lisaks peab toote juurdekasv vastama toote „valmis“ definitsioonile (*Definition of Done*), mille lepivad kokku omavahel tooteomanik ja erisuse tiimid (*Feature teams*). Realiseeritud nõue peab samuti vastama toote „valmis“ definitsioonile. Erisuse tiimid osalevad sprindis, mille jooksul valmib toote juurdekasv. Tiimid koordineerivad ja integreerivad oma arendused teiste tiimidega. Eesmärk on, et sprindi lõpuks oleks tiimide ühispingutuse tulemusel tekkinud terviklik tarnitav toote juurdekasv (*Shippable Product Increment*). [10] [52]

LeSS organisatsiooni struktuur on üsna lihtne, st siin puudub projekti ja programmi juhtimine. Nende üksuste tavapärased ülesanded on erisuse tiimide ja tooteomaniku vastutused. LeSS organisatsiooni struktuuri kuuluvad lisaks erisuse tiimidele ja tooteomanikule järgmised olulised osapooled.

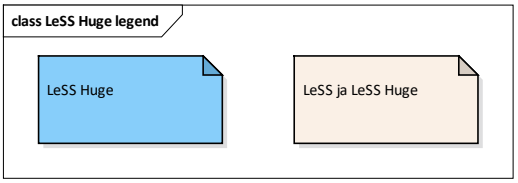
- **Tootegrupi juht** (*Head of Product Group*), kes on kõikide tiimide juht ning tema ülesandeks on aidata tiimidel lahendada erinevaid probleeme ja takistusi.
- **Defineerimata üksus** (*Undone department*), mida ideaalses maailmas organisatsioonis ei eksisteeri. On olukordi, kus tiim ei suuda ise toote juurdekasvu täielikult oma tiimisiseselt tekitada. Siia alla kuuluvad nende tegevuste läbiviijad, mis on vajalikud juurdekasvu saavutamiseks, näiteks testimine, arhitektuur, ärianalüüs ja mille tegijate esinemise tõenäosus tiimist väljaspool on LeSS *Huge* metoodika rakendamisel üsna suur. [10]

5.3.2 LeSS Huge metoodika

Kui ühte toodet arendab üle saja inimese, siis on arenduse lihtsustamine ja tükeldamine nii paljude erinevate nõuete ja osaliste tõttu keerukuse tõusu vältimiseks vältimatu. Kui tootearenduses osaleb üle kaheksa tiimi, siis nende jaotamisel lähtutakse *LeSS Huge* raamistikust. Keerukuse haldamine tükeldamise kaudu lähtub kliendi peamistest probleemivaldkondadest. *LeSS Huge* sisuks on LeSS raamistiku täiendav skaleerimine. [10]

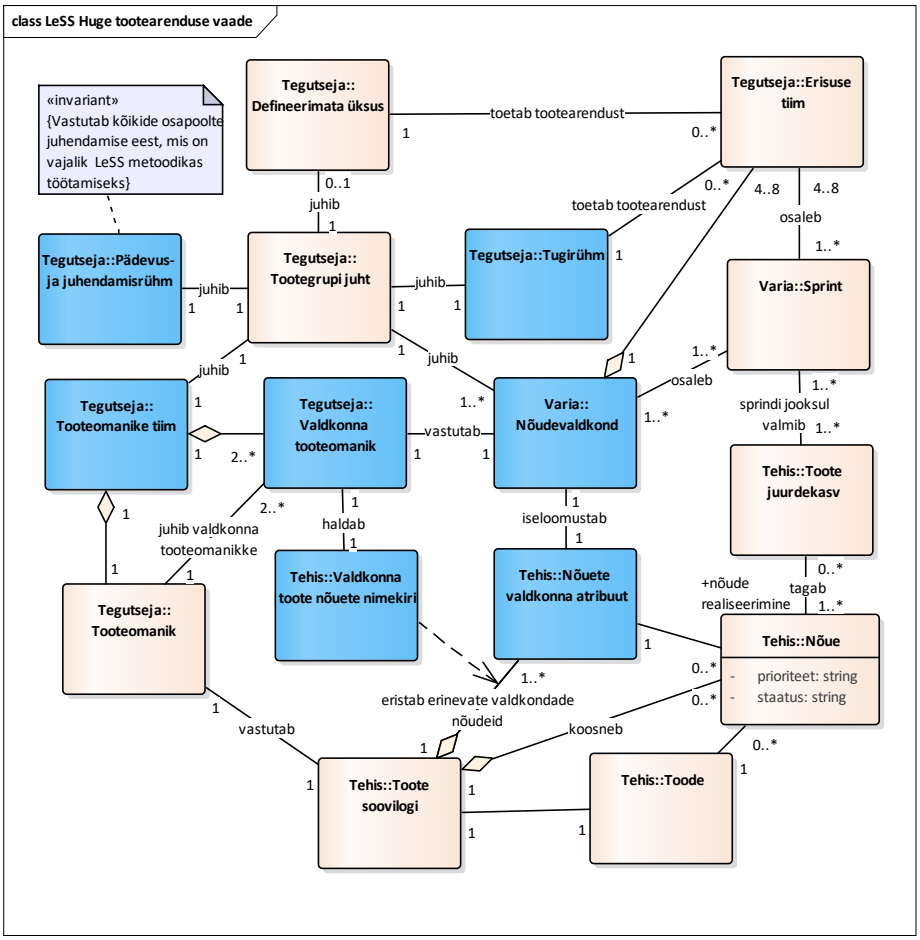
Kõik väiksema LeSS raamistiku kontseptsioon, st ka autori poolt loodud metamudelid ja nende kirjeldused, kehtivad ka LeSS *Huge* puhul. Eraldi esitab autor täiendava vaate LeSS *Huge* tootearenduse kohta, sest täiendav skaleerimine nõuab ka lisanduvaid rolle ja struktuuri. Kuigi täiendava struktuuri lisamine põhjustab tavaliselt ka vastutuste ümber jaotamist, mis tähendab väiksemaid vastutusi tiimidel, siis LeSS *Huge* proovib seda vältida hoides ettevõtte struktuuri võimalikult lihtsana. [10]

Järgnevalt esitatakse legend, mille alusel LeSS Huge tootearenduse vaade on loodud:



Joonis 16. LeSS *Huge* legend.

Joonis 16 on näha, et sinine värv eristab mudelil elemente, mis on täiendused LeSS *Huge* metoodikas. Beeži värviga näidatakse elemente, mis on mõlemal metoodikal ühised.



Joonis 17. LeSS *Huge* tootearenduse vaade.

Joonis 17 on metamudel kirjeldamaks LeSS *Huge* raamistiku tootearenduse vaadet. Nõudevaldkonnad (*Requirement Areas*) tekivad lähtudes kliendist. Iga valdkonnaga tegeleb neli kuni kaheksa erisuse tiimi. Töö valdkonnaga on dünaamiline ehk igal ajahetkel võib selleks ettenähtud tiimide või nende liikmete arv kasvada või kahaneda. See sõltub valdkonna tähtsusest ja selle muutumisest. Tiimid võivad ka liikuda ühest

valdkonnast teise. Kõikide nõudevaldkondade tootearendus toimub ühises sprindis. Sprint lõpeb ühise toote juurdekasvu tootmisega üle kõikide nõudevaldkondade. Toote juurdekasv tekib ühe või mitme nõude realiseerimisel. Iga toote kohta on mitmeid nõudeid. Igal tootel on üks toote soovilogi, mis koosneb nõuetest. Toote soovilogisse (*Product Backlog*) lisatakse nõudevaldkonna atribuut (*Requirement Area attribute*) ja iga logis olev nõue on klassifitseeritud ainult ühe valdkonna alla, mille tulemusel tekib toote valdkonna soovilogi (*Area Product Backlog*). [10] [52]

LeSS *Huge*'s on võrreldes LeSS raamistikuga täiendavad rollid:

- **Valdkonna tooteomanikud** (*Area Product Owners*) on eraldi igal nõudevaldkonnal. Valdkonna tooteomanik spetsialiseerub oma valdkonnale ja keskendub toote valdkonna soovilogi (*Area Product Backlog*) haldamisele ja prioritiseerimisele. Suurtes tootearenduse gruppides on tavaliselt mitmed toetavad tootejuhid spetasialiseerinud erinevatele kliendi valdkondadele ja mitmed neist tõenäoliselt kannavad ka valdkonna tooteomaniku rolli. Mõnikord on põhiline tooteomanik lisaks ka valdkonna tooteomanik ühele valdkonnale – seda esineb tõenäolisemalt väikemate tiimide arvuga LeSS Huge raamistikus. „Tooteomanik" LeSS tiimi vaates (Joonis 12) on LeSS *Huge* metoodikas „Valdkonna tooteomanik".
- **Valdkonna erisuse tiimid** (*Area Feature Teams*) on ühes ja samas nõudevaldkonnas töötavad tiimid. Tiimi perspektiivist vaadatuna ei erine LeSS *Huge* alusel töötamine LeSS aluselt töötamisest, sest tiimi töid juhib nende valdkonna tooteomanik samamoodi nagu tavaline tooteomanik. [10] [52]

Vastavalt sellele kui palju on nõudevaldkondi on LeSS *Huge*'l kaks või rohkem valdkonna tooteomanikku. Teiste sõnadega, igal nõudevaldkonnal on üks valdkonna tooteomanik. Lisaks on üks peamine tooteomanik, kes on keskendunud toote üldisele optimeerimisele, valdkondade edendamisele ja valdkonna tooteomanike juhendamisele. Kõik valdkonna tooteomanikud moodustavad tooteomanike tiimi (*Product Owner Team*), mille ülesandeks on tooteomanike omavaheline info jagamine. Iga nõudevaldkond töötab nagu (väikese) LeSS raamistiku realisatsioon. Kõikide valdkondade töö toimub ühes ühises sprindis, milles iga valdkonna tiim osaleb. [10]

LeSS *Huge* raamistiku puhul lisanduvad täiendavad struktuuriüksused ja mida juhib tootegrupi juht.

- **Tooteomanike tiim** (*Product Owner Team*), mille moodustavad kõik valdkonna tooteomanikud ning selle tegevuseks on tooteomanike omavaheline info jagamine. Selle meeskonna tegevust nimetatakse tihti ka tootehalduseks (*Product Management*). Suuremas organisatsioonis võib tooteomanikku toetada tootejuht (*Product Manager*).
- **Pädevus- ja juhendamisrühm** (*Competence and coaching*), kuhu kuuluvad praktikud ja eksperdid, kes LeSS raamistikku tunnevad ning oskavad erinevaid rolle nende töös juhendada.
- **Tugirühm** (*Support*), mille eesmärk on tagada tsentraliseeritud arenduskeskkondade tugi. Näiteks on nende vastutuses konfiguratsioonihaldus, pideva integratsiooni (*continuous-integration*) süsteemid ja muud operatiivselt vajalikud rakendused. [10]

Ühe valdkonna tiimi vaates on LeSS *Huge* sama nagu väiksem LeSS oma tseremooniatega ehk kehtivad LeSS metoodika metamudelid (Joonis 12, Joonis 13, Joonis 14). [52] LeSS tootearenduse tasemel tseremooniate joonis (Joonis 14) kehtib nüüd ühe nõudevaldkonna kohta. LeSS *Huge* metoodikasse lisandub vaid täiendav sündmus – tooteomanike tiimi kohtumine (*Product Owner Team meeting*), mida kasutatakse erinevate nõudevaldkondade tooteomanike info vahetuseks ja koordineerimiseks. [10]

5.4 Disciplined Agile Delivery metoodika

Disciplined Agile (DA) on metoodika, mis annab konkreetsed juhised aidates tiimidel ja organisatsioonil muuta oma protsesse sujuvamaks ja luues ettevõtte konteksti arvestades kindla aluse ettevõtte agiilsusele. DA metoodika autorid on Scott Ambler and Mark Lines. Antud töös käsitletakse kõige uuemat versiooni 4.x. DA on jagatud neljaks tasemeks, millest järgnevalt antakse ülevaade. [65]

Disciplined Agile Delivery (DAD) on protsessiraamistik, mis on inimestele ja pidevale õppimisele orienteeritud hübriidne ja paindlik lähenemisviis kogu IT-lahenduste tarne elutsükli jaoks. DAD on skaleeritav ja eesmärgile orienteeritud [66]. DAD pakub hübriidset lähenemist laiendades Scrum ja Kanban metoodikat teiste agiilsete

arendusmetoodikatega ja meetoditega nagu (*Agile Modeling* ehk AM), ekstreemprogrammeerimine (XP), *Lean Software Development* ning paljud teised. DAD defineerib kuus erinevat elutsüklit, mida autor kirjeldab lähemalt antud peatükis. [65]

Disciplined Devops on erinevate IT alaste tegevuste toetamine ja toimingute sujuvamaks muutmine IT-lahenduste väljatöötamiseks, et pakkuda organisatsioonile tõhusamaid tulemusi. Siia alla kuuluvad sellised funktsioonid nagu näiteks turvalisus, andmehaldus ja väljalaskehaldus (*Release Management*). [65]

Disciplined Agile IT (DAIT) määratleb, kuidas rakendada agiilseid ja *Lean* strateegiaid infotehnoloogilistes protsessides. Siia alla kuuluvad sellised funktsioonid nagu näiteks portfelli haldus, toote haldus ja inimeste juhtimine. [65]

Disciplined Agile Enterprise (DAE) omab võimekust turul toimuvaid muutusi kiiresti tunnetada ja neile reageerida. Seda tehakse organisatsiooni kultuuri ja struktuuri kaudu, mis hõlbustab muutusi antud olukorra kontekstis. Sellised organisatsioonid vajavad innovatsiooni pideva arengu tagamiseks ning selle aluseks olevaid *Lean* ja agiilseid protsesse. Siia alla kuuluvad sellised funktsioonid nagu näiteks turundus, müük ja finantsjuhtimine. [66]

DAD metoodika üks põhilisi võtmeaspekte on see, et see kirjeldab lahenduste täieliku arendamise ja tarnimise kulgu algusest lõpuni tervikuna ning see jaguneb järgnevasse kolme faasi.

1. **Algatus** (*Inception*), mille eesmärk on aru saada skoobist ja identifitseerida esmased nõuded, mille täitmine soovitakse saavutada. Antud etapis tegeletakse portfelli haldusega, määrates toodete või projektide tähtsusaste, et valida välja kõige tähtsamad. Lisaks koostatakse antud etapis lahenduse arhitektuuriline visioon ja tagatakse arenduste rahastamine. Samuti pannakse paika ka sobiv tiimi koosseis.
2. **Ehitamine** (*Construction*), mille eesmärk toota kasutatavat äriväärtust pakkuvat lahendust. Antud etapis toimub reaalne tarkvara arendamine. Kasutatakse erinevaid Scrum ja Kanban praktikaid, mida on kombineeritud erinevate teiste metoodikatega või meetoditega nagu ekstreemprogrammeerimine ja agiilne modelleerimine.

3. **Üleminek** (*Transition*), mille eesmärk on planeerida, korraldada ja koordineerida tarkvara kasutusele võtmine, et teha toodetud lahendus kliendile kättesaadavaks. Antud faasis tuleb tagada lahenduse testimine ja muud lahenduse *live* keskkonda paigaldamiseks vajalikud tegevused nagu näiteks andmete ülekandmine. Kindlasti tuleb läbi viia lõppkasutajaga testimine, et veenduda lahenduse sobivuses ja saada asjakohast tagasisidet. Siia etappi kuulub ka dokumentatsiooni korrastamine ning kasutajate- ja paigaldamisjuhendite loomine. Eesmärk on, et klient oleks lõpptootega rahul ja lahendus toimiks. [53] [65]

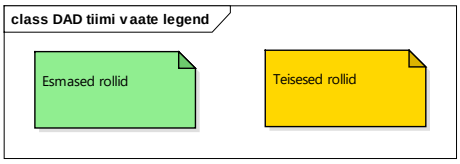
DAD metoodika defineerib kuus erinevat arenduse tsüklit.

1. **Agiilne elutsükkel** (*Agile Lifecycle*) on Scrum'i laiendav elutsükkel ning on mõeldud projektipõhise arendustegevuse jaoks.
2. ***Lean* elutsükkel** (*Lean Lifecycle*) on *Lean* printsiipe rakendav elutsükkel. Kui Scrum kirjeldab hulga tseremooniaid, mis toimuvad iteratsiooni käigus kindlal ajal, siis *Lean* seda ülemäära ette ei kirjuta ja ütleb, et neid tuleb läbi viia siis kui selleks tekib vajadus. Antud elutsükkel on samuti mõeldud projektipõhise arendustegevuse jaoks.
3. **Pideva tarne agiilne elutsükkel** (*Continuous Delivery Agile Lifecycle*) on sarnane agiilsele elutsüklile. Põhiline erinevus nende vahel seisneb selles, et antud elutsüklis peab uus funktsionaalsus tekkima iga iteratsiooni lõpus.
4. **Pideva tarne *Lean* elutsükkel** (*Continuous Delivery Lean Lifecycle*) on üsna sarnane eelmisele tsüklile. Erinevus on selles, et siin rakendatakse *Lean* põhimõtteid ja uus funktsionaalsus tarnitakse kindla ajaintervalli tagant, kokkuleppeliselt näiteks igapäevaselt või igakuiselt.
5. **Uurimuslik elutsükkel** (*Exploratory (Lean Startup) Lifecycle*) on mõeldud idufirmade või uue toote ideega tiimidele.
6. **Programmi elutsükkel** (*Program Lifecycle*) kirjeldab, kuidas panna mitmeid tiime korraga koos ühe toote kallal töötama. See nõuab programmi juhtimist (*Program Management*). [65]

Eeldatavasti suudab organisatsioon antud tsüklid kasutusele võtta, neid vastavalt vajadusele enda olude järgi kohaldades. Iga tiim peaks ise otsustama, millise elutsükli järgi ta soovib töötada. Iga tiim on unikaalne ja selleks, et DAD oleks efektiivne, peab

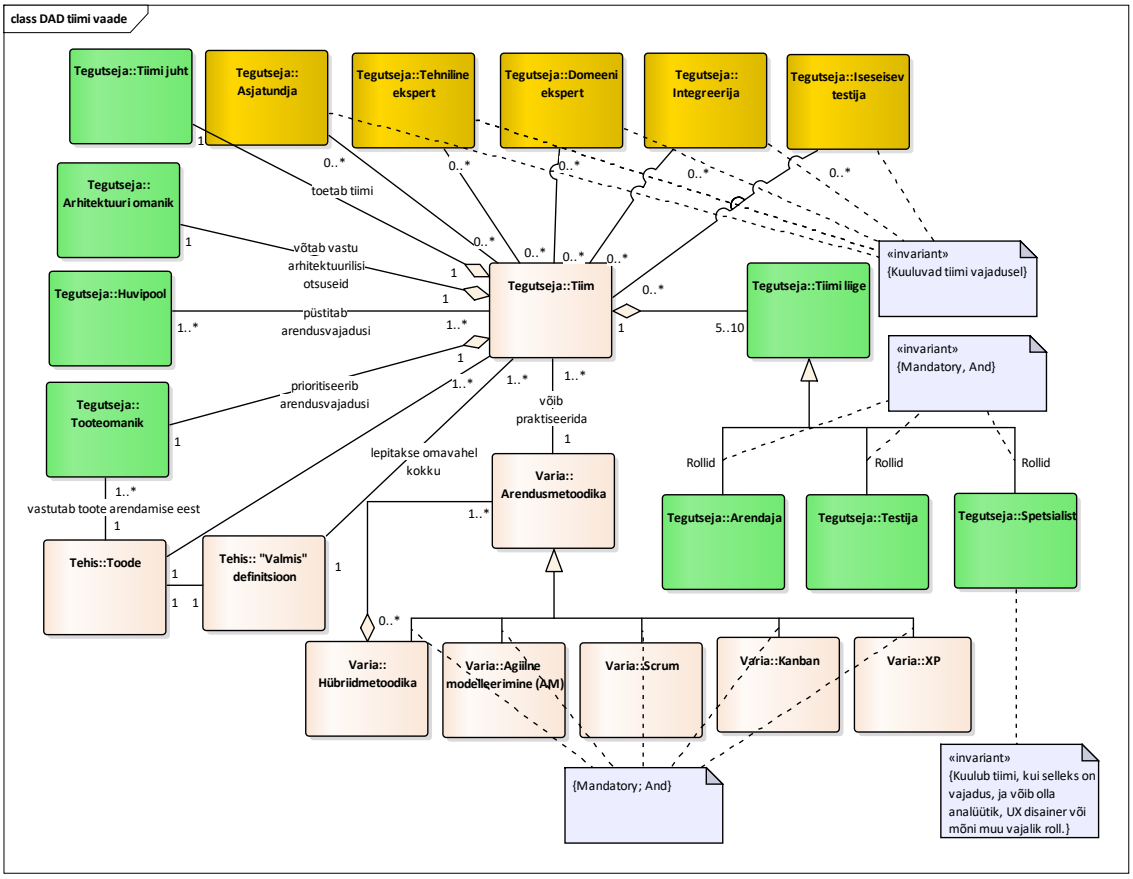
tagama võimalikult paindliku arenduse elutsükli kõikidele tiimidele. See on ka põhjus, miks DAD kirjeldab nii suure arvu erinevaid elutsükleid. [65]

Eelnevalt sai toodud välja DAD kuus erinevat elutsüklit. Kõik need on erinevad, kuid neil on omavahel ka palju kattuvusi. Autor ei näe põhjust siinkohal kõiki neid modelleerida. Esiteks seetõttu, et see oleks väga mahukas. Teiseks seetõttu, et antud elutsüklite lühikirjelduse järgi saab autor juba teha järelduse selle kohta, mida ettevõtte X tiimid saaksid kasutada. Esimesed kaks elutsüklit ei sobi, kuna need kirjeldavad projektipõhise arenduse töökorraldust. Autor leiab, et sobivaimaks modelleeritavaks elutsükliks on tiimide töökorraldust kirjeldav kolmas elutsükkel, kuna see põhineb Scrum'il ja ettevõtte eesmärk on ärile iga sprindi tagant väärtust anda. Lisaks tuleks ettevõttes kindlasti rakendada programmi elutsüklit, kuna see kirjeldab, kuidas koordineerida korraga mitmeid tiime hõlmavat tootearendust.



