HAMK
Hämeen ammatti-
korkeakoulu

ARVIOINTI JA PALAUTEJÄRJESTELMÄ OHJELMISTOKEHITTÄJILLE

Ammattikorkeakoulututkinnon opinnäytetyö

Tieto- ja viestintätekniikka, insinööri (AMK)

Kevät, 2024

Otto Kilpirinne

HAMK
Hämeen ammatti-
korkeakoulu

Tieto- ja viestintätekniikka AMK
Tekijä Otto Kilpirinne
Työn nimi Arviointi ja palautejärjestelmä ohjelmistokehittäjille
Ohjaaja Jari Mustajärvi

Tiivistelmä
Vuosi 2024

Tämä opinnäytetyö on laadittu toimeksiantona ohjelmistokehitysyritys Fisplaylle, joka pyrkii vastaamaan kasvavaan tarpeeseen kehittää menetelmiä ohjelmistokehittäjien taitojen arviointiin ja seurantaan nopeasti muuttuvassa työympäristössä. Työn keskeiset käsitteet ja teoriat liittyvät ohjelmistokehityksen, taitojen arvioinnin sekä ammatillisen kehityksen ympärille, tunnistaen alalla vallitsevan tarpeen jatkuvalle oppimiselle ja osaamisen päivittämiselle.

Opinnäytetyö on toteutettu analysoimalla kerättyä aineistoa, joka koostuu ohjelmistokehittäjien itsearvioinneista, yrityksen käyttämästä teknologiadatasta ja henkilöstöhallinnon tarjoamasta tiedosta. Aineisto on kerätty yhteistyössä yrityksen kanssa, ja sen avulla on kehitetty monipuolisia metodeja kehittäjien osaamisen tarkastelemiseen ja kehittämiseen. Tutkimusmenetelminä ovat olleet kvalitatiivinen ja kvantitatiivinen analyysi, jotka on toteutettu käyttäen kyselyitä sekä ohjelmistotyökalujen logiikan tarkastelua.

Tulokset osoittavat, että kehitetyt menetelmät tarjoavat tehokkaita tapoja seurata ja edistää ohjelmistokehittäjien ammatillista kasvua. Ne tarjoavat työkaluja henkilökohtaisten vahvuuksien tunnistamiseen, kehityskohteiden kartoittamiseen, yksilöllisten kehityssuunnitelmien rakentamiseen sekä resursoinnin kohdentamiseen. Opinnäytetyön puitteissa kehitetyt mallit ja työkalut voivat toimia perustana jatkokehitykselle ja laajemmalle soveltamiselle työelämässä.

Johtopäätöksissä korostetaan, että opinnäytetyön tulokset tarjoavat arvokkaita resursseja organisaatioiden henkilöstön kehityksen tukemiseen. Työelämä hyötyy erityisesti kehittäjien osaamisen systemaattisesta seurannasta ja kehityksestä, mikä mahdollistaa strategiset päätökset koulutusresurssien kohdentamiseksi. Tulevaisuudessa tulisi laajentaa tutkimusta ohjelmistokehittäjien työhyvinvoinnin ja motivaation vaikutuksesta heidän ammatilliseen kehitykseensä, jotta voidaan luoda kokonaisvaltaisempia kehitysmalleja.

Avainsanat Ohjelmistokehitys, arviointimenetelmät, ohjelmistokehittäjät

Sivut 24 sivua ja liitteitä 2 sivua

HAMK
Hämeen ammatti-
korkeakoulu

Bachelor of information and communication technology Abstract
Author Otto Kilpirinne Year 2024
Subject Assessment and Feedback System for Software Developers
Supervisors Jari Mustajärvi

This thesis has been prepared as a commission for the software development company Fisplay, which seeks to address the growing need to develop methods for assessing and monitoring software developers' skills in a rapidly changing work environment. The central concepts and theories of the work relate to software development, skills assessment, and professional development, recognizing the industry's prevailing need for continuous learning and skills updating.

The thesis has been carried out by analyzing collected data, which consists of software developers' self-assessments, technology data used by the company, and information provided by human resources management. The data has been collected in collaboration with the company, and it has been used to develop a variety of methods for examining and developing developers' competencies. The research methods have included qualitative and quantitative analysis, which have been carried out using surveys and examination of software tools' logic.

The results show that the developed methods offer effective ways to track and promote the professional growth of software developers. They provide tools for identifying personal strengths, mapping development targets, constructing individual development plans, and targeting resourcing. The models and tools developed within the framework of the thesis can serve as a basis for further development and broader application in the workplace.

The conclusions emphasize that the results of the thesis provide valuable resources to support organizations' personnel development. The professional world particularly benefits from the systematic monitoring and development of developers' skills, which enables strategic decisions to be made in allocating training resources. In the future, research should be expanded to examine the impact of software developers' well-being and motivation on their professional development, to create more comprehensive development models.

Keywords Software development, assessment methods, software developer

Pages 24 pages and appendices 2 pages

Sisällys

Kuvat, taulukot ja kaavat

Liitteet

1 Johdanto

Ohjelmistokehittäjä lyhyesti koodari on ammattilainen, joka suunnittelee, toteuttaa, parantaa ja ylläpitää tietokoneohjelmistoja tai -järjestelmiä. Ohjelmistokehittäjän työnkuvaan voi kuulua ohjelmiston tai järjestelmän suunnittelu ja toteutus, olemassa olevan järjestelmän ylläpito ja jatkokehitys ja ongelmatilanteiden ratkaisu (Kim 2023). Ohjelmistokehittäjä työskentelee erilaisten ohjelmointikielien, kehitystyökalujen ja -menetelmien parissa. Ohjelmistokehittäjät ovat usein erikoistuneet tiettyyn osa-alueeseen, kuten front-end (käyttöliittymä), back-end (järjestelmän taustapalvelut) tai full-stack (front-end ja back-end), mobiilikehitys, tietoturva tai tietokantasuunnittelu muutamia mainitakseni. Työssä menestyminen vaatii jatkuvaa oppimista, ongelmanratkaisutaitoja, analyyttista ajattelua ja hyvää yhteistyökykyä. Työ on myös hyvin monipuolista ja koodaamisen lisäksi kehittäjän työhön kuuluu usein myös asiakastapaamisia, tarpeiden määrittelyä, dokumentointia, testausta, ohjelmistojen julkaisua, ylläpitoa ja yhteistyötä muiden kehittäjien, testaajien ja asiakkaiden kanssa.

Ohjelmistoteollisuus kehittyy nopeasti ja ohjelmistokehittäjän pitää pysyä ajan tasalla teknologian muutoksista ja pystyä kehittämään omia taitojaan ja osaamistaan. Ohjelmistokehitys on ala, jossa osaamisen tasoa voi olla joskus vaikea arvioida. Ohjelmistokehittäjien taidot ovat avainasemassa ohjelmistokehitysprojektien onnistumisessa ja asiakastyytyväisyydessä. Lisäksi ohjelmistoyritysten tuottavuudessa on suuria eroja. Ylin neljännes ohjelmistokehitysyrityksistä kehittää ohjelmistoja kolme kertaa tehokkaammin kuin alin neljännes (Strålin ym., 2016, s. 7).

Tämän opinnäytetyöt tavoitteena on tutkia ja kehittää menetelmiä ohjelmistokehittäjien taitojen arviointiin. Pyrkimyksenä on kehittää kattava järjestelmä, joka mahdollistaa ohjelmistokehittäjien taitotason seurannan ja kehittämisen eri osa-alueilla. Tavoitteena on, että järjestelmä antaa paremman ymmärryksen kehittäjän vahvuuksista ja kehityskohteista. Tämän perusteella kehittäjä olisi tietoinen siitä minkä osa-alueiden kehittämiseen kannattaa panostaa ja missä edistystä on tapahtunut.

1.1 Työn tavoitteet ja motivaatio

Tämän opinnäytetyön keskeisenä tavoitteena on tutkia ja arvioida, kuinka ohjelmistokehittäjän taitoja voidaan tehokkaasti mitata ja seurata. Tavoitteena on myös

perehtyä käytettävissä oleviin teknologisiin ratkaisuihin, joista voisi olla apua taitojen seurannassa. Näiden perusteella pyrkimyksenä on kehittää järjestelmä tai systeemi, jossa näitä mittausmenetelmiä käytetään ohjelmistokehittäjien osaamisen ja tehokkuuden seurantaan.

Yksi avaintekijä ohjelmistokehitysprojektien onnistumisessa on kehittäjien osaaminen. Tukemalla heidän taitojen kehittymistä organisaatiossa voidaan saavuttaa korkeampi tuottavuus, parempi tuotteen laatu sekä lisätä asiakastyytyväisyyttä. Automatisoiduilla mittausjärjestelmillä voidaan saada objektiivista tietoa kehittäjän suorituskyvystä ja samalla ohjata ja auttaa heidän ammatillista kasvuaan.

1.2 Tutkimuksen kohde ja merkitys

Tutkimuksen kohteena on ohjelmistokehittäjien taitojen arviointi ja seuranta. Erityisesti keskitytään siihen, miten ohjelmistokehittäjien taitoja voidaan mitata objektiivisesti ja tehokkaasti, sekä miten näitä mittauksista saatuja tietoja voidaan hyödyntää kehittäjien ammatillisen kasvun tukemisessa. Tämä sisältää sekä teknologiset, että oppimisen ja koulutuksen näkökulmat, mukaan lukien erilaisten mittausmenetelmien ja teknologioiden arviointi sekä niiden soveltuvuus eri tilanteisiin.

