TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond
Tarkvarateaduse instituut

Siim Vene 132660IABMM

DEVOPS JUURUTAMINE SUURETTEVÕTTES

Magistritöö

Juhendaja: Truls Tuxen Ringkjob
MSc, Tallinna Tehnikaülikool

Konsultant Rain Rebane
MBA, Estonian Business School

Tallinn 2017

Autorideklaratsioon

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

Autor: Siim Vene

05.05.2017

Annotatsioon

Käesoleva töö raames uuritakse suurettevõtte näitel *DevOps* metoodika juurutamist suurorganisatsioonis.

DevOps (*Development and Operations*) on kultuur või praktika, mis käsitleb eneses nii arenduse kui halduse kompetentside koondamist omavahel kas läbi parema kommunikatsiooni või tihedalt seotud meeskondade. Protsesside automatiseerimise abil võimaldab tarkvara arendamist, testimist ning juurutamist teha kiiremini ja sagedamalt. Toetab teenuste arendamist vastavalt kiiresti muutuvatele turunõuetele, tehes seda odavamalt ning vähemate katkestustega.

Töös analüüsitakse Telia Eesti AS juba läbiviidud muudatust, tuuakse välja selle kitsaskohad ja kasutades kvalitatiivset uurimismetoodikat luuakse tegevuste nimekiri, mida olemasolevate protsesside ning organisatsioonistruktuuri muutmisel kasutades aitab *DevOop* juurutamist ettevõttes. Kaardistatakse peamised riskid ning ohukohad, millele peab muudatust läbi viies tähelepanu pöörama.

Kuna DevOps näol on tegemist populaarsust koguva metoodikaga, siis antud töös leitud järeldused on kasutatavad teiste suurettevõtete juures, kes plaanivad analoogset muudatust läbi viia.

Lõputöö on kirjutatud eesti keeles ning sisaldab teksti 65 leheküljel, 4 peatükki, 10 joonist, 8 tabelit.

Abstract

Implementing DevOps in large organizations

This thesis will explore the implementation aspects of DevOps for large organizations.

DevOps (Development and Operations) is a culture or practice covering both development and management competencies pooled together either through communication or a closely-related teams. Through the software development process automation enables to respond more quickly to changing market requirements.

Described in the work is the analysis of DevOps change in large telecom company, identifying the bottlenecks. Using qualitative research methodology author establishes a list of activities that supports implementing of DevOps. Mapped are the main risks and pitfalls that one must pay attention when going through the change.

As DevOps is a trending methodology, the present study findings can be used for other large companies, who are planning to carry out similar changes.

The thesis is in estonian language and contains 65 pages of text, 4 chapters, 10 figures, 8 tables.

Lühendite ja mõistete sõnastik

API	Ingl.k. *Application Programmable Interface* – programmiliides
CD	Ingl.k. *Continous Depoyment* – pidev juurutamine
CI	Ingl.k. *Continous Integration* – pidev integreerimine
CMDB	Ingl.k. *Configuration Management DataBase* - konfiguratsioonihalduse andmebaas
Evangelist	Töö kontekstis kui tehnoloogia saadik
HPSM	Ingl.k. *HP Service Manager* – teenusehalduse rakendus
IaaS	Ingl.k. *Infrastructure As A Service* – infrastruktuur teenusena
KPI	Ingl.k. *Key Performance Indicator* – võtmetähtsusega mõõdik
DevOps	Ingl.k. *Developement and Operations* – arendus ja haldus
LCM	Ingl.k. *Lifecycle Management* – elutsükli haldus
MVP	Ingl.k. *Minimum Viable Product* – minimaalselt kasutatav toode
PaaS	Ingl.k. *Platform As A Service* – platvorm kui teenus
RBAC	Ingl.k. *Role Based Access Control* – rollipõhine juurdepääsu kontroll
SOA	Ingl.k. *Service Oriented Architecture* – teenustepõhine arhitektuur
SSO	Ingl.k. *Single-Sign On* – erinevate tarkvarade juurdepääsukontolli meetod
TOGAF	Ingl.k. *The Open Goup Architecture Forum* – ettevõtte arhitektuuri raamistik

Sisukord

Jooniste loetelu

Tabelite loetelu

Sissejuhatus

Kaasaegse tarkvaraarenduse suunanäitajaks on paindlikkus ja kohanemisvõimekus kiiresti muutuvate turgude vajadustega. See on toetanud viimasel kümnendil paindlike arendusmetoodikate kasutuselevõttu kosemudeli tüüpi arendustsükli asemele [26]. Kui tarkvaraarenduses on toimunud paindliku arenduse muutused, siis olulise komponendina arendatud teenuseid teenindavat infrastruktuuri on see kohanemisvõimekus ees ootamas. Pilveteenuste tulekuga on muutunud suuresti see, kuidas võimaldatakse uusi platvormiteenuseid kasutusele võtta ning neid samamoodi dünaamiliselt kasutusest eemaldada, kuid klassikaline lõhe arendusmeeskondade ning haldusüksuste vahel on endiselt prevaleeriv enamikes ettevõtetes, kus arendustsüklid on raskendatud läbi omavahel vastanduvate eesmärkide (arendajatel tarnetsüklite sagedus, koodi kvaliteet, kiired muudatused ning haldajatel stabiilsed keskkonnad). Raskendatud on selliste paindlike metoodikate juurutamine suurettevõtetes, kus eksisteerivad jäikadena käsitletavad protsessid, juhtimisstruktuur, konfiguratsioonihaldus ja ettevõtte kultuur seda ei toeta. Suurettevõttes võib osutuda väljakutsuvaks skaleerida väikeses meeskonnas toimivat mudelit, sest projektide mastaapsusest lähtuvalt on osakondadevaheline kommunikatsioon keeruline. Üleminek ilma hoolika analüüsita võib osutuda kas vähetasuvaks või ei anna oodatud efekti.

Käesolev töö käsitleb ülalkirjeldatud halduse ning arenduse valdkondade vahelist konflikti vähendada üritava metoodika *DevOps* (*development* and *operations*) juurutamisega kaasnevaid tegevusi suurettevõttes.

Teema valiku põhjustas selle olulisus töös käsitletavas ettevõttes läbiviidud *DevOps* struktuurimuudatus, mille abil sooviti tõsta teenuste kvaliteeti, arenduse kiirust, kuluefektiivsust ning populaarsust tööandjana tööjõuturul, mis kokkuvõtvalt moodustavad võimaliku konkurentsieelise. Kuna läbiviidud muudatusega realiseerusid nii kaardistatud kui kaardistamata riskid, otsustas autor läheneda teemakäsitlusele süstemaatiliselt organisatsioonimuudatuse kontekstis. Probleem on laialdasem, kuna *DevOps* metoodikat tõlgendatakse erinevalt. Ei ole üheselt defineeritud, kas tegemist on

ainult organisatoorse, kultuurilise või tehnoloogilise vahendiga või mingi kombinatsiooniga mainitutest. See omakorda põhjustab situatsiooni, kus lahendatakse organisatsioonimuudatust juurutades kas valet probleemi või siis vale rõhuasetusega.

Esimeses peatükis tehakse ülevaade *DevOps* metoodikast, selle tekkeloost ja peamistest printsiipidest. Seatakse eesmärgiks koostada tegevuste nimekiri, mille abil lahendada metoodikaga kaasnevaid probleeme ning tingimusi. Määratakse uurimismetoodika, mille abil hakatakse küsimusele vastust otsima.

Teises peatükis kirjeldatakse Telia Eesti AS organisatsiooni, süsteemseid probleeme ning nende probleemide lahendamiseks kasutusele võetud *DevOps* metoodika juurutamise kava, ning sellega kaasnenud probleeme. Analüüsitakse võimalikke riske. Tuuakse välja muudatusega kaasnenud kitsaskohad.

Kolmandas peatükis luuakse struktureeritud tegevuskava organisatsioonimuudatuse läbi viimiseks suures ettevõttes. Käsitletakse protsesside ja vastutuste muutmiseks vajalikke tegevusi. Koostatakse soovituslik mõõdikute komplekt, millega selgitada välja muudatuse hetkeseisu ning juurutamise edukust.

Neljandas peatükis arutletakse *DevOps* fenomeni üle ning teostatakse tulemuste analüüs. Võrreldakse töös leitut teise suurorganisatsiooni muudatusega. Pakutakse välja võimalikud edasiarendused.

1 Ülevaade *DevOps* kontseptsioonist ja magistritöö metoodikast

Peatükis käsitletakse *DevOps* kontseptsiooni, võrreldakse seda paindlike tarkvara arendamise praktikatega, tuuakse esile selle metoodika peamised printsiibid ning kirjeldatakse selle juurutamisega kaasnevaid aspekte ettevõtte näitel. Püstitatakse töö uurimusküsimused ning kirjeldatakse uurimismetoodikat, mida kasutades küsimusi lahendama hakatakse.

1.1 *DevOps*'i olemus

DevOps on kaasaegne termin, mis tekkis kahe teineteisega seotud trendi koosmõjul. Esimest nimetatakse "paindlikuks süsteemi administreerimiseks", mis tekkis paindlike arendusmetoodikate rakendamisel süsteemide administreerimise juures. Teine trend on väärtuse mõistmine, mis tekib arendusmeeskondade ning haldusmeeskondade omavahelisest koostööst kõikide arenduse elutsükli etappide juures. [32]

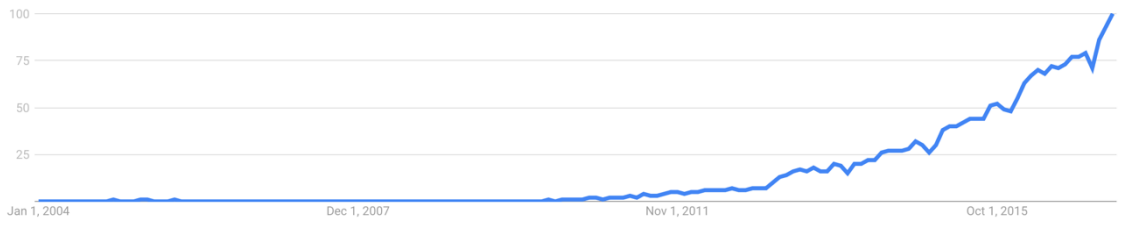
DevOps'i kirjeldatakse kui praktikat, kus halduse ja arenduse insenerid töötavad koos terve teenuse elutsükli jooksul, disaini faasist läbi arenduse kuni toodangukeskkonda paigaldamiseni. Lisaks käsitletakse *DevOps*'i ka kui arendustehnikate kasutamist halduse inseneride poolt nagu versioonihalduse ning testimise printsiipide rakendamine. [32]

Gartner defineerib *DevOps*'i kui perspektiivi, mis nõuab kultuurilist muudatust organisatsioonis ja fokusseerib kiirele IT teenuse tarnele läbi paindlike praktikate integreeritud lähenemise kontekstis. *DevOps* rõhutab inimesi ning kultuuri parendamaks koostööd erinevate IT huvigruppide nagu arenduse ja halduse üksuste vahel, kuid samas ka arhitektuuri ning infoturbega seotud osapooli arvestades. *DevOps* lahendused kasutavad tehnoloogiaid (eriti automatiseerimisvahendeid), mis elutsükli vaatest keskenduvad programmeeritavale ning dünaamilisele taristule. [11]

DevOps kui termin sai alguse 2009. aastal, mil Patrick Debois organiseeris Belgias konverentsi nimega *DevOps Days*. Peale konverentsi jätkus diskussioon suuresti Twitteris ning kuna Twitteris on kasutusel 140-tähemärgiline piirang, siis muudeti algset #devopsdays konverentsile viitamist kasutades #*devops* lühendit. [15]

Murrangupunktiks loetakse 2011. aasta Gartneri analüütiku Cameron Haight ettekannet, kus ta väitis aastaks 2015 *DevOps* arenevat pilveteenuste pakkujate nišistrateegiast peavoolu strateegiaks, leides kasutust 20% juhtiva 2000 globaalse organisatsiooni poolt. See sõnumit võeti tõsiselt ning kõik suuremad teenusepakkujad hakkasid *DevOps* kontseptsiooni oma müügisõnumitesse inkorporeerima. [15]

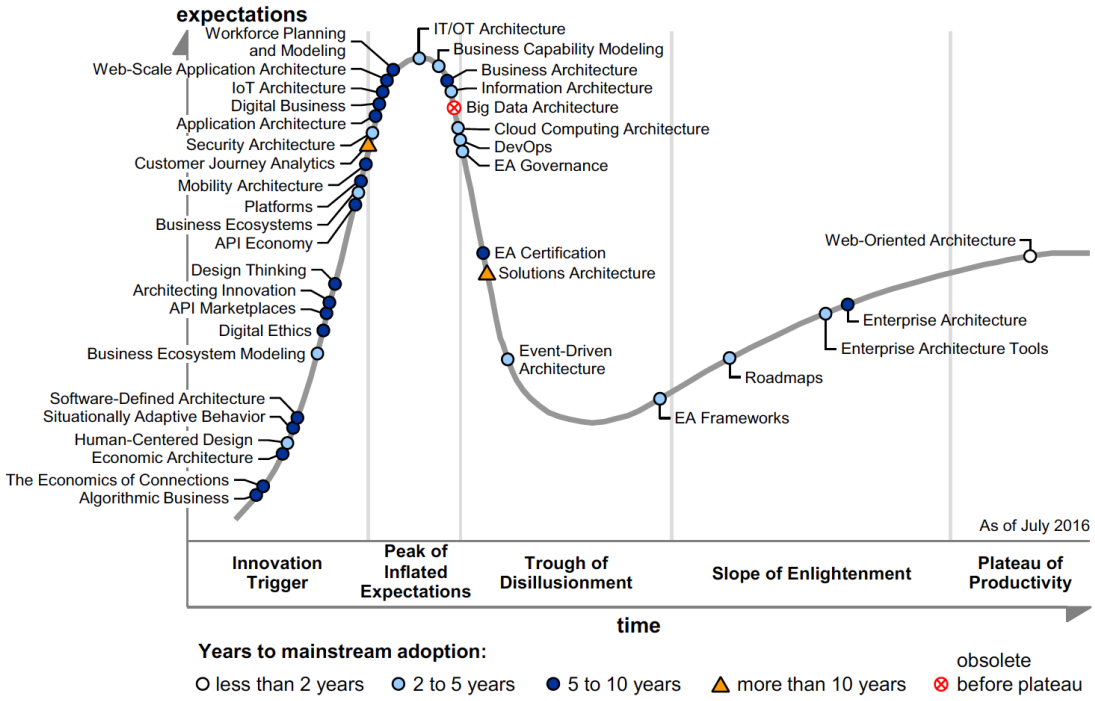
Google Trends statistika järgi on otsingumootoris termini kasutamise populaarsus endiselt tipus:



Joonis 1 Google Trends graafik sõnale "*devops*"

Ettevõtte arhitektuuri kontekstis Gartneri hinnangul on tegemist alles haibi tsükli kõvera tipus oleva lähenemisega, mis adapteeritakse peavoolu alles kahe kuni viie aasta pärast. [11]

Haibi tsükli kõver kujutab endast uute tehnoloogiate teket ning ettevõtete poolt adapteerimist. See koosneb Innovatsiooni Käivitajast, Kõrgendatud Ootuste Tipust, Pettumuse Orust, Valgustatuse Nõlvast ning Produktiivsuse Platoost. Lihtsustatult, innovaatilised trendid läbivad haibi kõvera ning alles siis, kui ootused on jõudnud realistlikele tasemetele ja ettevõtted on õppinud trendi kasutama nendele omases kontekstis, muutub trend alles tootlikkuse mõttes tasuvaks. [18]



Joonis 2 *DevOps* Gartneri haibitsükli kõveral [11]

Gartneri hinnangul puudub *DevOps*'il konkreetne standard või raamistik, nagu näiteks ITIL või CMM, mis loob tingimused liberaalsele subjekti interpretatsioonile. See omakorda tingib olukorra, kus metoodika juurutamine ning edu mõõtmine on raskendatud. [11]

