TARTU ÜLIKOOL
Majandusteaduskond

Mait Klaos

VÄLEDATE TARKVARAARENDUSMETOODIKATE KASUTAMISVÕIMALUSED TARKVARAETTEVÕTTE NÄITEL

Bakalaureusetöö

Juhendaja: dotsent Tõnu Roolaht

Tartu 2016

Soovitan suunata kaitsmisele ……..

(juhendaja allkiri)

Kaitsmisele lubatud “ “..... 2016. a

Rahvusvahelise ettevõtluse ja innovatsiooni õppetooli juhataja prof. Urmas Varblane

…….……

(õppetooli juhataja nimi ja allkiri)

Olen koostanud töö iseseisvalt. Kõik töö koostamisel kasutatud teiste autorite tööd, põhimõttelised seisukohad, kirjandusallikatest ja mujalt pärinevad andmed on viidatud.

……..

(töö autori allkiri)

SISUKORD

SISSEJUHATUS

Tarkvara arendavad ettevõtted muutuvad majanduse järjest olulisemateks osalisteks, kuna arvutid, nutiseadmed jms ning nendes töötav tarkvara mõjutab juba peaaegu kõiki inimelu aspekte. Tarkvara arendavas ettevõttes on turule viidavaks tooteks kas täiesti uus tarkvara ehk toode, selle lisaomadus või olemasoleva tarkvara/toote uus versioon. Tarkvara võib olla arendatud kindlale kliendile või mõeldud müügiks üldisele tarkvaraturule (Sommerville 2011: 6). Olenemata sihtgrupist on ettevõtte jaoks väga oluline vähendada turule viimise aega (*time to market*). Selle aja vähendamine, võttes ettevõttes kasutusele sobivamaid tarkvaraarendusmetoodikaid, aitab tõsta ettevõtte konkurentsivõimet.

Tarkvaraettevõttes tehtav töö tähendab põhiliselt töötamist informatsiooniga kasutades teadmisi, tegemist on teadmistepõhise majandusharuga (Staats *et al* 2010:377). Sellises ettevõttes loovad kliendi jaoks väärtust inimesed. Seega on töötajate töö korraldamine ettevõtte jaoks kriitilise tähtsusega. Takistused, ebaefektiivsus ja muud väärtuse loomist segavad faktorid raiskavad aega, mil töötaja saaks tegeleda ettevõtte jaoks kõige tähtsamaga – kliendi jaoks väärtust suurendava toote või teenuse loomisega.

Autotootmises nii aja kui toorme raiskamise vähendamise eesmärgil kujundatud Toyota tootmissüsteem pani aluse timmitud mõtteviisile (*lean thinking*). Käesoleva bakalaureusetöö olulised mõisted ja metoodikad on kõik timmitud mõtteviisiga seotud. Tiimitud mõtteviis on katusmõiste nii timmitud tootmisele kui ka väledatele tarkvaraarendusmetoodikatele nagu scrum, kanban ja scrumban. Need on kaasaegsed tarkvaraarendusmetoodikad, mille põhimõtted pärinevad timmitud mõtteviisist.

Bakalaureusetöö eesmärgiks on läbi väledate tarkvaraarendusmetoodikate kasutuselevõtu tulemuslikkuse hindamise, töötada välja ettepanekud Eesti tarkvaraettevõttele vähendamaks kasiino täislahenduse kliendile üleandmiseks kuluvat aega.

Eesmärgi täitmiseks püstitas autor järgmised uurimisülesanded:

- selgitada timmitud tootmise ja mõtteviisi olemust;

- uurida kirjanduse baasil väledate tarkvaraarendusmetoodikate olemust ja põhimõtteid;
- tuua välja väledate tarkvaraarendusmetoodikate seos timmitud mõtteviisiga;
- käsitleda varasemaid empiirilisi uurimusi väledate tarkvaraarendusmetoodikate rakendamise kohta;
- valida välja mõõdikud, mis aitavad hinnata väledate tarkvaraarendusmetoodikate tulemuslikkust;
- tutvustada analüüsitavat ettevõtet ning anda ülevaade ettevõtte kasiino üksuse väljalaskeosakonna (*release management*) olemasolevast tarkvaraarendusprotsessist;
- hinnata väleda tarkvaraarendusmetoodika, scrumbani kasutuselevõttu väljalaskeosakonnas;
- teha ettevõttele ettepanekud, kuidas saaks veelgi vähendada kasiino täislahenduse kliendile üleandmiseks kuluvat aega.

Töö on üles ehitatud kaheosalisena. Töö esimesest osa alustatakse timmitud mõtteviisi põhimõtete väljatoomisega, misjärel liigutakse infotehnoloogia valdkonna spetsiifiliste tarkvaraarendusmetoodikate käsitlemiseni. Töö teises, empiirilises osas hinnatakse uuritavas ettevõttes läbiviidud protsessimuutust toetudes teoreetilises osas väljatoodud käsitlustele ning saadud tulemustele toetudes tehakse ettevõttele ettepanekud.

Bakalaureusetöö teoreetilise osa alguses näidatakse Toyota tootmissüsteemi kujunemist kasutades Jeffrey K. Liker, T Spear & Bowen töid. Timmitud tootmise põhimõtted pärinevad Wood, Hines & Rich, ning Landeri töödest. Timmitud mõtteviis (*lean thinking*) on põhjalikku käsitlemist leidnud J.P. Womack ja D.T. Jones raamatutes. Teoreetilise osa teises pooles kirjeldatakse väledaid tarkvaraarendusmetoodikaid (*agile software development*) ja nende rakendamist. Selleks toetutakse J. Highsmithi ja A. Cockburni artiklitele. Scrumi metoodika põhimõtted, mida töös tutvustatakse, pärinevad selle loojate J. Sutherlandi ja K. Schwaberi töödest. Kanbani käsitlusest ülevaate andmiseks toetutakse selle kasutusele võtja D. J. Andersoni aga ka Stellman & Greene raamatutele. H. Kniberg ja M. Skarini raamat on väärtuslik allikas saamaks ülevaadet scrumi ja kanbani praktikas rakendamisest.

Bakalaureusetöö empiirilises osas viiakse läbi juhtumiuuring tarkvaraettevõtte kasiino üksuse väljalaskeosakonnas. Peatüki alguses tutvustatakse esmalt kasutatavaid uurimismetoodikaid ja ettevõtet. Uuringu käigus kaardistatakse osakonnas kasutusel olevad protsessid, tuuakse välja protsessimuudatuse peamised aspektid ja toimunu võrdlus teooriaga. Seejärel viiakse läbi kvalitatiivanalüüs, mille käigus intervjuueritakse protsessis osalejaid, teostatakse dokumendi analüüs ja autoripoolne vaatlus. Ettevõtte poolt kogutud andmete põhjal viiakse läbi ka kvantitatiivanalüüs. Analüüsi käigus võrreldakse ja sünteesitakse kõikide metoodikate rakendamisel saadud tulemusi.

Analüüsi tulemustele toetuvad autori poolt tehtavad parandusettepanekud on kasulikud eelkõige uuritavale ettevõttele, kuid üldised järeldused on sobilikud ka teistele sarnast protsessimuudatust kaaluvale ettevõttetele. Majandusteaduskonnas ja Tartu Ülikoolis ei ole timmitud mõtteviisi ega väledaid tarkvaraarendusmetoodikaid palju uuritud. Autori panus selles valdkonnas on käsitleda timmitud tarkvaraarendusmetoodikate rakendamist maailma mõttes keskmise suurusega tarkvara tootva ettevõtte näitel.

Märksõnad: kulusäästlik mõtlemine, kulusäästlik tootmine, tarkvarafirmad, väle tarkvaraarendus.

1. VÄLEDATE TARKVARAARENDUSMETOODIKATE TEOREETILINE KÄSITLUS

1.1. Timmitud tootmine ja mõtteviis kui väledate tarkvaraarendus-metoodikate mõjutajad

Käesolevas alapeatükis antakse ülevaade timmitud tootmise ajaloost ja kujunemisest. Seejärel avatakse timmitud mõtteviisi olemus ja kirjeldatakse selle põhimõtteid. Väledate tarkvaraarendus metoodikate põhimõtted pärinevad timmitud tootmisest alguse saanud timmitud mõtteviisist.

1937. aastal asutas Jaapani suurima kangastelgede tootmisettevõtte juht Sakichi Toyoda autotootmisettevõtte, nimetades selle *Toyota Motor Company*'ks ja määrates ettevõtte juhiks oma poja Kiichiro Toyoda (75 years…2016). Sakichi Toyoda oli automaatsete kangastelgede väljamõtleja. Need kangasteljed tegi unikaalseks funktsioon, mis seiskas automaatselt tootmise kui lõim katkes. Eelnev võimaldas viga tuvastada, seda koheselt analüüsida ja parandada. See kontseptsioon sai nimeks *jidoka* ehk „inimlik automatiseerimine". *Jidoka* tähendab sellist töökorraldust, kus defekti ei lasta läbi järgmisse tööetappi. (Liker 2004: 33)

Kiichiro Toyoda järgis oma isa Sakichi filosoofiat, mida ta aga omalt poolt täiendas. Kiichiro Toyoda õppis tundma kogu autotootmisprotsessi Fordi autotehases Michiganis, kus ta viibis terve aasta. Samuti mõjutas Kiichirot Ameerika supermarketite süsteem, kus ostjad võtsid ise riiulitelt kauba ja poetöötajad täiendavad kaubaga riiuleid. (Liker 2004: 35) Jaapanisse naastes võttis Kiichiro Toyoda kasutusele Fordi tootmissüsteemi selle osa, kus allhankijatelt telliti ainult nõudluse jagu autoosi. Seda süsteemi hakati nimetama *Just-In-Time* (JIT) ehk täppisajastatud tootmiseks. (Smith ja Hawkins 2004: 6) *Jidoka* ja *Just-In-Time* moodustasid ning moodustavad veel tänapäevalgi kaks Toyota tootmissüsteemi alustala.

Taiichi Ohno oli Toyota tootmissüsteemi alusepanija. Ta alustas oma karjääri Toyoda kangastelgi tootvas ettevõttes 1932. aastal, kus ta katsetas ja rakendas Kiichiro Toyoda arendatud *JIT* põhimõtteid ja meetodit. Pärast teist maailmasõda sai Toyota Motor Company juhiks Eiji Toyoda (Kiichiro Toyoda nõbu). Koos Taiichi Ohnoga asusid nad üles ehitama sõjas kannatada saanud Toyota autotootmist. (Liker 2004: 40) Sõjajärgne majanduskriis, mis seisnes toorme, maa ja finantskapitali vähesuses, sundis peatootmisinseneriks saanud Taiichi Ohnot välja mõtlema järjest säästlikemaid meetodeid (Smith ja Hawkins 2004: 7).

Ressursi vähesus sundis Toyotat tootma ainult seda, mida klient tahtis. Selline Ameerika supermarketist tuttav tõmbepõhine süsteem võimaldas koos *jidoka* ja *Just-In-Time* kasutamisega luua tootmisvoo, kus puudusid üleliigsed varud ning toodeti ainult seda, mida klient soovis. Tulemuseks saadi tõmbepõhine (*pull*) tootmissüsteem, kus iga toode liikus läbi tootmise erinevate etappide (*flow*). Eelnevat nimetati timmitud tootmiseks. Timmitud tootmises koosneb tootmisrakk tööjärjekorras lähestikku olevatest inimestest, masinatest või tööjaamadest, rakuline tootmine on eeldus sujuva töövoo loomiseks. Igal rakul on täita kaks olulist rolli – järgmisele rakule ollakse tarnija, eelmisele klient. (Liker 2004: 48)

Ka Liker toob välja (Liker 2004: 44), et Toyota tootmissüsteemi rakendamine algab kliendi seisukohast tootmisprotsessi jälgimisega. Põhiline, väärtust defineeriv küsimus on– mida klient sellest protsessist saada soovib? Klient võib asuda nii ettevõttes kui väljaspool ettevõtet. Kliendi seisukohast on võimalik jälgida protsessi ning eraldada väärtust loovad ja väärtust mitteloovad etapid. Womack täpsustab (1996: 38), et tootmises peab kaardistama väärtusahelas asuvad tegevused, jagades need kolme kategooriasse:

- tegevused, mis tegelikult loovad tarbija jaoks väärtust;
- tegevused, mis ei loo väärtust aga mis on vajalikud protsessis ;
- tegevused, mis ei loo tarbija jaoks väärtust ja saab protsessist kohe eemaldada.

Spear ja Bowen (1999:98) selgitasid põhjaliku uuringu tulemusena välja Toyota tootmissüsteemi paradoksaalsuse põhjuse – range spetsifikatsioon võimaldab olla paindlik ja loov. Muudatuse tegemiseks on Toyotas kasutusel probleemi lahendamise protsess, mis sisaldab olukorra detailset hindamist ja parenduste plaani. Eelnev on sisuliselt plaanitud muudatuste test. Selline süsteem innustab töötajaid ja juhte

eksperimenteerima, mis on omakorda õppiva organisatsiooni alus. Spear ja Bowen (1999:98) on kirja pannud neli Toyota tootmissüsteemi kirjeldavat reeglit:

1. kõik tööga seotu on detailideni kirjeldatud – töö sisu, järjekord, ajastus ja tulemus;
2. suhtlus tarnija ja kliendi vahel peab olema otsene ja ühemõtteline;
3. toote- ja teenusevoog kulgeb mööda lihtsat kindlaksmääratud rada;
4. kõik täiustused, mis on seotud kas tootmise, masina ja inimese vahelise seosega, peavad olema kooskõlas teadusliku meetodiga[1], läbiviidud õpetaja juhtimise all ning madalaimal organisatsiooni tasemel.

Taiichi Ohno ja Eiji Toyoda väljatöötatud Toyota tootmisfilosoofia „*Toyota Production System*“ (Toyota tootmissüsteem), avaldati Ohno raamatus „Toyota tootmissüsteem: suurtootmise tagamaad” 1988. aastal. Samal aastal avaldas John Krafcik oma artikli „Kulusäästliku tootmissüsteemi triumf“ („*Triumph of the Lean Production System*“), kus ta võttis kasutusele Toyota tootmissüsteemi kirjeldamiseks kasutusele termini „*lean*“, millega tähistas tõhusat Toyota autotehastes kasutatavat tootmissüsteemi (Krafcik 1988:41).

„Lean Enterprise Estonia“ koostöös Ettevõtluse Arendamise Sihtasutusega (EAS) on oma 2013. aastal väljaantud kulusäästliku mõtlemise terminite sõnastikus soovitanud eestikeelseks kasutamiseks terminit „kulusäästlik“ (Kulusäästliku …2013:1). Ingliskeelsele terminile „*lean production*“ on eestikeelseks vasteks „kulusäästlik tootmine“ või „timmitud tootmine“. Kasutusele on võetud „timmitud tootmine“ „*kuna see annab kõige paremini edasi timmitud tootmise sisu, s.t. pidevat parandamist, pidevat tegevuse ja protsesside täiustamist selle nimel et vähendada kulusid, lühendada tarneaegu ja parendada kvaliteeti, samas kui kulusäästlik viitaks ainult kulude vähendamisele*“ (Womack *et al* 2010: x). Tartu Ülikoolis magistritöö kaitsnud M. Säinas (2014) ja P. Vitsur (2014) on oma töödes kasutanud mõistet „timmitud“. Ka käesolevas töös tõlgitakse termin *lean* kui “timmitud”.

Timmitud tootmise on James P. Womack, Daniel T. Jones ja Daniel Roos oma raamatus „Masin, mis muutis maailma „defineerinud järgmiselt: „*Timmitud tootmine on*

[1] Teadusliku meetodi aluseks on vaatluste, mõõtmiste või katsete põhjal hüpoteeside püstitamine, hüpoteeside põhjal järelduste tegemine ja korratavate katsete teel järelduste paikapidavuse kontrollimine ning kui vaja siis hüpoteeside muutmine (Scientific method: 2016).

„timmitud“, kuna võrreldes masstootmisega kasutatakse selle puhul igat ressurssi vähem: poole vähem tehase inimressurssist, tootmispinnast, investeeringutest töövahenditesse ja konstrueerimise ajakulust, et välja töötada uusi tooteid poole lühema ajaga. Samuti on tootmiskohas vaja hoida vähem kui pooli varusid, defekte tekib palju vähem ning võimalik on toota suuremat ja üha kasvavat valikut tooteid.“ (Womack *et al* 2010: 11)

Smith ja Hawkins (2004: 16) tõid välja, et timmitud tootmine on raiskamise eemaldamine igast tootmise etapist. Need etapid on kliendisuhted (müük, arveldamine, teenuse ja tootega rahulolu), tootedisain, tarnijate võrgustik, tootmine, hooldamine, arendamine, kvaliteedi tagamine ja tootmise korraldus. Eesmärk on vähendada inimtööd, valmis- ja pooltoodangut, ajakulu kliendiga suhtlemise alustamisel, arendamiseks kuluvat aega ja ruumi valmistamaks kõrgeima kvaliteediga tooteid kõige efektiivsemalt ja kasumlikumalt.

Liker (2004:41) defineerib timmitud tootmise läbi tarneahela (*supply chain*): eemaldades raiskamise igast protsessi astmest vähendatakse läbimisaega, millega saavutatakse parim kvaliteet ja madalaim hind, suurendades samaaegselt töötajate turvalisust ja tõstes moraali.

Eelpool toodud timmitud tootmise definitsioonid keskenduvad raiskamise vähendamisest saadavale kasule. Raiskamisena defineerib Wood (2004a: 8) igasugust inimtegevust, mis kulutab ressursse, aga ei loo väärtust. Timmitud tootmises eristatakse kolme liiki raiskamist, mille jaapanikeelsed terminid on *muda*, *mura* ja *muri* (Liker 2004: 129):

- *Muda* – väärtust mitte lisav tegevus või protsess*;*
- *Mura* – raiskamine, mille põhjustab ebaühtlase koormusega tootmine;
- *Muri* – inimeste või masinate üle koormamine.

Mura ja *Muri* põhjustavad sageli tegevusi või protsesse, mis ei loo väärtust ehk viivad *Muda* tekkimiseni. Taiichi Ohno jagas *Muda* seitsmesse gruppi (Hines, Rich 1997: 47-48):

1. Ületootmine. Ilma tellimuseta tootmine, mis tekitab raiskamist tööjõukuludes, ladustamises ja transpordis.
2. Ootamine (masina või tööriista järgi). Automatiseeritud masina järgi või halvasti korraldatud protsessist tulenevalt töökorra või tööriista ootamine.

3. Tarbetu transport/edasivedu. Pooltoodangu vedamine tootmisüksuste vahel või vaheladustamine, mis tekitab liigset transpordikulu.
4. Protsessisisesed kaod. Toote disainimisel ja tootmisel tekkiv raiskamine lisandväärtust mitte lisavate liigutuste tõttu või vajamisevast kvaliteetsemate osade tootmine.
5. Üleliigsed laojäägid. Transpordi ja ladustamiskulude kasv, mis on tingitud üleliigsest toormaterjalist, pooltoodangust ja/või valmistoodangust. Eelnev põhjustab nii pikemat tarneaega, tarvituselt kadumist kui ka riknenud kaupa. Üleliigsed laojäägid võivad varjata tootmise tasakaalust väljas olemist, tarnijate hilinemist, defekte, tootmisseadmete katkiolemist ja/või pikemat seadistusperioodi.
6. Tarbetu liikumine. Töötajate ebavajalik liikumine tootmisprotsessi ajal saamaks kätte tööriistu või osasid.
7. Defektid. Defektsete osade tootmine, parandamine, asenduseks tootmine ja kontroll.

Liker (2004:45) on lisanud kaheksanda raiskamise grupi, töötaja kasutamata jäänud loomingulisus. See on aja, ideede, oskuste, parenduste ja õppimisvõimaluste tähelepanuta jätmine, kuna ei kuulatud töötaja ettepanekuid.

Ohno arvates on kõige olulisem raiskamine ületootmine, kuna see põhjustab enamuse teistest raiskamistest. Tootes mis tahes tootmisprotsessi faasis rohkem kui tarbija soovib, tekitatakse pooltoodangu kuhjumist järgmistes faasides. Suurte varudega ei saavutata optimaalset käitumist. Näiteks, kui defektset komponenti kohe ei tuvastata ja ei eemaldata, siis hilisemas tootmisfaasis, kus seda komponenti kasutatakse, põhjustab defektne sisend suurt aja- ja töökulu. Traditsiooniline kulude säästmine keskendub ainult väärtust loovale aga timmitud mõtteviis keskendub väärtusahelale eemaldamaks väärtust mitteloovat. (Liker 2004: 46)

Timmitud tootmine on laiema mõiste timmitud mõtteviisi osa. Timmitud mõtlemisest paremaks arusaamiseks on Arlbjørn & Freytag jaganud oma töös kõik timmituga seotu kolmele tasemele. Kõige kõrgemal tasemel, filosoofia tasemel määratletakse, et timmitud mõtteviis vähendab raiskamist ja suurendab kliendile pakutavat väärtust. Keskmisel tasemel asuvad Toyota tootmissüsteemist alguse saanud viis timmitud mõtteviisi

põhimõtet. Kolmas ja kõige alumine tase on tööriista/tehnika tase, kus paiknevad järgmised metoodikad – 5S töökoha organiseerimise printsiip (Sorteeri, Sea korda, Sära, Standardi, Säilita); väärtusahela kaardistamine; kanban; pidev parendamine (*kaizen*); JIT; TQM; piirangute teooria jne.

Kanban tähendab jaapani keeles kaarti, märki või piletit ja on töövahend juhtimaks töö voogu ja materjali tootmist Toyota laadsetes tõmbetootmis süsteemides (Liker 2004: 53). Kanban on üks olulisemaid timmitud mõtlemise praktikas rakendamise elemente. See on võtmetähtsusega timmitud tootmise juhtimise vahend, mis ütleb mida, kui palju ja kuna toota. Kanban aitab visualiseerida töövoogu, piirata pooleli oleva töö hulka igas töö staadiumis ja mõõta tsükliaega. Kanbani töövoos liiguvad füüsilised kaardid koos materjali või pooltoodetega. Kaardid käituvad nagu piletid, mis võimaldavad töövool liikuda erinevate tööjaamade või protsessi osade vahel. Tihti kasutatakse kanban metoodika juures ka valget seinatahvlit, millele on võimalik kleepida märkmepabereid. Valget tahvlit kasutades liigub töö edenedes tööülesanne (kaart) ühest tööjaamast teise. Tahvlil kujutatakse tööjaamu/tööetappe tulpadena, nii et reeglina liigub „kaart“ vasakult paremale ühest tulbast teise. Nõnda tekib tahvlile vaadates kogu projekti hetkeseisust silmapilkselt terviklik ülevaade – kaartide liikumine näitab, kuidas projekt edeneb, või vastupidi, kus esinevad pudelikaelad. (Ikonen *et al* 2011: 306) Lander (2007:3686) järgi on kanbanil tähtis roll timmitud tootmises ja mõtteviisis:

- kanban on autonoomne planeerimissüsteem, mis töötab reaalajas ja tagab, et töötatakse oluliste tööülesannete kallal;
- kanban toetab töö protsessi parendust, sest kanban lihtsustab arusaamist komponentide ja info voost ning loob selge arusaamise omavahel ühendatud, kuid parandamist vajavate protsesside kohta;
- filosoofilistel põhjustel, kuna tõmbepõhist tootmissüsteemi rakendatakse läbi kanbani kasutuselevõtu, ning kanban võimaldab *JIT* tootmise poole liikuda vähendades näiteks raiskamist ületootmise tõttu.

Womack ja Jones on oma raamatus „Timmitud mõtlemine“ välja toonud viis peamist timmitud mõtteviisi põhimõtet (Womack, Jones 1996:16-26):

- Väärtuse määratlemine (*value*). Ainult lõppklient saab väärtuse olemasolu hinnata.

- Väärtusahela tuvastamine (*value stream*). Väärtusahel on kooslus kõikidest tegevustest, mis on toote jõudmiseks tarbijani vajalikud läbida.
- Voogtootmine (*Flow*). Moodustada tuleb väärtust loovate tegevuste voog. Tootmine tuleb muuta sujuvaks, ilma tõrgeteta kulgevaks.
- Tõmbepõhine tootmine (*Pull*). Oluline on saavutada olukord, kus tegutsetakse alles siis, kui lõpptarbijal tekib toote järgi nõudlus.
- Püüdlemine täiuslikkuse poole (*Perfection*). Protsessi on võimalik piiritult efektiivsemaks muuta, vähendades aega, ruumi, tootmiskulusid ja vigu.

