TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond
Informaatikainstituut
Infosüsteemide õppetool

IDU70LT
Mario Maisto 143665IABM

SAP ERP SÜSTEEMI MOODULILE TEHTAVA ARENDUSE HALDAMINE AVALIKU SEKTORI PROJEKTI NÄITEL

Magistritöö

Juhendaja: Karin Rava
MSc
Lektor

Tallinn 2017

Autorideklaratsioon

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

Autor: Mario Maisto

03.01.2017

Annotatsioon

Antud töö eesmärk on analüüsida avaliku sektori arendusprojekti (arendus ERP süsteemi moodulile) käigus tekkinud probleeme ning leida nende juur-põhjused, kasutades selleks põhjus-tagajärg metoodikat. Samuti on töö eesmärgiks positsioneerida projekti asukoht arendusmetoodikate skaalal (traditsiooniline metoodika või agiilne lähenemine) ning leida sobivaim töös uuritud probleemide lahendamist võimaldav arendusmetoodika. Samuti otsitakse töös lahendusi esitatud projekti probleemide juur-põhjustele, lähtudes valitud sobivaima arendusmetoodika võimalustest.

Magistritöö tulemusena selgus, et töös uuritud avaliku sektori projekt sobinuks agiilse arendusmetoodikaga, kus sobivaimaks osutus Scrum, mis võimaldanuks vältida või leevendada mitmeid arendusfaasis tekkinud probleeme. Täiendavalt tuli põhjus-tagajärg uuringust välja projektist puudunud põhjaliku riskianalüüsi vajalikkus, mis võimaldanuks paremini arvestada projekti käigus tekkinud tehnoloogiliste probleemide ning muutuvate nõuetega.

Lõputöö on kirjutatud eesti keeles ning sisaldab teksti 72 leheküljel, 6 peatükki, 9 joonist, 12 tabelit.

Abstract

Development Management for SAP ERP System Module Based on a Public Sector Project

The aim of this thesis is to analyse and find out the root-causes of the problems of public sector IT projects, that are limited by the requirements set by the public sector and which have large technical complexities due to the systems used. The thesis focuses on a public-sector project, of which the author was closely related to. By finding the root-causes of the problems, the aim is to provide solutions by analysing the nature of the project for finding applicable development methodologies and thus selecting the best suitable way to implement similar projects in the future.

The thesis described the main problems of the project, which were related to changing requirements, problems regarding technological complexities, managing scope creep without exceeding the time limit and difficulties caused by poor risk-management and lack of recourses. One of the reason these problems occur can be caused due to unsuitable choice of development methodology as well as overlooking of some crucial aspects when planning the project. Therefore, an analysis was conducted to highlight the main causes of these problems.

The results obtained with this master thesis show that the project in question should have been managed by an agile development methodology, more precisely with Scrum, which could have avoided or lessened the extent of multiple problems that occurred in the development phase. In addition, the problem-cause analysis highlighted the fact, that the issues with technological difficulties and changing requirements could have been lessened with a sufficient risk-analysis.

The thesis is in Estonian and contains 72 pages of text, 6 chapters, 9 figures and 12 tables.

Lühendite ja mõistete sõnastik

Agiilne arendusmetoodika	Tuntud ka kui väledad või paindlikud arendusmetoodikad on kontseptuaalse raamistikuna agiilse manifesti printsiipe järgivad tarkvara-arendusmetoodikad, mis põhineb iteratiivsel ja protsessi-kesksel arendamisel, kus nõuded ja lahendused arenevad läbi koostöö iseorganiseeruva meeskonnaga.
Agiilne manifest	2001 aastal 17 inimese poolt loodud agiilse tarkvara arendusmetoodika põhiväärtuste ja põhimõtete kogum (inglise keeles *Agile Manifesto*). Manifest koosneb neljast põhiväärtusest ning kaheteistkümnest põhimõttest.
Back-End	Põhi- (taga-) arvuti, põhi- (taga-) programm. "Tagarakenduse" (*back-end application*) või -teenuse ülesandeks on kaudselt toetada eesteenusi ning nad on tavaliselt lähemal vajalikule ressursile ja/või suudavad sellega suhelda.
DSDM	(*Dynamic Systems Development Method*) - Dünaamiline tarkvaraarendusmeetod on agiilse lähenemise perekonda kuuluv meetod, mis kirjeldab põhiprintsiibid, võimaldamaks saavutada soovitud funktsionaalsust väikse ajaga, kaasates kontrolli ning probleemide lahendamise komponendid.
ERP	(Ettevõtte ressursside planeerimine) - lai tegevuste ring, kus kasutatakse erinevaid rakendusprogramme ja mis aitab tootmis- või muul ettevõttel hallata olulisi ärikomponente nagu tooteplaneerimine, detailide sisseostmine, laoseisu haldamine, suhtlemine tarnijatega, kliendihaldus, tellimuste täitmise jälgimine jne. ERP võib sisaldada ka finants- ja inimressursside halduseks mõeldud tarkvara.
Extreme Programming	(lühend XP) - Agiilne arendusmetoodika, mis keskendub kvaliteetsele koodi kirjutamisele, kasutades selleks lühikesi iteratsioone, paarisprogrammeerimist, pidevat integreerimist ning testimisest tulenevat arendust (*TDD*).

Front-End	Eessüsteem, eeskomponent.Veebilehele ilmuv kasutajaliides, mis võimaldab veebisaidi külastajal kahepoolselt suhelda saidi dünaamiliste osadega nagu andmebaasid, ostukorviprogrammid ja ostutöötlustarkvara.
HTML5	HTML5 on ülemaailmse veebi tuumikkeele HTML viies põhiredaktsioon.
IKT	(Info- ja kommunikatsioonitehnoloogia) Andmete töötlemise, salvestamise ja edastamise tehniliste vahendite, meetodite ning võtete koondnimetus.
Ishikawa diagramm	Kaoru Ishikawa poolt loodud meetod, mida kasutatakse tänapäeval analüüsimaks probleemide tekkimise põhjuseid.
Iteratsioon	Töö-tsükkel, mille jooksul toodetakse töötav tarkvara.
Javascript	Netscape'i poolt välja töötatud skriptikeel, mis võimaldab veebiautoritel luua interaktiivseid veebisaite.
Kanban	Agiilne tõuke-tõmbe lähenemist järgiv arendusmetoodika, mis fokusseerib töövoo visualiseerimisele ning käimasoleva töö hulga piiramisele.
Kasutajalugu	(inglise keeles *User Story*) - Scrum ja teiste agiilsete arendusmetoodikate juures kasutatav mõiste, mis sätestab toote üksiku(te) funktsionaalsuse(-te) nõuded.
Kohanduv veebidisain	Seadme omadustele reageeriv disain. Erinevaid ekraani resolutsioone toetav disain.
Kristall-meetod	(inglise keeles *Crystal*-Method) - Agiilsete tarkvaraarendusmeetodite perekond, mille puhul arendatakse tarkvara ühiste põhimõtete järgi.
Läbipõlemisgraafik	(inglise keeles *Burndown Chart*) - näiteks Scrum arendusmetoodikas kasutatav graafik visualiseerimaks kogu töö ja tehtud töö suhet.
Püstijala koosolek	(inglise keeles *Daily Scrum Meeting*) - Scrum arendusmetoodika juures kasutatav mõiste, mis tähendab arendusmeeskonna-siseseid igapäevased lühikesi koosolekuid kus iga meeskonna liige räägib, mis on tema seatud eesmärgid ning millised on selle saavutamist takistavad probleemid.
SAP	Saksa firma SAP AG on asutatud 1972.a. ja kuulub juhtivate e-äri lahendusi pakkuvate firmade hulka maailmas.

SAP BW	(inglise keeles *SAP Business Information Warehouse*) - terviklik ärireegleid koondav SAP tootekeskne andmeladu.
SAP Fiori	SAP poolt 2013 aastal tutvustatud kohanduva disainiga SAP rakendused ning nende loomise disaini-printsiibid.
SAP GUI	SAP graafiline liides.
SAP HR	Tuntud ka kui SAP HCM (*Human Capital Management*) on SAP moodul inimressurssidega seounduvate tegevuste haldamiseks.
SAP NetWeaver Gateway	SAP NetWeaver toode, mis võimaldab luua SAP süsteemidega seotud veebiteenuseid.
SAP Portal	Keskkond, mis tagab ühtse juurdepääsukoha nii SAP kui ka SAP-välistele informatsiooniallikatele.
SAP UI5	Lühend inglisekeelsele mõistele *SAP User Interface for HTML5.*
SAP Web-Dynpro	SAP tehnoloogia veebipõhiste kasutajaliideste loomiseks.
SAP WebIDE	Veebipõhine SAP süsteemidega integreeritav arenduskeskkond.
Scrum	Agiilne tarkvara-arenduse metoodika, mis kasutab kindlaks määratud pikkustega iteratsioone ehk sprinte lõpp-toote kvaliteetseks loomiseks, kasutades seehulgas mõisteid nagu toote kogum, sprindi kogum, Scrum meister ning toote omanik.
Scrum meister	(inglise keeles *Scrum Master*) - Scrum arendusmetoodika juures kasutatav roll, mis representeerib isikut, kes vastutab arendusmeeskonna-sisese töövoo sujumise läbi takistuste kõrvaldamise, koosolekute läbiviimise ning projekti üldist kulgemist kontrollides.
SOA	(inglise keeles *System Oriented Architecture*) - Teenustepõhine arhitektuur, kus tarkvara protsessid on jagatud teenusteks.
Sprindi kogum	(inglise keeles *Sprint Backlog*) - Scrum arendusmetoodika juures kasutatav mõiste, mis hõlmab järgmise sprindi raames arendatavate funktsionaalsuste kogumit.
Sprindi planeerimine	(inglise keeles *Scrum Master*) - Scrum arendusmetoodika juures kasutatav mõiste, mis representeerib meeskonna, toote omaniku ning Scrum meistri koosolekut, mille käigus arutatakse toote omaniku juhtimisel, millised

	kasutaja-lood on järgmise sprindi sees saavutatavad, mille alusel koostatakse sprindi kogum.
Sprint	Iteratsioon, kui väikseim võimalik töötsükkel, mille jooksul toodetakse töötav tarkvara.
Stacey maatriks	Maatriks, mis aitab leida projektile sobivat tarkvara-arendusmetoodikat, hinnates nõuete selgust ning tehnoloogilist keerukust.
Testimisest Tulenev Arendus	(inglise keeles *Test Driven Development*, ehk *TDD*) - tarkvara arendamise protsess, kus esimese etapina kirjutatakse valmivale osale ühiktestid ning seejärel jätkatakse arendamisega.
Toote kogum	(inglise keeles *Product Backlog*) - Scrum arendusmetoodika juures kasutatav mõiste, mis hõlmab kõiki kliendi kasutaja-lugude prioritiseeritud nimekirja.
Toote omanik	(inglise keeles *Product Owner*) - Scrum arendusmetoodika juures kasutatav roll, mis representeerib tellija-poolset isikut, kes haldab toote kogumit, juhendab meeskonda ning võtab vastu töö tulemid.
Traditsiooniline mudel	Kosemudel (inglise keeles *Waterfall*) - tarkvaraarenduse mudel, kus tegevuse etapid kindlas suunas järjestikuse voona.
Töös olevate ülesannete arvu piirang	(inglise keeles *Work In Progress*, ehk *WIP*) - Kanban arendusmetoodika poolt kasutatav mõiste, mis tähendab maksimaalset tööülesannete arvu igas töö etapis.

Sisukord

Jooniste loetelu

Tabelite loetelu

1 Sissejuhatus

Majandus- ja Kommunikatsiooniministeeriumi väljastatud Eesti Infoühiskonna arengukava 2020 kohaselt on tehnoloogia, inimeste kasutusharjumused ning seadusandlus pidevas muutumises [1].

„Efektiivne on see, kes oskab neid muutusi õigeaegselt ette näha ja omab paindlikkust nende muudatuste elluviimiseks. Paindlikkus aga tähendab vajadust vältida oma aja ära elanud tehnoloogiatesse takerdumist.“ [1]

Arengukava toob täiendavalt välja vajaduse Eesti avalike (e-)teenuste reformile, et vältida suuri halduskulusid tulevikus ning mitte takerduda „spagetiarhitektuuri“. Reform näeb ette Eesti (e-)teenuste kaasajastamist, mis vastaks ühtsele kvaliteedinõudele, lisanduvalt nõudega, et avalikus sektoris ei tohi olla olulise tähtsusega IKT-lahendusi vanemad kui 13 aastat [1].

Muudatustega kaasnevates arendusprojektides on avalik sektor aga maha jäämas üldiste trendidega, kus erasektoris on juhtimas muudatusi aktsepteerivad agiilsed arendusmetoodikad, ning mida ei ole avalikus sektoris siiani laialt kasutusel [2].

Käesolevas töös otsib autor töös käsitletava avaliku sektori projekti peamiseid murepunkte ning puudujääke, leidmaks parimad praktikad projektide jaoks, mis on piiratud avaliku sektori määratud nõuetega ning millel on suur tehniline keerukus ning projektiga seonduvatest süsteemidest tulenevad nõuded (töös käsitletud SAP moodul ning selle keerukuse haldamine projektis). Antud projekti analüüsiga loodab autor leida võimalikud lahendused seal esinenud probleemidele ning vahendid sarnaste takistuste vältimiseks tulevastes projektides.

Antud magistritöö ning selles läbiviidav analüüs võimaldab märgata võimalikke probleemide allikaid ning tekkimise ilminguid nii autori enda edasistes projektides kui ka üldiselt tehnoloogiliselt keerukate avaliku sektori IKT projektide arenduste haldamisel.

Käesoleva magistritöö põhieesmärkideks on:

1. Leida töös käsitletava projekti probleemide juur-põhjused
2. Analüüsida probleemide juur-põhjuseid ning leida neile võimalikud lahendused
3. Analüüsida töös käsitletava projekti sobivust agiilse arendusmetoodikaga
4. Leida töös käsitletava projekti iseloomule sobivaim arendusmetoodika
5. Teadmiste omandamine taoliste projektide arenduse haldamisel

Töös püstitatud eesmärkide saavutamiseks kasutab autor juhtumianalüüsi metoodikat, kus analüüs toetub teaduslikele uuringutele ning isiklikele kogemustele seoses töös kirjeldatava projektiga.

Töö koosneb 6 peatükist:

- Esimene peatükk on töö sissejuhatav osa, kus autor esitab töö eesmärkide püstituse, kasutatava metoodika tutvustuse ning töö vajaduse määratluse.
- Teises peatükis annab autor ülevaate avaliku sektori IKT projektide peamistest valupunktidest ning nende piirangutest.
- Kolmandas peatükis esitab autor töös analüüsitava projekti eesmärgi, teostuse koos muudatuste ning vigade haldusega ja toob välja selle käigus ilmnenud peamised probleemid.
- Neljandas peatükis toob autor läbi põhjus-tagajärg analüüsi välja peatükis 3 esitatud probleemide juur-põhjused ning võrdleb neid peatükis 2 esitatud üldiste probleemidega.
- Viiendas peatükis analüüsib autor läbi kolme erineva metoodika peatükis 3 esitatud projekti paiknemist arendusmetoodikate skaalal (traditsiooniline või agiilne lähenemine) ning seejärel uurib projekti sobivust kolme agiilse arendusmetoodikaga. Peatüki lõpus leitakse kõige sobivam arendusmetoodika ning antakse ülevaade võimalikest lahendustest probleemide juur-põhjustele.

- Kuuendas peatükis võtab autor kokku töös käsitletu ning kontrollib, kas töös püstitatud eesmärgid täideti.

2 IKT projektide valupunktid avalikus sektoris

Antud jaotises kirjeldab autor peamisi probleeme, mis esinevad avaliku sektori projektide korral, tuues välja probleemide üldisema tausta ning avaliku sektori projektide eripärad. Täiendavalt võimaldab jaotises kirjeldatu hilisemalt kontrollida üldiste avaliku sektori probleemide vastavust töös uuritud projekti probleemidega

2015 aasta Standish Grupi uuringu kohaselt 31,1% Ameerika Ühendriikides alustatud IT projektidest tühistatakse enne kui need jõutakse lõpetada ning 52,7% projektide maksumus on 189% esialgsest hinnangust, kus vaid 16,2% projektidest valmivad esialgseks tähtajaks jäädes samal ajal eelarve piiridesse [3].

Lisanduvalt toob raport välja küsitlusest lähtunud põhjused, mis tagavad projektide edukuse. Kolmeks juhtivaks faktoriks olid kasutajate seotus projektiga, tegevjuhtide tugi ning selgelt määratletud nõuded. Samuti määrasid olulist osa projekti pädev planeerimine, reaalselt seatud ootused ning võimekus projekti vahe-etappide määramiseks [3].

Täiendavalt uuriti peamisi põhjuseid, mis oluliselt raskendavad projekti edukat lõpetamist ning kolmeks peamise põhjusena toodi välja vähene kasutajapoolne panus, poolikud nõuded ja spetsifikatsioon ning kolmandana muutuvad nõuded ja spetsifikatsioon [3].

Kolmandana uuriti peamisi põhjuseid, mis põhjustavad projektide ebaõnnestumist. Kolmeks peamiseks põhjuseks toodi sarnaselt eelnevale välja puudulikud nõuded, vähese kasutajapoolse panuse ning vajaminevate ressursside puudumise [3].

Avatus suhtlusele, muutustega toime tulemine ning läbi tagasiside hindamise esialgse ebamäärasusega toime tulemine on aga iseloomujooned, millega suhestub agiilne lähenemine tarkvaraprojektide käsitlemisel [4].

