TARTU ÜLIKOOL
Pärnu kolledž
Ettevõtlusosakond

Hannu Kikkas

SÕEP3

VÕIMALUSED AGIILSE TARKVARAARENDUSE JUHTIMISE KÜPSUSTASEME TÕSTMISEKS

Lõputöö

Juhendaja: Arvi Kuura, PhD

Pärnu 2021

Soovitan suunata kaitsmisele (allkirjastatud digitaalselt)

Arvi Kuura

Kaitsmisele lubatud

TÜ Pärnu kolledži programmijuht (allkirjastatud digitaalselt)

Margus Kõomägi

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

(allkirjastatud digitaalselt)

Hannu Kikkas

SISUKORD

SISSEJUHATUS

Võrreldes mõnekümne aasta taguse ajaga on tänapäeva ühiskonnas paljud protsessid oluliselt kiiremad, paindlikumad ning orienteeritud kohanema äkiliste muutustega väliskeskkonnas. Toimuvaid muutuseid on kirjeldatud kui liikumist vedelale tänapäevasusele kus sotsiaalsed raamid pole selgelt piiritletud ja sageli kaovad need enne kui korralikult taheneda jõuaks suurendades seeläbi ebamäärasust ühiskonna osapoolte jaoks (Bauman, 2005). Vajadust arvestada väliskeskkonnas suureneva ebamäärasusega organisatsioonide ja projektide juhtimises märgati juba varakult ning tulevikus valitsevat keskkonda määratleti kirjeldava terminiga VUCA (*Volatility, Uncertainty, Complexity, Ambiguity* – muutlikkus, ebamäärasus, keerukus, mitmetähenduslikkus). Suureneva muutuse ja ebamääruse tõhusaks lahenduseks VUCA tingimustes loetakse agiilsust (*agility* – väledus, nobedus, paindlikkus) – kiiret kohanemist muutustele nii isiku, meeskonna kui organisatsiooni tasanditel (Baran & Woznyj, 2020).

Kuusinen ja Karvonen leiavad mõlemad oma töödes, et agiilsuse juurutamine organisatsiooni, programmi ning projekti tasanditel võib osutuda märkimisväärseks väljakutseks ja sellega võivad muuhulgas kaasneda pinged organisatsioonisiseses kommunikatsioonis, sest sõltuvalt organisatsiooni vanusest ning valdkonnast ei pruugi agiilse lähenemise põhimõtted olla laiendatavad kogu organisatsioonile tervikuna ja agiilsete ning mitte-agiilsete meeskondade nägemused probleemide lahendamisel võivad kardinaalselt erineda vähendades nõnda pakutavate teenuste ja toodete kvaliteeti (Kuusinen *et al.*, 2016; Karvonen *et al.*, 2018). Kuivõrd IKT (info- ja kommunikatsioonitehnoloogia) valdkonna ettevõtted on agiilsuse juurutamisel esirinnas, siis keskenduvad paljud teaduskirjanduses avaldatud artiklid ning ka käesolev töö agiilsuse efektiivse juurutamise analüüsile kas osaliselt või täielikult tarkvaraarendusega tegelevates organisatsioonides.

Tänapäevaseid tarkvaralisi lahendusi pakkuvates ettevõtetes on märkimisväärselt suurenenud sisemine vajadus olla pidevalt konkurentsivõimeline, millega kaasneb

vajadus liikuda edasi varasemalt väljatöötatud jäikadelt arendusmetoodikatelt sellistele, mis võimaldaksid arenduse käigus kiirelt kohanduda muutunud või täpsustatud nõudmistele ning seeläbi võimalikult efektiivselt peegeldaksid ühiskonnas toimuvad protsesse. Teisalt on viimasel dekaadil kasvanud tarkvaralahenduste kasutuselevõtt mitmeid sektoreid ja tööstusharusid kaasavatesse terviklahendustesse ning sellest tulenevalt on kasvanud surve tarkvaraarenduse kvaliteedi ja kiiruse tõstmiseks. Sobivaks võimalikuks lahenduseks on agiilsed tarkvaraarenduse metoodikad, mis peaks katma nii sisemised kui välised vajadused.

Paraku on paljud tarkvaraarendusega seotud organisatsioonid süvenenud agiilsete metoodikate teoreetilistesse põhimõtetesse ning parimatesse praktikatesse pinnapealselt, mistõttu hiljem juurutamise efektiivsust analüüsides osutub, et agiilsed metoodikad on kasutusel kas poolikult, lihtsalt turundusliku märksõna tasemel ja tegelikult on kasutusel hoopis mõni muu (aeglasem, vanem) arendusprotsess või on organisatsioon leiutanud hoopis enda mugandatud hübriidversioonid agiilsest tarkvaaraarendusest. Seda ilmestab 2015. aasta „*CHAOS Report*“-i statistika mille kohaselt ainult 5% raporti koostamisel kasutatavatest projektidest on teostatud agiilsete metoodikatega (Johnson & Mulder, 2015, lk 6). Sarnaselt leiavad Tuncel *et al.*, et agiilse küpsuse hindamismudelid (AMM – *Agile Maturity Model,* AMA – *Agile Maturity Assessment Model*) on küll aidanud vähendada lõhet teooria ja praktika vahel, kuid jätkuvalt on märkimisväärne puudujääk agiilsuse juurutamisel (Tuncel *et al.*, 2020, lk 51).

Töö eesmärk on sünteesida agiilse tarkvaraarenduse jaoks sobiv küpsusmudel, tulemuskaart ja soovitused mõõdikute formuleerimiseks.

Eesmärgi täitmiseks püstitas autor järgnevad uurimisküsimused:

- millisel määral sobivad olemasolevad küpsusmudelid rakendamiseks agiilse tarkvaraarendusega tegelevates organisatsioonides;
- millisel määral on agiilse tarkvaraarenduse põhimõtted juurutatud tarkvaraarendusega tegelevates organisatsioonides tuginedes sünteesitud küpsusmudelile.

Lõputöö eesmärgi saavutamiseks defineeris autor järgnevad uurimisülesanded:

- kirjeldada tarkvaraarenduse teoreetilist tausta ja tuua välja seotus projektijuhtimisega;
- tuua välja küpsusmudelite roll tarkvaraarenduse protsesside tõhustamisel;

- hinnata teaduskirjanduses kirjeldatud agiilsete küpsusmudelite sobivust praktiliseks rakendamiseks tarkvaraarendusega tegelevates organisatsioonides;
- sünteesida sobivad mõõdikud ja tulemuskaart agiilse tarkvaraarendusega tegelevatele organisatsioonidele tuginedes teaduskirjanduses leiduvatele agiilsetele küpsusmudelitele;
- mõõta ja analüüsida agiilse tarkvaraarenduse küpsustaset organisatsioonides baseerudes sünteesitud tulemuskaardile ja mõõdikutele;
- tuua välja üldistatud soovitused agiilse tarkvaraarenduse küpsustaseme tõstmiseks.

Lõputöö koosneb kahest osast. Töö esimeses osas käsitletakse tarkvaraarenduse juhtimise jaoks olulisi teoreetilisi printsiipe. Teooria esimeses alapeatükis kirjeldatakse projektide rolli tarkvaraarenduses, teises ja kolmandas alapeatükis tutvustatakse traditsioonilise ning agiilse tarkvaraarenduse projektijuhtimise meetodikaid. Neljandas, viiendas ja kuuendas alapeatükis analüüsitakse küpsusmudelite rolli tarkvaraarenduse protsesside tõhustamisel, tutvustakse küpsustaseme hindamise mudelite teoreetilist tausta ning mõõdikuid. Esimese osas viimases alapeatükis eraldi tuuakse välja agiilse küpsustaseme hindamismudelite rakendatavuse problemaatika koos hindava võrdlusega sobivusest Eesti organisatsioonidele.

Töö teises osas kirjeldatakse esmalt sünteesitud küpsusmudeli väljatöötamise metoodikat ja tulemuskaarti, seejärel tutvustakse uuringu meetodit, valimit ning struktuuri. Sünteesitud küpsusmudelile tuginedes viiakse läbi kvantitatiivne uuring neljas tarkvaraarendusega tegelevas Eesti organisatsioonis ning tuuakse välja uuringu tulemused ja analüüs. Lõpetuseks tehakse teostatud uuringu tulemuste põhjal üldistatud ettepanekud agiilse tarkvaraarenduse küpsustaseme tõstmiseks.

1. TARKVARAARENDUSE JUHTIMISE PRINTSIIBID

1.1. Projektijuhtimine tarkvaraarenduses

Projektistumine on 1995. aastal esimest korda mainitud termin, millega kirjeldatakse arenenud ühiskonda tabavat nähtust, kus järjest rohkem igapäevaseid (töö)ülesandeid teostatakse kas alaliste või ajutiste projektidena. Projektid on kõikjal ja läbivalt kasutusel kõigis ühiskonna kihtides ning etappides (Schoper, 2017, lk 71). Kui majanduses laiemalt hakati projektistumist märkama 90-ndate keskpaigas, siis võib väita, et mitmetes sektorites toimus orgaaniline projektistumine ajaliselt oluliselt varem. Selleks et mõista projektide ja projektijuhtimise olulisust ning mõju (ühiskondlikele) protsessidele tuleb esmalt aru saada individuaalsete terminite tähendustest.

Projekt on ühe PMI (*Project Management Institute*) definitsiooni järgi ajutine ettevõtmine loomaks unikaalselt toodet, teenust või tulemit (PMI, 2017, lk 4). IPMA (*International Project Management Association*) definitsiooni järgi on projekt ajutine, kuid unikaalne ja organiseeritud ettevõtmine kokkulepitud tulemi realiseerimiseks eeldefineeritud nõuete ning piirangutega (IPMA, 2015, lk 27). Ajutisus tähendab, et igal projektil on olemas kindel algus ja lõpp, kuid see ei tähenda, et projektid on lühikese kestvusega. Samuti pole projektide ajutisus seotud projekti tulemi elueaga (PMI, 2017, lk 5).

Projektijuhtimine on teadmiste, oskuste, vahendite ja tehnikate rakendamine projekti eesmärkide huvides, mis toimub läbi projekti jaoks kindlaksmääratud protsesside asjakohase kasutamise. Projektijuhtimine võimaldab organisatsioonidel projekte tõhusalt ja tulemuslikult teostada (PMI, 2017, lk 10). Projektide juhtimisel lähtutakse sageli varasematest kogemustest ja kirjanduses defineeritud standarditest, kuid iga projekti tuleb käsitleda erinevalt ning projektijuhi töö projekti juhtimisel on projektimeeskonna ja huvigruppide abil leida tasakaal varasemate kogemuste ning standardite kasutamisel arvestades projekti eripära. Seetõttu on PMI poolt väljaantav PMBOK (*Project*

Management Body Of Knowledge) kirjeldanud projektijuhtimise standardit kui soovituslikku praktikat, mitte kui ranget ettekirjutust (*Ibid.*, lk 2).

Tarkvaraarendus on algusest peale tihedalt olnud seotud erinevate projektijuhtimise metoodikatega ning protsessipõhise lähenemisega, sest tarkvaraarenduse alguspäevil sooviti esmalt tugineda juba olemasolevatele kogemustele teistest tööstusharudest kus projektide efektiivsus oli ennast juba tõestanud. Kuna tarkvara arendamine oli tollal ning on siiamaani väga sarnane järkjärguliselt maja ehitamisega, kus algelt arendatakse välja tarkvaraline põhi (nö vundament) ning seejärel juurutatakse täiendavalt erinevad funktsionaalsused (seinad, aknad, uksed), siis esimeste tarkvaraprojektide süstemaatilisel elluviimisel eelistati kasutada ehituses ennast tõestanud projektijuhtimise metoodikaid ning rangeid protsessipõhiseid mudeleid.

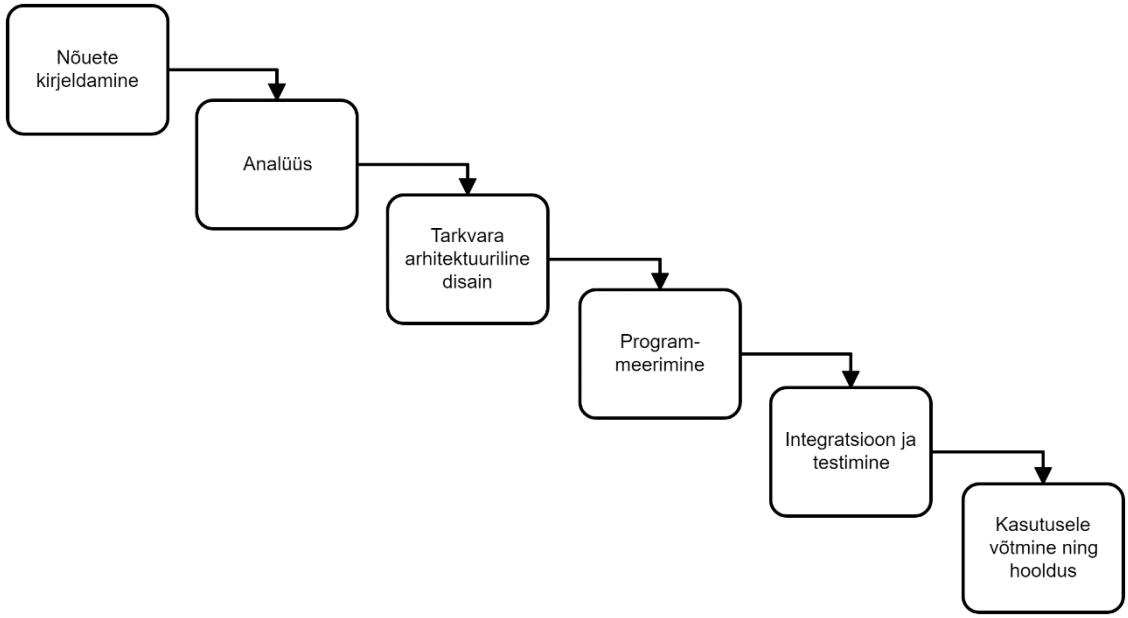
1.2. Traditsioonilised tarkvaraarenduse metoodikad

Kõige levinumaks projektijuhtimise metoodikaks tarkvaraarenduses on koskmudel (*waterfall*), mille kohaselt on kogu protsess võimalikult detailselt planeeritud üksteisele järgnevate lineaarsete sammudena, kus iga järgnev samm eeldab sisendit eelnevast sammust. Laialt kasutusel projektijuhtimise protsessina ehitustööstuses oli koskmudeli metoodika eelistamise põhjuseks oluliste muudatuste raskendatud teostamine projekti hilisemates faasides, mistõttu tuli kõik algselt põhjalikult läbi planeerida.

Esimene teadaolev formaalne kirjeldus tarkvaraarenduseks koskmudeli kohandamisest pärineb aastast 1970, kus Royce (1970, lk 328) analüüsib ning toob välja läbiviidud tarkvaraarenduse projektide baasil seitse üksteisele järgnevat ja kindlas järjekorras täidetavat sammu, mis on ühised erinevates projektides sõltumatult projekti suurusest või keerukusest:

- süsteemi nõuete kirjeldamine;
- tarkvara nõuete kirjeldamine;
- analüüs;
- tarkvara arhitektuuriline disain;
- programmeerimine / teostamine;
- integratsioon ja testimine;
- kasutusele võtmine ja hooldus.

Praktikas leidis tarkvaraarenduses rohkem kasutust Royce`i poolt kirjeldatud mudeli kuuesammuline optimeeritud ja kohendatud alternatiiv, kus kaks esimest nõuete kirjeldamist käsitlevat sammu ühendati oma sisuliste sarnasuste tõttu ning ülejäänud protsess jäi samaks (joonis 1). Awad (2005) juhib sealjuures tähelepanu, et kuigi koskmudeli konkreetsete sammude nimetused võivad erinevates organisatsioonides ning projektides erineda, siis koskmudeli metoodikat kasutav projekt on alati jaotatud sarnase sisuga etappideks (protsessiks) (lk 3).



