TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond

Eliise Juhansoo 164030IABB

TARKVARAARENDUSE PROJEKTISISENE PROTSESSIPARENDUS TARKVARAARENDUSETTEVÕTTE NÄITEL

Bakalaureusetöö

Juhendaja: Maili Markvardt
MSc

Tallinn 2019

Autorideklaratsioon

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

Autor: Eliise Juhansoo

20.05.2019

Annotatsioon

Käesoleva bakalaureusetöö eesmärk on parendada tarkvaraarenduse projektisisest töökorraldust, kuid esmalt kaardistada sellega seotud probleemid, mis kogu meeskonnale muret teevad. Protsessiparendus viiakse läbi Gitlab *boardi* abil, mis on virtuaalne vahend tarkvaraarenduslike meeskondade töö korraldamiseks nii kontoris kui eemal.

Esmalt tutvustatakse lähemalt projekti ja selle eripärasid ning Kanban *boardi*, mille põhimõtetele vastab ka Gitlabi *board*. Tööst ei puudu ka Gitlab *boardi* erinevate osade ja funktsioonide tutvustus ning nende kasulikkuse analüüs.

Praktiline osa näeb ette küsitluse läbi viimist kogu lõputööga seotud ettevõtte ühes valdkonnad, mille alla kuulub ka uuritav projekt. Peale seda küsitluse põhjal projektis tehtud muudatused ning nende kasulikkuse analüüs.

Töö praktilist osa ehk projektis sisse viidud muudatusi võib pidada positiivseks, sest nende põhjal said lahendatud kõik probleemid, mida oli võimalik Gitlab *boardiga* lahendada. Samuti sai lahendatud põhiline eesmärk vähendada *stand-upidele* kuluvat aega.

Lõputöö on kirjutatud eesti keeles ning sisaldab teksti 42 leheküljel, 5 peatükki, 7 joonist, 1 tabel.

Abstract

In-house Process Improvement of Software Development Based on the Example of Software Developing Company

The goal of this bachelor thesis is to improve the software development process by examining the problems in the software developing project and finding solutions to them. To get to know the in-house problems, a survey was taken in company X. Based on the results of the survey, many of the problems can be solved by using the Gitlab board application. Gitlab board was chosen because it has already been used in different projects in company X.

Firstly, an overview is given of Kanban project's leading method and its main part - Kanban board. A more thorough description is given about the Gitlab company, the Gitlab board, and its main functions such as lists, issues, labels, milestones, assignees and merge requests. There is also a short overview of the Jira application, which is used for communicating with the client.

Secondly, in the practical part, the survey and its results, which are the problems connected with in-house workflow, are described in detail. That is followed by the description of the changes made in the Gitlab board to improve the in-house process and the results caused by these changes.

Overall, the result was positive because half of the members of the project were involved with the changes and started using them. A solution was found to all the problems that could be solved with the Gitlab board. Also, the main goal was to reduce the time that stand-up meetings take. After the changes, the meetings were a lot shorter, therefore, that goal was achieved.

The thesis is in Estonian language and contains 42 pages of text, 5 chapters, 7 figures, 1 table.

Lühendite ja mõistete sõnastik

Gitlab	Tarkvaraarendusega tegelev ettevõte [1].
Board	Gitlab ettevõtte toode tarkvaraarenduslike protsesside jälgimiseks [2].
Kanban	Töökorralduse juhtimise vahend, meetod [3].
Jira	Ülesannete jälgimise seade, projektijuhtimise vahend [4].
Stand-up	Lühike koosolek, kus osalejad seisavad püsti [5].
Task	Ülesanne, töö [6].
Issue	Ülesanded ja ideed, mida märgitakse Gitlab *boardile* [7].
Front end	„Veebilehele ilmuv kasutajaliides, mis võimaldab veebisaidi külastajal kahepoolselt suhelda saidi dünaamiliste osadega nagu andmebaasid, ostukorviprogrammid ja online-ostutöötlustarkvara“[8].
Back end	Tagarakendus, mille ülesandeks on kaudselt toetada eesteenuseid [8].
Agiilne arendus	Tarkvaraarenduse juhtimise tehnika.
Backlog	Tarkvaraarendamise nõuete loetelu [9].
Closed	Suletud, kinni, lõpetatud [6].
List	Nimekiri, loend [6]. Antud kontekstis *boardil* olevad tulbad või veerud, mille alla kirjeldatakse *issuesid* [2].
Label	*Issuede* külge lisatavad värvilised märked [10].
Milestone	Meetod või funktsioon *issuede* ja *merge requestide* grupeerimiseks [11].
Assignee	Funktsioon, millega saab ülesandeid määrata kindlale isikule [2].
Merge request	Taotlused, mis võimaldavad muuta lähtekoodi mitmel inimesel samal ajal erinevates harudes [12].
UI	*User Interface* ehk kasutajaliides [8].
API	*Application Programming Interface* ehk rakendusliides, programmiliides [8].

Sisukord

Jooniste loetelu

Tabelite loetelu

1 Sissejuhatus

Tänapäeval kiiresti arenev tehnoloogiavaldkond on tekitanud turule väga palju erinevaid infotehnoloogilisi ettevõtteid, kes kõik rohkemal või vähemal määral pakuvad oma klientidele kõige uuemaid tehnoloogilisi lahendusi. Ühes ettevõttes võidakse tegeleda mitmete erinevate klientide tellimuste täitmisega ning see nõuab väga head töökorraldust.

Ka ettevõttes X tegeletakse igapäevaselt mitmete erineva valdkonna rakenduste arendamisega ning selle juurde kuuluvate tegevustega, millest üks on kogu projekti juhtimine. Antud lõputöö eesmärk on uurida, kas ja kuidas on võimalik projektisisest töökorraldust parendada ning kogu protsessi edukamalt juhtida. Seda aga kasutades projektisiseseks töökorralduseks mõeldud rakendust nimega Gitlab, mis põhineb Kanban *boardi* põhimõtetel.

1.1 Taust

Ettevõtte X üheks suurimaks kliendiks on ettevõte Y. Lõputöösse valiti antud asutuse projektid seetõttu, et töö autor on ise nendega väga tihedalt seotud. Nimelt on ta kaasa löönud kahes suuremas projektis nimedega A ja B, millest esimeses viiakse läbi lõputööga seotud tegevused projektisiseseks protsessiparenduseks. Kõikides ettevõtte Y projektides kasutatakse töökorralduseks peamiselt kahte selleks loodud rakendust.

Esimene neist kannab nime Jira, mis on mõeldud kliendiga suhtlemiseks. See on keskkond, kuhu klient raporteerib nii rakendustest leitud vigu, uuendatud nõudeid kui ka testimise käigus tekkinud soove. Rakendus võimaldab mugavalt kõiki eelmainitud aspekte ühte kohta kirja panna ning neid vastavalt vajadusele sorteerida. Lisaks on kogu tegevus kõikidele osapooltele nähtav ehk kõigil on seal leiduvast informatsioonist võimalik osa saada.

Teine kannab nime Gitlab, mis oma võimaluste poolest on küll väga lai, kuid antud projektis kasutatakse sealt ainult *board* funktsiooni sisemise töökorralduse haldamiseks. Gitlabi *board* põhineb Kanban *boardil*, millest tuleb täpsemalt juttu juba järgmises

peatükis. Gitlabi *board* annab väga hea ülevaate projekti üldisest seisust ning ka sellest, millega projektiliikmed täpsemalt tegelevad. Seda aga sellisel juhul, kui kõik asjaosalised ühistest kokkulepetest kinni peavad ning *boardi* kaasajas hoiavad.

Selleks, et saada aru, mis täpsemalt projektisisest protsessi sujuvust takistab, viidi kogu valdkonnas läbi uuring, milles selgusidki põhilised murekohad. Uuring oli üles ehitatud Google Forms rakenduse abil, kus erinevad inimesed said anonüümselt vastata eelnevalt uuringu läbiviija poolt ettevalmistatud küsimustele, mille vastuste analüüsist selgusid täpsemad muudatust vajavad kohad. Lõputöö on üles ehitatud nende küsimuste vastustest selgunud probleemide lahendamiseks Gitlab *board* rakenduse toel.

Projekt A näeb ette ettevõtte Y kõikide rakenduste üleviimist uude keskkonda ning selle raames valmib ka uus töölaud, mida on nii kliendil kui ametnikul mugavam kasutada. Selleks, et kõik rakendused saaksid vigadeta uude keskkonda üle viidud tuleb need üle testida ning vigade avastamisel ka ära parandada. Projektiga on seotud 22 inimest, kellest 16 tegelevad igapäevaselt teemaga ning keda lõputööga seonduv konkreetselt mõjutama hakkab. Ülejäänud kuus inimest on projekti kaasatud seetõttu, et nad on mõne rakendusega rohkem kursis ning mõni antud projekti rakendus on nende inimeste rakendustega seotud. Need kuus inimest oskavad kõige kiiremini anda projekti probleemidele, mis on seotud teiste projektidega, lahendusi.

Bakalaureusetöö autor usub, et antud töö käigus loodava Gitlab *boardi* kasutamine tuleb kasuks kõikidele tarkvaraarendusega seotud tiimidele olenemata nende suurusest. Mida suurem on inimeste arv, seda vajalikum on *board*, sest kui kogu info on ühest kohast leitav ja kaasajane, siis hoitakse kõigi liikmete aega kokku.

1.2 Probleem

Juba enne lõputöö kirjutama hakkamist pani autor tähele mõningaid probleeme, mis põhinesid sel ajal veel ainult isiklikul kogemusel, kuid millele lahenduse leidmine oleks kõigi töö tegemist palju mugavamaks ja kiiremaks muutnud. Koosolekutel ja *stand-upidel* kulus põhiline aeg selleks, et saada aru kes millega tegeleb ning mis seisus projekt üldiselt on. Selline informatsioon on oluline põhiliselt ainult projektijuhtidele ning muud liikmed ootavad sellistelt koosviibimistelt olulisemat infot, mis tuleks töötegemisel ka kuidagi

kasuks. Lähemal uurimisel said oletused rohkem kinnitust ning tekkis mõte uurida teemat täpsemalt ning panna ka tulemused kirja.