Avaliku sektori projektid on tihedalt seotud riigihankega, nõudes fikseeritud hinda ja skoopi, mis aga võib saada takistuseks agiilsete põhimõtete järgimisel [5].

Samuti on antud töös käsitletav projekt avaliku sektori projekti raamlepinguga käsitletav osa ning seega fikseeritud hinnaga nagu seda sätestab riigihanke seadus, mille kohaselt

tuleb välja kuulutada hange, millega ühtlasi pannakse paika ka projekti maksumus ning selle ulatuses kaetav funktsionaalsus. Fikseeritud hinnaga projektide puhul lasub skoobi kontrolli kohustus tarnija poolel, tagades selle, et kõik hankija esitatud nõuded saaksid täidetud ning soovitud lõpp-tulem hankijale tarnitud [5].

Oluline on, et hankija-poolne tehniline dokumentatsioon luuakse võimalikult detailne, sest see määrab ära millest tarnija nõuete realiseerimise käigus lähtub. Ebapädev tehniline dokumentatsioon võib varjata nõuete tegelikku keerukust või lahenduse realiseeritavust, mis omakorda mõjutab valesti arvestatud rahalisi ning ajalisi kulusid. Praktikas on sellised ootamatused IT projektide arendamisel sagedased juhtuma, ning teatud puhvri arvestamine ajakuludesse on üsna tavapärane [6].

Suuremad ebakõlad võivad põhjustada ka puhvri varude lõppemise ning selline olukord võib põhjustada mitmeid probleeme. Selline olukord tähendab tarnija jaoks teostamata tööde tegemist enda ressursside arvelt. Samuti võib see omakorda tekitada nn 'doomino-efekti', kus üle aja ning lubatud ressursside läinud projekti etapp takistab omakorda järgneva etapi õigeaegset tarnimist. Püüdes eelnimetatud probleemide kuhjumist vältida, kiirustatakse hetkel aktuaalse etapi lõpetamisega, mistõttu langeb funktsionaalsuse kvaliteet puuduliku testimise ja koodi kvaliteedi kontrolli tõttu, põhjustades omakorda pingeid tarnija ja hankija vahel [6].

Samuti toob 2010. aasta riigikontrolli aruanne välja, et paljud riigi infotehnoloogia valdkonna projektid ebaõnnestusid eelkõige puuduliku ettevalmistuse tõttu. Arendustöid alustati juhtkonna otsusel sageli ilma võimalikke lahendusteid ja oodatavat kasu ning kulu hindamata [7].

Suurte avaliku sektori projektide probleeme on uuritud ka rahvusvahelisel tasandil artiklis „Managing large-scale IS/IT projects in the public sector: Problems and causes leading to poor performance", kus analüüsiti IT projekte üle maailma [8].

Uurimuse kohaselt mõjutab projektide edukust see, kui hästi on osapooled mõistnud tulevase projekti funktsionaalset keerukusastet, kus selle alahindamine võib põhjustada ebatäpseid või puudulikke nõudeid. See suurendab võimalust, et tarnija poolt tehtud pakkumine ei arvesta varjatud keerukuse võimalusega ning selle maksumus ja ajakava ei samastu reaalsusega [8].

Üheks raskuseks avaliku sektori projektide juures on ka nõuete muutumine ning muudatuste haldus. Kui projekti summas on eelnevalt kokku lepitud, moodustuvad skoobi ümber üsna jäigad piirid rahalistes ning ajalistes ressurssides. Suuremahuliste projektide puhul võivad nõuded hankija poolel muutuda ning tekib vajadus muudatusteks, kus nende esialgsesse skoopi mahutamine võib osutuda problemaatiliseks [8].

Lähtudes riigihanke seaduse paragrahv 28 punktist 5, on hankijal siiski hilisemalt õigus korraldada väljakuulutamiseta läbirääkimistega hankemenetlus, kui tellitakse samalt tarnijalt esialgses projektis või hankelepingus mittesisaldanud, kuid esialgses hankelepingus kirjeldatud töödeks või teenuste osutamiseks ettenägematute asjaolude tõttu vajalikuks osutunud täiendavaid ehitustöid või teenuseid kuni 20 protsendi ulatuses esialgse hankelepingu maksumusest, kus täiendavaid töid või teenuseid ei saa tehniliselt või majanduslikult esialgse hankelepingu esemest eraldada ilma hankijale suuri kulutusi põhjustamata või kui täiendavad tööd on hankelepingu täitmiseks vältimatult vajalikud [5].

Samuti lihtsustab olukorda avaliku sektori projekti puhul ka raamleping, mille kohaselt võib tarnijalt lepingu kehtivusaja vältel selle alusel tellida lisahankeid vajalike lisaarenduste, mis ei jäänud 20 protsendi hankelepingu maksumuse sisse, teostamiseks. See vähendab riske nii hankija kui ka tarnija jaoks, sest raamleping lubab hankijal skoopi mitte mahtuvad muudatused ning lisaarendused tellida samalt tarnijalt ning võimaldades seeläbi rasketes olukordades edasi liikuda kartmata, et tellitu ei saa jäiga skoobi tõttu soovitul kujul üle antud [5].

3 Projekti ülevaade

Töös kajastatav projekt on raamlepinguna osa suuremast avaliku sektori projektist, millest viimane hõlmas endas uue personali- ja palgaarvestuse toimemudelile üleminekut. Töös käsitletav alam-projekt nägi ette veebipõhise andmeedastuskanali arendamist, mis võimaldab tellijal hallata oma personaliga seotud tegevusi (edaspidi andmeedastuskanal). Autor oli kirjeldatava projekti algse arendusmeeskonna liige ning tihedalt seotud kõikide selle etappidega, alates esmastest koosolekutest, tehniliste lahenduste valikust kuni lahenduse üleandmiseni kliendile.

Järgnevates jaotistes käsitleb autor täpsemalt projekti kirjeldust, selle analüüsi- ning arendusfaasi eripärasid, muudatuste ja vigade haldust ning lisanduvalt mainib ära töö käigus esinenud peamised probleemid.

3.1 Andmeedastuskanali kirjeldus

Andmeedastuskanali eesmärk on tõhustada personaliandmete liikumist hallatavatelt asutustelt ametiasutustesse ning andmete talletamist ühtses SAP HR-moodulis.

Loodavat andmeedastuskanalit hakkavad kasutama allasutuste personalitöötajad/juhid ning ametiasutuse personalitöötajad. Andmeedastuskanali abil edastavad allasutuse personalitöötajad/juhid personaliandmed ametiasutuse personalitöötajale kinnitamiseks.

Andmeedastuskanal asub SAP Portal keskkonnas ning on selle kasutajatele ligipääsetav veebilehitseja abil. Andmevahetus andmeedastuskanali rakenduse ja SAP süsteemi vahel on realiseeritud veebiteenustega.

Andmeedastuskanal hõlmab endas tööde loetelu kuvavat peavaadet ning personalihaldusega seotud tööalaseid vorme, milleks on ametikoha loomine, muutmine ja lõpetamine; tööle võtmine, töötaja andmete muutmine, töölepingute, puhkuste ning lähetuste haldamine.

Andmeedastuskanalis saab protsess alguse ametiasutuse personalitöötaja/juhi poolt personaliandmete sisestamise või muutmisega ja lõpeb andmete salvestamisega SAP ERP süsteemi või palga-arvestaja poolsete lisategevustega SAP ERP süsteemis. Äriprotsessi osa andmeedastuskanalis kulgeb läbi kahe kuni kolme osaleja – ametiasutuse

personalitöötaja, hallatava asutuse personalitöötaja ja hallatava asutuse juht (viimased kaks on sageli ühes isikus).

Arvuliselt koosnes lahendus esmase spetsifikatsiooni järgi peavaatest ning üheksast vormist, millest töös analüüsitud perioodi käigus realiseeriti kuus ning kolm jäeti hilisemaks arendamiseks (ühtlasi täiendavaks täpsustamiseks). Töö käigus lisandus vajaduspõhiselt neile vormidele juurde veel üks esialgu käsitlemata vaade. Samuti täienesid kolme vormi nõuded ning lisandus funktsionaalsust üle kõigi realiseeritud vormide.

3.2 Projekti teostus

Kuigi töös käsitletava projekti haldamise metoodika ei olnud üheselt paika pandud, läheneti sellele kui traditsioonilist kosemudelit järgivale projektile, kus enne arendusfaasi toimus kogu rakendust hõlmav analüüs, mis kestis 3 kuud ning millele järgnes arendusfaas, mille algseks kestuseks oli samuti määratud 3 kuud. Siinkohal on autori arvates oluline mainida, et arendusfaas oli jagatud kolmeks eraldi tarnitavaks osaks ehk iteratsiooniks, mis koosnesid andmeedastuskanali erinevatest vormidest (alates kriitilisematest kuni vormideni, mille realiseerimine polnud nii ajakriitiline). Iteratsiooni valmimisel toimusid sinna kuuluvate vormide vastuvõtutestid ning samal ajal alustati järgmisesse iteratsiooni vormide arendusega, kus puudus agiilsele lähenemisele omane tagasivaade või täiendav analüüs kasutajatega.

Järgnevates jaotistes käsitlebki autor täpsemalt projekti analüüsi- ning arendusfaasi iseärasusi ühes selle üldise tehnilise lahenduse, projekti muudatuste ning vigade haldusega. Peatüki lõpetuseks kirjeldab autor ka projekti käigus ilmnenud peamisi probleeme.

3.2.1 Analüüsifaas

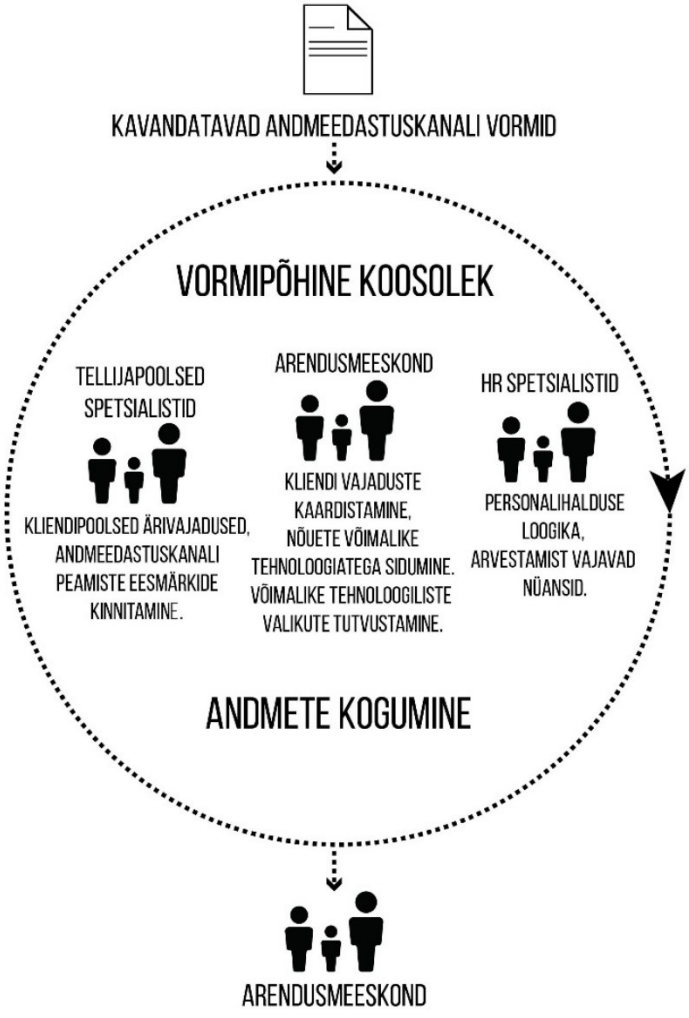
Projekti analüüsifaasi esimeses järgus toimusid koosolekud kliendiga, kus määratleti kliendi põhivajadused loodava lahenduse juures. Koosolekutel käsitleti kindla tegevuse (ja sellega loodava andmeedastuskanali vormide) defineerimist, millest võtsid osa nii tellija, arendajad kui ka inimressursside spetsialistid. Koosolekutele eelnevalt oli kliendi poolt koostatud loodavate vormide põhised dokumendid, kus oli lühidalt esitatud järgmised punktid:

- Vormiga seotud protsessi hetkeolukorra kirjeldus
- Tegevuse rollipõhine protsess
- Protsessi loodava andmeedastuskanali rollipõhine tegevus-diagramm
- Kliendi määratud üldised vormipõhised nõuded

Loodud dokumendid vaadati koosolekutel HR-spetsialisti ning arendusmeeskonna juuresolekul üle ning tehti vajalikke täiendusi ning muudatusi.

Samuti määrati analüüsifaasi alguses ära projekti nii tellija- kui tarnijapoolsed meeskonnaliikmed ning analüüsifaasis tellija ja tarnija poolt täidetavad protsessid.

Analüüsi vormipõhiste koosolekute üldine koosseis ja osavõtjate ülesanded on toodud joonisel 1.



Joonis 1. Vormipõhise analüüsi koosseis.

Osapoolte ülesanded analüüsifaasis jagunesid järgmiselt:

1) Tellija

Tellija esitab (vajadusel täitjaga konsulteerides) konkreetse personalihalduse äriprotsessi nõudedokumendi täitjale, mis sisaldab kirjeldust all-toodud ulatuses [9]:

- Äriprotsessi mudelit
- Äriprotsessi käigus tekkivaid andmeid – nende loomist/ muutmist/kustutamist/ arhiveerimist
- Töövoo kirjeldust antud äriprotsessi osas
- Tekkivaid dokumente ja nende arhiveerimise nõudeid
- Kasutusõigustele esitatavaid nõudeid

2) Täitja

Täitja koostab (vajadusel tellijaga konsulteerides) esitatud info alusel omapoolse esmase spetsifikatsiooni, mis sisaldab [9]:

- Äriprotsessi sisu ja vajaduse kirjeldust
- Arendustöö funktsionaalset kirjeldust
- Kujunduse ja ekraanivormide prototüüpe
- Valitud tehnoloogia kasutamise eeliseid teiste alternatiivsete lahenduste ees
- Tehnilise lahenduse kirjeldust
- Vajaminevate seadistuste kirjeldusi
- Autoriseerimise kirjeldusi

Loomisele kuuluvad osad jaotati vormide kaupa kolme ossa, kus esimesena pidid valmima kõige kriitilisema tähtsusega osad. Lisaks planeeriti esimesse osasse ka kõige lihtsama vormi valmimine, mis hõlmas endas märkmete ning manuste vormi.

Kuna andmeedastuskanali lahendus hõlmas endas liidestamist SAP ERP süsteemiga, hõlmas analüüs ka *back-end* andmete haldamise loogikat, defineerides andmetabelid, nende seosed, samuti esitati esialgsed kasutatavad staatused, veahalduse ning andmeid haldavate kasutajate rollid ja õigused.

Koosolekutel oli üheks teemaks ka f*ront-end* tehnoloogia valik, kus võimalus oli kanali vastav osa teha juba vanas ja varasemalt SAP süsteemide poolt kasutatud Web-Dynpro programmina või otsustada SAP toodete seas uue HTML5 tehnoloogiale baseeruva SAP UI5 kasuks.

Esimestel koosolekutel toimus arutelu tehnilise lahenduse valiku üle, kus arendusmeeskond ning enamus kliendi-poolseid esindajaid pooldasid uudset SAP UI5 lahendust.

Mõlema lahenduse demonstreerimiseks loodi ka lihtsad näited, kus Web-Dynpro versioon loodi Soome SAP HR mooduli spetsialisti ning UI5 näidis projekti arendusmeeskonna poolt. Kui Web-Dynpro prototüüp valmistati võimalikult loodava andmeedastuskanali avakuva sarnaseks, siis SAP UI5 prototüüp oli mõeldud visualiseerimaks üldist SAP UI5 kontseptsiooni kuvades juba andmesisestuseks mõeldud valmis vormipõhist SAP UI5 standard-lahendust.

Kuigi prototüübid ei toonud oma erinevuse tõttu absoluutset selgust, otsustati SAP UI5 lahenduse kasuks peamiselt selle paindlikumate ja ajaga kaasas käivate võimaluste tõttu nagu kohanduv veebidisain ning erinevate seadmete tugi. Siinkohal soovib autor välja tuua, et töö edasistes osades käsitletaksegi projekti käsitlust valitud tehnoloogia (SAP UI5) kontekstis.

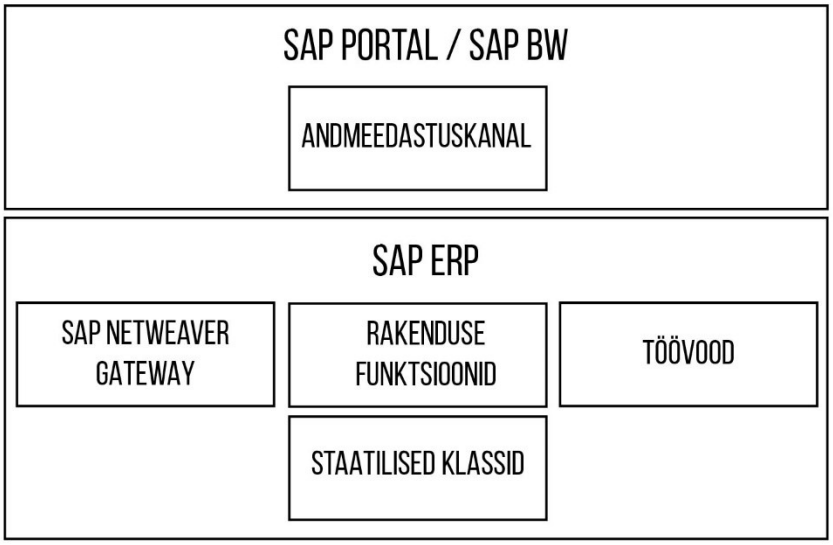
Pärast vormipõhiste põhinõuete ning tehnoloogia fikseerimist moodustati arendusmeeskonna poolt kogutud andmete põhjal analüüsifaasi tulemina valminud esimene projekti spetsifikatsioon, kus kirjeldati ära rollide vahelised äriprotsessid ning üldine tehniline protsessi kirjeldus koos andmete kuvamise ja talletamise loogikaga.