Moyano-Funtes, Sacristian-Diaz (2012:556) toovad oma töös välja teaduskirjanduses lisandunud põhimõtted:

- pühendunud juhtkond,
- austus inimeste vastu,
- kaasa tarneahela juhtkond.

Bhasin *et al* (2006: 64) toovad kirjanduses leitu põhjal oma töös välja, et timmitud mõtteviis peab olema tervet ettevõtet haarav filosoofia. Tähtis on muuta organisatsiooni liikmete mõtlemist ja käitumist. Õige kultuuri olemasolu on eeldus timmitud toimimisele. Timmitud mõtteviisi peaks suhtuma kui teekonda. Lander *et al* (2007: 3685) järgi eeldab pidev parandamine ehk *kaizen* õppivat organisatsiooni, mille kindlustab teadaolevate ning soovitavalt nähtavate standardite kehtestamine. Kui standardeid ei kehtestata, jääb ühe töötaja uutmoodi tegemine ainult individuaaltasemel õppimiseks. Ohno kasutas analoogiat, kus vesi on varud ja kivid on probleemid. Kui vähendada veetaset (varusid), tulevad välja kivid (probleemid). See sunnib organisatsiooni probleeme kas lahendama või protsess lakkab toimimast. Kui protsessid on tihedalt seotud, kerkivad probleemid kiirelt esile ja sunnivad kiirelt tegutsema suunates sellega organisatsiooni õppima. Sarnaselt on seotud raiskamine ja õppimine, mis tähendab, et kesksel kohal on õppimine protsess mitte mingi kindel tehnika või töövahend.

Muutes inimesed võimeliseks oma tööd tegema ning vastutavaks oma töö parandamise eest, standardiseerides seosed individuaalsete varustajate ja tellijate vahel ning surudes voo ja ühenduse probleemide lahendamise madalaimale tasemele, luuakse reeglid moduleeritud organisatsiooni. Eelnev võimaldab ühes tootmisprotsessi osas disainida muudatusi ilma teisi osasid ülemäära mõjutamata ning delegeerida juhtidel vastutust ilma,

et tekiks kaos. (Spear, Bowen 1999:106) Lander *et al* toovad välja (2007: 3695), et timmitud mõtteviisi rakendamisel on vajalik luua süüdistamisvaba keskkond, kus probleemi nähakse kui võimalust parendamiseks. Sel juhul julgevad inimesed katsetada ja teha vigu. Töötajatel peab olema tahe teha asju paremini, et saavutada püstitatud eesmärgid. Selleks et parendamine oleks järjepidev peab olema paika pandud kindel süsteem. Samuti peab olema paigas, kuidas individuaalse õppimisega saadud teadmised muutuksid organisatsiooni õppimise osaks. Selline süsteem toetab inimesi igapäevases pingutuses oma tööd parandamisel. Hästi rakendatuna on see süsteem võtmeks töötajate arengule.

Timmitud mõtteviisist lähtuvat Toyota tootmissüsteemi on eeskujuks võetud ja rakendatud peamiselt traditsioonistest tööstusharudes. Alates 2000. algusest on proovitud rakendada timmitud mõtteviisi ka teadmistepõhises töös. Kui enamus tööd traditsioonilistes ettevõtetes seisneb füüsiliste asjade ja infoga manipuleerimises, siis teadmistepõhine töö sisaldab peamiselt informatsiooniga tegelemist. Järgnevalt tuuakse välja põhjused, miks timmitud mõtteviisi põhimõtted võiks sobida teadmistepõhise töö juhtimiseks Esiteks on teadmistepõhine töö oluliselt muutlikum, näiteks tarkvaraarenduses on küllalt levinud, et klient muudab poole tootmise pealt tingimusi[2]. Töö ebakindlus ei ole seotud ainult kliendiga, vaid ka kasutatavast tehnoloogiast tulenevate piirangutega, aga ka väliskeskkonnamõjudega. Teiseks, teadmistepõhise töö protsessid ja nendevahelised seosed on tihti nähtamatud, erinevalt tootmisest, kus kõik protsessiosad ja nende omavaheline sobivus on nähtaval. Tähtsad tükid on inimeste peades või esindatud sümbolitena arvutites. Töö valmisoleku järku näeb alles siis kui see on (peaaegu) valmis. Seega protsessi peidetus takistab probleemi varakult tuvastamist ja efektiivset parandamist. Kolmandaks, teadmistepõhine töö on sageli disainiülesanne, mis ulatub kõrgetasemelisest arhitektuurist madalataseme detailideni. (Staats *et al* 2010:377) Seosed timmitud tootmise ja mõtteviisi ning väledate tarkvaraarendusmetoodikate vahel on toodud alapeatükk 1.2 lõpus.

[2] Kui Toyotal on Camry sedaani tootmine pooleli, siis kliendi nõudmise peale ei hakata seda ümber ehitama Tacoma veoautoks (Staats *et al* 2010:377).

Alapeatükis toodi välja Toyota tootmissüsteemi tekke eeldused ning selle aluspõhimõtted, mis on kasutusel ka timmitud tootmise defineerimisel. Näidatakse kuidas timmitud mõtteviis koondab suuremaks filosoofiaks kõik timmituga seotu.

1.2. Väledate tarkvaraarendusmetoodikate põhimõtted

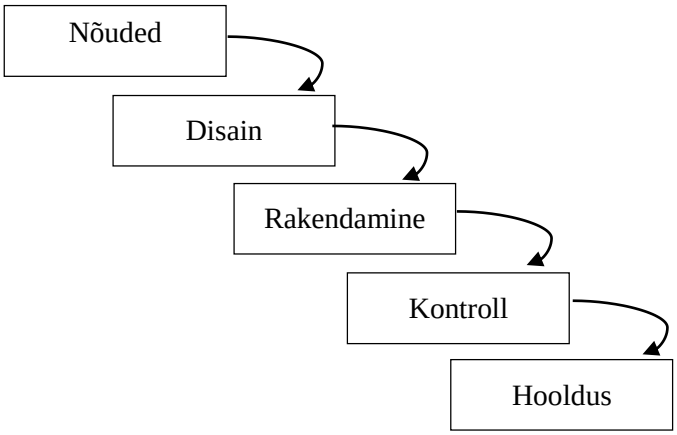
Alanud alapeatüki alguses tuuakse välja tarkvara ja tarkvaraarenduse mõiste, tutvustatakse ja võrreldakse tarkvaraarendus mudelite teooriaid. Alapeatüki teises pooles keskendutakse töös kasutatavate väledate tarkvaraarendusmetoodikate põhjalikumale tutvustamisele, näidatakse nende põhimõtteid, võrreldakse nende erinevust ja sarnasust. Alapeatüki lõpus näidatakse timmitud mõtteviisi seoseid väledate tarkvaraarendusmetoodikatega.

Käesolevas töös uuritav teadmistepõhine töö on tarkvaraarendus. Sommerville (2011: 6) defineerib tarkvara (*software*) kui arvutiprogrammid ja selle dokumentatsioon, mis võivad olla arendatud kindlale kliendile või üldisele tarkvaraturule.

Foster annab oma raamatus laiema definitsiooni tarkvarale: tarkvara on süsteemne kombinatsioon programmidest, andmebaasidest ja dokumentatsioonist ainsa eesmärgiga lahendada spetsiifilisi süsteemi probleeme ja täita etteantud eesmärke. Süsteem on omavahel seotud, üksteisega suhtlevad ja üksteisest sõltuvad komponendid, mis funktsioneerivad ühtse tervikuna täitmaks kindlat eesmärki. Efektiivne süsteem on sünergiline. Süsteemi võib ka defineerida kui kombinatsiooni koostöötavatest inimestest, materjalidest, hoonetest ja varustusest mis loob sisendist mõtestatud ja soovitud väljundi. Tarkvaraarendus on Fosteri (2014) järgi protsess, millega infosüsteeme või tarkvara uuritakse, planeeritakse, modelleeritakse, arendatakse, rakendatakse ja hallatakse. Tarkvaraarendusel on oma elutsükkel – periood, mille jooksul tarkvara välja mõeldi, disainiti, arendati ja on kasutusel. Kasutusel on erinevad tarkvaraarenduse elutsüklit kirjeldavad mudelid. Käesolevas töös keskendutakse kose (*waterfall*) ja väledatele (*agile*) tarkvaraarendusmudelite võrdlemisele. (Foster 2014)

Kose ehk plaanipõhise mudel on alguse saanud tootvast- ja ehitustööstusest. Joonisel 1 on näidatud klassikaline kose mudel, kus iga protsessi etapp (tarkvarale esitatud nõuded (*requirements*), disain, rakendamine, kontroll ja hooldus) voogab järjestikuliselt allapoole järgmisse etappi. Erinevalt tootmisprojektidest ei ole tavapärases

tarkvaraarendusprojektis algusfaasis piisavalt nõudeid teada, et sellist mudelit kasutada. (Lehman, Sharma 2011:750) Seepärast on tarkvaraarenduse jaoks klassikalist kose mudelit pisut muudetud.



Joonis 1. *Klassikaline kosetüüpi tarkvaraarendus mudel*
Allikas: (Lehman, Sharma 2011: 751).

Kruchten (2003: 6) defineerib tarkvaraarenduse kose mudeli järgnevalt: see on tarkvaraarenduse protsessi mudel, kus tegevused edenevad ühest etapist järgmisesse järjestikvoona – kontseptsiooni loomine, algatamine, analüüsimine, projekteerimine, ehitamine, testimine ja haldamine. Staats *et al* (2010: 377) toonitavad, et soorituse kohta on võimalik tagasiside anda alles protsessi lõpus, kus probleemide lahendamine on raskendatud ja kallis. Tabelis 1 on esitatud Kruchteni (2003: 4) väljatoodud tarkvaraarendusprojektide läbikukkumise sümptomid, kui kasutatakse kose mudelit, ja peamised juurpõhjused, miks projektid läbi kukuvad. Lisaks tabelis esitatud põhjustele on Kruchten probleemidena välja toonud veel kontrollimatu muudatuste leviku ning ebapiisava automatiseerimise.

Tabel 1. Tarkvaraarendusprojektide läbikukkumise sümptomid ja juurpõhjused

Sümptomid	**Juurpõhjused**
ebaselge arusaamine lõppkasutaja vajadustest	*ad hoc* nõuete muutmine tarkvara kohta
suutmatus kaasas käia muutunud nõuetega, mis on esitatud tarkvara kohta	mitmetähenduslik ja ebatäpne suhtlus nii tellija kui meeskonna liikmete vahel
tarkvara moodulid ei sobitu omavahel	tarkvara habras arhitektuur
tarkvara loomine, mida on raske hooldada või laiendada	üleliigne keerukus

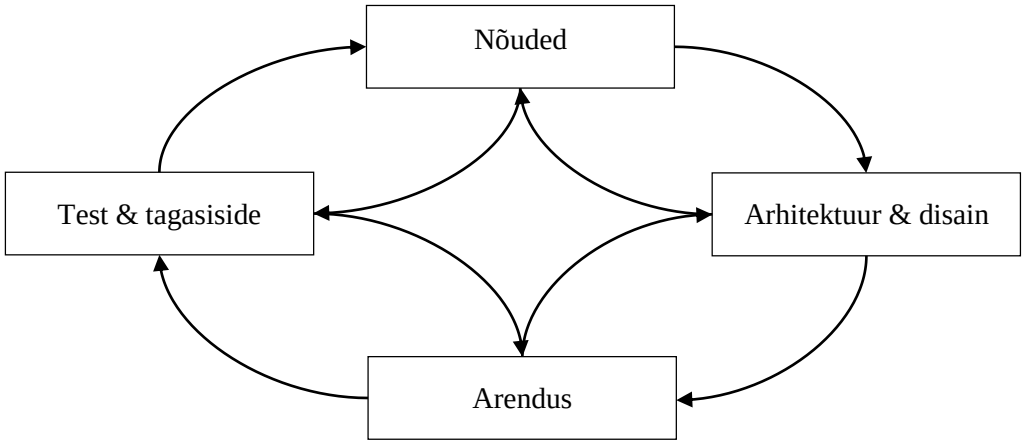
Sümptomid	Juurpõhjused
tõsiste vigade ilmnemine projekti hilisemas faasis	märkamatuks jäänud vastuvääkivus nõuetes, disainis ja/või rakendamises
madal tarkvara kvaliteet	ebapiisav testimine
vastuvõetamatu tarkvara jõudlus	projekti staatuse subjektiivne hindamine
meeskonnaliikmete vaheline miskommunikatsioon, pole teada kes muutis mida, kuna, kus ja milleks	riskide vale hindamine ja nendega tegelemine
	kontrollimatu muudatuste levik
	ebapiisav automatiseerimine

Allikas: (Kruchten 2003: 3), autori koostatud.

Timmitud mõtteviisi rakendamine läbi väleda tarkvaraarendusmudeli rakendamine aitaks paljusid eelpool nimetatud juurpõhjuseid vältida või parandada esinevaid sümptomeid. Lehman & Sharma (2011:751) järgi on põhiline erinevus väledate tarkvaraarendusmetoodikate ja plaanipõhiste metoodikate (nt. kose mudel) vahel see, et kui plaanipõhised metoodikad tõstavad esile plaani kui sellist, siis väledad metoodikad keskenduvad kliendile. Iteratiivne tarkvaraarendus, mida võimaldab väledate tarkvaraarendusmetoodikate kasutamine, hoiab eelpool toodud tarkvaraarenduse juurprobleemid ära, kuna (Kruchten 2003:8):

1. tõsised möödarääkimised tulevad välja arenduse alguses, kui nendega on võimalus tegeleda;
2. iteratiivse tarkvaraarendus võimaldab ja julgustab kasutajat andma tagasisidet, mis omakorda võimaldab defineerida nõuded tarkvara kohta;
3. iteratiivse tarkvaraarenduse arendusmeeskond peab keskenduma projekti jaoks kõige kriitilisematele ülesannetele ja on kaitstud segavate vähemtähtsate ülesannete eest;
4. toimub pidev, iteratiivne testimine, mis võimaldab objektiivselt hinnata projekti staatust;
5. vasturääkivused nõuetes, disainis ja rakendamisel leitakse üles varakult;
6. meeskonna, eriti testijate, töökoormus jaotub ühtlasemalt üle kogu projekti elutsükli;
7. meeskond saab jooksvalt vigadest õppida ja tegeleda pidevalt protsessi parendamisega;
8. projekti tellijad (*stakeholders*) omavad reaalajas infot projekti staatuse kohta.

Väleda tarkvaraarendusmudeli aluseks on iteratiivsed tsüklid. Joonisel 2 on näha arenguetappide omavahelised seosed. Igal etapil on tagasisidemehhanism eelmisse etappi. On üldine kokkuleppe, et ühelgi etapil pole lõppu, kõik etapid on pidevas arengus, see on vastuolus kose tüüpi planeerimisega, kus eeldatakse, et ennem järgmise etapi algamist lõppeb eelmine etapp ära. (Lehman, Sharma 2011:751)



Joonis 2. Väleda tarkvaraarendusmudeli skeem
Allikas: (Lehman, Sharma 2011:751).

Töö autor on koondanud kirjanduses väljatoodud kose mudeli ja väleda tarkvaraarendusmudeli põhimõtted Tabelisse 2. Põhimõtteliselt ei saa pidada ühte metoodikat teistest paremaks, kuigi väleda mudeli kasutamine võimaldab kose mudelis tekkida võivad probleemid kiiremini üles leida. Mõlemal mudelil on omad plussid ja miinused ja seega sõltub arendusmetoodika valik tingimustest. Awad (2005:35) toob välja kolm põhilist määrajat mudeli valikul – projekti suurus, inimesed ja risk. Mida suurem projekt (eelarve, kestvus ja organisatsiooni suurus), seda sobivam on põhjalikumat planeerimist, dokumenteerimist ja erinevate osapoolte koordineerimist soosiv kose tüüpi mudel. Mida paindlikum on organisatsioon, seda lihtsam on kasutusele võtta väledad mudelid.

Tabel 2. Kose ja väleda mudeli võrdlus

	Kose mudel	**Väle tarkvaraarendusmudel**
Nõudmised tarkvarale	Defineeritakse alguses, jäigad	Viimasel hetke muudatused on võimalikud
Planeerimise aeg	Pikk	Lühike
Skoop (*Scope*)	Arendatakse täpselt vastavalt nõuetele	Arendatakse seda, mida kliendil vaja on

	Kose mudel	Väle tarkvaraarendusmudel
Kliendi seotus arendusega	Projekti sisendi saamisel ja valmis tarkvara üleandmisel	Igapäevane koostöö
Arendusprotsess	Lineaarne	Iteratiivne
Probleemide avastamisele kuluv aeg	Pikk	Lühike
Väljalase	Lõpus „valmis“ versioon	Sage, väikeste muudatustega
Testimine	Eraldi faas, pärast arenduse lõppu	Iga iteratsiooni sees tehakse funktsionaalne ja ühiku test
Muudatuse hind	Kõrge	Madal
Võime muudatusele kiirelt reageerida	Madal	Kõrge
Dokumentatsioon	Tehakse ette ja kogu tarkvara jaoks	Vajadusel ainult tehtud töö jaoks
Meeskonna suhtlus	Iga faasi alguses	Pidev, funktsioonide üleselt

Allikas: (Tsui *et al* 2013:99, Awad 2005:35-36) põhjal autori koostatud

2001 aasta veebruaris koostasid väledate arendusmetoodikate propageerijad kokku Väleda Tarkvaraarenduse Manifesti (Highsmith, Cockburn 20011:121). Selles on kirjas (Highsmith, Cockburn 20011:121): „*Me leiame paremaid võimalusi, et ise tarkvara luua ja aidata teistel seda teha. Seeläbi me hindame rohkem:*

- ***inimesi ja nendevahelist suhtlust*** *kui protsesse ja arendusvahendeid,*
- ***töötavat tarkvara*** *kui kõikehõlmavat dokumentatsiooni,*
- ***koostööd kliendiga*** *kui läbirääkimist lepingute üle,*
- ***reageerimist muutunud oludele*** *kui algse plaani järgimist.*

Hoolimata nimekirjas paremal pool väljatoodud tegurite väärtusest, hindame me nimekirjas vasakul pool [3]olevaid tegureid kõrgemalt.“

Järgnevalt peab töö autor vajalikuks tuua ära samade autorite (täielik autorite nimekiri toodud Lisas 5) poolt loodud kaksteist väleda tarkvaraarenduse printsiipi (12 Principles…2016):

1. Kliendi rahulolu maksimeerimine läbi tarkvara kiire ja pideva väljalaske.
2. Tarkvarale esitatavate muutuvate nõuete tervitamine, isegi hilises arendusfaasis.
3. Töötava tarkvara korrapärane väljalase (nädalate, mitte kuude kaupa).
4. Arendajate ja tellija vaheline igapäevane näost-näkku suhtlus.
5. Projektis motiveeritud ja usaldusväärsete inimeste osalemine.
6. Näost-näkku suhtluse kui parima suhtlusmeetodi kasutamine (kokku istumine).

[3] Rõhutamaks originaali väljenduslikkust on toodud Agile Manifesto tsitaadina, kus „vasakul olevaid“ on rasvases kirjas.

7. Projekti edukuse hindamine läbi töötav tarkvara.
8. Jätkusuutliku arendus edendamine väledate protsesside abil.
9. Pideva tähelepanu pööramine tehnilisele väljapaistvusele ja heale disainile.
10. Lihtsuse väärtustamine.
11. Iseorganiseeruvate meeskondade kasutamine.
12. Kohanemine pidevalt muutuva olukorraga.

Paljudes tarkvaraarendusest kirjutatud raamatutes ja artiklites, kus käsitletakse väledaid tarkvaraarendusmudeleid, tsiteeritakse Väleda Tarkvaraarenduse Manifesti. See näitab, et 2001. aastal loodud Väleda Tarkvaraarenduse Manifest on ka mitmeid aastaid hiljem aktuaalne. Sealsed põhimõtted on kinnistunud väledatesse tarkvaraarendusmetoodikatesse, nagu scrum ja kanban. Väleda tarkvaraarenduse printsiipide ideestik kattub timmitud mõtteviisi põhimõtetega. Mõlemad seavad eesmärgiks kliendile väärtuse loomise, keskendudes väärtusahela leidmisele, mis võimaldab tekitada väärtust loovate tegevuste pideva voo. Panustamine inimeste loovusele, andes neile võimaluse protsessi pidevalt parendada, võimaldab vähendada raiskamist. Mõlemad rõhuvad ka suhtlemisele ja pidevale olukorraga kohanemisele ning iseorganiseerumisvõimele.

Eelpool toodud printsiibid on juhised arendamaks kõrgekvaliteedilist tarkvara väledalt. 2001. aastal on „Agile Alliance“ toonud välja järgmised väleda mudeli põhiomadused, mis sarnanevad eespool mainitud printsiipidega (Bustard *et al* 2013:139):

- oluline on programmeerijate meeskonna ja äriekspertide tihe koostöö;
- kommunikatsioon peab toimuma näost-näkku, mis on efektiivsem kui kirjutatud dokumentatsioon;
- ärilist väärtust omavaid versioone tuleb lasta välja sageli, lühikeste ajavahemike tagant;
- olemas peab olema kokkuhoidev iseorganiseeruv meeskond, mis suudab pidevast tarkvara nõuete muutumisest hoolimata kirjutada koodi.

Kui eespool on toodud välja väleda tarkvaraarenduse printsiibid ja väleda mudeli põhiomadused, siis väleda tarkvaraarenduse definitsiooni annab Ambler (Agile…2010): väle tarkvaraarendus on evolutsiooniline, koostööl põhinev, kvaliteedile fokusseeriv lähenemine tarkvaraarendusele, kus potentsiaalselt töötav variant väljastatakse kliendile

regulaarselt. Dingsøyr *et al* (2012:1214) toovad ära veel laiema väleda tarkvaraarenduse definitsiooni: tarkvaraarendus on väle, kui ollakse kogu aeg valmis kiirelt muudatusi looma või tagantjärele ellu viima, õppides samaaegselt muutunud olukorrast ning pakkudes kliendi jaoks olulist väärtust.

Väle tarkvaraarenduse mudel iseenesest ei anna ette täpseid juhiseid, kuidas igapäevast tööd korraldada. Selleks on loodud metoodikad ja raamistikud. Olemasolevad väledad metoodikad on Abrahamsson *et al* (2002:18) järgi *Extreme Programming (XP), Scrum, Crystal family of methologies, Feature Driven Development, the Rational Unified Process (RUP), Dynamic Systems Developemtn Method, Adaptive Software Development* ja *Open Source Software Development*. Kniberg, Skarin (2011: 9) toovad välja väledate tarkvaraarendusmetoodikate ettekirjutatuse (kui täpselt on tarkvaraarendusprotsessi reeglid ja protseduurid ning tööriistad etteantud) võrdluse, kus on toodud, et RUP metoodikal on üle 120 reegli, XP 13 reeglit on natuke rohkem kui scrumi 9 reeglit, kanbanil on ainult kolm (visualiseeri oma töövoog ja limiteeri pooleliolev töö). Kuna autori uuritava ettevõtte protsessimuudatuse tuumaks on scrumi, kanbani ja nende kahe sünteesi – scrumbani kasutuselevõtt, siis ka käesolevas töö fookuses on need kolm väledat tarkvaraarendusmetoodikat. Oluline on samas ka ära märkida, et väledad tarkvaraarendusmetoodikad, eesotsas scrumiga, on 2014 aasta seisuga levinuimad[4] tarkvaraarendusmetoodikad (State of…2016)

Aastal 1995. esitlesid Schwaber ja Sutherland teadusartiklit, kus oli esimest korda kirjeldatud scrum metoodikat (OOPSLA: 2016). Scrum ei ole akronüüm, vaid termin pallimängust ragbi, millega tähistatakse moodust, kuidas läbi meeskonna pingutuse pall mängu tagasi tuua (Babar et al 2013: 39). Ken Schwaber ja Jeff Sutherland on mõlemad Väleda Tarkvaraarenduse Manifestile allakirjutanud, väledaid metoodikaid propageeriva organisatsiooni „Agile Alliance“ alusepanijad ning scrumi metoodika autorid.