DevOps metoodikat seostatakse peamiselt IT süsteemide pideva integratsiooni (*continuous integration*) ja pideva tarnega (*continuous delivery*) kui vahendina optimeerimaks töö voogu rakenduse elutsükli vältel arendusest toodangusse. [11]

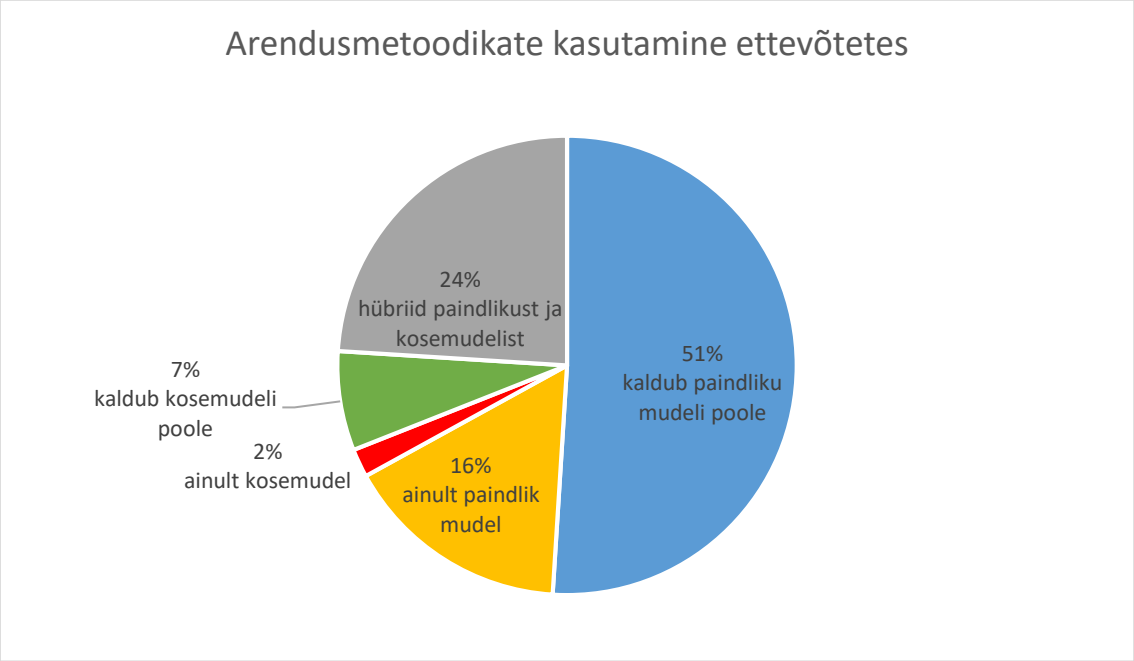
1.2 *DevOps* ja paindlik tarkvaraarendus

DevOps tugineb sarnastele printsiipidele paindliku tarkvaraarendusega, järgides "*The Agile Manifesto*" [5] kirjeldatud printsiipe, kus:

1. Protsesside ja töövahendite asemel eelistatakse indiviide ja suhtlust;
2. Põhjaliku dokumentatsiooni asemel eelistatakse toimivat tarkvara;
3. Lepinguliste läbirääkimiste asemel eelistatakse kliendisuhtlust;

4. Kinda plaani järgmisele eelistatakse muudatustele reageerimist.

Paindlikku tarkvaraarendust hakati kasutusele võtma 15 aastat tagasi algselt liikumisena, mis vastandus kosemudeli arenduspraktikatele. Peamiseks motivaatoriks kasutuselevõtjate seas oli veendumus, et sedasorti arendusmetoodika on rohkem kasutajakeskne ning parendab arendusmeeskondade omavahelist koostööd. TechBeacon 2016 aastal tehtud uuringu järgi, kus küsitleti 600 tarkvaraarendusega tegelevat professionaali, on paindlikud printsiibid kasutuses umbes 2/3 ettevõtetes. [26]



Joonis 3 Peamine arendusmeetod organisatsioonides HP uuringu kohaselt [26]

Kuigi nii paindlik tarkvaraarendus kui *DevOps* tuginevad sarnastele printsiipidele, on neil teatavaid erinevusi:

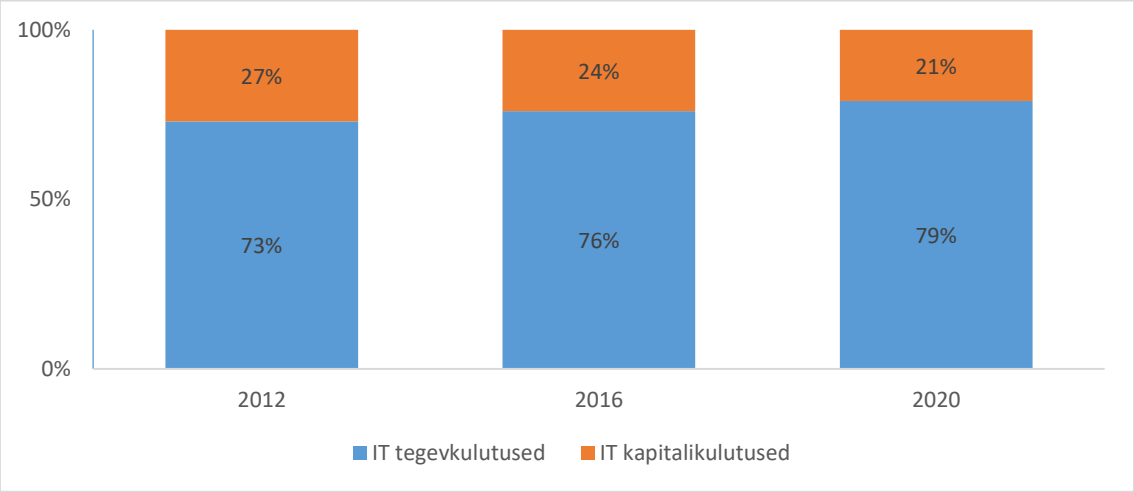
- Paindlik tarkvaraarendus keskendub peamiselt tarkvara arendusele. Kui tarkvara on arendatud ning tarnitud, siis formaalselt arendusmeeskond sellega enam ei tegele ning keskendub järgmisele iteratsioonile. *DevOps* seevastu keskendub tarnimiskõlbulikule tarkvarale ning selle võimalikult kindlale kasutuselevõtule. *DevOps* ei tugine otseselt paindlikule tarkvaraarendusele ning seda võib kasutada ka tarkvarade puhul, mis on arendatud kosemudeli metoodikat kasutades.

- Scrum metoodika järgi on meeskonnaliikmed võimelised tegema kõiki vajalikke töid projekti käigus. *DevOps* puhul seda ei eeldata - arendus ning haldusmeeskonna paralleelne eksisteerimine on täiesti võimalik, kus spetsialistid ei teosta teise üksuse tegevusi, vaid tuginevad oma tööd tehes efektiivsele ning sagedale kommunikatsioonile.

- Paindliku arendusmeeskonna toimimiseks ei ole mingeid eeldusi tööriistadele. Kogu kommunikatsioon võib toimuda e-kirjade ning märkmepaberite abil. *DevOps* sõltub suurel määral vahendite ning töölõikude automatiseeritusest. [17]

1.3 *DevOps* metoodika kasulikkus

Kaasaegsetes turutingimustes on ootused ettevõtete paindlikkusele reageerimaks kiiretele muudatusetele kasvanud. Pilveteenuste arenemisega on ettevõtetes muutunud infotehnoloogiale tehtavate kapitalikulutuste ning tegevkulutuste osakaal, kus kapitalikulutused kas ei kasva või vähenevad aasta-aastalt ning tegevkulud suurenevad sõltumata üldisest tehnoloogiaeelarve suurenemisest või vähenemisest. *CEB 2013-16 IT Budget Benchmark* uuringu kohaselt moodustavad aastaks 2020 tegevkulud 80% infotehnoloogiale tehtavatest kulutustest, kasvades aastas 1,4%. [1]



Joonis 4 IT tegev- ning kapitalikulude osakaal [1]

Võttes kasutusele automatiseeritud tarneprotsessid on võimalik suuremate tegevuskulude pealt säästa proportsionaalselt rohkem kui kapitalikulutustelt. *Puppet Labs* 2016 aasta aruandest tuleneb, et hästitoimivad ettevõtted on oluliselt efektiivsemad

võrreldes teiste ettevõtetega, juurutades koodi toodangukeskkondadesse 200 korda sagedamini, täidavad tellimusi 2,555 korda kiiremini, taastavad teenuste tööd 24 korda kiiremini ning kogevad muudatustest tingitud katkestusi kolm korda vähem. Samas kulutavad 22% vähem aega erakorralistele ja planeerimata tegevustele ning 29% rohkem aega uusi lisaväärtust loovatele tegevustele. [9]

Amazon.com arendus- ning testimisüksustes on *DevOps* metoodika juurutamine aidanud vähendada planeerimata katkestusi 75% ning katkestustele kulunud minutite hulka 90%. [10]

CA Technologies korraldatud küsitluse kohaselt 1425 ettevõtte seas leiti, et tänu *DevOps* metoodika juurutamisele vähenes rakenduste haldamisele ning parandamisele kulunud aeg 21%, arendamisele ning testimisele kuluv aeg vähenes 19% ning sama aja jooksul suurenes kasum 19%. [6]

2016 aastal Oulu ja Helsinki ülikoolides viidi läbi küsitlusel põhinev uuring [38], mille tulemusena kaardistati *DevOps* praktikate juurutamisega kaasnevad järgmised positiivsed tagajärjed:

- Rohkem juurutatud funktsionaalsust ning sagedasemad tarnetsüklid – automatiseeritud arendus-, testi- ning juurutusprotsessid võimaldavad ettevõtetel rohkem funktsionaalsust luua. Automatiseerimine aitab vähendada tarnetele kuluvat töömahtu, võimaldades tarnida nii sagedasti kui tarvis.
- Suurenenud kvaliteedikontrolli tase – automatiseeritud arendusprotsess aitab kaasa sellele, et iga muudatus on kontrollitud enne kasutuselevõttu. Kuna muudatusi kontrollitakse igas arendusetapis, avastatakse eksimusi sagedamini ja varem, mis tähendab, et tarkvaratoodetes on vähem vigu ning nende parandamine on odavam.
- Paranenud koostöö ning suhtlus – *DevOps* metoodika kasutamine soodustab arendus- ning haldusmeeskondade vahelist suhtlust, mis aitab lõhkuda müüre klassikaliste valdkonnapõhiste üksuste vahel. Suurenenud suhtlus ja koostöö omakorda soodustavad kogemuste ning teadmiste ülekandumist meeskondade vahel. Multifunktsionaalsed meeskonnad kasutavad paremini ära kompetentse, kuna laialdasem oskuste hulk on kättesaadaval.

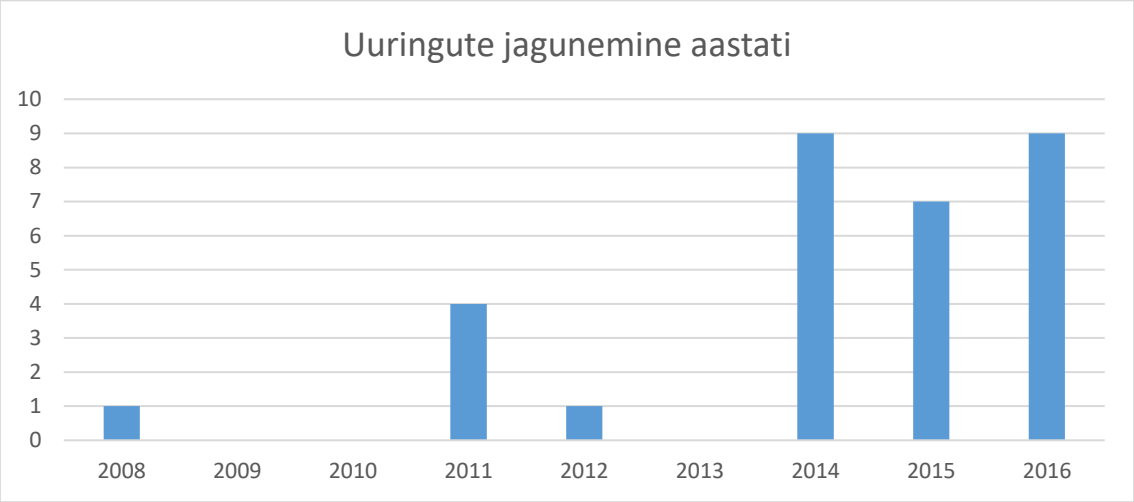
- Kõrgem moraal meeskondades – sagedaste tarnete tõttu väheneb spetsialistide stressitase võrreldes harvade kuid suurte tarnetega, kuna võimalike vigade arv uute funktsionaalsuste kasutuselevõtuga on väiksem.
- Kasutuselevõetud funktsionaalsuse parem nähtavus klientidele – uusi funktsionaalsusi saab klientidele kiirema ajaga kasutusele anda. Lisaks saavad arendusmeeskonnad kiiremini tagasisidet klientidelt, mis aitab toodete parendamisele kaasa. Samuti aitab see kaasa turunõuetele reageerimisele ning võimaldab erinevaid ideid kiirelt katsetada. [38]

Ülaltoodut arvesse võttes võib väita, et organisatsioonijuhtimises *DevOps* mudeli kasutuselevõtt toob endaga kaasa reaalseid ning mõõdetavaid kasusid.

1.4 *DevOps* akadeemilise uurimisobjektina

Arvestades *DevOps* termini populaarsust ning sellega seotud investeeringute hulka, on akadeemilisel tasemel uurimustöid suhteliselt vähe. Hetke kirjandus tugineb peamiselt fenomeni kirjeldamisele ning parimate praktikate jagamisele, mis tuginevad suuremal või vähemal määral autorite subjektiivsele kogemusele. Akadeemilisi töid antud teemal on vähe ning temaatikal puudub raamistik või ühtne definitsioon. Lahendusi käsitletakse läbi erinevate vaatenurkade ning seetõttu võivad need üksteisest erineda. [24]

Anand Srivatsav Amaradri ja Swetha Bindu Nutalapati oma magistritöös "*Continuous Integration, Deployment and Testing in DevOps Environment*" kaardistasid 31 uurimustööd, mis on avaldatud ajavahemikus 2008 kuni 2016. Uurimustööde hulgast 22 käsitlevad *DevOps* kasutamisega kaasnevaid kasusid, 7 uurimustööd puudutavad *DevOps* kasutamisega kaasnevaid väljakutseid ning 10 uurimustööd on kirjutatud *DevOps* praktikatest ning printsiipidest. [2]



Joonis 5 *DevOps* uurimustööde jagunemine aastati [2]

Töö kirjutamise hetkel kättesaadavad põhjalikumad uurimustööd on sarnase rõhuasetusega, keskendudes peamiselt *DevOps* metoodikat võimaldatavate või toetavate tarkvarade, tehnoloogiate ja protsesside kirjeldamisele, võrdlemisele ning juurutamisele. Organisatsiooni muudatuse vaatevinklist teemakäsitlus on olnud pealiskaudsem. Joonas Hamunen oma 2016. aasta magistritöös "*Challenges in Adopting a Devops Approach to Software Development and Operations*" käsitleb seda temaatikat neljal leheküljel, kuid peamine rõhuasetus on siiski arendus- ja haldusvahendite automatiseerimisel.

Tuginedes ülaltoodule on autor püsitanud järgmise uurimisküsimuse:

"Kuidas juhtida organisatsioonimuudatust *DevOps* metoodika kasutuselevõtmiseks?"

Vastused küsimusele peavad andma muudatuse läbiviimiseks vajalikud suunised sõltumata IT juhtimise raamistikest või juurutamise hetkel kasutusel olevast ettevõtte organisatoorsest struktuurist.

1.5 Uurimismetoodika valik

Metoodika valikul peab arvestama lahendatava probleemi olemust. Kaks levinumat uurimismetoodikat on kvantitatiivne ning kvalitatiivne uurimismetoodika[49]. Kvantitatiivne metoodika uurib numbrilisi andmeid, samas kvalitatiivne metoodika kasutab kirjeldavat lähenemist.

