TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond
Informaatika instituut
Infosüsteemide õppetool

# Tarkvara arendusprotsessi parendamine Tieto Estonia AS projekti näitel

magistritöö

Üliõpilane: Vahur Kaar
Üliõpilaskood: 143673IAPM
Juhendaja: Mart Roost

Tallinn
2016

# Autorideklaratsioon

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

---------- ----------

(*kuupäev*) (*allkiri*)

# Annotatsioon

Käesolev magistritöö kirjeldab tarkvaraarendusprotsessi parendamise protsessi Tieto Estonia AS ühe projekti näitel. Töö eesmärgiks on välja töötada käsitluse all oleva projekti jaoks sobilik arendusprotsess, mis suurendab arendamise kiirust, maandab riske, parandab meeskonnasuhtlust ning suurendab projekti kasumlikkust. Algselt antakse ülevaade tegevusliku disainuuringu metoodikast, mida kasutatakse projektis paranduste läbiviimisel. Töös on kirjeldatud ja analüüsitud olemasolev arendusprotsess ja selle puudujäägid. Samuti on ära dokumenteeritud ka paranduste läbiviimine projektis. Töö viimases osas võrreldakse uut arendusprotsessilahendust esialgsega.

Töö tulemusena valmib mudel ja veebisait (protsessiveeb, mis publitseerib mudeli erinevatele osapooltele ja huvilistele vajalikes vaadetes), mida on reaalses elus võimalik kergelt rakendada. Mudel kirjeldab arendusprotsessi abstraktsel kujul ning seda on võimalik rakendada mitmetes sarnastes projektides.

Lõputöö on kirjutatud eesti keeles ning sisaldab teksti 70 leheküljel, 6 peatükki, 18 joonist ja 12 tabelit.

# Abstract

The current thesis describes the software development process improvement process of one of Tieto Estonia AS projects. The goal of this thesis is to design a better development process for the given project, which increases development velocity, reduces risks, improves team communication and increases the project's profitability. Firstly, the author gives an overview of the Action Design Research methodology, which is used to carry out improvements in the project. The existing software development process and its flaws are described and analyzed, the process of carrying out improvements to the development process are documented. Lastly, the new designed process is compared against the initial one.

As a result of this thesis, a model and a website (a process web, which is published for different parties in different views) is assembled, which are easily applicable in the real world. The model describes the software development process in an abstract level and it can be applied in many similar projects.

The thesis is in estonian and contains 70 pages of text, 6 chapters, 18 figures and 12 tables.

# Jooniste nimekiri

# Tabelid, lähtekood ja näited

# Sisukord

# 1. Sissejuhatus

Tänapäeva maailm on pidevas muutuses. Selle tagajärjel muutuvad sageli vajadused ja nõuded toodetes, teenustes ja neid arendavates protsessides. Enamasti ei ole muutused selgelt etteennustatavad, mille tulemusena on vaja proaktiivselt olukordadele kohaneda. Pidev kohanemine aga tähendab, et toimub pidev areng. Selleks, et inimene saaks rohkem kasu olukordadest, on vaja anda oma panus arendamise juhtimisele. Arendamist ei vaja üksnes müüdavad tooted, vaid ka protsessid.

Läbi aastate on rakendatud erinevaid lähenemisi ja metoodikaid tarkvaraarenduses. Igal lähenemisel on omad eelised ja puudujäägid. Esimese generatsiooni tarkvaraarenduse metoodikateks peetakse „kose“ mudeli põhiseid traditsioonilisi metoodikaid. Neid metoodikaid iseloomustab konkreetsele plaanile orienteeritus, kus protsess koosneb kindlas järjekorras täidetavatest etappidest, mis on selgelt tuvastatavad, ettenähtavad ja korratavad. Teise generatsiooni metoodikaid iseloomustab iteratiivne lähenemine. Iteratiivse lähenemise puhul pööratakse tähelepanu suure projekti tükeldamisele väiksemateks täidetavateks iteratsioonideks, mille käigus on võimalik töö tellijale toodet esitleda, varajast tagasisidet saada ja sellest lähtuvalt muudatusi läbi viia. [1]

Aastakümnete jooksul on välja kujunenud palju erinevaid metoodikaid ja raamistikke tarkvaraarendusprotsesside kujundamiseks. Kuigi tänapäeval on agiilsed ja iteratiivsed metoodikad kujunenud suureks ja läbilöövaks trendiks, on iga olukord eriline ja omanäoline. Praktikas selgub sageli, et ühte konkreetset metoodikat pole mõttekas puhtal kujul kasutada, kuna organisatsioonis esinevad teatud isikupärased jooned, mis lähevad vastuollu teoorias pakutuga. Sellistes olukordades tavaliselt võetakse elemente erinevatest raamistikest ja luuakse spetsiifiline protsess organisatsiooni jaoks. Protsessi väljakujundamisel ja selle arendamisel peab võtma arvesse just eristuvaid isikupäraseid omadusi ja kasutama elemente erinevatest metoodikatest, sest ainult siis on võimalik saavutada efektiivsem protsess.

## 1.1 Probleemi ja tausta kirjeldus

Ülesande püstitus pärineb ettevõttes Tieto Estonia AS töötamisest. Autor määrati uude projekti, mis on ettevõtte hoolduses olnud viimased 13 aastat. Projektiga ühinemise hetkel on meeskond väike ning projekti töökorralduses esineb palju probleeme:

- tarkvara tarnimisel esineb väga tihti vigu;
- omavaheline suhtlemine on ebaefektiivne ja raskendatud töökorraldusest tulenevalt;
- uute tellimustööde arendamine on väga aeganõudev;
- uute tellimustööde arendamine sisaldab kõrget riski.

Koos autoriga ühineb projektiga veel teisigi töötajaid ning senine töökorraldus ei saa enam jätkuda. Vastasel juhul kahaneb projekti kasumlikkus vastuvõetamatult väikeseks.

## 1.2 Ülesande püstitus

Käesoleva töö peamiseks eesmärgiks on parandada käsitluse all oleva projekti arendusprotsessi. Paranduste tulemusena peavad olema täidetud järgmised kriteeriumid:

- parandatud arendamise kiirus;
- maandatud riskid;
- efektiivsem meeskonnasuhtlus;
- paranenud projekti kasumlikkus.

Lõputöö täiendavateks eesmärkideks on luua kergesti loetav ja jälgitav mudel parandatud tarkvaraarendusprotsessi kohta ja publitseerida veebisait selle põhjal. Mudel peab kirjeldama tarkvaraarendusprotsessi abstraktsemal tasemel, mis sobiks ka teistele sarnase organisatsioonilise ja töökorraldusliku struktuuriga projektidele. Veebisait peab kujutama endast protsessiveebi, mis publitseerib mudeli erinevatele osapooltele ja huvilistele vajalikes vaadetes, muutes protsessi rakendamise mudeli lugejatele võimalikult kergeks.

## 1.3 Metoodika

Lõputöös püstitatud eesmärkideni jõutakse läbi pideva koostöö projekti erinevate osapooltega. Tarkvaraarendusprotsessi parendamisele lähenetakse agiilsel viisil ning selle läbiviimisel kasutatakse tegevusliku disainuuringu metoodikat. Lähtutakse agiilse tarkvaraarenduse parimatest praktikatest. Töö tulemusena valmib dokumentatsioon paranduste läbiviimisest projektis. Dokumentatsioon koosneb üksikute iteratsioonide kokkuvõtetest. Parandatud tarkvaraarendusprotsess modelleeritakse SPEM 2.0 raamistiku alusel.

## 1.4 Ülevaade tööst

Teises peatükis käsitletakse arendusprotsessi parandamise läbiviimisel kasutatavaid võtteid ja metoodikaid. Antakse ülevaate tegevusliku disainuuringu teoreetiliste aluste kohta. Kirjeldatakse kuidas see lähenemine on moodustunud juba varasemalt eksisteerinud metoodikate baasil. Samuti käsitletakse varasemalt praktikas läbiviidud uuringuid.

Kolmas peatükk sisaldab Tieto Estonia AS projekti esialgse arendusprotsessi ülevaadet ja analüüsi. Kirjeldatakse olukorda projektiga ühinemisel ning koostatakse SPEM 2.0 raamistikul baseeruv mudel lähteolukorrast. Teostatakse analüüs lähteolukorra kohta ning tuuakse välja protsessi tugevused, mida on vaja säilitada ning nõrkused, mis peab likvideerima.

Neljandas peatükis dokumenteeritakse arendusprotsessi paranduste läbiviimine. Dokumentatsiooni vormistamisel on lähtutud tegevusliku disainuuringu metoodikast. Arendusprotsessi arendamine on viidud läbi iteratsioonide kujul. Parandused viidi läbi kolmes eristuvad iteratsioonis.

Viies peatükk esitab mudeli kujul paranduste tulemusena kujunenud protsessi. Sarnaselt kolmandale peatükile, kasutatakse SPEM 2.0 raamistikku mudeli koostamisel. Lõpuks analüüsitakse paranduste tulemusi ning võrreldakse uut lahendust lähteolukorraga.

# 2 Tegevusliku disainuuringu metoodikate ja tarkvara protsesside parendamise alused

## 2.1 Disainuuringu metoodika

Disainuuring on uurimismeetod, mille eesmärgiks on formaliseerida ja valideerida disainimise mudeleid ja teooriaid. Samuti puudutatakse teadmiste, meetodite ning vahendite arendamise ja valideerimise protsesse. Ülimaks eesmärgiks on parandada disainimise protsessi. [2]

Disainuuringuga seostatakse ka disainkatsetusi, mille olemuseks on erinevate disainide proovimine probleemi lahendamise eesmärgil. Disainkatsetused kujunesid välja hariduskavade ümberdisainimise käigus. Nende abil sooviti viia läbi kujundavat uurimistööd hariduskavade ümberdisainimisel, et paremini testida ja täiustada muudatusi, mis põhinesid varasemal uurimusel. Disainuuringud pöörasid tähelepanu erinevatele probleemidele, mis teadustöö käigus ilmnesid. Olulisemateks kujunesid järgnevad:

- pöörata tähelepanu teoreetilistele küsimustele õppimise olemuse kohta;
- uurida õppimise fenomeni reaalse maailma kontekstis, mitte üksnes piiritletud ja kontrollitud laboratooriumis;
- uurida õppimise olemuse kitsamaid suundasid;
- kujundava hinnangu baasil uute teaduslike avastuste tuletamine. [3]

### 2.1.1 Disainuuringu tunnusjooned

Järgnevalt on välja toodud selgemini eristuvad tunnusjooned, mis iseloomustavad disainuuringu metoodikat:

1. Korrapäratud olukorrad – uuringud ja eksperimendid viiakse läbi osaliselt piiritlemata (ja vahel isegi kaootilistes) keskkondades, mis emuleerib reaalselt maailma. Keskkonna liigne piiritlemine võib anda väärtulemusi.
2. Palju teineteisest sõltuvaid muutujaid – eksperimenti mõjutavad mitmed erinevad muutujaid ja mõjutegureid, mis mõjutavad ka teineteist. Uurimuse läbiviijad ei pruugi kõiki tegureid isegi eriliselt jälgida.

3. Olukorra iseloomustamine – jälgitavaid muutujaid ei üritata eriliselt kontrollida, vaid pigem toimub üksnes olukorra kirjeldamine ja iseloomustamine. Üritatakse avastada kõik tegurid, mis mõjutavad eksperimendi tulemit kas otsesel või kaudsel kujul.

4. Muutlikud võtted – eksperimentide algfaasides pannakse paika esialgsed planeeritavad võtted ja meetodid, mis hakatakse rakendama. Võtted ei ole täielikult ära defineeritud, vaid on jäetud võimalus teha muudatusi eksperimendi käigus. Sõltuvalt edufaktorist viiakse läbi täiendusi ja parandusi.

5. Sotsiaalne koosmõju – uuritakse subjektide omavahelist kokkupuudet ja mõju teineteisele, selle asemel, et nad teineteisest eraldada ja isolatsiooni panna.

6. Profiili arendamine – eesmärgiks on jälgida disaini erinevaid külgi ja luua kvalitatiivne ning kvantitatiivne profiil, mis iseloomustab disaini praktilise külje pealt.

7. Osapoolte kaasamine disainis – disaini arendamisel üritatakse kaasata erinevaid osapooli ja rakendada nende ekspertteadmisi disaini kujundamisel ja analüüsimisel.

Disainuuring soosib ettekirjutatud disain-teadmiste moodustamist ja loomist, mis saavutatakse pideva innovaatilise tegevuse ja hindamiste läbiviimise kaudu. Uuritavad ja arendatavad subjektid on tavaliselt seotud ühe kindla tüüpprobleemiga, mis on abstraktsemal tasemel ära kirjeldatud. Disainuuringu tulemused ei hõlma endas üksnes innovaatilist artefakti, vaid ka teadmisi, mis võimaldavad luua teisi sarnaseid artefakte, mis kuuluvad samasse kategooriasse.

Tavaliselt eelneb artefakti arendamisele ja hindamisele probleemi teadvustamine ja defineerimine, mille alusel toimub edasine tegevus. Filosoofilises mõttes pööratakse rohkem tähelepanu kasumlikkusele, mis saavutatakse uuringu alguses püstitatud probleemi lahendamisega.

Suhteliselt vähesed uurimused on suutnud balansseeritult esitada üldist probleemi tüüpi ja rakendada praktikas muudatusi, mis toetavad defineeritud üldistusi. Disainuuringu meetodid pakuvad erinevaid lähenemisi, mis seovad omavahel probleemid, eesmärgid, kontseptide formaliseerimise, disaini ja tulemuse hindamise. Selline tegevuste järjestus, mis eraldab artefakti ehitamise ja hindamise faasid teineteisest, ei paku uurimise meetodit, mis sisaldaks endas pidevat innovaatilist tsüklilist arendust. [3]

### 2.1.2 Disainuuringu rakendamise praktikas

Näitena käsitleme Ann Brown'i ja Joseph Campione'i programmi „Õppijate kogukonna kujunemise soodustamine" („Fostering a Community of Learners"), mis viidi läbi erinevates USA algkoolides. Uurimust viiakse läbi erinevates klassiastmetes samaaegselt ning uuritakse õpilaste õpetamise ja õppimise viise.

Uuringu esimene faas viidi läbi labori keskkonnas üksikute lastega. Esimesed eksperimendid tehti vastastikuse õppimise teooriaga. Õpilastele anti ühe kindla ainega seostuv tekst, mille lapsed pidid läbi lugema. Peale seda pidid lapsed täitma neli ülesannet:

- Küsida küsimus teksti kohta
- Anda ülevaade, mis jäi ebaselgeks
- Teha lühikokkuvõte loetu põhjal
- Teha oletused järgmise teksti kohta, mis neile antakse

Labori tulemused olid väga positiivsed ning uuringu skoopi otsustati järgmises etapis suurendada. Teises etapis hakati uurima reaalseid olukordi kooli klassiruumides. Õpilastele anti korda-mööda õpetaja roll, mille käigus pidi teiste õpilaste käest üleval mainitud küsimused esitama. Tulemused olid jällegi väga positiivsed, kuid vestlused olid väga pinnapealsed. Nendest tulemustest kujunes välja õppijate kogukonna moodustamise idee, mida järgmistes etappides hakati rakendama.

Järgmises etapis muudeti eksperimenti selliselt, et iga kursuse alguses toimus suunitleva mõttega tund, mille käigus esitas õpetaja või koolivälise külaline õpitava temaatika ja skoobi. Samal ajal lisati programmi ka laboripõhised tegevused, kus õpilased said eksperimente läbi viia antud temaatikaga seoses. Lisades programmi ka koolivälised eksperdid, suurenes koheselt programmi skoop. Praktika kogukond suurenes, õpitav materjal ei piirdunud ainult koolis pakutavaga. Selles faasis hakkas kujunema arusaam, kus õpilased käsitlevad igal aastal aina sügavamaid aspekte ainest. Seda arusaama hakati järgmises etapis rakendama.

Viimase etapi jaoks valiti välja uus kool, kus hakati rakendama teise kuni kaheksanda klassi õpilaste seas uut parandatud programmi, kus õpilased peaksid järk-järgult minema aines aina sügavamale. Etapi käigus õpilased pakkusid välja, et klassiastmete vahelised diskussioonid oleksid väga kasulikud. See toetas suurel määral kogukonna arengut, mis ühendas kõik vajalikud tükid õppijate kogukonna moodustamisel. [3]

Näitena käsitletud „Õppijate kogukonna kujunemise soodustamise“ programm vastab mitmetele üleval pool mainitud tunnusjoontele. Eksperimendid viidi läbi reaalsetes koolides ja klassiruumides, mille käigus ei piirdutud isegi üheklassiastmega. Projekti käigus skoop laienes ja projekti osapooled said ise anda oma panuse disaini kujundamisse.

## 2.2 Tegevusliku uuringu metoodika

Tegevuslik uuring on osalusdemokraatial põhinev protsess, mille eesmärgiks on arendada inimese sihtide saavutamisele suunatud praktilisi teadmisi. Protsess baseerub osalusdemokraatia põhimõtetel, mis on tänapäeval levinud. Tegevusliku uuringu käigus üritatakse viia kokku tegutsemine ja mõtlemine, teooria ja praktika. Püüeldakse praktiliste lahenduste poole, mis on inimeste jaoks suure tähtsusega ning üleüldise üksikisikute ja kogukondade õitsengu poole. [5]

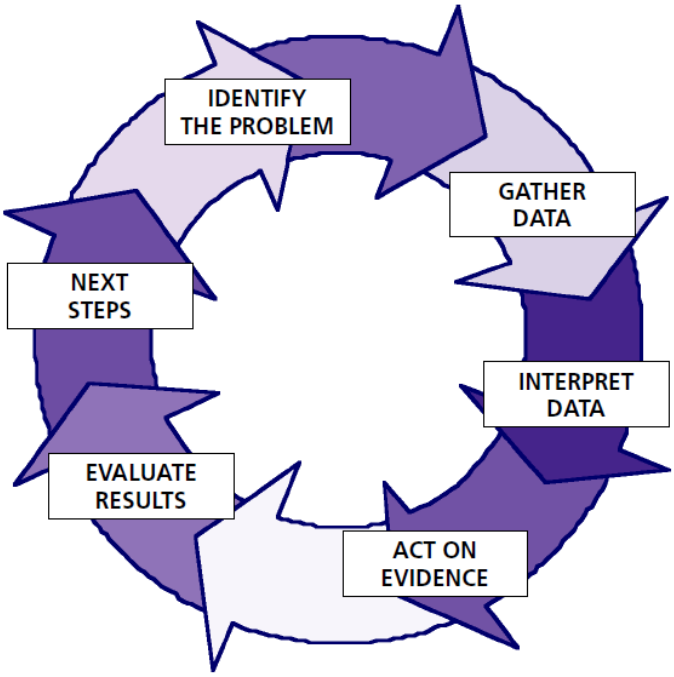
Tegu on protsessiga, milles osalejad pidevalt jälgivad oma kasutatavaid praktikaid süstemaatiliselt ja hoolikalt. Metoodika baseerub järgnevatel eeldustel:

- Uuringus osalejad on kõige efektiivsemad, kui nad tegelevad oma valdkonna probleemidega.
- Osalejad muutuvad efektiivsemaks, kui julgustada neid kriitiliselt hindama omaenda tööd ja teostama muudatusi.
- Osalejad abistavad teineteist tehes pidevat koostööd.
- Osalejate professionaalne areng on oluliselt parem, kui tehakse koostööd. [6]

Tegevuslik uuring kombineerib teooria loomise uurija vahelesekkumisega selleks, et paremini lahendada otseseid organisatsioonilisi probleeme. Tavaliselt on tegu iteratiivse protsessiga, mis baseerub toimivatel hüpoteesidel, mida arendatakse korduvate tsüklite käigus pidevalt edasi. [4]

Tegevuslik uuring ei haaku tavapärase teadustööga. Tegu pole võttega, mille käigus üritatakse lahendada ühte konkreetset probleemi, pigem keskendutakse hoopis enesearengu edendamisele. Tähelepanu pööratakse rohkem inimeste oskuste, tehnika ja strateegiate arendamisele. [6]

### 2.2.1 Tegevusliku uuringu protsess




**Joonis 2.1. Tegevusliku uuringu protsess**

Tegevusliku uuringu protsess hõlmab endas nelja suuremat temaatikat: osalejate mõjuvõim, koostöö läbi aktiivse kaasalöömise, teadmiste omandamine ja sotsiaalsed muutused. Nimetatud temaatikad on kaetud järgneva viie faasiga (protsessi illustreerib ka Joonis 1. [6]):

1. Probleemi valdkonna tuvastamine

   Oluline on piirata püstitavate küsimuste arvu ja valida lähtuvalt töökoormusest ainult reaalselt lahendatavad probleemid. Kui esimeses faasis planeerida hoolikalt, on võimalik palju probleeme ennetada. Püstitatavad küsimused peavad olema kõrgematasemelised (mitte jah/ei-tüüpi küsimused), tähendusrikkad, sisukad ja ilma valmislahenduseta.