Sutherland ja Schwaber defineerivad scrumi kui metoodika järgnevalt: scrum on iteratiivne, järkjärguline raamistik projektidele toote või tarkara arendamiseks (Sutherland:16). Goldsteini definitsioon keskendub suurima ärilise väärtuse andmisele

[4] 2015. aastast pärit 3925 ettevõtet hõlmanud 9ndat korda läbiviidu väledate metoodikate kasutusuuringu tulemuste põhjal kasutab 94% vastanutest väledaid metoodikaid, sellest scrumi 56%, scrum/xp segu 10%, scrumbani 6% ja kanbani 5%. (State of…2016)

(Goldstein…2016): scrum on väle raamistik, mis võimaldab võimalikult lühikese ajaga kasvatada ettevõtte väärtust. Overhage *et al* (2011: 2) definitsioon rõhutab ärivajaduste rahuldamist: scrum on juhtimis ja kontrollimehhanism, mis võimaldab fokusseerida tegevust turgu/klienti rahuldava tarkvara loomisele. Need kolm definitsiooni toovad välja scrumi väga olulised aspektid – iteratiivsus, ettevõtte väärtuse kasvatamine ning kliendi soovide täitmine.

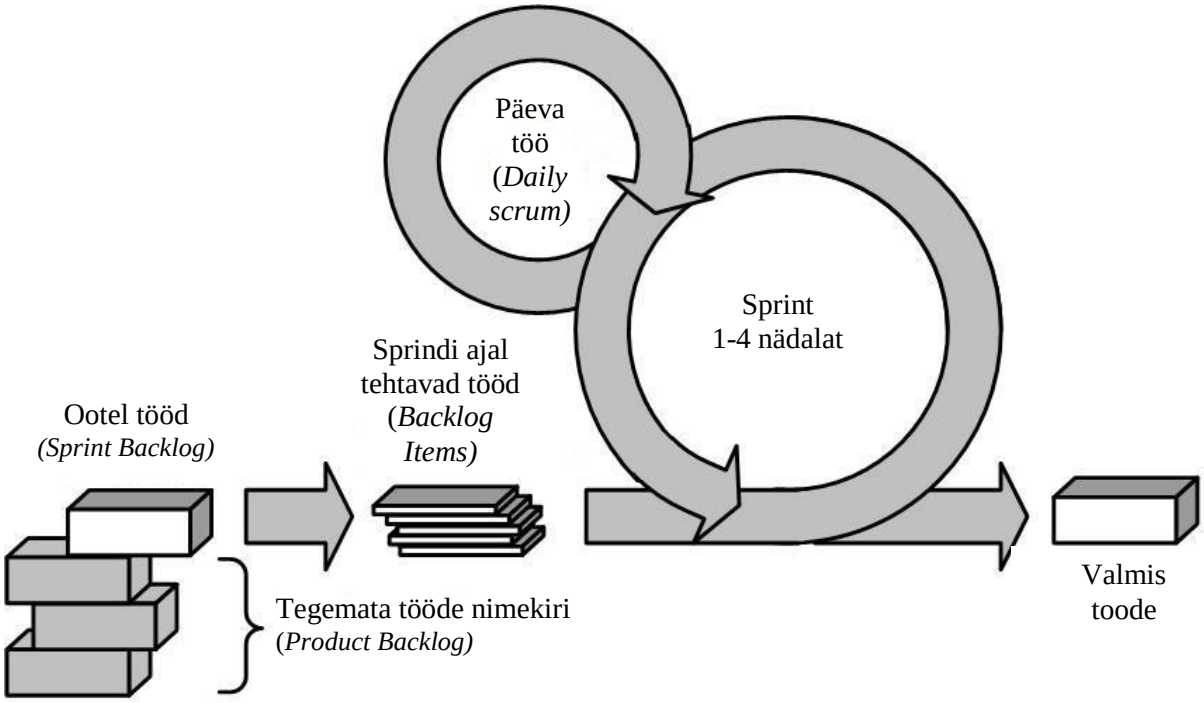
Scrum kasutab timmitud mõtteviisist tuntud „tõmbe" tehnikat saavutamaks läbi süsteemi kulgevat voogu ning vältimaks ülekoormamist. Selline süsteem eksisteerib arenduse lühitsükli ehk sprindi planeerimises, kus meeskonna võimuses (*empowered*) on valida just niipalju tööd kui antud perioodil on jõukohane teha. Meeskonnalt eeldatakse, et nad annavad endast kõik, et sprindi aja sees lubatud tööga kvaliteetselt valmis saada. Scrum rakendab raiskamise ehk *muda* vähendamise protsessi. Scrumi meeskonda julgustatakse leidmaks üles takistused ning kui meeskond ei saa iseseisvalt neid eemaldada, siis scrumi meister (s*crum master)* töötab selle nimel, et tõkked eemaldatud saaks. Kui scrumi meister ise ei saa neid eemaldada, kommunikeerib ta probleemi tervele organisatsioonile ning prioritiseerib selle lahendamise. (Barton 2009:1)

Töö on scrumis organiseeritud lühitsüklite kaupa, iga sprindi pikkus on 1–4 nädalat, sprintide vahel pause pole. Sprindid on ajalahterdatud (*timeboxed),* lõppevad ettemääratud ajal, olenemata sellest, kas töö on lõpetatud või mitte. Sprinte ei pikendata kunagi. Iga sprindi alguses toimub sprindi planeerimise koosolek (*sprint planning*), kus meeskond (*development team*) valib endale ootel tööde nimekirjast (*product backlog*) tööd, mille ta usub saavat valmis sprindi lõpuks. Ootel tööde nimekiri on kogum tellija nõuetest (*customer requirements)* ja tarkvarale esitatud omadustest (*features*). Selle koostab ja prioritiseerib toote omanik (*product owner*) koos huvitatud osapooltega (*stake-holders*), kelleks on tarkvara tellija esindaja. Toote omanik esindab projektis tellija huvisid. (Sutherland, Schwaber: 15)

Seda väikest segmenti tegemata tööde nimekirjast, mis valitakse sprinti, nimetatakse sprindi nimekirjaks (*sprint backlog*). Meeskond pühendub valitud töid valmis saama sprindi lõpuks. Sprindi ajal tööde nimekirja ei muudeta ja nõudmisi ei täiendata. Sprindi langustrendi graafik (*burndown chart*) näitab igapäevaselt, kui palju on veel sprindi jooksul tööd teha. Tavaliselt on see graafik, kus Y-teljel on tööde arv ja X-teljel

kuupäevad. Peaks moodustama langev graafik, kus on visuaalselt kohe näha kuidas käesolev sprint edeneb. (Kniberg, Skarin 2011: 54,57)

Meeskond on scrumis funktsioonideülene (*cross-functional*). See tähendab, et meeskond koosneb kõikidest arenduseks vajaminevatest rollidest. Sprindi lõpus peab olema valmis väljalastav toode. Sprindi ajal koguneb meeskond lühikeseks (kuni 15min) igapäevaseks püstijalakoosolekuks (*daily Scrum meetings/Stand Up)*, kus iga liige saab vastata kolmele küsimusele: Mida Sa eile tegid? Millega Sa täna tegeled? Kas on töö tegemist segavaid takistusi? (Sutherland, Schwaber: 16)



Joonis 3. Scrumi protsess
Allikas: (Sutherland & Schwaber:16), autori kohandused.

Igapäevased koosolekud võimaldavad jälgida tööülesannete progressi ja projekti juhtida. Scrumi kui metoodika rakendamist projektis kontrollib ja koordineerib scrumi meister viies läbi igapäevased koosolekuid ja kindlustades scrum kasutamise projektis. Iga sprindi lõpus tehakse koos tellijaga sprindi ülevaate koosolek (*sprint review*), kus demonstreeritakse tehtut ja saadakse tagasisidet aspektidele, millele meeskond peab keskenduma järgmise sprindi ajal. Iga sprindi lõpus toimub sprindi tagasivaade (*sprint retrospective*), kus osaleb terve meeskond, scrum meister ja tooteomanik, seal arutatakse mis õnnestus ja mida saab teha paremini. Sprindid kestavad, kuni viimane töö ootel tööde

nimekirjast on tehtud. Siis on tarkvarale seatud nõuded (*requirements*) täidetud ja tarkvara valmis. Selliselt võimaldab scrumi metoodika rakendamine tarkvaraarendusprojektidele vajalikku paindlikkust läbi faaside/sprintide. (Kaleshkova *et al*: 2015:184)

David Anderson tutvustas väledat tarkvaraarendusmetoodikat kanban esimest korda 2007. aastal „Agile2007" konverentsil (Scotland: 2015). Anderson *et al* (2010: 6) järgi on kanban metoodika, mis pärineb timmitud tootmise mõtteviisist ning mille alustalaks on kolm põhikomponenti: visualiseeritud tööde juhtimine, käimasoleva töö hulga piiramine ja töö süsteemist läbi "tõmbamine. Kanbani rakendamise eesmärk on vähendada läbimisaega ehk tarneaega (*lead time*). Stellman & Greene (2014:319) toovad oma raamatus välja, et kanban ei ole projektijuhtimissüsteem, vaid metoodika parandamaks kasutusel olevaid tarkvara loomise protsesse. Kanbani kasutuselevõtuga ei pea ka looma ega andma meeskonnale uusi rolle, nagu scrumi kasutuselevõtuga. Käesolevas töös tähistatakse edaspidi terminiga kanban väledat metoodikat, täpsemalt on välja toodud, kui autor peab silmas kanban tahvlit või kanbani Toyota tootmissüsteemi.

Scotland (2010) toob välja neli põhilist kanbani põhiprintsiipi:

1. keskendumine töövoo (*flow*) läbimisele,
2. töö visualiseerimine,
3. protsessis oleva töö (*Work in Progress, WIP*) limiteerimine,
4. pidev täiustamine.

Kanbani metoodikas tähendab töövoost arusaamine mõistmist, kuidas äriline või kliendiväärtus liigub läbi töövoo. Väärtusahela kaardistamine aitab saada ülevaadet, kuidas protsess on korraldatud ja mida annaks muuta, et suurendada kliendile pakutavat väärtust. Samuti tuleb üles leida protsessis leiduvad blokeerijad, etappide vahele tekkivad järjekorrad ning pudelikaelad ja neid vähendada või eemaldada. (Scotland 2010) Olulisel kohal on tõmbe süsteemi kasutamine, kus järgnevas etapis tõmmatakse tööd eelnevast etapist.

Visualiseerimine toimub kanban tahvli abil. See on vahend mõistmaks töövoogu ning saamaks ülevaadet sellest voost. Ikonen *et al (*2011: 306) rõhutavad, et kanban motiveerib visualiseerima. Kanban tahvlil on kujutatud töövoo visuaalne mudel ehk kuidas töö läbib arenduse erinevad etapid. Nagu juba alapeatükis 1.1 kirjeldatud, on kanban tahvlil

staatuse nimelised tulbad ja töö liigub ühest tulbast järgmisesse, vasakult paremale. Töö nähtavaks tegemine läbi tööülesannete liikumise eri staatuste vahel toob esile protsessis leiduvad takistused ja kitsaskohad ning võimaldab neid vähendada. Hoolimata visualiseerimise nõudest ei ole kehtestatud reegleid, kuidas kanban tahvlil töid kujutada.

Kolmas printsiip, pooleli oleva töö limiteerimine tähendab, et arvestama peab kõige suurema ajakuluga töövoo faasiga. Näiteks, kui testimine võtab kaua aega, pole arendajal mõtet teha rohkem tööd kui testija jõuab testida. Vastasel juhul koormatakse testijat liialt. Kanbani tahvli kasutamine aitab näidata pooleli olevaid töid ja aru saada, kuhu on tekkinud pudelikael. Meeskond koos juhiga (tavaliselt juhtival kohal olev isik, või keegi kes sellele meeskonnale töid annab) lepivad kokku mõistliku arv korraga pooleli olevatest töödest. Keskmist aega, mil üks tööülesanne viibib töövoos, nimetatakse läbimisajaks (*lead time*), aega, mil tööülesanne viibib on ühes staatuses, nimetatakse tsükliajaks (*cycle time*). (Stellman, Greene 2014:332)

Timmitud mõtteviisist lähtuv pidev muutumine aitab kanbani rakendaval meeskonnal keskenduda protsessist raiskamise (peatükis 1.1 defineeritud *muda*, *mura* ja *muri*) eemaldamisele. Stellman, Greene (2014:316) toovad välja, et kanban aitab meeskonnal parendada moodust, kuidas ehitada tarkvara. Meeskonnal, kes kasutab kanbani, on selge ülevaade mida neil on tarkvara loomiseks vaja teha, kuidas nad suhtlevad ülejäänud ettevõttega, kus ebaefektiivsus ja ebaühtlus tingib raiskamise ning kuidas on võimalik probleemide juurpõhjused eemaldada, et läbimis- ja tsükliaega vähendada.

Kanbani fookus erineb scrumist. Scrum keskendub peamiselt projektijuhtimisele, tähelepanu on tehtaval tööl, kuna see valmib ja kas see vastab kasutaja ja tellija vajadustele (Stellman, Greene 2014:315). Tabelis 3 on välja toodud scrumi ja kanbani metoodikate erinevused:

Tabel 3. Scrumi ja kanbani metoodikate erinevused.

Scrum	**Kanban**
Ajalahterdatud iteratsioonid (*timeboxed*) on ettekirjutatud	Ajalahterdatud iteratsioonid on vabatahtlikud. Võib olla eraldi takt planeerimisel, väljaandmisel ja protsessi parendamisel. Võib olla piiritletud sündmustega mitte ajalahtritega.
Meeskond kohustab iteratsiooni lõpuks tööga valmis saama	Töökohustus võtmine (*commitment*) on valikuline

Scrum	Kanban
Vaikimisi on planeerimise ja protsessi parendamise mõõdupuu suutlikkus (*velocity*)	Vaikimisi on planeerimise ja protsessi parendamise mõõdupuu läbimisaeg (*lead time*)
Funktsioonideülesed (c*ross-functional)* meeskonnad	Funktsioonideülesed meeskonnad on valikulised, lubatud ka spetsialistidest koosnevad meeskonnad.
Töö jaotatakse osadeks, mida on võimalik ühe sprindi ajal ära teha	Töö maht ei ole ettekirjutatud
Sprindi langustrendi graafik *(Burndown chart*) on ettekirjutatud	Diagram joonise tüüp pole ettekirjutatud
Sprindi suurus limiteerib *WIP*	*WIP* limiteeritud töövoo staatusega
Ajahinnangu andmine (*estimation*) on ettekirjutatud	Ajahinnangu andmine on valikuline
Pooleli olevale iteratsioonile ei saa tööd lisada	Uut tööd võib lisada, kui on piisavalt vaba ressurssi (*capacity*)
Sprint backlog on ainult meeskonna oma	Kanban tahvlit saab jagada erinevate inimeste ja/või meeskondade vahel
Ette on kirjutatud 3 rolli (*PO/SM/Team*)	Rollid pole ettekirjutatud
Sprintide vahel “nullitakse” scrum tahvel	Kanban tahvel on püsiv
Prioritiseeritud *product backlog*	Prioritiseerimine on valikuline
Olemas on protsessi artefaktid	Puuduvad protsessi artefaktid

Allikas: (Kniberg, Skarin 2011: 50, Nikitina *et al* 2012:140)

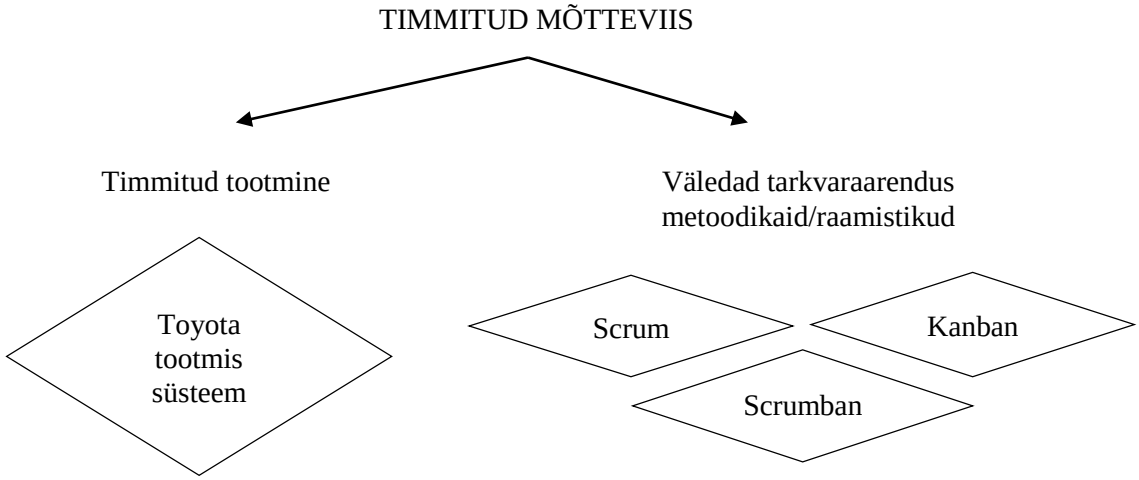
Scrum ja kanban sarnasused on (Kniberg, Skarin 2011: 49):

- mõlemad on timmitud ja väledad,
- mõlemad kasutava tõmbepõhist graafikute koostamist,
- mõlemad seavad piirangud (erineval viisil) pooleli olevale tööle (*WIP*),
- mõlemad kasutavad läbipaistvust innustamaks protsessi parendamisele,
- mõlemad keskenduvad töötava tarkvara varajasele ja sagedasele väljalaskmisele,
- mõlemad baseeruvad iseorganiseeruvale meeskonnale,
- mõlemad eeldavad töö juppideks jagamist,
- mõlemal optimeeritakse töövoogu pidevalt vastavalt empiirilistele andmetele (*velocity /lead time*).

Corey Lada 2008. aastal väljapakutud terminiga scrumban tähistati algselt üleminekut scrumilt kanbanile. Edaspidi on scrumban arenenud juhtimis raamistikuks, kus meeskonnad korraldavad tööd scrumi põhimõtteid järgides ja kanbani metoodikat kasutatakse kui läätse, mis võimaldab näha, mõista ja pidevalt parendada oma töökorraldust. (Reddy 2015:83,89) Scrumi ja kanbani sarnasused lubavad kahte metoodikat kombineerida ühendades scrumi praktikaid ja kanbani printsiipe, saades

scrumbani (Nikitina *et al* 2012:140). Reddy (2015: 40) toob välja et, scrumban ei ole ainult mõnede scrumi ja kanbani elementide kasutamine tarkvaraarendus protsessi loomiseks, vaid kanbani süsteemi kasutamine scrumi kontekstis. Lõppeesmärk on aidata ja võimendada scrumi omadusi ja pakkuda uusi omadusi ja võimalusi.

Kokkuvõttes võib öelda, et väledad metoodikad, erinevalt kosetüüpi mudelitest, järgivad samu reegleid, mis timmitud tootmine ja terviklik kvaliteedijuhtimine (Poppendieck 2001). Nikitina *et al* (2012:140) toovad välja, et kanban baseerub täppisajastatud tootmisel ja timmitud tootmisel. Joonisel 4 on esitatud autori koostatud joonis, mis seob eelnevalt käsitletud mõisted. Sellel joonisel on timmitud mõtteviis esitatud kui katusmõiste, mille alla kuuluvad timmitud tootmine ja väledad tarkvaraarendusmetoodikad.



Joonis 4. Väledate tarkvaraarendusmetoodikate/raamistike seosed timmitud mõtteviisiga.
Allikas: autori koostatud.

Womack ja Jones (1996:16-26) kirja pandud timmitud mõtteviisi viis peamist põhimõtet – väärtus, väärtusahel, voog, tõmme ja täiuslikkus – on kasutusel ka väledates tarkvaraarendusmetoodikates. Kasutajale väärtuse loomine on scrumi üks põhilisi eesmärke (Goldstein 2016). Arendatakse ainult seda, mis loob kasutajale väärtust ehk mida kasutaja vajab (Sutherland 16). Väärtusahela tundmine võimaldab aru saada, kuidas protsess on korraldatud ja mida annaks muuta, et suurendada kliendile pakutavat väärtust (Scotland 2010). Kanban tahvel on visuaalne vahend, mis annab kiire ülevaate tööst ja

aitab mõista töövoogu. Barton (2009:1) toob välja, et scrum kasutab tõmbe tehnikat ühtlustamaks läbi süsteemi kulgevat voogu ja vältimaks ülekoormamist. Täiuslikkuse poole pürgitakse läbi pideva raiskamise vähendamise, raiskamiseks on liigne toodang, hilinemine, rööprähklemine ja ületootmine.

Alapeatükis anti ülevaade tarkvaraarenduse kose ja väledast mudelist. Näidati klassikalise koskmudeli piirangutest tulenevaid probleeme ja kuidas timmitud mõtteviisist mõjutatud väleda mudeli rakendamine neid lahendada võimaldab. Alapeatüki teises pooles kirjeldati väledaid tarkvaraarendusmetoodikaid scrum ja kanban, toodi välja nende erinevused ja sarnasused ning nende kahe metoodika printsiipide koos rakendamisel tekkivat scrumban metoodikat.

1.3. Timmitud mõtteviisist lähtuvate metoodikate rakendamine ettevõtetes

Järgnevas alapeatükis antakse ülevaade timmitud mõtteviisist lähtuvate metoodikate rakendamisest tarkvaraettevõtetes. Alapeatüki teises pooles keskendutakse seejuures teaduskirjanduses leitud väledate tarkvaraarendusmetoodikate rakendamisele.

Bhasin *et al* (2006: 65) analüüsisid oma töös varasematele uuringutele toetudes, miks kukub ettevõtetes läbi Toyota tootmissüsteemi rakendamine. Leiti, et tihti unustatakse süsteemi rakendamisel ära, kui oluline on edukuse saavutamiseks ja süsteemi toimimiseks inimeste vaheline suhtlus, probleemide korrektne ja kiire lahendus, meeskonnatöö ja eestvedamine. Autorid järeldasid, et keskkond koos sinna kuuluvate inimeste ja nende kultuuriliste eripäradega on peamisteks põhjusteks, miks timmitud tootmise rakendamine sageli läbi kukub. Süsteemi rakendamisel on tähtis iseorganiseerumisvõime, fookuse nihutamine kontrollimiselt aitamiseni, hindamiselt võimustamiseni ja arengu toetamiseni ning planeerimisest kuulamiseni. Tähtis on muudatuste elluviimine ja nendest muudatustest õigeaegne kõigi organisatsiooni liikmete teavitamine, sisekommunikatsioon. Suurimad raskused tekivad ettevõtetel timmitud tootmise rakendamisel sellest, et puuduvad planeerimine, juhised ja projekti osade järjestamine, puudu ei jää aga nö tehnilistest teadmistest meetodi või rakendatavate tehnikate kohta.

Bhasin *et al* (2006: 65) võtavad kokku, et Toyota tootmissüsteemi edukaks rakendamiseks on vajalikud järgmised tingimused:

- samaaegselt tuleb rakendada viit või rohkem Toyota tootmissüsteemi tehnikat,
- oluline on nägemus timmitust kui pikaaegsest teekonnast,
- tuleb omaks võtta ja rakendada pidevat parendamist,
- võimustumise võimaldamine organisatsioonikultuuris ja timmitud põhimõtete sponsoreerimine on oluline omaks võtta terve väärtusahela ulatuses.

Conti *et al* (2006: 1025) viisid läbi põhjaliku uuringu, timmitud tootmise rakendamise ja tööstressi vahelise seose leidmiseks, kus küsitleti 1391 töötajat 21 tehasest. Tulemustest lähtub, et timmitud tootmise kasutuselevõtt organisatsioonis sunnib töötajaid kasutama uut tehnoloogiat ning looma uusi tööalaseid suhteid. Eelnev loob eeldused töö kvaliteedi ja töötajate produktiivsuse kasvuks, kuna uute tehnoloogiate kasutuselevõtust tulenevad positiivsed efektid kanduvad üle ka teistele organisatsiooni liikmetele. Uue süsteemi kasutuselevõtuga kaasnev ebakindlus võib aga tuua kaasa kõrgenenud stressitaseme, mis omakorda vähendab töötajate rahulolu ja pühendumust. Ka Hines *et al (*2004:998) rõhutavad inimfaktori tähtsust timmitud tootmise kasutuselevõtul. Nad toovad välja, et timmitud tootmise rakendamisel on oht, et tehase töötaja on sageli ülekoormatud ja pidevas pingeseisundis. Seega peab süsteemi rakendamisel pidevalt ja tähelepanelikult tegelema töötajatega ehk arvestama inimliku dimensiooniga. Töötajaid tuleb motiveerida, anda neile iseotsustamise õigust ehk rakendada võimustamist ja suhtuda töötajatest austusega.