Töös püstitatud uurimisküsimuse lahendamiseks valis autor kvalitatiivse uurimismetoodika. Metoodika sobib antud teema käsitlemiseks, kuna võimaldab kasutada väikeseid valimeid, interpretatsioon põhineb kogutud andmete ning uurija tõlgenduse kombinatsioonil, andmete kogumine sisaldab näiteks intervjuude kasutamist, uurimisküsimust toetavate dokumentide analüüsi ning juhtumi uurimist.

Töös ei kajastata võimalikke rakendusi, tehnoloogiaid või teenuseid, mille abil saab rakendada *DevOps* printsiipe. Töös ei käsitleta põhjalikult erinevaid tarkvaraarendusmetoodikaid, vaid keskendutakse peamiselt IT arenduse ja halduse koostoimimise printsiipidele ja nende juurutamise uurimisele rõhuasetusega organisatsioonimuudatusele.

1.6 Teoreetilise peatüki kokkuvõte

Peatükis tutvustati *DevOps* terminit, selle sarnasusi ning erinevusi paindlike tarkvaraarendusmeetoditega. Selgitati metoodika kasulikkust teiste suurettevõtete näitel. Anti ülevaade senisest akadeemilisest uurimustöödest antud teemal, valiti uurimismetoodika ning ja püstitati uurimisküsimus, millele töö käigus hakatakse vastust otsima.

2 *DevOps* juurutamise analüüs Telia Eesti AS näitel

Peatükis antakse ülevaade antud magistritöös analüüsitavast ettevõttest. Kirjeldatakse lähtuvalt ettevõtte eesmärkidest, mida *DevOps* juurutamisega soovitakse saavutada. Autor teostab eelanalüüsi *DevOps* metoodika sobivuse analüüsimiseks halduse vaatevinklist ning kaardistab sellega kaasnevad võimalikud riskid. Viimasena antakse ülevaade probleemidest, mis ilmnesid pärast juurutamist.

2.1 Ettevõtte kirjeldus

Magistritöös käsitletav ettevõte on Telia Eesti AS, üks Eesti suurimaid telekommunikatsiooniettevõtteid. Ettevõttes töötab üle 2000 inimese. Nendest otseselt IT teenuste arendamise ning haldamisega on seotud ligikaudu 500 inimest. Antud töö eesmärki sobib ettevõte toetama just seetõttu, et tegemist on tehnoloogiaorganisatsiooniga, kus on juba juurutatud ettevõtet tervikuna läbivad protsessid, mis baseeruvad Info- ja Telekommunikatsioonisektoris laialt kasutatavale e-TOM äriprotsesside raamistikule [12]. Samuti on ettevõttel ISO 27001 sertifikaat [23] ning IT teenuste halduses lähtutakse ITIL v3 soovitustest [48]. Kogu tegevus on jaotatud funktsionaalseteks kompetentsiüksusteks nagu näiteks arendusüksused, teenuste haldusüksused ja erinevad IT infrastruktuuri funktsioone toetavad üksused ning ettevõtte erinevate üksuseid läbivaid protsesse koordineerivad dedikeeritud protsessijuhid.

Tarkvaraarendust teostatakse ettevõttes tuginedes Lean Startup printsiipidele[45]:

- Selle asemel, et kulutada nädalaid või kuid planeerimisele ja uurimustele, lepivad ettevõtjad teadmisega, et neil on hulk testimata hüpoteese. Selle asemel et kirjutada detailset äriplaani, kogutakse hüpoteesid kokku raamisikku, mida kutsutakse ärimudeliks, mis kirjeldab, kuidas ettevõte loob väärtust oma klientidele.
- Hüpoteeside testimiseks kasutatakse kliendikeskset arendust, kus potentsiaalsetelt klientidelt või kasutajatelt küsitakse tagasisidet kõikidele arendusmudeli elementidele, näiteks funktsionaalsused, hinnastamine, turustuskanalid jms. Rõhuasetus on sealjuures paindlikkusel ja kiirusel: luuakse

minimaalselt elujõuline toode ning kohe küsitakse sellele klientidelt tagasiside. Selle abil hinnatakse ümber ärimudeli eeldused ning algatatakse tsükkel uuesti, tehes kas väikeseid või pöördelisi muudatusi ideede juures, mis ei tööta.

- Kasutades paindlikku arendusmetoodikat koos kliendikeskse arendusega. Vastupidiselt aastapikkusele arendustsüklile, kus arvatakse teadvat klientide probleeme ja tootevajadusi, toimub paindlik arendamine iteratsioonidena ning järk-järguliselt.

2.2 *DevOps* metoodikal baseeruv struktuurimuudatus

Organisatsioonis tehnoloogiaüksuse toimimist mõjutavat muudatust otsima ajendas situatsioon, kus arendusüksused tootsid uusi teenuseid juurde ning rakendushaldusüksuste võimekus neid toetada ei vastanud enam organisatsiooni ootustele. Esines probleeme arendusüksuste ning platvormi haldusüksuste vahel nii ressursside tellimisel, haldamisel ja tarnetsüklite planeerimisel. Sooviti rakendada muudatust, kus moodustatud meeskonnad võtavad arendatava teenuse elutsükli vaates tervikvastutuse, kiirendades tarnetsükleid ning läbi selle tõsta teenuste kvaliteeti ja konkurentsivõimet turul. Lisaks sooviti nende tegevuste abil vähendada tarkvaravigade hulka toodangusüsteemides. [39]

Juhtkond püstitas hüpoteesi, et neid probleeme saab lahendada *DevOps* metoodika juurutamisega ettevõttes.

Moodustati teenuste portfelli aluseks võttes uued teenusepõhised üksused ja meeskonnad ning komplekteeriti need nii arenduse kui halduse spetsialistidest. Eesmärgiks võeti, et meeskond (*DevOps* tiim) peab võtma vastutuse toote kui terviku eest. Kui varasemalt olid teenuse arendamise funktsioaalsus erinevates osakondades, siis uue korra järgi pidid olema kaetud projektijuhtimine, toote arendamine, testimine, koodi paigaldamine toodangusse, platvormi haldamine ja turvalisuse tagamine ühes teenuse eest vastutavas meeskonnas. Projektijuhtimise funktsiooni liikumisega teenuste eest vastutavadesse üksustesse kaotati sisemine projektijuhtimise resurss. Kaotati kesksed andmebaasi ja operatsioonisüsteemi eest vastutavad osakonnad vastavate kompetentside viimisega teenuste eest vastutavatesse osakondadesse. Lisaks soovitati

vähendada juhtimistasandite hulka ettevõttes ning sellega seoses kaotati üks keskastmejuhtide kiht. Muudatus jõustati 01.juuni 2016. [39]

2.3 *DevOps* juurutamise analüüs

Juurutamisega kaasnevatele haldusorganisatsiooni muudatustele teostas autor analüüsi. Analüüsi skoopi piirati autori rolli ja vastutusvaldkonnaga ettevõttes töötamise hetkel milleks oli IT infrastruktuuri arhitekti positsioon.

2.4 Muudatusega kaasnevad oodatavad kasud ettevõttele

- Valdkonna üksustes süveneb omanikutunnetus ja vastutus.
- Ristkompetentside jagamine halduse/arenduse vahel.
- Kommunikatsioon sama valdkonna sees efektiivsem.
- Ressurss teenustele on dedikeeritud.
- Eeldatav tarnekiirus paraneb võrreldes senisega.
- Tegevused muutuvad vahetumaks, kõik tegevused ei eelda töökäske.
- Teenuse toimimise terviklik vaade ja dokumenteeritus teenuse/toote üleselt paraneb.
- Infrastruktuuriteenuste vaates (kettamassiivid, võrk, virtuaalserverid) soodustab *DevOps* selgemini defineerima tsentraalset tehnilist teenust ja seda pidevalt arendama.
- Tekib teenusega seoses terviklikum kuluvaade. Eelduseks on valdkonnasisene otsustusõigus.

2.5 Muudatusega kaasnevate riskide kaardistus

Autor kaardistas võimalikud organisatsiooni muudatusega kaasnevad riskid. Kirjeldati risk, selle esinemise tõenäosus, mõju realiseerumise korral ning võimalikud rakendatavad abinõud, mis on toodud välja tabelis 1.

ID	Riski kirjeldus	Tõe-näosus	Tagajärje mõju	Rakendatavad abinõud
R1	Seniste IaaS/PaaS kompetentsikeskuste planeeritava muutmisega kaasnev teenuse kvaliteedi taseme langus	Kõrge	Suur	• Valdkonna sisesed valveringid ja kompetentside ristkasutus. • Kogenud spetsialistide värbamine • Koolitused
R2	Platvormide kompromiteerimise tõenäosuse kasv (arendajatele *root* õigused, halduse kompetentside puudumine)	Keskmine	Suur	• Valdkondade ülene protsessimudel • Tarneprotsessi automatiseerimine • Koolitused • Lahendada turve võrgukihis
R3	Võtmeisikute lahkumine	Keskmine	Suur	• Rahuoluuuringust välja tulnud probleemide lahendamine ja positivsete aspektide tugevdamine • Personaalne lähenemine • Mõjutatud spetsialistide kaasamine planeerimisse
R4	Dokumenteerituse taseme langus (töökäskude, muudatuste logi, HPSM kirjed)	Kõrge	Keskmine	• Tsentraalsed vahendid automaatseks dokumenteerimiseks ning süsteemide avastamiseks
R5	Tehnoloogiate elutsükli juhtimine killustub	Keskmine	Madal	• Rakenduste ning arhitektuuriliste komponentide kataloogi käsitlus peab olema tsentraalselt jõustatud
R6	Mittestabiilsete tehniliste toodete kasutamisest tulenevad turva ja teenuse kvaliteedi riskid. (beeta-versioonid jms)	Keskmine	Kõrge	• Rakenduste ning arhitektuuriliste kompoentide kataloogi kasutamine • Koolitused • Lahendada turve võrgukihis
R7	Rakenduste ning ressursside haldusvastutuse määramine	Keskmine	Keskmine	• Teenuste kataloog • Meeskonnajuhid peavad võtma vastutuse
R8	Valvete mehitamatus (senine minimaalselt 3 valdkonna spetsialisti nõue)	Keskmine	Kõrge	• Valvete üleandmise plaan • Senised valvegrupid jätkuvad, ehkki liikmed on erinevates valdkondades • Koolitused
R9	Platvormi haldusrutiine ei järgita (OS paikamised, hooldusrutiinid jne)	Kõrge	Keskmine	• Valdkondade ülesed serverite seadistamise standardid ja standardite järgimise kontroll (operatsioonisüsteemi, andmebaasi seadistatamine) • Valdkondade ülene “platvormi halduse grupp” peab kirjeldama ja järgima ühtseid reegleid • Keskne platvormide halduslahenduse teenus üksustele
R10	Kesksete platvormi ja andmebaasi ressursside haldamine (Oracle, PostreSQL, RedHat HA	Keskmine	Keskmine	• Kesksed jagatud platvormid ja klastrid võimalusel tükeldada ning valdkondade vahel ära jagada • Valdkondadeülene eksperditaseme

	klastrid)			kompetentsi kaardistamine ning juhtimine
R11	Intsidendi, probleemi, halduse protsesside rikkumine, kui ei kasutata keskset HPSM vahendit (arenduste meeskonnad juhivad töid JIRA abil)	Keskmine	Madal	• ITSM süsteemide liidestamine JIRA'ga • Uute protsesside loomine
R12	Struktuuri muutudes ligipääsuõiguste grupid ei muutu (RBAC, rakendusepõhised)	Kõrge	Suur	• Protsessihalduse vahendites muudatuse eelselt õiguste kaardistamine ning vajadusel ette uute õiguste gruppide tegemine
R13	Süsteemsete ning priviligeeritud kasutajakontode üleminek	Keskmine	Keskmine	• Süsteemide üleandmisel õiguste üleandmise protsessi kirjeldamine • Keskse paroolihoidla kasutamise jõustamine
R14	Tehnoloogiliste platvormide arenduse ja elutsükli edasise vastutuse ebaselgus (Hetkel vastutus vastavas keskses kompetentsi meeskonnas)	Madal	Väike	• IAAS/PAAS teenuste arendusmeeskondade kompletkeerimine • Koolitused
R16	Haldusvahendite kasutamine killustub (tarnimine, monitooring, varundamine, logimine, automatiseerimine)	Keskmine	Keskmine	• IaaS, PaaS võimaldamine meeskondadele • Tarneprotsessi automatiseerimine • Arhitektuurinõukogu kontrollorganina • Valdkonnapõhiste vs üldiste tehnoloogiate strateegia juhtimise printsiibid
R17	Integratsiooniteenuste vastutuse määramatus	Keskmine	Keskmine	• Määrata selge vastutus integratsiooniteenuste arenduse ja halduse eest. Näiteks: SOA, RabbitMQ, SSO, VPN, Seif/EPS.
R18	Projektijuhi rolli kaotamine langetab infrastruktuuri komponentide tarnekiirust	Keskmine	Keskmine	• Tarneprotsessi automatiseerimine
R19	Olemasolevate mõõdikute sobimatus uue töökorraldusega	Keskmine	Madal	• *DevOps* projekti (organisatsiooni muudatuse) KPI'd • Uusi protsesse toetavad üksuste KPI'd

Tabel 1 Kaardistatud teadaolevad riskid ning nende vastumeetmed

2.6 *DevOps* juurutamisejärgsed probleemid

Peatükis tuuakse välja järgmised kaardistatud ja kaardistamata probleemid, mis ilmnesid *DevOps* metoodika juurutamise järgselt ettevõttes ajavahemikus september 2016 kuni veebruar 2017.

2.7 Realiseerunud kaardistatud riskid

1. Volituste ning IT varade üleandmine oli formaalne mitte sisuline, uute meeskondade juhid ei olnud ette valmistatud ning varustatud ressurssidega tagamaks teenuste kvaliteeti.
2. Haldusvaates senise tsentraalse teenuste halduse detsentraliseerimisega kaasnevad riskid jäid käsitlemata, näiteks platvormihalduse korraldamine uutes struktuuriüksustes arendajate poolt.
3. Ei kehtestatud suuniseid, kuidas süsteemiõiguste või haldusvastutuse jagunemine peaks toimima hakkama tingimustes, kus vanad reeglid enam ei kehtinud.
4. Puudusid selged mõõdikud muudatuse eesmärkide saavutamise hindamiseks.

2.8 Realiseerunud autori poolt kaardistamata riskid

1. Olemasolevate töötavate protsesside (kokkulepped ja kord, kuidas arendatakse ja hallatakse) asemele ei pakutud uusi tegevusi toetavaid protsesse. See omakorda tingis uue struktuuri ning vanade protsesside paralleelse toimimise, mis kohati olid teineteisele vasturääkivad.
2. Kompetentse ei kaardistatud, osakonnad ning spetsialistid liigutati uutesse struktuuriüksustesse ilma sisulise analüüsita.
3. Spetsialistidel puudus selge arusaam, mida nad tegema peavad hakkama. Ootuste kommunikatsioon juhtkonna erinevatel tasemetel ei olnud konsistentne.
4. Puudus täpne ülevaade, millisel küpsustasemel erinevad *DevOps* meeskonnad olid enne ja peale struktuurimuudatust ning sellest tulenevalt puudus koolitusplaan puuduste likvideerimiseks.
5. Teenuste eest vastutavates üksustes nii tehnoloogiline kui kompetentside tase erines. Puudus arusaam, milliseid teenuseid uus metoodika enim toetab ning millistele on seda raskem rakendada.
6. Muudatusejärgselt vähenenud arendusvõimekus, arendajate halduskompetentside omandamine ning haldusprotsesside õppimine teostati arendusetegevuse arvelt.
7. Puudus portfellihalduse raamistik uues struktuuris valdkondadeüleste projektide prioriteetide määramisel.