Analüüsifaasis aga detailsusastet, milleni sooviti lahenduse prototüüpimisel jõuda, ei defineeritud ning prototüüpide detailsus jäi seetõttu esimeste vormide puhul madalaks ning visuaalseid kuvatõmmised eksisteerisid vaid kahel kõige prioriteetsemal vormil ning teiste vormide puhul piirduti vormipõhiste erinõuete ning infotüüpide defineerimisega.

3.2.2 Andmeedastuskanali tehniline lahendus

Andmeedastuskanali tehniline lahendus SAPUI5 tehnoloogia kasutamise korral on kujutatud joonisel 2. Kasutaja saab ligipääsu andmeedastuskanalile SAP NetWeaver Portali kaudu. ERP süsteemiga suhtleb rakendus NetWeaver Gateway vahendusel veebiteenuste kaudu. Veebiteenused on üles ehitatud funktsioonimoodulitele, millega on teostatud andmetabelitest andmete lugemine, kirjutamine ja muutmine.

Loodav lahendus koosneb eraldi süsteemides paiknevatest *front-end* ja *back-end* osast, kus andmeedastuskanali front-end on seotud kliendi personali poolt juba kasutatava SAP Portal veebikeskkonnaga SAP Business Warehouse süsteemis ning *back-end* ühes andmeid haldavate funktsioonimoodulite, töövoogude, veebiteenuste ning staatiliste klassidega SAP ERP süsteemis.



Joonis 2. Andmeedastuskanali tehnilise lahenduse skeem

Andmeedastuskanali poolt teostatavad tegevused on seotud seda kasutavate töötajatele määratud SAP õiguste ja rollidega. Järgnevalt on esitatud kasutajate autoriseerimise ning kasutajarollide üld-põhimõtteid ja definitsioone.

Autoriseerimise üld-põhimõtted [9]:

- Andmeedastuskanalile ligipääsemiseks ning sellega töötamiseks kasutatakse SAP ERP ja BW/Portali kasutaja-kontosid. Kasutaja ei pea Portalis andmeedastuskanali kasutamiseks end korduvalt identifitseerima.

- Kasutusõiguste defineerimine, omistamine ja eemaldamine toimub standardsete SAP rollide kaudu

- Andmeedastuskanalis kasutusele võetavad rollid on kirjeldatavad pärimisseosega. Defineeritakse üldine roll, mis võimaldab ligipääsu kõigile andmetele ning pärimisseose abil loodud rollid seavad täpsemaid piiranguid organisatsiooniüksuste lõikes

- Ühele kasutajale saab anda korraga mitu andmeedastuskanali kasutajarolli

3.2.3 Arendusfaas

Spetsifikatsiooni esimese versiooniga algas ka andmeedastuskanali arendus, kus alustati kõige prioriteetsema vormi arendusega. Esimese tellitud osana kuulusid arendamisele vormid tööle võtmine, tehniliselt lihtsaim märkmete vorm (iteratsioon I), töölepingu muutmine ja lõpetamine (iteratsioon II). Kolmanda iteratsioonina oli kavandatud ka puhkuste ja töösuhte lõpetamise vormi arendamine, mis aga esialgse ajakava sisse ei mahtunud ning realiseeriti muudatustaotlusena. Ülejäänud vormid kuulusid küll projekti spetsifikatsiooni, kuid jäid ootele hilisemaks arendamiseks. Samaaegselt toimusid aga andmeedastuskanali spetsifikatsiooni pidev parandamine ja täiendamine, kus dokument oli tellijapoolse pideva jälgimise all ning pidevalt tuli tarnija-poolele täpsustusi ning parandusnõudeid.

Arendus toimus nii *front-end* kui ka *back-end* poolel ning tarnijal kasutas selleks enda süsteeme ning ei vajanud algselt tellijapoolset keskkonda. *Front-end* arendus toimus uues SAP enda loodud interneti-põhises arenduskeskkonnas SAP WebIDE, mis sisaldas koheselt SAP UI5 kõige uuemaid komponente ning võimaldas lihtsat liidestust nii Git-hoidla kui SAP *back-end* teenustega.

Vormide arendamisel järgiti spetsifikatsioonis kirjeldatud vormile seatud erinõudeid ja piiranguid. Samuti oli spetsifikatsioonis viited veakontrollidele (täienes projekti käigus), teadetele ning toodud olid ka vormipõhise andmeliiklusega seotud veebiteenuse väljatüübid. Spetsifikatsioonis ei olnud algselt toodud kõikide vormide poolt kasutatavaid veebiteenused, vaid need lisati sinna töö käigus arendajate poolt. Samuti ei olnud kõikide andmeedastuskanali vormide vaadetele visuaalseid prototüüpe. Vajalikud väljad olid defineeritud vastavalt SAP süsteemis olevate samade infotüüpidega.

Esialgu oli arendusmeeskonnas kaks inimest, kellest mõlemad olid kaasatud ka koosolekutesse ning seeläbi ka analüüsi protsessi. Mõlemad arendajad tegelesid esimese iteratsiooni vormide nii *front-end* kui *back-end* arendusega ning lahenduse üldise vajalike struktuuride loomisega SAP ERP süsteemis.

Kui *front-end* arendus kätkes endas HTML5 tehnoloogiat järgnevate javascriptis loodavate SAP UI5 vormide loomist, siis *back-end* lahendus tähendas nii andmeedastuskanali põhiste tabelite ja nendevahelise struktuuri koostamist kui ka *ABAP* funktsioonimoodulite ühes neid kasutatavate veebiteenuste loomist.

Esimese iteratsiooni vormide koostamise ajal oli arendusmeeskonnal vaja ka tegeleda projekti edasise analüüsi ning sellest lähtuva spetsifikatsiooni täiendamisega. Arendusmeeskonna esialgse väikese suuruse tõttu ei kasutatud ka arendusülesannete haldamist hõlbustavaid vahendeid, nagu KanBan tahvel või selle veebipõhised analooge (nt. Trello). Esimese iteratsiooni keskel liitus arendusmeeskonnaga veel kolmas liige, misjärel hakati korraldama nädalas korra peetavaid arendusmeeskonna-siseseid koosolekuid, kus liikmed said koos lahendada tekkinud probleeme ning jagada uusi ülesandeid. Viimase iteratsiooni algusfaasis, milleks oli töötajate puhkuste ja töösuhte lõpetamise vormi loomine, lisandus meeskonnaga ka neljas arendusmeeskonna liige. See võimaldas meeskonnal jagada ressursse uute vormide nii *back-end* kui *front-end* arenduse kui eelmistes iteratsioonides valminud vormide paranduste ja kliendi poolt soovitud täienduste vahel.

Arendusmeeskonna koosseis ja liikmete peamised ülesanded (puudusid jäigad rollid) iteratsiooniti on toodud järgnevas tabelis 1.

Tabel 1. Arendusmeeskonna koosseis erinevates iteratsioonides.

Iteratsioon-Vorm	**Arendusmeeskond**	**Tellija**
Iteratsioon 1 – Vorm 1	Liikmeid: 2 Arendajaid:2 Ülesanded: • Analüüs • Arendus	Nõuete täpsustamine Vastuvõtutestid

	• Testimine	
Iteratsioon 1 – Edasised vormid	Liikmeid: 3 Arendajaid: 2 Ülesanded: • Arendus • Testimine Analüütikuid: 1	Nõuete täpsustamine Vastuvõtutestid
Alates iteratsioon 2	Liikmeid: 4 Arendajaid: 3 Ülesanded: • Arendus • Testimine • Muudatuste ja täienduste läbiviimine Analüütikuid: 1	Nõuete täpsustamine Muudatuste täpsustamine Vastuvõtutestid

3.3 Muudatuste haldus

Esimese iteratsiooni vormide valmimisel viidi lahendus ka kliendi arendus- ning testkeskkonda, kus tellijapool sai valminud tööd testida ning anda tagasisidet. Muudatuste alla kvalifitseerusid sellised ettepanekud, mis ei olnud tingitud arendusmeeskonna poolt põhjustatud veast lahenduse koodis või lahendused, mis olid jäänud projekti spetsifikatsioonis kirjeldamata.

Muudatustaotlus sisaldas endas:

- Viidet projekti lepingule
- Muudatuse algatajat
- Muudatuse koondnimetust
- Töö hinnangulist mahtu
- Tellija ja teostaja ressurssi

- Töö teostamise ja testimisse andmise kuupäevi
- Viiteid nõuetele
- Lõppolukorra detailset lahendust ning võimalikke lahendusvariante
- Lahenduse vajalikke seadistusi ja programmeerimisvajadusi
- Võimalikke muudatusi süsteemi kasutamises

Muudatustaotluste iseloom varieerus komponentide visuaalse järjestuse muutmisest (näiteks tabeli tulpade järjekorra muutmine) vormi komponentide välja vahetamiseni.

3.4 Vigade haldus

Arendusmeeskonna poolt loodud vormid edastati kliendile omapoolseks kontrollimiseks, kus iga vormi tegevuse jaoks viidi kliendi poolt läbi vastuvõtu test. Testi tulemuste kohta koostati koond-dokument, koos järgmiste testipõhiste andmetega:

- Vormi nimetus
- Vormi eesmärk
- Vastuvõtu testi sooritamise kuupäev
- Testijad
- Testi tulemus

Iga testi juures oli ka viide vastuvõtu testi detailsele dokumendile, mis kirjeldas testi kulgu ning vea tekkimisel selle asjaolusid. Vormid loeti vastuvõetuks, kui kõik testimise skoobis olnud testid läbiti edukalt ning tellijapoolsed tingimused olid täidetud.

Vastuvõtu testide dokumendid koostati iteratsioonide või tellitud muudatuste valmimisel ning neid hallati kõiki test-dokumente koondavas Microsoft Exceli tabelis.

Tabel 2. Vastuvõtu testi tulemuse näide

Vorm	Vorm “ Töösuhte lõpetamine“ toodangus
Vormi eesmärk	Tööle võtmise vormi eesmärk on töötaja (sh sotsiaalsetele töökohtadele tööle tulevad töötajad (v.a. lepingulised töötajad) isikuandmete ankeedis ja temaga sõlmitud töölepingus kajastuvate andmete sisestamine SAP HR süsteemi.
Vastuvõtu testid	17.12.2015
Testijad (13in)	
Testi tulemus	Aktsepteerimata

3.5 Probleemid ja takistused, garantiitööd

Projekti käigus esines andmeedastuskanaliga loomisel erinevat tüüpi probleeme, mis tulenesid arendusetapis tehtud vigadest, analüüsis koostatud spetsifikatsiooni prototüüpide detailsusest, SAP UI5 lahenduse uudsusest ja disainist. Järgnevalt kirjeldab autor projekti käiku seganud asjaolusid ning toob lühidalt välja nende peamised põhjused.

Autori käsitletud probleemid liigitusid järgmiselt:

- Tehnoloogilised takistused
- Disainist lähtuvad takistused
- Koostöö-alased takistused
- Arendusmeeskonna-sisesed takistused

3.5.1 Tehnoloogia-põhised takistused

Projekti arenduse algusfaasis, kui alustati esimeste vormide arendamisega, olid nii SAP UI5 kui ka lahenduse *front-end* pooleks valmistamiseks kasutatav SAP WebIDE uued tehnoloogiad.

SAP UI5 tuli turule koos SAP Fiori rakendustega, mille eesmärk oli viia varasemalt SAP GUI's tehtavad tihti raskesti hoomatavad tegevused visuaalselt meeldivale ja lihtsale veebilehitsejas hallatavale kujule. SAP standard Fiori lahendused olid aga disainitud lihtsakoeliste tegevuste jaoks. Et tagada erinevate turgude klientide soove, võimaldas SAP luua samu komponente (SAP UI5) kasutades Fiori rakenduste disainiga ühtivaid kliendi ärivajadusi luues ka UI5 rakenduste loomiseks orienteeritud arenduskeskkonna. Tellitud andmeedastuskanal oli aga kogum keerulistest vormidest, kus andmeväljad olid suuresti teineteisesega seotud, kus tuli arvestada väärtuspõhiste sisestusabide ja kasutaja rollidest mõjutatud piirangutega.

Keerulised vormid tähendasid, et paljud andmeväljad tuli siduda kontroll-loogikatega, mis vormi täitmisel suhtlesid *back-end* funktsioonimoodulitega läbi hulgaliste veebiteenuste. See tähendas, et *back-end* funktsioonimoodulite kui ka *front-end* lahendus pidi olema võimalikult ökonoomne, et lahenduse keerukus ei aeglustaks ja seeläbi pärsiks lahenduse kasutatavust. Algsete prototüüpide andmeväljade detailsus ei olnud spetsifikatsiooni esimeste versioonide puhul aga väga sügav ning esimesena valminud vormide loogikale lisandus hilisemalt erinevaid kontrolle, mille koormusega esimestes etappides arvestatud ei oldud. Seetõttu tuli hilisemas etapis siiski tegeleda funktsionaalsuse lisamise ning sellest tuleneva andmeedastuskanali kasutuskogemuse parandamise ning teenuste ja andmete liikluse optimeerimisega.

Samaaegselt projektiga arenes pidevalt ka SAP UI5 tehnoloogia, kus lisandus uusi komponente või nende funktsionaalsust. Siiski esines esimeste vormide puhul olukordi, kus SAP UI5 poolt teatud toimingu täitmiseks ettenähtud lahendus ei täitnud andmeedastuskanali vormis oma ülesannet täielikult või ei ühtinud vormi mõne teise loogilise komponendiga. See tähendas arendusmeeskonnale omakorda lisatööd ning täiendavat aega.

Analüüsi käigus ei lepitud kokku vaikimisi veebilehitsejaid, mis andmeedastuskanalit peaksid toetama ning üldine eeldus nii arendusmeeskonna kui ka tellija pool oli, et

andmeedastuskanali töö ei ole pärsitud üheski laialt-levinud veebilehitsejas. Samuti oli SAP uut UI5 tehnoloogiat tutvustanud kui erinevatel süsteemidel ja nutiseadmetel toimivat lahendust. Siiski tekkisid arendusmeeskonnal projekti üsna varases staadiumis probleemid, kus mõningad kasutatud komponendid käitusid eri veebilehitsejates teisiti, kus suurimad takistused tulid ilmsiks Internet Exploreriga. Antud veebilehitseja puhul olid probleemsed lahendused nagu manuste laadimine, mõningate komponentide andmemudelite uuendamine ning visuaalne kuju ja üleüldine käitumine. Asjaolu, et just Internet Explorer oli kliendi poolt kõige enim kasutatav veebilehitseja tõi arendusmeeskonnale üsna suuri probleeme, sest see nõudis sageli paljude juba töötavate lahenduste ümber disainimist.

Kogu projekti vältel oli andmeedastuskanali spetsifikatsioon kliendi ja arendusmeeskonna jälgimise all ning seda täiendati pidevalt. Esimeste vormide koostamisel avaldunud nii UI5 komponentide kui ka vormide disainist tulenenud probleemide tõttu täienes arendusfaasi käigus ka hilisema iteratsiooni prototüüpide detailsus. Sellega kaasnesid ka olukorrad, kus vormi tuli kliendi soovil täiendada või ümber-disainida, et see vastaks täpsemini tellija ootustele. Kui arendusmeeskond leidis, et tegemist on muudatustaotlusega, sest juba varasemalt spetsifikatsioonis olnud prototüübile tehtud töö tuli ümber disainida, siis esines juhtumeid, kus tellija leidis, et probleem on tingitud analüüsi faasis läbi arutamata jäänud detailist ning seega tuleks tööd võtta kui garantiitööd. Sellised olukorrad arutati poolte vahel läbi ning seejärel otsustati, mille sisse tööaeg arvestatakse. Samuti oli võimalus selliseid töid teatud ulatuses arvestada iga iteratsiooniga lisanduvasse varu-puhvrisse, mis oli ette nähtud tööst tulenevate ootamatute raskuste lahendamiseks.

3.5.2 Disain

Analüüsi etapis madala prototüüpide detailsuse tõttu tekkis arendusmeeskonnal ka küsimusi andmeväljade erinevate kuva- ja kasutusaspektide suhtes. Andmeedastuskanali üheks ülesandeks oli viia personalihalduse-alased tegevused kergelt arusaadavale ning lihtsasti täidetavale kujule, säilitades siiski kogu tegevuse keerukuse ning võimalused. Vormid pidid olema arusaadavad nii SAP GUI samade vaadetega tuttavate kui ka väga vähese SAP kogemustega kasutajate jaoks. Seetõttu tuli arendusmeeskonnal luua prototüüpe ka arendusfaasis, kus koos kliendiga konsulteerides tuli vorme muuta ning leida kliendi jaoks sobivaim lahendus. Samuti osutus üheks keerulisemaks etapiks SAP

GUI programmiga samaväärsete andmeväljade kontrollide loomine nii, et see ei mõjutaks andmeedastuskanali kiirust. Kui SAP GUI programmide kontrollid käivitasid päringuid vastu otseselt programmi väljadega seotud andmetabeleid ja välja tüüpe, siis andmeedastuskanal pidi suutma sama kiirust silmas pidades haldama nii *front-end* kui *back-end* liiklust ning logima andmeedastuskanalis tehtavad tegevused. Selline andmevormidega kaasnev keerukus osutus esimeses kahes vormide iteratsioonis andmeedastuskanali tööd siiski koormavaks, mistõttu tuli viimases iteratsioonis tegeleda ka garantiitööna lahenduse koormustaluvuse ja töökiiruse optimeerimisega.

