OULUN AMMATTIKORKEAKOULU

Tuomas Lipponen

HAJAUTETTU VERSIONHALLINTA TEOLLISUUSROBOTIIKASSA

Tutkimus versionhallinnan mahdollisuuksista

HAJAUTETTU VERSIONHALLINTA TEOLLISUUSROBOTIIKASSA

Tutkimus versionhallinnan mahdollisuuksista

Tuomas Lipponen
YAMK-opinnäytetyö
Syksy 2022
Robotiikan tutkinto-ohjelma
Oulun ammattikorkeakoulu

TIIVISTELMÄ

Oulun ammattikorkeakoulu
Tekniikan ylempi ammattikorkeakoulututkinto, robotiikan tutkinto-ohjelma

---

Tekijä: Tuomas Lipponen
Opinnäytetyön nimi: Hajautettu versionhallinta teollisuusrobotiikassa – Tutkimus versionhallinnan mahdollisuuksista
Työn ohjaaja: Esa Kontio
Työn valmistumislukukausi ja -vuosi: syksy 2022 Sivumäärä: 80 + 2 liitettä

Hajautettu versionhallinta teollisuusrobotiikassa on laadullinen tutkimus versionhallinnan mahdollisuuksista teollisuusrobotiikan alalla. Tutkimuksessa selvitettiin, onko versionhallinta käytössä robottiohjelmoijien tai -loppukäyttäjien työkaluna sekä millaisia kokemuksia ja ennakkoasenteita siihen liittyy. Tiedetään, että hajautettu versionhallinta on ollut avainasemassa tehokkaan ohjelmistokehityksen mahdollistajana. Tutkimusongelma oli selvittää, miksi teollisuusrobotiikassa versionhallinta ei ole niin suuressa merkityksessä tai niin suuresti esillä kuin muun ohjelmoinnin alalla.

Primääriaineisto kerättiin internetkyselyllä joulukuun 2021 ja tammikuun 2022 välisenä aikana hyödyntäen Webropol-kyselyalustaa. Kysely lähetettiin 17:lle sattumanvaraisesti valitulle Suomessa toimivalle teknologiateollisuuden yrityksen edustajalle, jotka ovat tekemisissä teollisuusrobotiikan kanssa. Kyselyyn vastasi 12 yrityksen edustajaa. Kyselyllä kysyttiin taustoittavien kysymysten jälkeen, onko yrityksellä käytössä versionhallintaa. Vastauksen perusteella esitettiin jatkokysymyksiä syistä versionhallinnan käyttämättömyyteen tai sitä, millaisena käyttö on koettu.

Tutkimusmenetelmänä käytettiin laadullista tutkimusmenetelmää, koska tiedonintressinä oli saavuttaa tietoa, joka auttaa ymmärtämään tutkittavan ilmiön merkitystä. Tutkimuksen heuristisella työotteella tavoiteltiin omista ennakkokäsityksistä luopumista ja mahdollisimman kokonaisvaltaista ilmiön ymmärtämistä. Aineiston analysoinnissa käytettiin metodeina luokittelua ja hermeneuttista analyysiä.

Kyselyn tulosten mukaan kaikille vastaajille versionhallinta oli vähintäänkin osittain tuttu käsite. Robottimerkillä ei ollut vaikutusta versionhallinnan käyttöön. Versionhallintaa hyödynnettiin suhteellisesti enemmän suurissa ja keskisuurissa yrityksissä kuin pienissä yrityksissä. Lähes kaikki versionhallintaa käyttävät kokivat, että työtavat toimivat paremmin, kun käytössä on versionhallinta. Missään yrityksessä ei koettu, etteikö versionhallinnasta olisi hyötyä.

Yksi syy versionhallinnan käyttämättömyydelle oli, ettei versionhallinta integroidu ohjelmointiympäristöön. Monipuoliselle integraatiolle ohjelmointiympäristön ja versionhallinnan välillä on tarvetta. Versionhallinnan käyttäjämäärä osoittautui taustaolettamusta suuremmaksi. Versionhallinnan käyttäjät ja nekin, jotka eivät käytä versionhallintaa, tiedostivat sen edut. Versionhallinnan käyttö teollisuusrobotiikassa on selvästi mahdollista. Koska versionhallinnan käyttö parantaa työtapoja, on sen käyttö kannattavaa työn tuottavuudenkin näkökulmasta. Robottimerkki ei aseta rajoitteita versionhallinnan käytölle, lukuun ottamatta yhteistyörobotteja. Muita selkeitä rajoittavia tekijöitä ei tutkimuksessa tullut esille.

Asiasanat: hajautettu versionhallinta, versionhallinta, teollisuusrobotti, robotiikka, automaatio

ABSTRACT

Oulu University of Applied Sciences
Master's degree programme, Robotics

Author: Tuomas Lipponen
Title of thesis: Distributed version control in industrial robotics — A research of the possibilities of version control
Supervisor: Esa Kontio
Term and year when the thesis was submitted: autumn 2022 Number of pages: 80 + 2 appendices

Distributed version control in industrial robotics is a qualitative research of the possibilities of version control in the field of industrial robotics. The goal of the research was to find out whether version control is used as a method by robot programmers or end users and what experiences and prejudices are related to it. Distributed version control has played a key role in software development. The research problem was to find out, why is version control not as important or prominent in industrial robotics as it is in the field of other programming.

The data was collected via an internet survey conducted between December 2021 and January 2022 using the Webropol survey platform. The survey was sent to seventeen randomly selected technology industry company representatives dealing with industrial robotics in Finland. Twelve company representatives took part in the survey. After background questions there was a question whether the company uses version control. Based on the answer, follow-up questions were asked about the reasons for not using version control or what kind of experiences they have about version control.

The research used the heuristic approach aiming comprehensive understanding of the phenomenon. Classification and hermeneutic analysis in the form of a hermeneutic circle were used as methods in the analysis of the data.

According to the survey results version control was at least partially familiar to all participants. The robot brand had no effect on the use of version control. Version control was more commonly used in large and medium-sized companies than in small companies. Almost all version control users felt that work methods are better when version control is used. None of the companies felt that version control was not useful.

One reason for not using version control was that version control is not integrated into the programming environment. Versatile integration between the programming environment and version control is needed. Even those who do not use version control recognize its benefits. Because the use of version control improves working methods, its use is also profitable regarding work productivity. Version control works in industrial robotics and the robot brand does not set any restrictions on the use of version control, except with collaborative robots. The study did not reveal any other clear limiting factors for the use of version control in industrial robotics.

Keywords: distributed version control, version control, industrial robotics, robotics, automation

SISÄLLYS

LYHENTEET

CVCS	Centralized Version Control System, keskitetty versionhallintajärjestelmä
CVS	Concurrent Versions System, yksi versionhallintajärjestelmä
DevOps	Software development and IT operations, ohjelmistokehitys ja IT-palvelutoiminnot
DVCS	Distributed Version Control System, hajautettu versionhallintajärjestelmä
PLC	Programmable Logic Controller, ohjelmoitava logiikka
RCS	Revision Control System, yksi versionhallintajärjestelmä
SCCS	Source Code Control System, yksi versionhallintajärjestelmä
SCM	Source Code Management, versionhallinta
SVN	Subversion, yksi versiohallintajärjestelmä
TFS	Team Foundation Server, versionhallintapalvelu, nykyään Azure DevOps Server
TFVC	Team Foundation Version Control, yksi keskitetty versionhallintajärjestelmä
VCS	Version Control System, versionhallintajärjestelmä

1 JOHDANTO

Teollisuusrobotiikan sovellukset ovat aiempaa modulaarisempia sekä tehokkaampia, ja ne tulee toteuttaa lyhyessä aikataulussa. Aiempaa vaativammat sovellukset yhä lyhyemmässä ajassa aiempaan verrattuna vaativat ohjelmointityöltä tehokkuutta ja mahdollisesti aiemmin tehdyn työn hyödyntämistä. Näiden tavoitteiden saavuttamisessa voivat auttaa oikeanlaiset työkalut, kuten versionhallinta.

Hajautettu versionhallinta on ohjelmistokehittäjän tärkeä voimavara (1, s. 1). Versionhallintaan on ollut saatavilla useita järjestelmiä jo 2000-luvun alkupuolelta saakka. Ohjelmistokehityksessä se on saavuttanut suuren suosion, ja ohjelmistokehittäjät käyttävät sitä työssään lähes poikkeuksetta. On aika selvittää, käyttävätkö teollisuusrobottien ohjelmoijat versionhallintaa ja olisiko teollisuusrobottien ohjelmointiin mahdollista tuoda versionhallinnan edut, joita ohjelmistokehittäjät ovat muissa ohjelmointiympäristöissä jo pitkään hyödyntäneet. Tästä ajatuksesta alkoi hahmottua tutkimuksen asiaongelma. Asiaongelmasta, tutkimusongelmasta ja siitä johdetuista tutkimuskysymyksistä on kerrottu tarkemmin luvussa 4.2.

Hajautettu versionhallinta teollisuusrobotiikassa on laadullinen tutkimus versionhallinnan mahdollisuuksista teollisuusrobotiikan alalla. Tutkimuksessa selvitetään, onko versionhallinta käytössä teollisuusrobottiohjelmoijien tai -loppukäyttäjien työkaluna ja millaisia kokemuksia tai ennakkoasenteita siihen liittyy. Tutkimuksessa selvitetään hajautetun versionhallinnan tarpeellisuutta ja hyötyjä teollisuusrobotiikassa sekä edellytyksiä sen käyttöön. Tuloksien perusteella voidaan päätellä, voiko hajautettu versionhallinta toimia teollisuusrobotiikassa ja millä reunaehdoilla kaikki robotiikan parissa työskentelevät voisivat siitä hyötyä. Tavoitteena on, että teollisuusrobotiikan ohjelmointiin voitaisiin saada näillä tiedoilla uutta tehokkuutta.

Työn tilaaja Algol Technics Oy toteuttaa teollisuuden robotisoidun automaation ratkaisuja ja on osa suomalaista Algol-konsernia. Algol Technicsillä uskotaan, että maailma automatisoituu, nopeutuu ja muuttuu aiempaa kompleksisemmaksi. Näihin muutoksiin Algol Technics tarjoaa tämänhetkiset ja tulevaisuuden ratkaisut. Teknologinen kyvykkyys perustuu tietoon, taitoon ja kokemukseen toteuttaa vaativia teollisuuden robotisoidun automaation kokonaistoimituksia. Hajautetun versionhallinnan merkitys ohjelmointityön apuvälineenä on tiedostettu, ja soveltuvien toimintatapojen sekä työnkulun vaikutus käytön sujuvuuteen on tunnistettu. Tavoitteena on, että moni voisi saada hyötyä

tämän tutkimuksen sisällöstä robotisoidun automaation viemiseksi entistä tehokkaampaan suuntaan.

1.1 Työn tausta

Versionhallinta teollisuusrobotiikassa on harvinaista. Siihen viittaa ainakin se, ettei aiheesta ole löytynyt aikaisempia tutkimuksia tai opinnäytetöitä. Tästä muodostui tutkimuksen taustaolettamus ja työhypoteesi, jota tutkimuksella haluttiin koetella. Kirjallisuutta itse versionhallinnasta on saatavilla runsaasti, mutta ei täsmällisesti kohdentuen teollisuusrobotiikkaan. Tietojenkäsittelyssä versionhallinta on ollut jo pitkään jokapäiväisessä käytössä. Ohjelmoitavien logiikoiden ja etenkin tietokoneohjelmien versionhallintaa on tutkittu. Koska teollisuusrobotiikan versionhallintaa ei ole juurikaan aiemmin tutkittu, tämän tutkimuksen yksi tavoite on tuottaa uutta tietoa aiheesta.

Tiedetään, että hajautettu versionhallinta on ollut avainasemassa tehokkaan ohjelmistokehityksen mahdollistajana. Tällä hetkellä vaikuttaa siltä, ettei teollisuusrobottien ohjelmoinnissa juurikaan hyödynnetä hajautettua versionhallintaa. Tutkimusta tarvitaan selvittämään syitä tälle ilmiölle ja toisaalta osoittamaan, mitä etuja robottien ohjelmointiin voitaisiin saada hajautetulla versionhallinnalla.

Jokaisessa robotisoidun automaation projektissa automaatio- ja robottisuunnittelu etenee omalla tavallaan. Jokaisella suunnittelijalla voi olla hieman erilaiset tyylit tehdä suunnittelu- ja ohjelmointityötä. Ohjelmointityön eteneminen ja seurattavuus pohjautuvat usein erillisiin dokumentteihin, joihin voidaan kirjoittaa historiaa siitä, mitä ohjelmaan on tehty. Usein ohjelmista tallennetaan eri versioita kansiorakenteisiin ja niille annetaan päiväyksen mukaisesti yksilöity nimi. Versionhallinnan tarkoituksena on mahdollistaa ohjelmakoodin muutosten seurattavuus: ilmaista, kuka teki muutoksen, mitä muutettiin, milloin se tehtiin ja minkä vuoksi. Versionhallinnasta voidaan jälkeenpäin helposti havaita mitä on muuttunut, mikäli laitteistossa esimerkiksi ilmenee ongelmia. Näistä versionhallinnan eduista voisivat hyötyä myös teollisuusrobotiikan ohjelmien suunnittelijat.

Ammatillinen historiani painottuu robotisoidun automaation asiantutijatehtäviin. Työkokemusta alan asiantuntijatehtävistä on hieman yli 9 vuotta, joista säännöllistä teollisuusrobottien ohjelmointia on ollut 7 vuotta. Versionhallinta käsitteenä on tullut tutuksi jo aivan asiantuntijatehtävissä saadun työ-

kokemuksen alkuvaiheessa. Kokemusta versionhallinnan käytöstä olen niin ikään saanut jo varhaisessa vaiheessa. Versionhallinnan hyödyntämistä olen kokeillut ohjelmoitavien logiikoiden ja teollisuusrobottiohjelmien hallintaan sekä todennut sen olevan tietyin rajoituksin mahdollista. Aikaisemmin rajoitteita oli logiikoiden ohjelmoinnissa siinä, että lähdekoodi oli binäärimuodossa. Tutkimusaiheen valintaan on vahvasti vaikuttanut aiempi tieto versionhallinnan mahdollisuuksista sekä kokemus robottien ohjelmoinnista. Kokemus vaikuttaa oleellisesti ajatteluun ja valintoihin, joita tutkimuksessa on tehty.

1.2 Tavoitteet

Versionhallinta on tarjonnut sovelluskehittäjille yhden tehokkaan menetelmän ketterään ohjelmistokehitykseen. Tässä tutkimuksessa selvitetään, hyödynnetäänkö versionhallintaa teollisuusrobotiikan yhteydessä, millaisia hyötyjä versionhallinnalla voidaan saavuttaa ja millaisena työskentely versionhallinnan kanssa ja ilman koetaan. Lisäksi tavoitteena on tuoda esille hyödyllisiä sovelluksia, jotka voivat helpottaa versionhallinnan hyödyntämistä teollisuusrobottiikassa.

Tutkimuksen kohteena ovat teollisuusrobotit ja niiden versionhallinta. Teollisuusrobottivalmistajia on maailmassa useita. Tutkimuksen teoreettisessa viitekehyksessä keskitytään neljään robottimerkkiin, jotka olivat kyselyyn vastanneiden tyypillisesti käyttämiä merkkejä. Hajautettu versionhallinta on yleisimmin käytössä oleva versionhallintatapa, ja siksi se painottuu tässä tutkimuksessa sekä valikoitui myös tutkimuksen otsikkoon. Se ei kuitenkaan rajaa tutkimuksen ulkopuolelle muita versionhallintatapoja, jotta tutkimusaineistoa olisi riittävästi.

1.3 Työn toteutus

Tutkimuksen viitekehys painottuu versionhallintaan ja robotisoituun automaatioon. Tietoa viitekehykseen ja tutkimuksen tekemiseen hankittiin kirjallisuudesta, internetistä saatavissa olevista konferenssimateriaaleista ja muista asiantuntijaorganisaatioiden internetlähteistä. Tutkimuksen aineisto kerättiin internetkyselyllä hyödyntäen Webropol-kyselyalustaa. Tietoa robottimerkkien ohjelmointiympäristöjen mahdollistamasta versionhallinnasta kerättiin viitekehykseen sähköpostilla robottivalmistajille lähetetyillä kyselyillä.

Ennen varsinaista aineiston analysointia aineistoa luokiteltiin, jotta vuoropuhelu aineiston kanssa helpottuisi. Aineiston analysoinnissa käytettiin metodeina luokittelua ja hermeneuttista analyysiä. Luokittelun ja analyysin avulla voitiin esittää tutkimuksen johtopäätökset ja tulkinnat tutkittavasta aiheesta. Varsinainen tieto ja tutkimuksen tulos ilmenee johtopäätöksien ja tulkinnan kautta.

1.4 Raportin rakenne

Johdannossa esitellään aihetta, taustaa ja tutkimuksen merkitystä. Siinä kerrotaan tutkimuksen tavoitteet, tarkoitus ja toteutus. Johdantoa seuraa viitekehyksen eli teoriataustan osuus, jossa selvitetään versionhallinnan perusteita, menetelmiä sekä eri versionhallintajärjestelmiä ja versionhallintapalveluita. Luvun 2 Versionhallinta jälkeen lukijalla on käsitys siitä, mitä versionhallinta tarkoittaa ja miten sitä voidaan käyttää. Luvussa 3 Robotisoitu automaatio kuvataan keskeisimmät käsitteet ja ohjelmoitavien logiikoiden versionhallinta. Luvussa myös kuvataan, millainen lähtötilanne neljällä eri robottimerkillä on versionhallinnan hyödyntämiseen sekä löytyykö ohjelmointiympäristöistä integraatiota versionhallintajärjestelmiin.

Tutkimuksen seikkaperäinen toteutus ja metodologia on selitetty luvussa 4 Tutkimuksen toteutus. Luvussa käsitellään tutkimustehtävät, aineiston keruu, aineiston analysointi, tutkimusetiikka, tutkimuksen pätevyys ja luotettavuus. Lukua seuraavat tutkimuksen tulokset.

Tutkimuksen tuloksissa esitellään ensin tiivistetysti analyysin tulokset. Analyysin tuloksien pohjalta esitellään tutkimuksen johtopäätökset ja tulkinnat tutkittavasta aiheesta. Näiden yhteyttä muuhun tutkimusaineistoon pohditaan ja argumentoidaan. Varsinainen tieto ja tutkimuksen tulos ilmenee johtopäätöksien ja tulkinnan kautta. Vastaukset tutkimuskysymyksiin tulevat esille johtopäätöksistä ja tulkinnasta. Yhteenveto kokoaa keskeisimmät tulokset ja johtopäätökset. Siinä arvioidaan valittujen menetelmien etuja ja heikkouksia, tutkimuksen yleistettävyyttä ja esitetään näkemyksiä jatkotutkimuksesta.

2 VERSIONHALLINTA

Versionhallinta on menetelmä järjestää tietoa niin, että eri versiot ovat erotettavissa toisistaan ja niiden historiaa on mahdollista seurata. Versionhallinta mahdollistaa palaamisen aiempaan versioon tarpeen vaatiessa. Tietojenkäsittelyssä huolimattomasti toimimalla tietty versio voi vahingossa korvaantua uudella versiolla. Ilman versionhallintaa palaaminen vanhaan voi olla vaivalloista. Versionhallinnassa versioinnin kohteena ovat tiedostot ja niiden sisältämä tieto. Tieto on tietokoneympäristössä tiedostoissa, ja versionhallinnan kannalta versiointi onnistuu kaiken tyyppisten tiedostojen kanssa. (2, s. 8.)

Tiedostojen tyypillä ja tiedoston sisältämällä tiedolla ei versionhallinnassa ole merkitystä. Kuitenkin jos versionhallinnasta halutaan saada paras hyöty, tulisi tiedoston sisältää luettavissa olevia merkkejä eikä käännettyä binääristä tietoa, joka ei ole ihmisen luettavissa. Esimerkkinä lukukelpoisesta tiedostosta on tekstitiedosto.txt ja binäärisestä tiedostosta sovellus.exe. Lukukelpoisien tiedostojen etuna on, että muutos voidaan esittää visuaalisesti vaikkapa koodirivin sisältämän käskyn vaihtamisesta toiseen (kuva 1).

```
$ git diff a434 2dc5
diff --git a/RAPID/TASK1/PROGMOD/MainModule.mod b/RAPID/TASK1/PROGM
.mod
index 64aeb03..e5f942a 100644
--- a/RAPID/TASK1/PROGMOD/MainModule.mod
+++ b/RAPID/TASK1/PROGMOD/MainModule.mod
@@ -10,7 +10,7 @@ MODULE MainModule
     FUNC num Calculate(num nNumerator, num ndenomerator)
             VAR num nresult:=0;

-            nresult:=nNumerator / ndenomerator;
+            IF ndenomerator<>0 nresult:=nNumerator / ndenomerator;

             RETURN nResult;
```

KUVA 1. Kahden eri version välillä muuttuneen tiedon vertailu diff-komennolla

Keskeisin ajatus nykyaikaisessa versionhallinnassa on muutosten seuranta. Jokaisen muutoksen tulisi aikaleiman lisäksi vastata kysymyksiin kuka, mitä ja miksi. Näitä tietoja sisältävää kommenttia yhdessä versionhallintaan tallennettavan muutoksen kanssa kutsutaan sanalla commit.

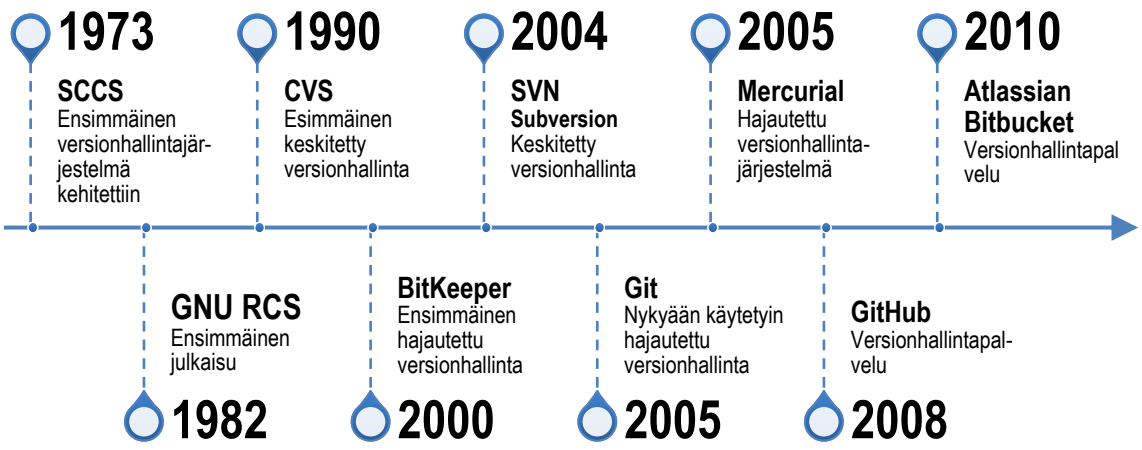
Ohjelmistoja on kehitetty useita vuosikymmeniä. Uudet teknologiat ovat tuottaneet mahdollisuuden ohjelmistojen kasvuun ja monipuolistumiseen. Kun lähdekoodin määrä alkoi kasvamaan, eri versioiden tarvitsema tallennustila alkoi muodostumaan moninkertaiseksi. Tallennustilaa tarvitsivat versiot, joihin oli korjattu jokin ongelma ja eri asiakkaiden eri versiot ohjelmasta. Kun oli aika selvittää, mistä löytyy juuri se versio ohjelmakoodista, johon tietty ongelma on korjattu tai miksi juuri tietynlainen muutos on tehty ohjelmaan, saattoi työ osoittautua haastavaksi. Näihin haasteisiin alettiin hakemaan ratkaisua lähdekoodin hallintajärjestelmästä (SCCS). (3, s. 364.)

Source Code Control System (SCCS) oli ensi askel kohti nykyaikaista versionhallintajärjestelmää, ja sen kehitys alkoi vuoden 1972 loppupuolella. Se suunniteltiin helpottamaan ohjelmakoodin muutosten hallintaa. Kaikki versiot ohjelmakoodista voitiin tallentaa yhteen tiedostoon. Eri versiolle yhteinen koodiosuus voitiin tallentaa kerran, ja täten vältyttiin saman koodin monistamiselta. Samaan aikaan pääsy kaikkiin eri versioihin säilyi. Järjestelmä merkitsi koodiosuuteen versionumeron, päiväyksen sekä tiedon siitä, kuka version oli tehnyt, mikä versiossa oli muuttunut aiempaan versioon verrattuna ja miksi muutos oli tehty. SCCS kehitettiin IBM 370 -keskustietokoneen käyttöjärjestelmässä käytettäväksi sekä PDP 11 -tietokoneelle UNIX-ympäristössä käytettäväksi. (3, s. 364, 369.)

Kun SCCS-järjestelmää verrataan nykypäivän versiohallintajärjestelmiin, huomataan selkeitä yhtäläisyyksiä. Perusperiaatteet ovat pysyneet samoina. Tarkoitus on helpottaa ohjelmakoodin muutosten seurantaa ja hallintaa. Nykyaikaiset versionhallintajärjestelmät tallentavat eri versioihin yksilöllisen numeron, päiväyksen ja tiedon siitä, kuka muutoksen teki sekä mitä muutettiin ja minkä vuoksi.

Yksi tunnetuista versionhallintajärjestelmistä on The Revision Control System (RCS), jota kehitetään edelleen. RCS on paranneltu versio edeltäjästään SCCS:stä. Sen jälkeen on kehitetty useita versionhallintajärjestelmiä. Usein syy uuden järjestelmän kehittämiseksi on ollut edeltävän järjestelmän puutteiden korjaus tai sen lisensoinnissa tapahtuneet muutokset. RCS kehitettiin hyödyntäen ajatuksia SCCS-järjestelmästä, ja Concurrent Version Control (CVS) kehitettiin täydentämään RCS-järjestelmän puutteita. Samoin Subversionin (SVN) kehityksen käynnisti tarve kehittää CVS-järjestelmässä havaittuja puutteita. BitKeeper oli aluksi ilmainen hajautettu versionhallintajärjestelmä avoimen lähdekoodin projekteille. BitKeeper kuitenkin muutti strategiaansa, ja palvelusta tuli maksullinen myös avoimen lähdekoodin projekteille. Muutos sai aikaan Gitin eli hajautetun versionhallintajärjestelmän kehittämisen. (2, s. 12; 4, s. 9; 5, s. 24.)

Kuvassa 2 on havainnollistettu eräiden keskeisimpien versionhallintajärjestelmien ja -palveluiden julkaisuja. Kuvasta voi havaita, että versionhallintaa on suunniteltu ja kehitetty jo vuosikymmeniä. Koska erilaisia versionhallintapalveluita on useita, kaikki eivät mahdu aikajanalle. Yksi mainitsemisen arvoinen aikajanalta puuttuva palvelu on vuonna 2011 aloitettu ja suuren suosion saavuttanut GitLab.



KUVA 2. Eräiden keskeisimpien versionhallintajärjestelmien ja -palveluiden aikajana (2; 3; 5; 6; 7; 8; 9; 10)

Seuraavaksi selvitetään, ketkä oikeastaan versionhallintaa käyttävät, mihin sitä käytetään, miten sitä käytetään sekä mitä eroa on versionhallinnalla ja revisionhallinnalla. Kuvan 3 kysymyksiin on annettu vastauksia seuraavissa kappaleissa. Kysymysten avulla on tuotu esiin seikkoja, jotka auttavat ymmärtämään lisää versionhallinnan perusperiaatteista.

KUVA 3. Kysymyksiä, joihin seuraavat kappaleet antavat vastauksia

Nykyään monet käyttävät eräänlaista versionhallintaa, jopa mahdollisesti tietämättään. Google Drive ja Microsoftin OneDrive tallentavat tiedostoista eri versioita, joihin on mahdollista palata, eli muodostavat tiedostojen versiohistorian (11; 12). Yleensä versionhallinnalla tarkoitetaan ohjelman lähdekoodin muutosten seurantaa ja hallintaa (5, s. 17). Näin ollen versionhallinnan tyypilliset käyttäjät ovat ohjelmoijia. Ohjelmistokehittäjien parissa versionhallinta onkin elintärkeä työkalu (1, s. 1).