Kuna töö autor ise on projektis A testija kohal, siis on tema jaoks väga oluline teada, kes millise ülesandega konkreetselt tegeleb ning kus maal on lahendus, et oleks selge, millal valminud arendusi testida. Selleks on loodud ja kasutusele võetud Gitlab *board*, kust on säärast informatsiooni väga lihtne järgi uurida, kuid kui meeskonnaliikmed ei hoia seda kaasajas ning ei uuenda ülesandeid ehk *taske* vastavalt, siis tekib palju mõttetut üleküsimist ning ootamist. Samuti kui inimesed ei kirjelda täpselt ülesande sisu, siis on jällegi vaja üle küsida, mida ta täpsemalt ülesannet püstitades saavutada soovis ning millist tulemust ta ootab. Kuna *boardile* võib projekti käigus tekkida isegi sada või rohkem avatud taski ehk *issuet*, siis on väga oluline ka neile õigeid märkeid külge panna. Selle tulemusena on konkreetse teema ülesanded paremini leitavad. Veel hiljuti pidasid äsja välja toodud reeglitest kinni väga vähesed ning sellest ajendatult tuli muudatused projekti tööprotsessi sisse viia.

1.3 Eesmärk

Bakalaureusetöö eesmärk on probleemist lähtuvalt panna inimesi kasutama rohkem Gitlab *boardi* rakendust. Tegemist on väga kasuliku ning vajaliku abivahendiga, aga ainult juhul, kui inimesed kokkulepitud töökorraldusest kinni peavad ning seda kaasajas hoiavad. Loomulikult peab olema ka *board* ise hästi üles ehitatud ning projektisiseseid vajadusi rahuldama.

Bakalaureusetöö eesmärk on lahendada järgnevaid probleeme Gitlab *boardi* rakenduse abil:

1. Ülesannete täitjad ei saa piisavalt ülesande täitmiseks vajaminevat infot ning nad peavad seda ülesande püstitajalt juurde küsima. Kui püstitaja paneks kohe kogu olulise info kirja, hoiaks see nende mõlema aega kokku.

2. Projektijuhid ei tea, kes millega täpselt tegeleb ning milline on projekti üldine seis. *Stand-up'idel* kulub kümneliikmelise tiimi puhul ligikaudu 20 minutit selleks, et teada saada, kes millega tegeleb, kuid see info on oluline ainult projektijuhile ning ülejäänud liikmed raiskavad sellega oma töötegemise aega. Mida rohkem inimesi, seda rohkem aega kulub.

3. Ei ole teada, kas keegi on mingisuguse teema juurde liiga kauaks püsima jäänud ning kas tal oleks sellega abi vaja.

4. Ei ole teada, mis seisus on *boardil* olevad ülesanded. Näiteks mõni inimene lahendab ülesande, kuid ei märgi seda lahendatuks ning teine hakkab sama otsast peale tegema. *Boardi* kaasajas hoidmine hoiaks jällegi liikmete aega kokku.

5. Projekti liikmed ei tegele *issuedega*, millel pole tegijat küljes ning märkeid „TODO“, „KIIRE“ ja muud.

6. Ülesandeid jagatakse erinevaid kanaleid pidi ning ei ole arusaadav, millised on tähtsamad. Lisaks võivad mõned ära ununeda.

7. Koodis muudetakse sama kohta, kuid teistele ei öelda ning hiljem *mergemisel* tuleb konflikt välja.

Põhiliseks eesmärgiks on muuta *boardi* olemust selliselt, et selle kasutamine tooks tiimile ja projektile üldiselt rohkem kasu ning parandaks ka töökorralduslikku protsessi.

1.4 Ülevaade tööst

Antud töö järgnevates peatükkides räägitakse kõikidest juba mainitud rakendustest ja nende võimalustest lähemalt. Tutvustatakse üldiselt ka tänapäeval väga populaarset tarkvaraarendusprojekti juhtimise tehnikat, milleks on agiilse arenduse juurde kuuluv Kanban.

Peale teoreetilist osa jõutakse töö põhiosani ehk uurimuse ja selle tulemuseni. Tuuakse välja täpsemad küsitluses selgunud asjaolud ning nende põhjal läbiviidud põhjaliku analüüsi tulemused. Peale seda juba konkreetses projektis vastavalt analüüsile läbiviidud muudatused protsessi parenduseks ning sealt edasi kogu töö tulemus nendest samadest muudatustest ajendatud tulemuste põhjal.

2 Metoodika

Käesolevas peatükis tutvustatakse kõiki bakalaureusetöös kasutatavaid metoodikaid. Tutvustatakse lühidalt ka ettevõtte X olemust. Põhiline osa läheb siiski töös kasutatavate teoreetiliste osade kirjeldamiseks ning nende põhimõtete tutvustamiseks.

2.1 Ettevõtte X lühitutvustus

Organisatsioon X tegeleb igapäevaselt tarkvaraarenduste loomise ja pakkumisega erinevatele ettevõtetele. Kliente võib leida nii avalikust kui ka erasektorist. Avalikus sektorist on Y ettevõtte, kellele on suunatud suurem osa ettevõtte X tööjõust. Erasektoris on põhilisteks klientideks erinevad pangad. Tarkvaralisi lahendusi pakutakse veel ka muudele ettevõttetele, mille osakaal ettevõttes X ei ole väga suur. Lisaks teenustele on ettevõte X lähiajal välja tulemas ka oma esmase tootega, mis on mõeldud peamiselt pankadele.

Tänasel päeval on ettevõtte töötajate arv juba natuke üle saja ning pidevalt värvatakse uusi inimesi juurde. Ühes tavalises tarkvaraarenduslikus projektis on ametid jagatud selliselt: arendajad, analüütikud, testijad ja projektijuht. Arendajaid saab jagada veel ka *front end* ja *back end* arendajateks. Projektid võib koondada klientide järgi valdkondadeks, mida juhivad valdkonnajuhid.

Ettevõttest ei puudu ka personaliosakond, mida nimetatakse *People Op's Teamiks*. Nemad sisustavad töövälist aega põnevate seminaride ning muude pidudega, kus tavaliselt pakutakse söögipoolist ning ei puudu ka muusikalised esinejad. Lisaks sellele tegelevad nemad ka uute töötajate värbamisega ning keskenduvad lisaks oskustele ka inimese sobivusega üldisesse kollektiivi.

2.2 Kanban

Kanban metoodika sai alguse juba rohkem kui viiskümmend aastat tagasi ühes autotootmisettevõttes nimega Toyota. Nad võtsid üle toidupoodides kasutatava meetodi,

mis aitas paremini hallata nende varusid ning hoida neid täpselt sellises koguses nagu tarbija vajab. Selleks jälgiti ostjate tarbimisharjumusi ning nii sai pidevalt jälgida, et vajalikud tooted oleksid laos olemas, kuid samas ei olnud laod jällegi üle ladustatud. Toyota hakkas oma tehastes kasutama sama meetodit, mis nägi välja selline, et tootmisosakondade tiimid jagasid omavahel kaarte kasutatud materjalidest ning kui midagi hakkas otsa saama, siis anti kaart edasi lattu, kes neid uute materjalidega varustas. Kui aga laost hakkasid varud otsa saama, siis saadeti kaart materjali tootjale, et oma varusid jällegi täiendada. Sellist käitumisviisi nimetatakse ka „just õigel ajal“ või inglise keeles „*just in time*“ (JIT) tootmisprotsessiks. [3]

Toyota meeskond hakkas ka ülesandeid ja probleeme vaatama täiesti uue nurga alt. Nimelt vaadati ülesandeid kui ühte tervikut ehk jälgiti seda kogu protsessi vältel algusest lõpuni. Jälgimine tähendas reaalset visuaalset vaatamist ehk probleemid pandi kirja kaartidele, mille liikumise põhjal sai jälgida selle seisu. Teiseks suunati suurem tähelepanu olulisematele probleemidele, mis tuli kiiresti või mingiks kindlaks ajaks ära teha. Selline käitumisviis muutis kogu protsessi kordades paremaks, sest probleemid ilmnesid kiiresti ning inventaari tegemine kahanes miinimumini. [13]

Kanban on tänapäeval kõige populaarsem agiilse arenduse alla kuuluv tehnika, mida kasutatakse paljudes tarkvaraarendusega tegelevates meeskondades. Kanban arvestab tiimi võimekusega, mis annab neile rohkem plaanimise painduvust ja valikuid, kiiremat väljundit, kindlamat fookust ehk sihti ning läbipaistvust kogu projekti vältel. Erinevalt tolleaegsetest konkreetsete asjade tootmise ettevõtetest, ei ole tarkvaarendusega seotud meeskondadel vaja nii palju füüsilisi produkte, kui siis ainult tahvlit ja kaarte, mis võivad tänapäeval olla samuti virtuaalsed. [3]

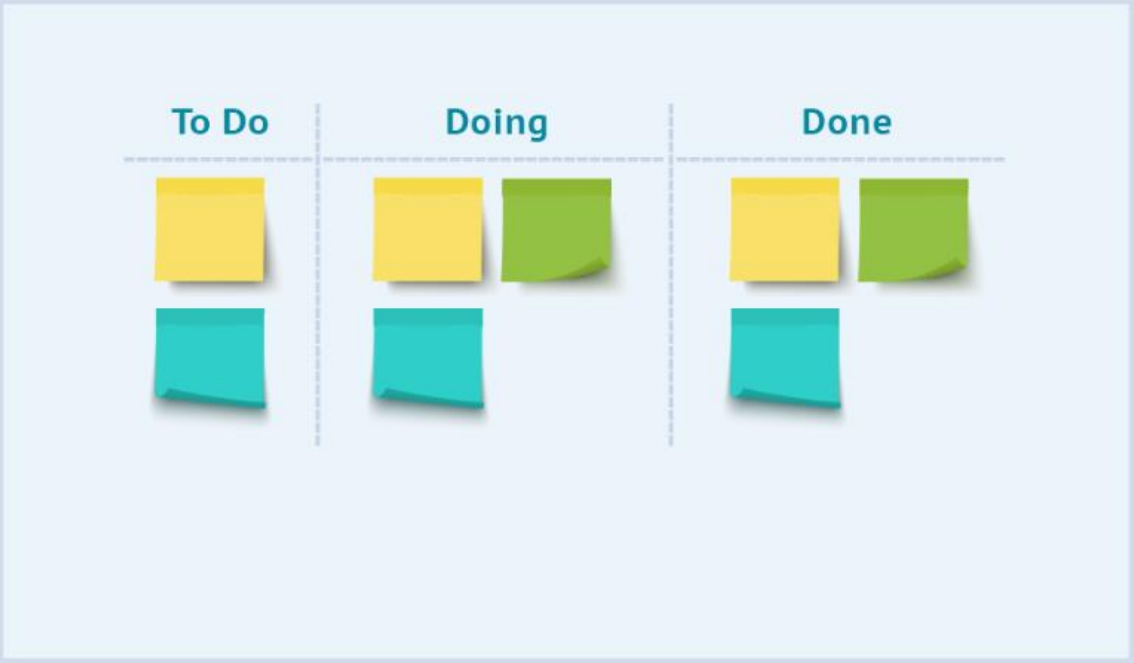
Lühidalt kokku võttes ongi kanban töökorralduse juhtimise vahend või meetod, mis aitab eesolevaid ülesandeid meeles pidada ning hetke probleeme kiiremini tuvastada [14]. Kanban meetodi põhiline tööriist on tahvel ehk *board*, mida kirjeldatakse täpsemalt juba järgmises alapeatükis, kuid põgusalt võib öelda, et see annab kogu töövoolust ja üldiselt projekti seisust väga hea ülevaate ning see hoiab kõigi asjaosaliste aega kokku.

