TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond
Informaatikainstituut
Infosüsteemide õppetool

DevOps printsiipide rakendamise analüüs TransferWise'i näitel

Bakalaureusetöö

Üliõpilane: Liis Rush
Üliõpilaskood: 101804IABB
Juhendaja: lektor Karin Rava

Tallinn
2016

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.

---------- ----------

(*kuupäev*) (*allkiri*)

Annotatsioon

Käesoleva bakalaureusetöö eesmärgiks on koondada DevOps liikumise organisatsioonilised ja tehnilised printsiibid ning analüüsida nende rakendamist ettevõtte TransferWise näitel. Töö annab ülevaate DevOps printsiipide järgimise võimalikust kasust ja riskidest IT-meeskonnas.

Töös on analüüsitud IT arendus- ja infrastruktuuritiimide vahelisest barjäärist tulenevaid suhtlusprobleeme ja tehnilisi ebakõlasid ning pakub antud probleemidele lahendusi DevOpsist lähtuvalt organisatsiooniliste mõtteviisi- ja struktuurimuutuste näol. Tehniliste küsimuste lahendamisel on käsitletud mitmeid DevOpsiga seostatavaid arendusmetoodikaid ja -vahendeid, mis aitavad kaasa süsteemide pidevale integreerimisele, automatiseerimisele ja juurutamisele.

Töö tulemusena on kirjeldatud DevOps liikumise põhiprintsiibid ning vastandatud need TransferWise'i arendusmeeskonna töövõtetega. Töö tulemusest võib järeldada, et TransferWise'i arendusmeetodid ja -vahendid vastavad suures osas DevOpsi poolt pakututele ning sihipärane DevOps printsiipide järgimine ei annaks ettevõttele olulist lisakasu. Samuti ei ole vajalikud DevOpsi poolt soovitatavad tiimidevahelised struktuurimuutused, kuna ettevõtte meeskondade vahel puuduvad DevOpsi poolt kirjeldatud sotsiaalsed barjäärid.

Lõputöö on kirjutatud eesti keeles ning sisaldab teksti 44 leheküljel, 7 peatükki, 12 joonist, 2 tabelit.

Abstract

The aim of this thesis is to draw together the organizational and technical principles associated with DevOps movement and to analyze their implementation in TransferWise. The thesis will give an overview of the advantages and risks related to following DevOps in an IT team.

The thesis will analyze the problems of possible social barriers and miscommunication between development and operations teams and the technical misconceptions resulting from disorganized collaboration between the two parties. The thesis will provide solutions to both the organizational and technical problems through social and structural changes in teams and through providing development methodologies and tools for IT automation and integration to benefit the teamwork and development process.

As a result the thesis describes the main principles of DevOps movement and compares them to the development approach used in TransferWise. As a conclusion it is found that Transferwises development methods and tools meet the concepts of DevOps for the most part without a purposeful following of the principles. It is also found, that there is no necessity for the structural changes in teams suggested by DevOps due to the lack of social barriers.

The thesis is in Estonian and contains 44 pages of text, 7 chapters, 12 figures, 2 tables., etc.

Lühendite ja mõistete sõnastik

DevOps	***DevOps*** Tarkvaraarendusmeetod, mis peab oluliseks tarkvaraarendajate ja teiste IT-spetsialistide omavahelist koostööd ja suhtlust kogu arendusprotsessi vältel. Tehnilisest aspektist on põhirõhk arendusprotsessi automatiseerimisel ja pideval integreerimisel.
(IT) süsteemihaldus/ infrastruktuuritiim	***IT operations, IT Ops, Ops*** Üldmõiste IT-teenuste ja protsesside kohta, mis ei ole otseselt seotud tarkvaraarendusega, vaid IT-infrastruktuurisüsteemide halduse ja toega. Kasutatakse ka vastava valdkonna spetsialistide üldnimetusena.
Pidev tarne	***Continuous delivery (CD)*** Pidev tarne tarkvaraarendusprotsessis. Pidev tarne on võimalik tänu agiilsete tarkvaraarenduspõhimõtete järgmisele ja tarneprotsessi maksimaalsele automatiseerimisele.
Pidev integreerimine	***Continuous integration (CI)*** Pidev tarkvarakoodi integreerimine tarkvaraarendusprotsessis. Pideval integreerimisel ühendatakse kõik eri tarkvaraarendajate poolt tehtud koodimuudatused mitmel korral päevas suuremas koodibaasis
Agiilne infrastruktuur	***Agile infrastructure*** IT-infrastruktuuride süsteemihalduse käsitlus, mis juhib tähelepanu vajadusele infrastruktuuri kiirete muutuste järele toetamaks agiilset tarkvaraarendust
Tarkvara väljalase	***Software release*** Uue tarkvarafunktsionaalsuse või -täienduse avalikustamine lõppkasutajatele

Tarkvara juurutamine	***Software deployment/deploy*** Uue tarkvarafunktsionaalsuse või -täienduse paigaldamine selle kasutamiseks ette nähtud keskkonda
Mikroteenused	***Microservices*** tarkvaraarhitektuuri stiil, milles keeruline tarkvara on jaotatud ja arendatud väikeste sõltumatute protsessidena, mis suhtlevad omavahel üle arenduskeeltest sõltumatute arendusliideste
Tarkvara tarneahel	***Software delivery pipeline*** Pideva tarkvaratarne protsessi kirjeldav ahel, milles on esitletud tegevused arendustsükli eri etappidel.
Infrastruktuur koodina	***Infrastructure as Code (IaC)*** Programmeeritav infrastruktuur. IT-infrastruktuuri haldamine kasutades kõrgema taseme programmeerimiskeeli
Testimisel põhinev tarkvaraarendus	***Test-driven development (TDD)*** Tarkvaraarendusprotsess, kus võimalikult väiksele tarkvara funktsionaalsusele kirjutatakse eelnevalt tarkvaratest ja seejärel tarkvarakood

Jooniste nimekiri

Tabelite nimekiri

Sisukord

1. Sissejuhatus

DevOps liikumine ja sellega seotud praktikad on viimastel aastatel kogunud suurt populaarsust. Samas on DevOps mõiste paljudele raskesti defineeritav ja mõistetav, mõnedele ka lihtsalt moeväljend. Antud töö koondab endas DevOpsi põhiprintsiibid ja võrdleb neid Eesti ühe edukaima idufirma TransferWise'i tööpõhimõtetega, eesmärgiga selgitada välja DevOpsi kasu praktikas.

1.1 Taust ja probleem

Töö eesmärk on välja selgitada, kas DevOps põhimõtete sihipärane ja süstemaatiline järgmine ning vastavate tehniliste vahendite kasutamine võiks parandada arendusprotsessi efektiivsust.

Töö teema on eelkõige ajendatud autori isiklikust kogemusest IT arendus- ja haldusstiimide vahelise kommunikatsiooni vahendamisel ja tehniliste küsimuste lahendamisel. Töö on vajalik, et analüüsida DevOps liikumise sisu ja välja selgitada töömeetodid ja vahendid arendus- ja haldustiimide vahelise koostöö parendamiseks. Samuti annab töö analüüsitavale ettevõttele võimaluse vaadelda oma töövõtteid DevOps printsiipidest lähtuvalt ja teha kogutud informatsioonist lähtuvalt enda jaoks vajalikud järeldused.

Käesolev töö on koostatud 2015. aasta sügisel Tallinnas. Ettevõtte Transferwise esindajatega on läbi viidud intervjuud vahemikus november-detsember 2015 ja antud töös esitletud informatsioon peegeldab ettevõtte struktuuri, protsesse ja vahendeid antud ajahetkel.

1.2 Ülesande püstitus

Bakalaureusetöö eesmärgid:

Eesmärk 1: Defineerida DevOps liikumise olemus, protsessid ja vahendid ühtse dokumentatsioonina.

Eesmärk 2: Kaardistada ettevõtte TransferWise arendusmetoodikad võrdluses DevOps printsiipidega.

Eesmärk 3: Analüüsida DevOps meetodite ja vahendite sihipärase juurutamise võimalikku kasu ettevõtte arendusprotsessidele ja ärilistele eesmärkidele.

1.3 Metoodika

Antud töö eesmärkide saavutamiseks on läbi viidud järgnevad tegevused:

Eesmärk 1 saavutamiseks on kirjeldatud autori kogemusele ja erinevatele usaldusväärsetele allikatele tuginedes DevOps liikumise põhiolemust ja populaarsemaid arendusvahendeid.

Eesmärk 2 saavutamiseks on viidud läbi intervjuud ettevõtte esindajatega, eesmärgiga kaardistada nende poolt tarkvaraarenduses kasutatavad põhimõtted, meetodid ja vahendid, pidades silmas DevOps põhimõtteid. Kogutud informatsioon on struktureeritud ja vastandatud DevOps põhiprintsiipidega.

Eesmärk 3 saavutamiseks on ettevõtte Transferwise töökorralduse ja DevOps printsiipide võrdlusel tekkinud informatsiooni põhjal järeldusena leitud täna kasutusel olevate DevOps võtete kasu ja täna mitte kasutusel olevate võtete võimalik kasu arendusprotsessi efektiivsusele.