Versionhallinta voi olla merkittävässä roolissa virheellisen koodin korjaamiseksi, kun ohjelmakoodissa havaitaan ongelma. Ongelmaa selvittävä voi seurata ohjelmaan tehtyjä muutoksia ja tarvittaessa palata edelliseen versioon todetakseen, esiintyykö virhe jonkin tietyn muutoksen jälkeen. Ongelmanratkaisu helpottuu, kun tunnetaan, mitä muutoksia ohjelmaan on tehty. Ilman versionhallintaa palaaminen edelliseen versioon voisi olla jopa mahdotonta. Mahdollista varmuuskopiota käyttämällä palaaminen tiettyyn ohjelmaversioon on mahdollista, mutta ohjelmaan tehdyt muutokset voivat olla tuntemattomia.

Versionhallinnan käyttämiseen on monia syitä, tai itse asiassa monet syyt ovat johtaneet versionhallinnan kehittämiseen ja käyttöönottamiseen. Esiin tulleet ongelmat on ratkaistu luomalla tai kehittämällä versionhallintajärjestelmä. Välttääkseen nämä todetut ongelmat voi apuna käyttää versionhallintaa. Aluksi versionhallinnalla pyrittiin ratkaisemaan, kuinka saada helposti selville, mitä tiedostossa on muuttunut, milloin ja miksi (3, s. 364). Seuraavan sukupolven versionhallinnalla ratkaistiin ongelma, kuinka työskennellä samojen tiedostojen kanssa yhtä aikaa. Tähän ratkaisuna oli

keskitetty versionhallintajärjestelmä (2, s. 10). Keskitetyn versionhallintajärjestelmän haasteena oli, että se vaati aina pääsyn palvelimelle, jolle versionhallinta oli keskitetty. Mikäli pääsy estyi syystä tai toisesta, työ saattoi keskeytyä (2, s. 10). Samaan tiedostoon tehtyjen muutosten yhdistäminen on yksi tärkeä ominaisuus versionhallinnassa. Se mahdollistaa ketterän yhteistyön samojen tiedostojen kanssa. Versionhallintaa käytetään, koska usean henkilön yhteistyön mahdollistaminen samojen tiedostojen kanssa on paljon helpompaa versionhallinnan kanssa kuin ilman sitä. Versionhallinnan tarkoitus on helpottaa ihmisten tekemää työtä.

Versionhallinnasta käytetään useita nimityksiä. Englannin kielellä käytössä esiintyvät muodot Source Code Management (SCM), Version Control System (VCS) ja Revision Control System. Kaikilla näillä tarkoitetaan usein samaa asiaa. Kirjassa The Source Development with CVS verrataan sanojen versio ja revisio eroja. CVS-versionhallintajärjestelmässä revisio on numero, jolla CVS yksilöi tiedoston ja siihen tehdyt muutokset. Kun useasta eri tiedostosta tehdään julkaisu, se saa versionumeron. Näin ollen kaikki versionhallinnassa olevat sisäiset yksilölliset muutokset ovat revisioita. Vaikka versionhallintajärjestelmä olisikin todellisuudessa revisionhallintajärjestelmä, kirjassa VCS:ää kutsutaan versionhallintajärjestelmäksi (Version Control System) eikä revisionhallintajärjestelmäksi (Revision Control System). Kirjoittaja perustelee valintaa sillä, että revisionhallintajärjestelmä kuulostaisi omituiselta ja päätyy siksi käyttämään versionhallinta-sanaa. (5, s. 44.)

Kehittäjä tai ohjelmoija ei käytännössä ole kiinnostunut siitä, minkä version kanssa hän työskentelee vaan, minkä revision. Revisio on tietty commit eli muuttuneet tiedostot yhdessä selityksen kanssa, jolle järjestelmä antaa tietyn revision. Versionhallintajärjestelmän historia antaa ohjelmoijalle kuvan siitä, mitä on tapahtunut, ja mahdollisuuden luoda oma haara tietystä revisiosta. Git-versionhallintajärjestelmässä tällainen on mahdollista. (13.)

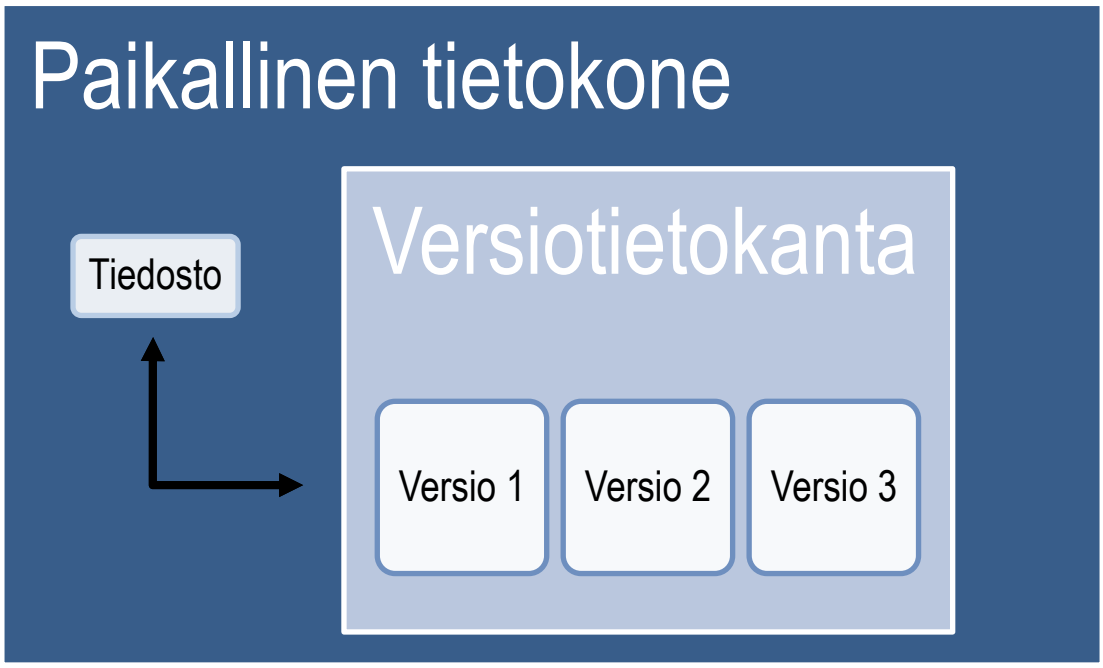
2.1 Versionhallintamenetelmät

Versiohallintamenetelmät ovat kehittyneet ajan saatossa. Ensimmäisten versionhallintajärjestelmien periaate oli mahdollistaa paluu edeltäneisiin tiedostoversioihin tiedosto kerrallaan. Yhdenaikainen työskentely tiedostojen kanssa ei ollut mahdollista. Versionhallinta oli paikallista. Seuraava sukupolvi versionhallintajärjestelmistä mahdollisti usean tiedoston yhdenaikaisen käsittelyn usean

tekijän toimesta. Versionhallinta tapahtui keskitettynä palvelimelle. Kolmannen sukupolven versionhallinnassa yhdistyivät edeltäneen sukupolven ominaisuudet hajautettuun strategiaan. Versionhallinta oli hajautettuna jokaiselle työasemalle pelkän keskitetyn palvelimen sijaan. (14.)

Paikallinen versionhallinta

Moni käyttää edelleen tapaa versioida tiedostoja yksinkertaisesti kopioimalla ne eri nimellä toiseen kansioon sekä mahdollisesti lisäämällä nimeen päiväyksen. Myös ennen versionhallintajärjestelmien kehitystä käytettiin versionhallintana tiedostojen nimeämistä eri versioiksi samalle työasemalle. Yksinkertaisuutensa vuoksi toimintatapa on vieläkin hyvin yleinen. Kuten jo aiemmin luvun 2 alussa mainittiin, eri tiedostoversiokopioiden tarvitsema tila alkoi tuottaa haasteita. Lisäksi tällainen menetelmä oli altis erehdyksille. Tärkeä versio saattoi ylikirjoittua toisella. Tämä epäkohta saikin aikaan ensimmäisten versionhallintajärjestelmien kehityksen, kuten SCCS:n, joka edustaa paikallisten versionhallintajärjestelmien (kuva 4) kategoriaa. (2, s. 8.)



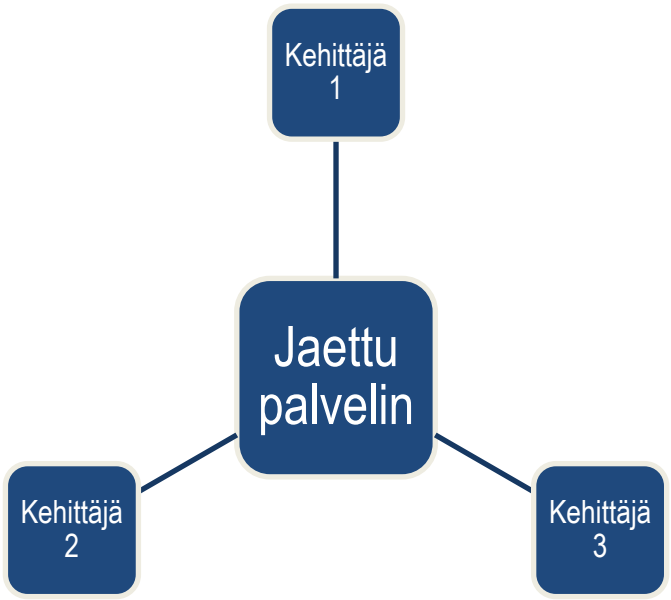
KUVA 4. Paikallinen versionhallinta, jossa kaikki toiminnot ovat yhdellä tietokoneella (mukaillen 2, s. 8)

Paikallisessa versionhallinnassa muutosten seuranta ja versiot ovat tallennettuina pelkästään paikalliselle työasemalle. Käytössä ei ole palvelinta, ja versionhallinta on suunniteltu pidettävän paikallisessa tietokannassa. Kun kaikki tieto on keskitetty yhdelle tietokoneelle, yhteistyö muiden kanssa voi osoittautua haasteelliseksi tai jopa lähes mahdottomaksi. Yhden tietokoneen versionhallinta on myös altis koko versionhallinnan menettämiselle, jos tietokone vaurioituu. (2, s. 8, 10.)

Paikallinen versionhallinta voi olla riittävä silloin, kun kyse on yhdestä yksittäisestä ohjelmoijasta. Versionhallinnan perusperiaatteet, kuten muutosten seuranta, toteutuvat myös paikallisessa versionhallinnassa. Yksittäinen ohjelmoija voi kuitenkin saavuttaa lisähyötyä muista versionhallintatavoista paikalliseen versionhallintaan verrattuna.

Keskitetty versionhallinta

Paikallinen versionhallinta toi lähinnä helpotusta versioiden seurantaan versiohistorian muodossa sekä tilanpuuteongelman ratkaisemisessa. Seuraava haaste ratkaistavaksi oli yhteistyön mahdollistaminen muiden tiedostojen käyttäjien kanssa (2, s. 9). Ratkaisuksi kehitettiin keskitetty versionhallinta eli Centralized Version Control Systems (CVCS). Keskitetyssä versionhallinnassa (kuva 5) on yksi palvelin, jolle jokainen kyseiselle projektille työskentelevä ohjelmoija lähettää versioitavat tiedostonsa (2, s. 9). Palvelin ja ohjelmoijien tietokoneet ovat toisistaan erillisiä laitteita ja voivat sijaita missä vain toisistaan riippumatta, kunhan verkkoyhteys palvelimeen saadaan muodostettua. Keskitetty versionhallinta on ollut vuosia versionhallinnan standardi (2, s. 9).



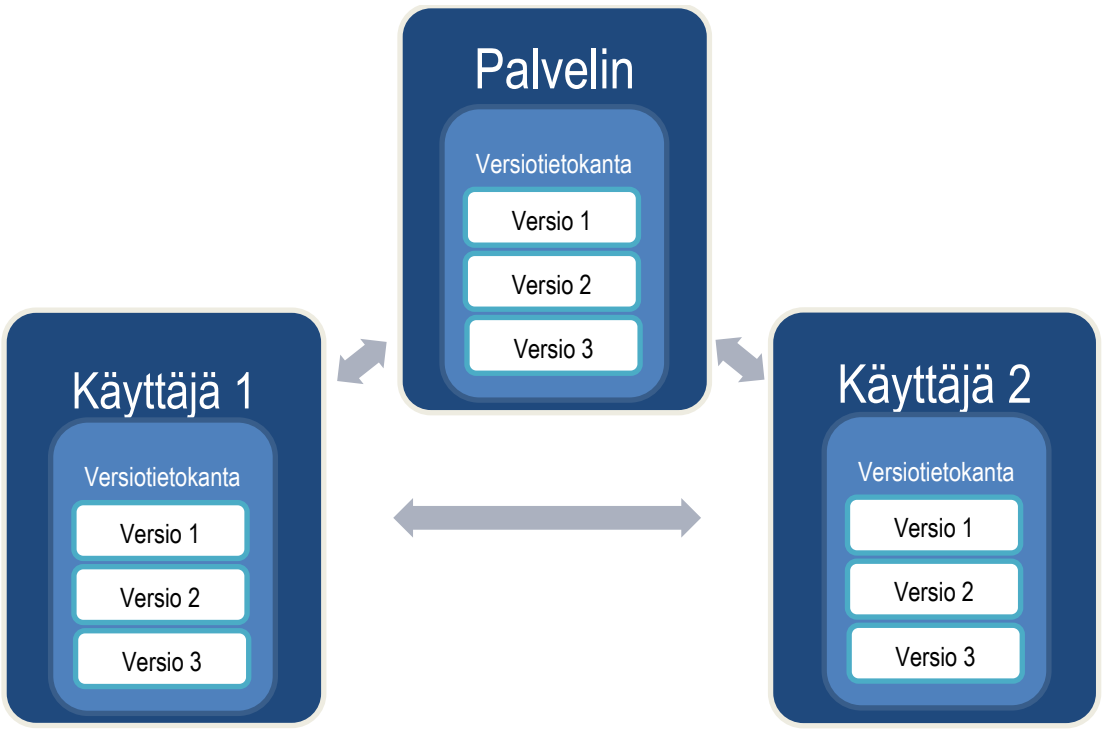
KUVA 5. Keskitetty versionhallinta, jossa ohjelmoijat siirtävät ja noutavat muutokset keskitettyyn palvelimeen (mukaillen 2, s. 10)

Paikalliseen versionhallintaan verrattuna keskitetty versionhallinta tuo monia etuja. Ohjelmoijat voivat saada tiedon, missä vaiheessa toiset ohjelmoijat ovat, ja työn edistymisen valvonta on helpompaa, kun tieto on yhdessä paikassa. Keskitetyssä versionhallinnassa on myös heikkoutensa. Yksi

selvimmistä on työn keskeytyminen, jos yhteys palvelimeen katkeaa. Keskitetty versionhallinta vaatii yhteyden palvelimeen sujuvan työskentelyn mahdollistamiseksi. Järjestelmä on myös hyvin haavoittuvainen. Mikäli palvelimen kiintolevy vaurioituu, koko versionhallintajärjestelmä ja kaikki sen tiedot voidaan menettää, koska palvelin on ainoa paikka, jossa tieto muutoksista on. Vastaava heikkous on myös paikallisessa versionhallinnassa. (2, s. 9.)

Hajautettu versionhallinta

Hajautettu versionhallinta (kuva 6) eli Distributed Version Control Systems (DVCS) on järjestelmä, jossa nimensä mukaisesti versionhallinta on hajautettuna eri tietokoneille. Hajautettu versionhallinta on ikään kuin yhdistelmä paikallista versionhallintaa ja keskitettyä versionhallintaa. Siinä jokaisella koneella on täydellinen kopio kaikesta tallennettavasta tiedosta (2, s. 10). Versionhallinta on suojattu hyvin tiedon katoamiselta, koska jokaisella ohjelmoijalla sekä palvelimella on täydellinen kopio tiedosta. Mikäli yhden käyttäjän versionhallinnan tiedot tuhoutuvat, hänelle on helppo palauttaa koko versiohistoria tietoineen toiselta käyttäjältä tai palvelimelta.



KUVA 6. Hajautettu versionhallinta, jossa jokaisella on oma versiotietokanta (mukaillen 2, s. 11)

Merkittävänä etuna on lisäksi se, että työtä voidaan tehdä myös ilman yhteyttä palvelimeen. Keskitetyssä versionhallinnassa tämä ei välttämättä ole mahdollista, tai työ ilman palvelinyhteyttä ei ole niin helppoa kuin hajautetussa versionhallinnassa. Ilman palvelinyhteyttä muutoksia ei voida

lisätä versionhallintaan, koska tietokanta sijaitsee palvelimella. Hajautettu versionhallinta mahdollistaa useita erilaisia työnkulkutapoja, jotka eivät ole toteutettavissa keskitetyssä versionhallinnassa. (2, s. 11.)

Hajautetun versionhallinnan keskeisimmät edut on koottu kuvaan 7. Muutosten seurannalla nähdään, miten ohjelmakoodi on kehittynyt ajan kuluessa. Toisin sanoen, se mahdollistaa koodin historian seuraamisen. Koska jokainen versionhallintaan tallennettu muutos on yksilö, voidaan siihen helposti palata tarpeen tullen. Näin ollen versioita voidaan helposti vertailla keskenään. Toinen ohjelmoija voi vertailla, mitä muutoksia muut saman ohjelmakoodin parissa työskentelevät ovat tehneet, tai tarkistaa, mitä muutoksia edelliseen versioon verrattuna on tehty. Hajautetun versionhallinnan ominaisuuksiin kuuluu, että sama koodipohja on palvelimen lisäksi kaikilla työasemilla, joilla ohjelmakoodia kehitetään. Koska sama koodipohja löytyy useasta paikasta, ei yksittäisellä työasemalla tai palvelimella tapahtunut ohjelmakoodin häviäminen kadota koko työhistoriaa ja koodia. Kaikki on palautettavissa ohjelmakoodin hävittäneelle koneelle toisen saman ohjelmakoodin parissa työskentelevän henkilön työasemalta. Koodipohjan hajautus jokaiselle ohjelmoijalle mahdollistaa ketterän ohjelmakoodin kehityksen usealla työasemalla yhtä aikaa. Ohjelmakoodiin tehdyt muutokset voidaan yhdistää palvelimelle, josta muutokset ovat kaikkien ohjelmoijien noudettavissa.

KUVA 7. Hajautetun versionhallinnan keskeisimmät edut

2.2 Versionhallintajärjestelmät

Luvun 2 kuvassa 2 esiteltiin eräiden keskeisimpien versionhallintajärjestelmien aikajana. Lähempää tarkastelua varten on poimittu tutkimusaineiston mukaan suosituimmat versionhallintajärjestelmät. Git osoittautui suosituimmaksi järjestelmäksi ja valikoitui tarkasteluun. SVN eli Subversion oli myös yritysten käytössä. Bitbucket mainittiin myös, ja koska kyseessä on versionhallintapalvelu, joka Gitin lisäksi on aiemmin tukenut myös Mercurial järjestelmää, valikoitui myös Mercurial viitekehyksen sisältöön.

2.2.1 Git

Git on avoimen lähdekoodin hajautettu versionhallintajärjestelmä, jonka ensimmäinen julkaisu oli vuonna 2005. Git kehitettiin korvaamaan maksulliseksi muuttunut BitKeeper-versionhallintajärjestelmä Linux-ytimen kehityksessä. Käytännössä ensimmäisen version Git-versionhallinnasta kehitti Linux-ytimen kuuluisa kehittäjä Linus Torvalds. Gitin tuli olla nopea, yksinkertainen, mahdollistaa epälineaarinen ohjelmistokehitys eli usean ominaisuuden yhtäaikainen kehitys, täysin hajautettu ja soveltuva suurten projektien hallintaan. Tähän kaikkeen Git pystyy ja on pystynyt myös kehittymään siinä entisestään. Gitistä on tullut maailman käytetyin moderni versionhallintajärjestelmä. (2, s. 12; 16.)

Suurin ero Gitin ja muiden versionhallintajärjestelmien välillä on Gitin tapa käsitellä tietoa. Muut versionhallintajärjestelmät tallentavat tiedot tiedostoperusteisesti eli tallennettava tieto on tiedostokokonaisuus. Yleisesti tätä kutsutaan deltaperusteiseksi versionhallinnaksi. Gitin tapa käsitellä tietoa on erilainen. Git tallentaa tiedot kuvakaappauksina tiedostojen sisällöstä eli tiedon siitä, miltä tiedostojen sisältö tallennushetkellä on näyttänyt. Ainoastaan muuttuneiden tiedostojen sisältö tallennetaan ja muihin seurattaviin tiedostoihin viitataan linkillä, jolloin saman sisältöistä dataa ei uudelleen tallenneta. (2, s. 12–13.)

Gitiä hyödynnetään myös teollisuusautomaation ohjelmointiympäristöissä. Omronin The Team Edition integroi Git-versionhallintajärjestelmän osaksi Sysmac Studio -ohjelmointiympäristöä. Team Editionin myötä koneohjainten ohjelmoinnissa päästään hyötymään kaikista Git-versionhallintajärjestelmän eduista, kuten hajautetusta rakenteesta, jossa jokainen ohjelmoija voi työskennellä itse-

näisesti oman paikallisen koodipohjan parissa ja yhdistää valmiit osat palvelimelle muiden kehittäjien saataville. Koneohjainten ohjelmien yhdenaikainen kehitys mahdollistuu, ja ohjelmaosien yhteen liittäminen on ketterää. (15.)

Gitin edut muihin hajautettuihin versionhallintajärjestelmiin verrattuna ovat edellä mainituissa Gitille alun alkaen asetetuissa vaatimuksissa ja Gitin tavassa käsitellä tietoa, mutta etenkin sen tehokkuudessa. Tietojen tallennus, kehityshaarojen luonti, haarojen yhdistäminen ja muutosten seuranta ovat ketteriä (16). Gitiä voidaan käyttää usealla eri tavalla, eli työnkulku on sovittavissa organisaation tarpeiden mukaisesti. Tämän ansiosta Git skaalautuu hyvin pieniin sekä suuriin projekteihin (16).

Vaikeinta Gitin käytössä lienee sen käytön oppiminen. Sujuva käyttö vaatii komentojen ymmärtämisen, ja käytettyjen termien toiminnallisuus saattaa poiketa muista versionhallintajärjestelmistä. Tämä saattaa aiheuttaa haasteita etenkin aloittelijoille. (16.)

Vaikka Gitin käyttö voi aluksi tuntua haastavalta, on sen opettelu vaivan arvoista. Kun perustoiminnot ovat hallussa, käyttö on sujuvaa ja uusia toimintoja voi opetella lisää tarpeen tullessa. Käyttöön löytyy paljon hyviä oppaita ja esimerkkejä. Käyttöä helpottavat myös eri apuohjelmat. Perusteet on kuitenkin syytä opetella, vaikka käyttäisikin selkeitä ja helppokäyttöisiä graafisia apuohjelmia. Ilman perusteiden ja peruskomentojen osaamista komentoja ja toimintoja on helppo käyttää väärin ja saada järjestelmä palauttamaan virheilmoituksia, mikä lopulta voi johtaa turhautumiseen. Pro Git on hyvä kirja, joka opettaa kaiken, mitä Gitistä tarvitsee tietää.

2.2.2 Mercurial

Mercurial on hajautettu versionhallintajärjestelmä, jonka ensimmäinen versio valmistui vuonna 2005. Mercurialin kehitykseen olivat vaikuttamassa aiempien versionhallintajärjestelmien tietyt ominaisuudet. Järjestelmästä haluttiin saada helppokäyttöinen ja korkean suorituskyvyn versionhallintajärjestelmä, joka skaalautuu myös suurille projekteille. (8.)

Mercurialissa on suhteellisen vähän komentoja, joten sen käyttö on helppo oppia. Järjestelmä on kevytrakenteinen ja erisuuruisille projekteille hyvin skaalautuva. Mercurialissa ja SVN:ssä on sa-

mankaltaiset komennot, joten SVN:n osaaminen helpottaa Mercuralin käyttöä. Keskitetyn versionhallintamenetelmänsä vuoksi SVN ei tallenna versiohistoriaa käyttäjän koneelle toisin kuin Mercurial ja muut hajautetut versionhallintajärjestelmät. Mercurialiin pystyy tuomaan versiohistorian useasta eri versionhallintajärjestelmästä. (8.)

Eri versionhallintajärjestelmillä on omat etunsa, joten yhtä järjestelmää, joka sopisi optimaalisesti kaikkeen, ei toistaiseksi ole tehty. Mercurial pitää vahvuuksinaan helppokäyttöisyyttä, suorituskykyä ja hyviä muutosten yhdistämisominaisuuksia. Toisaalta verrattaessa Mercurialin ja Gitin suorituskykyä toisiinsa Git selviytyy useammassa tapauksessa Mercurialia nopeammin. (8.)

2.2.3 Subversion (SVN)

SVN-versionhallintajärjestelmä on avoimen lähdekoodin keskitetty versionhallintamenetelmä (8; 9). SVN:n kehitys alkoi tarpeesta kehittää parempi versionhallintajärjestelmä valta-asemaan nousseen CVS-versionhallintajärjestelmän tilalle (9). Vuonna 2005 julkaistiin version 1.0 (9).

SVN:n edut keskitettynä versionhallintajärjestelmänä tulevat esiin, kun versioitavat tiedostot ovat binäärimuotoisia. Binääritiedostot ovat tiedostoja, jotka eivät ole enää tekstimuotoista lähdekoodia vaan käännettyä koneen ymmärtämää binääristä dataa. Keskitetyssä versionhallinnassa kaikki tiedot ovat palvelimella, joten suuret binääritiedostot eivät ole viemässä tilaa paikalliselta tietokoneelta. Hajautetussa versionhallinnassa binääritiedostot veisivät paljon tilaa paikallisella tietokoneella. Hajautetussa versionhallinnassa kaikki tieto on myös paikallisella tietokoneella ja jokaisesta muuttuneesta tiedostosta joudutaan ottamaan uusi kopio. Uusi kopio joudutaan ottamaan, koska binääritiedoston sisällön muutoksia ei voida lukea. SVN mahdollistaa tiedostojen lukituksen, jolloin muut käyttäjät eivät pääse käyttämään tiedostoa. Toiminnallisuus on eduksi binääritiedostoja käsiteltäessä. (8.)

SVN:n heikkoudet ovat suorituskyvyssä, jossa Mercurial ja Git ovat parempia (8). Keskitetyn versionhallintatavan takia, yhteys palvelimeen tulee säilyttää, jotta työnteko järjestelmän kanssa on sujuvaa. Työskentely ilman verkkoyhteyttä keskeytyy viimeistään silloin, kun muutokset tulisi lisätä versionhallintaan.

2.3 Versionhallintapalvelut

Ohjelmoinnissa, jossa halutaan hyödyntää versionhallintapalvelua tai DevOps-työkaluja, sopivan palvelun valinta on oleellinen tehtävä. Palveluilla on paljon yhtäläisyyksiä, mutta erojakin löytyy. Jotkin palveluista ovat ilmaisia ja toiset maksullisia. Usein palvelun maksullisuus riippuu käyttäjämäärästä, käytettävissä olevista ominaisuuksista, ohjelmakoodin avoimuudesta ja hallinnointitavasta.

Versionhallintapalvelut helpottavat järjestelmän tai palvelun hallinnointia. Lisäksi useat palvelut tarjoavat DevOps-menetelmien mukaisia työkaluja. Ketterä ohjelmistokehitys liittyy oleellisesti DevOps-menetelmiin. DevOps-työkaluihin lukeutuu muun muassa suunnittelu-, katselmointi-, wiki-sivusto-, jatkuvan integraation-, jatkuvan toimituksen-, tehtävienhallinta- ja projektinhallintatyökalut.

Kaikki uudet palvelut, järjestelmät ja työkalut vaativat aina jonkinlaisen panoksen käyttöönottovaiheessa. Mikäli käyttöönottoa ja menetelmien jalkautusta tekijöille ei tehdä huolellisesti, saattavat ne menettää merkityksensä ja koitua enemmän rasitteeksi kuin hyödyksi ohjelmointityössä. Siksi on syytä kiinnittää huomiota, millainen järjestelmä tai palvelu tukee parhaiten juuri kohdeyrityksen tehokkuutta ohjelmointiprojekteissa.

