TARTU ÜLIKOOL
Arvutiteaduse instituut
Informaatika õppekava

Siim-Morten Ojasalu

DevOps ja selle rakendatavuse uuring Põhja-Eesti Regionaalhaiglas

Bakalaureusetöö (9 EAP)

Juhendaja: Ph.D Pelle Jakovits

Tartu 2021

DevOps ja selle rakendatavuse uuring Põhja-Eesti Regionaalhaiglas

Lühikokkuvõte:

DevOps on veel päris uus võrreldes teiste tarkvaraarendusmeetoditega. Selle loomisel püüti vältida varasemate metoodikate vigu ning sellepärast on DevOps arenenud üheks populaarseimaks ja tõhusaimaks metoodikaks. Seda kasutavad mitmed suured tehnoloogiaettevõtted üle kogu maailma. Töö eesmärgiks on uurida DevOps'i teooreetilist poolt – kirjeldada selle olemust ja kasutust ettevõttes ning seejärel luua võimalik lahendus vastava metoodika integreerimiseks Põhja-Eesti Regionaalhaigla IT-teenistusse. Selleks koostab autor IT-teenistuse praeguse olukorra kohta analüüsi, toob välja hetkel esinevad probleemid ja annab hinnangu, kuidas võiks DevOps tulla kasuks sealses keskkonnas. Lõpetuseks koostab autor plaani, mille abil on samm-sammult võimalik DevOps'i IT-teenistusse integreerida ja arendusmetoodikat muuta.

Võtmesõnad:

DevOps, pidev integreerimine, pidev tarne, testimine, automatiseerimine, tarkvaraarenduse metoodika

CERCS: P170 Arvutiteadus, arvutusmeetodid, süsteemid, juhtimine

DevOps and its applicability study in North Estonia Medical Centre

Abstract:

DevOps is still quite new compared to other software development methods. It was created in order to avoid the past mistakes of other procedures, which is why DevOps has evolved into one of the most popular and effective methodologies. It is used by many large technology companies around the world. The aim of this thesis is to study the theoretical side of DevOps - describe its nature and the use in a company and then create a possible solution for integrating the respective methodology into the IT service of the North Estonia Medical Centre. Therefore, the author prepares an analysis of the current situation of the IT service, highlights the current problems and assesses how DevOps could be useful in the environment. Finally, the author draws up a plan for step-by-step integration of DevOps into the IT service.

Keywords:

DevOps, continuous integration, continuous delivery, testing, automation, software engineering methodology

CERCS: P170 Computer science, numerical analysis, systems, control

Sisukord

Sissejuhatus

Põhja-Eesti Regionaalhaigla on Eesti suurim aktiivravi asutus. Taolise suurusega kaasnevad väga tihedasti külastatavad infosüsteemid, mis on pidevalt väga suure koormuse all. Neid kasutavad nii kliendid, haigla enda töötajad kui ka haiglas kasutusel olevad arvutid ja masinad, mis peavad informatsiooni edastama ja vastu võtma. Taolised süsteemid vajavad katkematult nii hooldust, uuendusi kui ka parandusi. Seetõttu on vajalik võimalikult efektiivne töö IT-teenistuses, kelle ülesandeks ongi vastavate infosüsteemide ülalpidamine. Haigla infosüsteemi rike võib põhjustada väga kriitilisi tagajärgi ja sellepärast peab IT-teenistus oma tööd tegema täpselt, tõhusalt ja ilma ressursside raiskamiseta.

DevOps (liitsõna, mis tuleneb ingliskeelsetest sõnadest *development* ehk arendus ja *operations* ehk käitus) on oma olemuselt lähenemine tarkvara arendusele ja juurutamisele, mis keskendub võimalikult kiirele uute toodete või teenuste arendamisele, mida on lisaks võimalik pärast tarnet lihtsamini hooldada. Tänapäeval kasutavad suurtest firmadest sellist lähenemist näiteks Amazon, Netflix ja Hertz [14]. Seega on DevOps leidnud rakendust suurtes firmades, mis tähendab, et õige kasutuse juures võib see ettevõttele tulusaks osutuda.

Bakalaureusetöö eesmärk on analüüsida DevOps'i eeliseid ja kasutusvõimalusi Põhja-Eesti Regionaalhaigla (edaspidi PERH) IT-teenistuse näitel. PERH soovib muuta oma tööd ja arendust efektiivsemaks ja sellisel kujul uurimistööd ega analüüsi veel ei ole tehtud. Praeguse lähenemise juures on nõrgaks küljeks vananenud töömetoodika, mille tõttu esineb süsteemides palju tõrkeid ja ressursside raiskamist. Eelnevatele probleemidele oleks DevOps'i rakendamisega võimalik leida lahendus. Sellepärast on koostöös PERH-iga valitud just DevOps vaadeldavaks meetodiks, mille abil oleks sobivuse korral võimalik muuta IT-teenistuse tööd tõhusamaks. Efektiivsem töö PERH-i IT-teenistuses aitaks kogu haigla infosüsteeme paremini töös hoida. Selle töö eesmärk uurida, kas ja millisel viisil rakendada DevOps'i PERH-is, et resultaadiks oleks kiirem ja kvaliteetsem toodete või teenuste arendus ning tarne. Kuna Põhja-Eesti Regionaalhaigla on Eesti üks suurimaid raviasutusi, siis on nende IT-teenistuse abistamine, nõustamine ja tõhusamaks muutmine väga oluline probleem, millele autor otsib lahendust.

Töö läbiviimiseks tutvustab lõputöö autor esialgu DevOps'i teoreetilist poolt, seletades lahti selle olemuse, kasutuseelised ning tuues välja olulised tavad selle rakendamise puhul. Seejärel viib autor läbi küsitlused PERH-i IT-teenistuses, analüüsib sealset töö struktuuri ja teeb võrdluse seal kasutuses oleva arendusmeetodi ja DevOps'i vahel. Töö tulemuseks on

autoripoolne analüüs, mis kasu DevOps'i kasutuselevõtt tooks asutusele ning soovitused, kuidas DevOps'i metoodikat asutuses rakendada.

1. DevOps

Tarkvara arendamiseks ja tarneks on olemas mitmeid erinevaid metoodikaid. Näiteks kosk-arendusprotsess, pidevintegratsioon, iteratiivne arendusprotsess ja nende hulgas ka DevOps. Lisaks nendele on veel palju teisi ja neil kõigil on omad eelised ja puudujäägid. Sellepärast on ka iga metoodika jaoks õige rakenduskoht. Selleks, et olla firmana edukas, peab leidma viisi, kuidas olla võimalikult efektiivne ja seda on DevOps'i abiga saavutanud näiteks Amazon, NASA või Netflix [14].

DevOps'i defineeritakse tihtipeale väheste erinevustega. Jessica Díaz jt [4] on defineerinud seda kui põhimõtete kogumit, mille eesmärgiks on parandada ja edendada tarkvaraarendajate ja süsteemiadministraatorite omavahelist koostööd. Samas ei ole arendajad ja administraatorid ainukesed, keda see puudutab. Teistsuguse määratluse järgi võib DevOps'i kirjeldada kui meetodit, mille abil edendada ühistööd kõikide erinevate sidusgruppide vahel, kes vastava toote või teenuse arendamisprotsessis osalevad, eesmärgiga saavutada võimalikult kiire arendus ja tarne. Lisaks on DevOps justkui uus viis määratlemaks edu – seda ei pruugi alati määrata ainult lõpliku toote ülesehitus või kvaliteet, vaid ka kiirus ning kes sarnase tootega esimesena turule tuleb. Seetõttu üritavad firmad keskenduda kiirele tootearendusele ja seeläbi vähendada aega, mis kulub toote turule jõudmiseks. Firmad, mis suudavad tooted kiiremini lõppkasutajani tuua, on tihtipeale edukamad ning suudavad paremini konkureerida samas valdkonnas tegutsevate teiste firmadega.

Mali Senapathi jt [5] määratlevad DevOps'i kui silda toote arenduse ja juurutamise vahel, mida on võimalik saavutada avatud suhtluse, usalduse, austuse ning vastutuse ning ajendite sidumisel. Selle põhilisteks eesmärkideks on saavutada järjepidev ja kiire tootearendus ning ka juurutus, kuid lisaks tagada ka teenused, mis aitaksid toetada välearenduse elutsüklit. Selle taustal on arengusuunaks saanud toodete juurutamine läbi interneti, et jõuda otse kliendini. See omakorda aitab üha kiiremini ja tihemini uute toodete ja uuendustega turule jõuda.

Sanjeev Sharma [3] kirjeldab DevOps'i kui tarkvaraarendusmeetodit, mis tõstab esile tarkvaraarendajate ja süsteemiadministraatorite omavahelise suhtluse, koostöö ning integratsiooni tähtsuse. See on lahenduseks nende vastastikusele sõltuvusele ja aitab firmal või organisatsioonil kiiresti toota tarkvara või teenuseid.

Lühidalt ajaloost

Steve Mezak [10] toob DevOps'i ajaloo kirjelduses välja aastad 2007-2008, kus belglasest konsultant Patrick Debois sai valitsuselt ülesandeks aidata andmekeskuse üleviimise valmisolekut testida. Ta sai kohustuseks jälgida arenduse ja käituse omavahelisi suhteid ning koostööd. Tulemusena tuli välja suur lõhe nende kahe sidusgrupi vahel ja Patrickul tekkis tahe seda parandada. Antud soov ei saavutanud algselt väga edu, kuid aastal 2009 tegid John Allspaw ja Paul Hammond, mõlemad Flickr'i töötajad, ühel konverentsil ettekande, mängides rollides arenduse ja käituse esindajaid ja süüdistades üksteist süsteemi vigades. See ettekanne on tänaseks saanud kuulsaks, kandes nime „10+ Deploys per Day: Dev and Ops Cooperation at Flickr". Selle tulemusena mõisteti, et tulevikus peavad mõlemad osapooled olema palju läbipaistvamad, sujuvamad ja täielikult integreeritud. Ettekannet nägi ka Patrick Debois, kes sai sellest inspiratsiooni ja pani aluse konverentsile „Devopsdays" Belgias. Nii jõudiski DevOps ametliku terminina kasutusse ja hakkas üha enam kuulsust ja kasutust leidma.

Kasutuseelised

Põhja-Eesti Regionaalhaigla puhul on tegemist Eesti suurima aktiivravi asutusega, mis pakub raviteenuseid peaaegu kõigil arstlikel erialadel. Seetõttu on sealsete töötajate koormus tihti väga suur ja seda ka IT-teenistuses. Pidevalt ja üha kiiremini arenevas informaatika valdkonnas on ka neil vaja edasi liikuda ja uute toodetega turule tulla, kuid võimalikult efektiivse töö saavutamiseks on vajalik sobiv metoodika. Üheks selliseks võimaluseks võiks olla DevOps.

Sanjeev Sharma jt [1] toovad välja DevOps'i rõhu agiilsele metoodikale, mida rakendatakse igale sidusgrupile tarkvaraarenduses. Selle abiga on võimalik kasvatada toote või teenuse arenduse kiirust nii esimeseks tarneks kui ka edasiarendusteks nii klientide tagasisidel kui ka vigade ilmnemisel. Just vastav kiirus ja reaktsioon tagasisidele muudavad DevOps'i väga vajalikuks palju suuremas plaanis kui vaid arenduses – selle abil on võimalik parandada ja edendada nii klientide rahulolu kui ka suurendada asutuse innovaatilisi võimeid ning tagada parem ajakasutus.

Ideaalis võiksid Gene Kimi jt [2] väitel arendajad DevOps'i abiga saada pidevat ja kiiret tagasisidet enda tehtud töö kohta, mis edaspidi aitab neil kiiresti seda hooldada ning ka tarnida vastavasse keskkonda. Seda on võimalik saavutada automaattestide, väiksemate koodijuppide ülevaatuse ja ka proovitestimisega. Sellisel juhul saavutavad arendajad kindluse, et loodud toode või teenus on töökõlblik ja töötab nii, kuidas on ette nähtud ning juhul, kui esinevad probleemid, siis need on kiirelt parandatavad.