1.4 Ülevaade tööst

Käesolev töö on jagatud viie sisuosa vahel.

Töö esimeses osas on ülevaatlikult defineeritud ja kirjeldatud DevOps liikumise põhimõtted, ajalooline taust ja sisu.

Töö teises osas on lühidalt tutvustatud töös analüüsitavat ettevõtet Transferwise.

Töö kolmandas osas on kirjeldatud DevOps põhimõtted organisatsioonilisest vaatepunktist lähtuvalt ja võrreldud antud aspekte uuritava ettevõtte tööpõhimõtetega, eesmärgiga leida ühised jooned ja kirjeldada erinevused.

Töö neljandas osas on välja toodud levinumad DevOpsiga seostatavad tehnilised meetodid ja vahendid ning vastandatud need uuritavas ettevõttes kasutusel olevatega.

Töö viiendas osas on võrdluse tulemused koondatud ja välja toodud analüüsi tulemusel leitud järeldused.

2. DevOpsi käsitlus antud töös

DevOps käsitluse selgitamiseks antud töö kontekstis on järgnevalt toodud DevOpsi definitsioon, DevOps ajaloo lühitutvustus ja levinud valearusaamad.

2.1 DevOps definitsioon

DevOps on tarkvaraarendusmetoodikate kogum. DevOpsi käsitletakse tihti ka kui liikumist või kultuuri.

Mõiste DevOps on lühend terminitest *development* ja *operations*. DevOps koondab enda alla hulga tarkvaraarendusmetoodikaid ja - põhimõtteid, mis võivad erinevates käsitlustes ka mõneti varieeruda. Eelkõige tähendab DevOps tarkvaraarendusspetsialistide (*development*) tihedat koostööd teiste IT-spetsialistidega (*operations*), kelle erialaks ei ole otseselt tarkvara arendamine, näiteks süsteemi-, võrgu- või andmebaasiadministraatorid, aga ka kasutajatugi ja tarkvaratestijad. Laiemas käsitluses ei puuduta DevOps koostööd vaid IT arendus- ja haldusosakonna vahel, vaid kaasab ka ettevõtte äripoole.

DevOpsi põhiprintsiipideks on lisaks arendus- ja süsteemihaldustiimide efektiivsele koostööle ka süsteemide integreerimine ja automatiseerimine, eesmärgiga saavutada võimalikult sujuv, kiire ja veakindel tarkvara tarneahel. DevOps on tihedalt seotud pideva integreerimise ja pideva tarne põhimõtetega ja aitab ettevõtetel seeläbi oma arendustegevusega kiiremini reageerida turuolukorra muutustele.

2.2 DevOps liikumise ajalugu

DevOps liikumine sai alguse 2008-ndal aastal, kui Andrew Clay Shafer ja Patrick Debois tõstsid Agile 2008 konverentsil Torontos esile mõiste *agile infrastructure* - agiilne infrastruktuur [4]. Selle teemakäsitlusega juhiti tähelepanu probleemile, et enamus tänapäeva agiilseid tarkvaraarendusmetoodikaid (Scrum, test-driver-development, jm) keskenduvad vaid tarkvaraarendusele, jättes kõrvale infrastruktuuri haldusprotsessid, mis on tihti endiselt jäigad ja aeganõudvad, pidurdades tarkvara väljalaskesagedust. Agiilne lähenemine oli vaja viia ka IT-infrastruktuuriprojektidesse.

Debois't sai peagi üks DevOps liikumise juhtfiguure. Ta korraldas esimese DevOpsDays ürituse, mis pani alguse samanimelisele globaalsele konverentsiseeriale, mis on omakorda tänaseni üheks edasiviivaks jõuks kogu DevOps liikumisele [11].

2011-ndal aastal hakkasid ilmuma esimesed DevOps kogukonna poolt arendatud töövahendid, nagu näiteks virtuaalsete arenduskeskkondade konfiguratsioonivahend Vagrant, mis täiendas juba olemasolevaid süsteemihaldusvahendeid Chefi ja Puppetit [11].

Veel üks tähtis nimi DevOpsi propageerimisel on Mike Loukides (*Vice President of Content Strategy*, O'Reilly Media). Debois kõrval koostas ka tema olulisimaid DevOps teemalisi materjale, näiteks 2013. aastal ilmunud raport "What is DevOps?". Loukides juhtis palju tähelepanu sellele, et DevOps ei ole vaid kindlate tööriistade kasutamine ("me kasutame Chef'i, seega teema DevOpsi"), vaid eelkõige tähendab see arendus- ja süsteemitiimide vahelist üksteisemõistmist [7].

2013.-ndast aastast kuni tänaseni on DevOps põhimõtted levinud plahvatuslikult. Mobiilsete pilvelahenduste levik on suurendanud veelgi vajadust paindlike tarkvaratarneahelate järele. 2014.-ndal aastal läbi viidud CA Technologies uuring "DevOps: The Worst-Kept Secret to Winning in the Application Economy" tõi välja, et 88 protsendil 1425-st IT ja ärivaldkonna täitevjuhist on plaan DevOpsi juurutada lähema viie aasta jooksul [5]. Ettevõtted nagu Amazon, Netflix, Target, Facebook ja paljud teised järgivad juba DevOps metoodikaid [8].

2.3 Levinud valearusaamad

Kuna DevOps mõiste on oma olemuselt lai ja kohati mitmeti mõistetav, siis on tekkinud mitmed levinud valearusaamad DevOpsi sisust.

DevOps ei ole sama mis NoOps

Üks levinud hirme on see, et DevOps tähendab sama mis NoOps [7]. Arvatakse, et tänu tänapäevastele konfiguratsioonihaldusvahenditele muutub infrastruktuur nii automatiseerituks, et kaob vajadus süsteemiadministraatorite ja teiste IT-haldusspetsialistide järele. Tegelikkuses tähendab DevOps eelkõige IT-infrastruktuuri efektiivsemat ja agiilsemat haldus, eesmärgiga püsida tempos ärinõuete kiire muutumisega. DevOps küll eeldab, et süsteemiadministraatorid või programmeerijad kirjutaksid IT-süsteemide haldusprotsesside automatiseerimiseks koodi,

kuid sellele vaatamata jääb alles vajadus kompetentsete spetsialistide järele mõlemas valdkonnas, seda eriti keerulisemate küsimuste lahendamisel. Põhirõhk on koostööl.

DevOps ei tähenda ainult ettevõtte töökultuuri ega ainult õigeid vahendeid

Samuti ei piisa DevOpsi rakendamiseks ainult õigete tööriistade kasutuselevõtust või ainult DevOps "kultuuri" loomisest ettevõttes. Automatiseerimisvahendite nagu Chef või Puppet rakendamine üksi ei tähenda, et ettevõte järgib DevOps põhimõtteid [7]. Suur osa DevOpsist on arendajate ja haldustiimi vahelise suhtluslõhe elimineerimine. Samuti ei piisa ainult sellest kui programmeerijad ja süsteemiadministraatorid omavahel hästi läbi saavad. DevOpsi juurutamiseks peavad olema täidetud nii inimlikud kui ka tehnilised aspektid.

DevOps ei ole ametinimetus

DevOps ei ole lihtsalt ametinimetus. Arendajate nimetamine DevOpsideks või DevOps spetsialisti ametikoha loomine ei ole DevOps. Kuigi teatud juhtudel on DevOpsi juurutamisel spetsiaalse ametikoha loomine õigustatud (uutele printsiipidele üleminekut peab keegi juhtima), siis on kõige alus jällegi koostöö ja kommunikatsioon eri tiimide vahel.

DevOps ei ole KÕIK

DevOps ei lükka kõrvale kõiki varasemaid arendusmeetodeid ja vahendeid, vaid pakub eelkõige lahendusi arendus- ja haldustiimide vahelisele lõhele ning sellest tulenevatele kvaliteediprobleemidele - ta toetab teisi agiilseid meetodeid.

3. Analüüsitav ettevõte TransferWise

Antud töös on analüüsitud Eesti ja kogu Euroopa ühe edukaima idufirma TransferWise arendusprotsesse DevOps printsiipidest lähtuvalt [6].

TransferWise on Eestis loodud ja tänaseks UK-s baseeruv rahvusvahelisi rahaülekandeid pakkuv ettevõte. TransferWise loodi 2011-ndal aastal Kristo Käärmanni ja Taavet Hinrikuse poolt.