3.5.3 Koostöö projekti partneritega

Töös kirjeldatava projekti ülesanne oli luua andmeedastuskanal, mis toimetaks andmed SAP ERP süsteemi selleks arendusmeeskonna poolt koostatud andmetabelitesse HR moodulisse edastataval kujul. Vormide kinnitamisel andmete HR moodulisse salvestamise ülesanne oli aga välisel partneril, kes koostasid vastavad funktsioonimoodulid ERP süsteemis. Kuigi arendusmeeskonna ja välise partneri vahel toimus vormipõhiste andmete vahetus ja koostöö juba analüüsifaasi alguses, esines siiski olukordi, kus enda ülesannetega jätkamiseks oli vajalik HR mooduli mõne funktsiooni eelneva vea parandamine, mis pärssis arendusmeeskonna tööd ja lükkas edasi tähtaegu. Samuti oli aeganõudvam arendusmeeskonna ja välise partneri vaheliste funktsioonimoodulite ühendamisel tekkinud vigade analüüs ning põhjuse leidmine, sest mõlemal pool puudus kohese tagasiside võimalus ning suheldi läbi kliendi või elektron-postitsi.

3.5.4 Meeskond

Projekti arendusmeeskonna suurus ei võimaldanud esialgu piisavalt suures mahus eraldada kindlaid rolle vormide arendamisel. See tähendas, et iga meeskonna liige pidi tegelema nii vormide ja prototüüpide edasise analüüsi, arenduse ja testimisega. Seda pehmendas aga asjaolu, et võimalik oli pidev suhtlus kliendiga, kes ühtlasi ka oli kaasatud lõplikku testimisfaasi.

Projekti erinevates staadiumites lisandus meeskonnaga veel kokku kaks arendajat, mis andis võimaluse ressursse ühtlasemalt jagama hakata, kuid mis jäi siiski liiga väikeseks, et oleks võimalik olnud analüüsi, arendusse ja testimisse eraldi. Siiski ei saa siinkohal täielikult nõustuda Brooks'i seadusega, mis väidab, et hilises staadiumis tööjõu lisamine

ainult suurendaks arenduseks kuluvat aega. Meeskonna suuremine andis siiski võimalusi tuua sisse rohkem kvaliteeti, sest see võimaldas esialgsetel arendusmeeskonna liikmetel samaaegselt rohkem suhelda klientidega ning vajadusel ka delegeerida ülesandeid.

Eelpool toodud probleemide tulem oli aga pikenenud iteratsioonid, mis nõudsid töö teostamiseks rohkem ajalist ressurssi, kus täiendavalt tuli arvestada nii lisanduvate muudatuste, tehnoloogilistest iseärasustest ning ebatäpsustest tingitud parandus-vajaduste olemasoluga. Kõik see takistas töö mahtumist algselt paigutatud ajalistesse raamidesse ning projekti valmimisaeg pikenes oluliselt.

4 Peamiste probleemide põhjus-tagajärg analüüs

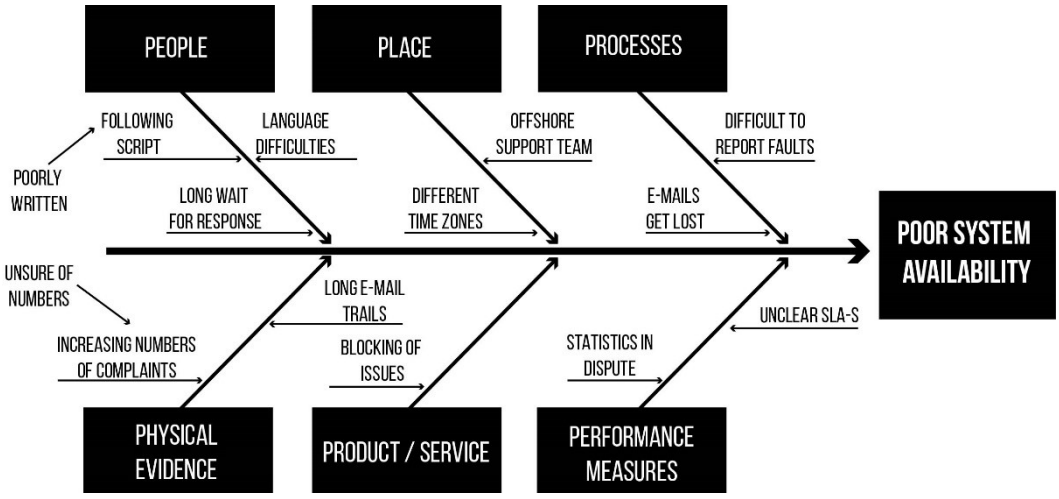
Käesolev peatükk käsitleb eelnevalt esitatud projekti põhjus-tagajärg analüüsi, kus alam-peatükkides on toodud kasutatava meetodi kirjeldus, selle teostus ning tulemuste analüüs.

4.1 Meetodi tutvustus

Järgnevas peatükis analüüsib autor projektiga seotud probleeme läbi põhjus tagajärg meetodi, kasutades selleks Ishikawa ehk niinimetatud f*ish bone* meetodi lähenemist. Ishikawa diagramm on 1960ndatel Kaoru Ishikawa poolt loodud meetod, mis oli algselt ette nähtud mõõtma kvaliteedikontrolli protsessi laevaehituses. Tänapäevases võtmes kasutatakse seda diagrammi aga ka analüüsimaks probleemide tekke põhjuseid [10].

Mudelis defineeritakse arusaadavalt probleem, mida analüüsitakse. See moodustab mudelis niinimetatud kala pea. Probleemist lähtuvad omakorda probleemi mõjutavad kategooriad ja nende alam-kategooriad (6P kategooriad – Inimesed (*People*), Asukoht (*Place*), Protsessid (*Processes*), Füüsilised tõendid (*Physical evidence*), Toode/Teenus (*Product/Service*) ja Sooritus-mõõdikud (*Performance measures*) [11].

Diagrammis kasutatavad kategooriad ei ole aga täiesti kindlaks määratud ning neid võib lisada lähtudes olukorrast. Järgneval joonisel on toodud ka diagrammi üldine ehitus (Joonis 3) [11].



Joonis 3. Ishikawa diagramm

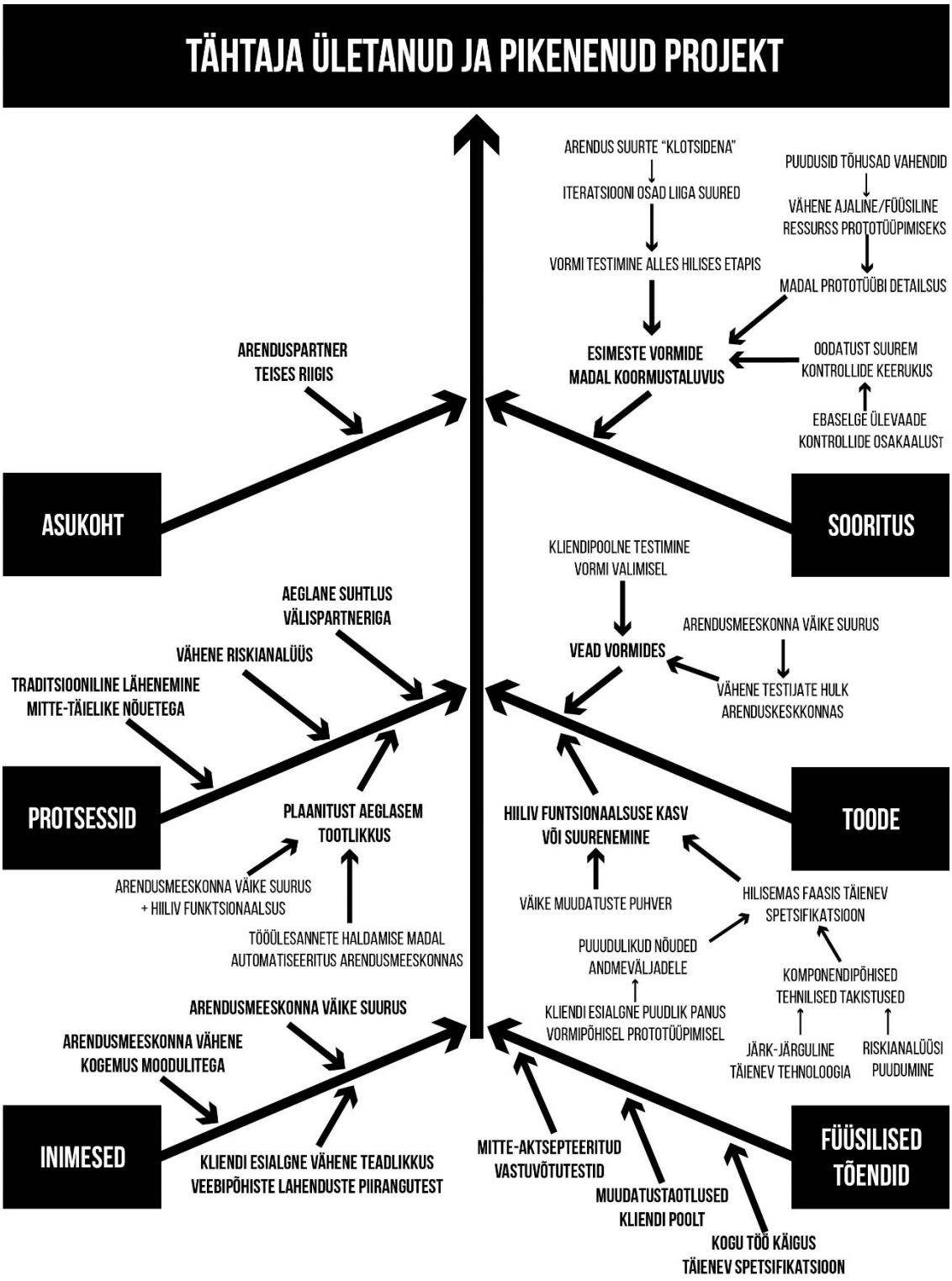
4.2 Projekti probleemide Ishikawa diagramm

Ishikawa 6P diagrammi koostamisel lähtub autor peatükis 3 esitatud projekti elutsükli käigus tekkinud probleemidest ja takistustest. Loetavuse huvides kuvatakse diagramm joonisel 4 vertikaalselt (erinevalt traditsioonilisele horisontaalsele paigutusele), kus tipus paikneb lahatav põhiprobleem.

Ishikawa analüüsis käsitletud probleemid on seotud järgnevas loetelus toodud andmeedastuskanali projekti probleemidega.

Andmeedastuskanali projekti probleemid:

- Uue tehnoloogia kasutuselevõtust tingitud tõrked
- Hiiliv funktsionaalsuse kasv või suurenemine
- Lahenduse madal koormustaluvus
- Vormide ümber disainimise vajadus
- Prototüüpide liigne madal esialgne detailsus
- Veebilehitseja tõrked
- Aeglane koostöö välise partneriga
- Esialgne väikseliikmeline arendusmeeskond



Joonis 4. Projekti probleemide analüüs Ishikawa diagrammiga

4.3 Projekti probleemide analüüs Ishikawa diagrammi põhjal

Käesolevas peatükis esitab ning analüüsib autor diagrammi koostamisel saadud tulemusi, mille peamine eesmärk on leida projekti probleemide juur-põhjused ja ühisosad.

Diagrammi tipu-probleemiks määratles autor tähtaja ületamise, sest kõik projekti käigus ilmnenud vead tähendasid esialgselt ettenähtud kava pidurdamist ning tõrkeid tulemi tarnimisel. Mudelis kasutatavate kategooriatena kasutati juba eelnevalt kirjeldatud 6P lähenemist.

Diagrammi kõige tihedamad osad on seotud kategooriatega Toode, Sooritus ja Protsessid, kus esimese kahe kategooria ülemiste tasemete harud kirjeldavad lahenduse tehnilisi takistusi, milleks on vead vormides, hiiliv funktsionaalsuse kasv või suurenemine ja esimeste vormide madal koormustaluvus. Tõrked nende kategooriate sujuvuses tähendasid projekti pikenemist läbi takistuste töö vastuvõtmisel ning suurenenud parandus- ja muudatustaotluste hulgas.

Üheks korduvaks niinimetatud juur-taseme probleemiks diagrammil on aga arendusmeeskonna suurus ja selle ressursi kättesaadavus. Nimetatud aspekt oli seotud probleemidega nagu plaanitust aeglasem tootlikus arendamise etapis ning samuti ka üheks mõjutajatest arendusmeeskonna poolt märkamata jäänud vigade ja prototüüpide algse madala detailsuse korral.

Meeskonna suuruse tegurit, kui ühte mõjutajat plaanitust aeglasemal tootlikkusel, põhjendab autor sellega, et esialgselt ei osatud arvestada hiiliva funktsionaalsuse asjaoluga, kus tuli tegeleda nii uute vormide arendamisega kui ka eelnevate vormide täiendamisega ning ühtlasi andmeedastuskanali spetsifikatsiooni uuendamisega. Seetõttu sai esialgne ajakava liiga lühike, et sinna mahuksid ka täpsustustega kaasnenud tööd.

Esimese mitmeid kategooriaid siduva probleemina toob autor välja kategoorias Toode toodud vigade olemasolu vormides. Peale väikese-liikmelise meeskonna omasid mõju ka tööülesannete madal automatiseeritus arendusprotsessis koos asjaoluga, et arendatavad osad ei olnud jaotatud piisavalt väikesteks alam-osadeks, et neid siis testida sellisel moel, et tõenäosus vea-aspektide märkamata jäämiseks oleks võimalikult madal. Samuti võimaldas vormide erinevaid andmevälju siduvate loogilistes kontrollides peituvate vigade tähelepanuta jäämine kliendi-poolsete testide sooritamine alles keeruka vormi

lõplikul valmimisel. See tähendas, et arendusmeeskonna poolt võimalik, et varakult tekkinud viga avastati alles hilises etapis.

Teise mitmeid kategooriaid siduva probleemina toob autor välja esimeste vormide madala koormustaluvuse, mis hõlmasid kategooriaid nagu Sooritus, Inimesed, ja Protsessid. Kirjeldatav probleem oli tehniliselt tingitud suurest arvust kontrollidest, mis tähendas, et kasutaja interaktsioonil vormiga toimus läbi veebiteenuste tihe liiklus *back-end* süsteemis paiknevate funktsioonimoodulitega. Ebaselge ülevaade kõikidest nõuetest esimese iteratsiooni arendusfaasis tähendas seda, et kontrollid kas muutusid keerulisemaks või lisandus neid juurde, milleks aga esialgne vormi karkass valmis ei olnud. Samuti soodustasid seda madal prototüüpide detailsus, mille tõttu ei saadud selget ülevaadet reaalselt saadetavate andmete mahust ja nende kontrollide tihedusest. Prototüüpide madal detailsus oli samuti tingitud ka UI5 komponentide visualiseerimiseks mõeldud tõhusa vahendi puudumisest (selline keskkond tuli turule pärast uuritava projekti valmimist) ja prototüüpide loomiseks olnud vähesest ressursist (esialgne kaheliikmeline arendusmeeskond). Lisanduvalt leiab autor, et kombinatsioon, mille osad olid arendusmeeskonna võrdlemisi vähene kogemus mooduli kõigi iseärasustega, kliendi esialgne vähene teadlikkus veebipõhise lahenduste piirangutest ja analüüsi etapis vähe rõhku saanud riskianalüüs, oli täiendavalt probleemi soodustav faktor.

Täiendava aspektina on seotud probleemina välja toodud arendusobjektide ebapiisav jagamine sobivalt väikesteks alam-osadeks, vaid arendamine suurte tükkidena, mis takistas varasemat vea märkamist ühe loogilise komponendi tasemel (varakult näha, et mõni teatud kontroll laeb andmeid sobimatult kaua). Samuti leiab autor, et kuna analüüsi faasis prooviti läbi traditsioonilise lähenemise analüüsida projekti, mille kõik detailid polnud selged, jäi tähelepanuta ka paindlik muudatuste lisamise tugi, mistõttu oli hilisemalt koormustaluvusega seotud probleemide lahendamine ka kulukas.

Kolmanda suurema haruna Ishikawa diagrammis võib käsitleda kategoorias Toode asetsevat hiiliva funktsionaalsuse haru, mille põhjused on seotud ka kategooriates Protsessid, Sooritus ja Inimesed esitatud takistustega. Peamiseks juur-põhjuseks, miks projekti käigus esines hiilivat funktsionaalsust, leiab autor olema kliendipoolse esialgse puuduliku panuse vormide prototüüpimisel. See, omakorda tingituna esialgsest nõuete ebatäpsusest kliendi poolel, tähendas seda, et esimestest vormidest jäid puudu mõningased kliendi jaoks vajalikud, kuid täpsustamata jäänud detailid. Siinkohal oleks

autori hinnangul kasuks tulnud ka arendusmeeskonna varasem kogemus projektis käsitletud mooduliga.

Lisaks tingis projekti spetsifikatsiooni hilisemas osas täienemise uuest tehnoloogiast tulenevad probleemid, mis teatud olukordades sundisid arendusmeeskonda vormide komponente ümber-disainima. Täiendavalt oli see seotud nii komponentide varjatud puudujääkide, detailse riskianalüüsi puudumise kui ka üleüldise vastava tehnoloogia-alase info puudulikkusega.