Sanjeev Sharma [3] näitel aitab DevOps'iga kaasnev agiilne arendus vähendada traditsiooniliste tarkvaraarenduste lõppfaasi probleeme, kus erinevate tarkvara osade kallal töötanud arendajate või tiimide lõplik ühiseks tooteks integreerimine on kulukas ja ettearvamatu. Sellisel puhul leiti vigu ja kokkusobimatuid osi alles arenduse lõppfaasis, mis tõstis riski toote ümbertöötamise vajaduseks. DevOps'i pidevintegratsiooniga lõimivad arendajad oma tööd ülejäänud tiimiga palju tihemini, et pidevalt testida valminud tarkvara osa ja vigade olemasolul neid koheselt avastada. See aitab vältida kulukat ümbertöötlemist toote või teenuse lõppfaasis.

Uuringud on näidanud, et DevOps'i rakendavate firmade infotehnoloogiline töö on tihtipeale efektiivsem. Nicole Forsgren Velasquez jt [6] toovad oma aastases DevOps'i kokkuvõttes välja tähelepaneku, et mida kauem on organisatsioonid kasutanud DevOps'i kui tarkvaraarendusmeetodit, seda tõhusamaks muutub IT-teenistuse tehtav töö. Ajas paranevad nii äritulemused kui ka üleüldine infotehnoloogiline sooritusvõime. Lisaks tuuakse kokkuvõttes välja tähtsad aspektid IT-teenistuse tootluse võrdluseks. Nendeks on toote juurutuse sagedus, aeg, mis kulub muudatuste tegemiseks ning keskmine aeg, mis kulub taastumiseks vastavatest muudatustest. Need kolm tegurit määravad läbilaskevõime ja stabiilsuse. Infotehnoloogilise jõudluse parandamiseks peavad firmad keskenduma vastavalt läbilaskevõime ja stabiilsuse edasiarendusele ja kasvatamisele ning seda on võimalik saavutada DevOps'i abiga.

Puppet Labs'i korraldatud 2015. aasta uuringust [7] selgub, et suure jõudlusega ja hästi toimivad IT-organisatsioonid, kes on võtnud kasutusele DevOps'i tavad, juurutavad oma arendusi 30 korda tihemini ja seda 200 korda kiiremini. Lisaks taastuvad sellised organisatsioonid riketest 168 korda kiiremini, tehes neid üleüldiselt ka 60% vähem. Sellega lükatakse ümber müüt, et infotehnoloogias peab valima kas kiiruse või töökindluse vahel. Kuna rikked ja vead on vältimatud, siis olulisele kohale kerkibki kiirus, millega vastavad tõrked eemaldatakse ja kui kiiresti neist taastutakse. See omakorda võib saada määravaks faktoriks, kas organisatsioon on turul edukas või võitleb ellujäämise nimel.

Eelnevaid punkte kokku võttes võib öelda, et DevOps'i rakendades on firmal või asutusel võimalik paremini ära kasutada kõikide oma tootearenduse sidusgruppide aega, vältides suuri vigasid arenduse lõppfaasis ja vajadust toodet ümber töötada ning seega tõsta tehtud töö väärtust aja suhtes. Lisaks, tänu pidevale testimisele ja tagasisidele, on võimalik luua lähedasem ja lojaalsem side oma klientidega, kelle arvamust võetakse kuulda ja mille abil on võimalik toodet kiirelt paremaks muuta. Samas tuleb sellise keskkonna saavutamiseks üles seada vastavad automaattestid ja luua pidev, aus ning avatud suhtlus kõikide osapoolte vahel, kes

vastava teenuse või toote juures töötavad. Seepärast peaks ettevõte uurima, kas vastav panus oleks neile ka tulus.

2. DevOps’i metoodika rakendamine ettevõttes

Iga suure otsuse puhul ettevõttes on vaja kaaluda läbi erinevad võimalused ja analüüsida nende võimalikke tulemusi. Seda peab tegema ka tarkvaraarenduses ja valminud toote juurutamises. Võrreldes traditsiooniliste arendusmeetoditega võib DevOps algselt tunduda kallis ja liialt suur muutus. Samas on näiteks Puppet Labs’i poolt läbi viidud uuringust selgunud, et DevOps’i edukalt rakendavad organisatsioonid võivad olla tootearenduses palju kiiremad, samal ajal säilitades parema töökindluse [7]. Samas ei pruugi selline metoodika sobida iga organisatsiooni puhul ja seetõttu tuleb veenduda, kas DevOps’i rakendamine oleks ettevõtte jaoks tulus ja kasulik. Selleks võiks läbi viia uuringu.

2.1 Valearusaamad

Iga uue või tundmatu muudatuse puhul on tähtis teada selle tagamaid ja õppida tundma selle mõju tulevikus. Ka DevOps’i puhul on oluline olla tuttav selle printsiipidega ning sellega kaasnevalt ka paljude valearusaamadega, mis esmapilgul võivad tekkida. Autor toob siinkohal välja mõned tavalisemad ja lihtsasti segi aetavad arusaamad, mis tegelikult ei ole korrektsed ja vajavad lahti seletamist.

DevOps tähendab automatiseerimist

Andy Lipnitski [9] toob välja, et automatiseerimisel on DevOps’i juures suur roll, kuid neid ei ole võimalik võrdsustada. DevOps kultuurina sisaldab endas palju rohkemat kui puhtalt tarkvaraarenduse automatiseerimist. See on vaid väike osa antud metoodika võimalikust edukast rakendamisest.

DevOps on vaid kindlate vahendite kasutamine

Jason Hand [11] kirjeldab, et uute konfiguratsioonihalduse instrumentide kasutuselevõtt ei loo metoodilist muutust organisatsioonis. Selliste vahendite kasutamine on DevOps’is oluline ja kasulik, kuid vaid nendest ei piisa uue töökultuuri loomiseks ja muudatusi on vaja sisse tuua ka muus osas.

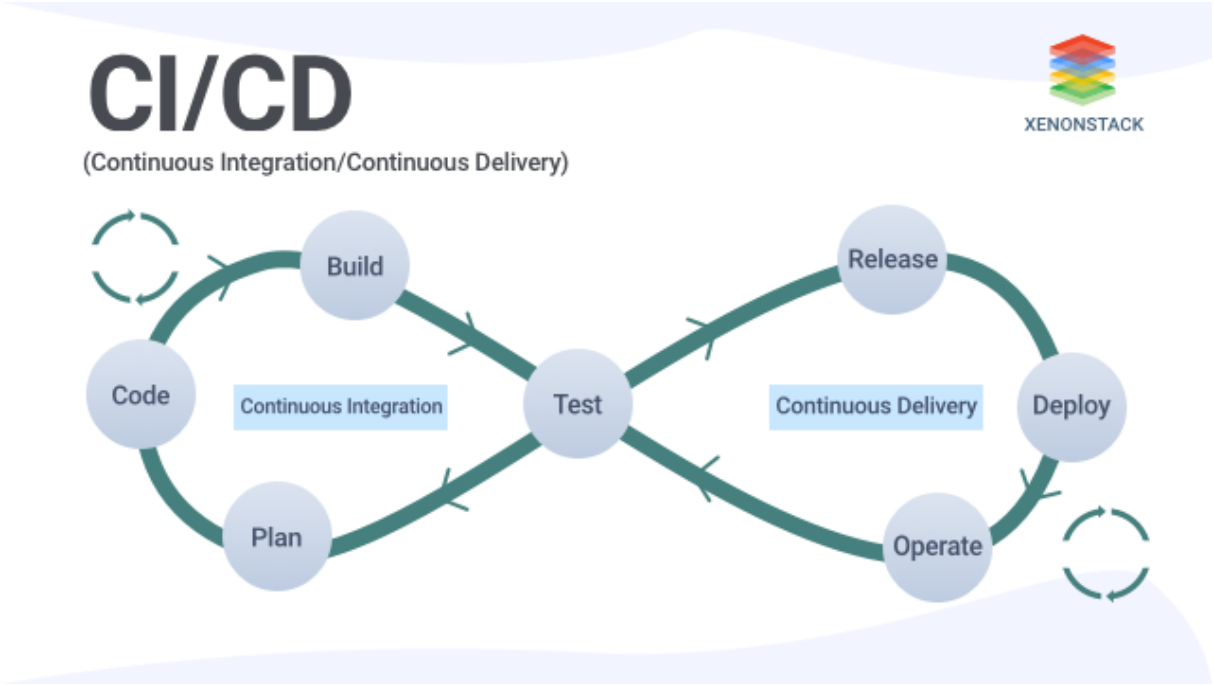
Samal arvamusel on ka Andy Lipnitski [9], kes toob esile, et on olemas väga palju vahendeid, mille abil on võimalik muuta DevOps’i efektiivsemaks. Samas peavad töötajad peale nende tehnoloogiate kasutamise õppima rakendama pidevat testimist, integreerimist ja ka tarnet. Selle kõige eesmärk on edendada tarkvara lõplikku kvaliteeti ja kiirendada selle arendust.

DevOps'i kasutuselevõtt tähendab selle edukat rakendamist

Edukaks metoodika rakendamiseks juhiste andmisest ja DevOps'i spetsialisti palkamisest Jason Handi [11] sõnul ei piisa. Selle nimel peavad kõik koostööd tegema, sest ühe inimese pingutustest ei piisa vastava töökultuuri loomiseks. Selle edukaks toimimiseks on vaja panuseid igaühelt ning seda vastavalt DevOps'i nõuetele.

2.2 Rakendamise alused

Oma teoses kirjeldab Sanjeev Sharma [3], et DevOps'i aluseks on pidev integratsioon (inglise keeles *continuous integration)* ja pidev tarne (inglise keeles *continuous delivery*). Ilma nendeta ei ole võimalik DevOps'i rakendada. Kõik muud laiendused ja lisad on boonusteks, kuid vajaliku baasi moodustavad just need kaks kontseptsiooni. Nende eesmärgiks on minimeerida tsükli aega, mis kujutab endast mõne protsessi või toote tarnimist kliendini, alustades kasutusloost. Joonis 1 ja joonis 2 illustreerivad eeltoodud käsituse olulisust ja täpsust.



Joonis 1. Pideva integratsiooni ja pideva tarne joonis [12].

Joonis 2. DevOps'i konveieri diagramm [13].

2.2.1 Continuous integration

Tänapäeval uue toote või teenuse väljastamine nõuab tööd nii selle enda kallal, kui ka süsteemi juures, kuhu antud toode või teenus soovitakse integreerida. Sanjeev Sharma [3] kirjeldab, et tihtipeale võib selline lõimimine osutuda palju keerulisemaks kui algselt võiks tunduda. Kuna uue teenuse arendajad ei pruugi olla juba olemasoleva süsteemi arendajad, siis tuleb neil integreerida oma loodud toode kellegi teise loodud süsteemiga ja/või juba olemasolevate teiste teenustega. Selline olukord tõstab integratsiooni tootearenduses väga olulisele kohale. *Continuous integration* viitabki sellisele pidevale teguviisile ehk pidevale integreerimisele. See tuleneb eelkõige välearenduse metoodikast.

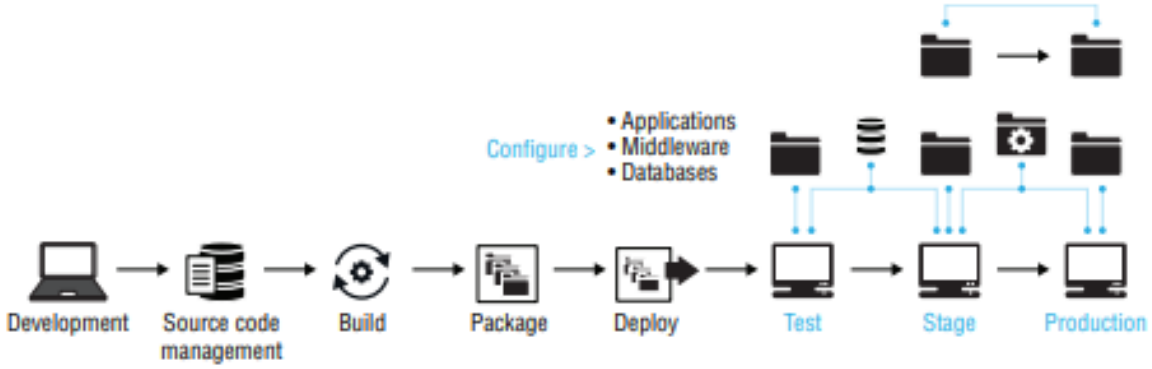
Sanjeev Sharma [3] toob välja, et traditsioonilises tarkvaraarenduses oli lõimimine alles üks viimastest või vähem tähtsatest osadest arendusprotsessis, mil tihtipeale tuli juba terviklik toode, teenus või osa nendest, mille kallal töötasid erinevad arendustiimid, integreerida olemasolevasse süsteemi või teiste tiimide loodud osadega. Selline lahendus oli kulukas ja lisaks veel etteaimamatu, sest lõimimise käigus tuleb ette probleeme ja kokkusobimatust. Tekkinud defektide eemaldus võttis aga palju aega, sest muutusi tuli sisse viia terviklikus tootes. Seepärast kujunes välearenduses välja lahendus, kus arenduse käigus lõimiti osasid

kokku järjepidevalt. Nii avastati probleemid varakult ja ümber tuli töötada ainult olemasolev lahendus, mitte juba valmisolev toode. Lisaks on sellise metoodika puhul võimalik juurde lisada ka pidev tarne, mis tähendaks, et vastavad integreeritavad osad viiakse üle süsteemi testkeskkonda, kus kontrollitakse ühilduvust ja sobivust.