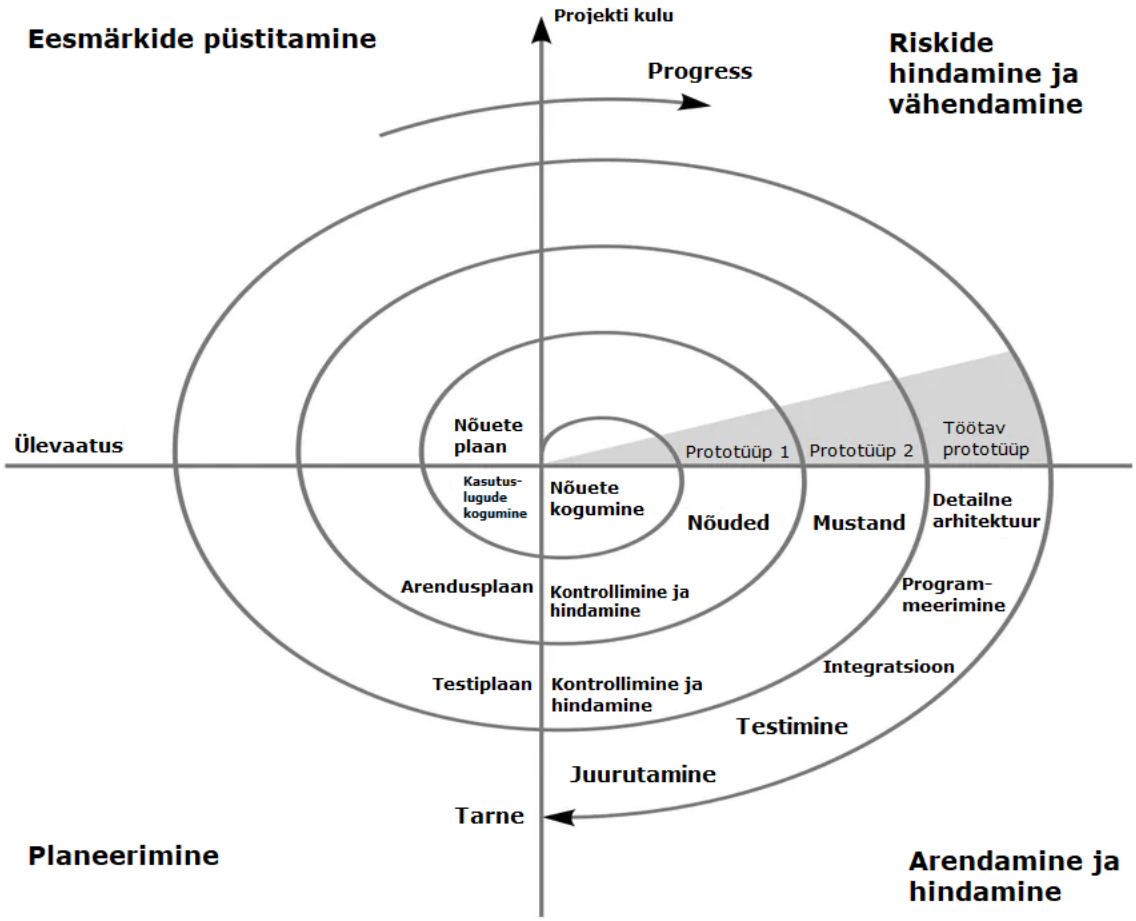
Joonis 1. Optimeeritud koskmudel tarkvaararenduses, kus on ühendatud nõuete kirjeldamise sammud, Royce 1970 alusel

Paraku on nii ideaalse kui optimeeritud koskmudeli (joonis 1) baasil tarkvaraarenduse kitsaskohaks ajas kasvavate ja suuremate mahtudega tarkvaraarenduste keerukus ning tundmatute ja teadmata tegurite mõju. Nõnda võtab nõuete defineerimine, detailne analüüs ning planeerimine väga suure osa projekti ajast, kuid samas ei välista, et hilisemal programmeerimise või juurutamise sammul esineb kitsaskohti, millele nõuete kirjeldamise või analüüsi etapis ei mõeldud või mille esinemine oli raskesti ennustatav. Sellisel juhul tuleks eelnevalt jäigalt lõpuni planeeritud protsessis minna sammudes (vastuvoolu) tagasi, teostada täiendavaid analüüse, täiendada analüüsidokumenti ja kohandada projektiplaani vastavalt. Lõpptulemuse valmimise seisukohalt ei aita

analoogsed hilised täpsustamised kaasa projekti kokkulepitud tähtajaks valmimisele, mis võib ettevõtluses omakorda vähendada organisatsiooni usaldusväärsust.

Iteratiivne ning inkrementaalne tarkvaraarenduse mudel (*iterative and incremental model*, iteratsioon – sammude jada korduva sooritamise protsess, tsükkel; inkrementaalne – astmeline, järk-järguline) on paralleelselt koskmudeliga väljakujunenud tarkvaraarenduse metoodika mille taust on 60-ndate aastate kosmosetööstuses, kus lisandus igapäevaselt kardinaalselt uusi ja kosmosest arusaamu muutvaid avastusi ning polnud võimalik arvestada nõuete lõplikkusega. Metoodika järgselt on iga projekt jaotatud inkrementaalseteks arendusiteratsioonideks (arendustsükliteks) ja iga iteratsioon on võrdväärne iseseisva miniprojektiga, mis hõlmab endast nõuete analüüsi, disaini, arendamise arhitektuurilist disaini ning testimist. Iteratsioonide eesmärk on välja arendada testitud tükk tarkvara mis sisaldab kõigest osalist funktsionaalsust võrreldes lõppeesmärgiga. Iga iteratsiooni järel on planeeritud aeg tagasisideks ja täiendavate analüüside tegemiseks, mille käigus täpsustuvad nõuded tarkvarale, mis annavad omakorda sisendi järgmiseks arendusiteratsiooniks. Arendustsüklite tulemusel valminud tükk tarkvara lisab lõpptulemusse uut funktsionaalsust samm-sammult (inkrementaalselt) ning viib järjest lähemale projekti eesmärgile, milleks on lõpliku tarkvaratoote tarnimine. Tavaliselt on kuni viimase iteratsioonini kogu valmiv lõpplahendus tarkvaraarendust teostava organisatsiooni käsutuses ning kliendile tarnitakse alles viimase iteratsiooni lõplik tulemus (Larman, 2004, lk 10). Kokku on arendustsükleid minimaalselt kolm, maksimaalset arvu pole piiratud. Lihtsustatud iteratiivse ja inkrementaalse tarkvarametoodika protsessijoonis on toodud lisas 1.

Pideva koskmudeli kasutamise ning viimistlemise käigus töötas Barry Boehm 1988. aastal välja spiraalmudeli eesmärgiga ühendada samm-sammult prototüüpimist ja nõuete muutlikkust, kuid sooviga säilitada planeerimise olulisust protsessis.



Joonis 2. Spiraalmudeli faasid lihtsustatud neljale ringile, iga spiraaliring tähistab järjekordset projekti etappi. Allikas: Boehm, 1988, lk 64

Spiraalmudel (joonis 2) ei kirjelda konkreetset metoodikat ja on suunatud riskide maandamisele kasutades teisi tarkvaraarenduse protsessijuhtimise mudeleid individuaalsete projekti (alam)eesmärkide saavutamiseks (nt koskmudel, iteratiivne ja inkrementaalne mudel). Ringide arv mudelis pole fikseeritud ning sõltub tarkvaraprojekti eesmärkidest. Kokku on Boehmi poolt defineeritud neli faasi projekti arenguetappidest lähtuvalt, kuid samas pole kõigi faaside kaasamine igas projektis kohustuslik:

- eesmärkide püstitamine – projekti järjekordse etapi eesmärgid tuvastatakse;
- riskide hindamine ning vähendamine – riskid tuvastatakse, analüüsitakse ning leitakse võimalused riskide maandamiseks;

- arendamine ja hindamine – projekti etapi teostamiseks valitakse sobiv arendusmetoodika ning vahendid;
- planeerimine – projekt ülevaatus ning koostatakse plaanid spiraali järgmise ringi tarbeks (Boehm, 1988, lk 64).

Lõputöös kirjeldatud traditsioonilistele metoodikatele on kirjanduses viidatud ka kui raskekaallastele, kus projekti eelduseks on kindel ja juba projekti alguses lõplikult kindlaksmääratud nõuete kogum ning rõhk on planeerimisel (Meso & Jain, 2006, lk 20). Awad (2005) kirjeldab üldistatult raskekaallaste järgmised ühised jooned:

- ennustav lähenemine – suur osa projektist planeeritakse tüüpiliselt detailselt pikaks ajaks ette, seega on kõik projekti lõpetamiseks vajalikud sammud ennustatavad; rõhuasetus on algse plaani järgimisel mis omakorda võimaldab piisava täpsusega arvutada tähtaegu ja eelarvet;
- põhjalik dokumentatsioon – nõudeid kirjeldav dokument on võtmetähtsusega projekti edukaks lõpetamiseks; valitseb uskumus, et enne arendamise alustamist on võimalik koguda kokku kõik projekti jaoks olulised nõuded, seega edasised sammud sõltuvad kogutud nõuete dokumendi põhjalikkusest;
- protsessile suunatus – metoodika on välja töötatud selliselt, et sõltumatult individuaalse töötaja rollist projektis kirjeldab valitud metoodika täpselt rollipõhised protseduurid protsessis, mida töötaja peab projektis tegema ja jälgima;
- töövahenditele suunatus – kõigi metoodika sammude jaoks on olemas täpsed töövahendid (lk 6).

Kokkuvõtvalt leiab autor, et traditsiooniliste tarkvaraarenduse projektijuhtimise metoodikate puhul on tegemist protsessidega mis on märkimisväärselt viinud edasi tarkvaratööstuse arengut ja nende väljakujunemine on tugevalt seotud tarkvaraarenduse sektori noorusega. Kuid tänapäevases muutuvas maailmas on arenduste keerukuse kasvust lähtuvalt vajalik ka metoodikate enda areng või kohendamine. Traditsiooniliste metoodikate ebaedu ning kitsaskohti on kinnitanud ametlik statistika. 1995. aastal avaldatud „*CHAOS Report*“ (iga-aastane uurimus tarkvaraarenduse projektide edukuse mõõtmiseks) raporti kohaselt kõigist uuritud tarkvaraprojektidest koguni 53% ületasid eelarvet ja ei valminud õigeaegselt. Sama raporti 2015. aasta väljaande järgi oli koskmudeli metoodikat kasutanud tarkvaraprojektide edukus ainult 11% (Barden, 2017,

lk 24). Rohkelt ebaõnnestunud projektid peegeldavad autori arvates kõige selgemalt traditsiooniliste metoodikate ebasobivust pidevate muutustega kohanemisel.

1.3. Agiilse tarkvaraarenduse metoodikad

Agiilseid (agiilne – väle, kiire, paindlik, kohanev) tarkvaraarenduse metoodikaid on võimalik lugeda traditsiooniliste metoodikate loogiliseks edasiarenguks sooviga vähendada projektide ebaõnnestumist. Kuna arvukate jätkuanalüüside käigus on leitud, et tarkvaraprojektide ebaõnnestumine on olnud sageli seotud halva omavahelise kommunikatsiooniga projekti jaoks oluliste huvirühmade (*stakeholder*) vahel (Yin *et al.*, 2011, lk 20), mida süvendab arendusprotsessi nõrk paindlikkus, siis viis kasvav rahulolematus traditsiooniliste metoodikate oluliste puudujääkide osas 2001. aastal „Agiilse tarkvaraarenduse manifest“-i (*Manifesto for Agile Software Development*) avaldamiseni pikaajalise tarkvaraarenduse kogemusega autorite poolt.

Manifest on mõeldud suunama tarkvaraarendust varasemalt põhjalikult planeeritud protsessidelt inimeste omavahelise suhtlemise väärtustamisele ja kiirematest ühiskonnas toimuvatest protsessidest tuleneva määramatusega arvestamisele. Avaldatud dokumendis tuuakse välja 12 järgitavat põhimõtet, millele lisaks on neli olulist väärtust autorite poolt eraldi esile tõstetud:

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

Esiletõstetud väärtuste puhul kehtib reegel, et nö paremal pool kirjeldatud ja varasemalt olulised tegurid pole väärtusetud, kuid nende vasakpoolseid alternatiive väärtustatakse rohkem (Beck *et al.*, 2001).

Kuigi manifesti avaldamisest on möödas juba 20 aastat, siis on agiilse tarkvaraarenduse metoodika ühese definitsiooni kirjeldamine jätkuvalt väljakutse. Kuna manifestis toodud põhimõtteid on võimalik vabalt tõlgendada ja juurutada, siis jäävad erinevad autorid erinevatele seisukohale, et milliseid konkreetseid metoodikaid üldse lugeda agiilseteks ning milliseid mitte. Protsessina on kõik agiilse tarkvaraarenduse metoodikad tugevalt

seotud traditsioonilise iteratiivse ning inkrementaalse arendusmetoodikaga. Seda kinnitab Larman (2004) enda raamatus ja leiab, et pidev agiilsete metoodikate ja iteratiivsuse koosmainimine on tekitanud väärarusaama nagu iteratiivne arendus on agiilsete meetodikate väljatöötamisega kaasnenud leiutis (lk 35). Tema arvates pole võimalik täpselt defineerida, et mida täpselt lugeda agiilsete metoodika definitsiooniks, sest kõik väidetavalt agiilsed metoodikad erinevad praktikas mõnevõrra üksteisest, valdavaks ühisosaks on lühikesed ajaraamidega (arendus)iteratsioonid järkjärgulise plaani ning eesmärkide täpsustamisega. Lihtsustatult on ainuke kohustuslik komponent agiilsus (*agility*) – kiire ja paindlik kohanemine muutustele (lk 25).

Tuntuimad agiilse tarkvaraarenduse metoodikad on XP (*Extreme Programming*, ekstreemprogrammeerine, autor Beck), *Kanban* (jaapani keeles „stend“) ning Scrum (autoriteks Schwaber & Sutherland). Mitmete enamlevinud agiilsete metoodikate autorid on ühtlasi agiilse tarkvaraarenduse manifesti autorite hulgas.

Kõige tuntum agiilne metoodika on Scrum, mis on oma sisult iteratiivne ja inkrementaalne protsess ükskõik millise toote arendamiseks või arendustöö juhtimiseks. Metoodika keskendub juhtnööridele kuidas meeskonnaliikmed peaksid töötama, et saavutada paindlikkus pidevalt muutuvas keskkonnas. Scrum ise on enda nimetuse saanud nimetuse ragbist kus sarnase nimega kutsutakse strateegiat, mille abil saada mängust väljas pall taas mängu läbi meeskonnatöö (Schwaber, viidatud Awad, 2005, lk 10 vahendusel).

Scrum jagab meeskonnaliikmed kolme rolli:

- *Scrum Master* – vastutab korrektse Scrumi kasutamise ning juurutamise eest (organisatsioonis, projektis, meeskonnas, ...);
- toote omanik – vastutab (tarkvara)toote äriliste huvide ning (tarkvara)arenduse ühes suunas hoidmise eest;
- meeskond – vastutab lahenduse valmimise eest (Yin *et al.*, 2011, lk 21).

Scrumi meeskond koosneb viiest kuni üheksast liikmest, kes on ristfunktsionaalsete oskustega (*cross-functional skills*) ja edukalt võimelised iseseisvalt töötama. Scrum ei määratle iga iteratsiooni käigus kasutatavat tarkvaraarenduse metoodikat, vaid pakub juhiseid ning tööriistu ettearvamatuse vältimiseks projektis tervikuna. Kesksel kohal

Scrumi tööprotsessides on sprint – kuni neljanädalane arendustsükkel osalise funktsionaalsusega toote tarnimiseks. Juhi roll projektis on meeskonna igapäevatöö detailse organiseerimise asemel takistuste kõrvaldamine ja sujuva töö võimaldamine, meeskond organiseerib enda tööd ise tulenevalt sprindi edukaks lõpetamiseks vajalikest eesmärkidest. Scrumis on defineeritud neli olulist artefakti millega iga sprindi jooksul arvestatakse:

- tegemata tööde loetelu – kõigi toote valmimiseks vajalike tööde nimekiri, prioriseeritult;
- sprindi tegemata tööde loetelu – kõigi sprindi käigus tehtavate tööde nimekiri, prioriseeritult;
- tarnimise graafik (*Release Burndown Chart)* – projekti progressi kujutav graafik;
- sprindi graafik (*Sprint Burndown Chart*) – sprindi progressi kujutav graafik (Yin *et al.*, 2011, lk 21).

Awad (2005) toob Scrumi metoodika juures täiendavalt välja igapäevase (lühi)koosoleku (*Scrum Daily*) olulisuse, mille eesmärk on saada ülevaade sprindi progressist, kõrvaldada takistusi sprindi lõpetamiseks ja lahendada jooksvalt meeskonna probleeme (lk 10). Scrumi lihtsustatud protsessijoonis on toodud lisas 2.

Lisaks suuremale paindlikkusele on võrreldes traditsiooniliste metoodikatega agiilsetes metoodikates kasutusel nö parimad praktikad, mis originaalselt pärinevad mõnest konkreetsest metoodikast (eriti ekstreemprogrammeerimisest), kuid mida kasutatakse laiemalt kõigis agiilse tarkvaraarenduse projektides, vähendades nõnda erinevate metoodikate omavahelisi piire veelgi. Rohkem esile tõstetud praktikateks on igapäevased koosolekud, paarisprogrammeerimine (kaks tarkvaraarendajat kasutamas sama töövahendit), testimispõhine programmeerimine (esmalt arendatakse testid ja siis funktsionaalsus) ja kasutajalood (*stories / user stories*) (*Ibid.*, lk 29).

Agiilsete ning traditsiooniliste meetodite erinevuste ülevaatlik kokkuvõte on välja toodud tabelis 1. Võrdlemisel on vaadeldud erinevate lähenemiste mõju nii manifestis eraldi väljatoodud eritingimuste kui projekti seisukohalt.

Tabel 1. Traditsiooniliste ning agiilsete tarkvaraarenduse metoodikate erinevused

Mõõdik	Traditsiooniline	Agiilne
Nõuded	Nõuded kaardistatakse projekti alguses, muudatused raskendatud	Alustamise faasis nõuded ebaselged, pidev täpsustamine (arendus)iteratsioonide vahel
Dokumentatsioon	Põhjalik, oluline sisend tarkvaraarenduseks	Pole kriitiline lõpliku tulemuse tarnimiseks, tase ning maht lepitakse kliendiga kokku
Klient / kasutaja	Klient kaasatud nõuete kirjeldamise ning tarkvara juurutamisel, pikk ajaline vahe soovide ning tulemuse vahel lisab riski, et lõplik tulemus ei vasta enam soovidele / on aegunud	Klient ning kasutajad kaasatud iga iteratsiooni järel, eesmärk tarnida soovitud tarkvara
Organisatsiooni struktuur	Sobib suurtele organisatsioonidele, protsessid ja rollid rangelt paigas	Sobib organisatsioonidele, mis on valmis protsesside järgimisel kohanduma kliendi tagasisidest lähtuvalt
Projekti struktuur	Ettemääratud protsessid, kõik sammud mõeldakse eelnevalt läbi, tarnimine viimase sammuna, projektimeeskond rohkearvuline	Orgaanilised protsessid, iteratiivne ja ennustamatu, tarnimine iga iteratsiooni järgselt, projektimeeskond maksimaalselt 20 inimest
Projekti tulemuslikkuse näitaja	Plaanile vastavus, võimalik väärtus ärile tekib projekti lõpus	Ärilise lisandväärtuse loomine, võimalik väärtus ärile tekib koheselt peale esimest tarnet

Allikas: Awad 2005, Yin *et al.* 2011 ja Beck *et al.* 2001; autori kohandatud

Paradoksaalselt on üheks agiilsete metoodikate suurimaks kitsaskohaks hinnatud täpselt sama omadust, mida loetakse ka agiilsuse tugevuseks – huvirühmade tihedat omavahelist suhtlemist ja avatud organisatsioonisisest kultuuri. Yin *et al*. (2011, lk 21) toovad välja, et kui projekti eduks on vajalik pidev ning efektiivne kliendi ja arendusmeeskonna omavaheline koostöö just läbi näost-näkku suhtlemise, siis suurimaks puuduseks võib saada järjest suureneva globaliseerumise juures inimestevaheline distants. Organisatsioonide struktuurid kus arendusüksused asuvad erinevates linnades või suisa eri riikides, pole enam haruldased. Modernsete IKT vahendite kasutamine (videosillad jms) aitab probleemi leevendada, kuid kõik huvirühmad ei pruugi vastava suhtlusformaadiga hakkama saada või jääb selle käigus mõni oluline mõte edastamata.