Joonis 18. DAD rollide legend.

Joonis 18 esitab DAD rollide legendi. DAD metoodika esmaseid rolle tähistatakse tiimi vaate joonisel rohelise värviga ja teiseseid rolle kollase värviga. Ülejäänud elemente näidatakse beeži värviga.



Joonis 19. DAD tiimi vaade.

Joonis 19 on toodud ära DAD metoodika tiimi vaade. Tiimis on tavaliselt viis kuni kümme liiget, kuhu kuuluvad arendajad ja testijad ning vajadusel ka analüütik, UX disainer või mõni muu tiimile vajalik roll. Üks ja sama liige võib olla mitmes rollis. Tiim võib praktiseerida erinevaid arendusmetoodikaid nagu Scrum, XP, AM, Kanban ja mitmeid veel. Lisaks toetab DAD laialdaselt nende hübriidide kasutamist. DAD metoodika järgi võivad eksisteerida komponent-tiimid kui ka erisuse tiimid, aga ka arenduse funktsiooni järgi organiseeritud funktsionaalsed tiimid (*Functional team*), nagu näiteks arenduse tiim, testimise tiim ja paigaldamise tiim. Tootele on loodud „valmis“ definitsioon, mille tiim omavahel kokku lepib. DAD defineerib eraldi esmased rollid, millest enamik moodustavad tiimi ning need on alati esindatud. Lisaks defineerib DAD teisesed rollid, mis on tiimi tööd toetavad. Teisesed rollid täidetakse tavaliselt ajutiselt, eesmärgiga tiimi mõne konkreetse probleemi korral nõustada või lahendamaks skaleerimisega seotud probleeme. [65]

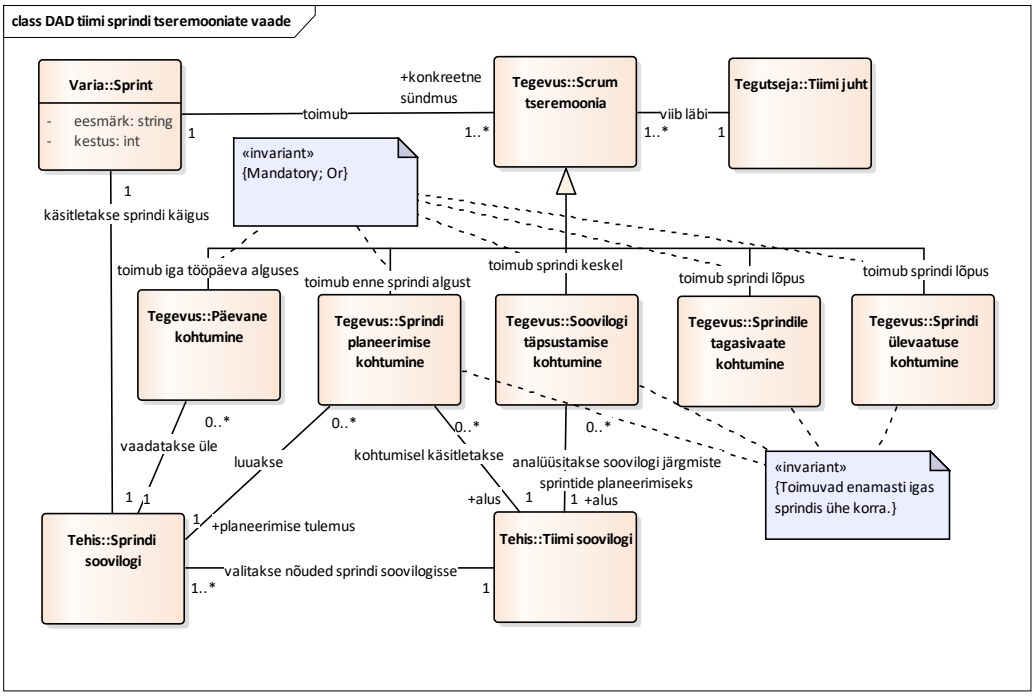
DAD metoodika kirjeldab järgnevad esmased rollid.

- **Tiimijuht** (*Team Lead*) on sarnane rüsinameistri rolliga. Selle vastutus on luua tiimi sujuvaks tööks vastavad tingimused. Igal tiimil on üks tiimijuht ning üks ja sama tiimijuht võib juhtida korraga ühte tiimi.
- **Tooteomanik** (*Product Owner*) esindab arendusvajadusi, määrab nende tähtsuse järjekorra ning vastatab toote arendamise eest. Ta selgitab välja lahenduste detailsed nõuded ja määrab tiimi tööde tähtsuse järjekorra. Tooteomaniku ülesanne on tehtud arendusi huvirühmadele tutvustada. Igal tiimil on üks tooteomanik ning üks ja sama tooteomanik on korraga ühel tiimil.
- **Tiimi liige** (*Team Member*) on keskendunud reaalse lahenduse arendamisele huvipoolte jaoks. Tiimi liikmed võivad tegeleda testimise, analüüsi, arenduse, (arhitektuurilise) disaini, ja planeerimisega. Iga liige ei ole võimeline kõiki neid tegevusi katma, kuid DAD metoodika eesmärk on ajaga liikmete võimekusi kasvatada.
- **Arhitektuuri omanik** (*Architecture Owner*) vastutab tiimis arenduste ja komponentide arhitektuuriliste otsuste eest. Arhitektuuri omanikke on tiimis üks.
- **Huvipool** (*Stakeholder*) on isik, keda lahenduse tulemus oluliselt mõjutab. Selleks võib olla näiteks äriline tellija, lõppkasutaja või peakasutaja. Tiim teeb parima lahenduse saavutamiseks pidevalt huvirühmaga koostööd. Üks ja sama huvipool võib olla mitmes tiimis ning ühes tiimis võib olla mitu erinevat huvipoolt. [65]

DAD metoodika kirjeldab järgnevad teisesed rollid. Neid rolle võib tiimides ühel ajahetkel olla mitu.

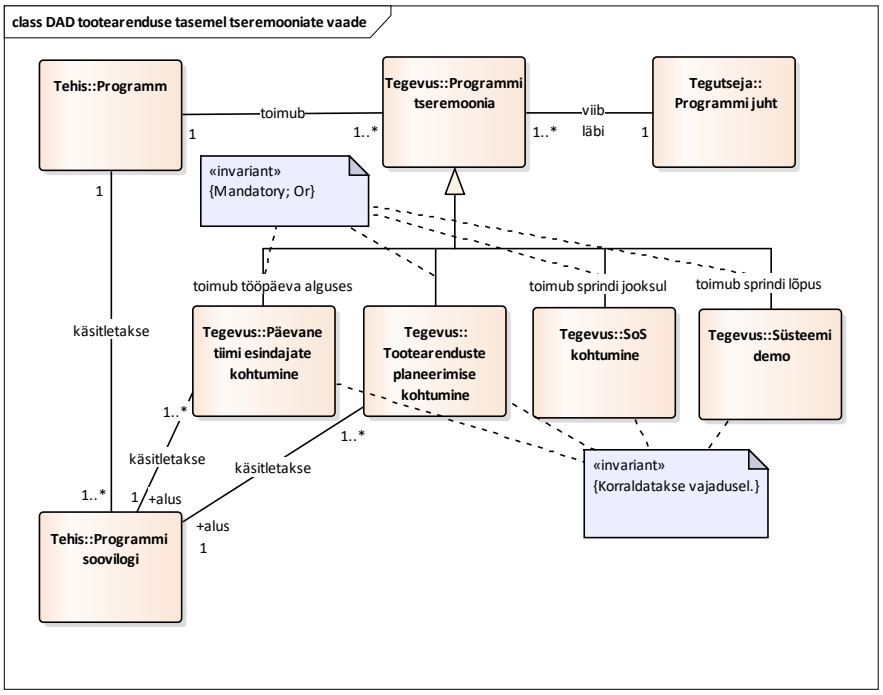
- **Asjatundja** (*Specialist*) võib olla näiteks ärianalüütik või muu koordineeriv roll.
- **Iseseisev testija** (*Independent Tester*) kaasatakse kogu lahenduse valideerimiseks. Kuigi suur osa testimisest tehakse tiimides sees, siis iseseisev testija suudab tihitpeale anda objektiivsemat tagasisidet, kuna tiimi liikmed ise oma lahendusi valideerides alati vigu ei märka.
- **Valdkonna ekspert** (*Domain Expert*) tunneb detailselt arendatavat valdkonda ning teda on vaja tiimi kaasata lahenduse nõuete kirjeldamiseks.
- **Tehniline ekspert** (*Technical Expert*) toetab tiime tehniliste lahenduste nõustamisega. Tehniline ekspert võib olla näiteks andmebaasi administraator.

- **Integreerija** (*Integrator*) on vaja tiimidesse kaasata tavaliselt juhul kui tehtud arendus puudutab palju süsteeme. Antud roll aitab neid süsteeme üheskoos tööle ja suhtlema panna. [65]



Joonis 20. DAD tiimi sprindi tseremooniate vaade.

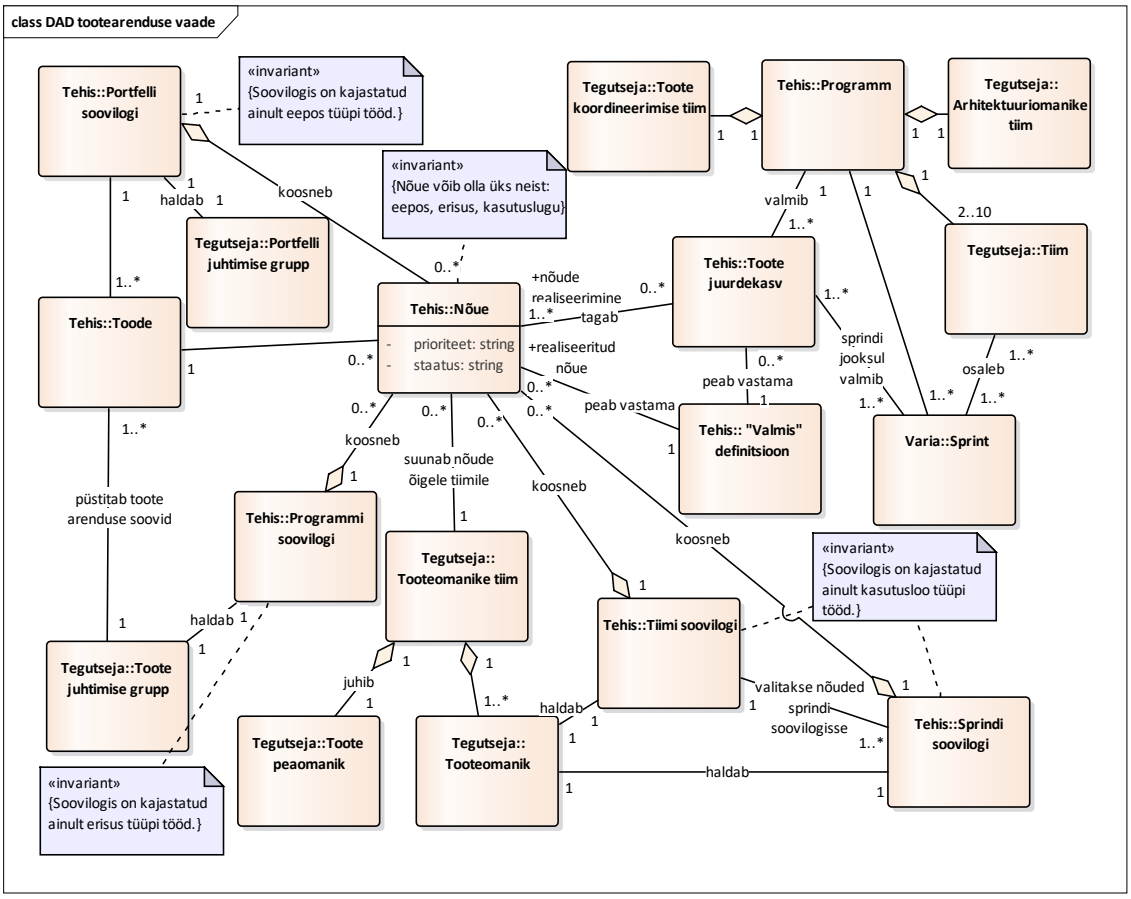
Joonis 20 esitab DAD metoodika tiimi sprindi tseremooniate vaate, mis on vastavuses DAD elutsükliga „Pideva tarne agiilne elutsükkel“. Tiimijuht viib läbi Scrum tseremooniad. Spirndil on kindel eesmärk ja kestus. Enamasti kestab sprint kaks nädalat, kuid võib kesta vahemikus üks kuni neli nädalalt. Sprindis toimub mitu Scrum tseremooniat, üks konkreetne tseremoonia toimub ühes sprindis. Sprindi käigus käsitletakse sprindi soovilogi (*Iteration Backlog*), millesse kuuluvad nõuded (*Work Item*) on valitud tiimi soovilogist (*Work item list*). Sprindis toimuvad klassikalised Scrum tseremooniad: päevane kohtumine, sprindi planeerimise kohtumine, soovilogi täpsustamise kohtumine, sprindile tagasivaate kohtumine ja sprindi ülevaatuse kohtumine. Nende sisu saab lugeda SAFe metoodika peatükist (vt Joonis 5 kirjeldused). [65]



Joonis 21. DAD programmi tasemel tseremooniate vaade.

Joonis 21 esitab DAD metoodika programmi tasemel tseremooniate vaate. Programmi juht (*Program Manager/Coordinator*) vastutab tootearenduse tseremooniate läbiviimise ja koordineerimise eest. Programmis toimub üks või mitu tseremooniat, mille jooksul käsitletakse programmi soovilogi. DAD metoodika märgib, et ületiimilisi tseremooniaid korraldatakse vastavalt vajadusele. Peamised DAD metoodika tseremooniad on järgmised.

- Päevane tiimi esindajate kohtumine (*Daily Coordination meeting*) toimub tiimide esindajatega (enamasti tiimijuht või tooteomanik) tööpäeva alguses, et vahetada jooksvat informatsiooni ja arutada tekkinud probleeme. Kohtumine toimub vastavalt vajadusele.
- Tootearenduste planeerimise kohtumine (*Big room planning*) on sündmus, mille käigus kõik tiimid planeerivad eesolevaid töid. Antud üritus on sarnane nagu SAFe metoodika PI planeerimine. Metoodika märgib, et seda tuleb korraldada vastavalt vajadusele. Antud kohtumisel on aluseks programmi soovilogi.
- SoS kohtumine (*Scrum of Scrums meeting*) on enamasti iganädalaselt toimuv kohtumine kõikide tiimijuhtide vahel, et vaadata üle eesmärkide saavutamise rütmis püsimine ja rääkida üle tiimidevahelised sõltuvused.
- Süsteemi esitlus ehk demo (*System Demo*) on üritus, mille raames esitletakse huvirühmadele ja äri osapooltele sprintides valminud funktsionaalsust. [65] [66]



Joonis 22. DAD tootearenduse vaade.

Joonis 22 esitab DAD tootearenduse vaate. Programmi juhtimine (*Program Management/Program Coordination*) haldab suure tootearenduse korral mitmete tiimide tööd ja aitab hallata tiimide omavahelisi sõltuvusi. Programmi juhtimise eesmärk on koordineerida tiimide tööd, et nad saaksid ühiselt koos toota tarbitavaid lahendusi ja toote juurdekasvu oma huvirühmadele. Arhitektuuriomanike tiim (*Architecture Owner Team*) vastutab programmi üldise arhitektuurilise suuna eest ja arhitektuuriliste tehniliste sõltuvuste üle läbirääkimiste pidamiseks. Toote koordineerimise tiim (*Product Coordination Team*) vastutab programmis olevate tiimideüleste koordineerimisprobleemide lahendamise eest. Programmi (*Program*) moodustavad ühise toote kallal töötavad tiimid, kes töötavad paralleelselt ühises sprindis ja tarnivad toote juurdekasvu. Ühte programmi võib kuuluda optimaaselt kaks kuni kümme tiimi. Sprindi jooksul valmib toote juurdekasv, mille tagab ühe või enama nõude realiseerimine ning mis peab vastama „valmis“ definitsioonile. Nõudel on staatus ja prioriteet. Tiimi soovilogi ja sprindi soovilogi koosnevad nõuetest, mis võivad olla kasutusloo tüüpi. Programmi soovilogi koosneb nõuetest, mis on erisuse tüüpi ja programmi soovilogi haldab toote juhtimise grupp. Toote juhtimise grupp (*Product Management*) tegeleb toote

haldamisega, kujundades toote strateegiat ja ärivisiooni ning püstitab toote või toodete arenduse soovid. Portfelli soovilogi koosneb eepose tüüpi nõuetest. Portfelli soovilogi on nagu äriline teekaart (*roadmap*), mis kajastab prioriteete ja mida haldab portfelli juhtimise grupp. Portfelli juhtimise grupp (*Portfolio Management*) vastutab toodete identifitseerimise ja prioritiseerimise eest, luues seeläbi äriväärtust. Portfelli juhtimise grupp suhtleb pidevalt toote juhtimise grupiga, et anda vastastikku tagasisidet ja sisendit toodete kohta. Tooteomanike tiim (*Product Owner Team*) vastutab arendustööde prioritiseerimise eest ja konkreetsete tööülesannete (nõuete) suunamise eest õigele tiimile. Tooteomanike tiimi juhib toote peaomanik (*Chief Product Owner*). [65]

5.5 Metamudelite valideerimise läbiviimine

Mudelite valideerimiseks viidi läbi nende ülevaatus autori kahe kolleegiga – ettevõtte *Agile coach* ja arendusjuht. Antud isikud valiti valideerimise protsessi, kuna nad tunnevad töös käsitletavaid raamistikke ning on nende juurutamisega varasemalt kokku puutunud. Loodud metamudelite üheks eesmärgiks on, et mudelid kirjeldaksid metoodikat ja nende vaatleja saaks aru, kuidas metoodika on ülesehitatud. Seetõttu viis autor lisaks läbi mudelite ülevaatuse ka kahe IT valdkonnas töötava isikuga, kes antud metoodikaid ei tunne.

Ülevaatuse alguses tegi autor lühikese sissejuhatuse metoodikast, mudelite vaadete põhimõttest ja pakettide struktuurist. Valideerimisel olid tähelepanu all järgmistele küsimustele vastuste leidmine.

- Kas kõik olulised metoodikat iseloomustavad elemendid on olemas?
- Kas terminite kasutus ja nende tõlked on arusaadavad?
- Kas tähtsamad seosed on kirjeldatud?
- Kas lisatud kitsendused on mõistetavad ja nende esitus on piisav?
- Kas võimsustike määratlus lähtub sellest, mida metoodika kirjeldab (nö ideaaljuht ehk „*happy path*“ stsenaarium)
- Kas antud mudelid annavad metoodikast tervikliku pildi või on midagi antud metoodika puhul olulist täiesti välja jäänud?
- Kuidas metoodikast aru saadakse? (Autor palus testitaval metoodikat kirjeldada)
- Kas midagi olulist on mudelis kajastamata?

Lisaks esitas autor küsimusi elementide ja nendevaheliste seoste kohta, mille õigsuses ta kahtles. Ülalmainitud küsimused aitavad tagada erinevate osapoolte sarnase mudelite tõlgendamise.

Mudelite ülevaatajad tõdesid, et ei ole varemalt sellise lähenemisviisiga kokku puutunud, kus paindmetoodikate põhimõisted ja nende seosed esitatakse klassidiagrammidena. Nad leidsid, et see on huvitav viis metoodika sisu edasiandmiseks. Lisaks märgiti, et mudelitest arusaamine osutus keerukaks juhul kui seal esines tundmatu nimetusega elemente. Kuna mudelid on esitatud eestikeeles, siis mitmed mõisted ei ole IT valdkonnas kasutusel, vaid selle asemel on kasutusel inglise keelne termin. Üks näide selle kohta on „rüsinameister“, mis ei olnud ülevaatajate jaoks tuntud mõiste, kuigi see on ka standardipõhises tarkvaratehnika sõnastikus [6]. Inglise keelne termin sellele on „*Scrum Master*“, mis IT valdkonnas on aga teada-tuntud termin. Kuna metamudelitele on lisatud mõistete selgitused ja tekstilised kirjeldused, siis paralleelselt mudeleid uurides ja teksti lugedes ei tohiks selliseid arusaamatusi tekkida.