2. Andmete kogumine ja korrastamine

   Kogutud andmetest sõltub, milliseid tegevusi hakatakse hiljem rakendama. Selleks, et saada parem ülevaade olukorrast, on parem koguda andmeid erinevatest allikatest. Vaja on koguda üksnes püstitatud probleemide lahendamiseks vajalikke andmeid. Andmed peab organiseerima selliselt, et need oleksid kasulikud iseloomujoonte ja trendide tuvastamisel.

3. Andmete tõlgendamine

   Andmed võivad olla erinevat tüüpi, seega võib tõlgendamine toimuda ka erinevalt. Näiteks tabelite, graafikute näol.

4. Rakendatud tegevustel põhinevate andmete kogumine

   Kogutud andmete ja teooria põhjal on vaja koostada tegevusplaan, mis toetab püstitatud eesmärke ja võimaldab jälgida ning uurida rakendatavaid muudatusi. Selleks, et muudatuste mõju paremini jälgida ja analüüsida, on kasulikum mõjutada ainult ühte tegurit korraga.

5. Tagasipeegeldamine

   Hinnatakse rakendatud muudatusi. Kui esines edusamme, siis on vaja kontrollida andmete korrektsust, kui olukord läks aga halvemaks, on vaja planeerida järgmised tegevused, mis annaksid vastupidise tulemuse.

6. Järgmine iteratsioon

   Toimub uute küsimuste püstitamine. Suundutakse tagasi 1. faasi. [6]

### 2.2.2 Tegevusliku uuringu rakendamine praktikas

Aastal 1991 algatati uuring, mille käigus uurida, kuidas algatada loodusturismi Windwardi saartel (St. Lucia, Grenada, Dominica ja St. Vincent), mis asuvad Kariibi meres. Kohalik valitsus üritas toetada kohaliku ühiskonna ja riikliku majanduse arengut. Uuringu formaadiks otsustati valida tegevusliku uuringu metoodika, kuna oli vaja konsulteerida väga erinevate osapooltega, nagu ministeeriumite esindajad, keskkonnakaitse grupid, ühiskonna organisatsioonid, noorte organisatsioonid, farmerite ühingud ja eraettevõtted.

Projekti viisid läbi kaks Yorki Ülikooli teadlast, kes olid varasemalt tegutsenud selles regioonis. Moodustati riiklike huvigruppide komisjonid. Esimeseks ülesandeks oli organiseerida konverents, kus püstitada ideid ja luua plaanid, mida läbi viia. Konverentsid pidid toimuma igal saarel. Läbiviidud konverentside tulemusena püstitati soovitused ja tegevusplaanid, viimaks läbi mitmeid väiksemaid alamprojekte erinevates regioonides. Selles faasis loodi täiendavad nõuandjate grupid.

Igas regioonis viidi läbi korduvalt projekti koosolekuid, millest võtsid osa mõjukamad koordinaatorid ja nõunikud. Jagati kogemusi, hinnati oma tööd ning arendati plaane edasi. Eesmärgiks oli paremini hallata tervet protsessi.

Projekti tulemused olid erinevates regioonides erinevad. St. Vincenti saarel tõi teadustöö suuri tulemusi, kohalik turism paranes oluliselt. Grenada ja St. Lucia saartel olid vastakad tulemused ja Dominica saarel uurimus üleüldse ebaõnnestus. Peamiseks probleemiks kujunes valitsuse juhtide hirm kaotada oma mõju regioonis. Kuna projekt püüdles selle poole, et kõik seotud osapooled võtavad aktiivselt juhtimise protsessist osa, siis sellega kaasneb ka kohaliku valitsuse võimu vähenemine. Tugeva koostöö tulemusena võib saavutada palju rohkem. [7]

## 2.3 Metoodikate ühendamine

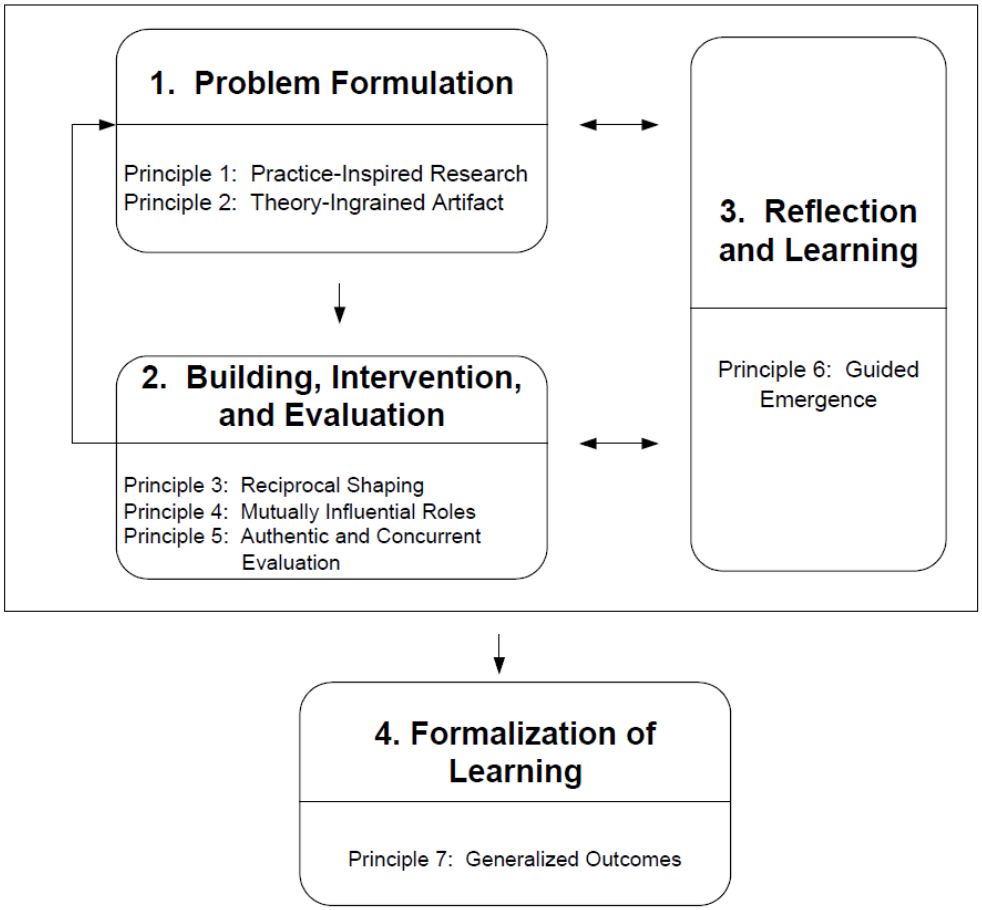
Kahes eelnevas peatükis on antud lühike ülevaade disainuuringust ja tegevuslikust uuringust eraldi koos paari reaalelulise näitega nende rakendamisest. M. Sein, O. Henfridsson, S. Purao, M. Rossi ja R. Lindgren on välja pakkunud uut liiki disainuuringu lähenemise, mis ühendab mõlema ülalpool nimetatud metoodikad omavahel. Väljapakutud meetodiga saab teostada disainuuringut, mis arvestab ka asjaoluga, et uuritav artefakt puutub kokku ja on mõjutatud uuringu praktilisest korralduslikust poolest, olgugi, et esialgne disain allub ainult uurija tahtele.

Tegevuslik disainuuring on metoodika, mille eesmärgiks on luua ettekirjutatud disainialased teadmised läbi pideva arendamise ja hindamise, mis viiakse läbi organisatsioonilises keskkonnas. Käsitletakse kahte suuremat probleemi:

- Organisatsioonilises keskkonnas ilmneva probleemi määratlemine läbi pideva hindamise ja sekkumisega;
- Üldisemat tüüpi probleemi kirjeldava artefakti loomine ja hindamine.

Nimetatud probleemide käsitlemisel saadaksegi protsess, mille fookuseks on probleemi kirjeldava artefakti arendamine, vahelesekkumine ja hindamine mitte ainult teoreetilise poole pealt, vaid ka artefakti kasutamise ja lõppkasutajate mõju jälgimine ja hindamine. [4]

### 2.3.1 Tegevusliku disainuuringu protsess




**Joonis 2.2. Tegevusliku disainuuringu protsess ja printsiibid**

Järgnevalt on ära kirjeldatud tegevusliku disainuuringu protsessi etapid ja rakendatavad printsiibid (protsessi illustreerib ka Joonis 2. [4]):

1. Probleemi formaliseerimine

   Esimese etapi käigus panevad uurijad paika uuritava probleemi. Probleemi määratlemisel võivad kaasa aidata lõppkasutajad, uurijad, praktiseerijad, olemasolevad tehnikad või varasem uurimistöö. Pannakse paika projekti skoop ning erinevatele osapooltele määratakse rollid, mida nad projekti jooksul peavad täitma. Probleemi määratlemisel tuvastatakse uurimuse võimalused, mis põhinevad olemasolevatel teooriatel ja tehnikatel. Üheks oluliseks ülesandeks on ka üldise probleemi grupi määratlemine. Esimene faas põhineb kahel printsiibil:

   - **Praktikast inspireeritud uuring** – probleemide formaliseerimisel pööratakse rohkem tähelepanu praktikas eksisteerivatele olukordadele.

- **Teoorial põhinev aftefakt** – püstitatavad probleemid ja nende lahendamise käigus loodavad artefaktid peavad teooriat täiendama.

2. Arendamine, vahelesekkumine ja hindamine

   Teises faasis ehitatakse esmase disainiga artefakt vastavalt püstitatud eesmärkidele ja probleemidele. Pakutud disaini hakkab ümber kujundama ja parandama organisatsiooni poolne tegevus ja järgmised disainimise tsüklid. See faas viiakse läbi iteratiivselt, ühendades disaini arendamise, organisatsioonilise vahelesekkumise ja hindamise. Käesoleva faasi lõpptulemiks on realiseeritud artefakti disain. Faasi vältel toimub artefakti pidev hindamine ja ümberkujundamine ning rakendatavad disainimise võtted häälestatakse vastavalt probleemi tüübile. Pööratakse rõhku ka innovatsioonile.

   Faasile saab läheneda kahel erineval viisil:

   1) Tehnikatele orienteeritult – esialgsetes disainides ja iteratsioonides keskendutakse organisatsiooni piiritletud kontekstile. Hilisemates iteratsioonides rakendatakse disaini juba suuremal skaalal.

   2) Organisatsioonile orienteeritult – peamiseks innovatsiooni allikaks on pidev praktiline organisatsiooniline sekkumine. Disain rakendatakse juba varajastes staadiumites laiemal skaalal.

   Teine faas põhineb järgmistel printsiipidel:

   - **Vastastikune kujundamine** – nii tehniline kui ka organisatsiooniline kontekst mõjutavad teineteise arengut.

   - **Vastastikku mõjutavad rollid** – erinevad projekti pädevusalad õpivad koos ning mõjutavad teineteise tegevust uurimuse käigus.

   - **Ehtne paralleelselt toimiv hindamine** – hindamine ei ole omaette protsess, vaid seda teostatakse pidevalt disaini kujundamise käigus.

3. Peegeldamine ja õppimine

   Peegeldamise ja õppimise protsess toimub paralleelselt esimese kahe nimetatud etapiga. Eesmärgiks on teha üldistusi konkreetse probleemi lahendamise osas ning rakendada saadud teadmisi üldisema probleemi lahendamisel. Peegeldamine ja õppimine sarnaneb *lean* metoodikas rakendatud valideeritud õppimisele, mille käigus üritatakse korduva eksperimenteerimise abil leida viidis, kuidas jätkusuutlikku äri arendada. [8] Kolmas faas põhineb järgmisel printsiibil:

   - **Juhitud kujunemine** – pööratakse tähelepanu sellele, et arendatav artefakt toetaks nii uurija püstitatud esialgset disaini kui ka samal ajal toimuvat organisatsiooni poolt läbiviidavat ümberkujundamise protsessi.

4. Õppimise formaliseerimine

   Viimase etapi eesmärgiks on formaliseerida õppimise tulemusel saadud teadmised. Teadmised formaliseeritakse üldisemat tüüpi probleemide jaoks. Viimane faas põhineb järgmisel printsiibil:

   - **Üldistatud lõpptulemid** – tulemused viiakse spetsiifilise ja unikaalse keskkonna kontekstist üle üldisemale ja abstraktsemale tasemele.

### 2.3.2 Tegevusliku disainuuringu metoodika rakendamine praktikas

Tegevuslikku disainuuringut on rakendatud näiteks sotsiaalmeedia uuringutes. Kuna üheks osapooleks uuringu teostamisel on lõppkasutajad, siis peavad ka nemad väga aktiivselt disainimise protsessist osa võtma, kuna lõppkasutajate vajadused on vaja rahuldada. Sellega kaasneb aga väljakutse. Kuna kasutajaid on palju ja igaühel on oma soovid ja ootused, siis pole probleemide formaliseerimine sugugi kerge. Disaini skoop võib ka liiga suureks paisuda. Samas, kui põhineda rohkem teooriale, siis võib uuringu juhtimine langeda suuremas osas uurijate õlgadele, kuid vaja on kaasata kõik osapooled, et tulemused annaksid rohkem kasu. Järelikult on väga oluline pöörata tähelepanu ühisele erinevate osapoolte vahelisele õppimisele ja koostööle ning ühise disaini keskkonna piiritlemisele. [8]

Tegevuslikku disainuuringut on rakendatud ka Volvo IT-osakonnas. Uuriti ettevõtte kompetentside haldamise süsteemi. Üldiseks probleemide valdkonnaks valiti organisatsioonide kompetentsihaldus. Projekti alguses leiti, et olemasolev süsteem sisaldas vigaseid andmeid ning erinevad osapooled ei saanud andmeid kasulikult kasutada. Osapooled olid teineteisest eraldatud, peamine tähelepanu oli pööratud minevikus omandatud kompetentsidele, puudus

tuleviku prognoosimine ja juhtimisevõimalus ning aruandlus oli liiga primitiivne. Kompetentside struktuur oli töökohapõhine.

Projekti käigus kaasati uurijaid ja praktikante, et võimendada teoreetilisi, tehnilisi ja praktilisi vaateid. Peamiseks probleemiks oli andmete halb kvaliteet. Probleemidega tegeleti iteratiivsel kujul, kus igas iteratsioonis määratleti ja lahendati probleemid ära. Üleüldiseks teoreetiliseks suunaks valiti oskustel põhinev kompetentside struktuur, mida hakati realiseerima. Esimene disaini versioon oli formaliseeritud teooria kujul, kus mudel vastas üksnes oskustel põhineval struktuuril. Teine versioon lahendati prototüübi kujul. Läbi mitmete iteratsioonide laiendati pidevalt skoopi ning lisati uusi lähenemisi süsteemi. Projekti lõpus formaliseeriti kompetentside haldamise jaoks printsiibid ning esitati Volvo uus kompetentside haldamise süsteem ühe praktilise näitena. [4]

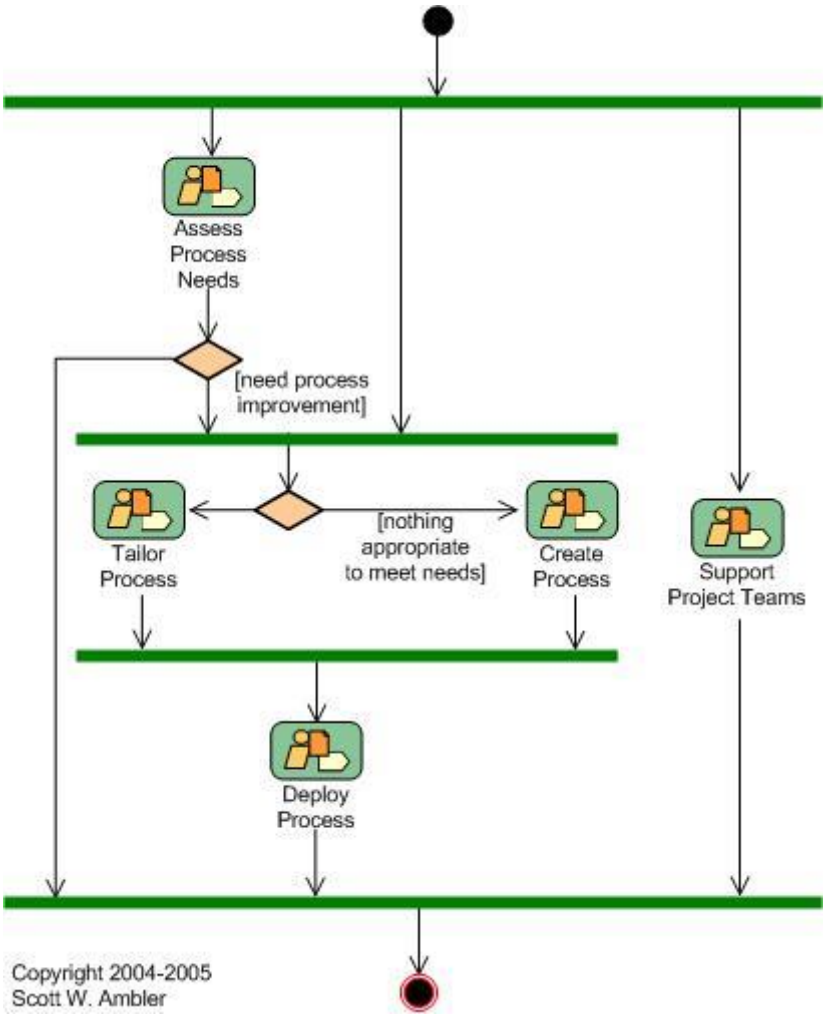
## 2.4 Tarkvara protsesside parendamine (SPI)

Käesolevas lõputöös teostatakse „Tarkvara protsesside parendamise“ distsipliin. Tarkvara protsesside parendamine (Software Process Improvement) on Enterprise Unified Process raamistiku üks distsipliinidest, mille ülesandeks on luua ja pidada ülal projektide tarkvaraarendusprotsesse. Tavaliselt agiilsetes meeskondades kujuneb protsess iseenesest, kui alustatakse projektiga ja läbitakse esimesed iteratsioonid. Protsessi arendamise korraldamise peale erilist rõhku ei panda.

Tarkvara protsesside parendamisele on võimalik läheneda kolmel erineval viisil: ülalt-alla, alt-üles ning kombineeritult. Ülalt-alla lähenemisviisi puhul valitakse välja üks levinud näidisprojekt (näiteks Scrum) ning seda hakatakse organisatsioonis rakendama, lähtudes organisatsiooni protsessinõuetest. Alt-üles lähemisviisi korral analüüsitakse organisatsiooni arendusprotsessi nõudeid ning ehitatakse uus unikaalne protsess üksnes vajaduste baasil. Kombineeritud lähenemise puhul ehitatakse arendusprotsess olemasolevate protsesside meetodifragmentidest. Meetodifragmendid valitakse erinevatest allikatest ning ühendatakse ainult organisatsiooni ärivajadustele vastavad komponendid. Kombineeritud lähenemine on praktikas kõige rohkem levinud.