Teise suure kitsaskohana on välja toodud projektimeeskonna suurus. Kuna agiilsed metoodikad eeldavad tihedat meeskonnaliikmete koostööd ja suhtlust (omavahel, kliendiga), siis projektimeeskonna liikmete arvu kasvades muutub töö efektiivne koordineerimine raskendatuks. Nii Constantine kui Fowler lisavad, et näost-näkku

suhtlemine muutub tülikalt keeruliseks agiilsetes projektimeeskondades kus on rohkem kui 20 liiget (Fowler, 2004; Constantine, 2001, viidatud Awad, 2005, lk 26 vahendusel).

Autori arvates võib organisatsioonides lisaks eelpool väljatoodud puudustele, tekitada arusaamatust agiilse tarkvaraarenduse metoodika täpse definitsiooni puudumine, millele omakorda aitab kaasa parimate praktikate läbisegi kasutamine. Sealjuures võib segadust veelgi suurendada võrdlus iteratiivse ja inkrementaalse traditsioonilise metoodikaga, mida loetakse raskekaallaseks ning iganenuks, kuid millele tuginevad mitmed agiilsed metoodikad. Kuid neist väljatoodud kitsaskohtades hoolimata loetakse agiilsed metoodikaid efektiivseks vahendiks organisatsiooni äriliste väljakutsete lahendamisel tänapäevastel kiiresti muutuvatel ja killustunud globaalsetel turgudel, mis hindavad kõrget kvaliteeti, tõhusust ning kliendile suunatud tooteid ja teenuseid (Highsmith, *Ibid.*, lk 18 vahendusel).

1.4. Tarkvaraarendus ja küpsusmudelid

Enamike kasumit taotlevate organisatsioonide eesmärgiks on konkurentidega võrreldes omada mõnda eelist mis võivad olla näiteks oluliselt sügavamad tööstusharu (eri)teadmised, parem kvaliteet või lihtsalt odavam hind. Tarkvaraarenduse sektori sisenemiskulud on võrreldes töötleva tööstusega väga madalad ja konkurents suureneb koos tehnoloogiliste võimaluste arvu kasvuga, seega on konkurentidest eristumisel lisaks eelpool toodule täiendavaks müügiargumendiks tõusnud oluliselt kiiremad tarkvara arendamise protsessid, millega kaasneb tavalisest kiirem tarkvaratoote tarnimise aeg (*time to market*).

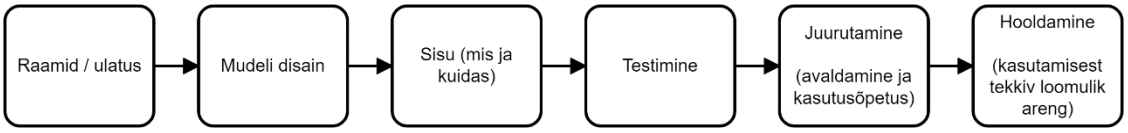
Selleks et objektiivselt hinnata ettevõtte turupositsiooni võrreldes konkurentidega on vaja esmalt mõista mida mõõta, kuidas ning millega mõõtmistulemust võrrelda. *AS-IS* olukorra kaardistamiseks on vaja abivahendeid, mille kasutamisest on võimalik vajadusel järeldada ja defineerida meetmed olukorra parendamiseks ning mis hiljem aitavad lisaks kontrollida parendamise progressi (Becker, *et al.*, 2009, lk 213). Soovitud eesmärkide saavutamisel võib olla abi küpsusmudelite kasutamisest.

Henriques *et al.* (2017, lk 56) defineerivad küpsusmudelit rangelt protsessist lähtuvalt ning leiavad, et küpsusmudel kirjeldab kuidas protsess saab ajas areneda (küpseda). De

Bruin *et al.* (2005, lk 8) kirjeldavad küpsusmudelit kui vahendit, millega hinnata valitud domeeni küpsust (kompetentsust, võimekust, keerukuse taset) enam-vähem kõikehaaravate kriteeriumite põhjal.

Becker *et al.* (2009, lk 213) võtavad asja kokku kõige konkreetsemalt sõnades, et küpsusmudel koosneb järjestikustest küpsuse tasemetest ühe või mitme konkreetse mõõdiku piires. Küpsuse tasemed tähistavad nende mõõdikute eeldatavat, soovitud või tüüpilist arenguteed mis on kirja pandud etappidena. Esimene tase (tavaliselt kujutatuna kõige all) tähistab algolekut mida võib iseloomustada kui minimaalsete oskuste / teadmiste kogumit vaadeldava mõõdiku piires. Kõrgeim tase tähistab seevastu maksimaalset küpsust ning küpsuse arengutee kõigi tasemete läbimist. Küpsusmudel toimib arenguteel oleva organisatsiooni positsiooni hindamise skaalana ja pakub välja kriteeriumid ning tunnused mida tuleb täita teatud küpsustaseme saavutamiseks.

Küpsusmudelite väljatöötamisel on kasutusel mitmed teooriad. Üheks levinumaks võib pidada De Bruini *et al.* 2005. aastal avaldatud käsitlust mis jagab küpsusmudeli arendamise kuueks iseseisvaks sammuks. Väljatoodud sammude järjekord on oluline ning kriitiline lõpliku mudeli seisukohalt – valed otsused küpsusmudeli raamide / ulatuse defineerimisel mõjutavad oluliselt mudeli testimist (joonis 3). De Bruin *et al.* (2005, lk 9) rõhutavad, et küpsusmudeli lihvimisel on vaja vahel minna testimise sammu tulemuste analüüsil tehtud järelduste tõttu küpsusmudeli sammudes nö tagasi ning muuta vastuvõetud otsuseid.



Joonis 3. Küpsusmudeli arendamise etapid. Allikas: De Bruin *et al.*, 2005, lk 9

Küpsusmudeli tasemed määratakse De Bruini *et al.* järgi ära mudeli disainimise sammus, mõõdikud sisu sammus. Praktikas on kõigi rohkem levinud viietasemelised mudelid, kuigi De Bruini *et al.* teoorias pole tasemete arvu piiratud ning ainukesteks olulisteks tingimusteks on viimase taseme definitsiooni konkreetsus ja loogiline areng läbi tasemete. Tasemete arvu valikul on De Bruini *et al.* arvates sagedasti lähtutud kõige

esimesest küpsusmudelist CMM (*Capability Maturity Model*), kus oli samuti kirjeldatud viis küpsuse taset (*Ibid.*, lk 11).

Küpsusmudelite küpsuse hindamisel soovitavad De Bruin *et al.* kasutada mõõtmisinstrumendina ankeetküsimustikku. Kvantitatiivsete mõõtevahendite kasutamine võimaldab kogutud andmete järjepidevat statistilist analüüsi ja lihtsustab tulemuste võrdlemist. Küsimustike edastamine ning andmete kogumine IKT vahendite abil muudab küsimustiku hõlpsasti levitatavaks globaalselt (*Ibid.*, lk 14).

Teaduskirjanduses käsitletud küpsusmudelite baasil koostatud küsimustikud on sageli struktureeritud tabelina, kus ühes dimensioonis esitatakse küpsusmudeli sisulist mõõdikut ning küsitluse täitjal palutakse hinnata mõõdiku tunnetuslikku väärttust vastaja jaoks, teisest dimensioonist lähtuvalt on küsimused esitatud enamasti küpsustasemete mõttes kasvavalt. Kõige levinum viis küpsuse väärtuse hindamiseks on viiepunktiline Likerti skaala, kus tähis „5“ tähistab kõrgeimat küpsuse taset (De Bruin *et al.*, 2005, lk 8).

Autor leiab, et head küpsusmudelid aitavad mõista organisatsioonis juba eksisteerivate (tarkvaraarenduse) protsesside ning meeskondade puudujääke ja pakuvad lisaks võimalust kvantitatiivselt või kvalitatiivselt võrrelda organisatsiooni sees individuaalseid töötajaid, erinevaid meeskondi ning käimasolevaid projekte. Iga küpsusmudeli järgmise taseme saavutamine tähistab sammu teekonnal paremuse suunal ja viib lähemale soovitud tulemuse saavutamisele (Fontana *et al.*, viidatud Henriques, 2015, lk 56 vahendusel).

1.5. Traditsioonilise tarkvaraarenduse küpsusmudelid

Esimene laialdast kasutust leidnud ja üldse üks vähestest traditsioonilise tarkvaraarenduse küpsusmudelitest CMM (*Capability Maturity Model*, võimekuse küpsusmudel) pärineb 1986. aastast USA Carnegie Melloni ülikoolist. Oma nimetuse on küpsusmudel saanud USA valitsusepoolsest soovist mõõta koostööpartnerite tarkvara arendamise võimekust (*capability*). Küpsusmudeli teoreetilisel kavandamisel võeti aluseks, et tarkvaratoote kvaliteedi määrab suures osas selle loomiseks kasutatud tarkvaraarenduse protsesside kvaliteet, kuivõrd protsessi loetakse selleks, mis ühendab ja seob omavahel organisatsiooni jaoks olulised komponendid – inimesed, meetodid ning töövahendid (Team, 2006, lk 4). CMM küpsusmudeli järkjärgulise tasemete struktuuri puhul on

juhindutud TQM (*Total Quality Management*) kvaliteedijuhtimise põhimõtetest kohandades need tarkvaraarendusega tegelevatele organisatsioonidele. Kokku kirjeldab CMM ära viis taset kasvavas küpsuse järjekorras:

- esialgne (*initial)* – tarkvaraarenduse protsessid kaootilised;
- korratav (*repeatable*) – algelised projektijuhtimise protsessid, et jälgida kulusid ja tähtaegu;
- defineeritud (*defined*) – tarkvaraarenduse protsessid on dokumenteeritud ning standardiseeritud;
- hallatud (*managed*) – tarkvaraarenduse protsesside efektiivsuse ja tarkvaratoodete kvaliteedi mõõtmine;
- optimeerimine (*optimising*) – järjepidev protsesside jälgimine ning parendamine (Paulk, 2009, lk 8).

Algselt kutsuti mudelis viimast küpsustaset „optimeeritud“ tasemeks, kuid kuna järjepidev protsesside parendamine toimub kestvalt ja pidevalt, siis kohendati taseme nimetust 1988. aastal. Paulk hinnangul on CMM küpsusmudel midagi, mis loob tugeva põhja projektijuhtimise ja organisatsioonilise arengu tõenduspõhiseks juhtimiseks (*Ibid.*, lk 6).

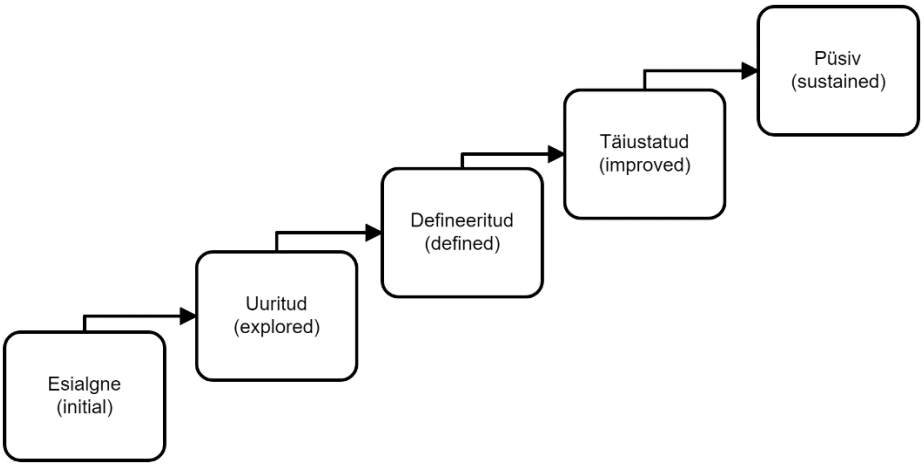
Lõputöö autor leiab, et kuna CMM on suunatud protsesside optimeerimisele ning mitte mõne konkreetse tarkvaraarenduse metoodika probleemide lahendamisele, siis pole üllatav traditsiooniliste tarkvaraarendusmetoodikate küpsusmudelite vähesus – CMM oli ja on jätkuvalt edukalt rakendatav kõigile metoodikatele. Traditsiooniliste tarkvaraarenduse metoodikate kasutuse vähenemine projektide läbiviimisel on kindlasti aidanud kaasa traditsiooniliste küpsusmudelite ja eeskätt CMM kasutuse hääbumisele, kuid esitatud põhimõtted ning kriteeriumid pole alati valed ega iganenud. Seetõttu nõustub autor Beckeri *et al.* (2009, lk 213) väitega et küpsusmudel on tõhus IT juhtimise vahend, mis võimaldab organisatsiooni paremini positsioneerida ja aitab leida paremaid lahendusi muutusteks. Nagu agiilse tarkvaraarenduse iteratsioonid on uuesti leitud ning täiustatud vana, siis võivad ka vanade traditsiooniliste metoodikate küpsusmudelid pakkuda väärtuslikku sisendit uuemaid tarkvaraarenduse metoodikaid katvatele küpsusmudelitele.

1.6. Agiilse tarkvaraarenduse küpsusmudelid

Sarnaselt agiilse tarkvaraarenduse metoodikatega on agiilsete küpsusmudelite arengut võimalik lugeda orgaaniliseks arenguks uute tarkvaraarenduse metoodikate järjest suurema kasutuselevõtu tõttu erinevates organisatsioonides, mis on ühtlasi aidanud ka agiilse tarkvaraarenduse definitsiooni täpsustada. Seda kinnitavad Turetken *et al.* (2017, lk 1) ning toovad välja seose, et kui agiilsed arendusmetoodikad leiavad tarkvaraarenduses üha rohkem kasutust ja sellega on kaasnenud tänapäevaste organisatsioonide mõttemaailmas muudatused, siis varasema agiilse arendusprotsessi kasutuselevõtmise vajalikkuse põhjendamise asemel suunatakse tähelepanu agiilse arendusprotsessi maksimaalsele efektiivsusele ning heale laiendatavusele.

Organisatsiooni agiilsuse mõõtmiseks on mitmeid võimalusi, millest mõned eeldavad eksperthinnangu saamist ja teised keskenduvad enesehindamise meetoditele, sest need võimaldavad organisatsiooni taset mõõta kiiremini (Yürüm & Demirörs, 2017, lk 392). Gren *et al.* (2015, lk 40) leiavad viitega Sidky tööle, et organisatsiooni agiilsuse mõõtmine taandub lihtsustatult kasutusele võetud agiilse põhimõtete kogusele ning seega on võimalik välja töötada mõõtmisinstrument, mis võimaldab agiilsuse taset üheselt määrata – organisatsioon, mis järgib kümmet agiilse tarkvaraarenduse põhimõtet on agiilsem kui organisatsioon, mis järgib ainult kolme. Kõiki eelpool toodud probleeme lahendavateks universaalseteks vahenditeks sobivad teoorias agiilse tarkvaraarenduse küpsusmudelid.

Esimene teadaolev väljatöötatud agiilne küpsusmudeli pärineb 2009. aastast (joonis 4), eesmärgiks parendada ja täiustada agiilset tarkvaraarendust ning edendada agiilse tarkvaraarenduse põhimõtete levikut nagu madalam hind, kliendi rahulolu, kvaliteet jne (Patel *et al.*, 2009, lk 6). Küpsusmudeli väljatöötamisel on Yini *et al.* hinnangul saadud inspiratsiooni CMMI küpsusmudelist (*Capability Maturity Model Integration*, CMM mudeli edasiarendus erinevate CMM versioonide ühendamiseks), sest kattuvad nii tasemete arv (5) ning osaliselt ka tasemete nimetused (Yin *et al.*, 2011, lk 22).



Joonis 4. Algne agiilse tarkvaraarenduse küpsusmudel. Allikas: Patel *et al.*, 2009, lk 6

Yini *et al.* märgatud seosed on agiilse tarkvaraarenduse suunitlusega küpsusmudelite puhul sagedased – mitmed küpsusmudelid on saanud inspiratsiooni juba olemasolevatest väljatöötatud ning tarkvaraarenduses varem levinud agiilsetest küpsusmudelitest. Tihti eksisteerib omavaheline ühisosa mõõdikute põhimõttelise sisu osas, kuid erinevus sõnastuse ja / või tasemete piires paiknevate kriteeriumite osas. Viimane asjaolu raskendab küpsusmudelite omavahelist võrdlemist ning Schmitt *et al.* (2019, lk 662) tõdevad enda töös, et juba olemasolevate agiilsete küpsusmudelite sarnasusi ning erinevusi võrdlevaid artikleid pole piisavalt. Sama leiavad ka Yürüm ja Demirörs, kes toovad välja, et agiilsete arendusprotsesside küpsusmudelite kirjeldustes on paralleelselt ühisosadega selliseid mõõdikuid, mis välistavad üksteist ja seetõttu raskendavad erinevate varasemate küpsusmudelite üksteisega võrdlemist (Yürüm & Demirörs, 2017, lk 392).