Sarnaselt toovad Gene Kim jt [2] välja, et traditsiooniliselt lõimiti koodi alles projekti lõpus ja siis esines tihtipeale sellepärast palju tõrkeid. Tuli tööd ümber kirjutada, et jõuda positsiooni, kus oleks võimalik toodet üldse tarnida. Lisaks, kuna seda tehti just projektide lõppfaasis, võttis see ka rohkem aega. Tähtaegadest kinnipidamiseks oli seetõttu vaja sellistel puhkudel jätta osa tõrkeid lahendamata või lahendada need hooletult. Arenduse käigus võis selline probleem üha rohkem süveneda – kuna koodi lõimimine oli ebameeldiv ja kulukas, tehti seda üha harvemini. Lahenduseks kujuneski pidev integratsioon, mille käigus arendajad lõimisid oma töid vähemalt iga päev. Selle abiga tõusis töö produktiivsus ja vähenes aeg, mida kulutati tehniliste tõrgete lahendamiseks.

2.2.2 Continuous delivery

Eelpoolmainitud pidev tarne on teine väga oluline aspekt DevOps'i juures. Sanjeev Sharma [3] kirjeldab seda kui lisasammu edasi pidevast integreerimisest. Lõimimise juures on võimalik antud versioon saata edasi testimiseks, et tagada selle kvaliteet. Iga versiooni puhul seda tegema ei peaks. Nii tuleks käituda alles sellises etapis, kus toode või teenus on funktsionaalne ja seda on võimalik testida. Seejärel on võimalik see edastada käitusesse, kus saab vastava versiooni tarnida tootmiskeskkonda. Ka selles etapis tuleks sõeluda välja õiged versioonid, mida edasi saata. Funktsionaalsed ja stabiilsed variandid saab tarnida, muu vajaks veel tööd ja parandusi. Kogu sellise ahela eesmärk on uued tooted või teenused võimalikult kiiresti kliendini toimetada ja selle jaoks on vajalik tarnekonveier (inglise keeles *delivery pipeline*).



Joonis 3. Tarnekonveier arendusest kuni produktsioonikeskkonda [3].

Sanjeev Sharma [3] selgitab, et väljastus on tarvis automatiseerida, et selline tarnekonveier üldse oleks võimalik luua. Pidev integreerimine toob konveierile stabiilse tempoga uusi versioone ja seetõttu peab iga variant kiiresti ja automaatselt läbima testimiskeskkonna funktsionaalsuse testid, integratsioonikeskkonna stabiilsuse testid ja lõpetuseks testide läbides jõudma tarnimiseni. Tähtis on teada, et selline lähenemine ei tähenda, et iga versioon jõuab tootmiskeskkonda ja lisaks ei pruugita korraga tarnida tervet toodet või teenust, vaid ainult osasid sellest.

2.2.3 Abistavad tavad DevOps'i edukaks rakendamiseks

Varasemalt välja toodud pidev tarne ja pidev integreerimine on kaks põhilist aspekti, ilma milleta ei ole võimalik DevOps'i rakendada. Samas ei ole puhtalt nende kahe abil selline rakendamine eriti efektiivne ja edukas. Autor toob siinkohal välja mitmeid aspekte, mis tulevad abiks DevOps'i edukaks kasutamiseks ning vastavad lähenemised on välja toodud järgnevalt:

Koodipõhine infrastruktuur

Sanjeev Sharma [3] kirjeldab, et pideva integreerimise tõttu väljastavad arendajad varasema traditsioonilise tarkvaraarendusega võrreldes palju kiiremini uusi muudatusi ja seetõttu kasvab keskkondade arv, mida käitus peab haldama, üha suuremaks. Enam ei piisa olemasoleva keskkonna konfiguratsiooni muutmisest konsooli abiga, vaid on vajalik iga uue toote või teenuse tarne puhul luua uus keskkond, kus vajadusel oleks muudetud konfiguratsioon ja saaks vastavat arendust testida. Lisaks on sellise tegevuse puhul eriti tähtis kiirus, sest on halb, kui arendused jäävad keskkonna puudumise või sobimatuse tõttu tööjärge ja kuhjuvad. Selle vältimiseks ei jaksa käitus enam manuaalselt luua keskkondi ja tuleb üle minna skripti- ehk koodipõhisele infrastruktuurile. Skriptide abil on võimalik kiiresti ja automaatselt luua uued

vajalikud keskkonnad ning spetsiifiliste vajaduste rahuldamiseks tuleb vastavalt välja valida õige skript. Jääb vaid üle selliste skriptide haldamine. Paljud raamistikud on abiks sellise tegevusviisi lihtsustamiseks, näiteks Chef, Puppet, Salt ja Ansible.

Pidev tagasiside

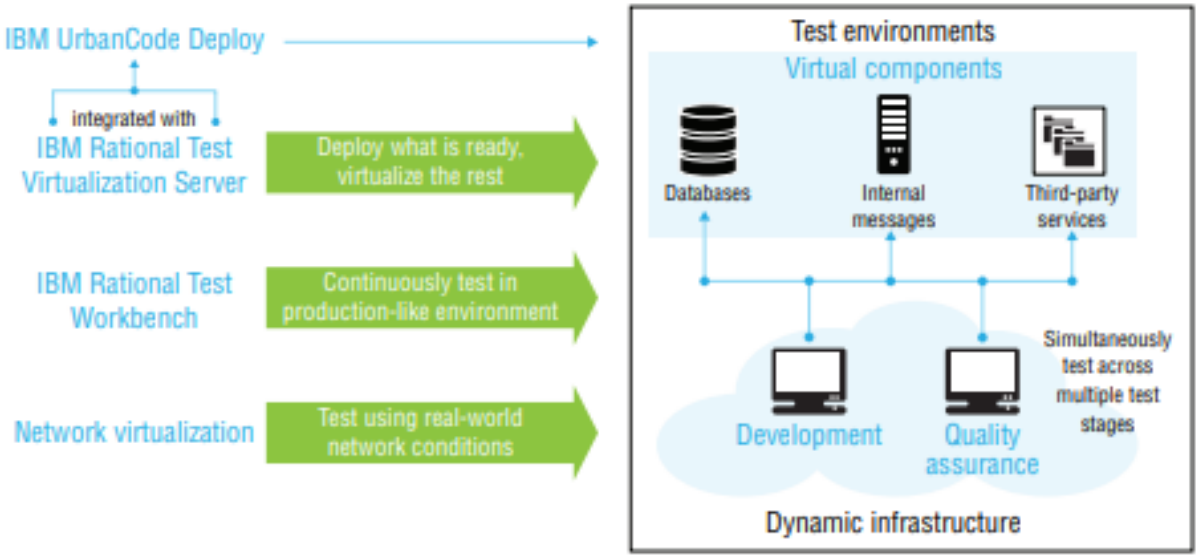
Nick Galbreath viis läbi pideva juurutuse uuringu, milles kajastas ka tagasiside olulisust [20]. Gene Kim jt [2] toovad selle läbiviidud uuringu tulemusena välja, et pidev tagasiside julgustab töötajaid olema enesekindlamad ja tundma end töökeskkonnas turvaliselt. Uuringu alguses pandi tähele, et töötajad nii käituses kui ka arenduses olid ebakindlad ja ei soovinud valminud koodi juurutada. Justkui valitses kartus, et kui juurutusel läheb midagi valesti ja kogu süsteem kannatab, on see just selle juurutuse teostanud töötaja süü. Lõpuks kui keegi julges oma muudatused produktsioonikeskkonda juurutada ja seal tekkiski tõrge, puudus piisav telemeetria, et see viga koheselt esile kerkiks ning tihtipeale jõudis teade veast kohale alles kliendi kaudu. Lahenduseks püüti iga arenduse puhul suurendada ka produktsioonikeskkonna telemeetriat. Juhul kui juurutuse tulemusena tekkis tõrge, said arendajad sellele enne jälile kui klient ning üheskoos jõuti probleem ära parandada. Tulemusena kasvas ka töötajate enesekindlus, sest isegi kui tekkis tõrkeid, jõuti nendele kiirelt jälile ja otsustati, kas tehakse tagasipööre või kiire parandus.

Oma teoses toob Sanjeev Sharma [3] välja, et inglise keeles *Continuous feedback* tähendab eelkõige varasemalt joonisel 3 kajastatud tarnekonveieri iga funktsionaalse protsessi puhul tagasiside andmist kõikidele protsessidele, mis paiknevad joonisel sellest vasakul. Näitena peaksid arendajad andma tagasisidet teenuse või toote arhitektidele ja analüütikutele, käitus peaks tagasisidet andma aga peaaegu kõigile – kvaliteedi kontrollile, testijatele, arendajatele ja kõikidele muudele tootega seotud olevatele sidusgruppidele. Sellise teguviisi eesmärgiks on tagada, et loodud tooted või teenused töötavad nagu ette nähtud.

Pidev testimine

Sanjeev Sharma [3] seletab lahti, et pidev testimine eeldab, et tarnitavat toodet või teenust kontrollitakse tarnekonveieri igas etapis. See algab arendajatest endast, kes loodavaid komponente testivad. Seejärel integreeritakse vastavad komponendid kokku tooteks või osaks sellest ning viiakse uuesti läbi testid. Peale seda lahkub loodud toode arenduskeskkonnast kvaliteedi kontrolli, kus pannakse kood tööle uues keskkonnas, milles on võimalik läbi viia kõik vajalikud testid antud etapis. Kõige lõpuks jõuab toode käitusesse, kus omakorda viiakse läbi viimased süsteemi stabiilsuse testid. See toimub tootmiskeskkonnaga võrdses olustikus ja

kui kõik läheb plaanipäraselt, on võimalik see reaalsesse tarbekeskkonda tarnida. Siinkohal saab probleemiks eelarve. Kõikide vajalike testkeskkondade üleval hoidmiseks ning rakenduste ja muude teenuste vabastamiseks uue toote testimiseks on vaja palju ressursse. Seda on võimalik vähendada olemasolevate protsesside virtualiseerimisega, mida on kujutatud joonisel 4. Nii ei ole vaja reaalseid teenuseid või rakendusi vabastada, et teste läbi viia.



Joonis 4. Testide virtualiseerimine [3].

Gene Kim jt [2] kirjeldavad, et automatiseeritud testid aitavad arendajatel pidevalt veenduda, et nende kirjutatud kood reaalselt töötab. Üksuste testimine käib kiiresti, ainult mõne minutiga. Iga lõimimise juures töötavad testid kolmel tasandil ning lisaks tehakse need testid läbi iga kahe kuni nelja tunni tagant. Sellise testimise abil tõusis tohutult töö produktiivsus – päeva jooksul lõimiti kordades rohkem oma koodi ja arendajad said teha seda individuaalselt. Samas tuuakse välja, et sellist tootlikkust ei oleks võimalik toetada ilma pideva integratsioonita. Lõpptulemuseks võisid ettevõtte arenduskulud langeda ligikaudu 40% ja ühe programmi tasandil langesid vastavad kulud lausa kuni 78%. Lisaks suurenes pidevalt töös olevate arenduste hulk kuni 140%.