2.2.1 Kanban board

Kanban *board* koosneb peamiselt kahest väga olulisest elemendist: veerud või tulbad ning *issued* ehk probleemid, mida nimetatakse ka *taskideks* ehk ülesanneteks. *Issued* ehk *taskid*

on jagatud vastavalt nende staatusele erinevatesse veergudesse ning nad liiguvad *boardil* vasakult paremale. Elemendid võivad olla nii füüsilised kui ka virtuaalsed. Füüsiline tähendab seda, et ülesanded või probleemid pannakse kirja kaartidele, mis võivad olla näiteks märkmepaberid, ning need kinnitatakse tahvlile. Virtuaalne tähendab aga seda, et ülesanded või probleemid pannakse kirja internetipõhisele tahvlile, millele kõik liikmed saavad ligi igal ajal igast kohast interneti olemasolul.

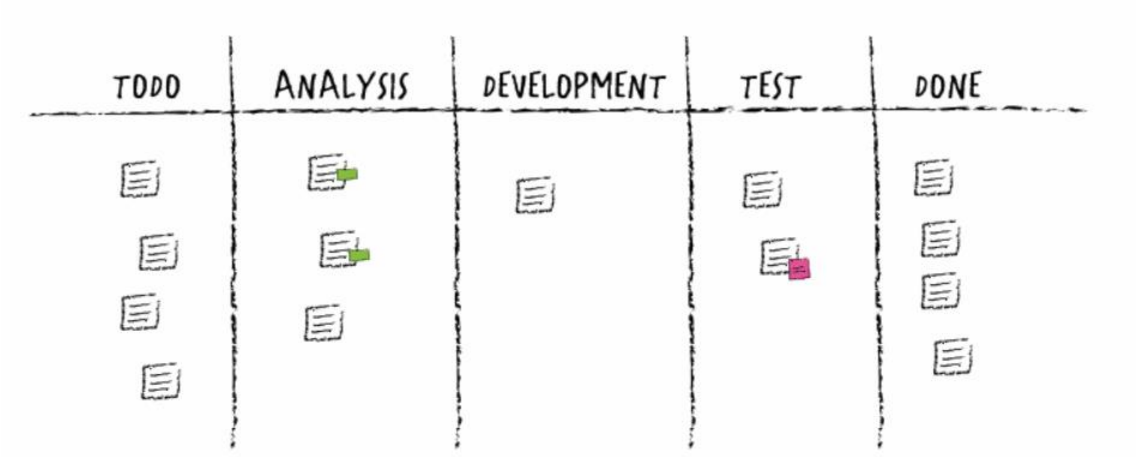
Kanban *boardi* veerud jaotatakse, nagu joonisel 1 näha, tavaliselt kolmeks osaks: *To Do*, *Doing* ja *Done* ehk valmis tegemiseks, tegemisel ja tehtud [3].



Joonis 1. Kanban board [15]

Eelnev kombinatsioon on väga üldine ning näiteks tarkvaraarendustiimides on *boardil* veerge palju rohkem. Nagu näidatud joonisel 2, siis need võivad olla näiteks sellised: *backlog*, analüüs, arendus, testimine ja *closed* ehk suletud. *Backlog* on mõeldud tulevaste tööde ja ideede kirja panekuks ning nende juurde asutakse siis, kui tekib aeg ja vajadus [16]. Tähtsamad tööd hoiab projektijuht eespool, sest nii on suurem tõenäosus, et need võetakse töösse [16]. Kui üldiselt liiguvad *taskid* vasakult paremale, siis siin võib juhtuda ka vastupidi, sest testimisel võib selguda, et arendus ei vasta nõuetele ning siis tuleb *task* juba vastavasse veergu tagasi liigutada. Kanban meetodil pannakse veergudele ka limiite, mis tähendab seda, et ühes veerus võib olla kindel arv ülesandeid. Kui see arv saab täis, siis suunatakse kogu tiimi tähelepanu sellele, et sealt *taske* edasi liigutada ning peale seda

juba tavapärase tööga jätkata. Selline käitumisviis aitab kiirelt avastada kitsaskohti ning hoiab töövoo maksimaalselt sujuvana [16].



Joonis 2. Kanban board tarkvaraarendustiimides [13]

Üheks Kanban *boardi* kasulikkuse põhjuseks tuuakse veel seda, et see loob ühtse pildi kogu projekti seisust. Teaduslikult on kindlaks tehtud, et aju suudab töödelda visuaalset pilti 60 000 korda kiiremini kui tavalist teksti. 90 protsenti andmetest tuleb ajju just läbi visuaalide, mille põhjal võib öelda, et aju eelistab pilti tekstile. Kanban *board* annabki edasi selle sama informatsiooni, mida tavaliselt kirjeldatakse tekstina, edasi pildina ning nii suudab aju sellest paremini aru saada. [14]

2.3 Gitlab company

Gitlab on avatud lähtekoodiga ettevõte, mis toodab tarkvara tarkvaraarenduslikule protsessile, mida kasutavad tänaseks rohkem kui 100 000 ettevõtet, ja kuhu panustavad igapäevaselt rohkem kui 2200 inimest. Gitlabi põhimõtted on koostöö, tulemus, efektiivsus, mitmekesisus ja läbipaistvus. Gitlab on ettevõte, mis usub, et igaüks kogu maailmas saab ühekorraga ja võrdselt panustada olenemata sellest, kus ta parajasti asub. [1]

Avatud allikatega ettevõte tähendab, et rakenduse kood, mida nad arendavad, on kõigile kättesaadav ning muudetav. Ka Gitlabi oma töötajad teevad kaugtööd ehk ei eksisteeri kontorit, kus kõigil oleks oma töökoht, vaid kogu töö tehakse ära kodust. See seletab, miks Gitlabi töötajad pärinevad viiekümne kolmest erinevast riigist. [1]

Gitlab ettevõte on ise väga suur eemalt töötamise propageerija, sest oma ettevõtte on nad peamiselt sellele üles ehitanud. Selleks, et eemalt töötamine ka sujuks ja oleks efektiivne, peab töökorraldus olema väga hoolikalt läbi mõeldud. Võrreldes tavaliste ettevõtetega, tehakse kaugtööd edendavates ettevõtetes kahte asja teisiti:

1. Ei saa oodata, et inimesed töötavad kõik samal kellaajal, sest suure tõenäosusega võivad nad asuda erinevates ajatsoonides ehk kõik töötavad siis, kui neil on selleks sobivaim aeg.

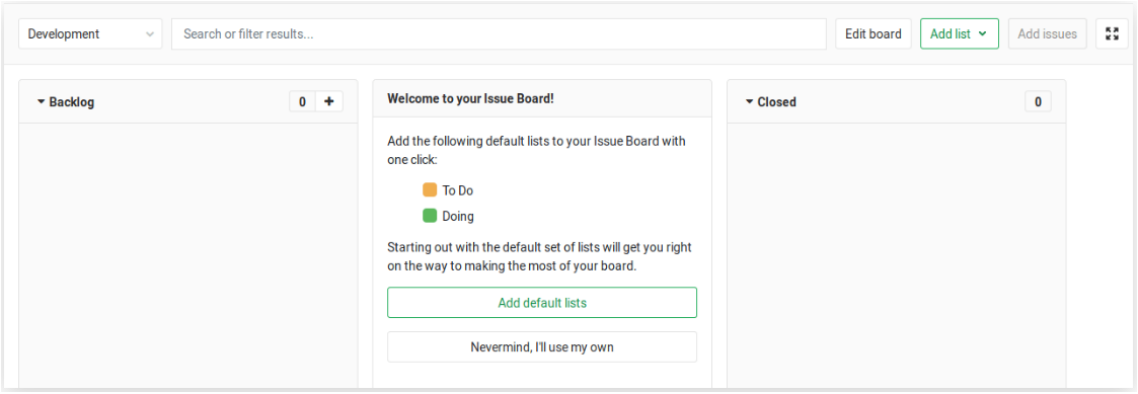
2. Informatsiooni ei saa hoida ainult oma peas, sest inimene, kes hoiab teistele olulist infot oma peas võib samal ajal magada, kui teised töötavad. Selleks tuleb absoluutselt kõik kirja panna.

Sellised piirangud mõjutavad organisatsiooni käitumisviisi ja olemust suurel määral. Esiteks ei pea keskenduma selle, kui kaua keegi kontoris on, vaid selle asemel saab keskenduda hoopis konkreetse inimese tehtud tööle. Teiseks, kui kõik pannakse kirja, siis ei teki informatsiooni puuduse ja halva kommunikatsiooni tõttu ootamatuid probleeme. Kolmandaks võib ettevõte palgata professionaale üle kogu maailma, mis on kasulik selle poolest, et ei pea muretsema nende lennutamise või elamistingimuste pärast. [17]

Gitlab toode on kooskõlas nende enese põhimõtetega. Nimelt toodavad nad tarkvaraarendusega tegelevatele ettevõtetele ja tiimidele töö paremaks jälgimiseks ja korraldamiseks erinevaid rakendusi. Üheks selle põhiosaks on Kanban *boardil* põhinev Gitlab *board*, mis kuulub juba eelnevalt mainitud virtuaalsete *boardide* hulka.