Lõplikku väljatöötatud agiilsete küpsusmudelite arvu on raske määrata ning tuleb arvestada võimalusega, et erasektori organisatsioonid ei kipu enda sisemisi küpsusmudeleid väljapoole jagama, sest efektiivsuse saladuse jagamine võib vähendada eelist konkurentide ees. Mitmete avaldatud küpsusmudelite puhul on piirdutud agiilsete põhimõtete rakendamisega mõnele varasemale traditsioonilisele protsessijuhtimise küpsusmudeli põhjale ning tulemiks on olnud hübriidküpsusmudelid, mida ikkagi ametlikult presenteeritakse agiilse tarkvaraarenduse küpsusmudelitena (Schmitt *et al.*, 2019, lk 661). Neid väiteid illustreerib Tunceli *et al.* (2020, lk 54) töös toodud statistika kus on välja toodud 30 akadeemilist ning 10 erasektori taustaga mudelit. Nurdiani *et al.*

(2019, lk 2) hinnangul on teadaolevaid agiilse tarkvaraarenduse erinevaid küpsusmudeleid ainult teaduskirjanduses mainitud vähemalt 40, autoriteks nii agiilse arenduse eksperdid kui akadeemikud. Agiilsete küpsusmudelite nõrkusi analüüsivas Anderseni *et al.* (2020, lk 261) töös on võetud aluseks 47 erinevat küpsusmudelit kirjeldavat teadusartiklit, Schmitt *et al.* (2019, lk 663) on kaasanud enda võrdlevas artiklis rohkem juhuslikke allikaid internetist ning on piirdunud 28 agiilse küpsusmudeli leidmisega.

Küpsusmudelite rohkus on käesoleva lõputöö autori hinnangul viinud olukorrani kus küpsusmudelite vahel navigeerimine ja võimalikust rakendamisest saadava kasu (eel)hindamine organisatsiooni või projekti seisukohalt muutub ajas järjest keerulisemaks.

1.7. Agiilsete küpsusmudelite rakendatavus

Agiilsete küpsusmudelite rakendatavuse osas on välja toodud väga palju puudujääke ning kriitikat. Fundamentaalsel tasemel on probleem termini endaga – küpsus tarkvaraarenduses on väga suhteline ning muutlik termin mis muutub ajas, täieneb pidevalt ning harva saavutab lõpliku täiuslikkuse. Seega pole kunagi võimalik koostada küpsusmudelit, mille abil püüelda maksimaalsete tasemeteni kõigis mõõdikutes (Andersen *et al.*, 2020, lk 264).

Paljude mudelite puhul saab öelda, et on olemas agiilsuse eraldi paradigmana käsitlemine, kuid sageli on sellistel juhtudel mudelite sisendiks agiilsete tarkvaararenduse metoodikate puhtalt teoreetiline taust või juba varasemalt avaldatud mudelid. Keskmiselt ainult umbes pooled mudelid arvestavad agiilse tarkvaraarenduse manifestiga (Tuncel *et al.* 2020, lk 58). Lihtsustatult algab mitmetes teadustöödes protsess esmalt katsega luua ning kirjeldada järjekordset teoreetilist küpsusmudelit, mille pädevuse valideerimiseks teostatakse seejärel uuring kus valimi hulk on parimal juhul kuni 50 vastajat, halvimatel juhtudel puudub valideerimine täielikult.

Kui järgida De Bruin'i *et al.* küpsusmudelite väljaarendamise teooriat, siis kõige rohkem eksitakse küpsusmudelite kavandamise esimeses sammus mille käigus määratakse mudeli sihtrühmad ning mis võimaldab sihtrühma liikmetel vajadusel kaasa aidata

küpsusmudeli edasisel arendamisel (De Bruin *et al.*, 2005, lk 10). Valdavas enamuses on küpsusmudelite kirjeldamisel jäetud tähelepanuta just konkreetne suunatus – millise suurusega organisatsioonidele, kui suure töötajate hulgaga projektimeeskondadele, jne. Ühtlasi on see oluline aspekt mudelite hindamisel Eesti kontekstis, sest heal küpsusmudelil on olemas praktiline rakendatavus VKE-des (väikesed ja keskmise suurusega ettevõtted, valdav osa Eesti tarkvaraarendusega tegelevatest organisatsioonidest).

Lõputöö autori arvates saab (agiilsete) küpsusmudelite väljatöötamise protsessi veelgi täiustada, kui mudeli väljatöötamise ajendiks oleks ühe või paremal juhul suisa mitme erineva suurusega organisatsiooni vajadused ja soovid ning kirjeldatud küpsusmudel on mitmete (agiilsete) iteratsioonide tulem, mille käigus (teaduslik) teooria täpsustub sisendi või tagasisidena reaalelust. Sarnase protsessi puudumisest tulenevalt on väljatöötatud küpsusmudelid kas liiga üldistatud või vastupidiselt väga sageli eeldavad mõne väga konkreetse tarkvaraarenduse metoodika kasutust, tuginevad varasemalt kirjeldatud ning ajaloolise lähenemisega protsessidele tarkvaraarenduses või sobivad rohkem organisatsioonidele, mis asuvad pika tarkvaraarenduse ning projektijuhtimise ajalooga regioonides.

Erinevate küpsusmudelite rakendatavust hinnates on jõutud sarnase kriitikani. Leppänen enda 2013. aasta analüüsis leiab, et agiilsed küpsusmudelid on veel oma varajases staadiumis ning ühtegi analüüsitud küpsusmudelit pole loodud või vaadeldud eduka agiilse tarkvaraarenduse seisukohalt (lk 336). Ta järeldab et nende praktikas edukaks rakendamiseks on vaja teostada veel palju nii kontseptuaalset kui ka empiirilist uurimistööd (lk 338). Tuncel *et al.* 2020. aasta artiklis uurivad rakendatavust võrreldes Leppäneniga suisa 14-st erinevast aspektist lähtuvalt ning lõpuks järeldavad sarnaselt, et mitte ükski analüüsitud küpsusmudelitest pole piisavalt terviklik, et praktikas rakendada. Ühe vähese positiivse asjaoluna toovad Tuncel *et al.* välja, et mõnedes mudelites on huvitavaid elemente, mida saaks taaskasutada mõne uue agiilse küpsusmudeli väljatöötamiseks (Tuncel, *et al.*, 2020, lk 51).

Rakendatavuse täpsemaks uurimiseks toetub autor 2019. aastal ilmunud Schmitti *et al.* uurimustööle, mis üritab süstematiseeritult hinnata leitud agiilsete küpsusmudelite

(AMM, valim 28 mudelit) sobivust vastavalt püstitatud kriteeriumitele. Püstitatud kriteeriumite puhul piisab mudeli vastavusest vähemalt ühele kriteeriumile:

- AMM peab omama erinevaid küpsustasemeid, kus tasemed peavad olema piisavalt konkreetselt kirjeldatud, et nii ekspertidel kui mitte-ekspertidel on võimalus hinnata praegust taset ning on arusaadavad järgmise taseme saavutamiseks vajalikud tegevused;
- AMM peab võimaldama hinnata nii ettevõtte kui meeskonna omaenda küpsustaset sisaldades küsimusi, mis toetavad ja juhivad hindamisprotsessi;
- AMM peab sisaldama nõuandeid agiilsuse suurendamiseks organisatsioonides (Schmitt *et al.*, 2019, lk 663).

Antud lõputöö raames hindab autor küpsusmudelite sobivust VKE, avaliku sektori asutuste ning geograafilise sobivuse (Eesti organisatsioonikultuur) aspektist. Kuna mitmed agiilsed küpsusmudelid on rohkem kõlapinda leidnud võrreldes analoogidega, siis leidub küpsusmudeleid, mis on esitatud nii Tunceli *et al.* 2015. aasta kui Schmitti *et al.* 2019. aasta artiklites ning nende mudelite puhul on võimalik osaliselt ära kasutada juba varasemalt teostatud analüüsi.

Kõigis aspektides on autori poolt küpsusmudelite hindamisel kasutusel kolmepalline skaala:

- „-„ – ei sobi / pole võimalik hinnata;
- „+“ – nõrk sobivus;
- „++“ – tugev sobivus.

Küpsusmudeli raamid ja ulatus võivad toetada erineva suurusega organisatsioone ning Tuncel *et al.* (2020, lk 54) eraldavad ulatuse erinevateks kategooriateks – meeskond, toode / projekt, portfell / programm ja organisatsioon. Küpsusmudel võib korraga vastata mitmele ulatuse väärtusele. Kuigi suurim lisaväärtus tekib mudelitest mis on maksimaalse ulatusega ehk kasutatavad nii väikeste kui väga suurte organisatsioonide puhul, siis tabelis 2 kasutab autor mõnevõrra erinevat hindamismetoodikat hindamisel VKE sobivuse aspektist:

- tugeva sobivusega on hinnatud küpsusmudelid, mille ulatus on meeskond ja toode / projekt või meeskond ja organisatsioon;

- nõrga sobivusega on küpsusmudelid, mille ulatuseks on märgitud ainult meeskond või ulatus sisaldab portfelli / programmi;
- mitte sobivad on küpsusmudelid, kus on eraldi välja toodud sobivus väga suurtele korporatsioonidele või kus on mudeli väljatöötamisel olnud sisendiks väga suure korporatsiooni probleemid.

Portfellid / programmid on märgitud nõrga sobivusega, sest on tavapäraselt kasutusel väga suurtes organisatsioonides ning sellist ulatust toetavad küpsusmudelid võivad olla raskemini rakendatavad VKE-des.

Avaliku sektorile sobivuse hindamisel vaatleb autor küpsusmudelite seotust mõne konkreetse agiilse tarkvaraarenduse metoodikaga ja küpsusmudelite mõõdikutest tulenevaid kohustuslikke tarkvaraarenduse praktikaid. Praktikatest tulenevalt võivad meeskonnas, projektis või organisatsioonis olla vajalikud konkreetsed rollid ja tegevused, mille saavutamiseks peaks muutma avaliku sektori organisatsiooni struktuuri ning töökorraldust või palkama täiendavat tööjõudu. Kõik need võivad omakorda seada täiendavaid väljakutseid asutuse eelarvele või tekitada ebakõla näiteks riikliku tarkvaraarenduse standardiga. Tabelis 2 väljatoodud küpsusmudelid on hinnatud põhimõttel, et need on teostatavad pikema aja vältel võrreldes VKE protsesside kiirusega:

- tugeva sobivusega on hinnatud küpsusmudelid, mille puhul on mõõdikutes väljatoodud tasemete järgimine minimaalse lisakeerukusega (nt paarisprogrammeerimine või igapäevased (lühi)koosolekud) või puudub keerukus üldse ning küpsusmudeliga seotud metoodika ei põhjusta uute rollide tutvustamist organisatsioonis;
- nõrga sobivusega on küpsusmudelid, mille puhul mõõdikutes väljatoodud kriteeriumid vajavad suuri muudatusi (mitu metoodikat paralleelselt kasutuses) või kus küpsuse saavutamiseks on kohustuslikud täiesti uued rollid (nt *Scrum master*);
- mitte sobivad on küpsusmudelid, kus on eraldi välja toodud sobivus väga suurtele korporatsioonidele või kus on mudeli väljatöötamisel olnud sisendiks väga suurte korporatsioonide probleemid.

Geograafilise sobivuse hindamisel arvestab autor Eesti (organisatsiooni)kultuuri eripäradega. Kuna ükskõik millisel tasemel agiilsete tarkvaraarenduse meetodite

juurutamine on strateegiline otsus organisatsiooni seisukohalt, siis Schneider *et al.* on enda 1991. aasta töös näidanud, et erinevates regioonides üle maakera on kultuurilised erinevused, mis omakorda väljenduvad organisatsioonikultuuris ning mõjutavad strateegiate sisu ja reageerimist väljakutsetele. Kui lääne kultuuriruumis juhid maandavad muutustest tulenevat teadmatust ning riske ja inglisekeelses mõjuruumis minimeeritakse muutuste tähtsust, siis Jaapanis juhid hoopis kohandavad organisatsiooni strateegiaid muutuste paremaks mõistmiseks (Schneider *et al.*, 1991, lk 316). Ayed *et al.* (2017, lk 153) viivad kultuurilised erinevused igapäevasesse töökeskkonda ja hindavad, et meeskonnaliikmete kultuuriline taust mõjutab tarkvaraarenduse praktikaid ning sellega tuleb arvestada kui tarkvaraarendusega tegelevad meeskonnad soovivad agiilseid tarkvaraarenduse metoodikaid kasutada efektiivselt ning edukalt. Eesti IKT organisatsioonide kultuuri ja strateegiaid mõjutavad enim Skandinaavia riikide organisatsioonid oma geograafilise läheduse, sarnase kultuuriruumi ja tihedate majanduslike sidemete tõttu:

- tugeva sobivusega on hinnatud küpsusmudelid, mis on konkreetselt suunatud Skandinaavia regiooni organisatsioonidele või mille valideerimiseks kasutati Skandinaavias asuvate ettevõtete tagasisidet;
- nõrga sobivusega on kõik ülejäänud küpsusmudelid.

Organisatsioonikultuuriliselt mitte sobivaid küpsusmudeleid põhimõtteliselt ei eksisteeri, sest tarkvaraarenduse ning tehnoloogiasektori üheks edasiviivaks jõuks on tihe konkurents ja enda parimate praktikate teistega jagamine. Jagamise tulemusel analüüsitakse, mugandatakse ja kohandatakse teiste organisatsioonide praktikaid globaalselt. Seega on kõigi küpsusmudelite sisu alati vähemalt osaliselt kasutatav kõigis riikides.

Tabel 2. Agiilsete küpsusmudelite rakendatavus VKE, avaliku sektori ning geograafilise sobivuse aspektist

Mudel, esimene autor, aasta	VKE	Avalik sektor	Geograafiline sobivus
The Agile Adoption Framework (Sidky, 2007)	++	++	+
Agile Maturity Model (AMM): A Software Process Improvement framework for Agile Software Development Practices (Patel, 2009)	+	+	+
The Agile Maturity Model – Applied to Building and Releasing Software (Humble, 2009)	++	++	+
Agile Karlskrona test (Seuffert, 2009)	++	++	++
Seven Dimensions of Agile Maturity in the Global Enterprise (Benefield, 2010)	+	++	+
Agile Maturity Model (AMM): The 5 Levels of Maturity (Proulx, 2010)	+	+	+
An Agile BI Maturity Model (Woods, 2011)	-	-	+
Scrum Maturity Model (Yin, 2011)	+	+	+
Enterprise Continuous Integration Maturity Model (Minick, 2014)	-	+	+
A Reference Model for Software Agility Assessment: AgilityMod (Ozcan-Top, 2015)	+	++	+
Assessing the adoption level of scaled agile development: a maturity model for Scaled Agile Framework (Turetken, 2017)	+	-	+

Allikas: autori koostatud, Schmitt *et al.* 2019 ja Tuncel *et al.* 2020 alusel
„-„ sobivus puudub või on raskesti hinnatav, „+“ nõrk sobivus, „++“ tugev sobivus

Avaliku sektori sobivuse hindamisel on erandiks Minicku 2014. aasta küpsusmudel, mis on küll konkreetselt suunatud korporatsioonidele, aga samas on lõputöö autori hinnangul mudel suures osas kasutatav ka avalikus sektoris.

Tarkvaraarenduse juhtimise printsiipide kokkuvõttena saab välja tuua järgnevat. Tarkvaraarenduse sektori noorusest tulenevalt on tarkvaraarenduse metoodikate areng olnud orgaaniline protsess kus esmalt sooviti ära kasutada teiste sektorite projektijuhtimise kogemust ja nende protsesse, mida seejärel üritati kohandada tarkvaraarendusele sobivaks. Kohandatud traditsioonilisest koskmudeli metoodikast eristus iteratiivne ning inkrementaalne tarkvaraarenduse metoodika, mis esmalt oli suunatud kitsalt kosmosetööstuse ebamäärasuste lahendamisele, kuid mille sobivus teatava määramatusega toimetulekul oli tõukeks tänapäevaste agiilsete tarkvaraarenduse metoodikate tekkimisele. Agiilsetes metoodikates on olulisel kohal pidev muudatustest

tulenevate riskide maandamine läbi tiheda tarkvara tarnimise ning klientide pidev kaasatus tarkvaraprojekti vältel.

Tarkvaararenduse metoodikate kasutust hinnates selgus täiendavalt, et organisatsioonidel on tarkvaraarenduse metoodikate juurutamisel kasu tarkvaraarenduse küpsusmudelitest, mis võimaldavad kaardistada puudujääke organisatsiooni tarkvaraarenduse protsessides, juhendavad kasutama paremaid tarkvaraarenduse praktikaid ja seeläbi suurendavad organisatsiooni konkurentsieelist. Küpsusmudelite analüüsimisel osutus, et paljude mudelite väljatöötamisel on piirdutud puhtalt teoreetilise lähenemisega, praktilist rakendatavust pole sageli kontrollitud ning küpsusmudelite omavaheline võrdlemine organisatsiooni vajadustele sobivaima selgitamiseks on keeruline mõõdikute vasturääkivuse tõttu. Seetõttu viidi läbi täiendav agiilse tarkvaraarenduse küpsusmudelite sobivuse analüüs VKE, avaliku sektori asutuse ning Eesti organisatsioonikultuuri aspektist.

2. KÜPSUSTASE EESTI IKT ORGANISATSIOONIDES

2.1. Sünteesitud küpsusmudel

Käesolevas lõputöös püstitatud uurimisküsimustele vastamiseks tuleb esmalt sünteesida ja kirjeldada agiilse tarkvaraarenduse küpsusmudel koos mõõdikute ning tasemetega. Seejärel on sünteesitud küpsusmudeli ja tulemuskaardi baasil võimalik hinnata agiilse tarkvaraarendusega tegelevate organisatsioonide küpsustaset läbi empiirilise uuringu.

Uuringu jaoks kasutatava küpsusmudeli sünteesimisel on tuginetud peatükis 1.7 esitatud küpsusmudelite rakendatavust analüüsivale tabelile (tabel 2). Kuna oma olemuselt keskenduvad Sidky (2007) ja Ozcan-Top (2015) koostatud küpsusmudelid kõigile agiilse tarkvaraarenduse põhimõtetele ning Seuffert (2009), Benefield (2010) ja Humble (2009) pigem tehnilisele täiuslikkusele, siis on sünteesimisel aluseks võetud eelkõige Sidky (2007) väljatöötatud mudel, millele on lisatud mõõdikute väärtuseid, ideid ning täiustusi teistest mudelitest.