Suurimmaksi osaksi DevOps- ja versionhallintapalvelut liitetään muuhun ohjelmointityöhön kuin teollisuusrobottien ohjelmointiin. On mahdollista, että tulevaisuudessa samat menetelmät ja työkalut ovat laajasti jokapäiväisessä käytössä myös teollisuusrobotiikan ohjelmoinnissa.

GitHub

GitHub on suurin yksittäinen versionhallintapalvelu, joka tarjoaa hallintapalvelun Git-versioarkistoille (repository). GitHub tarjoaa sekä ilmaisia että maksullisia tuotteita muun muassa ohjelmakoodin säilytykseen, vian seurantaan (issue tracking) ja koodin katselmointiin. (2, s. 167–168.)

GitHub tarjoaa GitHubin käyttäjille graafista GitHub Desktop -asiakasohjelmaa GitHubin käyttöliittymäksi. GitHubin etuja ovat itsenäisesti toimiva (self-hosting) versioarkistointi, tehtävienhallinta, koodin kehitystoiveiden ilmaiseminen, wiki ja muut edellä mainitut ominaisuudet samassa palvelussa. Palvelua käytettäessä ei ole tarvetta ylläpitää Git-palvelinta ja näin säästetään aikaa etenkin sellaisissa organisaatioissa, joissa ei ole omaa IT-tukea. GitHub toimii eräänlaisena versioiden pilvitallennustilana.

Bitbucket

Atlassian Bitbucket on vastaavanlainen versionhallintapalvelu kuin GitHub. Bitbucket kehitettiin alun perin Mercurial-versionhallintajärjestelmälle (10). Bitbucket alkoi tukemaan Git-versionhallintajärjestelmää, ja Mercurialin tuki jätettiin sittemmin pois. Sivustonsa mukaan Bitbucket-pilvi on Git-versionhallintajärjestelmään perustuva hallinnointi- ja yhteistyötyökalu (17).

Bitbucketilla on kolme eri hallinnointivaihtoehtoa. Pilvipohjaisena hallinnointi tapahtuu Atlassianin palvelimilla. Palvelinpohjaisena palvelu suoritetaan käyttäjäyrityksen omalla palvelimella. Kolmas vaihtoehto on tarkoitettu suurille organisaatioille. Siinä käytössä on datakeskus, joka näkyy käyttäjille yhtenä instanssina, mutta on hajautettu usealle käyttäjäyrityksen hallinnoimalle palvelimelle. (17.)

Bitbucketin etuja ovat helppo yhdistettävyys moniin DevOps-työkaluihin, kuten tehtävienhallintaohjelmisto Jiraan ja Trelloon (17). Kumpikin ohjelmistoista ovat Agile-projektinhallintatyökaluihin kuuluvia sovelluksia.

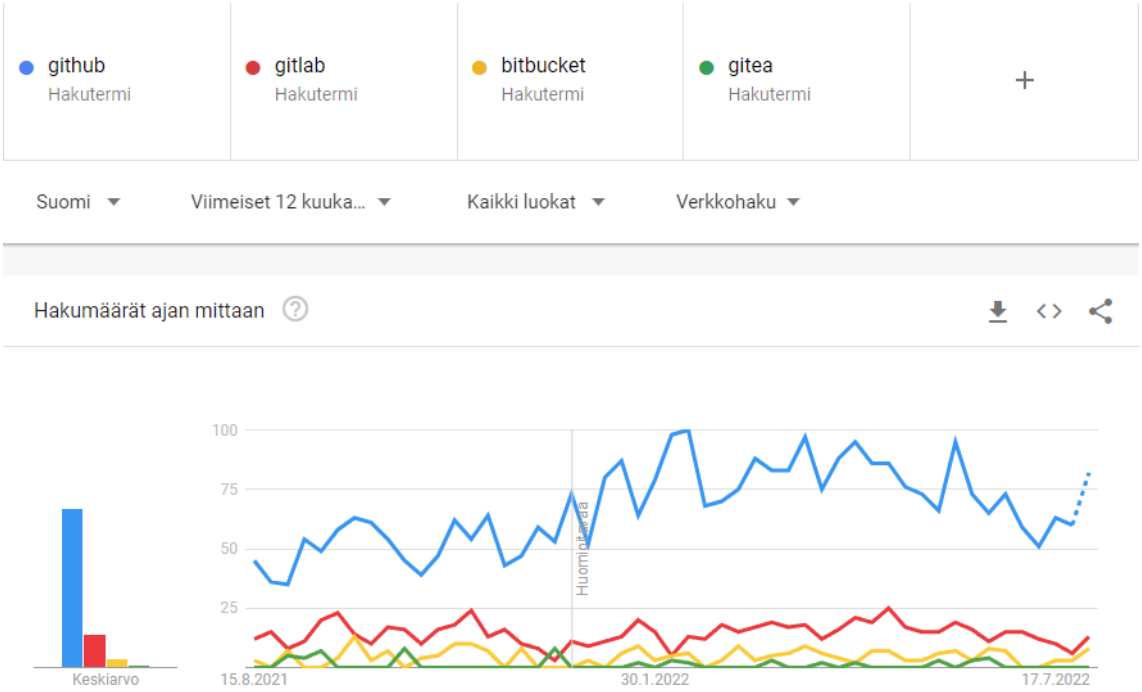
GitLab

GitLab kehitettiin vaihtoehtoiseksi versionhallintapalveluksi GitHubille ja Bitbucketille. GitLab on DevOps-alusta, joka sisältää SCM:n lisäksi useita DevOps-työkaluja (18). GitLabilla voidaan suunnitella (plan), hallita lähdekoodia (SCM), katselmoida koodia, hyödyntää jatkuvaa integrointia (Continuous Integration), hyödyntää jatkuvaa jakelua (Continuous Delivery) ja hyötyä monista muista DevOps-työkaluista (18). GitLab on todellisuudessa paljon muutakin kuin pelkkä versionhallintapalvelu. GitLabilla on arviolta 30 miljoonaa rekisteröityä käyttäjää (18).

Gitea

Gitea on samankaltainen versionhallintapalvelu kuin GitHub, Bitbucket ja GitLab. Gitea kuvaa itsensä kivuttomaksi itsehallitsevaksi palveluksi (painless self-hosted service). Gitean tavoitteena on tarjota helpoin, nopein ja kivuttomin tapa toteuttaa itsehallinnoiva Git-versionhallintapalvelu. Ohjelmointi on toteutettu Go-ohjelmointikielellä, jonka ansiosta Gitea tukee useita eri käyttöjärjestelmiä ja arkkitehtuureita kuten Linux, Windows, macOS, UNIX, x86 sekä amd64. Jopa Raspberry Pi 3:n suorituskyky riittää suorittamaan palvelua. Gitean sivustolla on kattava vertailu ominaisuuksista muihin vastaaviin palveluihin. (19.)

Kaikista versionhallintapalveluista ei löytynyt käyttäjämääriä. Kuvasta 8 voidaan arvioida eri palveluiden suosiota. Kuvassa on Google-hakukoneen hakutuloksia Suomessa, kun hakusanoina on edellä mainitut versionhallintapalvelut. Google on ottanut tietojen keräämisen parannuksen käyttöön 1.1.2022, jonka jälkeen trenditaso on noussut. Kuvaajasta nähdään, että hakusanoista GitHub on selkeästi suosituin. GitHubilla on sivustonsa mukaan yli 83 miljoonaa rekisteröitynyttä ohjelmoijaa (20). Githubin ja GitLabin ilmoittamat rekisteröityjen käyttäjien määrät korreloivat hyvin kuvaajan kanssa.



KUVA 8. Google Trends -sivuston tulos versionhallintapalveluhakutermien suosiosta 9.8.2022 viimeisen 12 kk:n ajalta (21)

2.4 Muut apuohjelmat

Apuohjelmilla tarkoitetaan versionhallinnan käyttöä helpottavia sovelluksia. Versionhallinnan käytön helpottamiseksi on saatavilla useita erilaisia sovelluksia tiedon tarkasteluun, vientiin ja tuontiin järjestelmästä sekä versionhallinnan käyttämiseen. Sopivan sovelluksen löytäminen voi tuoda helpotusta käyttäjälle versionhallinnan käyttöön ja siten lisätä motivaatiota sen käyttöön sekä parantaa työn tehokkuutta. Eräitä versionhallinnan käyttöä helpottavia sovelluksia on esitelty tässä luvussa.

Gitweb

Gitweb on yksinkertainen web-pohjainen visualisointityökalu, jonka avulla on helppo seurata useita versioarkistoja ja niiden eri versioita yhdestä paikasta (22). Työkalun avulla voidaan nähdä myös eri versioiden sisältö ja haarat (22). Tietyn version haku onnistuu käyttäen commit-viestiä tai sen osaa hakukentässä. Gitweb voidaan helposti käyttöönottaa paikalliselle Linux- tai Mac-koneelle yhden henkilön käyttöön. Työkalun hyödyntäminen Windows koneella vaatii, että Gitweb on asennettu palvelimelle, jonne otetaan yhteys web-selaimella Windows koneelta. Gitweb voi helpottaa versionhallinnan seurantaa etäpalvelimelta ilman, että muutoksia tarvitsee noutaa omalle työasemalle.

TortoiseGit ja TortoiseSVN

TortoiseGit ja TortoiseSVN ovat Windows-käyttöjärjestelmän graafisia käyttöliittymiä versionhallintajärjestelmään. Kumpikin ovat itsenäisiä ohjelmia. TortoiseGit kehitettiin TortoiseSVN:n pohjalta. Nimensä mukaisesti sovellus sopii joko Git- tai SVN-versionhallintajärjestelmään. Kumpikin on avoin ja ilmainen sovellus. TortoiseGit ja TortoiseSVN tukevat useita versionhallintajärjestelmän peruskomentoja ja näyttävät ne suoraan Windowsin resurssienhallinnassa (file explorer) tiedoston tai kansion sisältövalikossa. Ohjelmissa on myös useita muita käyttöä helpottavia toimintoja, jotka voivat olla eduksi Windows-ympäristöön tottuneelle käyttäjälle. Komentoriviä työssään paljon hyödyntävät pystyvät kuitenkin tekemään toiminnot komentoriviltä nopeammin, mutta aloitteleva versionhallinnan käyttäjä voi saada TortoiseGit- tai TortoiseSVN-sovelluksesta helpotusta versionhallinnan käyttöön.

Sourcetree

Sourcetree on myös Windows käyttöjärjestelmän graafinen käyttöliittymä Git-versionhallintajärjestelmään ja Mercurial-versionhallintajärjestelmään. Sourcetree on ilmainen ja on saatavissa myös Mac tietokoneelle. Sourcetree helpottaa ja yksinkertaistaa versionhallintajärjestelmän käyttöä ja voi madaltaa kynnystä aloittaa versionhallinnan käyttö. Hyvin visuaalinen käyttöliittymä tarjoaa käyttäjälleen helpotusta moniin versionhallintaan liittyviin tehtäviin. Se visualisoi Gitwebin tapaan versioarkistot ja niiden eri versiot, haarat, tagit sekä pystyy hallinnoimaan useita paikallisia- (local repository) ja etäversioarkistoja (remote repository).

Visual Studio Code

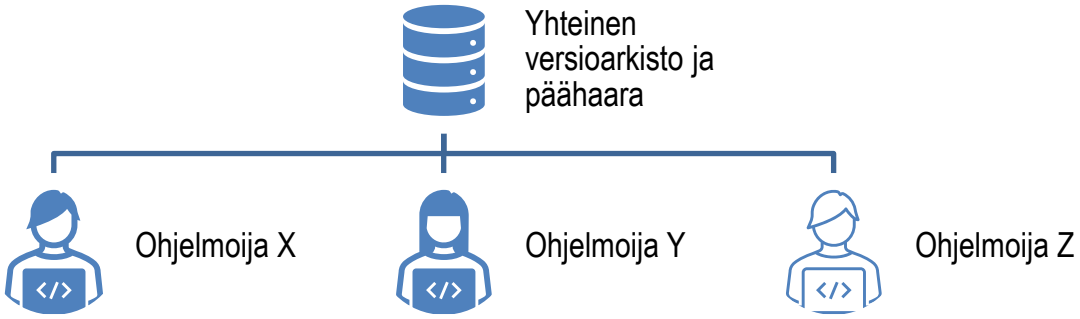
Visual Studio Code on käytännössä monipuolinen tekstieditori. Siinä on sisäänrakennettu tuki usealle eri versionhallintajärjestelmälle ja -palvelulle. Sovelluksella voi tehdä yleisimmät versionhallintaoperaatiot ja se kykenee luomaan suoran yhteyden muun muassa Githubiin (2, s. 477). Gitiä käytettäessä sovellus näyttää Git-tilapalkin, jossa on muun muassa haarat, ja saapuvat ja lähetettävissä olevat muutokset (2, s. 477). Visual Studio Code voi olla hyödyksi teollisuusrobottien versionhallinnassa etenkin eri versioiden eroavaisuuksien vertailussa.

2.5 Hajautetun versionhallinnan työnkulku

Työnkululla tarkoitetaan, kuinka versionhallintaa käytetään. Hajautettua versionhallintaa voidaan käyttää monin eri tavoin. Työnkulkuun ei ole suoraan standardia, kuinka tulisi toimia. Toimintatavat kannattaa sopia tiimin kesken, jotta yhteistyöstä tulee sujuvaa ja tehokasta (15). Git on yksi versionhallintajärjestelmistä, jolle on kehitelty erilaisia työskentelytapoja. Jokaisella työpaikalla voi olla omat toimintatapansa, jotka on koettu heille sopivaksi. Hajautetulla versionhallinnalla voidaan luoda erilaisia hajautettuja työnkulkuja, joita ei voida toteuttaa perinteisemmällä keskitetyllä versionhallinnalla (13). Saadakseen käsityksen millaista työnkulku Git-versionhallinnan kanssa on, seuraavana on esitelty muutamia työnkulkuja, joista eri tiimit voivat tarvittaessa jatkokehittää omaan tarpeeseensa sopivimmat toimintamallit.

Keskitetty työnkulku

Keskitetyissä versionhallintajärjestelmissä on yleisesti käytössä yksi ohjelmoijien yhteistyömalli, joka on keskitetty työnkulku (kuva 9). Keskitetyssä työnkulussa jokainen ohjelmoija lähettää muutoksensa jaettuun versioarkistoon. Käytännössä keskitetyn työnkulun käyttäminen tarkoittaa, että ensimmäinen ohjelmoija lisää versionsa versionhallintaan, jonka jälkeen seuraava ohjelmoija hakee uusimman version järjestelmästä ja yhdistää sen omaansa. Yhdistämisen jälkeen hän voi lisätä oman versionsa versionhallintaan. (2, s. 126–127.)



KUVA 9. Keskitetyssä työnkulussa kaikki ohjelmoijat lähettävät muutoksensa suoraan yhteiseen versioarkistoon

Keskitetty työnkulku sopii sekä keskitettyyn- että hajautettuun versionhallintajärjestelmään (2, s. 127). Keskitetyssä versionhallinnassa keskitetty työnkulku on käytössä luonnollisesti, mutta se toimii myös hajautetun versionhallintajärjestelmän kanssa.

Keskitetyssä työnkulussa riittää yksi päähaara (main branch). Tosin silloin hajautetun versionhallinnan edut eivät tule hyödynnettyä kattavasti. Gitin yksi etu on jokaisen ohjelmoijan oma versiotietokanta omalla koneella, jolloin ohjelmointia voi tehdä ja versioita lisätä versionhallintaan riippumatta muista ohjelmoijista. Toinen merkittävä etu on haarat. Niitä hyödyntämällä ohjelmoija voi helposti yhdistellä koodia eri versioarkistojen välillä ja hyötyä haarojen mahdollistamasta koodin suojaamisesta. Kun ohjelmakoodi on omassa haarassaan, siihen ei vaikuta päähaaraan tehdyt muutokset. (23.)

Keskitetty työnkulku voi olla sopiva pienille tiimeille tai niille, jotka ovat tottuneet keskitettyyn versionhallintaan. Mikäli käytössä on hajautettu versionhallintajärjestelmä, seuraavana esiteltävistä työnkuluista voi löytyä tehokkaampi työskentelytapa.

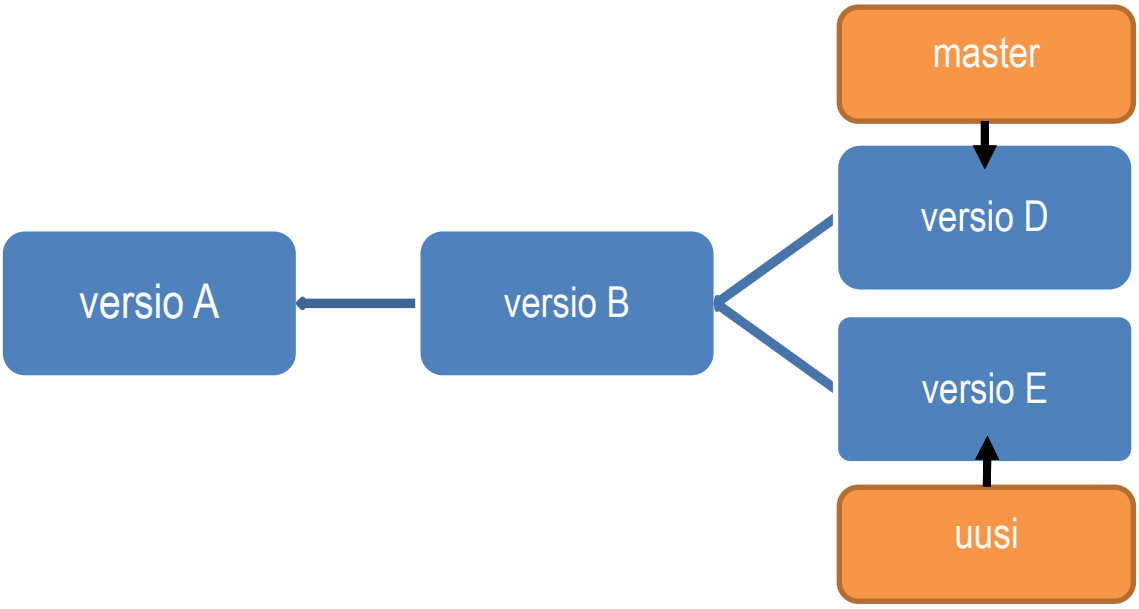
Ominaisuushaara (feature branch) -työnkulku

Tämän työnkulun pääajatus on, että jokainen kehitettävä ominaisuus tehdään omassa sivuhaarassaan päähaaran rinnalla (24). Jotta työnkulun voi helposti ymmärtää, on syytä esitellä versionhallinnan yksi merkittävimmistä ominaisuuksista eli haarojen käyttö.

Etenkin Git-versionhallintajärjestelmässä, haarat ja niiden hyödyntäminen on todella vaivatonta. Siksi haarojen käyttöä pidetään yhtenä Gitin merkittävimmistä ominaisuuksista. Muissa versionhallinajärjestelmissä sen käyttö voi olla aikaa vievää. (2, s. 63.)

Gitin oletuspäähaara on master-haara. Joka kerta, kun ohjelmoija vie muutokset Gitin päähaaran, eli tässä tapauksessa master-haaran, osoitin siirtyy yhden commitin eteenpäin. Commit on termi, jolla tarkoitetaan kommentoitua versionhallintaan vietyä versiota. Kun päähaarasta luodaan uusi haara, päähaara sekä uusi haara osoittavat samaan kohtaan versiohistoriassa. (2, s. 64.)

Kuvassa 10 on kuvattu tilanne, jossa B-version kohdalla on luotu haara nimeltään uusi ja master- ja uusi-haaraan on tehty yksi muutos. Haarat ovat eriytyneet toisistaan. Kumpikin haara perii version A ja B historian, ja lisäksi ne sisältävät oman yksilöllisen muutoksen.



KUVA 10. Master-haara ja uusi-haara ovat eriytyneet, mutta sisältävät myös yhteistä historiaa (mukaillen 2, s. 66)

Ominaisuushaara -työnkulun (feature branch workflow) idea on siis hyödyntää haaroja. Jokainen koodiin tehtävä ominaisuus kehitetään omassa haarassaan, jolloin ominaisuuden kehitys voidaan tehdä erillään muista ominaisuuksista tai päähaaran kehityksestä. Kun ominaisuus on saatu valmiiksi, se yhdistetään päähaaraan. Tämä työnkulku poikkeaa keskitetystä työnkulusta siinä, ettei muutoksia lisätä suoraan päähaaraan, vaan kuvaavalla nimellä nimetyn palvelimella olevan versio-

arkiston (kutsutaan myös nimellä origin) ominaisuushaaraan. Ominaisuushaara ei poikkea toiminnoiltaan päähaarasta, vaan se on yksi haara siinä missä päähaarakin, ainoastaan omalla kuvaavalla nimellään. Ominaisuushaaran vieminen palvelimen versioarkistoon mahdollistaa muiden ohjelmoijien yhteistyön kyseisen ominaisuuden kanssa. (24.)

Tällaista työnkulkua voidaan hyödyntää myös havaitun ongelman korjaamiseen. Kun koodissa havaitaan ongelma, sen ratkaisemiseksi voidaan luoda uusi haara. Tarvittavat muutokset tehdään koodiin, toiminto testataan ja kun varmistutaan, että korjaus on toiminut, voidaan haara yhdistää päähaaraan. Tällaisessa tapauksessa haara voidaan poistaa yhdistämisen jälkeen. Toimimalla näin, voidaan suojata päähaaraa turhilta muutoksilta. Mikäli korjausyritys ei heti onnistu, voidaan kehitystä jatkaa haarassa tekemättä lisää ongelmia päähaaraan.

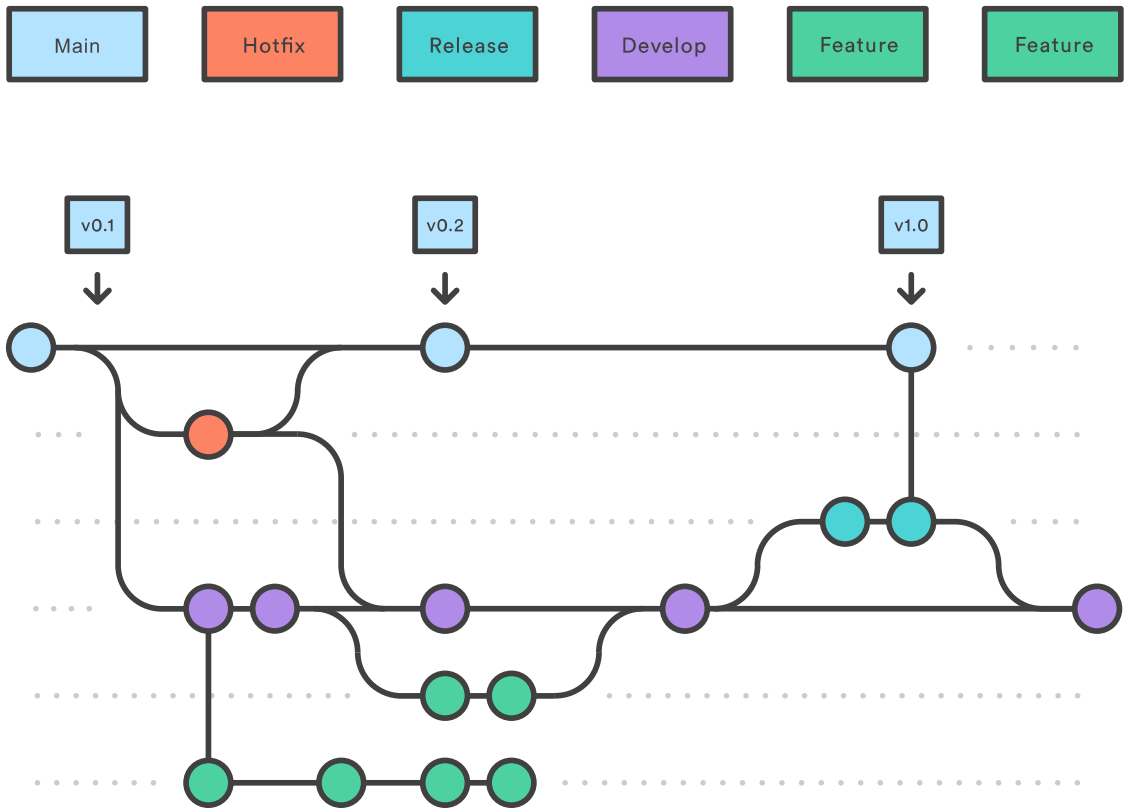
Haarat mahdollistavat myös keskustelun muutoksesta ennen kuin se yhdistetään päähaaraan. Keskustelu voidaan avata niin sanotun pull request tai merge request -pyynnön kautta. Termit liittyvät eri versionhallintapalveluiden, kuten Bitbucket tai GitLab ominaisuuksiin. Pyynnön tarkoituksena on katselmoida tehdyt muutokset muiden ohjelmoijien toimesta ennen päähaaraan yhdistämistä (24). Katselmoinnin tarkoituksena on löytää mahdollisia virheitä tai tuoda esille parannusehdotuksia ja täten saada yhdistettävästä koodista parempi.

Gitflow-työnkulku

Gitflow-työnkulun pääajatus on kahdessa päähaarassa main- ja develop-haarassa. Main-haarassa on kaikki ohjelmakoodin julkaistut, kuten asiakkaalle toimitetut versiot. Develop-haara toimii integrointihaarana kehitetyille ominaisuuksille. Jokaiselle kehitettävälle ominaisuudelle luodaan uusi haara samaan tapaan kuin ominaisuushaaratyönkulussa. Lisäksi käytössä on release-haara. Kun tavoiteltu määrä ominaisuuksia on saavutettu esimerkiksi asiakasta varten, luodaan release-haara develop-haarasta. Release-haaran luonnin jälkeen uusia ominaisuuksia ei enää lisätä, vaan ainoastaan korjauksia virheisiin tai dokumentaatiota. Kun mahdolliset korjaukset on tehty, release-haara yhdistetään päähaaraan. (25.)

Kuvassa 11 on havainnollistettu, millainen Gitflow-työnkulku on. Kuvassa näkyvät haarat Main, Hotfix, Release, Develop, Feature sekä toinen Feature. Main-haaraan on luotu merkkejä (tags) julkaistuista versioista. Merkkien avulla on helppo selvittää, mistä koodipohjasta esimerkiksi asiakkaalle julkaistu versio on tehty. Kuvassa yksi ominaisuus on saatu yhdistettyä Develop-haaraan ja

yhden ominaisuuden kehitys on vielä kesken. Main-haara on saanut yhden väliversion, johtuen pikakorjauksesta, joka on toteutettu omassa nimikko haarassaan.

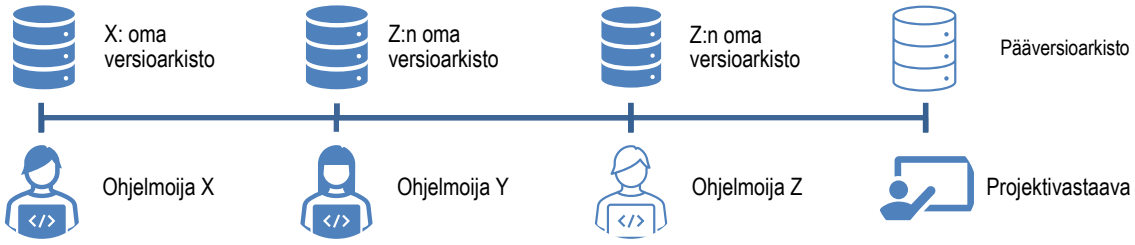


KUVA 11. Gitflow-työnkulkua havainnollistava esimerkki (25)

Gitflow-työnkulku on perinteinen tapa käyttää Gitiä (25). Sen edut tulevat esiin julkaisuperusteisessa työnkulussa, jossa ohjelmaa kehitetään aina tiettyyn pisteeseen asti ennen sen lähettämistä tuottavaan toimintaan (25). Gitflow voi olla eduksi suurissa organisaatioissa, joissa on useita ohjelmoijia samalle koodipohjalle. Siitä voi myös hyödyntää itselle sopivia osia ja muodostaa omaan työympäristöön parhaiten soveltuvan työnkulun.