2.4 Gitlab board

Gitlab Issue *board* on Kanban *boardil* põhinev rakendus, mis aitab tiimidel oma projekti ja tööd visualiseerida ning töökorraldust paremini hallata. Joonisel 3 on näha, et *board* on algselt tühi ning sinna on märgitud ainult *backlog* ja *closed* listid, mille vahele jääv ala tuleb igal tiimil endal vastavalt vajadusele ära täita. Rakendus pakub ka võimalust lisada põhilised Kanban *boardi* juurde käivad listid, kui tiimil pole algselt kohe selge, mida neil täpsemalt oleks vaja *boardil* kajastada. Töökäigus saab *boardi* täiustada, vastavalt tiimi vajadustele kohandada ning ka üleliigseid liste ja muud eemaldada. Gitlab *board* koosneb osadest, mida inglise keeles nimetatakse nii: *lists*, *issues*, *labels*, *milestones*, *assignees* ja *merge requests*. [2]



Joonis 3. Gitlab Issue board [2]

Projektis võib kasutada ühe *boardi* asemel ka mitut [2]. Näiteks tarkvaraarenduslikus projektis võib eraldi *boardide* peal välja tuua UI ja API-ga seotud *issued*. Teise variandina võib *boardid* jagada ka näiteks analüüsi, arenduse ja testimise *boardideks*. Selline lahendus tuleb pigem kasuks rohkemate liikmetega ja suuremates projektides, sest seal on ka ülesandeid ja probleeme vaja rohkem raporteerida. Gitlab on *boardide* vahel liikumise ja *issuede* liigutamise teinud väga mugavaks ning mitme erineva *boardi* kasutamine ei ole kindlasti töötegemisel mingiks takistuseks.

2.4.1 Lists

Listid on *boardil* olevad tulbad või veerud, mille alla kirjeldatakse *issuesid* [2]. Vastavalt projektile võivad need olla väga erinevad, kuid tarkvaraarenduses koosneb *board* üdiselt kaheksast osast: *backlog*, analüüs, valmis arendamiseks, arendamisel, arendatud, valmis testimiseks, testimisel ja *closed*. *Listide* pealt saab füüsiliselt jälgida kogu töö voolu ja seisu. Kui ühte *listi* tekib palju *issuesid*, siis on see kohe näha ning tasub hakata uurima, mis põhjusel *issued* kuhjuvad. Kuhjumist aitab vältida *listides* olevate *issuede* arvu piiramine ehk kui *list* saab nii-öelda täis, siis suunatakse sellele rohkem ressurssi ja tehakse vajadusel tiimisisesed korrektuurid, et neid edaspidi vältida [2].

2.4.2 Issues

Issued ehk ülesanded on *boardi* kõige tähtsamad osad. Nimelt kirjeldatakse nendes järgnevad ülesanded, hetkeprobleemid ning ka tuleviku soovid. *Issue* koosneb pealkirjast ja kirjeldavast osast. Kirjeldava osa maksimaalselt efektiivseks ära kasutamiseks on Gitlab loonud *template* funktsiooni ehk malli tegemise võimaluse, mida kasutades ei jää

kindlasti ükski oluline info märkimata, sest see vastab projektisiseselt tehtud kokkulepetele ehk projektiliikmed saavad endale sobiva malli ise luua. [7]

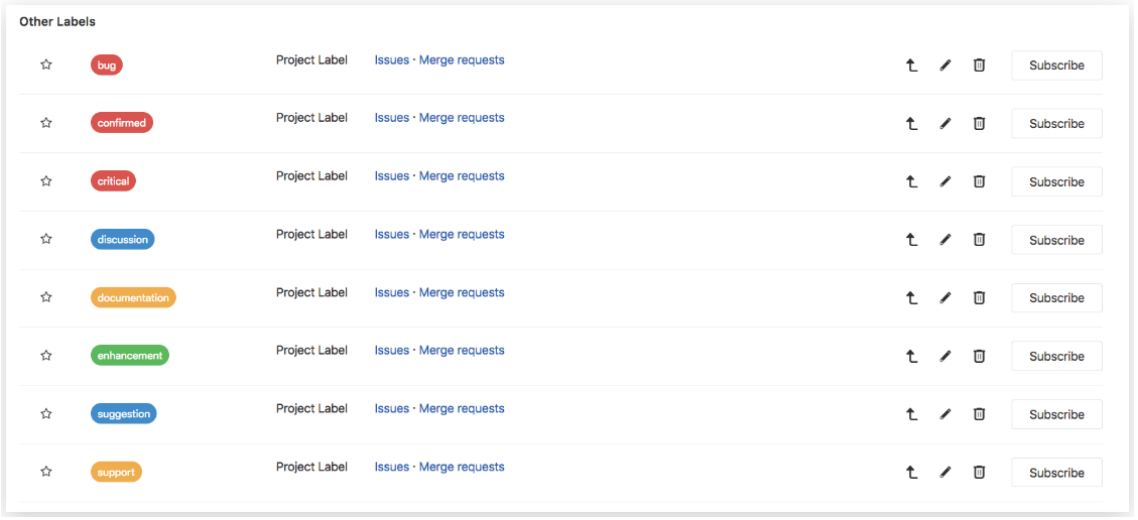
Issuede alla saab lisada kommentaare ehk kui ülesande tegijal tekib püstitajale küsimusi, siis saab ta seal samas *issue* all selle ära küsida ning nad ei pea omavahel füüsiliselt kokku saama [7]. Nii jääb kogu teemaga seotud informatsioon ühte kohta ning inimesed, kes antud teemaga tulevikus liituvad saavad kogu vahepeal tekkinud arutelust osa.

Lisaks on Gitlab teinud veel ka *issuede* ühendamise funktsiooni, mis tähendab, et ühe alla saab kommenteerida teist ning seal samas on ka selle teise *issue* staatus näha [7]. Näiteks tekib olukord, kus üks *issue* sõltub teisest ehk esimesega saab jätkata siis, kui teine on tehtud ja vastupidi. Kuna Gitlab näitab ka kommenteeritud *issuede* staatust, siis ei pea ühelt teisele liikuma selleks, et näha, mis teisega toimub. Selline tegutsemisviis hoiab jällegi kõigi *issuega* seotud inimeste aega kokku.

2.4.3 Labels

Labelite funktsioon on struktureerida ja kategoriseerida *issuesid* ja *merge requeste* kasutades kirjeldavaid tiitleid. Igale *labelile* vastab oma värv, mis aitab neid üksteisest eristada. Nende abil saab kiiresti ja mugavalt olulisi *issuesid* või *merge requeste* filtreerida, sest otsingu tulemusel kuvatakse kõik ainult konkreetse *labeliga* seonduvad *issued* või *merge requestid*. [10]

Joonisel 4 on näidatud *labelite* nimekiri, mida kasutatakse nende muutmiseks, kustutamiseks, prioritiseerimiseks ja muuks.



Joonis 4. Labels [10]

Gitlabis saab *labeleid* muuta ka projekti või lausa valdkonnapõhiseks, mis tähendab seda, et samasuguseid *labeleid* kasutatakse läbivalt kogu valdkonnas. *Labeleid* ei pea iga *boardi* jaoks uusi tegema, vaid piisab ühest korrast ning edaspidi saab neid kõiki juba uutel *boardidel* kasutada. [10]

2.4.4 Milestones

Milestone'id on Gitlabi loodud selleks, et jälgida *issuede* ja *merge requestide* liikumist saavutamaks laiemat eesmärki mingil kindlal ajaperioodil. Joonisel 6 on kujutatud ühte versioonipõhist *milestone'i*, kus visuaalne halli ja rohelise värviga tähistatud riba näitab *milestone'i* täidetavust. *Milestone'id* annavad võimaluse grupeerida ühe teemaga seotud *issued* ühte gruppi, millele saab määrata algus- ja lõppkuupäeva. [11]

Joonis 5. Milestone [11]

Milestone'id võivad olla nii funktsionaalsuse kui ka versioonipõhised. Esimene variant tähendab, et kõik ühe teemaga seotud *issued* seotakse ühe *milestone'iga* ning kui kõik need *issued* saavad *closed* oleku, siis on antud teema funktsionaalselt valmis. Kui *issued* on saanud endale versioonipõhise *milestone'i* külge, siis nende *closed* olek märgib versiooni valmis olekut. Versioonipõhised ei pea olema ühe rakenduse funktsionaalsusega seotud ning võivad koosneda ka mitme rakenduse erinevatest

issuedest. Versioonipõhiseid *milestone'e* kasutatakse näiteks kliendile tarnete tegemiseks.

Sarnaselt *labelitega* võib ka *milestone'e* laiendada projektipõhisest valdkonnapõhiseks [11]. Jällegi hoiab see aega kokku, sest igale uuele projektile ei pea sarnase sisuga *milestone'e* tegema, vaid võib kasutada samu läbivalt.

2.4.5 Assignees

Assignee funktsiooni kasutatakse konkreetsele isikule ülesande ehk *issue* määramiseks [2]. Ülesannet võib määrata nii projektijuht, ülesande täitja ise kui ka keegi kolmas, kes on kindel, et just tema valitud isik peab hakkama selle ülesandega tegelema. Ülesande tegija määramine sõltub jällegi projektisisesest kokkuleppest, kas seda teeb ülesande täitja ise või ülesande püstitaja. *Assignee* funktsiooni kasutamine aitab vältida seda, et mitu inimest sama ülesannet korraga lahendavad. Lisaks on *assignee* kõigile nähtav ehk küsimuste ja probleemide korral saab konkreetse isiku poole pöörduda.

2.4.6 Merge requests

Merge requestid võimaldavad muuta lähtekoodi mitmel inimesel samal ajal erinevates harudes [10]. Kogu valmis osa kood rakendusest asub põhiharus ning kui arendaja tahab midagi muuta, siis ta teeb sellest põhiharust kõrvalharu ning viib oma muudatused selles kõrvalharus sisse. Selleks, et oma muudatusi põhiharusse viia esitataksegi *merge request* ehk kahe haru ühendamise taotlus. Peale seda vaatavad juhtivarendaja ehk arhitekt või teised kaasarendajad tehtud muudatused üle ning kontrollivad, kas kirjutatud kood on korrektne. Muudatusi võib kontrollida üks või mitu inimest. Gitlab on loonud funktsiooni *merge requestid issuedega* siduda, mis võimaldab näiteks *merge requesti* sulgemisega ka *issuet* sulgeda või lihtsalt muuta *issue* asukohta *boardil* [12].