Conti *et al* (2006: 1025) leidsid oma uurimuses, et timmitud tootmise elluviimise ettevõttes võib jagada kolme etappi.. Esimene etapp kestab esimesed kaks aastat, mil organisatsioonid valitsevad erimeelsused ning muudatuste-vastasus. Teine etapp ehk kolmas aasta on muutuste stabiliseerumise ja pikaajalise vundamendi ehitamise aasta. Viimane etapp, alates neljandast aastast muutuvad tehtud muudatused normaalsuseks, igapäevaseks. Samuti tekib uhkus timmitud tootmisest tulenevatest saavutustest, kuna saavutatud on esimesed nähtavad tulemused. Viimases etapis tõuseb ka töötajate rahulolu ning väheneb stressitase. Sellest järeldab töö autor, et läbiviidud muudatuste edu või ebaedu hindamist ei tohi teostada liiga vara. Conti *et al* tulemuste alusel peab töötama

süsteemi rakendamise nimel ning edu ootuses vähemalt kolm aastat, et näha tegelikult saavutatud tulemusi.

Timmitud mõtteviisi järgimine tähendab alati teatud spetsiifiliste tehniliste lahenduste kasutamist (nagu näiteks Toyota tootmissüsteemi tööriistad). Et nende lahenduste kasutamine oleks efektiivne ja toetaks läbiviidavat muudatust, peab samaaegselt arendama ka sotsiaalset süsteemi ehk ettevõttes töötavaid inimesi. Inimesel on väga tähtis roll probleemide lahendajana ning edasise arengu võimaldajana. Ettetulevate probleemide lahendamiseks ei saa kasutada üks-suurus-sobib-kõigile (*one-size-fits-all*) lahendust, vaid tuleb olemasolevaid tehnilisi lahendusi kasutada paindlikult probleemide lahendamiseks., Samuti eeldatakse protsesside erinevustest tuleneva varieerumisega arvestamist, inimeste poolset vastutuse võtmist muutuva keskkonna mõistmist ning keskkonnaga kohanemist. (Lander *et al*: 3684) Eelnev on väga sarnane scrumbani kasutuselevõtuga tarkvaraettevõtetes. Nagu alapeatükis 1.2. välja toodud, tähendab scrumban metoodika seda, et ettevõttes võetakse kasutusele kahe erineva väleda arendusmetoodika, scrumi ja kanbani parimad ja konkreetsele ettevõttele sobivaimad osad. Kuna iga ettevõte on unikaalne oma eriliste tööprotsesside, metoodikat rakendatavate inimeste ja klientide soovidega, on scrumban just selline paindlik tööriist, mis võimaldab kombineerida scrumi ja kanbani sobivaimaid osasid.

Layton (2012) järgi peab väledate tarkvaraarendusmetoodikate rakendamiseks looma ettevõttes sobiva töökeskkonna ja muutma inimeste käitumist. Viimane on väga raske, kuna nagu eelpool välja toodud, suurendavad muudatused inimeste stressitaset. Töökeskkond peab väledate tarkvaraarendusmetoodikate rakendamisel võimaldama töötajatel töötada üksteisele füüsiliselt lähedal asuvatel töökohtades, et võimaldada näost-näkku suhtlust. Soovitavalt võiks koostöötav meeskond asuda ülejäänud ettevõtte töötajatest eraldi asuvas tööruumis. Võimalusel tuleks meeskonnale hankida ka liigutatavad toolid ja lauad ning valge tahvel. Oluline on eemaldada meeskonna keskendumist segavad takistused, milledeks on mitme projektiga paralleelselt tegelemine, rööprähklemine, üleliigne kontroll, väliste mõjutajate ja juhtkonna otsene sekkumine tooteomaniku töösse, et mõjutada prioriteetide üle otsustamist. Tähtis on taas avastada traditsiooniline suhtlus ja töövahendid (näost-näkku, paber-pliiats), mille kasutamine on otsekohene ja võimaldab meeskonnal tunnetada, et nad saavad protsessides osaleda, neid vajadusel muuta ja olla innovaatilised. Lisaks traditsioonilistele

töövahenditele on oluline on leida ja kasutusele võtta väledaks tarkvaraarendustööks vajalikud ja organisatsioonile sobivad tehnilised tööriistad, nagu näiteks tarkvaraline lahendus kanban tahvli jaoks ja tööülesannete haldamise tarkvara.

Inimeste käitumise muutmise juures on scrumi rakendamisel olulisel kohal inimestele rollide andmine. Rollid võetakse tavaliselt kasutusele ka scrumban metoodika rakendamisel. Rollideks, mille peab inimestele määrama, on toote omanik, scrum master ja meeskonna liige. Väga tähtis on ka uute väärtuste loomine, milleks on pühendumus, keskendumine, avatus, austus ja julgus. Meeskonna formeerimisel ja selle filosoofia loomisel/muutmisel on olulised funktsioonideülesus, ise-organiseerumisvõime, iseseisvus oma töö juhtimisel, meeskonna liikmete arv ja vastutuse võtmine. (Layton 2012)

Bustard *et al* (2013: 141) uurisid väledate arendusmetoodikate kasutuselevõttu 40 tarkvaraarendusega tegelevas Põhja Iirimaa ettevõttes. Leiti, et väle tarkvaraarendusmetoodika on muutumas uueks tarkvaraarenduse standardiks traditsioonilise kosk tüüpi mudeli asemel. Tulemused näitasid, et (Bustard *et al* 2013: 141):

- 2/3 uuritud ettevõttetest kasutab igapäevaselt väledaid arendusmetoodikaid;
- nende metoodikate rakendamisel väheneb segavate faktorite arv;
- ettevõtetes tunnistatakse, et väledate arendusmetoodikate kasutuselevõtt on ettevõttele kasutoov.

Staats *et al* (2010: 380) näitavad oma töös, et väledate tarkvaraarendusmetoodikate rakendamisel luuakse ettevõtetes tavaliselt esimese asjana pilootmeeskond. Nikitina *et al* (2012:149) toovad välja, et kanbani tööpõhimõtete rakendamise positiivse tulemuse üheks põhjuseks oli protsessi pidevaks üle vaatamiseks ja parendamist vajavate olukordade lahendamiseks loodud foorum. Protsessi pidevat parendamist võimaldava mehhanismi loomine ja selleks hästi koolitatud ja pühendunud töötajad on kaks võtmekomponenti lahendamaks ettevõttes olevaid protsessist ja/või organisatsioonist endast lähtuvaid probleeme. Uurimuse autorid leidsid, et need kaks elementi võivad tuua kaasa ettevõte jaoks oodatud tulemusi isegi ilma uute metoodikate kasutuselevõtuta. Need kaks elementi on aga väledate tarkvaraarendusmetoodikate oluliseks osaks. .

Ennem scrumbani rakendamist tuleb kõigepealt tunda konteksti, millesse metoodikat sisse viima hakatakse. Reddy 2015:100 toob välja, et ennem metoodika rakendamist tuleb kindlasti läbida järgmised sammud:

- identifitseerida organisatsiooni hetkeseis ning prioriteedid (näiteks kvaliteedi tõstmine, tootlikkuse kasv, parem protsessi planeerimine);luua head ja töötavad suhted organisatsiooni juhtkonna ja meeskonnajuhtidega;
- määrata kindlaks valdkonnad, milles muudatust saavutada soovitakse;
- tutvustada scrumbani kui evolutsioonilt muutust võimaldavat raamistikku;
- alustada kaasatavate töötajate koolitamist scrumban põhimõtete ja parimate praktikate osas;
- selgitada välja, milliseid riske ja takistusi võib tuua kaasa scrumbani tutvustamine seda rakendama hakkavatele meeskondadele.

Tarkvaraarendusmetoodikaid rakendatakse ettevõtetes protsessi parendamise eesmärgil. Sellepärast on tähtsal kohal küsimus, missuguseid mõõdikuid saab kasutada protsessi parendamise mõõtmiseks. Kniberg, Skarin (2011: 87) toovad välja, et üldise ja otsekohese ülevaate kaasnevatest muutustest saab, kui analüüsida kanban tahvlilt. Analüüsimisel tuleks hinnata, kui palju tööülesandeid on „teha“ ning kui palju on „tehtud“ tulbas. Väledate arendusmetoodikate rakendamise edukust on võimalik aga ka numbriliselt mõõta kasutades selliseid näitajaid ja mõõdikuid nagu läbimisaeg (*lead time*), tsükliaeg (*cycle time*), kogusuutlikus (*total velocity*), suutlikkus töötüübi kohta (suutlikust mõõdetakse tööülesandeid jõutakse ühe iteratsiooniga ära teha) ja tööülesannete järjekorra pikkus, ning koostades sprindi langustrendi (*burndown)* graafiku. Tabelis 4 on toodud erinevate mõõdikute plussid ja miinused.

Antud töös suutlikkuse näitajaid ja sprindi langustrendi graafikut ei kasutata, sest neid kasutatakse pigem ainult scrum meeskonna töö hindamiseks. Kuna käesolevas töös analüüsitakse scrumban metoodika rakendamist, saab kasutada kanbani tahvli kasutamisega sobituvaid näitajaid nägu näiteks läbimis- ja tsükliaeg ning järjekorra pikkus.

Tabel 4. Väledate tarkvaraarendusmetoodikate mõõdikute võrdlustabel

Kasutatav mõõdik	Mõõdiku plussid	Mõõdiku miinused
Tsükliaeg	Lihtne mõõta Ei vaja ajahinnangut Algab ja lõppeb kasutajaga	Ei arvesta tööülesande mahuga
Kogusuutlikkus, agregeeritud üle kõigi tööülesannete tüüpide	Näitab kätte varieeruvuse võimaliku parendamise suuna	Ei aita ennustada konkreetsete tööde lõpptähtaegu
Suutlikus töötüübi kohta	Täpsem kui kogusuutlikkus	Võrreldes kogusuutlikkusega on raskem arvet pidada
Järjekorra pikkused	Kiire nõudmiste kõikumise näitaja Lihtne visualiseerida	Ei näita, kas järjekorra põhjuseks on ebaühtlane nõudmine või ebaühtlane tootmine Järjekorra puudumine võib näidata ületootlikust

Allikas: (Kniberg, Skarin 2011: 87)

Käesolevas alapeatükis tutvustati timmitud mõtteviisist lähtuvate metoodikate rakendamist ettevõtetes. Alapeatüki lõpus keskenduti väledate tarkvaraarendusmetoodikate rakendamise erinevatele aspektidele, seejuures mõõdikutele, mis aitavad hinnata metoodika rakendamise edukust. Töö empiirilises osas analüüsitakse ühe väleda tarkvaraarendusmetoodika, scrumbani rakendamist Eesti tarkvaraarendusettevõttes, kus valdavalt on kasutusel kose tüüpi mudel, ning tuuakse välja soovitused, kuidas scrumbani kasutuselevõtuga vähendada kliendile väljastatava tarkvara väljalaske aega.

2. VÄLEDATE TAKVARAARENDUS METOODIKATE RAKENDAMINE TARKVARAETTEVÕTTE KASIINO ÜKSUSE VÄLJALASKEOSAKONNAS

2.1. Uuringu metoodika ja tarkvaraettevõtte tutvustus

Teise peatüki eesmärgiks on anda ülevaade uurimisobjektist ja kasutatavatest uurimismetoodikatest, kirja panna uuritava ettevõtte kasiino üksuse väljalaske osakonna tootearendus- ja tootmisprotsessid, analüüsida protsessi uuenduse tulemuslikkust ning pakkuda välja timmitud tootmise ja väledate tarkvaraarendusmetoodikate teooriast tulenevaid parendusvõimalusi.

Bakalaurusetöö uurimisobjektiks on multinatsionaalse *online* kasiino tarkvara tootva korporatsiooni Tartus asuv väljalaskeosakond. Uurimismetoodikatest on töös kasututatud juhtumiuuring (*case study research*). Käesolevas töös kasutatud juhtumiuuringu kvalitatiivseteks alammetoodikateks on dokumendianalüüs, osalusvaatlus ning intervjuud. Kvantitatiivsed andmed, mida kasutatakse pärinevad andmebaasist. Yin'i järgi on juhtumiuuring empiiriline analüüs, millega uuritakse kaasaegset reaalses elus juhtuvat nähtust. Nähtuse ja teda ümbritsev kontekst on sageli eristamatud, empiirilise materjali kogumiseks kasutatakse seejuures erinevaid allikaid. Juhtumiuuring vastab küsimustele kuidas ja miks, kontroll uuritava käitumise üle on väike ning tähelepanu on kaasaegsel sündmusel. (Yin 2009:18) Käesolevas töös vaadeldakse konkreetset juhtumit, milleks on väljalaskeosakonna tarkvaraarendusprotsesside parendamine timmitud mõtteviisi ja väledate tarkvaraarendusmetoodikate abil.

Töös kasutatavaks esimeseks alametoodikaks on vaatlus, mis võimaldab uuritavaid protsesse jälgida kõrvaltvaatajana sellistena nagu nad on oma loomulikus keskkonnas, ilma nende kulgu mõjutamata. Selle uurimismeetodi kasutamist toetab fakt, et käesoleva töö teoreetilises osas kasutatud eelnevates uuringutes on kasutatud vaatlust (Ikonen *et al* 2011: 308). Vaatlust on võimalik käesoleva töö autoril teostada määratlemata arv

tööpäevi, kuna uurija töötab uuritavas ettevõttes. Samas, töökirjutaja osakonnas läbiviidavas protsessimuudatuses ei osale, kaasatud on teised meeskonnad, mis võimaldab analüüsimisel jääda objektiivseks.

Teine kasutatud alammetoodika on dokumendianalüüs, mis võimaldab võrrelda senini kasutusel olnud ja parendatud tarkvaraarendusprotsesse. Selleks on autor kasutanud ettevõte siseseks kasutuseks mõeldud dokumentatsiooni, nagu näiteks retrospektiivi koosolekute protokollid, ettevõtte töötajatele sisemiseks kasutamiseks mõeldud koolitusmaterjalid jms.

Kolmandaks kvalitatiivseks alammeetodiks on intervjuu protsessis osalevate töötajatega. Struktureeritud intervjuu võimaldab saada täpsemat nägemust protsessimuudatusega kaasnevatest probleemidest, ning valideerida kvantitatiivse analüüsiga saadud tulemusi. Samuti võimaldab intervjuu töökirjutajal dokumendianalüüsist leitud probleemide kohta lisaküsimusi esitades neid täpsemini mõista ja pakkuda lahendusi protsessi parandamiseks. Intervjuu küsimused (lisa 2) koostati pärast läbiviidud dokumendianalüüsi ja JIRAst esialgsete andmete analüüsi. Intervjuu viidi läbi kõigi protsessis osalenud töötajatega (vt. tabel 5).

Tabel 5. Intervjuude läbiviimise aeg ja kestus Ettevõtte töötajatega aprillis 2016 a.

Amet	Intervjuu toimumisaeg	Intervjuu kestus
Arendaja 1	12.04.2016	1,2h
Projektijuht 1	12.04.2016	1h
Testija 1	12.04.2016	0,5h
Projektijuht 2	13.04.2016	0,8h
Projektijuht 3	13.04.2016	1h
Testija 2	13.04.2016	1h
Arendaja 2	13.04.2016	0,5h
Arendusmeeskonnajuht	15.04.2016	1h
Testija 3	18.04.2016	Vastas emailiga

Allikas: autori koostatud.

Töös kasutatud kvantitatiivsed andmed pärinevad ettevõtte tööülesannete haldamis infosüsteemist (JIRA), kuhu kogutakse, nii automaatselt kui võimalik, andmeid iga tööülesande (*taski*) kohta.

Uuritava tarkvaraettevõtte lõid ettevõtjad kasiinoärist ning tarkvara- ja multimeediatööstusest 1999. aastal Tartus. Algsetele loojatele lisandusid turunduse ja tootega tegelejad ning 2001. aastal sõlmiti leping esimese litsentsiaadiga (*licensee*) samal

aastal lisandus lisaks kasiino tootele ka „*Live Casino*“ (*online* mängud, kus kaarte jagavad päris diilerid video otseülekande vahendusel), bingo ja pokkeri võrgustikud, mänguterminalide tarkvara ja mobiilimängud. 2006. aastal noteeriti ettevõtte aktsia Londoni börsi alternatiivnimekirjas AIM ning avati arenduskeskus Bulgaarias. Järgnevatel aastatel lansseeriti spordivõistlustele panustamist võimaldav tarkvara ning kasiino toodetega siseneti riiklikult reguleeritud turgudele (Itaalia, Hispaania, Eesti ja Serbia). 2010. aastal toimunud ühinemised tegid korporatsioonist juhtiva mitmekanaliga mängulahenduste pakkuja. 2012. aastal noteeriti korporatsioon Londoni börsi põhinimekirjas. Korporatsiooni üheks konkurentsieeliseks on kasiino, pokkeri, bingo jt olemasolevate platvormide ülene lahendus, mis pakub mängijatele sama mängukogemust kõikidel pakutavatel platvormidel läbi ühe kasutajakonto. (About us…2016)

2016. aasta seisuga töötab korporatsioonis üle Euroopa paiknevas 13 keskuses ligikaudu 5000 inimest. Eestis asuvad korporatsiooni arendus-, tootmis- ja teenindavad funktsioonid. 670 töötajaga, kellest üle 500 inimese töötab Tartus enam kui 10 osakonnas, on ettevõtte üks suuremaid tarkvaraettevõtteid Eestis. Tartus asuvateks osakondadeks on toodete ülene üksus (tarkvaraarendus, infrastruktuur, tootearendus jt), kasiino üksus, projektijuhtimise meeskond, teenuste toeosakond, graafikute meeskond infosüsteemide- ja tehnoloogia üksus, administratsioon jt. (Playtech Eestis…2016)

Kasiino üksus, mille protsesse antud töös analüüsitakse, jaguneb kaheks osakonnaks: kasiino arendusosakond (C*asino Development*) ja kasiino väljalaskeosakond (*Casino Release Management*). Uute mängude loomise ning kasiinokliendi nö „tooriku“ ja kõige sinna juurde kuuluva arendamise eest vastutab arendusosakond. Väljalaskeosakonna eesmärk on kokku panna ja mängijateni viia kliendisoove arvestav kasiino. See hõlmab suhtlust kliendi esindajatega, kontrollimist, et arendusosakonnast tulev ilma kujunduseta kasiino töötab, mängude lisamist kasiinoprogrammi, kliendi tellitud kujunduse kohaldamist (näiteks logode lisamine, kliendi soovitud värvilahendused ja eriarendused rakendamine) ja testimise tulemust. Samuti on osakonna eesmärk viia sisse arendusosakonnast tulevad versiooniuuendused olemasolevatesse kasiinodesse ning teha need mängijatele kättesaadavaks. Antud töös keskendutakse väljalaskeosakonnale, mis koosneb viiest meeskonnast: kliendihaldurite meeskond (*Client Delivery Managers*), arendusmeeskond (*Casino branding team*), lansseerijate meeskond (*Launch Team*),

projektijuhtide meeskond (*Release process managers*) ning kvaliteeditagamise meeskond (*QA Team*).

2015. aasta teises pooles otsustas analüüsitav ettevõte rakendada väleda tarkvaraarendusmetoodikat scrumban väljalaskeosakonnas. Selleks loodi väljalaskeosakonda kuuluvate funktsioonideülese meeskonnaga pilootprojekt. Töös analüüsitav periood algas pilootprojekti käivitamisega 2015. aasta oktoobris, uurimisperiood kestis kuus kuud, perioodi lõppes 2016. aasta aprillis.

Ettevõte kasutab töö planeerimiseks ning tööülesannete ohjamiseks infosüsteemi JIRA. See on enimkasutatud tarkvaraarenduse vahend, millega on lihtne planeerida ja juhtida projekte, ning jagada meeskondadele ja nende liikmetele ülesandeid, samuti on JIRA-sse sisse ehitatud scrum ja kanban metoodikate tugi (JIRA… 2016). Andmeid tööülesannete kohta on infosüsteem JIRA-st võimalik välja võtta vastavalt soovitud kuupäeva vahemikule. Väledate tarkvaraarendusmetoodikate scrum ja kanban teooriast lähtuvalt on võimalik mõõta (Kniberg, Skarin 2011: 87):

- tsükliaega (*cycle-time),*
- läbimisaeg (*lead time*) ja
- järjekorra pikkuseid.

Kvantitatiivanalüüsi läbiviimiseks kasutatakse numbrilisi andmeid, milleks on JIRA-st väljavõetavad tööülesannete staatuste keskmised ajakulud. Perioodi alguse ja perioodi lõpu keskmisi võrreldakse seejärel omavahel, samuti võrreldakse pilootprojekti meeskonna tööülesannete ajakulu ülejäänud osakonna tööülesannete keskmise ajakuluga.

Kvalitatiivanalüüs koosneb dokumendianalüüsist, mis viiakse läbi autorile kättesaadavatest pilootprojekti meeskonna koostatud retrospektiivi koosoleku protokollidest, vaatlusele lisandub ka pilootprojekti liikmetega läbi viidud struktureeritud intervjuu. Nende kolme meetodi kombineerimine töösoorituse mõõtmisetulemustega võimaldab teha kindlaks, kas protsessi muudatus parandas töösooritust ning analüüsida, kas protsessis esineb veel raiskamist. Viimasel juhul saab pakkuda välja protsesside parandamisvõimalusi. Järgmises alapeatükis analüüsitakse kasiino üksuse väljalaskeosakonna protsesse ennem ja pärast väledate tarkvaraarendusmetoodikate kasutuselevõttu.

2.2. Kasiino üksuse väljalaskeosakonna tarkvaraarendusprotsesside analüüs

Käesoleva alapeatüki eesmärgiks on anda ülevaade kasiino üksuse väljalaskeosakonnast ja seal kasutusel olevast kose tüüpi tarkvaraarendusprotsessist.

Kasiinodes mängivad lõppkasutajad on litsentsiaatide kliendid, litsentsiaadid (*licensee*) ehk kasiino operaatorid on aga analüüsitava tarkavaraettevõtte kliendid, seega on tegemist *business-to-business (B2B*) ärimudeliga. Korporatsioonil on 2016 aasta kevade seisuga üle 120 kliendi. Litsentsiaatide huve esindavad ettevõttes kliendihaldurid (*Account Manager*). Litsentsiaadi kliendihaldur on ettevõtte töötaja, kes esindab kliendi ärihuve ning ärihuvidest lähtuvalt juhib kõikide erinevate toodetega (kasiino, pokker, bingo jt) seotud muudatusi. Kasiino osakonna toode on kliendi jaoks kohandatud *online* kasiino.

Väljalaskeosakonna, mis on kasiino osakonna üks kahest osakonnast ning milles toimuvaid muudatusi antud töös analüüsitakse, moodustavad meeskonnad on toodud tabelis 6. Tegemist on funktsionaalsete meeskondadega, kus igal meeskonnal on etteantud roll ja meeskonda juhib meeskonnajuht.