Forking-työnkulku

Forking-työnkulussa jokaisella ohjelmoijalla on oma versioarkistonsa (kuva 12). Jokaisella on kaksi versioarkistoa: paikallinen ja palvelimella oleva versioarkisto. Tässä työnkulussa projektin hallinnoija poimii muutokset jokaisen ohjelmoijan omasta palvelimelle tallennetusta versioarkistosta varsinaiseen pääversioarkistoon. Tässä työnkulussa on etuna se, ettei jokaiselle kehittäjälle tarvitse antaa oikeuksia pääversioarkistoon. Forking-työnkulussa käytetään tyypillisesti Gitflow-työnkulun toimintamalleja. (26.)



KUVA 12. Forking-työnkulku havainnollistettuna (mukaillen 26*)*

Jokin työnkulku on syytä hallita, jotta versionhallinnan käyttö on sujuvaa. Esitetyt toimintamallit ovat esimerkinomaisia toimintamalleista ja niiden tarkoituksena on herättää ajattelua, miten yrityksessä voitaisiin parhaalla mahdollisella tavalla käyttää tehokkaasti versionhallintaa. Jos kaikki eivät tunne yhteisiä toimintatapoja tai noudata sovittua työnkulkua, saattaa se johtaa tehottomaan versionhallinnan käyttöön. Lopulta seurauksena voi olla, ettei motivaatiota versionhallinnan käyttöön enää ole.

3 ROBOTISOITU AUTOMAATIO

Maailma automatisoituu jatkuvasti ja on tehnyt sitä jo vuosisatojen ajan. Automaatio ei siis ole uusi käsite. Teknologiat yhdistyvät monipuolisiksi kokonaisuuksiksi, joihin voi kuulua teollisuusrobotteja, konenäköratkaisuja, liikkeenohjausta yhdistettynä tekoälyyn ja pilvipohjaiseen laskentaan sekä huippunopeisiin tietoliikenneyhteyksiin. (32.)

Mikä on robotti?

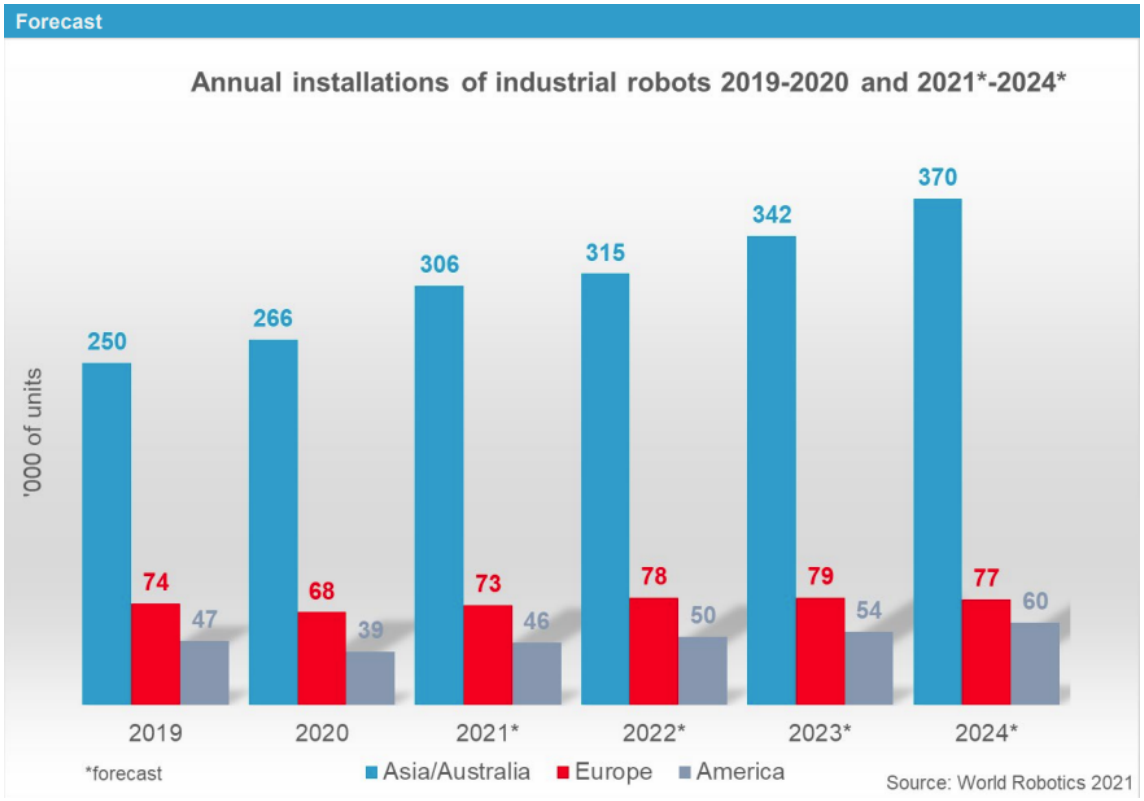
Kansainvälinen standardisoimisjärjestö määrittelee robotiikan sanaston kattavasti standardissaan ISO 8373:2021. Standardin mukaan pelkkä robotti on autonominen ohjelmoitu mekanismi, joka sisältää ohjausjärjestelmän (33). Määritelmä on muuttunut standardin edellisestä versiosta vuodelta 2012. Vuoden 2012 versiossa robotti määriteltiin autonomiseksi ohjelmoiduksi mekanismiksi, jolla on kaksi tai useampia akseleita (34). Uusimmassa versiossa määritelmästä on poistunut akselivaatimus robotille. On hyvä huomioida, että robotti ja teollisuusrobotti ovat kaksi eri termiä, ja niille kummallekin on omat määritelmänsä.

IFR eli International Federation of Robotics jaottelee robotit kahteen ryhmään: teollisuusrobotteihin ja palvelurobotteihin (35). Näiden kahden lisäksi lääketieteellisiä robotteja ei pidetä teollisuusrobotteina eikä palvelurobotteina (33). Näin ollen lääketieteelliset robotit muodostavat IFR:n ryhmien lisäksi oman ryhmänsä.

Teollisuusrobotit ovat määritelty ISO 8373:2021 standardin mukaan uudelleenohjelmoitaviksi vähintään kolmiakselisiksi laitteiksi, joka sisältää manipulaattorin toimilaitteineen, robottiohjaimen ja käyttöliittymän robotin opettamiseksi ja/tai ohjelmoimiseksi. Saman standardin mukaan palvelurobotit ovat robotteja, jotka toteuttavat hyödyllisiä tehtäviä henkilökohtaisessa tai ammatillisessa käytössä ihmisille tai laitteille. Lääketieteelliset robotit ovat robotteja, jotka ovat tarkoitettu käytettäväksi lääketieteellisessä tarkoituksessa. (33.)

Tavanomaisesti robotisoidussa automaatiossa yhdistyvät teollisuusrobotti, ohjelmoitava logiikka ja tarvittavat oheislaitteet. Sovelluksissa teollisuusrobotti hoitaa kappaleiden tai työkalujen siirtelyn ja logiikka koordinoi robotin ja oheislaitteiden yhteistoimintaa. Tällaista kokonaisuutta kutsutaan robottisoluksi.

IFR esittää lehdistökonferenssissa 28.10.2021, että maailmanlaajuisesti robotteja asennetaan vuosittain yhä enemmän. IFR ennustaa kasvun jatkuvan myös lähitulevaisuudessa (kuva 13). Markkinakehityksessä IFR korostaa toimitusketjujen paikallisuuden vaikutusta kehitykseen. Toimittajien olisi hyvä olla lähempänä asiakasta, jolloin toimitusketjut olisivat kestävämpiä logistisesti ja poliittisesti. (35.)



KUVA 13. Teollisuusrobottien vuosittaiset asennukset ja ennuste (35)

Toimitusketjujen lyhentämisellä voitaisiin mahdollisesti karsia komponenttien pitkiä merikuljetuksia, joissa tavara siirtyy hyvin kaukaa Eurooppaan. Paikallisella robotisoidulla tuotannolla voisi olla ympäristön kannalta edullisia vaikutuksia. Teknologisesta kestävän kehityksen kehityssuunnasta IFR mainitsee modernien robottien kuluttavan vähemmän energiaa kevyiden rakenteiden, älykkään energiankäytön ja edistyksellisten robottien tarttujien avulla, jotka eivät kuluta energiaa lähes ollenkaan. (35.)

Ennusteen perusteella roboteilla vaikuttaisi olevan paljon kysyntää myös tulevaisuudessa ja kysyntä myös vaikuttaisi kasvavan. Kasvavan kysynnän vuoksi robotteja tulee pystyä käyttöönottamaan entistä nopeammin, eli alan tulee kasvaa tai toimintojen tehostua.

Daedal Researchin vuonna 2022 tekemän markkinatutkimuksen mukaan markkinatrendi suuntautuu yhteistyörobotteihin, joita kutsutaan myös coboteiksi (36). Cobot-nimitys tulee englanninkielinen sanoista collaborative robots (36). Markkinatutkimuksen mukaan avainyhtiöitä teollisuusrobottimarkkinoilla ovat muun muassa Fanuc, ABB, KUKA, Kawasaki ja Epson Robots (36). Nämä robottimerkit muodostivat suurimman osan tämän tutkimuksen tyypillisimmistä roboteista.

3.1 Teollisuusrobotit

Teollisuusrobotit ovat määritelty ISO 8373:2021 standardin mukaan uudelleenohjelmoitaviksi vähintään kolmiakselisiksi laitteiksi, jotka sisältävät manipulaattorin toimilaitteineen, robottiohjaimen ja käyttöliittymän robotin opettamiseksi ja/tai ohjelmoimiseksi (33). Standardi SFS-EN ISO 10218-1:2011 määrittelee teollisuusrobotin mukautetusti edellä mainitun standardin vanhemmasta painoksesta ISO 8373:1994 (37, s. 12). Määritelmän mukaan teollisuusrobotti on teollisuuden automaatiosovelluksissa käytettävä automaattisesti ohjattu, uudelleen ohjelmoitavissa oleva monikäyttöinen käsittelylaite, jonka kolme akselia tulee olla ohjelmoitavissa (37, s. 12). Laite voi olla kiinteästi asennettu tai liikkuva (37, s. 12).

Määritelmään mahtuu monenlaisia teollisuusrobotteja. Kaikkia robottityyppejä yhdistää se, että niissä on itse käsittelylaite, esimerkiksi robotin käsivarsi, ja ohjauslaite, jota kutsutaan myös robottiohjaimeksi. Teollisuusrobottijärjestelmään kuuluu teollisuusrobotin lisäksi robotin työkalut ja kaikki oheislaitteet, jotka tukevat robotin tehtävän toteuttamista (37, s. 14).

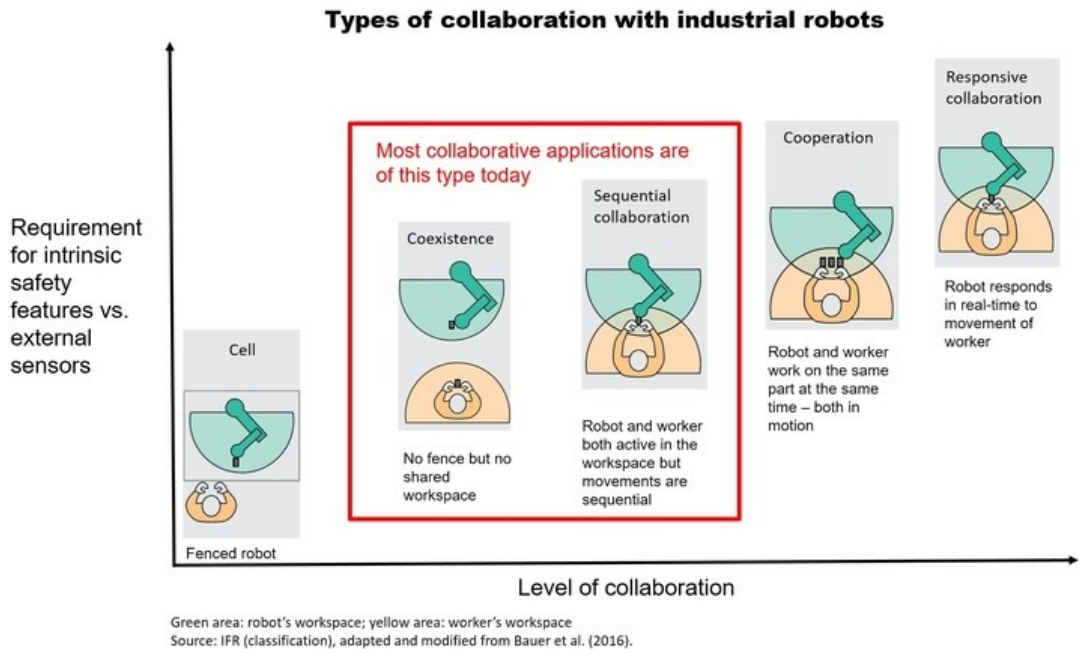
Teollisuusrobotin ohjauslaitteeseen eli robottiohjaimeen kuuluu ohjausohjelma, joka on käskysarja, jolla robotti saa kyvyn toimia ja vastata sille annettuihin käskyihin (37, s. 14). Tämä toiminnallisuus on ohjelmoitu teollisuusrobotin ohjauslaitteeseen siten, ettei käyttäjä normaalisti pääse muuttamaan sitä (37, s. 14). Joissain tapauksissa käyttäjän on mahdollista päivittää ohjausohjelma robottivalmistajan toimittamalla firmware-päivityksellä. Käyttäjän kannalta oleellisempi osa ohjausta on työohjelma, johon ohjelmoidaan liikkeet ja aputoiminnot käskysarjana (37, s. 14). Sovellus on tehty tiettyä tarkoitusta varten ja on aina muutettavissa.

3.2 Yhteistyörobotit

Yhteistyörobotit eli niin sanotut cobotit (collaborative robots) ovat teollisuusrobotteja, joka tekevät työtään yhteistyössä työntekijöiden kanssa. Vuorovaikutteisuus robotin ja työntekijän välillä tekee robotista cobotin. IFR jakaa yhteistyörobotit kahteen ryhmään. Ensimmäisessä ryhmässä ovat teollisuusrobotit, jotka täyttävät standardin ISO 10218-1 vaatimukset. Toisessa ryhmässä ovat teollisuusrobotit, jotka ovat yhteistyörobotteja, mutta eivät täytä edellä mainitun standardin vaatimuksia. Vaikka standardin vaatimuksia ei täytettäisikään, se ei suoraan tee robotista vaarallista. Koneturvallisuuden vaatimustenmukaisuus on mahdollista saavuttaa myös ilman standardeja. (39.)

Suomessa ISO 10218-1 -standardia vastaa kansallinen standardi SFS EN ISO 10218-1 (37, s. 1). Yhteistyörobottien vaatimustenmukaisuus, joka käytännössä tarkoittaa turvallista konetta, voidaan saavuttaa myös ilman standardia. Yhteistyörobotit, jotka liikkuvat ihmisen ollessa kosketusetäisyydellä robottiin, on suojattu määrittämällä robotille tilanteeseen sopivat parametrit kuten nopeus ja voima. Suureita mitataan jatkuvasti, ja mikäli raja-arvo ylitetään, niin robotti saatetaan turvalliseen tilaan. Viimekädessä robotin käyttöönottaja huolehtii riskiarvioinnin avulla tilanteeseen ja käyttötarkoitukseen soveltuvien parametriarvojen määrittämisestä.

Kuvassa 14 on havainnollistettu yhteistyörobotiikan eri toimintatapoja. Tekninen raportti ISO/TS 15066 tarjoaa ohjeistusta yhteistyörobottien operaatioihin, joissa ihminen ja yhteistyörobotti jakavan yhteisen työalueen. Tekninen raportti täydentää standardeja ISO 10218-1 ja ISO 10218-2 tarjoamalla lisäohjeistusta yhteistyörobottien hyödyntämiseen turvallisesti. Teknisessä raportissa esitetään tiettyyn ruumiinosaan kohdistuvan paineen ja voiman sallitut arvot sekä niiden välisen suhteen. (38, s. v.)



KUVA 14. Teollisuusrobotin ja ihmisen välisen yhteistyön eri tapoja (39)

Usein yhteistyörobottien ohjelmoiminen poikkeaa jossain määrin muista teollisuusroboteista. Omronin yhteistyörobottien ohjelmointiympäristö on TM Flow, joka on eri kuin muiden Omronin teollisuusrobottien ohjelmointiympäristö ACE. KUKA LBR iisy yhteistyörobotilla niin ikään on eri ohjelmointiympäristö ja uusi käyttöjärjestelmä verrattuna robottivalmistajan muiden teollisuusrobottien ohjelmointiympäristöön ja käyttöjärjestelmään. LBR iisy ohjelmoidaan käyttäen graafista käyttöliittymää. Fanucin yhteistyöroboteilla on graafinen, raahaa ja pudota (drag & drop) -tyyppinen ohjelmointimahdollisuus ohjelmoinnin helpottamiseksi.

Yhteistyörobottien ohjelmoinnissa suuntaus vaikuttaisi olevan graafisiin käyttöliittymiin, jonka tarkoituksena on helpottaa ohjelmointia ja madaltaa kynnystä aloittaa robotin käyttöönotto ja käyttö. ABB tarjoaa myös graafisen käyttöliittymän cobottiensa ohjelmointiin. Graafinen käyttöliittymä, Wizard Easy Programming, on sovelluskerros uudessa OmniCore-käyttöjärjestelmässä. Wizard Easy Programming tuottaa samaa Rapid koodia, kuin normaali robotin ohjelmointikin.

Yhteistyörobottien paikoituspisteiden määrittämisessä hyödynnetään usein käsin avustamista. Käsin avustamisessa robotti ohjataan haluttuun pisteeseen ja orientaatioon esimerkiksi kappaleen poimimiseksi. Käsin avustetussa pisteiden opetuksessa robotin akseleiden jarrut vapautetaan, mutta robotti säilyttää riittävän momentin akseleiden moottoreilla, jotta robotti ei pääse hallitsemattomasti romahtamaan. Paikka tallennetaan, kun haluttu piste ja orientaatio on saavuttu. Tällä tavoin käyttäjän ei tarvitse määrittää halutun pisteen koordinaatteja, orientaatiota ja akselikonfiguraatioita.

Versionhallinnan kannalta yhteistyörobottien ohjelmia voidaan joutua tallentamaan binäärimuodossa, koska lähdekoodia ei välttämättä saada vietyä ohjelmointiympäristöstä tekstimuotoisena. Monet robottivalmistajien graafisen ohjelmoinnin ohjelmointiympäristöistä tuottavat binäärimuotoista lähdekoodia. Jos yhteistyörobotin ohjelmointiympäristö tuottaa samaa koodia, jota kirjoitettaisiin tavanomaisilla teollisuusroboteilla, niin ohjelmakoodi voidaan saada vietyä tekstimuotoisena versionhallintaan ja hyötyä paremmin versionhallinversionhallintajärjestelmän eduista.

3.3 Koneautomaatio-ohjaimet

Ohjelmoitavat logiikat (PLC) toteuttavat koneen ohjauksen sille ohjelmoidun ohjelman mukaisesti. Yksinkertaistettuna logiikka lukee siihen kytketyt tulot, suorittaa ohjelman mukaiset operaatiot ja kytkee logiikan lähdöt ohjelman mukaisesti. PLC:n logiikkatoimintojen lisäksi koneen ohjauksessa voidaan hyödyntää muitakin teknologioita kuten turvaluokiteltuja koneturvatoimintoja, liikkeenohjausta, useita kenttäväylätekniikoita ja liityntärajapintoja eri järjestelmiin. Laitetoimittajat käyttävät omista ohjausjärjestelmistään erilaisia nimityksiä. Nykyaikaisissa edistyneissä ohjausjärjestelmissä hyödynnetään useita eri teknologioita pelkän logiikkaohjauksen lisäksi, mitkä on saatu integroitua yhteen ohjauskomponenttiin. Teknologioiden integroituminen logiikkaohjaimen kanssa, on todennäköisesti vaikuttanut ohjainten nimeämiseen.

Omronin koneautomaatio-ohjaimissa (Machine Automation Controller) yhdistyvät edellä mainitut ominaisuudet sekä konenäkö (40). Siemens käyttää nimitystä teollisuusautomaatiojärjestelmä, jonka alaryhmänä on erityyppisiä ohjaimia, kuten edistyneet ohjaimet (41). Ohjaimet sisältävät logiikkaohjauksen lisäksi integroituja teknologioita, joiden laajuus ja edistyneisyys riippuu valitusta ohjaimesta.

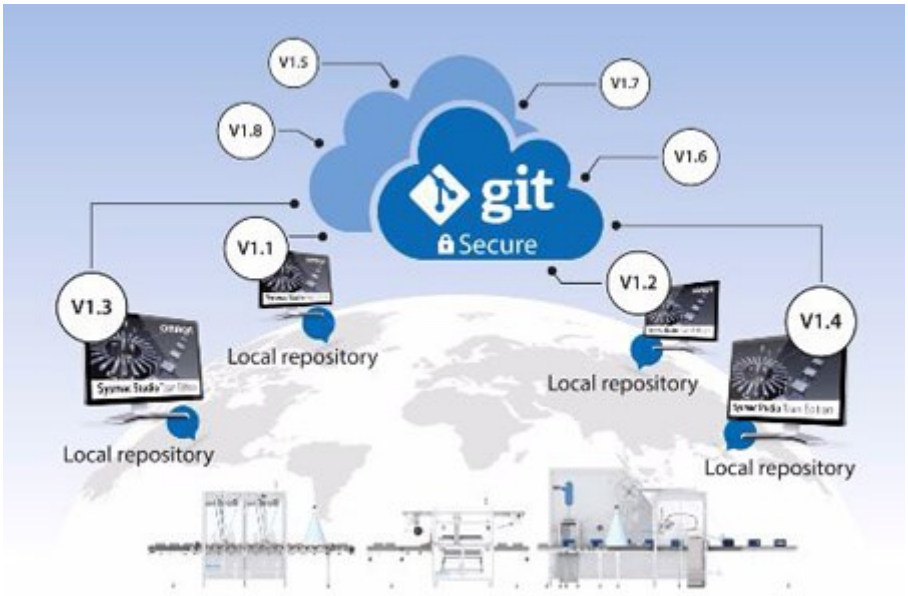
3.4 Koneautomaatio-ohjainten versionhallinta

Koneautomaatio-ohjainten osalta on havaittavissa selkeää suuntautumista kohti versionhallinnan hyödyntämistä. Useat valmistajat ovat jo integroineet versionhallintajärjestelmärajapinnan omaan ohjelmointiympäristöönsä. Hajautettu versionhallinta on vahvasti edustettuna versionhallintamenetelmänä niin kuin se on PC ohjelmoinnin alalla. Tutkimuksen viitekehykseen sisällytettiin muutaman versionhallintaa hyödyntävän logiikkavalmistajan ratkaisut versionhallinnan toteuttamiseksi. Tällä

tiedolla on tarkoitus osoittaa, kuinka logiikkaohjelmoinnin versionhallinnan hyödyntäminen vertautuu teollisuusrobottiohjelmoinnin versionhallintaan. Teollisuusrobottiohjelmoinnin versionhallinnan tämänhetkistä tilaa käsitellään luvussa 3.5.

Omron Team Edition

Sysmac Studio on Omronin koneautomaatio-ohjaimien ohjelmointiympäristö. Team Edition on laajennus, jolla Sysmac Studioon saadaan integroitua hajautettu Git-versionhallintajärjestelmä. Team Editionilla Sysmac Studio saavuttaa kaikki hajautetun versionhallinnan edut, kuten vuorovaikutteisen ohjelmistokehityksen yhdessä usean ohjelmoijan kanssa, helpon graafisesti ohjelmoitujen ohjelmien vertailun versioarkiston kanssa, muutosten seurannan, mahdollisuuden palata eri versioihin, koodin hajauttamisen ja mahdollisuuden ohjelmointiin käyttäen versionhallintaa, vaikka yhteyttä palvelimeen ei juuri silloin olisi. Kuvassa 15 on havainnollistettu hajautettua ohjelmointia Sysmac Studiolla. (15.)



KUVA 15. Git-versionhallintajärjestelmä Omron Sysmac Studio -ohjelmointiympäristössä (15)

Rajapintana Gitiin on käytössä graafinen Git-asiakasohjelma TortoiseGit, joka on integroitu Sysmac Studio -ohjelmointiympäristöön. TortoiseGitin avulla Git-työkalut ovat suoraan käytössä ohjelmointiympäristössä ilman tarvetta käyttää erillistä sovellusta muutosten viemiseksi versionhallintaan. Ohjelmakoodin lisäksi Sysmac Studio Team Edition pystyy vertailemaan käyttöliittymäsuunnittelun (HMI) graafista sisältöä. Se mahdollistaa ketterän versionhallinnan hyödyntämisen myös käyttöliittymäsuunnittelussa.

Beckhoff TwinCAT source control

Beckhoffin TwinCAT 3:ssa on SCM-integraatio, joka tukee itsenäisen työskentelyn lisäksi muun muassa tiimeissä työskentelyä. TwinCAT 3 -ohjelmointiympäristössä on valittavissa useita eri versionhallintajärjestelmiä. Ohjelmointiympäristö tukee versionhallintajärjestelmistä muun muassa Gitiä, TFS:ää ja SVN:ää. Muutosten vertailu tehdään TwinCAT Project Compare Tool -työkalulla TwinCAT ohjelmointiympäristössä. (27. s. 7, 8, 13.)

Beckhoffin TwinCAT 3 -ohjelmointiympäristö tarjoaa kattavan tuen versionhallintajärjestelmille. Ohjelmointiympäristössä voidaan saavuttaa kaikki versionhallinnan edut kuten Omronin Sysmac Studion kohdalla jo mainittiin. Näiden lisäksi on mahdollisuus valita eri versionhallintajärjestelmien väliltä omaan tarpeeseen sopivin vaihtoehto.

B&R Team functions

B&R Automation Studio 4.11 -ohjelmointiympäristö tukee alkuperäisesti Microsoft Visual SourceSafe-, SVN- ja TFS-versionhallintajärjestelmiä (28). Microsoft Visual SourceSafe on vanha, ja sen tuki on päättynyt vuonna 2017 (29). Automation Studion ohjelmointimalli ja avoin datarakenne mahdollistavat muidenkin versionhallintajärjestelmien käytön. Se ei vaadi mitään erityistä versionhallintajärjestelmältä, joten esimerkiksi Git-versionhallintajärjestelmän käyttäminen on mahdollista. Kaikki versionhallinnan edut ovat saavutettavissa myös B&R Automation Studio -ohjelmointiympäristön kanssa.

Siemens Version Control Interface (VCI)

Siemens SIMATIC STEP 7 (TIA Portal) logiikkaohjelmien ohjelmointiympäristö on tukenut Git- ja SVN-versionhallintajärjestelmiä versionsa V16 alkaen (42). Siemens kutsuu rajapintaa nimellä Version Control Interface (VCI). Rajapinta saadaan käyttöön ohjelmointiympäristöön laajennuksella (TIA Portal Add-Ins) VCI Git Connector versio 1.0.0 (43, s. 4). Käyttöohjeessa mainitaan rajattu määrä Git-komentoja. Joitain hyödyllisiä komentoja kuten rebase, merge ja tag ei mainita käyttöohjeessa. Tämän perusteella Gitin käyttö voi olla rajoittunut vain tiettyihin peruskomentoihin. Myöskään kaikkea tietoa ei voida viedä versionhallintaan, kuten suojattuja lohkoja sekä kirjastoja (44, s. 40).

Siemens on ottanut askeleen kohti versionhallinnan hyödyntämistä logiikkaohjelmoinnissa. Git-versionhallintajärjestelmänä on saanut merkityksen myös Siemens ympäristössä. Tämä kertoo vahvasti siitä, että versionhallinta koetaan merkitykselliseksi.

3.5 Teollisuusrobottien versionhallinta

Teollisuusrobottien ohjelmointiympäristöissä ei tällä hetkellä ole valmiita rajapintoja versionhallintajärjestelmiin, lukuun ottamatta Omronin teollisuusrobottien ohjelmointiympäristöä. Havaintoja suuntautumisesta rajapinnan kehittämiseksi on kuitenkin saatu. Tulevaisuus tulee todennäköisesti muuttamaan tilannetta versionhallintamyönteisemmäksi. Mahdollisuuksia versionhallinnan käyttöön on kuitenkin olemassa. Kuten tutkimuksen tuloksissa on mainittu, versionhallintaa hyödynnetään teollisuusrobottienkin ohjelmoinnissa, vaikka ohjelmointiympäristöt eivät ainakaan vielä sisällä suoraa rajapintaa versionhallintajärjestelmiin.