Pidev seire

Seire ülesandeks on püsivalt jälgida tarbekeskkonna seisundit. Sanjeev Sharma [3] toob näitena välja, et oluline on kindlustada, et süsteemid jookseksid veatult ja protsessid töötaksid ka kõige madalamal tasemel. Selliseks pidevaks seireks on olemas erinevad vahendid, kuid lisaks vaatlusele tuleks selliseid andmeid ka koguda ja analüüsida. Puhtalt toote või teenuse tasandil on käitusel võimalik koostöös arendajatega juba kindlasse produkti spetsiaalne jälgimistarkvara

sisse ehitada. Selline teguviis lihtsustaks tootepõhist seiret. Kõiki kogutud andmeid tuleks analüüsida ja jagada iga sidusgrupiga. Ainult nii on selliste andmete kogumisest reaalselt kasu. Heade andmete ja nende analüüsimise abil on võimalik pidevalt areneda ning kõik muudatused, ükskõik, millise sidusgrupi tasemel, on andmepõhised. Mida madalamal ja väiksemal tasemel sellist seiret teha, seda kasulikum on see organisatsioonile üleüldiselt.

Gene Kim jt [2] toovad välja, et hea teenuse tasemega ettevõtted taaskäivitavad oma servereid tõrgete eemaldamiseks kuni kakskümmend korda vähem kui traditsioonilised infotehnoloogiaettevõtted. Nad kirjeldavad, et varasemalt produktsioonikeskkonnas probleemide tekkel ei pruugitud teada, mis vastavat viga tekitas. Lahenduseks taaskäivitati järjest servereid ja kui see ei aidanud, siis süüdistati arendajaid vigases koodis. Edukamad ettevõtted oskasid paremini diagnoosida süsteemis esinevaid probleeme. Selleks loodi kaugmõõtmise võimalused toodetesse ja keskkondadesse sisse. Suurema telemeetria abil oli võimalik pidevalt erinevate toodete ja keskkondade seisukordi jälgida ja seeläbi teostada pidevat seiret. Tulemusena jõuti probleemidele kiiremini jälile ning tõrkeid tuvastati täpsemalt ja teenuse- või keskkonnapõhiselt.

Pidev äriplaneering

Lõpetuseks kirjeldab Sanjeev Sharma [3], et tänapäeval peavad organisatsioonid oma klientide tagasisidele kiirelt reageerima. Seetõttu peab äriplaneering olema kiire, kohanemisvõimeline ja mitte liiga suuremahuline. Planeerida tuleks väiksemaid osi korraga ja nendest edasi liikuda vastavalt tagasisidele. Kergemate eesmärkide saavutamiseks peavad organisatsioonid välja uurima, mida kliendid tegelikult tahavad ja seejärel pidevalt uuendama oma väljavaateid ja nägemust.

Kõik eelnevalt väljatoodud tavad on justkui lisad pidevale integreerimisele ja tarnele, kuid ilma nendeta oleks keeruline DevOps'i edukalt rakendada. Seepärast ongi DevOps palju rohkem kui puhtalt automatiseerimine või kindlate tehnoloogiliste vahendite kasutuselevõtt. See hõlmab endas kogu organisatsiooni kultuurilist arengut ja seisukohta.

2.2.4 DevOps'i kultuur

Muutused tehnoloogilisel tasemel ei tähenda muutusi töökultuuris. Sanjeev Sharma [3] toob esile probleemi, millega võib kaasneda DevOps'i rakendamise läbikukkumine. Selleks on kogu organisatsiooni kultuuriline muutus, ei piisa vaid testide automatiseerimisest ja kindlate tavade kasutuselevõtust. Kui inimesed on ühte kindlat moodi juba tööd teinud, siis on keeruline neile

selgeks teha, et korraga see kõik muutub. Tekivad arusaamatused, sest varasem versioon võis edukalt töötada ja töötajad ei näe vajadust millegi muutmiseks. Ühtsuse tekkeks on vaja kõikide töötajate panust. Isegi kui kõik näevad eeliseid DevOps'i rakendamiseks, aga ei näe selle nimel vaeva, ebaõnnestub vastav kultuuriline uuendus. Nii suure muudatuse edukaks sisseviimiseks on vaja esiteks usaldust. Muudatuste läbiviimiseks on vajalikud usaldus ja läbipaistvus kõikide töötajate vahel. Need on DevOps'i alustalad ja seetõttu pole ilma nendeta võimalik edasi liikuda. Lisaks tulevad juurde ka efektiivne kommunikatsioon ja ühtne hindamine läbi terve organisatsiooni. Puhtalt sõnumite ja e-maili teel suhtlemine ei ole piisav, vaja on otsest inimestevahelist kontakti ja seda kõikide töötajate vahel, kellel on parajasti omavahel vaja suhelda. Kõik töötajad peavad olema vastutavad tarbekeskkonda tarnimisel ja ainult nii on kõikide sidusgruppide tööd võimalik ühtsete või vähemalt sarnaste parameetrite järgi hinnata. DevOps'i rakendamine nõuab aega ja on tõenäoline, et algselt võib tootlikkus langeda, kuid samm-sammult õigete tavade poole liikudes on võimalik see varasemaga võrreldes peagi hoopis kõrgemale tasemele tõsta.

3. Põhja-Eesti Regionaalhaigla IT-teenistus

Käesolevas peatükis kirjeldab autor hetkelist PERH-i IT-teenistuse metoodikat ja töö ülesehitust. Autor loob eelnevas kahes peatükis välja toodud DevOps'i eeliste ja nõuete põhjal analüüsi, millega jõuab lõpuks arvamuseni, kas DevOps on IT-teenistuse jaoks sobilik metoodika. Lisaks pakub autor välja ühe kindla plaani, mille abil oleks soovi korral võimalik efektiivselt DevOps'i PERH-i IT-teenistusse integreerida ja seda ka edukalt rakendada.

Metoodika

Hetkeseisu analüüsiks viis autor läbi kolm eraldiseisvat küsitlust PERH-i IT-teenistuses. Nendeks olid küsitlused arendajatele, projektijuhtidele ja süsteemiadministraatoritele, mis on välja toodud lisas 1. Vastuste põhjal koostas autor analüüsi praegusest korraldusest, tuues sealt välja nii häid kui ka halbu tavasid. Seejärel loodi võrdlus kahes varasemas peatükis välja toodud DevOps'i teoreetilise poolega. Selle abil oli võimalik analüüsida, kui palju DevOps'i rakendamine IT-teenistuses muudaks, missugused antud muudatused oleksid ja kuidas nende kasutuselevõttu korraldada. Lõpetuseks koostas autor analüüsi ja eeliste ning puuduste alusel plaani, mille abil oleks võimalik DevOps'i integreerida IT-teenistusse.

Küsitluste koostamisel seadis autor eesmärgiks saada võimalikult hea ülevaade erinevate sidusgruppide probleemidest hetkelise töökorralduse koha pealt. Küsimuste loomisel oli näidiseks sarnasel teemal kirjutatud bakalaureusetöö [19]. Põhjalikuma eeskujuna kasutas autor magistritööd, mille eesmärk oli juurutada DevOps'i suurettevõttes [18]. Lisaks toetus autor küsimuste loomisel ka varasemalt läbiviidud uuringutele, kus kajastusid DevOps'i edukust määravad tehnilised näitajad [8]. Iga küsimuse sihiks oli vastavalt DevOps'i printsiipidele leida probleeme näiteks töö efektiivsuses nii arenduses kui ka käituses, suhtluses omavahel, töökultuuriga rahulolus ja ressursside kasutuses. Lisaks uuris autor lühidalt ka DevOps'i tundmise ja pooldamise kohta ning kuidas võiks uus töökorraldus olla töötajatele endile kasulik.

Küsitluste valmimisel saatis autor need PERH-i IT-teenistuses olevale kontaktisikule, kes omakorda jagas need vastavate töötajate vahel laiali. Küsitlused olid vajadusel vastaja jaoks töö autori suhtes täiesti anonüümsed, sest nad ei pidanud oma vastuseid saatma otse autorile ega ka lisama sinna oma nime või muud isiklikku informatsiooni. Samas kui nad vahendasid oma vastuseid läbi autori kontaktisiku PERH-is, siis kontaktisiku jaoks ei jäänud vastajad anonüümseks. Eesmärgiks oli saada vähemalt kaks vastajat igale küsitlusele, kuid kahjuks erinevate asjaolude tõttu jõudsid vastata nii projektijuhi küsitlusele kui ka administraatori

küsitlusele vaid üks inimene. Arendajate vastuseid oli kaks erinevat. Sellegipoolest tuli vastustest välja piisavalt probleeme ja murekohti, mida autoril oli järgnevas alampeatükis võimalik analüüsida ning edaspidi muudatusi soovitada.

3.1 Hetkeseisu analüüs

Käesolevas peatükis loob autor küsitluste vastustena saadud tulemuste põhjal analüüsi hetkelise töökorra kohta PERH-i IT-teenistuses. Eesmärgiks on välja tuua eelkõige probleeme ja muresid, et näidata, mida oleks võimalik DevOps'i rakendamisega edendada ja parandada. Välja on toodud ka positiivsed märkused, kuid vähesemal määral, sest küsitluse eesmärk oli välja tuua töö osad, mis vajaksid muudatusi.

Rahulolu

Rahulolu PERH-i IT-teenistuses on hetkeseisuga väga erinev. Kaks töötajat hindasid isiklikku rahulolu töökultuuriga kümne punkti skaalal hindega 7, üks valis hindeks 8, aga leidus ka vastaja, kes valis hindeks 3. Sellest võib järeldada, et vaja oleks sisse viia muudatusi. Töökeskkond, kus töötajad ei ole rahul, ei saa olla tõhus. Mitte keegi ei vastanud etteantud skaala põhjal hindega 10, mis tähendab, et iga küsitlusele vastanud töötaja meelest on ühel või teisel moel võimalik hetkest seisu parandada.

Efektiivsus

IT-teenistuse üldist toimimist ja efektiivsust hinnati kõige tihemini kümne punkti skaalal hindega 7, mis oli ühtlasi ka kõrgeim hinne. Leidus ka töötajaid, kes andsid sellele madalama hinde – kõige madalam oli hinne 5. Võrreldes isikliku rahuloluga, hinnatakse hetkelist üldist töökeskkonna toimimist kõrgemalt. Samas ei ole ka selle tulemused märkimisväärsed – ühtegi 10 puntkilist hinnangut vastusena ei tulnud. Seetõttu oleks vaja viia sisse muudatusi, mis võiksid kogu IT-teenistuse tõhusust tõsta.

Efektiivsuse osas toovad arendajad välja, et ressursside raiskamist esineb hetkel suhteliselt laiamahuliselt. Järgnevalt toob autor välja küsitlusest võetud näiteid:

- Esineb arendusi, mida ei viida kunagi lõpuni, kuigi sinna on kulunud palju ressursse.
- Vana koodi ümberkirjutamine on kulukas ja aeglane. On hetki, kus uue tarkvara loomine oleks odavam.

- Arenduste loomisel puudub loogiline planeerimine süsteemidevahelist sõltuvust arvestades. Näiteks alustatakse süsteemi B arendusega, mis sõltub süsteemist A. Süsteem A on aga läbimas muudatusi ja nende valmimisaeg on ebaselge.
- On olukordi, kus luuakse vajadusepõhiselt uut funktsionaalsust, kuid töö valmimisel jäetakse see tahaplaanile ning testimine lükkub oluliselt hiljemaks.
- Esineb ajaraiskamist puhtalt sellepärast, et erinevates keskkondades on töötajatel puudu vajalikud õigused ning muudatuste sisseviimiseks on vaja leida keegi teine, kellel need on olemas.

Süsteemiadministraator toob ressursside raiskamisena välja järgnevad probleemid:

- Esineb ebavajalikke koosolekuid.
- Tellitakse teenuseid, mida hiljem ei lähe vaja või osutuvad need valeks.

Projektijuht toob probleemidena välja järgnevad asjaolud:

- Projektid jäävad lõpetamata, sest prioriteedid muutuvad.
- Korraga proovitakse töölaual hoida liiga palju asju.
- Eelnev probleem põhjustab omakorda kvaliteedi langust ja tähtaegadeks mitte valmis jõudmist.