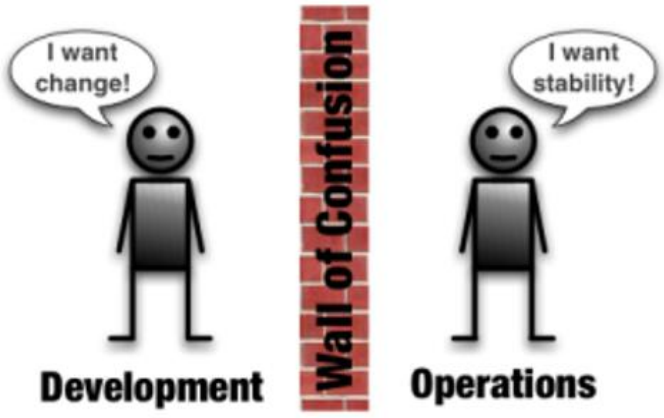
TransferWise pakub alternatiivi muidu kulukatele pankade poolt pakutavatele rahvusvahelistele rahaülekannetele. TransferWise on veebipõhine *peer-to-peer* valuutavahendusteenus, mille efektiivsus tuleneb viisist, kuidas TransferWise raha liigutab. Nimelt ei liigu saatja raha otse vastuvõtjale, vaid vastuvõtja saab raha enda valuutas teiselt sama vääringu saatjalt. Niiviisi välditakse kulukaid rahvusvahelisi ülekandeid ja valuuta konverteerimist [10].
Kuigi TransferWisei peakontor asub Londonis, toimub peamine tarkvaraarendustegevus ettevõtte Tallinna kontoris. Töö aluseks olevate intervjuude läbiviimise hetkel oli TransferWise'is kokku üle 450 töötaja, neist 70 tarkvaraarendajat ja 4 IT-infrastruktuuri eest vastutavat töötajat. Kuna tegu on äärmisel kiiresti kasvava ettevõttega, siis töötajate arv pigem tõuseb.
TransferWise ei kasuta oma arendusprotsessis rangelt paika pandud metoodikaid ja tööriistu, sealhulgas ka mitte DevOpsi. Töövõtted ja vahendid varieeruvad tiimide lõikes ja vastavalt vajadustele. Kasutusel on antud olukorra jaoks optimaalsed protsessid ja vahendid, mis üldjoontes kattuvad DevOps printsiipide poolt esile tõstetutega. Seni on DevOps TransferWise'i jaoks olnud pigem tänapäevase tarkvaraarenduse loomulik osa, mitte sihipärane tegevus.

4. DevOps - organisatsiooniline vaatenurk

Antud töös on DevOps printsiibid jagatud üldistatult kahte valdkonda – organisatsioonilisteks ja tehnilisteks aspektideks. DevOpsi organisatsiooniline käsitlus koondab enda alla arendus- ja haldustiimide vahelise kommunikatsiooni ja rollide jaotusega seotud küsimused, alampeatükkidena *Wall of Confusion*, rollide jaotus, DevOps mustrid ja DevOps kultuuri loomine ettevõttes.

4.1 Wall of Confusion

DevOps toob põhiprobleemina välja arendus- ja süsteemihaldustiimide vahelise barjääri - *"Wall of Confusion"* [14]. Barjäär tuleneb traditsiooniliselt välja kujunenud arusaamadest kummagi tiimi tööülesannetest ja -kultuurist.



Joonis 1. Wall of Confusion – Devs vs Ops [14]

Tarkvaraarendajad on tavapäraselt avatud muutustele. Nende töö põhineb suuresti muutuva ärimaailma vajaduste realiseerimisel. Tänapäevased agiilsed arendusmetoodikad toetavad muutuvaid protsesse ja enamus arendustiime on agiilse arenduse põhimõtted omaks võtnud ja rakendavad neid igapäevaselt.

Klassikaliselt põhineb süsteemiadministraatorite töö stabiilsuse tagamisel. Kuna igasugused muutused ohustavad süsteemi stabiilsust, siis suhtutakse nendesse skeptiliselt. Väidetavalt on 80% IT-süsteemide töö tõrgetest põhjustatud kehvasti planeeritud süsteemimuudatustest [1].

Sellest mõttemaailmade erinevusest tulenevalt võivad ettevõttes meeskondade omavahelise üksteisemõistmise puudumisel tekkida konfliktid ja tehnilised probleemid, mis enamjaolt avalduvad alles tarkvara tarneahela väljalaskefaasis.

Kahe meeskonna vahelist lõhet süvendavad tihti ka erinev juhtimine ja geograafiline asukoht.

Joonis 2. Wall of Confusion [14]

DevOps liikumine keskendub just süsteemihaldustiimi kaasamisele tarkvaraarendusprotsessi varasemates faasides, et vältida viimase hetke probleeme.

TransferWise'is ei ole üldjuhul arenduse- ja haldustiimi vahelist barjääri. Kuna eesmärgid on ärilisest aspektist ühised, siis töötatakse ka ühiselt nende eesmärkide saavutamise nimel. Siiski on töö üleandmise punkt arendusprotsessis alati kriitiline ja veaaldis koht. Seetõttu pannakse erilist rõhku muudatuste juhtimisele ja töö pidevale tagasisidestamisele. Kuna ettevõtte töökultuur on üleüldiselt tolerantne ja sõbralik, ei tegeleta probleemide korral üksteise süüdistamisega, vaid püütakse viga parandada ja võimalusel neid edaspidi ennetada.

4.2 Rollide jaotus

DevOpsi levik toob endaga kaasa mõningaid olulisi muutusi klassikalistes IT-rollides. Nii tarkvaraarendajatelt kui ka süsteemiadministraatoritelt eeldatakse uusi oskusi, mis aitavad kaasa süsteemide efektiivsemale automatiseerimisele ja integreerimisele.

4.2.1 Tööülesannete jaotus

DevOps annab arendusprotsessis osalejatele mitmeid uusi tööülesandeid, mis klassikalises käsitluses selgelt ei kajastu. Uued tööülesanded ei ole rangelt jaotunud arendus- ja haldusmeeskondade vahel – olenevalt kompetentsist, meeskonna struktuurist, olukorrast või muust võib antud küsimusi lahendada ka teine pool.

Tööülesannete jaotus on kirjeldatud järgneva tabelina.

Tabel 1. Tööülesannete jaotus arendus- ja haldustiimide vahel

Klassikalised arendusmeeskonna tööülesanded	**Klassikalised haldusmeeskonna tööülesanded**
Tarkvara analüüs	Serverite, andmebaaside, arvutivõrkude ja erinevate seadmete konfiguratsioon ja hooldus
Tarkvara disain	Valmis tarkvarade paigaldus ja konfiguratsioon
Programmeerimine	Süsteemide monitooring
Testimine	*Helpdesk*
Arendusprotsessi juhtimine ja dokumenteerimine	Jm
Jm	
Täiendavad "DevOps" tööülesanded	

Lisaks programmeerimisele *unit* testide kirjutamine, testimisprotsessi automatiseerimine.	Pideva tarkvara integratsiooni ja tarne strateegiate välja töötamine ja realiseerimine nii *host*- kui ka pilvepõhisel infrastruktuuril.
Tarkvara iseseisev väljalase.	Teenuse hea kättesaadavuse tagamine nii *production* kui ka *preproduction* keskkondades ka väga tihedate tarkvara väljalasete puhul.
Rakenduse pidev monitooring ja võimalike kitsaskohtade kõrvaldamine.	Automatiseeritud dünaamiliste keskkondade planeerimine ja üles seadmine.
Tihe koostöö ja tegevuste tagasisidestamine infrastruktuuritiimiga.	Tihe koostöö ja tegevuste tagasisidestamine arendustiimiga.

4.2.2 DevOps ametinimetusena

Vaatamata muutunud tööülesannetele, ei tohiks DevOps'ist saada eraldiseisev ametnimetus või roll. Eelkõige peaks olema tegu vastava kompetentsi omamise või loomisega olemasolevate rollide raames.

Sellele vaatamata on nimetus "DevOps" töökuulutustes tõusev trend [9]. Järjest enam otsitakse kandidaate ametisse *DevOps Engineer* või *DevOps Automation Engineer*. Seda ilmselt seetõttu, et ülaltoodud oskused, millele rõhub ka DevOps liikumine, on tööandjate seas nõutud ja aitavad eristada "klassikalisi" spetsialiste kaasaegsete tehnoloogiatega kursis olevatest.

Töö koostamise hetkel töötas TransferWise'is 70 tarkvaraarendajat ja 4 IT infrastruktuurihaldusega tegelevat töötajat. Siiski ei ole TransferWise'is ranget rollide erisust. Vastava kompetentsi olemasolul tegelevad huvi või vajaduse korral ka tarkvaraarendajad infrastruktuuri puudutavate küsimustega, süsteemiadministraatorid programmeerimisega. Erinevalt klassikalisest tarkvaraarendustiimist puuduvad TransferWise'is rollidena IT-projektijuhid ja tarkvaratestijad.

Meeskond on jaotunud suhteliselt eraldiseisvateks vertikaalseteks tiimideks, kes viivad oma projektid ellu iseseisvalt, ilma vastava projektijuhi abita. Kõik meeskonna liikmed on kaasatud tehniliste lahenduste realiseerimisse.

Manuaalse tarkvaratestimise vajadus on viidud miinimumini. Loodav kood kaetakse automaattestidega, arendajad testivad oma koodi ise ja vastutavad selle kvaliteedi eest. Enne koodi töösse andmist vaadatakse lahendus üldjuhul üle mõne teise tarkvaraarendaja poolt, et püüda kinni võimalikud vead või anda täiendavat tagasisidet lahenduse parandamiseks. Vigade esinemisel tegeletakse nende likvideerimisega esimesel võimalusel, sealjuures on oluline vea olemasolust teavitada ka ülejäänud tiimi. Läbi kogemuste jagamise õpitakse ka teiste vigadest.