ABB

ABB teollisuusrobottien ohjelmointiympäristö on RobotStudio. RobotStudiossa yhdistyvät ohjelmointiympäristön lisäksi simulointi virtuaalisessa 3d ympäristössä sekä sisäänrakennettu virtuaalikontrolleri. Mallinnus, ohjelmointi ja simulointi hoituvat yhdellä sovelluksella. Virtuaalikontrolleri on täysin vastaavanlainen kuin todellinen robottikontrolleri, ja tämä mahdollistaa realististen simulaatioiden toteuttamisen. (45.)

RobotStudio kehittyy jatkuvasti ja parannuksia sovellukseen saadaan aika-ajoin. RobotStudion kehityssuuntana on pian nähtävissä laajentuminen pilvipalvelupohjaiseksi. Uusia ominaisuuksia on myös odotettavissa. Niihin lukeutuu muun muassa versionhallinta. RobotStudion käyttäjäkunta pitää versionhallintaa toivottuna ominaisuutena RobotStudioon. Siksi versionhallinta tulee integroitumaan osaksi RobotStudiota lähitulevaisuudessa. Nähtäväksi jää, millaiset ominaisuudet versionhallinta lopulta tuo RobotStudioon. (46.)

Tälläkin hetkellä ABB:n robottien ohjelmoinnissa voi hyödyntää versionhallintaa. Ohjelmointi tapahtuu korkean tason RAPID-ohjelmointikielellä. RAPID-kielellä kirjoitettu lähdekoodi sekä konfiguraatiotiedostot ovat tekstimuotoisia, joten niiden vienti versionhallintaan on helppoa ja kaikki versionhallinnan ominaisuudet ovat hyödynnettävissä. Lähdekoodi ja konfiguraatiotiedostot voidaan tal-

lentaa RobotStudiosta joko erikseen tai varmuuskopion muodossa kaikki kerrallaan. Tiedostot tallennetaan versionhallinnan työkansioon ja edetään kuten minkä tahansa versioitavien tiedostojen kanssa. Kun versionhallintaa hyödynnetään yhteistyössä muiden ohjelmoijien kanssa, tulee tarve siirtää muuttuneita tiedostoja myös versionhallinnasta RobotStudioon. Muutokset voidaan tuoda RobotStudioon, josta ne voidaan siirtää edelleen robottikontrolleriin.

Fanuc

Fanuc-robotteja voidaan ohjelmoida hyödyntäen RoboGuide-ohjelmointi- ja simulointiympäristöä. RoboGuide käyttää Fanuc Virtual Robot Controller -teknologiaa, jonka avulla käyttäjä voi luoda samanlaisen robottiohjelman kuin tuotannossa. Offline-ohjelmointia hyödyntämällä robottiohjelmat saadaan etukäteen lähes valmiiksi ennen varsinaista käyttöönottoa. (47.)

RoboGuidesta saadaan vietyä TP-ohjelmat lukukelpoisessa muodossa. Fanuc-robotteja voidaan ohjelmoida sekä TP- että Karel-ohjelmointikielellä. Robotin varmuuskopioon ei tule automaattisesti lukukelpoisia tiedostoja, mutta ohjelmista voidaan ottaa lukukelpoinen ASCII-muotoinen varmuuskopio erikseen. Lukukelpoiset ohjelmatiedostot voidaan viedä versionhallintaan ja muutokset on helppo jäljittää. Versionhallintaan voidaan viedä myös binäärimuotoisia varmuuskopioita. Silloin saavutetaan versionhallinnan hyödyt muilta osin lukuun ottamatta tarkempaa muutosten seurantaa.

KUKA

KUKA-teollisuusroboteilla on kolme järjestelmäohjelmistoa (System Software) KSS, Sunrise.OS ja iiQKA.OS (48). Eri järjestelmäohjelmistoilla on oma ohjelmointiympäristönsä. Suurin osa KUKAn teollisuusroboteista käyttää KUKA System Software (KSS) -järjestelmäohjelmistoa, jonka ohjelmointiympäristönä on KUKA WorkVisual. Sunrise-alustan teollisuusrobottien ohjelmointiympäristönä on Sunrise.Workbench. Sunrise-alusta on käytössä LRB iiwa -yhteistyörobotilla ja mobiilituotteilla (48). Uusin järjestelmäohjelmisto on iiQKA.OS, jota käytetään tällä hetkellä KUKA LBR iisy -yhteistyörobotilla. Ohjelmointi tehdään tällä alustalla robotin ohjaimen (pendant) kautta. LRB iisy on saatavilla tällä hetkellä vain rajatulle asiakasjoukolle vuonna 2022 ja saatavuus laajenee sen jälkeen (48).

Versionhallinnan hyödyntäminen on mahdollista KUKAn teollisuusroboteilla. KSS ja Sunrice.OS alustoilla on projektipohjainen tiedostorakenne, josta on mahdollista viedä (export) kaikki tiedostot

versionhallintajärjestelmään (49). Uudella iiQKA.OS alustalla ei toistaiseksi ole samanlaista projektipohjaista hallintaa kuin kahdella muulla alustalla, mutta sitä kehitetään. Samankaltaiset ominaisuudet tiedostojen viemiseksi versionhallintaan pitäisi tulla mahdolliseksi (49). Kun versionhallintaa hyödynnetään yhteistyössä muiden ohjelmoijien kanssa, tulee tarve siirtää muuttuneita tiedostoja myös versionhallinnasta projektiin. WorkVisualissa voidaan myös tuoda tiedostoja ja kansiota projektirakenteeseen. WorkVisualista muutokset voidaan siirtää robottikontrolleriin.

Omron

Omron teollisuusrobottien ohjelmointiympäristö on Automation Control Environment (ACE). Useat sovellukset voidaan konfiguroida hyödyntämällä wizard-toimintoa ilman varsinaista ohjelmointia. Vaativimmissa sovelluksissa ohjelma kirjoitetaan V+-ohjelmointikielellä. ACE sisältää emulaattoritilan, joka tarjoaa virtuaaliympäristön emuloimaan fyysisiä laitteita. Tämä mahdollistaa offline-ohjelmoinnin ilman yhteyttä fyysisiin laitteisiin. V+-ohjelmointikielen lisäksi voidaan kirjoittaa C#-ohjelmia. (50.)

ACE on tukenut 4.0 versiosta alkaen Git-versionhallintajärjestelmää (50). ACE-ohjelmointiympäristön versionhallinta toimii samalla tavalla kuin Omronin koneautomaatio-ohjaimien ohjelmointiympäristössä Sysmac Studiossa. Tarkennuksia Omronin ohjelmointiympäristöjen versionhallinnasta on kerrottu luvussa 3.4. TM Flow on Omronin yhteistyörobottien ohjelmointiympäristö. Ohjelmointi tehdään graafisesti. Yhteistyörobottien ohjelmointiympäristö ei sisällä samanlaista versionhallintaintegraatiota kuin ACE-ohjelmointiympäristössä on.

3.6 Versionhallinnan soveltuvuus robottiohjelmointiin

Seuraavaksi selvitetään, millaisia reunaehtoja sujuvalla hajautetun versionhallinnan käytöllä on teollisuusrobotiikassa. Optimaalisen käytön mahdollistamiseksi on tiettyjä vaatimuksia tai suosituksia. Millaisia ominaisuuksia teollisuusrobotin ohjelmointiympäristöstä olisi hyvä löytyä, jotta saavutettaisiin hyvä käytettävyys versionhallinnalle.

Mikäli teollisuusrobotista tai sen ohjelmointiympäristöstä voidaan viedä ja tuoda ohjelmakoodi tekstimuotoisena, hyödyttää se hajautetun versionhallinnan käyttöä. Vientiä tarvitaan, kun muutokset halutaan tallentaa versionhallintajärjestelmään. Tuontia tarvitaan, kun halutaan tuoda jonkun muun tekemät muutokset robotille tai sen ohjelmointiympäristöön.

Teollisuusroboteissa on työohjelmien lisäksi konfiguraatiotiedostoja, joihin on tallennettu tietoa robottijärjestelmästä. Konfiguraatiotiedostoissa voidaan määritellä useita robottijärjestelmäkohtaisia parametrejä, jotka liittyvät kommunikointiin, signaaleihin, liikkeenohjaukseen ja robotin asetuksiin. Mikäli nämä tiedot saadaan vietyä robotista, saadaan versionhallintaan tallennettua hyödyllistä tietoa konfiguraatiomuutoksista.

Yhteistyörobottien ohjelmointi poikkeaa usein tavanomaisten teollisuusrobottien ohjelmoinnista. Teollisuusroboteilla ohjelma kirjoitetaan usein tekstinä, ja ohjelmakoodi voidaan viedä ohjelmointiympäristöstä ja tuoda ohjelmointiympäristöön versionhallinnasta. Yhteistyöroboteilla ohjelmointi tehdään graafisesti, jolloin ohjelmakoodin vienti ja tuonti ei ole niin yksinkertaista. Etenkin eri versioiden yhdistäminen voi olla vaikeaa. Jotta versionhallinta toimisi sujuvasti, pitäisi versionhallinta integroida graafisen ohjelmoinnin ohjelmointiympäristöön. Logiikkaohjelmoinnin versionhallinnassa integraatioita on olemassa eri ohjelmointiympäristöissä, mutta yhteistyörobottien osalta siinä on havaittavissa puute. Teknologia on jo olemassa versionhallinnan hyödyntämiseksi graafisessa ohjelmoinnissa, joten lienee vain ajan kysymys, koska versionhallinta on ketterästi hyödynnettävissä myös yhteistyöroboteilla.

4 TUTKIMUKSEN TOTEUTUS

Tutkimuksen perustan luovat tieteenfilosofiset taustaoletukset, ja ne määrittävät tutkimuksen tavoitteita, toteutustapoja sekä tuloksia (51, s. 25). Tutkimuksen tekijällä on aina, joko implisiittiset tai eksplisiittiset tieteenfilosofiset taustaoletuksensa (51, s. 25). Tämän luvun tavoitteena on kertoa eksplisiittisesti tämän tutkimuksen tienteenfilosofisista taustaoletuksista, minkälaisia ovat tämän tutkimuksen tutkimusmetodologiset valinnat ja kuinka niihin on päädytty.

Tutkimusmenetelmänä käytettiin laadullista tutkimusmenetelmää, koska tiedonintressinä oli saavuttaa tietoa, joka auttaa ymmärtämään tutkittavan ilmiön merkitystä ja synnyttämään uutta ajattelua. Emansipatorinen tiedonintressi on vanhojen ajatusmallien kyseenalaistamista ja hermeneuttinen tiedonintressi ilmiön selittämistä ymmärrettäväksi, niin että syntyy uutta ajattelua (52, s. 120). Näistä molemmista tiedonintresseistä on kyse tässä tutkimuksessa. Laadullisen tutkimusmenetelmän erityispiirteenä on selvittää välittömän havainnoinnin ulottumattomissa olevia asioita (52, s. 96). Tutkimuksen heuristisella työotteella tavoiteltiin omista ennakkokäsityksistä luopumista ja mahdollisimman kokonaisvaltaista ilmiön ymmärtämistä. Heuristisen työotteen tavoitteena oli myös päästä ketterästi riittävän lähelle parasta mahdollista toteutusta. Käytännössä ennakkokäsityksistä ja asenteista pyrittiin luopumaan hermeneuttisen kehän avulla.

4.1 Metodologia

”Metodologia on oppi tieteen menetelmistä” (51, s. 37). Tutkimusmetodologia (kuva 16) voidaan käsittää tapana, jolla tutkimusta lähestytään. Toisaalta metodologialla voidaan viitata siihen, miten uutta tietoa hankitaan. Metodilla tarkoitetaan tiettyä toimintaa tietyn tehtäväkokonaisuuden toteuttamiseksi (52, s. 179). Tässä tutkimuksessa aineiston analyysiin käytettiin hermeneuttista metodia. Tutkimuksen tavoite ohjaa tutkimusmetodien valintaa (51, s. 14).

*KUVA 16. Metodologia on osa tieteenfilosofiaa ja on suppeasti määriteltynä oppijärjestelmä metodeista uuden tiedon hankkimiseksi todellisuudesta (*52, *s. 182; 57, s. 20)*

Heuristiikalla tarkoitetaan metodeja tai strategioita ongelman ratkaisemiseksi (53). Tutkimuksessa on kyse tutkimusongelman ratkaisusta tai tutkimuskysymyksiin vastaamisesta (52, s. 183). Heuristiikka voidaan yleisellä käytännön tasolla käsittää olevan merkityksellisimmän tiedon poimimista ongelman ratkaisemiseksi (53). Esimerkiksi tietyn laitteen toiminnon selvittämiseksi ei välttämättä tarvitse lukea kokonaista käyttöohjetta, vaan selata käyttöohje siihen kohtaan, mistä tieto löytyy ja selvittää toiminto. Tällaista toimintatapaa voidaan kutsua heuristiseksi. Yritys ja erehdys sekä nyrkisäännöt edustavat heuristisia metodeja (53). Tutkimuksen heuristinen työote toteutuu tapana poimia mahdollisimman merkityksellistä tietoa tutkimusongelman ratkaisemiseksi. Kaikki kyselylomakkeella kerätty data ei välttämättä ole oleellista tutkimuksen kannalta, mutta tietoa kerättiin sen hetkisen suunnitelman mukaisesti riittävän laajasti, jotta data olisi käytettävissä, jos tarve siihen ilmenee.

Heuristiikalla, niin kuin kaikella, on hyvät ja huonot puolensa. Heuristisella työotteella voidaan saavuttaa luotettava tulos tutkimukselle, mutta ei välttämättä täydellistä tulosta (53). Heuristisen työotteen tavoite ja käyttötarkoitus oli saavuttaa luotettava tutkimustulos riittävän hyvällä laadulla. Laadulla tarkoitetaan tässä yhteydessä tutkimuksen pätevyyttä. Pätevyys on ennen kaikkea tutkimuksen merkityksellisin tekijä, jotta tutkimus saavuttaa arvonsa. Tutkimuksen heuristisella työotteella

haluttiin kokeilla eri lähestymissuuntia ja samalla yrittäen löytää uutta näkökulmaa ja ajattelua tutkimuksen toteutukseen.

Hermeneuttisessa metodissa tutkija käy vuoropuhelua tutkimusaineistonsa kanssa. Vuoropuhelu muodostaa kehän, jota kutsutaan hermeneuttiseksi kehäksi (kuva 18). Tutkija ei koita suostutella aineistoa vastaamaan omaa alkuolettamaa, vaan tulkintoja tehdään useassa vaiheessa ja sen uskottavuutta arvioidaan. Tästä muodostuu kehä, joka johtaa ajatteluun ja sen avulla ymmärtämiseen ja tulkinnan korjaamiseen sekä syventämiseen. Jokainen tulkintakierros vie lähemmäksi tutkittavaa ilmiötä ja sen kokonaisvaltaisempaa ymmärtämistä. Tutkimustekstiä kirjoitetaan jokaisen tulkintakierroksen jälkeen, kun tutkija kokee, että on tulkinnut asian uudella tavalla. (52, s. 143.)

Heuristinen työote hermeneuttisen kehän muodossa johti uuden tiedon saavuttamiseen ja uudenlaiseen ajatteluun. Ajattelun sai aikaan uusia näkemyksiä, joille haettiin merkitystä tutkimuskehyksessä. Jokainen vuoropuhelukerta tutkimusaineiston kanssa muodosti uuden kehän, joka muokkasi vastauksen mallintamista tutkimuskysymyksiin. Tarkemmin aineiston analysoinnista on kerrottu luvussa 4.5.

4.2 Tutkimustehtävä

Tutkimuksen perusta on hyvin määritellyssä tutkimustehtävässä. Tutkimuksessa on kyse tutkimusongelman ratkaisusta tai tutkimuskysymyksiin vastaamisesta. Tutkimuskysymykset koostuvat pääkysymyksestä sekä alakysymyksistä, joita voidaan kutsua myös teoreettisiksi tutkimuskysymyksiksi. Pääkysymys eli itse tutkimusongelma on johdettu asiaongelmasta ja teoreettiset tutkimuskysymykset tutkimusongelmasta. Teoreettiset tutkimuskysymykset ovat kysymyksiä, joihin tutkimuksella halutaan vastata. (52, s. 51, 183; 54.)

Tutkimuksen asiaongelma muodostui siitä lähtökohdasta, miksi teollisuusrobotiikassa versionhallinta ei ole niin suuressa merkityksessä tai niin suuresti esillä kuin muun ohjelmoinnin alalla. Vaikka versionhallinta on suuressa roolissa muussa ohjelmoinnissa ja selkeästi kasvavassa roolissa PLC ohjelmoinnissa, niin teollisuusrobottien ohjelmoinnissa se ei esiinny yhtä selkeästi. Asiaongelmasta muodostui tutkimuksen pääkysymys eli tutkimusongelma, josta muodostettiin tutkimuskysymykset.

Tutkimusongelmasta muodostettiin kolme tutkimuskysymystä:

- Millaisissa yrityksissä hyödynnetään teollisuusrobotiikan ohjelmoinnissa hajautetun versionhallinnan potentiaalia?
- Mitkä ovat ne tekijät, jotka vaikuttavat versionhallinnan käyttöön tai käyttämättömyyteen yrityksissä?
- Onko teollisuusrobottien ohjelmointiin mahdollista tuoda kaikki versionhallinnan edut?

4.3 Rajaus

Tutkimuksen kohteena oli teollisuusrobottiohjelmakoodin ja robotin konfiguraatiotiedostojen versionhallinta. Robotin konfiguraatiotiedostoilla tarkoitetaan robottiin määriteltyjä signaaleja, kenttäväyläkonfiguraatiota, robotin asentotietoa suhteessa ympäristöön ja muita tallennettavia parametreja, joita teollisuusrobotti tarvitsee tarkoituksenmukaiseen toimintaansa. Tutkimuksessa ei tutkittu PLC:n tai muiden välitason ohjausjärjestelmien versionhallintaa. Tutkimuksessa ei tutkittu erikseen yhteistyörobottien versionhallintaa, mutta viitekehyksessä tuotiin esille havaintoja yhteistyörobottien ohjelmointiympäristön soveltuvuudesta versionhallintaan.

Primääriaineiston kerääminen kohdistettiin suomalaisiin teknologiateollisuuden yrityksiin, jotka työskentelevät robotiikan parissa. Aineistonkeruujoukkoon ei sisällytetty suuria ulkomaisia teollisuusrobottivalmistajia, koska tarkoitus ei ollut tutkia robotin valmistus- ja kehitysvaiheen versionhallintaa, vaan robotin sovellusvaiheen versionhallintaa.

Sekundääriaineistossa sivutaan PLC:n versionhallintaa, jotta saadaan näkemys, millä tasolla PLC:n versionhallinta on teollisuusrobotiikan versionhallintaan verrattuna. Tarkempaan tarkasteluun, versionhallinnan tämänhetkisistä mahdollisuuksista teollisuusrobottivalmistajien ohjelmointiympäristöissä, valittiin neljä yleisimmistä teollisuusrobottimerkeistä. Valitut robottimerkit aakkosjärjestyksessä olivat ABB, Fanuc, KUKA ja Omron.

4.4 Aineiston keruu

Tutkimuksen primääriaineisto kerättiin internetkyselyllä joulukuun 2021 ja tammikuun 2022 välisenä aikana hyödyntäen Webropol kyselyalustaa. Kysely lähetettiin 17 sattumanvaraisesti valitulle suomalaiselle yritykselle, jotka ovat tekemisissä teollisuusrobotiikan kanssa. Kyselyyn vastasi 12

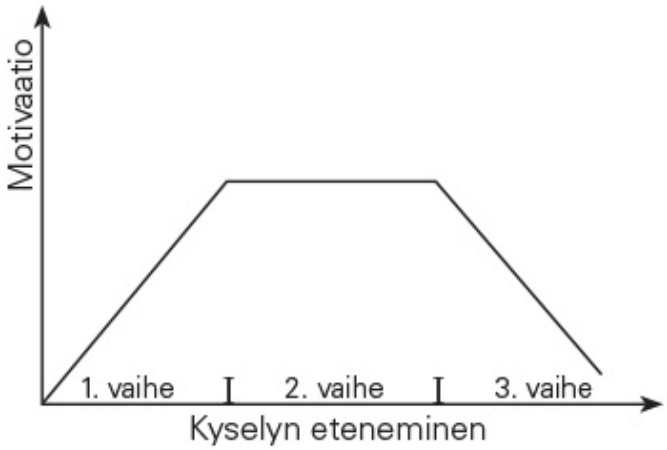
yrityksen edustajaa. Sähköposti osoitettiin henkilökohtaisesti yrityksen edustajalle. Sähköpostissa pyydettiin välittämään viesti yrityksen henkilölle, jolla olisi parhaat tiedot vastata kyselyyn. Kohdejoukossa oli robottien loppukäyttäjiä, robotti-integraattoreita ja robottivalmistajia. Sähköpostilla yleisimpien teollisuusrobottien valmistajille lähetetyllä kyselyllä kerättiin sekundääriaineistona tietoa integraattoreille suunnattujen teollisuusrobottimerkkien mahdollistamasta versionhallinnasta. Sähköpostikyselyllä saatu tieto on kerrottu luvussa 3.5 Teollisuusrobottien versionhallinta

Kyselylomake on tyypillisempi tapa kerätä tietoa määrällisessä tutkimuksessa kuin laadullisessa tutkimuksessa. Kysely tiedonkeruun tapana kuitenkin valittiin tähän laadulliseen tutkimukseen sen nopeuden ja taloudellisuuden vuoksi. Internet-kyselyllä hankittu aineisto on valmiiksi digitaalisessa muodossa ja näin ollen helppo käsitellä. Digitaalisen kyselyaineiston analysoiminen onnistuu myös helpommin kuin haastatteluaineiston. Kyselyn laatiminen vaatii tarkkuutta, jotta sillä voidaan saavuttaa tutkimukselle asetetut tavoitteet ja että tutkimus olisi pätevä. Aineiston kerääminen kyselylomakkeella ajateltiin olevan riittävä tutkimuksen tavoitteen saavuttamiseksi.

Tutkimuksen alussa on varmistuttava siitä, että tutkittava ilmiö on mitattavissa ja testattavissa. Tutkittavan ilmiön muuttamista mitattavaan muotoon kutsutaan operationalisoinniksi. Käytännössä tämä tarkoittaa, että teoreettiselta tasolta siirrytään empiiriselle tasolle eli kyselylomakkeelle. Kun tuloksia tulkitaan, toimitaan päinvastoin siirtymällä kyselylomakkeelta takaisin teoreettiselle tasolle. (52, s. 83.)

Vaikka laadullisessa tutkimuksessa ei varsinaisesti tunneta mittaamista tai operationalisointia, niin silti teoreettiset käsitteet tulee muuttaa empiiriseen muotoon, jotta ilmiötä voidaan tutkia (55). Vaikka käsitteestä operationalisointi ei laadullisen tutkimuksen tapauksessa ehkä puhutakaan, niin käytännössä operationalisointia käytettiin, kun kyselylomakkeelle laadittiin näkemystä ja mielipidettä mittaavat Likert-asteikkoa käyttävät kysymykset.

Kyselylomake laadittiin siten, että alkuun sijoitettiin taustoittavat kysymykset, joihin oli helppo vastata. Helpoilla alkukysymyksillä pyrittiin kasvattamaan vastaajan motivaatiota kyselyn edetessä (kuva 17). Kyselyn kohdassa, jossa mitattiin mielipiteitä versionhallinnasta, käytettiin Likert-asteikkoa. Likert-asteikossa on tyypillisesti viisi vastausvaihtoehtoa, joista keskimmäisin on neutraali (56, s. 92). Parittomuuden ajatus on antaa vastaajalle vaihtoehto olla ottamatta kantaa kysymykseen (56, s. 92). Vastausvaihtoehdot nimettiin sanallisesti, jotta vastaamisessa ei tulisi lomakkeen tulkintavirheitä.



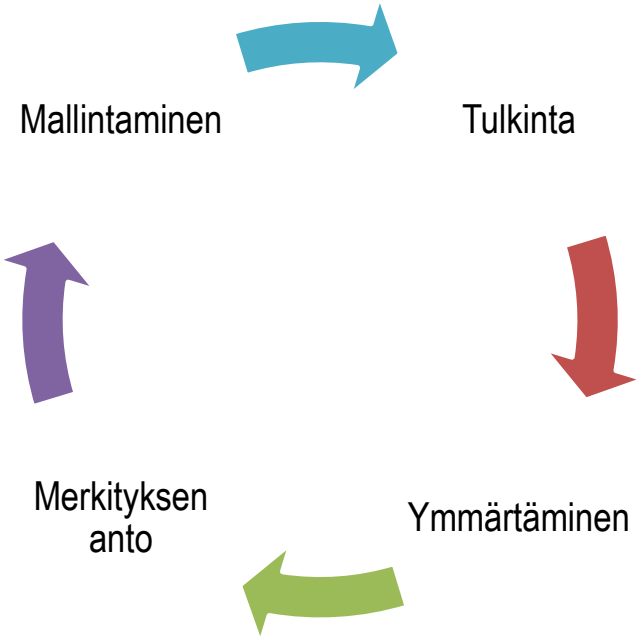
KUVA 17. Motivaation määrä kyselyn edetessä (56, s. 82)

Kysymykset valittiin kyselyyn tutkimusongelman ja siitä johdettujen teoreettisten tutkimuskysymysten pohjalta. Kyselylomakkeen (liite 2) alussa hankittiin taustatietoa vastaajasta, yrityksestä ja minkä valmistajan robotteja he tyypillisesti käyttävät. Taustoittavilla kysymyksillä oli tarkoitus saada tietoa, millaisia vaikutuksia taustatiedoilla on versionhallinnan hyödyntämiseen. Lean-filosofian ja ohjelmistokehityksen ketterien menetelmien kartoituksella oli tarkoitus saada tietoa, onko niillä yhteyttä tutkimusongelmaan. Loput kysymykset liittyivät versionhallinnan käytön tai käyttämättömyyden selvittämiseen sekä syiden taustojen selvittämiseen.

Seuraavassa vaiheessa kartoitettiin suppeasti kokemusta Lean-filosofiasta ja ohjelmistokehityksen -ketteristä menetelmistä. Ajatuksena oli tutkia, liittyykö Lean-filosofian tai ketterien menetelmien osaaminen versionhallinnan hyödyntämiseen. Seuraavana esitettiin yksi oleellisimmista kysymyksistä, joka oli: onko yrityksessä käytössä jokin versionhallintajärjestelmä? Vastaus vaikutti jatkokysymyksiin, joissa ei-vastanneilta kartoitettiin tarkemmin syitä miksi käytössä ei ole versionhallintajärjestelmää. Mikäli vastaus oli kyllä, kartoitettiin mikä järjestelmä yrityksellä on käytössä ja miten se on soveltunut yrityksen käyttöön. Kyselylomakkeessa oli lisäksi vapaatekstikenttiä täydentämään vastauksia. Lopuksi kaikilta kyselyyn osallistuneilta kysyttiin, onko heillä jotain muuta mitä he haluaisivat mainita versionhallinnasta. Kysymyksen tarkoitus oli saada tärkeää lisätietoa siitä, mitä kyselylomakkeella ei ollut huomattu kysyä.

4.5 Aineiston analysointi

Aineiston analysoinnissa käytettiin metodeina luokittelua ja hermeneuttista analyysiä. Hermeneuttinen analyysi toteutettiin hermeneuttisen kehän muodossa (kuva 18). Hermeneuttisessa analyysissä tulkinta etenee tutkimusaineiston yksityiskohdista kokonaisuuteen ja takaisin, jonka jälkeen tulkinta kohdistetaan tulkitsijan viitekehyksen kautta aineistoon ja takaisin viitekehykseen (57, s. 109). Aineisto itsessään ei ole tutkimuksen tieto vaan vuoropuhelu aineiston kanssa tuottaa tiedon (58, s. 31).