Küpsusmudelite tasemete nimetused on tuletatud Sidky (2007) ja Ozcan-Topi (2015) mudelite tasemete nimetuste ühendamisel nende nimelise ning osalt sisulise sarnasuse tõttu:

- esmane – tarkvara arendamine toimub traditsioonilistel meetoditel (koskmudel), kasutusel on mõningad juhuslikud agiilse tarkvaraarenduse elemendid;
- uuritud – tarkvaraarenduses on esimestes projektides kasutusel iteratiivsed metoodikad, tarkvara tarnimise sagedus suureneb;
- efektiivne – arendusprotsessi tõhususe suurendamine, kvaliteetse töötava tarkvara väljatöötamiseni viivate agiilse tarkvaraarenduse praktikate tihe rakendamine;
- täiustatud – agiilsete tarkvaraarenduse põhimõtteid võetakse kui efektiivset ja elementaarset protsessijuhtimise vahendit muutustega toimetulekuks;
- püsiv agiilsus – agiilsed metoodikad on saavutanud organisatsioonis laia kõlapinna, organisatsioonis julgustatakse õppima, õpetama ning ennast pidevalt täiendama.

Mõõdikute valikul on lähtutud agiilse tarkvaraarenduse manifesti 12 põhimõttest (peatükk 1.3). Selleks et kutsuda mõnda (tarkvaraarenduse) protsessi agiilseks on kõigi agiilsete põhimõtete tunnuste esinemine analüüsitavas protsessis hädavajalik. Sidky *et al.*

(2007, lk 205) hinnangul on mitmeid põhimõtteid võimalik omavahel grupeerida ja kohendada ning seeläbi kirjeldada viis lõplikut mõõdikut mis katavad kõigi manifestis väljatoodud põhimõtete sisulise poole:

- pidev muutus kui lisaväärtus kliendile;
- planeerimine tarkvara tihedaks tarnimiseks;
- inimesed;
- tehniline täiuslikkus;
- klient kui lisaväärtus.

Küpsusmudeli mõõdikute igal tasemel on välja toodud üks kuni kolm aspekti ja väärtust, mis peavad organisatsioonis kõik esinema või olema täidetud, et lugeda küpsus konkreetsel tasemel saavutatuks. Küpsusmudelisse valitud aspektid ja väärtused järgivad kirjeldatud tasemete struktuuri meenutades astmelist püramiidi kus iga järgneva taseme tegevused on lähemal püsivale agiilsusele, kuid eeldavad eelneva taseme edukat läbimist. Sünteesitud küpsusmudel on toodud lisas 3.

2.2. Uuringu struktuur ja korraldus

Uuringu valimiks on nelja Eestis tegutseva organisatsiooni töötajad – kahe tarkvaraarendusega tegeleva ettevõtte töötajad (VKE, töötajate arv < 100) ning kahe erineva ministeeriumi IT-keskuse tarkvaraarendusega seotud töötajad. Kõigi valitud organisatsioonide puhul on vastajad valitud meeskondadest kus on vähemalt kaks tarkvaraarendusega tegelevat töötajat. Uuringusse valitud VKE-d on oma profiililt ja tarkvaraarenduse protsessidelt erinevad. Esimene ettevõte keskendub tarkvaraarenduse konsultatsiooni ja projektipõhise tellimustarkvara teenuse pakkumisele ning palju tarkvara valmib projektipõhiselt kliendiga koostöös. Sellise ettevõtte võimalik tugevus on tarkvaraarenduse metoodikate pidev optimeerimine ja efektiivsus. Teine ettevõte on seotud enda tarkvaratoote pideva arendamisega tugevalt reguleeritud valdkonnas. Regulatsioonidest tulenevalt vajavad kõik tarkvaraarenduse sammud pidevalt eraldi kontrollimist ning lisaks vähendab enda tarkvaratoode vajadust mahtuda rangetesse projekti tähtaegadesse ja seetõttu ei pruugi ettevõttes kasutusel olevad (tarkvaraarenduse) protsessid olla kõige optimaalsemad ega küpsemad. Avaliku sektori uuringusse kaasamine annab võimaluse hinnata kas ja kui palju erinevad tarkvaraarenduse protsessid

ning nende kvaliteet sektoripõhiselt. Lisaks võimaldab see mõõta kuidas mõjutab tarkvaraarenduse projekte ja valitavaid metoodikaid kliendi lähedus (klientideks üldjuhul vastava ministeeriumi teised osakonnad) ning projektide ranged eelarved ja tähtajad. Kõik uuringusse kaasatud organisatsioonid on enda hinnangul agiilse tarkvaraarenduse metoodikatega tuttavad ning kasutavad neid igapäevaselt.

Uuringu teostatakse kasutades ankeetküsitlust. Ankeetküsitluse küsimustik (lisa 4) tugineb käesoleva lõputöö peatükis 2.1 sünteesitud küpsusmudeli mõõdikutele ja individuaalsete tasemete aspektidele. Küsitlus viiakse läbi Google Forms keskkonna vahendusel ning viited küsimustikule edastatakse vastajatele elektrooniliste suhtluskanalite vahendusel. Küsimustikule vastamine on vastaja seisukohalt anonüümne, kuid uuringusse on lisatud kaks valikvastustega taustküsimust vastajaga seotud meeskonnaliikmete arvu ja aktiivsete tarkvaraprojektide arvu mõõtmiseks. Taustküsimused suunavad vastajat järgima agiilse tarkvaraarenduse põhimõtteid– liiga suur meeskond on tahes-tahtmata vähem paindlik ning peaks peegelduma ka madalamas küpsustasemes. Uuringu läbiviimisel on küsimustikust kasutusel neli identset koopiat mis võimaldab edastada igale kaasatud organisatsioonile enda isiklik küsimustik. Eraldamine välistab vastajapoolse tahtliku väärinfo sisestamise enda organisatsioonilise kuuluvuse osas ning samas võimaldab täpsemalt tulemuste analüüsis hinnata ja võrrelda erinevate organisatsioonide ning sektorite küpsustaset.

Uuringu ülesehituse ning küsimustiku küsimuste tüübi valikul on määravaks varem edukalt teostatud ning agiilset küpsusastet mõõtvad uuringud, mis on näidanud et küsitletavate enda poolt täidetavad küsimustikud on uuringu läbiviijale vähem ajakulukad võrreldes kvalitatiivsete uuringutega. Sarnaselt on Gren *et al.* enda 2015. aasta uuringu järeldustes leidnud, et ankeetküsitlus sobib hästi agiilse küpsustaseme kiireks hindamiseks, kuid mitte niivõrd ebapiisava juurutamise põhjuste väljaselgitamiseks, mis on sageli organisatsiooni eripäradega seotud ning vajab eeskätt kvalitatiivset lähenemist (lk 48).

Küsimustikus on kokku 33 kohustuslikku küsimust, millest valdav osa on esitatud mina vormis väidetena. Iga konkreetse küpsusmudeli mõõdiku kohta esitatavad väited on grupeeritud kokku ja kuvatakse vastajale seotud väidete plokina. Väidetele vastamisel peab vastaja hindama väitega nõustumist või mittenõustumist viiepallisel Likerti skaalal,

kus väärtust 1 tõlgendatakse küsimustikus läbivalt kui „tugev mittenõustumine väitega“ ning väärtust 5 kui „tugev nõustumine väitega“. Küsimuste esitamine mina vormis väidetena ja vastajapoolne tunnetuslik subjektiivne hindamine Likerti skaalal järgib Greni *et al.* 2015. aasta agiilse küpsusmudeli valideerimiseks teostatud uuringu struktuuri. Viiepallise Likerti skaala sobivuse küpsustasemete hindamiseks on kiitnud heaks ka De Bruin *et al.* oma küpsusmudeleid käsitlevas töös (De Bruin *et al.*, 2005, lk 8).

Mõõtmise lihtsustamiseks on mõõdikutega seotud väited küsimustikus esitatud mõõdikute piires kasvavate tasemetega. Samas ei kata küsimustik kõiki sünteesitud küpsusmudelis esitatud tasemeid ega aspekte. Kuna uuringusse valitud organisatsioonid tegelevad igapäevaselt tarkvaraarendusega, siis on loogiline uuringu seisukohalt eeldada, et uuringus osalevates organisatsioonides on olemas vähemalt minimaalselt defineeritud protsessid mille tulemusel valmiv tarkvara tarnitakse aeg-ajalt klientidele. Sellisel tasemel protsesside ning seotud tegevuste olemasolu kontrollimine eraldi küsimuse või väitena ei oma antud uuringu seisukohalt lisaväärtust ja koormaks liigselt küsimustikku. Seega on kõigil uuringusse kaasatud organisatsioonidel saavutatud vähemalt agiilse tarkvaraarenduse esimene küpsustase nelja mõõdiku puhul. Erandiks on mõõdiku „tehniline täiuslikkus“ tasemed mille puhul minimaalsete tarkvaraarenduse protsesside kasutus ei garanteeri edukat küpsustase 1 läbimist.

Lisaks küpsusmudeli mõõdikutega seotud väidetele on küsimustikus kaks mina vormis kontrollväidet, mis on järjestuses paigutatud vahetult vastavalt enne ja pärast mõõdikutega seonduvaid väiteid:

- „Minuga seotud organisatsioonis arendatakse tarkvara kasutades agiilse tarkvaraarenduse metoodikaid“ – vastaja subjektiivne hinnang organisatsioonis kasutatavate tarkvaraarenduse metoodikate kohta enne küsimustiku põhilise osas nägemist, tulemuste analüüsil on kõrgem väärtus parem;
- „Ma leidsin eelnevate väidete seast aspekte ja tegevusi, mida minuga seotud meeskonnas / organisatsioonis veel ei tehta, kuid mida peaks tegema“ – vastaja subjektiivne hinnang organisatsiooni puudujääkide osas olles eelnevalt lugenud ja hinnanud individuaalsete aspektide ja väärtuste esinemist organisatsioonis, tulemuste analüüsil on madalam väärtus parem.

Esimene kontrollväide kuvatakse küsimustikus vastajale enne küpsusmudeli väidete kuvamist selliselt, et vastaja ei saa näha ülejäänud küsimustiku sisu esmalt sellele kontrollväitele vastamata. Mõlemad kontrollväited on esitatud viiepallisel Likerti skaalal analoogselt teiste väidetega küsimustikus.

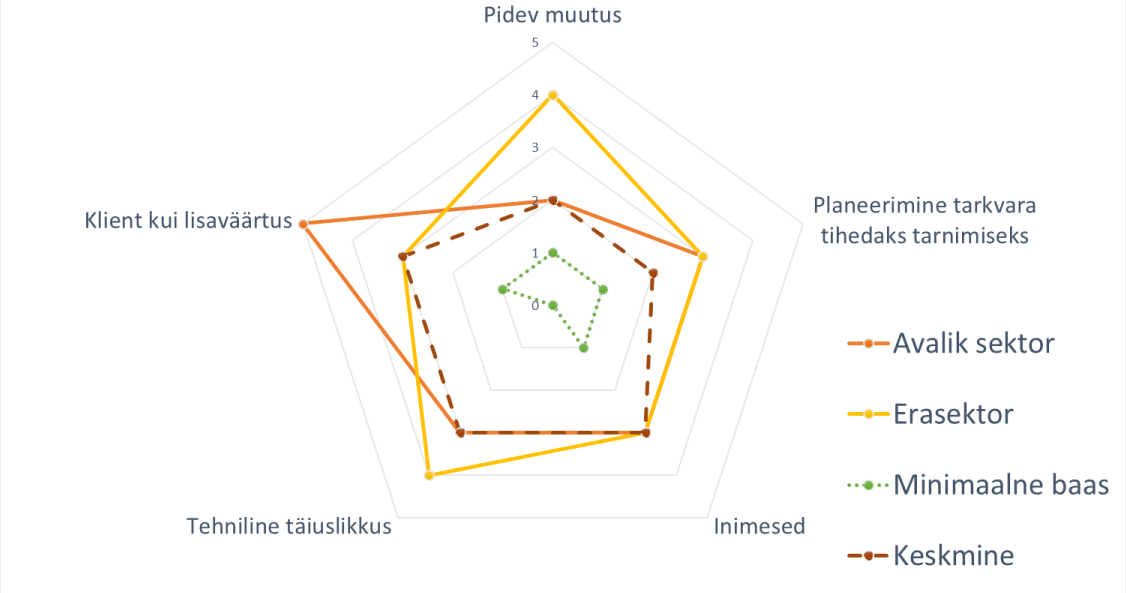
Uuringu tulemuste analüüsi seisukohalt eeldatakse, et osalevad organisatsioonid omavad positiivseid (> 3) väärtuseid paralleelselt mitmetes mõõdikute kategooriates ja erinevatel tasemetel. Seetõttu ei paku küsimustik ühest vastust küsimusele „mis on organisatsiooni agiilse tarkvaraarenduse küpsustase“, kuid annab võimaluse sünteesitud küpsusmudeli abil tuvastada puudujääke iga mõõdiku ning taseme kohta eraldi. Iga mõõdiku juures loetakse organisatsiooni küpsustasemeks viimane positiivne väärtus alates esialgsest tasemest pärast mida esineb küsimustikus sama mõõdiku piires neutraalse (3) või negatiivse (< 3) väärtusega vastus. Kui konkreetse mõõdiku tasemel on kaks või rohkem aspekti, siis peavad positiivselt olema hinnatud kõik vastava taseme aspektid. Isegi kui organisatsioonis esineb maksimaalse väärtusega mõõdikute vastuseid mõnest kõrgemast sama mõõdiku tasemest, siis neid taseme määramisel enam ei arvestata ning maksimaalseks tasemeks jääb viimane positiivsete väärtustega tase enne neutraalset või negatiivset väärtust.

2.3. Uuringu tulemused

Uuring viidi läbi märts 2021. aastal Google Forms keskkonna vahendusel vastavalt uuringu struktuurile. Kokku osales uuringus 26 vastajat – 10 vastajat avalikust sektorist ja 16 erasektorist. Täpne vastajate jagunemine individuaalsete organisatsioonide vahel ning uuringu koondatud tulemused on toodud lisas 5.

Keskmine vastaja töötab 5 – 10 liikmelises meeskonnas ning on paralleelselt seotud vähemalt kahe projektiga (keskmine 2,83). Avalikus sektoris on vastajatega seotud projektide arv üldisest keskmisest mõnevõrra suurem (keskmiselt 4,10), mis võib-olla tingitud valdavalt domineerivast olukorrast Eesti IKT tööjõuturul kus erasektor palkab vähegi võimekamad töötajad avalikust sektorist endale. See tõstab omakorda avaliku sektori töötajate töökoormust, kaasatud aktiivsete projektide arvu ja teoorias vähendab võimalusi ning aega agiilse tarkvaraarenduse parimate praktikate juurutamiseks.

„Minuga seotud organisatsioonis arendatakse tarkvara kasutades agiilse tarkvaraarenduse metoodikaid“ kontrollväite hindamisel leidsid paljud vastajad enne ülejäänud väidete nägemist, et nendega seotud organisatsioon juba kasutab agiilseid metoodikaid (üldine keskmine 4,29). Samas küsimustiku läbimisel hinnati kontrollväitele „Ma leidsin eelnevate väidete seast aspekte ja tegevusi, mida minuga seotud meeskonnas / organisatsioonis veel ei tehta, kuid mida peaks tegema“ vastuseks pigem nõustumist (üldine keskmine 3,64), mis eriti tugevalt oli esindatud just avalikus sektoris (keskmine 4,30). Lisaks avaldus täpsemal analüüsimisel, et nende kahe kontrollväite vahel on tugev korrelatsioon (kordaja r = -0,87) – mida madalamalt tunnetati organisatsiooni agiilsust enne väidete nägemist, sest tugevamalt tunnetati küsimustikule vastamisest lisaväärtust.

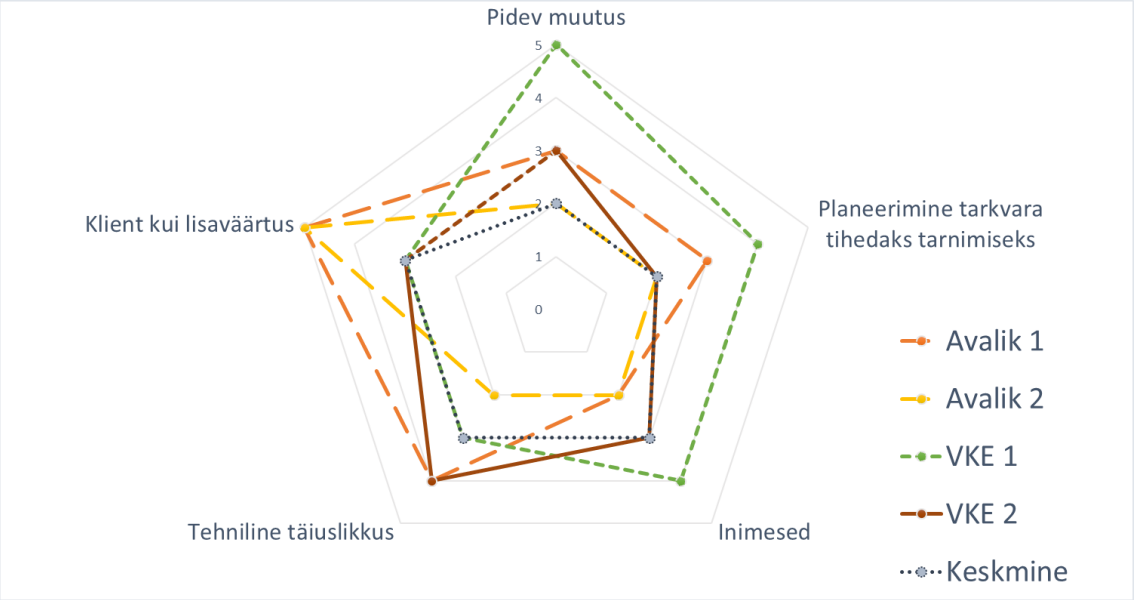


Joonis 5. Agiilse tarkvaraarenduse küpsustasemed koondatud võrdluses

Organisatsioonide küpsustasemete mõõtmisel arvutati esmalt küsimustikus esitatud väidete keskmised väärtused ühe mõõdikuteploki piires iga organisatsiooni kohta eraldi. Seejärel kasutati peatükis 2.2 välja toodud taseme hindamise protsessi maksimaalse taseme määramiseks vastava mõõdiku piires. Avaliku ja erasektori küpsustasemete tulemused koos keskmisega üle kõigi vastajate on välja toodud joonisel 5. Lisaks on joonisele lisatud „minimaalne baas“, mis tähistab iga tarkvaraarendusega tegeleva organisatsiooni alati eksisteerivat küpsustaset ja annab võimaluse võrrelda kui kaugele

minimaalsest tasemest on jõutud. Kõigi uuringusse kaasatud organisatsioonide võrdlused on välja toodud joonisel 6.