Ka TransferWise otsis analüüsi koostamise perioodil oma meeskonda *DevOps Engineer* ametinimetusega töötajat. Tegelikkuses liitub antud spetsialist olemasoleva infrastruktuuri meeskonnaga ja tema tööülesanded ei erine oluliselt teiste tiimi liikmete tänastest tööülesannetest.

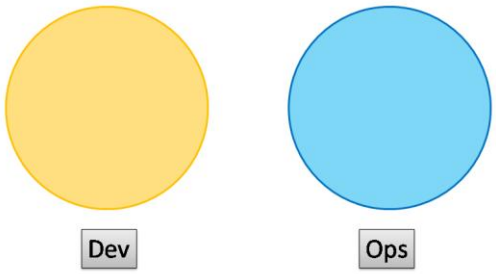
4.3 Mustrid

Kuna arendusmeeskonnad võivad ettevõtete lõikes oluliselt erineda, ei saa defineerida ühte kindlat DevOps meeskonna struktuuri. Järgnevalt on välja toodud mõned levinud DevOps tiimide mustrid ja anti-mustrid [13].

4.3.1 Anti-mustrid

Ettevõtete püüdlused tuua arendus- ja süsteemihaldustiimid üksteisele lähemale ei vii alati soovitud tulemuseni. Järgnevalt on kirjeldatud kolm levinud DevOps mustrit, mis ei ole üldjuhul pikas perspektiivis jätkusuutlikud.

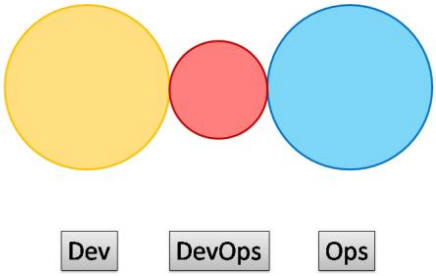
1. **Anti-tüüp A – eraldiseisvad huvigrupid**



Joonis 3. Anti-tüüp A

Tüüpiline näide Dev ja Ops meeskondade vahelisest barjäärist. Ühisosa puudub. Tarkvaraarendajad ei mõista rakenduse infrastruktuuri poolset konteksti, süsteemiadministraatoritel ei ole üldjuhul aega ega motivatsiooni tekkinud konfliktidega tegeleda.

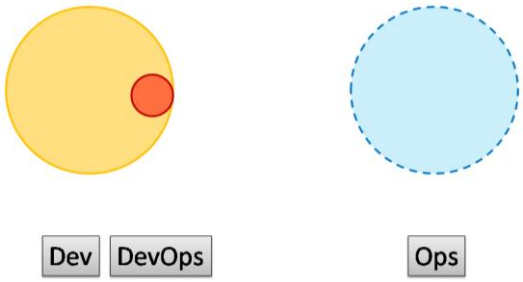
2. **Anti-tüüp B – eraldi DevOps tiim**



Joonis 4. Anti-tüüp B

Otsustatakse ka "DevOpsi tegema hakata". Luuakse eraldiseisev DevOps tiim. Arendus- ja infrastruktuuritiim kaugenevad üksteisest veelgi.

3. **Anti-tüüp C – *Ops*'i ei ole vaja**

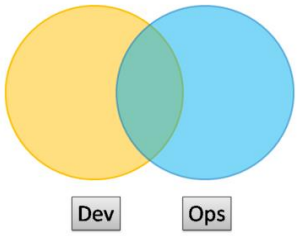


Joonis 5. Anti-tüüp C

Otsustatakse, et IT-haldusosakonda ei ole vaja. Arendajad alahindavad süsteemi keerukust ja loodavad, et tänapäevased pilvelahendused ja konfiguratsioonivahendid võimaldavad neil süsteemi iseseisvalt hallata. Selle tulemusel jääb keerukamate ülesannete või suuremate probleemide korral tihtilugu puudu tehnilisest kompetentsist.

4.3.2 Jätkusuutlikud DevOps mustrid

Tüüp 1. Sujuv koostöö.

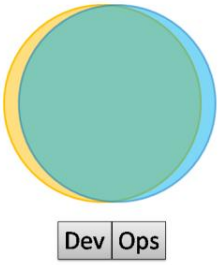


Joonis 6. DevOps tüüp 1 - Sujuv koostöö

Mõlemal poolel on oma spetsiaalsed vastutusalad, kuid ka ühisosa, mille piires tehakse koostööd. On olemas selgelt väljendatud ühine eesmärk ja tehniline kompetents uute meetodite ja tehnoloogiate rakendamisel.

Muster on omane ettevõtetele, kus infrastruktuuri tiimil on lisaks arendustiimi keskkondade toe pakkumisele ka teisi äri toetavaid ülesandeid.

Tüüp 2. Täielikult ühildunud.

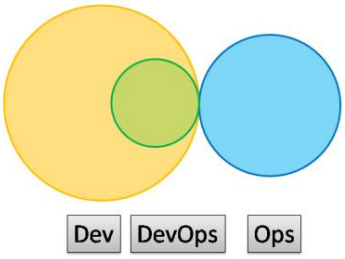


Joonis 7. DevOps tüüp 2 - Täielikult ühildunud

Infrastruktuuri tiim on pea täielikult sulandunud tootearendustiimi. Mudel töötab enamasti ühe veebipõhise toote arendusega tegelevate ettevõtete, näiteks Facebooki puhul. Ei ole üldjuhul rakendatav väljaspool kitsast ühe toote põhist fookust, kuna tiimide huvid ei kattu nii suures ulatuses.

Antud DevOps muster on omane ka TransferWise'ile. Arendus-ja infrastruktuuritiim töötavad ühiselt ettevõtte äriliste eesmärkide realiseerimise nimel. Infrastruktuuritiimi tööülesanded on peamiselt seotud TransferWise'i rakenduse infrastruktuuri toega. Rollid ei ole rangelt jaotatud ega lõhestunud.

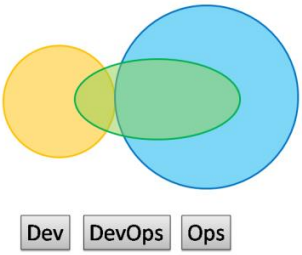
Tüüp 3. IaaS - (Infrastructure as a Service)



Joonis 8. DevOps tüüp 3 – IaaS

Arendustiimi sees moodustatakse ekspertide meeskond, kes rakendab ja haldab ka DevOpsile omaseid infrastruktuuri lahendusi. Üldjuhul kasutusel, kuna oma "traditsiooniline" infrastruktuuritiim ei tule muutustega piisavalt kiiresti kaasa või on fokuseeritud traditsiooniliste lahenduste toele mitme toote- või teenusliiniga ettevõttes.

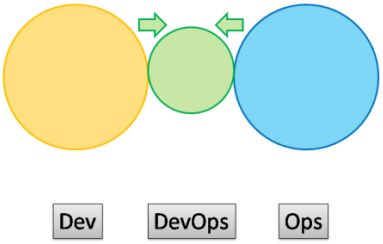
Tüüp 4. DevOps-as-a-Service



Joonis 9. DevOps tüüp 4 - DevOps as a Service

Ettevõttes, kus napib kogemusi, rahalisi vahendeid või tööjõudu DevOps vahendite kasutuselevõtuks, võib olla mõistlik DevOps kompetents sisse osta. Leitakse teenuspakkuja, kes aitab ehitada testkeskkonnad ja automatiseerida infrastruktuuri ja monitooringulahendused ning pakub konsultatsiooni, kuniks tekib oma meeskonnas vajalik kompetents.

Tüüp 5. Ajutine DevOps tiim



Joonis 10. DevOps tüüp 5 - Ajutine DevOps tiim

Ettevõttes luuakse vahetiim, kes "tõlgib" arendus- ja haldustiimi omavahelist suhtlust, aidates seeläbi juurutada DevOps printsiipide kasutust. Selleks, et vältida anti-tüüp B tekkimist, ei anta vahetiimile pikaajalisemat vastutust tarkvara väljalasete ja toote diagnostika osas.

4.4 DevOps kultuuri loomine ettevõttes

DevOps liikumise entusiastid rõhutavad muuhulgas õige töökultuuri loomisele, mis aitaks kaasa tiimidevaheliste barjääride murdmisele ja koostöö innustamisele. Kultuuriline muutus on eelkõige vajalik pika ajalooga ettevõtetes, kelle IT-osakond ei ole muutuva IT maailmaga piisavalt kiiresti kohanenud. Näiteks USA jaemüügikett Target on viimasel ajal innukalt jaganud oma ettevõtte IT-osakonnakonna transformatsiooni traditsioonilisest DevOpsi

järgivaks. Target, nagu paljud teised ettevõtted, oli hädas venivate projektide, aeglase arendusprotsessi ja töötajate madala motiveeritusega, samal ajal kui infosüsteemid muutusid aasta-aastalt järjest raskemini hallatavateks. Lahendust nähti DevOpsis. DevOps kultuuri juurutamiseks leiti tarkvarainseneride seas entusiastid, kes muudatusi juhtisid. Samuti korraldati ettevõtte-siseseid konverentse väliskülalistega, automatiseerimisdemosid, *hackathone* ja palju muud – kõik eesmärgiga juurutada IT-meeskonnas DevOps mõtteviisi ja töövõtteid [2].

