TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond
Tarkvarateaduse instituut

Julia Švedova 111903IABB

AGIILSETE ARENDUSMETOODIKATE RAKENDAMISE ANALÜÜS LOGISTIKA LAHENDUSI PAKKUVA FIRMA NÄITEL

Bakalaureusetöö

Juhendaja: Karin Rava
MSc. Eng

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: Julia Švedova

16.05.2017

Annotatsioon

Lõputöö eesmärgiks on analüüsida Scrum ja Scrumban arendusmetoodikate rakendamist logistika lahendusi pakkuva ettevõtte ühe arendusmeeskonna näitel selgitamaks välja, miks Scrum metoodika rakendamine ei osutunud vaadeldava meeskonna jaoks edukaks ning kuivõrd edukas on olnud Scrumban metoodika rakendamine. Lõputöö kirjutamise hetkel on Scrumban arendusmetoodika olnud meeskonnal kasutusel üks aasta ning enne Scrumbanile üleminekut rakendas vaadeldav meeskond Scrum metoodikat mitme aasta vältel.

Töös analüüsitakse probleeme, mis võivad esineda Scrum ja Scrumban arendusmetoodikate rakendamisel, tuvastamaks nende probleemide põhjusi. Samuti viiakse läbi Scrum ja Scrumban metoodikate sobivuse hindamist vaadeldavale arendusmeeskonnale, võttes aluseks analüüsitavate metoodikate põhimõtted ja praktikad ning töö teoreetilises osas formuleeritud agiilsete arendusmetoodikate rakendamise edu võtmefaktorid.

Töö tulemusena on tuvastatud Scrum ja Scrumban arendusmetoodikate rakendamisel esinenud probleeme ning analüüsitud nende põhjusi. Scrum metoodika rakendamise analüüsi tulemusena on järeldatud, et selle metoodika ebaõnnestumise põhjusteks on eelkõige tooteomaniku madal pühendumus, meeskonna liikmete ükskõiksus arendusprotsessi suhtes ning üldine distsiplineerimatus valitud metoodika rakendamisel. Scrumban metoodika rakendamise analüüsi tulemusena on järeldatud, et Scrumban metoodika sobib vaatluse all olevale meeskonnale ning et meeskond saab selle metoodika rakendamisega jätkata. Scrumban metoodika rakendamisel tuvastatud probleemide põhjuste analüüsi tulemusena on tehtud konkreetsed ettepanekud nende probleemide lahendamiseks.

Lõputöö on kirjutatud eesti keeles ning sisaldab teksti 58 leheküljel, 6 peatükki, 4 joonist, 1 tabelit.

Abstract

Analysis of the Implementation of Agile Software Development Methodologies by the Example of a Logistics Solutions Provider

The aim of the thesis is to analyze the implementation of Scrum and Scrumban methodologies by one of the software development teams in the company providing logistics solutions. The purpose of the analysis is to understand why the implementation of Scrum turned out to be unsuccessful for the team and to determine whether the implementation of Scrumban has been successful so far. By the time of writing the thesis, the team under examination has implemented Scrumban for a year. Before adopting Scrumban the team used to implement Scrum for a couple of years.

The thesis examines the issues that might arise when implementing Scrum and Scrumban methodologies with the aim of identifying the root causes of the issues. In addition, Scrum and Scrumban suitability for the team under investigation is evaluated based on the principles and practices of Scrum and Scrumban and the success factors of agile methodologies implementation determined in the theoretical part of the thesis.

As a result of the thesis, the issues encountered during the implementation of Scrum and Scrumban are identified and the root causes of the issues are analyzed. The main conclusion made as a result of Scrum implementation analysis is that the main reasons behind the failure with Scrum were low commitment of the product owner, the development team's indifference towards the development process and general lack of discipline in adoption and implementation of the selected agile methodology.

The main conclusion made as a result of Scrumban implementation analysis is that Scrumban suits the team under examination well and the team can continue with Scrumban implementation. As a result of the analysis of the root causes of the issues identified during Scrumban implementation, several specific suggestions were made for solving the issues found.

The thesis is in Estonian and contains 58 pages of text, 6 chapters, 4 figures, 1 table.

Lühendite ja mõistete sõnastik

BU	*Business Unit*, ettevõtte äripool
IT	*Infotechnology Unit*, ettevõtte infosüsteemide arendamisega tegelev pool
Multitegumtöö	*multitasking*, mitme asjadega (tööülesannetega) korraga tegelemine
vs	*versus*, vastu
WIP	*Work In Progress*, töös olevad tööd (tööülesanded)

Sisukord

Jooniste loetelu

Tabelite loetelu

1 Sissejuhatus

Agiilne tarkvaraarendus on saavutanud väga suure populaarsuse viimase kümne aasta jooksul [1]. Selles on suur rõhk rakendataval agiilsel arendusmetoodikal, kuna üldjuhul just arendusmetoodika määrab, kuidas teostatakse arendusprotsessi ning kuivõrd see protsess toetab agiilse tarkvaraarenduse põhimõtteid.

Tänapäeval eksisteerib palju erinevaid agiilseid arendusmetoodikaid, millest igaüks pakub oma meetodeid ja praktikaid agiilse tarkvara arendusprotsessi teostamiseks. On üldtunnustatud, et ei eksisteeri sellist agiilset arendusmetoodikat, mis sobiks hästi igas situatsioonis [2]. Seetõttu tuleb vastava arendusmetoodika valimisel lähtuda konkreetse arendusprojekti eripärast ja vajadustest ning metoodika sobivuse hindamisel arvestada mitte ainult metoodika potentsiaalse kasuga, vaid ka võimalike takistustega selle rakendamisel.

Agiilse arendusmetoodika valik ei ole sugugi triviaalne ülesanne ning sobivaima arendusmetoodika leidmiseks peab arendusmeeskond mõnikord katsetama mitme erineva metoodika rakendamist. Käesolevas lõputöös analüüsitakse agiilsete arendusmetoodikate rakendamist ühe konkreetse arendusmeeskonna näitel, mis püüabki lahendada sobivaima arendusmetoodika leidmise mittetriviaalset ülesannet.

1.1 Taust ja probleem

Käesoleva lõputöö teema on ajendatud autori kogemusest agiilsete arendusmetoodikate rakendamisel, millest on kasvanud soov aru saada, mis võib olla agiilsete metoodikate rakendamise ebaõnnestumise põhjusteks ning millistele olulistele faktoritele tuleb nende rakendamisel eelkõige tähelepanu pöörata, et rakendatav metoodika tooks kasu.

Töös vaadeldakse logistika lahendusi pakkuva ettevõtte ühte arendusmeeskonda, kus töö autor täidab süsteemianalüütiku rolli. Mõned aastad tagasi otsustas antud arendusmeeskond ümber kujundada oma traditsioonilise arendusprotsessi agiilseks arendusprotsessiks, võttes kasutusele Scrum arendusmetoodika. Scrum metoodika

rakendamine ei osutunud aga edukaks ning arendusprotsessis esines mitmeid probleeme, mille tõttu otsustas meeskond minna Scrumban metoodikale üle. Käesoleva lõputöö kirjutamise hetkel rakendab veel vaadeldav meeskond Scrumban metoodikat, kuid arendusprotsessis ikka esineb teatud probleeme.

Käesolev lõputöö on koostatud 2017. aasta kevadel Tallinnas ning selles analüüsitakse Scrum ja Scrumban arendusmetoodikate rakendamist vaatluse all olevas arendusmeeskonnas selgitamaks välja, miks Scrum metoodika rakendamine ei osutunud edukaks ning kuivõrd edukas on siiani olnud Scrumban metoodika rakendamine. Analüüsi perioodiks on august 2015 kuni aprill 2017.

1.2 Ülesande püstitus

Lõputöö eesmärkideks on:

- jõuda arusaamisele, millest on tingitud Scrum metoodika rakendamisel esinenud probleemid;
- jõuda arusaamisele, millest on tingitud Scrumban metoodika rakendamisel esinevad probleemid;
- välja selgitada, mis on need olulisemad faktorid, mis mõjutavad agiilse arendusmetoodika rakendamise edukust;
- jõuda järeldusele selle kohta, kas vaatluse all olev meeskond saab ka edaspidi jätkata Scrumban metoodika rakendamisega;
- teha ettepanekuid arendusmetoodikate rakendamise analüüsi tulemusena tuvastatud probleemide lahendamiseks, tuginedes analüüsi järeldustele.

Käesolev töö on kasulik eelkõige vaatluse all olevale arendusmeeskonnale, kuna töös tehtud analüüsi järeldused ja ettepanekud võimaldavad meeskonnal tuvastada parenduskohti oma arendusprotsessis ning teha selles vastavad täiendused. Samuti võib käesolev töö pakkuda huvi ka teistele arendusmeeskondadele, kes kaalutlevad Scrum või Scrumban metoodika kasutuselevõtmist ning soovivad teada, millistele faktoritele tuleb nende rakendamisel enim tähelepanu pöörata, et metoodikate rakendamine õnnestuks.

1.3 Metoodika

Käesoleva lõputöö eesmärkide saavutamiseks viiakse läbi järgnevad tegevused:

- töötatakse läbi teoreetilise materjali ja juhtumiuuringud järgmistel teemadel:
    - agiilse arendusprotsessi olemus ja põhimõtted,
    - Scrum ja Scrumban arendusmetoodikate põhimõtted,
    - agiilse arendusmetoodika sobivuse hindamine;
- formuleeritakse agiilsete arendusmetoodikate rakendamise edu võtmefaktorid, tuginedes läbitöötatud materjalidele;
- analüüsitakse vaadeldava arendusmeeskonna poolt teatud perioodil teostatud arendusprotsessi vastavust sellel perioodil kasutuselolnud agiilsele arendusmetoodikale;
- analüüsitakse vaadeldaval arendusmeeskonnal teatud perioodil kasutuselolnud arendusmetoodika rakendamist läbi formuleeritud agiilsete arendusmetoodikate rakendamise edu võtmefaktorite;
- tehakse järeldused arendusmeeskonnal teatud perioodil kasutuselolnud arendusmetoodika rakendamise kohta, tuginedes analüüsi tulemustele.

1.4 Ülevaade tööst

Töö esimeses osas antakse ülevaate agiilsest tarkvaraarendusest üldiselt. Eeskätt tuuakse välja agiilse arendusprotsessi põhimõtted ning selle eelised ja puudused. Siis antakse põhjaliku ülevaate töös analüüsitavate arendusmetoodikate – Scrum ja Scrumban – põhimõtetest ja praktikatest. Lõpuks formuleeritakse agiilsete arendusmetoodikate rakendamise edu võtmefaktorid ning tutvustatakse lähenemist agiilse arendusmetoodika sobivuse hindamisele, mida kasutatakse töö analüüsi osas.

Töö teises osas tutvustatakse vaatluse all olevat meeskonda ning arendatavaid süsteeme.

Töö kolmandas osas analüüsitakse Scrum ja Scrumban metoodikate rakendamist vaatluse all oleva meeskonna näitel. Mõlema arendusmetoodika rakendamise analüüsimisel esmalt vaadatakse, kuivõrd korralikult järgis vaadeldav meeskond rakendatava arendusmetoodika põhimõtteid ning seejärel analüüsitakse selle metoodika rakendamist läbi töö teoreetilises osas formuleeritud agiilsete arendusmetoodikate edu võtmefaktorite.

Töö neljandas osas kirjeldatakse arendusmetoodikate rakendamise analüüsi tulemusena tehtud järeldusi ning tehakse ettepanekuid tuvastatud probleemide lahendamiseks.

2 Agiilne tarkvaraarendus

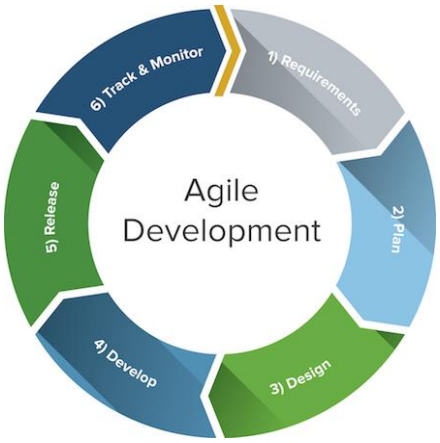
Agiilne tarkvaraarendus baseerub põhimõtetel, mille järgi tarkvara nõuded ja lahendused arenevad inkrementaalselt läbi pideva koostöö iseorganiseeruvate multifunktsionaalsete arendusmeeskondade ja tarkvara tellijate vahel. Agiilne tarkvaraarendus võimaldab kiiret ja paindlikku muutustele reageerimist läbi kohanduva planeerimise, iteratiivse arendusprotsessi, kiire ja inkrementaalse tarkvara tarnimise ning pideva arendusprotsessi hindamise ja täiustamise [1].

Käesolevas peatükis antakse ülevaade agiilse arendusprotsessi olemusest ja põhimõtetest ning töös analüüsitavatest agiilsetest arendusmetoodikatest (Scrum ja Scrumban). Viimases alapeatükis defineeritakse agiilsete arendusmetoodikate eduka rakendamise võtmefaktorid ning kirjeldatakse lähenemist agiilse metoodika sobivuse hindamisele, mida kasutatakse käesoleva lõputöö analüüsi osas.

2.1 Agiilse arendusprotsessi ülevaade

Agiilseks võib nimetada sellist tarkvara arendusprotsessi, mis järgib Agiilse Manifesti printsiipe ning kasutab meetodeid ja praktikaid, mis võimaldavad kiiret inkrementaalset arendust ja sagedast tarkvara tarnimist, vähendades seejuures protsessi üldkulusid ja tagades tarkvara koodi kõrget kvaliteeti. Agiilses tarkvaraarenduses on põhirõhk tarkvara disainil ja realiseerimisel, kusjuures nõuete kogumine ja testimine on disaini ja realiseerimise lahutamatud osad, mistõttu tarkvara nõuded ja disain arenevad ja valmivad üheskoos [2].

Agiilset protsessi järgides, arendatakse tarkvara iteratiivselt (tsükliliselt). Joonisel 1 on toodud agiilse arendustsükli põhifaasid. Agiilse arendustsükli faasid ei pea tingimata toimuma kindlas järjekorras, need on paindlikud ning võivad toimuda ka üksteisega paralleelselt.



Joonis 1. Agiilse arendusprotsessi elutsükkel [3]

2.1.1 Agiilse arendusprotsessi põhimõtted

Agiilsete meetodite filosoofia tuleneb Agiilsest Manifestist, mis sai valmis 2001. aasta veebruaris ning mille autoriteks on 17 tarkvaraarenduse spetsialisti. Agiilne Manifest paneb paika agiilse tarkvaraarenduse väärtused ja eesmärgid, mis kajastuvad 12-s agiilse tarkvaraarenduse põhimõttes. Agiilses Manifestis formuleeritud põhimõtteid võib vaadelda kui eeskirju ja reegleid, mida iga „agiilseks“ nimetatav arendusprotsess peab järgima [4].

Agiilse Manifesti väärtused

- *Inimesed ja suhtlemine* on tähtsam kui protsessid ja tööriistad.
- *Töötav tarkvara* on tähtsam kui põhjalik dokumentatsioon.
- *Kliendiga koostöö* on tähtsam kui lepingute läbirääkimised.
- *Reageerimine muutustele* on tähtsam kui plaani järgimine [5].

Agiilse Manifesti väärtused avalduvad 12-s põhimõttes, mis toetavad agiilse tarkvaraarenduse väärtusi ning mida võib vaadelda juhtnööridena agiilse arendusprotsessi teostamiseks. Agiilse Manifesti põhimõtteid võib kokku võtta järgmiselt:

- *Kliendi kaasatus*

  Kliendid (tarkvara tellijad) peavad olema arendusprotsessi tihedalt kaasatud. Kliendi peamised ülesanded on tarkvara nõuete kirjeldamine ning valminud funktsionaalsuse ülevaatamine ja tagasiside andmine selle kohta, kuivõrd valminud tarkvaratoode vastab oodatule.