Valideerimise tulemusel tegi autor mudelites kümmekond muudatust. Peamiselt seisnesid muudatused seosetüüpide kirjelduste täiendamises või võimsustike muutmises. Näitena võib välja tuua muudatuse, mis tehti tiimi vaadetes (Joonis 4, Joonis 8, Joonis 12, Joonis 19), kus arendusmetoodika seosetüüp oli eelnevalt tootega seotud, kuid mudelite ülevaatajate jaoks oli see arusaamatu ja nad leidsid, et korrektsem oleks antud seos näidata arendusmetoodika ja tiimi vahel, sest tiim praktiseerib arendusmetoodikat, mitte toode.

5.6 Metamudelitel põhinev võrdlus

Metoodikate võrdlemisel kasutatakse ontoloogilise analüüsi meetodit [67]. Metoodikate võrdlemisel lähtutakse loodud metamudelitest, mille elemente omavahel võrreldakse. Võrdlemisel võetakse aluseks *Scaled Agile Framework* metamudel (kui "ontoloogia") ning seda metoodikat võrreldakse *Large-Scale Scrum Huge* ja *Disciplined Agile Delivery* metamudelitega. Võrdlus tehakse nende metoodikate põhjal, kuna autor on jõudnud järeldusele, et antud metoodikad on sobilikud valikud otsustusmudeli alternatiivideks. SAFe metamudelis on 45 klassi, LeSS Huge metamudelis 37 klassi ning DAD metamudelis 48 klassi. Võrdluses uuritakse järgmise nelja ontoloogilise lahknevuse esinemist. Need võivad kahjustada modelleerimise ontoloogilist selgust.

- Mõiste ülelaadimine (*construct overload*) – vaadeldud metoodikas olev mõiste vastab mitmele SAFe metoodika mõistele.
- Mõiste dubleerimine (*construct redundancy*) – vaadeldud metoodikas on rohkem kui üks mõiste, mis vastab ühele SAFe metoodika mõistele.
- Mõistete üleküllus (*construct exess*) – vaadeldud metoodika mõistele ei vasta ükski SAFe metoodika mõiste.
- Mõiste puudumine (*construct deficit*) – SAFe metoodika mõistele ei vasta ükski vaadeldud metoodika mõiste. [67]

Tabel 2. Metamudelite võrdlus.

ID	**SAFe**	**LeSS *Huge***	**DAD**
1	Toode	Toode	Toode
2	„Valmis“ definitsioon	„Valmis“ definitsioon	„Valmis“ definitsioon
3	Agiilne tiim	Erisuse tiim	Tiim
4	Arendustiim	Erisuse tiim	Tiim
5	Tooteomanik	Tooteomanik	Tooteomanik
6	Rüsinameister	Rüsinameister	Tiimi juht
7	Süsteemi arhitekt	Spetsialist	Arhitektuuriomanik
8	Liige	Tiimi liige	Tiimi liige
9	Arendaja	Arendaja	Arendaja
10	Testija	Testija	Testija
11	Spetsialist	Spetsialist	Spetsialist
12	Arendusmetoodika	Arendusmetoodika	Arendusmetoodika
13	Hübriidmetoodika	-	Hübriidmetoodika
14	Scrum	Scrum	Scrum
15	ScrumXP	-	Hübriidmetoodika?
16	Kanban	-	Kanban
17	Äriomanik	-	Huvipool
18	Scrum tseremoonia	Scrum tseremoonia	Scrum tseremoonia
19	Sprint	Sprint	Sprint
20	Innovatsiooni sprint	-	-
21	Päevane kohtumine	Päevane kohtumine	Päevane kohtumine
22	Sprindi planeerimise kohtumine	Sprindi planeerimise kohtumine 2	Sprindi planeerimise kohtumine

ID	SAFe	LeSS *Huge*	DAD
23	Soovilogi täpsustamise kohtumine	Tiimi toote soovilogi täpsustamise kohtumine	Soovilogi täpsustamise kohtumine
24	Sprindi tagasivaate kohtumine	Sprindi tagasivaate kohtumine	Sprindi tagasivaate kohtumine
25	Sprindi ülevaatuse kohtumine	-	Sprindi ülevaatuse kohtumine
26	Sprindi soovilogi	Tiimi sprindi soovilogi	Sprindi soovilogi
27	Tiimi soovilogi	-	Tiimi soovilogi
28	Programmi soovilogi	Toote soovilogi	Programmi soovilogi
29	Agiilne väljalaske rong (ART)	-	Programm
30	Väljalaske rongijuht	-	Programmi juht
31	PI sündmus	LeSS tseremoonia	Programmi tseremoonia
32	Programmi juurdekasvu ajaraam (PI)	-	-
33	PI planeerimine	Sprindi planeerimise kohtumine 1	Tootearenduste planeerimise kohtumine
34	Kontrolli ja kohanda kohtumine (I&A)	-	-
35	Süsteemi demo	Sprindi ülevaatuse kohtumine	Süsteemi demo
36	Ettevalmistus PI planeerimiseks	-	-
37	Tooteomanike kohtumine	-	Tooteomanike kohtumine
38	SoS kohtumine	-	SoS kohtumine
39	Programmi Kanban	-	-
40	Nõue	Nõue	Nõue
41	Toote juurdekasv	Toote juurdekasv	Toote juurdekasv
42	Tootejuhtimise grupp	Tooteomanike tiim	Toote juhtimise grupp
43	Portfelli soovilogi	Toote soovilogi	Portfelli soovilogi
44	Portfelli juhtimise grupp	-	Portfelli juhtimise grupp
45	Väärtusahel	Nõudevaldkond	-

Järgnevalt tuuakse välja võrdluse tulemusel (Tabel 2) tehtud tähelepanekuid.

SAFe metoodika võrdlemisel **LeSS *Huge*** metoodikaga tuuakse välja nelja liiki ontoloogilised lahknevused.

- Järgnev LeSS *Huge* metoodikas olev mõiste vastab mitmele SAFe metoodika mõistele (mõiste ülelaadimine): Erisuse tiim, Spetsialist, Toote soovilogi.
- Mõiste dubleerimine LeSS *Huge* metoodikas puudub.
- LeSS *Huge* metoodika järgmistele mõistetele ei vasta ükski SAFe metoodika mõiste (mõistete üleküllus): Mitme tiimi toote soovilogi täpsustamise kohtumine, Üldine toote soovilogi täpsustamise kohtumine, Sprindi ühine tagasivaate kohtumine, Algne toote soovilogi täpsustamise kohtumine, Defineerimata üksus, Tugirühm, Tootegrupijuht, Pädevus- ja juhendamisrühm, Valdkonna tooteomanik, Valdkonna toote nõuete nimekiri, Nõuete valdkonna atribuut.
- Järgmistele SAFe metoodika mõistetele ei vasta ükski LeSS *Huge* metoodika mõiste (mõiste puudumine): Hübriidmetoodika, ScrumXP, Kanban, Äriomanik, Innovatsiooni sprint, Sprindi ülevaatuse kohtumine, Tiimi soovilogi, Agiilne väljalaske rong, Väljalaske rongijuht, Programmi juurdekasvu ajaraam (PI), Kontrolli ja kohanda kohtumine (I&A), Ettevalmistus PI planeerimiseks, SoS kohtumine, Tooteomanike kohtumine, Programmi Kanban, Portfelli juhtimise grupp.

SAFe metoodika võrdlemisel **DAD** metoodikaga tuuakse välja nelja liiki ontoloogilised lahknevused.

- Järgnev DAD metoodikas olev mõiste vastab mitmele SAFe metoodika mõistele (mõiste ülelaadimine): Tiim, Hübriidmetoodika.
- Mõiste dubleerimine DAD metoodikas puudub.
- DAD metoodika järgmistele mõistetele ei vasta ükski SAFe metoodika mõiste (mõistete üleküllus): Tehniline ekspert, Domeeni ekspert, Integreerija, Asjatundja, Iseseisev testija, XP, Agiilne modelleerimine, Päevane kohtumine, Toote peaomanik, Tooteomanike tiim, Toote koordineerimise tiim, Arhitektuuriomanike tiim.
- Järgmistele SAFe metoodika mõistetele ei vasta ükski DAD metoodika mõiste (mõiste puudumine): Innovatsiooni sprint, Programmi juurdekasvu ajaraam (PI), Kontrolli ja kohanda kohtumine (I&A), Ettevalmistus PI planeerimiseks, Programmi Kanban, Väärtusahel.

Analüüsides Tabel 2 ja mõistete jaotust ontoloogilisteks lahknevusteks järeldab autor, et SAFe ja DAD paindmetoodikad on omavahel sarnasemad kui SAFe ja LeSS *Huge*, sest SAFe ja DAD metoodikal on rohkem mõistete kattuvusi kui SAFe ja LeSS *Huge* vahel.

Teiseks täheldas autor, et DAD ja LeSS *Huge* metoodikate metamudelis oli sarnase sisuga mõiste, SAFe metamudelis seda aga ei olnud, kuigi antud mõiste metoodikas eksisteerib. Täpsemalt, DAD metoodikas on tootearenduse vaates mõiste „Toote koordineerimise tiim“. Sarnane mõiste on ka LeSS Huge metoodikas – „Pädevus- ja juhendamisrühm“. SAFe’s sellise sisuga mõistet metamudelis ei eksisteeri, küll aga on selline roll SAFe metoodikas samuti olemas – LACE (*The Lean-Agile Center of Excellence*) [58]. Modelleerimisel ei pidanud autor vajalikuks sarnast mõistet SAFe metamudelis kujutada, sest selle tähtsus SAFe metoodika kontekstis ei olnud niivõrd oluline.

Kolmas võrdlemisel tehtud tähelepanek oli, et LeSS *Huge* metoodika metamudelis puudub SAFe metoodika mõistega „Tooteomanike kohtumine“ sarnane mõiste. Küll aga mainiti sarnast mõistet „Tooteomanike tiimi kohtumine“ tekstis eraldi (vt jaotis 5.3.2) Antud mõiste puudub metamudelis, sest LeSS *Huge* metoodika kohta ei tehtud eraldi tootearenduse tasemel tseremooniate vaadet, sest sellele kehtis LeSS metoodika vaade (Joonis 14) koos tooteomanike tiimi kohtumisega nii nagu jaotises kirjeldatud.

6 Paindmetoodika valik konkreetsele ettevõttele

Eelmises peatükis toodi välja viie skaleeritava paindmetoodika metamudelid ja nende kirjeldused ning võrreldi loodud mudeleid. Käesolevas peatükis koostatakse otsustusmudel nende hulgast sobiva paindmetoodika valikuks. Kõigepealt kirjeldatakse lühidalt Saaty metoodikat, seejärel tehakse ülevaade põhi- ja alamkriteeriumitest, valitakse sobivad alternatiivid ning tehakse tundlikkuse analüüs.

6.1 Saaty meetodi lühikirjeldus

Saaty analüütiliste hierarhiate meetodi (AHP) töötas välja 1970ndatel USA matemaatik Thomas L. Saaty. Saaty meetod püüab otsustussegaduses ja suures subjektivismis korda luua nii, et pärast otsustuse tegemist saaks olla veendunud nii tehtud otsustuse õigsuses kui ka osata veenda teisi, miks otsustati just nii ja mitte teisti. Saaty meetod põhineb (otsustajate poolt) objektide paarikaupa võrdlemisel. [2] [36]

Meetodi kasutamise võib jagada järgmisteks etappideks.

1. Probleemi defineerimine.
2. Eesmärgi defineerimine.
3. Modelleerimine ja süntees (hierarhia koostamine).
    - Kriteeriumite ehk mõjurite leidmine.
    - Alternatiivide ehk valikute leidmine.
4. Alternatiivide paarikaupa võrdlemine iga kriteeriumi suhtes.
5. Kriteeriumite paarikaupa võrdlemine.
6. Alternatiivide osakaalude leidmine.
7. Tulemuste analüüs, sealhulgas ka tulemuste tundlikkuse analüüs, mille eesmärk tekitada mudeli struktuuris põhjendatud muudatusi ning vaadelda uut tekkinud tulemust.
8. Vajaduse korral eelnevate sammude täpsustamine ehk järgmine iteratsioon.

Kui kriteeriumid ja alternatiivid on leitud, siis korrastatakse need mitmetasemelisse hierarhilisse struktuuri. Kõigepealt tuleb eesmärk, siis kriteeriumid, teatud kriteeriumitel võivad olla alamkriteeriumid ja viimasel tasemel on alternatiivid. [2] [36]

Tabel 3. Saaty fundamentaalskaala otsustuste jaoks [36].

Intensiivsus	**Definitsioon**	**Selgitus**
1	Võrdselt tähtis	Kaks kriteeriumit pole mõjus eristatavad.
3	Mõõdukalt eelistatud	Üks kriteerium on teisest eristatavam.
5	Oluliselt eelistatud	Ühel kriteeriumil on teisega võrreldes tugev eelistus.
7	Väga tugevalt eelistatult	Ühel kriteeriumil on teisega võrreldes väga tugev eelistus.
9	Ekstreemselt eelistatud	Tugevaim võimalik paremus või eelistus.

Tabel 3 on esitatud, millise skaala alusel kriteeriume, alamkriteeriume ja alternatiive teineteisega võrreldakse.

Subjektiivsete hinnangute järjepidevuse jälgimiseks on Web-Hipre rakenduses välja toodud arv kooskõla määr CM (*consistency measure*). T. Veskioja väidab, et CM valem erineb Saaty suhtelise kooskõlaindeksist, kuid kasutamine on sama. Seega kui CM väärtus on alla 0.1, siis on tegemist kooskõlaliste hinnangutega. [7]

6.2 Otsustusmudeli koostamine

Ühed olulisemad kriteeriumid skaleeritava paindmetoodika valikul on kindlasti organisatsiooni tüüp ja tootearenduse ulatuslikkus. Ettevõte, mille näitel otsustusmudel luuakse, on suurettevõte. Selle tootearenduses osaleb kaheksateist tiimi (Vt peatükk 3). Alternatiivide valikul lähtutakse nendest kriteeriumitest (vt jaotis 6.2.5) ja seetõttu neid kriteeriumeid enam otsustusmudelisse ei kaasata. Loodav otsustusmudel arvestab ettevõtte kontekstiga.

6.2.1 Põhi- ja alamkriteeriumid ning nende kirjeldused

Antud jaotises esitatakse otsustusmudeli põhi- ja alamkriteeriumid. Autor on välja valinud neli põhikriteeriumit, mis on ettevõtte konteksti arvestades skaleeritava paindmetoodika valikul olulised. Kriteeriumite valikuni jõuti kirjanduse uurimise ja metamudelite loomise tulemusena. Metamudelite võrdlemisel tuvastati peamised

erinevused ja sarnasused, mis aitas metoodikate erisusi mõista ja selle põhjal kriteeriumeid välja mõelda. Näiteks tõdes autor modelleerimise käigus, et erinevad metoodikad kirjeldavad hulga erinevaid rolle. Seega leidis autor, et oluline on sobitada olemasolevad ettevõtte rollid metoodika rollidega. Sellest tulenes ka rollide sobivuse kriteerium (vt jaotis 6.2.1). Autor kaalus lisaks sellist kriteeriumit nagu metoodika populaarsus, kuid otsustas selle siiski välistada. Selle kriteeriumiga oleks otsitud, millist metoodikat kasutatakse kõige rohkem. Autor välistas selle kriteeriumi, sest ei saa lähtuda metoodika populaarsusest, vaid tuleb leida sobiv metoodika vastavalt ettevõttele. Kõige populaarsem ei tähenda alati, et see oleks ettevõttele kõige sobivam.

- **Eksistents**, mis määratleb, millised ettevõtte jaoks olulised aspektid on metoodikas olemas ja millised mitte.
- **Rakendatavus**, mis määratleb, kuidas metoodika võimaldab tiime moodustada ja milliseid arendusmetoodikaid kasutada.
- **Rollide sobivus**, mille raames hinnatakse, kuidas sobivad metoodika pakutavad rollid ettevõttes kasutusel olevate rollidega.
- **Juurutamine**, mis määratleb aspekte, mida on oluline arvestada metoodika kasutuselevõtmisel.

Tabel 4 esitab põhikriteeriumitele vastavad alamkriteeriumid ja nende kirjeldused koos põhjendustega, miks on oluline antud alamkriteeriumeid otsustusmudelisse kaasata.

Tabel 4. Põhi- ja alamkriteeriumid ning nende kirjeldused.

Põhikriteerium	**Alamkriteerium**	**Kirjeldus**
Eksistents	Portfelli haldus	Portfelli juhtimine on oluline, et identifitseerida ja prioritiseerida erinevate toodetega seotud arendustöid [58]. Antud ettevõtte puhul on see tähtis, sest siiani ei ole projektipõhine arendus seda lahendanud. Mida detailsemalt metoodika seda kirjeldab, seda parem.
	Kliendikesksus	Kliendikesksus (*Customer Centricity*) on mõtte- ja äritegevuse viis, mis keskendub kõikide ettevõtte pakutavate teenuste ja toodete kaudu kliendi jaoks positiivsete kogemuste loomisele [4]. Ettevõtte jaoks on tootearendusel oluline keskenduda kliendi vajadustele ning leida võimalus kontrollida uute funktsionaalsuste vastavust kliendi soovidele.

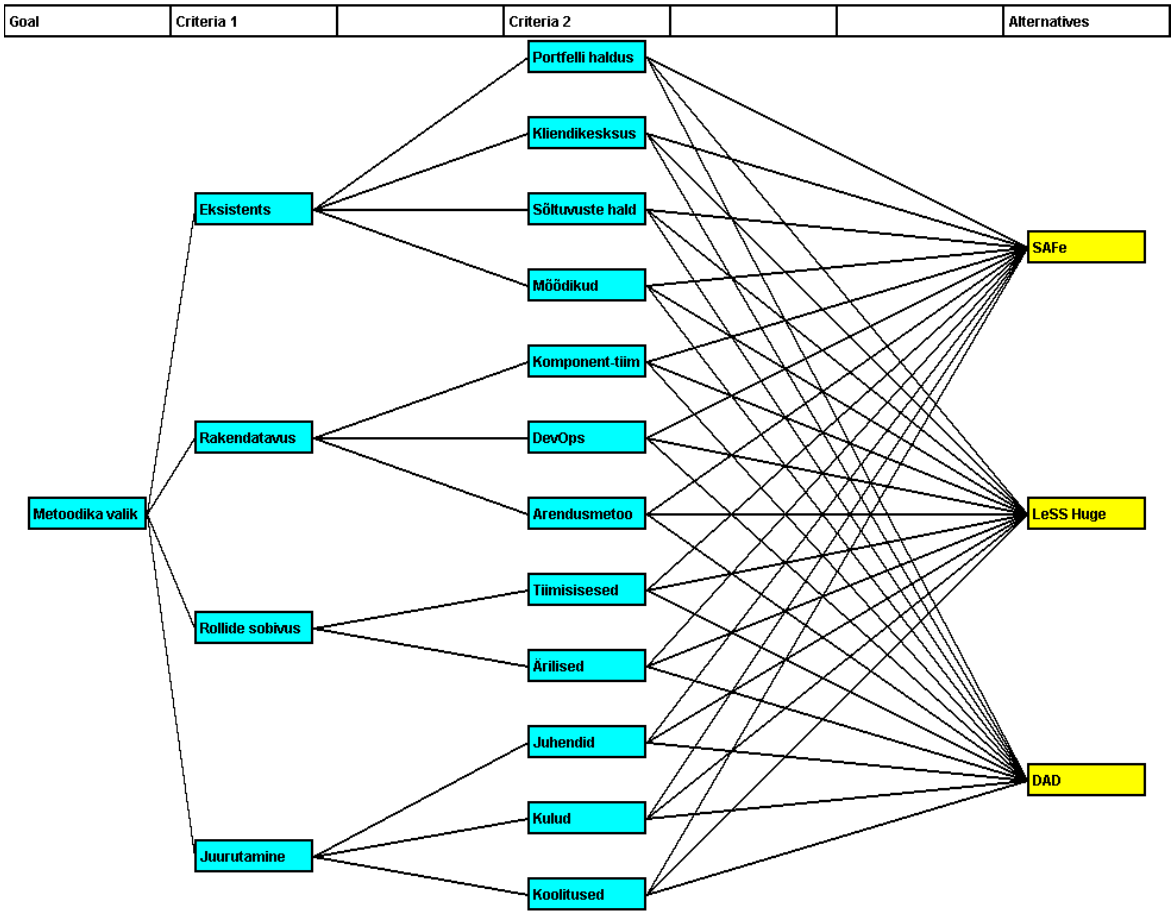
Põhikriteerium	Alamkriteerium	Kirjeldus
	Sõltuvuste haldus	Suure arvu tiimide korral sõltuvad tiimid omavahel erinevate arendustööde kaudu. Valitav metoodika peab pakkuma välja (erinevate tseremooniate näol) viisid, kuidas tiimidevahelisi sõltuvusi hallata.
	Mõõdikud	Ettevõtte huvi on mõõta rakendatava metoodika edukust. Seetõttu peab metoodika välja pakkuma mõõdikud ehk KPId (*Key Performance Indicator*), kuidas metoodika rakendamise efektiivsust erinevatel tasemetel mõõta.
Rakendatavus	Komponent-tiim	Praegused tiimid on moodustatud arendades arhitektuurilisi komponente. Tiimidel ei ole võimekust arendada täielikku klienditeekonda (korraga kõike, mida klient soovib). Ettevõtte soov on, et tiime ei peaks uuesti koostama ja kompetentsi jagunemine tiimides võiks suurel määral jääda samaks. Seega metoodika võiks toetada tiimide struktureerimist komponentide järgi.
	DevOps	DevOps on tarkvaraarenduse kultuur, mis ühendab omavahel tarvaraarenduse ja -operatsioonid [8]. Devops kultuur on ettevõttes juba varem kasutusele võetud ja see on olnud edukas. Ettevõttel ei ole soov seda muuta, seega metoodika peaks seda toetama.
	Arendusmetoodikad	Valitav metoodika peab toetama/lubama erinevaid tiimides rakendatavaid arendusmetoodikaid (Scrum, Kanban, XP jne..). Tiimide jaoks on see oluline paindlikkuse seisukohast, kuna siiani ei ole kõik tiimid ühise arendusmetoodika järgi töötanud.
Rollide sobivus	Tiimisisesed	Olemasolevad tiimide rollid peavad sobituma metoodika pakutavate rollidega.
	Ärilised	Olemasolevad ärirollid peavad sobituma metoodika pakutavate rollidega.
Juurutamine	Juhendid	Juhendmaterjalide piisavus, kättesaadavus ja põhjalikkus, et nende põhjal metoodikat kasutusele võtta.
	Kulud	Metoodika kasutusele võtmisega kaasnevad võimalikud lisakulud. Juurutamise algfaasis on ettevõttel soov tellida vastava metoodika ekspert, kes teeks vajalikele osapooltele koolituse. Eesmärk on hoida kulud madalana. Antud alamkriteeriumi

Põhikriteerium	Alamkriteerium	Kirjeldus
		alusel alternatiive võrreldes võrreldakse koolituste hindu.
	Koolitused	Kuna ettevõttes ei ole skaleeritavaid metoodikaid hästitundvaid praktikuid, siis on oluline erinevate koolituste olemasolu, mida vajadusel saab tellida. Antud alamkriteeriumitega mõõdetakse, kas ja kui palju erinevad metoodikad seda pakuvad.