TransferWise ei propageeri ettevõttes DevOps kultuuri. Ollakse entusiastlikud kõige suhtes, mis toob arendusprotsessis kasu. Noore ettevõttena ei ole TransferWise'il aastatega juurdunud ebaefektiivseid töömeetodeid, mida oleks vaja läbi kultuurilise muutuse murda. Ollakse kursis tänapäevaste arenduspraktikatega ja rakendatakse neid vastavalt vajadusele.

5. DevOps – tehniline vaatenurk

Lisaks ettevõttes DevOps kultuuri loomisele, käib DevOps liikumisega kaasas mitmeid tehnilisi vahendeid ja töövõtteid, mis aitavad kaasa arendus- ja haldustiimide vahelisele koostööle, protsesside automatiseerimisele, integreerimisele ja süsteemide töökindluse tagamisele.

Klassikalises organisatsioonis tekitab tihti probleeme arendajate ja infrastruktuurihaldurite tööriistade ühisosa puudumine. Kasutatakse eraldiseisvaid töövahendeid- ja keskkondi, mis tihtilugu omavahel ei ühildu. Probleemid tekivad enamasti alles tarkvara juurutusfaasis, mille käigus arendustiim annab arenduse üle infrastruktuuritiimile. Vea tuvastamine on kulukas, kuna üksteise töökeskkondi ei tunta. Sellest tulenevalt tekib tahes-tahtmata olukord, kus iga järgmine tarkvara väljalase on stressirohke ja probleemide tekkimine on pigem reegel kui erand.

DevOps koondab enda alla alltoodud töömeetodid ja -vahendid, mis aitavad võimalikku lõhet vältida.

5.1 Agiilsed metoodikad

Viimase kümne aasta jooksul on toimunud suured muudatused tarkvaraarenduspraktikates. Klassikalisest kosemudelist on liigutud järk-järgult iteratiivsete ja agiilsete metoodikate juurde, mis võimaldavad tarkvara lõppkasutajani viia osade kaupa loogiliste „tükkidena“. Sellele aitab kaasa MVP – *minimum viable product* kontseptsioon, kus tarkvara planeeritakse, arendatakse ja tarnitakse võimaikult väikeste äriliselt kasulike osadena [12]. See omakorda toob ärilise lahenduse kasutajani kiiremini ja sujuvamalt. Arendajad saavad oma tootele kohese tagasiside ja muudatuste ja paranduste sisseviimine toimub kogu arenduse elutsükli vältel. DevOps täiendab agiilset tarkvaraarendust, juhtides tähelepanu IT- haldusosakonna rollile selle realiseerimisel. DevOps tõstab esile mõiste „agiilne infrastruktuur“, mis tähendab, et koos ärivajaduste agiilse realiseerimisega tarkvarakoodis peab toimima ka infrastruktuuri agiilne haldus [4].

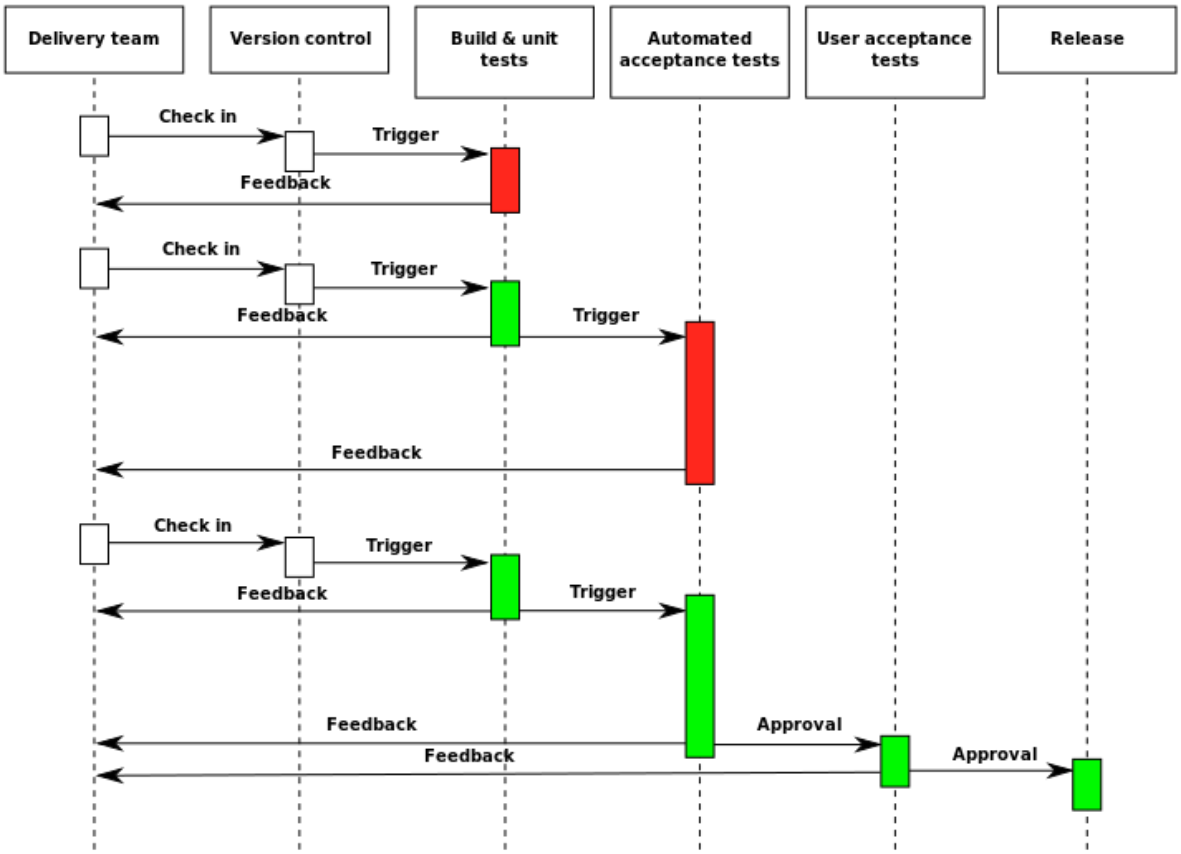
TransferWise arendab tarkvara agiilselt. Tarkvara luuakse pidevalt eri tiimide poolt väikeste „tükkidena“. TransferWise’i arendustegevused toetavad ärilisi eesmärke. Uued lahendused ja ideed juurutatakse võimalikult kiiresti. Kui lahendus ei osutu äriliselt otstarbekaks, minnakse edasi järgmise ideega. Lähtutakse MVP kontseptsioonist. Tarkvara luuakse arhitektuuriliselt suures osas mikroteenustena, mida saab tarnida eraldiseisvate üksustena.

Mikroteenused annavad arendajatele arenduskeele valikul vabamad käed, kuna need suhtlevad üksteisega üle arenduskeelest sõltumatute arendusliideste. Neid saab tihedalt juurutada ka arendajate endi poolt ja need on lihtsalt monitooritavad. Siiski kaasneb mikroteenuste kasutamisega ka lisakeerukus. Seda eelkõige andmete pärimisel analüüsiks, kuna selleks on vajalik andmete denormaliseerimine ja teenuste vaheliste halduskihtide loomine.

5.2 Tarkvara pidev integreerimine ja tarne

Tarkvara pidev muutmine väikeste osade kaupa tekitab vajaduse efektiivsema koodihalduse järgi, et vältida ühilduvusprobleeme. Pideva integreerimise puhul koondatakse kõik eri arendajate poolt tehtud koodimuudatused mitmel korral päevas keskses või jagatud repositooriumis. Selleks on mõistlik kasutada versioonihaldustarkvara nagu SVN või Git. Koodimuudatused kompileeritakse järkudena ja testitakse koheselt automaattestidega. See teeb võimalikuks vigade avastamise ja parandamise arenduse varajases faasis. Paljud DevOps tiimid on lisaks versioonihaldussüsteemidele võtnud kasutusele Jenkins tarkvara, mis võimaldab automatiseerida *build*- ja testimisprotsesse, aidates seeläbi kaasa pidevale integreerimisele.

Tarkvara pidev tarne on samm edasi pidevast integreerimisest. Pidev tarne on üks DevOps tehnilise käsitluse põhiprintsiipidest. Järjest vähem on ettevõtteid, kes tarnivad oma tarkvarauuendusi mitmenädalaste või -kuuliste vahedega. Näiteks tarkvarahiiud Netflix ja Amazon annavad tarkvarauuendusi välja sadu või isegi tuhandeid kordi päevas [8]. Selle realiseerimiseks on vajalik tarkvara tarneprotsessi automatiseerimine, mille tulemusel on tagatud, et kõik tehtud muudatused programmis on igal hetkel *live* süsteemi üle viidavad. Koodimuudatused läbivad enne toodangukeskkonda viimist mitmeid kontrolletappe lähtekoodi kompileerimisest kuni detailsete testprotseduurideni, kogu protsessis on oluline pidev tagasisidestamine. Pideva tarne protsessi kirjeldab tarkvara tarneahel.