Kategooria, mis otseselt ei seondu eelpool esitatud probleemidega on Asukoht, millega seotud takistus oli aga samuti akuutne projekti tähtaegade ületamisel ning oli seotud koostööpartneri välisriigis asumisega. See tähendas aeglast suhtlust ning raskendatud koostööd omavaheliste lahenduste ühendamisel.

4.4 Järeldused

Ishikawa diagrammi analüüsile tuginedes saab välja tuua järgmised projektiga seondunud juur-põhjused:

1. Sooritus- ja kvaliteedialased vead olid suuresti tingitud madalast prototüüpide detailsusest ning esimeste vormide mitte-valmisolekust hiilivaks funktsionaalsuseks

2. Vähene riskianalüüs tõid arendusmeeskonnale lisatööd mille jaoks ressurssi ei jagunud (tehnoloogia piirangud, hiiliv funktsionaalsuse kasv või suurenemine)

3. Madalale koormustaluvusele ning vigadele aitas kaasa tööülesannete liiga madal segmenteeritus arendusmeeskonnas

4. Vigadele vormides aitas kaasa kliendi liiga hiline testimisse kaasamine

5. Ümber-disainimise vajadusele aitas kaasa kliendi esialgne väike panus vormide prototüüpide koostamisel

6. Koormustaluvuse probleeme vähendanuks arendusmeeskonna varasem kogemus projektiga seotud mooduliga

7. Raskesti teostatavad/kallid muudatused on seotud projekti spetsifikatsiooni loomisel puudulike nõuetega (analüüsimata muudatustega kaasnevaid riske)

8. Arendusprotsessi raskendas aeglane suhtlus välispartneriga

Võrreldes antud jaotises esitatud juur-põhjuseid jaotises 2 toodud avalike projektide üldiste probleemidega, saab järeldada, et uuritud projekti probleemid, nagu vähene kasutajapoolne panus, poolikud nõuded ning puudulikud ressursid, ühtisid suurel määral tarkvaraarendusega seotud üldiste probleemidega.

5 Projekti eeldatav haldus valitud metoodikatega

Järgnevas peatükis uurib töö autor eelnevalt esitatud projektist lähtudes, kas antud peatükis vaatluse all olevate metoodikate järgimine oleks vähendanud projekti käigus tekkinud probleemide või raskuste arvu ning seeläbi lühendanud protsessi kulgu. Peatükis kasutatakse projekti võimalike agiilsete omaduste kontrolliks DSDM sobivus-küsimustikku, kristallmeetodit ning Stacey maatriksit. Projekti omaduste analüüsile järgneb projekti sobivus-analüüs agiilsete arendusmetoodikatega nagu Scrum, Kanban ning Extreme Programming (sageli kasutatud lühendina XP).

5.1 Projekti paiknemine traditsioonilise ja agiilse metoodika suhtes

Projektide soovitatava arendusmetoodika tuvastamiseks saab selle iseärasusi, nagu nõuete selgus, tehnoloogiline keerukus, meeskonna suurus, arvestades hinnata, kas projektile sobib traditsiooniline või agiilne lähenemine. Järgnevalt uurib autor kolme erineva ülesehitusega metoodikaga saadud tulemusi. Nendeks metoodikateks on DSDM sobivus-küsimustik, Kristallmeetodi paiknevus-tabel ning Stacey maatriks.

5.1.1 DSDM sobivus-küsimustik

DSDM (*Dynamic Systems Development Method*) ehk dünaamiline tarkvaraarendusmeetod on agiilse lähenemise perekonda kuuluv meetod, mis kirjeldab põhiprintsiibid, võimaldamaks saavutada soovitud funktsionaalsust väikse ajaga, kaasates kontrolli ning probleemide lahendamise komponendid [12].

DSDM poolt määratud põhiprintsiipidele tuginedes saab kontrollida, kas projekt on agiilseks lähenemiseks sobiv. Esitatavad küsimused on järgmised [13]:

1. Kas agiilne filosoofia on projektis vastuvõetav?
2. Kas meeskonnaliikmed on volitatud otsuste tegemiseks?
3. Kas kasutaja/kliendi pidev seotus projektiga on võimalik?
4. Kas astmeline arendus on võimalik?
5. Kas arendajate ja lõpp-kasutajate vaheline vahetu suhtlus on tagatud?

6. Kas meeskonna keskne koosseis on stabiilne?
7. Kas meeskonnaliikmetel on oskused meeskonnatööks?
8. Kas meeskonna suurus võimaldab näost-näkku stiilis suhtlust?
9. Kas projekti osapoolte vahel on usaldus?
10. Kas arendustehnoloogia/-keskkond toetavad astmelist arendust, kiiret prototüüpimist, muudatusi?

Mida rohkem on negatiivseid vastuseid, seda ebatõenäolisem on tulemus, et agiilne lähenemine on projektile sobiv [13].

Käsiteldava projekti põhiselt küsimustele vastates on küsimuste 1, 3, 4, 5, 6, 7, 8, 9 vastused positiivsed, kus projekt võimaldas kasutaja seotust valmiva tootega ning iteratiivset arendus-tsüklit. Samuti ei muutunud projekti käigus ka negatiivselt meeskonna koosseis - projekti algus-faasist tööga kaasatud olnud liikmed jätkasid andmeedastuskanaliga tööd ka selle hilistes etappides ning meeskonna suurus ei pärssinud näost-näkku suhtlemise võimalikkust. Osapoolte vahel oli usaldus, kus peamiseks eesmärgiks oli seatud lõpp-tulemi ootuspärane valmimine.

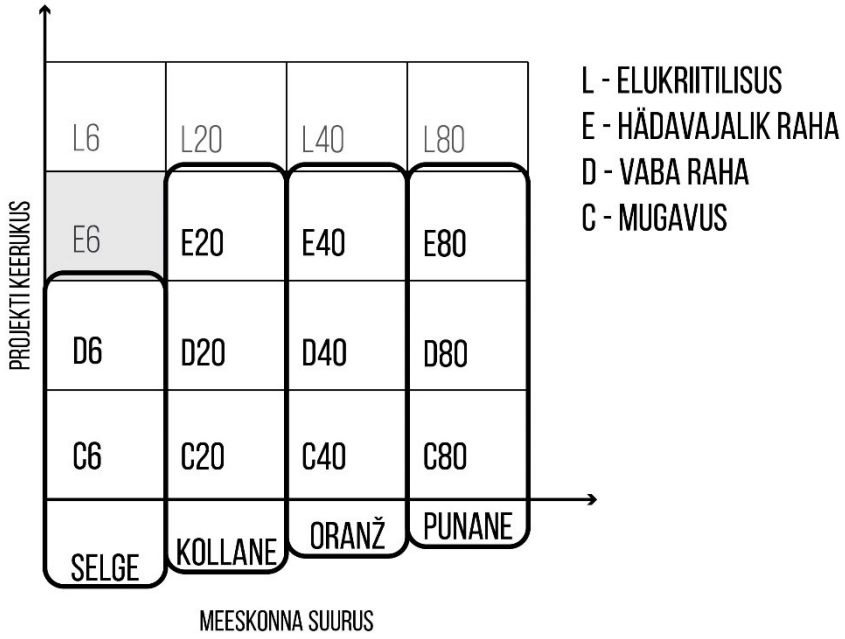
Negatiivse või siis mitte täielikult positiivse vastuse said aga küsimused 2 ja 10, mis on autori hinnangul ka mõnevõrra seotud. Kuigi projekt andis arendusmeeskonna-siseselt selle liikmetele õigusi nii disaini kui ka komponentide lõpliku valiku suhtes, siis just töö käigus ilmnenud selgusetus vormi komponentide mõningate nõuete juures sundis arendusmeeskonna liikmeid lahenduse otsuste valideerimiseks suhtlema (ja ootama nende vastust) kliendiga. Selle põhjuseks võib pidada ka küsimus 10 eitavat vastust, kus esialgu puudusid SAP kontekstis detailsed UI5 komponentidega prototüüpimise vahendid.

Autor järeldab kirjeldatust, et küsimustiku tulemused ei kirjelda projekti sobimatust agiilsetele metoodikatele, küll aga tuleks täheldada esineda võivaid probleeme, mis on seotud prototüüpide madala tasemega.

5.1.2 Kristall-meetod

Teiseks meetodiks leidmaks projektile sobivat arendusmeetodit on kasutada Alistair Cockburni kristall-meetodi tabelit, mis kasutab mõõdikutena projekti keerukust ning kaasatud meeskonna suurust [13].

Järgnev joonis visualiseerib kristall-meetodite ülesehitust.



Joonis 5. Kristallmeetodite ülesehitus [14]

Eelpool toodud joonisel kirjeldavad värvid selge (*Clear,* 1-4 inimest), kollane (*Yellow*, 6-20 inimest), oranž (*Orange*, 20-40 inimest) ja punane (*Red*, 50-100 inimest) kristallmeetodeid, kus värvus viitab meeskonna suurusele – mida tumedam toon, seda suurem meeskond. Selge grupist (*Clear*) välja jääv halli tooni ruut E6 viitab sellele, et selge kristalli meetod ei ole otseselt E-tüüpi projektidele suunitletud, kuid on siiski võimalik nimetatud meetodiga katta. Meeskonna suuruse teljeks tabelis on horisontaalne telg. Joonisel oleva tabeli vertikaalne telg representeerib töö keerukust või tähtsust. Väikse-liikmelised ning kerged projektid kvalifitseeruvad kui selged projektid, samal ajal kui keerukatel ning suurt meeskonda haaravad projektid on punased. Mida suurem on projekti meeskond, seda raskem on järgida agiilsete metoodikate üheks põhimõtteks olevat näost-näkku suhtluse vajadust ning tekib vajadus põhjalikuma kooskõlastatud dokumentatsiooni järele. Samuti töötab ka põhimõte, et mida keerukam ja olulisem on

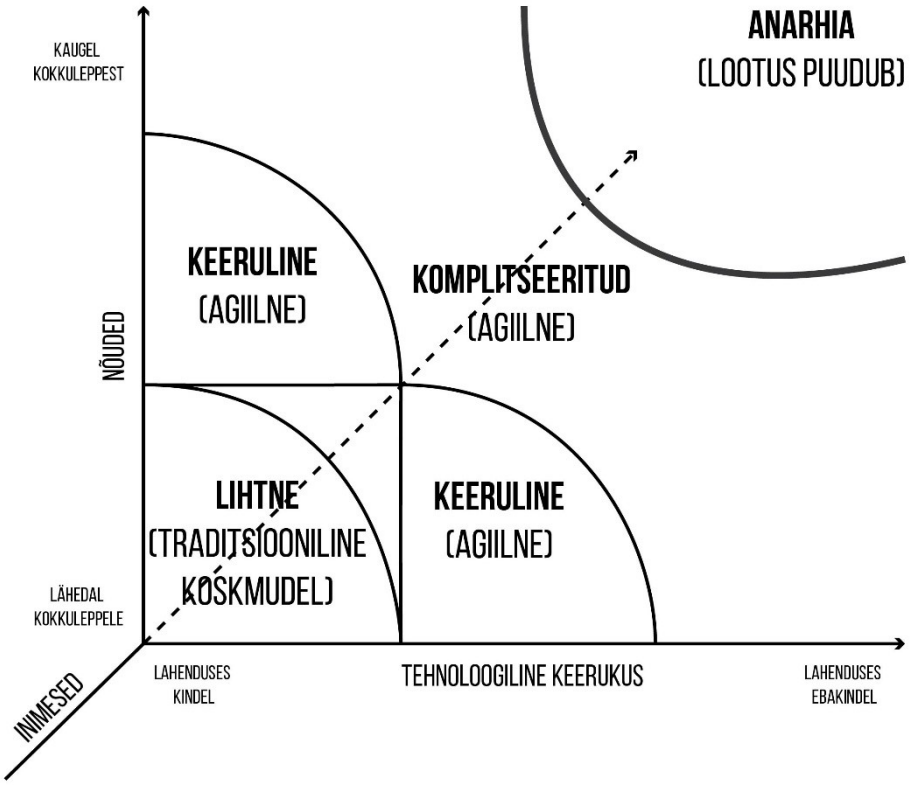
projekt, seda rohkem inimesi projektis kaasatakse. Seetõttu tuleks tumedamate värvitoonide juures kasutada rohkem traditsioonilist lähenemist. Selge kristalli vahemikku kvalifitseeruva projekti puhul võib aga arvestada vabama ning agiilset lähenemist järgiva metoodikaga [14].

Vaadates eelpool toodud joonisel 5 olevat Cockburni tabelit leiab autor, et töös käsitletud projekt kuulub selle väikese meeskonna ning keskmise keerukuse taseme tõttu selge (*Clear*) grupi lahtrisse D6 (keskmine keerukus/tähtsus). Projekti keskmise keerukustaseme põhjenduseks toob autor fakti, et projekti tulemiks oli inimressursside haldust lihtsustava kanali valmimine, mille funktsionaalsus (küll keerulisem) oli olemas ka iga kliendi SAP ERP süsteemis. Seega ei saa projekti käsitleda kui hädavajaliku või elukriitilise projektina, sest see oli ettenähtud pigem kliendi töö protsessi ja kasutusmugavust lihtsustava edendava. Arvestades aga fakti, et projekti tulem oli seotud suurema, suurel hulgal kliendi all-asutusi hõlmava projekti osana, siis käsitles autor projekti tähtsust keskmiseks, sest tulemi hilinemine või projekti ebaõnnestumine mõjutaks omakorda ka kliendi jaoks olulisema ning suurema projekti kulgu.

Saadud tulemuste kohaselt leiab autor, et Cockburni kristall-meetodi tabeli põhjal sobib uuritud projekt agiilsete arendusmetoodikatega hästi.

5.1.3 Stacey maatriks

Kolmanda vahendina võib projekti soovitatava arendusmetoodika hindamiseks kasutada Ralph Stacey 2011 aastal disainitud maatriksit, mille mõõdikuteks on vertikaalteljel paiknev nõuete selgus ning horisontaalteljel kindlus tehnilises keerukuses. Mida suurem on mõõdikute selgus ja kindlus seda suurem on väärtus telgedel (vaata joonis 6) [15].



Joonis 6. Stacey maatriks [15]

Selged nõuded ning tehnoloogilise keerukuse kindlus viitavad lihtsale, vähese keerukus-tasemega projektile ning võib eeldada, et sellistes projektides esineb vähe muutusi ning kasutada saab kindla struktuuriga jäigemaid metoodikaid nagu traditsiooniline kosemudel. Kui aga ilmneb ebakindlust nõuete või tehnoloogiliste aspektide suhtes, on Stacey maatriksi kohaselt tegemist keerulise projektiga ning soovitatav on kasutada paindlikku agiilset metoodikat. Olukordades, kus ühe telje mõõdik on madal ja teise mõõdik kõrgem, on tegemist keerulise lahendusega. Kui kõrged on aga mõlema telje mõõdikud, on tegemist komplitseeritud projektiga. Maatriksil on toodud ka anarhia sektor, kus puudub igasugune kindlus tehnoloogilisest aspektist ning projekti nõuded on täiesti ebaselged. Sellisel juhul viitab maatriks, et tegemist on väga ohtliku projektiga, mille eesmärke on raske või hoopis võimatu saavutada [15].

Asetades töös uuritud projekti antud maatriksi mõõdikutesse, saab väita, et kuna kõik andmeedastuskanali nõuded ei olnud enne arendusfaasi algust fikseeritud, vaid täienesid ka töö käigus (põhjusena kas kliendi poolsest hilisemast täpsustusest või SAP UI5 poolt määratud tehnoloogilistest takistustest) kuulub esitatud projekt keerulise ja komplekse

piirimaile ning soovituslik on agiilne lähenemine. Autori hinnangul ei lähene projekt anarhiale, sest ebatäpsusi esines siiski väheste nõuetega. Samuti ei saa eeldada, et tehnoloogilist keerukust oleks Stacey maatriksil määratud analüüsi etapis väga kõrgeks, sest projekti arendusfaasi alguses ei olnud tehnoloogilise keerukuse probleemid koheselt teada, vaid need ilmusid töö käigus. Siiski tõstaks tehnoloogilise keerukuse taset koheselt akuutsete prototüüpimise vahendite puudumine.

5.2 Metoodikate valik

Põhinedes peatükis 5.1 saadud tulemustele, kus selgus, et töös käsitletud projekt sobinuks kõigi kolme paiknevus-analüüsi põhjal agiilse lähenemisega, võrdlebki autor projekti probleemidest lähtudes olukorda populaarsete agiilsete metoodikatega, kus kriteeriumiteks on:

- Reageerimine muutustele
- Koostöö kliendiga
- Töö prioritiseerimine
- Kiire tootlus

Lähtudes 2011. aastal korraldatud uuringute tulemustest, kus uuriti 35 erineva riigi seast enim kasutatavaid agiilseid arendusmetoodikaid, osutusid vastavalt eelnevalt paika pandud kriteeriumeid arvestades valitud metoodikateks Scrum, Extreme Programming (XP) ja KanBan [2].

2001. aastal loodi 17 inimese poolt agiilsete lähenemiste põhiväärtusteks saanud manifest, mis rõhutab arendus-projektide projekti liikmete ja tellija vahelise funktsioneeriva koostöö vajalikkusele ning väärtuse edastamisele [16].

Agiilse lähenemise neli põhiväärtust on [16]:

- Inimeste ja suhete väärtustamine on olulisem kui töövahendid ja protsessid
- Töötava tarkvara eelistamine dokumentatsioonile
- Kliendi kaasamine olulisem kui lepingulised läbirääkimised
- Muutustega adapteerumine olulisem kui plaani järgimine

Neid põhiväärtusi järgivaid mitmed tänapäevased arendusmetoodikad, nagu Scrum, Kanban ja Extreme Programming klassifitseeritaksegi kui agiilseid metoodikaid [17].