- *Inkrementaalne tarkavara tarnimine*

  Tarkvara arendatakse inkrementaalselt iteratsioonide kaupa. Iga iteratsiooni käigus valmib iteratsiooni alguses kliendiga kokku lepitud funktsionaalsuste hulk.
- *Inimesed on tähtsamad kui protsess*

  Arendusmeeskonna kogemused ja oskused on väga olulised, kuna nendest suurel määral sõltub projekti produktiivsus ja edukus. Arendusprojekti tuleb kaasata motiveeritud inimesed, kes on suutelised organiseerida tööprotsessi kõige paremal moel.
- *Muutused on oodatavad*

  Tarkvara arendamisel tuleb alati olla muutusteks valmis ning disainida tarkvara juba algusest peale selliselt, et muutusi tarkvara nõuetes oleks kerge sisse viia.
- *Tarkvara ja arendusprotsessi lihtsus*

  Arendatava tarkvara disain peab olema võimalikult lihtne ning peab võimaldama kiiret muutuste sisseviimist tarkvara nõuete muutumisel. Alati tuleb püüelda arendusprotsessi lihtsustamise poole ning tuleb elimineerida tööprotsessist liigset komplekssust ja mittevajalikke tegevusi.
- *Tehniline tipptase*

  Agiilne tarkavaraarendus rõhutab kõrge kvaliteediga tarkvara koodi tähtsust, kuna kvaliteetne ja hästi dokumenteeritud kood hõlbustab tarkvara tundma õppimist ning edasiarendust [2], [6].

2.1.2 Agiilse arendusprotsessi eelised ja puudused

Vastavalt „The 10th Annual State of Agile Report“ aruandele [7] on agiilsel arendusprotsessil järgmised peamised eelised:

- Pidevalt muutuvate prioriteetide parem hallatavus
- Kõrgem arendusmeeskonna efektiivsus
- Suurem arendusprotsessi läbipaistvus
- Parem tarkvara tarnete planeerimine
- Lühendatud toodangusse jõudmise aeg
- Kõrgem tarkvara kvaliteet
- Kõrgem arendusmeeskonna moraal ja motivatsioon
- Madalamad projekti riskid
- Parem kooskõlastus arendajate ja äripoole vahel

Agiilse protsessi paindlikkuse ja kohandatavusega käivad kaasas ka mõned võimalikud puudused:

- Ebapiisav planeerimine
- Suurte kogemuste ja teadmistega projekti meeskonna eeldus
- Kõrgelt pühendunud projekti meeskonna eeldus
- Ebapiisav dokumentatsioon
- Lõpliku toote suur erinevus esialgselt planeeritud tootest [3]

2.2 Agiilsed arendusmetoodikad

Kui agiilne tarkvaraarendus on pigem raamistik agiilsuse realiseerimiseks, siis agiilsed arendusmetoodikad on konkreetsed meetodid, mis kirjeldavad seda, kuidas võib agiilsust tarkvaraarenduses rakendada. Kõik agiilsed arendusmetoodikad põhinevad ühisel filosoofial ning neil on palju ühiseid tunnuseid ja tavasid, kuid rakendamise seisukohast vaadates, on igal metoodikal oma terminoloogia, taktika ning praktikate ja toimingute kogum [8]. Kõige levinumad agiilsed arendusmetoodikad on *Extreme Programming (XP)*, *Scrum*, *Kanban*, *Scrumban*, *Crystal*, *Adaptive Software Development (ASD)*, *Dynamic Systems Development Method (DSDM)*, *Lean Development* ja *Feature Driven Development (FDD)* [1], [3].

Käesoleva lõputöö analüüsi osas vaadeldav arendusmeeskond on lõputöö kirjutamise hetkeks katsetanud kahte agiilset arendusmetoodikat; nendeks on Scrum ja Scrumban. Lõputöös analüüsitakse nende metoodikate rakendamise edukust selles meeskonnas. Seetõttu vaadeldakse järgmistes alajaotistes neid metoodikaid detailsemalt.

2.2.1 Scrum

Scrum oli välja töötatud tarkvara arendusprotsessi juhtimiseks, seetõttu ei defineeri Scrum mingeid konkreetseid meetodeid või praktikaid, mida tuleks järgida/kasutusele võtta tarkvara realiseerimise faasis. Pigem keskendub Scrum sellele, kuidas peaksid arendusprojekti liikmed organiseerima oma tööd saavutamaks arendusprotsessi vajalikku paindlikkust ja kohandatavus pidevalt muutuvate nõuete keskkonnas [9].

Scrum metoodika ehitusplokkideks on Scrum meeskond ja sellega kaasnevad rollid (*roles*), tseremoniaalsused (*events*) ja artefaktid (*artefacts*).

Rollid

- *Tooteomanik (Product Owner)*

  Tooteomanik vastutab tarkvaratoote tegemata tööde loetelu (*product backlog*) loomise ja haldamise eest ning arendusmeeskonna tehtud töö kohta tagasiside andmise eest. Tooteomanik suhtleb pidevalt arendusmeeskonna ja tarkvara tellijaga tagamaks, et mõlemal osapoolel on ühine arusaamine *backlog*'is olevate tööde kohta [3].
- *Scrum Master*

  Scrum Master'i põhiülesandeks on vastutada Scrum praktikate korrektse kasutamise ja protsessi järgimise eest. Lisaks kuulub tema vastutusalasse protsessist tulenevate takistuste ja probleemide lahendamine, tagamaks, et projekti meeskond saab kokkulepitud töid tähtajaks valmis [10].
- *Arendusmeeskond (Development Team)*

  Arendusmeeskond koosneb kompetentsetest spetsialistidest, kes tegelevad tarkavara arenduse, testimise ja tarnega. Scrumis on arendusmeeskond iseorganiseeruv ja multifunktsionaalne, mis tähendab, et kõiki otsuseid võetakse vastu kogu meeskonnaga ning meeskonna liikmetel on kõik vajalikud oskused kokkulepitud tööde teostamiseks [10].

Tseremoniaalsused

Scrum metoodika tseremoniaalsused tagavad arendusprotsessi läbipaistvuse ning soodustavad protsessi pidevat täiustamist. Tseremoniaalsuste mittekasutamise tulemuseks on tavaliselt madalam protsessi läbipaistvus ning efektiivsus.

- *Sprint*

  Sprint on ajaliselt piiratud (1-4 nädalat) periood, mille jooksul peavad sprindi alguses kokkulepitud tööd valmis saama.
- *Sprindi planeerimine (Sprint Planning)*

  Sprindi planeerimise koosolekul otsustavad tooteomanik ja arendusmeeskond, milliseid *backlog*'i töid võtta tulevasel sprindil arendusse. Arendusse võetud tööd peavad sprindi lõpuks valmis saama.

- *Igapäevane koosolek (Daily Standup)*

  Lühike (mitte rohkem kui 15 minutit) igapäevane koosolek, mille jooksul annab iga meeskonna liige lühiülevaate eelmisel päeval tehtud tööst, jooksva päeva tööplaanidest ning probleemidest, mis takistavad tema töö tegemist.
- *Sprindi ülevaatus (Sprint Review)*

  Sprindi ülevaatus toimub iga sprindi lõpus. Ülevaatusel demonstreerib arendusmeeskond sprindi jooksul valminud tarkvara osa. Tooteomanik kontrollib, et valminud funktsionaalsus vastab vastuvõtmise kriteeriumitele (*acceptance criteria*). Tarkvara tellija annab tagasiside selle kohta, kas valminud funktsionaalsus vastab ärinõuetele.
- *Retrospektiiv (Retrospective)*

  Retrospektiiv toimub peale sprindi ülevaatust enne uue sprindi algust. Retrospektiivil arutab arendusmeeskond mis läks hästi, mis ei läinud hästi ning milliseid parandustegevusi saab järgmises sprindis ette võtta kõrvaldamaks tuvastatud probleeme/puudujääke. Retrospektiiv on väga oluline tseremoniaalsus, kuna võimaldab arendusprotsessi pidevat parendamist [8], [10].

Artefaktid

- *Tarkvaratoote tegemata tööde loetelu (Product Backlog)*

  Tarkvaratoote tegemata tööde loetelu on prioriteedi järgi järjestatud nimekiri kõikidest töödest, mida on vaja teostada toote valmimiseks. Tarkvaratoote *backlog* muutub pidevalt, kajastades kõiki tarkvara nõuete muutusi [10].
- *Sprindi tegemata tööde loetelu (Sprint Backlog)*

  Sprindi tegemata tööde loetelu on teatud hulk töid tarkvaratoote tegemata tööde loetelust, mida plaanitakse valmis saada sprindi vältel.
- *Inkrement (Increment)*

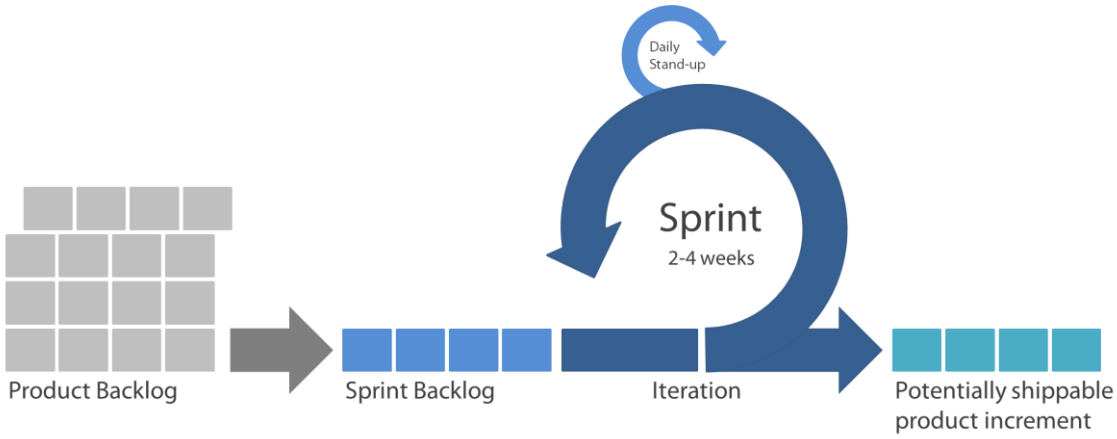
  Inkrement on kogum tarkvaratoote *backlog*'i töid, mis on valmis saanud peale viimast tarnet. Tooteomanik otsustab, millal tuleb toote inkrementi toodangusse lasta. Arendusmeeskond tagab, et selleks ajaks on kõik inkrementi kuuluvad tööd valmis toodangusse minekuks [8].

Scrumi elutsükkel

Scrumi elutsükkel koosneb järgmistest põhietappidest:

- Sprindi planeerimisel lepib arendusmeeskond tooteomanikuga kokku, milliseid töid tarkvaratoote tegemata tööde loetelust (*product backlog*) võetakse sprinti. Need tööd moodustavad sprindi tegemata tööde loetelu (*sprint backlog*).
- Sprint kestab tavaliselt 2-4 nädalat. Selle aja jooksul tegeleb arendusmeeskond sprindi *backlog*'is olevate tööde arendamise ja testimisega. Meeskonna liikmed annavad lühiülevaate tehtud tööst igapäevastel koosolekutel (*daily stand-up*).
- Sprindi lõpuks peavad kõik sprindi jooksul valminud funktsionaalsused olema potentsiaalselt tarnitavad.
- Sprint lõpeb sprindi ülevaatuse (*sprint review*) ja retrospektiiviga (*retrospective*) [3].

Joonis 2 kujutab Scrumi elutsüklit.



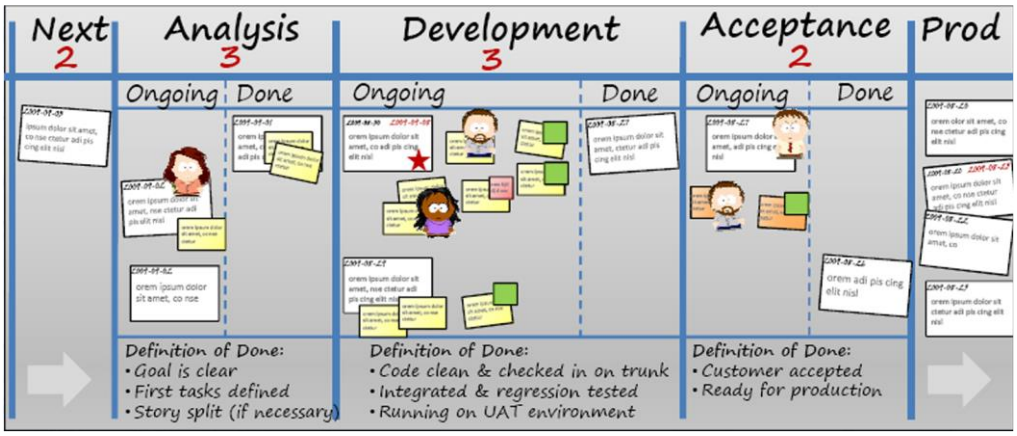
Joonis 2. Scrumi elutsükkel [11].

2.2.2 Scrumban

Scrumban arendusmetoodika on kombinatsioon Scrum ja Kanban metoodikatest, mis ühendab endas parimaid praktikaid mõlemast metoodikast [12].

Järgmised Scrumbani praktikad on üle võetud Kanbanist:

- *Töövoo visualiseerimine*
  Arendusmeeskond visualiseerib tahvlil erinevaid faase, mida tarkvaratoote *backlog*'i tööd läbivad arendustsükli käigus alates *backlog*'ist lõpetades „valmis" faasiga [13]. Joonis 3 esitab Scrumbani tahvli ühte võimalikku varianti.



Joonis 3. Scrumban tahvli näidis [14].

Veerud tahvlil kujutavad töövoo erinevaid faase ning „post-it“ kaardid kujutavad toote *backlog*'i töid. Selliselt on tööde liikumine läbi iga töövoo faasi kergesti jälgitav.

- *„Pull“ põhimõte*

  Pull“ põhimõtte järgi ei „lükata“ töid järgmisesse arendusfaasi, vaid „tõmmatakse“ neid alles siis, kui järgmisele faasile seatud tööde limiit lubab seda. Näiteks, kui arendusmeeskonna liige saab oma töö tehtud, „tõmbab“ ta arendusse järgmist tööd prioriteedi järgi järjestatud toote *backlog*'ist. Selline lähenemine eeldab, et toote *backlog* on hästi hallatud ning alati korrektselt kajastab toote nõuete ja prioriteetide muutusi [15]. Kui aga toote *backlog*-i pideva prioritiseerimisega esineb raskusi ning selles on liiga palju töid, võib võtta kasutusele ühe lisaveergu, mis järgneb *backlog*-i veerule (nt. „*Next*“ veerg Joonisel 2 kujutatud näidistahvlil) ning sisaldab teatud arvu prioritiseeritud töid, mida arendusmeeskond võib järgmisena töösse võtta. Parema ülevaatlikkuse mõttes võib mõned arendusfaasid (nt. analüüs, arendamine, testimine) jagada „töös“ ja „valmis“ veergudeks („*Ongoing*“ ja „*Done*“ veerud Joonisel 2 kujutatud näidistahvlil). Sel juhul, kui töö saab ühe arendusfaasi raames valmis, „lükatakse“ seda selle faasi „töös“ veerust „valmis“ veergu ning järgmise arendusfaasi „töös“ veergu „tõmmatakse“ siis töid eelmise faasi „valmis“ veerust [14].
- *Töös olevate tööde (WIP- Work In Progress) limiteerimine*

  Töös olevate tööde limiit seab maksimaalselt lubatud tööde arvu igale töövoo faasile ning kui tööde arv mõnes faasis ületab seatud limiiti, peavad kõik arendusmeeskonna liikmed sõltumata rollist töötama selle kallal, et selle faasi tööd saaksid ennem valmis, kui uusi töid võetakse *backlog*'ist arendusse. Selline

lähenemine lubab ennetada pudelikaelte tekkimist, tagab sujuvat töövoogu ning soodustab meeskonna liikmete tihedamat koostööd. WIP limiidi seadmine lubab vältida ka iga individuaalse meeskonna liikme ületunni- ja multitegumtööd (*multitasking*) [15], [16].