Tutkimuksen merkitys on monitahoinen. Ensinnäkin ohjelmistokehitys on ala, joka on voimakkaasti riippuvainen kehittäjien osaamisesta ja taidoista. Kehittäjien taitojen tehokas ja objektiivinen arviointi sekä kehityksen seuranta mahdollistavat paremman ymmärryksen heidän vahvuuksistaan ja kehityskohteistaan. Tämä voi johtaa parempaan tuottavuuteen, tuotteen laatuun sekä asiakastyytyväisyyteen. Toiseksi objektiiviset ja tehokkaat arviointimenetelmät voivat auttaa yrityksiä ja organisaatioita tunnistamaan koulutus- ja kehitystarpeita mikä puolestaan voi johtaa parempaan henkilöstön kehittymiseen ja sitoutumiseen. Kolmanneksi, kun organisaatiot ymmärtävät kehittäjien taitoja paremmin, ne voivat tehdä tietoisempia päätöksiä rekrytoinneista, tiimien muodostamisesta ja projektien resursoinnista.

Ohjelmistokehitys on nopeasti kehittyvä ala ja tämän vuoksi on tärkeää, että organisaatiot olisivat kykeneviä seuraamaan kehittäjien taitoja säännöllisesti sekä reaaliaikaisesti. Tämä tutkimus pyrkii tarjoamaan työkaluja ja menetelmiä, jotka auttavat organisaatioita vastaamaan näihin haasteisiin. Tutkimus on toteutettu tilaajayrityksen tarpeet huomioon ottaen sekä yrityksessä käytössä olevia teknologioita hyödyntäen. Tutkimuksen havainnot kuitenkin pätevät suurimmalta osin muihinkin kehittäjäorganisaatioihin.

2 Sanasto

- **Authentication** - Autentikointi: Prosessi, jossa tarkistetaan käyttäjän henkilöllisyys esimerkiksi käyttäjänimen ja salasanan avulla.
- **Authorization** - Autorisointi: Oikeuksien myöntäminen käyttäjälle tietyille järjestelmäresursseille tai toiminnoille.
- **Backend development** - Palvelinpuolen kehitys: Sovelluksen taustajärjestelmän kehittäminen, joka käsittelee tietokantojen hallintaa, palvelinlogiikkaa ja sovellusliittymiä (APIs).
- **Branch** - Haara: Versiohallinnassa käytettävä termi, joka viittaa kehitystyön haarautumiseen pääkehityslinjasta. Haara mahdollistaa kehittäjien työskennellä erillään pääkoodikannasta (yleensä kutsutaan "master"- tai "main"-haaraksi) tekemällä muutoksia, parannuksia tai kokeiluja ilman, että se vaikuttaa heti pääkoodikantaan. Kehityksen valmistuttua haarat voidaan yhdistää takaisin päähaaraan yhdistämispyynnön (merge request) avulla.
- **CI/CD** - Jatkuva Integrointi ja Jatkuva Toimitus: Kehityskäytäntöjä, joissa kehitystiimin tekemät muutokset integroidaan päähaaraan usein ja automaattisesti, ja sovelluksia toimitetaan tuotantoon automatisoidusti.
- **Cloud service** - Pilvipalvelu: Internetin kautta tarjottavat palvelut, jotka voivat sisältää tallennustilaa, laskentatehoa tai sovellusalustoja.
- **Container** - Kontti: Kevyt virtualisointiyksikkö, joka sisältää sovelluksen ja kaikki sen toiminnan kannalta tarpeelliset osat, kuten kirjastot ja riippuvuudet.
- **DevOps** - Kehitys ja toiminta: Kulttuuri- ja käytäntökokonaisuus, joka pyrkii yhdistämään ohjelmistokehityksen (Dev) ja tietotekniikan toiminnan (Ops) parantamaan järjestelmän toimitusnopeutta ja laatua.
- **DevOps Engineer** - DevOps-insinööri on IT-ammattilainen, joka yhdistää ohjelmistokehityksen ja järjestelmäoperaatiot parantaakseen tuotekehityksen nopeutta ja laatua, keskittyen automatisointiin, CI/CD-prosesseihin ja järjestelmänvalvontaan.
- **Frontend development** - Käyttöliittymäkehitys: Sovelluksen käyttäjän näkymän kehittäminen, joka sisältää käyttöliittymän suunnittelun ja toteutuksen.
- **Infra** - Infrastruktuuri: Tietotekniikan perusrakenteet, kuten palvelimet, verkkolaitteet ja muu laitteisto sekä ohjelmistot, jotka mahdollistavat tietojärjestelmien käytön.
- **Issue** - Ongelmanseuranta: Järjestelmässä raportoitujen ongelmien tai kehitysehdotusten hallintaan käytettävä toiminto.

- **Julkaisutaajuus** - Kuinka usein sovellus tai ohjelmiston päivitykset julkaistaan käyttäjille tai tuotantoympäristöön.
- **Lead Developer** - Pääkehittäjä johtaa teknisesti ohjelmistokehitystiimiä, suunnittelee järjestelmien rakennetta, tekee keskeisiä teknisiä päätöksiä ja mentoroi muita kehittäjiä.
- **Merge Request** - Yhdistämispyyntö: Koodin yhdistämisen pyyntö, jossa kehittäjä ehdottaa muutoksiaan integroitavaksi päähaaraan (master branch) tai muuhun kehityshaaraan.
- **Pipeline** - Putkilinja: Automaattinen prosessi, joka kuvaa vaiheet, joita koodin on kuljettava versionhallinnasta testauksen, rakentamisen ja mahdollisesti tuotantoon siirron kautta.
- **Repository** - Koodivarasto: Keskitetty säilytyspaikka ohjelmistoprojektin tai koodikirjastojen lähdekoodille, dokumentaatiolle ja muille projektiin liittyville tiedostoille. Koodivarasto mahdollistaa versionhallinnan, eli siinä voidaan seurata ja hallita eri kehitysvaiheiden muutoksia, tallentaa useita versioita tiedostoista ja mahdollistaa useiden kehittäjien yhteistyön saman koodikannan parissa. Tyypillisesti käytössä ovat järjestelmät kuten Git, ja palvelut kuten GitHub, GitLab tai Bitbucket tarjoavat koodivarastojen hosting-palveluita.

3 Tutkimusmenetelmät ja materiaalit

Taustatutkimuksen tekeminen ennen arviointiperusteiden laatimista on olennaisen tärkeää useista syistä. Se tarjoaa ymmärryksen jo olemassa olevista menetelmistä, mikä auttaa välttämään tarpeettoman työn ja mahdollistaa nykyisten menetelmien parannusten tekemisen. Samalla taustatutkimus tuo esille teorian ja käytännön väliset erot, ja auttaa tunnistamaan tehokkaimmat ja hyödyllisimmät menetelmät, jotka voivat toimia mallina uusille arviointiperusteille.

Taustatutkimus auttaa myös tunnistamaan nykyisten menetelmien ongelmakohdat, mikä välttää samojen virheiden toistamisen uudessa arviointimallissa. Lisäksi se tarjoaa laajemman kontekstin, joka mahdollistaa tietopohjaisten päätösten tekemisen ja auttaa ymmärtämään käyttäjien todelliset tarpeet. Taustatutkimus tarjoaa mahdollisuuden myös benchmarkingiin eli vertailuun muiden menetelmien kanssa, joka auttaa asettamaan tavoitteet uudelle arviointimallille.

Taustatutkimuksen avulla voidaan hahmottaa myös tulevia suuntauksia ja muutoksia ohjelmistoteollisuudessa, jotka ovat tärkeitä uusien arviointiperusteiden luomisessa. Se

auttaa myös ymmärtämään arviointiin liittyviä eettisiä kysymyksiä ja miten ne voidaan ottaa huomioon uusien perusteiden luomisessa. Kaiken kaikkiaan, taustatutkimus tarjoaa laajemman ymmärryksen siitä, kuinka arviointiperusteet vaikuttavat ohjelmistokehitysprojektien laatuun, tiimien toimintaan ja ohjelmistoteollisuuden trendeihin.

3.1 Kyselytutkimuksen toteutus

Kyselytutkimus kohdistettiin omassa työyhteisössä työskenteleville kehittäjille. Kysely keskittyy ohjelmistokehittäjien taustoihin, heidän osaamisensa kehittämiseen ja siihen liittyviin haasteisiin sekä toiveisiin. Se pyrkii ymmärtämään kehittäjien koulutustaustaa, työhistoriaa ja nykyistä työnkuvaa. Erityisen tärkeää on selvittää, miten kehittäjät arvioivat omaa osaamistaan, mitä mittareita he käyttävät ja kuinka he näkevät osaamisen kehittämisen tärkeyden. Lisäksi kyselyssä etsitään tietoa kehittäjien haasteista ja toiveista työkalujen suhteen, jotka auttaisivat heitä seuraamaan ja mittaamaan osaamistaan sekä kehittymään alallaan.

3.2 Kyselytutkimuksen tulokset

Kysyttäessä työkokemuksesta ohjelmistoalalla, vastaukset vaihtelivat 5–6 vuodesta 15 vuoteen. On tärkeää huomata tämä hajonta, sillä se heijastaa eri ohjelmistokehittäjien erilaista kokemuspohjaa. Työnkuvien suhteen vastaukset viittaavat siihen, että kehittäjät ovat erikoistuneet erilaisiin tehtäviin ohjelmistoalalla. Työtehtävistä oli mainittu muun muassa DevOps engineer sekä Lead Developer. Tämä korostaa tarvetta yksilöllisille kehitysreiteille ja mahdollisesti yksilöllisille mittareille. Koulutustaustan osalta vastaukset vaihtelevat ammatillisesta koulutuksesta yliopistokoulutukseen. Tämä heijastaa monimuotoisuutta koulutustaustoissa ja vahvistaa ajatusta, että osaamisen kehittäminen ei rajoitu vain tiettyihin oppimispolkuihin.