Tabel 6. Ülevaade väljalaskeosakonna meeskondadest

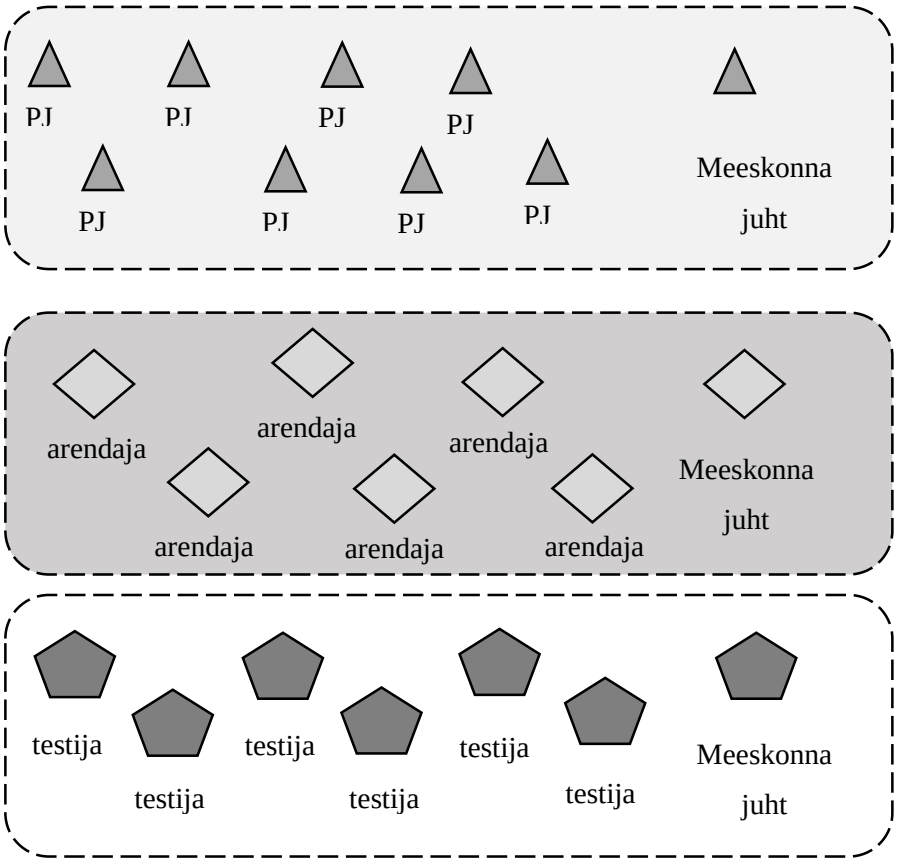
Meeskonna ingliskeelne nimetus	**Meeskonna eestikeelne nimetus**	**Meeskonnaliikme roll**	**Meeskonna töö lühike kirjeldus**
Client Delivery Managers	Kliendimuudatuste projektijuhtide meeskond	Kliendimuudatuste projektijuht	Kliendi esindajatega suhtlemine, töö planeerimine
Casino branding team	Kasiino kujunduse lisamise meeskond	Arendaja (*layout developer*)	Lisada kliendi tellitud graafika ja kliendi spetsiifilised arendused
Launch Team	Lansseerimis meeskond	Lansseerimisinsener	Tagada toote võimalikult sujuv ja kiire kliendini toimetamine, töövahendite arendamine
Release process managers	Väljalaske protsesside projektijuhid	Väljalaske protsesside projektijuht	Suhtlus kasiino arendusosakkonnaga
QA Team	Kvaliteeditagamise meeskond	Kvaliteedi tagamise spetsialist ehk testija	Kasiino tarkvara testimine

Allikas: (Organization 2016) autori koostatud.

Väljalaskeosakonna sisenditeks on kasiino arendusosakonnast tulevad uued kasiino versioonid (*builds*) ning kliendi poolt tulevad nõuded. Osakonna väljundiks on uus töötav kasiino või olemasoleva kasiino uus versioon. Kasiino arendusosakonnast tulevate uute versioonide sujuvama ja kvaliteetse väljalaskmise koordineerimise eest vastutavad kaks väljalaske protsesside projektijuhti. Käesoleva töö analüüsist jääb nende roll välja, sest tehtud muudatus neid ei puuduta. Samuti jääb käesoleva töö analüüsist välja lansseerimismeeskond, kuna selle meeskonna töö on väljalaskeosakonna töö automatiseerimine. Töö kirjutamise ajaks on kasiinode kokkupaneku automatiseerimine saavutatud piisavas ulatuses, nii et ülejäänud osakonna töötajad saavad iseseisvalt uusi kasiino versioone väljastada, ilma et oleks vaja lansseerimis inseneri poolset sekkumist.

Litsentsiaadid, töös edaspidi kliendid, on jaotatud tähtsuse järgi kategooriatesse. Igal litsentsiaadil on mitmeid brände, igal brändil on oma kindla kujundusega kasiino. Madalama kategooria brändide kasiinosid uuendatakse paar korda aastas, kõrgeima astme brändide kasiinosid uuendatakse mitu korda kuus. Samuti on klientidele lubatud erilahendused, nagu näiteks brändi graafikaga kasiino fuajee, spetsiaalselt brändi jaoks arendatud mängud jms. Madalama kategooria brändid saavad standardse kasiino, mille ainsateks võimalikeks muudatusteks on logo ja värvid.

Käesolevas töös uuritav protsessimuudatus mõjutab kasiino väljalaskeosakonna kolme meeskonna tööd – kliendimuudatuste projektijuhtide meeskonda, arendajate meeskonda ja kvaliteeditagamise meeskonda. Joonisel 5 on kujutatud neid meeskondi funktsionaalsetena – ülemine punktiirjoontega markeeritud kast on kliendimuudatuste projektijuhtide meeskond (joonisel lühend *PJ*), keskmises kastis on kujutatud arendajate meeskond (joonisel lühend *arendaja*) ja alumises kastis tarkvara kvaliteeditagamise meeskonda ning nende meeskondade töötajaid (joonisel lühend *testija*).



Joonis 5. Funktsionaalsed meeskonnad kasiino väljalaskeosakonnas
Allikas: (Organization 2016), autori koostatud.

Ennem muudatusi olid kõik töötajad oskuste järgi grupeeritud funktsioonide põhiselt eri meeskondadesse ja töö liikus ühest meeskonnast järgmisesse. Igal meeskonna oli täita oma kindel funktsioon. Kliendimuudatuste projektijuhtide meeskonna ülesanne oli suhelda litsentsiaadi kliendihalduritega või vajadusel otse kliendi esindajatega, saadud info põhjal teha valmis tööülesanne ning planeerida ülesande täitmine. Arendusmeeskonna roll oli viia sisse ülesandega tellitud muudatused ning kvaliteeditagamise meeskonna eesmärk oli kindlaks teha, kas soovitud muudatused vastavad kliendi nõudmistele ning veenduda toote kvaliteedis (Workflow & process description 2016).

Tööülesande elukäik on kaardistatud ja seejärel defineeritud JIRA-s, kirja on pandud kõik võimalikud vahepealsed tööülesande staatused (olekud), mis võimaldab leida üles etapid, kus tööülesanne viibib ning mis etapid on vaja läbida, et ülesanne saaks täidetud. Igal meeskonnaliikmel on oma roll, mis vastab teatud tööülesande JIRA staatusele (Workflow

& process description 2016). Tööülesande staatused koos eestikeelsete vastetega on esitatud tabelis 7. Tabeli viimases veerus on toodud meeskonna liige, kelle käes tööülesanne on. Töö liigub tabelis ülevalt alla. Igale staatusele saab eelneda eelmine ning staatust vahele jätta ei saa.

Tabel 7. Tööülesannete staatused JIRA-s

Tööülesande staatus ingliskeeles	**Tööülesande staatuse eestikeelne vaste**	**Staatuse muutja roll**
To Do	Teha	PJ
Ready for Development	Arenduseks valmis	PJ
In development	Arenduses	arendaja
In deployment	Paigalduses	arendaja
Ready for Testing	Testimiseks valmis	arendaja
In testing	Testimises	testija
Sanity Failed	Test ebaõnnestunud	testija
On hold	Ootel	testija
Ready for Preview	Valmis üleandmiseks	PJ

Allikas: (Workflow & process description 2016), autori koostatud

Projektijuhtide meeskonna sisendiks on litsentsiaadilt tulenevad nõuded, mis vormistatakse JIRA-s tööülesanneteks. Igal litsentsiaadil on kindlaks määratud projektijuht, iga projektijuht haldab mitut klienti. Tööülesande formuleerimise faasis suhtleb projektijuht tihedalt kliendiga, et tööülesanne peaks olema piisava detailsusastmega järgmistes faasides tööülesannet täitvate spetsialistide jaoks. Tööülesanne saab külge staatuse „*teha*“ ning lisatakse ka kuupäev, millal ülesanne võiks olla täidetud ja kliendile üle antud. Selleks on kasutusel väljalaske kalender, mis arvestab eelnevalt kogutud info põhjal, kui pikk on ühe tööülesande tsükliaeg ehk kui kaua ülesanne viibib väljalaskeosakonnas ennem kui toote saab anda üle kliendile (Workflow & process description 2016). Väljalaske kalender arvestab nii töötajal töö tegemiseks kuluvat aega kui ka arvuti tööaega. Kasiino programmi kokkupanek (*compile*) ja serverisse üleslaadimine (*upload*) on automatiseeritud tegevused, mida teevad masinad.

Arendusmeeskonna liikmed võtavad „*arendustööks valmis“* nimekirjast tööd sisse tulemise järjekorras töösse. Töö saab JIRA-s staatuse „*arenduses*“. Kui arendaja oma töölõigu lõpetab, liigutatakse tööülesanne staatusesse „*paigalduses*“, kus kasiino kompileeritakse ja laetakse üles serverisse nii, et kliendil on võimalik näha muudatusi ja teha need mängijatele kättesaadavaks. Kui staatus „*paigalduses*“ on edukas, muutub tööülesande staatus „*testimiseks valmis*“. Kui arenduse käigus tekib tõrge, näiteks polnud

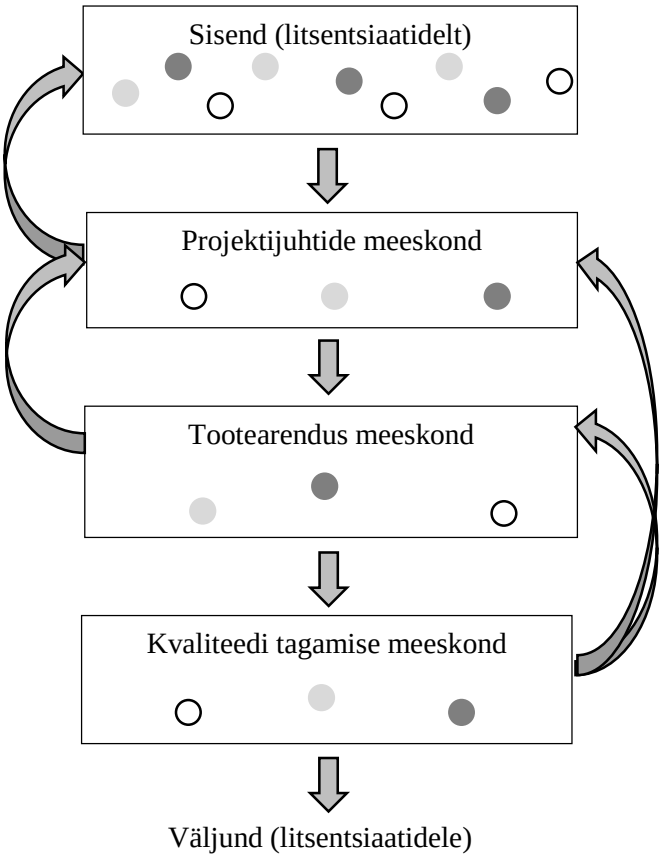
võimalik kliendi täpset soovi täita, märgitakse see JIRA-s ära ja tööülesanne jääb kas pausile või liigub tagasi projektijuhile, kes peab saama kliendilt lisainfot või otsustama, mida edasi teha. (Workflow & process description 2016)

Kvaliteeditagamisemeeskonna liikmed võtavad "testimiseks valmis" nimekirjast tööd sisse tulemise järjekorras töösse. Kui kvaliteeditagamisemeeskonna liige lõpetab töö ülesandega, märgib ta staatuseks „*testimine lõpetatud"*. Kui testimisel ilmnesid vead või tekkisid lisaküsimused, pöördub ta vastuse saamiseks kas arendaja või projektijuhi poole. Niikaua on tööülesande staatus JIRA-s kas „*ootel*" või „*test ebaõnnestunud".* Kui testimisel vigu ei leitud, saab tööülesanne staatuse „*Valmis üleandmiseks*". (Workflow & process description 2016)

Kuna iga väljalaskeosakonna meeskonna ressurss on piiratud, ei jätku kõikidele projektidele piisavalt tööaega ja seega peab töid jagama. Iga meeskonna sees toimub ressursside jagamine ja tööülesannete meeskonnasisene prioritiseerimine. Alati võetakse JIRAst ülesandeid sisse tulemise järjekorras, kuid igapäevaselt on projekte või kliente millel on kõrgem prioriteet või lühem tähtaeg. See teadmine on olemas aga ainult projektijuhil, kes suhtleb kliendiga, mitte funktsionaalse meeskonna liikmetel või selle meeskonna juhil. Eelnevast johtuvalt tekib ressursside jagamisel prioritiseerimise konflikt.

Tööülesande liikumine ühest funktsionaalsest meeskonnast järgmisesse on toodud joonisel 6. Tegemist on teoreetilises osas käsitletud kosk-tüüpi tööülesande liikumisega ühest protsessi faasist teise. Erinevatele klientidele tehtavad tööülesanded on tähistatud eri toonides ringidega. Erinevate toonide kasutamine aitab näitlikustada, et iga tööülesanne konkureerib teiste ülesannetega ja iga projektijuht konkureerib teiste projektijuhtidega meeskondades olevate töötajate ressursi pärast. Tekib huvide konflikt, kus projektijuhid võistlevad töötaja ressursi pärast, kuna soovivad, et nende kliendi töö võetaks esimesena ette, kuigi vastav tööülesanne tuli meeskonna tööde järjekorda hiljem. Antud olukorras võidab sageli see projektijuht, kes agressiivsemalt oma tööülesande järjekorras ettepoole tõstmise eest seisab, viimane sõna jääb osakonnajuhatajale. Funktsionaalse meeskonna juht peab oma meeskonna liikmete ressurssi planeerima. Selle ressursi kättesaadavust mõjutavad töötajate puhkused, haigused, koolis käimine ja teised põhjused.

Tegemist on kose tüüpi protsessiga, kus allapoole suunatud nooltega on joonisel tähistatud tööülesande liikumine eelnevast meeskonnast järgenvasse. Ülespoole suunatud nooled tähistavad olukorda, kus on leitud viga või ei saa tööd edasi teha ning tööülesanne liigutatakse tagasi eelmisesse või veel varasemasse faasi. Siinkohal on oluline rõhutada, et reeglina see tööülesanne, mida liigutatakse varasemasse etappi lisatakse tehtavate tööde nimekirja ning kui vastav tööülesanne on lubatud valmis saada kindlaks kuupäevaks, peab projektijuht ülesande täitmiseks ressursi leidma. Selline asjade käik on osakonnas igapäevane konfliktide allikas.



Joonis 6.Töö liikumine funktsionaalse struktuuriga meeskondade vahel (autori koostatud)
Allikas: (Asian agile team info), autori koostatud.

Tööülesande liikumine funktsionaalsete meeskondade vahel tekitab võimaliku pudelikaela, kus tööd võivad kuhjuda ühe meeskonna kätte ehk jääda ühte staatusesse pidama. Põhjusteks võib olla ebapiisav kommunikatsioon meeskonna sees ja meeskondade vahel või ebaühtlaselt jaotunud töökoormus (*mura*), mis tekitab omakorda *muri* ehk inimeste ülekoormamist. Kokkuvõtvalt võib öelda, et töötaja ressurssi planeerib

nii projektijuht, kes seisab oma klientide soovide eest, kui meeskonnajuht, kes seisab selle eest, et tööülesanded saaksid täidetud.

Eelnevalt kirjeldatud probleemid ja konfliktid vajasid lahendamist. Leiti, et olemasolevaid probleeme aitab lahendada väledate tarkvaraarendusmetoodikate kasutuselevõtt. Protsessi muudatuse elluviimiseks loodi osakonnajuhataja initsiatiivil pilootmeeskond, mille koodnimeks sai „Aasia meeskond“. Kuna Aasia turul tegutsevad litsentsiaadid on ettevõtte jaoks väga olulised, otsustati alustada just nende klientide nõudmiste teenindamise paremaks muutmisest.

Väljalaskeosakonnas läbiviidava protsessimuudatuse elluviimiseks kasutati koolitusfirma Ciklum väledatele arendusmetoodikatele spetsialiseerunud koolitajaid. Pilootmeeskonna liikmetega viidi läbi väledate tarkvaraarendusmetoodikate scrum ja kanban koolitused. Koolituse lõpuks tundsid pilootmeeskonna liikmed nii scrumi kui kanbani tööpõhimõtteid ja omasid ülevaadet võimalustest töökorraldust paremaks teha. Õpiprotsessis õppisid ka koolitajad, viisid ennast kurssi ettevõtte protsessidega selleks, et soovitada sobivaimaid metoodikaid ja nende komponente. Koolitajad koos pilootmeeskonnaga analüüsisid väljalaskeosakonna tööd ja protsesse ning leidsid eesmärgi täitmiseks kõige paremini sobivad arendusmetoodika komponendid, parimad nii scrumist kui kanbanist.

Scrum metoodikast võeti kasutusele (Casino Release Workshop 2016):

- rollid: scrum meister, toote omanik (*PO – product owner*);
- funktsioonideülene meeskond;
- tseremooniad, päevane scrum, tööjärjes oleva selgindamine (*backlog grooming*), retrospektiivkoosolek.

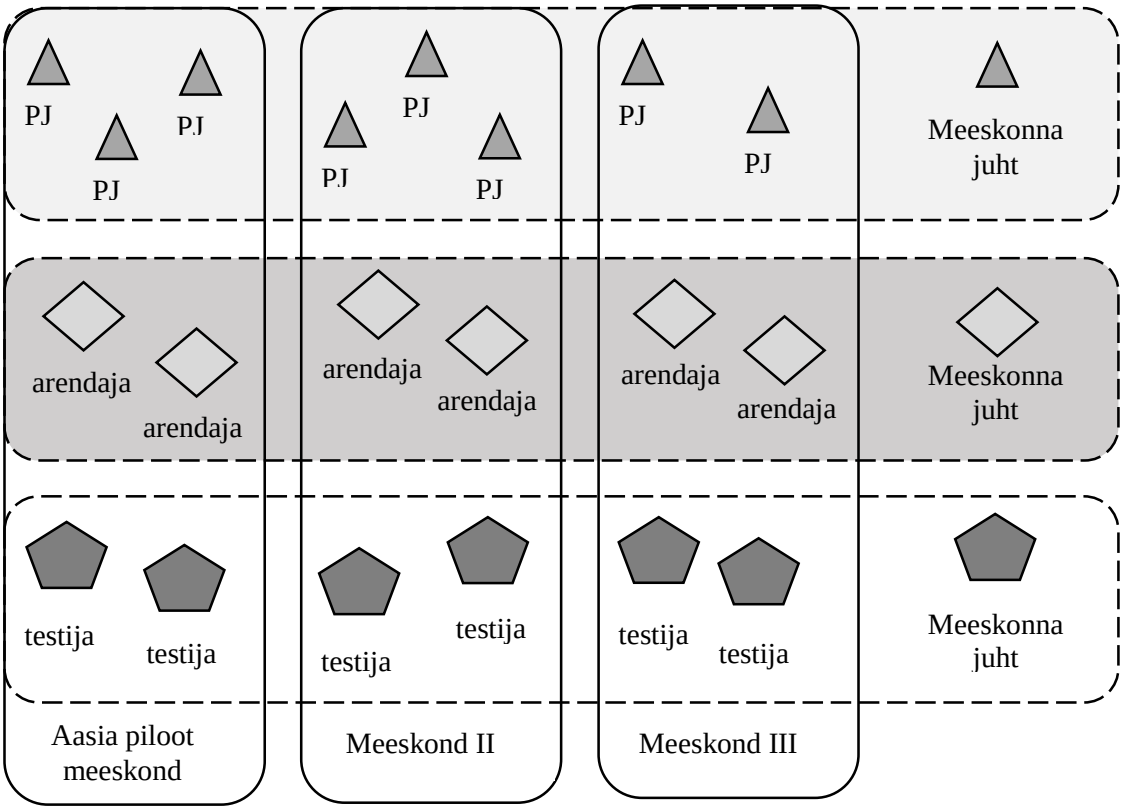
Kanban metoodikast võeti kasutusele (Casino Release Workshop 2016):

- Kanban tahvel (*board*);
- Tööülesanne tõmmatakse eelmisest etapist *vs* tõugatakse järgmisse etappi (*pull vs push*);
- Pooleli oleva töö hulk (*WIP Limit*);
- Läbimisaja (*lead time)* parandamine.

Koolitusel pandi paika ka väledates arendusmetoodikates kasutatavad igapäevaste ja retrospektiivkoosolekute pidamise kord: kuna toimub, kus toimub, kes osalevad, kaua

kestab ning et koosolekutel kasutatakse kanban tahvlit (JIRA-s olev virtuaalne tahvel, mis kuvatakse koosolekuruumis ekraanile).

Meeskonna töö ümberkorraldamine on aega ja pühendumust nõudev protsess, mistõttu on sellele muudatusele ettevalmistamist põhjalikult analüüsitud. Timmitud tootmise teooriast pärit rakuline tootmine on väljendunud scrum metoodikas funktsioonide ülese meeskonna loomises. Selleks on masstootmisest tuntud funktsioonide põhjal loodud meeskonnad ümberjaotatud funktsioonideülestesse meeskondadesse. Meeskond peab sisaldama liikmeid kõikidest vajalikest rollidest nii, et tööülesanne täidetakse ühe funktsioonide ülese meeskonna sees ilma tööülesande liikumiseta ühest meeskonnast teise. Joonisel 7 on toodud üldine skeem, kuidas väljalaskeosakonnas moodustatakse funktsioonideülesed meeskonnad. Funktsiooniüleste meeskondade moodustumisega suurenes projektijuhtide panus kasiino testimisse ja lihtsamate arendustööde endale selgeks tegemine, suurenes kasiino testijate osalemine arendustegevuses ning kasvas arendajate võimekus vajadusel kasiinosid testida.



Joonis 7. Funktsioonideülesed võimalikud meeskonnad kasiino väljalaskeosakonnas
Allikas: (Casino Release Workshop 2016), autori koostatud.

Väljalaskeosakonnas moodustatakse funktsioonideülesed meeskonnad nii, et sarnaste omadustega klientide jaoks tehtavad tööd oleks ühe meeskonna käes. Aasia meeskond koosneb kolmest projektijuhist, kahest arendajast ja kolmest testijast. Scrum rollid jaotuvad järgmiselt: kliendimuudatuste projektijuht omab tooteomaniku rolli (*Product Owner*), ühel testijatest on *Scrum Masteri* roll, ülejäänud arendajad ja testijad moodustavad scrum meeskonna. (Agile Training…2015)

Vastavalt teoreetilises osas välja toodud väledate tarkvaraarendusmetoodika põhimõtetele toetudes peaks olukord pärast protsessimuudatust paranema seetõttu et:

- kuna projektijuht on tooteomanikuna kliendi esindajana meeskonnas, jõuavad kliendi soovid meeskonnani kiiremini ja täpsemini;
- suhtlus töötajate vahel on vahetum;
- töö on efektiivsemalt koordineeritud;
- keskendutakse ühisele eesmärgile, mis loob ühtse meeskonna tunde;
- kuna kõik meeskonna liikmed osalevad otsustuse juures, suureneb motivatsioon tööülesannet kiiremini ja kvaliteetsemalt täita.

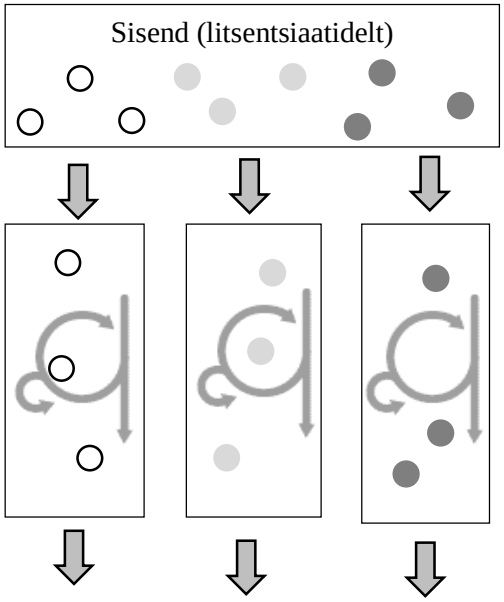
Ciklumi koolitusmaterjalist lisanduvad nimekirja veel järgmised positiivsed tagajärjed (Agile Training…2015):

- arendajal tekib arusaamine kliendi soovidest, mis võimaldab väiksema ajaga soovitud muudatused kasiino tarkvarasse viia;
- testijal tekib brändipõhine teadmine (*know-how*), mis lühendab testimisele kuluvat aega;
- kuupäevadest kinnipidamine (*commitment*) paraneb läbi töö parema planeerimise.