- *Meeskonna reeglite defineerimine*
  Reegleid defineerib arendusmeeskond ise. Selgesõnalised reeglid ehk protsessi sammude lihtsustatud kirjeldused aitavad meeskonna liikmetel protsessist paremini aru saada ning paremini protsessi järgida. Lisaks, paraneb protsessi jälgitavus ning tekivad võimalused protsessi täiustamiseks [13].

Scrumilt on Scrumban mõningaid tseremoniaalsusi üle võtnud:

- *Planeerimise koosolek (Planning Meeting)*
  Scrumbanis planeeritakse järgmiseid arendusse minevaid töid siis, kui selleks tekib vajadus ehk siis kui *backlog* või muu arenduseks valmis tööde loetelu hakkab tühjaks saama. Arendusmeeskond ise otsustab, millal järgmist planeerimise koosolekut teha. Ideaalne tööde planeerimise protsess peab alati tagama, et arendusmeeskonnal on täpselt nii palju tööd kui nad oskavad normaalse töökoormusega ära teha, mitte rohkem ega vähem [16], [17].
- *Ülevaatuse koosolek (Review Meeting)*
  Ülevaatuse koosolekul demonstreerib arendusmeeskond äripoolele valminud tarkvara funktsionaalsust ning saab selle kohta äripoolelt tagasisidet. Kuigi Scrumbanis ei ole sprinte, eelistatakse siiski teha ülevaatuse koosolekuid regulaarselt.
- *Retrospektiiv* (*Retrospective Meeting)*
  Retrospektiive tehakse tavaliselt peale ülevaatuse koosolekut ning püütakse tuvastada protsessi parenduskohti ning leida viise protsessi efektiivsemaks muutmiseks.
- *Igapäevane koosolek* (*Daily Stand-up Meeting)*
  Igapäevase koosoleku põhiidee on saada arendusmeeskonna liikmetelt informatsiooni töös olevate tööde ja tööd takistavate asjaolude kohta, et paremini koordineerida töövoogu ning kõrvaldada töövoos tekkivaid takistusi [16].

2.2.3 Scrum vs Scrumban

Tabel 1 esitab Scrum ja Scrumban metoodikate võrdlust.

Tabel 1. Scrum ja Scrumban metoodikate võrdlus [14], [18], [19], [12].

	Scrum	**Scrumban**
Iteratsioonid	1-4-nädalased sprindid	Pidev töövoog lühemate planeerimistsüklitega ning pikemate tarnetsüklitega
Artefaktid	Tarkvaratoote *backlog*, sprindi *backlog*, toote inkrement	Tarkvaratoote *backlog*, toote inkrement
Tahvel	Lihtne tahvel, mida tühistatakse/uuendatakse iga sprindi alguses	Püsiv töövoogu kajastav tahvel
Tööde jagamine	„Push“ ja „pull“ põhimõte, arendusmeeskond võtab kohustuseks teostada kokkulepitud hulk töid sprindi jooksul	„Pull“ põhimõte, arendajad tõmbavad töid arendusse, kui saavad oma eelmise tööülesandega valmis
Töös olevate tööde (WIP) limiit	Piiratud sprinti võetud tööde arvuga	Piiratud töövoo igale faasile seatud maksimaalse tööde arvuga
Planeerimine	Sprindi planeerimine	Planeerimine kui *backlog* hakkab tühjaks saama
Tseremoniaalsused	Igapäevane koosolek, sprindi planeerimine, sprindi ülevaatus, sprindi retrospektiiv	Igapäevane koosolek; planeerimine, ülevaatus ja retrospektiiv vajadusel
Rollid	Tooteomanik, *Scrum Master*, aredusmeeskond	Arendusmeeskond ja teised vajaminevad rollid
Tarkvaratoote *backlog*	Prioriseeritud tegemata tööde loetelu, mis katab kogu tarkvaratoote funktsionaalsuse	Prioriseeritud loetelu töödest, mis on valmis arendusse võtmiseks
Töö suurus	Töö peab olema ühe sprindi jooksul äratehtav	Ei ole piiratud
Uute tööde lisamine iteratsiooni käigus	Ei ole lubatud	Lubatud
Tootluse mõõdikud	Läbipõlemise graafik (*burndown chart*), tööde valmimise kiirus (*velocity*)	Keskmine tsükli aeg (*average cycle time*), keskmine toodangusse jõudmise aeg (*average lead time*)

	Scrum	**Scrumban**
Millistel juhtudel sobib kasutamiseks?	Nõuded on hästi defineeritud ning saab tagada sprinti võetud nõuete mittemuutmist, on olemas pühendunud tooteomanik ja *Scrum Master*, arendusmeeskond koosneb kogenud spetsialistidest ning saab hakkama iseorganiseerimisega.	Ilma selge visioonita toote arendamine, pidevalt muutuvad nõuded, ootamatud muutused tööde prioriteetides, tarkvara hooldus on arendusprotsessi lahutamatu osa, Scrumi jäikus piirab arendusprotsessi kohandamist/täiustamist vastavalt toimuvatele muutustele.

2.3 Agiilse arendusmetoodika sobivus

Agiilseid arendusmetoodikaid vaadeldakse tihti kui vahendeid arendusprotsessi üldkulude vähendamiseks ning olemasoleva protsessi efektiivsemaks muutmiseks. On üldtunnustatud, et ei eksisteeri sellist agiilset arendusmetoodikat, mis sobiks hästi kõikidele projektidele. Samas eksisteerib palju erinevaid agiilseid praktikaid, mida tasub kasutusele võtta mõnes konkreetses situatsioonis [4]. Sageli juhtub aga, et arendusprotsessi kujundamisel jäetakse olulised agiilse metoodika põhimõtted ja praktikad mingil põhjusel välja. Pealegi, mõnedes ettevõtetes puuduvad agiilse arendusmetoodika kasutuselevõttu toetav ärikultuur ja -keskkond. Tulemusena agiilse metoodika potentsiaalne kasu jääb kas osaliselt või täielikult realiseerimata või halvemal juhul muutub situatsioon halvemaks [20].

Seetõttu on ülitähtis teha regulaarseid arendusprotsessi ülevaatusi hindamaks valitud arendusmetoodika ja praktikate sobivust ja efektiivsust ning leidmaks protsessi võimalikud parenduskohad ja edasiarendamise võimalused.

2.3.1 Agiilsete arendusmetoodikate rakendamise edu võtmefaktorid

Agiilse arendusmetoodika sobivuse hindamiseks on esialgu vaja tuvastada need olulised faktorid, mis mõjutavad selle rakendamise edukust. Teema-alase kirjanduse läbitöötamise tulemusena on tuvastatavad allpooltoodud agiilsete arendusmetoodikate rakendamise edu võtmefaktorid. Võtmefaktorite tuvastamisel kontrollis töö autor, et need toetavad ka Agiilses Manifestis sõnastatud väärtuste täitmist.

Organisatsioonilised

- *Ettevõttesisene kultuur ja väärtused* [7], [21], [22], [23], [24]
  Peaaegu kõik agiilsete protsesside eksperdid omistavad suurt tähtsust ettevõttesisesele kultuurile ja väärtustele agiilsete protsesside juurutamisel. Kuna agiilsed meetodid eeldavad pidevat koostööd kliendiga (tarkvara tellijaga), peab ettevõtte kultuur võimaldama kiiret muutustele reageerimist vastavalt kliendilt saadud tagasisidele. Samuti ettevõtte kultuur ja väärtused peavad soodustama töötajate vahelist koostööd ja pidevat teineteiselt õppimist.
- *Tippjuhtide toetus* [7], [22], [25], [26], [27], [28]
  Tippjuhtide toetuse puudumine võib osutuda tõsiseks takistuseks agiilsele protsessile üleminekul. Paljudel juhtudel just tippjuhid peavad võtma endale ettevõtte agiilse transformatsiooni eestvedajate rolli, kuna neil on võimu teha vajalikke organisatsioonilisi ja kultuurilisi muutusi.

Kliendisuhted

- *Koostöö kliendiga* [6], [24], [25], [26]
  Pidev ja tihe koostöö kliendiga on äärmiselt vajalik tagamaks, et nii klient kui ka arendusmeeskond saavad tarkvara nõuetest ühtemoodi aru ning et arendatav tarkavara vastab kliendi ootustele.
- *Kliendi pühendumus* [2], [4], [23], [24], [27], [29]
  Kuna agiilsed meetodid eeldavad tihedat koostööd kliendiga, on oluline, et klient tunneks tõelist huvi arendatava tarkvara vastu ning et temal oleks aega arendusprotsessis aktiivselt osaleda ning arendusmeeskonnale tagasisidet anda.
- *Kliendi rahulolu* [24], [26], [27], [29]
  Kliendi rahulolu on agiilse tarkvaraarenduse üks tähtsamaid prioriteete, mida saavutatakse läbi tiheda koostöö ning varajase ja kiire tarkavaratoote tarnimise. Kliendid hindavad agiilseid meetodeid, kuna need annavad neile võimaluse võtta arendusprotsessist osa ning vajadusel mõjutada seda.

Töötajad

- *Arendusmeeskonna pädevus* [4], [6], [7], [24], [26], [27], [28]
  Arendusmeeskonna pädevus on otseselt seotud meeskonna võimekusega tähtajaliselt tarnida nõuetelevastavat ja kvaliteetset tarkvara pidevalt muutuvate

nõuete ja prioriteetide tingimustes. Lisaks peab arendusmeeskond olema võimeline kujundama ning vajadusel kohandama kasutuseloleva arendusprotsessi.

- *Meeskonna liikmete vaheline koostöö* [6], [7], [24], [26], [27], [28], [29]
  Agiilsed arendusprotsessid eeldavad pidevat suhtlemist ning tihedat koostööd meeskonna liikmete vahel. Pidev koostöö ja suhtlemine soodustavad teineteiselt õppimist ning üksteise aitamist tööülesannete lahendamisel. Selle tulemusena saavutatakse sujuvamat ja efektiivsemat arendusprotsessi.
- *Meeskonna pühendumus ja rahulolu* [22], [25], [26], [28], [29], [30]
  Väga oluline on, et arendusmeeskonna liikmed tunneksid, et nende tööd väärtustatakse ning et arendatav tarkvara vastab kliendi ootustele. See tõstab meeskonna liikmete rahulolu ja pühendumust, mis omakorda loob positiivse töökeskkonda, soodustab tõrgeteta arendusprotsessi ning tagab tarkvara kvaliteeti.

Protsess

- *Agiilse meetodi ja selle praktikate järgimine* [7], [22], [23], [28], [29], [31]
  Agiilsete arendusmeetodite loojad ja neid rakendavad spetsialistid on ühel meelel: selleks, et valitud agiilne meetod tooks oodatavaid tulemusi, on vaja ustavalt järgida selle põhimõtteid ja praktikaid. Agiilse arendusmetoodika järgimine tagab, et arendusprotsess ei välju kontrolli alt, omab selget struktuuri, on läbipaistev ja vastab agiilse tarkvaraarenduse põhimõtetele.
- *Pidev protsessi täiustamine* [7], [26], [25], [28], [29], [30]
  Agiilse protsessi juurutamise järel on väga oluline pidevalt jälgida protsessi ning vajadusel teha selles kohandusi, et see paremini vastaks projekti/ettevõtte nõuetele. Protsessi ülevaatus ja efektiivsuse hindamine peavad toimuma regulaarselt tuvastamaks protsessi kitsaskohti ning võimalusi protsessi täiustamiseks.
- *Projektijuhtimine* [22], [23], [24], [26], [28], [31]
  Agiilne projektijuhtimine peab toetama inkrementaalset, iteratiivset tarkvara arendust läbi kohanduva planeerimise, pideva nõuete haldamise, kiire inkrementaalse tarnimise ja kliendiga koostöö. Samuti peab agiilne projektijuhtimine soodustama meekonna liikmete vahelist koostööd ning pidevat suhtlemist. Protsessi seisukohast vaadates, agiilse projektijuhtimise olulisemad

valdkonnad on tööde planeerimine, nõuete haldamine (s.h. prioritiseerimine) ja tarkvara tarnete korraldamine.

- *Koolitused ja mitteformaalne õppimine* [24], [26], [28]
  Agiilsele protsessile üleminekul on väga oluline korraldada mitmeid koolitusi protsessi ja arendusmetoodika kohta kõikidele arendusprotsessi osalejatele (s.h. klient ja teised huvigrupid). Koolitused kindlustavad, et kõikidel on ühine arusaamine protsessist, mistõttu saavad kõik protsessi paremini järgida. Tähtis koht on agiilses arendusprotsessis ka mitteformaalsel teineteiselt õppimisel, kuna see on üks lihtsamaid viise meeskonna pädevuse suurendamiseks.

Tehnilised

- *Tehniline tipptase ja hea disain* [22], [23], [25], [31]
  Agiilsed praktikad rõhutavad puhta ja hästi struktureeritud programmikoodi tähtsust, kuna siis on koodist kergemini aru saada ning vajadusel muudatusi sisse viia. Puhta ja kvaliteetse koodi tagavad selgelt defineeritud koodi kirjutamise standardid, lihtne tarkvara disain ning põhjalik ja regulaarne koodi refaktoorimine.
- *Tehnoloogiad ja tööriistad* [25], [27], [32]
  Tehnoloogiad ja tööriistad peavad toetama agiilse arendusprotsessi põhimõtteid ning tagama protsessi tõhusust. Samuti peavad nad lihtsustama ning muutma efektiivsemaks ja kvaliteetsemaks meeskonna liikmete tööd. Oluline on ka, et meeskonna liikmetel oleksid tugevad teadmised kasutatavate tehnoloogiate ja tööriistade kohta.

2.3.2 Arendusmetoodika sobivuse hindamine

Selleks, et agiilne arendusprotsess jääks ka edaspidi agiilseks, on väga oluline regulaarselt teha kasutuseloleva arendusprotsessi hindamist selgitamaks välja, kui hästi see toimib [25]. Lähtudes alapeatükis 2.1.1 toodud agiilse arendusprotsessi põhimõtetest võib öelda, et arendusprotsess on agiilne ja toimib hästi siis, kui:

- arendusmeeskond suudab arendada ja tarnida kõrgekvaliteetset töötavat tarkvara inkrementaalselt ning tihedas koostöös kliendiga;

- klient võtab arendusprotsessist aktiivset osa ning pidevalt annab valminud tarkvara funktsionaalsustele tagasisidet tagamaks, et arendatav tarkvara vastab oodatule;
- soodustatakse meeskonna liikmete vahelist suhtlemist ja koostööd;
- suudetakse kiiresti reageerida ärinõuete ja prioriteetide muutumisele;
- kliendi ja meeskonna rahulolu püsib alati kõrgel tasemel.

Kui arendusprotsess ei toimi enam hästi, tuleb viia läbi selle põhjalikku ülevaatust selgitamaks välja võimalikke parenduskohti. Agiilse arendusprotsessi ülevaatuse läbiviimisel tuleb eelkõige alustada kasutuseloleva agiilse arendusmetoodika rakendamise analüüsist, kuna just agiilne arendusmetoodika määrab, kuidas tuleb arendusprotsessi teostada, et saavutada ja säilitada selle agiilsust ja tõhusust.