Avaliku sektori tulemuste hindamisel on selgelt näha puudujääke „pidev muutus kui lisaväärtus kliendile“ ja „tehnilise täiuslikkuse“ mõõdikute küpsuses. Need puudused avalduvad visuaalselt ka joonisel kus avaliku sektori joon on rohkem kaldu ühte graafiku nurka võrreldes erasektori rohkem balansseeritud joonega. Samas on avaliku sektori puhul väga tugevalt esindatud „klient kui lisaväärtus“. Kõik need tulemused näivad kinnitavat teooriat, et avaliku sektori organisatsioonides on ranged eelarved ja projektid piirava mõjuga mistõttu protsesside optimeerimine ning valitud tehnoloogiliste lahenduste asemel paremate otsimine on organisatsioonis madala prioriteediga tegevused mille jaoks pigem aega ei leita.



Joonis 6. Agiilse tarkvaraarenduse küpsustasemed organisatsioonide võrdluses

Kuna peaaegu eranditult hindasid kõik avaliku sektori vastajad enda koostööd klientidega väga heaks, siis võib seda lugeda püüdeks kompenseerida teiste mõõdikute puudujääke väga kliendikeskse lähenemise ja suhtumisega.

Küpsustasemete koondtulemuste võrdlused seda otseselt ei näita, kuid paljud vastajad oleks palju kõrgemal küpsustasemel kui lõplikult arvutatud küpsustaseme mõõtmise tulemus. Näiteks mõõdiku „tehniline täiuslikkus“ esimene väide „Minuga seotud

organisatsioonis on kasutusel kodeerimisstandardid“ osutus probleemseks kohaks paljudele vastajatele mõlemast sektorist kuigi samal ajal olid sama mõõdikuteploki kõrgema küpsustaseme väited väga tugevalt esindatud. Autori hinnangul võib see viidata kohati kaootilisele agiilsete tarkvarametoodika praktikate juurutamisele kus mõned väärtused ja aspektid leiavad organisatsioonis rakendamist, kuid on välditud ühtlaselt tugeva vundamendi rajamist. Selle põhjused võivad omakorda tuleneda organisatsiooni tööprotsessi eripäradest või tuginemisest ebapiisava kvaliteediga infoallikatele.

Ainult erasektori tulemuste hindamisel tuleb arvestada, et üks uuringusse kaastatud organisatsioon tegeleb reguleeritud tarkvaratoote arendamisega kus tarkvaraarenduse protsessid on täpselt defineeritud. Sellest tulenevalt viisid selle organisatsiooni vastajad mõõdiku „planeerimine tarkvara tihedaks tarnimiseks“ erasektori keskmist oluliselt alla ja mõjutasid negatiivselt „pideva muutuse kui lisaväärtus kliendile“ tulemust. Negatiivne mõju pole tulemuste analüüsi seisukohalt üllatav – mõlemad eelpool nimetatud mõõdikud on seotud tarkvaraarenduse protsesside pideva optimeerimisega ning võimalikult tiheda tarnimisega ja seega lähevad nad mõlemad reguleeritud tarkvaraarenduse protsessiga vastuollu. Erasektori tulemuste analüüsi suurimaks avastuseks võib lugeda kliendiga koostöö ühtlaselt nõrka taset. Kuna erasektoris on konkurents tihe, siis üks võimalik meetod konkurentsieelise saavutamiseks on näiteks kliendikesksus ning seetõttu on ootamatu avastada sedavõrd passiivset klientide kaasamist.

Hoolimata hoolikalt valitud ja võimalikult erineva profiiliga organisatsioonide uuringusse kaasamisest on autori arvates teostatud uuringu suurimaks puudujäägiks valimi väiksus nii kaasatud organisatsioonide arvu kui vastajate arvu osas. Sedavõrd väikese valimi puhul on individuaalsetel negatiivselt meelestatud vastajatel lihtsam mõjutada kogu sektori tulemust. Sellest hoolimata võimaldas uuring saada positiivse vastuse esitatud uurimisküsimustele. Ühest küljest aitas läbiviimiseks koostatud küsimustik edukalt valideerida nii sünteesitud küpsusmudelit kui küsimustikku ennast ning teisalt pakkus tulemuste analüüs võimalust mõõta erinevate organisatsioonide agiilse tarkvaraarenduse küpsustaset. Uuringu lisaväärtuseks saab lugeda eriti avaliku sektori tugevat nõustumist väitega „Ma leidsin eelnevate väidete seast aspekte ja tegevusi, mida minuga seotud meeskonnas / organisatsioonis veel ei tehta, kuid mida peaks tegema“, mis loob autori hinnangul küsimustikule vastaja teadlikkuse tõstmise läbi soodsad

eeltingimused organisatsioonidele kes soovivad juurutada agiilse tarkvaraarenduse parimaid praktikaid. Autor arvates annab korduvalt sarnase uuringu teostamine regulaarsete intervallide järel üle pikema ajalise perioodi koos kvalitatiivse intervjuuga võimaluse mõõta erinevate organisatsioonide küpsustaseme arengut ajas ning leida paremaid põhjendusi kitsaskohtadele.

2.4. Järeldused ja soovitused

Ankeetküsitluse tulemuste sisuline analüüs toob välja mitmed kitsaskohad mis lähtuvad ühest konkreetsest sektorist või organisatsioonist, kuid mille põhjal koostatud soovitused on laiendatavad kõigile tarkvaraarendusega tegelevatele organisatsioonidele. Järelduste tegemisel ning võimalike soovituste kirjeldamisel juhindub tugineb autor nii enda kogemusele kui varasemalt teaduskirjanduses avaldatud agiilse tarkvaraarenduse parimatele praktikatele. Ülevaatlik järelduste ja soovituste nimekiri on välja toodud tabelis 3. Muutuse keerukuse puhul arvestab autor võimaliku soovituse skoobiga (projekt / meeskond / organisatsioon) ning lisaks hindab võimalikku ajalist kulu.

Tabel 3. Uuringu järelduste ja soovituste ülevaatlik loetelu

Leid	Järeldus	Soovitused	Muutuse keerukus
Ühtlaselt madal küpsustase üle kõikide mõõdikute mõnede organisatsioonide puhul	Võimalik töötajate ülekoormatus ja iganenud tööprotsessid, pole aega tegeleda küpsuse suurendamisega	Anda töötajatele aega õppimiseks, kohendada organisatsiooni võtmenäitajaid	Keeruline ja aeganõudev, organisatsioon peab kohendama enda strateegiaid
Ebaühtlased mõõdikute väärtused ning aspektid – ühtlaselt tugev agiilne vundament puudu, kuid kõrgemad tasemed tugevalt esindatud	Kaootiline parimate praktikate juurutamine, võimalik tuginemine ebapiisava kvaliteediga allikatele	Vähendada auke agiilses vundamendis, tugineda parimate praktikate valikul mitmetele allikatele (praktikud ja teadus)	Keerukus sõltub aukude arvust vundamendis, potentsiaalselt aeganõudev, organisatsioon võib vajada enda strateegia kohendamist
Madal kliendikesksuse tase (erasektori „klient kui lisaväärtus“ mõõdiku küpsustase)	Kliendid füüsiliselt kaugel, tööprotsessid iganenud, ranged rollid kus üks konkreetne töötaja	Kaasata klienti sõltumatult füüsilisest kaugusest läbi tänapäevaste IKT lahenduste,	Keskmine keerukus, potentsiaalselt aeganõudev, võib vajada

	suhtleb kliendiga („telefon“)	väärtustada ja julgustada meeskondades klientidega suhtlemist	tööprotsesside ülevaatamist ja rangete rollide vähendamist
Koodistandardite puudumine („tehniline täiuslikkus“ mõõdiku esimene küpsustase)	Kaootiline parimate praktikate juurutamine, töötajate võimekusse liigne panustamine	Defineerida ja juurutada kasvõi osaliselt koodistandardid	Väike, võib vajada organisatsiooni-sisest kokkulepet mis võib ajakulu pikendada
Keskmine tarkvaraprojektide arv üle kahe, avalikus sektoris üle nelja („keskmine projektide arv“, lisa 5)	Võimalik tööjõupuudus pidurdab töökoormuse ühtlast jagamist rohkemate töötajate vahel, projektid ebamääraste tähtaegadega ning jäävad „venima“	Võimalusel vähendada projektide arvu maksimaalselt kahele, et vältida töötajate läbipõlemist ja liigset stressi	Keskmine keerukus, potentsiaalselt aeganõudev, organisatsioon võib vajada enda strateegia kohendamist

allikas: autori koostatud

Autori arvates on kõige suuremaks probleemiks ühtlane madal küpsustase mitmetes mõõdikutes mis võib olla seotud teadmatusega, et püsiva agiilsuse küpsustaseme saavutamiseks on vaja eesmärgipäraselt investeerida tegevustesse ning aega. On vaja leida aega, et tegeleda süsteemselt organisatsioonisiseste protsesside pideva analüüsimise ja optimeerimisega ning on vaja anda aega töötajatele, et nad saaksid muude töökohustuste kõrval end aeg-ajalt täiendada ja teistele enda teadmisi edastada. Kui analüüsida lisas 5 väljatoodud tabelis keskmist projektide arvu ja vastava organisatsiooni küpsustasemeid, siis on näha seost kus kõige rohkemate projektidega organisatsioonis on küpsustasemed kõige madalamad. Autori hinnangul on tõenäoline, et sellises organisatsioonis kulub töötajate aeg pigem projektide vahel orienteerumisele ning seetõttu tegeletakse tarkvaraarenduse küpsustaseme tõstmisega nii meeskonnas kui organisatsioonis ebapiisavalt. Seega soovitab autor ükskõik millisel tarkvaraarendusega tegeleval organisatsioonil võtta organisatsiooni strateegias üheks võtmenäitajaks (KPI – *Key Performance Indicator*) teadlikult agiilse tarkvaraarenduse küpsustaseme tõstmise. Läbivalt püsiva agiilsuse saavutamine on igas organisatsioonis erineva keerukusega ning selleni jõudmine on võrreldav iteratiivse ja inkrementaalse tarkvaraarenduse projektiga, kus mitmete iteratsioonide tulemusel saavutatakse töötav lahendus. Sealjuures tuleb arvestada, et küpsus agiilses tarkvaraarenduses pole lõplik väärtus – tehnoloogilised

arengud ning uued tarkvaraarenduse metoodikate parimad praktikad võivad oluliselt muuta organisatsiooni arusaama optimeeritud protsessidest või püsivast agiilsusest. Seetõttu võib organisatsiooni küpsustase ajutiselt isegi langeda ja vajada küpsustaseme tõstmiseks vajalike tegevuste ülevaatamist ning kordamist. Seega ei ole soovitatud strateegilise võtmenäitaja lõpptulemus kiiresti saavutatav ega saa olla määratud ajaliselt lühikese perspektiiviga, kuid suurendab teiste organisatsioonide ees konkurentsieelist alates esimestest planeeritud tegevustest. Käesoleva lõputöö uuringus osalenud avaliku sektori organisatsioonide puhul võib siinjuures üles kerkida küsimus sellise strateegia mõttekuses – nad ei pea otseselt kellegagi konkureerima, sest nende garanteeritud klient on alati ministeerium. Küll aga leiab autor, et kõik organisatsioonid konkureerivad tööjõuturul ja mõnikord võib potentsiaalse töötaja jaoks saada otsustavaks teguriks võimalus õppida ning proovida küpsemaid tarkvaraarenduse metoodikaid. Lisaks aitab uuenduslikkus lõhkuda levinud arusaamu avaliku sektori iganenud lähenemisest.

Teise olulise kitsaskohana saab välja tuua ühtlase tugeva agiilse vundamendi puudumise kõigi organisatsioonide puhul. Praktikas kajastub see ebaühtlaselt juurutatud väärtustes kus kõrgema küpsustaseme väärtused ja tegevused on ühe mõõdiku piires rohkem kasutusel kui madalama taseme omad. Tabelis 3 kirjeldatud koodistandardite puudumine on üks näide auklikust vundamendist. Kui tarkvaraprojekti valmimist võrrelda ühiselt raamatu kirjutamisega, kus iga peatükki kirjutab erinev autor, siis peatükkides ühtse kirjutamisstiili (koodistandardi) kasutamine lihtsustab uute autorite (arendajate) lisamist projekti ning annab hilisemale lugejale arusaadavama pildi tervikust (projektist). Lõputöö autori hinnangul võivad sarnased augud vundamendis viia suuremate projektide puhul teoreetiliselt tarbetu ajakuluni. Samas saab eeldada, et kõrgemate küpsustasemete väärtuste tugevam esindatus tähendab teatavate tööprotsesside ning agiilsete tarkvaraarenduse praktikate eelnevat olemasolu organisatsioonis. Seega on organisatsioonis olemas platvorm mis peaks hõlbustama puudujääkide kiiret likvideerimist.

Kolmandaks puudujäägiks loeb autor madalat mõõdiku „klient kui lisaväärtust“ küpsustaset osades uuringus osalenud organisatsioonides eeldusel, et vastajad on termini „klient“ all mõistnud nendega seotud organisatsiooni kliente ja mitte lõpp-kasutajaid (*end user*). Kuivõrd tänapäeva kliendid on modernsete IKT töövahenditega kursis ja

konkurents IKT sektoris on väga tihe, siis on autori jaoks arusaamatu teadlik distantsi hoidmine kliendiga. Autori arvates on kliendikeskne lähenemine üks lihtsamaid viise konkurentsieelise saavutamiseks kui rahulolev klient soovitab tarkvaraarendusega tegelevat organisatsiooni oma tuttavatele. Lisaks pole klientide kaasamisel tarkvaraarenduse projekti ühtegi negatiivset tulemit kui tarkvaraprojekti arendamise käigus selgitatakse kliendile pidevalt tema tagasiside potentsiaalset mõju projekti eesmärkidele, eelarvele ning tähtaegadele. Samas on kliendid tänapäevaste töövahendite abil lihtsasti pidevalt kättesaadavad ja teavad tihti paremini lahendatava probleemi sisu ning ärireegleid. Seega võimaldab kliendiga sage konsulteerimine tarnida täpselt soovitud tarkvara. Autori ettepanek on vaadata üle ja vajadusel parendada organisatsiooni tööprotsesse klientide suuremaks kaasamiseks (koostöö). Vajadusel tuleks restruktureerida organisatsiooni, et poleks defineeritud rangeid rolle ja reegleid mis lubavad kliendiga suhtlemist üksnes üksikute töötajate grupil (nö „telefonimäng“). Täiendavalt soovitab autor teostada regulaarseid rahuolu küsitlusi näiteks NPS-i (*Net Promoter Score*) või IPA (*Importance-Performance analysis*) abil, et saada hinnangut kaasatusele klientidelt endalt.

Autori hinnangul on eelnevalt välja toodud soovitused rakendatavad kõigis organisatsioonides sõltumata probleemi enda päritolust. Kuigi mõne soovituse järgimine pole ajaliselt kiiresti saavutatav ning võib vajada küllaltki suurt mõttemaailma muutust või organisatsiooni strateegia ülevaatamist, siis tuleb lähtuda arusaamast, et iga kavandatud ja teostatud samm tõstab organisatsiooni küpsustaset ning viib lähemale püsiva agiilsuse saavutamisele. Organisatsiooni tippjuhtkonna veenmisel võib kasutada väidet, et kõik sarnased eesmärgipärased sammud kajastuvad suuremas töötajate ja klientide rahulolus ning võimaldavad organisatsioonil müüa enda oskusi ja teadmisi konkurentidest kallimalt ning teenida suuremaid kasumeid.

KOKKUVÕTE

Käesolevas lõputöös uuriti võimalusi agiilse tarkvaraarenduse küpsustaseme tõstmiseks tarkvaraarendusega tegelevate organisatsioonide näitel. Autori arvates õnnestus lõputöös püstitatud eesmärk – avaldatud tarkvaraarenduse küpsusmudelite baasil sünteesida agiilse tarkvaraarenduse jaoks sobiv küpsusmudel, tulemuskaart ja mõõdikud – saavutada ning lisaks formuleerida praktilised üldistatud ettepanekud agiilse tarkvaraarenduse küpsustaseme tõstmiseks.

Lõputöö teoreetilise baasi ja tarkvaraarenduse metoodikatest ülevaate koostamisel selgus, et agiilse tarkvaraarenduse metoodikad on traditsiooniliste metoodikate loogiliseks edasiarenguks sooviga vähendada tarkvaraarenduse projektide ebaõnnestumist. Agiilsete metoodikate puhul kasutatakse ära traditsioonilistes tarkvaraarenduse metoodikates esinenud parimaid praktikaid. Kuid agiilse tarkvaraarenduse mõiste ning agiilse tarkvaraarenduse manifestis väljatoodud põhimõtted jätavad väga palju ruumi tõlgenduseks ja seetõttu on üksteisega konkureerivaid agiilseid metoodikaid välja töötatud mitmeid. See omakorda on tekitanud organisatsioonidele väljakutseid agiilsete metoodikate juurutamisel. Puudujääke saab vähendada kasutades tarkvaraarenduse taseme hindamise küpsusmudeleid, mis samm-sammult suunaks organisatsiooni protsesse õiges suunas, kuid agiilse tarkvaraarenduse küpsusmudeleid on ainult teadusartiklites kirjeldatud kümneid mis täiendavalt raskendab õigete valikute tegemist. Keerukust lisab praktikute ja teoreetikute vähene koostöö tarkvaraarenduse küpsusmudelite väljatöötamisel.