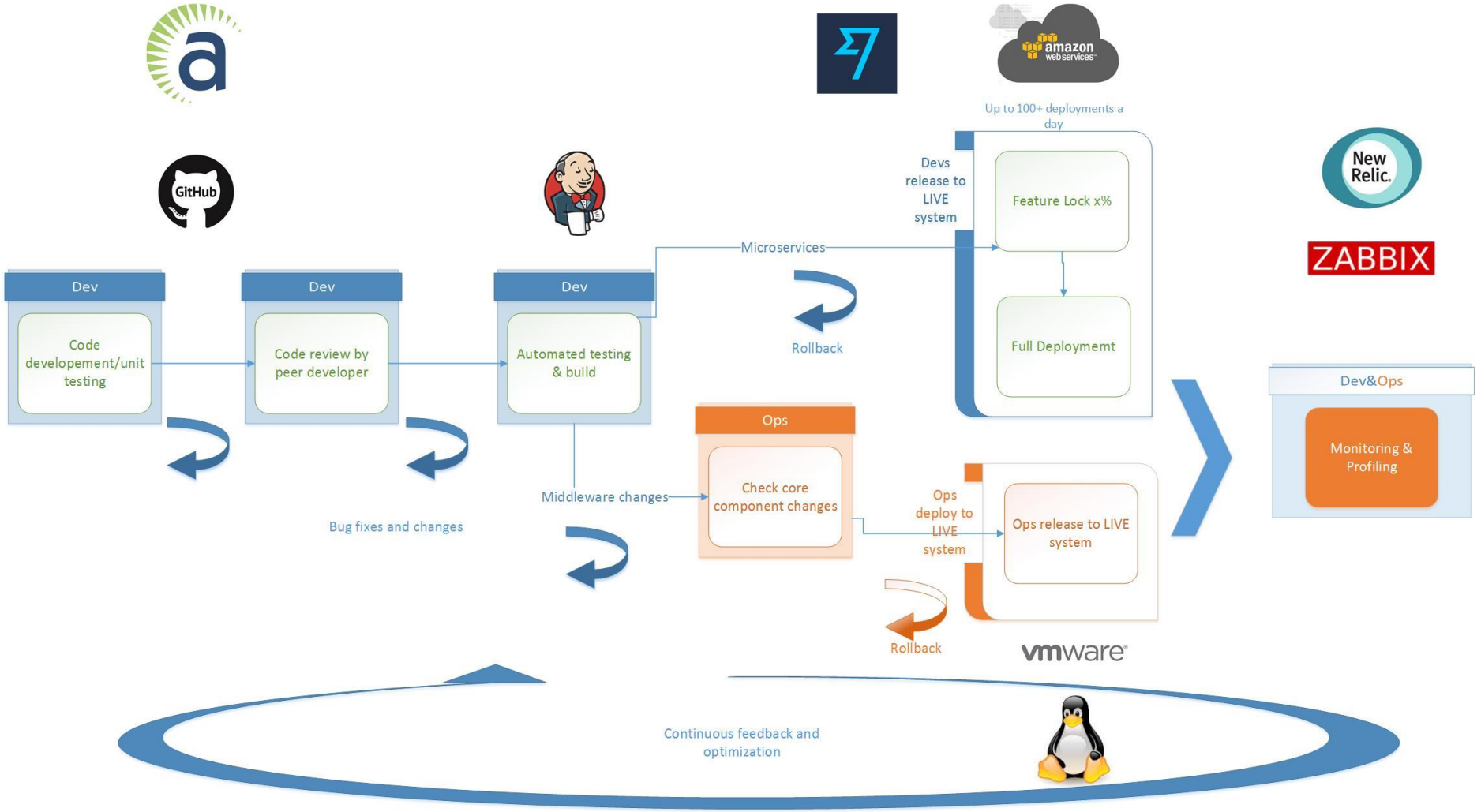


Joonis 11. Pidev tarkvara tarneahel [3]

TransferWise'i programmikood on üldjuhul lihtsalt hallatav ja automaattestidega kaetud, manuaalne testimine ja muu „käsitöö" on viidud miinimumini. Koodi haldamiseks ja testimiseks on kasutusel GitHub ja Jenkins. Protsessid mida on vähegi võimalik ja mõttekas automatiseerida automatiseeritakse. Esmajärjekorras lahendatakse probleemid, kuhu kulub asjatult aega.

Tarneprotsess on suures osas automatiseeritud. Tarkvara väljalasked toimuvad üldjuhul vähemalt korra päevas. Mikroteenuseid viiakse *live* keskkonda vajadusel ka enam kui 100 korda päevas, tarkvara väljalasete sagedus ei ole limiteeritud. Muudatuste tagasivõtmine on üldjuhul lihtsalt teostatav. Siiski on muudatusi, mis vajavad enne *live* süsteemi viimist infrastruktuuritiimi poolset manuaalset kontrolli, eelkõige kehtib see vahevaras tehtavate muudatuste puhul. Need muudatused on vigade puhul ka raskemini tagasi võetavad, kuna seovad omavahel tarkvara rakendus- ja andmekihid.

TransferWise'i tarkvara tarneahel on kirjeldatud alltoodud joonisena. Lisatud on ka näited kasutatavatest töövahenditest ja vastutusalad.



Joonis 12. TransferWise'i tarkvara tarneahel koos vastutusalade ja näidetega töövahenditest

5.3 Tiimitöö vahendid

Pideva tarkvara tarneahela saavutamiseks on oluline tõhus kommunikatsioon ja pidev tagasisidestamine tarkvaraarendusprotsessis osalejate vahel. Samuti muudatuste juhtimine riskide minimeerimiseks. Efektiivse tiimisuhtluse loomisele ja muudatuste ajaloo salvestamisele aitavad kaasa mitmed tänapäevased koostöövahendid, näiteks interaktiivsed suhtlus- ja versioonihaldustarkvarad, koodihoidlad ja IT-spetsiifilised projektihalduskeskkonnad.

Tiimitöövahendid peaksid toetama töötaja produktiivsust. Klassikaline suhtlus emaili ja telefoni teel ei ole üldjuhul enam primaarseks suhtlusvormiks. Tarkvaraarendustiimid eelistavad järjest enam moodsaid koostöökeskkondi. Suhtlus on pigem *chat*i põhine, kasutatakse vahendeid nagu Slack ja HipChat. Tööülesandeid ja projektifaile hallatakse näiteks Jiras, Youtrackis või Assemblas, kuhu on võimalik koondada nii projektikood, eri tüüpi failid kui ka tööülesanded näiteks Scrum või Kanban tahvlitena.

Kommunikatsiooni- ja koostöövahendid ei ole TransferWise'is kuidagi paika pandud, need varieeruvad tiimiti. Kasutusel on lisaks tavapärastele telefonidele, emailidele ja Skype'ile näiteks suhtluskeskkond Slack, arendusmeeskonna *ticketing* süsteemina on kasutusel Assembla.

5.4 Infrastruktuuri haldus

DevOps paneb suurt rõhku paindlikumale infrastruktuuri haldusele. Süsteemihaldustiimi vaatenurgast on kõige suurem muutus tänapäevases tarkvaraarenduses rakenduse üleminek üksikutelt serveritelt kümnetele, sadadele või tuhandetele serveriharudele, seda näiteks AWS-i või Rackspace'i pilves [7]. Sellise muutuse järel ei saa infrastruktuuri hallata vanaviisi mõne käsureaskriptiga, probleeme ei saa käsitleda ainult eraldiseisvate „tulekahjudena". Infrastruktuur peab skaleeruma vastavalt ärivajaduste ja arendusmahtude kasvule. Süsteemihaldus eeldab pidevat optimeerimist ja ettenägelikkust.

Olukorraga aitab kohaneda infrastruktuuri haldamine koodina (*infrastructure as code – IaC*). Töökindel infrastruktuur peab olema taastoodetav ja programmeeritav, nagu ka tarkvarakood [7]. Infrastruktuuri haldamise seisukohast ei ole enam oluline, kas riistavaraline ressurss on füüsiline, tarkvaraline või kombinatsioon mõlemast. Oluline on, et süsteemiadministraatorid mõistaksid automatiseerimise tähtsust skaleeruva ja töökindla infrastruktuuri loomisel. Kui skriptid sisaldavad enamasti staatilisi korratavaid samme, siis IaC-i puhul kasutatakse infrastruktuuri konfigureerimiseks kõrgema taseme programmeerimiskeeli, mis võimaldavad kirjutada mitmekülgsemat koodi süsteemide haldamiseks ja tarneprotsessi automatiseerimiseks. Seda koodi saab hallata sarnaselt programmikoodiga, sealjuures hoida ka infrastruktuuri koodi versioonihaldussüsteemis, kus on kõik muudatused jälgitavad.

Koos DevOpsi levikuga on populariseerinud ja täienenud erinevad konfiguratsioonivahendid [11]. Riistavaralistest erinevustest tulenevatele konfiguratsiooniprobleemidele pakuvad lahendusi virtuaalmasinad ja konteinertehnoloogiad (*containerization)*. Kaks levinumat vahendit infrastruktuuri automatiseerimiseks on Chef ja Puppet. Need aitavad tagada, et kõigi seadmetel töötavad õiged teenused ja on õige tarkvaraline konfiguratsioon. Eelmainitutega ühilduv Vagrant kindlustab, et kõik virtuaalmasinad on algusest peale identselt konfigureeritud, Ansible aitab realiseerida infrastruktuuri koodis. Antud vahenditega saab kirjutada tarkvara oma riistvara haldamiseks, olenemata sellest, kas tarkvara jookseb enda serveris, *hosting*us või pilves.