Kyselyn perusteella kehittäjille ei ole tarjolla standardoituja tapoja mitata omaa taitotasoaan. Useimmat kehittäjät turvautuvat henkilökohtaisiin ja epävirallisiin tapoihin, kuten virheistä oppimiseen tai työsuorituksen itsearviointiin. Tämä viittaa siihen, että tarvitaan enemmän strukturoidumpia tapoja seurata ja arvioida osaamisen kehittymistä. Kaikki vastaajat pitivät osaamisen kehittämistä erittäin tärkeänä, korostaen ohjelmistokehitysalan jatkuvaa muutosta ja tarvetta pysyä ajan tasalla. Koodin laadun arvioimiseksi käytetään sekä itsearviointia että vertaisarviointia. Vaikuttaa siltä, että laadun mittaamisen työkalut ja käytännöt vaihtelevat yksilöiden välillä.

Vastaajat toivoivat erilaisia mittareita ja työkaluja osaamistasonsa seuraamiseen. Näitä olivat esimerkiksi suunnittelun, tuottavuuden ja ylläpidettävyyden seuraaminen. Tässäkin yhteydessä mittareiden tulisi huomioida laatu ja tehokkuus. Vastaajat kokivat haasteita osaamisensa kehittämisessä ja laadun parantamisessa, joista yksi yleinen oli ajanpuute. Aikataulupaineet näyttävät olevan merkittävä este oppimiselle ja itsensä kehittämiselle.

Ohjelmoijan osaamistason arvioinnin uskottiin vaikuttavan positiivisesti motivaatioon ja kehitykseen, kunhan se on tehty oikein. Kuitenkin arvioinnin väärinkäytön seuraukset saattavat olla negatiiviset.

Vastaajat toivoivat saavansa rehellistä, perusteltua ja rakentavaa palautetta työstään. Lisäksi he pitivät tärkeänä sitä, että palaute annetaan oikealla tavalla. Osaamisen seurantajärjestelmän uskottiin parantavan työpaikan tehokkuutta ja kilpailukykyä, kunhan se on suunniteltu ja toteutettu oikein. Tällainen järjestelmä saattaa tuoda mukanaan uudenlaisia haasteita, mutta se voi myös auttaa identifioimaan kehityskohteet ja auttaa yksilöitä kehittymään omassa työssään.

4 Taitojen mittaamisen merkitys

Mittaroinnin tarkoitus ei ole lisätä kenenkään työkuormaa vaan kehittää järjestelmä, joka toimisi mahdollisimman automaattisesti niiltä osin kuin se on järkevää ja mahdollista. Automaattiset mittarit mahdollistavat reaaliaikaisen seurannan sekä objektiivisen tiedon keräämisen, eikä ihmisten subjektiiviset havainnot tai muistikuvat pääse vaikuttamaan lopputulokseen. Tämä parantaa mittausten luotettavuutta ja tehokkuutta. Automaattisesti kerätty tieto on myös vertailukelpoista samalla lailla kerätyn tiedon kanssa.
Pitää muistaa, että automaattisesti kerättyyn tietoon liittyy myös haasteita. Mittareiden tulisi olla hyvin suunniteltuja, jotta ne tuottavat relevanttia tietoa. Myös tietosuojakäytännöt pitää olla kunnossa ja henkilöstön täytyy ymmärtää, miksi mittauksia tehdään ja mihin niitä hyödynnetään.

4.1 Mittaamisen tarkoitus ja tavoite

Mittaamisen perimmäinen tarkoitus on tarjota selkeä ja objektiivinen kuva kehittäjien suorituskyvystä ja osaamisesta. Tämä antaa yksilölle mahdollisuuden ymmärtää paremmin omia vahvuuksia ja kehityskohteita sekä auttaa organisaatiota tunnistamaan koulutustarpeita paremmin sekä auttaa resurssien kohdentamiseen eri projektien välillä.

Tavoitteena on luoda järjestelmä, joka on sekä kattava, että joustava. Kattavuus tarkoittaa sitä, että mittaamme kaikki olennaiset osa-alueet, kun taas joustavuudella haetaan järjestelmän mukauttamista käytön aikana ilmeneviin tarpeisiin tai järjestelmän puutteisiin. On tärkeää, että mittaamista ei koeta rasitteena tai uhkana, vaan pikemminkin työkaluna, joka auttaa sekä yksilöitä, että organisaatiota kehittymään ja kasvamaan.

Mittaamisen tulee olla läpinäkyvää sekä mittaamisen perusteet ja metodologia tulee olla helposti ymmärrettävissä sekä saatavilla kaikille mittauksen osapuolille. Tämä edistää luottamusta järjestelmään sekä varmistaa, että mittaustuloksia käytetään rakentavasti ja eettisesti.

4.2 Mittaamisen haasteet asiantuntijatyössä

Asiantuntijatyön mittaamiseen liittyy useita haasteita, joista osa on yleisiä kaikille aloille ja osa erityisiä ohjelmistokehitykselle. Seuraavassa käsitellään näitä haasteita:

Subjektiivisuus: Asiantuntijatyön tuloksia on usein vaikea mitata objektiivisesti. Esimerkiksi ohjelmistokehittäjän työn laadun arviointi voi perustua moniin tekijöihin, kuten koodin selkeyteen, tehokkuuteen ja virheettömyyteen. Nämä tekijät voivat olla subjektiivisia ja riippua arvioijan näkökulmasta.

- Monimutkaisuus: Ohjelmistokehittäjän työ on monimutkaista ja sisältää useita eri osa-alueita. Yksittäisen mittarin luominen, joka kattaisi kaikki nämä osa-alueet, on haastavaa.
- Kvantitatiivisten ja kvalitatiivisten mittareiden yhdistäminen: Vaikka jotkin osa-alueet, kuten koodirivien määrä, voidaan mitata kvantitatiivisesti, monet tärkeät tekijät, kuten koodin laatu tai yhteistyökyky, ovat kvalitatiivisia. Näiden kahden tyyppisten mittareiden yhdistäminen yhdeksi kokonaisuudeksi voi olla haastavaa.
- Mittareiden väärinkäyttö: On riski, että mittareita käytetään väärin, esimerkiksi rankaisemaan tai palkitsemaan työntekijöitä ilman koko kontekstin ymmärtämistä.
- Muuttuva teknologia: Ohjelmistokehitys on ala, joka muuttuu jatkuvasti. Tämä tarkoittaa, että mittareiden on oltava joustavia ja niitä on päivitettävä säännöllisesti vastaamaan alan uusimpia trendejä ja teknologioita.
- Henkilökohtaiset erot: Jokainen ohjelmistokehittäjä on yksilö, jolla on omat vahvuutensa, heikkoutensa ja työskentelytapansa. Yhden koon mittarit eivät välttämättä sovi kaikille.
- Motivaation vaikutus: Jos ohjelmistokehittäjät tuntevat, että heitä arvioidaan liian tiukasti tai epäreilusti, se voi vaikuttaa heidän motivaatioonsa ja sitoutumiseensa.
- Tietosuoja ja eettiset kysymykset: Mittaamiseen liittyvät tietosuojakysymykset, kuten se, mitä tietoja kerätään, kuka pääsee niihin käsiksi ja miten niitä käytetään, ovat erittäin tärkeitä. Eettiset kysymykset, kuten yksityisyyden kunnioittaminen ja reilu kohtelu, ovat myös keskeisiä.

Yhteenvetona voidaan todeta, että vaikka ohjelmistokehittäjien taitojen mittaaminen on tärkeää ja voi tuoda monia hyötyjä, siihen liittyy myös monia haasteita. On tärkeää lähestyä mittaamista harkiten, ottaen huomioon sekä tekniset että inhimilliset tekijät, jotta saadaan aikaan tehokas, oikeudenmukainen ja motivoiva arviointijärjestelmä.

5 Työkalut ja teknologian osaamisen arvioinnissa

Ohjelmointi ja IT-alalla teknologinen maisema on jatkuvassa muutostilassa. Tämän myötä on yritysten olisi ensiarvoisen tärkeää etsiä tehokkaampia keinoja arvioida työntekijöiden teknistä osaamista. Monesti on tiedossa se mitä työntekijä osaa, kun hänet otetaan töihin yritykseen mutta on tärkeää myös ymmärtää, miten hän kehittyy ja minkälaista osaamista hän hankkii työuransa aikana. Osaamisen systemaattinen arviointi antaa mahdollisuuden yksilöiden vahvuuksien ja kehityskohteiden tunnistamiseen. Näin voidaan myös tunnistaa minkä osa-alueiden kehittämiseen yrityksessä kannattaa panostaa.

Osaamisen arviointi on kuitenkin monimutkainen tehtävä, ja siihen liittyy useita haasteita. Yksi keskeisimmistä haasteista on määritellä, mitä "osaaminen" itse asiassa tarkoittaa tietyssä kontekstissa, ja miten sitä voidaan mitata objektiivisesti ja oikeudenmukaisesti. Lisäksi on otettava huomioon, että eri työtehtävissä ja eri aloilla osaamisen määritelmät ja mittarit voivat vaihdella huomattavasti.

Tässä luvussa keskitymme erilaisiin työkaluihin ja teknologioihin, joita voidaan hyödyntää osaamisen arvioinnissa. Tarkastelemme sekä perinteisiä että uudempia lähestymistapoja, kuten itsearviointilomakkeita, osaamismatriisia ja automatisoituja analyysityökaluja, kuten GitLabin kaltaisia ohjelmistojen versionhallintatyökaluja. Käymme läpi näiden työkalujen ja teknologioiden vahvuuksia ja heikkouksia sekä pohdimme, miten niitä voidaan yhdistellä saavuttamaan kattava ja monipuolinen näkemys yksilön tai tiimin osaamisesta.

5.1 Osaamismatriisi

Osaamismatriisi on työkalu, jota käytetään systemaattisesti kartoittamaan ja arvioimaan yksilöiden tai tiimien osaamista eri aihealueilla. Se tarjoaa visuaalisen, helpon ymmärrettävän tavan esittää, millä tasolla taidot ovat eri aihealueilla (Beck, S. (2003) s.2). Osaamismatriisi on taulukkomuotoinen esitys, jossa yksilön tai tiimin taidot ja osaaminen jaotellaan eri kategorioihin. Kategorioille annetaan arvio tai taso, joka kuvastaa kyseisen osaamisalueen hallintaa (Beck, 2003, s. 2). Tasoja voidaan asettaa esimerkiksi numeraalisesti tai sanallisesti, esimerkiksi aloittelija, perustaso, edistynyt ja asiantuntija.