Lõputöös hinnati olemasolevate agiilse tarkvaraarenduse küpsusmudelite rakendatavuse määra lähtudes kolmest parameetrist – rakendatavus väikeses- või keskmise suurusega ettevõttes, rakendatavus avaliku sektori asutuses ning sobivus geograafiliselt Eesti organisatsioonikultuuriga. Rakendatavuse määra hindamisel tugineti esmalt juba varasemalt teadusartiklites teostatud analüüsidele mis võimaldas vähendada võrreldavate küpsusmudelite hulka. Seejärel koostati võrdlustabel kus hinnati igat küpsusmudelit

individuaalselt eelpool toodud parameetrite alusel kolmepallisel skaalal. Kõrgema rakendatavuse määraga küpsusmudelite sisulisel võrdlemisel ilmnes, et mitmed küpsusmudelid omavad omavahel sisulist ühisosa. Ühisosade analüüsi kasutati seejärel sisendina lõputöö ühe eesmärgi osa – agiilse tarkvaraarenduse jaoks sobiva küpsusmudeli – sünteesimisel. Küpsusmudelite teoreetilise analüüsi käigus selgus lisaks, et tõhusaim meetod küpsustaseme mõõtmiseks küpsusmudeli baasil on ankeetküsitlus.

Sünteesitud küpsusmudeli baasil viidi läbi uuring tarkvaraarendusega tegelevate organisatsioonide küpsustaseme määramiseks. Tulemuste mitmekesistamiseks kaasati uuringusse neli Eestis tarkvaraarendusega tegelevat organisatsiooni võrdselt avalikust ja erasektorist. Teostatud uuring võimaldas määrata kõigi uuringusse kaasatud organisatsioonide agiilse tarkvaraarenduse küpsustasemed sünteesitud küpsusmudeli mõõdikute piires. Ankeetküsitluse tulemusi analüüsides ilmnes, et uuringusse kaasatud organisatsioonid erinesid üksteisest mitmetes valdkondades. Avaliku sektori organisatsioonid on pigem liigselt koormatud projektidega ning tehnoloogiliselt vähem küpsed, kuid samas kaasavad klienti rohkem enda arendusprotsessidesse. Erasektori tugevus on tehnoloogiline pagas ja paremad protsessid, kuid vähem kaasatakse klienti. Uuringu tulemusel selgus ka, et mitmetel organisatsioonidel on augud agiilse küpsusmudeli esimestel tasanditel ning agiilseid põhimõtteid on juurutatud ebaühtlaselt. Ebaühtlane küpsustaseme mõõdikute jaotus ning sektorite erinevused võimaldasid autoril kirjeldada üldistatud meetmed küpsustaseme tõstmiseks sõltumatult organisatsioonist.

Lõputöös teostatud uuring on autorile teadaolevalt esimene sarnane uuring Eestis, mis on üritanud määrata agiilse tarkvaraarenduse küpsustaset Eesti organisatsioonides. Autori hinnangul on uuringu tarbeks sünteesitud küpsusmudel ja uuringu tulemusel tehtud praktilised ettepanekud pikemas perspektiivis abiks kõigile tarkvaraarendusega tegelevatele organisatsioonidele. Hoolimata uuringu soovitust väiksemast vastajate valimist loeb autor lõputöö eesmärgi saavutatuks ning leiab, et käesolev lõputöö on hea platvorm edasiste põhjalikemate sarnaste uuringute teostamiseks.

VIIDATUD ALLIKAD

Andersen, K. N., Lee, J., Mettler, T., & Moon, M. J. (2020). Ten Misunderstandings about Maturity Models. *The 21st Annual International Conference on Digital Government Research,* 261–266. doi:https://doi.org/10.1145/3396956.3396980

Awad, M. A. (2005). A comparison between agile and traditional software development methodologies. *University of Western Australia, 30.*

Ayed, H., Vanderose, B., & Habra, N. (2017). Agile cultural challenges in Europe and Asia: insights from practitioners. *2017 IEEE/ACM 39th International Conference on Software Engineering: Software Engineering in Practice Track (ICSE-SEIP),* 153–162. doi:10.1109/ICSE-SEIP.2017.33

Baran, B. E., & Woznyi, H. M. (2020). Managing VUCA: The human dynamics of agility. *Organizational Dynamics, 100787, 2020*. doi:https://doi.org/10.1016/j.orgdyn.2020.100787

Barden, E. (2017). Don't Go Chasing Waterfalls. *NZ Business + Management*, 31(8), M23 – M26.

Bauman, Z. (2005). *Liquid life.* Polity. Loetud aadressil https://www.academia.edu/download/40418957/_Zygmunt_Bauman__Liquid_Times_Living_in_an_Age_ofBookZZ.org.pdf

Beck, K., Beedle, M., Bennekum, A., Cockburn, A., Cunningham, W., Fowler, M., Grenning, J., Highsmith, J., Hunt, A., Jeffries, R., Kern, J., Marick, B., Martin, R. C., Mellor, S., Schwaber, K., Sutherland, J., &Thomas, D. (2001). Agiilse tarkvaraarenduse manifest. Loetud aadressil http://agilemanifesto.org/iso/et/manifesto.html

Becker, J., Knackstedt, R., & Pöppelbuß, J. (2009). Developing Maturity Models for IT Management – A Procedure Model and its Application. *Business & Information Systems Engineering*, *1*(3), 213–222. doi:10.1007/11576-009-0167-9

Benefield, R. (2010). Seven Dimensions of Agile Maturity in the Global Enterprise: A Case Study. *2010 43rd Hawaii International Conference on System Sciences System Sciences (HICSS)*, 1–7. doi:10.1109/HICSS.2010.337

Boehm, B. W. (1988). A spiral model of software development and enhancement. *Computer*, *21*(5), 61–72. doi:10.1109/2.59

De Bruin, T., Rosemann, M., Freeze, R., & Kaulkarni, U. (2005). Understanding the main phases of developing a maturity assessment model. *Australasian Conference on Information Systems (ACIS):. Australasian Chapter of the Association for Information Systems*, 8–19.

Gren, L., Torkar, R., & Feldt, R. (2015). The prospects of a quantitative measurement of agility: A validation study on an agile maturity model. *Journal of Systems and Software*, *107*, 38–49. doi:https://doi.org/10.1016/j.jss.2015.05.008

Henriques, V., & Tanner, M. (2017). A systematic literature review of agile and maturity model research.. *Interdisciplinary Journal of Information, Knowledge and Management*, *12*, 53–73.

Humble, J., & Russell, R. (2009). The Agile Maturity Model. Applied to Building and Releasing Software. Loetud aadressil https://info.thoughtworks.com/rs/thoughtworks2/images/agile_maturity_model.pdf

IPMA. (2015). *Individual Competence Baseline (4th Version).*

Johnson, J., & Mulder, H. (2015). Factors of Succes 2015. doi:10.13140/RG.2.2.28300.67208

Karvonen, T., Sharp, H., & Barroca, L. (2018). Enterprise Agility: Why Is Transformation so Hard? *Agile Processes in Software Engineering and Extreme Programming – Workshops. XP 2018*, 131–145. doi:10.1007/978-3-319-91602-6_9

Kuusinen, K., Gregory, P., Sharp, H., & Barroca, L. (2016). Strategies for doing Agile in a non-Agile Environment. *Proceedings of the 10th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement*, *5*, 1–6. doi:10.1145/2961111.2962623

Larman, C. (2004). *Agile and iterative development: a manager's guide*. Addison-Wesley Professional.

Leppänen, M. (2013). A Comparative Analysis of Agile Maturity Models. *Information Systems Development: Reflections, Challenges and New Directions*, 329–343. doi:https://doi.org/10.1007/978-1-4614-4951-5_27

Licht, N. (2015). *Scrum testimisprotsessi struktureerimine Ignite OÜ näitel*. (Magistritöö). Loetud aadressil https://digikogu.taltech.ee/testimine/et/Item/cf217ef3-8b01-4950-a641-0ec6d9fc7940

Meso, P., & Jain, R. (2006). Agile software development: adaptive systems principles and best practices. *Information systems management*, *23*(3), 19–30. doi:10.1201/1078.10580530/46108.23.3.20060601/93704.3

Minick, E., & Fredrick, J. (2014). Enterprise Continuous Integration Maturity Model. Loetud aadressil https://www.urbancode.com/resource/continuous-delivery-maturity-model/

Nurdiani, I., Börstler, J., Fricker, S., Petersen, K., & Chatzipetrou, P. (2019). Understanding the order of agile practice introduction: Comparing agile maturity models and practitioners' experience. *Journal of Systems and Software*, *156*, 1–20. doi:10.1016/j.jss.2019.05.035

Ozcan-Top, O., & Demirörs, O. (2015). A Reference Model for Software Agility Assessment: AgilityMod. *International Conference on Software Process Improvement and Capability Determination*, 145–158.

Patel, C., & Ramachandran, M. (2009). Agile maturity model (AMM): A Software Process Improvement framework for agile software development practices. *International Journal of Software Engineering, IJSE*, 2(1), 3–28.

Paulk, M. C. (2009). A History of the Capability Maturity Model for Software. *ASQ Software Quality Professional*, *12*(1), 5–19. doi:10.1.1.216.199

PMI, Inc. (2017). *A Guide to the Project Management Body of Knowledge (6th ed.)*.

Proulx, M. (2010). Yet Another Agile Maturity Model (AMM) – The 5 Levels of Maturity. Loetud aadressil https://danossia.wordpress.com/2010/07/12/yet-another-agile-maturity-model-the-5-levels-of-maturity/

Royce, W. W. (1970). Managing the development of large software systems. *In Proceedings of IEEE WESCON*, 1–9. Loetud aadressil http://www-scf.usc.edu/~csci201/lectures/Lecture11/royce1970.pdf

Schmitt, A., Theobald, S., & Diebold, P. (2019). Comparison of Agile Maturity Models. *Product-Focused Software Process Improvement (PROFES 2019)*, 661–671. doi:https://doi.org/10.1007/978-3-030-35333-9_52

Schoper, Y. G., Wald, A., Ingason, H. T., & Fridgeirsson, T. V. (2018). Projectification in Western economies: A comparative study of Germany, Norway and Iceland. *International Journal of Project Management*, 36(1), 71-82. doi:http://dx.doi.org/10.1016/j.ijproman.2017.07.008

Schneider, S. C., & De Meyer, A. (1991). Interpreting and responding to strategic issues: The impact of national culture. *Strategic management journal*, 12(4), 307-320.

Seuffert, M. (2009). Agile Karlskrona test. Loetud aadressil https://mayberg.se/media/downloads/karlskrona-test.pdf

Sidky, A., Arthur, J., & Bohner, S. (2007). A disciplined approach to adopting agile practices: the agile adoption framework. *Innovations in systems and software engineering*, *3*(3), 203-216.

Team, C. P. (2006). CMMI for Development, version 1.2. Loetud aadressil http://cc.ee.ntu.edu.tw/~farn/courses/SE/CMMI_DEV_V12.pdf

Tuncel, D., Körner, C., & Plösch, R. (2020). Comparison of Agile Maturity Models: Reflecting the Real Needs. *2020 46th Euromicro Conference on Software Engineering and Advanced Applications (SEAA)*, 51–58, doi:10.1109/SEAA51224.2020.00019

Turetken, O., Stojanov, I., & Trienekens, J. J. M. (2017). Assessing the adoption level of scaled agile development: a maturity model for Scaled Agile Framework. *Journal of Software-Evolution and Process, 29*(6). doi:10.1002/smr.1796

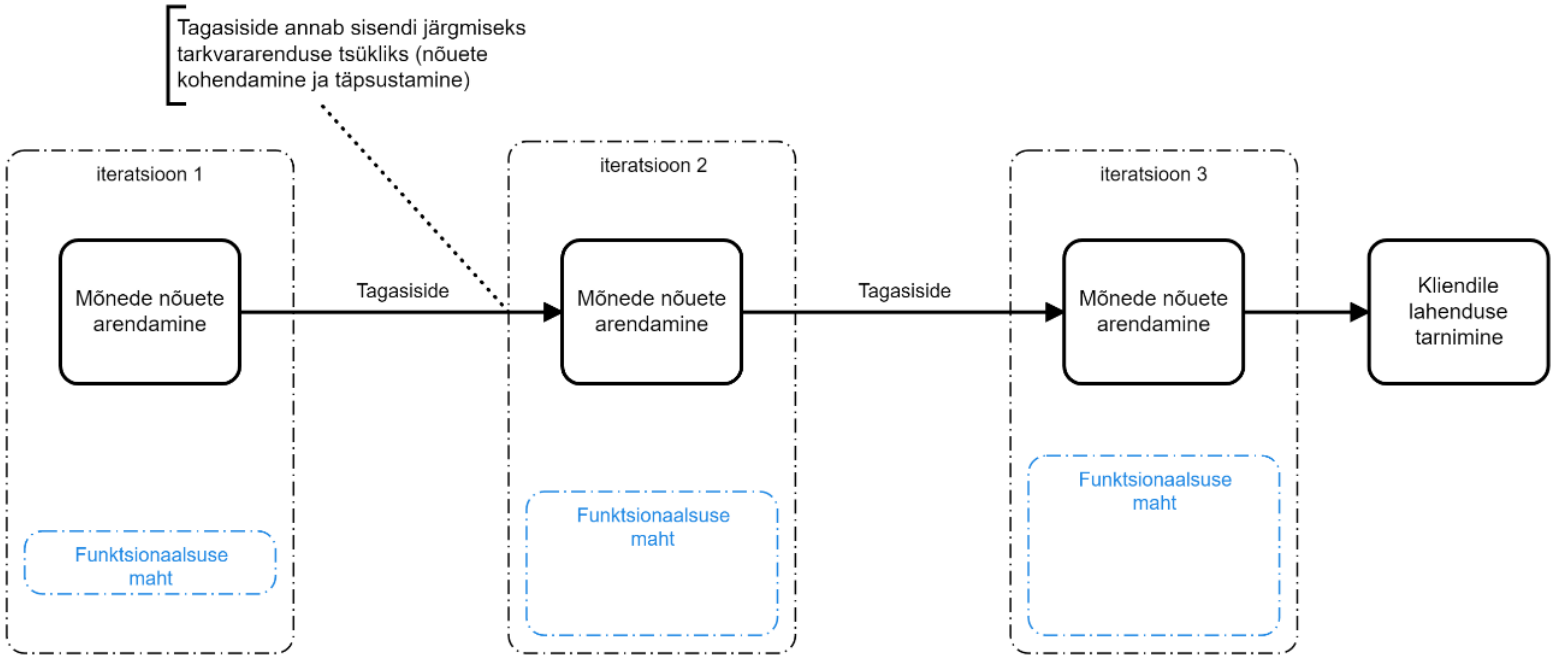
Woods, D. (2011). An Agile BI Maturity Model. Loetud aadressil https://www.forbes.com/sites/danwoods/2011/10/26/an-agile-bi-maturity-model

Yin, A., Figueiredo, S., & da Silva, M. M. (2011). Scrum maturity model. *Proceedings of the ICSEA, 20-29.*

Yürüm, O. R., & Demirörs, O. (2017). Agile Maturity Self-Assessment Surveys: A Case Study. *2017 43rd Euromicro Conference on Software Engineering and Advanced Applications (SEAA)*, 392–399. doi:10.1109/SEAA.2017.58

Lisa 1. Iteratiivse ja inkrementaalse tarkvaraarendusmetoodika etapid

Iteratsioonide eesmärk on välja arendada testitud tükk tarkvara mis sisaldab kõigest osalist funktsionaalsust võrreldes lõppeesmärgiga. Tavaliselt on kuni viimase iteratsioonini kogu valmiv lõpplahendus tarkvaraarendust teostava organisatsiooni käsutuses ning kliendile tarnitakse alles viimase iteratsiooni lõplik tulemus (Larman, 2004, lk 10).



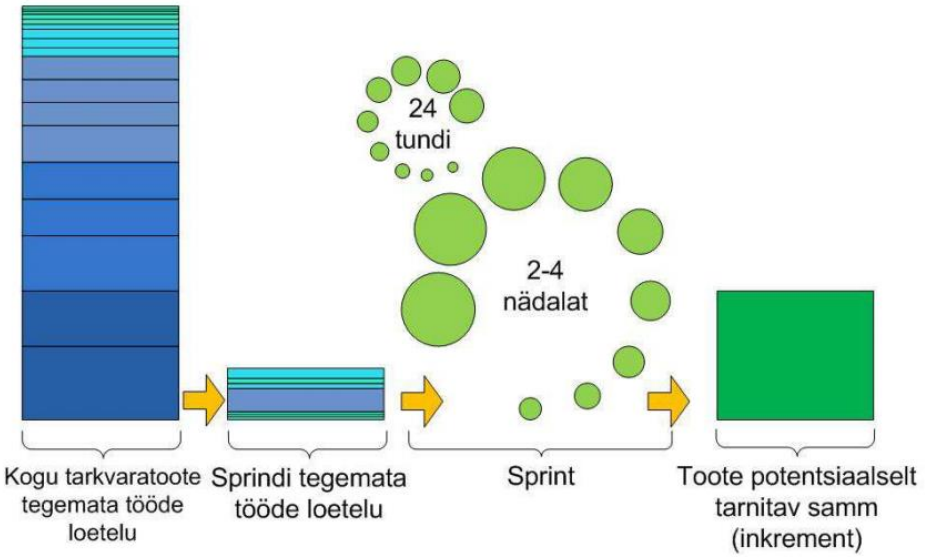
Joonis. Iteratiivne ja inkrementaalne mudel kolme iteratsiooni näitel, Larman, 2004 alusel

Lisa 2. Agiilse tarkvaraarendusmetoodika Scrum etapid

Kesksel kohal Scrumi tööprotsessides on sprint – kuni neljanädalane arendustsükkel osalise funktsionaalsusega toote tarnimiseks. Scrumis on defineeritud neli olulist artefakti millega iga sprindi jooksul arvestatakse:

- tegemata tööde loetelu – kõigi toote valmimiseks vajalike tööde nimekiri, prioriseeritult;
- sprindi tegemata tööde loetelu – kõigi sprindi käigus tehtavate tööde nimekiri, prioriseeritult;
- tarnimise graafik (*Release Burndown Chart)* – projekti progressi kujutav graafik;
- sprindi graafik (*Sprint Burndown Chart*) – sprindi progressi kujutav graafik (Yin *et al.*, 2011, lk 21).