Joonisel 8 on toodud tööülesannete liikumine funktsiooniülestes meeskondades. Tekkinud väärtusahelad võimaldavad vähenda tööaja raiskamist. Litsentsiaate tähistatakse joonisel eri toonides ringidega. Sellisel tööjaotusel on ühe litsentsiaadi brändidega seotud tööülesanded ainult ühe meeskonna käes. Kuna spetsialistid, eriti testijad, ei pea enam teadma (või otsima sise veebist abimaterjalidest) kõigi litsentsiaatide ja nende brändide spetsifikatsioone, vähendab keskendumine ainult teatud brändidele tööajakulu. Oluline tööülesande täitmise kuluva aja vähenemine saadakse sellest, et projektijuht ei konkureeri enam ülejäänud projekti- ning meeskonnajuhtidega teatud tööülesande täitmiseks kuluva tööjõu ressursi pärast. Eelnev vähendab tekkida võivaid

pingeid ning aitab planeerida tööülesandeid, kuna on selge ülevaade töökoormusest ja töötaja ressursi saadavusest.

Tööülesanne ei liigu funktsioonide ülese meeskonna puhul enam funktsionaalsete meeskondade vahel, vaid on nii kaua funktsioonide ülese meeskonna käes, kuni tööülesanne on täidetud. Kui töö käigus tekib tõrkeid või lisaküsimusi, läbib tööülesanne vajaliku hulga iteratsioone meeskonna sees mitte meeskondade vahel. Näiteks, kui testija leiab tööülesande täitmisel puudujääke, ei pea tööülesanne minema tagasi arendusmeeskonna tööde nimekirja ning ootama, kuni tekib arendajal vaba aeg, vaid testija saab arendajale otse kommunikeerida, mis puudujäägid leiti, või suhelda projektijuhiga, et saada vajaminevat infot ja test edukalt lõpetada. Iteratsioonid on kujutatud joonisel 8 ringikujuliste joontena.



Joonis 8. Tööülesannete liikumine funktsiooniülestes meeskondades
Allikas: (Agile Training), autori koostatud.

Alampeatükis kirjeldati väljalaskeosakonnas kasutusel olevat kose tüüpi protsessimudelit. Samuti toodi välja väledast tarkvaraarendusmetoodikast lähtuvad osakonnas pilootmeeskonnaga läbiviidud protsessimuudatused. Järgmises peatükis analüüsitakse, kas tarkvaraarendusprotsesside muudatusega õnnestus vähendada

inimressursi raiskamist väljalaskeosakonna meeskondade töös ja vältida prioriteetide konflikti.

2.3. Kasiino üksuse väljalaskeosakonna tarkvaraarendusprotsesside parandamisvõimalused väledate tarkvaraarendusmetoodikate põhimõtetest lähtuvalt

Alapeatükk sisaldab väljalaskeosakonna töö ümberkorraldamise analüüsi scrumban põhimõtetest lähtuvalt. Alapeatüki lõpus tehakse ettevõttele analüüsi põhjal välja töötatud muudatusettepanekuid. Autorile pole teada, et vaadeldaval perioodil oleks toimunud osakonnas teisi muudatusi, mis oleks võinud mõjutada Aasia pilootmeeskonna tööd ja sellega seoses ka teostatud analüüsi ja saadud tulemusi.

Ciklum koolitajad koos ettevõtte pilootmeeskonna liikmetega panid koolituse algusel kirja protsessimuudatusest tulenevad võimalikud positiivsed muutused, aga ka potentsiaalsed riskid, murekohad ja puudused. Põhiline muudatus, mida sooviti saavutada oli läbimisaja vähenemine. Positiivsete tulemustena nähti veel, et (Agile Training…2015):

- meeskonna liikmed tunnevad põhjalikumalt oma klienti,
- meeskonnasisese kommunikatsioon paraneb,
- testimisele kuluv aeg kindlatele brändidele väheneb,
- suureneb lubadustest kinnipidamine (*commitment*).

Esimesel retrospektiivkoosolekul, mis toimus koos koolitajatega pärast poole kuu möödumist muudatuste rakendamisest, lisati eelmainitud tulemustele ka (Asian agile team info):

- parem ülevaade oma meeskonna tööst,
- parem teadmiste jagamine meeskonna sees,
- meeskonna kõrge motivatsioon teha oma tööd.

Ennem muudatuste rakendamist leiti koos koolitajatega ka eeldatavad ohukohad, milleks olid (Agile Training…2015):

- arendusest tulevate uute kasiino versioonide töösse rakendamine (ressursi konflikt, meeskonna liikmed peavad tegelema mitte Aasia brändidega seotud tööga);
- meeskonna tööaja jagamine meeskonnavälistele tööülesannete täitmiseks? (kui Aasia meeskonnast väljaspool on tööjõudu puudu, rakendatakse Aasia meeskonna töötajaid);
- arendajate vähesed teadmised toote kokkupanekust (arendajad peavad end nende jaoks uue toote kokkupanekuga kurssi viima);
- raskused istumisplaani muutmisel;
- osakonnasisesed projektid (tööaeg kulub mujale kui pilootmeeskonna tööle);
- puhkused, haigused, kool jt töölt puudumise põhjused, mis takistab tööülesande sujuvat liikumist;
- projektijuhtide omavaheline töödele prioriteetide seadmine (iseorganiseeruv vs meeskonnajuhi põhine, kuna meeskonnas on kolm projektijuhti, peavad nad omavahel kokku leppima prioriteetides);
- töökoormus vs inimesed (kuna tegeldakse ainult kindlate brändidega, ei jää aega tegeleda karjääri soodustavate ja motivatsiooni tõstvate uute projektidega).

Esimesel retrospektiivkoosolekul lisandusid ohukohtadena veel (Asian agile team info):

- töö planeerimisega tekkida võivad probleemid,
- võimalikud probleemid arendajate/testijate tööjõu ressursiga,
- hirm liigsete koosolekute ees,
- stressi põhjustav mõttemaailma muutus.

Aasia pilootmeeskond on töö kirjutamise hetkeks töötanud kuus kuud. Selle ajaga on jõutud läbi viia seitse põhjalikumat retrospektiivkoosolekut. Igal koosolekul panid kohalolnud meeskonnaliikmed kirja, mis neile eelmisest perioodist positiivsena meelde jäi ja mis mõjus negatiivsena ning vajaks seetõttu parandamist. Autor liigitas koosolekute protokollides väljatoodud probleemid ja hästi läinud asjad üldisteks teemadeks ning jaotas need teemad omakorda läbimisaega suurendavateks või vähendavateks teguriteks. Tulemused on toodud Tabelis 8.

Tabel 8. Läbimisaega mõjutavad tegurid

Läbimisaega parandavad ⇧	Uute funktsioonideüleste oskuste õppimine
	Meeskonna kõrge moraal
	Omavaheline otsene ja efektiivne suhtlus
	Kiirem info jagamine
	Parem töö planeerimine
	Uute inimeste meeskonda lisamine (tööjõu hulga suurendamine, *HR*)
Läbimisaega suurendavad ⇩	Meeskonna liikmete ajakulu meeskonnavälise töö tegemiseks
	Ebapiisav tööjõu hulk
	Raskused õppimise jaoks aja leidmisega
	Üldine kehva kommunikatsioon ettevõttes
	Raskused seoses mõtlemise muutumisega
	Mittekvaliteetne sisendinfo
	Tehnilised probleemid, mille lahendused on meeskonnast väljas

Allikas: (Asian agile team info 2016) Autori koostatud.

Scrumban arendusmetoodika kasutuselevõtt parandab oluliselt senist töökorraldust, mis omakorda suurendab läbi õppimise ja meeskonna moraali tõusu ka töötajate võimekust. Uues funktsioonideüleses meeskonnas olemisega kaasneb oma tavapärasele rollile õppimine. **Õppimist** on mainitud retrospektiivkoosoleku protokollides positiivse tulemusena seitsmel korral. Uute rollidega kaasnevad funktsioonideülese õppimisega seotud näited: „*Projektijuhid osalevad ise kasiinotarkvara testimises*“, „*Projektijuhid on funktsioonideülesed*“, „*Projektijuhid hakkasid õppima GMM*[5]*i*“. Intervjuudes juhiti tähelepanu tõigale, et töö parem planeerimine ja litsentsiaatide nõudmiste/soovidega arvestamine on tuntavam lisakoormus. Seda võib lugeda õppimisprotsessi osaks.

Eranditult kõik intervjueeritavad mainisid meeskonna sisekliima paranemist. Töötajate võimekust suurendab ka kõrge **meeskonna kõrge moraal,** juba teisel retrospektiivkoosolekul toodi välja, et pilootmeeskonna moraal on olnud algusest peale hea. Kõrget moraali toodi peaaegu kogu uuritava perioodi retrospektiivkoosolekute protokollides esile kuuel korral. Näiteks „*Hea meeskonnaliikmete vaheline keemia*“, „*Sujuv meeskonnatöö*“, „*Meil valitseb hea meeskonna sünergia*“. Ka intervjuud toetavad kõrge moraali olemasolu meeskonnas.

„*…tunda on ühisosa ja isiklikku panust, arvan, et see on seotud eneseteostuse võimaluse tunnetamisega ja isiklikul tasandil 'ma olen tähtis ja vajalik' tundmisega, mis säärases korporatiivvõtmes kipub tihti ära kaduma“*

[5] GMM on ettevõtte poolt loodud programmeerimiskeel kasiinode kokkupanekuks

Arendaja 1

„*kõik teavad, mis on kellegi roll abistavad ja toetavad teineteist*“.

Projektijuht 1

Töökorralduse parendamise alla kuuluvad ka **suhtluse paranemine** ja **kiirem info jagamine**. Neid tegureid on mainitud positiivsena vaadeldava perioodi retrospektiivkoosolekutel seitsmel korral. Sama trend on leitav ka intervjueeritavate vastustes.

„*Kõik osapooled on probleemidesse kaasatud ja nendel teemadel arutlemine mõjub motiveerivalt, puudutab otseselt inimesi“*.

Arendaja 1

„Väiksel sega tiimil (ametite mõttes) on palju eeliseid – soojem sisekliima, parem planeerida ja sujuvam/külluslikum infovahetus“

Projektijuht 2

„Info liigub kiiremini ning ei pea erinevaid inimesi taga ajama küsimustega, kuna päevaplaanid pannakse paika hommikustel koosolekutel “.

Testija 2

Eelpool toodud läbimisaega parandavad tegurid saab jagada kolme suurde gruppi. Läbimisaega on võimalik vähendada suurendades töötajate võimekust, parendades töökorraldust või suurendades töötajate hulka.

Kuna läbi on viidud töötajate jaoks keerukas tööprotsessi muudatus, on **töö planeerimisega** seotud probleeme retrospektiivkoosolekutel välja toodud kõige rohkem, kaheksal korral. Näiteks: „*Kasiino arendusosakonnast tulevate uute versioonide tööülesanded ei ole kanbani tahvlil nähtavad*“, „*'Valmis üleandmiseks' ja 'Ootel' on eristamatud*“, „*Liiga palju rööprähklemist tööülesannete vahel*“, „*Liiga palju ülesandeid korraga kanban tahvlil“*. Viimased kaks probleemi lahendati jooksvalt kasutades kanban metoodikas saadaolevat võimalust piirata pooleli oleva töö hulka (*WIP Limit*). Selle rakendamisel määratakse JIRA-s ära igas tulbas maksimaalne tööülesannete arv. Kui see arv ületatakse, visualiseerib süsteem maksimaalset arvu ületavad ülesanded punasena, mis annab koheselt probleemist märku.

Tööjõu ressursi puudust või sellest johtuvat **tööaja planeerimise** keerukust mainiti negatiivse poole pealt viiel korral. Enamustel kordadel tööressurssi ebapiisavalt, kui töötaja peab olema koolis, puhkab või on haigestunud. Sellisel juhul on keeruline leida asendajat, kuna sama töö tegijaid on vähe. Uute rollide õppimine küll leevendab antud olukorda, kuid näiteks suures funktsionaalses meeskonnas olevat ressursi varu see ei asenda.

Probleemid tööaja planeerimise ja suhtlusega, mis on seotud **meeskonnast väljapoole** suunatud suhtlusega**,** on välja toodud kuuel korral, peamiselt seoses uute versioonide (*build task)* ning litsentsiaatidelt tuleva ebapiisava infoga nõudmistega. Suhtlusprobleemi kliendi ehk litsentsiaadiga tõi intervjuul välja ka näiteks Arendaja 2.

„*…kõik on kinni siis tellija ja tellimuse täitja vahelise ja osapoolte kommunikatsiooni kvaliteedis*“.

Arendaja 2

Uute rollide **õppimise** keerukust tuuakse välja neljal korral, näiteks: „*Liigselt uusi asju, vaja on rohkem aega õppimiseks*“, „*Pole aega õppimaks uusi funktsioone*“. **Mittekvaliteetse sisendinfo** all peetakse silmas seda, et rohkem infot peaks panema sise veebi, projektijuhtidelt peaks tulema täpsem töökirjeldus ning osakonnavälised osapooled võiksid oma tööd detailsemalt kirja panna. **Tehnilised probleemid** on seotud mittestandardsete (häkk) lahenduste kasutamisega ning automatiseeritud tööde masinressursi vähesusega.

Eelnevalt väljatoodud läbimisaega pikendavad tegurid on võimalik koondada nelja suurde gruppi. Töö tegemiseks kuluvat aega suurendavad: töötajate ülekoormatus, halb töökorraldus, ebakvaliteetne infovahetus ja meeskonnavälised tegurid.

Osakonnas läbiviidud pilootmeeskonnas tehtud muudatuse alguses oli rõhk õppimisel ja töökorralduse muutmisel, perioodi lõpupoole, kui muudatused said uueks ja hästi töötavaks standardiks, on võimalik näha hästi toimivat timmitud meeskonda. Ühise eesmärgi nimel ühtse meeskonnana koos töötamist ilmestab järgnevad näited :

„*…näiteks selline väide, et pole minu asi see on sellisele tiimile võõras.*“

Arendaja 2

„*…kui kellelgi on mure, on see kogu tiimi mure ja kes saab, püüab aidata.*“

Testija 1

„*…igaüks pusis oma nurgas klapid silme ees enam vähem keskendudes otsejoones enda finišijoonel asuvatele tõketele ja see mis mujal toimus küll võis mõjutada, aga reaalset adumist miks ja kuidas ja mis põhjustel üks või teine osapool nii või naa seda tajub näeb lahendab/käitub ning miks siis lõpuks takistused on sellised nagu nad on, seda polnud.*“

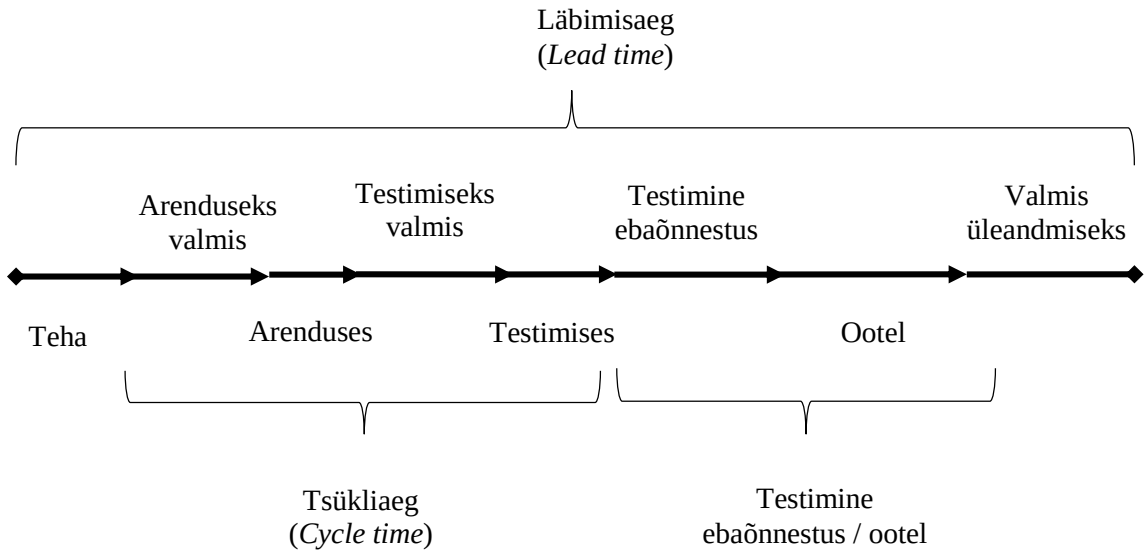
Arendaja 1

Viimane toodud tsitaat näitab ilmekalt, kuidas meeskonnaliikmed on aru saanud ja omaks võtnud timmitud mõtlemise põhilise idee – pideva parendamise, *kaizeni*. Mille alustala on Sakichi Toyoda väljatöötatud viie miks küsimuse tehnika, mille järgi korrates miks? viis korda, tuleb välja probleemi olemus ja selle lahendus saab selgeks (Liker 2004: 269). Inimesest, kes masinlikult teeb oma tööetapi õpitud liigutust, on saanud küsiv inimene, õppiva organisatsiooni alustala.

Eelnevalt analüüsiti protsessimuudatust kvalitatiivses võtmes, toetudes Aasia pilootmeeskonnaliikmetega läbiviidud intervjuudele, retrospektiivkoosolekute protokollidele ja ekspert hinnangutele (vt. lisa 4). Järgnevalt viiakse läbi kvalitatiivne analüüs toetudes andmetele ettevõtte infosüsteemist JIRA. Väljalaskeosakonna tööülesande elutsüklit on kujutatud Joonisel 9. Eraldi on välja toodud tööülesande läbimisaeg ja tsükliaeg. Läbimisaega (*lead time*) mõõdetakse infosüsteem JIRA-s tööülesande loomisest ja esmase staatuse andmisest kuni viimase staatuseni, kus tööülesanne on veel väljalaskeosakonna käes. Läbimisaega hakatakse mõõtma staatusest „Teha“ (*To Do*) hetkeni kui tööülesanne on jõudnud staatusesse „Valmis üleandmiseks“ (*Ready for Preview*). Läbimisaja sisse kuulub ka staatuste vahemik, mil tööülesanne on osakonnavälise osapoole, litsentsiaadi käes. Osakonna töötajad ootavad sel ajal litsentsiaadi tagasisidet, mille saamise järel tööülesanne kas jätkab tsüklit või saadetakse see tagasi mõnda varasemasse staatusesse.

Tsükliaeg (*cycle time*) näitab JIRA-s ainult tööülesande täitmiseks kulutatud aega. Selle aja paneb töötaja JIRA-sse ise kirja. Antud töös järgitakse ettevõtte poolt kasutatavat tsükliaja mõõtmise põhimõtet, milleks on vahemik staatuse „Arenduseks valmis“ algusest kuni staatuse „Testimises“ lõpuni. Oluline on ära märkida et tsükliajas ei kajastu staatused „*Testimine ebaõnnestus + ootel*“, kus tööülesandega meeskonna poolt tööd ei tehta. Töös võeti arvesse Aasiapilootmeeskonna poolt ühiselt otsustatud töökorralduslikku

muudatust, mille järgi alates 1. veebruarist 2016 asendati staatus „*Testimine ebaõnnestunud*“ staatusega „*Ootel*“. See muudatus täpsustas tööülesande staatust olukorras, mil testimine pole võimalik litsentsiaadist tuleneval põhjusel. Kuna need kaks staatust väljendavad antud töö kontekstis ühte ja sama ehk töö seisab osakonnast mitte sõltuvatel põhjustel, loetakse nende staatuste sisu samaks. Samuti on oluline märkida, et läbimisaega pikendab tööde nn ette ära tegemine. Töö tehakse ära ennem tähtaega, kuid infosüsteemis muudetakse staatuste alles tellitud kuupäeval. Selline käitumine ei järgi timmitud tootmise *Just-In-Time* põhimõtet. Eelnev käitumine ei mõjuta seda kui kaua tööülesanne on meeskonna käes (tööülesanne peab nii või naa tehtud saama, nii staatustes: *„Arenduses“*, *„Testimises“* viibiks tööülesanne ikkagi töö tegemiseks kulunud aja), vaid mõjutab kogu läbimisaega.



Joonis 9. Tööülesande läbimisaja ja tsükliaja mõõtmispõhimõtted väljalaskeosakonnas
Allikas: autori koostatud.

JIRA-sse on kanbani töövoo loomine ning andmetest automaatne graafiku tegemine sisse ehitatud. Aasia meeskond kasutab töö jälgimiseks JIRA-s kanban tahvlit, kus on sarnaselt tootmises kasutatava kanban tahvliga näitavad tulbad staatust. Tööülesanded saavad viibida teatud ajahetkel ainult ühes tulbas. Staatuste vahel liikumine toimub vasakult paremale, nii et kõige vasakpoolne on esimene tulp – staatus „*Teha*“ – tahvli paremas servas on viimane tulp – staatus “*Valmis üleandmiseks*“. Käesolevaks tööks vajalikud mõõtmisandmed pärinevad Aasia litsentsiaatide jaoks tehtud tööülesannete Aasia JIRA

tahvlilt ning ülejäänud brändidele mõeldud tahvlilt, see on vajalik eristamaks Aasia pilootprojekti brände ülejäänutest. Töökirjutajal on juurdepääs JIRA-sse sisse ehitatud graafilisele liidesele, kus on võimalik ise määrata, mis staatuste ja ajaperioodide kohta andmeid soovitakse. Kõiki staatuseid hõlmava graafiku visuaalne kujutis on toodud Lisas 3.

Analüüsiperioodiks on antud töös valitud kuue kuu pikkune periood, mis kattub pilootprojekti pikkusega.. Ciklumi koolitajatelt saadud eksperthinnangu järgi on võimalik juba kuue kuu möödudes teha järeldusi protsessi edukuse kohta. Ideaalis võiks aga analüüsitava perioodi pikkus olla üks aasta või pikem. Uuritav periood algas 10. oktoobril 2015 ja lõppes 11. aprillil 2016.

Töös kasutatud andmed tööülesannete kohta pärinevad infosüsteemist JIRA. Sellel perioodil on Aasia pilootmeeskonna käes olnud 279 tööülesannet. Selleks, et leida perioodi alguse ja perioodi lõpu muut, peab paika panema mis on algus ja mis on lõpp. Kuna antud perioodil on tööülesande keskmine läbimisaeg väga pikk – 2 nädalat, 4 päeva ja 6 tundi, siis autor valis protsessimuudatuse mõju hindamiseks algusperioodi ja lõpuperioodi pikkuseks ühe kuu. Sestap võrreldaksegi tabelis 9 vaatlusaluse perioodi esimese ja viimase kuu keskmisi tööülesande erinevatest staatustes olemise aegu (ehk siis perioodi algus on 30. päeva, algusega 10.okt.2016 ja perioodi lõpp on 30. päeva lõpuga 11.apr. 2016). Samuti on tabelis toodud nii aritmeetiline keskmine kui ka mediaan, millele toetudes saab järeldada, et valdavalt lühikese ajakuluga ülesannete hulgas esineb väga suure ajakuluga tööülesandeid.

Tabel 9. Aasia pilootmeeskonna tööülesannete staatuste ajad.