Eelnimetatud agiilsete metoodikate ühisosa tuleneb projekti sihtide väljaselgitamises koos kliendiga, samal ajal kui lõpp-produktil lubatakse ajas muutuda. Töö toimub iteratiivsete tsüklitena, kus iga tsükli lõpus teostatakse selle hinnang. Hinnangu tulemusele lähtudes vormitakse lõpp-produkti paremini vastama kliendi vajadustele, kus võtme-tähtsusega osa on koostööl ning paindlikkusel [17].

5.2.1 Scrum

Scrum on agiilsete arendusmetoodikate perekonda kuuluv metoodika, mis kasutab kindlaks määratud pikkustega iteratsioone ehk sprinte lõpp-toote kvaliteetseks loomiseks, kus iteratsioon antud kontekstis tähendab väikseimat võimalikku töötsüklit, mille jooksul toodetakse töötav tarkvara. Nimetatud metoodika järgib kindlaid reegleid, kohustusi ja koosolekuid mille struktuur metoodika käigus ei muutu [18].

Iga sprindiga on seotud järgmised protsessid [18]:

- Sprindi planeerimine (*Sprint Planning*)
- Igapäevased püstijala-koosolekud (*Daily Scrum Meeting*)
- Sprindi demo (*Sprint Demo*)
- Sprindi tagasivaade (*Sprint Retrospective*)

Sprindi käigus kasutab meeskond ka töö progressi visualiseerivaid jooniseid nagu ülesannete tahvlid ja läbipõlemis-graafikud [18].

Scrum rollid jagunevad järgmiselt [18]:

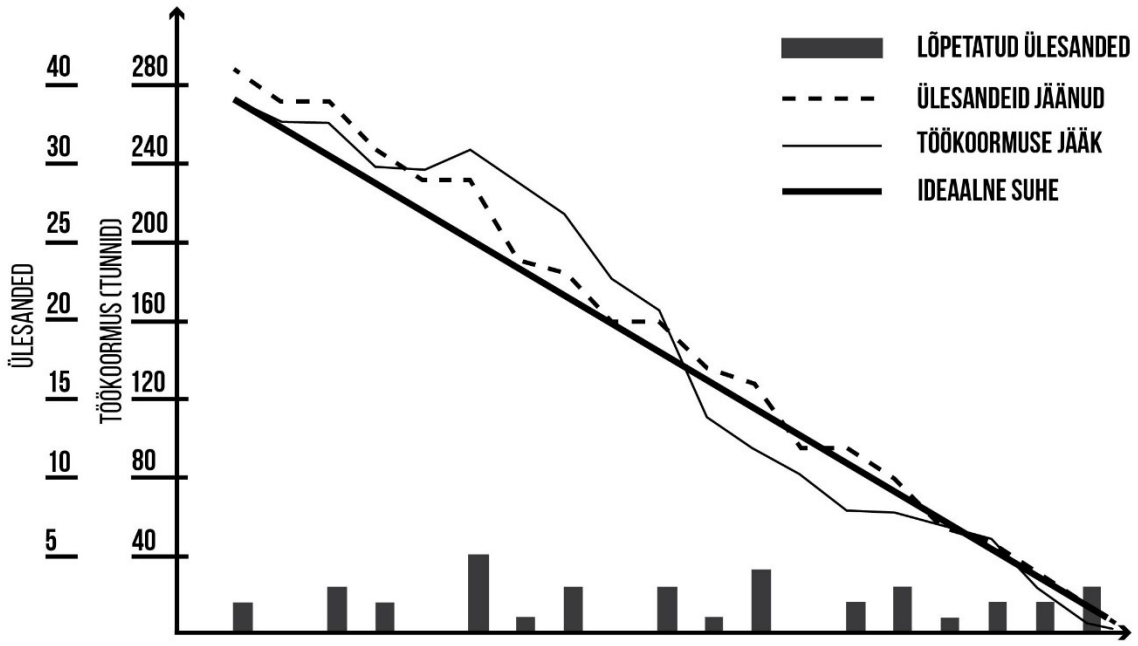
- Toote omanik
- Scrum meister
- Meeskond

Scrumi meeskonnaliikmed kasutatavad töös järgnevaid mõisted [18]:

- Toote kogum (*Product Backlog)*
- Sprindi kogum (*Sprint Backlog)*
- Sprindi läbipõlemisgraafikud (*Sprint Burndown Charts)*

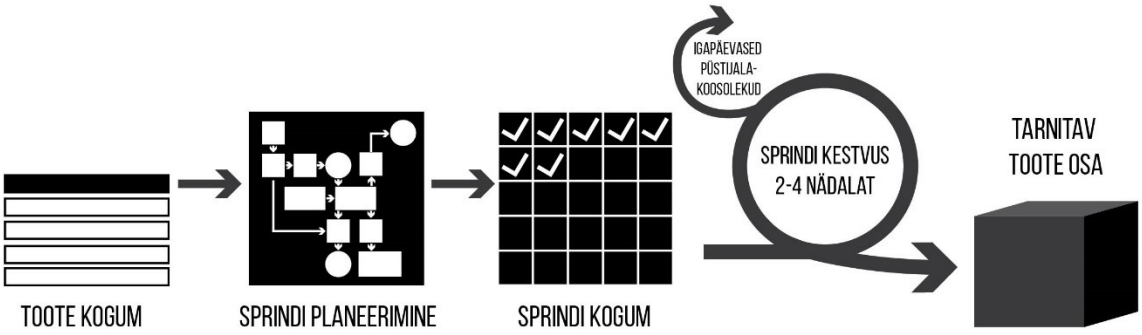
Scrum protsessi kirjeldus

Kui Scrumi lõpp-tulemuseks on tarnitav toode, siis see valmib projekti käigus toodetavate väiksemate sprintide käigus loodud osadena. Esimesena jagatakse lõpp-tulem toote omaniku poolt loodud kasutajalugude (*User story*) põhjal väikesteks tarnitavateks funktsionaalseteks osadeks, mis toote omaniku ning Scrum meeskonna poolt prioritiseeritakse. Lõpp-tulemi osad moodustavadki toote kogumi (*Product Backlog*), kus see siiski ei ole nimekiri osadest mis igal juhul peavad saama realiseeritud, vaid pigem nimekiri soovitud funktsionaalsusest ning vigadest. Toote kogumist valitakse seejärel välja toote omaniku prioriteetseteks määratud osad ning peetakse sprindi planeerimise koosolek (*Sprint Planning*), kus arutatakse millised neist on järgmise sprindi sees saavutatavad ning koostatakse selle kogum (*Sprint Backlog*). Seejärel algabki arendusfaasi etapp ehk sprint, kus valmimisele kuuluvad sprindi kogumisse lisatud osad. Sprint on fikseeritud kestusega ning tavaliselt 2-4 nädalat pikk, mille jooksul peetakse igapäevaseid kuni viietest minuti pikkuseid püstijala-koosolekuid, kus iga meeskonna liige räägib, mis on tema seatud eesmärgid ning millised on selle saavutamist takistavad probleemid. Arenduse käigus kasutatakse ka läbipõlemisgraafikuid (*Burndown Chart*), mis näitavad kogu töö ja tehtud töö suhet ning aitavad seeläbi märgata takistusi ning hinnata sprindi üleüldist kulgu [19].



Joonis 7. Läbipõlemisgraafik [18]

Sprindi lõpus toimub valminud osa esitlus toote omanikule (*Sprint Demo*). Samuti korrastatakse enne järgmise sprindi algust üle toote kogum, kus vaadatakse üle prioriteedid, mille käigus võidakse kogumi osi eemaldada või uusi juurde luua. Arendusfaasi etapi viimane osa on sprindi tagasivaade, kus analüüsitakse sprindi edukust ning kirjeldatakse mis oli positiivne ning negatiivne. Samuti arutatakse, mida võis lõppenud sprindist õppida ning järgmisesse etappi kaasa võtta [19].



Joonis 8. Scrum protsess [18]

Scrum rollide kirjeldus

Toote omanik – omab visiooni lõpp-tulemusest ja jagab seda informatsiooni ülejäänud meeskonnaga. Nimetatud rolliga isik keskendub äri- ja turuvajadustele ning prioritiseerib tegemisele kuuluva töö ning haldab toote kogumit. Toote omaniku ülesanne on teha kindlaks, et kõik meeskonna liikmed mõistavad toote kogumis olevad osasid ning samuti võtta vastu tarnitud toote osad [18].

Scrum meister – tagab meeskonna töö sujumise. Organiseerib koosolekuid toote omanikuga, aitab eemaldada töö käigus esinevaid takistusi, koostöö toote omanikuga toote kogumi sprindi-eelsel ettevalmistamisel. Scrum meister ei oma autoriteeri meeskonnaliikmete üle vaid kontrollib Scrum protsessi [18].

Scrum meeskond – soovitatavalt üks kuni kümme liiget [20], kus töö protsessi iseloomustab pidev koostöö toimumine. Erinevalt traditsioonilisest arendusmeeskonna struktuurist, puuduvad Scrum meeskonnas kindlad rollid nagu programmeerija või testija vaid töö valmib kogu meeskonna ühtse panusena. Scrum meeskond kontrollib iga sprindi kava ning määravad kui palju tööd nad suudavad iga sprindi jooksul realiseerida [18].

Võttes arvesse eelpool kirjeldatud Scrum'i põhilisi karakteristikuid, saab kontrollida, kas töös käsitletav projekt oleks taganud põhivajadused metoodika kasutuselevõtuks.

Tabel 3. Scrumi baas-vajaduste vastavus projektiga

Vajadus	Sobivus projektiga	Kommentaar
Pidev suhtlus kliendiga	Raskendatud	Sobib: Kogu projekti vältel oli tagatud infovahetus kliendiga kasutades selleks kas koosolekuid, telefonikõnesid või e-maile. Mitte-range suhtlusstiil. Ei sobi: raskendatuks jääb suhtlus välise partneriga.
Iteratsioonideks jagatav projekt	Sobib	Vormipõhised sprindid.

Meeskonnasisene suhtlus/ igapäevased püstijala koosolekud	Sobib	Suur avatud tööruum koos meeskonnaruumiga.
Kasutuslugudeks jagatavad ülesanded, agiilset arendust võimaldav lahendus	Sobib	Arenduse jagamine vormipõhisteks komponentideks/funktsionaalsusteks. Teenustepõhine lahendus.

Samuti saab vaadelda Scrum'i sobivust töös käsitletud projekti (kui avaliku projekti) üldtingimustega.

Tabel 4. Projekti sobivus Scrumiga

Vajadus	Sobivus Scrumiga	Kommentaar
Ajaline piiritletus	Sobib	Sprindid võimalik luua kasutades eesmärgi-põhist ajaraamistikku (*timeboxing*)
Väike arendusmeeskonna suurus	Sobib	Metoodika sobib väikestele meeskondadele, suuruses 1-10 [20]
Puuduvad kindlad rollid arendusmeeskonnas	Sobib	Scrum metoodika ei määra arendusmeeskonna rolle [20]

Scrumi kasutusevõtul projekti probleemide esinemise tõenäosus:

Tabel 5. Projekti probleemide esinemise tõenäosus Scrum'i puhul

Probleem	Esinemise tõenäosus	Kommetaar
Uue tehnoloogia kasutuselevõtust tingitud tõrked	Suur	Kuna antud metoodika on vigade käsitlemisel pigem reaktiivne, siis vead esinevad sprindi käigus. Tekkinud vead kaardistatakse esimese sprindi retrospektiivis ning sarnaste tõrgete

		tekkimise tõenäosus järgmistes sprintides langeb.
Hiiliv funktsionaalsuse kasv või suurenemine	Suur	Võimalus sprintide planeerimisel muuta toote kogumit luues sinna uut funktsionaalsust suurendab probleemi esinemise tõenäosust.
Lahenduse madal koormustaluvus	Keskmine	Jagades lõpp-tulemi väikesteks funktsionaalselt tarnitavateks osadeks kaasates toote omaniku arendusprotsessi toote kogumi loomisel ja prioritiseerimisel tekib üldisem parem arusaam ka vormide peidetud keerukusest. Lisades andmeedastuskanali üldist tööd koormavaid nõudeid aga hilisemates iteratsioonides probleemi esinemise tõenäosus jällegi tõuseb.
Vormide ümber disainimise vajadus	Madal	Toote omanik on tegev kogu arendusprotsessi vältel ning osaleb igas sprindijärgsel koosolekul. Samuti lihtsustab asjaolu lihtne muudatuse haldamine kindla kasutusloo tasemel järgmises sprindis.
Prototüüpide liiga madal detailsus	Keskmine	Kliendi kaasatus toote-kogumi loomisel sunnib osapooli kasutusloo tasemel detailsemalt mõtlema. Samas prioriteetsete vormide puhul, kus kliendil puudub kindel kohene hoiak prototüübi nõuete kohta, defineeritakse ebatäpsed tingimused ning tulem võib erineda soovitust [21].
Veebilehitseja tõrked	Suur	Kuna antud metoodika on vigade käsitlemisel pigem reaktiivne, siis vead esineks sprindi käigus. Tekkinud vead kaardistataks esimese sprindi retrospektiivis ning sarnaste tõrgete tekkimise tõenäosus järgmistes sprintides langeb.

Aeglane koostöö välise partneriga	Suur	Kuigi Scrum lihtsustaks suurel määral suhtlust arendusmeeskonna ja kliendi vahel, siis välise partneriga suhtlust ei määratleta.
Liiga väikseliikmeline arendusmeeskond haldamaks arenduse protsessi	Madal	Scrum ei rõhu rollide olemasolule arendusmeeskonnas vaid võimalikult tõhusale suhtlusele. Igapäevased koosolekud koos sprindi kogumi ja isiklike ülesannete haldamisega võimaldab hoida järge käesoleval arenduskäigul ning meeskonnaliikmete lisandumisel lihtsamalt delegeerida ülesandeid käesoleva sprindi kogumist.

5.2.2 Kanban

Kanban metoodika on inspireeritud 1940ndate aastate lõpus Toyota Motoration Corporation inseneri Taiichi Ohno poolt disainitud meetodist, mis oli mõeldud tootmisprotsessi efektiivsuse tõstmiseks, kus kasutatakse tõuke-tõmbe lähenemist läbi visuaalsete kaartide (Kanban tähendab jaapani keeles visuaalset märki või kaarti) [18].

Kanban metoodika loob väärtust informeerides mida ning millal toota ning samal ajal ei nõua suurte muutuste tegemist hetkel toimivas tootmis-süsteemis [18].

Agiilses kontekstis rakenduvad aga samad reeglid, kus inventari mõiste asendub käsil oleva töö kogusega ehk WIP (inglise keeles *work-in-progress*) ning kus tööülesandeid saab juurde lisada vaid siis, kui Kanban metoodika poolt kasutataval tahvlil on selle jaoks ruum vabanenud. Töös olev lubatud ülesannete kogus sõltub arendusmeeskonna suurusest. Selline lähenemine toob juurde paindlikkust, läbipaistvust ja parandab tulemusi [18].

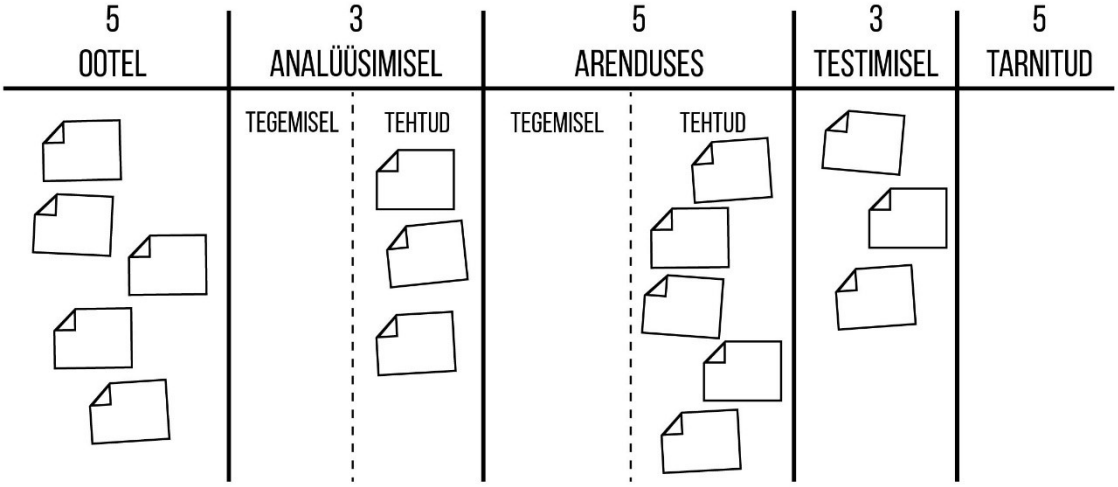
Kanban metoodika keskseks osaks on selle poolt kasutatav töö protsessi käiku visualiseeriv tahvel (kas siis füüsiline või virtuaalne), mis on jaotatud tööülesande etappi kirjeldavate tulpade vahel. Lihtsaim tabel koosneb tulpadest nagu tegemisele kuuluv, käsil olev töö, ning lõpetatud. Tarkvara-arendus projektide puhul kasutatakse tihti aga ka suuremat tulpade arvu, nagu näiteks: Toote kogum, tegemisele kuuluv, arendamisel, testimisel, vastuvõtu-testimisel, valmis [18].