TransferWise ei ole infrastruktuuri puhul paika pannud kindlat platvormi ja töövahendeid. Infrastruktuuri ja vahendite valimisel lähenetakse vajaduspõhiselt ja valitakse antud olukorraks sobivaim lahendus. Eksperimenteerimise mõttes proovitakse erinevaid tehnoloogiaid, et testida nende sobivust TransferWise'i vajadustega. Kasutusel on nii oma riistvara kui ka pilvehostingus olevad serverid. Majasisene riistvara on eelkõige vajalik turvakaalutlustel kriitilisemate komponentide hoidmiseks, veebilahenduse front-end on üldjuhul Amazoni pilves (AWS) selle hea skaleeruvuse tõttu. Aga kasutatakse ka teisi pilvehostingu pakkujaid, nt Herokut. Virtualiseerimiseks on üldjuhul kasutusel VMWare, aga ka näiteks KVM või Qemu. Kohaliku arenduskeskonna haldamisel lihtsustab tööd konteinertarkvara Docker, kuid *live* keskkondades seda praktiliselt ei kasutata. Operatsioonisüsteemina on kasutusel Linux.

Infrastruktuuri manuaalset konfigureerimist kasutatakse võimalikult vähe. Kui mingi tegevus sisaldab mustrit, mida saab koodis esile kutsuda, siis viiakse see koodi ja hoitakse näiteks

GitHubis. Samas on alles ja ei kao ära manuaalne konfigureerimine, kuna kõiki tegevusi ei saa automatiseerida.

Muidu populaarseid Chefi või Puppetit TransferWise'is tarneprotsessi automatiseerimiseks ei kasutata. Selleks otstarbeks on olemas ise arendatud tööriist, mis on küll väiksema funktsionaalsusega, kuid just TransferWise'i tarkvara tarneprotsessile vastav. TransferWise'i suuruses tugevat arenduskompetentsi omavas ettevõttes on oma vahendite loomine teatud olukordades loomulik ja efektiivsem kui üldtuntud vahendite kasutamine.

5.5 Tarkvara pidev testimine

DevOps paneb suurt rõhku tarkvara pidevale testimisele (*continuous testing*). Varasemad metoodikad, kus testimine järgnes arendusfaasile ja eelnes tarnefaasile ei ole DevOpsi seisukohast jätkusuutlikud. Testimine peaks olema pidev ja maksimaalselt automatiseeritud. Läbi automaattestide saab arendaja oma koodile kohese tagasiside ja vigade parandamine arenduse varajases faasis on oluliselt odavam.

Tarkvara tarnimisel kasutatakse tihti A/B testimist, mille puhul viiakse muudatus *feature lock'ina* esialgu piiratud kasutajagrupini. Kui lahendus ei tööta tehniliselt või äriliselt, võetakse see tagasi (ka muudatuse tagasivõtmine peaks olema automatiseeritud protsess). Kui lahendus osutub edukaks, saab selle jagada suuremale kasutajaskonnale.

TransferWise'is ei ole eraldi testija rolli ega kvaliteedi tagamise meeskonda. Eraldi testijate puudumine kiirendab tarkvara tarneprotsessi, kuna puudub kolmas osapool arendus- ja haldustiimi vahel, kes lahendust õpib, kontrollib ja tagasisidestab. Samuti aitab see tagada, et aredajad võtavad täie vastutse tarkvarakoodi kvaliteedi eest. Vigade esinemissagedus on viidud miinimumini kasutades automaatteste, kus vähegi võimalik. Arendajad kontrollivad oma lahendusi ja vastutavad nende eest ise, toodangus leitud vigu analüüsitakse ja nendest teavitatakse ka teisi. Vigu parandatakse süsteemselt. Valmis kood läbi enamasti ka ülevaatuse, see tähendab, et kood kontrollitakse üle teise tarkvaraarendaja poolt. Mõned tiimid rakendavad ka testimisel põhinevat arendust. Testimiseks kasutatakse muuhulgas Jenkinsit. Jõudlus- ja turvatestid on väga mahukad. Lahenduse testimiseks väiksemal

kasutajabaasil viiakse arendus esialgu *live* keskkonda *feature lock*'ina. Mikroteenuste puhul on vea korral muudatuse tagasi võtmine sama lihtne kui selle ülespanek.

5.6 Monitooring

Üks suurimaid väljakutseid tänapäevastele süsteemihaldustiimidele on probleemide ennetamine hajussüsteemides [7]. Monitooringu tähtsus muutub järjest suuremaks, ja seda mitte ainult infrastruktuuri tasemel. Monitooring peaks olema muuhulgas ka rakenduse osa. Tänapäevased monitooringuvahendid võimaldavad jälgida praktiliselt kõiki süsteemi aspekte. Arendus- ja infrastruktuuritiimide ülesandeks on sellest informatsioonitulvast õigete mõõdikute alusel analüüsiks vajalike andmete kogumine ja nende andmete töötlemine, et jõuda õigel hetkel jälile võimalikele vigadele või kitsaskohtadele. See on taaskord aspekt, mille puhul on ülimalt oluline arendus- ja haldustiimide omavaheline koostöö. Süsteemi stabiilsus on jagatud vastutus, kuna viga võib esineda nii rakenduse kui infrastruktuuri tasemel. Oluline on, et monitooringust tuleva info alusel oleks veakoht võimalikult lihtsalt tuvastatav, et vastutav osapool saaks kiirelt vea kõrvaldatud.

TransferWise'i puhul toimub süsteemi monitoorimine nii rakenduse kui ka infrastruktuuri tasemel. Infrastruktuuri puhul kasutatakse peamiselt Zabbix tarkvara, tarkvaraarenduse poolel on kasutusel NewRelic ja teised tarkvarad.

Anomaaliad on tuvastatavad mitmel kihil, monitooringuandmete kättesaadavus on hea. Süsteemi vastu on mitutuhat parameetrilist kontrolli, sealhulgas kontrollitakse näiteks piisava kõvakettaruumi olemasolu ja andmebaasitabelitesse vajalike kirjete tekkimist, aga ka protsessitulemuste vastavust äriparameetritele. Monitooringuparameetrite lisamine on sarnane tarkvaratestide kirjutamisele, ehk parameetrid täienevad pidevalt süsteemi kasvades. Kõiki muudatusi ei pea ka käsitsi lisama, näiteks Zabbix on piisavalt intelligentne vahend, et võtab lisandunud serveri ise monitooringusse.

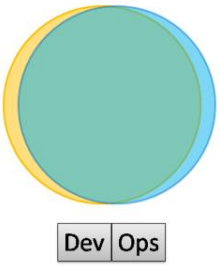
Üldjuhul administreerivad monitooringusüsteeme tarkvaraarendajad, lahendaja otsimisele ei kulu üleliigset aega, kuna on seadistatud automaatsed teavitused vastavalt veatüübile. See tähendab, et veateade läheb emaili või tekstisõnumina kellele vaja ja seda nii *live* kui *prelive* süsteemides.

Tulemuseks on kvaliteetne infosüsteem, mis teavitab probleemidest.

6. TransferWise’i töömeetodite ja –vahendite vastavus DevOps printsiipidele

Vastavalt analüüsi käigus käsitletud alampunktidele, on järgnevalt esitatud TransferWise’is kasutatavate arendusmeetodite ja -vahendite võrdlus tabelina.

Tabel 2. TransferWise'i töömeetodite ja -vahendite vastavus DevOps printsiipidele

Printsiip	**Kas on TransferWise’is rakendatud?**	**Kommentaar**
Arendus- ja infrastruktuuritiimi vahelise sotsiaalse barjääri puudumine	Jah	Eesmärkide nimel töötatakse ühise meeskonnana.
Range rollide jaotuse puudumine organisatsioonis	Jah	Tarkvaraarendajad ja infrastruktuuritiim teevad omavahel tihedat koostööd, vajadusel tööülesanded ristuvad.
Jätkusuutlik DevOps tiimi muster	Jah	 Tiimid on pea täielikult ühildunud.
DevOps kultuuri sihipärane loomine ettevõttes	Osaliselt	Arendusprotsessis rakendatud põhimõtted kattuvad enamjaolt DevOpsiga, kuid sihipärast printsiipide järgimist ega „DevOps“ mõiste pidevat kasutamist ei ole.

Agiilne/iteratiivne tarkvaraarendus	Jah	Tarkvara luuakse peamiselt mikroteenustena ja tarnitakse pidevalt.
Pidev integreerimine	Jah	Koodimuudatused ühendatakse pidevalt koodihoidlas.
Pidev tarne	Osaliselt	Mikroteenuste puhul toimib pidev tarneahel, keerulisemate vahevara puudutavate muudatuste puhul on vajalik täiendav kontroll, protsess ei ole täielikult automatiseeritav.
Efektiivsed koostöövahendid	Jah	Suhtlus- ja koostöövahendid vastavad kasutajate vajadustele, kuid varieeruvad tiimiti.
Infrastruktuur koodina	Jah	Võimalusel hallatakse infrastruktuurimuudatusi koodis ja hoitakse muudatuste ajalugu koodihoidlas.
Automaattestimine	Jah	Tarkvarakood on maksimaalselt automaattestidega kaetud, manuaalse testimise vajadus on viidud miinimumini.
Laiaulatuslik monitooring	Jah	Monitooring on nii infrastruktuuri kui rakenduse tasemel, kaetus on hea.