Osaamismatriisi tarjoaa selkeän ja jäsennellyn tavan esittää osaamista, mikä helpottaa sekä yksilön osaamisen tunnistamisessa että kehittämistarpeiden määrittelyssä (Beck, S. 2003 s.2). Kun osaamismatriisi on vapaasti ohjelmoijien käytössä, heidän on helppo itse asettaa

konkreettisia tavoitteita omalle kehitykselleen ja seurata tavoitteisiin pääsemistä (Beck, 2003 s. 2).

Sovelluskehittäjien taitojen mittaamisessa osaamismatriisin avulla voidaan tunnistaa sekä tekniset että pehmeät taidot. Teknisten taitojen lisäksi on tärkeää tunnistaa ja arvioida esimerkiksi tiimityöskentelytaidot, ongelmanratkaisukyky ja jatkuva oppiminen. Putzmeister, Inc. -yhtiön tapauksessa he käyttivät osaamismatriisia tunnistaakseen pitkäaikaisten asiantuntijoiden tiedot ja siirtääkseen ne nuoremmille työntekijöille (Beck, 2003, s. 4). Tämä osoittaa, että osaamismatriisin avulla voidaan paitsi tunnistaa ja kehittää yksilöiden taitoja, myös siirtää arvokasta tietoa organisaatiossa.

5.2 Osaamismatriisin toteutus

Osaamismatriisin toteutuksessa otin huomioon sekä tekniset että pehmeät taidot, jotka ovat olennaisia sovelluskehittäjille. Kuvassa 1 nähdään, että matriisi on jaettu useisiin osioihin, kuten "Development", "Frontend development", "Backend development", "DevOps and Infra", "Container Technologies and Orchestration", "Cloud Services", "Authentication and Authorization", "Collaboration and Teamwork" ja "Soft skills". Jokaisessa osiossa on erilaisia taitoja, jotka on listattu ja joihin kehittäjät voivat merkitä oman osaamisensa tason.

Osaamistasot on määritelty viiteen kategoriaan: "No experience", "Beginner", "Intermediate", "Expert" ja "Master". Jokaiselle tasolle on annettu selkeä kuvaus, joka auttaa kehittäjiä arvioimaan omaa osaamistaan objektiivisesti. Esimerkiksi "Beginner" -taso kuvaa henkilöä, jolla on perustiedot aiheesta, mutta vähän tai ei lainkaan käytännön kokemusta, kun taas "Master" -taso kuvaa henkilöä, jolla on erittäin syvällinen ymmärrys aiheesta, paljon käytännön kokemusta ja kyky opettaa muita.

Tämä osaamismatriisi on kaikkien kehittäjien vapaassa käytössä, mikä mahdollistaa sen, että jokainen voi itse päivittää ja seurata omaa osaamistaan. Matriisin arvot ovat näkyvissä kaikkien kehittäjien osalta kaikille, jotta kehittäjillä on myös mahdollisuus nähdä keneltä saa parhaiten apua tiettyihin osa-alueisiin. Matriisi toimii myös erinomaisena työkaluna kehityskeskustelujen pohjana. Esimiehet ja kehittäjät voivat yhdessä tarkastella matriisia, tunnistaa vahvuudet, kehittämiskohteet ja asettaa tavoitteita tulevalle kaudelle. Lisäksi matriisia voidaan käyttää koulutussuunnitelmien laatimisessa. Jos tietyllä osa-alueella havaitaan puutteita tai tarvetta syventää osaamista, voidaan suunnitella koulutuksia tai työpajoja kyseiseen aiheeseen liittyen. Resursoinnin näkökulmasta matriisi auttaa

tunnistamaan, ketkä kehittäjät ovat parhaiten soveltuvia tiettyihin projekteihin tai tehtäviin, perustuen heidän osaamiseensa ja erikoistumiseen.

Kuva 1. Anonymisoitu versio käyttöön otetusta osaamismatriisista.

SKILLS MATRIX			
Skill section	**Skill**	**Example developer**	**Another developer**
Development	Code quality and maintainability	Master	Expert
	Design and Architectural Practices	Expert	Intermediate
	Testing and Test Automation	Expert	Expert
	Performance Optimization	Expert	Intermediate
	Refactoring	Expert	Intermediate
Frontend development	User Interface Design and Implementation	Intermediate	Intermediate
	Responsive Design	Intermediate	Intermediate
	Client-side Performance Optimization	Intermediate	Intermediate
	User Experience (UX) Enhancement	Intermediate	Intermediate
	Vue / Nuxt / Vuetify	Intermediate	Intermediate
Backend development	API Design and Implementation	Expert	Expert
	Security	Expert	Intermediate
	Database Design and Optimization	Master	Intermediate
	Server-side Performance Optimization	Master	Intermediate
	Laravel	Master	Expert
DevOps and Infra	Continuous Integration and Continuous Deployment (CI/CD) Practices	Beginner	Beginner
	Infrastructure as Code	No experience	No experience
	Server and Network Management	Beginner	No experience
	Troubleshooting and Monitoring	Intermediate	Intermediate
Container Technologies and Orchestration	Container Management and Optimization	Beginner	Beginner
	Service Scaling and Management	Beginner	Beginner
	Network and Storage Management in Container Environments	Beginner	Beginner
Cloud Services	Cloud Service Management and Configuration (especially Google Cloud)	Beginner	Beginner
	Cloud Cost Management and Optimization	No experience	No experience
	Cloud Security and Privacy	No experience	Beginner
Authentication and Authorization	User Management and Access Rights	Master	Intermediate
	Single Sign-On (SSO) and Token-Based Authentication	Master	Intermediate
	Security Protocols and Practices	Beginner	Beginner
Collaboration and Teamwork	Code review	Expert	Expert
	Documentation	Expert	Expert
Soft skills	Problem solving	Master	Expert
	Time handling	Expert	Expert
	Leader skills	Intermediate	Intermediate
	Teaching and mentoring	Intermediate	Master

Skill levels	**Explanation**
No experience	No knowledge or experience of the subject.
Beginner	Basic knowledge of the subject, but little to no practical experience.
Intermediate	Good understanding of the subject and practical experience.
Expert	Deep understanding of the subject and extensive practical experience
Master	Profound understanding of the subject, extensive practical experience, and the ability to teach others

5.3 GitLabin hyödyntäminen osaamisen arvioinnissa

Nykyaikaisesta ohjelmistokehityksestä puhuttaessa työkalujen merkitystä ei voida sivuuttaa. Niiden avulla ohjelmistokehittäjät pystyvät työskennellä tehokkaammin, varmistaa työnsä laadun ja hallita koodia tehokkaammin. Modernissa ohjelmistokehityksessä kaikki toiminnot kulkevat jonkun järjestelmän läpi, jossa hallitaan ohjelmiston versioita ja tuotantoputkia ja tämä mahdollistaa monenlaisen tiedonkeruun ohjelmistokehittäjän toiminnoista. Yrityksessä, johon tämä tutkimus tehtiin, on käytössä GitLab mutta monia muitakin vaihtoehtoja tähän on, muun muassa GitHub, BitBucket ja OneDev muutamia mainitakseni (WeAreDevelopers, n.d,).

5.3.1 Mikä GitLab on?

GitLab on avoimen lähdekoodin ohjelmisto, joka yhdistää versiohallinnan CI/CD prosessit (Continuous Integration ja Continuous Deployment) ja muita ohjelmistokehityksen elinkaaren vaiheita yhdeksi yhtenäiseksi palveluksi. GitLabin perusominaisuudet ovat ilmaiseksi saatavilla mutta sen kaupalliset version tarjoavat yrityksille monia eri työkaluja ohjelmistokehityksen prosessien hallintaan. Kun näitä työkaluja hyödynnetään oikein, GitLabista voi saada arvokasta tietoa ohjelmoijien taitotasosta ja osaamisesta.

GitLab tarjoaa myös monipuolisia ominaisuuksia yhteistyöhön, kuten koodin tarkistusprosessit (Merge Requests), ongelmanseuranta (Issues) ja wikin, jotka helpottavat tiimien välistä kommunikaatiota ja dokumentaation ylläpitoa. GitLabin integroitu issue tracker mahdollistaa tehtävien seurannan ja priorisoinnin suoraan koodivaraston (repository) yhteydessä, mikä vähentää tarvetta käyttää erillisiä projektinhallintatyökaluja.

Käyttöliittymän kautta käyttäjät voivat hallita erilaisia asetuksia, kuten käyttöoikeuksia, yksityisiä tai julkisia projekteja ja webhookkeja, jotka mahdollistavat integraatioita kolmansien osapuolien sovelluksien kanssa. Webhookkien avulla voidaan esimerkiksi automatisoida tehtäviä tai saada ilmoituksia ulkoisiin järjestelmiin tietystä tapahtumasta GitLabissa.

GitLabin DevOps-alusta tukee myös konttiteknologiaa, kuten Dockeria, ja Kubernetes-integraatiota, mikä helpottaa sovellusten skaalausta ja hallintaa tuotantoympäristöissä. Tämä mahdollistaa jatkuvan integraation ja toimituksen prosessien optimoinnin, jolloin sovelluksia voidaan päivittää nopeasti ja luotettavasti. Lisäksi GitLab tarjoaa laajat raportointi- ja analytiikkatyökalut, jotka auttavat mittaamaan ja parantamaan ohjelmistokehityksen prosesseja. Näiden työkalujen avulla organisaatiot voivat seurata koodin laadun muutoksia, havaita pullonkauloja työvaiheissa ja arvioida koodin ylläpidettävyyttä.