Kanban metoodika põhiprintsiipideks on [18]:

- Töövoo visualiseerimine
- Töös olevate ülesannete arvu piirang (WIP)
- Toovöö pidev mõõtmine ja kontrollimine
- Töövõtted tuleb selgelt defineerida
- Pidev areng läbi väikeste muudatuste

Kanban protsessi kirjeldus

Metoodika kasutusele võtmisel visualiseeritakse töövoog, kus alustatakse kasutajalugudest koostatud tööülesannetest ning lõpetatakse sellest lähtuva funktsionaalsuse tarnimisega. Kanban metoodika puhul määratakse seejärel töös olevate ülesannete arvu piirang (WIP), mis tähendab, et igas töövoo etapis olevate ülesannete arv on piiratud kindale arvule. Kui ülesanne valmib ning lahkub töövoost, teeb see ruumi järgmisele toote kogumis olevale ülesandele. Nii luuakse kergesti hoomatav ning meeskonna ressursse arvestav protsess lõpp-tulemi loomiseks. Sarnaselt Scrum'ile toimuvad Kanban metoodika korral igapäevased püstijala-koosolekud, kus iga meeskonnaliige kirjeldab lühidalt millega ta tegeles, tegeleb ning millega kavatseb tegelema hakata. Erinevalt Scrum'ist ei ole Kanbali puhul tööülesanded määratud kinnistesse iteratsioonidesse, vaid kogu töö on pidevas protsessis läbi tahvlil kasutatud osade, mille piirangud lähtuvad limiteeritud tööülesannetest töövoo etappides. Kanban'i puhul ei pea arendusmeeskond olema ühtsena kõiki protsesse hõlmav ning analüütikud, arendajad ja testijad võivad olla eraldiseisvad. Samuti puuduvad Kanbanil kindlad rollid metoodika haldamiseks ning ajaraamistikust lähtuvad iteratsioonid üheskoos iteratsiooni tagasivaate kohustusliku osaga [18].



Joonis 9. Kanban tabel [22]

Võttes arvesse eelpool kirjeldatud Kanban metoodika põhilisi karakteristikuid, saab kontrollida, kas töös käsitletav projekt oleks taganud põhivajadused metoodika kasutuselevõtuks.

Tabel 6. Kanban baas-vajaduste vastavus projektiga

Vajadus	Sobivus projektiga	Kommentaar
Iteratsioonideks jagatav projekt	Sobib	Võimalikud komponendipõhised kasutajalood
Meeskonnasisene suhtlus	Sobib	Suur avatud tööruum koos meeskonnaruumiga.
Kasutuslugudeks jagatavad ülesanded, agiilset arendust võimaldav lahendus	Sobib	Arenduse jagamine vormipõhisteks komponentideks/funktsionaalsusteks. Teenustepõhine lahendus.
Eraldi analüütikud, arendajad ja testijad	Ei sobi	Esimeste vormide korral ei jaguks piisavalt meeskonnaliikmeid, et kõik protsessid mehitada.

Samuti saab vaadelda Kanban sobivust töös käsitletud projekti (kui avaliku projekti) üldtingimustega.

Tabel 7. Projekti sobivus Kanbaniga

Vajadus	Sobivus Kanbaniga	Kommentaar
Ajaline piiritletus	Ei sobi	Puudub kindlat ajaraamistikku määrav lähenemine
Väike arendusmeeskonna suurus	Ei sobi	Esialgne meeskonna suurus piisavalt väike, et täita tahvli erinevad protsessid
Puuduvad kindlad rollid arendusmeeskonnas	Ei sobi	Kanban tahvli tõuke-tõmbe protsesside rollide määramiseks (nagu analüütik, testija, arendaja) oli esialgu liiga vähe meeskonnaliikmeid

Kanban kasutusevõtul projekti probleemide esinemise tõenäosus:

Tabel 8. Projekti probleemide esinemise tõenäosus Kanban puhul

Probleem	Esinemise tõenäosus	Kommetaar
Uue tehnoloogia kasutuselevõtust tingitud tõrked	Suur	Kuna antud metoodika on vigade käsitlemisel pigem reaktiivne, siis vead esinevad töö käigus. Tekkinud vead kaardistatakse selle esimesel tekkel ning sarnaste tõrgete tekkimise tõenäosus järgmistes sprintides langeb.
Hiiliv funktsionaalsuse kasv või suurenemine	Suur	Puudub ajaraamistiku käsitlus ning muudatusi võib lisada kogu protsessi käigus. Siiski piirab seda mõneti ülesannete arvu piirand töövoos (WIP). Kui klient soovib töösse tuua lisafunktsionaalsust tuleb enne kindlaks teha selle prioriteetsus, mille järel arvestatakse kui

		kiiresti see töösse võetakse. Teades WIP ja prioriteetsuse aspekti tekib mingil määral ülevaade hiilivast funktsionaalsusest ning sellest, et lisanduv töö ei mahu esialgsesse planeeritud aega [23].
Lahenduse madal koormustaluvus	Keskmine	Jagades lõpp-tulemi väikesteks funktsionaalselt tarnitavateks osadeks kaasates toote omaniku arendusprotsessi toote kogumi loomisel ja prioritiseerimisel tekib üldisem parem arusaam ka vormide peidetud keerukusest. Lisades andmeedastuskanali üldist tööd koormavaid nõudeid aga hilisemates iteratsioonides probleemi esinemise tõenäosus jällegi tõuseb.
Vormide ümber disainimise vajadus	Keskmine	Kuigi toote osasid tarnitakse tihti, siis ei defineeri Kanban kindlaid koosolekuid kliendiga, mil osaliselt valminud funktsionaalsus üle vaadatakse (puuduvad iteratsioonid). Probleemi lihtsustab siiski pidev muudatuse haldamine kindla kasutusloo tasemel.
Prototüüpide liiga madal detailsus	Keskmine	Kliendi kaasatus toote-kogumi loomisel sunnib osapooli kasutusloo tasemel detailsemalt mõtlema. Samas prioriteetsete vormide puhul, kus kliendil puudub kindel kohene hoiak prototüübi nõuete kohta, defineeritakse ebatäpsed tingimused ning tulem võib erineda soovitust.
Veebilehitseja tõrked	Suur	Kuna antud metoodika on vigade käsitlemisel pigem reaktiivne, siis kaardistataks sellist tüüpi viga alles selle esinemisel, sarnaste tõrgete tekkimise tõenäosus küll hilisemalt langeb (pideva arengu printsiip).

Aeglane koostöö välise partneriga	Suur	Kanban ei lihtsusta suhtlust välise partneriga
Liiga väikseliikmeline arendusmeeskond haldamaks arenduse protsessi	Suur	Töövoo protsesside mehitamiseks (analüüs, arendus, test) ei jätkunuks esialgselt piisavalt ressursse.

5.2.3 Extreme Programming (XP)

Ekstreemprogrammeerimine (edaspidi XP) on metoodika, mis töötati välja tegelemaks selliste tarkvaraarenduse projektidega, kus traditsiooniline kosemudel ei sobi. XP välja töötatud sobima väikese meeskonna ning ajas muutuva projektiga, kasutades lähenemist, mis seob endas pidevat suhtlust tellijaga ning soodsat muudatuste haldust [24].

Seda võimaldab metoodika seatud viis põhiprintsiipi, milleks on [24]:

1. Kiire tagasiside
2. Luua lihtsaim lahendus – teha vaid seda, mida sooviti ning ei midagi muud
3. Järk-järgulised muudatused – kõige korraga muutmine ei toimi
4. Avatus muudatustele – enamasti puudub ainult üks lahendus
5. Luua kvaliteeti

Esitatud printsiibid suurendavad töö efektiivsust ning ühtlasi selle kvaliteeti. Tagasiside kiire olemus näeb ette õpitu kohandamist ning kohest rakendamist protsessi edasises käigus, koos põhimõttega, et suur osa probleemidest on lihtsasti lahendatavad ning neile tuleb reageerida võimalikult ruttu. Samuti tuleb kõik vajalik luua võimalikult lihtsalt arvestades sellega juba ette, et võimalusel oleks probleemide lahendamine vähem keerukas. Lisaks näeb metoodika ette, et lahendus peaks valmima lühikestes iteratsioonides, mida tuleks tellijale esitada tihti – nii suudetakse XP kohaselt hallata ka tarkvaraarenduse suurimat probleemi, milleks on muudatuste haldus [25].

Samuti näevad põhiprintsiibid ette avatust muudatustele, kus tuleks arvestada sellega, et lahendused ja nendeni viivad teed võivad aja jooksul muutuda ning seetõttu tuleks vältida

vaid ühele võimalikult lahendile keskendumist. Neid punkte järgides tuleb meeles pidada ka asjaolu, et luua võib ainult kvaliteetset koodi, mis vastab kõikidele esitatud nõuetele ning on põhjalikult testitud [24].

XP näeb nende eelpool mainitud põhiprintsiipide täitmist läbi järgnevate lähenemiste kasutamise [26]:

1. Lihtne disain – kood peab olema võimalikult lihtne ja loetav
2. Ühiktestimine ja Testimisest Tulenev Arendus (*Test Driven Development*)
3. Paarisprogrammeerimine
4. Mitte-koormavad töönädalad (maksimum 40 tundi nädalas)
5. Kollektiivne omand – meeskonnaliige võib koodi parendada kogu töö ulatuses
6. Lahenduse pidev integratsioon – lahendust uuendatakse kohe, kui mingi ülesanne saab täidetud. Lühikesed iteratsioonid (tavaliselt 1 nädal)
7. Kodeerimis-standardid – kood kirjutatakse eelnevalt kindlaks määratud reeglite alusel
8. Tööülesanded jagatud võimalikult väikesteks ning lihtsalt kirjeldatud osadeks
9. Planeerimismäng – töö osade planeerimine iteratsiooniti kasutajalugude põhjal
10. Meeskonna ja kliendi vahelist suhtlust võimaldava keskkonna kasutamine (soovitatavalt ühine tööruum kliendiga)

XP protsessi kirjeldus

Sarnaselt varasemalt uuritud Scrum'i protsessile, toimub XP puhul planeerimisfaasis kasutajalugude kirja panemine, kus sisend saadakse kliendiga suhtlemisel. Kasutajalood jagatakse seejärel väiksemateks funktsionaalseteks osadeks ning jaotatakse iteratsioonide vahel ära. Erinevalt Scrum'i iteratsioonist, on XP vastavad tsüklid tavaliselt lühemad (tavaliselt 1 nädal) ning seega tarnitavad osad väiksemad. Enne iteratsiooni arendustegevusega algamist toimub disaini faas, kus pannakse paika kasutatavad standardid ning metafoorid arendusprotsessile. Samuti kasutatakse funktsionaalsete

ülesannete koostamisel kasutajakaarte, kus meeskonnaliikmed saavad kaartidel välja pakkuda lahendusi erinevatele probleemidele ning väljakutsetele ning arutelu käigus valida sobivaimad lahendused. XP eripära tuleneb veel selle detailsemalt määratletud normidest koodi kirjutamisel, kus iga tööülesande arendamine peab algama sellele ühiktesti loomisest ning toodet ei ole lubatud tarnida enne, kuni kõik olemasolevad vastuvõtu-testid saavad läbitud. Samuti tuleb XP puhul ära mainida paarisprogrammeerimise vajadus ning ületundide tegemise keelu, kus viimase juures nähakse põhjust vigade tekkes ja kvaliteedi languses. Viimaseks etapiks XP arendusprotsessis on tagasiside kogumine, kus hinnatakse kliendi rahulolu tarnituga ning tehakse vajadusel muudatusi käimasolevas ning tulevastes iteratsioonides. Kui klient ei ole sooritatuga rahul algab tagasisidest kogutud andmete põhjal ebaõnnestunud osade täiendamine läbi disain-arendus-testimine-tagasiside ahela [27].

Võttes arvesse eelpool esitatud XP põhilisi karakteristikuid, saab kontrollida, kas töös käsitletav projekt oleks taganud põhivajadused metoodika kasutuselevõtuks.

Tabel 9. XP baas-vajaduste vastavus projektiga

Vajadus	Sobivus projektiga	Kommentaar
Pidev suhtlus kliendiga	Raskendatud	Osaliselt sobiv: kuigi puudus otsene võimalus kliendiga samas ruumis töötada, oli kogu projekti vältel oli tagatud infovahetus kliendiga kasutades selleks kas koosolekuid, telefonikõnesid või e-maile. Mitte-range suhtlusstiil. Ei sobi: raskendatud suhtlus välise partneriga
Iteratsioonideks jagatav projekt	Sobib	Võimalikud vormi komponendipõhised iteratsioonid

Meeskonnasisene suhtlus/ igapäevased püstijala koosolekud	Sobib	Suur avatud tööruum koos meeskonnaruumiga.
Kasutuslugudeks jagatavad ülesanded, agiilset arendust võimaldav lahendus	Sobib	Arenduse jagamine vormipõhisteks komponentideks/funktsionaalsusteks. Teenustepõhine lahendus.
Kogu funktsionaalsuse katmine ühiktestidega	Raskendatud	Nõudnuks läbirääkimisi märksa pikema ajakava suhtes
Paarisprogrammeerimine	Sobib	Arendajaid oli alati vähemalt kaks, paarisprogrammeerimist soodustav keskkond
Kodeerimis-standardite määramine	Sobib	
Lahenduse pidev uuendamine	Sobib	

Samuti saab vaadelda XP sobivust töös käsitletud projekti (kui avaliku projekti) üldtingimustega.

Tabel 10. Projekti sobivus XP metoodikaga

Vajadus	Sobivus XP metoodikaga	Kommentaar
Ajaline piiritletus	Ei sobi	Ei ole otseselt käsitletud ajast piiritletud arendust – oht hiilivaks funktsionaalsuseks.
Väike arendusmeeskonna suurus	Sobib	Metoodika sobib väikestele meeskondadele [24].

Puuduvad kindlad rollid arendusmeeskonnas	Sobib	XP metoodika ei määra arendusmeeskonna rolle [26]

XP kasutusevõtul projekti probleemide esinemise tõenäosus:

Tabel 11. Projekti probleemide esinemise tõenäosus XP puhul

Probleem	Esinemise tõenäosus	Kommetaar
Uue tehnoloogia kasutuselevõtust tingitud tõrked	Madal	Võimalus disainifaasis kasutajakaartidega leida eelnevalt erinevaid lahendusi ülesande realiseerimiseks. Kiire vea avastamine arenduse käigus TDD tõttu.
Hiiliv funktsionaalsuse kasv või suurenemine	Suur	Võimalus iteratsioonide planeerimisel muuta projekti nõudeid luues sinna uut funktsionaalsust suurendab probleemi esinemise tõenäosust.
Lahenduse madal koormustaluvus	Keskmine	Jagades lõpp-tulemi väikesteks funktsionaalselt tarnitavateks osadeks ning kaasates toote omaniku tihti toimuvatesse testidesse oleks probleem kiiremini tuvastatud. Lisades andmeedastuskanali üldist tööd koormavaid nõudeid aga hilisemates iteratsioonides probleemi esinemise tõenäosus jällegi tõuseb.
Vormide ümber disainimise vajadus	Madal	Toote omanik on tegev kogu arendusprotsessi vältel ning osaleb igas iteratsiooni lõpus toimuval koosolekul. Samuti lihtsustab asjaolu lihtne muudatuse haldamine kindla kasutusloo tasemel järgmises iteratsioonis.
Prototüüpide liiga madal detailsus	Keskmine	Kliendi kaasatus detailsete kasutajalugude loomisel sunnib osapooli detailsemalt

		mõtlema. Samas prioriteetsete vormide puhul, kus kliendil puudub kindel kohene hoiak prototüübi nõuete kohta, defineeritakse ebatäpsed tingimused ning tulem võib erineda soovitust.
Veebilehitseja tõrked	Keskmine	Kuna nõuded veebilehitsejate suhtes jäid defineerimata, poleks arvatavasti seda esialgu ka testidesse lisatud ning esmased probleemid oleksid siiski avaldunud. Samas probleemide varajane avastamine seoses lühikeste iteratsioonidega ja sellega seotud kasutaja-testidega lihtsustanuks olukorra ulatust.
Aeglane koostöö välise partneriga	Suur	Kuigi XP lihtsustaks suurel määral suhtlust arendusmeeskonna ja kliendi vahel, siis välise partneriga suhtlust ei määratleta.
Liiga väikseliikmeline arendusmeeskond haldamaks arenduse protsessi	Keskmine	XP ei rõhu rollide olemasolule arendusmeeskonnas vaid võimalikult tõhusale suhtlusele. Igapäevased koosolekud hõlbustaksid sarnaselt Scrumile igapäevatööd. Arvestades, et XP on nõudlik oma põhjaliku standardiseerimise ning testide suhtes, võib eeldada, et esmasel väga väikseliikmelisel meeskonnal kuluks XP kõikide nõuete järgimisel kauem aega kui näiteks muude metoodikatega.

5.3 Arendusmetoodikate analüüsi järeldused

Järgnevas jaotises koondab autor kokku arendusmetoodikate sobivus-analüüsis saadud tulemused. Arendusmetoodikatele antaval üldisel hinnangul lähtub autor nende analüüsi juures kasutatud kolmest kriteeriumist, milleks on:

- Projekti sobivus arendusmetoodikaga
- Arendusmetoodika sobivus avaliku projekti üld-tingimustega
- Probleemide lahenemise tõenäosus

Esimese kriteeriumi juures, milleks on projekti sobivus arendusmetoodikaga, oli kõigi kolme metoodika puhul mõni faktor, mis takistas metoodika hindamist kui projektiga täielikult sobivaks. Kõige vähem mitte-sobivusi antud kriteeriumi puhul oli Scrum metoodika puhul, kus ainukeseks raskendavaks asjaoluks sai keeruline suhtlus välise partneriga (pideva suhtluse nõue). Samuti osutus see probleemiks XP puhul, kus takistustena lisandusid veel kliendiga samas ruumis töötamise võimaluse puudumine ning kogu koodi ühiktestidega katmiseks vajalik vähene ressurss. Kui eelmainitud kahe metoodikaga puudusid otsest mitte-sobivust indikeerivad hinnangud, siis Kanban'i puhul sai selleks liiga vähene arendusmeeskonna ressurss metoodika poolt ette-nähtud meeskonna segmenteerimiseks (analüüs, arendus, testimine).