Kui agiilse arendusmetoodika rakendamine mõnes konkreetses projektis/ettevõttes võimaldab hästi toimiva agiilse arendusprotsessi teostamist, võib väita, et kasutuselolev arendusmetoodika on antud situatsioonis efektiivne, vastasel juhul tuleb kaalutleda selle metoodika väljavahetamist. Kasutuseloleva agiilse arendusmetoodika väljavahetamist kaalutledes, on esmalt vaja teha metoodika rakendamise põhjalikku analüüsi välja selgitamaks selle ebaefektiivsuse põhjused. Põhjuseks võib olla nii metoodika ebasobivus antud konkreetses situatsioonis kui ka metoodika väär kasutamine või mittejärgimine. Lähtudes tuvastatud ebaefektiivsuse põhjustest, tuleb kas otsida uut sobivamat arendusmetoodikat, kohandada olemasolevat metoodikat või teha muudatusi selles, kuidas käesolevat metoodikat rakendatakse.

Nii tarkavaraarenduse spetsialistid [22], [25], [30] kui ka akadeemilised uurijad [6], [21], [24], [23], [33], [34] on pakkunud erinevaid lähenemisi ja raamistikke agiilsete arendusmetoodikate sobivuse ja efektiivsuse hindamiseks. Üks võimalik lähenemine on analüüsida kasutuselolevat agiilset metoodikat läbi agiilsete arendusmetoodikate edu võtmefaktorite, mis olid defineeritud eelmises alajaotises. Agiilsete arendusmetoodikate edu võtmefaktorid juhivad tähelepanu just nendele võtmealadele, mis kõige rohkem mõjutavad agiilsete tarkvara arendusprojektide edukust. Seetõttu lubavad nende võtmealade analüüsi tulemused teha järeldusi arendusmetoodika efektiivsuse kohta, mis omakorda näitab, kas kasutuselolev arendusmetoodika on käesolevale projektile/ettevõttele sobiv või vajab täiustamist/väljavahetamist. Käesoleva lõputöö analüüsi osas kasutatakse just sellist agiilsete arendusmeetodikate hindamise lähenemist.

3 Meeskonna ja arendatavate süsteemide tutvustus

Järgnevalt tutvustatakse käesolevas töös vaatluse all olevat meeskonda ja meeskonna poolt arendatavaid süsteeme.

3.1 Meeskonna tutvustus

Meeskonda kuulub kaheksa inimest, kes tegelevad kahe ettevõttesisese süsteemi arendamise ning hooldusega (süsteeme tutvustatakse täpsemalt alapeatükis 3.2). Meeskonna rollide jaotus on järgmine: tooteomanik, arendusmeekonna juht, süsteemianalüütik, süsteemiarhitekt/tarkvaraarendaja, 3 tarkvaraarendajat, tarkvara kvaliteedi juht. Kogu arendusmeeskond peale tooteomaniku asub Tallinnas.

Järgnevalt on kirjeldatud vaadeldava meeskonna liikmete põhiülesanded:

- *Tooteomanik*

  Tooteomanik on ettevõtte äripoole esindaja. Ettevõtte äripool esineb arendustööde tellija (kliendi) rollis. Tooteomaniku põhiülesanneteks on ärinõuete kogumine ja ärianalüüs, toote *backlog*'i koostamine ja prioritiseerimine, valmis tööde vastuvõtutestimine, tarnete planeerimine koostöös arendusmeeskonna juhiga, tarnete kinnitamine ning kasutajate dokumentatsiooni loomine. Lisaks vastutab tooteomanik süsteemide visiooni kujundamise ja üldise arengukava loomise eest ning süsteemide superkasutajate koolitamise ja toetamise eest. Tooteomaniku üheks tähtsaks ülesandeks on ka süsteemide sõltuvuste haldamine tihedas koostöös integreeritud naabersüsteemide tooteomanikega.
- *Arendusmeeskonna juht*

  Arendusmeeskonna juhi põhiülesandeks on arendusmeeskonna töö organiseerimine, s.h. igasuguste organisatsiooniliste küsimuste lahendamine, igapäevast tööd segavate takistuste elimineerimine, konfliktide lahendamine, juhtkonnale raporteerimine jms. Lisaks vastutab meeskonna juht selle eest, et kõigil meeskonna liikmetel on tööülesanded alati olemas ja et tööülesanded on

arusaadavad. Vajadusel tegeleb meeskonna juht ka süsteemide testimise ja dokumenteerimisega.

- *Süsteemianalüütik*

  Süsteemianalüütiku põhiülesanneteks on prioriteetsemate toote *backlog*'is olevate tööde süsteemianalüüs, nõutud funktsionaalsuste ärikirjelduste valideerimine ja täiendamine tihedas koostöös tooteomanikuga ning detailiseeritud nõuetega tööülesannete koostamine arendajatele. Lisaks pakub süsteemianalüütik arendajatele ja kvaliteedi juhile igapäevast tuge, vastates nende küsimustele tööülesannete kohta ja täpsustades ärinõudeid tooteomanikuga. Süsteemianalüütiku tööülesannete hulka kuulub ka valminud tööde manuaalne testimine, süsteemide superkasutajatele tugiteenuste osutamine, valminud funktsionaalsuse dokumenteerimine ning tarnete planeerimine koostöös tooteomaniku ja meeskonna juhiga.

- *Süsteemiarhitekt/tarkvaraarendaja*

  Süsteemiarhitekt vastutab süsteemide üldise arhitektuuri ja disaini eest. Tema põhiülesanneteks on süsteemide arhitektuuri analüüs ja arendamine tagamaks süsteemide vajalikku jõudlust ja stabiilsust, tehniliste lahenduste väljapakkumine nõutud funktsionaalsuse realiseerimiseks koostöös süsteemianalüütiku ja tooteomanikuga, arendajate nõustamine tööülesannete täitmisel, koodi ülevaatuste tegemine ja süsteemide arendamine (lähtekoodi kirjutamine).

- *Tarkvaraarendaja*

  Tarkvaraarendaja ülesandeks on süsteemide edasiarendus vastavalt tööülesannetes kirjeldatud nõuetele. Koostöös süsteemiarhitektiga mõtleb tarkvaraarendaja tööülesandes kirjeldatud funktsionaalsuse disaini ja lahenduse läbi ning teeb vajalikku süsteemi edasiarendust. Tarkvaraarendaja tegeleb nii kasutajaliidese kui ka serveripoolsete (s.h. andmebaasi) arenduste realiseerimisega.

- *Tarkvara kvaliteedi juht*

  Tarkvara kvaliteedi juht vastutab tarnitava tarkvara kvaliteedi eest. Kvaliteedi juhi põhiülesandeks on süsteemide automaattestide (sh integratsioonitestide) kirjutamine, haldamine ja käivitamine ning testiraportite jälgimine ja vigade raporteerimine. Automaattestide kirjutamisel teeb kvaliteedi juht tihedat koostööd süsteemianalüütiku ja arendajatega tagamaks automaattestide usaldusväärsust ja kvaliteeti. Kvaliteedi juht tegeleb ka süsteemide uute versioonide paigaldamisega

erinevatesse arenduskeskkondadesse. Kvaliteedi juht ei tegele üldjuhul manuaalse testimisega, see on süsteemianalüütiku ja tooteomaniku ülesanne.

3.2 Arendatavate süsteemide tutvustus

Vaatluse all olev meeskond tegeleb kahe süsteemi arendamise ja hooldusega. Esimene süsteem on operatiivsüsteem, mis vastutab erinevat liiki tellimuste vastuvõtmise, valideerimise ja töötlemise eest ning integreeritud ettevõttesisestesse süsteemidesse korrektses formaadis ja nõutud andmetega edastamise eest. Tellimused tulevad operatiivsüsteemi kas integreeritud klientide süsteemidest (nt. SAP) või ettevõtte kliendiportaalist (klient sisestab tellimused käsitsi). Operatiivsüsteem on ettevõttesisene süsteem ning selle töö on täiesti automatiseeritud, st kui kõik on seadistatud korrektselt süsteemis, toimub tellimuste töötlemine ja edasisaatmine automaatselt sekundite jooksul. Käsitsi töö on vajalik vaid erandjuhtude korral, näiteks kui tellimuste töötlemisel esineb vigu või kui on vaja olemasolevat seadistust muuta vastavalt kliendi uutele nõuetele (sellega tegelevad süsteemi superkasutajad). Operatiivsüsteem on ärikriitilise tähtsusega, mistõttu peab arendusmeeskond tagama süsteemi usaldusväärsust (korrektne andmete töötlus), teatud jõudlust (kiire andmete töötlus) ja töökindlust (süsteem peab alati töökorras olema).

Teine süsteem on kliendi asukohtade, toodete ja tellimuste lisaandmete (*master data*) ning äriprotsesside seadistuste (*business processes configuration*) haldussüsteem. See on suhteliselt uus süsteem, mille arendus algas 1,5 aastat tagasi ning mille eesmärkideks on korrastada ja ühtlustada *master data*, mida kasutatakse ettevõtte teatud tegevusvaldkonda teenindavates integreeritud süsteemides ning luua keskne süsteem, kus saaks hallata nende süsteemide äriprotsesside konfiguratsioone ja üldseadistusi (nt. hallata süsteemile ligipääsu õigusi ja andmetele ligipääsu piiranguid, seadistada süsteemi põhiprotsesside äriloogikat erinevate klientide jaoks jms.). Hetkel on haldussüsteemiga täielikult integreeritud vaid üks süsteem ning osaliselt selle funktsionaalsust kasutab eelpool kirjeldatud operatiivsüsteem. Tulevikus plaanitakse kõikide ettevõtte teatud tegevusvaldkonda kuuluvate süsteemide integreerimist keskse haldussüsteemiga. Hetkel on haldussüsteem ärikriitilise tähtsusega sellega täielikult integreeritud süsteemi jaoks, kuna integreeritud süsteem vajab oma töös haldussüsteemis hallatavaid andmeid. Seetõttu peab arendusmeeskond tagama süsteemi kõrget töökindlust.

4 Arendusmetoodikate rakendamise analüüs

Käesolevas peatükis analüüsitakse Scrum ja Scrumban arendusmetoodikate rakendamist eelnevas peatükis kirjeldatud meeskonna näitel eesmärgiga hinnata metoodikate sobivust vaadeldavate süsteemide arendamiseks.

Arendusmetoodika sobivuse hindamisel analüüsitakse eelkõige kuivõrd meeskonna poolt teostatav arendusprotsess vastab metoodikas kirjeldatud põhimõtetele. Läbitöötaud materjalidele [25], [28], [29], [31], [35] ja töö autori kogemustele tuginedes, ei saa oodata, et metoodika toob kasu ja muudab arendusprotsessi efektiivsemaks, kui seda ei järgita korralikult.

Seejärel analüüsitakse arendusmetoodika sobivust läbi alapeatükis 2.3.1 defineeritud agiilsete metoodikate rakendamise edu võtmefaktorite. Edufaktoreid analüüsides, kontrollitakse ka, et oleksid täidetud Agiilse Manifesti põhimõtted.

4.1 Scrum arendusmetoodika rakendamine

Scrum arendusmetoodika rakendamise analüüsiperioodi alguseks võetakse august 2015, kuna sel ajal liitus käesoleva töö autor vaatluse all oleva meeskonnaga süsteemianalüütiku rollis. Scrum metoodika rakendati märtsini 2016, peale seda mindi Scrumban metoodikale üle. Siin tuleb mainida veel seda, et jaanuarini 2016 tegeles meeskond aktiivselt vaid ühe süsteemi (operatiivsüsteemi) arendusega.

4.1.1 Arendusprotsessi vastavus Scrum arendusmetoodikale

Järgnevalt analüüsitakse kuivõrd korralikult rakendas analüüsitav meeskond Scrum metoodikat. Analüüsi aluseks on Scrum metoodika kirjeldus alapeatükis 2.2.1.

Rollid

- *Tooteomanik*

  Tooteomanik oli olemas, kuid ta ei olnud projektile pühendunud. Teiste tööülesannetega hõivatuse tõttu ei olnud tal tihti võimalust osaleda Scrumi

tseremoniaalsustes ning operatiivselt vastata täpsustavatele küsimustele nõutud süsteemi funktsionaalsuste kohta. Selle taustal tooteomaniku poolt formuleeritud ärinõuete kirjeldused olid väga ähmased ja ebapiisavad, kuna tema arvamusel arendusmeeskond pidi ise vajalikke detaile lisama. Vahepeal ei olnud tooteomanikul aega ärinõudeid üldse kirja panna ning sellega tegeles kas meeskonna juht (kes pidi täitma ka analüütiku ülesandeid) või süsteemianalüütik. Ajanappuse tõttu olid tööülesannete kirjeldused väga pealiskaudsed ning suur äriloogika osa jäi tihti dokumenteerimata. Tooteomanik ei tegelenud *backlog*'i pideva prioritiseerimisega ning kuna mitu inimest lisasid töid *backlog*'i, ei omanud keegi selget pilti prioriteetsematest töödest ning toote *backlog* kasvas väga suureks. Vastuvõtutestimisega ei tegelenud tooteomanik peaaegu üldse usaldades, et arendusmeeskond tagab vajalikku kvaliteeti. Osaliselt on vastuvõtutestimine ka süsteemi superkasutajate vastutus, kuid kahjuks ei olnud ka neil kunagi aega põhjalikuks testimiseks. Tooteomaniku koostöö integreeritud süsteemide tooteomanikega süsteemi sõltuvuste juhtimisel oli puudulik. Mitu korda oli situatsioon, kus integreeritud süsteemides ei olnud vajalikud arendused tehtud, mis takistas vaatluse all oleva süsteemi uue funktsionaalsuse toodangusse laskmist. Kui algas teise süsteemi arendus, oli tooteomaniku pühendumus selle süsteemi arendamisele umbes sama, mistõttu pidi arendusmeeskond veel rohkem vaeva nägema, et tooteomaniku tegemata töö ei takistaks arenduste kulgu.

- *Scrum Master*

  Eraldi *Scrum Master*'it meeskonnal ei olnud. Osaliselt täitis tema ülesandeid meeskonna juht. Näiteks tegeles ta organisatsiooniliste küsimustega ning protsessist tulenevate takistuste ja probleemide lahendamisega. Keegi ei vastutanud Scrum praktikate korrektse kasutamise ja protsessi järgimise eest, mistõttu kaldus arendusprotsess Scrum metoodikast kõrvale aina rohkem ja rohkem.

- *Arendusmeeskond*

  Scrum arendusmeeskonna koosseisu kuulusid meeskonna juht/süsteemianalüütik, süsteemianalüütik, süsteemiarhitekt/tarkvaraarendaja, 2 tarkvaraarendajat ja tarkvara kvaliteedi juht. Arendusmeeskond vastas Scrum meeskonna põhimõtetele: selle koosseisu kuulusid kompetentsed spetsialistid ning meeskond oli iseorganiseeruv ja multifunktsionaalne. Arendusmeeskond paiknes ühes ruumis, mis tagas pideva suhtlemise ja soodustas tihedat koostööd liikmete vahel.

Kahjuks ei olnud koostöö eemaloleva tooteomanikuga väga tihe tooteomaniku pideva kättesaamatuse tõttu. Seetõttu pidi arendusmeeskond pidevalt töötama pooleliolevate tööülesannetega, mis raskendas lahenduste väljatöötamist ning vahepeal põhjustas tehtud töö ümbertegemist. Samuti muutis tooteomanik tööde prioriteete tihti ning lisas uusi töid sprinti, mille tõttu pidid arendusmeeskonna liikmed jätma mõned tööülesanded pooleli ja tegelema uue ülesandega. Kuna toote *backlog*'i tööd ei olnud prioritiseeritud, ei olnud arendusmeeskonnal selget ettekujutust toote (süsteemi) visiooni kohta ning lähimatest arenguplaanidest. Meeskonna juht ja süsteemianalüütik pidid pidevalt tooteomaniku tööülesandeid enda peale võtma, tegeledes ärinõuete väljaselgitamise ja kirjeldamise, *backlog*'i tööde prioritiseerimise ning superkasutajate toetamisega oma põhiülesannete kõrvalt. Arendusmeeskonna liikmed olid ülekoormatud ning teise süsteemi arenduse algusega halvenes situatsioon rohkem. Kõik see põhjustas arendusmeeskonna rahulolematust, eriti tooteomaniku osas.