EUP projektides on tarkvara protsesside parendamise näol tegu keskkonna distsipliini alamosaga, mis kirjeldab ära protsessid terves organisatsioonis ja annab juhised, kuidas läbi viia parandusi protsessides ja neid jagada meeskondade vahel. Arendusprotsessi parendamist

viib läbi protsessiinsener. Tegu on konkreetse rolliga, mida üks organisatsiooni töötaja täidab. Joonis 3 kirjeldab ära üldise protsesside haldamise protsessi, mida protsessiinsener läbi viib. [9]




**Joonis 2.3. Tarkvara protsesside parendamise distsipliini töökorraldus**

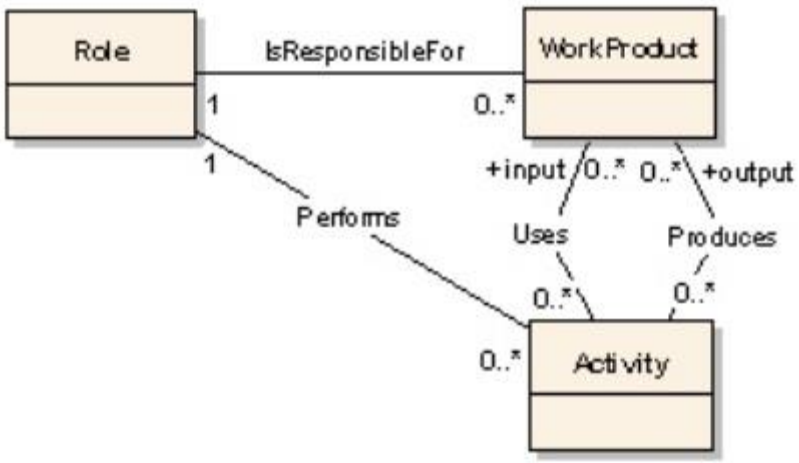
Tarkvara protsesside arendamisel peab silmas pidama, et kõik muudatused peavad lähtuma ärilisest kontekstist. Vaja on jälgida, et protsess toetab organisatsioonis elutsevat kultuuri ning eesmärke. Mida väiksemad on läbi viidavad muudatused, seda paremini suudetakse olukorda juhtida ja kontrollida. [10]

Joonisel 2.3 välja toodud protsess sobitub suures osas kokku tegevusliku uuringu metoodikaga. Tarkvara protsesside parendamise distsipliin vajab selgete rollide eristamist ning osapoolte poolset initsiatiivi ja koostööd, et teostada parandusi, mis on kõigile kasulikud. Parandused viiakse läbi iteratiivselt ning toimub sagedane protsessi hindamine, täiendamine või loomine ja rakendamine. Põhiline erinevus seisneb selles, et protsessist ei võta osa teadusliku suunaga

inimesed. Teooria ja teaduse arendamise asemel panustatakse organisatsiooni äri edendamisele ning reaaleluliste probleemide lahendamisele.

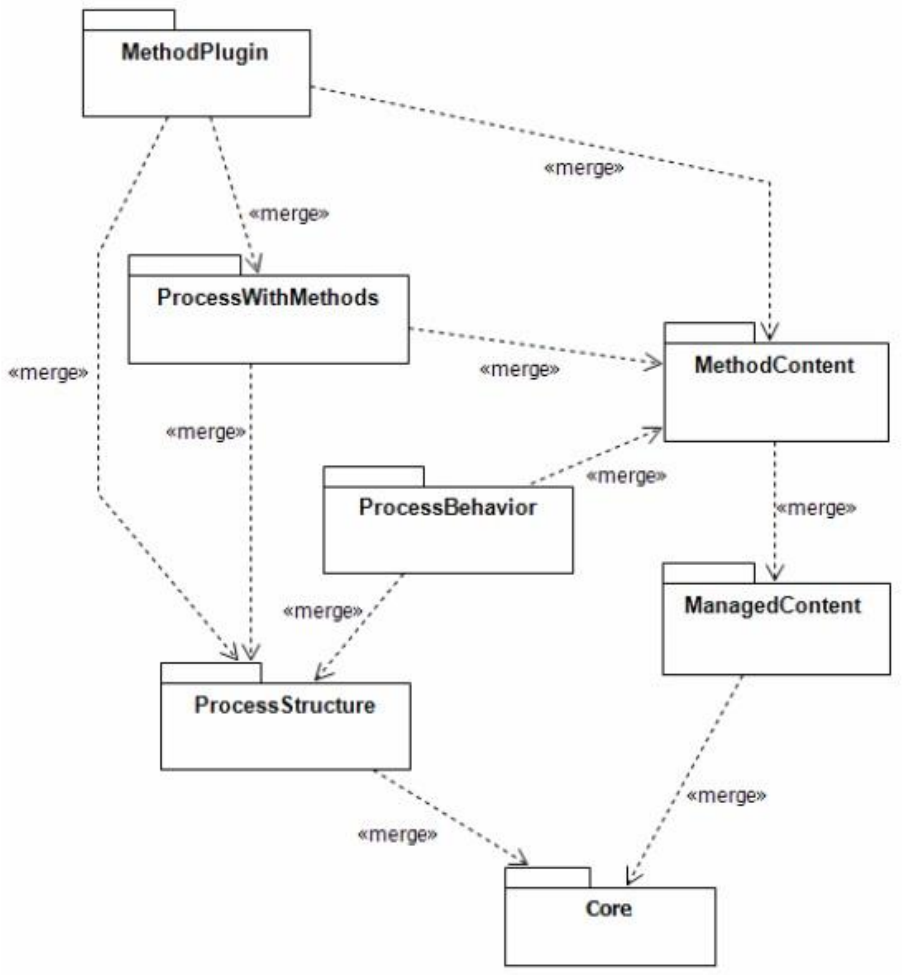
## 2.5 SPEM 2.0

Käesolevas töös kirjeldatakse ja modelleeritakse rakendatavad tarkvara protsessi parandused Software & Systems Process Engineering Meta-Model 2.0 spetsifikatsiooni (SPEM 2.0) alusel. SPEM 2.0 spetsifikatsioon kirjeldab ära protsesside modelleerimise semantika ning metamudeli, mille abil esitada tarkvara arendusprotsesse. SPEM kasutab UML'i objektorienteeritud viisil, et kirjeldada tarkvaraarendusprotsesse. SPEM baseerub joonisel 2.4 kujutatud kontseptuaalmudelil. [12]




**Joonis 2.4. SPEM'i kontseptuaalmudel**

Joonisel 2.5 on esitatud SPEM 2.0 metamudeli struktuur.




**Joonis 2.4. SPEM 2.0 metamudeli struktuur**

SPEM 2.0 metamudeli pakettide tähendused:

- Tuum – sisaldab klasse, mis on kõigile teistele pakettidele ühised.
- Protsessi struktuur – pakett defineerib protsesside kirjeldamiseks vajalikud komponendid ning võimaldab luua lihtsaid ja paindlikke protsessimudeleid.
- Protsessi käitumine – selle paketi sisu võimaldab protsessi esitada staatilise struktuurina, mis võimaldab luua seoseid erinevate tegevuste vahel.
- Hallatav sisu – võimaldab protsessi mudelite dokumentatsiooni esitada inimloetaval kujul. Pakub välja struktuuri protsessi täpsemaks kirjeldamiseks teksti kujul.

- Meetodi sisu – võimaldab kasutajatel ja organisatsioonidel luua teadmusbaasi, mis ei sõltu mitte ühestki protsessist ega projektist. Võimaldab kirjeldada korduvkasutatavaid protsessi osasid. Paketi peamised komponendid on Ülesanded, Rollid ja Tooted. Meetodi sisu on võimalik jaotada pakettidesse, kus on eraldi välja toodud eelpool nimetatud komponendid.

- Meetoditega protsess – võimaldab siduda omavahel protsessi struktuuri paketi meetodi sisu paketiga.

- Meetodi plugin – pakub vahendid, millega hallata suurema ulatusega ja taaskasutatavaid meetodite ja protsesside teekisid ja repositooriumeid. [10]

## 2.6 Kokkuvõte

Peatükis 2 anti kokkuvõte edasises uurimistöös kasutatavate võtete ja teooriate kohta. Tegevuslik disainuuring on välja arenenud tegevuslikust uuringust ja disainuuringust, võttes kokku ühelt poolt praktilise lähenemise organisatsioonikesksele probleemile ja sidudes selle tihedamalt teoreetilise poolega ja püüdlusega lahendada üldisem probleem. Järgnevas peatükis kirjeldatakse ja analüüsitakse käesoleva uurimistöö raames vaatluse all oleva projekti tarkvaraarendusprotsessi.

# 3 Esialgse arendusprotsessi mudel ja analüüs

## 3.1 Arendusprotsessi mudel

Esialgse arendusprotsessi mudel on jaotatud kolme perspektiivi: juhtimise, tarnimise ja arendamise perspektiiv. Järgnevalt on ära kirjeldatud igasse perspektiivi kuuluvad rollid, ülesanded, töötulemid ning protsess tervikuna.

### 3.1.1 Juhtimise perspektiiv

Juhtimise perspektiivi pakett vastutab toote haldamise eest.

#### 3.1.1.1 Rollid

**Tabel 3.1. Juhtimise perspektiivi rollid**

| **Roll** | **Toote juht** |
|---|---|
| Lühikirjeldus | Toote äriline juht, kes töötab välja müügistrateegiaid, visioone ja paneb paika organisatsiooni eesmärgid. |
| Põhikirjeldus | Tegu on organisatsiooni ärilise juhiga, kes jälgib suuremat pilti toote arendamisel. Toote juht hoolitseb selle eest, et toote areng liiguks organisatsiooni vaatevinklist lähtudes õiges suunas.<br>Toote juht vastutab toote müügi eest. Ta töötab välja müügistrateegiaid ja tuvastab turu vajadusi, mida rahuldada. Selleks on vaja omada üleüldist visiooni toote hetke ja tulevase arengu kohta. Toote juht paneb paika organisatsiooni eesmärgid turul konkureerides ning on ärilises mõttes lõplik otsustaja, milliseid arendusi meeskond peab arendama. Toote omanik konsulteerib pidevalt toote juht, et kaasa rääkida eesmärkide moodustamisel ning et oleks selge arusaam visioonist. |
| **Roll** | **Toote omanik** |
| Lühikirjeldus | Toote arendamise tugiisik, kes räägib kaasa toote visiooni moodustamisel ning annab arendusmeeskonnale kätte prioriteedid. |
| Põhikirjeldus | Tugiisik, kes räägib aktiivselt kaasa toote visiooni moodustamises. Toote omanik formaliseerib toote visioonist lähtuvad uued tööd ning räägib arendusmeeskonna juhiga kaasa tööde prioriteetide ja aja planeerimise osas. Tema vastutusalasse jääb uute arenduste vastuvõtutestimine ja uute tarkvara |

| | |
|---|---|
| | versioonide tarnete kinnitamine. Selles osas toimub suhtlus süsteemi administraatoritega, kes viivad läbi versioonide paigaldamisi.<br>Toote omanik on arendusmeeskonna jaoks ka peamine tugiisik, kelle käest saada täpsustusi uue tarkvara versiooni nõuetest. Toote omanik on arendusmeeskonna jaoks ühene kliendipoolne esindaja, kelle najal püsib suhtlus organisatsiooni ja meeskonna vahel. |

#### 3.1.1.2 Ülesanded

**Tabel 3.2. Juhtimise perspektiivi ülesanded**

| **Ülesanne** | **Toote strateegia arendamine** | | |
|---|---|---|---|
| **Kirjeldus** | Toote omanik ja juht arendavad toote visiooni ja tulevikuperspektiive. Toote strateegia kujundatakse koos müügistrateegiaga. Toote strateegiat planeerib põhiliselt toote omanik, konsulteerides toote juhiga. | | |
| **Eesmärk** | Eesmärgiks on hoolitseda selle eest, et toode vastaks organisatsiooni vajadustele ja visioonile. | | |
| **Sisend** | – | **Väljund** | Toote arenduse strateegia |
| **Rollid** | *Primaarne tegutseja* | Toote omanik | |
| | *Kõrvaline tegutseja* | Toote juht | |
| **Ülesanne** | **Müügistrateegia arendamine** | | |
| **Kirjeldus** | Toote omanik ja juht arendavad toote müügi strateegiat. Müügistrateegia arendamise käigus selgitatakse välja lõppkasutajate vajadused ja kohandatakse toote üleüldist visiooni. Strateegia arendamisest sõltub, milliseid tellimusi arendusmeeskond hakkab saama. Strateegia kujundamist teostab toote juht, kaasatakse ka toote omanik. | | |
| **Eesmärk** | Eesmärgiks on hoolitseda selle eest, et süsteem või toode oleks klientide jaoks ajakohane. | | |
| **Sisend** | – | **Väljund** | Toote müügistrateegia |
| **Rollid** | *Primaarne tegutseja* | Toote juht | |
| | *Kõrvaline tegutseja* | Toote omanik | |
| **Ülesanne** | **Ärinõuete hindamine** | | |
| **Kirjeldus** | Toote omanik hindab koos toote juhiga organisatsiooni ärilisi vajadusi ning nõudeid süsteemile. Süsteemi nõuded formaliseeritakse tellimustööde kujule. | | |

| | | | |
|---|---|---|---|
| **Eesmärk** | Eesmärgiks on rahuldada organisatsiooni ärilised vajadused. | | |
| **Sisend** | Toote arenduse strateegia | **Väljund** | Toote arenduse strateegia |
| **Rollid** | *Primaarne tegutseja* | Toote omanik | |
| | *Kõrvaline tegutseja* | Toote juht | |
| **Ülesanne** | **Tellimustöö koostamine** | | |
| **Kirjeldus** | Toote omanik koostab uue tellimustöö, mis sisaldab töö kirjeldust. Tellimustööks võib olla mõni halduslik tegevus, keskkonna uuendused, tarkvara uue funktsionaalsuse arendamine, vigade parandamine või tarkvara uue versiooni paigaldamine. | | |
| **Eesmärk** | Eesmärgiks on luua tellimustöö, mille käigus täieneks või paraneks süsteemi funktsionaalsus. | | |
| **Sisend** | Toote arenduse strateegia<br>Toote müügistrateegia | **Väljund** | Tellimustöö |
| **Rollid** | *Primaarne tegutseja* | Toote omanik | |
| | *Kõrvaline tegutseja* | – | |
| **Ülesanne** | **Töölaua prioritiseerimine** | | |
| **Kirjeldus** | Toote omanik vaatab üle tellimustööde prioriteedid lähtudes toote- ja müügistrateegiatest. Arvestatakse arendusmeeskonna ja süsteemi administraatorite kommentaare, kuna teatud tellimustööde paigaldamine sõltub kindlatest asjaoludest. | | |
| **Eesmärk** | Eesmärgiks on luua arendusmeeskonna jaoks tööde eelisjärjekord, mille alusel tellimusi peaks täitma. | | |
| **Sisend** | Töölaud | **Väljund** | Töölaud |
| **Rollid** | *Primaarne tegutseja* | Toote omanik | |
| | *Kõrvaline tegutseja* | Toote juht | |
| **Ülesanne** | **Tellimustöö mahuhinnangu kinnitamine** | | |
| **Kirjeldus** | Toote omanik otsustab arendusmeeskonna edastatud mahuhinnangu alusel, kas tööd on mõtet realiseerida ajalise kulu suhtes. Mahuhinnangu saamine aitab toote omanikul planeerida järgmisi tarkvara versioone, kuna järgmiste versioonide valmimine sõltub tulevaste tööde ajalisest kulust. | | |
| **Eesmärk** | Eesmärgiks on saada arendajapoolne hinnang töö realiseerimise mahule. | | |

| | | | |
|---|---|---|---|
| **Sisend** | Tellimustöö hinnang | **Väljund** | Tellimustöö hinnang |
| **Rollid** | *Primaarne tegutseja* | Toote omanik | |
| | *Kõrvaline tegutseja* | – | |
| **Ülesanne** | **Tellimustöö mahuhinnangu tagasilükkamine** | | |
| **Kirjeldus** | Toote omanik ja töö raporteerija võib arendusmeeskonna esitatud mahuhinnangut ka tagasi lükata, kui see ei vasta oodatud prognoosidele või kui projekti eelarve seda ei võimalda. Selle tulemusena läheb tellimustöö ümbervormistamisele ja ümberhindamisele. | | |
| **Eesmärk** | Eesmärgiks on vältida liiga kulukate ja vähese kasuteguriga tellimustööde realiseerimist. | | |
| **Sisend** | Tellimustöö hinnang | **Väljund** | – |
| **Rollid** | *Primaarne tegutseja* | Toote omanik | |
| | *Kõrvaline tegutseja* | Toote juht | |
| **Ülesanne** | **Uue tarkvara versiooni vastuvõtutestimine** | | |
| **Kirjeldus** | Kui tarnepakk on koostatud ja arendusmeeskonna poolt kinnitatud, siis paigaldatakse see kliendi testimiskeskkonda. Seal viib toote omanik läbi omapoolsed testid, veendumaks, et järgmine versioon ei sisalda kriitilisi vigu. Kui esineb väiksemaid vigu, mis ei ole nii kriitilised, siis need vead raporteeritakse uute tellimustöödena. | | |
| **Eesmärk** | Eesmärgiks on tagada järgmise tarkvara versiooni kvaliteet. | | |
| **Sisend** | Uus tarkvara versioon | **Väljund** | Vastuvõtu testimise aruanne |
| **Rollid** | *Primaarne tegutseja* | Toote omanik | |
| | *Kõrvaline tegutseja* | – | |
| **Ülesanne** | **Uue tarkvara versiooni kinnitamine** | | |
| **Kirjeldus** | Toote omanik ja juht kinnitavad järgmise tarnitava tarkvara versiooni paigaldamise. Järgmise tarkvara versiooni kinnitamine viiakse läbi peale vastuvõtutestimisi. Kinnitus saadetakse süsteemi administraatoritele ja arendusmeeskonnale. Administraatorid viivad läbi tarkvara paigalduse, arendusmeeskond on tarne paigaldamise käigus tegutsemise ootel. | | |
| **Eesmärk** | Eesmärgiks on avaldada, et järgmine versioon on valmis lõppkasutajatele tarbimiseks. | | |
| **Sisend** | Uus tarkvara versioon | **Väljund** | – |
| **Rollid** | *Primaarne tegutseja* | Toote omanik | |

| | | |
|---|---|---|
| | *Kõrvaline tegutseja* | – |

### 3.1.2 Tarnimise perspektiiv

Tarnimise perspektiivi pakett vastutab tarkvara tarnimise ja sellega seotud haldustööde eest.