6.2.2 Otsustusmudeli struktuur

Autor koostas Web-Hipre tarkvaras otsustusmudeli, kus põhi- ja alamkriteeriumid koos alternatiividega viidi hierarhiasse.

Joonis 23 kajastab otsustusmudeli struktuuri ettevõttele sobiva paindmetoodika valikuks.



Joonis 23. Otsustusmudel paindmetoodika valikuks.

6.2.3 Põhikriteeriumite võrdlus

Põhikriteeriumite kaalud kooskõlastati ettevõttes töötava eksperdiga. Põhikriteeriumite võrdlust kirjeldab Joonis 24.

Eksistents *vs.* Rakendatavus – mõlemad kriteeriumid on võrdtähtsad. Saaty skaala hinnang 1.

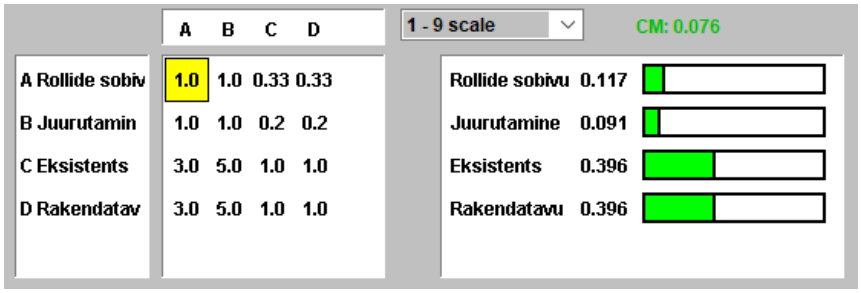
Eksistents *vs.* Rollide sobivus – Eksistents on mõõdukalt eelistatud. Saaty skaala hinnang 3.

Eksistents *vs.* Juurutamine – Eksistents on oluliselt eelistatud. Saaty skaala hinnang 5.

Rakendatavus *vs.* Rollide sobivus – Rakendatavus on mõõdukalt eelistatud. Saaty skaala hinnang 3.

Rakendatavus *vs.* Juurutamine – Rakendatavus on oluliselt eelistatud. Saaty skaala hinnang 5.

Rollide sobivus *vs.* Juurutamine – mõlemad kriteeriumid on võrdtähtsad. Saaty skaala hinnang 1.



Joonis 24. Põhikriteeriumite võrdlus.

Joonis 24 esitab põhikriteeriumite võrdluse, millest võib järeldada, et ettevõtte jaoks on eksistentsi ja rakendatavuse põhikriteeriumid võrdselt kõige olulisemad. Seejärel on hinnatud rollide sobivust kolmandaks. Juurutamise kriteerium on suhteliselt kõige vähem oluline.

6.2.4 Alamkriteeriumite võrdlus

Alamkriteeriumite kaalud kooskõlastati ettevõttes töötava eksperdiga.

Eksistents alamkriteeriumitele on antud järgmised hinnangud.

Portfelli haldus *vs.* Mõõdikud – Portfelli haldus on väga tugevalt eelistatud. Saaty skaala hinnang 7.

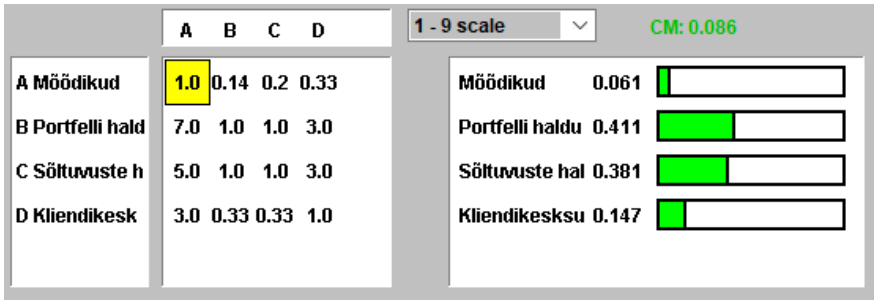
Portfelli haldus *vs.* Sõltuvuste haldus – kriteeriumid on võrdtähtsad. Saaty skaala hinnang 1.

Portfelli haldus *vs.* Kliendikesksus – Portfelli haldus on mõõdukalt eelistatud. Saaty skaala hinnang 3.

Mõõdikud *vs.* Sõltuvuste haldus – Sõltuvuste haldus on oluliselt eelistatud. Saaty skaala hinnang 5.

Mõõdikud *vs.* Kliendikesksus – Kliendikesksus on mõõdukalt eelistatud. Saaty skaala hinnang 3.

Sõltuvuste haldus *vs.* Kliendikesksus – Sõltuvuste haldus on mõõdukalt eelistatud. Saaty skaala hinnang 3.



Joonis 25. Eksistentsi alamkriteeriumite võrdlus.

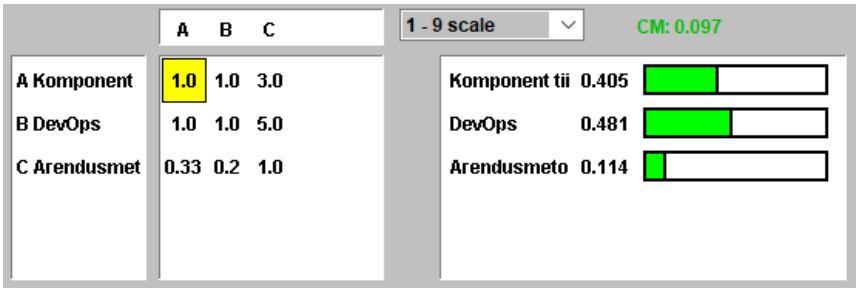
Joonis 25 esitab eksistentsi alamkriteeriumite võrdluse. Jooniselt võib järeldada, et portfelli halduse alamkriteeriumit peetakse kõige olulisemaks. Peaaegu sama oluline on sõltuvuste haldus. Kliendikeskuse kriteeriumit on hinnatud kolmandaks ja kõige vähem tähtis on mõõdikute olemasolu.

Rakendatavus alamkriteeriumitele on antud järgmised hinnangud.

Komponent-tiim *vs.* DevOps – kriteeriumid on võrdtähtsad. Saaty skaala hinnang 1.

Komponent-tiim *vs.* Arendusmetoodikad – Komponent-tiim on mõõdukalt eelistatud. Saaty skaala hinnang 3.

DevOps *vs.* Arendusmetoodikad – DevOps on oluliselt eelistatud. Saaty skaala hinnang 5.



Joonis 26. Rakendatavus alamkriteeriumite võrdlus.

Joonis 26 esitab rakendatavuse alamriteeriumite võrdluse, millest järeldub, et DevOps kriteerium on kõige olulisem. Peaaegu sama tähtis on komponent-tiimi kriteerium ja kõige vähem tähtsamaks on ostunud erinevate arendusmetoodikate olemasolu.

Rollide sobivus alamkriteeriumitele on antud järgmised hinnangud.

Tiimisisesed *vs.* Ärilised – Tiimisisesed rollid on mõõdukalt eelistatud. Saaty skaala hinnang 3.

Joonis 27. Rollide sobivuse alamkriteeriumite võrdlus.

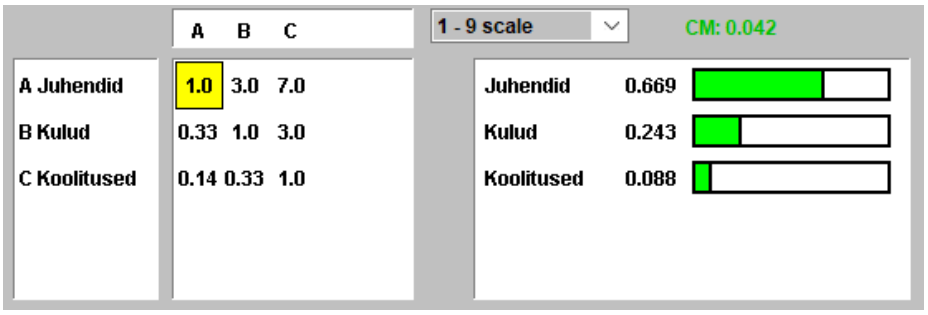
Joonis 27 esitab rollide sobivuse alamkriteeriumite võrdluse, millest järeldub, et tiimisiseste rollide sobivuse kaal on olulisem kui äriliste rollide sobivus.

Juurutamise alamkriteeriumitele on antud järgmised hinnangud.

Juhendid *vs.* Kulud – Juhendid on mõõdukalt eelistatud. Saaty skaala hinnang 3.

Juhendid *vs.* Koolitused – Juhendid on väga tugevalt eelistatud. Saaty skaala hinnang 7.

Kulud *vs.* Koolitused – Kulud on mõõdukalt eelistatud. Saaty skaala hinnang 3.



Joonis 28. Juurutamise alamkriteeriumite võrdlus.

Joonis 28 esitab juurutamise alamkriteeriumite võrdluse, millest järeldub, et ettevõtte jaoks on juhendite alamkriteerium kõige suurema kaaluga. Kulude kriteerium on tähtsuselt järgmine ja kõige väiksema kaaluga on koolituste alamkriteerium. Kuna koolituste tellimine on planeeritud ühekordselt, siis selle kulusid väga kõrgeks ei hinnata.

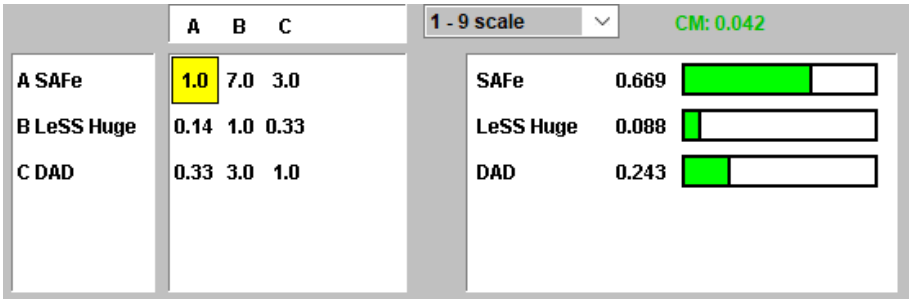
6.2.5 Alternatiivide valik

Lähtudes loodud metamudelitest ja nende kirjeldustest, on autor teinud valiku otsustusmudeli alternatiivide suhtes. Otsustusmudelisse sobivad alternatiivid *Scaled Agile Framework*, *Large-Scale Scrum Huge* ja *Disciplined Agile Delivery*, sest need metoodikad on sobivad ettevõtte tiimide arvule, neis on portfelli juhtimise tase ja need on sobivad eelkõige suuremahulise tootearenduse puhul. Seega otsustusmudelisse ei võeta alternatiivideks *Nexust* ja tavalist *LeSS* metoodikat. Autor välistas *Nexus* metoodika, sest modelleerimise käigus selgus, et antud metoodika on sobilikum väiksema tootearenduse mahuga ettevõttele, kus on kolm kuni üheksa tiimi (Joonis 8), ning seda ei ole võimalik nii suuremahuliselt skaleerida kui ettevõtte X puhul see vajalik oleks. Samuti ei eksisteeri *Nexus* metoodikal sellisel tasemel portfelli juhtimist. *LeSS* ei sobi alternatiiviks samal põhjusel nagu *Nexus*. See on mõeldud väiksema tootearenduse jaoks ja tiimide arv jääb kahe kuni kaheksa vahele (Joonis 12). Küll aga sobib valikusse *LeSS Huge*, mis on *LeSS* raamistiku edasiarendus suuremate organisatsioonide ning nende arenduste jaoks.

Skaleerimise maatriksi (Joonis 1) järgi paigutuvad *Nexus* ja *LeSS* metoodikad skaleeritud arendusmetoodika lahtrisse, kuna tiimide arv on üle kahe ja kõik meeskonnad arendavad ühte toodet. Antud ettevõtte puhul sobivad metoodikad, mis paigutuvad maatriksil sinna, kus metoodika kirjeldab ka portfelli (ja programmi) juhtimist ning SAFe, DAD ja LeSS *Huge* seda ka võimaldavad.

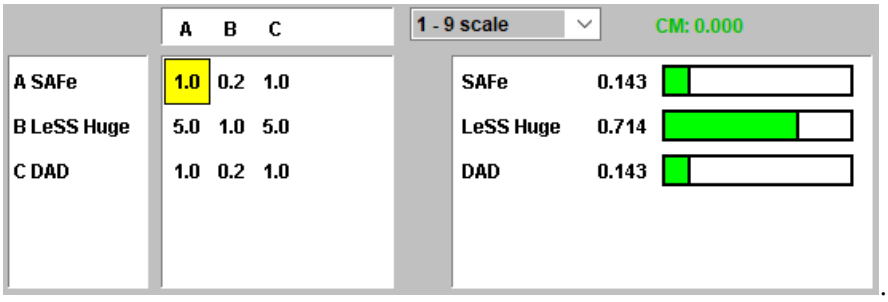
6.2.6 Alternatiivide võrdlus

Käesolevas jaotises võrreldakse omavahel kolme väljavalitud alternatiivi iga alamkriteeriumi suhtes. Selle käigus esitatakse kaalude hinnanguid kajastav joonis ja antud kaalude põhjendused. Kaalude määratlemine on lõputöö autori enda hinnang. Siinkohal täiendavaid osapooli kaalude kooskõlastamiseks ei kaasatud.



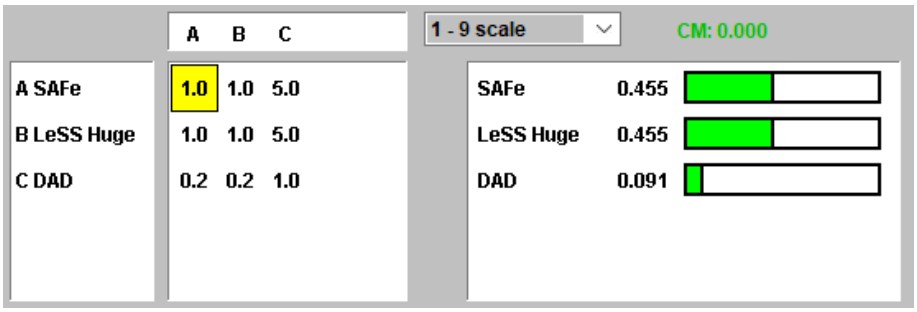
Joonis 29. „Portfelli haldus“ Saaty skaala hinnangud

Joonis 29 esitab metoodikate portfelli haldusele antud hinnangud. SAFe annab väga head juhised, kuidas portfelli tasemel toodete juhtimine ja prioritseerimine käima peab ning millised rollid selleks vajalikud on. DAD kirjeldab portfelli juhtimist samuti üsna detailselt. Siiski annab SAFe selles osas paremad juhised [65]. LeSS Huge metoodika ei paku välja väga põhjalikku portfelli juhtimise reeglistikku, kuna tootearenduse vastutus on suurel määral tiimide sees [10].



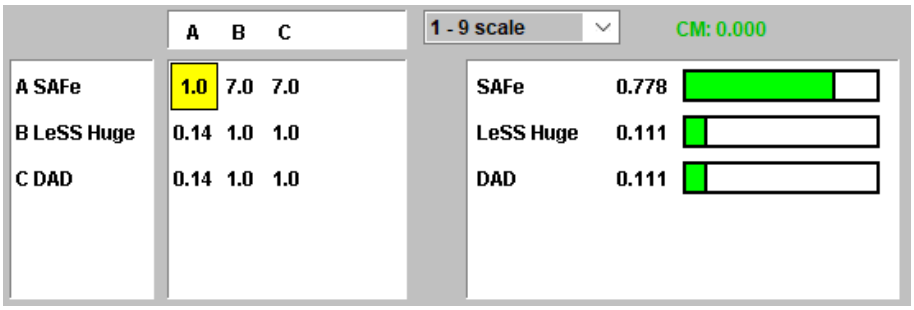
Joonis 30. "Kliendikeskus" Saaty skaala hinnangud.

Joonis 30 esitab metoodikate kliendikesksuse kaalud. LeSS metoodika üks olulisemaid põhiväärtusi on kliendikesksus ehk metoodika on ehitatud üles selliselt, et tootearendus oleks võimalikult kliendikeskse lähenemisega [10]. SAFe ja DAD kirjeldavad samuti kliendikesksuse aspekti, kuid võrreldes LeSS’iga ei ole see neil nii tugev põhiväärtus [58] [65].



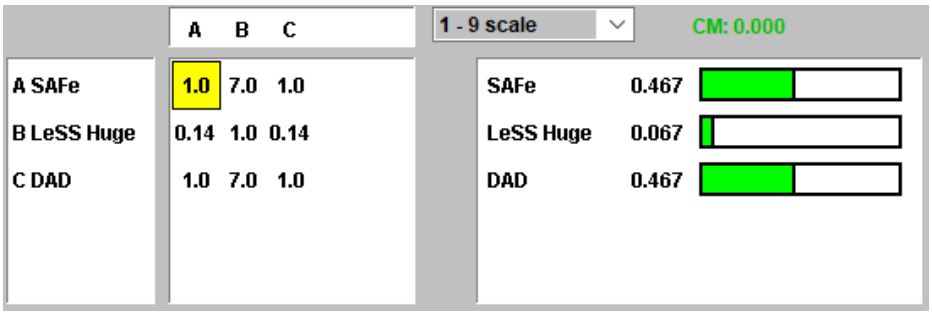
Joonis 31. "Sõltuvuste haldus" Saaty skaala hinnangud.

Joonis 31 on esitatud metoodikate sõltuvuste haldamise omavahelised kaalud. LeSS metoodikat järgides toimuvad arendustööde sõltuvusi käsitlevad kohtumised igas sprindis kõiki tiime või nende esindajaid kaasates (Joonis 14). SAFe puhul on sõltuvuste haldamiseks PI planeerimise üritus, kus järgmise PI tiimidevahelised sõltuvused saavad kindlaks tehtud. Lisaks toimub sprindi jooksul tooteomanike kohtumine ja rüsinameistrite SoS kohtumine, kus vajadusel saab sõltuvusi arutada (Joonis 6). DAD metoodika ütleb, et tiimid võivad ise valida, milliseid koordineerimise võimalusi nad sõltuvuste haldamiseks kasutavad, näiteks SoS kohtumised, ületiimilised suured planeerimise sessioonid või jooksvad sprindi kohtumised (Joonis 21) [65]. Autor leiab, et nii LeSS kui ka SAFe metoodika puhul on sõltuvuste käsitlemiseks vajalikud tegevused hästi ettekirjutatud, DAD aga ei paku välja ühte head lähenemisviisi, vaid kinnitab üle juba üldlevinud praktikaid. See võib aga autori arvates olla riskikoht.