Vaadeldud kriteeriumi järgi sobivaimaks metoodikaks leiab autor seega olema Scrum'i, kus raskendatud olekus oli üks nõue (ülejäänud metoodikate puhul: XP – 2 nõuet olekus raskendatud, Kanban – 1 nõue mitte-sobivas olekus).

Teise kriteeriumi puhul, milleks on arendusmetoodika sobivus avaliku projekti üld-tingimustega, sobis projekti üldiste nõuetega Scrum, kus võrreldes XP metoodikaga, oli tänu ajaraamistiku määratlemisele sobiv ka projekti ajaline piiritletus. Kuna XP ja Kanban metoodika korral otsene ajapiirangute määratlemise käsitlus puudub, hindas autor sellele nõudele vastavust mitte-sobivaks. Projekti iseärasustest tingitud kriteeriumitega täielikult mitte-sobivaks hindas autor aga Kanban metoodika, kus lisanduvalt ajalise piiritluse probleemile, said takistuseks ka meeskonna protsesside vahel

jaotamise vajadus (liiga väike arendusmeeskond) ning arendusmeeskonnas puuduvad rollid (puudusid kindlaks määratud analüütikud, arendajad, testijad).

Seega hindab autor ka teise kriteeriumi sobivaimaks arendusmetoodikaks Scrum metoodika, mille juures oli positiivseks teguriks võrreldes teiste vaadeldud metoodikatega ajaraamistiku käsitlus.

Kolmanda kriteeriumiga sobivuse arvestamisel kasutab autor punktisüsteemi, kus probleemide lahenemise tõenäosuse punktid arvutatakse vastavalt metoodikate analüüsis toodud probleemide esinemise tõenäosuse järgi (madal, keskmine, suur). Hinnangu andmisel arvutatakse punktid järgmiselt:

- Madal = 1 punkt (lahendatud probleem)
- Keskmine = 0,5 punkti (osaliselt lahendatud probleem)
- Suur = 0 punkti (lahendamata probleem)

Esitatud arvutuses on käsitletud probleemid võrdsete kaaludega.

Tabel 12. Projekti probleemide lahenemise tõenäosus arendusmetoodikatega

Arendusmetoodika	Punktid	Lahendatud probleemid	Lahendamata probleemid
Scrum	3	2	4
Kanban	1,5	0	5
XP	4	2	2

Jagatud punktide tulemusel selgub, et XP poolt rakendatavate printsiipide kasutamisel oleks kõige suurem positiivne mõju projektis tekkinud probleemide lahendamisel. Tabelis 12 toodud tulbas „lahendatud probleemid“ kuvatakse probleemide arvu, mille esinemise tõenäosus metoodika kasutuselevõtul olnuks madal ning samaväärselt on „lahendamata probleemid“ tulbas olev arv seotud probleemidega, mille esinemise tõenäosus oleks jäänud suureks. XP edukus probleemide lahendamisel peitub selle põhjalike ühiktestide, lühikeste iteratsioonide, arendusele eelneva meeskonnasisese arutelu ning kliendi kaasamise tulemusel.

Siiski oli meeskonna suurust käsitleva probleemi tõenäosus XP puhul suurem kui Scrum metoodikaga, kuna kõikide XP printsiipide järgimine oleks olnud esialgses kaheliikmelises meeskonnas raskendatud. Kanban metoodika kasutamine tõi küll kaasa osalisi lahendusi tekkinud probleemidele, kuid autori hinnangul olnuks esitatud takistuste tekkimise tõenäosus antud metoodikaga siiski suurem kui eelnevalt kirjeldatud lähenemiste korral.

Arvestades kolme kriteeriumi analüüsist saadud tulemusi, sobinuks autori hinnangul projektiga kõige paremini Scrum arendusmetoodika, sest esitatud metoodika puhul ei ole suuri takistusi selle sobivusel projekti iseloomuga ning samuti ei takista projekti iseärasused metoodika põhilisi printsiipe järgimast. Lisanduvalt võib probleemide analüüsist järeldada, et metoodika vähendaks oluliselt mõningate takistuste esinemise tõenäosust või vähemalt vähendaks nende mõjude edasist ulatust projektis.

5.4 Projekti probleemide juur-põhjuste võimalikud lahendused

Käesolevas jaotises toob autor välja võimalikud lahendused peatükis 4 käsitletud probleemide juur-põhjustele, arvestades ka agiilsete arendusmetoodikate analüüsi tulemusi.

1.1 Juur-põhjus: Esimeste vormide prototüüpide madal detailsusaste.

1.2 Lahendus: Agiilse arendusmetoodika, nagu Scrum, kasutuselevõtt ning kliendi kaasamine kasutajalugude defineerimisel.

2.1 Juur-põhjus: Hiiliv funktsionaalsuse kasv.

2.2 Lahendus: Agiilse arendusmetoodika kasutuselevõtul funktsionaalsuse lisandumisega arvestamine, kasutades samal ajal liigse funktsionaalsuse märkamiseks läbipõlemisgraafikuid ning ajaraamistikku. Scrum sprindi-järgsed tagasivaated.

3.1 Juur-põhjus: Vähene riskianalüüs.

3.2 Lahendus: Varuda aega lahenduse teostamise põhjalikuks riskianalüüsiks analüüsifaasis, kus arvestatakse nii kristall-meetodi kui ka Stacey maatriksi hindamis-kriteeriume nagu nõuete kindlus, tehnoloogiline keerukus, projekti keerukus ning meeskonna suurus.

4.1 Juur-põhjus: Tööülesannete liiga madal segmenteeritus arendusmeeskonnas.

4.2 Lahendus: Agiilse arendusmetoodika, nagu Scrum, kasutuselevõtt, mille käigus jagatakse tööülesanded kasutajalugude põhjal väikesteks, kergesti hallatavateks funktsionaalseteks osadeks.

5.1 Juur-põhjus: Kliendi liiga hiline testimisse kaasamine.

5.2 Lahendus: Agiilse arendusmetoodika, nagu Scrum, kasutuselevõtt, kus kasutaja tagasisidet kogutakse pärast igat lühikest sprinti.

6.1 Juur-põhjus: Kliendi esialgne väike panus vormide prototüüpide koostamisel.

6.2 Lahendus: Agiilse arendusmetoodika, nagu Scrum, kasutuselevõtt ning kliendi kaasamine kasutajalugude defineerimisel. Prototüüpide loomise sprint.

7.1 Juur-põhjus: Arendusmeeskonna vähene varasem kogemus projektiga seotud mooduliga.

7.2 Lahendus: Detailselt kirjeldatud kasutajalood (kasutades Scrum'i), toote kogum (*product backlog*) kliendi hallata.

8.1 Juur-põhjus: Spetsifikatsiooni loomine puudulike nõuetega.

8.2 Lahendus: Detailselt kirjeldatud kasutajalood (kasutades Scrum'i), toote kogum (*product backlog*) kliendi hallata. Agiilse arendusmetoodika kasutuselevõtul funktsionaalsuse lisandumisega arvestamine, kasutades samal ajal liigse funktsionaalsuse märkamiseks läbipõlemisgraafikuid ning ajaraamistikku. Scrum sprindi-järgsed tagasivaated.

9.1 Juur-põhjus: Raskendatud suhtlus välispartneriga.

9.2 Lahendus: Kuna agiilne lähenemine väärtustab osapoolte-vahelise suhtluse tõhusust, siis tuleb ka välispartnerit kaasata iga iteratsiooni ülevaatusesse (näiteks Scrum puhul). Siiski ei paku otseselt ükski uuritud arendusmetoodika otsest lahendust väliste partneritega suhtlemiseks.

Analüüsides esitatud probleemide juur-põhjuseid ning nende lahendusi järeldab autor, et avaliku sektori projektide käsitluses võimaldab töös analüüsitud vigu aidata vältida kahe

aspektiga arvestamine, milleks esimene on sobiva arendusmetoodika valimine ning teine tõhusa riskianalüüsi teostamine projekti planeerimisel, kus riskianalüüs aitaks vältida vigade tegemist enne arendusetapi algust ning sobiv arendusmetoodika nende tegemist selle käigus. Täiendavalt selgus, et uuritud projektiga sobinuks SAP UI5 tehnoloogia uudsusest, nõuete ebatäpsusest ning lahenduse üldise keerukuse tõttu hästi Scrum arendusmetoodika, mis oma paindliku lähenemisega suutnuks edukamalt hallata ajas täienevaid nõudeid ning väikestest funktsionaalsetest osadest koosnevate lühikeste sprintide läbi luua ülevaade valmivast tootest nii tellija kui ka tarnija poolel, samal ajal kaotamata kontrolli ajaliste piirangute üle.

6 Kokkuvõte

Käesoleva magistritöö põhieesmärkideks oli läbi juhtumianalüüsi leida töös käsitletud projekti probleemide juur-põhjused ning pakkuda neile võimalikud lahendused. Nimetatud eesmärkide saavutamiseks teostati käsitletud projekti analüüs kahes etapis, kus esimeses etapis koostati lähtudes töös uuritud projekti probleemidele Ishikawa põhjus-tagajärg diagramm ilmnenud probleemide juur-põhjuste leidmiseks. Teises etapis vaadeldi projektile sobiva arendusmetoodika paiknevust skaalal traditsiooniline-agiilne ning saadud tulemustele põhinedes teostati kolme agiilse arendusmetoodika sobivus-uuring.

Töös sooritatud analüüsi olulisemate tulemustena toob autor välja magistritöös uuritud projekti sobivuse agiilse lähenemisega, mille järeldus tulenes kolme erineva projekti paiknevust määratleva metoodika analüüsil saadud tulemustel. Samuti järeldas autor, et kui projekti arendusfaasis oleks kasutatud Scrum arendusmetoodikat, võimaldanuks see vältida või leevendanud mitmeid arendusfaasis tekkinud probleeme. Täiendavalt tuli põhjus-tagajärg uuringust välja projektist puudunud põhjaliku riskianalüüsi vajalikkus, mis võimaldanuks paremini arvestada projekti käigus tekkinud tehnoloogiliste probleemide ning muutuvate nõuetega.

Olulisemate järeldustena toob autor välja, et vaatamata projekti pikale ning üpris põhjalikule analüüsifaasile ei olnud võimalik täielikult fikseerida ebakindlaid nõudeid ning keerukat tehnoloogilist lahendust (lähtudes Stacey maatriksi parameetritest) ning sellistel juhtudel oleks tark kasutada agiilseid lähenemisi, nagu Scrum.

Lähtudes arendusmetoodikate sobivuse analüüsist järeldab autor lisanduvalt, et esialgne ajakava projekti vormide teostamiseks oli liiga lühike, et mahutada lisanduvat keerukust hilisemates faasides, kus ka XP sobivusanalüüs viitas, et võimalike probleemide lahendamiseks kasutatavate printsiipide jaoks napiks ressursse.

Kolmanda järeldusena toob autor välja nii Standish grupi uuringu kui ka Ishikawa analüüsi ning agiilsete metoodikate printsiipide poolt rõhutatud kliendi kaasamise vajalikkuse projekti kõikides faasides.

Antud töö edasiarendamise ühe võimalusena näeb autor toodud tulemuste ning järelduste rakendamist sarnases projektis. Samuti näeb autor võimalust töös käsitletud

arendusmetoodikate Scrum ja XP koos kasutamise analüüsi sarnases projektis ning võimaliku hübriid-metoodika sünteesimist. Täiendavalt saab uurida ka agiilsete arendusmetoodikate rakendamist juhtudel, kus otsene näost-näkku koosoleku võimalus on raskendatud (töös käsitletud väline partner).

Leides töös uuritud projekti probleemide juur-põhjused ning nende võimalikud lahendused koos kõige sobivama arendusmetoodikaga, leiab autor, et töös püstitatud eesmärgid said täies mahus täidetud, mööndusega, et pakutud lahendused on teoreetilised ning töö ei käsitlenud nende toimimist praktikas. Sellegipoolest on käsitletud magistritöös saadud tulemused olulised töö autori jaoks, sest magistritöö protsess pakkus autorile paremat ülevaadet sarnastes tingimustes ilmnevatest peamistest takistustest ning vahendid tulevastes projektides neid vältida.

Kasutatud kirjandus

[1] Majandus- ja Kommunikatsiooniministeerium, „Eesti Infoühiskonna Arengukava 2020," 2015. [Võrgumaterjal]. https://www.mkm.ee/sites/default/files/elfinder/article_files/eesti_infouhiskonna_arengukava.pdf.

[2] A. Gayane, „Survey of Agile Tool Usage and Needs," 2011. [Võrgumaterjal]. https://agilealliance.org/wp-content/uploads/2016/01/Agile2011_GA-low-q-2.pdf.

[3] J. Crear, L. Gesmer, J. Johnson, J. Lynch, H. Mulder, T. Mulder ja L. Vianna, „Standish Group Report," 2015. [Võrgumaterjal]. https://www.projectsmart.co.uk/white-papers/chaos-report.pdf.

[4] Beedle, Bennekum, Cockburn, Cunningham, Fowler, Highsmith, Hunt, Jeffries, Kern, Marick, Martin, Schwaber, Sutherland ja Thomas, „Agile Manifesto," 2001. [Võrgumaterjal]. http://agilemanifesto.org/principles.html.

[5] Riigiteataja, „Riigihangete seadus," 02 04 2015. [Võrgumaterjal]. https://www.riigiteataja.ee/akt/123022011040.

[6] Shared Services Canada, „What prevents large IT projects from being successful?," 2013. [Võrgumaterjal]. http://www.ssc-spc.gc.ca/media/documents/ae-ce-eng.pdf.

[7] Riigikontroll, „Riigi infosüsteemide arendusprotsessi tulemuslikkus," 12 02 2010. [Võrgumaterjal]. http://www.riigikontroll.ee/DesktopModules/DigiDetail/FileDownloader.aspx?FileId=11071&AuditId=2128.

[8] P. Patanakul, „Managing large-scale IS/IT projects in the public sector: Problems and causes leading to poor performance," *Elsevier,* 2014.

[9] Tallinna Linnakantselei, „SAP Portal andmeedastuskanali analüüsifaas," 2014.

[10] A.-A. Yazdani ja R. Tavakholi-Moghaddam, „The International Journal of Advanced Manufacturing Technology,"*Integration of the fish bone diagram, brainstorming, and AHP method for problem solving and decision making—a case study*, Springer, 2012, lk 651-657.

[11] J. Cadle, D. Paul ja P. Turner, „Business Analysis Techniques,"*99 Essential Tools for Success*, BCS, The Chartered Institute, 2014, lk 74-75.

[12] J. Stapleton, DSDM: Business Focused Development, Pearson Education, 2003.

[13] M. Griffiths, „Agile Suitability Filters," 2007. [Võrgumaterjal]. http://leadinganswers.typepad.com/leading_answers/files/agile_suitability_filters.pdf.

[14] A. Cockburn, „Agile Software Development,"*The Cooperative Game*, Addison-Wesley Professional, 2006, lk 335-351.

[15] B. Kurtulaj, „Why Stacey Matrix model can help you understand the applicability of Agile to your organisation," 2015. [Võrgumaterjal]. https://www.linkedin.com/pulse/why-stacey-matrix-model-can-help-understand-agile-baf-kurtulaj-mba.

[16] Beedle, Bennekum, Cockburn, Cunningham, Fowler, Highsmith, Hunt, Jeffries, Kern, Marick, Martin, Schwaber, Sutherland ja Thomas, „Agile Alliance,“ 2001. [Võrgumaterjal]. https://www.agilealliance.org/agile101/the-agile-manifesto/.

[17] R. Coffin ja D. Lane, „A Practical Guide to Seven Agile Methodologies,“ 11 10 2006. [Võrgumaterjal]. http://www.devx.com/architect/Article/32761.

[18] „Agile vs Scrum vs Waterfall vs Kanban,“ 2015. [Võrgumaterjal]. https://www.smartsheet.com/agile-vs-scrum-vs-waterfall-vs-kanban.

[19] I. Năftănăilă, „Critical analysys of the scrum project management methodology,“ 2008, lk 435-440. [Võrgumaterjal]. http://steconomice.uoradea.ro/anale/volume/2008/v4-management-marketing/077.pdf.

[20] L. Rising ja N. S. Janoff, „The SCRUM software development process for small teams,“ IEEE Software, 2000, lk 26-32.

[21] J. Highsmith ja A. Cockburn, „Agile Software Development,“*The Business of Innovation*, IEEE Computer Society Press, 2001, lk 120-122.

[22] D. Peterson, „What is Kanban?,“ 2009. [Võrgumaterjal]. http://kanbanblog.com/explained/.

[23] P. Stapelson, „Agile Extension to the BABOK®Guide,“ International Institute of Business Analysis, 2013, lk 17-18.

[24] K. Beck, Extreme Programming Explained: Embrace Change, Addisson-Wesley, 2000.

[25] D. Wells, „Agile Software Development,“ 2009. [Võrgumaterjal]. http://www.agile-process.org/.

[26] J. Hunt, „Agile Software Construction,“ Springer, 2006, lk 69-87.

[27] N. Nayab, „The Extreme Programming Life Cycle,“ 2011. [Võrgumaterjal]. http://www.brighthubpm.com/methods-strategies/88996-the-extreme-programming-life-cycle/.