Tseremoniaalsused

- *Sprint*

  Alguses olid sprindid kahenädalased, hiljem hakkas sprintide pikkus venima ning Scrumi perioodi viimastel kuudel ei olnud sprinte üldse. Sprintide pikkuse venimine oli tingitud sellest, et tooteomanik pidevalt lisas uusi töid sprinti, mistõttu ei olnud tihti võimalik kõik tööd sprindi lõpuks valmis saada ning sprinti pikendati. Sprintide ärajätmise põhjuseks oli see, et tooteomanikul tihti ei olnud aega teha sprindi planeerimist ja ülevaatust.
- *Sprindi planeerimine*

  Alguses tehti sprindi planeerimist, milles osales kogu arendusmeeskond ja tooteomanik. Tooteomanik osales tavaliselt telefoni või *Lync*'i vahendusel, harva käis ta Tallinna kontoris ka isiklikult. Sprindi planeerimised olid ebaefektiivsed, kuna tooteomanik oli tavaliselt ettevalmistamata: *backlog*'i töö kirjeldused olid puudulikud, tööd ei olnud prioriteetide järgi järjestatud, vahepeal ei olnud tooteomanik ise kindel, mis tööd on prioriteetsemad. Seetõttu sprindi planeerimise asemel tegeles meeskond tihti pigem *backlog*'i tööde korrastamisega. Samuti ärinõuete ähmasuse tõttu oli arendajatel väga raske sprindi töid hinnata, vahepeal keeldus süsteemiarhitekt tööle hinnangut andmast ebaselgete nõuete tõttu. Hiljem hakkas tooteomanik sprindi planeerimistel puuduma muude tööülesannetega

hõivatuse tõttu ning edastas informatsiooni tööde prioriteetide kohta meeskonnajuhile kas e-maili või telefoni teel. Esialgu jätkas arendusmeeskond sprindi planeerimise koosolekutega, kuid hiljem jäeti need ära ajanappuse tõttu ning meeskonna juht lisas ise tööd sprinti tooteomaniku informatsiooni põhjal.

- *Igapäevane koosolek*

  Igapäevased koosolekud toimusid regulaarselt iga tööpäeva alguses. Koosolekutel rääkis iga arendusmeeskonna liige lühidalt, millega ta tegeles eile, millega plaanib tegelda täna ning tema tööd takistavatest probleemidest.

- *Sprindi ülevaatus*

  Alguses toimusid sprindi ülevaatused iga sprindi lõpus. Sprindi ülevaatusest võttis osa kogu arendusmeeskond ja tooteomanik videokonverentsi vahedusel. Meeskond demonstreeris tooteomanikule tehtud töid ning tooteomanik võttis neid vastu ja andis omapoolse tagasiside. Hiljem aga jäeti sprindi ülevaatused ära, kuna tooteomanikul ei olnud aega ülevaatustel osaleda ning ta vaatas tööd ise üle, esitades küsimusi valminud funktsionaalsuste kohta analüütikutele kui midagi oli talle ebaselge. Üldjuhul oli selline tööde ülevaatus kiire ja pealiskaudne, kuna tooteomanik usaldas, et arendusmeeskond suudab tarnida nõuetelevastava funktsionaalsuse ning tagada selle kvaliteedi. Selline lähenemine on potentsiaalselt ohtlik, kuna arendusmeeskond alati võib ärinõuetest valesti aru saada, mistõttu valminud funktsionaalsus võib töötada mitte päris nii, nagu tooteomanik eeldab.

- *Retrospektiiv*

  Ajanappuse tõttu retrospektiive ei tehtud. Scrumi rakendamise perioodi lõpus tehti üks retrospektiiv, kus jõuti järeldusele, et olemasolev arendusprotsess ei ole enam üldse Scrumi moodi. Samas otsustas meeskond, et Scrum ei sobi käesolevas situatsioonis tooteomaniku madala pühendumuse, ebaselgete ja pidevalt ka sprindi sees muutuvate prioriteetide ning tseremoniaalsuste peale mineva ajakulu tõttu, eriti arvestades seda, et meeskonnal oli nüüd vaja arendada ja hooldada kahte süsteemi. Hiljem koostöös tooteomanikuga otsustati minna Scrumbanile üle.

Artefaktid

- *Tarkvaratoote tegemata tööde loetelu (Product Backlog)*

  Tarkvaratoote *backlog*'i haldamiseks kasutati projektijuhtimise vahendit *Jira*. Kõikidel meeskonna liikmetel oli sellele ligipääs. Kuigi Scrum metoodika kohaselt peab tooteomanik *backlog*'i töid pidevalt prioriteedi järgi järjestama, ei olnud toote *backlog* prioritiseeritud. *Backlog*'i tööde kirjeldused olid suures osas ebapiisavad, mistõttu olid ärinõuded arendusmeeskonnale tihti ebaselged või arusaamatud. Kuna mitu inimest (tooteomanik, meeskonnajuht, süsteemianalüütik, tehniliste tööde puhul ka süsteemiarhitekt) lisasid töid *backlog*'i ning tööde pidevat prioritiseerimist ei toimunud, ei olnud kellelgi selget pilti *backlog*'is olevatest töödest ning nende prioriteetidest. Lõpuks kasvas toote *backlog* väga suureks ning mõned tööd olid duplitseeritud (lisatud erinevate inimeste poolt). Teise süsteemi arenduse algusega hakati sellega seotud tööd samasse *backlog*'i lisama, mistõttu kasvas toote *backlog* veelgi suuremaks ning muutus peaaegu hallatamatuks.
- *Sprindi tegemata tööde loetelu (Sprint Backlog)*

  Sprindi *backlog*'i peeti samuti *Jira*'s. Sprindi *backlog*'i tõmmati tooteomanikuga kokkulepitud hulk töid toote *backlog*'ist. Kuigi Scrum metoodika keelab sprindi *backlog*'i uute tööde lisamist sprindi jooksul, toimus see analüüsitavate süsteemide puhul pidevalt, mistõttu ei jõutud tihti kõik sprindi tööd sprindi lõpuks valmis saada ning tuli sprinti pikendada (mõnikord mitu korda). Veel üheks põhjuseks sprintide pikendamiseks olid sprindi töödes ähmaselt sõnastatud ärinõuded, mille väljaselgitamisele kulus mõnikord päris palju aega ning tööd ei saanud sprindi lõpuks tehtud. Kõik ülalmainitud põhjustas mõnikord hilinemisi süsteemide tarnetes.
- *Inkrement*

  Süsteemi arendati ja tarniti inkrementide kaupa. Tarkvara tarned toimusid siis, kui tooteomanik pidas vajalikuks, mingit tarne planeerimist ei tehtud, seega tarned toimusid ebaregulaarselt ja pigem juhuslikult. Mõnikord süsteemide tarned hilinesid eelmises punktis kirjeldatud asjaolude tõttu. See põhjustas klientide rahulolematust, kuna lubatud funktsionaalsus ei jõudnud kokkulepitud ajaks toodangusse. Kuskil ei peetud tarnesse kuuluvate tööde nimekirja, mistõttu

vahepeal oli raske öelda, millise süsteemi versiooni koosseisus mingi töö läks toodangusse.

Scrumi elutsükkel

Alguses püüdis meeskond Scrumi elutsüklit järgida. Töö toimus kahenädalaste sprintide kaupa, iga sprindi alguses toimus sprindi planeerimise koosolek, kus tooteomanik koos arendusmeeskonnaga moodustasid sprindi *backlog*'i toote *backlog*'i töödest. Sprindi lõpus toimus sprindi jooksul valminud tööde ülevaatuse koosolek, kus arendusmeeskond demonstreeris tooteomanikule valminud funktsionaalsusi ning sai tooteomanikult nende kohta tagasiside. Üks kõrvalekalle Scrumi elutsüklist oli see, et retrospektiive ei peetud.

Hiljem aga hakkas meeskond aina rohkem Scrumi elutsüklist kõrvale kalduma peamiselt tooteomaniku madala projektile pühendumuse ja pideva kättesaamatuse tõttu. Sprindid hakkasid pikemaks venima, sprindi planeerimised ja ülevaatused toimusid aina harvemini ning hiljem jäeti need üldse ära, kuna tooteomanikul ei olnud aega nendes osaleda. Seetõttu jäeti Scrumi rakendamise perioodi lõpus ka sprindid ära. Retrospektiive endiselt ei peetud.

4.1.2 Scrum metoodika rakendamise analüüs

Järgnevalt analüüsitakse Scrum arendusmetoodika rakendamist läbi alapeatükis 2.3.1 defineeritud agiilsete metoodikate rakendamise edu võtmefaktorite.

Organisatsioonilised

- *Ettevõttesisene kultuur ja väärtused*
  Vaatluse all oleva ettevõttesisene kultuur ja väärtused toetavad agiilsete metoodikate rakendamist ja soodustavad töötajate vahelist koostööd. Isegi kui kõik meeskonna liikmed ei tööta samas kontoris, on töötajatele võimaldatud mitu erinevat võimalust suhtlemiseks (nt. telefoni- ja videokonverentsi seadmetega varustatud koosolekuruumid, igal töötajal on lauatelefon). Ettevõte soodustab töötajate pidevat õppimist, võimaldades käia tööalastel koolitustel ja rahvusvahelistel konverentsidel. Ettevõtte töökeskkond on töötaja sõbralik, töötajate arvamust kuulatakse ning julgustatakse seda avatult avaldama.

- *Tippjuhtide toetus*

  Ettevõtte tippjuhid toetavad agiilset lähenemist tööprotsesside organiseerimisel. Näiteks, otsustavad arendusmeeskonnad ise, kuidas oma arendusprotsessi paremini üles ehitada ning kuidas organiseerida oma igapäevast tööd. Tippjuhid usaldavad arendusmeeskondi ja ei sekku meeskondade tööprotsessidesse.

Kliendisuhted

Kuna vaatluse all olevad süsteemid on ettevõttesised süsteemid, esineb ettevõtte äripool kliendi rollis ning tooteomanik on äripoole esindaja arendusmeeskonnaga suhtlemisel. Seetõttu kliendisuhete all mõeldakse käesoleva analüüsi raames tooteomaniku ja arendusmeeskonna vahelisi suhteid.

- *Koostöö kliendiga (tooteomanikuga)*

  Koostöö tooteomanikuga oli ebapiisav, kuna tooteomanik oli pidevalt hõivatud muude tööülesannetega ning oli paljudel juhtudel kättesaamatu (nt. sprindi planeerimiste ja ülevaatuste tegemiseks, ärinõuete kohta täiendavate küsimustele vastamiseks jms.).
- *Kliendi (tooteomaniku) pühendumus*

  Tooteomaniku pühendumus oli väga madal ning ta ei osalenud arendusprotsessis nii palju, kui oli vajalik. Tooteomanik ei täitnud oma ülesandeid korralikult, näiteks, olid ärinõuete kirjeldused väga ähmased ja puudulikud, toote *backlog* ei olnud pidevalt prioritiseeritud ning arendusmeeskond ei saanud vajalikku tagasisidet õigeaegselt. Selle tõttu pidid mõned arendusmeeskonna liikmed tooteomaniku ülesandeid enda peale võtma. Ebaselgete ärinõuete tagajärjel pidid arendajad vahepeal tööd ümber tegema nõuete täpsustamisel.
- *Kliendi (tooteomaniku) rahulolu*

  Üldiselt oli tooteomanik arendusmeeskonna tööga väga rahul. Vahepeal tekkisid küll tema ja meeskonna vahel pinged, kuna tooteomaniku arvamusel pidi arendusmeeskond ise puuduolevaid detaile tööde kirjeldustesse lisama ning talle valmis lahendusi pakkuma. Samas ärinõuete kirjeldamine kuulub just tooteomaniku ülesannete hulka, kuna just tema suhtleb ettevõtte äripoolega ja kogub ärinõudeid. Arendusmeeskonnal oli väga raske pakkuda lahendusi, kui puudus selge arusaamine selle kohta, mida täpselt nõutakse.

Töötajad

- *Arendusmeeskonna pädevus*

  Arendusmeeskond omas vajalikku pädevust selleks, et tarnida nõuetelevastavat ja kvaliteetset tarkvara pidevalt muutuvate nõuete ja prioriteetide tingimustes. Süsteemiarhitekt tundis arendatavaid süsteeme väga hästi ning tarkvaraarendajatel oli suur kogemus süsteemide arenduses kasutatavate tehnoloogiate ja tööriistade osas. Kvaliteedi juht suutis tagada süsteemide suure automaattestidega katvuse (umbes 70% mõlema süsteemi puhul). Meeskonna juht ja süsteemianalüütik suutsid tagada enam-vähem tõrgeteta arendusprotsessi vaatamata ebaselgetele ja puudulikele ärinõutele. Ajanappuse tõttu ei tegelenud kahjuks arendusmeeskond arendusprotsessi regulaarsete ülevaatustega ning protsessi täiustamisega.
- *Meeskonna liikmete vaheline koostöö*

  Kuna kogu arendusmeeskond paiknes samas ruumis, oli meeskonna liikmete vaheline suhtlemine ja koostöö alati väga tihe. Meeskonna liikmed aitasid teineteist tööülesannete täitmisel, mistõttu pidevalt toimus ka teineteiselt õppimine.
- *Meeskonna pühendumus ja rahulolu*

  Arendusmeeskond ei olnud tooteomaniku tööga üldse rahul, kuna oli väga raske töötada ebaselgete ärinõuete ning sprindi jooksul järsult muutuvate prioriteetide kontekstis. Samuti mõned arendusmeeskonna liikmed olid pidevalt ülekoormatud, kuna pidid osaliselt täitma tooteomaniku ülesandeid oma tööülesannete kõrvalt. Vastuseks tooteomaniku madalale pühendumusele hakkas ka arendusmeeskonna pühendumus alanema.

Protsess

- *Agiilse meetodi ja selle praktikate järgimine*

  Eelmises alapeatükis kirjeldatud arendusprotsessi Scrum metoodikale vastavuse analüüsist on väga selgelt näha, et alguses püüdis küll meeskond järgida Scrumi põhimõtteid, kuid ajapikku järgiti neid aina vähem ja vähem jättes olulised Scrumi tseremoniaalsused ja praktikad protsessist välja. Võib öelda, et analüüsitava perioodi lõpus ei järginud meeskond ei Scrumi, ega mingit muud arendusmetoodikat, mistõttu muutus arendusprotsess ebaefektiivseks ning mõnes

mõttes ka kontrollimatuks. Peamiseks Scrum metoodika mittejärgimise põhjuseks oli tooteomaniku madal pühendumus ja pidev kättesaamatus. Scrum eeldab tooteomaniku suurt pühendumust ja aktiivset osavõttu arendusprotsessist. Ilma selleta ei ole mõtet Scrum metoodikat rakendada.

- *Pidev protsessi täiustamine*

  Arendusmeeskond ei tegelenud üldse arendusprotsessi ülevaatuste ja täiustamisega, mistõttu kaldus ta aina rohkem ja rohkem Scrum metoodikast kõrvale ning töötas pikka aega ebaefektiivse/ebasobiva protsessi järgi. Peamiseks põhjuseks, miks arendusmeeskond ei pidanud retrospektiive, oli pidev ajanappus, aga ka meeskonna ükskõiksus protsessi suhtes ning suutmatus tuvastada, et kasutuselolev protsess on ebaefektiivne ning vajab kohandamist.