#### 3.1.2.1 Rollid

**Tabel 3.3. Tarnimise perspektiivi rollid**

| **Roll** | **Süsteemi administraator** |
|---|---|
| Lühikirjeldus | Roll, kes vastutab süsteemi ja keskkondade ülalhoiu, seadistamise ja haldamise eest. |
| Põhikirjeldus | Süsteemi administraatori peamisteks ülesanneteks on tagada süsteemi kättesaadavus lõppkasutajatele, viib läbi hooldustöid ja uuendab tarkvara keskkondasid. Kuulub organisatsiooni IT-osakonda.<br>Lisaks süsteemi ülalhoiule hoolitseb süsteemi administraator selle eest, et tarkvara arendamise jaoks on olemas erinevad isoleeritud keskkonnad, kus on võimalik tarkvara uusi arendusi ja täiendusi ohutult testida ning toetab tarkvaraarendusosakonda selles osas. Hoolitseb selle eest, et erinevad keskkonnad oleksid võimalikult sarnased.<br>Lisaks keskkondade haldamisele, teostab süsteemi administraator ka uute tarkvara versioonide paigaldamist erinevatesse keskkondadesse, vastavalt toote omaniku kinnitatud tellimusele. |

#### 3.1.2.2 Ülesanded

**Tabel 3.4. Tarnimise perspektiivi ülesanded**

| **Ülesanne** | **Tarkvara keskkonna disainimine** | | |
|---|---|---|---|
| **Kirjeldus** | Tarkvara keskkonna disainimine ja loomine | | |
| **Eesmärk** | Eesmärgiks on hoolitseda selle eest, et nii arendajate, toote omaniku kui ka lõppkasutajate jaoks on olemas keskkond, kus pakutavat teenust kasutada. Arendajate ja toote omaniku huvideks on testimine. | | |
| **Sisend** | Süsteemi disain | **Väljund** | Tarkvara keskkond |
| **Rollid** | *Primaarne tegutseja* | Süsteemi administraator | |
| | *Kõrvaline tegutseja* | Arhitekt/analüütik | |
| **Ülesanne** | **Eksisteeriva keskkonna tagavarakoopia tegemine** | | |

| | | | |
|---|---|---|---|
| **Kirjeldus** | Toimivast tarkvara keskkonnast tagavarakoopia loomine. Teostatakse enne tarkvara uuenduste rakendamist. | | |
| **Eesmärk** | Eesmärgiks on maandada tarkvara uue versiooni paigaldamise riskifaktorit. Kui uue versiooni paigaldamise käigus ilmnevad vead, mida arendamise käigus ei tuvastatud, siis peab olema võimalik esialgne keskkond taastada. | | |
| **Sisend** | Tarkvara keskkond | **Väljund** | Tarkvara keskkond |
| **Rollid** | *Primaarne tegutseja* | Süsteemi administraator | |
| | *Kõrvaline tegutseja* | – | |
| **Ülesanne** | **Uue tarkvara versiooni paigaldamine** | | |
| **Kirjeldus** | Uue tarkvara versiooni paigaldamine lõppkasutajatele mõeldud keskkonda. | | |
| **Eesmärk** | Eesmärgiks on teha valmis arendatud tööd lõppkasutajatele avalikuks ja kasutatavaks. | | |
| **Sisend** | Tarkvara keskkond<br>Uus tarkvara versioon | **Väljund** | Tarkvara keskkond |
| **Rollid** | *Primaarne tegutseja* | Süsteemi administraator | |
| | *Kõrvaline tegutseja* | Andmebaasi administraator<br>Tarkvaraarendaja | |
| **Ülesanne** | **Uue tarkvara versiooni paigalduse tühistamine** | | |
| **Kirjeldus** | Kui uue tarkvara versiooni paigaldamisel ilmnes vigu, siis tühistatakse viimane tarne. Riskide maandamiseks peab olema valmisolek tühistada tarne paigaldamine ning taastada esialgne töötav versioon. Tühistamine tehakse varasemas staadiumis loodud tagavarakoopia alusel. | | |
| **Eesmärk** | Eesmärgiks on tagada süsteemi järjepidev toimimine. | | |
| **Sisend** | Tarkvara keskkond | **Väljund** | Tarkvara keskkond |
| **Rollid** | *Primaarne tegutseja* | Süsteemi administraator | |
| | *Kõrvaline tegutseja* | Andmebaasi administraator | |
| **Ülesanne** | **Tarkvara keskkonna uuendamine** | | |
| **Kirjeldus** | Süsteemi administraatorid viivad läbi süsteemi tarkvara uuendusi sobilikel aegadel. Tarkvara uuendusi on vaja läbi viia, kuna seeläbi kaotatakse ära teatud turvaaugud, ebaefektiivsused. Teatud tarkvara vanemaid versioone ei toetata ka enam. | | |

| | | | |
|---|---|---|---|
| **Eesmärk** | Eesmärgiks on hoolitseda selle eest, et süsteemi jooksutav tarkvara oleks uuendatud kõige uuema kasutatava versioonini. | | |
| **Sisend** | Tarkvara keskkond | **Väljund** | Tarkvara keskkond |
| **Rollid** | *Primaarne tegutseja* | Süsteemi administraator | |
| | *Kõrvaline tegutseja* | – | |
| **Ülesanne** | **Süsteemi monitoorimine** | | |
| **Kirjeldus** | Süsteemi monitoorimine. Süsteemi monitoorimine hõlmab endas vigade tuvastust, jõudluse jälgimist, lõppkasutajate aktiivsuse tuvastamist. | | |
| **Eesmärk** | Eesmärgiks on tagada süsteemi ja keskkonna veatu toimimine. | | |
| **Sisend** | Tarkvara keskkond | **Väljund** | Tarkvara keskkond |
| **Rollid** | *Primaarne tegutseja* | Süsteemi administraator | |
| | *Kõrvaline tegutseja* | – | |

### 3.1.3 Arendamise perspektiiv

Arendamise perspektiivi pakett vastutab tarkvara arendamisega seotud tegevuste ja pädevusalade eest.

#### 3.1.3.1 Rollid

**Tabel 3.5. Arendamise perspektiivi rollid**

| **Roll** | **Arhitekt/analüütik** |
|---|---|
| Lühikirjeldus | Tarkvara arhitekt/analüütik |
| Põhikirjeldus | Tehnilise kallakuga süsteemi analüütik, kes töötab välja tarkvara arhitektuuri ja disaini vastavalt süsteemile püstitatud ärinõuetele. Mahukamate tellimustööde korral teeb arhitekt/analüütik tihedalt koostööd toote omanikuga, et formaliseerida ülejäänud arendajatele detailsemad nõuded, mida realiseerida.<br>Pakub igapäevast tuge tarkvaraarendajatele süsteemi disainimisel ja tehniliste lahenduste väljapakkumisel. |
| **Roll** | **Andmebaasi administraator** |

| | |
|---|---|
| Lühikirjeldus | Andmebaasi administraator tegeleb andmebaasi arenduste realiseerimise ja hooldustöödega. |
| Põhikirjeldus | Andmebaasi administraatori ülesandeks on tagada andmebaasi ülalhoid. Ta tegeleb andmebaasi arenduste realiseerimisega, koostab eritellimusena aruandeid kliendi jaoks ning viib läbi hooldustöid ja andmebaasi optimeerimisi.<br>Andmebaasi administraator teeb koostööd analüütiku/arhitektiga süsteemi disaini osas. Pakub igapäevast tuge tarkvaraarendajatele süsteemi disainimisel ja tehniliste lahenduste väljapakkumisel. |
| **Roll** | **Tarkvaraarendaja** |
| Lühikirjeldus | Tarkvara arendaja realiseerib toote omaniku püstitatud nõuded tootele. |
| Põhikirjeldus | Tarkvara arendaja ülesanne on realiseerida uued toote tellimustööd vastavalt toote omaniku määratud prioriteetidele. Tarkvara arendaja teeb koostööd arhitekti/analüütikuga, et leida parim disain ja lahendus uutele tellimustele. Oma osa mängib ka koostöö toote omanikuga, kes täpsustab püstitatud nõudeid. Arendusmeeskonna juht määrab ära, millised tööd on kõige kriitilisemad ning mis vajavad kõigepealt tähelepanu. Tarkvaraarendaja ei tegele andmebaasipoolsete arenduste realiseerimisega, need tööd delegeeritakse andmebaasi administraatorile. |
| **Roll** | **Tarkvaraarendusmeeskonna juht** |
| Lühikirjeldus | Korraldab ja haldab tarkvara arendustöid meeskonnas. |
| Põhikirjeldus | Tarkvaraarendusmeeskonna juht vastutab selle eest, et kõigil meeskonnaliikmetel oleksid ülesanded, mida täita ja lahendada. Meeskonna juht osaleb kliendi koosolekutel ning esindab meeskonda. Ta jaotab ülesandeid meeskonnaliikmete vahel, vastavalt kliendi püstitatud prioriteetidele ja visioonile. Selle teostamiseks on vaja pidevalt koostööd toote omanikuga.<br>Meeskonna juhi üheks ülesandeks on ka tarkvaraarenduse üksuse majandusliku olukorra jälgimine. |
| **Roll** | **Testija** |
| Lühikirjeldus | Tarkvara kvaliteedi spetsialist, kes kinnitab, et tarnitava tarkvara funktsionaalsus toimib korrektselt. |

| | |
|---|---|
| Põhikirjeldus | Tarkvara kvaliteedi spetsialist, kelle ülesandeks on tagada tarnitava tarkvara kvaliteet. Testija suhtleb tihedalt toote omanikuga, et täpsustada nõudeid, mille vastu ta tarkvara testib ning tarkvaraarendajatega, et täpsustada läbi viidud arenduste mõju.<br>Tarkvara testija on arendusmeeskonna viimane isik, kelle pilgu alt käib tarnitav uus versioon läbi, mille käigus testitakse kõiki tarnitavaid uusi arendusi korraga. Testija testib arendusi ka eraldiseisvalt, kui arendaja on töö valmis saanud. |

#### 3.1.3.2 Ülesanded

**Tabel 3.6. Arendamise perspektiivi ülesanded**

| **Ülesanne** | **Tellimustöö hindamisele suunamine** | | |
|---|---|---|---|
| **Kirjeldus** | Tellimustöö suunatakse arendusmeeskonnale hindamiseks. Tellimustöö hindamiseks suunamine omab halduslikku tähendust. Tellimustööd koordineerimisega tegeleb tarkvaraarendusmeeskonna juht, kes valib koostöös ülejäänud meeskonnaga välja tööd realiseerivad osapooled ning suunab töö neile hindamiseks. | | |
| **Eesmärk** | Eesmärgiks on kergemini hallata tellimustöid arendusmeeskonna siseselt | | |
| **Sisend** | Tellimustöö | **Väljund** | – |
| **Rollid** | *Primaarne tegutseja* | Tarkvaraarendusmeeskonna juht | |
| | *Kõrvaline tegutseja* | Andmebaasi administraator<br>Arhitekt/analüütik<br>Tarkvaraarendaja<br>Testija | |
| **Ülesanne** | **Tellimustöö hindamine** | | |
| **Kirjeldus** | Tellimustöö hindamist viivad läbi kõik tarkvaraarendusmeeskonna liikmed, mille tulemusena kujuneb koondav hinnang. Koondava hinnangu alusel langetab toote omanik otsuse, kas kulutada ressursse töö realiseerimiseks või mitte. Tellimustöö hinnang saadetakse toote omanikule kinnitamiseks. Hinnangut mõjutab ka olemasoleva süsteemi disain, mida teeb ülesande kas keerulisemaks või lihtsamaks. | | |

| | | | |
|---|---|---|---|
| **Eesmärk** | Eesmärgiks on välja selgitada tellimustöö realiseerimiseks vajalik ajakulu, mis aitab meeskonnas aega planeerida ja toote omanikul otsuseid langetada. | | |
| **Sisend** | Tellimustöö<br>Süsteemi disain | **Väljund** | Tellimustöö hinnang |
| **Rollid** | *Primaarne tegutseja* | Andmebaasi administraator<br>Tarkvaraarendaja<br>Testija | |
| | *Kõrvaline tegutseja* | Arhitekt/analüütik<br>Süsteemi administraator | |
| **Ülesanne** | **Tellimustöö hinnangu kinnitamiseks suunamine** | | |
| **Kirjeldus** | Tellimustöö suunatakse koos mahuhinnanguga toote omanikule kinnitamiseks või tagasilükkamiseks. Tellimustöö hinnangu kinnitamiseks suunamine omab organisatoorset tähendust. Arendusmeeskonna juht vastutab selle eest, et tellimustöö oleks kõigi vajalike osapoolte poolt hinnatud. | | |
| **Eesmärk** | Eesmärgiks on saada toote omaniku nõusolek hakata tööd realiseerima. | | |
| **Sisend** | Tellimustöö hinnang | **Väljund** | Tellimustöö |
| **Rollid** | *Primaarne tegutseja* | Tarkvaraarendusmeeskonna juht | |
| | *Kõrvaline tegutseja* | – | |
| **Ülesanne** | **Tellimustöö analüüsimine** | | |
| **Kirjeldus** | Tellimustööd võivad olla äriliselt üsna keerulised, mille tõttu pole otstarbekas kohe alustada realiseerimisega. Detailide ja võimalike olukordade ning tagajärgede väljaselgitamiseks analüüsib arhitekt/analüütik tellimustööd ja selle mõjusid. Analüüsi tulemusena koostatakse dokumentatsioon, mida tarkvaraarendajad, testijad ja toote omanik võtavad arvesse, kui realiseeritakse ja testitakse antud tellimustööd. Arhitekt/analüütik teeb koostööd tarkvaraarendajate, andmebaasi administraatori ja toote omanikuga. | | |
| **Eesmärk** | Eesmärgiks on luua dokumentatsioon, mille alusel on tarkvaraarendajatel kergem tellimustööd realiseerida. | | |
| **Sisend** | Tellimustöö | **Väljund** | Tellimustöö dokumentatsioon |
| **Rollid** | *Primaarne tegutseja* | Arhitekt/analüütik | |
| | *Kõrvaline tegutseja* | Andmebaasi administraator<br>Tarkvaraarendaja<br>Toote omanik | |

| **Ülesanne** | **Süsteemi disaini hindamine** | | |
|---|---|---|---|
| **Kirjeldus** | Süsteemi arhitektuuri ja disaini hindamine ja parandusettepanekute tegemine. Süsteemi disaini läbivaatusi viib läbi arhitekt ja/või vanemtarkvaraarendaja. Hindamise käigus tehakse pööratakse tähelepanu kitsaskohtadele ja tehakse parandusettepanekuid, mida järgmiste tellimustööde käigus oleks mõistlik parandada. | | |
| **Eesmärk** | Eesmärgiks on parandada süsteemi disaini | | |
| **Sisend** | Süsteemi disain | **Väljund** | Süsteemi disain |
| **Rollid** | *Primaarne tegutseja* | Arhitekt/analüütik<br>Tarkvaraarendaja | |
| | *Kõrvaline tegutseja* | – | |
| **Ülesanne** | **Tellimustöö arendamine** | | |
| **Kirjeldus** | Tellimustööd hakkavad arendama erinevad tarkvaraarendusmeeskonna liikmed sõltuvalt töö iseloomust. Meeskonna liikmed on selgelt ära piiritlenud skoobi, mille raames nemad arendavad, st tarkvaraarendajad arendavad rakendust, andmebaasi administraatorid realiseerivad andmebaasi arendusi ja süsteemi administraatorid lahendavad keskkonnaga seotud probleeme. Kui tellimustöö sisaldab mitmeid skoope, siis meeskonna liikmed teevad tihedat koostööd, et kõigi realiseeritavad osad oleksid kooskõlas. | | |
| **Eesmärk** | Eesmärgiks on luua uut funktsionaalsust või parandada olemasolevat. | | |
| **Sisend** | Tellimustöö | **Väljund** | Tarnitav tellimustöö |
| **Rollid** | *Primaarne tegutseja* | Andmebaasi administraator<br>Süsteemi administraator<br>Tarkvaraarendaja | |
| | *Kõrvaline tegutseja* | Arhitekt/analüütik | |
| **Ülesanne** | **Täiendava informatsiooni küsimine tellimustöö kohta** | | |
| **Kirjeldus** | Kui tellimustöö kirjeldus on kahetimõistetav, ebaselge või ebatäpne, on töö realiseerijal kasulik küsida tellija käest üle ebaselged detailid. Tellijaga saab võtta otse ühendust, mille käigus ta selgitab täpsemalt töö ärilisi nõudeid ja vajadusi. | | |
| **Eesmärk** | Eesmärgiks on vähendada ebakõlade tekkimise ohtu | | |
| **Sisend** | Tellimustöö | **Väljund** | – |
| **Rollid** | *Primaarne tegutseja* | Andmebaasi administraator | |

| | | | |
|---|---|---|---|
| | | Arhitekt/analüütik<br>Tarkvaraarendaja<br>Testija<br>Toote omanik | |
| | *Kõrvaline tegutseja* | – | |
| **Ülesanne** | **Tellimustöö testimine** | | |
| **Kirjeldus** | Realiseeritud tellimustööd vaatab üle meeskonna testija, kes proovib läbi erinevad stsenaariumid funktsionaalsuse ja jõudluse osas. Testija koostab oma tulemuste kohta aruande, mis kinnitatakse tellimustöö tulemuste külge. Testija teeb koostööd ka toote omaniku arhitekti/analüütikuga ja tarkvaraarendajatega, et paremini hinnata funktsionaalsuse vastavust nõuetele. | | |
| **Eesmärk** | Eesmärgiks on maandada tarnitavate tellimustööde riske ning tagada süsteemi kvaliteet. | | |
| **Sisend** | Tellimustöö | **Väljund** | Tarnitav tellimustöö<br>Tellimustöö testimise aruanne |
| **Rollid** | *Primaarne tegutseja* | Testija | |
| | *Kõrvaline tegutseja* | Arhitekt/analüütik<br>Tarkvaraarendaja<br>Toote omanik | |
| **Ülesanne** | **Uue tarkvara versiooni koostamine** | | |
| **Kirjeldus** | Valmis arendatud tellimustöödest moodustatakse tarnepakk, mille peab lähitulevikus lõppkasutajatele publitseerima. Kui iteratsioon hakkab lõppema ning kõik nõutud tellimustööd on realiseeritud, siis arendused ühendatakse omavahel ning moodustatakse tarnepakk. Tarnepaki suunatakse süsteemi administraatorile, kes viib läbi paigaldamise. | | |
| **Eesmärk** | Eesmärgiks on teha realiseeritud tellimustööd kõikidesse keskkondadesse paigaldatavateks. | | |
| **Sisend** | Tarnitav tellimustöö | **Väljund** | Uus tarkvara versioon |
| **Rollid** | *Primaarne tegutseja* | Tarkvaraarendaja | |
| | *Kõrvaline tegutseja* | Andmebaasi administraator | |

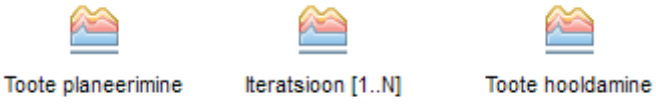
### 3.1.4 Töötulemid

**Tabel 3.7. Töötulemite kirjeldused**

| Töötulem | Kirjeldus |
|---|---|
| Tarkvara keskkond | Süsteemi tarkvara jooksutamiseks mõeldud keskkond, mida on võimalik replikeerida. |
| Uus tarkvara versioon | Tegu on uue tarnitava versiooniga valmis arendatud tellimustöödest. Uus tarkvara versioon sisaldab tellimustööd, selle dokumentatsiooni, testimise aruannet ja selle raames arendatud koodi. |
| Tarnitav tellimustöö | Tegu on valmisarendatud ja testitud tellimustööga, mida on võimalik hakata tarkvara järgmisesse versiooni paigutama. |
| Süsteemi disain | Süsteemi (tarkvara ja keskkonna) disain. |
| Töölaud | Tellimustööde nimekiri |
| Tellimustöö | Tellimustöö, mida toote omanik hakkab arendusmeeskonna käest tellima. Tellimustööks võib olla mõni halduslik tegevus, keskkonna uuendused, tarkvara uue funktsionaalsuse arendamine, vigade parandamine või tarkvara uue versiooni paigaldamine. |
| Tellimustöö dokumentatsioon | Dokumentatsioon, mis valmib tellimustöö detailsemal analüüsimisel. |
| Tellimustöö hinnang | Tellimustöö jaoks kuluv orienteeruv aeg. |
| Tellimustöö testimise aruanne | Testija koostab tellimustöö testimise tulemusena aruande, mis kirjeldab testimise tulemusi. |
| Vastuvõtu testimise aruanne | Kokkuvõte järgmise tarkvara versiooni testimisest. |
| Toote arenduse strateegia | Visioon, mille suunas soovib organisatsioon toote arendamisega liikuda. |
| Toote müügistrateegia | Visioon, kuidas toote uusi täiendusi ja parandusi saab kasutada käibe suurendamiseks. |

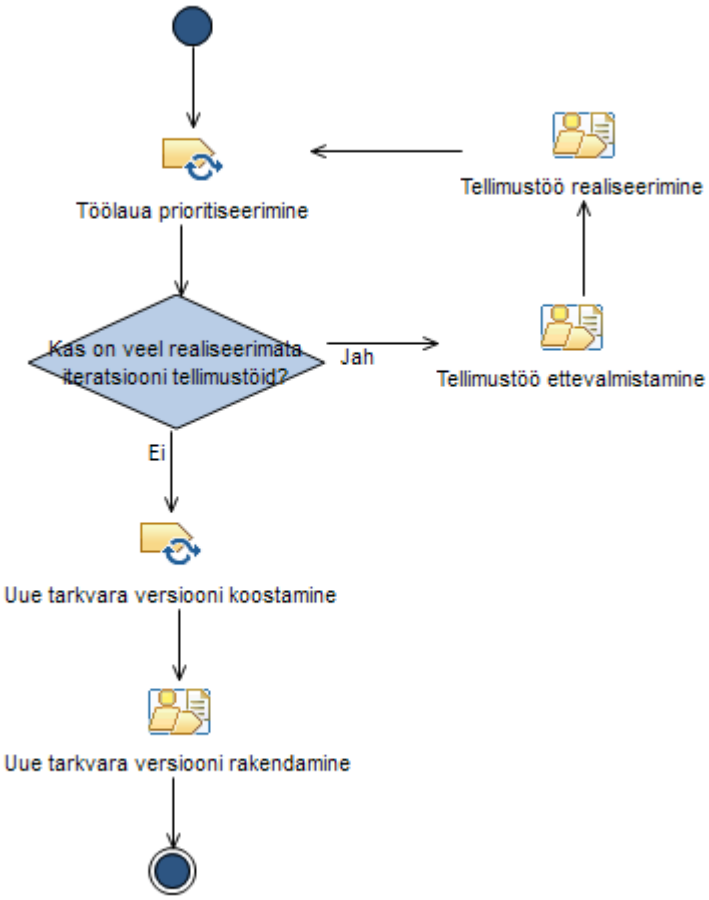
## 3.2 Arendusprotsessi analüüs

Käsitluse all olevas projektis on kasutusel iteratiivne lähenemine. Sellega paralleelselt toimuvad ka muud strateegilised ja halduslikud tegevused. Joonisel 3.1 on välja toodud arendusprotsessi põhilisemad tegevused. Toimub iteratiivne arendus, mille käigus planeeritakse toodet (sh arendatakse toote- ja müügistrateegiat) ning hooldatakse toodet (sh monitooritakse süsteemi ja viiakse läbi tarkvarauuendusi).