Kokkuvõtvalt selgub, et on palju asju, mille parandamisega oleks võimalik kogu IT-teenistuse efektiivsust tõsta. DevOps'i abiga saaks ära kaotada nii ebavajalikud arendused, kehvad planeerimised kui ka lihtsalt ajaliselt ebatõhusad koosolekud. Viimasele probleemile ei oleks DevOps otseselt lahenduseks, vaid vaja oleks lihtsalt konkreetsemaid plaane koosolekute jaoks. Uue metoodika edukal rakendamisel ei tohiks esineda projekte, mis jäävad vajaduse puudumise tõttu lõpetamata ning arenduses võiks valitseda parem arusaam ja kord.

Suhtlus

Omavahelise suhtluse osas töötajatel muresid otseselt ei ole. Tuuakse välja, et üksteisega suheldakse tihedalt ja inimesed on abivalmid ning avatud. Ühe küsitluse vastusena on kirjeldatud, et esineb probleeme puuduliku kommunikatsiooni ja info ebapiisava levimise tõttu. Sellised tulemused on siiski väga positiivsed ja kuna DevOps'i aluseks on hea suhtlus, siis võiks vajadusel üleminek uuele metoodikale olla üpris sujuv.

Kommunikatsioonivahenditest kasutatakse hetkel näiteks MS Teams'i, Jira't, Outlook'i ja ka tavapärast telefoniga helistamist. Üldiselt on e-maili kasutamine suhteliselt aeglane ja seetõttu

on hea, et kasutusel on ka näitaks MS Teams, kus on koheselt võimalik kindlatele töötajatele kiireid märkusi või ülesandeid anda. Lisaks on võimalik teatada ka tekkinud probleemidest või võimalikest puudujääkidest.

Arendus

Hetkel kasutatakse tarkvaraarenduses selliseid vahendeid nagu Atlassian Stack, Visual Studio koos Resharper laiendusega, Eclipse, Liquibase, Artifactory, Graylog, Enterprise Architect. Lisaks muidugi kasutatakse veel Stack Overflow'd ja Google'it. Versioonihalduseks kasutatakse Git'i. Erinevaid vahendeid on kasutuses suhteliselt palju ja nende tõttu probleeme ei esine.

Murekohana võib sisse tuua parajasti käesolevate toodete või teenuste arendused. Probleemina tuuakse küsitluses välja, et tööde omavahelist lõimimist esineb liiga vähe. Iga tiim töötab enda arenduse kallal ja teistega ei suhelda. Lisaks valmivad uued tooted lausa nii aeglaselt ja kulukalt, et vahel otsustatakse selle asemel hoopis olemasolevaid teenuseid täiendada, kuigi need on mõeldud hoopis millekski muuks. Sellise tegevuse all kannatab kogu süsteemi arhitektuur.

Seire

Süsteemi seireks kasutatakse hetkel selliseid vahendeid nagu Graylog ja Zabbix. Viimase abil on teostatud ka pidev seire. See on väga hea, sest pidev seire on üks DevOps'i abistavatest tavadest ning selle olemasolu muudab vajadusel uuele metoodikale ülemineku kindlasti lihtsamaks. Küsitluse põhjal võib uue toote tarnimine arendusest kliendini võtta aega paar kuud kuni aasta, mis on sellises kontekstis suhteliselt pikk aeg. Lisaks pikale tarneajale tuleb probleemina välja, et iga tarne puhul produktsioonikeskkonda on vaja teha muudatusi või parandusi. See tähendab, et mitte kunagi pole uus toode või teenus veel päris valmis ja korralikult läbi testitud. DevOps'i kultuur näeb ette, et muudatusi tehakse pidevalt, kuid igas tarnes ei tohiks esineda tõrkeid ja need tuleks võimalikult varajastes arendusetappides likvideerida.

Testimine ja automatiseeritus

Süsteemiadministraatorid ei vii uusi tooteid või teenuseid tarnides läbi ühtegi testi. Ainult arenduses viiakse läbi teste, kus on mõne toote arendamise juures kasutatud testijaid. Nende ülesandeks on kasutajaliidese vaates erinevaid tegevusi läbi teha ja vigu leida. Automaatteste enamasti ei kasutata. Siit võib tuleneda eelnevalt kajastatud probleem, et iga uue tarne puhul on

vaja läbi viia muudatusi ja parandusi. Väga vähene testimine laseb tihtipeale läbi vigased arendused ja seetõttu, et töö efektiivsust tõsta ja tõrgete arvu vähendada, oleks kindlasti vaja sisse viia rohkem teste nii arenduses kui ka tarnes.

Küsitlusest tuleb välja, et arenduse poolelt on uue toote ehitamine ja paigaldamine automatiseeritud – ainult varundada tuleb käsitsi. Samas varasemalt kajastatud testimine ega ka staatilised kontrollid ei ole automatiseeritud. Tarne poolelt ei ole ükski protsess automatiseeritud ning alati on vaja administraatori sekkumist. Pidev tarne on DevOps'i üks alustavasid ja selle edukaks toimimiseks on vaja suur osa tarnest automatiseerida. Seega on uuele metoodikale üle minnes vaja panna suurt rõhku erinevate tarneetappide automatiseerimisele.

Vead ja tõrked

Seire kajastusest tuli välja, et iga uue teenuse või toote tarnimisel esinevad vead või tõrked, millele kulub omakorda aega, et süsteem jälle töökorda saada. Arendajate vastustest tuleb välja, et probleemide eemaldamiseks läheb liiga kaua aega. Tuuakse välja, et see aeg peaks olema tundides, aga reaalsuses on hoopis päevades ja vahel isegi nädalates. Muidugi oleneb see probleemi suurusjärgust. Administraatorite poolelt tuleb välja, et probleemide lahendamine võtab väga erinevalt aega. Näiteks võib vigade likvideerimisele kuluda mõni tund, kuid võib minna ka kuu aega.

Tagasiside

Administraatorid toovad välja, et töö eest saadakse tagasisidet nii kirjalikult kui ka suuliselt oma kolleegidelt ja ülemuselt. Samas töö efektiivsust ei mõõdeta kuidagi ning seetõttu võib raske olla kellegi tööle otsest hinnangut anda. See võib omakorda põhjustada ebamäärase ja ebaausa hindamise erinevate töötajate vahel, sest pole alust, mille põhjal tööd hinnata. Arendajate vastustest tuleneb, et tagasisidet saadakse tootejuhtidelt. Samas kirjutatakse ka, et tagasiside tuleb alles siis kui arendus jõuab produktsioonikeskkonda ja tekivad vead. Positiivse poolena tuuakse välja, et vahel on jõudnud kliendi tänulik tagasiside ka lõpuks arendajateni. Samas ei tohiks see olla harv juhus, vaid iga kliendi tagasiside peaks jõudma kõikide töötajateni, kes vastava toote või teenusega seotud on.

Muutused ja DevOps

Küsitluse tulemusena on suur osa töötajatest vähemalt vähesel määral kursis DevOps'i ja selle kultuuriga. Lisaks pooldaksid kõik DevOps'i tundvatest töötajatest ka selle rakendamist IT-

teenistuses ja seda eriti just arendajad. Metoodika muudatustesse suhtuvad kõik väga positiivselt. Uute vahendite kasutuselevõtt ja õppimine oleks vastanute sõnul hea ning küsitluse põhjal võiks uusi vahendeid kasutades palju juurde õppida ja saada teada, millistest vahenditest on varasemalt tegelikult puudust tuntud. Lisaks mõjuks selline kultuuriline muutus positiivselt ka töötajate karjääridele. Tuuakse välja, et uued ja erinevad kogemused aitavad alati targemaks saada ning seda saaks ka oma CV-s kajastada.

3.2 Eelised ja puudused DevOps'i kasutamiseks Põhja-Eesti Regionaalhaiglas

Uue metoodika integreerimine suures asutuses on alati keeruline, kuid sellega võivad kaasneda eelised, mis näiteks tõstavad üldist töö efektiivsust või muudavad töö keskkonda meeldivamaks ja ka puudused, mis omavad vastupidist mõju. Järgnevalt toob autor välja, kuidas võiks DevOps olla nii kasuks kui ka kahjuks PERH-i IT-teenistusele. Eesmärgiks on esile tuua võimalikult olulisi tugevaid ja nõrku külgi, mis iseloomustaksid DevOps'i kasutust IT-teenistuses. Analüüsi aluseks on võetud varasemates peatükkides läbi töötatud materjalid ja hetkeseisu analüüs.

Kultuuriline muutus

Suurimaks muutuseks IT-teenistuse töös saab kindlasti olema üldine kultuuriline muutus ja DevOps oma sisult ning definitsioonide kohaselt ongi justkui omaette kultuur. Küsitluse vastustest tuleb välja, et praegune tööstruktuur on suhteliselt vana ja kaua kasutuses olnud. Seetõttu võib sellise kinnistunud süsteemi ja üleüldise kultuuri muutmine olla keeruline. Selleks, et luua edukalt uus süsteem töökeskkonnas, peavad kõik töötajad võrdselt sellesse panustama. Algselt võib töö efektiivsus langeda, sest metoodika on uus ja vajab harjumist ning harjutamist. Sellegipoolest, kui on ühiselt vastu võetud otsus DevOps'i peale üle minna, ei tohi algsetel negatiivsetel tulemustel lasta motivatsiooni ja tahet mõjutada. Uuringud on näidanud, et DevOps'i kasutavad ettevõtted võivad olla kordades efektiivsemad ja edukamad [7]. Sellise kultuuri omandamine võib osutuda tohutult kasulikuks nii üldisele töökorraldusele kui ka töötajatele individuaalselt. Läbiviidud küsitlustest selgus, et uute vahendite kasutuselevõtt ja uue metoodika õppimine oleks ainult positiivne kogemus, mida kõik vastanud töötajad toetaksid.

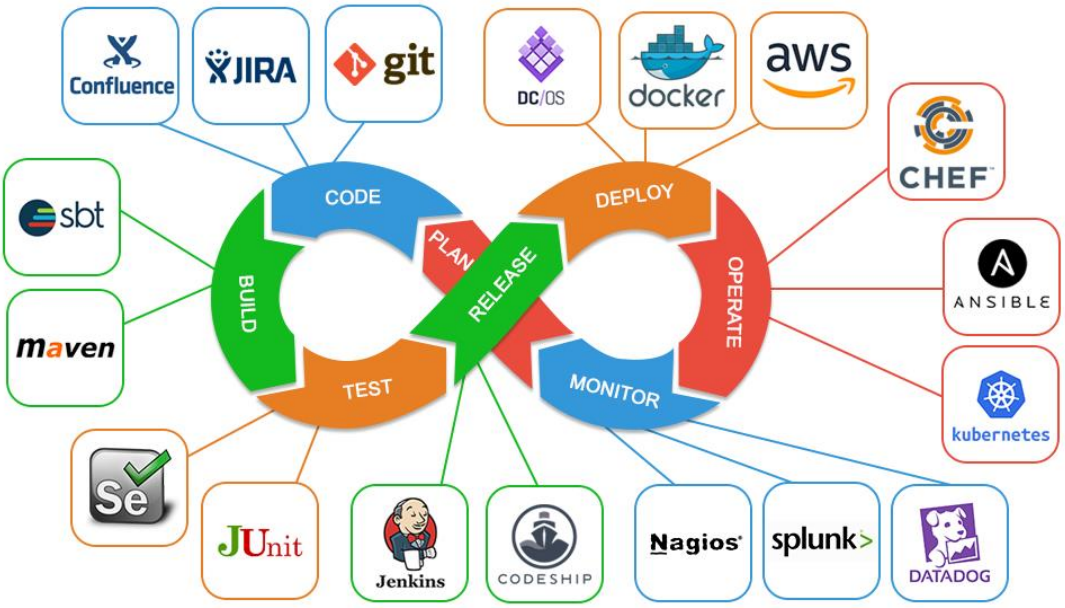
Uued vahendid

DevOps'i rakendamisel on kindlasti vaja olemasolevate vahendite asemele või nende kõrvale võtta erinevaid tööriistu. Olemasolevatest vahenditest võiks arenduses edasi kasutada Visual Studio't, Eclipse'i, Liquibase'i, Artifactory't, EnterPrise Architect'i ja Graylog'i. Uute tööriistadena võiks kasutusele võtta konteinerite tehnoloogia, mille kasutamine aitaks vältida tõrkeid uute teenuste ja toodete integreerimisel olemasolevasse süsteemi. Docker on üks selline vahend, mille abiga on võimalik arendajatel luua tarkvarale konteinereid ning nende abiga väga lihtsalt vastavaid loodud lahendusi jagada ka käitusega. Selliselt on võimalik ka eemaldada probleemid, kus arendajate loodud tarkvara töötab nende enda testkeskkonnas, aga kui loodud lahendus produktsioonikeskkonda integreeritakse, siis tekivad vead. Konteinerite abiga on arendajatele kättesaadav kõik sellisel kujul nagu see on ka süsteemis endas. Seega saavad arendajad testida täpselt sellistes tingimustes nagu need tegelikult on. Lisaks muudab selline lahendus ka administraatorite tööd lihtsamaks.