Merge requestid võimaldavad hoida üldist kontrolli koodi üle, sest halvasti tehtud arendust ei lasta põhiharusse ning nii saab vältida suuremaid probleeme kogu rakendusega. Lisaks on ka üldine kontroll ja ülevaade rakendusest ning arendajate tööst parem.

2.5 Jira

Jira on Atlassian ettevõttele kuuluv ülesannete jälgimise seade, mis võimaldab registreerida tarkvaralisi vigu ning teostada agiilset projekti arendust [18]. Ettevõtte X siseselt Jira rakendust ei kasutata, küll aga kasutab seda klient ning nii on paljud ettevõtte X töötajad sellega igapäevaselt tihedalt seotud.

Jira ja Gitlab *boardid* on omavahel väga sarnased. Nimelt kasutatakse ka Jira *boardi* agiilses arenduses ning seda saab vastavalt projekti vajadustele kujundada. Ka Jira *boardile* raporteeritakse rakendusega seotud vigu, mida nimetatakse ka *jiradeks*, ning neile saab märkida veaga seonduvaid projektisiseselt kokkulepitud märkeid ja selle lahendajaid. Lisaks saab ka Jiras ülesandeid mugavalt koodiga siduda ning nende staatust automaatselt uuendada vastavalt koodi seisukorrale.

3 Läbiviidud küsitluse tulemused

Käesolevas peatükis tutvustatakse ettevõttes X tegutsevas ettevõtte Y valdkonnas läbiviidud küsitlust, mille täpne struktuur on lisas 1, ja selle tulemusi. Valdkond koosneb viiekümne kolmest inimesest ning küsitlusele vastas neist 18. Autori arvates on tegu hea tulemusega, sest küsitlus oli küllaltki aeganõudev ja põhjalik ning inimesed pidid kulutama selleks oma vaba aega. Tabelis 1 on vastanud isikud jaotatud nende ametite järgi ning sealt on näha, et igast olulisest ametikohast on vastanud vähemalt üks inimene.

Tabel 1. Vastanute arvukus ja positsioonid

Positsioon ettevõttes	**Vastanute arv**
Analüütik	4
Arendaja	9
Projektijuht	3
Projektijuht/analüütik	1
Analüütik/arendaja	1

Anonüümne küsimustik viidi läbi Google Forms keskkonnas ja see oli jaotatud viite osasse:

1. Küsimused üldiselt olukorra kohta, mis puudutavad inimesi igapäevaselt ning mis takistavad neil efektiivset töötegemist. Paluti kirjeldada ka probleemi, mis on ette tulnud seoses tööülesannete saamise, jagamise või seisu jälgimisega.

2. Küsimused Jira rakenduse kohta. Kui palju seda kasutatakse, mis hinne antakse selle kasulikkuse eest ning põhjendused. Lisaks veel töökorraldust parandanud ja kahjustanud olukorrad.

3. Küsimused Gitlab rakenduse kohta. Kui palju seda kasutatakse, mis hinne antakse selle kasulikkuse eest ning põhjendused. Samuti ka töökorraldust parandanud ja kahjustanud olukorrad.

4. Küsimus, milliseid muid rakendusi on kasutatud ning rahulolu nendega.

5. Mida arvatakse koosolekutest ehk kas nendele kulub liiga palju aega ning millist infot sealt oodatakse. Lõpetuseks veel inimeste enda ettepanekud projektisiseseks protsessiparenduseks.

3.1 Üldised projektisisesed probleemid

Kaheksateistkümnest vastanust tõid üle poole välja selle, et enamasti seisab töö kliendi taga. Klient võib viivitada vastustega isegi päevi ning kui see vastus lõpuks saabub on teema juba ammu unustatud ja selle juurde tagasi tulemine võtab palju aega. Lisaks juhtub ka, et klient muudab tihti nõudeid ning vahel teeb ta seda isegi projekti lõpus. Sellest tulenevalt pole projektijuhtidel selgust, kas kõik vajalikud dokumendid said nõuete muudatustega uuendatud. Projekti lõpus muudetud nõudeid ei jõuta korralikult testida ning seetõttu võivad rakendusse vead sisse jääda, sest tähtaegadest tuleb kinni pidada.

Vastustes toodi ka probleemiks see, kui projekti vältel liitub sellega palju uusi inimesi, sest igaühe sisseelamine võtab aega ning tihti on neil palju küsimusi ja teemasid, millest kohe aru ei saa. Sellisel juhul võivad tekkida kommunikatsiooni probleemid, sest vanem olija ja teadja ei mäleta täpselt, kellele mida selgitatud on. Lisaks ei tutvustata uustulijatele projekti alguses suuliselt kokkulepitud reegleid ning võivad tekkida arusaamatused.

Seoses tööülesannete saamisega toodi välja, et tihti ei saa ülesande püstitusest kogu ülesande lahendamiseks vajaminevat informatsiooni ning seda tuleb püstitaja käest eraldi küsima minna. Lisaks ei muuda ülesande lahendaja ülesande seisu ehk ei märgi, kui see on tehtud ning kui keegi teine hakkab sama ülesannet lahendama, siis tekib muudatuste rakendusse sisseviimisel konflikt, mille lahendamine võtab aega. Arusaamatusi tekib ka siis, kui ülesandeid jagatakse erinevaid kanaleid pidi, sest ei saada aru, millised on tähtsamad ja olulisemad. Ülesannete erinevaid kanaleid pidi liikumine osutub takistuseks ka siis, kui isik on korraga seotud mitme projektiga, sest nii võivad mõned ära ununeda.

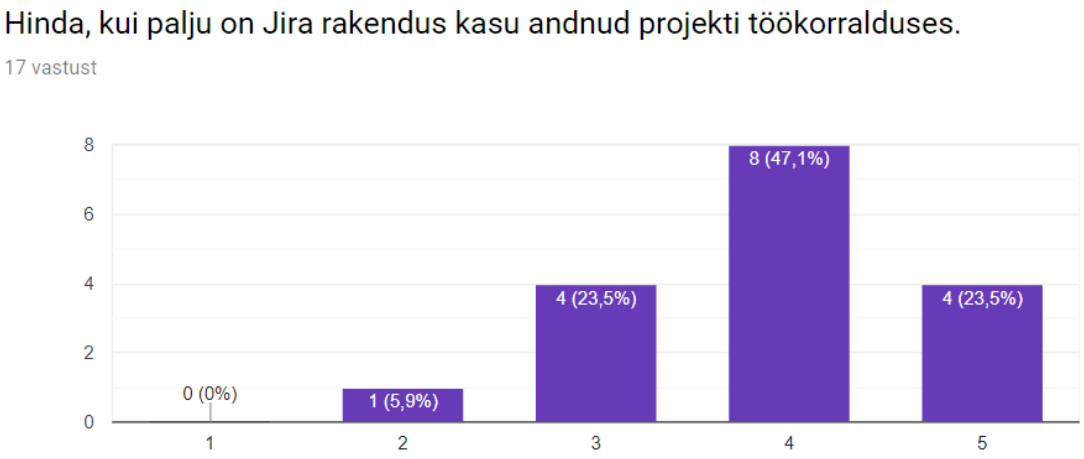
Valdkonnapõhiselt tehakse kliendile valmis arenduste näol tarneid ning ka nendega on seotud mõni probleem. Nimelt soovib klient tarnete kohta väga täpset infot, kuid see ei ole tihti projektisiseseltki selge. Üks vastajatest kirjutas, et tema arvates on tarnete tegemine mõttetult aeganõudev tegevus ning kood võiks kliendi keskkondadesse minna otse ja jooksvalt.

Kokkuvõtlikult on projektisisesed üldprobleemid sellised:

1. Klient viivitab vastustega päevi.
2. Klient muudab nõudeid projekti lõpus.
3. Uute projektiliikmete sisseelamine võtab palju aega ning nad ei saa piisavalt infot projekti kohta.
4. Ülesandel on puudulik püstitus, mille järgi pole koheselt võimalik ülesannet lahendama hakata ning tuleb küsida lisa informatsiooni.
5. Ülesandeid jagatakse erinevaid kanaleid pidi.
6. Pidev arusaamatus tarnetega nii projektisiseselt kui ka kliendi poolel.

3.2 Jira

Jirat kasutab kaheksateistkümnest inimesest seitseteist ning sellele pandi keskmiseks hindeks viiepallisüsteemis 3,88. Jooniselt 6 on näha, et kõige rohkem pandi hindeks neli. Üldiselt peetakse Jira rakendust kasulikuks, sest sealt saavad nii klient kui ka projektiliikmed vaadata tekkinud probleeme ning neid ka ise sinna püstitada. Kogu arutelu ajalugu jääb sinna alles ning kõik, kellel vaja, saavad sellega tutvuda. Lisaks saab Jirasse lisada ka kliendile küsimusi, mis hoiab ära liigset meilimist ja kõigile vastusest teavitamist ning koosolekuid. Ülesande kindlale isikule suunamise süsteem võimaldab näha, kes konkreetse ülesandega tegeleb ning vajadusel temalt täpsemat informatsiooni küsida.



Joonis 6. Jira hinded

Jira võimaldab mugavalt tarneid teha ning kogu tarneteemat paremini hallata. Enne Jirat tehti tarneid läbi meilide, kuid meilidest pole kindlasti võimalik kõigil osa saada, kellel tarnega seotud infot vaja võiks minna.