**Joonis 3.1. Süsteemi arendusprotsessi tegevused**

Toote arendus- ja müügistrateegia arendamine ning süsteemi perioodiliste uuenduste ja monitooringu korraldamine on organisatsiooni jaoks põhimõttelise tähendusega, mis on väga oluline osa infosüsteemi haldava projekti eluea kujunemisel.



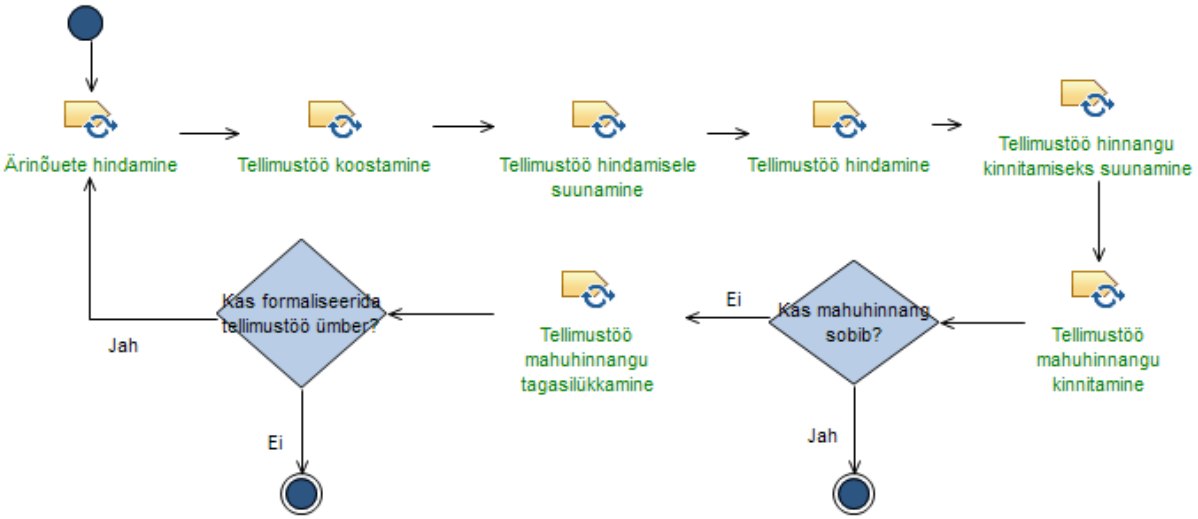

**Joonis 3.2. Iteratsiooni tegevusdiagramm**

Joonisel 3.2 on välja toodud tarkvara iteratsioonide põhiline tegevuste jada. Tellimustöid valmistatakse pidevalt ette ning realiseeritakse vastavalt prioriteetidele. Tellimustööde prioriteete hinnatakse pidevalt ümber, kuna tellija vajadused muutuvad üsna sageli. See on endaga kaasa toonud probleemid tarkvara versioonide planeerimisel. Projektis ilmneb väga sageli tarkvara paketeerimise vigu, mis on seotud uue versiooni pideva ümberplaneerimisega. Projekti versioonihalduse süsteemis puudub ühtselt defineeritud süsteem, mille järgi arendusi realiseerida ja hallata. See põhjustab palju vigu tarkvara tarnimisel.

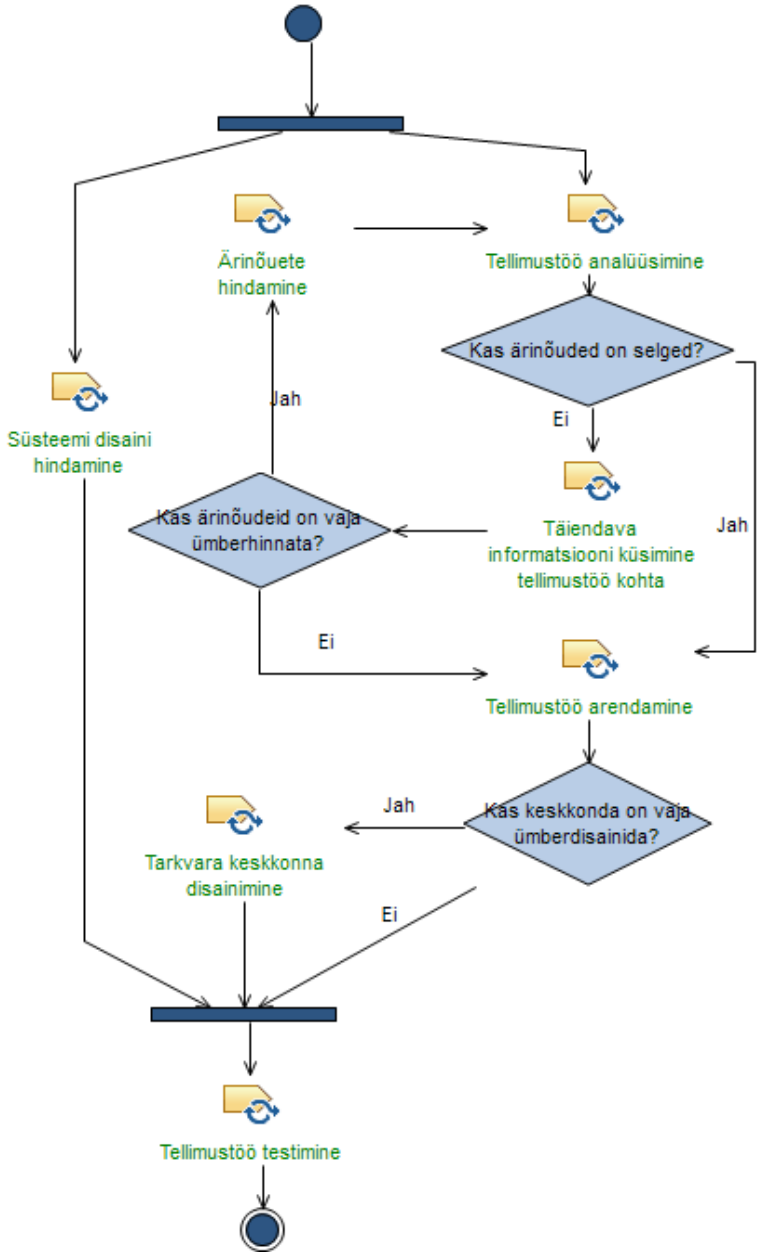
Tarkvaraarenduse iteratsioonide üheks osaks on tellimustööde ettevalmistamine, mis hõlmab tööde hindamist. Joonisel 3.3 on välja toodud hindamise protsess, mille käigus koostatakse mahuhinnang ning küsitakse töö tellijalt sellele kinnitust.

Üheks põhiliseks probleemiks antud alamprotsessis on hinnangu koostamise etapp, mille käigus kõik meeskonna liikmed esitavad oma hinnangud teineteisest sõltumata. Meeskonnaliikmed

töötavad erinevatest ruumides ning omavaheline suhtlus on häiritud. Alternatiivse lahendusena võiks meeskond regulaarselt korraldada mahuhinnangute andmise koosolekuid, kus meeskonnaliikmed saavad omavahel suhelda ning läbi arutada tellitavate tööde sisu.



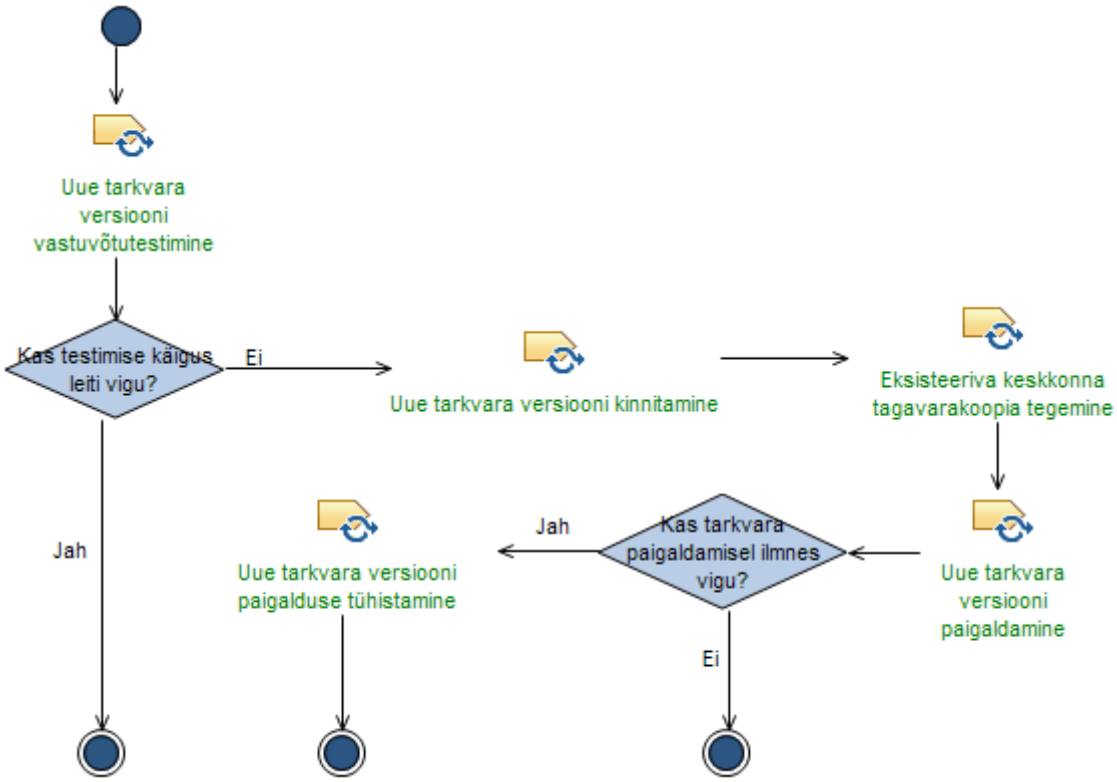

**Joonis 3.3. Tellimustöö ettevalmistamise tegevusdiagramm**




**Joonis 3.4. Tellimustöö realiseerimise tegevusdiagramm**

Tellimustöö realiseerimine on samuti üheks alamprotsessiks käesoleva projekti arendusprotsessis. Joonisel 3.4 on kirjeldatud tarkvara realiseerimise protsess. Üheks kriitilisemaks arendusprotsessi veaks on tellimustöö arendamise käigus tööde liigne piiritlemine rollide vahel. Lähtuvalt lähteolukorrale on projektis kasutusel põhimõte, et tarkvaraarendajad arendavad ainult süsteemi rakendust, andmebaasi administraatorid tegelevad ainult andmebaasi arendamisega jne. Selline töökorraldus pärsib süsteemialase kompetentsi edasiandmist inimeste vahel. Kui tarkvaraarendaja realiseeriks ka andmebaasi arendusi, suureneks tarkvaraarendajal ülevaade andmemudelist, mis omakorda aitab kaasa tulevaste

tööde hindamisel ja realiseerimisel. Andmebaasi administraator saab alati üle vaadata tarkvaraarendaja loomingu, mis ei pruugi olla kõige efektiivsem lahendus. Olgugi, et probleemi saaks efektiivsemalt lahendada, on süsteemi kui terviku mõistmise arendamine olulisem. Probleem on veelgi suurema faktoriga, kuna meeskonnaliikmed viibivad erinevates ruumides. Kui meeskonnaliikmete vastutusalad on sellisel määral piiritletud, siis on efektiivne meeskonnasuhtlus hädavajalik. Vastasel juhul tekib oluliselt rohkem vigu tellimustööde realiseerimisel. Meeskonnasuhtluse parandamiseks aitavad kaasa regulaarsed koosolekud, ühise suhtluskeskkonna kasutamine omavaheliseks suhtlemiseks ning meeskonnaüritused.




**Joonis 3.5. Uue tarkvara versiooni rakendamise tegevusdiagramm**

Viimaseks olulisemaks arendusprotsessi alamprotsessiks on tarkvara uue versiooni rakendamine (vt joonis 3.5). Käsitluse all olevas projektis polnud uute tarkvara versioonide rakendamisel olulisi probleeme. Tarkvara uuendamisele eelneb alati süsteemi tagavarakoopia tegemine olulisematest komponentidest. Tarkvara paigaldavad kogenud administraatorid, kes on oma ala spetsialistid. Uue versiooni paigaldamisel vigade ilmnedes on alati võimalik taastada esialgse tarkvara seis. Selle tulemusena on riskid maandatud.

## 3.3 Kokkuvõte

Kolmandas peatükis kirjeldati tarkvaraarendusprotsessi projekti alguses. Arendusprotsessi kirjeldamisel kasutati SPEM 2.0 metamudeli spetsifikatsiooni. Kirjeldati ära protsesside põhilised rollid ning nende poolt täidetavad tegevused ja ülesanded. Anti ülevaade olulisematest töötulemitest ning arendusprotsessist tervikuna. Järgnevas peatükis viiakse läbi arendusprotsessi parandused, mis leevendavad käesolevas peatükis kirjeldatud puudujääke. Parandusi viiakse läbi iteratiivsel kujul.

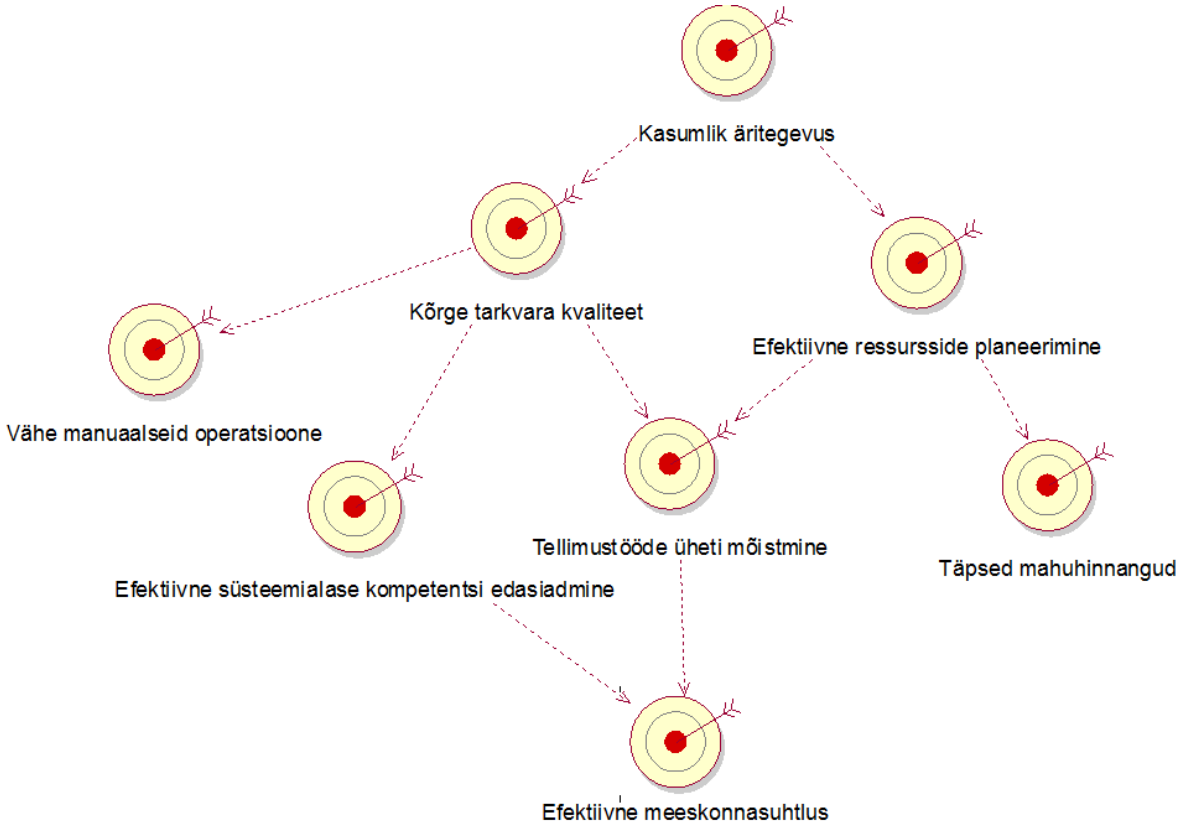
# 4 Arendusprotsessi paranduste läbiviimine

Käesolev peatükk käsitleb arendusprotsessi paranduste läbiviimist projektis, mis põhineb eelmises peatükis kirjeldatud metamudelil ja protsessi analüüsil. Arendusprotsessis paranduste läbiviimisel ja selle arendamisel rakendatakse kombineeritud lähenemist, kus lähtutakse olemasolevate raamistike näidisprotsessidest ning kasutatakse meetodite fragmente uue protsessi ehitamisel. Kombineeritud lähenemine võimaldab rakendada olemasolevate metoodikate elemente, samal ajal võttes arvesse nii organisatsioone ärilisi vajadusi kui ka struktuuri omapära. Arendusprotsessi parandused teostatakse lähtudes tegevusliku disainuuringu põhimõtetest (vt peatükk 2.3). Probleemide lahendamisele lähenetakse organisatsioonile orienteeritult, mille käigus hinnatakse põhiliselt organisatsiooni hetkeolukorda ja ärilisi vajadusi.

Analüüsi tulemusena tuvastati 5 põhilist probleemivaldkonda arendusprotsessis:

1) Süsteemi alase kompetentsi edasiandmine
2) Iteratsioonide käigus tarnitava tarkvara kvaliteet
3) Iteratsioonide planeerimine
4) Meeskonnasuhtlus
5) Tellimustööde hindamise protsess

Probleemsed valdkonnad on otseselt seotud ettevõtte infosüsteemi arendamise äriprotsessi eesmärkidega. Joonis 4.1 kujutab arendusprotsessi eesmärke. Ülimaks eesmärgiks on tagada äri kasumlik toimimine, kulude minimeerimine ja käibe suurendamine. Selle saavutamisele aitab kaasa efektiivne ressursside planeerimine ja kõrge tarkvara kvaliteedi tagamine. Kulude minimeerimist realiseeritakse automatiseerimise, korrektsete töömahuhinnangute koostamisega ja efektiivse meeskonnatöö korraldusega. Kõrge kvaliteet tagatakse süsteemialase kompetentsi hoidmise ja edasiandmisega, tellimustööde ühese mõistmisega ning samuti sujuva meeskonnatööga. Eesmärke saab mõõta juhtkonna ja meeskonna rahuloluhinnangute ning organisatsiooni majandustulemuste abil.




**Joonis 4.1. Infosüsteemi arendamise eesmärkmudel**

Käesoleva protsessi parandamisega üritatakse leida universaalne ja kergesti rakendatav tarkvaraarendusprotsess, mis sobiks võimalikult paljudele tarkvaraarendusprojektidele. Reeglina ei ole praktikas võimalik rakendada puhtal kujul tarkvaraarendusprotsesse, mis on erinevates metoodikates ja raamistikes kirjeldatud. Erinevused on tingitud äriliste huvide omapärast. Tegevusliku disainuuringu põhimõtetest võetakse uurimise alla tarkvaraarendusprotsess infosüsteemikesksetes ettevõtetes, mis sümboliseerib üldist tüüpi probleemi.