Kuigi meetodid ja vahendid vastavad DevOpsile, on protsesside parendamine pidev ja süsteemne töö ja kõigi aspektide puhul on veel arenguruumi.

Analüüsi tulemustest võib teha järgnevad järeldused:

- TransferWise'i arendusmeeskonna näitel võib öelda, et DevOps printsiipide range ja sihipärane järgimine ei ole tingimata vajalik, vaid on tänapäevase tarkvaraarenduse loomulik osa, mis tuleneb vajadusest püsida kiirelt muutuvas äri- ja tarkvaraarendusmaailmas konkurentsivõimelisena.

- DevOps metoodikate kasutamise suurimad kasutegurid on kiire tarkvaratarneprotsess, (TransferWise'i puhul vajadusel 100 või enam väljalaset päevas), süsteemide suurenenud töökindlus (automaattestimisest kinni peetavad vead ja monitooringuandmete hea kättesaadavus) ja inimressursi kokkuhoid (näiteks vähenenud vajadus manuaalse tarkvaratestimise järele).

7. Kokkuvõte

Käesoleva bakalaureusetöö põhieesmärgiks oli DevOps printsiipide koondamine, kirjeldamine ja analüüs ning võrdlemine ettevõtte TransferWise arendusprotsessiga. Töö tulemusena pidi selguma TransferWise'i arendusmeeskonna töövõtete ja -vahendite vastavus DevOpsi poolt kirjeldatutele ning kasutusel olevate DevOps võtete kasu arendusprotsessile, samuti täiendavate DevOps printsiipide jälgimise võimalik kasu.

Töö tulemusena said kaardistatud DevOps liikumise aluseks olevad organisatsioonilised ja tehnilised printsiibid ning vahendid. Kõigi alampunktide juures toodi võrdlusena välja antud põhimõtte järgimine või arendusmeetodi kasutamine TransferWise'is. Tööprotsessi käigus sai kirjeldatud ka TransferWise'i tarkvara tarneahel koos vastutusalade jaotuse ja näidetega töövahenditest, mida võib vaadelda kui DevOps põhimõtete rakendamise tulemust.

Töö tulemusena võib esile tõsta järgnevad järeldused:

- TransferWise'i arendusmeeskonna näitel võib öelda, et DevOps printsiipide range ja sihipärane järgimine ei ole tingimata vajalik, vaid on tänapäevase tarkvaraarenduse loomulik osa, mis tuleneb vajadusest püsida kiirelt muutuvas äri- ja tarkvaraarendusmaailmas konkurentsivõimelisena.

- DevOps metoodikate kasutamise suurimad kasutegurid on kiire tarkvaratarneprotsess, (TransferWise'i puhul vajadusel 100 või enam väljalaset päevas), süsteemide suurenenud töökindlus (automaattestimisest kinni peetavad vead ja monitooringuandmete hea kättesaadavus) ja inimressursi kokkuhoid (näiteks vähenenud vajadus manuaalse tarkvaratestimise järele).

Võimalike edasiarendustena võiks täiendavalt analüüsida ja võrrelda DevOpsiga seostavaid tehnilisi vahendeid ning nende kasu tarkvara tarneprotsessile. Samuti võiks detailsemalt analüüsida TransferWise'i arendusprotsessi kitsamaid aspekte, mis analüüsi käigus esile tõusid, kuid ei leidnud antud töö skoobis põhjalikumat kajastamist.

Antud töö koostamise käigus saavutati järgnevad eesmärgid:

1. Defineeriti DevOps liikumise olemus, protsessid ja vahendid ühtse dokumentatsioonina, tuginedes autori varasematele kogemustele ja erinevatele usaldusväärsetele kirjandusallikatele.

2. Kaardistati ettevõtte TransferWise arendusmetoodikad võrdluses DevOps printsiipidega. Meetodite kirjeldamiseks intervjueeriti korduvalt ettevõtte TransferWise esindajaid eelnevalt koostatud küsimustike, jooniste ja muude materjalide alusel.
3. Analüüsiti DevOps meetodite ja vahendite sihipärase juurutamise võimalikku kasu ettevõtte arendusprotsessidele ja ärilistele eesmärkidele. DevOpsi järgimise kasulikkust kirjeldab koostatud tarkvara tarneahel, kust avaldub arendusprotsessi kiirus ja äriliste lahenduste kiire tarnimine.

Vaatamata püstitatud eesmärkide saavutamisele, parandaks analüüsi detailsust ilmselt TransferWise'i arendusprotsessi jälgimine seestpoolt, kuna intervjuude alusel kõigi nüansside katmine oli keeruline.

Summary

The aim of this thesis was to collect, describe and analyze the DevOps principles and to compare them to the development processes used in TransferWise. As a result it was expected to find the common share and/or differences between the two and the benefits of the principles used and/or the possible benefits of the methods and tools currently not in use.

The thesis mapped the organizational and technical principles and tools suggested by DevOps with the methods and tools used in TransferWise. Within the process the software release pipeline was described with dev and ops responsibilities and examples of tools in use. The release pipeline serves as a description of a result of implementing DevOps.

The thesis has the following outcomes:

- Based on the example of TransferWise, it can be said that the strict and purposeful following of the DevOps principles is not a necessity, but it can be a natural part of modern software development driven by the need to stay in competition in the fast-changing business and development world.

- The greatest benefits of using DevOps methodologies are the agile software release process (as in the example of TransferWise up to a 100 or more software releases per day), increased system stability (thanks to faults caught in automated testing and good availability of monitoring data) and the savings in human resources (the reduced need for manual software testing for example).

As a further development it might be considered to analyze and compare the technical tools related to DevOps in greater detail. It might also be beneficial to further study the narrower aspects of TransferWises development process, which arouse within the process but did not find enough coverage in the scope of this thesis.

The following goals were achieved with this thesis:

1. The thesis defined the essence, processes and tools of DevOps in a single documentation, based of the previous experience of the author and supported by various trustworthy literary sources.

2. The thesis mapped the comparison of DevOps principles with the development methods of TransferWise. In order to describe the methods, various interviews were conducted with TransferWise representatives. The interviews were based on previously composed questionnaires, drafts and other materials.

3. The thesis analyzed the possible benefits of purposeful implementation of DevOps principles and tools to the development processes and business objectives. The draft of TransferWises release pipeline can be viewed as a description of successful implementation of DevOps as it shows the speed of the delivery process.

Despite all the goals of this thesis being achieved, the level of detail of this analysis could probably be improved through watching TransferWises development process from within, as it was difficult to cover all aspects through interviews.

Kasutatud kirjandus

[1] Behr, K., Kim, G., Spafford, G., The Visible Ops Handbook: Implementing ITIL in 4 Practical and Auditable Steps, Oregon: IT process Institute, 2005

[2] Build a DevOps Culture & Infrastructure for Success. [WWW] https://www.versionone.com/webcasts/agilelive-devops-success-webinar-part-1/. (28.10.2015)

[3] Continuous Delivery. [WWW] http://continuousdelivery.com/2010/02/continuous-delivery/#3. (16.12.2015)

[4] Debois, P. Agile Infrastructure & Operations. [WWW] http://www.jedi.be/presentations/agile-infrastructure-agile-2008.pdf. (16.10.2015)

[5] DevOps: The Worst-Kept Secret to Winning in the Application Economy, CA Technologies, 2014, 6.

[6] European Unicorns: Do They Have Legs?, GP Bullhound, 2015, 3.

[7] Loukides, M. What is DevOps?, CA: O'Reilly Media, 2012

[8] Null, C. 10 companies killing it at DevOps. [WWW] http://techbeacon.com/10-companies-killing-it-devops (20.12.2015)

[9] Ouimet, M. The DevOps Hiring Boom: The Numbers Behind the Numbers. [WWW] http://rewrite.ca.com/us/articles/devops/the-devops-hiring-boom-the-numbers-behind-the-numbers.html (20.12.2015)

[10] Price, R. How TransferWise Works. [WWW] http://uk.businessinsider.com/how-transferwise-works-2015-1 (10.11.2015).

[11] Rapaport, R. A Short History of DevOps. [WWW] http://rewrite.ca.com/us/articles/devops/a-short-history-of-devops.html. (16.10.2015)

[12] Ries, E. Minimum Viable Product: a guide. [WWW] http://www.startuplessonslearned.com/2009/08/minimum-viable-product-guide.html (14.12.2015)

[13] Skelton, M. What Team Structure is Right for DevOps to Flourish? [WWW] http://blog.matthewskelton.net/2013/10/22/what-team-structure-is-right-for-devops-to-flourish/. (05.10.2015)

[14]What is DevOps?, [WWW] http://dev2ops.org/2010/02/what-is-devops/. (05.10.2015)