2.9 *DevOps* juurutamise analüüsi peatüki kokkuvõte

Riskianalüüsi nõrkus on liigne keskendumine haldusprotsessidele ning tehnilistele tingimustele. Realiseerunud riskid viitavad süsteemsele probleemile planeerimises organisatsioonijuhtimise tasemel. Suurematekstakistajateks tehnilistest probleemidest olid eesmärkide selgusetus, spetsialistide mittekaasamine ning koolitusvajaduste tähelepanuta jätmine. Puudus konkreetne ajakava ning määramata jäid muudatuse edukust hinnata võimaldavad mõõdikud. Eksisteerivad protsessid jäid muutmata, mis tingis uue töökorralduse ning eksisteerivate protsesside omavahelise konfliktsuse.

Autor analüüsi käigus jättis kaardistamata arendusüksuseid ning -protsesse kätkevad riskid muudatuse kontekstis, kuna see ei olnud ettevõttes töötades tema vastutusalas. Arendusvaldkonna eest vastutajate poolt hinnati *DevOps* muudatuse mõju eksisteerivatele protsessidele minimaalseks. Sellele tuginedes keskendus autor peamiselt halduse valdkonna analüüsimsiele. Ilmnenud probleemid arendusüksustes olid seetõttu eelnevalt käsitlemata ning võimalike riskidena kommunikeerimata.

3 Ettepanekud *DevOps* juurutamiseks suurorganisatsioonis

Olemuslikult ei erine *DevOps* juurutamine mistahes muust organisatsiooni muudatuse protsessist. Kogu planeerimise juures võib kasutada juba väljatöötatud meetodeid nagu näiteks John P. Kotteri poolt välja pakutud muudatuste juhtimise mudel: [28]

1. Loo kiireloomulisuse vajadus	Uuri turge ning konkurentsivõimet tõstvaid muutujaid Tuvasta kriis, potentsiaalne kriis või suur võimalus
2. Loo tugev juhtiv koalitsioon	Kogu kokku piisavalt mõjukas grupp muudatust juhtima Julgusta gruppi töötama ühtse meeskonnana
3. Loo visioon	Loo visioon, mis aitaks suunata muudatusele tehtavaid pingutusi Loo strateegiad toetamaks seda visiooni
4. Kommunikeeri visiooni	Kasutades kõiki kommunikatsionikanaleid tutvusta visiooni ning strateegiaid Tutvusta uusi käitumismudeleid tuues juhtivat koalitsiooni kui näidet
5. Julgusta teisi toimima visiooni järgi	Eemalda muudatuse takistajad Muuda süsteeme või struktuure, mis õõnestavad muudatust Julgusta riski võtmist, ebatraditsioonilisi ideid ja tegevusi
6. Planeeri ja loo kiireid võite	Planeeri nähtavaid jõudluse parandajaid Loo need parandajad Tunnusta ja premeeri töötajaid, kes parendamistega seotud
7. Konsolideeri parendused ning tooda veel rohkem muudatust	Kasutades suurenenud usaldusväärsust muuda süsteeme, struktuure ning poliitikaid, mis ei toeta uut visiooni Palka, eduta ja arenda töötajaid, kes suudavad visiooni

	ellu viia Taaselusta protsessi uute projektide või muudatustega
8. Muuda uus lähenemine institutsionaalseks	Too välja seosed uutmoodi tegevuste ning ettevõtte edu vahel Arenda võimekust tagamaks juhtkonna arengut ning üleandmist

Tabel 2 Organisatsioonimuudatuse läbiviimine J.P. Kotteri kohaselt

Alljärgnevates punktides toob autor välja uurimustöö tulemusena välja selgitatud tegevuste nimekirja, mis vastavad püstitatud uurimusküsimusele "Kuidas juhtida organisatsioonimuudatust *DevOps* metoodika kasutuselevõtmiseks?". Punktid ei ole esitatud tingimata tähtsuse järjekorras, kuid on omavahel loogiliselt seotud.

Iga punkti lõpus võrdleb autor ettepanekut Telia Eesti AS tehtuga ja analüüsib punkti rakendatavust või vastavust. Võrdlus aitab praktilise näitena esile tuua, mida vastava punkti rakendamine või osaline rakendamine ettevõttes võib kaasa tuua.

3.1 Organisatsiooni kultuuri muutmine

Stanley McCrhrystal oma raamatus *Team of Teams: The Power of Small Groups in a Fragmented* World soovitab muuta lähenemist organisatsiooni juhtimisele kus soovitakse kontrollida keskelt kõiki detaile. Selle asemel, et organisatsiooni juhtida kontrollides igat organisatsiooni liigutust, soovitab McChrystal suunamise asemel võimaldamise lähenemist kasutada. Tema nägemuses juht opereerib kui "Silm peal, käed eemal" võimaldaja, luues ning hooldades ökosüsteemi, mille piires organisatsioon tegutseb. [30]

Kogu *DevOps* paradigma juurutamine saab alguse sellest, et ettevõtte erinevatel tasemetel mõistetakse sama moodi probleemi, mida soovitakse lahendada, meetmeid, millega soovitakse tulemust saavutada, ning takistajaid. See puudutab nii juhtkonda, osakondade juhte kui spetsialiste.

Juhtkonna tasemel on oluline aru saada initsiatiivi toetamisest ning sellele vajalike ressursside võimaldamisest. Kuna *DevOps* metoodika eeldab muutusi selles, kuidas

ettevõtte erinevad üksused üksteisega koos töötavad, siis potentsiaalsete konfliktide või vastasseisude puhul ilma juhkonna toeta üldist suunda või tooni rõhutada on raskendatud.

Oluline muudatus organisatsioonikultuuri tasemel on suurema otsustusvabaduse võimaldamine teenust haldavate meeskondade tasemel. Ühest küljest võimaldab see meeskondadel oma eksisteerivate kompetentside raames olla produktiivsem ning paindlikum muudatustele reageerimisel. Samas kaasneb sellega vajadus võimaldada spetsialistidel eksida, eriti algusfaasis, kus *DevOps* meeskonnas puuduolevate kompetentside omandamine on alles käsil.

Mathias Meyer rõhutab süüdistustevabade *post-mortem*'ide olulisust. [31] Lähtutakse eeldusest, et inimesed on üldjuhul heatahtlikud ning toimetavad parimas usus, oma parimate teadmiste kohaselt organisatsiooni piirangutest lähtuvalt.

Sageli on vead tingitud süsteemide komplekssusest. Tulenevalt spetsialiseerumisest puuduvad organisatsioonides inimesed, kes omaks kogu süsteemist tervikpilti detailsel tasemel. Juurutada filosoofiat, kus teadvustatakse, et vea teinud spetsialistile tegevuse hetkel tundus see tegevus tema teadmiste kohaselt mõistlik. Vastasel juhul ei oleks seda tegevust teostatud.

Erik Hollnagel'i sõnul on oluline mõista, et õnnetusjuhtumid ei juhtu sellepärast, et inimesed riskivad ning kaotavad, vaid seetõttu, et isik usub, et see, mis juhtub ei ole võimalik. [47]

Kui eksimuste järgselt spetsialistid teavad, et vea tegemine ei tähenda automaatselt vallandamist või karistamist, siis ollakse põhjuste avaldamisel avameelsemad. Vastasel juhul esineb info varjamist vähendamaks repressioone. Varjamine omakorda toob kaasa vigade kordumise tulevikus, kuna tervikpilti omamata pole sageli võimalik juurpõhjuseid likvideerida.

Telia Eesti AS töökorraldus vastab osaliselt sellele soovitusele. Struktuuriüksustele oli delegeeritud otsustusvabadust, meeskondade vahel oli kommunikatsioon kiire ning vahetu. Intsidentide järelanalüüsid olid avatud iseloomuga, kus eesmärgiks võeti kogetust õppimine süüdlase leidmise asemel. See soodustas avatud kultuuri spetsialistide vahel ning aitas kaasa teenuste kvaliteedi parandamisele.

Juhtkond toetas *DevOps* initsiatiivi, kuid selle eesmärkide ja juurutamise plaani kommunikatsioon ei olnud samamoodi mõistetav kõikidele organisatsioonikihtidele. Ei olnud üheselt arusaadav, mis probleemi lahendada tahetakse ja muudatusi takistavate aspektide käsitlus oli pealiskaudne. Sellest tingituna ei olnud kõikide spetsialistide ja esmatasandi juhtide kaasatus ning meelestatus muudatust toetav. See omakorda tekitas situatsiooni, kus initsiatiiv muudatuseks oli juhtkonnalt ülevalt alla, mitte spetsialistidelt alt üles.

3.2 *DevOps* küpsuse hindamine

Skoobi määramise eelselt tuleb mõista, millisel küpsustasemel on organisatsioon tervikuna ning osakonnad eraldiseisvatena. Küpsuse tasemest sõltub näiteks kompetentside täiendamise vajaduse ulatus, sealhulgas uute spetsialistide värbamine ning koolituskava planeerimine.

Küpsustaseme mõõtmine organisatsioonis aitab kaasa kompetentsikeskuste kaardistamisele ning potensiaalse osakonna või osakondade valikul, kus plaanitakse hakata *DevOps* metoodikat juurutama.

Kindlaks tuleb määrata nii mõõtmise hetkel olev küpsuse tase kui soovitud või saavutatav tase. Lisaks sellele, et määratakse ära hetkeseis, aitab küpsuse hindamine formuleerida tegevused või initsiatiivid, mille abil soovitatavale tasemele jõuda soovitakse.

Formaalset *DevOps* küpsuse hindamise raamistikku autor töö kirjutamise hetkel ei suutnud tuvastada. Lähimaid küpsusmudelite hindamise vahendeid tulevad pideva juurutamise hindamise mudelitest nagu InfoQ pideva juurutamise küpsusmudel [37], mis omakorda baseerub CMM küpsusmudelil[34]. Saamaks terviklikku ülevaadet on töös kombineeritud InfoQ mudelit, kus keskendutakse tehnilistele aspektidele, ning Steve Pereira poolt koostatud *DevOps Checklist* küsimustikku [35], mille abil mõõdetakse tööpõhimõtete küpsust.

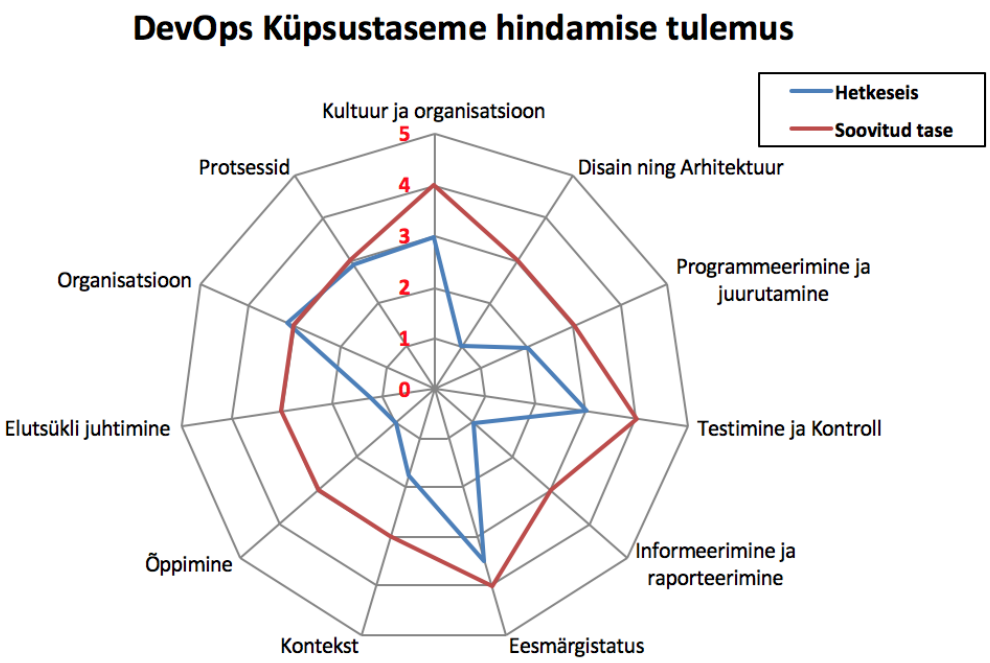
Alljärgnevalt on toodud tabel kombineeritud InfoQ ja *DevOps Checklist* mõõdetavatest aspektidest ning nende seletustest.

Protsessid	Rituaalid tagamaks konsistentsust ja kindlustunnet mis võimaldavad pidevat parendamist
Organisatsioon	Struktuurid mis toetavad erinevate osakondade omavahelist koostööd ja produktiivsust
Elutsükli juhtimine	Tarkvara elutsükli haldamine pidevalt muutuvas keskkonnas
Õppimine	Koolituse ning pideva arengu soodustamine
Kontekst	Informatsiooni kättesaadavus neile kellele seda vaja siis kui neil seda vaja
Eesmärgistatus	Organisatsiooni eesmärkide seostatus üksikisikute ja gruppide ülesannetega
Kultuur ja organisatsioon	Hinnatakse pidevat tarnimist toetavaid kultuurilisi ja organisatoorseid aspekte, koostööd struktuuriüksuste vahel
Disain ning arhitektuur	Toodete ja teenuste disaini ning arhitektuuri tase, süsteemide konsolideerituse tase
Programmeerimine ja juurutamine	Tarneahela komponentide automatiseerituse tase koos versioniseerimise, tarne automatiseerituse tasemega
Testimine ja kontroll	Testimise automatiseerituse tase, testitega kaetuse tase, testimist toetavad protsessid ja vahendid
Informeerimine ja raporteerimine	Mõõdikud mis toetavad tarneprotsessi, nende ajakohasus ning ligipääsetavus, seostatus äriliste eesmärkidega

Tabel 3 Küpsustaseme määramiseks vajalikud mõõdikud

Eraldi pole välja toodud vastavalt küpsustasemetele erinevaid mõõdikuid, kuid tabeli abil saab aimdust valdkondadest ning tegevustest, mida hinnata on vaja. Tuginedes nendele mõõdikutele saab kaardistada osakondade valmisolekut *DevOps* metoodikate kasutusele võtmiseks.

Näidisena on välja toodud radar-tüüpi graafik, kus sinise joonega on tähistatud hetkeseis ning punase joonega küpsustasemed, mida soovitakse saavutada.



Joonis 6 Näidismaatriks *DevOps* küpsustaseme hindamisest

Joonisel 6 kujutatud graafik aitab visualiseerida, millistes valdkondades on puudujäägid kõige suuremad võrdluses eesmärkidega.

Telia Eesti AS teostatu selle punkti soovitusele ei vasta. Muudatuse eelselt ettevõttes osakondade ning tööpõhimõtete küpsustaset ei mõõdetud. Sellest tulenevalt puudus eesmärk, millist küpsustaset saavutada sooviti ning seda toetav koolitus- või värbamiskava. Küpsustaseme mõõtmise etapi vahele jätmine raskendas muudatuse skoobi hindamist ning selle määramist. Polnud selge millises ulatuses oli organisatsioon muudatuseks valmis.

3.3 Muudatuse skoobi määramine

Muudatust planeerides tuleb alati paika panna esmalt tegevuse ulatus. Kriitilise edu faktorina tuuakse mitmetes *DevOps* käsitlevas materjalides, näiteks raamatutes [22, 41, 44], uuringutes [2, 24] või artiklites [7, 8, 40, 42] välja vajadus alustada nii väikese skoobiga kui võimalik. Õnnestunud piloodile tuginedes alles laiendatakse metoodikat teistele teenustele või ülejäänud organisatsioonile. Kuna metoodika juurutamine eeldab eksisteerivate protsesside ümber kohandamist või täiesti uute protsesside juurutamist, siis on sobivust soovitav testida esialgu väikesel skaalal ettevõttes.