Enne paranduste läbiviimist arendusprotsessis, koostas autor esialgse tegevuskava, mille järgi parandusi läbi viia. Esimeses etapp käsitleb arendusmeeskonna töökorraldusealaseid probleeme, mille alla kuuluvad meeskonna süsteemialase kompetentsi edasiandmine, meeskonnasuhtlus ning tellimustööde hindamise protsess. Teine etapp sisaldab tööde planeerimise, tarkvara kvaliteedi ja tarnimisega seotud probleeme. Paranduste planeerimisel ja läbiviimisel toimus pidev koostöö arendusmeeskonna juhiga ning toote omanikuga. Tagasisidet küsiti ka teistele meeskonnaliikmetelt.

Arendusprotsessi parandused viidi läbi iteratiivsel kulu. Kuna oli tegu vana projektiga, mille protsessid olid olnud kasutusel mitmeid aastaid, otsustati rakendada heuristilist äriprotsesside ümberdisainimise lähenemist. Protsessi ümberdisainimisel lähtuti soovist leida kesktee protsessi paindlikkuse, kvaliteedi ja ajalise efektiivsuse osas. Paindlik lähenemine on antud olukorras väga oluline, kuna protsessis osalejate nõuetele ja soovidele kiire reageerimine võimaldab rohkem probleeme efektiivsemalt ära lahendada. Heuristiline lähenemine võimaldab paremini leida ka universaalset tarkvaraarendusprotsessi lahendust, kuna kesktee sobib rohkemate projektidega. [12]

Paranduste läbiviimisel lähtuti Agiilse Manifesto põhimõtetest. Pöörati rohkem tähelepanu meeskonnaliikmete ja nendevahelistele interaktsioonidele, toimiva tarkvara, kliendisuhtluse ning muudatustele reageerimise peale. [15]

Paranduste realiseerimise käigus kujunes kolm eristuvat iteratsiooni. Iga iteratsiooni alguses püstitati eesmärgid, mida sooviti saavutada ning milliseid probleeme üritati lahendada. Planeerimise käigus pakuti välja ka võimalikud võtted, kuidas probleeme lahendada ning millistele teoreetilistele alustaladel need muudatused põhinesid. Iga iteratsioon lõppes parandustulemuste kokkuvõttest ja analüüsist, mille põhjal kujunes välja järgmise iteratsiooni plaan.

Tegevusliku disainuuringu kontekstis kujutas peatükk 3 endast probleemi formuleerimise faasi. Käesolevas peatükis kirjeldatud iteratsioonid peegeldavad arendamise, vahelesekkumise ja hindamise faasi ning hindamise ja peegeldamise faasi. Järgnevalt on ära kirjeldatud läbiviidud iteratsioonid ning nende tulemused.

## 4.1 I iteratsioon

Arendusprotsessi parandamise esimeses iteratsioonis fokusseeriti arendusmeeskonna töökorraldusalaste probleemide lahendamisele. Iteratsiooni eesmärkideks oli:

- parandada süsteemialase kompetentsi edasiandmist meeskonnas;
- parandada meeskonnasuhtlust;
- muuta tellimustööde hindamist täpsemaks.

Nimetatud probleemid on peamiselt tarkvara konstruktsiooni faasile orienteeritud ning mõjutavad kõige otsesemal kujul tarnitava tarkvara kvaliteeti. Teised kvaliteeti puudutavad probleemid, nagu tarkvara tarnete planeerimine otsustati jätta järgmiste iteratsioonide skoopi, kuna arendusmeeskonna töö efektiivsem toimimine on parema planeerimise ja kvaliteedi saavutamise eelduseks. Distsiplineeritud agiilse üleandmise (*Disciplined Agile Delivery*) metoodika järgi on üheks kõige olulisemaks põhimõtteks inimestega seotud probleemidega eelisjärjekorras tegelemine. Alastair Cockburni väitel on inimesed ja nendevaheline suhtlus peamised IT lahenduste realiseerimise edufaktorid.

### 4.1.1 Meeskonnasuhtluse parandamine

Üheks peamiseks probleemiks esialgses arendusprotsessis oli halb või puudulik meeskonnasuhtluse suunamine. Selleks, et tarkvaraarendusprojektid oleksid edukad, on vaja meeskonnaliikmetevahelist efektiivset koostööd, mille tulemusena toimub infovahetus ja teadmiste edasiandmine meeskonnaliikmete vahel. Olemasolevas arendusprotsessis puudus ühene keskkond, kus meeskond oleks igapäevaselt saanud jooksvate igapäevaste probleemide teemadel suhelda. Meeskonnakoosolekud toimusid ka ebakorrapärastel aegadel ning selget koosolekuplaani polnud. Tegu on arendusprotsessi kõige põhilisema osaga, millest algab enamik teisi probleeme.

Probleemi lahendamiseks loodi kõigile meeskonnaliikmetele kättesaadav keskkond, kus igapäevaselt omavahel suhelda. Lahendus realiseeriti Skype'i grupivestlusena. Samuti võeti vastu otsus hakata iga nädala esmaspäeva hommikuti korraldama koosolekuid, millest võtab osa terve arendusmeeskond koos toote omanikuga. Koosolekute jaoks lepiti kokku struktuur, mida jälgida. Struktuur koosneb järgnevatest temaatikatest:

- Arendusmeeskonna liikmete kokkuvõtete esitamine eelmisel nädalal tehtud töö tulemuste kohta
- Järgmise tarkvara tarne planeerimine või pooleli oleva tarne ülevaate andmine
- Arendusmeeskonna liikmetele järgmise nädala tööde prioriteetide määramine

Koosoleku formaat sarnaneb suures osas igapäevaste Scrumi metoodika stand-up koosolekutega, kus arendajad peavad andma ülevaate eelmisel päeval tehtud töödest, käesoleva päeva töödest ja nendega seoses ilmnevatest probleemidest. Samuti on sarnasused Scrumi sprindi planeerimise koosolekuga, kus pannakse paika järgmise iteratsiooni eesmärgid. [16]

Vaatluse all oleva projekti jaoks rakendatakse mõlema komponendi omadusi, kuid unikaalsel viisil.

### 4.1.2 Tellimustööde hindamisprotsessi parandamine

Meeskonnatöö probleemidega on seotud ka tellimustööde hindamise läbiviimine. Olemasolevas arendusprotsessis on mahuhinnangute andmise alamprotsess iteratsiooni osa, mis viiakse läbi iga tsükli alguses. Iteratsiooni käigus loodava järgmise tarkvara versiooni jaoks koostatakse nimekiri töödest ning alles siis viiakse läbi mahtude hindamised. Iteratsioonide ajal tellitakse ka töid, mis pole ühegi iteratsiooni osa veel. Sageli esinevad olukorrad, kus valmisarendatud tööd jäävad mitmeteks kuudeks ootele ning kaotavad oma aktuaalsuse. Sellise lähenemisega on tarkvaraarenduse fookus hajutatud ning ajaliselt väga ebaefektiivne.

Teise probleemina saab välja tuua asjaolu, et meeskonna liikmed annavad tellimustööde mahuhinnanguid teineteisest sõltumata. Meeskonnaliikmed ei suhtle omavahel ning sageli antakse valedel eeldustel hinnanguid ning selle tulemusena jääb töö realiseerimisel ajast puudu.

Probleemide lahendamiseks võeti vastu otsus pidada vastavalt vajadusele tellimustööde hindamise koosolekuid, kus arendusmeeskond saab kokku ning arutab omavahel läbi tööde lahendused ja keerukused ning selle alusel koostab mahuhinnangu. Koosolekutel võetakse kõigi meeskonnaliikmete arvamusi arvesse ning tekib suurem ülevaade projekti tulevaste tööde kohta. Tellimustööde hindamise koosolekud toetavad ka meeskonna kollektiivomandi põhimõtet, mis pärineb XP arendusmetoodikast. Põhimõte väidab seda, et kõikidel meeskonnaliikmetel on õigus vaadata ja muuta teineteise kirjutatud koodi. Põhimõte avaldub kõikidel tarkvara artifaktidel. [17] Kollektiivomandi põhimõtte suuremamahulisem rakendamine toimub järgmises alampeatükis.

### 4.1.3 Süsteemialase kompetentsi probleemide lahendamine

Süsteemialase kompetentsi edasiandmise probleemid avalduvad peamiselt arendusmeeskonna liikmete töö liigses piiritlemises. Käsitluse all oleva projektiga ühinedes oli välja kujunenud kindel printsiip, et tarkvaraarendajad tegelevad ainult veebirakenduste ja sellega seotud tarkvara arendamisega, jättes andmemudeli ja andmebaasi funktsionaalsuse arendamise täielikult enda skoobist välja. Andmebaasiga seotud arendustega tegeles üksnes andmebaasi administraator. Sarnane piiritlemine takistab süsteemialase ülevaate paremat omandamist, kuna teatud kompetents kuulub ainult ühel meeskonna liikmel.

Süsteemialase kompetentsi probleemide lahendamiseks rakendati meeskonnas kollektiivomandi põhimõtet. Projektis võeti vastu otsus, et tarkvaraarendajad hakkavad realiseerima ka andmebaasiarendusi ning tegema selles osas koostööd andmebaasi administraatoriga. Töö valmides kontrollib andmebaasi administraator üle kirjutatud andmebaasi muudatused ning hindab, kas tegu on kõige parema lahendusega. Kui lahendus ei sobi, siis viib andmebaasi administraator läbi parandused või suhtleb töö realiseeritud tarkvaraarendajaga ning pakub talle alternatiivseid lähenemiseviise. Seeläbi paraneb oluliselt nii süsteemialane kompetents meeskonnaliikmete vahel kui ka üleüldine tarkvaraarendajate tehniline võimekus ja silmaring.

### 4.1.4 Iteratsiooni tulemused

Esimese iteratsiooni saab lugeda õnnestunuks. Kõik iteratsiooni alguses püstitatud eesmärgid ja probleemid leidsid lahenduse. Iteratsiooni tulemusena realiseeriti järgnev:

- iganädalaste ühiste kliendikoosolekute pidamine;
- mahuhinnangute andmise koosolekute pidamine;
- ühise suhtluskeskkonna loomisega.

Realiseeritud paranduste tulemusena:

- meeskonnal kasvas ülevaade andmemudelist;
- tarkvaraarendajad suutsid palju paremini koosolekutel kaasa rääkida süsteemi disaini osas;
- oluliselt vähenes tellimustööde vääriti mõistmine, kuna iganädalastel koosolekutel käsitleti teemasid korduvalt ning küsiti infot hetkeolukorra kohta;
- tellimustööde mahuhinnangutest ei mindud enam nii suures mahus üle kui varem.

Parandused realiseeriti ja nende tulemusi uuriti 2 kuu vältel. Perioodi alguses ühines projektiga ka uus tarkvaraarendaja. Uus tarkvaraarendaja suutis suhteliselt kerge vaevaga projekti sisse elada ning omas üsna varakult ülevaate antud ajahetkel kriitilisest süsteemi funktsionaalsusest, mida hakati lähiajal arendama.

Kuigi iganädalased koosolekud aitavad paremini meeskonnatööd koordineerida ja see on parandanud meeskonna omavahelist suhtlust, jäi siiski iteratsiooni lõppedes püsti tarkvaratööde planeerimise probleem. Iteratsioonide jaoks tööde prioritiseerimine ja uute tarkvara versioonide ette valmistamine on endiselt häiritud. Toote omanik on sunnitud sageli tööde prioriteete hindama ning iteratsiooni plaan muutub iga nädala jooksul. Peamisteks põhjusteks on varjatud tehnilise keerukuse ja testimise mahu tekkimine, millega ei suudeta tellimustööde hindamise käigus alati arvestada. Seda probleemi üritatakse lahendada järgmises iteratsioonis.

Käesoleva iteratsiooniga loodi hea põhi tarnete planeerimise ja tellimustööde korraldamise parandamise jaoks, mida käsitletakse järgmistes iteratsioonides.

## 4.2 II iteratsioon

Teises iteratsioonis keskendutakse tellimustööde planeerimise ja tellimustööde arenduste haldamise peale. Tegu on kriitiliste valdkondadega, millest sõltub tarkvara uute versioonide koostamise kiirus ja riskitegur. Distsiplineeritud agiilse üleandmise metoodikast tulenevalt on väga oluline tarkvaraarenduse liigsete kulude minimeerimine. Oluline on tarnida ainult kõige olulisem lähtudes prioriteetidest ning jätta tegemata tööd, mis ei too lõppkasutajale lisaväärtust. Sellest tulenevalt on tarkvaraarendustsüklites üleval pidev tööde planeerimise ja prioritiseerimise probleem. Tellimustööde prioriteedid muutuvad kas ärivajaduste muutumise pärast, varjatud süsteemi keerukuse väljatulemise pärast ja/või suure testimise mahu pärast.

Uute tarkvara versioonide ettevalmistamise muudab ebamugavaks ka pidev käsitöö. XP arendusmetoodika rakendab *Continuous Integration* praktikat, mille abil toimub pidev uute tarkvara versioonide automaatne ettevalmistamine ja paigaldamine testimiseks mõeldud keskkonda.

Käesoleva iteratsiooni eesmärkideks on:

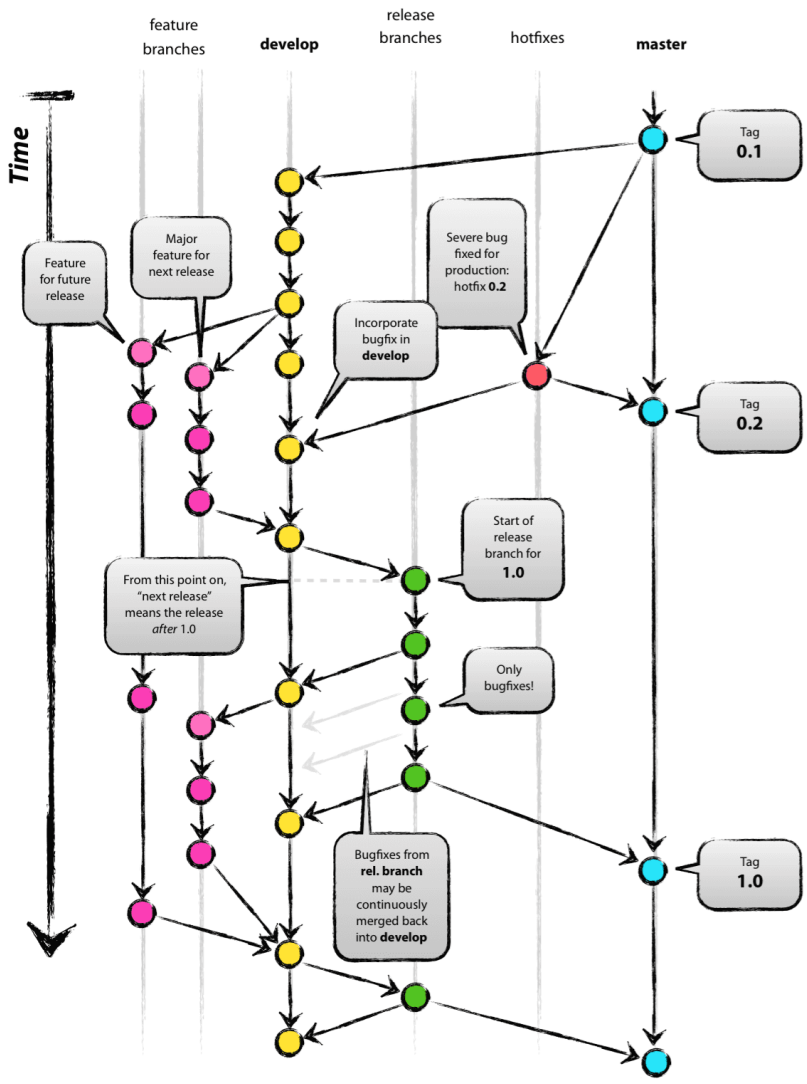
- parandada uute arenduste haldamist;
- automatiseerida uute tarkvara versioonide koostamine ja paigaldamine

Seeläbi paraneb oluliselt tarkvara arenduste haldamine ning seeläbi kõrgeneb tarnitava tarkvara kvaliteet. Agiilsete tarkvaraarendusprojektides peetakse õiget tarnimise strateegiat üheks kõige olulisemaks edufaktoriks [16].

### 4.2.1 Arenduste haldamise parandamine

Tarkvaraarendusprojektides kasutatakse reeglina versioonihalduse süsteeme tarkvara lähtekoodi ja uute arenduste haldamiseks. Esialgses tarkvaraarendusprotsessis puudus väljatöötatud mudel, mille järgi arenduste haldamine toimuks. Uued arendused ühendati omavahel versioonideks manuaalselt ning ebakorrapäraste võtete abil. Testimise faas toimus ka ebakorrapäraselt iga uue iteratsiooniga. Selle tulemusena anti sageli üle vigadega uusi tarkvara versioone. Tarnitud vead pidi tasuta ära lahendama, mille tõttu kannatas ka projekti kasumlikkus.

Arenduste paremaks haldamiseks otsustati rakendada Vincent Driessen'i väljatöötatud arenduste haldamise mudelit, mis põhineb versioonihaldussüsteemil Git. Joonis 4.2 illustreerib tarkvara arenduste haldamise puud. [12]




**Joonis 4.2. Vincent Driessen'i versioonihalduse mudel**

Vincent Driessen'i versioonihalduse mudelit iseloomustavad järgmised omadused:

- Põhiharudeks on *master* ja *develop* haru. *Master* haru peegeldab toodangu keskkonnas olevat süsteemi seisu ning *develop* haru peegeldab järgmise tarkvara versiooni seisu.
- *Feature* harud luuakse *develop* haru alusel ning peegeldavad uusi arendusi. Harudel on küljes tellimustööde tunnused.
- *Release* harud peegeldavad uusi üleandmiseks valmisolevaid arenduse seise. Samuti luuakse *develop* haru alusel.
- *Hotfix* harud peegeldavad kriitilisi veaparandusi, mis peab iteratsioonide vahel teostama. Põhinevad *master* harul.

Vincent Driessen'i versioonihalduse mudel võimaldab mugaval viisil viia läbi iteratiivset arendust. Uus versioonihalduse mudel võimaldab realiseerida ka *Continuous Integration* lahenduse, mis kaotab ära tarkvara uute versioonide koostamisel tehtava käsitöö.

### 4.2.2 Iteratsiooni tulemused

Teise iteratsiooni käigus realiseeriti järgnev:

- võeti kasutusele Vincent Driessen'i versioonihalduse mudel, mille abil üritati muuta arenduste haldamist mugavamaks ja efektiivsemaks;
- võeti kasutusele *Continuous Integration* lahendus, mis likvideeris suurel hulgal käsitööd uute tarkvara versioonide koostamisel.

Realiseeritud paranduste tulemusena:

- paranes üleantavate tarkvaraversioonide haldamine ja struktuur;
- vähenes uutel versioonidel ilmnevate vigade arv;
- paranes üleüldine tarkvara kvaliteet.

Parandused realiseeriti 3 kuu vältel ning selle käigus viidi läbi 5 tarkvaraarenduse iteratsiooni. Kuigi tarkvara kvaliteet paranes, ilmnes kõigi läbiviidud tarkvaraarenduse iteratsioonide käigus prioriteetide ja ärivajaduste ümberhindamisi, mille tulemusena kannatas uute tarkvara versioonide kvaliteet. Esines olukordi, kus planeeriti kahte järjestikust tarnet, mille järjekord

ning sisu vahepeal muutusid. Meeskonna hinnangul Vincent Driessen'i versioonihalduse mudel ei sobi ideaalselt käsitluse all oleva organisatsiooni ja projekti olemusega. Testimise korraldamises ilmnes samuti veel puudujääke. Nimetatud probleemid käsitletakse kolmandas iteratsioonis.