Tööülesande staatus	**Koguperioodi**		**Perioodi alguse**		**Perioodi lõpu**	
	aritm. keskmine	**mediaan**	**aritm. keskmine**	**mediaan**	**aritm. keskmine**	**mediaan**
Teha	1n 10t 52m	2p 2t	1n 9t 55m	5p 23t	1n 2p 2t	1t
Arenduseks valmis	22t 56m	2t 53m	1p 4t	22t 34m	12t 3m	37m
Arenduses	6t 27m	1t 2m	3t 57m	44m	5t 35m	1t 2m
Paigalduses	4t 30m	49m	4t 59m	2t 4m	3t 49m	21m
Testimiseks valmis	2p 11t	1p 18t	3p 22t	3p 1t	1p 21t	22t 51m
Testimises	8t 34m	2t 57m	7t 17m	3t 31m	12t 32m	2t 4m

Tööülesande staatus	Koguperioodi		Perioodi alguse		Perioodi lõpu	
	aritm. keskmine	**mediaan**	**aritm. keskmine**	**mediaan**	**aritm. keskmine**	**mediaan**
Test ebaõnnestunud[6]	2n 9t 27m	3p 10t	6p 14t	1p 23t	-	-
Ootel[7]	2n 18t 22m	4p 21t	5p 13t	3p 1t	3n 2p 7t	3n 5p 21t
Test ebaõnnestunud/ Ootel	1n 5p 17t	5p 22t	1n 1p 13t	4p 19t	3n 2p 7t	3n 5p 21t
Valmis üleandmiseks	2p 5t	20t 48m	20t 28m	15t 59m	1p 10t	12t 58m
Tsükliaeg	**4p 2t**	**3p 8t**	**5d 15t**	**5p 4t**	**2p 11t**	**2p 3t**
Tööaeg osakonnas	**1n 6p 10t**	**5p**	**2n 23t 27m**	**2n 18t 31m**	**1n 6p 16t**	**1n 2p 18t**
Läbimisaeg	**2n 4p 6t**	**1n 6p 22t**	**2n 5p 6t**	**2n 1p 22t**	**2n 2p 17t**	**1n 2p 22t**

Allikas: JIRA (Autori koostatud).

Kogu protsessimuudatuse üks põhiline eesmärk oli vähendada aega, mis kulub tööülesande tegemiseks, täpsemalt lühendada läbimisaega ja tsükliaega. Tabelisse 9. koondatud andmete järgi on Aasia pilootmeeskonna eesmärk täidetud – uuritud perioodi jooksul vähenes nii läbimis- kui tsükliaeg. Algne, koos koolitajatega seatud eesmärk vähendada tööaega kahe päevani, saavutati. Tsükliaeg vähenes keskmiselt viiest päevalt ja 15 tunnilt kahe päeva ja 11 tunnini. Samuti ei erine tsükliaja mediaan oluliselt aritmeetilisest keskmisest. Seega. Scrumban metoodika rakendamine on oluliselt vähendanud aega, mis kulub meeskonnal tööülesande tegemiseks.

Keskmise läbimisaja (2n 4p 6t) ja keskmise tsükliaja (4p 2t) vahe on kaks nädalat. Selle ajavahe (2 nädalat) moodustavad staatused, kus meeskond ei tee tööd ülesandega, ehk tööülesanne on erinevatel põhjustel teiste osakondade või tellija käes ootamas tagasisidet.. Siit on võimalik järeldada, et protsessis on veel potentsiaalselt palju raiskamist, sest selle kahe nädala jooksul ei anna meeskond mingit lisaväärtust tootele, järelikult võiks seda aega vähendada. Samasugusele järeldusele viitab ka mediaanide võrdlemine – läbimisaja mediaani (1n 6p 22t) ning tsükliaja mediaani (3p 8t) vahe, mis on 1 nädal ja 3 päeva.

[6] „Test ebaõnnestunud“ periood on 12 oktoober 2015 kuni 2. veebruar 2016
[7] „Ootel“ periood on 6.veebruar 2016 kuni 11.aprill 2016

Timmitud mõtteviisi järgi on olulised ainult väärtust loovad tööprotsessi osad. Sellepärast on autor loonud liitstaadiumi *„Tööaeg osakonnas“*, millega autor märgib ära kogu aja, mil tööülesanne on osakonnas, eega ei sisalda see staadium staatuseid „*Test ebaõnnestunud*“ ja „*Ootel*“. „Tööaeg osakonnas“ on keskmiselt peaaegu nädal lühem kui läbimisaeg. See näitab et probleemseid tööülesandeid on vähem kui kolmandik kogu tööülesannetest. Ettevõttel on kasulik „*Test ebaõnnestunud*“ ja „*Ootel*“. aega vähendada või teha nii, et ülesanne ei satukski nendesse staatustesse. See võimaldab pakkuda kliendile paremat teenust läbi väiksema ajakulu ja teha rohkem tööd teatud aja jooksul. Ettepanekud ettevõttele, kuidas vähendada staatuste „*Test ebaõnnestunud*“ ja „*Ootel*“ ajakulu, on välja toodud peatüki lõpus.

Tabelis 9 välja toodud staatused „*Arenduseks valmis*“ ja „*Testimiseks valmis*“ on teooria kohaselt vajalikud järjekorra pikkuse hindamiseks ning sobilikud vaid meeskonnasiseseks kasutamiseks. Nimetatud staatused on sobilikud sisemiseks kasutuseks, kuna osakonna välist inimest ehk tellijat huvitab eelkõige see kuna on töö valmis mitte vahepealsed staatused. Intervjuu käigus selgitati välja, et neid staatuseid kasutatakse meeskonnas sihtotstarbeliselt ehk neid saab kasutada hindamaks, kui pikk on ooteaeg ennem kui arendaja võtab tööülesande töösse või testija testimisse.

„*Arenduseks valmis*“ staatuse aritmeetiline keskmine on u 23 tundi, samas kui mediaan on. 3 tundi. Eelnev näitab, et enamus tööülesandeid on võimalik väga väikse ajaga töösse võtta, kuid aritmeetiline keskmise viivad üles tööülesanded, mis intervjuude kohaselt on põhjustatud ebaselgest tööülesande kirjeldusest/sisendinfost ja seetõttu peab ülesanne liikuma tagasi projektijuhile ja/või litsentsiaadile. Samas on näha scrumban põhimõtete järgimisest tulenevat suhtluse paranemist ja täpsemat info jagamist, sest perioodi algusega võrreldes on staatuse „*Arenduseks valmis*“ mediaan vähenenud 22 tunnilt 27 minutini ning keerukamate tööülesandeid keskmine ühelt päevalt 12 tunnini.

Intervjuust tuli välja, et staatuse „*Testimiseks valmis*“ ajakulus üle erinevate tööülesannete ei esine suuri kõrvalekaldeid. Seda kinnitab ka tööaja keskmise (2p 11t) suhteline väike erinevus mediaanist (1p 18t). Staatusele „*Testimiseks valmis*“ kuluv aeg vähenes vaadeldaval perioodil kolm korda. Keskmine aritmeetiline ajakulu perioodi algul oli 3p 22t, perioodi lõpus 1p 21t, mediaan vastavalt algul 3p 1h ja lõpus 22t. See on seletatav sellega, et jaanuarist 2016 eraldati meeskonnale uus kogenud testija, samuti ei

tohi alahinnata ka intervjuudest ning dokumendi analüüsist välja tulnud scrumban funktsioonideülesuse põhimõtet, mis sätestab, et projektijuhid õpivad juurde testija oskusi ja panustavad seeläbi märgatavalt testimisse.

Osakonnast väljapoole omab järjekorra näitamise tähendust staatus „*Teha*“. See võimaldab töö tellijal näha, kuna osakond tööülesande töösse saab võtta (tööülesanded võetakse tööle väljalaske kalendri alusel). Staatuse „*Teha*“ sobilikkust osakonna töö järjekorra pikkuse hindamisel uuriti meeskonna liikmete käest intervjuude käigus. Intervjuudest selgus, et sageli ootavad tööülesanded arendusest tulnud uute versioonide (*public build*) taga ehk tööülesandeid ei saa ennem töösse võtta kui kasiinokliendi versioon on arenduses valmis saanud.

Ülejäänud osakonna (mitte Aasia meeskonna) tööülesannete tegemiseks kuluv läbimis- ja tsükliaeg on toodud lisas 6. Uuritaval perioodil on ülejäänud osakonna käes olnud 922 tööülesannet. Andmetest on näha, et ülejäänud osakonna keskmine läbimisaeg (1n 5p 6t) ja keskmine tsükliaeg (3p 10t) on väiksemad kui Aasia pilootmeeskonna vastavad näitajad. See tuleneb osaliselt sellest, et Aasia klientidega seotud tööülesannete ajakulu on suurem. Näiteks lisandub ajakulu ajavahest, vastuse saamine viibib hetkeni, mil Aasias on alanud tööpäev. Lisaks eelnevale põhjusele toodi intervjuudes toodi välja, et suurim Aasia litsentsiaat teostab kliendiprogrammi testimist ise ning selleks kuluv tööaeg kajastatakse staatuses „*Ootel*“. Järelikult suureneb ka selle staatuse ajakulu, mis omakorda vähendab võimalust täpsemat analüüsi teostada. Samuti on oluline (organisatsiooni)kultuuriline erinevus Aasia klientidega, Awadi (2005: 34) toob välja, et paljud Aasia ettevõtted järgivad käsi ja kontrolli töömudelit, mis olles väga range hierarhiaga omakorda raskendab väledate tarkvaraarendusmetoodikate järgimist.

Lisasse 6 koondatud ülejäänud osakonna andmetest tuleb välja, et antud perioodil on ülejäänud osakonna läbimisaeg pikenenud keskmiselt nelja päeva võrra ning tsükliaeg pikenenud ühe päeva võrra. Antud töö raamistikust jääb sellise muutuse põhjuste analüüsimine välja. Vaadeldaval perioodil on 922st ülesandest. Nendest ennem staatusese „*Valmis üleandmiseks*“ jõudmist, läbis 302 tööülesannet staadiumid „*Test ebaõnnestunud*“ ja/või „*Ootel*“, ehk siis selleks et tööülesanne liiguks edasi neist kahest staatusest, pidi testimise ebaõnnestumise põhjus likvideeritud saama, või saama lahenduse miks tööülesanne jäi ootama meeskonnavälist sisendit.

Järgnevalt on esitatud autoripoolsed soovitused ettevõttele vähendamaks kasiino täislahenduse kliendile üleandmiseks kuluvat aega veelgi läbi scrumban põhimõtete rakendamise. Pilootmeeskonna tulemuste põhjal tasub kindlasti kaaluda kogu väljalaskeosakonnas scrumban põhimõtete rakendamise kasutamist.

Kogu läbimisaja keskmiselt suurima ajakuluga staatused on „*Teha*“ ning „*Test ebaõnnestunud + Ootel*“. Analüüsist selgus, et ajakulu põhjused on osakonna enamasti osakonnavälised ning seotud litsentsiaadiga. Osakond peab seetõttu parandama suhtlust litsentsiaadiga. Selle saavutamise võimalused on:

1. Eesmärgiks tuleb seada, et staatuses „Teha“ saaks tööülesanne piisava detailsusastme, mis vähendab tööülesande hilisemat edasi-tagasi liikumist erinevate staatuste vahel.
2. Mängude lisamise tööülesande üheks osaks peavad olema kliendihalduri tehtavad muudatused kasiino mängude ja mängijate haldamise infosüsteemis (*backend),* mille ajakulu on ühest päevast ühe nädalani. Selle asemel, et alustada nende muudatuste sisseviimisega siis kui mäng on kasiinosse lisatud (ehk arendus on lõppenud ja tööülesanne on testimeeskonna käes), võiks kliendihaldur alustada enda poolsete muudatustega juba staatuses „*Teha*“.
3. Intervjuudest selgus, et kasiino testimiseks staatuses „Test ebaõnnestunud + Ootel“ puuduvad vajalikud testkasutajad Seetõttu tuleb juba tööülesande koostamise ajal projektijuhil veenduda projektijuht testkasutajate olemasolus.

Retrospektiivkoosolekute protokollides mainiti korduvalt, et tööülesande kasiino arendusosakonnast tulevate uute kasiino programmi versioonide testimine vähendab märgatavalt meeskonna aja ressurssi. Töö autori arvates parandaks tekkinud olukorda:

- kui anda väljalaske protsesside projektijuhtidele (*Release process managers)* kasutada teistest sõltumatu testi ressurss (palgata juurde inimesi) või
- kui kogu väljalaskeosakond läheb üle funktsioonide ülestele meeskondadele ning seeläbi jagab brände mitte puudutava *build/common task* töö teatud ajahetkel väiksema koormusega meeskondade testijate vahel (protsesse parendada);

Lisaks tasuks läbi viia analüüs, mille tulemus näitaks kas tasuks luua funktsioonideülesed meeskonnad, kus on kaasatud väljalaskeosakonna ja arendusosakonna spetsialistid, see vähendaks osakondade vahelist suhtluse protseduurilisi takistusi (protsesse

parendada).Funktsioonideülese meeskonna põhimõtet võiks rakendada kasiino osakonnast väljapoole. Näiteks esindavad litsentsiaadi huve ettevõttes mitmed meeskonnad samaaegselt. Seetõttu tasuks väljalaskeosakonna projektijuhtidel analüüsida nende meeskondade keeruliste projektide teostamise ajaks kaasamise mõjusid. See vastaks väleda tarkvaraarendusmetoodika põhimõttele, kus klient kaasataks arendusprotsessi. Kuna projektijuht saab oma info erinevate meeskondade käest, aitaks nende kaasamine vähendada kommunikatsiooni probleeme.

Pideva parendamise põhimõttest lähtuvaks soovituseks on, et meeskond peaks analüüsima regulaarselt, näiteks igal retrospektiivkoosolekul, üle keskmise pikkusega tööülesandeid. Analüüs võiks välja selgitada üleliigse ajakulu põhjustajad, proovima leida nendele põhjustele lahendused.

Kokkuvõttes võib öelda, et väledate tarkvaraarendusmetoodikate scrumi ja kanbani parimate omaduste kombineerimine ja scrumbani kasutuselevõtt kasiino väljalaskeosakonna pilootmeeskonnas oli edukas. Protsessimuudatuse üheks ettevõtte poolt numbriliseks eesmärgiks seatud tööülesannete keskmine tsükliaeg vähendamine õnnestus. Funktsioonideülesemeeskonna loomisest tingitud suhtluse paranemine, ühisele eesmärgile keskendumine, efektiivsem töökorraldus ning uute rollidega seotud õppimine tõi kaasa pilootmeeskonnaliikmete moraali ja motivatsiooni kasvu. Vähendamaks veelgi kliendile kasiino uue versiooni üleandmiseks kuluvat aega (töö kasutatud mõiste läbimisaeg) on autor välja pakkunud parandusettepanekud.

KOKKUVÕTE

Olemasoleva ja sageli vähese ressursi efektiivsem kasutamine annab ettevõttele olulise konkurentsieelise. Tarkvaraettevõttes on põhiliseks ressursiks töötaja aeg. Seda aega on otstarbekas kasutada kliendi soovitud toote või teenuse loomiseks. Tarkvaratööstuses nimetatakse tarkvara loomise protsessi tarkvara arendamiseks ning tarkvaraarenduse protsessi juhtimise põhimõtteid tarkvaraarendusmetoodikateks. Oluline on leida ettevõtte jaoks sobivaim tarkvaraarendusmetoodika. Käesolevas bakalaureusetöös analüüsiti väledaid tarkvaraarendusmetoodikaid: scrum, kanban ja scrumban ning nende rakendamist tarkvaraettevõttes. Nende metoodikate paindlikkus, rõhutatult kliendi soovidega arvestamine protsessi algusest lõpuni ning protsessis leiduva raiskamise pidev vähendamine teeb need metoodikad väledaks (*agile*). Nende metoodikate põhimõtted pärinevad timmitud mõtlemisest (*lean thinking*).

Uurimustöö teoreetilises osas keskenduti väledate tarkvaraarendusmetoodikatest ülevaate andmisele teadus- ja erialakirjanduses leiduva põhjal. Töö kirjutamisel toodi välja timmitud mõtteviisi seosed väleda tarkvaraarendusmetoodikaga. Toyota tootmissüsteemist pärit raiskamise eemaldamine on timmitud tootmise põhiidee. Näidati, kuidas timmitud mõtteviisi viis peamist põhimõtet – väärtus, väärtusahel, voog, tõmme ja täiuslikkus – on aluseks väledate tarkvaraarendusmetoodikate põhimõtetele. Nii timmitud mõtteviis kui väledad tarkvaraarendusmetoodikad seavad eesmärgiks kliendile väärtuse loomise. Seda tehes keskendutakse väärtusahela leidmisele, mis omakorda võimaldab luua väärtust loovate tegevuste pideva voo. Olulisel kohal on inimeste loovusele, iseorganiseerumisvõimele ja pidevale olukorraga kohanemisele panustamine. Tänu eelnevale on võimalik protsesse pidevalt parendada (*kaizen*) ja selle läbi raiskamist vähendada. Kuna uuritava ettevõtte protsessimuudatus seisnes traditsiooniliselt kosk mudelilt üleminekule väledale mudelile, pidas autor oluliseks käsitleda ka kose mudelit ja selle kitsaskohti ning näidata, kuidas väleda mudeli rakendamine võimaldab neid kitsaskohti ellimineerida.

Uurimustöö teises pooles tutvustati analüüsitavat objekti, milleks oli tarkvara ettevõtte kasiino osakonna väljalaskeosakond. Osakonna tarkvaraarendusprotsessi analüüsimisel lähtuti teoreetilises osas toodud kose mudeli teooriast. Kuna tarkvaraarendusmetoodika vahetamine on oluline muudatus nii ettevõtte kui ka selle töötajate jaoks, moodustati uue metoodika sobivuse katsetamiseks funktsioonideülene pilootmeeskond. Väleda tarkvaraarendusmetoodika scrumbani põhimõtte rakendamist pilootmeeskonna töös hinnati kasutades teoreetilises osas toodud väledate tarkvaraarendusmetoodikate põhimõtteid ja nende rakendamise tulemuslikkust mõõtvaid näitajaid.

Töö autori positsioon uuritavas osakonnas võimaldas läbiviidud tarkvaraarendusprotsessi muudatust kõrvalt jälgida ja teostada uurimisprobleemi lahendamiseks juhtumiuuring, mille alammeetoditena kasutati dokumendianalüüsi, vaatlust ning intervjuusid pilootmeeskonna liikmetega kvalitatiivse info kogumiseks. Töös kasutatud kvantitatiivse analüüsi andmed on pärit osakonna infosüsteemist JIRA. Selles süsteemis on kirjas tööülesande detailine kirjeldus ja ajaliselt fikseeritud kulgemine läbi tööprotsessi eri etappide ehk staatuste. Vaatluse ja dokumendianalüüsi käigus saadud teadmised staatuste sisu kohta võimaldasid autoril analüüsida staatuste aritmeetilise keskmiste ja mediaanajakulu põhjal protsessis leiduvat ebaefektiivsust. Eelneva põhjal koostas autor intervjuuküsimused, et saada täiendavat infot andmetes esinenud kõrvalekallete kohta. Kõikidest andmeallikatest saadud andmetele toetudes analüüsis autor osakonnas läbiviidud protsessimuudatuse edukust ning tõi välja timmitud mõtteviisist ja väledast tarkvaraarendus mudelist lähtuvad parendusettepanekud.

Analüüsi tulemusest selgus, et väledate tarkvaraarendusmetoodikate, scrumbani kasutuselevõttu kasiino väljalaskeosakonna pilootmeeskonnas võib lugeda edukaks. Ettevõtte poolt numbriliselt mõõdetavaks eesmärgiks seatud tööülesannete keskmine tsükliaeg (aeg mil tööülesanne on meeskonna käes) vähenes viielt päevalt kahe päevani. Scrum metoodika järgi moodustatud funktsioonideülese meeskonna loomine tõi kaasa suhtluse paranemise meeskonna liikmete vahel, keskendumise ühisele eesmärgile, efektiivsema töökorralduse ning uute rollidega seotud õppimise. Kõik see on märgatavalt suurendanud pilootmeeskonna liikmete motivatsiooni ja tõstnud moraali.

Vähendamaks kliendile kasiino uue versiooni üleandmiseks kuluvat aega (töös kasutatud mõiste läbimisaeg) veelgi, on töö lõpus toodud teostatud analüüsist tulenevad parandusettepanekud ettevõttele, mis kokkuvõtvalt on järgmised:

- väledat tarkvaraarendusmetoodikat võiks rakendada terves osakonnas;
- oluline on kirjeldada tööülesanne võimalikult täpselt juba selle loomisel, et vältida hilisemat info otsimisega seotud ajakulu;
- osakonnavälised tööülesande täitmisega seotud osapooled võiksid alustada samaaegselt paralleelselt koos meeskonnaga oma tööd tööülesandega, et vältida hilisemat ooteaega;
- parandada oluliselt suhtlust ja infovahetust töö tellija ehk kliendiga;
- luua funktsioonideülesed meeskonnad, kuhu on kaasatud osakonna välised rollid sh klient või tema esindaja;
- pideva parendamise põhimõttest lähtuvalt analüüsima regulaarselt keskmisest oluliselt pikema ajakuluga tööülesandeid, et leida üles liigse ajakulu juurpõhjused ja sellest tulenevalt proovida protsessi parandada.

Olemasoleva teoreetilise materjali põhjal tasub laiendada käesolevat uurimust terves osakonnas läbiviidava scrumban metoodika kasutuselevõtu efektiivsuse uurimusele. Selleks võiks võrrelda loodavaid funktsioonideüleseid meeskondi omavahel, et leida üles parimalt koos töötavate meeskondade omadused. Laiemalt tasuks kaaluda väledate tarkvaraarendusmetoodikate kasutamise uuringut Eestis tegutsevate tarkvaraettevõtete hulgas, selgitamaks välja väledate metoodikate valiku ja leviku põhjused.

VIIDATUD ALLIKAD

1. 12 Principles Behind the Agile Manifesto, [https://www.agilealliance.org/agile101/12-principles-behind-the-agile-manifesto/], 2016.05.03
2. 75 years of TOYOTA. [http://www.toyota-global.com/company/history_of_toyota/75years/index.html]. 2016.04.26
3. **Abrahamsson P., Salo O., Ronkainen J., Warsta J.,** Agile software development methods. Review and analysis, 2002, VTT Publications 478, 118p
4. Agiilse tarkvaraarenduse manifest. [http://agilemanifesto.org/iso/et/]. 29.10.2015
5. Agile Training – Process Improvement Workshop, Ciklum Consulting, koolitusmaterjal, 2015
6. **Ambler S. W.,** Agile Software Development, Encyclopedia of Software Engineering, 2010, Wiley Online Library, 2010, 1: 1, pp. 29-46, doi: 10.1002/0471028959.
7. **Anderson D. J.,** Kanban: Successful Evolutionary Change for Your Technology Business, Blue Hole Press, 2010, 278p.
8. **Arlbjørn J.S., Freytag P.V.** Evidence of lean: a review of international peer-reviewed journal articles, European Business Review, 2013, Vol. 25 Iss: 2, pp.174 – 205.
9. Asian agile team info. Ettevõtte sise veeb. 2016.04.06
10. **Awad M. A.** A Comparison between Agile and Traditional Software Development Methodologies, 2005, Honours program thesis, University of Western Australia
11. **Babar M. A., Brown, A. W., Mistrik I.** Agile Software Architecture. Waltham: Morgan Kaufmann, 2013, 432p.
12. **Barton B.,** All-Out Organizational Scrum as an Innovation Value Chain, 2009, System Sciences, 2009. HICSS '09. 42nd Hawaii International Conference on, pages: 1-6.
13. **Bhasin S., Burcher P.** Lean viewed as a philosophy, Journal of Manufacturing Technology Management, 2006, Vol. 17 Iss: 1, pp.56 – 72.
14. **Bustard D., Wilkie G., Greer D.,** The Maturation of Agile Software Development Principles and Practice: Observations on Successive Industrial Studies in 2010 and

2012, Engineering of Computer Based Systems (ECBS), 2013, 2013 20th IEEE International Conference and Workshops on the, pp 139-146.