Koulutus ja tuki ovat myös tärkeitä osia GitLab-yhteisön tarjonnassa. GitLab järjestää säännöllisesti webinaareja, työpajoja ja tarjoaa laajan valikoiman dokumentaatiota ja ohjeita, jotka auttavat sekä aloittelevia että kokeneita käyttäjiä hyödyntämään alustan täydet mahdollisuudet. GitLab on siis paljon enemmän kuin pelkkä koodivarasto; se on kattava alusta, joka tukee koko ohjelmistokehityksen elinkaarta ja auttaa tiimejä toimimaan tehokkaammin yhteen. Sen avulla yritykset voivat nopeuttaa tuotekehitystä, parantaa ohjelmistojen laatua ja tehostaa tiimien yhteistyötä.

5.3.2 GitLabin osa-alueet

GitLab on monipuolinen työkalu ohjelmistokehitykseen. Seuraavassa on listattu joitakin GitLabin tärkeimmistä ominaisuuksista.

- Versionhallinta: GitLabin sydän on Git-versionhallinta. Kehittäjät voivat luoda projekteja, hallita koodimuutoksia, luoda haarautumisia (branches) ja yhdistää muutoksia merge requestien avulla.
- Issue tracking: GitLabin avulla tiimit voivat seurata ohjelmistoissa havaittuja ongelmia, tehtäviä ja ominaisuuspyyntöjä. Se mahdollistaa myös priorisoinnin, aikataulutuksen ja keskustelun.
- CI/CD: Kehittäjät voivat määrittää automaattiset työnkulut koodinsa testaamiseksi ja toimittamiseksi. Tämä vähentää manuaalisen työn tarvetta ja parantaa koodin laatua.
- Koodin laadun seuranta: GitLab tarjoaa työkaluja staattiseen koodianalyysiin, koodikattavuuteen ja turvallisuushaavoittuvuuksien havaitsemiseen.
- Wikit ja dokumentaatio: Projekteille voidaan luoda omat wiki-sivut ja dokumentaatio, joka auttaa tiimiä kommunikoimaan ja dokumentoimaan työnsä.
- Projektin hallinta: Työkaluja sprintsien, aikataulujen ja resurssien hallintaan, mukaan lukien Gantt-kaaviot ja burndown-kaaviot.
- Integraatiot: GitLabilla on laaja valikoima integraatioita muihin suosittuihin työkaluihin, kuten JIRA, Slack ja Kubernetes.

5.3.3 Arvovirta-analyysi

Gitlab, yhtenä johtavista DevOps-alustojen tarjoajista, esittelee ominaisuuden nimeltä Value Stream Analytics (VSA) joka voidaan suomentaa arvovirta-analyysiksi. Tämä ominaisuus on suunniteltu mittaamaan ja analysoimaan ohjelmistokehityksen eri vaiheita, auttaen tiimejä optimoimaan toimintojaan ja tehostamaan työnkulkujaan (Gitlab, n.d.). Opinnäytetyön kontekstissa VSA tarjoaa kriittistä tietoa, joka auttaa ymmärtämään pullonkauloja ohjelmistokehityksessä ja tunnistamaan mahdolliset parannuskohteet.

Esimerkiksi, yksittäisen kehittäjän työnkulun seuraaminen on elintärkeää projektien onnistumiselle. Oletetaan, että kehittäjä "A" on vastuussa useista tehtävistä tiimissään. VSA:n avulla voimme seurata hänen edistymistään tietyissä tehtävissä, määrittää kuinka kauan hänellä keskimäärin kestää siirtää tehtävä suunnitteluvaiheesta testausvaiheeseen ja lopulta tuotantoon. Jos huomaamme, että muutosten läpimenoaika on huomattavasti pidempi

kehittäjälle "A" verrattuna muihin tiimin jäseniin, se voi viitata siihen, että hän kohtaa esteitä tai tarvitsee lisäkoulutusta tietyissä osa-alueissa (Gitlab, n.d.).

Tämän lisäksi, VSA-työkalun avulla voidaan tarkastella erilaisia metriikoita kuten julkaisutaajuus, joka kertoo, kuinka usein sovelluksia tai muutoksia lähetetään tuotantoympäristöön. Tämän metriikan avulla voidaan tunnistaa, jos kehittäjän "A" tekemien muutosten määrä vähenee yhtäkkiä merkittävästi – tämä voi olla merkki ongelmista tai pullonkauloista hänen työssään (Gitlab, n.d.).

Yhteenvetona, Gitlab Value Stream Analytics ei ainoastaan tarjoa kattavaa näkymää ohjelmistokehitysprosessiin koko tiimin tasolla, vaan myös yksittäisen kehittäjän työnkulun analysointiin. Tämä tekee siitä erittäin arvokkaan työkalun opinnäytetyölle, joka keskittyy ohjelmistokehityksen parannuskohteiden tunnistamiseen ja tehokkuuden optimointiin yksilötasolla.

Kun puhumme ohjelmistokehityksestä, yksi keskeinen näkökulma on varmistaa, että todellinen kehitysprosessi vastaa organisaation määrittelemää ja suunnittelemaa arvovirtaa. Arvovirta määrittää sen, miten arvoa luodaan ja toimitetaan asiakkaalle, alkaen ideasta aina lopulliseen tuotteeseen tai palveluun. Kehitystavan on siis seurattava tätä määritettyä polkua tarkasti, jotta voidaan varmistaa, että asiakkaalle tuotettu arvo on mahdollisimman suuri ja prosessi on tehokas. Gitlab Value Stream Analytics (VSA) tarjoaa mahdollisuuden räätälöidä arvovirta-analyysi juuri yrityksen tai tiimin tarpeisiin sopivaksi. Sen sijaan, että yrityksen olisi mukauduttava tiettyyn ennalta määritettyyn arvovirtamalliin, VSA mahdollistaa arvovirran mukauttamisen niin, että se heijastaa tarkasti yrityksen omaa kehitysprosessia (Gitlab, n.d.). Tämä on erittäin arvokasta, sillä jokainen organisaatio on ainutlaatuinen, ja se, mikä toimii yhdelle yritykselle, ei välttämättä toimi toiselle.

Mukauttamalla arvovirta-analyysi vastaamaan tarkasti yrityksen työnkulkua, voidaan tunnistaa pullonkaulat, tehdä vertailuanalyyseja ja saada parempi ymmärrys siitä, miten resursseja voi kohdentaa tehokkaammin. Esimerkiksi, jos yritys on ottanut käyttöön uuden kehitysmenetelmän tai työkalun, VSA:n avulla voidaan analysoida, miten tämä muutos vaikuttaa arvon tuottamiseen ja kuinka se istuu osaksi olemassa olevaa arvovirtaa.

Lopulta, keskeinen tavoite on varmistaa, että kehitysputki vastaa määriteltyä arvovirtaa. Kun nämä kaksi elementtiä ovat linjassa, organisaation on helpompi tehostaa prosessejaan, vastata nopeammin markkinoiden muutoksiin ja tuottaa korkealaatuista arvoa asiakkailleen.

5.3.4 GitLabin tarjoama rajapinta

GitLab ei rajoitu pelkästään versiohallintatyökaluksi tai selaimen kautta käytettäväksi ohjelmistoksi vaan se tarjoaa myös kattavat ohjelmointirajapinnan (API) jonka avulla voidaan automatisoida monia toimintoja ja kerätä dataa GitLabista. Tämä rajapinta antaa mahdollisuuden automaattisesti kerätä dataa ohjelmoijien osaamisesta. Rajapinnasta on olemassa kattava dokumentaatio, jonka avulla on helppo suunnitella mitä tietoja ohjelmoijasta voidaan kerätä.

Tämä rajapinta mahdollistaa skriptien ja automaatioiden luomisen ohjelmoijien kehityksen seuraamiseen. Seuraavassa osiossa esittelen muutaman tavan seurata kehittäjien työskentelyä ja kehittymistä rajapinnasta saatavan tiedon avulla.

5.4 Liittämispyyntöjen hyödyntäminen ohjelmoijan työn arvioinnissa

Ohjelmistokoodin kehittäminen kehityshaaroissa (branch) on yleinen tapa, jolla hallitaan usean ohjelmoijan työtä saman ohjelmiston parissa. Muutokseen liittyvä koodi on omassa kehityshaarassaan, josta se yhdistetään (merge) päähaaraan (master branch). Näin tiimi voi hallita ja organisoida koodia tehokkaasti. (Brent Shiestl https://www.perforce.com/blog/vcs/branching-definition-what-branch)

Liittämispyyntöä (merge request) käytetään lähdehaaran liittämiseen kohdehaaraan, joka yleensä on päähaara (main branch) (Gitlab, n.d.). Yleisesti ottaen voidaan ajatella, että yksi päähaaraan liitetty lähdehaara edustaa jotain uutta ominaisuutta tai selkeää kokonaisuutta, joka voidaan viedä päähaaraan ja sitä kautta pääohjelmaan. Yrityksessä, johon tätä tutkimusta tehdään, käytetään jatkuvan kehityksen periaatetta CI (Continuous integration) jossa koodia viedään tuotantoon päivittäin mahdollisimman pienissä osissa, jolloin asiakas saa välitöntä hyötyä tehdyistä ominaisuuksista ja korjauksista.

Liittymispyyntöjen määrä ja niihin liittyvät tiedot ovat arvokkaita metriikoita ohjelmoijan työn arvioinnissa. Ne eivät kerro ainoastaan ohjelmoijan aktiivisuudesta ja tuottavuudesta, vaan myös kyvystä ottaa vastaan palautetta ja tehdä yhteistyötä muiden kanssa. Liittämispyyntöjen määrällä voidaan seurata ohjelmoijan tuottavuuden kehitystä pitkällä aikavälillä, kunhan otetaan huomioon kaikki liittymispyyntöjen määrään ja tyyppiin liittyvät seikat.

Seuraavaksi käyn läpi erilaisia asioita, joita liittymispyyntöihin liittyvästä datasta voidaan selvittää.