## 4.3 III iteratsioon

Eelmises iteratsioonis rakendati arendusprotsessis Vincent Driessen'i versioonihalduse mudelit, mis ei andnud ideaalseid tulemusi protsessi parandamisel. Kuigi mudel võimaldas realiseerida automaatikat, muutis tööde haldamist selgemaks, vähendas käsitööst tingitud vigu ning parandas üleüldist kvaliteeti, ei võimaldanud pakutud mudel tarneid eriti mugavalt ümber planeerida ega nende järjekorda muuta, mis võib paljude ettevõtete puhul esineda. Leidmaks universaalsema lahenduse tarkvaraarendusprojektide laiemale spektrile, on vaja versioonihalduse mudelit veel ümberdisainida. Käesoleva iteratsiooni eesmärgiks on:

- välja töötada versioonihalduse mudel, mis võimaldab tarkvara versioone mugavamalt ümber planeerida.

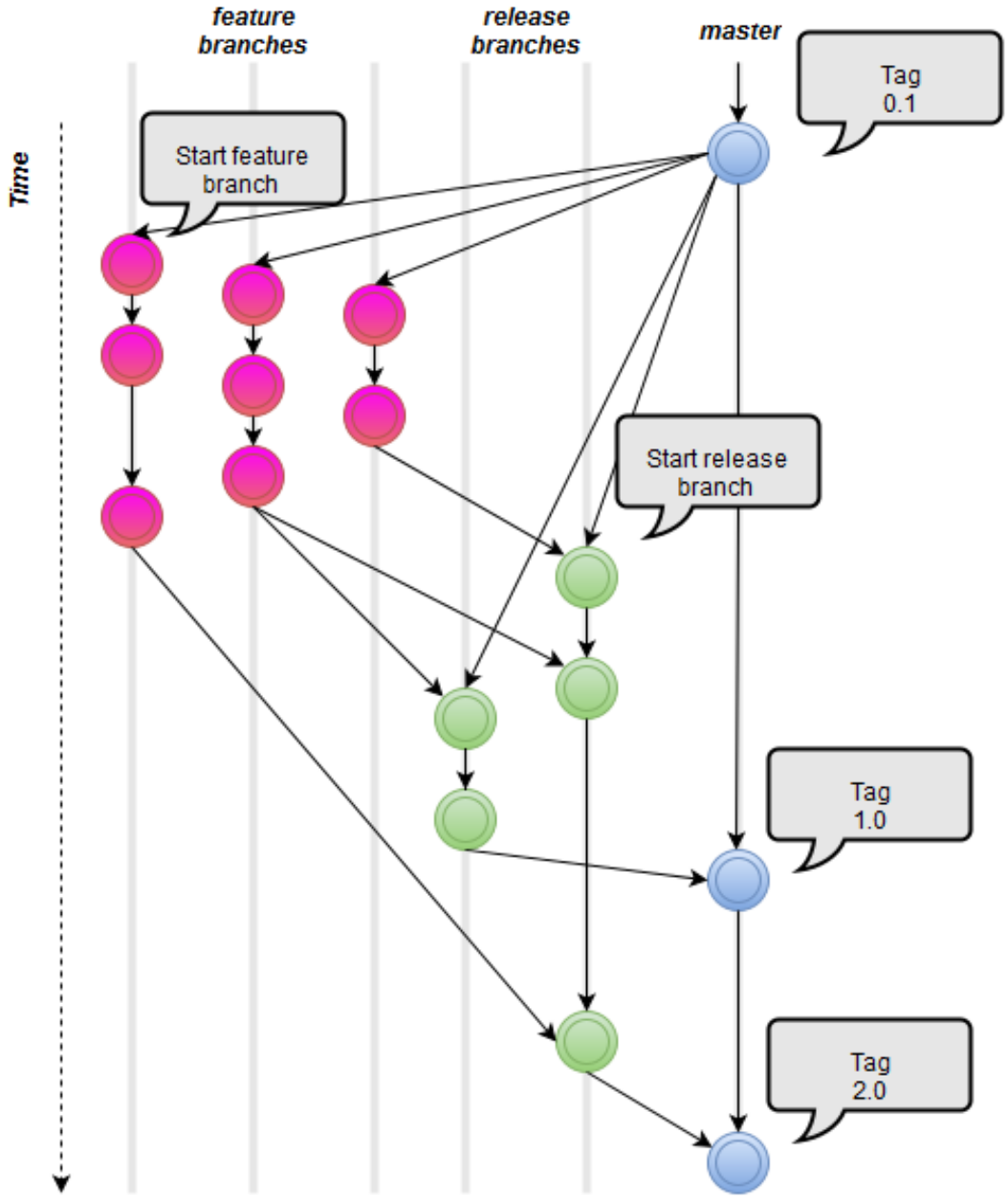
### 4.3.1 Arenduste haldamise parandamine

Eelmises iteratsioonis pakutud versioonihalduse mudeli peamiseks puudujäägiks oli *develop* haru staatilisus. Mudel sobib ideaalselt puhtamal kujul XP-d või Scrumi harrastavale meeskonnale, kuna nende metoodikate puhul pannakse iteratsiooni alguses paika kindel tööde nimekiri, mis peab iteratsiooni lõpuks valmis saama ning see nimekiri iteratsiooni vältel ei muutu. Kui on tegu aga muutliku ärilise kliimaga ettevõttega, siis isegi lühemate perioodide, nagu nädalate lõikes pole võimalik kindlat plaani paika panna.

Käesoleva iteratsiooni käigus pakuti välja uus versioonihalduse mudel. Mudeli disainimisel lähtuti Vincent Driessen'i mudelist ning teostati teatud lihtsustusi, mis teeb mudeli kergemini rakendatavaks erinevat tüüpi ettevõtete jaoks. Lihtsustatud mudelit iseloomustavad järgmised omadused:

- Ainsaks põhiharuks on *master* haru, mis peegeldab toodangu keskkonnas olevat süsteemi seisu.
- Tellimustööde arendused paigutatakse *feature* harudesse, mis luuakse *master* haru baasil.

❖ Tarned ja uued tarkvara versioonid koostatakse *release* harudesse, mis luuakse samuti *master* haru baasil. Töös võib olla mitu *release* haru paralleelselt, seega on võimalik tarneid igal ajahetkel ümber planeerida. Kui on vaja tarnida kriitilisi veaparandusi iteratsioonide vahel, siis erinevalt Vincent Driessen'i mudelist, käsitletakse neid samuti tavapäraste tarnetena ning nende kohta luuakse eraldi *release* harud.




**Joonis 4.3. Lõplik versioonihalduse mudel**

Joonisel 4.3. on välja toodud uus pakutud versioonihalduse mudel. Väljapakutud mudelit on võimalik rakendada ka puhtal kujul Scrumi, XP'd või mõnda muud agiilset metoodikat kasutavas projektis. *Continuous Integrationi* automaatikavahendeid on võimalik seadistada ka selliselt, et võetakse arvesse kõige uuem *release* haru.

Uus versioonihalduse mudel võimaldab kerge vaevaga tarneid järjekorra poolest ümber tõsta. Selle võimekuse tekkides otsustas arendusmeeskond ka uute tarkvara versioonide testimist ümber korraldada. Kuna tarneid peab üsna sageli ümber planeerima ja ümber tõstma, siis ajaliselt on mõistlikum ja efektiivsem viia testimist läbi vahetult peale tarne kokku panemist. Esialgses protsessis viidi läbi üksikute tellimustööde testimisi ka eraldi, kuid tarnete pideva ümberplaneerimise ja funktsionaalsuse erineva kombineerimise tulemusena tekkisid tarnitavasse uude versiooni ikkagi vead, mis tulid hilisemates staadiumites välja. Väljapakutud lahenduse järgi viiakse testimist läbi ainult siis, kui tarne koosseis on lõplikult kinnitatud ning kokku pandud. Sellisel juhul pühendab meeskond testimise ressurssi ainult versioonidele, mis antakse reaalselt tellijale üle.

### 4.3.2 Iteratsiooni tulemused

Kolmanda iteratsiooni käigus realiseeriti järgnev:

- arendati välja uus versioonihalduse mudel tellimustööde haldamiseks;
- korraldati ümber testimise protsess (testitakse ainult täielikult lõplikke tarkvara versioone, kus on kõik tellimustööd sees).

Uue mudeli koostamisel lähtuti teises iteratsioonis pakutud Vincent Driessen'i mudelist ning viidi läbi lihtsustusi, mis muutis mudeli universaalsemaks. Uut mudelit rakendati projekti 4 kuu ulatuses, mille käigus viidi läbi 6 tarkvaraarenduse iteratsiooni. Nii nagu ka teise iteratsiooni käigus, toimus tihti tarnete ümberplaneerimisi ning prioriteetide muutusi. Uus mudel võimaldas tarneid ümberplaneerimiste käigus oluliselt paremini hallata. Iteratsiooni tulemusena:

- suurenes oluliselt tarnitava tarkvara kvaliteet;
- vähenes eksimuste arv;
- paranes tarnete planeerimise võimekus.

Viimase iteratsiooniga täideti kõik arendusprotsessi parandamise eesmärgid, mis projekti alguses püstitati. Arendusprotsessi parandamise tulemusena arendati välja versioonihalduse mudel, mida on võimalik rakendada laiale ettevõtete spektrile.

## 4.4 Kokkuvõte

Neljandas peatükis viidi projektis läbi tarkvaraarendusprotsessi parandused. Parandusi planeeriti viia läbi kahes etapis. Esimene etapp sisaldas meeskonnatööalaste probleemide lahendamist ning teine etapp tellimustööde planeerimise ja haldamise parandamist. Paranduste tulemusena valmis uus versioonihalduse mudel, mis on võimalik rakendada erineva ärilise kliimaga ettevõtetes, nii stabiilsetes kui ka muutlikes organisatsioonides. Järgmises peatükis formaliseeritakse parandatud arendusprotsess SPEM 2.0 metamudeli kujul, mis tähistab tegevusliku disainuuringu kontekstis õppimise formaliseerimise faasi. Selle tulemusena valmib universaalne protsessi mudel, mis on rakendatav paljudes erinevat liiki organisatsioonides.

# 5 Parandatud arendusprotsessi mudel ja analüüs

## 5.1 Arendusprotsessi mudeli täiendused

Järgnevalt on välja toodud arendusprotsessi mudelisse lisandunud komponendid. Protsessi komponendid parandamise käigus enamjaolt ei muutunud. Peamised muudatused olid protsessi ülesehituses, mida käsitletakse ja analüüsitakse peatükis 5.2.

### 5.1.1 Rollid

**Tabel 5.1. Paranduste käigus lisandunud rollid**

| Roll | Tarkvaraarendusmeeskond |
|---|---|
| Lühikirjeldus | Süsteemi arendusega tegelev inimeste grupp |
| Põhikirjeldus | Tarkvaraarendusmeeskond on grupp inimesi, kelle ülesandeks on tegeleda süsteemi arendamisega ja täita tellimusi. Tarkvaraarendusmeeskond koosneb tarkvaraarendajatest, andmebaasi administraatorist, süsteemianalüütikust/arhitektist, testijast ja meeskonna juhist. Meeskond kollektiivselt vastutab toote arendamise ja sellega seotud tööde läbiviimise eest. Tarkvaraarendusmeeskond annab hinnanguid tellimustöödele ning suhtleb tellijaga kollektiivselt. Meeskonnaliikmete vahel toimub igapäevane pidev koostöö ning infovahetus. |

### 5.1.2 Ülesanded

**Tabel 5.2. Paranduste käigus muutunud ülesanded**

| Ülesanne | Tellimustöö hindamine |
|---|---|
| **Kirjeldus** | Tellimustöö hindamist viivad läbi kõik tarkvaraarendusmeeskonna liikmed, mille tulemusena kujuneb koondav hinnang. Koondava hinnangu alusel langetab toote omanik otsuse, kas kulutada ressursse töö realiseerimiseks või mitte. Tellimustöö hinnang saadetakse toote omanikule kinnitamiseks. Hinnangut mõjutab ka olemasoleva süsteemi disain, mida teeb ülesande kas keerulisemaks või lihtsamaks. |
| **Eesmärk** | Eesmärgiks on välja selgitada tellimustöö realiseerimiseks vajalik ajakulu, mis aitab meeskonnas aega planeerida ja toote omanikul otsuseid langetada. |

| | | | |
|---|---|---|---|
| **Sisend** | Tellimustöö<br>Süsteemi disain | **Väljund** | Tellimustöö mahuhinnang |
| **Rollid** | *Primaarne tegutseja* | Tarkvaraarendusmeeskond | |
| | *Kõrvaline tegutseja* | Süsteemi administraator | |
| **Ülesanne** | **Uue tarkvara versiooni testimine** | | |
| **Kirjeldus** | Uue tarkvara versiooni testimisel kontrollitakse üle, kas kõik tarnitav funktsionaalsus toimib korrektselt. Tarnes sisalduvad tellimustööd testitakse ükshaaval üle ning kontrollitakse ka nende koosmõju korrektset funktsioneerimist. Testimise tulemusena luuakse testimise aruanne, mis lisandub tarne tulemuste koosseisu. | | |
| **Eesmärk** | Eesmärgiks on tagada tarkvara tarne kvaliteet. | | |
| **Sisend** | Uus tarkvara versioon | **Väljund** | Uue tarkvara versiooni testimise aruanne |
| **Rollid** | *Primaarne tegutseja* | Testija | |
| | *Kõrvaline tegutseja* | Tarkvaraarendusmeeskond | |
| **Ülesanne** | **Tellimustöö arendamine** | | |
| **Kirjeldus** | Tellimustööd hakkavad arendama erinevad tarkvaraarendusmeeskonna liikmed sõltuvalt töö iseloomust. Tellimustöö realiseerimise ja selle tulemuste eest vastutab üheselt terve arendusmeeskond. Töö realiseerimise osas on kõigil meeskonnaliikmetel vastutus anda panus parima lahenduse saavutamiseks. Meeskonna liikmetel pole määratud selgeid piire skoobi osas. Tarkvara arendajad võivad realiseerida ka andmebaasi arendusi ning vastupidi, kuid töö valmimisel peavad vastavate skoopide spetsialistid tehtud töö üle vaatama enne üleandmist. Kui tellimustöö sisaldab mitmeid skoope, siis meeskonna liikmed teevad tihedat koostööd, et kõigi realiseeritavad osad oleksid kooskõlas. | | |
| **Eesmärk** | Eesmärgiks on luua uut funktsionaalsust või parandada olemasolevat. | | |
| **Sisend** | Tellimustöö | **Väljund** | Tarnitav tellimustöö |
| **Rollid** | *Primaarne tegutseja* | Tarkvaraarendusmeeskond | |
| | *Kõrvaline tegutseja* | – | |

### 5.1.3 Töötulemid

**Tabel 5.3. Paranduste käigus lisandunud töötulemid**

| Töötulem | Kirjeldus |
|---|---|
| Uue tarkvara versiooni testimise aruanne | Testija koostab uue tarkvara versiooni testimise tulemusena raporti, mis kirjeldab testimise tulemusi. Testimise aruanne antakse tarnitava tarkvara versiooniga koos tellijale üle. |

### 5.1.4 Juhised

**Tabel 5.4. Paranduste käigus lisandunud juhised**

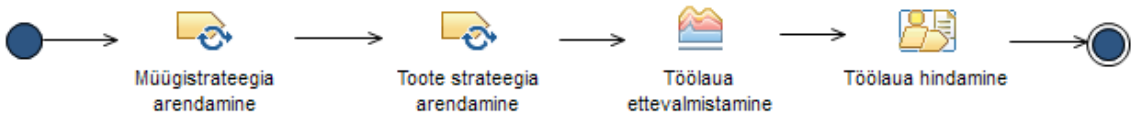
| Juhis | Kirjeldus |
|---|---|
| Tellimustööde hindamise koosolek | Tellimustööde asjakohaseks hindamiseks viiakse läbi koosolekuid vastavalt vajadusele. Koosolekutest võtab osa terve arendusmeeskond. Hindamisekoosolekul vaadatakse üle kogunenud uute tellimustööde nimekirja ning ühiselt koostatakse ajalised töömahu pakkumised tellijale. |
| Iganädalased arenduskoosolekud | Arendusmeeskond saab iga nädal kokku toote omanikuga. Koosoleku eesmärgiks on vahetada tootearenduse juhtkonna ja arendusmeeskonna vahel infot. Koosolekud annavad juhtkonnale parema selguse arendusmeeskonna võimekusest ning senisest progressist. See aitab kaasa toote strateegia planeerimisele. |
| Versiooni halduse mudel | Kasutusele võetud uus versioonihalduse mudel, mis võimaldab efektiivsemalt arendusi hallata ja planeerida. (vt. peatükk 4.3.1) |

## 5.2 Arendusprotsessi analüüs ja võrdlus esialgsega

Tarkvaraarenduse äriprotsessis viidi läbi muudatused toote planeerimise ja arenduse iteratsioonide osas. Esialgses protsessis kasutusel olnud etapid jäid suures osas samaks. Peamiselt muudeti etappide järjekorda ja lisati mõned etappide käigus rakendatavad praktikad. Järgnevalt on välja toodud protsessi muudatused, mis rakendati protsessi parendamise käigus.

### 5.2.1 Toote planeerimise protsess

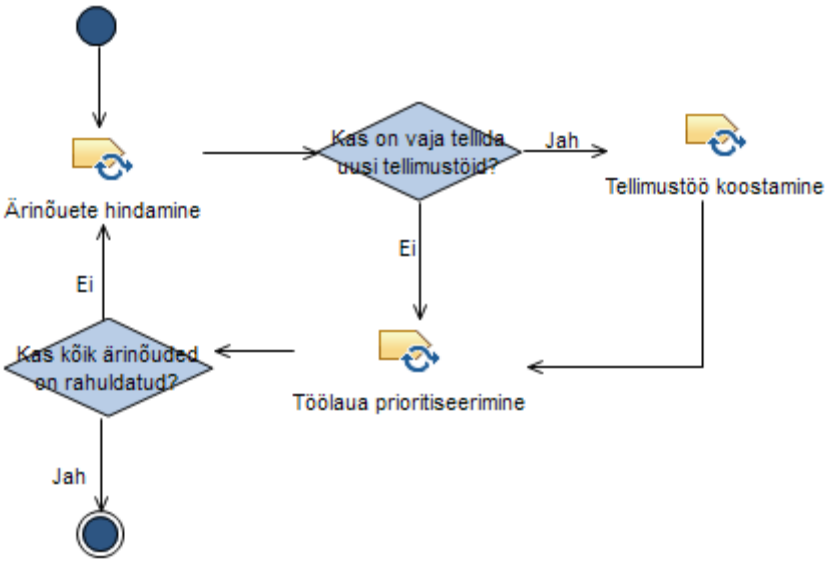
Esialgses arendusprotsessis arendati toote planeerimise faasis üksnes strateegiaid ja tulevikuplaane. Toote planeerimise faas kordub pidevalt, ka iteratsioonide ajal. Ärinõuete hindamine ja nende alusel tellimuste koostamine viidi aga läbi alles arendusiteratsioonide alguses. Parandatud protsessis on töölaua ettevalmistamine ja selle hindamine viidud toote planeerimise faasi (vt joonis 5.1.).