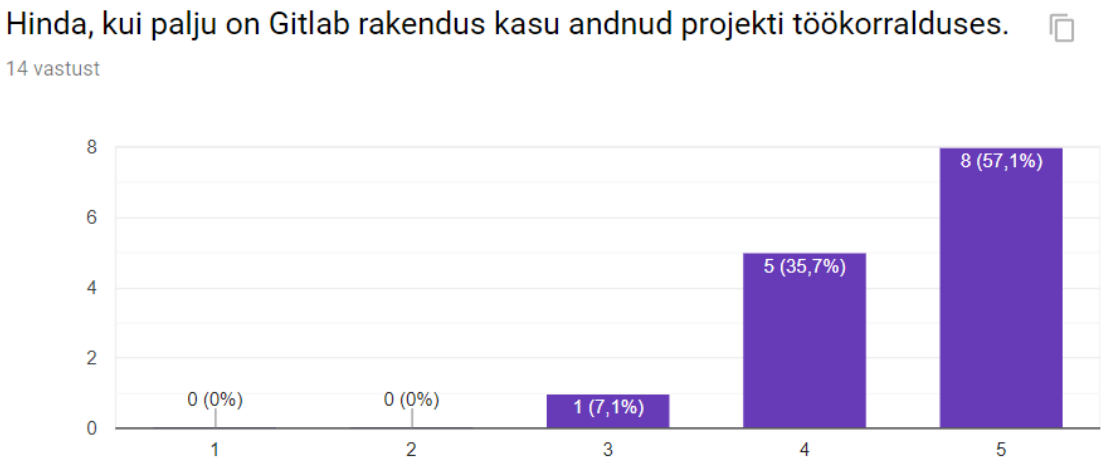
Jirat puudutavad probleemid:

- Ülesande püstitus on puudulik, ei kirjeldata täpset sisu ning on raske aru saada, kuidas seda lahendama peaks hakkama.
- Ülesande alla võib tekkida liiga palju kommentaare, kus alguses nõutakse üht ning pärast teist. Vahel küündib kommentaaride arv lausa kolmesajani ning nende hulgast on raske leida endale vajalikku infot.
- Klient ei vasta talle püstitatud küsimustele. Vastatakse erineva kiirusega.
- Algsetest kokkulepetest ei peeta kinni ehk ei kasutata ülesannete juures õigeid märkeid ning neid ei hoita kaasajas.
- Igal projektil uued kokkulepped ja reeglid. Ühtlus puudub ning raske on kokkulepetest kinni pidada. Vahel raskendavad seda ka ebaloogilised ja segased seadistused.
- Klient unustab uuendada tarnete paigalduse seisu ning ei teavita probleemidest.

- Mitu probleemi ühe Jira all ja ka vastupidi, ühe teema kohta mitu Jirat.

3.3 Gitlab

Gitlabi kasutab kaheksateistkümnest inimesest neliteist ning sellele pandi keskmiseks hindeks viiepallisüsteemis 4,5. Jooniselt 7 on näha, et kõige rohkem pandi hindeks viis. Vähesem kasutajate arv võib tuleneda sellest, et mõnes projektis on väga vähe inimesi ning ei ole tarvidust seda kasutada. On ka projekte, millega tegeleb ainult üks inimene ning see on juba tema enda otsustada, kas ta Gitlabi kasutab või mitte.



Joonis 7. Gitlab hinded

Gitlab rakendust peetakse projektisiseselt kasulikuks, sest sinna saab panna kirja viimsegi pisiasja, mis ei tohiks meelest ära minna. Eriti kasulikuks loetakse erinevate arendusharude kasutamist, mis võimaldavad konkreetset ülesannet siduda rakendusega. Sinna juurde käivad ka *merge requestid*, tänu millele saavad ka projektijuhid kindlad olla, et rakenduse kood on kontrollitud ja kvaliteetne.

Vastustes kirjutavad inimesed, et Gitlab *boardilt* on hea jälgida projekti üldist seisu ning sealt saab kiire ülevaate, kes millega tegeleb. Lisaks näeb sealt, mis staatuses ja kui palju on ülesandeid. Gitlab võimaldab *boardi* oma nägemuse järgi kujundada, et saada projektile võimalikult sobiv töövahend. Ülesannete küljes olevate märgete ehk *labelite* järgi saab ülesandeid ka mugavalt sorteerida ja otsida. Eriti peetakse Gitlab rakendust kasulikuks just suuremate projektide puhul, kus on ka rohkem liikmeid, sest kõigilt eraldi

küsida, millega nad tegelevad, ning ülesandeid personaalselt jagada muutub pikapeale liialt aeganõudvaks.

Gitlabi puudutavad probleemid:

- Ülesandeid ei hoita *up-to-date* ehk kaasajas.
- Ülesannete püstitus pole piisav ning nende põhjal ei ole võimalik kohe lahendamisega peale hakata.
- Ülesanded liiga killustatud.
- Vanu ülesandeid raske üles leida.
- *Boardi* seisu ei jälgita.

3.4 Muud rakendused

Küsitluses osalejate käest küsiti, kas ja milliseid rakendusi nad peale Jira ja Gitlabi veel kasutavad. Seitse inimest vastasid jah ning nendeks kasutatavateks rakendusteks olid:

- Fleep ja selle *taskinduse* funktsioon
- Excel
- Google drive files

Siinkohal on oluline mainida, et Fleep on kommunikatsiooni rakendus, mida kasutatakse valdavalt ettevõttes X, et üksteisega suhelda ning teadaandeid edastada. Üldiselt on iga projekti jaoks üldine *chat* ehk vestlus, kus jagatakse olulist informatsiooni, kuid vahel tehakse seal ka *taske* projektiliikmetele. Selline süsteem on *taski* loojale mugav, kuid selle täitjale ebamugav eriti siis, kui täitja on seotud mitme projektiga ning nendes projektides kasutatakse paralleelselt ka Gitlab *boardi*. Ebamugav sellel põhjusel, et selliselt loodud ülesanded lähevad tihti meelest ära ning isegi kui on meeles, siis on seda raske erinevatest vestlustest üles leida. Fleep taskid tulevad kasuks väikestes projektides, kus liikmete arv jääb alla kümne ning Gitlab *boardi* ei kasutata.

Excel on tabelihaldussüsteem ning seda kasutavad peamiselt analüütikud ja projektijuhid. Vastavalt isiku vajadustele teeb see projekti üldise vaate jälgimise lihtsamaks, kui projekt on jagatud osadeks ning nendel osadel on erinevad tähtajad. Lisaks aitab see ka töid planeerida ning nende mahtudest aimu saada. Excel tabeli valmistavad endale analüütikud ja projektijuhid ise ning üldiselt ei pea teised seda jälgima, vaid selle omanik annab vajadusel vastavat informatsiooni ise edasi.

Viimaseks toodi välja Google drive files rakendus, mis võimaldab teha dokumente, tabeleid, esitlusi, vorme ja palju muud. Selle peamine positiivne omadus seisneb selles, et faile saab projektiliikmetele jagada ning neid allalaadimata muuta. Muudatused on koheselt kõigile näha. Rakendus on seotud e-mailiga ehk eraldi kasutajat või muid kohmakaid protsesse selle kasutusele võtmiseks tegema ei pea.

3.5 Koosolekud

Bakalaureusetöö üks eesmärkidest oli koosolekutele kuluvat aega vähendada küsimuste „Kes mida teeb?“ arvelt. Selleks uuriti kõigepealt, kui palju kulutatakse üldse nädalas tunde koosolekutele ja *stand-upidele* ning kas seda aega oleks vaja vähendada. Lisaks uuriti ka, kas koosolekutelt saadav info vastab ootustele ning on kasulik.

Kõige rohkem kulub koosolekutele 3-5 tundi nädalas. Leidus ka inimesi, kellel kulub üks tund ning inimene, kellel kulub 35 tundi nädalas koosolekutele. Ainult ühe inimese arvates oleks vaja koosolekutele ja *stand-upidele* kuluvat aega vähendada ning toodi isegi ettepanek panna enne koosoleku algust selle täpne kava paika, et inimesed saaksid ise otsustada, kas neil on vaja sinna minna või mitte.

Küsimusele „Mis võiks olla maksimaalne tundide arv nädalas, mis kulub koosolekutele ja *stand-upidele*?“ vastati peamiselt, et 1-2 tundi ühe projekti kohta nädalas on piisav ning kokku kõikide projektide peale maksimaalselt 5-6 tundi. Toodi välja ka seda, et koosolekutele kuluv aeg sõltub palju projekti faasist. Näiteks analüüsi faasis on vaja palju arutada ja kogu tegevust hästi läbi mõelda ning tihti kliendile täpsustavaid küsimusi esitada.

Küsitluses küsiti veel, millist infot koosolekutelt oodatakse ning kas see vastab ootustele. Peamiselt vastab koosolekutelt saadav info ootustele, kuid ilmnesid ka mõningad probleemid:

- Räägitakse töösse mittepuutuvaid teemasid.
- Arutatakse probleeme, mis ei puuduta kõiki projektiliikmeid.
- Liiga palju aega kulub selle teada saamisele, kes mida teeb. Lisaks on see teadmine oluline ainult projektijuhtidele ja teisi see ei huvita.
- *Stand-upe* ei peeta püsti ning nendel arutatakse liiga üldiseid teemasid. Seetõttu venivad need ka liiga pikaks.
- Kui inimene ei saa koosolekul osaleda, siis hiljem ei anta talle teada, millised otsused vastu võeti ning mis projektisiseseid muudatusi tehti.

Üldiselt soovitakse koosolekutelt saada infot projekti ja selle seisu kohta, kuid oluline on ka tuleviku plaanide tegemine ning tähtaegade meelde tuletamine. *Stand-up* on jällegi jooksvate probleemide ja hetke takistuste aruteludeks ja lahendusteks ehk siis käsitleb lühemat ajaperioodi. Veel soovitakse teada, millist sisendit konkreetselt inimeselt oodatakse ning mida mis ajaks valmis peaks tegema. Vastanute arvates saab kõige suuremat kasu koosolekutelt projektijuht, kuid samas nad ka mõistavad, miks nende korraldamine vajalik on.

4 Läbiviidud muudatused ja nende analüüs

Käesolevas peatükis tutvustatakse projektis A läbiviidud muudatusi, mis on seotud kasutusel oleva Gitlab *boardiga*. Lisaks sellele viiakse läbi ka muudatuste tulemuste analüüs. Peatüki lõpust leiab ka eelnevas peatükis mainitud muude probleemide parandusettepanekud, mis on peamiselt seotud ühiste projekti ja valdkonna kokkulepetega ning Jira rakenduse kasutamisega.

4.1 Protsessiparendus

Projektisiseseks protsessiparenduseks viidi Gitlab *boardil* sisse järgnevad muudatused:

1. Muudeti olemasolevate listide sisu ja arvu. Lisandus analüüsi list, et ka analüütikute töö paistaks *boardilt* välja ning nii saab ka analüütikutele panna ülesandeid või täpsustavaid küsimusi mõne *issue* kohta. Peale muudatusi oli listide nimekiri selline: *backlog*, analüüs, valmis arenduseks, arenduses, arendatud/valmis testimiseks, testimisel, *closed*.

2. Loodi *issuede* tegemiseks *template* ehk mall, mille järgi tuleb ülesannet kirjeldada. Selle osad olid:

    a. Mis viga?

    b. Kus see viga tekib?

    c. Kellega see viga tekib?

    d. Milline oleks õige lahendus?

    e. Lisaks pildid ja videod kui tarvis.