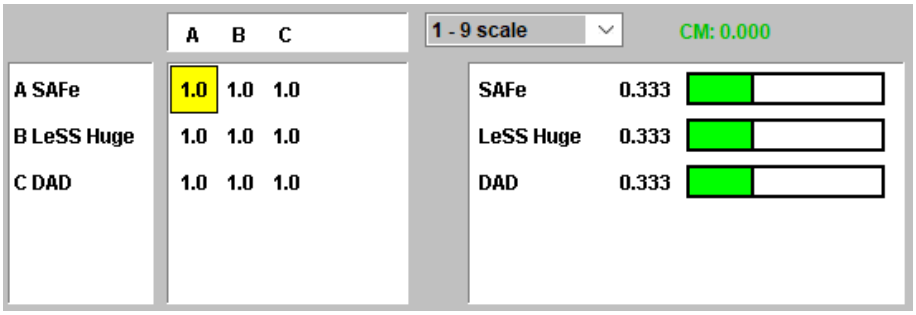
Joonis 32. "Mõõdikud" Saaty skaala hinnangud.

Joonis 32 kirjeldab mõõdikute hinnanguid. SAFe kirjeldab iga oma taseme jaoks mitmeid mõõdikuid, mis aitavad mõõta metoodika ja tootearenduse efektiivsust [58]. LeSS ja DAD pakuvad üsna minimaalse hulga mõõdikuid [10] [65].



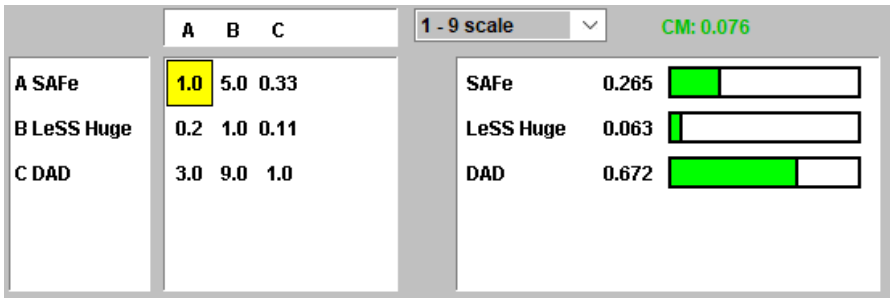
Joonis 33. "Komponent-tiim" Saaty skaala hinnangud.

Joonis 33 esitab hinnangud, kuidas metoodika toetab komponent-tiimide moodustamist. LeSS metoodika ütleb, et LeSSi rakendavad organisatsioonide arendustiimid peavad olema erisuse (*Feature*) tiimid, kes suudavad arendada mitut komponenti korraga, omades võimekust arendada välja üks terviklik kliendiväärtust loov funktsionaalsus. Samas ei välista LeSS täielikult ka komponent-tiimidega töötamist, kuid pakub juhised, kuidas struktureerida komponent-tiimid ümber erisuse tiimideks [10]. Selle asemel, et struktureerida tiimid äriväärtuse järgi, soovitab SAFe moodustada tiimid arhitektuurikomponentide ümber [68]. DAD ütleb, et metoodika rakendamiseks sobivad nii komponentide kui ka erisuse tiimid [65]. Antud ettevõte eelistab, et tiimid saaksid olla struktureeritud arhitektuuriliste komponentide järgi, sest nii tehakse ettevõttes ka praegu.



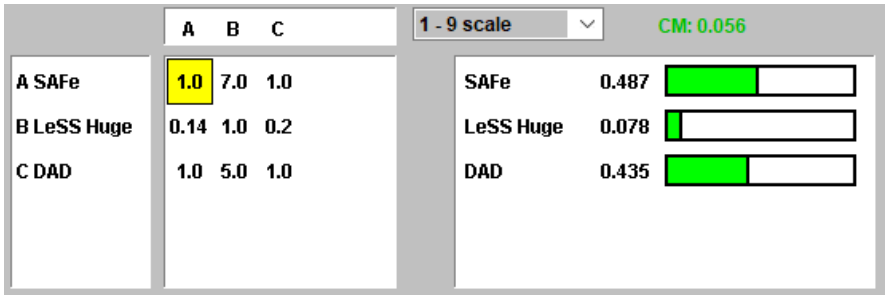
Joonis 34. "DevOps" Saaty skaala hinnangud.

Joonis 34 esitab metoodikate DevOps alamkriteeriumi kaalud. LeSS metoodika mitte ainult ei suuna DevOps lähenemist praktiseerima, vaid see on oluline osa LeSS metoodikast. Mida rohkem tiimid seda omaks võtavad, seda parem. [69] SAFe ütleb, et PI planeerimisel tuleb arvestada osa tiimi töömahust tehniliste tööde jaoks, mis samuti toetab DevOps lähenemist [58]. DAD metoodika puhul on DevOps kultuur samuti oluline osa metoodikast [65]. Autori hinnangul on DevOps kõikides metoodikates heakskiidetud ja seetõttu hindab kõiki metoodikaid selle alamkriteeriumi suhtes võrdsena.



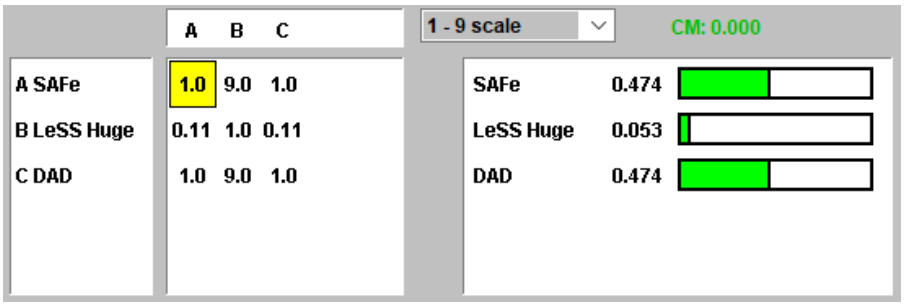
Joonis 35. "Arendusmetoodikad" Saaty skaala hinnangud.

Joonis 35 on näidatud arendusmetoodika kaalud. Vaadeldes metoodikate tiimi vaate metamudeleid (Joonis 4, Joonis 12, Joonis 19) võib järeldada, et LeSS metoodika puhul on tiimidel võimalik kasutada ainult Scrum arendusmetoodikat. SAFe metoodika võimaldab kasutada Scrum'i, Kanban'i, ScrumXP'd ja nende hübriide ning DAD võimaldab tiimidel praktiseerida veelgi rohkemaid arendusmetoodikaid ja nende hübriide.



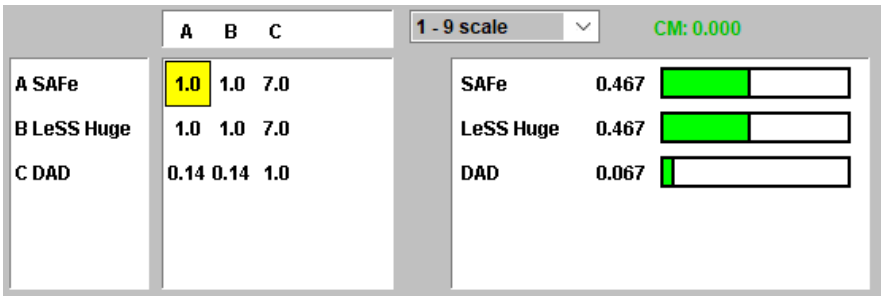
Joonis 36. "Tiimisisesed rollid" Saaty skaala hinnangud.

Joonis 36 esitab tiimisiseste rollide Saaty skaala hinnangud. DAD metoodikas võib tiimides ühel ajahetkel olla väga palju rolle, sest DAD kirjeldab teisesed rollid, mis kuuluvad tiimi ainult vajadusel (Joonis 19). Antud joonise järgi on igal tiimil enda tooteomanik. SAFe metoodika puhul eksisteerivad tiimides sarnased rollid (nagu DAD esmased rollid), kuid tiimijuhi asemel on rüsinameister (Joonis 4). SAFe metoodika järgi on samuti igal tiimil oma tooteomanik. LeSS Huge'l on sarnased rollid nagu DAD'il ja SAFe'l (Joonis 12), kuid iga valdkonna peale on üks tooteomanik. Teades ettevõttes valdkondade ja tiimide keerukust, leiab autor, et igal tiimil peaks olema enda tooteomanik, vastasel korral ei pruugi üks tooteomanik suuta kõikide tiimide tööd hallata. Vaadeldes ettevõtte olemasolevaid rolle (vt peatükk 3) leiab autor, et kõikide metoodikate rollid oleksid sobivad ja saaks võimalusel vastavusse viia olemasolevate rollidega. Kuna enamikes tiimides eksisteerib IT analüütik, siis nendest võiks saada tooteomanikud.



Joonis 37. "Ärilised rollid" Saaty skaala hinnang.

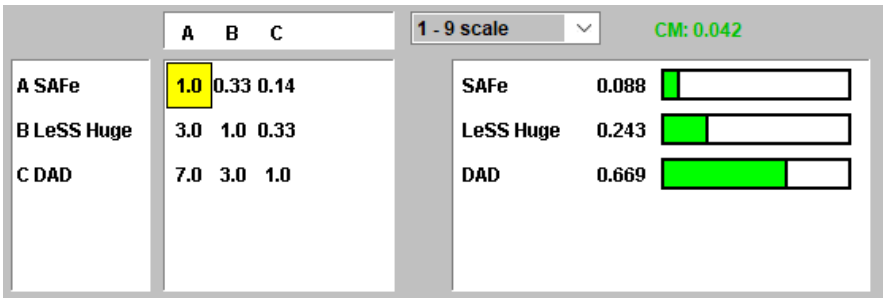
Joonis 37 esitab äriliste rollide Saaty skaala hinnangud. LeSS Huge ei kirjelda erinevaid ärilisi rolle, kuna vastutus tootearenduse eest on tiimide sees. Tootearenduse otsused tehakse tooteomanike tiimis koostöös tootegrupi juhiga. LeSS lähtub rollide kirjeldamisel oma põhiprintsiibist „*More with LeSS*“, mis tähendab, et LeSS ei kirjeldagi üleliia rolle, et vastutus ei hajuks. Olulised on eelkõige tiimisisesed rollid. SAFe määratleb sellised ärilised rollid nagu äriomanikud, eepose omanikud ja tootejuhid. DAD kirjeldab tootejuhi ja huvirühma rolli. Nii SAFe kui ka DAD metoodikate kirjeldatud ärilised rollid on antud ettevõttele sobivad, sest ettevõttes on erinevad teenustejuhid ja äriomanikud, kes on IT tellija rollis.



Joonis 38. „Juhendid“ Saaty skaala hinnangud.

Joonis 38 on toodud metoodikate juhendite põhjalikkuse hinnangud. SAFe metoodika kohta on kirjutatud 2018. aastal avaldatud raamat. Kuigi nüüdseks on tehtud metoodikas mõningad kohaldused, siis antud raamat on ikkagi veel sobiv alus metoodika kasutuselevõtmiseks [59]. Lisaks on SAFe kodulehel väga põhjalik kirjeldus metoodika eri versioonide kohta ja samuti pakub SAFe välja metoodika rakendamise tegevuskava (*Implementation Roadmap*) [58]. LeSS metoodika kohta on selle autorid avaldanud kolm raamatut, neist kõige uuem (2020. aasta kevade seisuga) ilmus 2016. aastal [52] [61] [62]. Samuti on LeSS kohulehel väga detailne info metoodika kohta ning välja on toodud erinevad juhtumiuuringud, kuidas LeSS'i on rakendatud [10]. DAD metoodika kohta on autorid kirjutanud neli raamatut, neist kõige uuem (2020. aasta kevade seisuga) ilmus

2019. aastal [66]. Samuti eksisteerib DAD koduleht, kus kirjeldatakse metoodikat ja antakse juhiseid. Küll aga on kodulehel olev info võrreldes SAFe ja LeSS kodulehega üsna pinnapealne.

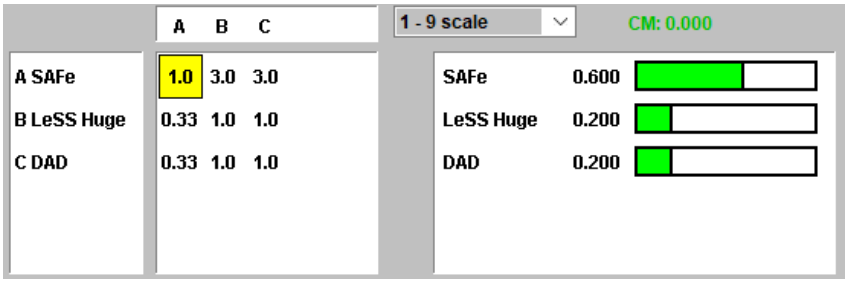


Joonis 39. "Kulud" Saaty skaala hinnangud.

Joonis 39 esitab kulude alamkriteeriumi Saaty skaala hinnangud. Autor on kulude kriteeriumi hindamisel lähtunud kodulehel pakutavate koolituste hindadest ühe inimese kohta. Ettevõttel on plaanis tellida üks põhjalik metoodika kasutuselevõttu käsitlev koolitus. Järgnevalt esitatakse kõikide metoodikate koolituste hinnad ühe inimese kohta.

- SAFe neljapäevane juurutamise koolitus – 2999 EUR [58]
- DAD neljapäevane juurutamise koolitus – 1699 EUR [65]
- LeSS *Huge* kolmepäevane juurutamise koolitus – 2499 EUR [10]

Kõige kallimad koolituste hinnad on SAFe metoodikal, veidi odavamad on LeSS Huge metoodikal ja DAD koolituse hind on SAFe koolituse hinnast peaaegu poole odavam.



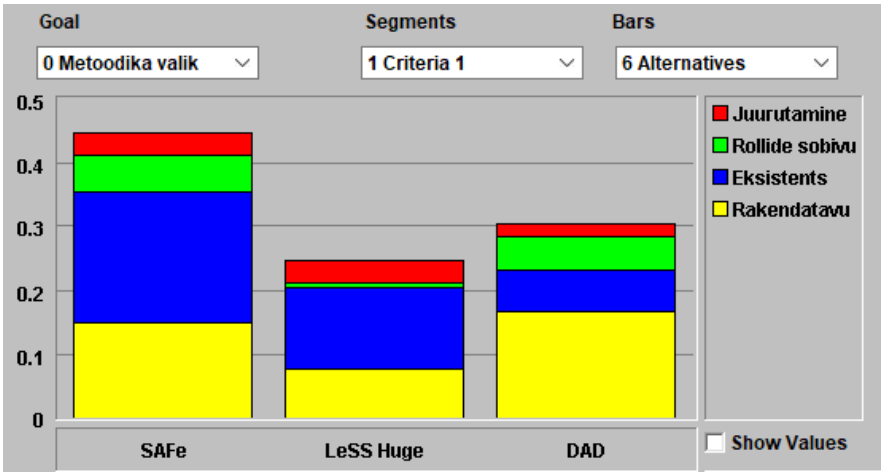
Joonis 40. „Koolitused“ Saaty skaala hinnangud.

Joonis 40 on näidatud metoodika juurutamiseks vajalike koolituste hinnangud. Vaadeldes nende metoodikate kodulehtedel pakutavaid koolitusi, võib järeldada, et kõikide metoodikate puhul pakutakse iganädalaselt eri riikides või virtuaalselt erinevaid konverentse ja koolitusi. Võrreldes nende sagedust, siis SAFe'l ja LeSS'il korraldatakse koolitusi igapäevaselt mitu korda, DAD metoodika koolitused nii tiheda intervalli tagant

ei toimu. SAFe pakub koolitusi konkreetsetele SAFe rollidele, LeSS ja DAD igale rollile eraldi suunatud koolitusi ei paku. [58] [10] [65]

6.2.7 Tulemused ja järeldused

Järgnevalt analüüsitakse otsustusmudeli tulemusi.



Joonis 41. Otsustusmudeli tulemused.

Joonis 41 ja Tabel 5 esitavad otsustusmudeli põhjal selgunud metoodikate järjestuse, millest järeldub, et otsustusmudeli tulemusel osutus kõige sobivamaks alternatiiviks *Scaled Agile Framework* metoodika. SAFe paremus teistest metoodikatest on üsna ülekaalukas. Jooniselt on näha, et SAFe sai kõige suuremad kaalud eksistentsi põhikriteeriumiga, DAD puhul sai kõige kõrgemad hinnangud rakendatavuse põhikriteeriumi alamkriteeriumid ning LeSS *Huge* puhul ei osutunud ühegi põhikriteeriumi alamkriteeriumite kaalud teiste metoodikate suhtes võitjaks.

Tabel 5. Otsustusmudeli arvulised tulemused.

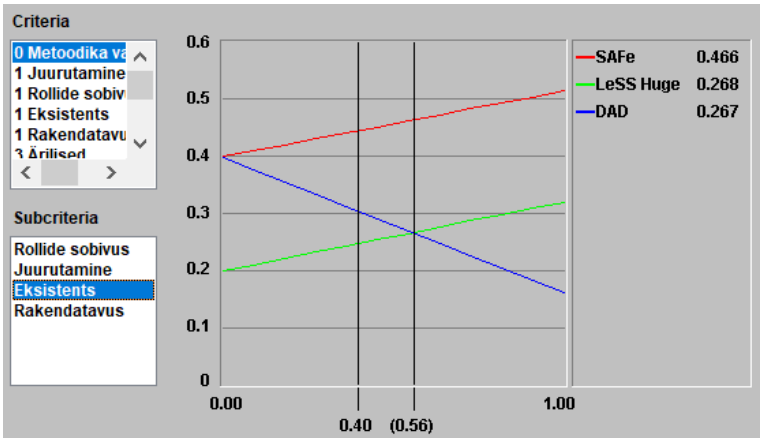
Põhikriteerium	**SAFe**	**LeSS Huge**	**DAD**
Eksistents	0.205	0.127	0.064
Rakendatavus	0.150	0.077	0.169
Rollide sobivus	0.057	0.008	0.052
Juurutamine	0.035	0.035	0.020
Kokku	**0.447**	**0.248**	**0.305**

6.3 Tundlikkuse analüüs

Antud jaotises viib autor läbi tundlikkuse analüüsi nii põhi- kui ka alamkriteeriumite suhtes. Tundlikkuse analüüs näitab, kas erinevate kriteeriumite osatähtsuse muutmine otsustusmudelis mõjutab võitjat ja alternatiivide järjestust. Lisaks annab tundlikkuse analüüs ülevaate sellest, kui palju tuleb teistsuguse tulemuse saamiseks osakaale muuta.

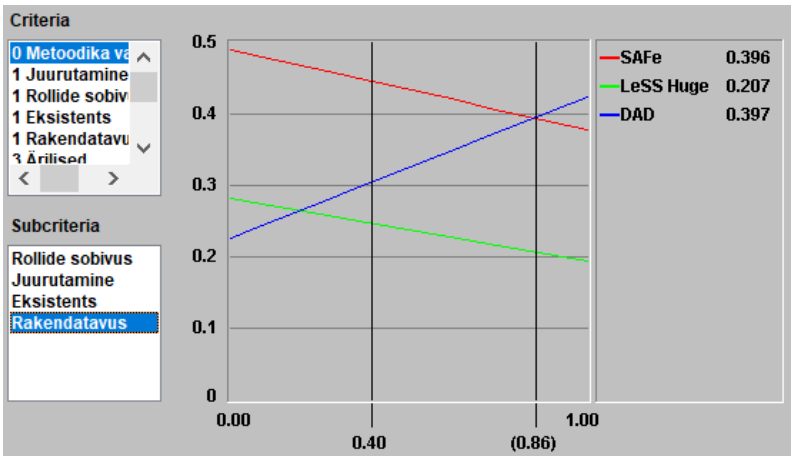
6.3.1 Põhikriteeriumite tundlikkuse analüüs

Käesolevas jaotises teostatakse otsustusmudeli põhikriteeriumitele tundlikkuse analüüs.



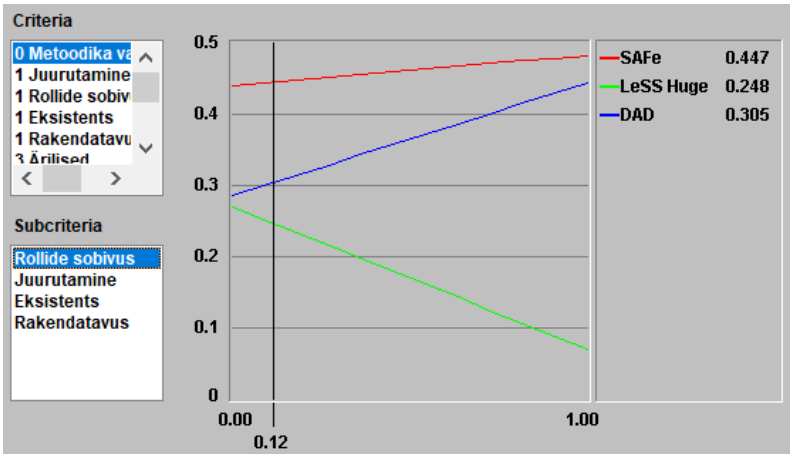
Joonis 42. "Eksistents" põhikriteeriumi tundlikkuse analüüs.