**Joonis 5.1. Toote planeerimise protsess**

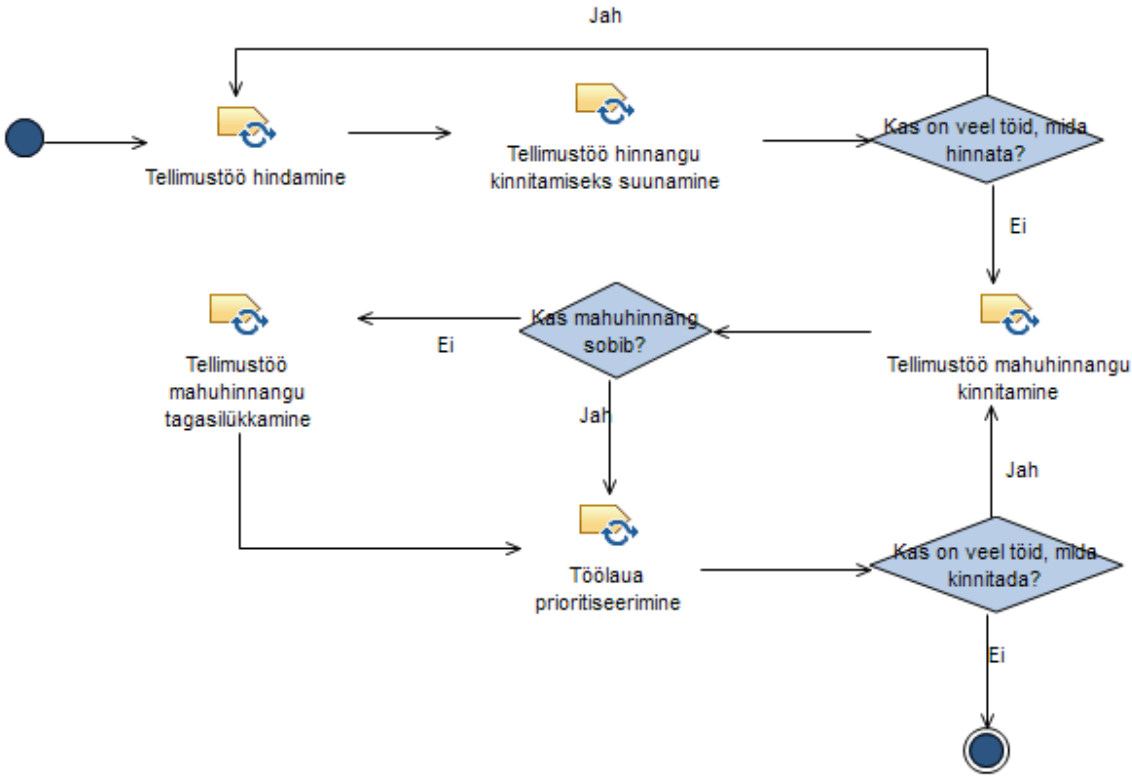
Osapoolte tihedam suhtlemine aitab parandada tootearenduse efektiivsust ja tulemuslikkust. Selle saavutamiseks viiakse parandatud protsessi kohaselt läbi iganädalasi meeskonna koosolekuid toote omanikuga, mille käigus saavad kõik osapooled parema ülevaate projektis toimuva üle.

Töölaua ettevalmistamise tõstmine toote planeerimise faasi aitas oluliselt suurendada planeerimise paindlikkust. Pidevalt, ka iteratsioonide läbiviimise ajal saab hinnata ärivajadusi, võtta arvesse vahepeal muutunud olukordasid ning ümberprioritiseerida. Äriprotsessides on paindlik reageerimine väga oluline, kuna see võimaldab mugavamalt likvideerida ebaefektiivsusi ja puudujääke. Joonis 5.2 illustreerib töölaua ettevalmistamise protsessi.




**Joonis 5.2. Töölaua ettevalmistamise alamprotsess**

Erinevalt esialgsele protsessile, kus töölauda hindasid arendusmeeskonna liikmed eraldi ning üksikute tellimustööde kaupa, hindavad lõplikus protsessis tellimustöid meeskonnaliikmed kõik koos. Töölauale kogunevad tööd võetakse päevakorda vastavalt tellija prioriteetidele. Koosolekutel räägivad arendusmeeskonnaliikmed omavahel võimalikud lahendused ja lähenemised läbi juba tööde planeerimise käigus. See võimaldab juhtkonnal juba tootearenduse varasemates staadiumites langetada otsuseid ning kohendada strateegiaid. Joonis 5.3 illustreerib tellimustööde ettevalmistamise protsessi.

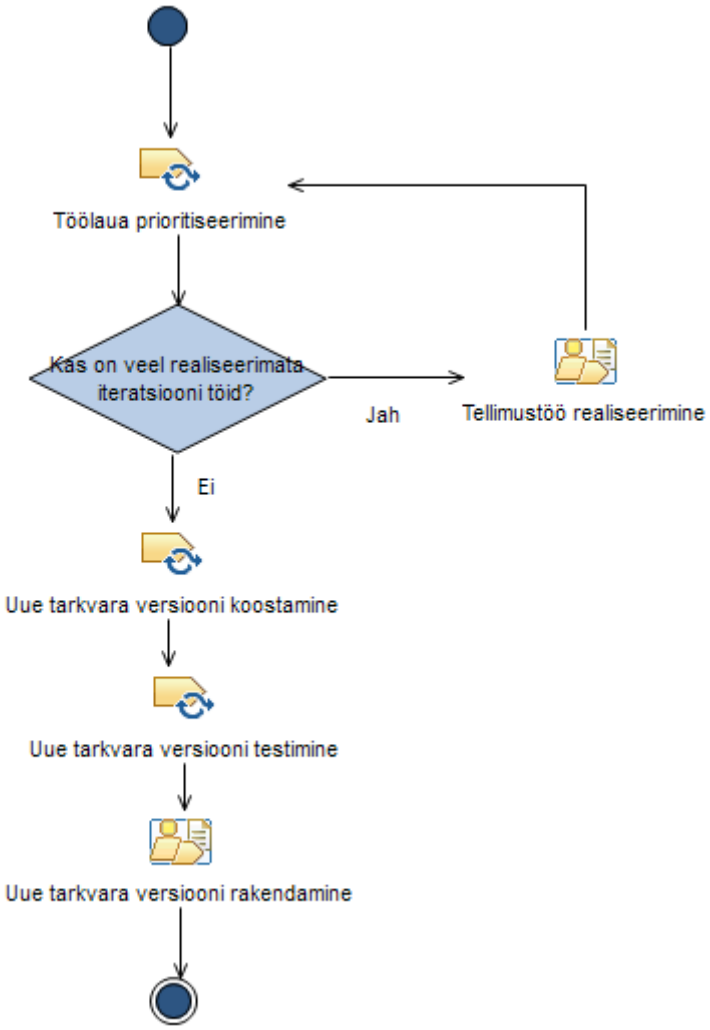



**Joonis 5.3. Tellimustööde ettevalmistamise protsess**

### 5.2.2 Iteratsiooni protsess

Iteratsiooni protsessi peamisteks muudatusteks on tellimustööde planeerimise likvideerimine iteratsiooni töötsüklist ning uue tarkvara versiooni testimise etapi lisamine (vt joonis 5.4). Iteratsiooni faasis kasutatakse uut väljatöötatud versioonihalduse mudelit, millega koos võeti kasutusele ka *Continuous Integration* lahendus, mis võimaldab automaatselt uusi tarkvara versioone kokku ehitada ja rakendada. Uus versioonihalduse mudel võimaldab ka mugavamalt hallata ning paindlikumalt planeerida tarkvara tellimustöid.

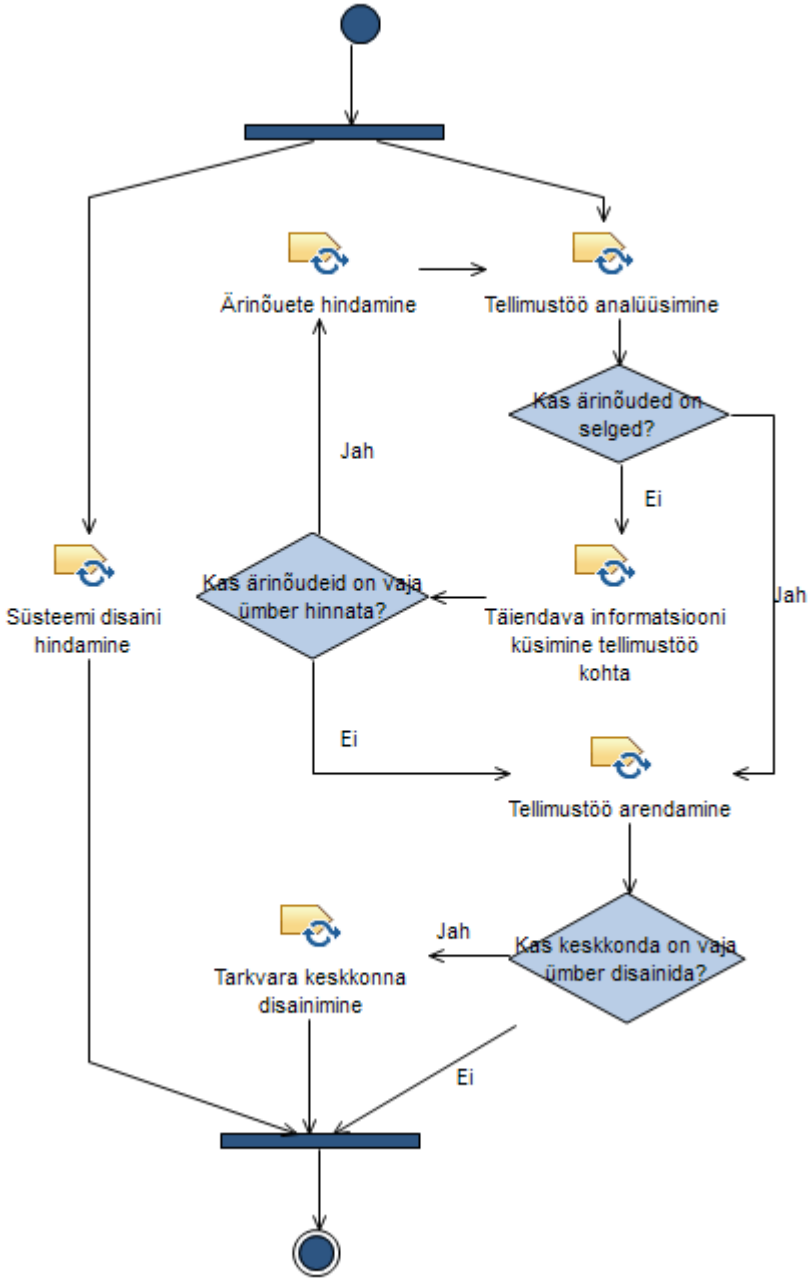
Peatükis 4.3.1 on dokumenteeritud arendusprotsessi parendamise viimane iteratsioon, mille käigus töötati uus mudel välja. Väljatöötatud mudel avas uued automatiseerimise võimalused, mida varasemalt polnud võimalik realiseerida. Mõju avaldus peamiselt toote planeerimise faasis, kuna mudel aitab paremini tagada uue koostatava tarkvara versiooni kvaliteedi sõltumata sellest, kuidas juhtkond soovib tellimusi ümber prioritiseerida.




**Joonis 5.4. Iteratsiooni protsess**

Iteratsiooni faasi lõppu lisandus uue tarkvara versiooni testimine. Esialgses arendusprotsessis testiti tellimustöid teineteisest eraldatud tingimustes, mille tulemusena tekkis tööde kombineerimisel vigu, mida polnud võimalik varasemates staadiumites avastada. Parandatud protsessis viiakse testimist läbi ainult siis, kui uue tarkvara versiooni koosseis on kindlalt paigas ning kõik tööd on realiseeritud ja kokku tõstetud. Selline lähenemine väldib töö mitmekordset tegemist, mis avaldaks negatiivset mõju projekti kasumlikkusele. Esialgses protsessis kulutati testimisele sageli mitmekordselt rohkem aega kui oli planeeritud. Kuna uus tarkvara versioon läbib mitu valideerimise kihti ning on mitmekordselt kontrollitud, siis tarkvara kvaliteet selles osas ei kannata. Üksnes aktuaalsete versioonide testimine tagab projekti ressursside efektiivse rakendamise. Läbiviidud muudatused on kooskõlas ka *lean development* põhimõtetega.

Joonis 5.5 illustreerib tellimustööde realiseerimise protsessi. Tellimustööde realiseerimine jäi parendamise tulemusena suures osas samaks. Üksikute tellimustööde testimise faas eemaldati ning asendati aktuaalse uue tarkvara versiooni testimise faasiga iteratsiooni lõpus.




**Joonis 5.5. Tellimustöö realiseerimise alamprotsess**

### 5.2.3 Eesmärkide täitmine

Tabel 5.5 peegeldab tarkvaraarendusprotsessi parandamise tulemusi protsessi eesmärkidega.

**Tabel 5.5. Töö tulemused eesmärkide suhtes**

| Eesmärk | Töötulem |
|---|---|
| Kasumlik äritegevus | • Arendusmeeskond suutis rohkem tellimustöid realiseerida, mille tulemusena kasvas organisatsiooni käive<br>• Automatiseerimise tulemusena minimeeriti kulusid |
| Kõrge tarkvara kvaliteet | • Automatiseerimise tulemusena tagati kõrgem kvaliteet<br>• Tellija tagasiside kohaselt oli tarkvara kvaliteet parandamise tulemusena oluliselt paranenud<br>• Uus versioonihalduse mudel minimeeris tarnitavate vigade arvu |
| Efektiivne ressursside kasutamine | • Ressursside kasutamine paranes, kuna mahuhinnangud muutusid täpsemaks ning tellimustööde mõistmine meeskonnaliikmete vahel paranes. |
| Vähe manuaalseid operatsioone | • *Continuous Integration* lahenduse realiseerimine |
| Efektiivne süsteemialase kompetentsi edasiandmine | • Meeskonna hinnangul paranes meeskonnasuhtluse keskkonna ja täiendavate koosolekute rakendamisega kompetentsi edasiandmise võimekus<br>• Uus meeskonnaliige suutis ilma suurema vaevata projekti sisse elada |
| Tellimustööde üheti mõistmine | • Tööde realiseerimise kiirus kasvas, kuna oli vähem arusaamatusi<br>• Realiseeriti rohkem tellimustöid kui varem |
| Täpsed mahuhinnangud | • Mahuhinnangutest peeti täpsemalt kinni<br>• Suhteline tellimustöö realiseerimise kulu vähenes |
| Efektiivne meeskonnasuhtlus | • Meeskonna ja tellija hinnangul oli kõikide osapoolte vaheline suhtlus oluliselt paranenud<br>• Vähem arusaamatusi |

# 6 Kokkuvõte

Käesoleva lõputöö eesmärgiks oli parandada ühe Tieto Estonia AS projekti arendusprotsessi. Ülesande püstitamise hetkel sisaldas arendusprotsess väga palju puudujääke ja ebaefektiivsusi. Tarkvara arendamise kiirus oli madal, uute tarkvara versioonide planeerimine ja realiseerimine võttis vastuvõetamatult kaua aega, meeskonnasuhtlus oli puudulik ning projekti üleüldine kasumlikkus oli liiga madal.

Lõputöö täiendavaks eesmärgiks oli luua kergesti loetav ja jälgitav protsessi mudel, mis illustreeriks üldist tarkvaraarenduse protsessi, mida saaks realiseerida paljudes projektides. Pöörati tähelepanu sellele, et mudel oleks lahendatud piisavalt abstraktsel tasemel ning et ta tooks välja tüüpiliste tarkvaraarendusprojektide omadused.

Lõputöös otsustati kasutada arendusprotsessi paranduste läbiviimisel SPI põhimõtteid koos tegevusliku disainuuringu metoodikaga. Kasutati ka äriprotsesside haldamise metoodikaid, protsessi analüüsimisel. Tarkvaraarendusprotsessi näol on tegu organisatsiooni ühe äriprotsessiga, mis allub äriprotsesside haldamise metoodikatele nii nagu iga teinegi protsess. Tegevuslik disainuuringu metoodika toetas universaalse protsessi väljatöötamise eesmärki.

Töö tulemusena töötati välja universaalne tarkvaraarendusprotsessi mudel, mida on võimalik rakendada tüüpilistes tarkvaraarendusprojektides. Protsess on piisavalt paindlik organisatsiooni suhtes. Mudel formaliseeriti protsessi veebisaidina, mis publitseerib protsessi erinevate osapoolte jaoks mõeldud vaadetena. Mudeli koostamisel lähtuti SPEM 2.0 metamudeli spetsifikatsioonist. Töö teiseks kaalukaks tulemuseks oli versioonihalduse mudeli väljatöötamine, mis kujunes arendusprotsessi tuumaks, mille najal tootearendus püsti seisab. Versioonihalduse mudelit on võimalik rakendada enamikes tarkvaraarendusprojektides.

Töö tulemusena võib järeldada, et tegevuslik disainuuring koos SPI (Tarkvara Protsesside Parendamise) põhimõtetega sobib väga hästi tarkvara arendusprotsesside parendamiseks. Töö autor täitis protsessi parendamise käigus nii praktiku (meeskonnaliikme), protsessiinseneri kui ka jälgija rolli (tegevusliku disainuuringu uurija). Töö näitab, et nimetatud praktikaid saab omavahel väga tulemuslikult kombineerida ning et teooriad teatud osas omavahel haakuvad.

# Summary

The aim of this thesis was to improve on of Tieto Estonia AS's project's software development process. There were many defects and inefficiencies in the process at the beginning of the project. The software development velocity was slow, the planning and implementation of new software releases took unacceptable amounts of time to complete, team chemistry was bad and the project's overall profitability was too low.

The secondary objective of this thesis was to create an easily readable and traceable process model that would illustrate a universal software development process, which is easily applicable in a wide variety of software development projects. The process model was supposed to be defined in a sufficiently abstract level. The model had to bring out typical software development project characteristics.

SPI (Software Process Improvement) principles along with Action Design Research was used to improve the process. Different business process management principles were also used for analysis. Software development processes are also business processes. The Action Design Research methodology successfully supported the goal to develop a universal process.

As a result of this thesis, a universal software development process model was developed that is applicable in typical modern software development projects. The process is flexible enough in relation to different organizations. The model was formalized as a process website that publishes the process in different views for different parties. SPEM 2.0 metamodel specification was used to assemble the process model. The second essential result of this thesis was the new version control model, which turned out to be the core of the product development process. The version control model is applicable in most software development projects.

It can be concluded that Action Design Research along with Software Process Improvement principles make a fine combination in software process improvement. During the project, the author assumed the role of practitioner (team member), process engineer and researcher (according to Action Design Research principles). This thesis shows that the named practices can be successfully combined and that the theories overlap in some aspects.

# Kasutatud kirjandus


[1] S. Ambler ja M. Lines, Disciplined Agile Delivery, Crawfordsville: R.R. Donnelley, 2012.

[2] *DRM: A Design Research Methodology*, 2002.

[3] A. Collins, D. Joseph ja K. Bielaczyc, „Design Research: Theoretical and," *The Journal of the Learning Sciences,* kd. 13, nr 1, pp. 15-42, 2004.

[4] M. K. Sein, O. Henfridsson, S. Purao, M. Rossi ja R. Lindgren, „Action Design Research," *MIS Quarterly,* kd. 35, nr 1, pp. 37-56, March 2011.

[5] M. Brydon-Miller, D. Greenwood ja P. Maguire, „Why action research?," *Action Research,* kd. 1, nr 1, pp. 9-28, 2003.

[6] F. Eileen, „Action Research," *Themes in Education,* 2000.

[7] *An Overview of the Methodological Approach of Action Research,* 1998.

[8] E. Ries, The Lean Startup, New York: Crown Business, 2011.

[9] T. T. Meum, „An Action Design to Developing Emergency Management Systems," Department of Information Systems, University of Agder, 2014.

[10] „Enterprise Agile: Software Process Improvement (SPI)," [Võrgumaterjal]. Available: http://www.enterpriseunifiedprocess.com/essays/softwareProcessImprovement.html.

[11] „Software Process Improvement (SPI) Best Practices," [Võrgumaterjal]. Available: http://www.ambysoft.com/essays/spiTips.html.

[12] A. Caplain ja B. Coulette, „Towards a Rigorous Process Modeling With SPEM," [Võrgumaterjal]. Available: http://people.rennes.inria.fr/Benoit.Combemale/research/phd/2006/iceis250406-CCCC-poster401.pdf.

[13] *Software & Systems Process Engineering Meta-Model Specification Version 2.0,* 2008.

[14] M. Dumas, „Äriprotsesside juhtimise kursus," [Võrgumaterjal]. Available: https://courses.cs.ut.ee/2016/bpm/.

[15] „Manifesto for Agile Software Development," [Võrgumaterjal]. Available: http://agilemanifesto.org/.

[16] S. Alliance, „The Scrum Guide," [Võrgumaterjal]. Available: https://www.scrumalliance.org/why-scrum/scrum-guide.

[17] D. Wells, „The Rules of Extreme Programming," 1999. [Võrgumaterjal]. Available: http://www.extremeprogramming.org/rules.html.

[18] M. Roach, „6 Top Success Factors for Agile Software Projects," *Software Alliance Wales,* 2015.

[19] V. Driessen, „A successful Git branching model," [Võrgumaterjal]. Available: http://nvie.com/posts/a-successful-git-branching-model/.