Liittymispyyntöjen määrä

Ohjelmoijan aktiivisuutta voidaan mitata liittämispyyntöjen määrällä. Suuri määrä liittämispyyntöjä voi viitata aktiiviseen kehitystyöhön, mutta on tärkeää huomata, että määrä ei aina korreloi laadun kanssa. On mahdollista, että ohjelmoija luo useita pieniä liittämispyyntöjä, kun taas toinen ohjelmoija saattaa keskittyä suurempiin ja monimutkaisempiin muutoksiin. Tämän vuoksi olisi hyvä seurata myös muutosten kokoja.

Liittymispyynnön tyyppi

Tutkimuksen kohteena olevassa yrityksessä liittymispyynnöt on jaettu erilaisiin kategorioihin.

- Feature: Tämä muutos sisältää uusia testattuja ominaisuuksia olemassa olevaan järjestelmään.
- Bug: Tämä muutos sisältää bugikorjauksia järjestelmään. Korjaukset pyritään tekemään
- Hotfix: Nopea korjaus akuuttiin ongelmaan. Tätä seuraa yleensä liittymispyyntö, jossa ongelma testataan ja korjataan huolellisesti.
- Bump: Jonkun ohjelman riippuvuuden päivitys uudempaan versioon tai ohjelman repositoryn päivitys isäntärepositorysta. Tämä liittymispyyntö ei sisällä muita muutoksia.

Tutkittaessa liittämispyyntöjen määrää on hyvä huomioida myös niiden tyyppi.

Liittämispyyntöjen hyväksymisprosentti

Jotta voidaan tarkastella luotujen muutosten laatua, on hyvä tarkastella millä prosentilla liittymispyynnöt on hyväksytty ilman muutoksia. Korkea hyväksymisprosentti voi olla viesti siitä, ohjelmoija ymmärtää projektin vaatimukset ja standardit hyvin. Toisaalta ongelmat tällä alueella voivat kertoa siitä, että vaatimusten määrittelyssä voi olla ongelmia ja puutteita.

5.4.1 Datan kerääminen GitLabista ja sen analysointi

Tutkimuksessa käytettiin GitLabin rajapintaa keräämään dataa liittämispyynnöistä eri projekteista. Datakeruun toteutti Python-skripti, joka käytti requests-kirjastoa API-kyselyiden tekemiseen. Skripti on usein pienehkö koodin pala, jolla pyritään automatisoimaan jokapäiväisiä toimenpiteitä. Skripti haki kaikki projektit, joihin käyttäjä on liittynyt, ja keräsi näistä projekteista kaikki sulautetut liittämispyynnöt vuodesta 2018 alkaen.

Tutkin myös mahdollisuutta saada dataan mukaan liittämispyynnöillä olevan koodimuutosten määrän. GitLabing Merge Requests API ei tarjonnut tähän kuitenkaan mallia, jolla pyyntöjen määrä rajapintaan määrä pysyisi maltillisena ja datankeruu olisi luotettavaa. Muutosten määrän lisääminen dataan jätettiin tässä vaiheessa pois mutta mahdollisesti tulevaisuudessa mitattavaksi asiaksi. Aluksi päätin hakea datapaketin vuoden 2020 alusta jotta on mahdollista nähdä trendejä sekä datan visualisoinnin suunnittelu on helpompaa, kun dataa on heti vähän enemmän.

5.4.2 Koodin toiminta

Valitsin skriptin kirjoitukseen ohjelmointikieleksi Pythonin. Vaikka en sitä itse työssäni käytäkään on se ylemmän tason ohjelmointikielenä hyvin kätevä tämänkaltaiseen työhön. Käyttämässäni Linux Ubuntu käyttöjärjestelmässä Python skriptejä voi kätevästi ajaa suoraan komentoriviltä käsin eikä koodi vaadi kääntämistä.

Koodi alkaa määrittelemällä API-avaimen ja tarvittavat otsikot. Se asettaa myös aloitusvuoden, josta alkaen dataa kerätään. Koodi hakee ensin kaikki projektit ja niiden yksilölliset tunnukset (ID). Tämän jälkeen se käy läpi jokaisen projektin ja hakee kaikki sulautetut liittämispyynnöt (Kuva 2).

Liittämispyynnöt ryhmitellään käyttäjän, kuukauden ja haaran nimen etuliitteen mukaan. Ryhmitelty data tallennetaan CSV-tiedostoon, joka voidaan viedä Looker Studioon edelleen analysoitavaksi.

Kuva 2. Python-skripti liittymispyyntöjen hakemiseen Gitlabin rajapinnasta

```python
# Fetch merged merge requests from all projects after the start year
all_merged_requests = []

for project_id in all_project_ids:
    url = f"https://gitlab.com/api/v4/projects/{project_id}/merge_requests"
    params = {"state": "merged", "per_page": 100, "updated_after": start_timestamp}

    page = 1
    while True:
        params["page"] = page
        response = requests.get(url, headers=headers, params=params)

        if response.status_code == 200:
            merged_requests = response.json()
            if not merged_requests:  # No more merge requests found
                break

            for mr in merged_requests:
                additions = mr['additions']
                deletions = mr['deletions']

            all_merged_requests.extend(merged_requests)
            print(f"Fetched merged requests for project {project_id}, page {page}, total merged requests: {len(all_merged_requests)}")
            page += 1
        else:
            print(f"Error fetching merged requests for project {project_id}: ", response.status_code, response.text)
            break

print(f"\nTotal merged merge requests fetched: {len(all_merged_requests)}")
```

5.4.3 Datan visualisointi

Kerätty data tässä muodossa on hankala tulkita, joten etsin sopivaa työkalua datan visualisointiin. Työkaluksi valikoitui Google Looker Studio koska se on yrityksessä käytössä jo muutenkin ja käyttö on tuttua. Looker studiossa on helppo yhdistää erilaisia datalähteitä, visualisoida niitä sekä jakaa niitä näkyviin tiimille. (Google, n.d.).

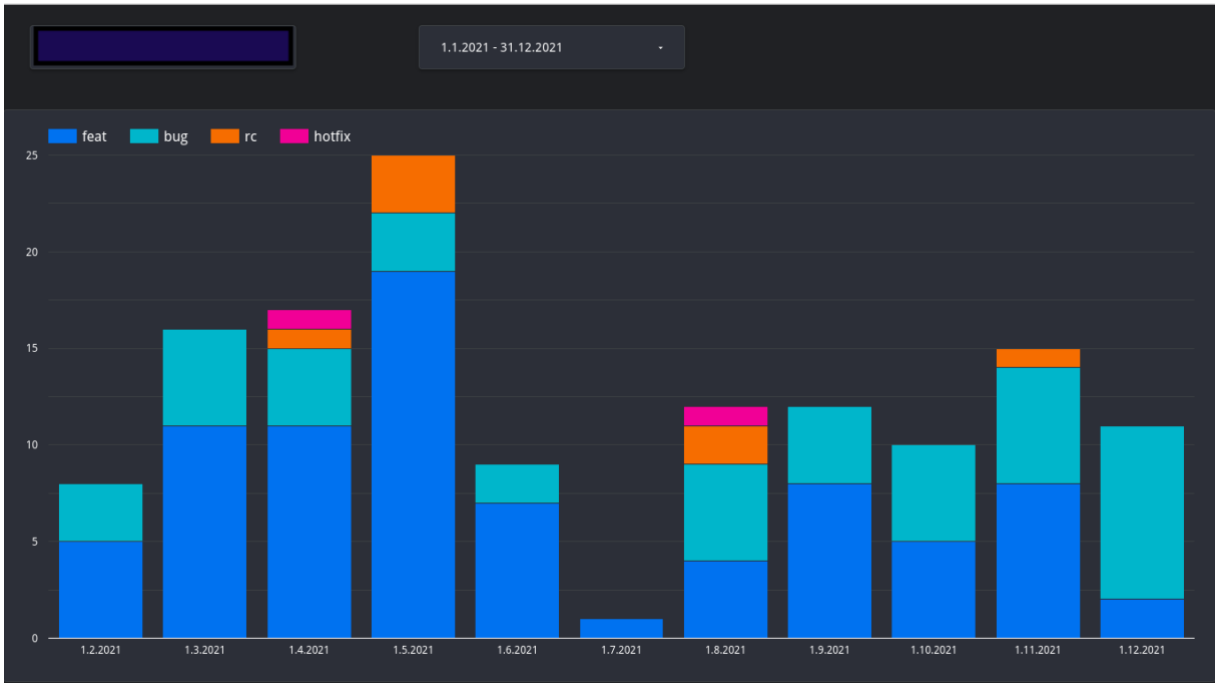
Valitsin kaavioksi pylväsdiagrammin, jossa jokainen pylväs edustaa siinä kuussa yhdistettyjä liittymispyyntöjä. Eri väreillä on eroteltu erityyppiset liittymispyynnöt toisistaan. Filttereiksi raportille asetin aikavälin sekä mahdollisuuden valita ohjelmoijat, joiden data raportissa esitetään (Kuva 3).

Raportin pääkohdat:

- Kuinka monta liittämispyyntöä kehittäjä on tehnyt kuukausittain.
- Millaisia haaroja on yleisimmin käytetty liittämispyynnöissä.
- Kuinka liittämispyyntöjen määrä on muuttunut ajan myötä.

Tämän analyysin avulla voidaan saada arvokasta tietoa kehitystiimin toiminnasta, kuten yksittäisen kehittäjän tuottavuudesta tai tiimin yleisistä käytännöistä.

Kuva 3. Looker Studion raportti liittymispyyntöjen yhdistämisestä päähaaraan



5.5 Aika-arvioiden ja toteumien seuranta

Ohjelmistoprojektien hallinnassa yksi keskeisimmistä haasteista on aikataulujen ja työmääräarvioiden tarkkuus. Kuten Helminen toteaa, työmääräarvioinnin ja aikataulusuunnittelun ongelma ei ole menetelmien puute, vaan niiden käyttö. Usein kohtaamme ongelmia, jotka johtuvat liian ylimalkaisesti tehdyistä suunnitelmista ja määrittelyistä. Kun nämä kaksi tekijää saadaan kehittymään, myös aikataulujen hallinta paranee (Helminen, 2008, s. 110).