- *Projektijuhtimine*

  Lähenemine projektijuhtimisele toetas inkrementaalset ja iteratiivset tarkvara arendust ning soodustas arendusmeeskonna liikmete tihedat koostööd. Samas protsessi seisukohast vaadates, ei tegeldud piisavalt selliste oluliste agiilse projektijuhtimise valdkondadega nagu tööde planeerimine, nõuete haldamine (s.h. prioritiseerimine) ja tarkvara tarnete korraldamine. Scrumis ei ole projektijuhi rolli ette nähtud, seega langeb suur osa agiilse projektijuhtimise ülesannetest tooteomaniku õlgadele. Muidugi peab arendusmeeskond tooteomaniku igatpidi toetama nende ülesannete täitmisel, kuid ei saa arendusmeeskond need ülesanded täielikult tooteomanikult üle võtta.

- *Koolitused ja mitteformaalne õppimine*

  Agiilsele protsessile üleminekul on väga oluline korraldada mitmeid koolitusi protsessi ja arendusmetoodika kohta kõikidele arendusprotsessi osalejatele (s.h. tooteomanik ja teised huvigrupid). Seda ei olnud tehtud, mistõttu oli kõikidel protsessi osalejatel oma arusaamine protsessist ning vastutusaladest, näiteks ei pidanud tooteomanik tähtsaks sprindi planeerimistel ja ülevaatustel osalemist. Mitteformaalne õppimine toimus arendusmeeskonnas pidevalt tänu tihedale koostööle meeskonna liikmete vahel.

Tehnilised

- *Tehniline tipptase ja hea disain*

  Agiilsed praktikad rõhutavad puhta ja hästi struktureeritud programmikoodi tähtsust ja lihtsa süsteemi disaini tähtsust. Operatiivsüsteem oli Tallinna

arendusmeeskonnale üle antud 5 aastat tagasi ning selleks ajaks olid selle disain ja programmikood kujunenud juba päris keerulisteks. Vana süsteemi lähtekoodi refaktoorimine oleks väga aeganõudev ja riskantne tegevus, seega sellega ei tegeldud. Uue koodi kirjutamisel järgisid arendajad puhta koodi põhimõtteid. Alguses toimusid ka regulaarsed koodi ülevaatused, kuid teise süsteemi arenduse algusega ei olnud selleks enam aega. Uue süsteemi ülesehitamisel püüti järgida lihtsa disaini ja hästi struktureeritud koodi põhimõtteid, kuid pideva ajanappuse ja puudulike ärinõuete tõttu ei olnud see alati võimalik.

- *Tehnoloogiad ja tööriistad*

  Operatiivsüsteemi arenduses kasutatavad mõned tehnoloogiad on pisut vananenud, kuid kahjuks ei saa neid suurte ajakulude ja riskideta välja vahetada. Uue süsteemi arenduses kasutati uusi tehnoloogiad. Töös kasutatavad tööriistad toetavad agiilse arendusprotsessi põhimõtteid, tagades protsessi tõhusust ja läbipaistvust.

4.1.3 Scrum metoodika analüüsi kokkuvõte

Alajaotistes 4.1.1 ja 4.1.2 tehtud analüüsi tulemusena võib välja tuua järgnevad peamised probleemid, millega puutus vaatluse olev meeskond kokku Scrum metoodika rakendamisel:

- Tooteomaniku madal pühendumus ja kaasatus arendusprotsessi
- Ebapiisav ärinõuete haldus (s.h. kirjeldamine ja prioritiseerimine)
- Ebaefektiivne sprindi planeerimine prioritiseerimata ja ähmaselt kirjeldatud ärinõuete tõttu
- Uute tööde lisamine sprinti ja prioriteetide järsk muutus sprindi jooksul, mille tõttu tuli sprintide pikkust pidevalt pikendada, mis omakorda põhjustas hilinemisi süsteemide tarnetes
- Hiline (mõnikord ka olematu) tooteomaniku tagasiside nii ärinõuete täpsustuste osas kui ka tehtud töö kohta
- Arendusmeeskonna liikmete ülekoormatus ning pühendumuse ja rahulolu alanemine
- Arendatavate süsteemide disaini ja lähtekoodi kvaliteedi alanemine ebaselgete ärinõuete ning pideva ajanappuse tõttu

- Scrum arendusmetoodika põhimõtete mittejärgimine tooteomaniku pideva kättesaamatuse tõttu
- Arendusprotsessi ülevaatuse ja täiustamisega mittetegelemine pideva ajanappuse ning alanenud meeskonna pühendumuse tõttu
- Arendusprotsessi madal kontrollitavus ja läbipaistvus

Scrum arendusmetoodika rakendamise analüüsi järeldusena võib öelda, et Scrum ei sobinud vaatluse all olevale meeskonnale eelkõige sellepärast, et meeskond ei suutnud peamisi Scrumi põhimõtteid järgida. Põhjuseks oli eelkõige tooteomaniku pidev kättesaamatus, mille tõttu kaldus arendusmeeskond kogu aeg Scrum metoodikast kõrvale, näiteks, sprinte pikendati, tseremoniaalsusi jäeti ära jms. Selle tõttu halvenes arendusprotsessi läbipaistvus ning see hakkas väljuma kontrolli alt. Samas kasvas ka arendusmeeskonna liikmete rahulematus nii tooteomaniku kui ka protsessi suhtes. Inimesed olid sunnitud pidevalt töötama puudulikult kirjeldatud ärinõuete ja sprindi sees muutuvate prioriteetidega, mis omakorda põhjustas hilinemisi süsteemide tarnetes ning hakkas avaldama negatiivset mõju ka arendatavate süsteemide kvaliteedile.

Arvestades sellega, otsustas arendusmeeskond koostöös tooteomanikuga võtta kasutusele arendusmetoodikat, mis ei vaja nii suurt tooteomaniku kaasatust ning mis lubaks tagada läbipaistvama ja sujuvama arendusprotsessi, vähendades seejuures mittevajalike tegevuste hulka. Arendusmeeskond arvas, et selliseks metoodikaks võiks olla Scrumban ning oli otsustatud Scrumbanile üle minna.

4.2 Scrumban arendusmetoodika rakendamine

Scrumban metoodikat hakkas vaadeldav meeskond rakendama alates aprillist 2016 ning käesoleva lõputöö kirjutamise hetkel on see metoodika meeskonnal veel kasutusel.

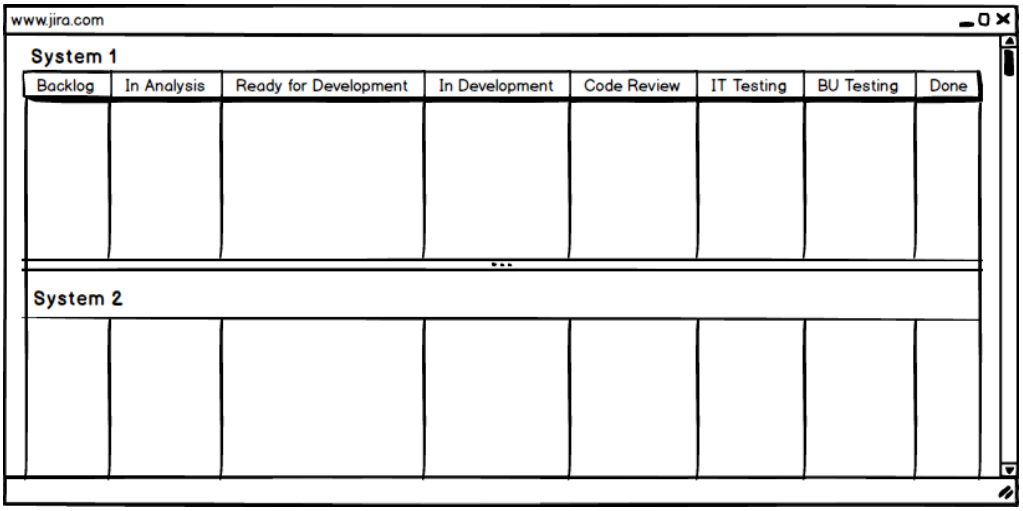
4.2.1 Arendusprotsessi vastavus Scrumban arendusmetoodikale

Järgnevalt analüüsitakse kuivõrd korralikult rakendab analüüsitav meeskond Scrumban metoodikat. Analüüsi aluseks on Scrumban metoodika kirjeldus alapeatükis 2.2.2.

- *Töövoo visualiseerimine*

  Töövoo visualiseerimiseks kasutatakse virtuaalset tahvlit projektijuhtimise vahendis *Jira*, kuhu igal meeskonna liikmel on ligipääs. Kuna korraga arendatakse

kahte süsteemi, on tahvel jagatud kaheks horisontaalseks alaks. Iga ala on tegelikult omaette tahvel, mis sisaldab ühe arendatava süsteemi töid. Mõlema süsteemi tahvlitel on samasugused veerud, mis kujutavad kõiki faase, mida tööd läbivad arendustsükli käigus. Selline lahendus suurel määral hõlbustab mõlema süsteemi tööde jälgimist, kuna erinevate süsteemide tööd on üksteisest visuaalselt eraldatud ning kunagi ei lähe omavahel segamini. Mõlema süsteemi töövoogude visualiseerimine kokkuvõtva tahvli abil pakub nii tooteomanikule kui ka arendusmeeskonnale terviklikku ülevaadet süsteemide arenduste kohta ning lubab kiiresti tuvastada töövoogude pudelikaelu ja probleemseid töid. See aga suurendab arendusprotsessi läbipaistvust ja kontrollitavust. Joonis 4 kujutab arendusmeeskonnal kasutuselolevat tahvlit.



Joonis 4. Meeskonnal kasutuselolev Scrumban tahvel (autori joonis). *IT – Infotechnology Unit, BU – Business Unit*

- *„Pull" põhimõte*

  „Pull" põhimõtet järgitakse osaliselt. *Backlog*'ist „ei tõmba" süsteemianalüütik üldjuhul töid ise, vaid töid lisab *„In Analysis"* veergu kas tooteomanik või meeskonna juht tooteomaniku informatsiooni põhjal. Põhjuseks on see, et *backlog*'i tööd ei ole endiselt prioritiseeritud ning *backlog* endiselt sisaldab liiga palju töid, millest paljud on ka vananenud sisuga. Samuti pidevalt muutuvate prioriteetide tõttu lisatakse mõned tööd analüüsi faasi kohe peale loomist. Kui analüüs on valmis ja töö sisaldab kõiki vajalikke detaile, lisab analüütik tööd *„Ready for Development"* veergu. Selles veerus olevaid töid püütakse hoida prioriteedi järgi järjestatuna. Sellega üldjuhul tegelevad meeskonna juht ja

süsteemianalüütik selgitades tooteomanikult välja tööde prioriteete. *„Ready for Development“* veerust „tõmbavad“ arendajad töid *„In Development“* veergu ise. Kui töö on valmis, lisab arendaja seda *„Code Review“* veergu. Kui koodi ülevaatus on tehtud ning arendaja on teinud koodis kõiki vajalikke parandusi, „lükkab“ ta tööd *„IT Testing“* veergu. Kui töö on analüütiku poolt testitud ning vastav süsteemi funktsionaalsus on automaattestidega kaetud kvaliteedi juhi poolt, „lükatakse“ tööd *„BU Testing“* veergu. Selleks ajaks on töös kirjeldatud funktsionaalsus juba vastava süsteemi toodangu eelsesse keskkonda (*system test environment*) lisatud ning tooteomanik ja süsteemi superkasutajad saavad valminud funktsionaalsuse äriloogikat testida. Kui äripoole testid on tehtud, „lükkab“ tooteomanik tööd *„Done“* veergu, millega kinnitab ta, et töö on toodangusse minekuks valmis.

- *Töös olevate tööde (WIP - Work In Progress) limiteerimine*

  WIP limiiti ei ole ühelegi töövoo faasile seatud, mistõttu mõlema süsteemi mõnedes töövoo faasides on pidevalt liiga palju töid. Nendeks faasideks on üldjuhul *„In Analysis“*, *„IT Testing“* ja *„BU Testing“*. Võib öelda, et need faasid on arendusprotsessi peamised pudelikaelad. See on ka arusaadav, kuna süsteemianalüütik peab tegelema nii tööde analüüsi ja arendajatele tööülesannete koostamisega kui ka manuaalse testimisega. Selle kõrvalt tuleb analüütikul tegelda ka superkasutajate toetamisega. Tulemusena on süsteemianalüütik ülekoormatud ning teeb pidevalt multitegumtööd (*multitasking*), mistõttu paljud tööd ootavad tema taga IT-testimise faasis. Selle tõttu hilineb ka arendajatele nende tööde kohta antav tagasiside ning vigade parandamiseks tuleb neil kulutada lisaaega, et meelde tuletada, mida oli tööülesande raames vaja teha. Lisaks tuleb pidevalt lükata edasi süsteemide tarnete tähtaegu, kuna kõik tööd ei saa kokkulepitud tähtajaks korralikult testitud. *„BU Testing“* on pudelikaelaks kuna nii tooteomanik kui ka süsteemide superkasutajad on pidevalt muude tööülesannetega hõivatud. Selle tõttu „lükkab“ tooteomanik tihti töid *„Done“* faasi neid testimata usaldades, et arendusmeeskond on töid piisavalt testinud ning taganud vajalikku tööde kvaliteeti. Selline lähenemine on potentsiaalselt väga riskantne, kuna alati eksisteerib oht, et arendusmeeskond sai ärinõuetest valesti aru ning valmis saanud funktsionaalsus ei tööta päris nii nagu tooteomanik seda kavandas.

- *Meeskonna reeglite defineerimine*

  Reeglid defineeriti alles hiljuti terve ettevõtte hõlmava kvaliteedi juhtimise süsteemi *(Quality Management System)* väljatöötamise raames. Selle raames pidi iga meeskond koostama detailseid arendusprotsesside ning vastutusalade kirjeldusi, mis on ettevõttesiseses *Wiki*'s kättesaadavad. Kuna need kirjeldused on väga detailsed ja mahukad, läheb vajaliku informatsiooni otsimiseks liiga palju aega. Aja kokkuhoidmise, parema arusaadavuse ja jälgimise mõttes oleks nüüd vaja luua ettevõtte *Wiki*'s eraldi lehekülg, kus peamised arendusprotsessi sammud ja meeskonna reeglid oleksid lihtsustatud vormis kirjeldatud. Nii oleks meeskonnal neid võimalik kergemini järgida ning vajadusel muuta, näiteks arendusprotsessi täiustamisel.

- *Planeerimise koosolek*

  Planeerimise koosolek, milles reeglina osaleb ainult arendusmeeskond, toimub iga nädala alguses. Tooteomanik osaleb planeerimise koosolekul ainult siis, kui ta külastab Tallinna kontorit. Viimasel ajal toimub see iga kuu. Planeerimise koosoleku alguses tutvustab meeskonna juht süsteemide arenduste lühiajalisi plaane ning räägib hetkel kehtivatest prioriteetidest kui vastav informatsioon oli tooteomanikult enne koosolekut saadud. Seejärel vaadatakse põgusalt tööde üldist seisu ning detailsemalt vaadatakse *„Ready for Development"* faasis olevate tööde kirjeldusi veendumaks, et neis sisalduvad kõik arendamiseks vajalikud detailid. Samuti annavad arendajad arenduseks valmis töödele hinnanguid tundides. Planeerimise koosoleku peamine eesmärk on tagada, et kõikidel meeskonna liikmetel on vähemalt järgmise planeerimise koosolekuni töö olemas ning kõikidel on nende tööülesanded arusaadavad.