Nish Anil jt on oma postituses [15] välja toonud, et Docker ja konteinerid on alus koostööks DevOps'is. Nad toovad välja analoogi Docker'i konteinerite ja reaalses elus kasutatava konteinerite transpordi vahel. Konteineri omanikud ei pea teadma, kuidas nende omandid kohale jõuavad ja laevafirmad ei pea teadma, mis antud konteinerite sees on. Arendajad loovad oma lahendused konteineritesse ning seejärel on käituses võimalik lihtsalt selle konteineri abiga loodud tarkvara süsteemi integreerida. Süsteemiadministraatorid ei pea teadma, mis täpselt konteineri sees on, justkui päriselus transpordis. Samamoodi ei pea arendajad täpselt teadma, kuidas nende konteiner produktsioonikeskkonda jõuab.

Lisaks Docker'ile peaks uute vahenditena kasutusele võtma ka näiteks Jenkins'i, Chef'i, Ansible'i või muu pidevat integratsiooni ja tarnet hõlpsustava vahendi. Autori soovitusena võiks kasutusele võtta Jenkins'i, sest seda on lihtne kasutada mugava veebiliidese abil, Jenkins'il on palju laiendusi, mis hõlbustavad käituse ja arenduse tööd ning lisaks on seda võimalik kasutada nii pideva tarne, pideva integratsiooni kui ka automatiseerimise jaoks[1]. Pidev tarne on DevOps'i alustala ning ilma selleta ei ole võimalik vastavat metoodikat edukalt rakendada. Eelnimetatud vahendite eesmärgiks on automatiseerida võimalikult suur osa tarnekonveierist. Selle tulemusena jääb omakorda vähemaks inimlike eksimuste sooritamine näiteks uue toote või teenuse tarnel. Rohkemate näidete eesmärgil on võimalikud vahendid DevOps'i sammude jaoks välja toodud joonisel 5.

[1] jenkins.io



Joonis 5. DevOps'i konveieris võimalikud kasutatavad vahendid [16].

Samas läheb DevOps'i metoodika juures vaja ka praegu kasutusel olevaid vahendeid. Tarkvara arenduses on tiimidel vaja pidevalt oma lahendusi teistega jagada ja lõimida ning selle jaoks on olemas Git. See on arendajatel ka hetkel kasutusel versioonihalduseks. Lisaks kasutatakse suhtluseks näiteks MS Teams'i, mis on väga hea vahend ja kõrgelt hinnatud DevOps'i kultuuris. Kui nendele vajalikele olemasolevatele vahenditele lisada ka uued ja samal ajal kaotada hetkel kasutuses olevad ebavajalikud vahendid, võib tekkida palju arusaamatusi. Töötajatel võib olla raske ümber harjuda uue tehnoloogiaga kui vana on juba väga pikalt kasutatud. See on üks probleem, mis võib DevOps'ile üle minnes esineda.

Kultuurilise poole pealt, võib algselt uue metoodika puhul eeldadagi efektiivsuse langust, sest töötajad on alles õppimise faasis. Motivatsiooni ja raske tööga on võimalik uute vahenditega aga kiiresti tuttavaks saada ja lisada omandatud oskused juba olemasolevate kõrvale. Seejärel efektiivsus jälle tõuseb ja ennustatavalt ka kõrgemale, kui see oli varasema metoodika puhul.

Kiiremad ja kvaliteetsemad tootearendused

DevOps'i abiga on võimalik luua PERH-i IT-teenistuses efektiivsem töökultuur kasvõi ainult kiirema ja kvaliteetsema tootearenduse poolest. Metoodika võiks lahenduse tuua eksisteerivatele probleemidele arenduses nagu näiteks vähene tööde lõimimine. Praegu võib IT-teenistuses ette tulla olukordi, kus erinevad arendustiimid ei suhtle üksteisega piisavalt ning töötavad vaid endale antud ülesannete kallal nii kaua, kuni need on valmis. DevOps'i praktika nõuab pidevat lõimimist ja koostööd, et erinevaid ühilduvusvigu ja muid tõrkeid võimalikult

varakult ennetada. Sellega on võimalik vähendada lõpliku toote või teenuse puhul esinevaid probleeme ning vältida vajadust iga tarne puhul parandusi teha. Lõpptulemuseks ongi tõhusam arendus, mis tagab kvaliteetsema toote ja meeldivama tulemuse kliendi jaoks.

Kui selle metoodika tavadest kinni pidada, ei edene ainult arendus, vaid ka kõik sellega kaasnev. Tagasiside põhjal loodavad edasiarendused on väledamad ning vigade likvideerimine toimub kiiremini ka käituses. Esimeses peatükis välja toodud eelisena, aitab DevOps juurutada efektiivsetel organisatsioonidel oma arendusi 30 korda tihemini ja seda kõike 200 korda kiiremini. Sarnased organisatsioonid taastuvad kordades kiiremini palju harvem esinevatest riketest. Sellepärast aitab see metoodika muuta toote või teenuse tarne ja juurutamise arendusest kliendini jõudmiseni kordades kiiremaks ilma, et peaks ohverdama töökindlust. DevOps oleks sarnaselt arendusele seega ka suureks abiks käituse efektiivsuse tõstmisele.

Tagasiside

IT-teenistuse hetkeseisu analüüsist selgus, et töötajate saadav tagasiside on ebamäärane ja pole kindlal viisil teostatud. DevOps'i puhul on kokkulepitud meetod, kuidas saab iga sidusgrupp tagasisidet tehtud töö eest. See aitab tagada, et loodud muudatused töötavad nii nagu vaja. Lisaks on sellisel viisil tagasiside iga töötaja jaoks ühesugune ja kõik saavad oma panuse eest harivat ja õpetlikku kriitikat. Muidugi ei tähenda see ainult negatiivset tagasisidet, vaid ka positiivset. Hea töökultuuri jaoks on vaja hästi tehtud tööd tunnustada. Arendaja küsitlusest tuli välja, et kui kliendi hea tagasiside jõuab arendajani, siis see on väga positiivne kogemus. Sarnaselt tuleks käituda ka organisatsioonisiseselt ja kiita üksteise hästi tehtud tööd. Selline teguviis võib aidata tõsta moraali ja inimeste töö kvaliteeti.

Finantseerimine

Algselt võib suurimaks probleemiks DevOps'i kasutuselevõtu puhul olla just finantseering. Uute vahendite kasutuselevõtt, koolituste läbiviimine, vajadusel uute töötajate palkamine – kõik eelnevad tegevused ja muudki, mis võivad esineda, vajavad rahastamist. Uute vahendite kaasamine töösse tähendab, et ettevõte peab vajadusel ostma vastavad litsentsid ja load, et sellist tarkvara üldse kasutada. Muidugi leidub ka tasuta tarkvara, kuid tihtipeale on sellistel tööriistadel omad hinnad. Kuna DevOps'iga kaasneb päris palju erinevaid vahendeid, kulub selle peale ka omajagu raha. Kulude alla lisanduvad ka veel koolitused ja õpetused, et töötajad saaksid uue metoodikaga ja kasutatavate vahenditega kiiremini tuttavaks.

Selline rahaline kulutus võib algselt tunduda liialt suur ja eemaletõukav, kuid arvestada tuleb ka edasist kokkuhoidu. DevOps'i üheks eesmärgiks on muuta töö võimalikult efektiivseks. Sellise tõhususe saavutamine kaotab ära varasemalt välja toodud probleemid ressursside liigse raiskamise osas. See tähendab, et kui algne investeering tundub suur siis selle juurde tuleb arvestada ka kogu seda ressurssi ja raha, mis eduka DevOps'i rakendamise pealt tulevikus kokku hoitakse. Muidugi tuleks sellisel juhul teha konkreetsed arvutused ja ennustused, mida autoril ei ole võimalik luua. Need annaksid täpsema pildi ja looksid arusaama, kas algne investeering võib saada probleemiks või hoopis suureks eeliseks.

Kokkuvõte

Võrreldes praeguse süsteemiga, nõuab DevOps'i rakendamine PERH-i IT-teenistuses päris suuri muutusi. Selliste muutuste edukaks läbimiseks on vaja nii tahet, planeerimist kui ka rahalisi võimalusi. Arvestades aga hetkelise olukorra analüüsi, tundub DevOps justkui lahendusena pea iga esineva probleemi jaoks. Kokkuvõtvalt on eelised ja puudused välja toodud tabelis 1.

Tabel 1. DevOps'i võimalikud eelised ja puudused PERH-i IT-teenistusse rakendamisel

	Eelised	**Puudused**
Töökultuur	• Suurem töötajate individuaalne rahulolu	• Töötajad peavad kohanema uue kultuuriga • Alguses tõenäoline töö efektiivsuse langus
Suhtlus	• Sarnane süsteem võrreldes olemasolevaga • Arendab praegust seisukorda edasi – muudab suhtlust tihedamaks	
Arendus	• Kvaliteetsem arendus • Kiirem arendus • Rohkem lõimimist • Vähem ressursside raiskamist	• Uued metoodikad vajavad õppimist ja harjumist
Käitus	• Rohkem automatiseeritud • Kvaliteetsem juurutus	• Uued metoodikad vajavad õppimist ja harjumist

	• Kiirem juurutus • Kiirem tõrgetest taastumise aeg	• Võib vaja minna rollide ja tööülesannete muutust
Töövahendid	• Uued oskused individuaalsel tasemel • Iga etapi on jaoks vajalikud tööriistad	• Koolituste läbimise vajadus, et tutvuda uute vahenditega • Algne koormuse suurenemine uute ja tundmatute vahendite õppimise tõttu
Finantseerimine	• Ressursside kokkuhoid ja vähesem raiskamine	• Suur algne investeering
Planeerimine	• Vähendab ebavajalike tööde ja projektide arvu • Aitab säästa aega	

Aeglane ning ebakvaliteetne tootearendus ja juurutamine asenduks palju tõhusama süsteemiga, mis võiks tuua paremaid muutusi ja uuendusi klientideni kiiremini vähesemate tõrgetega. Paljud sammud praeguses töökultuuris võiksid olla DevOps'iga automatiseeritud. Algselt see kindlasti tõstaks töötajate koormust, sest vastavad süsteemid on vaja üles seada. Hiljem aga lihtsustaks tööd oluliselt, sest paljud ajakulukad tegevused oleksid edaspidi automatiseeritud. Kõige lõpetuseks ja võibolla tähtsaima asjaoluna võiks DevOps tõsta inimeste rahulolu ning usku olemasolevasse töökultuuri. Vead käituses ja arenduses ning ka igas muus osas IT-teenistuses tekitavad lisastressi ja pingeid. Uus töökultuur võiks inimestele pakkuda paremat ja rahulikumat keskkonda, kus oma karjääri nimel vaeva näha.

3.3 DevOps'i integreerimine IT-teenistusse

Käesolevas alampeatükis loob autor potentsiaalse plaani PERH-i IT-teenistuse jaoks, mille abil oleks võimalik soovi korral edukalt integreerida ja rakendada DevOps'i. Selle tegemisel toetub autor varasemalt loodud analüüsidele ning läbi töötatud teooriale esimeses ja teises peatükis.

Esimene samm - arutelu

Esimese asjana peaks kokku koguma nii arendajad, projektijuhid, süsteemiadministraatorid kui ka kõik ülejäänud töötajad, kes on kaasatud vastavasse töökeskkonda, ning rääkima läbi DevOps'iga kaasnevad muutused ja nõuded. Varasemalt juba käsitletud punktina, DevOps'i

puhul on väga oluline kõigi töötajate panus ja ühtsus, sest muul viisil ei ole võimalik edukat kultuurilist muutust läbi viia. Seega tuleks korraldada suur küsitlus või näiteks koosolek, kus sellist üleminekut oleks võimalik arutada ja omavahel mõtteid jagada. Juhul kui tagasiside on positiivne ja kõik on valmis pingutama muutuste ja arengu nimel, on võimalik liikuda edasi.