Anand Srivatsav Amaradri ja Swetha Bindu Nutalapati oma oma magistritöös "*Continuous Integration, Deployment and Testing in DevOps Environment*" viisid läbi küsitluse ettevõtete seas üle maailma, kes olid *DevOps* juurutanud või lugesid ennast seda praktiseerivaks. Ülekaalukalt (88%) vastanutest leidis, et parem on alustada *DevOps* initsiatiivi väikese projektina. [2]

Piloodi skoobiks sobib valida sõltuvalt ettevõtte struktuurist mõni nähtav ning strateegiline teenus, selle arendamisega seotud spetsialistid või osakond. Tihtipeale on ettevõttes mõni osakond juba *DevOps* printsiipe osaliselt järgimas või saavutanud vajaliku küpsustaseme, mis võimaldab kergemat kaasamist spetsialistide vaates.

Muudatusega kaasnevad uued ning juba eksisteerivad protsessid ei pruugi tingimata teineteisega ühilduvad olla, seetõttu peab olema võimalik nende paralleelne eksistents ettevõtte töökorralduses.

Telia Eesti AS muudatus sellele soovitusele ei vasta. Planeeritud muudatuse nõrkuseks oli liiga suure skoobi valimine. Ettevõttes olid olemas *DevOps* mõistes sobivat küpsust omavad osakonnad. Samas rakendati muudatust ka nendele osakondadele, kelle küpsustase oli madal. Kehtima jäid endised protsessid, mis tekitasid konflikti uue struktuuriga organisatsioonis, kus vastutused ning kohustused olid muutunud.

3.4 *DevOps* programmi projektirühma loomine

Arvestades tõenäosust, et isegi suurima *DevOps* küpsustasemega mõõdetud ning piloteerimiseks valitud teenuse arendamisega seotud osakonnas ei ole kõiki vajalikke kompetentse, siis piloteerimisfaasis tuleb moodustada üle ettevõtte spetsialistidest koosnev töörühm, kes aitab pilooteeritavat teenust või osakonda viia uuele toimimismudelile. Kompetentside puudumisel võib töörühma kaasata väliseid partnereid või konsultante.

Töörühma liikmed kas eraldi või kombineerituna peavad omama kompetentse arenduses, testimises, süsteemihalduses, turvalisuses ja automatiseerimises.

Töörühma ülesandeks on aidata piloteerimiseks väljavalitud teenuse eest vastutaval osakonnal juurutada protsesse ning vahendeid automatseeritud tarnimiseks. See eeldab

ettekujutuse omamist teenuse väärtusahelast – millist tööd kelle poolt tehakse ning milliste sammudega seda töövoogu parandada.

Võimaldamaks töörühmal oma ülesandeid täita tuleb neile anda juhtkonna tasemel mandaat muudatuse läbiviimiseks ning eraldada ressursid nii aja kui võimalike investeeringute näol.

Töörühma etteotsa peab tekkima roll, mida nimetatakse DevOps evangelistiks [29]. Evangelisti vastutuseks on olla muudatuse juhtija ning *DevOps* printsiipide eeskõneleja ettevõttes. Rolli ülesandeks on propageerida *DevOps* metoodikaga kaasnevaid kasusid. Evangelisti rollis olev spetsialist peab tegelema nii äriliste tellijate kui arendus- ja haldusmeeskondade kaasamisega muudatusse, kandes erinevate distsipliinide vahelise tõlgi rolli. Soovitatavalt sügavate tehniliste teadmistega spetsialist või arhitekt, kes on võimeline ka sisuliselt probleeme lahendama.

Telia Eesti AS vastab osaliselt sellele soovitusele. Ettevõttes tekitati *DevOps* evangelistide rollid, kelle kanda oli initsiatiivi vedamine. Evangelisti tiitlit kandvad töötajad olid oma ala sügavate tehniliste teadmistega eksperdid.

Töörühma spetsialistidest, kes oleks hakanud teenuste üksuste tarneprotsesside muutmist toetama, ei moodustatud. Seetõttu ei tekkinud laiapõhjalist muudatust vedavat koalitsiooni, kogu vastutus protsesside ja süsteemide parendamiseks *DevOps* kontekstis oli üksikute inimeste õlul.

3.5 *DevOps* rollide ja kompetentside jaotamine

Sõltuvalt eksisteerivast organisatsiooni struktuurist või ärimudelist võivad olla kasutuses erinevate ülesannetega või vastutusaladega osakonnad. Uue struktuuri loomine ning ettevõtte reorganiseerimine ei ole põhifookuses *DevOps* juurutamise piloodi faasis.

DevOps planeerimisel tuleb välja selgitada, milline on optimaalseim mudel meeskonna komplekteerimisel nii piloteerimise faasis kui ettevõttes tervikuna.

DevOps metoodika kasutuselevõtuga kaasneb organisatsioonis olevate üksuste omavahelise suhtlemise ning vastutuse muutumine. *DevOps* ideoloogia rõhutab

tervikvastutuse võtmist teenuse elutsükli eest ühe meeskonna piires, mis tähendab arendusmeeskondades eksisteerivate kompetentside täiendamist ja rollide lisandumist.

DevOps kontekstis ideaalne meeskond on väike (5-9 liiget), alaline (meeskonnaliikmed ei ole jagatud teiste teenustega), iseorganiseeruv, katab teenuse toimiseks kõik vajalikud kompetentsid ning vastutab ühe või enama äriteenuse või toote eest. Väiksemad organisatsiooniüksused kasutavad ära inimestevahelise suhtlemise tugevust, kus üksteist tuntakse paremini, on tekkinud usaldus ning seeläbi panustatakse ühisemalt eesmärkide nimel.

Meeskonna komplekteerimisel ning koolitusvajaduste hindamisel võtta aluseks T-kujuliste kompetentside omandamine, mida on kirjeldanud Joshua Partogi oma artiklis "*Who are the Professional Scrum Developers?*" [33]. T-kujulisi kompetentse illustreeriv maatriks on kujutatud joonisel 7.

Analyst	Programmer	Test Engineer	Web Designer	System Engineer
Write Executable Documents	Write Unit Test Code (xUnit)	Write Automated Tests	UX Design	DevOps
Requirements Engineering	Write Production Code	Functional Testing	Java Script, HTML, CSS, LeSS	Python, Perl, Go, shell
Write User Manuals	Design System Architecture, DB	Write Test Plan	Image, Icon, Logo Design	System and OS
…	…	…	…	…

Broad

Deep

Joonis 7 T-kujuliste kompetentside näide[33]

Kuna tingimuseks on, et meeskond peab võtma teenuse eest tervikvastutuse, ning samas lisatingimuseks, et optimaalne suurus on alla 10 inimese, peavad spetsialistid olema võimelised rohkem kui ühes valdkonnas ülesandeid täitma. T-kujulise kompetentsi mudel tähendabki, et eksisteerib küll kitsam spetsialiseerumine, kuid kõikides teistes valdkonda puudutavates küsimustes vähemalt orienteerutakse. See tähendab, et osakonnas on arendamise, testimise, haldamise, automatiseerimise, turvalisuse,

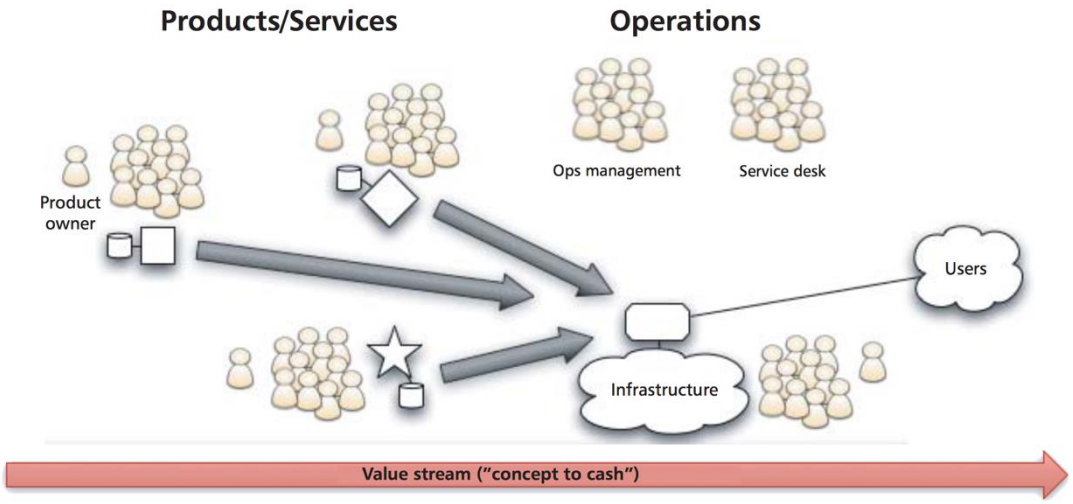
tarneijuhtimise rollid kas spetsialistide näol olemas või siis kompetentsidena kaetud.

Selise meeskonna mudeli kasuks räägib tõik, et kogu meeskonnaliikmete omavaheline kommunikatsioon on vahetu, mis omakorda aitab kaasa nii teenuse disaini, planeerimise, juurutamise kui intsidentide lahendamise kiirendamisele.

Kuna integreeritud meeskonnamudel on *DevOps* printsiipide üks osa, on oluline leida piir vajalike meeskonnasiseste funktsioonide ning teistelt osakondadelt või teenusepartneritelt teenusmudelina kasutatavate resursside vahele.

Ohukohaks on kõikidele vajalikele rollidele vastavaid spetsialistide mehitamise keerukus ning kompetentse juhtimine. Probleemiks võib kujuneda seniseid arendusmeeskondasid juhtinud juhtide oskamatus rakendada kompetentse valdkondades, millega nad varasemalt ei pidanud tegelema ning mille mõistmiseks puudub varasem kogemus. Probleem tekib *DevOps* metoodika laialdasel rakendamisel ettevõttes, kus igasse meeskonda ei pruugi olla võimalik leida kõikide vajalike kompetentsidega spetsialiste või on see teatud kitsaste kompetentside puhul ebaefektiivne (nt. igas meeskonnas andmebaasi spetsialist, võrguekspert, platvormi adnimistraator), eriti kui hallatavaid ühikuid on meeskonna või teenuse haldusalas vähe.

Autori hinnangul kõikidesse osakondadesse kõiki kompetentse ei ole otstarbekas planeerida, selle asemel võtta eesmärgiks kasutusele võtta IaaS või PaaS teenusemudelid ettevõttes arendusmeeskondade ning toetavate haldusüksuste vahel. Sõltumata teenuste omapärast on soovituslik viia sisse teenusemudel tsentraliseeritud teenustele, mida arendusüksused hakkavad kasutama. Selline lähenemisviis tagab töövahendite, praktikate ja tehnoloogiate elutsükli ühtlasema taseme ettevõttesiseselt.



Joonis 8 Teenusepõhise organisatsiooni mudel [21]

Teenuspõhise mudeli kasutamise puhul tuleb esmalt hinnata, millised kompetentsid on meeskonnas olemas ning millised on olulised teenuse osutamise vaatevinklist. Kompetentside kaardistamisel järgida ettevõtte IT strateegiat üldisemalt ning tehnoloogiaid valides lähtuda rakenduste ning arhitektuuriliste komponentide võimalutest.

Teenusepõhisuse kasutamine tingib ümberkorraldused õiguste ja vastutuste piiride tõmbamisel – eksisteerivate protsesside järgi tingisid muudatused vastavates süsteemides töökäske teistesse osakondadesse. Platvormi teenusena kasutamine võib tähendada selliste muudatustööde teostamist *DevOps* mudelit kasutava meeskonnasisese teostusena. Mudel lähtub printsiibist, kus teenust tarbival meeskonnal on otsustusvabadus ning iseteeninduse võimalus funktsionaalsuse kasutamisel.

Kuna rakenduse tervikvastutusega kaasneb rakenduse haldamise kohustus, siis kesksena figureeriv rakendushaldusüksuse funktsioon kaob ära, jaotudes vastavatesse osakondadesse laiali.

Tehnilised haldusfunktsioonid jäävad eraldi, kuid fookus muutub infrastruktuuri teenuste osutamisele (IaaS, PaaS) toodete eest vastutavatele üksustele. Teenuse osutamisele keskendumine vähendab haldusüksustes rutiinseid tegevusi ning töömahte võrreldes varasemaga. Teisalt toob selline toimimismudel kaasa suurenenud vajaduse arendada välja integratsioone tarneahela teiste lülidega. See omakorda tähendab, et

suureneb arendustegevuste ootus infrastruktuuri spetsialistide suunas automatiseerimise ning skriptimise läbi.

Telia Eesti vastab osaliselt sellele soovitusele. Plaaniti rakendada T-kujulist kompetentsimudelit, kus spetsialistid pidid olema suutelised oma üksuses üksteist asendama ja toetama valdkondades, kus neil endil puudus sügav kompetents.

Infrastruktuuriteenused viidi osaliselt IaaS/PaaS mudelile, kus teenuste eest vastutavatel üksustel oli võimalus kesksete infrastruktuuriteenuste nagu monitooring, varundamine, logimine jms kasutamiseks ning läbi iseteeninduse virtuaalserverite tellimiseks. Platvormi haldamise funktsiooni teenusepõhisena ei käsitletud. See tõi kaasa endaga situatsiooni, kus platvormi haldamise eest oma valdkonna piires hakkasid vastutama üksused, kus vastav kompetents oli puudulik. Kompetentside puudumist leevendati spetsialistide ristkasutamisega struktuuriüksuste vahel, mis tekitas õiguste ning kohustuste mittekattuvust.

3.6 Mõõdikute seadmine

Muudatuste mõju, hetkeseisu ja tulemuste hindamiseks on oluline määrata mõõdikud. Mõõdikud tuleb määrata auditeeritavatena, asja- ja ajakohastena.

Mõõdikute määramisel kasutada SMART (*Specific*, *Measurable*, *Agreed upon*, *Realistic, Time phased*) kriteeriume [14]:

1. Piiritleda Konkreetne parendusvaldkond, mida mõõtma plaanitakse hakata
2. Seatav kriteerium peab olema mõõdetav või vähemalt progressile viitav
3. Määratud teostaja(d)
4. Seatud tulemused peavad olema võimalikud saavutada
5. Teha kindlaks aeg, mille jooksul peavad olema tulemused saavutatud

Traditsiooniliselt hinnatakse erinevate üksuste tegevust erinevate mõõdikute järgi. Näiteks, arendusmeeskondade võimekust hinnatakse selle järgi, kui kiirelt suudavad nad tarnida uut funktsionaalsust. Samas haldusmeeskondade võimekust hinnatakse süsteemide stabiilsuse järgi, mis tingib püüdluse võimalikult vähestele muudatustele süsteemides. See on üks sagedasemaid vastasseisu põhjustajaid tarneprotsessi erinevates etappides. Võimalike eksimuste vältimiseks peavad arendajad läbima paljude

kontrollpunktidega protsessi, mille järgimine pikendab arendustsüklit ning põhjustab funktsionaalsuse harvema tarnimise.

Lähtuvalt punktis 3.2 toodud mudeli järgi mõõdetud *DevOps* küpsustasemest on soovitatav kasutada erinevaid mõõdikuid, mis toetavad paremini reaalset võimekust midagi parendada.

DevOps madalamatel küpsustasemetel mõõdetakse arendustsükli pikkust, erakorraliste muudatuste hulka, toodangusüsteemides avastatud vigade hulka, automaattestide osakaalu, teenuseseisakute hulka, vigade parandamiseks kulunud aega ja teenustaseme kokkulepete täitmist.

Edasijõudnud *DevOps* tasemel mõõdetakse lisaks katkestuseta funktsionaalsuse tarnimise hulka, automatiseerituse taset, taaskasutatavate süsteemsete komponentide hulka ja testitsüklite kiirust.

Gartner on põhjaliku dokumendi koostanud, kus käsitletakse mõõdikuid nii kultuuri, protsesside, toetava taristu, arhitektuuri, arendus ja testimise järgi [27]. Mõõdikute koondtabel organisatsiooni küpsusastmetele sobivate mõõdikutega on toodud välja lisades [Lisa 2].