15. Casino Release Workshop, koolitusmaterjal, [Casino release Workshop - 30th Oct.2015.pdf], 2016.04.18
16. **Conti R., Angelis J., Cooper C., Faragher B., Gill C.** The effects of lean production on worker job stress, International Journal of Operations & Production Management, 2006, Vol. 26 Iss 9 pp. 1013 – 1038.
17. **Dingsøyr T., Nerur S., Balijepally V., Moe N. B.,** A decade of agile methodologies: Towards explaining agile software development, Journal of Systems and Software, Volume 85, Issue 6, June 2012, Pages 1213–1221
18. **Foster E.C.,** Software Engineering: A Methodical Approach - 2014 edition, 2014, Apress, 588p.
19. **Goldstein I.** Scrum Shortcuts without Cutting Corners, 2013, Addison-Wesley
20. **Highsmith J., Cockburn A.** Agile software development: the business of innovation, Computer, 2001, Volume: 34, Issue: 9, pp. 120-122.
21. **Highsmith J., Cockburn A.** Agile software development: the people factor, Computer, 2001, Volume: 34, Issue: 11, pp. 130-133.
22. **Hines, P., M., Rich, N,** The seven value stream mapping tools, International Journal of Operations & Production, 1997, Vol. 17 Iss 1 pp. 46 - 64
23. **Hines, P., Holweg, M., Rich, N.** Learning to evolve. A review of contemporary lean thinking. – International Journal of Operations & Production Management, 2004, Vol. 24, No. 10, pp. 994-1011.
24. **Ikonen M., Pirinen E., Fagerholm F., Kettunen P., Abrahamsson P.,** On the Impact of Kanban on Software Project Work: An Empirical Case Study Investigation, 2011, Engineering of Complex Computer Systems (ICECCS), 2011 16th IEEE International Conference on, Page(s): 305 – 314.
25. JIRA tutvustus, [http://atlassian.trinidad.ee/et/atlassiani-tooted/], 2016-03-24
26. **Kaleshkova N., Josimovski S., Pulevska-Ivanovska L., Postolov K., Janecski Z.,** The contribution of scrum in managing successful software development projects, 2015, Economic Development / Ekonomiski Razvoj . Jun2015, Vol. 17 Issue 1/2, p175-194. 20p.

27. **Kniberg H., Skarin M.,** Kanban and Scrum - making the most of both (Enterprise Software Development), 2010, lulu.com (March 1, 2010), Enterprise Software Development, 120p.
28. **Krafcik, J.F**., Triumph of the lean production system. Sloan Management Review, 1988, 30 (1), 41–52
29. **Kruchten P.** The Rational Unified Process: An Introduction. Boston: Addison-Wesley, 2003, 336 p.
30. **Kulusäästliku mõtlemise terminite inglise – eesti ja eesti – inglise seletav sõnastik.** [http://www.lean.ee/files/Lean_eng-est_dictionary_2013-12-06_Vol3.pdf]. 01.11.2015
31. **Lander E., Liker J. K.** The Toyota Production System and art: making highly customized and creative products the Toyota way, International Journal of Production Research, 2007, Vol. 45, No. 16, 3681–3698.
32. **Layton M. C**., Agile Project Management for Dummies, 2012, John Wiley & Sons
33. **Lehman T.J., Sharma, A.,** Software Development as a Service: Agile Experiences, 2011, SRII Global Conference (SRII), 2011 Annual, Page(s): 749 – 758.
34. **Liker J. K.** The Toyota Way: 14 Management Principles from the World's Greatest Manufacturer, McGraw-Hill, 2004, 330p.
35. **Moyano-Fuentes, J., Sacristan-Diaz,** M. Learning on lean: a review of thinking and research. – International Journal of Operations & Production Management, 2012, Vol. 32, No. 5, pp. 551-582.
36. **Nikitina N., Kajko-Mattsson M., Stråle M.,** From scrum to scrumban: a case study of a process transition, 2012, Proceeding ICSSP '12 Proceedings of the International Conference on Software and System Process, Pages 140-149.
37. **Ohno T.,** Toyota Production System: Beyond Large-Scale Production, 1988, Productivity Press; 1St Edition, 152p.
38. OOPSLA ’95, [http://jeffsutherland.org/oopsla/oo95wrkf.html], 2016.02.10.
39. Organization, [Outlook, Contacts, Organization], 2016.04.18
40. **Overhage S., Schlauderer S., Birkmeier D,. Miller J.,** What Makes IT Personnel Adopt Scrum? A Framework of Drivers and Inhibitors to Developer Acceptance, 2011, System Sciences (HICSS), 2011 44th Hawaii International Conference on, Pages: 1 - 10, DOI: 10.1109/HICSS.2011.493

41. Playtech – About us – history, [https://www.playtech.com/about_us/history], 2016.03.23
42. Playtech Eestis, [http://playtech.ee/?nav=playtechgroup_ee], 2016.03.23
43. **Poppendieck M.,** Lean Programming, 2001.05.01, [http://www.drdobbs.com/lean-programming/184414734], 2016.05.13
44. **Reddy A.,** The Scrumban [R]Evolution, Addison-Wesley Professional, 2015, 384p.
45. **Sainas, M.** Timmitud tootmise ja piirangute teooria koosrakendamine tootmisprotsesside juhtimises AS Saku Metall Allhanke Tehas näitel. TÜ Ettevõttemajanduse instituut, 2014, 105 lk. (magistritöö)
46. Scientific Method, Oxford English Dictionary, [http://www.oed.com.ezproxy.utlib.ut.ee/view/Entry/383323?redirectedFrom=scientific+method#eid, 2016.05.18]
47. **Scotland K.,** Aspects of Kanban , Software Development Magazine, 2010, Methods & Tools - Summer 2010, [http://www.methodsandtools.com/archive/archive.php?id=104], 2016.02.10.
48. **Schwaber K.,** Agile Project Management with Scrum, 2004, Microsoft Press, 172p.
49. **Smith, R., Hawkins B.** Lean maintenance: Reduce costs, improve quality, and increase market share, Elsevier Butterworth-Heinemann, 2004, 287p.
50. **Sommerville I.,** Software Engineering 9th ed., 2011, Pearson/Addison-Wesley, 790p.
51. **Spear S., Bowen, H.K.,** Decoding the DNA of the Toyota Production System. Harvard Business Review, 1999. 77 (5), 97–106.
52. State of Agile Survey 9th annual, VersionOne, [https://www.watermarklearning.com/downloads/state-of-agile-development-survey.pdf], 2016.05.20
53. **Stellman A., Greene J.,** Learning Agile Understanding Scrum, XP, Lean, and Kanban, 2014, O'Reilly Media, Pages: 420.
54. **Sutherland J., Schwaber K.,** The Scrum Papers:Nut, Bolts, and Origins of an Agile Framework, [http://www.scruminc.com/scrumpapers.pdf]. 2016.02.10.
55. **Tsui F., Karam O., Bernal B.,** Essentials Of Software Engineering 3rd Edition, 2013, Jones & Batylett Learning, 334p

56. **Vitsur, P.** Timmitud mõtteviisist lähtumine ettevõtte väärtusahela juhtimisel AS Balbiino näitel. TÜ Ettevõttemajanduse instituut, 2014, 100 lk. (magistritöö)
57. **Womack P. J., Jones T. D.** Lean Thinking Banish Waste and Create Wealth in Your Corporation, 1996, Publisher: Simon & Schuster Inc., 350 p.
58. **Womack, J. P., Jones, D. T., Roos, D.** Masin mis muutis maailma. Kuidas timmitud tootmine tõi pöörde ülemaailmsetesse autosõdadesse. Tallinn: Külim, 2010. 337 lk.
59. **Wood, N.** Lean Thinking: What it is and what it isn't. – Management Service,2004a, Vol. 48, No. 2, pp. 8-10
60. Workflow & process description, JIRA 5 migration, [https://confluence.playtech.corp/pages/viewpage.action?pageId=72158211], 2016.04.18
61. **Yin, R. K.** Case Study Research: Design and Methods. 4th edition. Thousand Oaks, Sage Publication, 2009, 219p.

LISAD

Lisa 1. Toyota Tee 14 põhimõtet

Pikaajaline filosoofia

1. Juhtimisotsused peavad põhinema pikaajalisel filosoofial, vajadusel ignoreerides lühiajalise finantseesmärke.

Õigel protsessil on õiged tulemused

2. Protsessi pidev voog toob probleemid esile
3. Kasuta tõmbe süsteemi vältimaks ületootmist
4. Jaota töö koormus (heijunka)
5. Loo töökultuur peatumaks et parandada, tõstmaks kvaliteeti juba enne veatekkimist
6. Standardiseeritud tööülesanded on aluseks pidevale parandusele ja töötaja võimustamine
7. Kasuta visuaalse kontrolli mehhanismi, toob probleemid esile
8. Kasuta ainult kindlat, põhjalikult katsetatud tehnoloogiat mis vastab su töötajate ja protsesside vajadustele

Arendades oma töötajaid ja koostööpartnereid lisad organisatsioonile väärtust

9. Kasvata juhte, kes hoomavad sügavalt tööd, elavad filosoofiat ja õpetavad teisi
10. Arenda väljapaistvaid inimesi ja meeskondi, kes järgivad su ettevõtte filosoofiat
11. Austa ja arenda oma koostööpartnereid

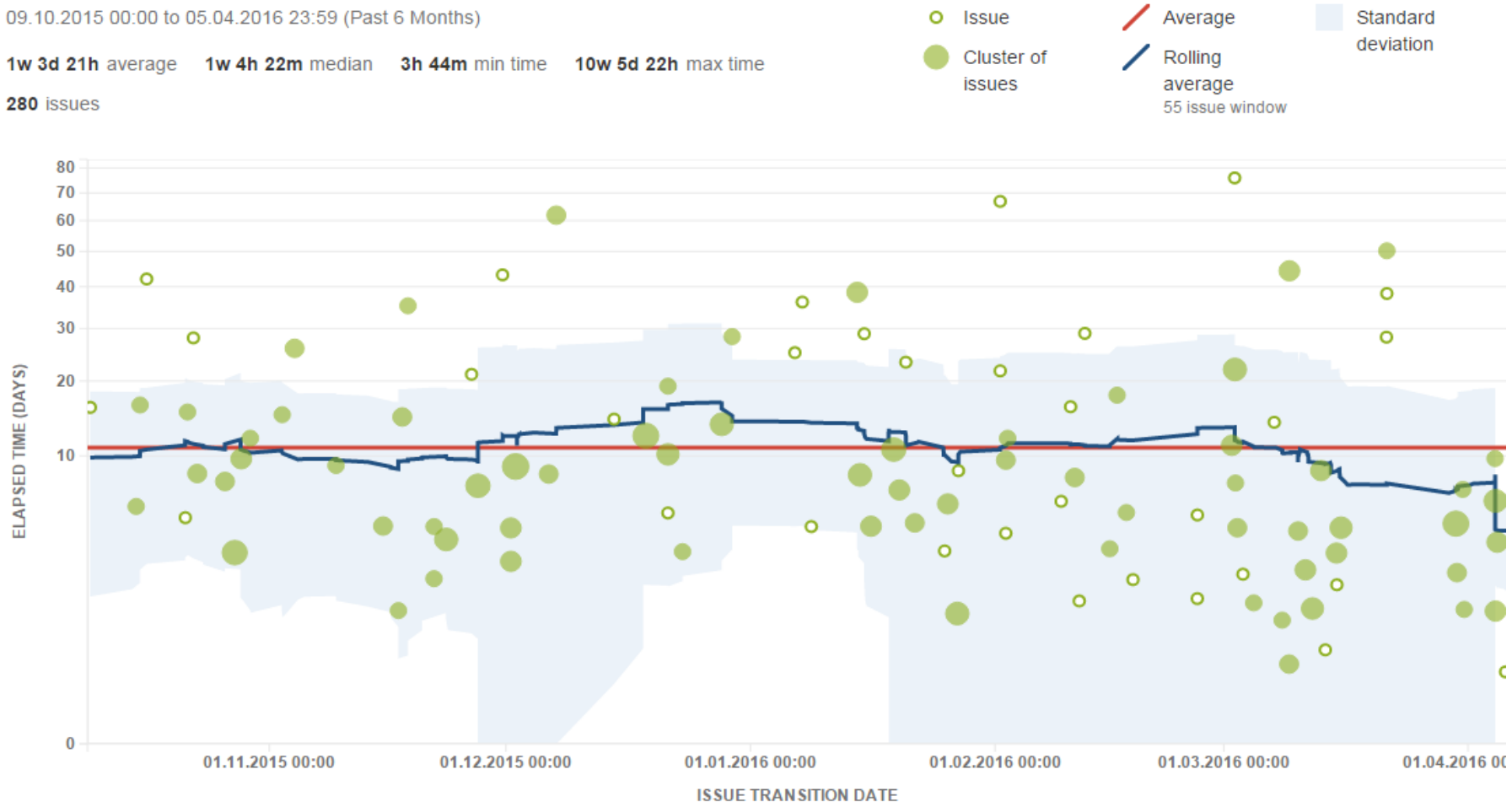
Pidev põhiprobleemide parendamine juhib organisatsiooni õppimisvõimet

12. Osale igapäeva töös, et hoomata täielikult olukordi (*genchi genbutsu*)
13. Võta otsused vastu konsensuslikult ja aeglaselt, kaaludes kõiki võimalusi, vii ellu kiirelt (*nemawashi*)
14. Õppiva organisatsiooni kultuur saavuta läbi järjekindla peegeldamise (*hansei*) ja pideva parandamise (*kaizen*) (Liker 2004: 54-58).

Lisa 2. Aasia pilootmeeskonna liikmetega läbiviidud intervjuu küsimused.

1. Mis on läinud Teie arvates scrumbani kasutuselevõtuga paremaks/halvemaks? (lisaküsimus: miks kokku ei istutud?)
2. Retrospektiivkoosolekutel mainiti korduvalt probleemsetena töökoormuse jagamist meeskonnast väljapoole (*build task/common task*). Mida annaks nende osas paremini teha? (Lisaküsimus: Kuivõrd need meeskondade ülesed ülesanded on Teie nägemuses ametialaselt arendavad?)
3. Läbimisaeg (*lead time*) on aeg mis tööülesanne on meie osakonnas. Mis saaks veel teha, et vähendada läbimisaega veelgi? (tutvustan joonist 9)
4. Mida tähendavad Teie jaoks staatused „Arenduseks valmis“ (*Ready for development*) ja „Testimiseks valmis“ (*Ready for testing*) ning „Valmis üleandmiseks“ (*Ready for Preview*)? (vajadusel näitan nende staatuste keskmist ajakulu, vastavalt 22t, 1p 18t ja 2p 5t)
5. Tööülesanne on staatustes „Test ebaõnnestunud“ (*Sanity failed*) ja „Ootel“ (*On Hold*) kokku keskmiselt 1 nädal ja 5 päeva. Palun tooge välja kuni kolm enim aega nõudvaimat põhjust, miks tööülesannet on võimatu jätkata ja see peab liikuma litsentsiaadile tagasi. (vajadusel toon välja, et arendaja ja testija käes viibis keskmiselt tööülesanne 8t) mis neil staatustel vahet on?
6. Mis tööd tehakse tööülesandega staatuses „Teha“ (*To Do*)? (Projektijuhtidele suunatud, vajadusel näitan selle staatuse keskmist ajakulu, 1n 10t 52m)

Lisa 3. Aasia pilootmeeskonna tööülesannete ajakulu graafik



Joonis 10. Aasia pilootmeeskonna tööülesannete ajakulu graafik vaatlusalusel oleval perioodil

Lisa 4. Aasia meeskonna scrum meistri ja toote omaniku eksperthinnang

Aasia meeskonna scrum meistri eksperthinnang :

- iga kahe nädala tagant toimuvad retrospektiiv meetingud on väga tähtsad.
- Testi skoopi on saanud muuta,
- 3 PMOd on suutnud omavahel töö ära jagada
- JIRA-s cycle time pole paranenud aga sellest pole ka väga hullu, sest seal olevad numbrite taga olevad taski staatuse muudatustel on sisuliselt teine tähendus, mis võimaldab spetsialistil ette tööd ära teha
- palju on ootamist on-hold ja sanity failed staatuses olevate taskide taga,
- licensee taga on palju ootamist, mis veab üles kogu cycle time (sanity failed asendas on holdi)

Aasia meeskonna ühe toote omaniku eksperthinnang :

- alguses oli kohanemine väga raske
- vaadeldava periood lõpuks on tööle saanud funktsioneerimise
- teised inimesed vanades ehk funktsionaalses meeskonnas kiidavad, et Asia meeskonna töö sujub

Lisa. 5 Väledate tarkvarametoodikate Manifesti loojate nimekiri

Väledate tarkvarametoodikate Manifesti loojad, kes lõid ka „Agile Alliance“ Kent Beck, Mike Beedle, Arie van Bennekum, Alistair Cockburn, Ward Cunningham, Martin Fowler, James Grenning, Jim Highsmith, Andrew Hunt, Ron Jeffries, Jon Kern, Brian Marick, Robert C. Martin, Steve Mellor, Ken Schwaber, Jeff Sutherland, Dave Thomas

Lisa. 6 Ülejäänud osakonna tööülesannete staatuste ajad.

Tööülesande staatus	Koguperioodi		Perioodi alguse		Perioodi lõpu	
	aritm. keskmine	**mediaan**	**aritm. keskmine**	**mediaan**	**aritm. keskmine**	**mediaan**
Teha	5p 22t	1p 23t	3p 8t	1p 17t	4p 12t	1p 16t
Arenduseks valmis	22t 22m	2t 54m	1p	19t 56m	7t 49m	15m
Arenduses	7t 18m	57m	3t 33m	22m	7t 34m	51m
Paigalduses	7t 4m	1t 29m	6t 49m	1t 43m	4t 23m	1t 21m
Testimiseks valmis	1p 18t	1p 1h	1p 3t	21t 3m	2p 19t	2p 2h
Testimises	7t 9m	2t 21m	4t 23m	1t 53m	7t 45m	2t 10m
Test ebaõnnestunud	1n 3p	2p 23t	4p 8t	21t 10m	2n 3p 2t	1n 18m
Ootel	4p 3t	1p 23t	3p 2t	1d 1h	5p 3t	3p 4t
Test ebaõnnestunud + Ootel	5p 16t	2p 2t	3p 20t	1d 5h	1n 1p 3t	3p 21t
Valmis üleandmiseks	1p	2t 32m	16t 29m	1h 26m	22t 16m	1t 34m
Tsükliaeg	**3p 10t**	**3p**	**2p 18t**	**2p 16t**	**3p 14t**	**2p 20t**
Tööaeg osakonnas	**1n 3p 9t**	**1n 1t 2m**	**2p 16t**	**5p 2t**	**1n 3p**	**6p**
Läbimisaeg	**1n 5p 6t**	**1n 1p**	**1n 1p 20t**	**5p 18t**	**1n 5p 5t**	**6p 16t**

Allikas: JIRA (Autori koostatud).

SUMMARY

THE USE OF AGILE SOFTWARE DEVELOPMENT METHODS IN A SOFTWARE DEVELOPMENT COMPANY

Software development companies are becoming increasingly important players in the economy as a computers, smart devices, etc, and software running in those affects almost all aspects of human life. Product of the software development companies is an entirely new software, additional properties of existing software, or new version of the product. The software can be developed for a specific customer or for sale to the general software market (Sommerville, 2011: 6). Regardless of the target group, it is very important for the company to reduce the time to market. Reduction of the time, by introducing the most suitable software development method, will increase the company's competitiveness.

Software companies work basically means working with information, it is a knowledge-based industry (Staats et al 2010: 377). In such companies people create value for the customer. Thus, organizing the work is critical fot the company. Obstacles, inefficiency and other factors distracting from value creation, waste the time when the employee is able to deal with the most important for the company –create the value to the customer – by creating products or services.

Toyota production system was foundation for lean thinking. This thesis fundamental concepts and methods are all related to lean thinking. It is the umbrella concept of lean production as well as agile software development methodologies such as scrum, kanban and scrumban. These are modern software development methodologies, the principles of which come from lean thinking.

Purpose of this thesis is while evaluating performance of applying agile software developemt methods, develop proposals for Estonian software company to reduce the casino software delivery time.

To reach the goals author set the following research tasks:

- explain the nature and philosophy of lean production;
- examine the literature on the basis of the principles and essence of agile software development methods;
- to highlight the link between agile software development methods and lean thinking;
- to address previous empirical studies about agile software development methods implementation;
- select indicators that help to evaluate the effectiveness of agile software development methods;
- introduce the company and give an overview of companys casino units release managements departments existing software development process;
- evaluate the agile software development methodology applying to release management department;
- make proposals to the company, how to further reduce the casino delivery time.

More efficient use of existing resources gives the company a significant competitive advantage. Employees time is the main resource for software companies. This time is meant to be used for creating the customer's desired product or services. In IT industry creating software process is called software development and process is called software development method. It is important to find most suitable software development method for the company. In this thesis agile software development methods scrum, kanban and scrumban were analyzed as also implementation of these methods in the software company. Flexibility of these methods, taking into account the wishes of the client and continuous removal of waste, makes these methods agile. These methods are come from the principles of lean thinking.

In theoretical part of the thesis the research focused on overview of the agile software development methodologies based on the literature. Removal of the waste is the lean manufacture main idea. It was shown how lean thinking’s five basic principles - the value, the value chain, flow, pull and perfection – are bases of the principles of agile software development. Lean thinking and agile software development methods aim at creating a value for teh customer. By doing this, they focus on the value chain, which in turn allows to create a continuous stream of value-adding activities. An important part is people's

creativity, self-organization and continuing to adapt to improving. Due to the foregoing processes can be continually improved (*Kaizen*) and reducing waste through it. Since the business process change involved the transition from traditional waterfall model to agile model, the author considered it is important to address the shortcomings of the waterfall model and to show how agile model implementation allows processes to be enhanced by eliminating bottlenecks.

Second half of the thesis starts with describing the casino release management department. Departments existing software development process was described based on the analysis presented in the theoretical part of the waterfall model theory. Since replacing software development method is a big thing for company and its people, a cross-functional pilot team was formed. Implementation of the agile software development method principles for the pilot team was evaluated using a theoretical findings of the agile methods presented in the theoretical part of thesis. As was measured the performance of those methods.

Author's position in the company allowed to monitor and conduct case study of the process change, by using document analysis, observation and interviews with pilot team members for the qualitative information. Quantitative data comes from the information system JIRA, it stores all the tasks with description and measured time task spent in status. Observation and document analysiz gave enough knowledge to proceed with quantitative analyzis, which in return allowed to form interview questions that helped to understand the issues behind tasks taking longer than average. Based on all the sources of the data the author analyzed the process change effectiveness and created suggestions according to lean thinking and agile software development methods.

The analysis revealed that applying agile software development methods, introduction of scrumban casino releasemanagement department's pilot team can be considered a success. Companys set target time to improve cycle time of the task was successful, it reduced from five days to two days. Applying scrum method to create crossfunctional pilto team led to an improvement in communication between the team members, focusing on the common objective, more effective work procedures and learning new skill to allow being crossfunctional. All of this has significantly increased the pilot team members' motivation and morale.

To even further reduce the delivery times of casino software (*lead time*), these suggestions were given:

- apply agile software development methods to the whole department;
- it is most important to describe all the aspects and requirements at creation of the task to avoid information asking ping-pong later on;
- Work within task, done by outside people from the team, should be started in parallel with the team, this should also help to reduce time;
- improve communication between the client and the team;
- to create crossfunctional teams that include specialists outside of the department;
- agile team should analyze regularly tasks that take much longer than average to complete, it would be following continuous improvement (*kaizen*) principal.

According to the theoretical material laid in the first part of the work, it is worthwhile to consider studying applying agile software development methods for the whole department. Performance of the teams should be then compared to find the most suitable set of personal skills for working together in agile team. The wider use of agile software development methods in Estonia's software companys could show the choice and usage of agile methods.

Lihtlitsents lõputöö reprodutseerimiseks ja lõputöö üldsusele kättesaadavaks tegemiseks

Mina, MAIT KLAOS

(sünnikuupäev 08.11.1979)

1. annan Tartu Ülikoolile tasuta loa (lihtlitsentsi) enda loodud teose

VÄLEDATE TARKVARAARENDUSMETOODIKATE KASUTAMISVÕIMALUSED TARKVARAETTEVÕTTE NÄITEL

mille juhendaja on dots. Tõnu Roolaht,

1.1. reprodutseerimiseks säilitamise ja üldsusele kättesaadavaks tegemise eesmärgil, sealhulgas digitaalarhiivi DSpace-is lisamise eesmärgil kuni autoriõiguse kehtivuse tähtaja lõppemiseni;

1.2. üldsusele kättesaadavaks tegemiseks Tartu Ülikooli veebikeskkonna kaudu, sealhulgas digitaalarhiivi DSpace´i kaudu kuni autoriõiguse kehtivuse tähtaja lõppemiseni.

2. olen teadlik, et punktis 1 nimetatud õigused jäävad alles ka autorile.

3. kinnitan, et lihtlitsentsi andmisega ei rikuta teiste isikute intellektuaalomandi ega isikuandmete kaitse seadusest tulenevaid õigusi.

Tartus, 23.05.2016