Scrum ei määratle iga iteratsiooni käigus kasutatavat tarkvaraarenduse metoodikat, vaid pakub juhiseid ning tööriistu ettearvamatuse vältimiseks projektis tervikuna.



Joonis. Arendusmetoodika Scrum etapid lihtsustatult. Allikas: Licht, 2015, lk 17

Lisa 3. Agiilse tarkvaraarenduse sünteesitud küpsusmudel

	Mõõdikud				
Taseme nimetus	Pidev muutus kui lisaväärtus kliendile	Planeerimine tarkvara tihedaks tarnimiseks	Inimesed	Tehniline täiuslikkus	Klient kui lisaväärtus
Esialgne	On olemas protsessid tarkvaraarenduse juhtimiseks	Tarkvaraarenduses on kasutusel koskmudel	Töötajate tööülesanded tulevad juhtidelt	Tarnitav tarkvara ehitatakse käsitsi	Koostöö kliendiga minimaalne, tüüpiliselt tarkvaraprojekti alguses ning lõpus
	Olemasolevate protsesside analüüs, optimeerimisvõimaluste leidmine	Tarkvara tarnimine juhuslik		Organisatsioonis on kasutusel kodeerimisstandardid	
Uuritud	Protsessid fookusega liikuda varasematelt metoodikatelt agiilse tarkvaraarenduse metoodikatele	Esimesed agiilsete iteratsioonidega tarkvara tarned, juhuslikud kuid usaldusväärsed	Tõhus suhtlemine meeskonnaliikmete vahel	Tarnitav tarkvara ehitatakse automaatselt, on kasutusel automaatsed testid	Koostöö võimaldab kliendile pooliku / osalise funktsionaalsusega tarneid
		Planeerimine toimub mitmel organisatoorsel tasandil		Kasutusel tarkvara sisuliste muudatuste jälgimise töövahendid	
Efektiivne	Protsessid fookusega võimalikult tihedaks tarkvara tarnimiseks	Täiendavate töövahendite kasutamine igapäevase töö efektiivsuse suurendamiseks	Ise organiseeruvad meeskonnad	Tarkvara arendamisel analüüsitakse kriitiliselt olemasolevat lahendust ning vajadusel parandatakse omaalgatuslikult vigu	Klient on kursis enda rolliga tarnitava tarkvara kvaliteedi tõstmisel
		Tihe tarnimine tarkvaras olevate vigade parandamiseks	Tihe näost-näkku suhtlemine		
Efektiivne		Planeeritakse ärilise lisaväärtuse tarnimist, mitte tööülesandeid			

Täiustatud	Protsesside fookus kliendi tagasisidega arvestamisel	Tarkvara tarned tihedad (iteratsioonid 1 – 6 nädalat)	Meeskonnad planeerivad enda tööd enne arendusiteratsiooni kliendipoolsest tagasisidest lähtuvalt	Igapäevased lühikoosolekud projekti(de) progressi jälgimiseks	Klient vajadusel alati kättesaadav ning valmis tagasisidet andma
		Planeerimine arvestab jooksvalt tarkvara arendamise progressi		Arendamisel pannakse üritatakse panna end kliendi „kingadesse“	
Püsiv agiilsus	Maksimaalselt optimeeritud protsessid, protsesside fookus maksimaalset lisaväärtust pakkuva tarkvara tarnimisel	Meeskonnad kohtuvad tihedalt, et arutada võimaluse tarkvara tarnimise kiiruse suurendamiseks	Üle organisatsiooni inimestevahelise koostöö ning jagatud vastutuse rõhutamine	Meeskonnad kohtuvad tihedalt, et arutada tehnilise kvaliteedi suurendamiseks vajalikke meetodeid	Igapäevane tihe koostöö kliendiga kogu tarkvaraprojekti kestuse vältel
		Projekt planeeritud algusest lõpuni täielikult agiilsete põhimõtete järgi	Üle organisatsiooni pideva õppimise, õpetamise ning enda täiustamise toetamine	Tarkvara vigade parandamise aeg minimaalne	
				Kasutusel agiilse tarkvaraarenduse parimad praktikad (nt paarisprogrammeerimine)	
Taseme nimetus	Pidev muutus kui lisaväärtus kliendile	Planeerimine tihedaks tarkvara tarnimiseks	Inimesed	Tehniline täiuslikkus	Klient kui lisaväärtus

Lisa 4. Agiilse küpsustaseme mõõtmise küsimustik

Küsimustikus on küsimused valdavas enamuses püstitatud mina kujul väidetena, millele vastaja annab subjektiivse hinnangu viiepallisel Likerti skaalal. Kõik Likerti skaala küsimused on vahemikus 1 („tugev mittenõustumine“) kuni 5 („tugev nõustumine“). Erandiks on arvsisendi tüüpi küsimused mille puhul vastajal palutakse teha valik:

- „Minuga seotud meeskonnas töötab tarkvaraarendusega X arv inimesi“ küsimusel on valikvastused „0 – 5, „5 – 10“, „10 – 15“, „15 - ...“;
- „Ma olen paralleelselt seotud X arvu tarkvaraarenduse projektidega“ küsimusel on valikvastused „1“, „2“, „3“, „4“, „5“, „rohkem kui 5“.

Uuringu tulemuste analüüsi käigus tõlgendatakse valikvastus „rohkem kui 5“ väärtuseks 6. Likerti skaalal esitatud küsimuste puhul loetakse analüüsis positiivseteks tulemusteks vastused väärtustega 4 ja 5.

Mõõdik	Tase	Küsimus	Tüüp
		Minuga seotud meeskonnas töötab tarkvaraarendusega X arv inimesi	Arvsisend (valikvastus)
		Ma olen paralleelselt seotud X arvu tarkvaraarenduse projektidega	Arvsisend (valikvastus)
		Minuga seotud organisatsioonis arendatakse tarkvara kasutades agiilse tarkvaraarenduse metoodikaid	Likerti skaala
Pidev muutus kui lisaväärtus kliendile	2	Minuga seotud organisatsioonis toimub pidev protsesside analüüs ja protsesside optimeerimisvõimaluste otsimine	Likerti skaala
	3	Minuga seotud projektides on protsesside fookuses muuhulgas tarkvara võimalikult tihe tarnimine	Likerti skaala
	4	Minuga seotud projektides on protsesside fookuses muuhulgas kliendi tagasisidega pidev arvestamine	Likerti skaala
	5	Minuga seotud projektides on protsesside fookuses muuhulgas maksimaalset (ärilist) lisaväärtust pakkuva tarkvara tarnimine	Likerti skaala
Planeerimine tarkvara tihedaks tarnimiseks	2	Minuga seotud projektides kasutatakse tarkvara arendamisel agiilseid tarkvaraarenduse metoodikaid (XP, Scrum, ...)	Likerti skaala
	3	Minuga seotud projektide tulemil valminud tarkvara tarnitakse kliendile testimiseks vähemalt kord 3 nädala jooksul	Likerti skaala

	3	Minuga seotud meeskond kasutab täiendavaid töövahendeid enda töö efektiivsuse suurendamiseks (tarkvara arendamise lihtsustamiseks, tööülesannete jälgimiseks, ...)	Likerti skaala
	3	Minuga seotud meeskond tähtsustab kliendile lisaväärtust pakkuva tarkvara tarnimist	Likerti skaala
	4	Minuga seotud meeskond arvestab tööde planeerimisel jooksvalt projekti(de) progressi	Likerti skaala
	5	Minuga seotud organisatsioonis kohtuvad meeskonnad pidevalt, et arutada võimalusi tarkvara tihedamaks tarnimiseks	Likerti skaala
	5	Minuga seotud projektid on algusest lõpuni planeeritud agiilse tarkvaraarenduse põhimõtete järgi	Likerti skaala
Inimesed	2	Minuga seotud meeskonnas on tõhus omavaheline suhtlemine	Likerti skaala
	3	Minuga seotud projektides on tööülesanded (projekti piirides) tööd teostatava meeskonna enda valida	Likerti skaala
	3	Minuga seotud meeskonnas on tavapärane tihe näost-näkku suhtlemine töö efektiivsemaks muutmiseks	Likerti skaala
	4	Minuga seotud projektides on tööülesanded koostatud kliendi tagasisidest lähtuvalt	Likerti skaala
	5	Minuga seotud organisatsioonis on väärtustatud inimestevaheline koostöö ning jagatud vastutuse printsiip	Likerti skaala
	5	Minuga seotud organisatsioon toetab pidevat õppimist, enda arendamist ja vajadusel teistele töötajatele teadmiste edastamist	Likerti skaala
Tehniline täiuslikkus	1	Minuga seotud organisatsioonis on kasutusel kodeerimisstandardid	Likerti skaala
	2	Minuga seotud projektides ehitatakse tarkvara automaatselt	Likerti skaala
	2	Minuga seotud projektides on kasutusel tarkvara sisuliste muudatuste jälgimise töövahendid (Git, ...)	Likerti skaala
	3	Minuga seotud projektides analüüsitakse töö käigus võimalikke parenduskohtasid ning vajadusel parandatakse omaalgatuslikult vigu tarnitavas tarkvaras	Likerti skaala
	4	Minuga seotud projektides toimuvad igapäevased lühikoosolekud projekti(de) progressi jälgimiseks	Likerti skaala
	4	Minuga seotud projektides pannakse igapäevaseid tööülesandeid teostades end tihti kliendi „kingadesse“	Likerti skaala
	5	Minuga seotud organisatsioonis kohtuvad meeskonnad tihedalt, et arutada tehnilise kvaliteedi suurendamiseks vajalikke meetodeid	Likerti skaala
	5	Minuga seotud projektides on kasutusel agiilse tarkvaraarenduse parimad praktikad (nt paarisprogrammeerimine)	Likerti skaala

Kliendiga koostöö	2	Minuga seotud projektides on klient teadlik ning nõus ainult osalist funktsionaalsust sisaldava tarkvara tarnetega	Likerti skaala
	3	Minuga seotud projektides on klient teadlik enda rollist tarkvara kvaliteedi tõstmisel (nt on määranud endapoolse kontaktisiku)	Likerti skaala
	4	Minuga seotud projektides on kliendipoolne kontaktisik vajadusel alati kättesaadav	Likerti skaala
	5	Minuga seotud projektides on läbivalt tihe koostöö minuga seotud meeskonna ning klientide vahel	Likerti skaala
		Ma leidsin eelnevate väidete seast aspekte ja tegevusi, mida minuga seotud meeskonnas / organisatsioonis veel ei tehta, kuid mida peaks tegema	Likerti skaala

Lisa 5. Küpsustaseme ankeetküsitluse koondatud tulemused

Tulemuste tabelis on järgnevad seosed küsimustikus (lisa 4) esitatud küsimustele:

- tulp „Agiilsuse hinnang enne vastamist“ on vastus küsimusele „Minuga seotud organisatsioonis arendatakse tarkvara kasutades agiilse tarkvaraarenduse metoodikaid“, kõrgem väärtus tähistab vastaja suuremat kindlust agiilsete metoodikate kasutamise osas;
- tulp „Küsimustiku kasulikkus“ on vastus küsimusele „Ma leidsin eelnevate väidete seast aspekte ja tegevusi, mida minuga seotud meeskonnas / organisatsioonis veel ei tehta, kuid mida peaks tegema“, kõrgem väärtus tähistab vastaja suuremat kriitikat meeskonna ja / või organisatsiooni puudujääkidele tuginedes küsimustikus esitatud väidetele.

Maksimaalne väärtus kõigis Likerti skaalal hinnatavates tulpades on 5.

		Vastajate arv	Meeskonna-liikmete arv (mood)	Projektide arv (keskmine)	Agiilsuse hinnang enne vastamist (keskmine, kõrgem on parem)	Küsimustiku kasulikkus (keskmine, väiksem on parem)	Mõõdikute tasemed				
							Pidev muutus kui lisaväärtus kliendile	Planeerimine tarkvara tihedaks tarnimiseks	Inimesed	Tehniline täiuslikkus	Klient kui lisaväärtus
Avalik		10	5 – 10	4,10	4,00	4,30	2	3	3	3	5
	org 1	6	5 – 10	2,50	4,50	3,50	3	3	2	4	5
	org 2	4	0 – 5	5,00	3,75	4,75	2	2	2	2	5
Erasektor		16	5 – 10	2,06	4,45	3,25	4	3	3	4	3
	org 1	9	5 – 10	2,77	5,00	3,00	5	4	4	3	3
	org 2	7	5 – 10	1,71	4,14	3,42	3	2	3	4	3
Kokku		26	5 – 10	2,81	4,30	3,65	2	2	3	3	3

SUMMARY

POSSIBILITIES TO INCREASE THE MATURITY LEVEL OF AGILE SOFTWARE DEVELOPMENT FOR THE MANAGEMENT

Hannu Kikkas

In companies offering modern software solutions, the internal need to be constantly competitive has increased considerably, which entails the need to move forward from the rigid development methods developed previously to those that would allow for rapid adaptation to changed requirements during the development process and which would effectively reflect the processes taking place in the society. On the other hand, in the last decade, the introduction of integrated software solutions involving several sectors and industries has increased, and as a result the external pressure to improve the quality and speed of software development has increased. The appropriate solution is an agile software development method, which should cover both internal and external needs.

The aim of the thesis is to synthesize the maturity model, score card and to formulate recommendations of measurements suitable for agile software development.

When compiling an overview of the theoretical base and the methodology of software development, it became clear that the methods of agile software development are a logical development of traditional methods with the desire to reduce the failure of software development projects. The best practices used in traditional software development methods were used as an input for agile methods. However, the concept of agile software development and the principles outlined in the Manifesto for Agile Software Development leave a lot of room for interpretation and therefore several methodologies have been developed, which all are competing against each other. This, in turn, has created challenges for organizations in the implementation of agile methodologies. The

shortcomings can be reduced using maturity models to assess the maturity level of software development, which would step by step guide the organization's processes in the right direction, but the maturity models of agile software development are in the dozens described in the scientific articles alone, which further complicates the making of the right choices. The complexity is amplified by the lack of cooperation between practitioners and theorists during the development of maturity models for software development.

The thesis assessed the applicability rate of existing agile software development maturity models based on three parameters – applicability in small or medium-sized enterprises, applicability in the public sector institution and compatibility geographically with Estonian organizational culture. The feasibility rate was first based on the analyses already carried out in research articles, which allowed to reduce the eventual number of maturity models compared. Then a comparison table was drawn up where each maturity model was assessed individually using the parameters described previously on a three-fold scale. The deeper analysis of maturity models with a higher applicability rate revealed that several maturity models share common elements with each other. The analysis of shared elements was then used as an input for the synthesis of the maturity model suitable for agile software development. Additionally, the theoretical analysis of mature models revealed that the most effective method of measuring maturity level based on a maturity model is a questionnaire survey.

A study was conducted based on the synthesized maturity model to determine the maturity level of organizations involved in software development. In order to diversify the results, four software development organizations in Estonia from the public and private sectors were equally involved in the study. The study carried out enabled to determine the maturity levels of agile software development for all the organizations involved in the study within the limits of the synthesized maturity model measurements. Analyzing the results of the questionnaire revealed that the organizations involved in the survey differed from one another in a number of areas. Public sector organizations are rather overloaded with projects and less technologically mature, but at the same time they involve the client more in their development processes. The strength of the private sector is the technological competence and more effective processes, but they involve the client less. The study also revealed that several organizations have holes at the first levels of the agile

maturity model and the agile principles have been implemented unequally. The uneven distribution of maturity level indicators and the differences in sectors enabled the author to describe and formulate the generalized measures to increase maturity level independently of the organization.

The study carried out in the thesis is the first similar study in Estonia, which has attempted to measure the maturity level of agile software development in Estonian organizations. The author finds that the synthesized maturity model for the study and practical proposals made as a result of the study will help all organizations involved in software development in the long term. Despite the relatively small participation rate of the study, the author considers that the goal of the thesis has been achieved and the work done can be used as a good platform for further in-depth similar studies.

Mina, Hannu Kikkas,

1. annan Tartu Ülikoolile tasuta loa (lihtlitsentsi) minu loodud teose "Võimalused agiilse tarkvaraarenduse juhtimise küpsustaseme tõstmiseks", mille juhendaja on Arvi Kuura, reprodutseerimiseks eesmärgiga seda säilitada, sealhulgas lisada digitaalarhiivi DSpace kuni autoriõiguse kehtivuse lõppemiseni.
2. Annan Tartu Ülikoolile loa teha punktis 1 nimetatud teos üldsusele kättesaadavaks Tartu Ülikooli veebikeskkonna, sealhulgas digitaalarhiivi DSpace kaudu Creative Commonsi litsentsiga CC BY NC ND 3.0, mis lubab autorile viidates teost reprodutseerida, levitada ja üldsusele suunata ning keelab luua tuletatud teost ja kasutada teost ärieesmärgil, kuni autoriõiguse kehtivuse lõppemiseni.
3. Olen teadlik, et punktides 1 ja 2 nimetatud õigused jäävad alles ka autorile.
4. Kinnitan, et lihtlitsentsi andmisega ei riku ma teiste isikute intellektuaalomandi ega isikuandmete kaitse õigusaktidest tulenevaid õigusi.

Hannu Kikkas

19.05.2021