Joonis 42 esitab eksistentsi põhikriteeriumi tundlikkuse analüüsi graafiku, millest järeldub, et eksistentsi põhikriteeriumi osatähtsuse muutmine ei muuda otsustusmudeli valiku tulemust võitja osas. Küll aga on võimalik eksistents kriteeriumi suurendamisega muuta ülejäänud kahe alternatiivi järjestust lõpptulemuses. Kui kriteeriumi osatähtsust suurendada 0,40 pealt 0,56 peale, siis oleks DAD metoodika asemel sobivuselt teine alternatiiv LeSS *Huge* metoodika. Kaalude suhe teistesse kaaludesse enne muudatust on 0,40/(1-0,40)=0,67 ja peale muudatust 0,56/(1-0,56)=1,27. Seega eksistents kriteeriumi tähtsust teiste kriteeriumite suhtes tuleks suurendada keskmiselt 1,27/0,67=1,9 korda.



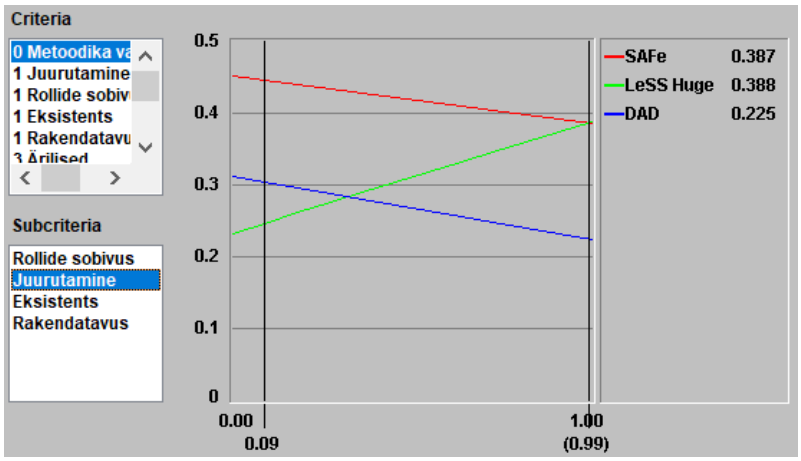
Joonis 43. "Rakendatavus" põhikriteeriumi tundlikkuse analüüs.

Joonis 43 esitab rakendatavuse põhikriteeriumi tundlikkuse analüüsi graafiku. Jooniselt järeldub, et kui kriteeriumi osatähtsust tõsta 0,40 pealt 0,86 peale, siis oleks sobivaks alternatiiviks DAD metoodika. Kaalude suhe teistesse kaaludesse enne muudatust on 0,40/(1-0,40)=0,67 ja peale muudatust 0,86/(1-0,86)=6,14. Seega rakendatavuse kriteeriumi tähtsust teiste kriteeriumite suhtes tuleks suurendada keskmiselt 6,14/0,67=9,16 korda. Antud muudatus rakendatavuse kriteeriumi suhtes oleks väga suur muudatus ja see ei oleks põhjendatud.



Joonis 44. "Rollide sobivus" põhikriteeriumi tundlikkuse analüüs.

Joonis 44 esitab rollide sobivuse põhikriteeriumi tundlikkuse analüüsi graafiku, millest järeldub, et rollide sobivuse põhikriteeriumi osatähtsuse muutmine ei muuda otsustusmudeli valiku tulemust.

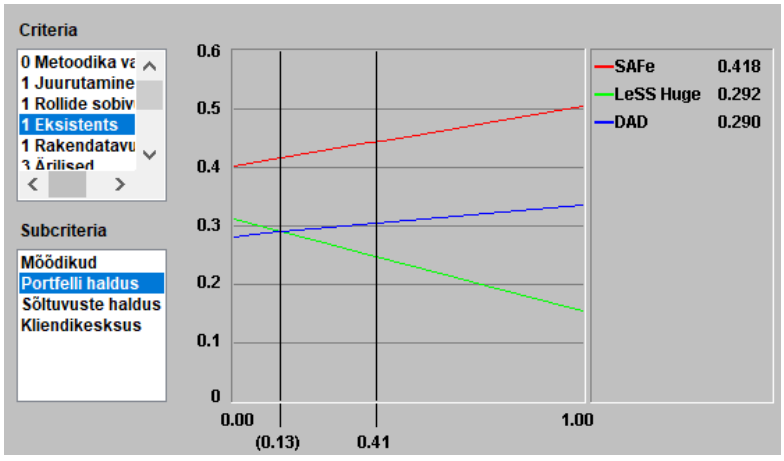


Joonis 45. "Juurutamine" põhikriteeriumi tundlikkuse analüüs.

Joonis 45 esitab juurutamise põhikriteeriumi tundlikkuse analüüsi graafiku. Jooniselt järeldub, et kui kriteeriumi osatähtsust tõsta 0,09 pealt 0,99 peale, siis oleks sobivaks alternatiiviks LeSS *Huge* metoodika. Kaalude suhe teistesse kaaludesse enne muudatust on 0,09/(1-0,09)=0,1 ja peale muudatust 0,99/(1-0,99)=99. Seega juurutamise kriteeriumi tähtsust teiste kriteeriumite suhtes tuleks suurendada keskmiselt 99/0,1=990 korda. Antud muudatus juurutamise kriteeriumi suhtes oleks liialt suur. Juurutamise kriteeriume ei oma nii suurt tähtsust, seega ei oleks selle suurendamine sellises mahus põhjendatud.

6.3.2 Alamkriteeriumite tundlikkuse analüüs

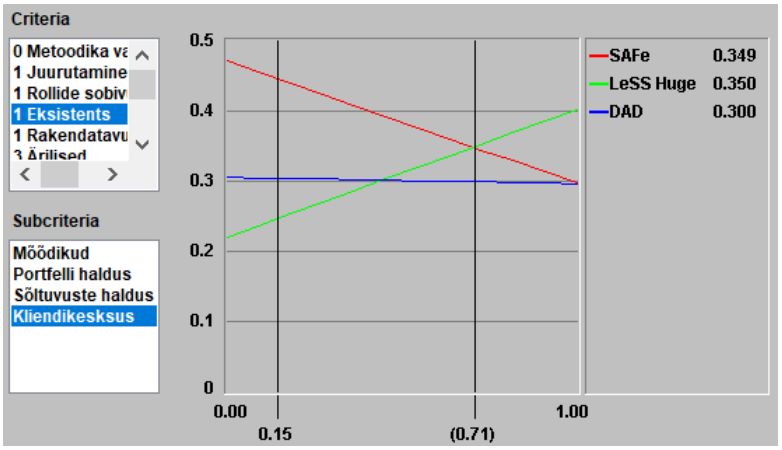
Käesolevas jaotises teostatakse otsustusmudeli alamkriteeriumitele tundlikkuse analüüs.



Joonis 46. "Portfelli haldus" alamkriteeriumi tundlikkuse analüüs.

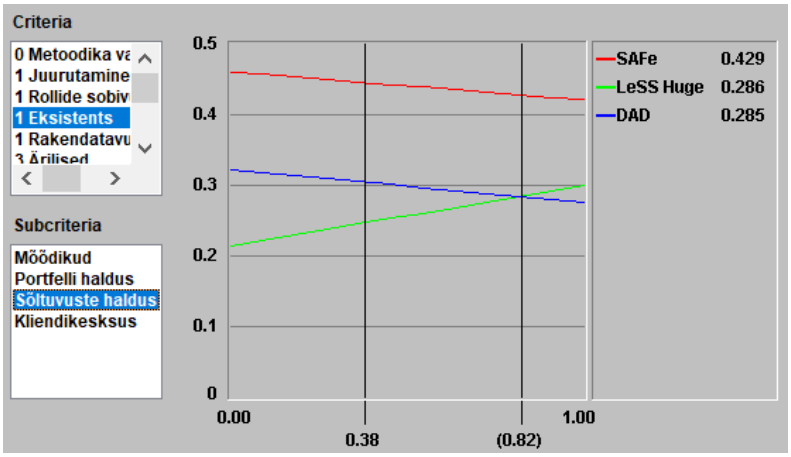
Joonis 46 esitab portfelli halduse alamkriteeriumi tundlikkuse analüüsi graafiku, millest järeldub, et portfelli halduse alamkriteeriumi osatähtsuse muutmine ei muuda otsustusmudeli valiku tulemust. Küll aga on võimalik portfelli halduse kriteeriumi

vähendamisega muuta ülejäänud kahe alternatiivi järjestust lõpptulemuses. Kui kriteeriumi osatähtsust vähendada 0,41 pealt 0,13 peale, siis oleks DAD metoodika asemel sobivuselt teine alternatiiv LeSS *Huge* metoodika. Kaalude suhe teistesse kaaludesse enne muudatust on 0,41/(1-0,41)=0,69 ja peale muudatust 0,13/(1-0,13)=0,15. Seega portfelli halduse tähtsust teiste kriteeriumite suhtes tuleks vähendada keskmiselt 0,69/0,15=4,6 korda. Ettevõtte jaoks on portfelli halduse kriteerium metoodikas väga oluline ja selle kriteeriumi osatähtsust vähendada ei tohi.



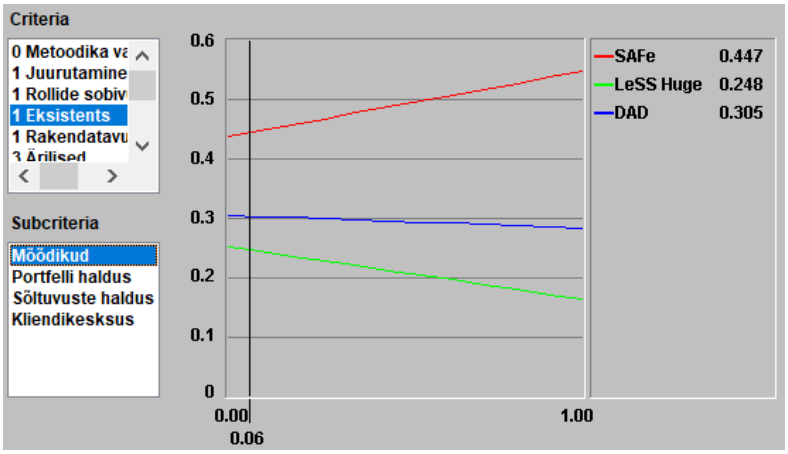
Joonis 47. "Kliendikesksus" alamkriteeriumi tundlikkuse analüüs.

Joonis 47 esitab kliendikesksuse alamkriteeriumi tundlikkuse analüüsi graafiku. Jooniselt järeldub, et kui kriteeriumi osatähtsust tõsta 0,15 pealt 0,71 peale, siis oleks sobivaks alternatiiviks LeSS *Huge* metoodika. Kaalude suhe teistesse kaaludesse enne muudatust on 0,15/(1-0,15)=0,18 ja peale muudatust 0,71/(1-0,71)=2,45. Seega kliendikesksuse kriteeriumi tähtsust teiste kriteeriumite suhtes tuleks suurendada keskmiselt 2,45/0,18=13,6 korda. Nii suur muudatus kliendikesksuse alamkriteeriumi suhtes oleks liialt suur. Kliendikesksuse olemasolu valitavad metoodikas on küll oluline, kuid lõpptulemuse muutmiseks vajalik sellises mahus kaalude suurendamine oleks põhjendamatu.



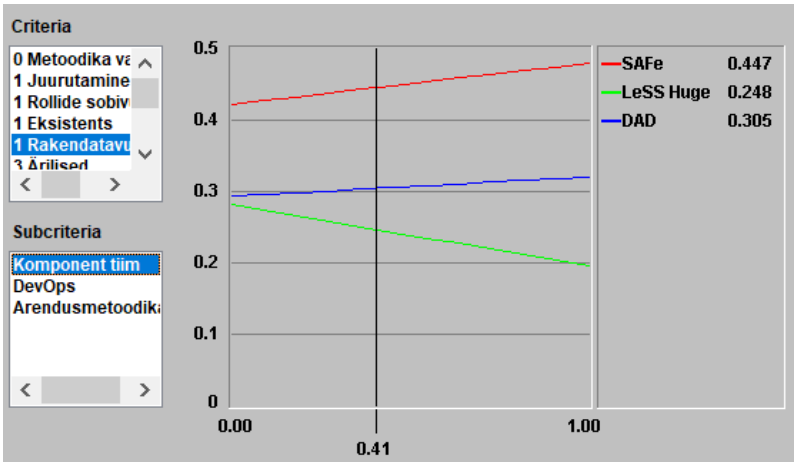
Joonis 48. "Sõltuvuste haldus" alamkriteeriumi tundlikkuse analüüs.

Joonis 48 esitab sõltuvuste halduse alamkriteeriumi tundlikkuse analüüsi graafiku, millest järeldub, et sõltuvuste halduse alamkriteeriumi osatähtsuse muutmine ei muuda otsustusmudeli valiku tulemust. Küll aga on võimalik sõltuvuste halduse kriteeriumi suurendamisega muuta ülejäänud kahe alternatiivi järjestust lõpptulemuses. Kui kriteeriumi osatähtsust tõsta 0,38 pealt 0,82 peale, siis oleks DAD metoodika asemel sobivuselt teine alternatiiv LeSS *Huge* metoodika. Kaalude suhe teistesse kaaludesse enne muudatust on 0,38/(1-0,38)=0,61 ja peale muudatust 0,82/(1-0,82)=4,56. Seega sõltuvuste halduse kriteeriumi tähtsust teiste kriteeriumite suhtes tuleks suurendada keskmiselt 4,56/0,61=7,48 korda. Ettevõtte jaoks on sõltuvuste halduse kriteerium metoodikas küll väga oluline, kuid nii suur kriteeriumi kaalu tõus ei oleks põhjendatud.



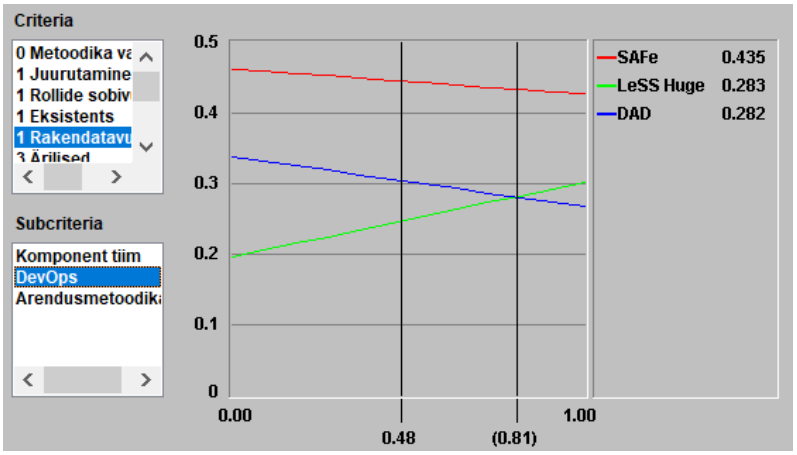
Joonis 49. "Mõõdikud" alamkriteeriumi tundlikkuse analüüs.

Joonis 49 esitab mõõdikute alamkriteeriumi tundlikkuse analüüsi graafiku, millest järeldub, et mõõdikute alamkriteeriumi osatähtsuse muutmine ei muuda otsustusmudeli valiku tulemust.



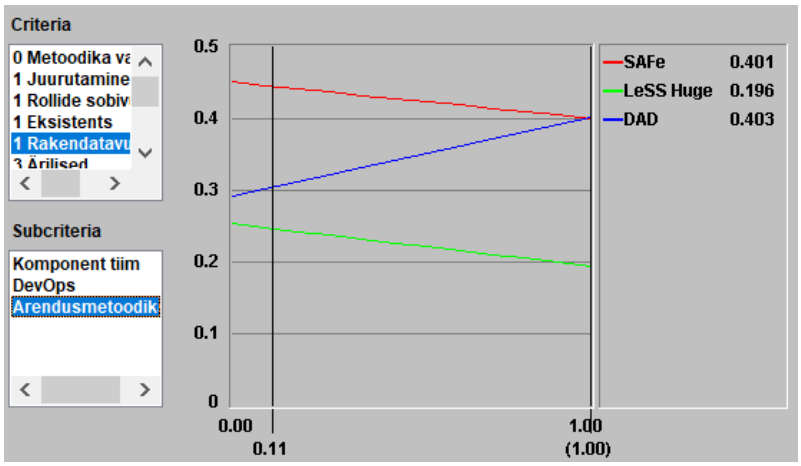
Joonis 50. "Komponent-tiim" alamkriteeriumi tundlikkuse analüüs.

Joonis 50 esitab komponent-tiimi alamkriteeriumi tundlikkuse analüüsi graafiku, millest järeldub, et komponent-tiimi alamkriteeriumi osatähtsuse muutmine ei muuda otsustusmudeli valiku tulemust.



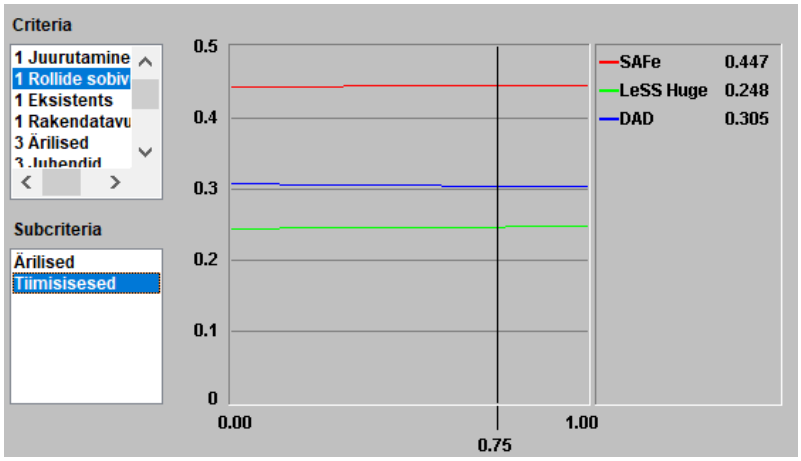
Joonis 51. "DevOps" alamkriteeriumi tundlikkuse analüüs.

Joonis 51 esitab DevOps alamkriteeriumi tundlikkuse analüüsi graafiku, millest järeldub, et DevOps alamkriteeriumi osatähtsuse muutmine ei muuda otsustusmudeli valiku tulemust. Küll aga on võimalik DevOps kriteeriumi suurendamisega muuta ülejäänud kahe alternatiivi järjestust lõpptulemuses. Kui kriteeriumi osatähtsust tõsta 0,48 pealt 0,81 peale, siis oleks DAD metoodika asemel sobivuselt teine alternatiiv LeSS *Huge* metoodika. Kaalude suhe teistesse kaaludesse enne muudatust on 0,48/(1-0,48)=0,92 ja peale muudatust 0,81/(1-0,81)=4,26. Seega DevOps kriteeriumi tähtsust teiste kriteeriumite suhtes tuleks suurendada keskmiselt 4,26/0,92=4,63 korda.



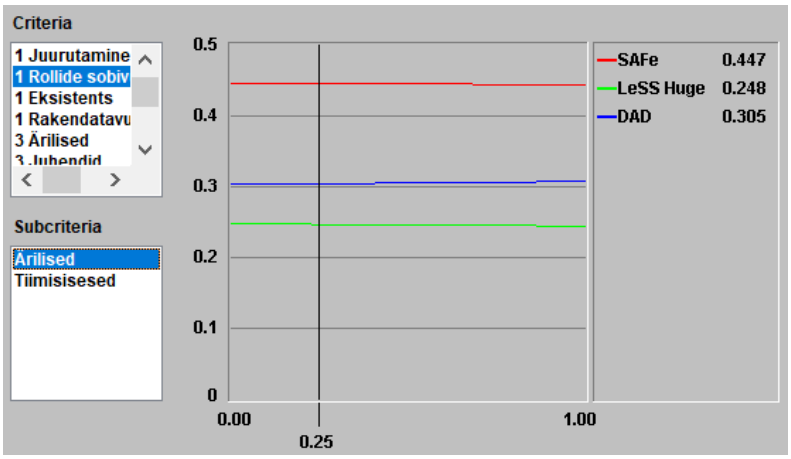
Joonis 52. "Arendusmetoodika" alamkriteeriumi tundlikkuse analüüs.

Joonis 52 esitab arendusmetoodika alamkriteeriumi tundlikkuse analüüsi graafiku, millest järeldub, et arendusmetoodika alamkriteeriumi osatähtsuse muutmine ei muuda otsustusmudeli valiku tulemust.



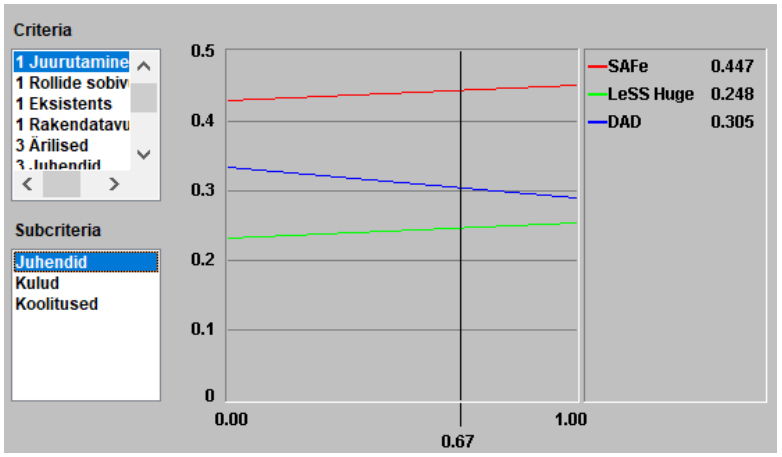
Joonis 53. "Tiimisisesed" alamkriteeriumi tundlikkuse analüüs.