Telia Eesti AS tehtu vastab sellele soovitusele. Ettevõttes olid eksisteerivad mõõdikud tegevuste kvaliteedi hindamiseks. *DevOps* muudatuse kontekstis kehtestati struktuuriüksustele üldisemad ning spetsialistidele personaalsed uued mõõdikud, mille järgi prioriteete seati. Kasutati *DevOps* madalama küpsustaseme mõõdikuid nagu näiteks teenuseseisakute hulka ja teenustaseme kokkulepete täitmist.

3.7 Tarneahela automatiseerimine

Tarneprotsess käsitleb tervikahelat, mis koosneb arendusvajaduse kaardistamisest, nende vajaduste arendamisest, testimisest ja arendatud funktsionaalsuse toodangukeskkondadesse paigaldamisest. Sõltuvalt organisatsiooni tehnoloogilisest küpsustasemest võivad olla osad arenduse- või haldustegevuste protsessid automatiseeritud.

PuppetLabs väljastas 2016. aastal uuringu "*2016 State of DevOps Report*" [9] kus küsitleti 4600 spetsialisti üle maailma. Kaardistati tarnimise võimekust edasijõudnute, keskpäraste ning algajate ettevõtete hulgas *DevOps* kontekstis. Koguti statistikat tarnimise sageduse, tarnimisest tingitud katkestuste ning katkestuste pikkuse kohta.

Uuringus kogutud andmete põhjal kirjeldatakse raportis meetodit, kuidas võrrelda erinevate küpsusastmete tarne kulukuse erinevust.

Katkestuse maksumuse arvutamise valem on järgmine:

Katkestuse maksumus = tarnimise sagedus × muudatusest tingitud katkestuse hulk × teenuse taastamiseks kuluv aeg × katkestuse maksumus tunnis

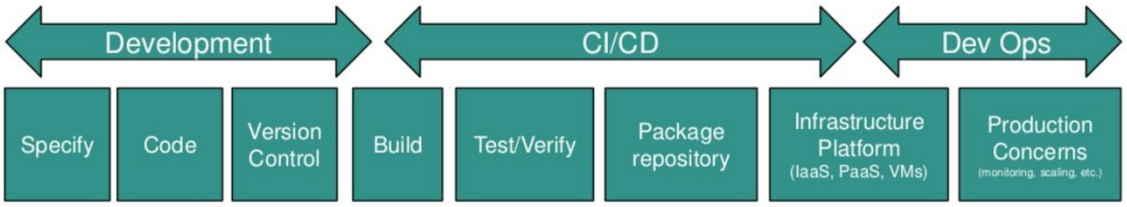
	Edasijõudnud	Keskpärased	Algajad
Tarnete sagedus aastas	1460	32	7
Muudatusest tingitud katkestuste hulk	7,5%	38%	23,5%
Teenuse taastamiseks kuluv aeg	1 tund	24 tundi	24 tundi
Katkestuse maksumus tunnis, keskmine	$500000	$500000	$500000
Katkestuste maksumus aastas (miljonites dollarites)	54,75	145,92	19,74
Katkestuse maksumus tarne kohta (tuhandetes)	37,5	4560	282

Tabel 4 Erinevused tarnimisega seotud katkestuste maksumuses [9]

Tabelist 4 võib järeldada, et sagedasem funktsionaalsuse tarnimine on odavam nii katkestuste hulga, kestuse kui maksumuse poolest. Raporti kontekstis eristati algajad ettevõtteid edasijõudnutest selle poolest, kui hästi oli toetatud pideva tarnimise võimekus. [9]

Tarneahel koosneb alamsüsteemidest nagu versioonikontrolli süsteem, komponentide repositoorium, tarnehaldussüsteem, rakenduste ning platvormide haldussüsteem,

tarnimise vahendid, monitooring jne. DevOps'i ideoloogia toetub suuresti tarneahela omavaheliste seoste automatiseerimisele API-de abil.



Joonis 9 Tarneahela etapid [13]

Joonisel 9 kujutatud tarneahela etapid eksisteerivad kõikides tarkvaraarendusega tegelevates ettevõtetes. Mõningate etappide teostatuks lugemine (koodi valideerimine, testitulemuste hindamine, toodangukõlbulikuks tunnistamine) võib tähendada nelja silma printsiibi (ingl. k. *code review*) kasutamist või eraldi kooskõlastusi.

Probleemid saavad alguse, kui erinevate etappide eest on pandud vastutama erinevad osakonnad, kellel on erinevad arusaamad tarneprotsessist. Lisaks võivad neil olla vastandlikud prioriteedid ja mõõdikud. Tegevust muudab keerulisemaks ning ajamahukamaks protsessi käigus ühest etapist teise liikumisel kooskõlastusringide läbimine ning manuaalsete protseduuride kasutamine.

DevOps metoodika kasutamise edukus tugineb tarneprotsesside automatiseerimisel. Tarneahela automatiseerimise eesmärgiks on muuta funktsionaalsuse paigaldamine toodangusse rutiinseks ja minimaalse inimese sekkumisega tegevuseks. Sagedasel tarnimisel on mitmeid positiivseid tagajärgi. Märkimisväärseimad neist on kiiremad turuvajadustele reageerimise võimekus, väiksem muudatustest tekkinud vigade hulk ning kokkuvõttes väiksemad kulutused arendusele.

Tarneahela automatiseerimise abil luuakse üks teekond kõikideks toodangu muudatusteks, olgu selleks siis rakenduse kood, andmebaaside skeemid, süsteemide konfiguratsioon või infrastruktuur. Modelleeritakse protsess süsteemide ehitamiseks, testimiseks ning juurutamiseks.

Arendustegevuste vaatevinklist tarneahela automatiseerimise abil tegeletakse peamiselt olemasolevate protsesside optimeerimisega või automatiseerimisega. Operatsioonisüsteemide või platvormide haldamises kaasnevad suuremad protsessilised

muudatused. Traditsiooniliselt on operatsioonisüsteemide konfigureerimise teostajateks süsteemiadministraatorid. Arendajate poolt loodud juhendite alusel tegelevad nad süsteemide loomisega, paikamisega ning vigade tuvastamisega.

Automatiseeritud tarneahelat kasutavas organisatsioonis operatsioonisüsteemide haldust sellisel kujul ei toimu. Nii süsteemiadministraatoritel kui arendajatel ligipääsuõigusi serveritesse ei ole, seda võimaldatakse erandjuhtudel intsidentide lahendamiseks. Nii süsteemide kui rakenduste koodi hoitakse versioniseerimisvahendis ning rakendatakse automaatselt konfiguratsioonihaldusvahenditega.

Sellise töökorralduse abil on nii serverid kui terved keskkonnad reprodutseeritavad programmaatiliselt alati vajaliku konfiguratsiooniga, võimaldades väikesel meeskonnal hallata tuhandeid servereid.

Kuigi peamiselt käsitletakse versioonihalduse kasutamist tarkvaraarenduse kontekstis, siis taristu automatiseerimisega kaasnevat arendustegevust ei tohiks käsitleda erinevalt.

Infrastruktuuri arendusega kui haldamisega seotud muudatused (kood, skriptid, automaattestid, konfiguratsioonifailid) tuleb talletada kesksetesse versioonihalduse keskkondasse.

PuppetLabs "*2014 state of DevOps Report*" raporti järgi versioonihalduse kontrolli kasutamine haldusmeeskondade poolt on parim indikaator määramaks IT ning organisatoorset võimekust teostada pidevat tarnimist. [46]

Juhul kui rakenduse ning keskkonna muutujad on salvestatud versioonihalduse- ning versioonikontrollisüsteemi, pakub see lisaks muudatuste kiirele tuvastamisele ka võimalust naasta süsteemi eelmisele versioonile. See omakorda lubab taastada kiirelt teenuse töö.

Automaatinfrastruktuuri kasutamisega on tekkinud kasutusse termin "*immutable infrastructure*" Chad Fowler'i poolt[16], mis üldistatuna tähendab, et (toodangu)keskkondades muudatusi sisse ei viida, vaid vajadusel asendatakse infrastruktuuri komponent (nt operatsioonisüsteem, rakenduse versioon jms) uuega, mis

evib sobivat funktsionaalsust. Josh Stella toob välja võrdluse [43] käsitsi seadistatava ning automaatinfrastruktuuri vahele:

Käsitsi seadistatav infrastruktuur	Automaatinfrastruktuur
Suur operatiivne keerukus (uuendamine, seadistamine jms)	Kergem süsteeme provisjoneerida või uuendada
Aeglasem süsteemide kasutusele võtt, vigadele altid protseduurid	Pidev muudatuste paigaldamine toodangukeskkondadesse, lühema tsükliga
Vigasid ning ohtusid peab tuvastama ennetamaks nende poolt tekitatud kahju	Vähendab vigasid ning ohte (ka tuvastamata või raporteerimata), kuna instantse genereeritakse sageli. See vähendab konfiguratsiooni erinevusi, haavatavust ning teenustaseme kokkulepete täitmiseks tehtavaid pingutusi
Teenuse katkemine juhul kui alusinfrastruktuuris mingisuguseid katkestusi esineb	Automaatse teenuse taastamise meetoditega väheneb sõltuvus alusinfrastruktuurist ning selle katkestuste mõju teenuste tööle

Tabel 5 Võrdlus automaatselt ja käsitsi seadistatava infrastruktuuri vahel

Infrastruktuuri peab olema võimalik kergemini uuesti ehitada kui parandada. Massimo Re Ferrè toob paralleeliks serverite tekitamise ning parandamise vahel kui erinevuse lemmikloomade ning kariloomade vahel. Esimeste puhul on tegemist personaliseeritud serveritega, millel on loogilised või teenusele vastavad nimed. Kui juhtub intsident siis üritatakse serverit läbi parandamise taastada. Teisel puhul kasutatakse struktureeritud umbmääraseid nimetusi või koode ning probleemide tekkimisel server pigem kustutatakse ning tekitatakse uus, kui et kulutatakse aega probleemi parandamisele või varukoopiast taastamisele. [36]

Võrdluse võib taandada sellele, et kui üks või mitu serverit muutuvad kasutamatuks ning lõppkasutajaid teenuse tarbimisel see ei mõjuta, on tegemist “kariloomadega”. Juhul kui see põhjustab märkimisväärse intsidendi teenuse töös, siis on tegemist “lemmikloomadega”.

Kuigi antud töös ei keskenduta tehnoloogilistele vahenditele või toodetele, siis järgnevalt toob autor välja lühikese nimekirja tehnoloogiatest, mida kirjutamise hetkel loetakse automatiseeritud tarneahela juurutamise võimaldajateks.

Puppet Labs eristab 5 kategooriat koos tootenimedega [19]:

- Versioonihaldus (GitHub, Mercurial, Perforce, Subversion, Team Foundation Server)
- Konfiguratsioonihaldus (Puppet, SaltStack, Ansible, Chef)
- Pideva integratsiooni vahendid (Atlassian Bamboo, Go, Jenkins, TeamCity, Travis CI)
- Juurutamine (Capistrano, Mcollective)
- Monitooring (New Relic, Nagios, Splunk, Elastic, Loggly, AppDynamics)

Tehnoloogiliste vahendite kasutuselevõtu juures peab arvestama ettevõtte IT strateegilisi suundasid ning IT arhitektide poolt välja pakutud tehnoloogiaid. Antud töö skoobis ei ole sellesuunaliste soovituste andmine.

Telia Eesti vastab osaliselt sellele soovitusele. Tugevusena võib välja tuua konteinerlahenduse pilootprojekti, mis võimaldab tarneahela automatiseerimist. Samas olemasolevat rakenduste arhitektuur arvestades selle kasutatavus oli piiratud. Puudus tegevuskava viimaks olemasolevate rakenduste arhitektuuri konteinerlahendustele sobivaks. Seetõttu oli raskendatud funktsionaalsuse konteineritena tarnimine.

Nõrkusena võib välja tuua prioriteetsetes projektides tarneahela automatiseerimise puudumise. Kuigi erinevad valdkonnad tegelesid oma protsesside automatiseerimisega, siis puudus üleettevõtteline initsiatiiv automatiseerimaks tarneahelat tervikuna. Servereid käsitleti kui “lemmikloomi”. Infrastruktuuri kui IaaS teenuse pakkumine oli osaliselt automatiseeritud. Serverite tarnimine oli küll automatiseeritud, kuid rakenduste ning konfiguratsiooni paigaldamine eeldas endiselt manuaalseid protsesse. Seetõttu oli süsteemide tarnimine komplitseeritud protsess, kus süsteeme või keskkondi ei saanud parameetritele vastavalt automaatselt luua. Olemasolevale rakenduste arhitektuurile sobiva automatiseeritud tarneahela loomine võimaldanuks teenuste eest vastutavatel

meeskondadel oma protsesse oluliselt optimeerida ja pideva tarnimise printsiipide abil tarnete sagedust tõsta.

3.8 Rakenduste ja tehnoloogiliste komponentide kataloogi juurutamine

Rakenduste ja tehnoloogiliste komponentide kataloog võib olla raamistiku juurutamisel kasutusele võetud protsess või informaalselt kokku lepitud. Antud punkt on *DevOps*'i temaatika juures oluline, sest meeskondadele võimaldatakse suuremat otsustusvabadust oma vastutusvaldkonna piires tegutseda. See aga võib kaasa tuua ettevõttes kasutatavate tehnoloogiate mitmekesistumise, kuna meeskonnaliikmed hakkavad isiklikele või muudele eelistustele tuginedes valima tehnoloogilisi vahendeid.

Rakenduste ja tehnoloogiliste komponentide kataloogi olemasolu on oluline teenustaseme tagamise jaoks, mis võib muutuda näiteks seoses spetsiifilist tehnoloogilist kompententsi omavate inimeste lahkumisega.

Rakenduse ja tehnoloogiliste komponentide kataloog tugineb TOGAF 9.1 arhitektuurilise raamistiku tehnoloogia portfoolio kataloogil [3]. TOGAF'i järgi on kataloogis kajastatud üle ettevõtte kõik tehnoloogiad sh. riistvara, platvormide tarkvarad ning rakenduste tarkvarad. Portfooliole tuginedes moodustuvad ettevõtte tehnoloogilised standardid.

Olemuslikult on kataloog tehnoloogiate elukaarte juhtimist abistav vahend, kus on kirjeldatud eelistatud, lubatud, keelatud ning uurimise all olevad tehnoloogiad ja/või tarkvarad. Lisaks on kirjeldatud elukaare lõpus olevad tehnoloogiad, mis aitab teenuse elutsüklit planeerida. Sellele kirjeldusele tuginedes saavad meeskonnad valida oma tööks vajalikke tarkvarasid või tehnoloogiaid. Kataloogi jõustamine aitab vähendada riski, kus ettevõte pakub teenust, mille komponentide haldamise kompetentsi ei eksisteeri. Kataloogi kasutamise korral on võimalik teistest meeskondadest ajutiselt spetsialiste määrata teenust haldama.
Kataloogi väärtus tugineb sellele, et valitud tehnoloogiad jõuavad nimekirja läbi laiapõhjalise arutelu, kuhu on kaasatud nii halduse spetsialistid, arendajad, arhitektid kui ärilised tellijad. See tagab tehnoloogiate valiku juures vajaliku analüüsi ning võrdluse

teiste, konkureerivate toodetega. Lisaks saavutatakse seeläbi sidusus IT strateegiliste eesmärkide tätmisega ning struktureeritud tehnoloogiate tutvustamisega ettevõttesse.