Tämä opinnäytetyö keskittyy ohjelmistokehittäjien urakehityksen ja sen seurannan merkitykseen, erityisesti taitojen kehittämisen ja mittareiden soveltamisen näkökulmasta. Tavoitteena on luoda selkeä ja toimiva malli issueiden aika-arvioiden toteutumisen seurantaan, joka perustuu todellisiin tuntikirjauksiin ja työmääräarvioihin. Tämä malli mahdollistaa arvioiden ja toteutumien erojen seurannan Gitlabissa, ja auttaa näin ohjelmistokehittäjiä ja projektipäälliköitä paremman aikatauluhallinnan saavuttamisessa.

Koulutuksen merkitys korostuu myös Helmisen tutkimuksessa, jossa suurin osa vastaajista (12/16) oli saanut koulutusta projektiaikataulujen tai työmääräarvioinnin tekemiseen (Helminen, 2008, s. 102). Tämä viittaa siihen, että ammattitaito ja koulutus ovat avainasemassa tarkkojen ja realististen arvioiden tekemisessä.

Hyvät projektityökäytännöt, kuten aikataulun täsmennys, muutosten vaikutukset alkuperäiseen aikatauluun, arvion ja toteuman vertailu ja analysointi sekä toteumatiedon tilastointi seuraavia projekteja varten, ovat keskeisiä tekijöitä projektinhallinnassa (Helminen, 2008, s. 109). Tämä korostaa jatkuvan parantamisen ja oppimisen merkitystä projektinhallinnassa.

Lisäksi Nuutilan (2020) mukaan projektin edistymisen seurannan ja raportoinnin parantaminen on keskeistä projektin onnistumisen kannalta. Hänen työssään kehitettiin projektin edistymisen seurantatyökalu ja raportointipohja, jotka perustuvat laajasti käytettyihin projektin edistymisen mittaus-, seuranta- ja raportointikäytäntöihin (Nuutila, 2020, s. 3). Tämä työkalu ja raportointipohja voivat olla hyödyllisiä myös tämän opinnäytetyön yhteydessä, kun kehitetään mallia issueiden aika-arvioiden toteutumisen seurantaan.

5.5.1 Suunnitelma aika-arvioiden seuraamiseen

Suunnitelma alkaa vaatimuksella, että jokaiselle issuelle on tehtävä arvio kokonaistyömäärästä. Tämä auttaa varmistamaan, että projekti etenee suunnitellusti ja että resursseja on allokattu tehokkaasti. Tämän jälkeen käytetään Google Cloud Functions -palvelua luomaan komento, joka hakee tuntitiedot yrityksen ERP-rajapinnasta. Tiedot lisätään Gitlab APIa hyödyntäen issueille ja merge requesteille, mikä mahdollistaa tarkemman seurannan ja analysoinnin.

Merge requesteille merkattu tuntimäärä kumuloidaan myös issuelle, jolloin saadaan kokonaiskuva kunkin issuen työmäärästä. Tämä auttaa tunnistamaan mahdolliset pullonkaulat ja resurssien allokointiongelmat ajoissa. Lopuksi seurataan arvioiden ja toteutumien eroa Gitlabissa. Tämä antaa selkeän kuvan siitä, miten hyvin aika-arviot vastaavat todellisia työmääriä ja auttaa tekemään tarvittavia säätöjä projektin aikana.

Jotta aika-arvioiden ylitykseen johtavat syyt löydettäisiin luotiin sitä varten puuttumiskynnys taulukko. Tulevaisuuden suunnitelma on luoda järjestelmä, joka nostaa automaattisesti listalle tehtävät, joiden aika-arvio sekä toteuma johtaa johonkin toimenpiteeseen. Näin ongelmakohtiin voidaan puuttua ajoissa ja auttaa ohjelmoijaa eteenpäin toteutuksessa. Tässä kohtaa voidaan myös tarkastella alkuperäistä suunnitelmaa ja sitä onko aika-arviossa otettu huomioon kaikki tarvittava. Tarvittaessa tässä kohtaa voidaan myös informoida asiakasta viivästyksestä.

Parametrit	Toimenpide
Ylitysprosentti 0–25 %, työmäärä 0–4 h	Ei puututa
Ylitysprosentti 0–25 %, työmäärä 4–12 h	Seuranta
Ylitysprosentti 0–25 %, työmäärä > 12 h	Analyysi
Ylitysprosentti 25–60 %, työmäärä 0–4 h	Seuranta
Ylitysprosentti 25–60 %, työmäärä 4–12 h	Analyysi
Ylitysprosentti 25–60 %, työmäärä > 12 h	Puututaan
Ylitysprosentti > 60 %, työmäärä 0–4 h	Analyysi
Ylitysprosentti > 60 %, työmäärä 4–12 h	Puututaan
Ylitysprosentti > 60 %, työmäärä > 12 h	Puututaan

Vähäisiin ylityksiin pienehköissä tehtävissä ei ole tarvetta puuttua. Vähäisiin ylityksiin valittiin toimintatavaksi seuranta. Seurannalla tarkoitetaan sitä, että tehtävän kulkua pidetään silmällä mahdollisen viivästyksen lisäyksen osalta. Seuraavana puuttumistason on analyysi, joka tarkoittaa kohtuullista viivästymistä pienissä tehtävissä tai pientä viivästymistä isommissa tehtävissä. Tässä ei katsottu tarpeelliseksi eikä mahdolliseksi puuttua viivästymiseen tuntikirjauksien tekemisen viiveen vuoksi mutta tehtävän kulku on hyvä analysoida jälkeenpäin, jotta nähdään, oliko toteutuksessa tai suunnittelussa ongelmia. Huomattaviin ylityksiin arvoista isompien tehtävien osalta pyritään puuttumaan jo teko vaiheessa, jotta mahdolliset ongelmakohdat saadaan selville ja tehtävän läpivienti ei viivästyisi entisestään.

Kehittäjän ohjeisiin määriteltiin myös se, että jos toteutukseen kuluva aika näyttää ylittyvän siihen on hyvä puuttua kehittäjän itsensä toimesta ajoissa jo ennen määräajan ylittymistä. Tässä kohtaa voidaan tarkentaa tehtävän kuvausta ja päivittää aika-arviota, jos suunnitelma ei ole ottanut huomioon kaikkia vaadittavia riippuvuuksia.

5.5.2 Hyödyt ja merkitys

Tämä suunnitelma mahdollistaa aika-arvioiden ja todellisen työmäärän tarkan seurannan, mikä on avainasemassa projektien onnistuneessa hallinnassa. Se auttaa organisaatioita varmistamaan, että projektit pysyvät aikataulussa ja budjetissa, ja että resursseja käytetään tehokkaasti. Lisäksi se mahdollistaa nopeat korjausliikkeet, jos projekti on vaarassa ylittää aikataulun tai budjetin sekä ohjaa oikeanlaisien aika-arvioiden tekoon. Lisäksi ohjelmointikehityksen tehokkuus paranee, kun ongelmiin puututaan mahdollisimman aikaisessa vaiheessa.

6 Ohjelmistokehittäjän urakehitys ja sen seuranta

Digitaalisen maailman nopea kehitys on vaikuttanut merkittävästi ohjelmistokehitysalan dynamiikkaan. Kuten aiemmin mainittiin, ohjelmistokehittäjät edustavat laajaa ikä- ja kokemuspektriä. Kuten Parsons, Susnjak ja Mathrani (2015) toteavat, ”Ohjelmistokehitysyhteisöissä on laaja skaala eri ikäisiä ja erilaisen kokemuksen omaavia ihmisiä.” (Parsons, ym., 2015, s. 1). Tämä monimuotoisuus, yhdistettynä digitaalisen maailman jatkuvaan evoluutioon, tuo mukanaan sekä haasteita että mahdollisuuksia.

Ohjelmistokehittäjän ammatti on nyt tarpeeksi vanha kattamaan eri sukupolvet kokemuksineen ja taustoineen digitaalisen aikakauden valossa. Kehittäjän ura voi helposti kattaa useita vuosikymmeniä, ja tänä aikana ohjelmointikielet, työkalut ja kehitysprosessit voivat muuttua merkittävästi (Parsons, ym., 2015, s. 1). Tämä korostaa jatkuvan oppimisen ja sopeutumisen merkitystä alalla, joka on tunnettu nopeasta teknologisesta kehityksestään.

Tutkimuksessa kerätty data antaa oivalluksia globaalien ohjelmistokehittäjien profiileista ja auttaa ennakoimaan tulevia työskentelytapoja alalla digitaalisen maailman kontekstissa. Kyselyssä kartoitettiin muun muassa kehittäjien kokemusta, ohjelmointikielten taitoja ja ohjelmistokehityksen tyyppejä (Parsons, ym., 2015, s. 2). Tämä antaa arvokasta tietoa siitä, kuinka ohjelmistokehittäjien urat ja roolit voivat muuttua kokemuksen karttuessa.

Lisäksi on tärkeää huomata, että ohjelmistokehittäjien tuottavuuden ja ohjelmiston laadun parantamiseksi on keskityttävä ihmisiin (Graziotin, ym., 2014, s. 1). Ohjelmistokehitys on älyllinen toimintaa, jota hallitsevat usein laiminlyödyt inhimilliset tekijät. Ohjelmistokehittäjillä on oltava vahvat analyyttiset ongelmanratkaisutaidot ja luovuus ohjelmistojen rakentamisprosessissa. Tämä korostaa sitä, että ohjelmistokehittäjien tunne- ja mielialat

syvästi vaikuttavat heidän kognitiivisiin prosessointikykyihinsä ja suorituskykyynsä, mukaan lukien luovuus ja analyyttinen ongelmanratkaisu (Graziotin, ym., 2014 s. 1).