Teine samm – muutuste ulatuse määratlemine

Järgmise sammuna tuleks otsustada, missuguses ulatuses DevOps'i rakendada. Kõige olulisemateks aspektideks on pidev integratsioon ja pidev tarne. Ilma nende kahe metoodikata ei ole võimalik DevOps'i rakendada ja nende tavade edukaks toimimiseks on vaja praeguse süsteemiga võrreldes sisse viia palju muutusi. Näiteks on vaja suur osa tarnest automatiseerida ja arendajad peaksid harjuma oma töid tihemini lõimima. Samas on lisaks väga palju muid tavasid, mille abil on võimalik DevOps'i rakendada palju efektiivsemalt ja edukamalt. Nendeks olid varasemalt käsitletud infrastruktuur koodina, pidev testimine, pidev seire, pidev tagasiside ja pidev äriplaneering. Vähemalt üks neist tavadest on juba IT-teenistuses kasutusel – hetkel teostatakse pidevat seiret Zabbix'i abil. Seega on vähemalt üks abistav tava juba kasutamisel, mis kindlasti mingil määral lihtsustab üleminekut uuele metoodikale. Lisaks sellele tuleb ühiselt paika panna ka muud tavad, mida kavatsetakse kasutama hakata. Autori soovitusena oleks PERH-is vaja kindlasti kasutusele võtta ka ülejäänud abistavad tavad – pideva tagasiside ja testimise, infrastruktuuri koodina ning pideva äriplaneeringu. Hetkelise tagasiside puudulikkuse tõttu on väga keeruline arendada edasi omavahelist suhtlust, mis on DevOps'i puhul väga oluline. Testimine ja koodipõhine infrastruktuur aitavad vältida vigu ja tõrkeid produktsioonikeskkonnas, mis suurendab kindlasti efektiivsust ja vähendab ressursside raiskamist. Näiteks pideva testimise kasutuselevõtt ja automatiseerimine vähendab inimlike eksimuste võimalust ning muudab uute teenuste ja toodete tarnet kiiremaks. Pidev äriplaneering aitaks vältida ebavajalikke projekte ja arendusi.

Kolmas samm – uued vahendid ja koolitused

Pärast DevOps'i kasutuselevõtu ulatuse määratlemist tuleks alustada muudatuste tegemisega. Alustada tuleks sellest, et välja peab valima edaspidi kasutatavad vahendid. GitKrakeni uuringus [17] tuuakse välja reaalselt DevOps'i rakendavate töötajate tarkvaralised eelistused – versioonihalduseks Git, koodi kirjutamiseks näiteks IntelliJ või Visual Studio Code, testimiseks JUnit ja Selenium, suhtluseks Microsoft Teams ja Slack, käituseks Docker ja monitoorimiseks Grafana, tarne automatiseerimiseks Jenkins. Kindlam vahendite valik tuleks asutusel endal teha, sest see sõltub nii inimeste isiklikest eelistustest kui ka töö arhitektuurist. Võimalikult

kiireks ümberõppimiseks oleks soovituslik korraldada koolitusi vastavate vahendite edukaks omandamiseks. Sellisel viisil on võimalik muuta töötajate jaoks võõrad tööriistad tuttavamaks ja nad õpivad ühiselt ühte moodi kõike kasutama. Seega ei tohiks probleeme tekkida ka sellega, et ühte tööd tegevad inimesed teevad seda erinevalt, mis võiks omakorda tekitada suures plaanis probleeme ja näiteks ühilduvusvigasid.

Neljas samm – tavadest kinni pidamine ja harjumine

Uus metoodika põhjustab algselt tõenäoliselt vähesel või rohkemal määral probleeme. Kogu töökorralduse tõhusus arvatavasti langeb ning võib tunduda, et DevOps oli vale samm. Selline langus efektiivsuses on tavaline nähtus muutuste puhul. Uuringud on näidanud, et ettevõtted, mis on DevOps'i kauem rakendanud, on selles ka edukamad [7]. See tähendab, et neljanda ja viimase pika sammuna on vaja järjepidevust ja koostööd. GitKraken on välja toonud, et igas DevOps'ile ülemineku etapis on oluline keskenduda pidevale arenemisele – tuleb leida tööd tagasihoidvad piirangud, need eemaldada ja seejärel seda sammu jälle korrata [17]. Konstruktiivne kriitika võib õppimise käigus väga palju juurde anda, mistõttu on tagasiside selles sammus väga oluline. Näiteks motiveerib positiivne tagasiside hästi tehtud töö eest inimesi ka edaspidi vaeva nägema.

Lisasamm

Autori ettepanekud põhinevad puhtalt varem käsitletud materjalidel ja analüüsidel. Võimalikult edukaks DevOps'i rakendamiseks PERH-is oleks autori arvates lisaks kõigele muule vajalik ka vastava metoodika spetsialisti palkamine. DevOps on muutumas aina populaarsemaks ja seetõttu koolitatakse sel alal välja ka üha rohkem spetsialiste. Sellise töötaja professionaalse analüüsi ja kava põhjal on ettevõttesisene üleminek uuele metoodikale kindlasti sujuvam ja personaalsem.

Kokkuvõte

Käesoleva lõputöö eesmärgiks oli analüüsida, kas DevOps võiks olla tarkvaraarendusmeetod, millega saaks Põhja-Eesti Regionaalhaigla IT-teenistuse tööd muuta tõhusamaks. Uue töökultuuri abiga edeneks tootearenduse ja -tarne kiirus ja kvaliteet ning väheneks üleüldine ressursside raiskamine. Hetkeseisu analüüsimiseks koostati küsitlused projektijuhtidele, arendajatele ja süsteemiadministraatoritele. Nende tulemusena jõuti otsusele, et praegune süsteem on vananenud ja sellega kaasnevalt esineb probleeme nii arenduses, käituses kui ka projektide juhtimises. Samas leiti ka positiivseid külgi praeguses süsteemis, mis on DevOps'i tavadega kooskõlas ning võiksid olla abiks olemasoleva kultuuri asendamises.

Toetudes esimestes peatükkides läbitöötatud materjalidele ning praegusele töökorrale, leiti, et DevOps võiks olla lahenduseks paljudele esinenud probleemidele. DevOps'i tavade abil võiks muuta toodete arendust kiiremaks ja kvaliteetsemaks, sest arendajad lõimiksid omavahel töid tihemini ja seeläbi ennetaksid tõrkeid varajastes arendusfaasides. Käitus tuleks DevOps'i tavade järgi rohkem automatiseerida ja seeläbi muutuks toodete tarne ning juurutamine kiiremaks ja väheneks inimlike vigade võimalus. Pidev äriplaneerimine aitaks vähendada ebavajalike projektide arvu ning seeläbi säästa nii aega kui ka raha. Lisaks aitaks DevOps'i kultuur parandada ka töötajate üldist rahulolu ja arendada edasi nende oskusi uute vahendite kasutamisel.

Lõpetuseks koostati plaan, mille abil oleks võimalik DevOps'i PERH-i IT-teenistusse integreerida. Neljaosalises plaanis toodi sammhaaval välja võimalikult üksikasjalikult, kuidas oleks võimalik edukalt DevOps'i rakendada ja olemasolevat töökultuuri asendada. Plaan koostati toetudes läbitöötatud teoreetilistele materjalidele, varasematele uuringutele, kus analüüsiti reaalselt seda metoodikat kasutavaid asutusi ning DevOps'i kasutamise eelistele IT-teenistuses. Plaani eesmärgiks oli anda ülevaade võimalikest etappidest, mis tuleks uue metoodika integreerimise jaoks kindlasti läbida.

Käesoleva töö edasiarendusena võiks viia läbi veelgi üksikasjalikuma küsitluse ja analüüsi IT-teenistuses, kaasates sellesse rohkem töötajaid ja kogudes täpsemat informatsiooni. Näiteks tuleks uurida asutuse rahalisi võimalusi, kasutusele võetavate erinevate tarkvarade ühilduvust ja võimalikke muutusi töötajate rollides. Selle abil saaks luua konkreetsema plaani, mille järgimisel võiks DevOps'i PERH-is edukalt rakendada. Käesolevas töös DevOps'i integreerimist IT-teenistuses läbi ei viidud.

Kasutatud kirjandus

[1] Sharma S., Coyne B. (2015) DevOps for Dummies®, 2nd IBM Limited Edition. Hoboken, NJ: John Wiley & Sons, Inc.

[2] Kim G., Humble J., Debois P., Willis J. (2016) The DevOps Handbook: How to Create World-Class Agility, Reliability, & Security in Technology Organizations. Portland, OR: IT Revolution Press, LLC.

[3] Sharma S. (2017) The DevOps Adoption Playbook: A Guide to Adopting DevOps in a Multi-Speed IT Enterprise. Indianapolis, IN: John Wiley & Sons, Inc.

[4] Díaz J., Perez J.E., Yague A., Villegas A., de Antona A. (2019) DevOps in Practice – A Preliminary Analysis of Two Multinational Companies. In: Franch X., Männistö T., Martínez-Fernández S. (eds) Product-Focused Software Process Improvement. PROFES 2019. Lecture Notes in Computer Science, vol 11915. Springer, Cham. https://doi.org/10.1007/978-3-030-35333-9_23

[5] Senapathi M., Buchan J., Osman H. (2019) DevOps Capabilities, Practices, and Challenges: Insights from a Case Study. https://doi.org/10.1145/3210459.3210465

[6] Velasquez N.F., Kim G., Kersten N., Humble J. (2014). 2014 State of DevOps Report. Vaadatud 09.03.2021 https://services.google.com/fh/files/misc/state-of-devops-2014.pdf

[7] Puppet Labs (2015). 2015 State of DevOps Report. Vaadatud 09.03.2021 https://services.google.com/fh/files/misc/state-of-devops-2015.pdf

[8] Forsgren N., Smith D., Humble J., Frazelle J. (2019). State of DevOps 2019. Vaadatud 01.04.2021 https://services.google.com/fh/files/misc/state-of-devops-2019.pdf

[9] Lipnitski A. (2019). DevOps Implementation: Where to Start And How to Make It Result in Success? Vaadatud 10.03.2021 https://www.scnsoft.com/blog/devops-implementation-guide

[10] Mezak S. (2018). The Origins of DevOps: What's in a Name? Vaadatud 11.03.2021 https://devops.com/the-origins-of-devops-whats-in-a-name/

[11] Hand J. (2015). The Top 10 DevOps myths debunked. Vaadatud 16.03.2021 https://techbeacon.com/devops/top-10-devops-myths-debunked

[12] Singh G. (2020). A Quick Guide to Continuous Integration and Continuous Delivery. Vaadatud 18.03.2021 https://www.xenonstack.com/blog/continuous-integration-and-continuous-delivery/

[13] Danicic D. (2020). What is DevOps Pipeline & How to Build One. Vaadatud 18.03.2021 https://phoenixnap.com/blog/devops-pipeline

[14] 6 Companies That Are Doing DevOps Well. (11.01.2018). Vaadatud 20.04.2021 https://www.helpsystems.com/blog/6-companies-are-doing-devops-well

[15] Anil N., Ghosh S., Veloso M., Schonning N., Parente J., Wenzel M. (2020). Containers as the foundation for DevOps collaboration. Vaadatud 21.04.2021 https://docs.microsoft.com/en-us/dotnet/architecture/containerized-lifecycle/docker-application-lifecycle/containers-foundation-for-devops-collaboration

[16] Modi A. (2020) Tools That Will Boost Your DevOps Journey. Vaadatud 28.04.2021 https://aws.plainenglish.io/learning-this-tools-will-boost-your-journey-into-devops-bb5dd7a854ee

[17] GitKraken (2020) DevOps Tools Report 2020. Vaadatud 29.04.2021 https://www.gitkraken.com/reports/devops-report-2020

[18] Vene S. (2017) *DevOps* juurutamine suurettevõttes. Tallinna Tehnikaülikooli infotehnoloogia teaduskond. [Magistritöö]. Tallinn.