Alljärgneval on esitatud rakenduste ja tehnoloogiliste komponentide kataloogi näidis:

Tehnoloogia: **Rakenduskiht**	**Eelistatud**	**Aktsepteeritav**	**Mitte valida**	**Kommentaarid**
Rakenduste raamistikud	• Java: Spring MVC • Python: Pyramid • JS: AngularJS	• Ruby: Rails • Microsoft .Net	Oracle APEX, Macromedia Flex, GWT	Javascripti puhul kasutada Yeoman, Karma, Jasmine tööriistu
Rakendus-serverid	• JBOSS • uWSGI	• Apache Tomcat • WebLogic • Jetty • .NET: MS IIS	Java EE: Apache JServ, Sun Java System Application Server, SAP, IBM WebSphere, Oracle iAS, Oracle iPlanet Web Server	WebLogic elukaare lõpustaadiumis, kahe aasta pärast mitte valida kategoorias
Tehnoloogia: **Infrastruktuuri kiht**	**Eelistatud**	**Aktsepteeritav**	**Mitte valida**	**Kommentaarid**
Serveri operatsiooni-süsteemid	Linux: CentOS, Debian Windows Server 2016	Linux: RedHat Windows Server 2012R2	Solaris HPUX BSD	
Virtualiseerimis platvormid	VMware	Hyper-V XEN	OracleVM KVM	

Tabel 6 Rakenduste ning tehnoloogia komponentide kataloogi näidis

Autori poolt näidisena koostatud tabeli 6 esimeses veerus paiknevad arhitektuurikihtide tüübid. Nende määratlemine aitab grupeerida tehnoloogiaid teemade kaupa. Teises veerus on kirjeldatud eelistatud tehnoloogiate valikud. Kolmandas veerus on kirjeldatud aktsepteeritavad, kuid mitte soovituslikud valikud. Nende esiletoomine on tähtis, sest vastavad kõige paremini ettevõtte ärivajadustele ning eksisteerivatele kompetentsidele. Tabeli neljandas veerus paiknevad ettevõttes kasutamiseks mittelubatud tehnoloogiad. Neid põhjuseid, miks ettevõte teatud tehnoloogiaid kasutada ei luba, on erinevaid. Näitena võib tuua litsentsi tingimused.

Telia Eesti AS vastab sellele soovitusele. Ettevõttes oli kasutuses rakenduste ja tehnoloogiliste komponentide kataloog. Selle haldamine oli arhitektide vastutusalas, uusi tehnoloogiaid aktsepteeriti arhitektuurinõukogu otsusega. Kasutuses olev protsess tagas, et ettevõttes lubatud tehnoloogiad olid läbinud võrdleva analüüsi alternatiividega ja nende vastavus äriprotsesside toetamiseks oli kooskõlastatud.
Nõrkusena võib samas välja tuua sisulise kontrolli puudumise kataloogi poolt kehtestatud nõuete täitmisele. Rakendamist korraldati protsessiliselt, vahendeid ülevaate saamiseks kehtestatud nõuete rikkumise kohta juurutatud ei olnud.

3.9 Ettepanekud *DevOps* juurutamiseks peatüki kokkuvõte

Peatükis toodi välja organisatsioonijuhtimise ning planeerimisega seotud tegevused, mis on eelduseks ja aitavad kaasa *DevOps* metoodika juurutamisele suurettevõttes.

Tegevused on välja toodud alljärgnevas tabelis.

Organisatsiooni kultuuri muutmine	Aitab ette valmistada organisatsiooni *DevOps* mudeli kasutuselevõtuks
DevOps küpsuse hindamine	Võimaldab välja selgitada vajalikud kompetentsid ning koolitusvajaduse
Muudatuse skoobi määramine	Annab ette raamid, mille piires alustada
DevOps projektirühma loomine	Määrab initsiatiivi eest vastutajad
Rollide ja kompetentside jaotamine	Meeskonnamudel ning teenustepõhise organisatsiooni printsiibid
Mõõdikute seadistamine	Tegevuse edukust hindav meetrika
Tarneahela automatiseerimine	*DevOps* tegevust toetav protsess
Tehnoloogia komponentide kataloog	Erinevate valdkondade kokku juhtimist hõlbustav meetod

Tabel 7 Tegevused organisatoorseks *DevOps* muutuseks

Pakutud soovitusest on kõik rakendatavad Telia Eesti AS puhul. Telias tehtud muudatused kas ei vastanud, vastasid osaliselt või vastasid täielikult soovitustele. Samas on töö kaitsmise hetkeks ettevõte realiseerunud kitsaskohtadele vastumeetmeid tarvitusele võtmas või siis need lahendanud.

Leiud on rakendatavad analoogsete suurettevõtete puhul, kus planeerimist *DevOps* metoodika katsetamiseks alles alustatakse.

Kuigi soovitustes ei ole eraldi välja toodud tegevusriskide analüüsi, siis on selle teostamine oluline, kuid sõltub spetsiifilisemalt nii skoobist kui ettevõtte eripäradest, mistõttu seda ettepanekute juures ei ole eraldi välja toodud.

4 Arutelu

Peatükis tehakse tagasivaade tehtud tööle. Vaadatakse laiemalt järeldusi milleni jõuti ning vaadeldakse võimalikke edasiarendusi antud vallas.

4.1 DevOps fenomen

Tööd kirjutama asudes seadis autor endale eesmärgiks leida lahendus küsimusele, mida suurettevõtte juhtkond peab hakkama lahendama, soovides *DevOps*'i kui uut võimalikku metoodikat juurutada ettevõttes.

Temaatika uurimust raskendab kirjutamise hetkel termini haibi kõrgfaasis olek. Rohujuuretasandi liikumisest alguse saanuna puudub ühtne ning selge raamistik või arusaam, mis asi on *DevOps*. Hästi sobib hetkeseisu illustreerima Ed Baker'i blogipostituses *Demystifying DevOps Behaviours* toodud pilt:

Joonis 10 *DevOps* illustratsioon Ed Bakeri kohaselt [4]

Puuduv definitsioon ning erinevates ettevõtetes selle töömetoodika juurutamise kas õnnestumised või ebaõnnestumised süvendavad segadust veelgi. *DevOps*'i peetakse sõltuvalt allikast kaasaegsete tehnoloogiate kasutuselevõtuks organisatsioonis, arendusmeeskondadesse haldustegevuste integreerimiseks, suhtluse parandamiseks

erinevate osapoolte vahel, paindlikuks metoodikaks arendamises, haldamises ja protsesside automatiseerimises.

Metoodika rakendamine suurettevõtetes on raskendatud ka seetõttu, et muudatuse initsiaatoritel on erinev arusaam protsesside, vastutusvaldkondade ja struktuurimudelite kasutuselevõtust. Tihtipeale on kogu muudatuse juhtimine kallutatud subjektiivsetest tõlgendustest või eestvedajate kompetensidest.

Olukorda ei tee lihtsamaks turul osalevate tehnoloogia ning teenuste ettevõtete soov järjekordset haipi ära kasutada. *DevOps* segast definitsiooni on võimalik rakendada paljude halduse või arendamisega seotud tehnoloogiate turustamiseks.

Olemuslikult ei ole *DevOps* midagi revolutsioonilist. Väikeettevõtetes, kus ressursside piiratuse tõttu ei ole võimalik mehitada kõiki IT rolle ning tekitada protsesse järgivate kompetentside eraldatust, on rakendatud *DevOps* printsiipe aastakümneid. Resoneerima on jäänud mõte, et subjekti näol püütakse lahendada suurettevõtetes ebaõnnestunud protsesside automatiseerimise ja arhitektuuri ning lahenduste arhitektuuride printsiipide juurutamist. *DevOps*'i pakutakse lahendusena kommunikatsiooniprobleemidele, vananenud praktikatele ja paindlikkust ning tootlikust mitte toetavatele protsessidele.

Pärast valkonna uurimist on autor jõudnud seisukohale, et *DevOps* kui *Developement and Operations* asemel on paslikum kasutada sõnapaari *Developing Operations* ehk süsteemide halduse arendamine. Infrastruktuuri haldamine ning arendamine on liikumas paindlikke tarkvaraarenduspraktikaid paremini võimaldavasse etappi. Teenused, mida paindlike praktikate abil luuakse, ei ole toetatud samasuguste paindlike printsiipide järgi ettevõtetes eksisteerivas infastruktuuri kihis. Seega probleemi raskuskese langeb pigem infrastruktuui kaasajastamisele.

Võib väita, et läbiv ühine joon kõikide *DevOps* initsiatiivide juures on keskendumine infrastruktuuri automatiseerimisele tasemel, kus arendajad saaksid oma tegevusi teha minimaalsete takistustega. Ettevõtete struktuuride muutmine, kompetentside juhtimine, kommunikatsiooni parandamine, protsesside ümbertöötamine on kõik seda toetavad tegevused.

On eksitav pidada *DevOps* lähenemist millekski, kus arendajad ise peavad haldama alussüsteeme. Kuni tehnoloogia pole arenenud tasemele, kus kõik on automatiseeritud, vajavad tarkvaraarendajad partnerit platvormide ning toetavate funktsioonide valdkonnas. Klassikaliselt on see funktsioon eksisteerinud, kuid toimimismudel liigub teenuse osutamise suunas.

Ideaalis peab haldusmeeskond ehitama süsteemi ja tööriistad, mis võimaldavad arendajatel täielikku iseteenindust koodi tarnimisele nii testimis- kui toodangukeskkondades. Lisaks peab süsteem võimaldama meetrikaid tervele organisatsioonile.

Arvestades kaasaegset pilveteenuste populariseerimist seab see ettevõtete haldusüksustele suure väljakutse. On keeruline mõnekümne haldusspetsialistiga ehitada, arendada ning liidestada võrdväärset platvormiteenust Google, Amazoni või Microsofti pilveteenustele, tagades samal tasemel jätkusuutlikkust, skaleeruvust või stabiilsust. Pilveteenuse pakkujatelt automatiseeritud tarneahelat teenusena sisse osta jääb alati alternatiiviks, mille analüüsil käesolevas töös ei peatuta.

4.2 DevOps HP Software näitel

Tekitamaks võrdlusmomenti oma töös tehtud leidudega toob autor välja mittestruktureeritud intervjuu [25] Hewlett-Packard Enterprise Software (HP Software) *DevOps* evangelisti Olivier Jacques'iga. Intervjuu ei olnud planeeritud, mistõttu ei olnud võimalik selle läbi viimiseks kasutada struktureeritud küsimustikku. Sisuliselt andis Olivier ülevaate, mida viimase kahe aasta jooksul oli teostatud HP Software's *DevOps* metoodika sisseviimisega ettevõtte tööprotsessidesse.

Hewlett-Packard Enterprise on üks suurimatest tehnoloogiaettevõtetest maailmas, seal töötab kokku 350 000 töötajat[20]. Ettevõttel on eraldi divisjon 14000 töötajaga, kes on keskendunud tarkvaratoodete arendamisele ning juurutamisele.

DevOps juurutamise katalüsaatoriteks olid järgmised aspektid:

- 2014. aastal Hewlett-Packard ettevõtte jaotamine kaheks ettevõtteks, mille tulemusena tuli ümber hinnata IT funktsioonide jaotamine või dubleerimine uutes ettevõtetes.
- Juhtkond teadvustas vajadust kohaneda kiirete muudatustega tarkvaraarenduses ning olemasolevate protsesside võimetust seda toetada.

DevOps algatust ei käivitatud ettevõtte ülese juhtkonna korraldusena. Moodustasti erinevate valdkonande spetsialistidest koosnev väike töörühm, kes sai järgmised ülesanded:

- kaardistada erinevad üksused, kus juba kasutati mingisuguseid automatiseerimise vahendeid
- propageerida organisatsioonisiseselt automatiseerimisvahendeid
- integreerida ja uurida erinevaid vahendeid loomaks automatiseeritud tarneahelat

Tegevuskavana prioritiseeriti iga osakonna tarneahela disainimist faaside kaupa, alustades pidevast integreerimisest ja testimisest, liikudes edasi pideva tarnimiseni. Protsessi igal sammul tuli automatiseerida nii palju kui võimalik. Oluliseks peeti kõiki huvitatud osapooli igal sammul kaasata.

Eesmärgiks võeti luua tarneahela automatiseeritud standard, mis muutus kohustuslikuks järgida *DevOps* transformatsiooni läbivatele osakondadele. Olulise faktorina nähti erinevate töölõikude või protsesside kokkupuutepunktide automatiseerimist läbi API liideste. Liideste olemasolu oli oluline seetõttu, et meeskondadele, kes olid juba oma vahendid kasutusele võtnud, pakuti kompromissina nende vahendite liidestamist kokkulepitud tarneahela vastava lõiguga.

Kuna HP Software toodab erinevaid tarkvarasid, ei seatud eesmärgiks üleettevõttelist kohustuslikku tehnoloogiate komplekti. Rõhutati pigem tarneahela standardi olulisusele, sest see pidi vastama kõikide huvitatud osapooli puudutavatele nõuetele. Peale tarkvaraarendamise ning juurutamise aspektide võeti arvesse nõudeid turvalisusele ning vastavust lepingulistele kohustustele ettevõtte partneritega.

Olivier'i meeskond sai ülesandeks aidata kõige suurema *DevOps* potentsiaaliga arendusmeeskondi viia üle automatiseeritud tarneahelale.

Olulisena tõi Olivier välja piirangu, kus tarneahelat kasutavad meeskonnad toodangukeskkonnas asuvatele serveritele administratiivset ligipääsu ei omanud. Kõik muudatused tuli teostada läbi automaatset paigaldamist võimaldavate tööriistade.

Kõiki muudatustöid, sõltumata korralisest või erakorralisest iseloomust, käsitleti kui harilikke ITIL'i protsessis defineeritud muudatustöid.

Intervjuu toimumise ajal ei täpsustatud, kui palju teenuseid või osakondi oli *DevOps* meetodile viidud, kuid projekti loeti edukaks. Mõõdikutest paranesid arenduse kiirus 7% ning testimise kiirus 40%.

Projekti jätkusuutlikkuse tagamiseks ning kompetentside edastamiseks oli kokku lepitud kord, kus igast uuest *DevOps* osakonnast pidi üks inimene osalema järgmise teenuse elutsükli juhtimise üleviimist automatiseeritud tarneahelale.

Märkmed vestlusest on toodud välja töö lisas [Lisa1].

4.3 Tulemuste analüüs

Töö tulemused keskenduvad rohkem organisatoorsetele tegevustele ning vähem tehnoloogilistele aspektidele, kuigi 3.7 "Tarneahela automatiseerimine" ja 3.8 "Rakenduste ning tehnoloogia komponentide kataloogi juurutamine" peatükkides käsitles autor muudatuse tehnoloogilisemat külge.

Kokku koosneb töö tulem kaheksast punktist, kus käsitletakse *DevOps* juurutamise organisatoorseid aspekte, loomaks eelduseid selle metoodika laialdasemaks kasutuselevõtuks ettevõttesiseselt. Töö tulemina esitatud tegevused on aluseks ettevõttele teenuste eest vastutavatele üksustele suurema autonoomsuse võimaldamiseks. Eksisteerivat organisatsiooni struktuuri tingimata muutma ei pea,

piisab osade eksisteerivate funktsioonide ning protsesside kohaldamisest, eriti piloteerimise faasis.

Töö tulemit võib kritiseerida selle tehnoloogiakaugsuse pärast nii tehnoloogiakeskses valdkonnas. Samas teostatakse tehnoloogia rakendamist läbi inimeste, nende harjumuste kujundamise ja teadmiste kanaliseerimise.

Käesolev uurimus peatub seal, kus algab tarneahela arhitektuuri disaini faas ning tehnoloogiate valiku analüüs. Autor leiab, et nende tegevuste eduka läbiviimise eelduseks ongi käesolevas töös kaardistatud tegevused, mis loovad soodsa õhkkonna ning teadliku valmiduse evolutsiooniliselt järgmisele tasemele areneda.

4.4 Võimalikud edasiarendused

Antud magistritöö on esitatud uurimistööna.

Autor näeb ühe võimaliku ja vajaliku edasiarendusena töö tulemina esitatud tegevuste testimist ning tulemuste pealt analüüsi teostamist ühe või mitme suurettevõtte *DevOps*'ile ülemineku projekti näitel.

Teise võimaliku edasiarendusena analüüsida tarkvaraarendusele tekkivat mõju *DevOps* printsiipide laialdasema kasutuselevõtuga. Kuna *DevOps* metoodika keskendub tarneahelale tervikuna, siis selle etappide automatiseerimine toob kaasa muudatusi ka arendusprotsessides.

Kokkuvõte

Käesolevas magistritöös käsitleti *DevOps* metoodika kasutuselevõtuga kaasnevate probleemide ennetamist suurettevõtetes.