Yhteenvetona voidaan todeta, että ohjelmistokehittäjän urakehityksen ymmärtäminen ja seuranta digitaalisen maailman kontekstissa ovat keskeisiä tekijöitä, kun pyritään varmistamaan, että kehittäjät voivat mukautua ja kasvaa jatkuvasti muuttuvassa teknologiaympäristössä.

7 Mittareiden soveltaminen käytännössä

Koska kaikki mittarointi alkaa lähtötason selvittämisestä aluksi kaikkia ohjelmoijia pyydettiin arvioimaan oma taitotaso osaamismatriisiin. Tätä matriisia on tarkoitus jatkossa hyödyntää kehittäjien kehityskohteiden sekä mielenkiinnon kohteiden kartoittamiseen. Osaamismatriisia pidetään yllä säännöllisillä päivityskierroksilla joko itsearviointina tai yhdessä esimiehen kanssa.

Gitlabin tarjoamaa arvovirta-analyysia käytetään kehittäjien seurantaan eri projektien välillä. Kehitystapa ajetaan noudattamaan yhteisesti sovittua mallia, jotta arvovirta-analyysit ovat yhteneväisiä eri projektien välillä ja kehitystapa eri projektien välillä pysyy yhtenäisenä.

Liittymispyyntöjen määrää seurataan ohjelmistokehittäjän tuottavuuden seurantaan. Tässä mittarissa on kuitenkin otettava huomioon se, että liittymispyyntöjä on hyvin erilaisia ja eri kokoisia. Mittaria tuleekin tarkastella pitkien aikavälien trendien kautta eikä yksittäisten kuukausien tarkkuudella.

Aika-arvioiden ja toteumien seuranta on oleellinen mittari, kun seurataan yksittäisten tehtävien suunnittelun ja toteutuksen onnistumista. Ohjelmoijien ohjeistuksessa on määritelty, että jokaiseen tehtävään, joka on suunniteltu pitää olla asetettu aika-arvio. Tuntien takaisinkytkentää GitLabista ei tämän opinnäytetyön puitteissa tehty vielä mutta suunnitelma siitä on laadittu ja tarkoitus on ottaa se käyttöön. Ennen kuin tämä on toiminnassa, suoritetaan manuaalista seurantaa tuntien kertymisestä tehtäville. Jokainen tuntikirjaus on yhteydessä tehtävään, johon kirjaus liittyy, joten seuranta on mahdollista. Tämä mahdollistaa myös tuntien viemisen GitLabiin tulevaisuudessa, jotta arvio ja toteuma olisivat selvästi näkyvissä.

8 Johtopäätökset ja pohdinta

Tämän opinnäytetyön tavoitteena oli kehittää järjestelmä, joka helpottaa ohjelmistokehittäjien taitojen arviointia sekä seurantaa. Ohjelmistokehittäjä tuottaa työtä tehdessään paljon dataa, jonka perusteella seurantaa voidaan tehdä. Tämä mahdollistaa käytännössä rajattoman jatkokehityksen sopivien mittareiden löytämiseksi. On kuitenkin hyvä ottaa huomioon, että data, jota kerätään, on tasapuolista ja sitä tarkastellaan objektiivisesti ottaen huomioon datan asettamat rajoitukset. Ohjelmistokehittäjien taitotason kartoitus on avannut kehittäjille mitä erilaisia osa-alueita tässä työssä on mihin he eivät ole vielä paneutuneet sekä auttanut ymmärtämään yrityksen osaamisvajeita.

Pelkkään dataan luottaminen ohjelmistokehittäjän arvioinnissa ei ole järkevää eikä mahdollista. Data antaa jonkinlaista osviittaa mihin suuntaan sovelluskehittäjän taidot ja työkalupakki on kehittymässä mutta oikeaan arviointiin tarvitaan aina lopulta ihmistä, joka osaa ottaa huomioon työhön liittyvät olosuhteet ja projektit, joissa henkilö on ollut työskentelemässä.

Ohjelmistokehitys on nopeassa muutoksessa oleva ala ja tämä aiheuttaa haasteita myös mittaroinnille sekä ohjelmistokehittäjien osaamisen kehitykselle. Lisäksi nopeasti kehittyvässä yrityksessä työtavat ja tietolähteet voivat muuttua ja muuttuivat jo tämänkin tutkimuksen aikana. Tämän vuoksi mittarointijärjestelmä ei ole missään nimessä valmis paketti, vaan sitä tulee kehittää muun kehityksen mukana sekä tutkia uusia valmiita mittareita, joita yrityksen käyttämät järjestelmät julkaisevat ja tarjoavat.

Tutkimus rajoittui kohdeyrityksen työtapoihin ja dataan, jota oli saatavilla tutkimushetkellä. Kyselytutkimus rajoittui yrityksen työntekijöihin eikä laajempaa kyselytutkimusta aiheesta tehty. Tavoitteena oli kartoittaa kohdeyrityksen työntekijöiden taustoja ja työtapoja, vaikka otanta oli suhteellisen pieni, kehittäjät tulivat hyvin erilaisista taustoista ja erilaisilla työkokemuksilla ja työnkuvilla, joten otanta oli suhteellisen laaja niiltä osin.

Tätä opinnäytetyötä tehdessäni tekoäly on tehnyt suurta sisäänmarssia ohjelmistokehittäjien arkeen. Uusia ohjelmointikieliä ja teknologioita on jatkossa huomattavasti helpompi ottaa käyttöön tekoälysovellutusten avulla. Tekoäly auttaa paljon ihan perus ohjelmointityöskentelyssä ja jättää ohjelmoijalle enemmän aikaa miettiä koodin rakennetta, optimointia ja liiketoimintalogiikkaa. Tekoälyn käytön osaaminen tulee olemaan tulevaisuudessa avainasemassa, kun halutaan tuottaa ohjelmistoja mahdollisimman kustannustehokkaasti.

Lähteet

Beck, S. (2003). *Skill and Competence Management as a Base of an Integrated Personnel Development (IPD) - A Pilot Project in the Putzmeister, Inc./Germany. Journal of Universal Computer Science*, s. 2 & s.4.

Fonseca, L., Silva, M., Silva, S. ja Pereira, G. (2019). *Continuous-learning work environment: A study with developers in software development organizations. Knowledge Management & ELearning*, 11(3), 281–303. https://doi.org/10.34105/j.kmel.2019.11.015

GitLab. (11.8.2023). *About Gitlab*
https://about.gitlab.com/

Gitlab. (20.8.2023). *Merge requests*
https://docs.gitlab.com/ee/user/project/merge_requests/

Google. (10.12.2023). *Analytics-datan visualisointi Looker Studiossa*
https://support.google.com/analytics/answer/9849873?hl=fi

Graziotin, D., Wang, X., & Abrahamsson, P. (2014). *Happy software developers solve problems better: psychological measurements in empirical software engineering*. PeerJ, 2. https://doi.org/10.7717/peerj.289

Helminen, H. (2008). *Työmääräarviointi ja aikataulusuunnittelu IT-projekteissa.* [Pro gradu -tutkielma, Tampereen yliopisto]. https://urn.fi/urn:nbn:fi:uta-1-18763

Kim, A., Indeed (10.8.2023). *What Does a Software Developer Do? (Plus Salary and Skills)*
https://www.indeed.com/career-advice/careers/what-does-a-software-developer-do

Nuutila, L. (2020). *Projektien edistymäseurannan kehittäminen*. [Opinnäytetyö, Metropolia ammattikorkeakoulu] https://urn.fi/URN:NBN:fi:amk-202005138772

Parsons, D., Susnjak, T., ja Mathrani, A. (2015). *The Software Developer Cycle: Career demographics and the market clock: or, is SQL the new COBOL?*
https://doi.org/10.1145/2811681.2811698

Strålin, T., Gnanasambandam, C., Andén, P., Comella-Dorda, S. ja Burkacky, O. (2016). *Software development handbook – Transforming for the digital age.* Saatavissa: https://www.mckinsey.com/~/media/McKinsey/Industries/Technology%20Media%20and%20Telecommunications/High%20Tech/Our%20Insights/Software%20Development%20Handbook%20Transforming%20for%20the%20digital%20age/Software%20Development%20Handbook%20Transforming%20for%20the%20digital%20age.pdf

WeAreDevelopers (30.1.2024). *The Top 10 GitHub Alternatives* https://www.wearedevelopers.com/magazine/top-github-alternatives

Liite 1. Kyselytutkimus

Mikä on nykyinen työnkuvasi? *

Oma vastauksesi

Mikä on koulutustaustasi? *

Oma vastauksesi

Osaaminen ja sen arvioiminen sekä kehittäminen

Mitkä ovat nykyiset menetelmäsi seurata osaamistasoasi ja kehittymistäsi? *

Oma vastauksesi

Kuinka tärkeänä pidät oman osaamisen kehittämistä työssäsi? *

Oma vastauksesi

Miten arvioit koodisi laatua ja mitä työkaluja käytät siihen? *

Oma vastauksesi

Minkälaisia mittareita ja työkaluja toivoisit käytettävän oman osaamistasosi seuraamiseen? *

Oma vastauksesi

Millaisia haasteita koet oman ohjelmointiosaamisesi kehittämisessä ja työn laadun parantamisessa? *

Oma vastauksesi

Miten uskot ohjelmoijan osaamistason arvioimisen vaikuttavan ohjelmoijien motivaatioon ja kehittymiseen? *

Oma vastauksesi

Minkälaista palautetta toivoisit saavasi työstäsi, jotta voisit kehittyä ohjelmistokehittäjänä? *

Oma vastauksesi

Miten järjestelmä, joka seuraa ja arvioi ohjelmoijien osaamista, vaikuttaisi työpaikan tehokkuuteen ja kilpailukykyyn? *

Oma vastauksesi

Paljon kiitoksia vastauksista :)

Lähetä

Tyhjennä lomake