- *Ülevaatuse koosolek*

  Ülevaatuse koosolekuid ei peeta, aga Scrumban metoodika ei kirjuta neid ka ette. Kokkulepe on, et tooteomanik vaatab valminud tööd ise üle, kui nad jõuavad *„BU Testing"* faasi ning küsimuste tekkimisel suhtleb süsteemianalüütiku või tööd teinud arendajaga. Kui tooteomanik on valminud funktsionaalsusega rahul, „lükkab" ta vastavat tööd *„Done"* faasi, mis tähendab, et töö on toodangusse minekuks valmis.

- *Retrospektiiv*

  Retrospektiive ei peeta. Kuigi Scrumban ei kirjuta retrospektiive ette, neid mitte tehes, kaotab arendusmeeskond võimaluse tuvastada protsessi kitsaskohti ning

leida viise nende elimineerimiseks. Samuti, kuna ärivajadused muutuvad pidevalt, eksisteerib oht, et arendusmeeskond ei märka/ei tunnista, et kasutuselolev arendusmetoodika ei toimi enam nii hästi nagu alguses ning et seda on vaja kohandada.

- *Igapäevane koosolek*
  Igapäevased koosolekud toimuvad iga päev. Nendel koosolekutel annab iga arendusmeeskonna liige lühiülevaate tööülesannetest, millega ta hetkel tegeleb ning tema tööd takistavatest probleemidest.

4.2.2 Scrumban metoodika rakendamise analüüs

Järgnevalt analüüsitakse Scrumban arendusmetoodika rakendamist läbi alapeatükis 2.3.1 defineeritud agiilsete metoodikate rakendamise edu võtmefaktorite.

Organisatsioonilised

- *Ettevõttesisene kultuur ja väärtused*
  Vaatluse all oleva ettevõttesisene kultuur ja väärtused endiselt toetavad agiilsete metoodikate rakendamist ja soodustavad töötajate vahelist koostööd. Mingeid kardinaalseid muutusi võrreldes Scrumi rakendamise perioodiga toimunud ei ole.
- *Tippjuhtide toetus*
  Ettevõtte tippjuhid endiselt toetavad agiilset lähenemist tööprotsesside organiseerimisel. Ka selles osas mingeid kardinaalseid muutusi võrreldes Scrumi rakendamise perioodiga toimunud ei ole.

Kliendisuhted

Kuna vaatluse all olevad süsteemid on ettevõttesised süsteemid, esineb ettevõtte äripool kliendi rollis ning tooteomanik on äripoole esindaja arendusmeeskonnaga suhtlemisel. Seetõttu kliendisuhete all mõeldakse käesoleva analüüsi raames tooteomaniku ja arendusmeeskonna vahelisi suhteid.

- *Koostöö kliendiga (tooteomanikuga)*
  Võrreldes Scrumi rakendamise perioodiga, on koostöö tooteomanikuga oluliselt paranenud. Tooteomanik külastab nüüd palju tihedamini (umbes kord kuus) Tallinna kontorit ning reageerib täpsustavatele küsimustele ärinõuete kohta palju kiiremini. Endiselt ei tegele ta *backlog*'i tööde prioritiseerimisega ning analüüsi

faasi lisatavate tööde kirjeldused on endiselt puudulikud ja ähmased, kuid nüüd lisab tooteomanik vajalikke detaile kirjeldustesse ise vastuseks süsteemianalüütiku või teiste arendusmeeskonna liikmete küsimustele. Endiselt ei pööra tooteomanik vajalikku tähelepanu vastuvõtutestimisele usaldades, et arendusmeeskond on vajalikku testimist teinud.

- *Kliendi (tooteomaniku) pühendumus*

  Võrreldes Scrumi rakendamise perioodiga, on tooteomaniku pühendumus paranenud, kuid siiski ei ole see veel päris õigel tasemel, kuna mõlema süsteemi *backlog*'id on endiselt korrastamata, *backlog*'i tööd ei ole pidevalt prioritiseeritud, ärinõuete kirjeldused on puudulikud ning vastuvõtutestimisega tegeldakse väga vähe (kui üldse).

- *Kliendi (tooteomaniku) rahulolu*

  Üldiselt on tooteomanik arendusmeeskonna tööga väga rahul ning mingeid pretensioone temalt ei ole laekunud. Ta usaldab arendusmeeskonda nii palju, et kinnitab toodangusse laskmiseks neid töid, mida ta ei ole isiklikult üle vaadanud/testinud.

Töötajad

- *Arendusmeeskonna pädevus*

  Scrumbani rakendamise perioodil on arendusmeeskonna koosseisus toimunud mitmeid muudatusi: vahetusid meeskonna juht ja süsteemiarhitekt ning tuli 2 uut arendajat juurde. Nendele muudatustele vaatamata ei alanenud arendusmeeskonna pädevus, kuna süsteemiarhitektiks sai üks endistest arendajatest, kellel olid juba väga tugevad teadmised mõlema arendatava süsteemi osas ning igapäevane tihe koostöö meeskonna liikmete vahel võimaldas uutel liikmetel kiiresti saada vajalikke teadmisi, et oma igapäevaste ülesannetega hästi toime tulla. Süsteemianalüütik peab vahepeal endiselt täitma tooteomaniku tööülesandeid, aga vähemas mahus, võrreldes Scrumi perioodiga. Nüüd töötab pidevalt meeskonna juht selle kallal, et tooteomanik ise lisaks tööde kirjeldustesse kõiki vajalikke ärinõudeid. Kahjuks, pideva ajanappuse tõttu endiselt ei pööra arendusmeeskond tähelepanu arendusprotsessi ülevaatusele ja täiustamisele.

- *Meeskonna liikmete vaheline koostöö*

  Arendusmeeskond paikneb endiselt samas ruumis ning meeskonna liikmete vaheline suhtlemine ja koostöö on endiselt tihe.

- *Meeskonna pühendumus ja rahulolu*

  Võrreldes Scrumi rakendamise perioodiga, on arendusmeeskonna pühendumus ja rahulolu paranenud, kuna tooteomaniku pühendumus ja arendusprotsessi kaasatus on suurenenud, mistõttu ei pea enam meeskond ootama tooteomaniku tagasisidet nii kaua kui varem ning töösse minevad ülesanded sisaldavad nüüd paremini kirjeldatud ärinõudeid. Samuti sprintide ärajätmine mõjus positiivselt tööatmosfäärile ning soodustas tööpinge ja stressi maandamist. Siiski on aga mõned meeskonna liikmed ülekoormatud, mistõttu koguneb mõnedes töövoo faasides liiga palju töid, mis ootavad ühe inimese taga. Selle tõttu tekivad töövoos pidevalt pudelikaelad.

Protsess

- *Agiilse meetodi ja selle praktikate järgimine*

  Eelmises alapeatükis kirjeldatud arendusprotsessi Scrumban metoodikale vastavuse analüüsist võib järeldada, et metoodikat üldiselt järgitakse, kuid selle mõnesid olulisi praktikaid ei kasutata. Kuigi Scrumban metoodika on Scrum metoodikast paindlikum ning ei ole niivõrd ettekirjutav [3], [17], võib selle potentsiaalne kasu jääda osaliselt või täielikult realiseerimata oluliste praktikate mittekasutamise tõttu. Näiteks, WIP limiteerimine on üks tähtsamaid Scrumbani praktikaid, mis lubab ennetada pudelikaelte tekkimist ning vältida individuaalsete meeskonna liikmete ületunni- ja multitegumtööd (*multitasking*). Vaatluse all olev meeskond ei kasuta seda praktikat, mistõttu tekivad töövoos pidevalt pudelikaelad ning mõned meeskonna liikmed on sunnitud pidevalt tegema multitegumtööd. Töövoo pudelikaelad põhjustavad pidevaid hilinemisi süsteemi tarnetes ning pidev ülekoormatus põhjustab meeskonna liikmete motivatsiooni langust ja töö kvaliteedi alanemist.
- *Pidev protsessi täiustamine*

  Arendusmeeskond ei tegele endiselt arendusprotsessi ülevaatuste ja täiustamisega. Peamiseks põhjuseks, miks arendusmeeskond ei pea retrospektiive, on ikka pidev ajanappus, kuna korraga tuleb arendada ja hooldada kahte süsteemi. Kuid meeskonnal tuleb aru saada, et arendusprotsessi ülevaatustega mitte tegeledes kaotavad nad võimalusi protsessi parenduskohtade leidmiseks. Näiteks, uurides, kuivõrd hästi järgitakse Scrumban arendusmetoodika põhimõtteid, võiks meeskond tuvastada, et mõned olulised

Scrumbani praktikad, mis võiksid lahendada olemasolevaid probleeme, jäävad kasutamata.

- *Projektijuhtimine*

  Lähenemine projektijuhtimisele toetab inkrementaalset ja iteratiivset tarkvara arendust ning soodustab arendusmeeskonna liikmete tihedat koostööd. Protsessi seisukohast vaadates, hakati nüüd rohkem tegelema selliste oluliste agiilse projektijuhtimise valdkondadega nagu tööde ja tarkvara tarnete planeerimine. Tööde planeerimised toimuvad nüüd iga nädal ning planeerimisel tagatakse, et kõikidel meeskonna liikmetel on järgmiseks nädalaks tööülesanded olemas ning need on arusaadavad. Tarneid planeerib tooteomanik koostöös meeskonna juhiga. Nüüd on võimalik jälgida ka tarnesse kuuluvate tööde listi koos nende staatustega (nt. „analüüsis", „arendamisel", „testimisel"). Tööde listi luuakse automaatselt *Jira* tööde alusel ning see on kõigile kättesaadav ettevõtte *Wiki*'s. Kahjuks, lükatakse planeeritud tarnete tähtaegu pidevalt edasi töövoos esinevate pudelikaelte tõttu. Nõuete haldamisega (s.h. prioritiseerimisega) tegeldakse endiselt ebapiisavalt, kuid tänu *„Ready for Development"* veeru lisamisele on vähemalt lähima tuleviku nõuded selged ja prioritiseeritud.
- *Koolitused ja mitteformaalne õppimine*

  Arendusmetoodika vahetamisel tegi meeskonna juht tooteomaniku ja arendusmeeskonnale üheaegset koolitust, kus selgitas, kuidas muutub arendusprotsess uue metoodika kasutuselevõtuga. Kuna kõik arendusprotsessi osalejad võtsid koolitusest osa, oli kõikidel ühine arusaamine uuendatud protsessist ja vastutusaladest ning hiljem ei tekkinud mingeid lahkarvamusi selle osas. Mitteformaalne õppimine toimub arendusmeeskonnas pidevalt tänu tihedale koostööle meeskonna liikmete vahel.

Tehnilised

- *Tehniline tipptase ja hea disain*

  Süsteemide arendamisel järgitakse puhta ning hästi struktureeritud ja dokumenteeritud lähtekoodi põhimõtteid, kuid pideva ajanappuse tõttu ei ole alati võimalik tagada head koodi dokumenteerituse ja ühiktestidega katvuse taset. Uue haldussüsteemi ülesehitamisel püütakse järgida lihtsa disaini ja kergesti laiendatava arhitektuuri põhimõtteid. Vana operatiivsüsteemi lähtekoodi refaktoorimisega üldiselt ei tegeleta, kuna see on väga aeganõudev ja riskantne

tegevus. Jälle hakati tegelema Scrumi rakendamise perioodil ärajäetud koodi ülevaatustega.

- *Tehnoloogiad ja tööriistad*

  Operatiivsüsteemi arenduses kasutatavad mõned tehnoloogiad on pisut vananenud, kuid kahjuks ei saa neid suure ajakulu ja riskideta välja vahetada. Uue haldussüsteemi arenduses kasutatakse uusi tehnoloogiad. Töös kasutatavad tööriistad toetavad agiilse arendusprotsessi põhimõtteid tagades protsessi tõhusust ja läbipaistvust.

4.2.3 Scrumban metoodika analüüsi kokkuvõte

Alajaotistes 4.2.1 ja 4.2.2 tehtud Scrumban metoodika rakendamise analüüsi tulemusena võib järeldada, et võrreldes Scrum metoodika rakendamise perioodiga, on situatsioon paremaks muutunud eelkõige järgmiste punktide osas:

- Tooteomaniku pühendumus ja kaasatus arendusprotsessi
- Arendusmeeskonna liikmete pühendumus ja rahulolu
- Tööde ja süsteemide tarnete planeerimine
- Süsteemide lähtekoodi kvaliteet
- Arendusprotsessi kontrollitavus ja läbipaistvus

Samas endiselt esineb probleeme ja puudjääke järgmistel aladel:

- Ärinõuete haldus (s.h. kirjeldamine ja prioritiseerimine)
- Töövoo juhtimine (pidev pudelikaelte tekkimine töövoos)
- Arendusmeeskonna liikmete ülekoormatus ja pidev multitegumtöö
- Vastuvõtutestimine (tooteomanikul endiselt ei ole selleks aega)
- Süsteemide tarnete korraldamine (hilinemised tarnetes)
- Arendusprotsessi pidev täiustamine (sellega ei tegeleta)

Üldiselt analüüsi tulemusena võib järeldada, et vaadeldaval meeskonnal on lihtsam Scrumban põhimõtteid järgida, kuna Scrumban ei ole nii ettekirjutav kui Scrum ning lubab sujuvamalt organiseerida arendusmeeskonna koostööd tooteomanikuga (kliendiga). Samuti ei ole Scrumbanis vaja arendusprotsessi sprintideks jagada ning kulutada iga sprint aega kõikidele Scrumi tseremoniaalsustele, mis pideva ajanappuse tingimustes lubab aega kokku hoida ning kulutada seda pigem tööülesannete täitmisele.

Scrumban on paindlikum kui Scrum ning lubab kiiremini reageerida muutustele tööde prioriteetides, kuna uute prioriteetsete töödega (nt. toodangus tuvastatud kriitilised vead) võib hakata kohe tegelema, mitte oodates uue sprindi algust. Lisaks võimaldab Scrumban metoodika paremat arendusprotsessi ülevaatlikkust ja läbipaistvust ning töövoo juhtimise võimalusi, mis muudavad arendusprotsessi kontrollitavamaks. Kuigi Scrumi rakendamisel võib töövoo visualiseerimiseks samuti tahvli kasutusele võtta, ei anna Scrumi tahvel nii head ülevaadet arendusprotsessi tervikuna kohta, kuna Scrumi tahvli uuendatakse iga sprint ning see sisaldab ainult iga konkreetse sprindi töid. Scrumbani tahvel on seejuures püsiv ning täpselt peegeldab kõiki arendusprotsessi faase, mida tööd läbivad arendustsükli jooksul alates toote *backlog*'ist kuni toodangusse minekuni. Lisaks, WIP limiitide seadmine töövoo faasidele pakub töövoo juhtimise võimalusi vältimaks pudelikaelu töövoos ning individuaalsete meeskonna liikmete ülekoormatust ja multitegumtööd. Arendusprotsessi läbipaistvus ja kontrollitavus on väga olulised vaadeldava meeskonna jaoks, kuna kahe süsteemi arendamine ja hooldamine nõuab väga head arendusprotsessi nähtavust ning kiiret muutustele reageerimist. Samuti võib WIP limiitide seadmine töövoo faasidele lahendada probleemi vaadeldava arendusmeeskonna töövoos esinevate pudelikaelte ning meeskonna liikmete ülekoormatusega.