Joonis 53 esitab tiimisiseste rollide alamkriteeriumi tundlikkuse analüüsi graafiku, millest järeldub, et tiimisiseste rollide alamkriteeriumi osatähtsuse muutmine ei muuda otsustusmudeli valiku tulemust.



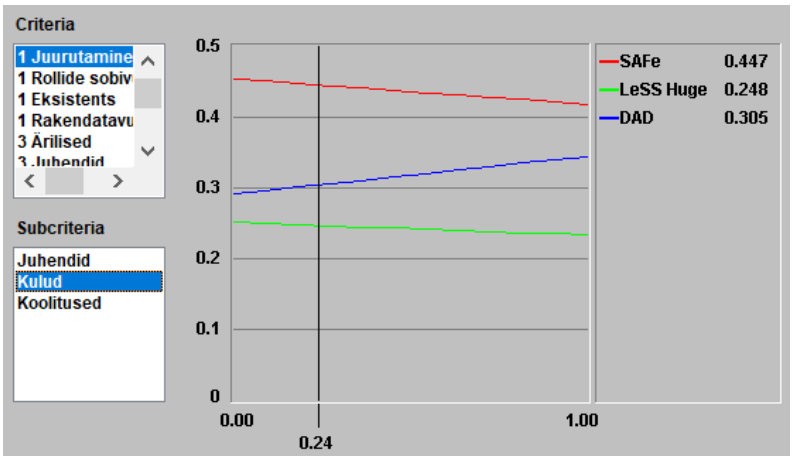
Joonis 54. "Ärilised rollid" alamkriteeriumi tundlikkuse analüüs.

Joonis 54 esitab äriliste rollide alamkriteeriumi tundlikkuse analüüsi graafiku, millest järeldub, et äriliste rollide alamkriteeriumi osatähtsuse muutmine ei muuda otsustusmudeli valiku tulemust.



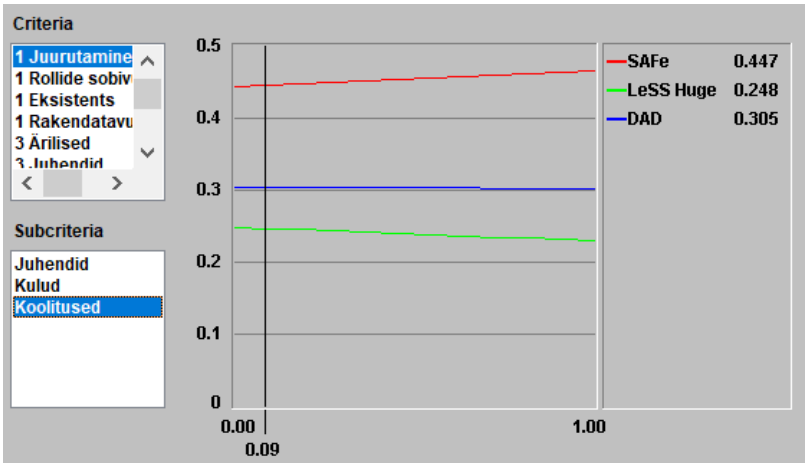
Joonis 55. "Juhendid" alamkriteeriumi tundlikkuse analüüs.

Joonis 55 esitab juhendite alamkriteeriumi tundlikkuse analüüsi graafiku, millest järeldub, et juhendite alamkriteeriumi osatähtsuse muutmine ei muuda otsustusmudeli valiku tulemust.



Joonis 56. "Kulud" alamkriteeriumi tundlikkuse analüüs.

Joonis 56 esitab kulude alamkriteeriumi tundlikkuse analüüsi graafiku, millest järeldub, et kulude alamkriteeriumi osatähtsuse muutmine ei muuda otsustusmudeli valiku tulemust.



Joonis 57. "Koolitused" alamkriteeriumi tundlikkuse analüüs.

Joonis 57 esitab koolituste alamkriteeriumi tundlikkuse analüüsi graafiku, millest järeldub, et koolituste alamkriteeriumi osatähtsuse muutmine ei muuda otsustusmudeli valiku tulemust.

7 Tulemuste analüüs ja järeldused

Käesoleval magistritööl oli kaks peamist eesmärki. Esimene eesmärk oli uurida erinevaid skaleeritavaid paindmetoodikaid ning luua uurmise käigus neid metoodikaid kirjeldavad metamudelid nendele, mida potentsiaalselt võiks ettevõttes X kasutada. Töö teiseks eesmärgiks oli valida ettevõttele X autori poolt uuritud paindmetoodikate hulgast sobiv. Selle eesmärgi täitmiseks tuli luua Saaty otsutusmudel. Käesolevas peatükis analüüsitakse töö tulemusi vastavalt püstitatud eesmärkidele ja tehakse järeldusi. Lisaks tuuakse antud peatükis välja töö võimalikud nõrkused ja esitatakse ettepanekud selle kohta, mida antud töö teemal edasi uurida.

7.1 Loodud metamudelite analüüs

Metamudelite loomiseks kirjeldati ära hulk põhimõtteid, mida modelleerimisel peaks alati järgima. Lisaks mõtles autor välja, milliseid metamudelite vaateid metoodikate kohta tuleks esitada. Kokku mõeldi välja neli vaadet, mis peaksid tagama metoodikast arusaamise. Autor lähtus vaadete loomisel sellest, et võimalikult palju omavahelisi seoseid omavad elemendid koondada samale diagrammile või lähedaselt seotud diagrammidele. Kirjeldatud põhimõtted ja vaated on üks võimalik metoodikate modelleerimise lähenemisviis, mis võivad olla ka edaspidi aluseks metoodikate metamudelite loomisel.

Kokku uuriti lähemalt viite metoodikat – *Scaled Agile Framework* (SAFe), *Disciplined Agile Delivery* (DAD), *Large Scale Scrum* (LeSS), *Large Scale Scrum Huge* (LeSS *Huge*) ja *Nexus*. Selle tulemusel loodi 17 diagrammi. Kuna LeSS on LeSS *Huge* metoodika osa, siis esimesed kolm LeSS vaadet kehtivad ka LeSS *Huge* metoodikal. LeSS *Huge* kohta esitati täiendav tootearenduse vaade.

Tabel 6. Klasside arv metamudelites.

Metoodika nimi	**Klasside arv**
Disciplined Agile Delivery	48
Scaled Agile Framework	45
Large-Scale Scrum Huge	37
Nexus	33
Large-Scale Scrum	30

Võrreldes metamudeleid üksteisega, võib elementide arvu põhjal (Tabel 6) järeldada, et SAFe ja DAD on kõige detailsemad metoodikad. *Nexuse*, LeSS ja LeSS *Huge* metoodikate metamudelites on elementide arv väiksem. Autor leiab, et *Nexuse* puhul on selle põhjuseks see, et antud metoodika on sobiv väiksemale tootearendusega organisatsioonile, seega see metoodika ei kirjelda erinevaid organisatoorseid üksuseid, mida on vaja koordineerimiseks või portfelli juhtimiseks. LeSS metoodikate puhul on see kooskõlas metoodika põhiprintisiibiga „*More with LeSS*“, mis tähendab, et LeSS üks eesmärkidest on hoida erinevate rollide ja tehiste arv võimalikult väiksena.

Metamudelite võrdlemise tulemusel järeldas autor, et SAFe on DAD paindmetoodikaga sarnasem kui SAFe ja LeSS *Huge*, sest nende mõisted kattusid omavahel kõige suuremal määral. Otsustusmudeli lõpptulemusel oli alternatiivide järjekorras esimeseks SAFe, seejärel DAD ja viimaks LeSS *Huge*. Autor leiab, et mudelite võrdlusel (vt jaotis 5.6) mõistete kattuvusest järeldatud SAFe ja DAD sarnasus ning otsustusmudeli alternatiivide järjekord on omavahel kooskõlas. Kuna SAFe ja DAD on sarnasemad, siis on loogiline, et nad paiknevad otsutusmudeli alternatiivide lõppjärjestuses järjestikku.

7.2 Otsustusmudeli tulemuste analüüs

Otsustusmudeli loomisel arvestati ettevõtte kontekstiga. Seega selliseid kriteeriumeid nagu organisatsiooni tüüp ja tiimide arv tootearenduses ei kaasatud, sest alternatiivide valikul arvestati juba nende kriteeriumitega. Otsustusmudelisse valiti neli põhikriteeriumit ja kaksteist alamkriteeriumit. Põhikriteeriumiteks olid eksistents, rakendatavus, rollide sobivus ja juurutamine. Kõige suurtemate kaaludega hinnati võrdselt eksistentsi ja rakendatavuse kriteeriumeid, rollide sobivus ja juurutamine olid

madalamate kaaludega. Kriteeriumite kaalud kooskõlastati autori kolleegiga. Eksistents ja rakendatavus kriteeriumid said kõige suuremad hinnangud, sest need mängivad kõige suuremat rolli skaleeritava paindmetoodika valimisel. Alamkriteeriumite osas olid suuremate kaaludega portfelli halduse kriteerium, sõltuvuste haldus, DevOps, tiimisisesed rollid ning erinevate juhiste olemasolu ja kättesaadavus metoodika kasutuselevõtmisel. Alamkriteeriumite kaalud kooskõlastati samuti ettevõtte eksperdiga. Alternatiivideks valiti SAFe, DAD ja LeSS *Huge* metoodikad. *Nexus* ja (väiksem) LeSS ei sobinud, kuna neid ei saa nii ulatuslikult skaleerida kui selleks on antud ettevõtte puhul vajadus.

Autori eesmärk oli uurida, kas praegu juurutatav SAFe on ettevõttele sobivaim metoodika. Otsustusmudeli kasutamise lõpptulemuseks oligi SAFe metoodika valik, mille võit oli üsna ülekaalukas. Teiseks sobivaks alternatiiviks oli DAD metoodika ja viimaseks jäi LeSS *Huge* metoodika. Kõige suuremate kaaludega alamkriteeriumid saidki SAFe metoodika puhul enamasti kõrgemad hinnangud, mis tõttu lõpptulemuseks oligi SAFe metoodika võit.

Otsustusmudelile tehti kõikide põhi- ja alamkriteeriumite suhtes tundlikkuse analüüs. Põhikriteeriumitest muudaks rakendatavuse ja juurutamise osatähtsuse muutmine otsustusmudeli valiku tulemust. Küll aga peaks antud kriteeriumite kaale suurendama mitmeid kordi, mis ei oleks antud kriteeriumite puhul põhjendatud (Joonis 43, Joonis 45). Alamkriteeriumitest mõjutaks lõpptulemust ainult kliendikesksuse kriteeriumi suurendamine 13,6 korda, mille lõpptulemus oleks seljuhul LeSS *Huge* metoodika valik (Joonis 47). Teiste alamkriteeriumite osatähtsuste muutmine ei mõjutanud lõpptulemust (võitjat). Portfelli halduse, sõltuvuste halduse ja DevOps alamkriteeriumite osatähtsuste muutmisel muudaks see lõpptulemuse järjestust teise ja kolmanda koha osas.

Loodud otsustusmudel on taaskasutatav, kuid alati tuleb lähtuda organisatsiooni kontekstist, mis määrab kriteeriumid, mida otsustusmudelis kasutada ning nende kaalud ja sellest tuleneva suhtelise olulisuse. Seega on antud otsustusmudel üks võimalik näidis, mida saab aluseks võtta mõne teise organisatsiooni puhul, mis seisab samuti skaleeritava paindmetoodika valiku ees. Antud otsustusmudel sai loodud sõltuvalt konkreetsest organistastiooni tüübist – suurettevõte. Samamoodi saab seda aluseks võtta ja rakendada ka väiksema ettevõtte tootearenduse korral. Mitmed kriteeriumid nagu näiteks sõltuvuste haldus, mõõdikute olemasolu, rollide sobivus ja juurutamine on universaalsed

kriteeriumid, mida on vaja hinnata ükskõik mis tüüpi organisatsiooni puhul sobivat metoodikat valides. Väiksema tootearendusega ettevõtte puhul peaks otsutusmudelist ära võtma portfelli halduse kriteeriumi, sest väikse tootearendusega ettevõttes ei ole vaja tooteportfelli juhtida kuna arendatakse ühte toodet.

7.3 Autori poolt väljapakutud terminid

Skaleeritavaid paindmetoodikaid kirjeldades märgistas autor tärniga (*) need mõisted, mille eestikeelset tõlget IT-alasest kirjandusest ei õnnestunud leida, mistõttu mõtles autor ise sobiva eestikeelse termini välja. Kokku oli selliseid mõisteid kuus. Tabel 7 tuuakse välja metoodika, kus antud mõiste oli kasutusel, selle ingliskeelne ja autori poolt väljapakutud eestikeelne tõlge.

Tabel 7. Väljapakutud terminite eestikeelsed tõlked.

Paindmetoodika	**Ingliskeelne termin**	**Eestikeelne termin**
SAFe	*Release Agile Train* (ART)	Agiilne väljalaske rong
	Program Increment (PI)	Programmi juurdekasvu ajaraam
Nexus	*Integrated Increment*	Integreeritud juurdekasv
LeSS	*Shippable Product Increment*	Toote juurdekasv
Kõik metoodikad	*Definition of Done*	„Valmis“ definitsioon
	Backlog Refinement	Soovilogi täpsustamise kohtumine

7.4 Tehtud töö nõrkused

Iga lõputöö kohta võib leida puudusi. Käesolevas jaotises analüüsib autor aspekte, mida ta peab antud töö nõrkusteks. Autori eesmärk on teadvustada ja põhjendada asjaolusid, mida lugejad võivad antud töö kitsaskohtadena näha.

George E. P. Box on (statistiliste mudelite kohta) öelnud, „Kõik mudelid on valed, aga mõned on kasulikud.“ [70]. Antud tsitaadi idee on, et iga mudelit võib pidada valeks, sest mudel ei kajasta kunagi täpselt reaalselt maailma – selle struktuuri ja käitumist vaatamata sellele, kui palju seda täpsustatakse. Autor leiab, et see kehtib ka sellise modelleerimise kohta, mida tehakse selles lõputöös. Hoolimata sellest, et mudel ei suuda reaalsust täpselt kirjeldada, on see siiski kasulik, kui selle esitatud ettekujutus modelleeritavast maailmast

on reaalsusele *piisavalt* lähedane ja samas ka piisavalt lihtne. [71] Nii on ka lõputöö autor mudelite koostamisel üritanud teha need võimalikult realistlikud. Lisaks mängib mudelite loomisel suurt rolli tõlgendamine. Mudelid on kasulikud, kui kõik osapooled nendest ühtemoodi aru saavad ja tõlkes midagi kaduma ei lähe. Autor näeb siinkohal ohtu, et töö lugeja võib mudeleid erinevalt tõlgendada ning seetõttu teha järeldusi, et tegemist ei ole korrektselt esitatud mudelitega.

Samuti võib nõrkuseks pidada, et mudelid esitavad ühe inimese (autori) arusaama metoodikast ja tema valikut olulistest elementidest ja nende seostest. Kuna käesoleva lõputöö mudelid kajastavad väga mitmeid erinevaid elemente ja nende omavahelisi seoseid, tekib palju võimalusi neid ka erineval viisil modelleerida. Tagamaks metamudelitest võimalikult sarnase arusaamise lisas autor igale loodud diagrammile tekstilise kirjelduse. Seega, kui diagrammi uurides jääb mõningad asjaolud arusaamatuks, siis tekstilise kirjelduse lugemine võiks selle probleemi lahendada. Lisaks vaatas autor mudeleid üle koos valdkonna ekspertidega, mis andis võimaluse arusaame ühtlustada ja vastavalt sellele teha mudelites korrektuure. Modelleerimisel on lähtutud põhimõttest väljendada mudelites seda, mida kirjanduses on teoreetiliselt öeldud. Seega näiteks võimsustikud kajastavad ideaaljuhtu, aga reaalsuses võib esineda eriolukordi, mis ei lange mudelites esitatud võimsustikega kokku.

Autor leiab, et veel üheks võimalikuks töö nõrkuseks on, et otsustusmudelis metoodikatele antud hinnangud põhinevad autori enese arvamusel. Autori töökohal viiakse läbi SAFe raamistiku juurutusprotsess ja autor osaleb selles, mistõttu on ta selle metoodikaga paremini kursis ja võib kasvõi alateadlikult kalduda seda eelistama. Küll aga vaadati kriteeriumite suhtelist olulisust autori töökohal oleva eksperdiga üheskoos üle ja arutati need läbi. Lisaks aitab tulemuse kvaliteeti parandada see, et saadud tulemusele tehti tundlikkuse analüüs, mis näitas, et lõpptulemust oleks võimalik muuta ainult teatud kriteeriumite kaalude suurendamisega mitmekordselt, mis aga poleks põhjendatud. Seega antud juhul on autor selle nõrkuse avaldumist ennetanud ja proovinud kirjeldatud meetmetega seda vähendada.

7.5 Ettepanekud edasisteks uuringuteks

Käesoleva lõputöö edasiarendusteks on mitmeid variante. Üks võimalik variant on arendada edasi loodud metamudeleid ning nende esitust. Näiteks võiks läheneda vaadete

loomisele mõnel muul viisil, st uusi vaateid luues muud moodi elemente diagrammile valida. Üks võimalik variant on luua vaateid erinevate tseremooniate järgi, näiteks sprindi planeerimise vaade ja selles võiks esitada kõik seda tseremooniat iseloomustavad olulisemad elemendid – tehised, sündmused ja rollid, mis antud sündmusesse kuuluvad (elementideks sprint, planeerimine, nõue, DoD, tooteomanik, arendustiim, soovilogid jne.). Teine variant metamudelitega seotud uurimistööks oleks uurida teisi, käesoleva lõputöö skoobist väljajäänud, metoodikaid. Kuna töö alguses tegi autor valiku võimalikest sobivamatest metoodikatest ja välistas RAGE ja *Spotify* metoodikad, siis võiks neid lähemalt uurida. Samuti võiks uurida DAD metoodika teisi elutsükleid ja SAFe teisi verisoone (sh lahenduse taset). Uurimise käigus tuleks samamoodi luua metoodika tähtsamaid elemente ja nende seoseid kirjeldavad metamudelid ning võrrelda neid metoodikaid metamudelite põhjal käesoleva lõputöö raames laiemalt käsitletud metoodikate metamudelitega.

Käesolevas lõputöös loodud otsustusmudeli edasise uuringu üheks võimaluseks on selle rakendamine mõne teise ettevõtte näitel, mis seisab skaleeritava paindmetoodika valiku ees. Eesmärgiks võiks olla võtta aluseks loodud otsustusmudel, seda kohaldada vastavalt ettevõtte kontekstile ning vaadelda tulemit. Sellega seoses võiks võrrelda käesoleva lõputöö ja uue uuringu käigus loodud otsustusmudeleid. Töö käigus võiks võrrelda mõlemas otsustusmudelis esitatud kriteeriumeid ja teha järeldusi selle kohta, millised kriteeriumid mängivad paindmetoodika valiku tegemisel (sõltumata kontekstist) alati rolli.

Kolmandaks edasisise uurimise suunaks pakub autor välja uurida mõne ettevõtte näitel antud töös käsitletud paindmetoodikate kasutusele võtmise protsessi. Kõigepealt tuleb uurida, milline on juurutava metoodika variandi sisu antud lõputöö põhjal loodud metamudelite ja nende kirjelduste põhjal. Seejärel tuleb juurutamise protsessi ning tulemuste põhjal hinnata, kas see, mida metoodika väidab ja milliseid probleeme väidetavalt lahendab, on tegelikult ka nii või mitte. Samuti tuleb kindlaks teha millised uued probleemid kerkivad esile metoodikat kasutusele võttes ja kuidas neid lahendada.

Nüüdseks on ettevõttes X SAFe’i juurutatud mõned kuud ja autor on selle aja jooksul täheldanud mõningaid probleeme. Näiteks, SAFe järgi toimub teatud intervalli tagant PI planeerimine, mis esmapilgul tundub hea viis, kuidas kõik tiimid saavad paari päevaga oma tööd ära planeerida. Selle sündmuse õnnestumiseks peab tiimidel olema väga hea

ülevaade eesolevatest arendustöödest. Selleks, et arendustöid planeerida, on vaja neid eelnevalt detailselt analüüsida ja teada nende töömahtu. Sage on olukord, kus enne PId ei ole tiimidel piisavalt aega sprintide kõrvalt sellega tegeleda ja PI planeerimisel ei õnnestu kogu PI kava kokku leppida. See on lihtsalt üks võimalik näide probleemidest, mida autor on metoodika kasutuselevõtu käigus täheldanud.

8 Kokkuvõte

Käesoleval lõputööl olid järgmised peamised eesmärgid.