3. Loodi iga rakenduse jaoks eraldi *label*, et neid oleks lihtsam eristada ning vajadusel *closed* listist otsida.

4. Kasutusele võeti *milestone* funktsioon ja seoti need kindla rakenduse *issuedega*. Eelnevalt on mainitud, et *milestone'e* saab kasutada nii funktsionaalsuse kui versiooni tegemiseks. Antud projektis näitavad *milestone'id* funktsionaalsuse valmisolekut.

5. Sõlmiti suulised kokkulepped, et kõik liikmed hoiavad *boardi up-to-date* ja võtavad oma nime *issue* küljest ära kui nad sellega töö lõpetavad. Samuti tuleb ülesandega alustamiseks see enda nimele panna, et teised projektiliikmed saaksid aru, kes millise ülesandega tegeleb.

Muudatused viidi sisse ainult Gitlab *boardil* seetõttu, et see on lõputööks valitud ettevõttes juba varasemast ajast kasutuses ning paljud liikmed olid sellega ka varem kokku puutunud, samuti on tegemist kaasaegse vahendiga ja omab kõiki tarkvaraarenduslikule projektile vajaminevaid funktsioone. Lisaks on vahend antud bakalaureusetöö põhifookuses. Küsitluses tulid välja veel ka kliendi ja Jira rakendusega seotud probleemid, kuid nende lahendamine nõuab juba rohkem ressurssi ja aega, sest mitmeid aastaid on koos kliendiga sellist töömustrit hoitud. Keeruline on kõiki osapooli äkitselt uut töökorraldust järgima panna. Selleks tuleks kaasata mõlemad osapooled ning teemat põhjalikult analüüsida. Peatüki lõpus on välja toodud ka mõned parandusettepanekud, mida võiks sisse viia kliendiga töökorralduse parenduseks.

Muud probleemid seoses koosolekutel jagatava info, uute inimeste projekti sisseelamise ja muude rakenduste kasutatavusega põhinevad peamiselt ühistel kokkulepetel ning nendest mittekinnipidamist ei ole võimalik Gitlab *boardiga* lahendada. Eriti oluliseks peab lõputöö autor projektijuhtide kokkulepetest kinnipidamist ja muudatustega kaasa tulemist, sest nemad on inimesed, kes saavad oma meeskondi kõige rohkem mõjutada ja nende tegevust juhtida. Antud projektis on kaks juhti, üks otsesem ja teine kaudsem. Otsene juht tuli muudatustega kaasa väga vähesel määral, sest tal puudus eelnev kokkupuude ja kogemus Gitlab *board* rakendusega ja paari nädala jooksul ei jõudnud ta sellega ära harjuda. Kaudsemal juhil oli aga varasem kokkupuude eelnevatest projektidest olemas ning muudatustega harjumine nii palju aega enam ei võtnud.

4.2 Tulemuste analüüs

Lõputöö tulemusel loodi uus parema süsteemiga Gitlab *board*, mille eesmärgiks oli lahendada vähemalt mõned töötajate poolt välja toodud probleemid. Eelmises alapeatükis väljatoodud muudatustele on selgunud järgnevad tulemused.

Esiteks lisatud analüüsi list tuli kasuks testijatele ja arendajatele, sest kui mõne *issue* kohta tekkis küsimus, siis sai selle kommenteerida *issue* juurde ning lisada see analüüsi listi, et analüütik teaks sellele vastata. Eelnevalt jäi *issue* oma lahendust ootama sellesse listi, kus ta parasjagu oli ning analüütikud pidid üle terve *boardi* endale suunatud ülesandeid otsima.

Teiseks kasutusele võetud malli hakkasid kasutama kõige enam testijad. Projektijuhid ja analüütikud sellest kinni ei pidanud ning nende loodud ülesanded olid väga üldsõnalised. Arendajal on raske aru saada, mida peaks parandama ning testijal, kuidas parandusi testida. Senikaua kuni kõik tegelevad enda loodud *issuedega* süsteem toimib, kuid kui ülesandega liitub keegi kolmas, siis tekib palju küsimusi. Malli kasutamine hoidis nende inimeste aega kokku, kes selle konkreetse *issuega* tegelesid, sest ei olnud vaja küsida täpsustavaid küsimusi.

Kolmandaks loodi igale rakendusele oma märge ehk *label* ning selle põhjal võib küll öelda, et kõik kes *boardi* aktiivselt kasutasid lisasid ka oma ülesannetele õigeid märkeid. *Board* on lihtsamini jälgitav, sest igal märkel on oma värv ning selle järgi on mugav inimesele olulisi teemasid teiste seast eristada. *Boardile* saab anda käsu kuvada ainult mingi kindla märkega ülesandeid. Muudatuse tulemusena ei pidanud projektijuht enam rakendusega tegelevalt inimeselt küsima, mis seisus see on, ning hoiti mõlema aega kokku.

Lisaks iga rakenduse eraldi *labelile* võeti kasutusele ka teisi. Nendega sai märkida *issue* juurde ka muud olulist informatsiooni, näiteks takistavat või kiiremini lahendust nõudvat viga, milleks olid vastavalt „takistav“ ja „kiire“. Nendega prioritiseeriti ülesandeid ning näiteks takistava vea puhul ei pidanud teine projektiliige selle taga ootama, vaid ülesanne võeti esimesena töösse ning leiti sellele kiiresti lahendus. Kõik liikmed said oma tavapärase tööga jätkata kellegi järel pikalt ootamata.

Neljandaks võeti kasutusele *milestone* funktsioon, mis pidi kõige rohkem kasu tooma projektijuhtidele, kes saavad nende pealt vaadata, mis seisus mõni rakendus on ehk kui palju on tehtud ja kui palju veel teha. Üks kahest projektijuhist hakkas funktsiooni kasutama ning see andis talle erinevatest projekti osadest hea ülevaate. Lisaks projektijuhile tuli see ka teistele liikmetele kasuks, kes *milestone'e* jälgisid. Nimelt vahel tekib olukord, kus jääb mõni väike ülesanne veel tegemata ning ise ollakse juba järgnevate teemade juures. Sellisel juhul saabki *milestone'i* pealt vaadata, kas midagi on veel ära ununenud, mis takistab teema lõpetatuks lugemist või saab selle siiski lõpetada. *Milestone* funktsiooni kasutuselevõtt tuli kasuks projekti üldseisu uurimisel, sest väga täpselt oli näha, mis seisus erinevad osad on.

Viiendaks loodi kokkulepe, et inimesed, kes *issuega* töö lõpetavad võtavad sellelt oma nime küljest ära. See kokkulepe oli suunatud pigem arendajatele, sest nende töö liigub edasi testimisse ning kui nad jätavad oma nime *taskile* külge, siis ei saa testija aru, kas võib tehtud arendusi testida või mitte. Kokkuleppest pidasid kõik arendajad kinni ning testijate töö muutus selle võrra paremaks. Juba enne muudatuste sisse viimist märkisid enamus arendajad ülesande juurde oma nime, kui nad seda lahendama hakkasid. Muudatusi tehes sai see meelde tuletatud ka teistele projektiliikmetele, kes kõik nime märkimist jälgima hakkasid. Sellega seoses lahenes probleem, kui mitu inimest tegelevad korraga sama ülesandega ning tagajärgedega tegelemise pealt hoiti aega kokku. Lisaks oli kohe *boardilt* näha, kes on mõne ülesande juurde liiga kauaks pidama jäänud, sest näiteks juba järgmise päeva *stand-upil* tuli see välja. Viimane oli projektijuhtide jaoks oluline probleem, mis sai *assignee* funktsiooniga lahendatud.

Töö üheks eesmärgiks oli vähendada *stand-upidele* kuluvat aega ehk vältida seda osa, kus uuritakse kes millega tegeleb. Nimelt projektis on 16 inimest ja *stand-upil* kulub igale inimesele keskmiselt kaks minutit, et teada saada, millega ta täpselt tegeleb. See teeb kõigi liikmete peale kokku 32 minutit ning seejärel arutatakse tekkinud probleeme ja nende lahendusi. Lisaks jagatakse ka muud olulist informatsiooni. Kes millega tegeleb osa tuli nende inimeste puhul, kes *boardi* täitsid, *boardilt* välja. Ülejäänud inimeste puhul tuli ka edaspidi *stand-upidel* uurida, millega nad tegelevad. Kuna *boardi* muudatustega tulid kaasa umbes pooled inimesed, siis läkski *stand-upi* aeg poole võrra lühemaks „kes millega tegeleb" teema arvelt.

Üldiselt võib tulemust pidada positiivseks, sest lõputöö alguses välja toodud probleemidele leiti lahendused kõigile, mida vähegi andis Gitlab *boardiga* parandada. Kõik, kes muudatustega kaasa tulid, ütlesid, et need on kasulikud ning mugavdavad tööprotsessi. Töö autor usub, et mida suurem on projektiliikmete ja tehtavate tööde arv, seda vajalikum on *board*, sest vastasel juhul on töökulgu pea võimatu jälgida.

4.3 Muud parandusettepanekud

Antud lõputöö keskendus ainult Gitlab *boardi* kasutamisele ja selle abil projektisisese tööprotsessi parendamisele, kuid nagu eelnevas peatükis selgus, siis on probleeme palju rohkem ning nendele pakutakse käesolevas alapeatükis erinevaid lahendusi. Nimekiri parandusettepanekutest muude probleemide jaoks, mida ei ole võimalik Gitlab *boardiga* lahendada:

- Teha kliendile ettepanek Jira kasutuse ühtlustamiseks.
- Vajadusel viia läbi Jira koolitused mõlemal osapoolel.
- Vajadusel viia läbi ettevõtte sisene Gitlab koolitus, et kõik saaksid selle kasutatavusest ühte moodi aru.
- Ülesannete jagamiseks kasutada projektisiseselt ainult Gitlab rakendust, mitte Fleepi, Excelit ega muud.
- Pidada kinni ühistest kokkulepetest. Tutvustada kokkuleppeid ka uutele projektiliikmetele.
- Kasutada ka tarnete tegemisel abivahendeid, et oleks selge, mis ja millal tarnitud on ning mis peaks olema järgmine.