Eesmärgiks seati välja töötada tegevuste kava, mille kasutamine looks soodsamat keskkonda *DevOps* metoodika kasutuselevõtuks.

Töös kasutati kvalitatiivset uurimismetoodikat. Autor teostas eelanalüüsi Telia Eesti As *DevOps* struktuurimuudatusele, tõi välja muudatusejärgsed kitsaskohad.

Uurimustöö tulemusena koostas autor ettepanekute nimekirja, mis koosneb kaheksast punktist. Iga punkti juures analüüsiti rakendatavust Teli Eesti AS puhul. Töö rõhuasetus on pigem organisatoorsetel meetmetel, mitte niivõrd tehnoloogilistel.

Lõpetuseks käsitleti *DevOps*'i kui fenomeni. Analüüsiti tulemusi, võrreldi neid maailmapraktikatega. Toodi välja võimalikud edasiarendused.

Kasutatud kirjandus

1. 2016 IT Budget Benchmark 2016 [WWW] https://www.cebglobal.com/content/dam/cebglobal/us/EN/best-practices-decision-support/information-technology/pdfs/ceb-cio-budget-benchmark.pdf (08.04.2017)

2. Amaradri, A.S., Nutalapati, S.B. Continuous Integration, Deployment and Testing in DevOps Environment: magistritöö. Karlskrona, Blekinge Institute of Technology, 2016. (09.03.2017)

3. Architectural Artifacts [WWW] http://pubs.opengroup.org/architecture/togaf9-doc/arch/chap35.html (11.04.2017)

4. Baker,E. Demystifying DevOps Behaviours [WWW] https://blogs.technet.microsoft.com/uktechnet/2015/12/18/demystifying-devops-behaviours/ (22.03.2017)

5. Beck, K., Beedle, M., Bennekum, A., Cockburn, A., Cunningham, W., Fowler, M., Grenning, J., Highsmith, J., Hunt, A., Jeffries, R., Jon, K., Marick, R., ;ellor, S., Schwaber, K., Sutherland, J., Thomas, D. Manifesto for Agile Software Development [WWW] http://agilemanifesto.org/principles.html (01.02.2017)

6. Bourne, V. RESEARCH PAPER: DEVOPS: THE WORST-KEPT SECRET TO WINNING IN THE APPLICATION ECONOMY [WWW] https://www.ca.com/content/dam/ca/us/files/white-paper/devops-winning-in-application-economy-2.pdf (13.03.2017)

7. Bradley, T. Inside the mind of a true DevOps evangelist [WWW] https://devops.com/inside-the-mind-of-a-true-devops-evangelist (17.02.2017)

8. Brown, A. Getting Started with DevOps: Build the Business Case. [WWW] https://puppet.com/blog/getting-started-devops-build-business-case (09.04.2017)

9. Brown, A., Forsgren, N., Humble, J., Kersten, N., Kim, G. 2016 State of DevOps Report Puppet Labs [WWW] https://puppet.com/system/files/2016-06/2016%20State%20of%20DevOps%20Report_0.pdf (11.03.2017)

10. Brown, J. DevOps— DevOps You Build It, You Own It! AWS Government, Education, and Nonprofit Symposium Washington, DC I June 25-26, 2015 [WWW] https://www.slideshare.net/AmazonWebServices/devopsyou-build-it-you-own-it (22.02.2017)

11. Burton, B., Burke, B., Blosch, M. Hype Cycle for Enterprise Architecture, 2016, Gartner. [WWW] https://www.gartner.com/document/3384417 (08.03.2017)

12. Business Process Framework (eTOM) [WWW] https://www.tmforum.org/business-process-framework/ (17.03.2017)

13. Coté, M. CI/CD from a donkey perspective [WWW] https://www.slideshare.net/cote/cicd-from-a-donkey-perspective (08.04.2017)

14. Doran, G.T. "There's a S.M.A.R.T. way to write management's goals and objectives".1981 Management Review. AMA FORUM. 70 (11): 35–36.

15. Edwards, D. THE HISTORY OF DEVOPS [WWW] http://itrevolution.com/the-history-of-devops/ (08.02.2017)

16. Fowler, C. Trash Your Servers and Burn Your Code: Immutable Infrastructure and Disposable Components [WWW] http://chadfowler.com/2013/06/23/immutable-deployments.html (08.02.2017)

17. Franklin Jr., C. Agile Vs. DevOps: 10 Ways They're Different[WWW] http://www.informationweek.com/devops/agile-vs-devops-10-ways-theyre-different/d/d-id/1326121?image_number=1 (09.03.2017)

18. Gartner Hype Cycle [WWW] http://www.gartner.com/technology/research/methodologies/hype-cycle.jsp (08.03.2017)

19. How to Build a High-Performing IT Team [WWW] https://puppet.com/resources/white-paper/how-to-build-a-high-performing-it-team/thank-you (27.03.2017)

20. HP people [WWW] http://www.hp.com/hpinfo/globalcitizenship/media/files/hp_fy11_gcr_hp_people.pdf (21.03.2017)

21. Humble, J., Molesky, J. Why Enterprises Must Adopt Devops to Enable Continuous Delivery, in: Cutter IT Journal, 2011, Vol. 24, No. 8

22. Humble, J., Molesky, J., O'Reilly, B. Lean Enterprise: How High Performance Organizations Innovate at Scale. Sebastopol, CA: O'Reilly Media. 2015

23. ISO/IEC 27000 family - Information security management systems [WWW] https://www.iso.org/isoiec-27001-information-security.html (08.04.2017)

24. J. Hamunen. Challenges in Adopting a Devops Approach to Software. Development and Operations: magistritöö. Aalto University School of Business, Aalto,2016

25. Jacques, O. (2016), Autori intervjuu. London, 30. November

26. Jeremiah, J. Survey: Is agile the new norm? [WWW] https://techbeacon.com/survey-agile-new-norm (17.03.2017)

27. Kenefick, S. Use Metrics to Drive Your Agile, DevOps and Continuous Delivery Initiatives [WWW] https://www.gartner.com/document/3168323 (13.03.2017)

28. Kotter, J.P. Leading Change. Boston, MA: Harvard Business School Press. 1996

29. Lucas-Conwell, F. Technology Evangelists: A Leadership Survey [WWW] https://www.gri.co/pub/res/pdf/TechEvangelist.pdf (14.04.2017)

30. McChrystal, S., Fussell, C. Team of Teams: The Power of Small Groups in a Fragmented World. Portfolio Hardcover. May 12th 2015

31. Meyer, M. What Adopting Blameless Post-Mortems Has Taught Me About Culture [WWW] http://www.paperplanes.de/2014/6/20/what-blameless-postmortem-taught-me.html (12.02.2017)

32. Mueller, E. What Is DevOps? [WWW] https://theagileadmin.com/what-is-devops/ (01.02.2017)

33. Partogi, J. Who are the Professional Scrum Developers? [WWW] https://www.scrum.org/resources/blog/who-are-professional-scrum-developers (09.04.2017)

34. Paulk, M.C., Weber, C.V., Curtis, B., Chrissis, M.B. (February 1993). "Capability Maturity Model for Software (Version 1.1)" (PDF). Technical Report. Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University. CMU/SEI-93-TR-024 ESC-TR-93-177

35. Prereira, S. THE DEVOPS CHECKLIST [WWW] http://devopschecklist.com/ (01.04.2017)

36. Re Ferré, M. vCloud, OpenStack, Pets and Cattle [WWW] http://www.it20.info/2012/12/vcloud-openstack-pets-and-cattle/ (02.03.2017)

37. Rehn, A., Palmborg, T., Boström, P. The Continuous Delivery Maturity Model [WWW] https://www.infoq.com/articles/Continuous-Delivery-Maturity-Model (01.03.2017)

38. Riungu-Kalliosaari, L., Mäkinen, S., Lwakatare, L.E., Tiihonen, J., Männistö, T. DevOps Adoption Benefits and Challenges in Practice: A Case Study. Product-Focused Software Process Improvement: 17th International Conference, PROFES 2016, Trondheim, Norway, November 22-24, 2016, Proceedings, LNCS 10027, pp. 590-597. Springer International Publishing, 2016.

39. Saar, K. DevOps in Telia Estonia. Presentatsioon. Tallinn 2016

40. Sattersten, T., Kim, G., Crummer-Olson, R., Broderick-Forster, A. The IT Revolution: DevOps Guide [WWW] http://itrevolution.com/wp-content/uploads/2016/06/itrev-devops-guide-5-2015.pdf (11.02.2017)

41. Sharma, S., Coyne, B. DevOps For Dummies®, 2nd IBM Limited Edition. Hoboken, NJ: John Wiley & Sons, Inc. 2015

42. Stadtmueller, L. Through the DevOps Looking Glass: Learnings from HP's Own Transformation Initiative [WWW] https://www.hpe.com/h20195/V2/getpdf.aspx/4AA5-8744ENW.pdf (08.04.2017)

43. Stella, J. An introduction to immutable infrastructure [WWW] https://www.oreilly.com/ideas/an-introduction-to-immutable-infrastructure (09.03.2017)

44. The DevOps Handbook. How To Create World-Class Agility, Reliability, & Security in Technology Organizations / G. Kim, J. Humble, P. Debois, and J. Wills. Portland, OR: IT Revolution Press. 2016

45. THE LEAN STARTUP METHODOLOGY [WWW] http://theleanstartup.com/principles (10.02.2017)

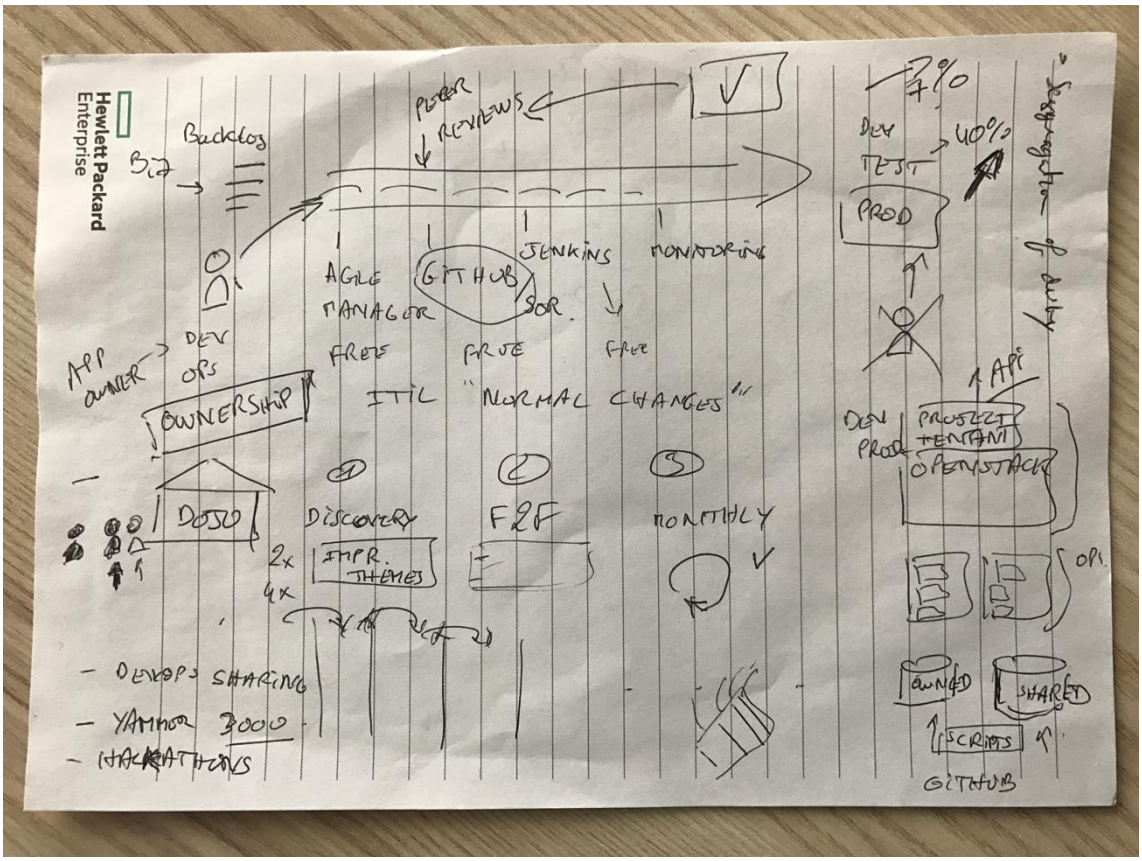
46. Velasquez, N.F., Kim, G., Kersten, N., Humble, J. 2014 State of DevOps report [WWW] https://puppet.com/system/files/2016-03/2014-state-of-devops-report.pdf (01.05.2017)

47. Wagenaar, W. A. & Groeneweg, J. Accidents at Sea: Multiple Cases and Impossible Consequences. In E. Hollnagel, G. Mancini & D. D. Woods (eds.), Cognitive Engineering in Complex Dynamic Worlds (pp. 133-144). London: Academic Press. 1988

48. What is ITIL® Best Practice? [WWW] https://www.axelos.com/best-practice-solutions/itil (08.04.2017)

49. Wilson, J. Essentials of business research: A guide to doing your research project. 2nd ed. Thousand Oaks: SAGE Publications Ltd, 2013

Lisa 1 – Jacques, O. (2016), Autori intervjuu.



Lisa 2 – Mõõdikud vastavalt küpsustasemele

Tabel 8 Meetrikad vastavalt küpsustasemetele, [27]

Küpsus-tase	Kasutatavad praktikad	Praktikate parendamiseks	Kasutades järgmisi mõõdikuid
Algaja	Vähesel määral kasutatakse tarkvara arendustsüklit toetavaid vahendeid (näiteks kasutatakse ainult versioonihaldust) ja toetutakse tugevalt käsitsi teostatavatele metoodikatele ning protsessidele.	Luua stabilised meeskonnad Tegeleda arenduse eraldatusega Versioonide märgistamine Automaatsed testid Statilise koodi analüüsid Teenustepõhine arhitektuur Taaskasutatavat ühiktestid	Teenimine investeeringutelt Kasutajate rahulolu Ressursside määramine Tsükli kestus Läbilaskevõime Arendamine vs stabiliseerimine Tulekahjude kustutamine Methodology Autentimiseks vajalike sammude hulk Tarnimiseks vajalike inimese arv Erakorraliste tarne suhe plaanilistesse tarnetesse Katkised tarkvaraversioonid Vigade arv
Keskmine	Arendust toetavate infrastruktuuri haldamise vahendid on kasutuses. Rakendatud on piiratult automatiseerimist ning mõningaid paindlikke praktikaid	Konfiguratsioon kui kood Versioniseeritud andmebaasid Tarneahelad Automaatsed turva- ning jõudlustestid Rakenduse jõudluse monitooring Teenuse virtualiseerimine ja testiandmete	Teenitud äriline väärtus Kasutajakogemus Meeskonna algatused Äriliselt põhjendatud funktsioonid Tsükli pikkus Läbilaskevõime Korratavad protsessid Andmete muutmise ning sünkroniseerimisega seotud vigade Automaatne paketeerimise funktsionaalsus Projektide hulk, mis seotud versioonikontrolli, probleemihalduse ja tarne vahenditega Koodi ja testitulemuste ülevaatused Lähtekoodi standardid ning staatilise koodi analüüside hulk Testidega katvus Testitsüklid, vigade

			leidmise aeg ning lahendamise kiirus Jõudlustestimine
Edasi-jõudnud	Infrastruktuur toetab tarkvaraarendust ning automatiseeritud tarne aspekte Kasutusel on paindlikud ning mõningad DevOps arenduse praktikad	Pidev arendamine ning testimine Infrastruktuur kui kood Automatiseeritud ärilise väärtuse testimine Ristkompetentsidega meeskonnad Katkestuseta tarnimine	Kasutajate poolt omaks võtmine Konfiguratsiooni ja infrastruktuuri automatiseeritud Konfiguratsiooni ning koormusega seotud probleemide hulk Komponentide liidestuste hulk ja keerukus Branching by abstraction Taaskasutatavad komponendid Katkestusteta tarnimiste hulka