- Uurida erinevaid skaleeritavaid paindmetoodikaid ja luua neid kirjeldavad metamudelid, kus on esindatud metoodikale iseloomulikud tegutsejad, tehised, tegevused ja nende omavahelised seosed ning mudelite sõnalised selgitused.
- Valida konkreetsele ettevõttele (ettevõte X) sobiv skaleeritav paindmetoodika kasutades Saaty analüütiliste hierarhiate meetodit.

Nimetatud eesmärgid saavutati. Kõigepealt toodi töös välja lõputööga sarnasel teemal varem ilmunud uuringud. Selliste tööde otsingu tulemusel tehti kindlaks, et kavandatud ulatuses ja tasemel ei ole varem sarnaseid uuringuid tehtud. Töö esimeses pooles analüüsiti viit skaleeritavat paindmetoodikat. Selle raames loodi neid metoodikaid kirjeldavad metamudelid ja mudelite sõnalised kirjeldused. Uuritavate metoodikate hulka valiti sellised, mis võiksid sobida konkreetsele ettevõttele. Enne modelleerimise alustamist kirjeldas autor ära, millistest põhimõtetest ta selle käigus lähtub ning milliseid vaateid planeerib ta töös esitada. Iga metoodika kohta loodi kuni neli vaadet. Valminud mudelid vaadati üle koos erinevate osapooltega. Selle eesmärgiks oli teha kindlaks mudelite arusaadavus ning leida parandamist vajavaid kohti. Metamudelite valideerimisel osales neli isikut, kellest kaks tundsid metoodikaid hästi ja teised kaks ei olnud varem neid praktikas kasutanud. Autor valis ülevaatuse protsessi nii metoodikate tundjad kui mittetundjad, et teha kindlaks, kas mudelitest saavad aru nii eksperdid kui ka need, kes metoodikaid ei tunne. Kolme metamudelit (ettevõttele X esialgsel hinnangul kõige sobivamate metoodikate kohta) võrreldi ka autori poolt omavahel ontoloogilise analüüsi meetodit kasutades, et teha kindlaks nende metoodikate peamised erinevused ja sarnasused.

Sobiva skaleeritava paindmetoodika valimiseks koostati Saaty analüütiliste hierarhiate meetodil põhinev otsustusmudel. Selle jaoks valiti välja otsustusmudeli kriteeriumid arvestades ettevõtte konteksti. Alternatiivid valiti modelleeritud paindmetoodikate seast. Viiest modelleeritud metoodikast valiti alternatiivideks kolm – *Large-Scale Scrum Huge*, *Scaled Agile Framework* ja *Disciplined Agile Delivery*. Otsustusmudeli alusel osutus

valituks *Scaled Agile Framework*. Lisaks tehti otsustusele tundlikkuse analüüs, mille sisuks oli teha põhjendatud muutusi ja vaadelda, kas otsustus selle tulemusel muutus. Tundlikkuse analüüs ei mõjutanud lõpptulemust.

Töö lõpus analüüsiti tulemusi ning tehti järeldused. Lisaks analüüsiti, mis on käesoleva lõputöö võimalikud nõrkused ning tehti ettepanekud edasisteks uuringuteks.

Valminud magistritöö võib pakkuda huvi organisatsioonide esindajatele, kes seisavad paindmetoodika valiku ees ning kellele lõputöö tulemusel loodud otsustusmudel annab eeskuju, kuidas ise samasugust valikut teha. Samuti sobib lõputöö lugejaskonnale, kes soovivad erinevatest paindmetoodikate mõistetest ja ülesehitusest metamudelite abil paremini aru saada.

Tuginedes ülalkirjutatule, võib järeldada, et magistritöös püstitatud eesmärgid saavutati.

Kasutatud kirjandus

[1] P. Diebold, A. Schmitt ja S. Theobald, „Scaling Agile – How to Select the Most Appropriate Framework,“ Proceedings of the 19th International Conference on Agile Software Development: Companion., 2018.

[2] T. L. Saaty, „Decision making with the analytic hierarchy process,“ Int. J. Services Sciences, Pittsburgh, USA, 2008.

[3] E. Eessaar ja R. Sgirka, „An Ontological Analysis of Metamodeling,“ Tallinn University of Technology, Tallinn, 2011.

[4] „Scaled Agile. SAFe Glossary.,“ [Võrgumaterjal]. Available: https://www.scaledagileframework.com/glossary/. [Kasutatud 23 02 2020].

[5] P. Parmakson, „Arendussõnastik,“ [Võrgumaterjal]. Available: https://agiil.github.io/sonastik/. [Kasutatud 25 03 2020].

[6] V. Hanson, M. Laur, M. Freudenthal, M. Seeba, B. Tulit ja J. Krasnosjolov, „Standardipõhine tarkvaratehnika sõnastik,“ Cybernetica, [Võrgumaterjal]. Available: https://stats.cyber.ee/. [Kasutatud 12 03 2020].

[7] K. Jaakson, „Aegridade esitamine SQL-andmebaasides OHLCV andmete näitel,“ Magistritöö. Tallinna Tehnikaülikool, Tallinn, 2019.

[8] „DevOps,“ Vikipeedia, [Võrgumaterjal]. Available: https://et.wikipedia.org/wiki/DevOps. [Kasutatud 22 04 2020].

[9] „Agile Alliance,“ [Võrgumaterjal]. Available: https://www.agilealliance.org/glossary/definition-of-done/. [Kasutatud 2020 03 15].

[10] „LeSS,“ The LeSS Company B.V., [Võrgumaterjal]. Available: www.less.works. [Kasutatud 29 03 2020].

[11] „Meta-Modeling and the OMG Meta Object Facility (MOF),“ Object Management Group, March 2017. [Võrgumaterjal]. Available: www.omg.org/ocup-2/documents/Meta-ModelingAndtheMOF.pdf. [Kasutatud 24 01 2020].

[12] K. Bittner, D. West ja P. Kong, The Nexus Framework for Scaling Scrum: Continuously Delivering an Integrated Product with Multiple Scrum Teams, Addison-Wesley Professional, 2017.

[13] D. Tegarden, B. H. Wixon ja A. Dennis, Systems Analysis and Design with UML, 4th Edition, Wiley, 2012.

[14] S. Denning, „What Does It Mean To Scale Agile?,“ Forbes, 15 Aprill 2016. [Võrgumaterjal]. Available: https://www.forbes.com/sites/stevedenning/2016/04/15/what-does-it-mean-to-scale-agile/#40a776e878b9. [Kasutatud 29 01 2020].

[15] B. Meyer, Agile! The Good, the Hype and the Ugly, Springer, 2014.

[16] „Scaled Agile,“ [Võrgumaterjal]. Available: https://www.agilest.org/scaled-agile/. [Kasutatud 25 11 2019].

[17] V. Basili, L. Briand, D. Bianculli, S. Nejati, F. Pastore ja M. Sabetzadeh, „Software engineering research and industry: a symbiotic relationship to Foster impact,“ IEEE Software, 35(5), 44-49, 2018.

[18] Y. Harab, C. Noteboom ja S. Sarnikar, „Evaluating Project Characteristics for Selecting the Best-fit Agile Software Development Methodology: A Teaching Case," *Journal of the Midwest Association for information Systems, 1(1),* pp. 33-52, 2015.

[19] V. B. Silva, F. Schramm ja A. C. Damasceno, „A multicriteria approach for selection of agile methodologies in software development projects," *IEEE International Conference on Systems, Man, and Cybernetics*, Budapest, Hungary, 2016.

[20] A. Sharma ja R. K. Bawa, „A multilevel hybrid approach for selection of agile development method using AHP, promethee and fuzzy logic," Punjab, India, 2017.

[21] T. Veskioja, „Täppismeetodid otsustuste vastuvõtmisel materjalid," [Võrgumaterjal]. [Kasutatud 03 02 2020].

[22] Y. Lin, V. Hilaire, N. Gaud ja A. Koukam, „A. Scrum conceptualization using K-CRIO ontology," *International Symposium on Data-Driven Process Discovery and Analysis,* Kd-d. Springer, Berlin, Heidelberg, pp. 189-211, ResearchGate, 2011.

[23] E. Eessaar, „Süsteemiarendus ja andmebaasi disaini koht selles.," [Võrgumaterjal]. Available: https://maurus.ttu.ee/download.php?aine=346&document=32454&tyyp=do. [Kasutatud 25 11 2019].

[24] H. Ayed, B. Vanderose ja N. Habra, „A metamodel-based approach for customizing and assessing agile methods.," *Eighth International Conference on the Quality of Information and Communications Technology.*, Lisbon, Portugal, IEEE, 2012.

[25] A. R. Hevner, S. T. March, J. Park ja S. Ram, „Design science in information systems research," *MIS Quart. 28,* pp. 75-105, 2004.

[26] J. Giles, The Nimble Elephant: Agile Delivery of Data Models using a Pattern-based Approach, Technics Publications, 2012.

[27] „Meta-," Vikipeedia, [Võrgumaterjal]. Available: https://et.wikipedia.org/wiki/Meta-. [Kasutatud 25 04 2020].

[28] M. Fowler, Analysis Patterns: Reusable Object Models, Clean Earth Books, 1996.

[29] E. Eessaar ja E. Käosaar, „Model Smells.," Github, [Võrgumaterjal]. Available: https://github.com/erki77/model-smells. [Kasutatud 25 04 2020].

[30] E. Eessaar, „Kontseptuaalne andmemudel. Andmebaasid I loengumaterjalid," [Võrgumaterjal]. Available: https://maurus.ttu.ee/download.php?aine=378&document=35581&tyyp=do. [Kasutatud 26 02 2020].

[31] E. Eessaar, „Kontseptuaalse andmemudeli/valdkonnamudeli antimustreid. Andmebaasid I loengumaterjalid," [Võrgumaterjal]. Available: https://maurus.ttu.ee/download.php?aine=346&document=33236&tyyp=do. [Kasutatud 25 11 2019].

[32] „Enterprise Architect," Sparx Systems, [Võrgumaterjal]. Available: www.sparxsystems.com/. [Kasutatud 11 02 2020].

[33] „Domain-specific modeling,“ Wikipedia, [Võrgumaterjal]. Available: https://en.wikipedia.org/wiki/Domain-specific_modeling. [Kasutatud 25 04 2020].

[34] „Web Ontology Language,“ W3, [Võrgumaterjal]. Available: https://www.w3.org/TR/owl-features/. [Kasutatud 25 04 2020].

[35] M. M. Mkhinini, O. Labbani-Narsis ja C. Nicolle, „Combining UML and ontology: An exploratory survey,“ univ. Bourgogne Franche-Comté, Dijon, France, 2020.

[36] L. Võhandu, Subjektiivsetest hinnangutest objektiivsete tulemusteni : loengukonspekt, Tallinn: Tallinna Tehnikaülikool, 1998.

[37] „Web-Hipre,“ [Võrgumaterjal]. Available: http://hipre.aalto.fi/. [Kasutatud 24 01 2020].

[38] M. K. Alqudah ja R. Razali, „Key factors for selecting an Agile method: A systematic literature review,“ *International Journal on Advanced Science, Engineering and Information Technology, 7(2),* pp. 526-537, 2015.

[39] D. Leffingwell, Agile Software Requirements: Lean Requirements Practices for Teams, Programs, and the Enterprise, Addison-Wesley Professional, 2010.

[40] „SAFe. SAFe Requirements Model,“ Scaled Agile, 2 10 2018. [Võrgumaterjal]. Available: https://www.scaledagileframework.com/safe-requirements-model/. [Kasutatud 06 02 2020].

[41] M. Zukova, „Enterprise Agile raamistiku valik Microsoft Dynamics toodete juurutamisel,“ Magistritöö. Tallinna Tehnikaülikool, Tallinn, 2016.

[42] „Visual Paradigm,“ Feature Team vs Component Team in Agile, [Võrgumaterjal]. Available: https://www.visual-paradigm.com/scrum/feature-team-vs-component-team-in-agile/. [Kasutatud 11 04 2020].

[43] J. Cabot, „What do their creators think about UML now?,“ Modeling Languages, 05 02 2018. [Võrgumaterjal]. Available: https://modeling-languages.com/uml-opinions-creators/. [Kasutatud 25 04 2020].

[44] „Unified Modeling Language,“ Object Management Group, 2017. [Võrgumaterjal]. Available: https://www.omg.org/spec/UML/About-UML/. [Kasutatud 25 04 2020].

[45] O. I. Lindland, G. Sindre ja A. Solvberg, „Understanding Quality in Conceptual Modeling,“ IEEE, University of Trondheim, 1994.

[46] „Eesti Keele Instituut. English-Estonian MT dictionary,“ [Võrgumaterjal]. Available: http://portaal.eki.ee/dict/ies/. [Kasutatud 2020].

[47] „IT terministandardi sõnastik,“ Eesti Keele Instituut, [Võrgumaterjal]. Available: http://eki.ee/dict/its/. [Kasutatud 12 02 2020].

[48] J. B. Warmer ja A. G. Kleppe, The object constraint language: getting your models ready for MDA, Addison-Wesley Professional, 2003.

[49] „13th State of Agile Report Released,“ InfoQ, [Võrgumaterjal]. Available: https://www.infoq.com/news/2019/05/13th-state-agile-report/. [Kasutatud 22 04 2020].

[50] M. LeMay, Implementing and Scaling Agile in the Enterprise, O'Reilly Media, Inc., 2017.

[51] K. Bittner, „Scaling Scrum with Nexus and Scrum Studio,“ Scrum.org, 17 01 2018. [Võrgumaterjal]. Available: https://www.scrum.org/resources/blog/scaling-scrum-nexus-and-scrum-studio. [Kasutatud 02 04 2020].

[52] C. Larman ja B. Vodde, Large-Scale Scrum: More with LeSS, Addison-Wesley Professional, 2016.

[53] M. Alqudah ja R. Rozilawati, „A review of scaling agile methods in large software development,“ *International Journal on Advanced Science, Engineering and Information Technology 6.6,* pp. 828-837, 2017.

[54] J. Paquet, „My Agile Partner,“ 31 05 2019. [Võrgumaterjal]. Available: https://www.myagilepartner.com/blog/index.php/2019/05/31/safe-vs-spotify/. [Kasutatud 04 02 2020].

[55] „Google Trends,“ [Võrgumaterjal]. Available: https://trends.google.com/trends/explore?q=Scaled%20Agile%20Framework,Disciplined%20Agile%20Delivery,Large%20Scale%20Scrum,Scrum%20Nexus. [Kasutatud 01 05 2020].

[56] T. Dingsøyr, N. B. Moe ja H. Holmström Ohlsson, „Towards an Understanding of Scaling Frameworks and Business Agility,“ A Summary of the 6th International Workshop at XP2018, Porto, Portugal, 2018.

[57] D. Leffingwell ja R. Knaster, SAFe 4.5 Distilled: Applying the Scaled Agile Framework® for Lean Enterprises, Second Edition, Addison-Wesley Professional, 2018.

[58] „SAFe,“ Scaled Agile, [Võrgumaterjal]. Available: www.scaledagileframework.com/. [Kasutatud 07 02 2020].

[59] D. Leffingwell, R. Knaster, I. Oren ja D. Jemilo, SAFe 4.5 Reference Guide: Scaled Agile Framework for Lean Enterprises, Second edition, Addison-Wesley Professional, 2018.

[60] „Scaling Scrum,“ Scrum.org, [Võrgumaterjal]. Available: https://www.scrum.org/resources/scaling-scrum. [Kasutatud 21 11 2019].

[61] C. Larman ja B. Vodde, Practises for Scaling Lean & Agile Development: Large, Multisite, and Offshore Product Development with Large-Scale Scrum, New Jersey, United States: Pearson Education (US), 2010.

[62] C. Larman ja B. Vodde, Scaling Lean & Agile Development: Thinking and Organizational Tools for Large-Scale Scrum, Addison-Wesley Professional, 2008.

[63] „Agiilne arendus,“ [Võrgumaterjal]. Available: https://agiil.github.io/IT/Agiil. [Kasutatud 02 04 2020].

[64] A. Vaidya, „Does DAD know best, is it better to do less or just be safe? Adapting scaling agile practices into the enterprise.,“ *PNSQC,* pp. 1-18, 2014.

[65] „Disciplined Agile Delivery,“ [Võrgumaterjal]. Available: https://disciplinedagiledelivery.com/. [Kasutatud 21 11 2019].

[66] M. Lines ja S. W. Ambler, Disciplined Agile Delivery: A Practitioner’s Guide to Agile Software Delivery in the Enterprise, IBM Press, 2012.

[67] A. L. Opdahl ja B. Henderson-Sellers, „Ontological Evaluation of the UML Using the Bunge-Wand-Weber Model,“ Springer-Verlag, Bergen, Norway, 2002.

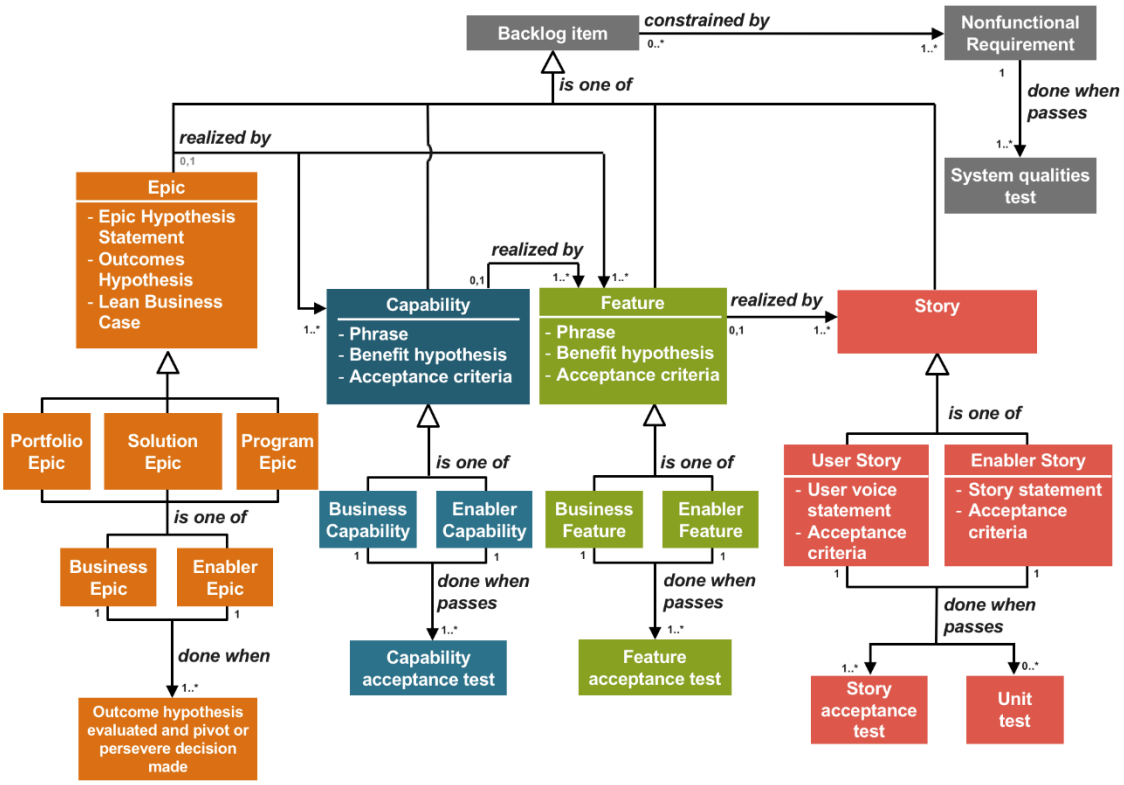
[68] J. Eckstein, „Architecture in Large Scale Agile Development,“ International Conference on Agile Software Development, Germany, 2014.

[69] D. Lynn, „LeSS and DevOps,“ Agile42, 2017. [Võrgumaterjal]. Available: https://www.agile42.com/en/blog/2017/09/22/less-and-devops/. [Kasutatud 03 04 2020].

[70] „All models are wrong,“ Wikipedia, [Võrgumaterjal]. Available: https://en.wikipedia.org/wiki/All_models_are_wrong. [Kasutatud 25 04 2020].

[71] A. Saltelli ja S. Funtowicz, „When All Models Are Wrong,“ University of Bergen, Norra, Bergen, 2014.

Lisa 1 – SAFe nõuete metamudel



Joonis 58. SAFe nõuete metamudel [40].

Joonis 58 esitab SAFe nõuete (*Backlog Item*) metamudeli, millelt on näha, et SAFe järgi jagatakse nõuded nelja gruppi – eepos (*Epic*), võimekus (*Capability*), erisus (*Feature*) ja kasutuslugu (*Story*). Eepos on kõige suurem nõue, mida käsitletakse portfelli, lahenduse või programmi tasemel. Iga nõue võib olla äriline (*Business*) või eeldus (*Enabler*) tüüpi. Äriliste nõuete realiseerimine annab äriväärtust, eeldus tüüpi nõuete realiseerimine on vajalik, et saaks realiseerida ärilisi nõudeid. Need on äriliste tööde tehnoloogilised ja arhitektuurilised eeldustööd. Eepos on tükeldatud võimekusteks, võimekus on tükeldatud erisusteks ja erisus on tükeldatud kasutuslugudeks.