[19] Rush L. (2016) DevOps printsiipide rakendamise analüüs TransferWise'i näitel. Tallinna Tehnikaülikooli infotehnoloogia teaduskond. [Bakalaureusetöö]. Tallinn.

[20] Galbreath N. (2012). Continuous Deployment – The New #1 Security Feature, from BSlidesLA 2012. Vaadatud 04.05.2021 https://www.slideshare.net/nickgsuperstar/continuous-deployment-the-new-1-security-feature

Lisad

I. Küsitlused

Administraator

Palun kirjeldage PERH-i IT-teenistuse hetkelist töökorda ja vastake igale küsimusele nii põhjalikult kui võimalik:

1. **Andke hinnang Põhja-Eesti Regionaalhaigla IT-teenistuse üldisele toimimisele ja efektiivsusele.**
    a. **Skaalal 1-10; 1 – väga ebaefektiivne ja halb; 10 – nii efektiivne ja hea kui võimalik**

Vastus:

2. **Kuidas olete rahul hetkelise töökultuuriga?**
    a. **Skaalal 1-10; 1 – pole üldse rahul; 10 – väga rahul**

Vastus:

3. **Nimetage kõige tähtsamaid omadusi, mis peavad tiimisiseselt eksisteerima, et antud tiim töötaks hästi ja efektiivselt.**

Vastus:

4. **Kas ja millisel kujul esineb praeguses töökultuuris ressursside raiskamist?**

Vastus:

5. **Milline on suhtlus nii arendajate, administraatorite kui ka projektijuhtide vahel? (Näiteks kui tihe, avatud ja abivalmis)**

Vastus:

6. **Milliseid vahendeid kasutavad tiimid omavaheliseks suhtluseks?**

Vastus:

7. **Kas omavahelist suhtlust erinevate sidusgruppide (näiteks administraatorite ja arendajate) vahel peaks parandama?**

Vastus:

8. **Milliseid vahendeid kasutate hetkel süsteemide monitoorimiseks?**

Vastus:

9. **Kuidas on teostatud pidev seire ehk missugusel viisil on teostatud süsteemide pidev haldus ja monitooring?**

Vastus:

10. **Kui kaua võtab aega uue toote / teenuse tarnimine arendusest kliendini?**

Vastus:

11. **Kui tihti vajab tarnitud teenus pärast produktsioonikeskkonda integreerimist muudatusi või parandusi?**

Vastus:

12. **Kui palju teste viiakse enne uue toote / teenuse tarnimist läbi? Kuidas seda tehakse? Millal neid teste läbi viiakse?**

Vastus:

13. **Kas testimiseks mõeldud keskkondasid on võimalik automaatselt juurde luua spetsiifiliste testide läbi viimiseks?**

Vastus:

14. **Kui kaua võtab keskmiselt aega vigade või rikete likvideerimine? (ühilduvusvead süsteemis, vigane kood jne)**

Vastus:

15. **Kui suur osa mingi toote / teenuse väljumisest arendusest kuni produktsioonikeskkonda tarnimiseni on automatiseeritud?**

Vastus:

16. **Kui osa tarnimisest on automatiseeritud, siis kuidas see teostatud on?**

Vastus:

17. **Kas rohkem automatiseerimist võiks uute toodete / teenuste kvaliteeti parandada ja tarnimist kiirendada?**

Vastus:

18. **Kui palju, kellelt ja mis kujul saate tagasisidet oma tehtud töö kohta?**

Vastus:

19. **Kas ja millisel viisil esineb tarnimise etappides ebavajalikke protsesse või ümbertöötamise vajadust?**

Vastus:

20. Kas ja millisel viisil mõõdetakse hetkel töö efektiivsust?

Vastus:

Palun vastake järgmistele küsimustele:

1. **Kas olete tuttavad DevOps'i kultuuriga?**

   Vastus:

2. **Kas pooldate DevOps'i rakendamist PERH-is?**

   Vastus:

3. **Kuidas suhtuksite töökultuuri ümber korraldamisse, mis võiks tuua muutusi Teie töösse näiteks kasutatava tarkvara või ajaliste nõuete kujul?**

   Vastus:

4. **Kas oleksite nõus õppima kasutama uusi vahendeid, et muuta töö efektiivsemaks?**

   Vastus

5. **Kas ja kuidas võiks teistsugune töökultuur olla kasulik Teie karjäärile?**

   Vastus:

Arendaja

Palun kirjeldage PERH-i IT-teenistuse hetkelist töökorda ja vastake igale küsimusele nii põhjalikult kui võimalik:

1. **Andke hinnang Põhja-Eesti Regionaalhaigla IT-teenistuse üldisele toimimisele ja efektiivsusele.**
    a. **Skaalal 1-10; 1 – väga ebaefektiivne ja halb; 10 – nii efektiivne ja hea kui võimalik**

Vastus:

2. **Kuidas olete rahul hetkelise töökultuuriga?**
    a. **Skaalal 1-10; 1 – pole üldse rahul; 10 – väga rahul**

Vastus:

3. **Nimetage kõige tähtsamaid omadusi, mis peavad tiimisiseselt eksisteerima, et antud tiim töötaks hästi ja efektiivselt.**

Vastus:

4. **Kas ja millisel kujul esineb praeguses töökultuuris ressursside raiskamist?**

Vastus:

5. **Milline on suhtlus nii arendajate, administraatorite kui ka projektijuhtide vahel? (Näiteks kui tihe, avatud ja abivalmis)**

Vastus:

6. **Milliseid vahendeid kasutavad tiimid omavaheliseks suhtluseks?**

Vastus:

7. **Kas omavahelist suhtlust erinevate sidusgruppide (näiteks administraatorite ja arendajate) vahel peaks parandama?**

Vastus:

8. **Milliseid vahendeid kasutate hetkel tarkvaraarenduseks?**

Vastus:

9. **Milliseid vahendeid kasutate hetkel versioonihalduseks?**

Vastus:

10. **Kui tihti lõimivad erinevate osade kallal töötavad arendajad oma parajasti valmisolevaid osi uue toote / teenuse arendamise juures?**

Vastus:

11. **Kui suur hulk tarnitud teenustest vajab pärast produktsioonikeskkonda integreerimist muudatusi või parandusi?**

Vastus:

12. **Kui palju teste viiakse uue toote / teenuse arendamisel läbi? Kuidas seda tehakse? Millal neid teste läbi viiakse?**

Vastus:

13. **Kui kaua võtab keskmiselt aega vigade või rikete likvideerimine? (vead koodis / ühilduvusvead süsteemis jne)**

Vastus:

14. **Kui suur osa toote / teenuse arendamisest kuni tarneni on automatiseeritud?**

Vastus:

15. **Kas rohkem automatiseerimist võiks uute toodete / teenuste kvaliteeti parandada ja arendust kiirendada?**

Vastus:

16. **Kui tihti või kiiresti valmivad uued tarnitavad tooted / teenused?**

Vastus:

17. **Kui palju, kellelt ja mis kujul saate tagasisidet oma tehtud töö kohta?**

Vastus:

18. **Kas ja millisel viisil esineb arendusetappides ebavajalikke protsesse või ümbertöötamise vajadust?**

Vastus:

19. **Kas ja millisel viisil mõõdetakse hetkel töö efektiivsust?**

Vastus:

Palun vastake järgmistele küsimustele:

1. **Kas olete tuttavad DevOps'i kultuuriga?**

Vastus:

2. **Kas pooldate DevOps'i rakendamist PERH-is?**

Vastus:

3. **Kuidas suhtuksite töökultuuri ümber korraldamisse, mis võiks tuua muutusi Teie töösse näiteks kasutatava tarkvara või ajaliste nõuete kujul?**

Vastus:

4. **Kas oleksite nõus õppima kasutama uusi vahendeid, et muuta töö efektiivsemaks?**

Vastus

5. **Kas ja kuidas võiks teistsugune töökultuur olla kasulik Teie karjäärile?**

Vastus:

Projektijuht

Palun kirjeldage PERH-i IT-teenistuse hetkelist töökorda ja vastake igale küsimusele nii põhjalikult kui võimalik:

1. **Andke hinnang Põhja-Eesti Regionaalhaigla IT-teenistuse üldisele toimimisele ja efektiivsusele.**
    a. **Skaalal 1-10; 1 – väga ebaefektiivne ja halb; 10 – nii efektiivne ja hea kui võimalik**

Vastus:

2. **Kuidas olete rahul hetkelise töökultuuriga?**
    a. **Skaalal 1-10; 1 – pole üldse rahul; 10 – väga rahul**

Vastus:

3. **Nimetage kõige tähtsamaid omadusi, mis peavad tiimisiseselt eksisteerima, et antud tiim töötaks hästi ja efektiivselt.**

Vastus:

4. **Kas ja millisel kujul esineb praeguses töökultuuris ressursside raiskamist?**

Vastus:

5. **Milline on suhtlus nii arendajate, administraatorite kui ka projektijuhtide vahel? (Näiteks kui tihe, avatud ja abivalmis)**

Vastus:

6. **Milliseid vahendeid kasutavad tiimid omavaheliseks suhtluseks?**

Vastus:

7. **Kas omavahelist suhtlust erinevate sidusgruppide (näiteks projektijuhtide ja arendajate) vahel peaks parandama?**

Vastus:

8. **Millised sammud läbitakse uue toote / teenuse idee loomisel?**

Vastus:

9. **Kui tihti tuleb uue toote / teenuse puhul peale arenduse alustamist ette muutusi, mille tõttu tuleb olemasolevat tööd ümber korraldada?**

Vastus:

10. Kes vastutab toote / teenuse praktilisuse ja õigeaegse valmimise eest?

Vastus:

11. Kas toote / teenuse tootmisahelas esineb ressursside raiskamist?

Vastus:

12. Kuidas oleks võimalik tootmisahelat efektiivsemaks muuta?

Vastus:

13. Kuidas saavad uued teenused / tooted tagasisidet?

Vastus:

14. Kui tähtis on toote / teenuse planeerimise ja uuendamise juures kliendi tagasiside?

Vastus:

15. Mille põhjal võib määratleda loodud projekti edukaks?

Vastus:

16. Kui kiiresti keskmiselt valmivad uued projektid?

Vastus:

17. Kas ja millisel viisil mõõdetakse hetkel töö efektiivsust?

Vastus:

Palun vastake järgmistele küsimustele:

1. **Kas olete tuttavad DevOps'i kultuuriga?**

Vastus:

2. **Kas pooldate DevOps'i rakendamist PERH-is?**

Vastus:

3. **Kuidas suhtuksite töökultuuri ümber korraldamisse, mis võiks tuua muutusi Teie töösse näiteks kasutatava tarkvara või ajaliste nõuete kujul?**

Vastus:

4. **Kas oleksite nõus õppima kasutama uusi vahendeid, et muuta töö efektiivsemaks?**

Vastus

5. Kas ja kuidas võiks teistsugune töökultuur olla kasulik Teie karjäärile?

Vastus:

II. Litsents

Lihtlitsents lõputöö reprodutseerimiseks ja üldsusele kättesaadavaks tegemiseks

Mina, Siim-Morten Ojasalu,

(autori nimi)

annan Tartu Ülikoolile tasuta loa (lihtlitsentsi) minu loodud teose

DevOps ja selle rakendatavuse uuring Põhja-Eesti Regionaalhaiglas,

(lõputöö pealkiri)

mille juhendaja on Pelle Jakovits,

(juhendaja nimi)

reprodutseerimiseks eesmärgiga seda säilitada, sealhulgas lisada digitaalarhiivi DSpace kuni autoriõiguse kehtivuse lõppemiseni.

Annan Tartu Ülikoolile loa teha punktis 1 nimetatud teos üldsusele kättesaadavaks Tartu Ülikooli veebikeskkonna, sealhulgas digitaalarhiivi DSpace kaudu Creative Commonsi litsentsiga CC BY NC ND 3.0, mis lubab autorile viidates teost reprodutseerida, levitada ja üldsusele suunata ning keelab luua tuletatud teost ja kasutada teost ärieesmärgil, kuni autoriõiguse kehtivuse lõppemiseni.

Olen teadlik, et punktides 1 ja 2 nimetatud õigused jäävad alles ka autorile.

Kinnitan, et lihtlitsentsi andmisega ei riku ma teiste isikute intellektuaalomandi ega isikuandmete kaitse õigusaktidest tulenevaid õigusi.

Siim-Morten Ojasalu

07.05.2021