KUVA 18. Hermeneuttisen analyysin eteneminen hermeneuttisen kehän muodossa *(mukaillen* 57, *s. 110)*

Ennakkokäsitys tutkittavasta aiheesta oli, ettei versionhallintaa juurikaan käytetä, versionhallinta tunnetaan heikosti ja sen käyttöönotto ja koetaan vaikeana. Kun metodologiset taidot olivat työn edetessä kehittyneet, oli aika pyrkiä irti omista ennakkokäsityksistä ja asenteista ja suunnata ajattelu kriittiseen ja reflektiiviseen ajatteluun.

Ensimmäisen kerran aineistoon tutustuttiin heti kun vastausaika oli umpeutunut. Tarkastelukerta oli lähinnä yleiskatsaus siihen, moniko kyselyyn oli vastannut. Jo tässä vaiheessa oli helppo irtaantua osasta omia ennakkokäsityksiä. Siitä alkoi muodostua uusi työhypoteesi, jota koeteltiin uusilla

vuoropuheluilla tutkimusaineistoon. Jokaisella hermeneuttisen kehän kierroksella aineistoa koitettiin lähestyä eri näkökulmasta. Ymmärtäminen oli osa jokaista kierrosta. Ymmärrystä korjattiin sitä mukaan, kun tulkinta toi uutta tietoa tutkimusongelmaan. Sitä mukaa ymmärrys myös syventyi.

Ennen varsinaista aineiston analysointia aineistoa luokiteltiin, jotta vuoropuhelu aineiston kanssa helpottuisi. Luokittelussa aineistoa jaettiin luokkiin, joka helpotti tiedon tuottamista aineistosta. Erilaisia luokkia kirjoitettiin taulukkoon ja niihin koottiin kyselyaineistosta vastauksia. Käytetyt luokat muodostuivat robottimerkeistä, yrityksen koosta, Lean- ja Agile-menetelmien osaamisesta sekä eri rooleista. Luokittelua pystyi tekemään myös Webropol-kyselyalustan työkaluilla, mikä helpotti työtä.

Varsinainen analyysi eteni hermeneuttisen kehän muodossa (kuva 18). Kehän kerroksilla luotiin ja tarkennettiin työhypoteesia, jota koeteltiin uudella hermeneuttisen kehän kierroksella. Kun hypoteesin todettiin kestävän tarkastelun, muodostui todennäköisin ja uskottavin tulkinta vastaamaan tutkimuskysymyksiin.

4.6 Tutkimusetiikka, tutkimuksen pätevyys ja luotettavuus

”Tutkimusetiikalla tarkoitetaan eettisesti vastuullisten ja oikeiden toimintatapojen noudattamista” (59). Tässä tutkimuksessa noudatetaan Tutkimuseettisen neuvottelukunnan (TENK) Hyvä tieteellinen käytäntö -ohjeistusta parhaalla mahdollisella tarkkuudella. Lisäksi tutkimuksessa on huomioitu Ammattikorkeakoulujen opinnäytetöiden eettiset suositukset.

Aineiston anonymisointi huomioitiin ennen aineistonkeruuvaiheen aloittamista. Kutsukirjeessä (liite 1) osallistujille kerrottiin tutkimuksen tausta, tavoite ja kuinka tulokset julkaistaan. Kutsussa oli yhteinen linkki kyselylomakkeeseen kaikille tutkimukseen kutsutuille eikä yhteystietoja kerätty aineistoon. Kirjeessä mainittiin, ettei tuloksia yksilöidä ja tutkimusaineisto säilytetään vain työn arviointiin saakka. Kyselylomakkeeseen (liite 2) pääsy oli salasanalla suojattu, jotta kyselyyn vastaisivat vain kutsutut asiantuntijat. Tulokset säilytettiin suojatussa Webropol ympäristössä.

Tutkimuksen mahdolliset rahoituslähteet tulee ilmoittaa (60, s. 23). Työn tilaaja — Algol Technics on tukenut tutkimuksen tekemistä tarjoamalla työaikaa tutkimuksen tekemiseen. Valtaosa tutkimuksesta on tehty työajan ulkopuolella.

Tutkimusetiikka asettaa vaatimuksia myös tutkimuksen toteutustavasta. Tutkimuksen tekemisen tulee olla kurinalaista, järjestelmällistä sekä täsmällistä (52, s. 25). Työn eteneminen suunnitelmallisesti, ohjauspalaverien koollekutsuminen säännöllisesti ja määrätietoinen itsenäinen työ ovat osoituksena edellä mainittujen ehtojen toteutumisesta.

Pätevyys

Tutkimuksen pätevyyteen vaikuttaa kuinka hyvin tutkimusote ja valitut metodit vastaavat tutkittavaa ilmiötä (61, s. 3). Tutkimus on pätevä, kun tutkimus mittaa juuri sitä ilmiötä, jota ollaan tutkimassa (61, s. 3). Heuristinen työote ohjasi kysymysten asettelua kyselylomakkeelle. Käytetty työote ja metodit tukivat tutkimuksen tavoitteen saavuttamisessa. Mittauksella saavutettiin sille asetetut tavoitteet. Tutkimuksessa pystyttiin vastaamaan tutkimuskysymyksiin eli selvittämään, millaisissa yrityksissä hyödynnetään teollisuusrobottien ohjelmoinnissa hajautetun versionhallinnan potentiaalia, mitkä ovat ne tekijät, jotka vaikuttavat versionhallinnan käyttöön ja selvittämään onko teollisuusrobottien ohjelmointiin mahdollista tuoda kaikki versionhallinnan edut.

Luotettavuus

Kun tutkimuksen tekijän ja tutkittavien käsitykset vastaavat toisiaan, voidaan laadullista tutkimusta pitää luotettavana (52, s. 154). Laadullisen tutkimuksen luotettavuuteen vaikuttaa kaikki siinä tehdyt teot, valinnat ja ratkaisut (52, s. 155). Kyselylomakkeen kysymysten asettelulla ja selkeästi kuvatuilla vastausvaihtoehdoilla pyrittiin minimoimaan tutkimuksen tekijän ja tutkittavien ajatusten erkaantuminen. Metodit pyrittiin valitsemaan mahdollisimman hyvin soveltuviksi tutkimuksen tavoitteen saavuttamiseksi. Työssä noudatettiin hyvää tutkimusetiikkaa, joka on kuvattu edellä. Tutkimuksen tuloksissa perustellaan, miten lopullisiin ratkaisuihin on päädytty ja arvioidaan niiden toimivuutta.

5 TUTKIMUKSEN TULOKSET

Tutkimuksen tuloksissa esitellään ensin analyysin tulokset. Analyysin tuloksien pohjalta esitellään tutkimuksen johtopäätökset ja tulkinnat tutkittavasta aiheesta. Näiden yhteyttä muuhun tutkimusaineistoon pohditaan ja argumentoidaan. Varsinainen tieto ja tutkimuksen tulos ilmenee johtopäätöksien ja tulkinnan kautta. Vastaukset tutkimuskysymyksiin tulevat esille johtopäätöksistä ja tulkinnasta.

Tutkimusaineisto kerättiin Webropol-kyselytyökalulla tehdyllä kyselylomakkeella. Kysely lähetettiin sähköpostitse 17 teollisuusrobotiikan parissa työskentelevän suomalaisen teknologiateollisuuden yrityksen edustajalle. Kyselyyn vastasi yhteensä 12 yrityksen edustajaa, joista yrityksen johdosta oli 1 henkilö, teknologiajohdosta 3 henkilöä, automaatio-, sähkö- tai robotiikkainsinöörejä 6 henkilöä ja muuhun ammattiryhmään kuuluvia 2 henkilöä. Suurin osa (n=9) vastaajista merkitsi roolikseen (kyselylomakkeessa toimialavalinta) robotisoitujen koneiden rakennuksen. Loput olivat robottien valmistajia (n=2) tai loppukäyttäjiä (n=1). Ennakkokäsitys tutkittavasta aiheesta oli, ettei versionhallintaa juurikaan käytetä, versionhallinta tunnetaan heikosti ja sen käyttöönotto sekä käyttö koetaan vaikeana. Tutkimusaineiston analyysin tulos — joka muutti heti ennakkokäsitystä — oli se, että puolet vastaajista hyödynsivät jo jotain versionhallintajärjestelmää.

5.1 Versionhallinta teollisuusrobottiohjelmoinnissa tällä hetkellä

Kaikille vastaajille versionhallinta on vähintäänkin osittain tuttu käsite. Suurin osa (n=11) pitää käsitettä versionhallinta tuttuna. Versionhallintaa hyödyntää 50 % (n=6) vastaajista teollisuusrobotin työohjelmien ohjelmoinnissa. Lisäksi yksi vastaajista hyödyntää versionhallintaa muussa kuin robotin työohjelmien ohjelmoinnissa.

Hajautettu versionhallinta on käytössä kolmella vastaajista. Käytössä oleviksi järjestelmiksi oli vastattu Git, GitHub ja Bitbucket. Bitbucket on tukenut Mercurial-versionhallintajärjestelmää vuoteen 2020 saakka. Mercurial on hajautettu versionhallintajärjestelmä, kuten aiemmin todettiin. Bitbucket on tukenut Git-versionhallintajärjestelmää vuodesta 2011 lähtien. Git on myös hajautettu version-

hallintajärjestelmä. Koska Bitbucketin voidaan katsoa hyödyntävän Gitiä ja GitHub on Git-version-hallintaa hyödyntävä palvelu, Gitin voidaan katsoa olevan käytetyin versionhallintajärjestelmä ja hajautetun versionhallinnan käytetyin versionhallintatapa.

Keskitetty versionhallinta on käytössä yhdellä vastaajista. Järjestelmäksi oli vastattu SVN eli Subversion-versionhallintajärjestelmä. Yhdessä vastauksessa versionhallintajärjestelmäksi mainitaan omavalmiste, jonka versionhallintamenetelmä ei käy aineistosta ilmi. Yksi vastaajista on vastannut versionhallintajärjestelmäksi TC, joka viitannee TeamCity-integrointipalvelimeen (build management and continuous integration server). TeamCity tukee useita versionhallintajärjestelmiä, kuten Gitiä, Mercurialia, SVN:ää ja CVS:ää. Aineistosta ei käy ilmi käytetty versionhallintamenetelmä.

Käytetyin teollisuusrobottien versionhallintamenetelmä on hajautettu versionhallinta. Se korreloi hyvin muun ohjelmistokehityksen suuntauksen kanssa, jossa hajautettu versionhallinta on myös käytetyin versionhallintamenetelmä. Hajautetun versionhallinnan suosioon voi vaikuttaa juuri sen yleisyys. Useat versionhallintapalvelut tukevat hajautettua versionhallintaa, ja siihen löytyy paljon oppaita ja esimerkkejä. Aiempi kokemus, muusta kuin robottien ohjelmoinnista, on voinut tuoda kokemusta hajautetun versionhallinnan käytöstä, jolloin se on luonteva vaihtoehto myös teollisuusrobottien versionhallintaan.

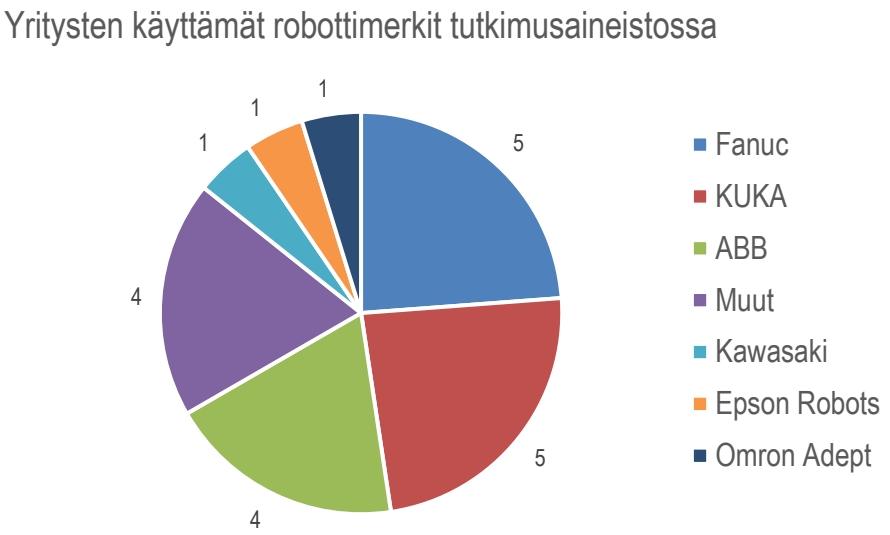
Taulukosta 1 nähdään, että versionhallintajärjestelmän käyttäjiä teollisuusrobottien työohjelmille on loppukäyttäjissä ja teollisuusrobottien integraattoreissa. Omia robotteja valmistavat yritykset (n=2) vastasivat, ettei käytössä ole versionhallintajärjestelmää robotin työohjelmille. Versionhallintaa kuitenkin käytetään muussa ohjelmointityössä ainakin yhden robottivalmistajan toimesta.

> *”Käytämme ohjelmointi työssä päivittäin versionhallintaa. Itse ajo-ohjelmien versiointi ei ole ollut vielä tarpeellista.”*

TAULUKKO 1. Versionhallinnan käyttö luokiteltuna rooleittain

			Versionhallintajärjestelmä			
		N	Käytössä	Ei käytössä	Käytössä	Ei käytössä
Rooli	Loppukäyttäjät	1	100 %	0 %	1	0
	Integraattorit	9	56 %	44 %	5	4
	Robottien valmistajat	2	0 %	100 %	0	2

Tutkimuksen tyypillisimpien robottimerkkien osalta otanta edustaa hyvin kansainvälistä teollisuusrobottien markkinatutkimusta vuodelta 2022, jossa avainyhtiöinä mainitaan muun muassa Fanuc, ABB, KUKA, Kawasaki ja Epson Robots (36). Tämän tutkimuksen tutkimusaineiston teollisuusrobottimerkit ovat kohdistuneet yleisimpiin robottimerkkeihin (kuva 19), jolloin yleistettävyys yleisimpien robottimerkkien osalta on parempi.



KUVA 19. Yritysten käyttämät robottimerkit tutkimusaineistossa

Kyselyssä oli mahdollista valita yksi tai useampia käytössä olevia robotteja. Taulukosta 2 nähdään, että käytössä olevia robotteja (n=21) on sen vuoksi enemmän kuin vastaajia. Useimmilla robottimerkeillä on sekä versionhallinnan käyttäjiä että niitä, joilla ei ole sitä käytössä. Tämä osoittaa käytännössä, että versionhallinta on mahdollista useimmilla teollisuusroboteilla. Aineiston perusteella vaikuttaisi siltä, ettei teollisuusrobottimerkki suoraan vaikuta siihen, käytetäänkö versionhallintaa vai ei. Epson Robots on ainoa poikkeus, jonka käytön yhteydessä ei hyödynnetä versionhallintaa. Epson Robots -robottimerkin käyttöön on tullut vain yksi vastaus. Luokitellusta tutkimusaineistosta voi havaita, että Epson Robots -käyttäjä ei ole ehtinyt perehtyä versionhallintaan, joten siitä ei voi tehdä päätelmiä, onko robottimerkki vaikuttamassa versionhallinnan käyttämättömyyteen. ABB-, Fanuc- ja etenkin KUKA-roboteista tulee esille selkeä viittaus siihen, että versionhallinta onnistuu kyseisillä roboteilla.

TAULUKKO 2. Versionhallinnan käyttö robottimerkeittäin luokiteltuna

			Versionhallintajärjestelmä			
Yritysten käyttämä robotti		N	Käytössä	Ei käytössä	Käytössä	Ei käytössä
	ABB	4	75 %	25 %	3	1
	Fanuc	5	60 %	40 %	3	2
	KUKA	5	100 %	0 %	5	0
	Epson robots	1	0 %	100 %	0	1
	Omron Adept	1	100 %	0 %	1	0
	Kawasaki	1	100 %	0 %	1	0
	Muut	4	25 %	75 %	1	3

Täyttä varmuutta ei saada siitä, hyödynnetäänkö versionhallintaa kaikilla robottimerkeillä, jotka vastaaja on valinnut. On mahdollista, että versionhallinta on käytössä vain yhdelle robottimerkille, mutta vastaaja on valinnut kaikki robottimerkit, joita yritys käyttää. Tästä seuraa jossain määrin epävarmuutta tulkintaan robottimerkin vaikutuksesta versionhallinnan käyttöön.

Versionhallinnan käyttö vaikuttaisi korreloivan yrityksen kokoon: etenkin suuryritykset hyödyntävät versionhallintaa, ja pienissä yrityksissä versionhallinnan hyödyntäminen oli vähäisintä (taulukko 3). Kaikista yrityskokoluokista kuitenkin löytyy versionhallinnan hyödyntäjiä, joten voidaan sanoa, että versionhallinnan hyödyntäminen on mahdollista kaiken kokoisissa yrityksissä.

TAULUKKO 3. Versionhallinnan käyttö yrityksen koon mukaan luokiteltuna

			Versionhallintajärjestelmä			
Yrityksen koko		N	Käytössä	Ei käytössä	Käytössä	Ei käytössä
	Suuryritykset	4	100 %	0 %	4	0
	Keskisuuret yritykset	2	50 %	50 %	1	1
	Pienet yritykset	6	17 %	83 %	1	5

Mitä tunnetumpia Lean- ja Agile-menetelmät ovat, sitä korkeampi käyttöaste versionhallinnalla on (taulukko 4). Osaamisen ja versionhallinnan käytön välillä on havaittavissa korrelaatiota, mutta ei välttämättä kausaliteettia. Yrityksissä, joissa Lean-menetelmät tunnetaan, on voinut syntyä ajattelua hukan poistamisesta ja työn tehostamisesta versionhallinnan avulla. Toisaalta voidaan ajatella, että versionhallintaa hyödyntävissä yrityksissä on henkilöitä, joilla on ollut mahdollisuus perehtyä

enemmän työtä tukeviin ja edistäviin metodeihin, kuten versionhallinnan käyttöön ja Lean-menetelmiin.

TAULUKKO 4. Versionhallinnan käyttö Lean- ja Agile-osaamisen mukaan luokiteltuna

			Versionhallintajärjestelmä			
		N	Käytössä	Ei käytössä	Käytössä	Ei käytössä
Lean-osaaminen	erittäin hyin, hyvin	4	100 %	0 %	4	0
	jonkin verran, vähän	6	33 %	67 %	2	4
	ei tuttua	2	0 %	100 %	0	2
Agile-osaaminen	erittäin hyvin, hyvin	4	75 %	25 %	3	1
	jonkin verran, vähän	6	37 %	63 %	3	3
	ei tuttua	2	0 %	100 %	0	2

Saman robotin parissa työskentelevien määrällä ei vaikuta olevan merkitystä versionhallinnan käyttöön. Luokitellusta aineistosta on havaittavissa, että suhteellisesti eniten versionhallintaa hyödynnetään yrityksissä, joissa saman robotin parissa työskentelee vain yksi henkilö. Versionhallinta on hyödyllistä sekä tiimeissä että yksittäisten henkilöiden käyttämänä. Kaikki versionhallinnan edut ovat mahdollisia yhden henkilön käyttämänä. Jos samaa koodipohjaa käyttää useampi henkilö, on etenkin hajautetun versionhallinnan etuna ketterä yhteistoiminta tiimin sisällä.

5.2 Tulkinnat ja päätelmät analysoidusta aineistosta

Kun versionhallinnan käyttöaste on analysoitu eri luokittelujen näkökulmasta, perehdytään tarkemmin, millaisissa yrityksissä versionhallintaa ei hyödynnetä ja millaisissa hyödynnetään. Mitkä ovat keskeiset syyt käyttöön tai käyttämättömyyteen ja millaisena työnteko koetaan versionhallinnan kanssa tai ilman. Tavoitteena on esittää tulkinta versiohallinnan käyttöön vaikuttavista tekijöistä ja tehdä johtopäätöksiä tutkittavasta ilmiöstä.

5.2.1 Versionhallinnan käyttämättömyyteen vaikuttavat tekijät

Tässä luvussa tutkitaan tarkemmin, millaisissa yrityksissä ei hyödynnetä teollisuusrobotiikan ohjelmoinnissa versionhallintaa ja millaisena työskentely ilman versionhallintaa koetaan. Lisäksi perehdytään tärkeimpiin syihin, jotka vaikuttavat siihen, ettei versionhallinta ole käytössä.

Tutkimusaineiston robottimerkillä ei ole vaikutusta versionhallinnan käyttöön. Sen sijaan käyttöaste on vähäistä pienissä yrityksissä, joissa ei ole paljoa kokemusta Lean- ja Agile-menetelmistä. Versionhallintaa hyödynnetään suhteellisesti enemmän suurissa ja keskisuurissa yrityksissä. Tähän voi vaikuttaa se, että suurissa ja keskisuurissa yrityksissä asiantuntijoilla voi olla enemmän aikaa selvittää ja perehtyä erilaisiin työntekoa tukeviin menetelmiin. Pienissä yrityksissä puolestaan samalla henkilöllä voi olla vastuullaan useita eri tehtäväkokonaisuuksia, jolloin työaika ei riitä uusien menetelmien selvittämiseen.

Lähes kaikilla teollisuusrobottimerkeillä on versionhallinnan käyttäjiä. Tutkimusaineiston perusteella hajautettua versionhallintaa käytetään ABB-, Fanuc- ja KUKA-teollisuusrobottien versionhallinnassa. Keskitetty versionhallinta on käytössä Kawasaki-teollisuusroboteille. Eri robottimerkit voivat hyödyntää eri versionhallintajärjestelmiä, mutta keskitetystä versionhallinnasta voi olla enemmän hyötyä, jos lähdekoodi joudutaan tallentamaan binäärisessä muodossa eli tiedostoina, joita ei voida lukea tekstinä. Hajautettua versionhallintaa käytettäessä binääritiedostot vievät suhteessa enemmän tilaa, koska tiedostot joudutaan säilyttämään sellaisena kuin ne ovat, useassa paikassa.

Koska versionhallinta käsitteenä on tuttu vähintäänkin osittain kaikille, syynä versionhallinnan käyttämättömyydelle ei ole se, ettei siitä ole mitään tietoa. Mikään yritys, joka ei käytä versionhallintaa, ei pidä versionhallintaa hyödyttömänä. Versionhallinnan potentiaali tiedostetaan, mutta kaikki eivät ole sitä käyttöönottaneet. Eräs vastaajista kertoi seuraavaa:

> *”Vaikka integraatioprojektit olisivatkin tyypillisesti lyhyehköjä ja normaalisti yhden robottiohjelmoijan projekteja, järjestelmän elinkaari varsinaisen projektin jälkeen on aina pitkä. Sen aikana tehdään tyypillisesti muutoksia myös robottiohjelmiin ja muutoskohdat on syytä kommentoida aina hyvin. Lisäksi muutoksia tehdään useamman ihmisen toimesta. Versionhallinta pitäisi automaattisesti kaikki kärryillä tilanteesta.”*

Tutkimusaineistossa omia robotteja valmistavat yritykset (n=2) eivät käytä versionhallintajärjestelmää robotin työohjelmille. Versionhallintaa kuitenkin käytetään muussa ohjelmointityössä ainakin yhden robottivalmistajan toimesta.

Kaikki versioivat ohjelmat tavalla tai toisella. Usein ohjelmatiedostot saavat versiotunnuksen tai muun tunnisteen tiedoston nimeen. Osa dokumentoi versionumeron ja tehdyt muutokset kommenttina ohjelmatiedostoon.

Työtapojen koetaan suurimmaksi osaksi toimivan ilman versionhallintaa. Kuitenkin kukaan vastaajista ei ole täysin samaa mieltä siitä, että työtavat toimisivat hyvin ilman versionhallintaa. Tästä voi päätellä, että versionhallinta koetaan enemmän työtapoja parantavana tai hyödyntävänä asiana kuin heikentävänä. Missään yrityksessä ei myöskään olla täysin sitä mieltä, ettei versionhallinalle olisi tarvetta.

Versionhallinnan käyttöönoton ei koeta vievän liikaa aikaa tai sen käyttöönoton olevan liian vaikea. Käyttöönottoa ei siis koeta esteeksi versionhallinnan käytölle. Viisiportaisella Likertin asteikolla vastausten keskiarvo puoltaa tätä. Versionhallintaa ei pidetä myöskään liian vaikeana oppia ja käyttää. Kysymys siitä, onko kokemusta tai tietoa tarpeeksi versionhallintajärjestelmistä, jakaa mielipiteitä. Osa on täysin eri mieltä ja osa täysin samaa mieltä.

Tärkeimmät syyt, miksi versionhallintaa ei käytetä

Versionhallintaa ei ole koettu tarpeelliseksi työohjelmille, mutta muussa ohjelmoinnissa versionhallintaa käytetään. Versionhallintaa ei käytetä, koska versioidulla tiedolla ei kyseisessä tapauksessa olisi hyötyä. Työohjelmat sisältävät mahdollisesti vain robottipisteistä muodostuvia liikeratoja. Muu robotin hyödyntämä koodi voi kuitenkin olla versionhallinnassa.

> *”Vielä ei ole ollut tarpeen. Ohjelmat ovat usein myös erittäin asiakas ja työkappalekohtaisia, joten sitä kautta pohjia ja eri rivisoita on hankala hyödyntää toisissa projekteissa. Yleensä ohjelmat hiotaan kuntoon lyhyen ajan sisällä ja hyväksyt ohjelmat tallennetaan järjestelmään.”*

> *”Käytämme ohjelmointi työssä päivittäin versionhallintaa. Itse ajo-ohjelmien versiointi ei ole ollut vielä tarpeellista.”*

Yksi keskeinen syy versionhallinnan käyttämättömyydelle on, ettei versionhallinta integroidu ohjelmointiympäristöön. Tutkituista robottimerkeistä Omron oli ainoa, joka on integroinut Git-versionhallinnan teollisuusrobottiensa ohjelmointiympäristöön. Versionhallinnan hyödyntäminen on mahdollista myös ilman integraatiota, mutta integraatio ketteröittäisi ohjelmointityötä.

Tarve monipuoliselle integraatiolle tulee esille tutkimusaineistosta. Vaikka rajapinta versionhallintaan olisi olemassa ohjelmointiympäristössä, mukaan kaivattaisiin monipuolisempia ominaisuuksia. Teollisuusrobotti muodostaa liikeratoja ohjelmoitujen pisteiden välissä usean eri parametrin funktiona. Liikerataan vaikuttaa mm. liikekäskyn tyyppi. Liikeratoja on vaikea tai mahdoton hahmottaa pelkästä ohjelmakoodista. Mikäli ohjelmointiympäristö kykenisi visualisoimaan eri ohjelmaversioiden väliset liikeratojen erot, se voisi tuoda toivotun ominaisuuden versionhallinnan hyödyntämiseksi. Eräät vastaajista kertoivat seuraavaa:

> *”Tämän hetken versionhallinta työkalut eivät integroidu meidän OLP ympäristöön.”*

> *”Pelkkä versionhallinta ei riitä. Pitäisi integroitua ohjelmointi ympäristöön. Kukaan ei pelkkää koodia katsomalla jaksa tai pysty havaitsemaan muutoksia robotinliikeradassa.”*

Yksi syy versionhallinnan käyttämättömyydelle on, ettei aikaa asiaan perehtymiseen ole ollut riittävästi. Ajan puute voi tulla eteen etenkin pienissä yrityksissä, joissa yhdellä henkilöllä voi olla useita tehtäviä. Versionhallinta vaatii perehtymistä, jotta siitä tulee hyödyllinen apuväline eikä työtä hidastava pakollinen suorite. Järjestelmän käytön lisäksi etenkin työtavat on syytä selvittää ja suunnitella omaa työtä tukeviksi. Koulutus tai perehdytys aiheeseen helpottaa versionhallinnan käyttöönottamista. Juuri itselle oleellisen tiedon löytäminen voi olla aikaa vievää ja siksi jäädä tekemättä:

> *”Ajan puute. Tiedostan, että tuosta olisi todennäköisesti hyötyä, mutta en ole ehtinyt perehtyä asiaan riittävästi.”*

Johtopäätökset versionhallinnan käyttämättömyydestä

Versionhallinnan käyttämättömyyttä voidaan selittää muutaman keskeisen seikan avulla. Robottivalmistajien kohdalla versionhallinnan käyttämättömyys selittyy sillä, ettei työohjelmien versionhallinnalla koeta olevan hyötyä. Muun ohjelman kohdalla versionhallintaa kuitenkin hyödynnetään. Kaikilla ei ole ollut riittävästi aikaa versionhallintaan perehtymiseen. Versionhallintajärjestelmät eivät integroidu riittävällä tasolla ohjelmointiympäristöihin.