Kogu bakalaureusetöös välja toodud probleemid esinevad rohkemal või vähemal määral paljudes tarkvaraarendusega seotud projektides ja ettevõtetes, kuid kindlasti ka muude valdkondadega seotud projektides ja ettevõtetes. Uuringu põhjal selgus, et Gitlab Issue *board* on hea abivahend töö paremaks korraldamiseks ning selle asjakohane kasutamine aitab igat protsessi parendada. Asjakohane seepärast, et kui projektisiseselt sellest ühtselt kinni ei peeta, siis neile, kes peavad, ei pruugi vahendist abi olla, sest kindlasti peab veel lisaks täpsustavaid küsimusi küsima või kolleegidega teema üle arutlema.

Antud töös käsitletud Gitlab *board* tuleb oma funktsionaalsuse poolest kasuks kõikidele tarkvaraarendusega tegelevatele meeskondadele, kuid kuna kogu funktsionaalsuse kasutamine ei ole kohustuslik, siis saavad seda hõlpsasti kasutada ka muude valdkondadega seotud meeskonnad. *Boardi* üks parimaid omadusi on see, et seda saab kujundada vastavalt vajadustele. Nii võib saavutada olukorra, kus mitu erinevat ettevõtet kasutavad sama vahendit, kuid väga erineval otstarbel ja eesmärgil.

5 Kokkuvõte

Käesolev bakalaureusetöö käsitleb tarkvaraarenduslikus projektis tekkivaid probleeme ning nende lahendusi. Probleemid on peamiselt seotud projektisisese töökorraldusega, millele leitakse ka lahendused, kuid lisaks sellele toodi välja ka probleemid seoses kliendiga suhtlemisega. Viimaseid probleeme antud lõputöös ei lahendata, kuid pakutakse parandusettepanekuid.

Töö alguses tutvustati lähemalt probleemi ja selle olemust ning sellega seotud asjaolusid. Toodi välja töö peamised eesmärgid ehk probleemid, mis esinevad paljudes tarkvaraarendusega seotud projektides. Tutvustati ka projekti, mille töökorraldust hakati lõputöö raames parendama, ja selle juurde käivaid töövahendeid, milleks olid Jira ja Gitlab, ning töömeetodeid.

Töö teoreetilises osas oli põhjalikult ära seletatud agiilses arenduses kasutatava Kanban *boardi* olemus, ajalugu ning selle kasutusviis. Kanban virtuaalsete *boardide* hulka kuuluvad töö põhifookuses olev Gitlab *board* ja Jira *board*, mille kasutuse parendus antud töö eesmärkidesse ei mahtunud. Kõige olulisema vahendi, Gitlab *boardi*, kõik osad on põhjalikult lahti seletatud ning nende kasutus ja kõik omadused välja toodud.

Töö praktilises osas tutvustati küsimustikku, mis viidi läbi ettevõttes X, et saada teada täpsemad projektisisesed ja ka üldisemad valdkonnas leiduvad probleemid. Küsimustik oli jaotatud viide osasse: üldised projektisisesed probleemid, Jira rakendusega seotud probleemid, Gitlab rakendusega seotud probleemid, koosolekute analüüs ning muude rakenduste kasutatavuse uuring. Selgus, et kasutatavad rakendused on kasulikud, kuid tihti tekivad takistused inimeste omavahelistes kokkulepetes ja nendest kinnipidamisel. Samuti polnud siiani rakenduste kasutatavus ja nende funktsioonid põhjalikult läbi uuritud ja projektile vastavusse seatud.

Töö viimases peatükis tutvustati projektis A sisse viidud muudatusi ning nende tulemuste analüüsi. Tulemused olid üldpildis positiivsed, sest lahendatud said kõik töö alguses välja toodud probleemid. Muudatustega tulid kaasa pooled projektiliikmetest ning kaasatulijate

arvates olid muudatused head ja tooksid veel hoolikamal kasutamisel projekti rohkem efektiivsust.

Kasutatud kirjandus

[1] „Company,“ Gitlab, 2019. [Võrgumaterjal]. Available: https://about.gitlab.com/company/. [Kasutatud 1 mai 2019].

[2] „Gitlab issue boards,“ Gitlab, 2019. [Võrgumaterjal]. Available: https://docs.gitlab.com/ee/user/project/issue_board.html. [Kasutatud 10 mai 2019].

[3] D. Radigan, „Kanban,“ Atlassian, 2019. [Võrgumaterjal]. Available: https://www.atlassian.com/agile/kanban. [Kasutatud 29 aprill 2019].

[4] „Jira,“ Wikipedia, 6 mai 2019. [Võrgumaterjal]. Available: https://en.wikipedia.org/wiki/Jira_(software). [Kasutatud 10 mai 2019].

[5] „Stand-up meeting,“ 2 aprill 2019. [Võrgumaterjal]. Available: https://en.wikipedia.org/wiki/Stand-up_meeting. [Kasutatud 15 mai 2019].

[6] „Inglise-eesti masintõlkesõnastik,“ EKI, 2019. [Võrgumaterjal]. Available: http://portaal.eki.ee/dict/ies/. [Kasutatud 15 mai 2019].

[7] „Issues,“ Gitlab, 2019. [Võrgumaterjal]. Available: https://docs.gitlab.com/ee/user/project/issues/. [Kasutatud 2 mai 2019].

[8] H. Vallaste, „e-Teatmik,“ 2019. [Võrgumaterjal]. Available: http://www.vallaste.ee/. [Kasutatud 15 mai 2019].

[9] „Backlog,“ Wikipedia, 22 november 2018. [Võrgumaterjal]. Available: https://en.wikipedia.org/wiki/Backlog. [Kasutatud 15 mai 2019].

[10] „Labels,“ Gitlab, 2019. [Võrgumaterjal]. Available: https://docs.gitlab.com/ee/user/project/labels.html. [Kasutatud 2 mai 2019].

[11] „Milestones,“ Gitlab, 2019. [Võrgumaterjal]. Available: https://docs.gitlab.com/ee/user/project/milestones/. [Kasutatud 2 mai 2019].

[12] „Merge requests,“ Gitlab, 2019. [Võrgumaterjal]. Available: https://docs.gitlab.com/ee/user/project/merge_requests/. [Kasutatud 3 mai 2019].

[13] K. Leopold, Kanban and change leadership: creating a cluture of continuous improvement, New Jersey: Wiley, 2015.

[14] J. Terry, „What is Kanban?,“ LeanKit, 2019. [Võrgumaterjal]. Available: https://www.planview.com/resources/articles/what-is-kanban/. [Kasutatud 29 aprill 2019].

[15] „What is a Kanban board?,“ Digite, 2019. [Võrgumaterjal]. Available: https://www.digite.com/kanban/kanban-board/. [Kasutatud 13 mai 2019].

[16] M. Rehkopf, „What is a Kanban board?,“ Atlassian, 2019. [Võrgumaterjal]. Available: https://www.atlassian.com/agile/kanban/boards. [Kasutatud 30 aprill 2019].

[17] J. v. d. Voort, „Medium,“ 9 jaanuar 2019. [Võrgumaterjal]. Available: https://medium.com/@jobv/im-leaving-gitlab-to-help-everyone-work-remotely-1f6828ec45d5. [Kasutatud 1 mai 2019].

[18] „Jira,“ Atlassian, 2019. [Võrgumaterjal]. Available: https://www.atlassian.com/software/jira. [Kasutatud 14 mai 2019].

Lisa 1 – Küsitlus

1. Mis on sinu positsioon tiimis? (projektijuht, arendaja, analüütik, jne)
2. Nimeta mõned projektisisesed probleemid, millega igapäevaselt pead tegelema ja mis takistavad või segavad projekti efektiivset realiseerimist.
3. Kirjelda mõnda probleemi, mis on sinul ette tulnud seoses tööülesannete saamise, jagamise või seisu jälgimisega.
4. Kas kasutad Jira rakendust?
5. Hinda, kui palju on Jira rakendus kasu andnud projekti töökorralduses.
6. Põhjenda (nt mõnes projektis on kasu olnud, mõnes mitte; isiklikult ja üldiselt)
7. Too näiteid oma projektist, kus Jira rakendus on töökorraldust parandanud. (Vähem meilimist, vähem koosolekuid, parem suhtlus kliendiga, jne)
8. Too näiteid oma projektist, kuidas Jira on töökorraldust kahjustanud. (Arusaamatu, aeganõudev, pole kaasajastatud, kõik ei jälgi täpselt, jne)
9. Kas kasutad Gitlab rakendust projektitööde jälgimiseks/korraldamiseks?
10. Hinda, kui palju on Gitlab rakendus kasu andnud projekti töökorralduses.
11. Põhjenda (nt mõnes projektis on kasu olnud, mõnes mitte; isiklikult ja üldiselt)
12. Too näiteid oma projektist, kus Gitlab rakendus on töökorraldust parandanud. (Vähem meilimist, vähem koosolekuid, parem meeskonnasisene suhtlus, jne)
13. Too näiteid oma projektist, kuidas Gitlab on töökorraldust kahjustanud. (Arusaamatu, aeganõudev, pole kaasajastatud, kõik ei jälgi täpselt, jne)
14. Kas kasutad mingit muud rakendust tööprotsesside paremaks organiseerimiseks?
15. Nimeta see rakendus.

16. Hinda selle rakenduse kasu tööprotsessidele.

17. Põhjenda (nt mõnes projektis on kasu olnud, mõnes mitte; isiklikult ja üldiselt)

18. Too näiteid oma projektist, kus sinu nimetatud rakendus on töökorraldust parandanud. (Vähem meilimist, vähem koosolekuid, parem meeskonnasisene suhtlus, jne)

19. Too näiteid oma projektist, kuidas sinu nimetatud rakendus on töökorraldust kahjustanud. (Arusaamatu, aeganõudev, pole kaasajastatud, kõik ei jälgi täpselt, jne)

20. Sinu ettepanekud, kuidas projektisisest töökorraldust veelgi efektiivsemaks muuta.

21. Kui palju kulub sul praegu aega koosolekutele ja *stand-upidele*? (tundi nädalas)

22. Kas sinu arvates oleks vaja koosolekutele ja *stand-upidele* kuluvat aega vähendada? Kuidas seda teha?

23. Mis võiks olla maksimaalne tundide arv nädalas, mis kulub koosolekutele ja *stand-upidele*?

24. Millist infot koosolekutelt/*stand-upidelt* ootad?

25. Kas koosolekutelt/*stand-upidelt* saadav info vastab sinu ootustele?

26. Kui sul tekkis soov veel midagi lisada, siis kirjuta see siia.