5 Järeldused ja ettepanekud

Scrum ja Scrumban metoodikate rakendamise analüüsi tulemusena jõudis käesoleva lõputöö autor järgmistele järeldustele:

- Üldiselt sobib Scrumban metoodika vaatluse all olevale meeskonnale paremini kui Scrum metoodika, kuna Scrumbani rakendamisel on täheldatud parendusi mitmetes alades võrreldes Scrumi rakendamisega. Põhjuseks on eelkõige see, et Scrumbani on lihtsam järgida situatsioonis, kus meeskond peab tegelema üheaegselt mitme süsteemide arendamise ja hooldusega ning kus uued prioriteetsed tööd tekivad ootamatult ning neid lisatakse pidevalt töövoogu. See järeldus tõendab ka agiilsete spetsialistide arvamust, et Scrumban sobib paremini situatsioonides, kus meeskonnal puudub selge visioon arendatava tarkvaratoote kohta, kus ärinõuded ja tööde prioriteedid muutuvad pidevalt ja ootamatult ning kus tarkvara hooldus on arendusprotsessi lahutamatu osa [12], [13], [17], [18].
- Probleemid, millega puutus vaadeldav meeskond kokku Scrum metoodika rakendamisel, ei ole niivõrd tingitud sellest, et Scrum metoodika ei sobi meeskonnale üldse, vaid pigem sellest, et meeskond ei suutnud järgida Scrumi põhimõtteid. Lõputöö autori arvamusel on Scrum metoodika ebaõnnestumise peamisteks põhjusteks tooteomaniku madal pühendumus, meeskonna liikmete ükskõiksus arendusprotsessi suhtes ning üldine distsiplineerimatus valitud metoodika rakendamisel. Sarnastele järeldustele on jõudnud oma uuringutes ka teised arendusmetoodikate rakendamise uurijad [28], [29], [31], [35]. Seega võib väita, et arendusprotsessi osalejate pühendumus ning valitud metoodika järgimine ja regulaarne täiustamine on arendusmetoodika rakendamise edukuse olulisemad faktorid.

Arvestades analüüsi järeldusi, arvab käesoleva lõputöö autor, et vaatluse all olev meeskond saab jätkata Scrumban metoodika rakendamisega. Scrumbani rakendamise analüüsi tulemusena tuvastatud probleemide lahendamiseks teeb autor järgmised ettepanekud:

- „*Pull*“ põhimõte paremaks realiseerimiseks ning parema ülevaatlikkuse saamiseks täiendada olemasolevat Scrumbani tahvlit järgmiselt:

    1. Lisada eraldi veerg (nt. „*To Do*“) „*Backlog*“ ja „*In Analysis*“ veergude vahel. Sellesse veergu hakkab tooteomanik või meeskonna juht tooteomaniku informatsiooni põhjal lisama töid, mis on süsteemianalüüsiks valmis. Selles veerus peavad tööd olema järjestatud prioriteetide järgi ning tööde kirjeldused peavad sisaldama kõiki ärinõudeid. „*To Do*“ veerust hakkab süsteemianalüütik „tõmbama“ töid *„In Analysis“* faasi.
    2. Jagada „*In Development*“, „*Code Review*“ ja „*IT Testing*“ arendusfaase kaheks veeruks, nt. „*In Process*“ ja „*Done*“. Kui töö saab teatud faasi raames valmis, „lükatakse“ see selle faasi „*In Process*“ veerust „*Done*“ veergu ning järgmisesse faasi „*In Process*“ veergu „tõmmatakse“ töö eelmise faasi „*Done*“ veerust.

- Seada mõlema arendatava süsteemi töövoo kõikidele faasidele (v.a. „*Backlog*“ ja „*Done*“) WIP limiite ning viia sisse uued meeskonna reeglid:

    1. Järgmisesse arendusfaasi et tohi tööd edasi lükata kui selle faasi WIP limiit on juba täis.
    2. Kui töövoo mingi faasi limiit on täis saanud, peavad kõik meeskonna liikmed töötama selle nimel, et selle faasi tööd saavad ennem ära tehtud kui võetakse uus töö „*Ready for Development*“ faasist.

    WIP limiidi seadmine ning ülalpool mainitud reeglite sisseviimine aitab elimineerida pudelikaelu töövoogudes, soodustab meeskonna liikmete vahelist koostööd ning lubab vältida individuaalsete meeskonna liikmete ülekoormatust ja multitegumtööd, mis omakorda avaldab positiivset mõju meeskonna liikmete rahulolule ning lubab vältida hilinemisi kokkulepitud süsteemide tarnete tähtaegades.
- Võtta endale reegliks regulaarselt tegelda arendusprotsessi pideva täiustamisega. Selleks tuleb regulaarselt (nt. kord kuus) viia läbi retrospektiivid, kus osaleb nii arendusmeeskond kui ka tooteomanik. Retrospektiividel vaatab meeskond kasutuseloleva arendusprotsessi üle, näiteks analüüsib kasutuseloleva arendusmetoodika rakendamist, püüab tuvastada arendusprotsessis esinevaid

probleeme, välja selgitada nende põhjused ning leida nendele lahendused. Selleks, et retrospektiiv tooks reaalset kasu, on väga oluline retrospektiivi tulemusi ja kokkuleppeid dokumenteerida ning määrata iga parendusettepaneku elluviimise eest vastutavat isikut. Nii on lihtsam järgmise retrospektiivi ajal probleemide staatust kontrollida ning vajadusel oma tegevusplaani korrigeerida.

6 Kokkuvõte

Lõputöö eesmärgiks oli analüüsida Scrum ja Scrumban arendusmetoodikate rakendamist logistika lahendusi pakkuva ettevõtte ühe arendusmeeskonna näitel selgitamaks välja, millest on tingitud mõlema metoodika rakendamisel esinenud probleemid. Edasiseks eesmärgiks oli jõuda järeldusele selle kohta, kas vaatluse all olev meeskond saab ka edaspidi jätkata Scrumban metoodika rakendamisega või tuleb kaalutleda selle väljavahetamist. Veel üheks eesmärgiks oli teha ettepanekuid arendusmetoodikate rakendamise analüüsi tulemusena tuvastatud probleemide lahendamiseks.

Püstitatud eesmärkide saavutamiseks töötati esmalt läbi teoreetilise materjali ja juhtumuuringud agiilse arendusprotsessi olemusest ja põhimõtetest, Scrum ja Scrumban metoodikate põhimõtetest ja praktikatest ning erinevatest lähenemistest agiilsete arendusmetoodikate sobivuse hindamisele. Teoreetilise materjali läbitöötamise tulemusena formuleeriti agiilsete arendusmetoodikate rakendamise edu võtmefaktorid. Seejärel analüüsiti Scrum ja Scrumban metoodikate rakendamist vaatluse all oleva meeskonna näitel. Mõlema arendusmetoodika rakendamise analüüsimisel esmalt analüüsiti, kuivõrd korralikult järgis vaadeldav meeskond rakendatava arendusmetoodika põhimõtteid ning seejärel analüüsiti selle metoodika rakendamist läbi töö teoreetilises osas formuleeritud agiilsete arendusmetoodikate edu võtmefaktorite.

Töö tulemusena tuvastati Scrum ja Scrumban arendusmetoodikate rakendamisel esinenud probleeme ning analüüsiti nende põhjusi. Scrum metoodika rakendamise ebaõnnestumise põhjuste analüüsi tulemusena jõuti järeldusele, et probleemid, millega puutus vaadeldav meeskond kokku selle metoodika rakendamisel, olid tingitud eelkõige tooteomaniku madalast pühendumusest, meeskonna liikmete ükskõiksusest arendusprotsessi suhtes ning üldisest distsiplineerimatusest valitud metoodika rakendamisel. Kuna sarnastele järeldustele on jõudnud oma uuringutes ka teised arendusmetoodikate rakendamise uurijad [28], [29], [31], [35], tehti üldistav järeldus arendusmetoodika rakendamise edukuse olulisematest faktoritest. Nendeks on arendusprotsessi osalejate pühendumus ning valitud metoodika järgimine ja regulaarne täiustamine.

Scrumban metoodika rakendamise analüüsi tulemusena jõuti järeldusele, et Scrumban metoodika sobib vaatluse all olevale meeskonnale ning et meeskond saab selle metoodika rakendamisega jätkata. Scrumban metoodika rakendamisel tuvastatud probleemide põhjuste analüüsi tulemusena tehti konkreetsed ettepanekud nende probleemide lahendamiseks. Nendeks on: lisada uued Scrumban tahvli veerud, seada WIP limiite igale töövoo faasile, viia sisse uued WIP limiitidega seotud meeskonna reeglid ning regulaarselt tegelda arendusprotsessi pideva analüüsi ja täiustamisega.

Kirjeldatud tulemitega täideti lõputööle püstitatud eesmärgid, kuna tuvastati põhjusi, millest olid tingitud mõlema metoodika rakendamisel esinenud probleemid, jõuti järeldusele, et vaadeldav meeskond saab jätkata Scrumban metoodika rakendamisega ning tehti ettepanekud arendusmetoodikate rakendamise analüüsi tulemusena tuvastatud probleemide lahendamiseks.

Kasutatud kirjandus

[1] VersionOne, „The 10th Annual State of Agile Report,“ 2016.

[2] D. E. Turk, R. France ja B. Rumpe, „Assumptions Underlying Agile Software Development,“ [Võrgumaterjal]. Available: https://arxiv.org/ftp/arxiv/papers/1409/1409.6610.pdf. [Kasutatud 15. märts 2017].

[3] Wikipedia, „Agile Software Development,“ [Võrgumaterjal]. Available: https://en.wikipedia.org/wiki/Agile_software_development. [Kasutatud 21. märts 2017].

[4] I. Sommerville, Software Engineering, Pearson, 2010.

[5] „What's the Difference? Agile vs Scrum vs Waterfall vs Kanban,“ [Võrgumaterjal]. Available: https://www.smartsheet.com/agile-vs-scrum-vs-waterfall-vs-kanban. [Kasutatud 21. märts 2017].

[6] "Manifesto for Agile Software Development," [Online]. Available: http://agilemanifesto.org. [Kasutatud 21. märts 2017].

[7] A. Sidky ja J. Arthur, „A Disciplined Approach to Adopting Agile Practices: The Agile Adoption Framework,“ *Innovations in Systems and Software Engineering,* kd. 3, nr 3, pp. 203-216, September 2007.

[8] „What is Agile Methodology?,“ [Võrgumaterjal]. Available: https://www.versionone.com/agile-101/agile-methodologies/. [Kasutatud 21. märts 2017].

[9] P. Abrahamsson, O. Salo, J. Ronkainen ja J. Warsta, „Agile Software Development Methods: Review and Analysis,“ VTT Publications, [Võrgumaterjal]. Available: http://www.vtt.fi/inf/pdf/publications/2002/P478.pdf. [Kasutatud 15. märts 2017].

[10] „The Scrum Guide™,“ [Võrgumaterjal]. Available: http://www.scrumguides.org/scrum-guide.html#definition. [Kasutatud 21. märts 2017].

[11] R. Costa, „Who, Where, What, When, Why and How to use Scrum - A simple guide,“ [Võrgumaterjal]. Available: https://www.linkedin.com/pulse/who-where-what-when-why-how-use-scrum-simple-guide-rui-costa. [Kasutatud 17. aprill 2017].

[12] „Scrum vs. Kanban vs. Scrumban,“ [Võrgumaterjal]. Available: http://webcache.googleusercontent.com/search?q=cache:6Vo04QlESOoJ:www.eylean.com/Publications/DownloadPublication/4e93cecc-4cb2-4e3f-849d-810d7aea33a5%3Fname%3DWhitepaper---Scrum-vs-Kanban-vs-Scrumban+&cd=14&hl=en&ct=clnk&gl=ee. [Kasutatud 25. märts 2017].

[13] Y. Yeret, „So What is Scrumban?,“ [Võrgumaterjal]. Available: http://yuvalyeret.com/so-what-is-scrumban/. [Kasutatud 21. märts 2017].

[14] H. Kniberg ja M. Skarin, Kanban and Scrum - Making the Most of Both, C4Media, 2010.

[15] C. Ladas, „Scrum-ban,“ [Võrgumaterjal]. Available: http://leansoftwareengineering.com/ksse/scrum-ban/. [Kasutatud 25. märts 2017].

[16] A. Thangaraj, „Scrumban – Being Differently Agile,“ [Võrgumaterjal]. Available: http://www.agilerecord.com/scrumban-%E2%80%92-differently-agile/. [Kasutatud 25. märts 2017].

[17] S. Pahuja, „What is Scrumban?,“ [Võrgumaterjal]. Available: https://www.agilealliance.org/what-is-scrumban/. [Kasutatud 25. märts 2017].

[18] S. Radics, „Scrum, Kanban, Scrumban – a fast overview and rough categorization when to use what method,“ [Võrgumaterjal]. Available: http://www.ontheagilepath.net/2013/09/scrum-kanban-scrumban-a-fast-overview-and-rough-categorization-when-to-use-what-method.html. [Kasutatud 25. märts 2017].

[19] „Whitepaper: Kanban vs Scrum,“ [Võrgumaterjal]. Available: http://www.belatrixsf.com/whitepapers/kanban-vs-scrum/. [Kasutatud 25. märts 2017].

[20] S. Soundararajan, J. D. Arthur ja O. Balci, „A Methodology for Assessing Agile Software Development Methods,“ *2012 Agile Conference*, 2012.

[21] A. Qumer ja B. Henderson-Sellers, „A framework to support the evaluation, adoption and improvement of agile methods in practice,“ *The Journal of Systems & Software,* kd. 81, nr 11, pp. 1899-1919 , 2008.

[22] C. G. Cobb, The Project Manager's Guide to Mastering Agile, John Wiley & Sons, Inc., 2015.

[23] T. Chow ja D.B. Cao, „A survey study of critical success factors in agile software projects,“ *The Journal of Systems & Software,* kd. 81, nr 6, pp. 961-971, 2008.

[24] C. S. Misra, V. Kumar ja U. Kumar, „Identifying some important success factors in adopting agile software development practices,“ *The Journal of Systems & Software,* kd. 82, nr 11, pp. 1869-1890, 2009.

[25] A. S. Koch, Agile Software Development: Evaluating the Methods for Your Organization, Artech House Books, 2004.

[26] T. Dybå ja T. Dingsøyr, „Empirical studies of agile software development: A systematic review,“ *Information and Software Technology,* kd. 50, nr 9, pp. 833-859, 2008.

[27] G. Kalus ja M. Kuhrman, „Criteria for Software Process Tailoring: A Systematic Review,“ [Võrgumaterjal]. Available: https://www4.informatik.tu-muenchen.de/publ/papers/kk2013a.pdf. [Kasutatud 07. aprill 2017].

[28] N. Nikitina, M. Kajko-Mattsson ja M. Strale, „From scrum to scrumban: A case study of a process transition,“ *2012 International Conference on Software and System Process*, 2012.

[29] N. Nikitina ja M. Kajko-Mattsson, „Developer-Driven Big-Bang Process Transition from Scrum to Kanban,“ *2011 International Conference on software and systems process*, 2011.

[30] J. Highsmith, Agile Project Management: Creating Innovative Products, Second Edition, Addison-Wesley Professional, 2009.

[31] N. Nikitna ja M. Kajko-Mattsson, „Historical Perspective of Two Process Transitions,“ *Fourth International Conference on Software Engineering Advances*, 2009.

[32] P. Gorans ja P. Kruchten, „A Guide to Critical Success Factors in Agile Delivery,“ [Võrgumaterjal]. Available: http://www.businessofgovernment.org/report/guide-critical-success-factors-agile-delivery. [Kasutatud 28. märts 2017].

[33] K. Petersen ja C. Wohlin, „A comparison of issues and advantages in agile and incremental development between state of the art and an industrial case,“ *The Journal of Systems & Software,* kd. 82, nr 9, pp. 1479-1490, 2009.

[34] S. Soundararajan, O. Balci ja J. D. Arthur, „Assessing an Organization’s Capability to Effectively Implement Its Selected Agile Method(s): An Objectives, Principles, Strategies Approach,“*2013 Agile Conference*, 2013.

[35] B. Schatz ja I. Abdelshafi, „Primavera Gets Agile: A Successful Transition to Agile Development,“ *IEEE Software,* kd. 22, nr 3, pp. 36-42, 2005.