Yrityksen koon vaikutusta versionhallinnan käyttöön voi selittää sillä, että mahdollisesti suuremmissa yrityksissä työkuorma on jakautunut tasaisemmin kuin pienissä yrityksissä, ja mahdollisuus tutkia ja käyttöönottaa uusia työtä edistäviä toimintatapoja tulee paremmin mahdolliseksi. Pienissä

yrityksissä ohjelman tekijöillä, voi olla ohjelmoinnin lisäksi useita eri tehtäviä. Heillä aika ei välttämättä riitä uusien toimintamallien tutkimiseen, kuten versionhallinnan käytön selvittämiseen. Tästä on selvä viittaus tutkimusaineistossa. Voi olla myös, että suurissa yrityksissä on laajempi asiantuntemus ja todennäköisemmin kokemuksia myös versionhallinnasta tai sen tuomista eduista.

Lean- ja Agile-osaaminen korreloi versionhallinnan käyttöasteen kanssa positiivisesti. Lean-periaatteiden mukaisesti ketteryyttä lisäämällä voidaan vaikuttaa vaihteluun ylikuormituksesta ja hukasta. Oikein käytettynä versionhallinnalla voidaan pienentää hukkaan menevää aikaa, kun ohjelmaan tehtyjä muutoksia ei tarvitse selvitellä vaan ne löytyvät dokumentoituina versionhallinnasta. Samalla versionhallinta voi myös pienentää ylikuormitusta helpottamalla usean ohjelmoijan yhteistyötä ja helpottamalla tiedon löytämistä siitä, mitä ohjelmassa on muuttunut, miksi muutos on tehty, milloin se on tehty ja kuka sen on tehnyt. On mahdollista, että Lean-periaatteiden ymmärtäminen on tuottanut ajattelua, joka on ollut osaltaan vaikuttamassa versionhallinnan käyttöön yrityksissä. Mikäli Lean-periaatteita ei tunne, ei tällaista ajattelua välttämättä synny. Lean-periaatteiden vähäinen tuntemus voi siis osaltaan selittää versionhallinnan käyttämättömyyttä.

Koska lähes kaikki versionhallintaa käyttämättömät kokevat versionhallinnan hyödylliseksi eivätkä koe käyttöönottoa liian vaikeaksi tai aikaa vieväksi, jää jäljelle riittävän tiedon ja taidon hankkiminen versionhallinnan käyttöönottamiseksi ja käyttämiseksi. Tietoa hajautetusta versionhallinnasta on koottu tähän raporttiin, ja taidot karttuvat versionhallintaa käyttämällä.

5.2.2 Versionhallinnan käyttöön vaikuttavat tekijät

Tutkitaan tarkemmin, millaisissa yrityksissä hyödynnetään versionhallintaa teollisuusrobotiikan ohjelmoinnissa ja millaisena käyttöönotto ja käyttö koetaan. Perehdytään tärkeimpiin syihin, jotka vaikuttavat versionhallinnan käyttöön.

Luvussa 5.2.1 todettiin, ettei robottimerkillä ole merkitystä versionhallinnan käyttöön ja että suurimmissa yrityksissä versionhallinnan hyödyntäminen on yleisempää kuin pienissä. Lean- ja Agile-osaamisella on positiivinen korrelaatio versionhallinnan käyttöön.

Lähes kaikki (n=5) versionhallintaa käyttävistä vastaajista ovat täysin samaa mieltä siitä, että työtavat toimivat paremmin, kun käytössä on versionhallinta. Yksi vastaajista on jokseenkin eri mieltä

siitä. Luvun 2.1 kuvassa 7 on esitelty hajautetun versionhallinnan etuja ja ne pätevät suurelta osin myös keskitetyn versionhallinnan kanssa. Se ettei versionhallinnan käyttö ole parantanut työtapoja voi olla selittävänä tekijänä, ettei versionhallinnan työnkulku ole selvä kaikille tai yhteisiä toimintatapoja ei noudateta.

Versionhallinnan käyttöönotto jakaa mielipiteitä. Puolet mieltävät versionhallinnan käyttöönoton jokseenkin helpoksi ja nopeaksi. Yksi vastaajista on täysin eri mieltä versionhallinnan helposta ja nopeasta käyttöönotosta. Tutkittaessa versionhallintajärjestelmän vaikutusta käyttöönoton haasteisiin havaitaan, että Gitin käyttöönotto on koettu haasteelliseksi, mutta Bitbucketin käyttöönotto ei. Bitbucket on versionhallintapalvelu, joka tukee Git-versionhallintajärjestelmää. SVN:n käyttöönotto on koettu jokseenkin haasteelliseksi.

Valtaosa (n=5) versionhallintaa käyttävistä vastaajista ovat jokseenkin tai täysin saamaa mieltä siitä, että versionhallinnan käyttö on helppoa, ja että heidän robottiohjelmansa soveltuvat hyvin versionhallintaan. Yksi vastaajista on jokseenkin eri mieltä asiasta. Fanuc-robotin käyttäjissä, jotka käyttävät versionhallintaa (n=3), on yksi käyttäjä, joka kokee versionhallinnan jokseenkin hankalaksi.

Tärkeimmät syyt, miksi versionhallintaa käytetään

Kaikki versionhallintaa käyttävistä pitävät tärkeänä tai erittäin tärkeänä syynä versionhallinnan käytölle sitä, että yrityksessä on useita ohjelmoijia samalle koodille. Versionhallinnan yksi eduista on ketterän yhteistyön mahdollistaminen. Versionhallintajärjestelmät osaavat varoittaa ristiriidoista, joissa samaa koodiosaa on muutettu usean eri ohjelmoijan toimesta. Tilanne voidaan ratkaista valitsemalla, kumman tekijän osuus tulee voimaan. Käytänteet määrittävät kuka päätöksen lopulta tekee.

Suurin osa (n=4) pitää ohjelmakoodin historian seurantaa tärkeänä tai erittäin tärkeänä. Ohjelmakoodin historialla saadaan dokumentointia tehdyistä muutoksista. Jokainen muutos sisältää tiedot kuka teki, mitä teki ja miksi teki sekä milloin teki. Näistä tiedoista voi olla hyötyä monessa eri vaiheessa ohjelman tekoa. Myös vanhoihin projekteihin on helppo palata ja tarkistaa esimerkiksi, miten tietty ongelma on korjattu ohjelmassa.

Ohjelmakoodin hajauttaminen koetaan hyödylliseksi ominaisuudeksi, joskin ei aivan yksipuolisesti. Mediaanilla mitattuna ohjelmakoodin hajauttaminen on annetuista vaihtoehdoista kolmanneksi tärkein syy versionhallinnan käytölle. Hajauttamisella voidaan suojata ohjelmakoodia mahdollisilta laiterikkojen aiheuttamalta ohjelmakoodin menettämiseltä. Jokainen kyselyyn vastannut tallentaa teollisuusrobottien ohjelmasta varmuuskopion. Tällä tavoin suojataan myös ohjelmakoodia varsinkin silloin, kun tallennus tapahtuu fyysisesti erilliseen paikkaan varsinaisesta ohjelman lähdekoodista.

Vähiten tärkein syy versionhallinnan käytölle on mahdollisuus palata helposti eri kehitysvaiheisiin. Se jakaa mielipiteitä eniten ja Likertin 5-portaisella asteikolla (arvot 1–5) keskiarvo tämän ominaisuuden merkittävyydelle on 2,8. Jos versionhallintaa käyttää aktiivisesti ja tekee paljon committeja, eri kehitysvaiheisiin paluulla on enemmän hyötyä. Ominaisuuden suosioon voi vaikuttaa se, ettei versionhallintajärjestelmät vielä integroidu kaikkiin ohjelmointiympäristöihin. Integraation puutos vaikuttaa siihen, kuinka helposti edelliseen versioon voidaan palata. Mikäli ohjelman eri versioihin vaihtaminen sujuisi helposti suoraan ohjelmointiympäristössä, sillä saattaisi olla positiivinen vaikutus kyseessä olevan versionhallinnan ominaisuuden suosiolle.

Versionhallinnan käytön haasteet

Yhtä lukuun ottamatta kaikki versionhallintaa käyttävät ovat täysin samaa mieltä siitä, että työtavat toimivat paremmin, kun käytössä on versionhallinta. Versionhallinnan käyttöön voi liittyä myös haasteita. Yksi vastaajista on täysin eri mieltä siitä, että versiohallinnan käyttöönotto on helppoa tai nopeaa. Toisin sanoen versionhallinnan käyttöönottoon voi kulua odotettua enemmän aikaa ja sen käyttöönotossa voi kohdata haasteita. Robottimerkeittäin luokitellusta aineistosta käy ilmi, että haasteita on kohdattu, kun käytössä on Fanuc- tai KUKA-teollisuusrobotit versionhallintatavan ollessa hajautettu ja järjestelmän Git. Aineistosta havaitaan, että KUKA-teollisuusrobottien käyttäjistä kolme viidestä on sitä mieltä, että versionhallinta on jokseenkin nopea ja helppo käyttöönottaa. Fanuc teollisuusrobottien osalta kaksi kolmesta on samaa mieltä edellisen maininnan kanssa tai ei osaa muodostaa kantaa ottavaa mielipidettä. Tästä voidaan päätellä, että todennäköisesti käyttöönoton haasteet liittyvät versionhallintajärjestelmään.

Versionhallinnan käytön sujuvuuteen ja tehokkuuteen vaikuttavat työtavat. Työtapoihin vaikuttavat millaista työnkulkua käytetään. Ellei työnkulkua ole sovittu, tai toimintatavoissa on eroavaisuuksia eri tekijöiden kesken, voi siitä aiheutua haasteita versionhallinnan käytössä. Luvussa 2.5 on selitetty hajautetun versionhallinnan eri työnkulkuja eli tapoja hyödyntää hajautettua versionhallintaa.

Yhteiset toimintatavat, käytännöissä pysyminen ja kulttuurin ylläpito mainitaan haasteina, joita on kohdattu versionhallinnan käytössä. Selittäviä tekijöitä näille haasteille voi olla, ettei työnkulku ole parhaalla mahdollisella tavalla sopiva, tai työnkulusta sekä sen noudattamisesta ei ole sovittu tiimin kesken. Erityisen tärkeää on, että kaikki noudattaisivat versionhallinnasta sovittuja käytäntöjä. Muutoin on mahdollista, että jotkut ohjelmoijat kokevat turhautumista, jos toiset eivät toteuta täsmällisesti esimerkiksi versioiden kommentointia, tai lisää versioita riittävän usein järjestelmään. Sellaisista commiteista, kuten ”Some updates”, jossa on lisäksi paljon muutoksia, voi olla vaivalloista selvittää mitä ja miksi on oikeastaan tehty.

Vaikka versionhallinnan käyttö helpottaa yhteistyötä saman ohjelmakoodin parissa, se ei poista tarvetta ohjelmoijien väliselle kommunikoinnille. Mikäli usein kohdataan tilanne, että samaa ohjelmakoodin osaa on muutettu usean ohjelmoijan toimesta ja päädytään konfliktiin eri versioiden välillä, työnteko voi tuntua vaivanloiselta. Tällaiset tilanteet voidaan välttää, tai niitä voidaan minimoida kommunikoimalla tiimin kesken ja sopimalla kuka tekee muutoksia mihinkin ohjelmakoodin osuuteen. Aina ei voida välttää saman koodiosuuden muutostarvetta usean ohjelmoijan toimesta varsinkin, jos ohjelma koostuu suurista yksittäisistä kokonaisuuksista. Silloin koodin pilkkominen pienempiin osakokonaisuuksiin voi auttaa konfliktien välttämisessä. Vastauksissa on seuraavanlaisia kommentteja, mikä on koettu haasteelliseksi versionhallinnassa.

> *”Yhteiset toimintatavat”*
>
> *”Käytännöissä pysyminen ja kulttuurin ylläpito*”
>
> *”Olemme käyttäneet PC-ohjelmistojen puolella versionhallintaa jo pidempään, ja ilman suurempia ongelmia. Jostain syystä robotti- ja PLC-ohjelmoijien parissa ei kuitenkaan tunnu olevan riittävää motivaatiota asian edistämiseksi. Jopa PC-ohjelmoija, joka tekee välillä robottiohjelmaa, saattaa oudosti laiskistua robottiohjelman kohdalla. Mielipiteeni on, että kyse on oikean motivoinnin puutteesta.”*

Työkalujen soveltuvuus tiettyyn tehtävään vaikuttaa työn sujuvuuteen. Esimerkiksi numeerista dataa on helpompi käsitellä taulukkolaskentaohjelmalla kuin tekstinkäsittelyohjelmalla. Mitä soveltuvammat ohjelmointiympäristöt ovat versionhallinnan hyödyntämiseen, sitä helpompaa versionhallinnan käytöstä tulee. Todennäköisesti teollisuusrobottien ohjelmointiympäristöt tukevat tulevaisuu-

dessa paremmin versionhallinnan hyödyntämistä. Omronin teollisuusrobottien ohjelmointiympäristön osalta se jo toteutuu. ABB:n osalta on saatu viitteitä siitä, että ohjelmointiympäristöön ollaan lisäämässä versionhallintaominaisuus. Toimintatavat helpottuvat, jos versionhallintaa voidaan toteuttaa suoraan teollisuusrobottien ohjelmointiympäristöstä.

Johtopäätökset miksi versionhallintaa hyödynnetään

Versionhallinnan käyttö selittynee enimmäkseen sillä, että yrityksessä on ollut riittävästi tietoa ja resursseja selvittää versionhallinnan hyödyntämistä ja toteuttaa versionhallinnan käyttöönotto. Yrityksessä on tiedostettu, että versionhallinnan käyttö vie hieman enemmän aikaa kuin ilman sitä, mutta se tuottaa pitkällä aikavälillä enemmän hyötyä kuin sen käyttämättömyys. Mahdollisesti yrityksen strategiselle johdolle on saatu tuotua esille versionhallinnan edut ja vaikutukset ohjelmointityöhön, ja lopulta saatu aikaan päätös toimenpiteiden toteuttamisesta versionhallinnan käyttöönottamiseksi. Päätös versionhallinnan käyttöönottamiseksi voidaan sanoa olleen hyödyllinen, koska lähes kaikki versionhallintaa käyttävät ovat täysin samaa mieltä siitä, että työtavat ovat parantuneet versionhallinnan käytön myötä.

Muita selkeitä versionhallinnan käyttöä selittäviä tekijöitä ei analysoidusta tutkimusaineistosta nouse esille. Käytössä oleva robottimerkki ei vaikuta versionhallinnan käyttöön. Lean- ja Agile-menetelmät korreloivat positiivisesti versionhallinnan käytön kanssa. Niillä saattaa olla vaikutusta ajatteluun, joka on ollut edesauttamassa versionhallinnan käyttöön päätymisessä.

5.3 Versionhallinnan tarpeellisuus ja hyödyt teollisuusrobotiikassa

Tuloksien perusteella voidaan sanoa työtapojen toimivan paremmin teollisuusrobottien ohjelmoinnissa, kun käytössä on versionhallinta. Tärkeimpänä syynä versionhallinnan käytölle on se, että yrityksessä on useita ohjelmoijia samalle ohjelmakoodille. Versionhallintajärjestelmän tehokas käyttö tuo etuja yhteistyön toteuttamiseen. Hajautetun tai keskitetyn versionhallinnan käyttäjät hyötyvät ohjelmakoodin hajautuksesta, joka suojaa ohjelmakoodia osaltaan tiedon häviämiseltä. Versionhallinta helpottaa muistamaan mitä muutoksia on tehty viimeksi ja täten helpottamaan työn jatkamista esimerkiksi loman jälkeen. Tätä pidettiin toiseksi tärkeimpänä syynä versionhallinnan käytölle.

Voidaan todeta, että versionhallinnan käyttö on selvästi mahdollista teollisuusrobotiikassa ja robottimerkki itsenäisenä tekijänä ei aseta rajoitteita versionhallinnan käytölle. Yhteistyörobottien osalta versionhallinnan hyödyntäminen on rajoittuneempaa, johtuen ohjelmointiympäristöstä. Rajoittavia tekijöitä ovat ohjelmakoodin vienti ja tuonti ohjelmointiympäristöön ja muutosten yhdistäminen, mitkä voi olla haastavaa tai mahdotonta graafisesti luotujen ohjelmien kanssa. Muita selkeitä rajoittavia tekijöitä ei tutkimuksessa tullut esille. Mikäli robotista tai robotin ohjelmointiympäristöstä saadaan lähdekoodi vietyä ja tuotua se versionhallintaan niin se parantaa versionhallinnan hyödynnettävyyttä. Silloin muutoksia voidaan seurata tarkemmin. Mikäli lähdekoodia ei saada vietyä, niin ohjelma voidaan kuitenkin tallentaa binäärisenä versionhallintaan. Silloin muutokset kannattaa kuvata tarkasti kommenttiin, koska aiempaa koodia ei voida vertailla uuteen versioon.

6 YHTEENVETO

Tutkimuksessa selvitettiin, onko versionhallinta käytössä teollisuusrobottiohjelmoijien tai -loppukäyttäjien työkaluna, ja millaisia kokemuksia tai ennakkoasenteita siihen liittyi. Tutkimuksessa selvitettiin hajautetun versionhallinnan tarpeellisuutta ja hyötyjä teollisuusrobotiikassa sekä edellytyksiä sen käyttöön.

Keskeisimmät tulokset

Tutkimuksen taustaolettamus oli, ettei versionhallinta ole teollisuusrobotiikassa kovin yleinen ilmiö. Siihen viittasi tekijän kokemuksen lisäksi ainakin se, ettei aiheesta löytynyt aikaisempia tutkimuksia, opinnäytetöitä tai artikkeleita. Versionhallinnasta yleisesti löytyy valtavasti julkaisuja. Taustaolettamuksesta tuli tutkimuksen työhypoteesi, jota tutkimuksella haluttiin koetella. Se osoittautui virheelliseksi, koska jopa puolet vastaajista hyödynsivät versionhallintaa teollisuusrobottikassa. Se on kuitenkin verrattain vähän, mikäli teollisuusrobotiikan versiohallinnan käyttäjämäärää verrataan muuhun ohjelmistokehitykseen, jossa sitä käytetään lähes poikkeuksetta. Versionhallinnasta ja sen eduista ollaan kuitenkin tietoisia, joten versionhallinta tullee yleistymään myös teollisuusrobotiikan ohjelmoinnissa. Siihen viittaavat tutkimuksessa esille tuodut ohjelmointiympäristöjen kehityssuunnat ja jo olemassa oleva ohjelmointiympäristöjen versionhallintaintegraatio.

Versionhallintaa hyödynnettiin suhteellisesti enemmän suurissa ja keskisuurissa yrityksissä. Koska versionhallinta käsitteenä oli tuttu vähintäänkin osittain kaikille, syynä versionhallinnan käyttämättömyydelle ei ollut se, ettei siitä ollut mitään tietoa. Mikään yritys, joka ei käyttänyt versionhallintaa, ei pitänyt versionhallintaa hyödyttömänä. Versionhallinnan potentiaali siis tiedostettiin, mutta kaikki yritykset eivät ole sitä käyttöönottaneet.

Lähes kaikilla teollisuusrobottimerkeillä oli versionhallinnan käyttäjiä. Lähtökohtaisesti versionhallinta onkin mahdollista toteuttaa kaikilla teollisuusrobottimerkeillä. Toisilla ketterämmin kuin toisilla. Lisäksi yhteistyöroboteilla versionhallinnan hyödyntäminen ei ole yhtä tehokasta kuin perinteisillä teollisuusroboteilla, johtuen yhteistyörobottien erilaisesta ohjelmointiympäristöstä. Versionhallinta on ketterämpää niillä teollisuusrobottien ohjelmointiympäristöillä, joista lähdekoodi saadaan vietyä ja tuotua lukukelpoisessa muodossa tiedostoihin ja jotka ovat integroineet versionhallinnan ohjelmointiympäristöönsä.

Työtapojen koettiin suurimmaksi osaksi toimivan ilman versionhallintaa, mutta kukaan versionhallintaa käyttämättömistä vastaajista ei väittänyt, etteikö versionhallinnasta voisi olla hyötyä. Versionhallinnan käyttöönoton ei koettu vievän liikaa aikaa tai sen käyttöönoton olevan liian vaikeaa. Käyttöönottoa ei siis koettu esteeksi versionhallinnan käytölle. Versionhallintaa käytettäessä yhteiset toimintatavat, käytännöissä pysyminen ja kulttuurin ylläpito olivat haasteita, joita oli kohdattu versionhallinnan käytössä. Selittäviä tekijöitä näille haasteille voi olla, ettei työnkulku ole parhaalla mahdollisella tavalla sopiva tai siitä sekä sen noudattamisesta ei ole sovittu tiimin kesken.

Yksi keskeinen syy versionhallinnan käyttämättömyydelle oli, ettei versionhallinta integroidu käytössä olevaan ohjelmointiympäristöön. Monipuoliselle integraatiolle ohjelmointiympäristön ja versionhallinnan välillä on tarvetta. Jos ohjelmaversioiden vertailu voitaisiin toteuttaa niin, että robotin liikeratojen erot olisivat visuaalisesti nähtävissä virtuaaliympäristössä, saavutettaisiin sillä uutta tehokkuutta teollisuusrobottien ohjelmointiin. Monipuolisen integraation toteuttamisen ei pitäisi olla mahdotonta. Simulointiympäristössä voitaisiin suorittaa kahta virtuaalikontrolleria, joissa toisessa olisi sen hetkinen ohjelmaversio ja toisessa versionhallintajärjestelmässä oleva versio. Eri versioiden robotit voisivat olla päällekkäin ja erot liikeradoissa visualisoitaisiin työympäristössä.

Versionhallinnan käyttämättömyyttä selittää se, ettei aikaa asiaan perehtymiseen ole ollut riittävästi. Ajan puute voi tulla vastaan etenkin pienissä yrityksissä. Lähes kaikki versionhallintaa käyttämättömät kokivat sen hyödylliseksi, eikä käyttöönoton koettu olevan liian vaikeaa tai aikaa vievää. Syyksi versionhallinnan käyttämättömyydelle jää vähäiset tiedot ja taidot. Tietoa voi kartuttaa tämän tutkimusraportin viitekehyksestä sekä lähteistä, ja taidot karttuvat versionhallintaa käyttämällä.

Versionhallinnan käyttö selittyy enimmäkseen sillä, että yrityksessä on ollut riittävästi tietoa ja resursseja selvittää versionhallinnan hyödyntämistä ja toteuttaa versionhallinnan käyttöönotto. Mitä soveltuvammat ohjelmointiympäristöt ovat versionhallinnan hyödyntämiseen, sitä helpompaa versionhallinnan käytöstä tulee. Todennäköisesti teollisuusrobottien ohjelmointiympäristöt tulevaisuudessa tukevat paremmin versionhallinnan hyödyntämistä teollisuusrobottien ohjelmoinnissa.

Kyselylomakkeessa ei eroteltu yhteistyörobotteja muista teollisuusroboteista, joten tutkimusaineisto ei sisällä informaatiota yhteistyörobottien versionhallinnasta. Yhteistyöroboteilla ohjelmointi on usein graafista ja aina ohjelmakoodia ei saada sellaiseen tekstimuotoon, jotta sen vienti versi-

onhallintaan ja versioiden yhdistäminen olisi ketterää. Koneohjaimien käyttöliittymien graafisen sisällön versionhallinta on jo mahdollista, mutta yhteistyörobottien osalta siinä on havaittavissa puute. Teknologia on siis sinänsä olemassa versionhallinnan hyödyntämiseksi graafisessa ohjelmoinnissa, joten lienee vain ajan kysymys, koska versionhallinta on ketterästi hyödynnettävissä myös yhteistyöroboteilla.

Tuloksien perusteella voidaan sanoa työtapojen toimivan paremmin teollisuusrobottien ohjelmoinnissa, kun käytössä on versionhallinta. Voidaan todeta, että versionhallinnan käyttö on selvästi mahdollista teollisuusrobotiikassa ja ettei robottimerkki aseta rajoitteita versionhallinnan käytölle, lukuun ottamatta yhteistyörobotteja. Tutkimuksessa ei tutkittu yhteistyörobottien versionhallintaa, mutta viitekehyksen aineiston perusteella voidaan todeta, ettei yhteistyörobottien ohjelmointiympäristö mahdollista samanlaista lähdekoodin vientiä ja tuontia kuin muiden teollisuusrobottien ohjelmointiympäristö. Muita selkeitä rajoittavia tekijöitä ei tutkimuksessa tullut esille. Versionhallinnan käyttöä selittää parhaiten sen tuoma hyöty työtapoihin.

Menetelmien etuja ja heikkouksia

Tutkimusmenetelmänä laadullinen tutkimusmenetelmä soveltui parhaiten tiedonintressiin, joka oli saavuttaa ymmärrystä ilmiön merkityksestä ja ennen kaikkea synnyttää uutta ajattelua. Tavoite oli selvittää välittömän havainnoinnin ulottumattomissa olevaa ilmiötä versionhallinnan käytöstä. Laadullisen tutkimusmenetelmän erityispiirteenä on juuri sama ominaisuus, joten se soveltui käyttötarkoitukseen hyvin.

Heuristisen työotteen tarkoituksena oli saavuttaa riittävä tarkkuus tutkimustehtävän toteuttamiseksi ja merkityksellisimmän tiedon selvittämiseksi. Metodi soveltui myös hyvin ajankäytöllisesti tutkimukselle töiden ja muiden toimien ohessa. Hermeneutiikka hermeneuttisen kehän muodossa osoittautui hyväksi metodiksi aineiston analysointiin. Vuoropuhelu aineiston kanssa tuotti uusia oivalluksia eri kierroksilla.

Tutkimusaineisto kerättiin web-pohjaisella kyselylomakkeella Webropol alustalla. Kyselylomake on kustannustehokas, nopea ja suhteellisen vaivaton tapa hankkia tutkimusaineisto. Koska tutkimus tehtiin muiden toimien ohessa, menetelmät valittiin tutkimusongelman ratkaisemiseksi olosuhteet huomioiden.

Tutkimusaineiston kerääminen kyselylomakkeella jättää pois mahdollisuuden tulkita haastateltavan emotionaalista ilmaisua. Toisaalta sen pois jättäminen vähentää virhetulkinnan riskiä. Lisäksi tarkentavat lisäkysymykset rajautuvat pois kyselylomaketta käytettäessä. Lomakkeessa pystyi valitsemaan useamman robottimerkin, joita yritys käyttää. Monivalinnan myötä ei saada varmuutta, käyttääkö versionhallintaa hyödyntävä yritys sitä kaikille robottimerkeillensä. Kyselyssä olisi ollut hyvä kysyä, millaista työnkulkua tai työtapoja yrityksessä noudatetaan, tai onko yrityksessä ylipäätään sovittua työnkulkua tai toimintamallia versionhallintajärjestelmän käyttämiseksi. Tämän tiedon myötä olisi voitu tutkia, millä tavalla tietty työtapa vaikuttaa versionhallintajärjestelmän käyttöön.

Yleistettävyys

Tutkimusaineistossa eniten käytetyimmät robottimerkit vastasivat yleisimpiä robottimerkkejä, jolloin se auttaa osaltaan tulosten yleistettävyydessä. Kyselyn vastaajaksi valikoituneilla yrityksien edustajilla on pitkä työkokemus alalta. Kahdestatoista vastaajasta 11:llä oli työkokemusta enemmän kuin 5 vuotta, jolloin kokemusta ja ensikäden tietoa tutkimuksen aihepiiristä oli riittävästi tutkimusaineistossa. Yleistettävyyttä heikentää se, ettei vertailukohtaa muihin tutkimustuloksiin ole. Teollisuusrobotiikan versionhallinnasta ei myöskään löytynyt artikkeleita tai muuta vastaavaa aineistoa.

Ne yritykset, jotka eivät hyödynnä vielä versionhallintaa, voivat saada tästä tutkimuksesta uutta tietoa versionhallinnan käyttöönottamiseksi. Niille, jotka käyttävät jo versionhallintaa, voi muodostua uutta ajattelua siihen, kuinka versionhallintaa voisi hyödyntää vielä tehokkaammin. Tutkimuksen johtopäätöksien perusteella voidaan osoittaa, että ennakkokäsityksistä siitä, ettei versionhallintaa voi tai ole hyödyllistä käyttää teollisuusrobotiikassa, kannattaa luopua.

Integraattoreille suunnattujen robottien valmistajille tutkimuksessa on osoitus siitä, että keskitetty tai hajautettu versionhallinta on robottiohjelmoijien käytössä, ja että versionhallintarajapinnalle on tarvetta.

Jatkotutkimus

Robotiikan asiantuntijatehtäviin tähtäävissä koulutuksissa ei yleisesti esiinny versionhallintaan liittyviä opintoja. Robotiikan asiantuntijatehtävissä olisi hyötyä, jos alaa opiskelevat pystyisivät hankkimaan osaamista versionhallinnasta. Versionhallinta vaikuttaa tämän tutkimuksen tiedon valossa olevan melko vahvasti osana teollisuusrobottien ohjelmointia, vakkakin ero käyttöasteessa suhteessa muuhun ohjelmistokehitykseen on suuri.

Jatkotutkimusta aihepiiristä on mahdollista tehdä esimerkiksi DevOps-menetelmien ja -työkalujen hyödyntämisestä teollisuusrobotiikassa, sekä niiden vaikutuksista robotisointiprojekteihin. AMK-opinnäytetyönä versionhallinnan käytännön hyödyntäminen teollisuusrobotiikassa tai cobotiikassa (collaborative robotics) tuottaisi tärkeää tietoa niille, jotka suunnittelevat versionhallintajärjestelmän käyttöönottamista teollisuusroboteille.

LÄHTEET

1. Majumdar, Rana, Jain, Rachna, Barthwal, Shivam & Choudhary, Chetna 2017. Source code management using version control system. 6th International Conference on Reliability, Infocom Technologies and Optimization (Trends and Future Directions) (ICRITO). IEEE Personal Account. Hakupäivä 19.2.2022. https://ieeexplore.ieee.org/document/8342438. Vaatii käyttöoikeuden.

2. Chacon, Scott & Straub, Ben 2014. Pro Git. Second Edition. Apress. Hakupäivä 19.2.2022. https://github.com/progit/progit2/releases/download/2.1.337/progit.pdf.

3. Rochkind, Marc J. 1975. The Source Code Control System. IEEE Transactions on Software Engineering (Volume: SE-1, Issue: 4, December 1975). IEEE Personal Account. Hakupäivä 3.3.2022. https://ieeexplore.ieee.org/document/6312866. Vaatii käyttöoikeuden.

4. Tichy, Walter F. 1985. RCS – A System for Version Control. West Lafayette, Indiana: Purdue University. Hakupäivä 3.4.2022. https://www.gnu.org/software/rcs/tichy-paper.pdf.

5. Foger, Karl & Bar, Moshe. 2003. Open Source Development with CVS. 3rd Edition. Scottsdale, Arizona: Paraglyph Press, Inc. Hakupäivä 3.3.2022. http://cvsbook.red-bean.com/OSDevWithCVS_3E.pdf.

6. Tichy, Walter F. 1982. Design, Implementation, and Evaluation of a Revision Control System. West Lafayette, Indiana: Purdue University. Hakupäivä 3.3.2022. https://dl.acm.org/doi/10.5555/800254.807748.

7. LWN.net 2004. subversion 1.0 is released. Hakupäivä 4.3.2022. https://lwn.net/Articles/72498/.

8. O'Sullivan, Bryan 2009. Mercurial: The Definitive Guide. O'Reilly Media.

9. Collins-Sussman, Ben, Fitzpatrick, Brian W. & Pilato, C. Michael 2008. Version Control with Subversion. 2nd Edition. O'Reilly Media.

10. Atlassian 2022. Atlassian Acquires Bitbucket.org, Distributed Version Control System Hosting. Hakupäivä 4.4.2022. https://www.atlassian.com/blog/archives/atlassian_acquires_bitbucket_org_distributed_version_control_system_hosting$.

11. Google 2022. Google Workspace Learning Center. Check or revert to earlier file versions. Hakupäivä 18.4.2022. https://support.google.com/a/users/answer/9308971?hl=en.

12. Microsoft 2022. Office support. View previous versions of Office files. Hakupäivä 18.4.2022. https://support.microsoft.com/en-us/office/view-previous-versions-of-office-files-5c1e076f-a9c9-41b8-8ace-f77b9642e2c2.

13. Spinellis, Diomidis. 2012. Git. IEEE Software (Volume: 29, Issue: 3, May-June 2012). IEEE Personal Account. Hakupäivä: 4.3.2022. https://ieeexplore.ieee.org/document/6188603. Vaatii käyttöoikeuden.

14. Sink, Eric. 2011. Version Control by Example. Hakupäivä 1.5.2022. https://eric-sink.com/vcbe/html/index.html.

15. Omron 2021. Sysmac Studio. Hakupäivä 16.6.2022. https://industrial.omron.eu/en/products/sysmac-studio#features.

16. Atlasian. Bitbucket. What is Git. Hakupäivä 10.5.2022. https://www.atlassian.com/git/tutorials/what-is-git.

17. Bitbucket. A brief overview of Bitbucket. Hakupäivä 12.8.2022. https://bitbucket.org/product/guides/getting-started/overview#a-brief-overview-of-bitbucket.

18. GitLab 2022. GitLab Features. Hakupäivä 12.8.2022. https://about.gitlab.com/features/.

19. Gitea 2022. What is Gitea. Hakupäivä 12.8.2022. https://docs.gitea.io/en-us/.

20. GitHub 2022. Let's build from here. The complete developer platform to build, scale, and deliver secure software. Hakupäivä 12.8.2022. https://github.com/about.

21. Google Trends. Compare. Hakupäivä 9.8.2022. https://trends.google.com/trends/explore?q=github,gitlab,bitbucket,gitea.

22. Git. Versio 2.37.3. Hakupäivä 20.7.2022. https://git-scm.com/docs/gitweb.

23. Atlassian. Comparing Workflows. Hakupäivä 28.7.2022. https://www.atlassian.com/git/tutorials/comparing-workflows#centralized-workflow.

24. Atlassian. Git Feature Branch Workflow. Hakupäivä 28.7.2022. https://www.atlassian.com/git/tutorials/comparing-workflows/feature-branch-workflow.

25. Atlassian. Gitflow workflow. Hakupäivä 28.7.2022. https://www.atlassian.com/git/tutorials/comparing-workflows/gitflow-workflow.

26. Atlassian. Forking Workflow. Hakupäivä 28.7.2022. https://www.atlassian.com/git/tutorials/comparing-workflows/forking-workflow.

27. Beckhoff 2021. TE1000 TwinCAT 3, Source-Control Manual. Version 1.2. Hakupäivä 16.6.2022. https://download.beckhoff.com/download/document/automation/twincat3/TC3_Source-Control_EN.pdf.

28. B&R Automation 2022. Automation Studio, B&R Help Explorer 4.2. Ohjelmointiympäristön mukana tullut sovellus.

29. Microsoft 2022. Microsoft Visual SourceSafe 2005. Hakupäivä 7.8.2022. https://docs.microsoft.com/en-us/lifecycle/products/microsoft-visual-sourcesafe-2005.

30. Atlasian. Bitbucket. Comparing Workflows. Hakupäivä 10.5.2022. https://www.atlassian.com/git/tutorials/comparing-workflows.

31. Siever, Ellen, Spainhour, Stephen, Figgins, Stephen & Hekman, Jessica P. 2000. Linux in a Nutshell. 3rd Edition. USA: O'Reilly & Associates, Inc. Hakupäivä 10.5.2022. https://docstore.mik.ua/orelly/linux/lnut/index.htm.

32. Association for Advancing Automation 2022. Getting Started with Automation: How to Start Your Journey. Hakupäivä 20.5.2022. https://www.automate.org/editorials/getting-started-with-automation-how-to-start-your-journey.

33. ISO 8373 2021. Robotics — Vocabulary. The International Organization for Standardization ISO. Hakupäivä 11.6.2022. https://www.iso.org/obp/ui/#iso:std:iso:8373:ed-3:v1:en:term:4.17.

34. ISO 8373 2012. Robots and robotic devices — Vocabulary. The International Organization for Standardization ISO. Hakupäivä 11.6.2022. https://www.iso.org/obp/ui/#iso:std:iso:8373:ed-2:v1:en.

35. International Federation of Robotics 2021. Press Conference World Robotics 2021. Hakupäivä 11.6.2022. https://ifr.org/downloads/press2018/2021_10_28_WR_PK_Presentation_long_version.pdf.

36. Daedal Research 2022. Global Industrial Robot Market: Analysis By Industry, By Type, By Region Size and Trends with Impact of COVID-19 and Forecast up to 2026. Hakupäivä 10.7.2022. https://www.marketresearch.com/Daedal-Research-v3440/Global-Industrial-Robot-Type-Region-31701189/.

37. SFS-EN ISO 10218-1 2011. Robotit ja robotiikkalaitteet. Turvallisuusvaatimukset. Osa 1: Teollisuusrobotit. Helsinki: Suomen Standardisoimisliitto SFS. Hakupäivä 15.5.2022. https://sales.sfs.fi/fi/index/tuotteet/SFS/CENISO/ID2/1/235190.html.stx. Vaatii käyttöoikeuden.

38. ISO/TS 15066 2016. Technical Specification. Robots and robotic devices — Collaborative robots. The International Organization for Standardization ISO. Hakupäivä 15.5.2022. https://sales.sfs.fi/fi/index/tuotteet/ISO/ISO/ID9998/1/402851.html.stx. Vaatii käyttöoikeuden.

39. International Federation of Robotics 2021. Industrial Robots. Hakupäivä 12.6.2022. https://ifr.org/industrial-robots.

40. Omron 2021. Machine Automation Controller. Hakupäivä 8.7.2022. https://assets.omron.eu/downloads/brochure/en/v4/p089_nj_nx_series_brochure_en.pdf.

41. Siemens 2022. Industry Mall. SIMATIC Industrial Automation Systems. Hakupäivä 8.7.2022. https://mall.industry.siemens.com/mall/en/WW/Catalog/Products/5009999.

42. Siemens 2019. Engineering framework ready for continuous software development. Hakupäivä 6.7.2022. https://press.siemens.com/global/en/pressrelease/engineering-framework-ready-continuous-software-development.

43. Siemens 2020. TIA Add-In for TIA Portal V16: VCI Git Connector. Hakupäivä 6.7.2022. https://support.industry.siemens.com/cs/attachments/109773999/109773999_TIA_Add-In_VCI_Git_Connector_1.0.0_APPL.zip. Vaatii kirjautumisen.

44. Siemens 2021. SIMATIC STEP 7 and WinCC Engineering V17 System Manual. Hakupäivä 6.7.2022. https://support.industry.siemens.com/cs/attachments/109798671/STEP_7_WinCC_V17_enUS_en-US.pdf?download=true. Vaatii kirjautumisen.

45. ABB 2022. RobotStudio. Hakupäivä 5.5.2022. https://new.abb.com/products/robotics/robotstudio.

46. Mainio, Juha 2022. Product Manager. ABB Oy, Robotics. Haastattelu 1.6.2022.

47. Fanuc. Robot Accessories. Hardware and software. Hakupäivä 7.8.2022. https://www.fanuc.eu/uk/en/robots/accessories.

48. KUKA 2022. KUKA Operating Systems. Hakupäivä 7.8.2022. https://www.kuka.com/en-de/products/robot-systems/software/system-software.

49. Wick-Hartmann, Daniel 2022. Support Engineer Core Software. KUKA Deutschland GmbH. Sähköpostikeskustelu 13.7.2022.

50. Omron 2022. Automation Control Environment (ACE). Version 4. Hakupäivä 6.6.2022. https://assets.omron.eu/downloads/manual/en/v6/i633_ace_4.0_users_manual_en.pdf.

51. Puusa, Anu & Juuti, Pauli 2020. Laadullisen tutkimuksen näkökulmat ja menetelmät (toim. Anu Puusa & Pauli Juuti). Gaudeamus.

52. Vilkka, Hanna 2021. Tutki ja kehitä. 5. päivitetty painos. Jyväskylä: PS-kustannus.

53. Frimodig, Benjamin 2022. What Are Heuristics? Simply Psychology. Hakupäivä 7.7.2022. https://www.simplypsychology.org/what-is-a-heuristic.html.

54. Tilastokeskus. Tietotarpeen määrittely ja tutkimusongelman täsmentäminen. Hakupäivä 18.8.2022. https://www.stat.fi/tup/htpalvelut/haastutk_toiminta_tutkasetelma.html.

55. Saaranen-Kauppinen, Anita & Puusniekka, Anna 2006. KvaliMOTV – Menetelmäopetuksen tietovaranto. Tampere: Yhteiskuntatieteellinen arkisto. Hakupäivä 25.6.2022. https://www.fsd.tuni.fi/menetelmaopetus/kvali/L2_3_2_2.html.

56. Valli, Raine 2015: Aineistonkeruu kyselylomakkeella. Teoksessa Valli, Raine (toim.): Ikkunoita tutkimusmetodeihin 1. Metodin valinta ja aineistonkeruu: vinkkejä aloittelevalle tutkijalle. 5. uudistettu painos. Jyväskylä: PS-kustannus.

57. Pitkäranta, Ari 2014. Laadullinen tutkimus opinnäytetyönä. Työkirja ammattikorkeakouluun. Jokioinen: e-Oppi Oy.

58. Laine, Timo 2018. Miten kokemusta voidaan tutkia? Fenomenologinen näkökulma. Teoksessa Valli, Raine (toim.): Ikkunoita tutkimusmetodeihin 2. Näkökulmia aloittelevalle tutkijalle tutkimuksen teoreettisiin lähtökohtiin ja analyysimenetelmiin. 5. uudistettu ja täydennetty painos. PS-kustannus. Jyväskylä.

59. Oulun ammattikorkeakoulu 2022. Opinnäytetyö. Hakupäivä 30.6.2022. https://www.oamk.fi/opinto-opas/opintojen-sisalto/opinnaytetyo.

60. Arene 2019. Ammattikorkeakoulujen opinnäytetöiden eettiset suositukset. Ammattikorkeakoulujen rehtorineuvosto Arene ry. Hakupäivä 30.6.2022. https://www.arene.fi/wp-content/uploads/Raportit/2020/AMMATTIKORKEAKOULUJEN%20OPINN%C3%84YTET%C3%96IDEN%20EETTISET%20SUOSITUKSET%202020.pdf?_t=1578480382.

61. Hiltunen, Leena 2009. Validiteetti ja reliabiliteetti. Graduryhmä 18.2.2009. Jyväskylän yliopisto. Hakupäivä 10.7.2022. http://www.mit.jyu.fi/ope/kurssit/Graduryhma/PDFt/validius_ja_reliabiliteetti.pdf.

KUTSUKIRJE LIITE 1

Yritys

Henkilön nimi
Tehtävä

Pyyntö osallistua tutkimukseen teollisuusrobottien ohjelmakoodin versionhallinnasta

Taustaa

Teollisuusrobotiikan sovellukset ovat koko ajan modulaarisempia, tehokkaampia ja ne tulee toteuttaa lyhyessä aikataulussa. Vaativammat sovellukset yhä lyhyemmässä ajassa vaativat ohjelmointityöltä tehokkuutta ja mahdollisesti aiemmin tehdyn työn hyödyntämistä. Näiden tavoitteiden saavuttamisessa voivat auttaa oikeanlaiset työkalut, kuten **versionhallinta**.

Mistä on kysymys?

Varteenotettavat ohjelmistokehittäjät käyttävät lähes poikkeuksetta työssään versionhallintaa. Versionhallintaan on ollut saatavilla jo pitkään useita järjestelmiä. Hyödyntävätkö teollisuusrobottien ohjelmoijat versionhallintaa? Olisiko teollisuusrobottien ohjelmointiin mahdollista tuoda kaikki versionhallinnan edut, joita ohjelmistokehittäjät ovat muissa ohjelmointiympäristöissä jo pitkään hyödyntäneet?

Tavoite ja tulokset

Tutkimus toteutetaan osana tutkimuksen tekijän robotiikan ylemmän ammattikorkeakoulutukinnon suoritusta. Tutkimuksessa selvitetään, **onko versionhallinta käytössä robottiohjelmoijien tai loppukäyttäjien työkaluna**, ja millaisia kokemuksia tai ennakkoasenteita siihen liittyy. Tutkimuksessa **selvitetään hajautetun versionhallinnan tarpeellisuutta ja hyötyjä teollisuusrobotiikassa**, sekä edellytyksiä sen käyttöön. Tutkimuksessa osoitetaan käytännössä versionhallinnan toimivuus robottien ohjelmoinnissa. Tuloksien perusteella voidaan päätellä, voiko hajautettu versionhallinta toimia teollisuusrobotiikassa ja millä reunaehdoilla kaikki robotiikan parissa työskentelevät voisivat siitä hyötyä. Tavoitteena on, että teollisuusrobotiikan ohjelmointiin voitaisiin saada näillä tiedoilla uutta tehokkuutta.

Tutkimusaineisto

Tutkimuksen on tavoite valmistua kesään 2022 mennessä. Tutkimus on valmistumisen jälkeen julkisesti saatavilla valtakunnallisessa Theseus-opinnäytetyö- ja julkaisutietokannassa ja näin ollen kaikkien hyödynnettävissä. Tutkimusaineisto ei yksilöi ketään vastaajaa ja aineisto säilytetään vain työn arviointiin saakka.

Vastaaminen

Vastaaminen kyselyyn käy ketterästi oheisen linkin kautta, ja linkki on sama kaikille tutkimukseen valituille yrityksille. Kyselyn tuloksia ei yksilöidä ja tiedot jäävät vain opinnäytetyöntekijän ja työn arvioijan tietoon siihen saakka, kunnes työ on arvioitu. Tämän jälkeen kyselyn vastauksena saadut tiedot hävitetään. **Vastaamalla tähän kyselyyn voitte olla edistämässä teollisuusrobotiikan kehittymistä**

yhä tehokkaampaan ja ketterämpään suuntaan. **Pyydän vastaamaan kyselyyn 19.12.2021 menessä. Mikäli koette, että joku toinen henkilö yrityksessänne sopisi paremmin kyselyn vastaajaksi, pyydän ohjaamaan kyselyn hänelle. Pyydän vastaamaan kyselyyn vain yhden henkilön toimesta.**

Kyselyn **salasana**: Robotiikka2021

TÄSTÄ PÄÄSETTE KYSELYYN.

Ystävällisin terveisin,
Tuomas Lipponen
Automation Engineer, Insinööri (AMK)
Robotiikan tutkinto-ohjelman opiskelija (YAMK)
[redacted]

OAMK OULUN AMMATTIKORKEAKOULU

Versionhallinta teollisuusrobotiikassa

Pakolliset kysymykset merkitty tähdellä (*)

Ammattiryhmä, johon katsotte lähinnä kuuluvanne? *

- ○ Yrityksen johto
- ○ Teknologiajohtaja
- ○ Myyntipäällikkö
- ○ Automaatio/sähkö/robotiikkainsinööri
- ○ Projekti-insinööri
- ○ Tekninen myyjä
- ○ Asentaja
- ◉ Muu

Toimiala, johon katsotte lähinnä kuuluvanne?

- ○ Robottien loppukäyttäjä
- ○ Robotisoitujen koneiden rakennus
- ○ Robottien valmistaja
- ○ Robottien ohjelmointipalvelujen tuottaja
- ○ Konsultointi
- ○ Muu

Kuinka paljon yrityksessänne on työntekijöitä?

- ○ >200
- ○ 100-200
- ○ 50-100
- ○ 10-50
- ○ <10

Kuinka moni henkilö tyypillisesti työskentelee projektissa saman robotin parissa yhtäaikaa?

- ○ >10
- ○ 5-10
- ○ 2-4
- ○ 1

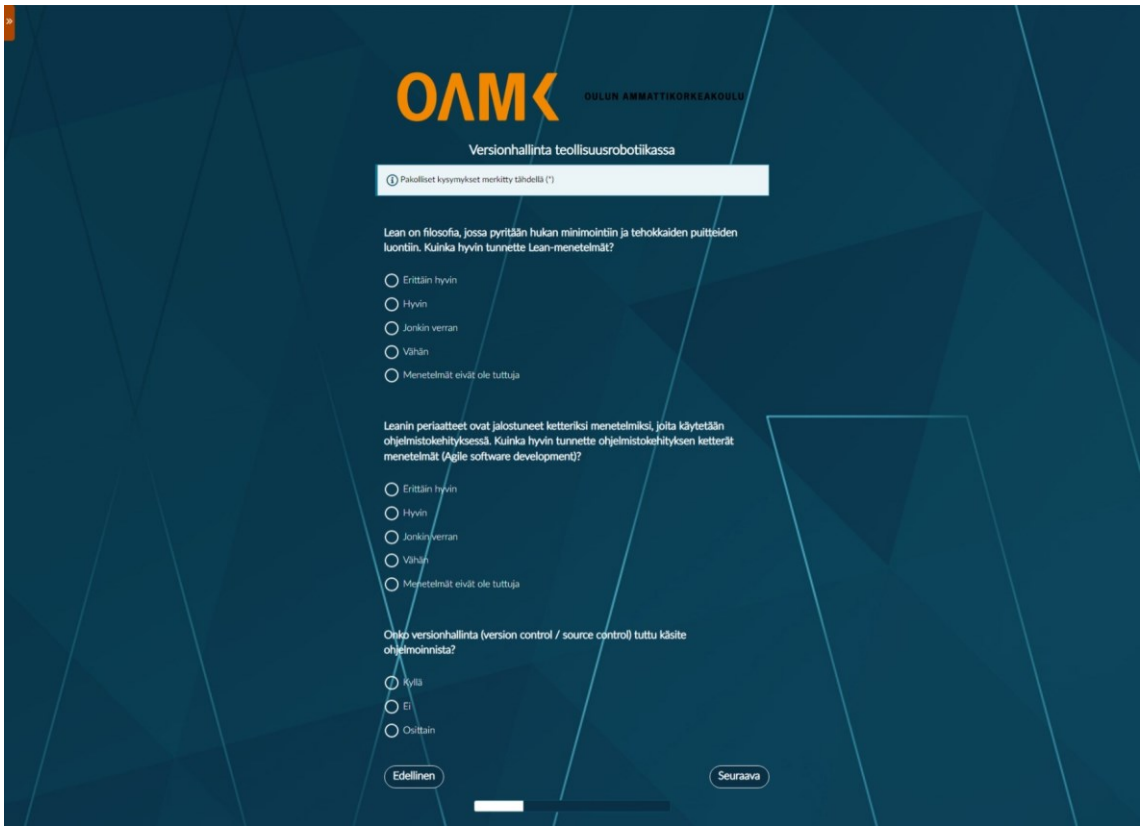
Kuinka monta vuotta olette työskennelleet robotiikkaan liittyvissä työtehtävissä?

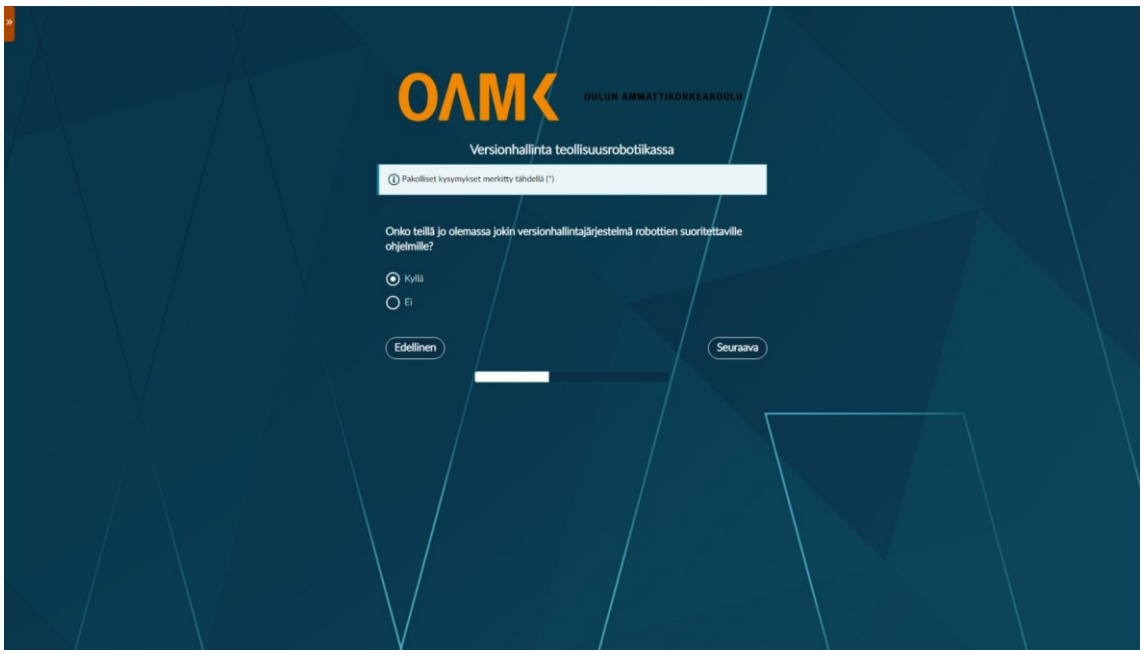
- ○ 0-5
- ○ 5-10
- ○ 10-20
- ○ > 20

Minkä valmistajan robotteja käytätte tyypillisimmin?

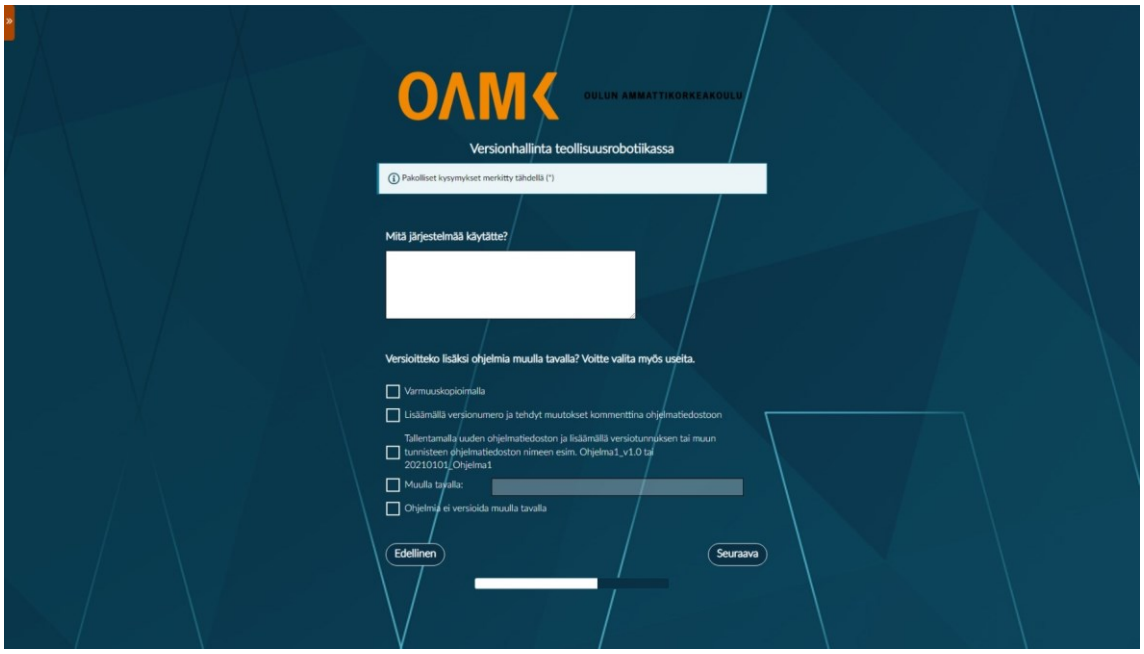
- ☐ ABB
- ☐ Comau
- ☐ Epson Robots
- ☐ Fanuc
- ☐ Kawasaki
- ☐ KUKA
- ☐ Omron Adept
- ☐ Stäubli
- ☐ Yaskawa
- ☐ Universal robots
- ☐ Muu

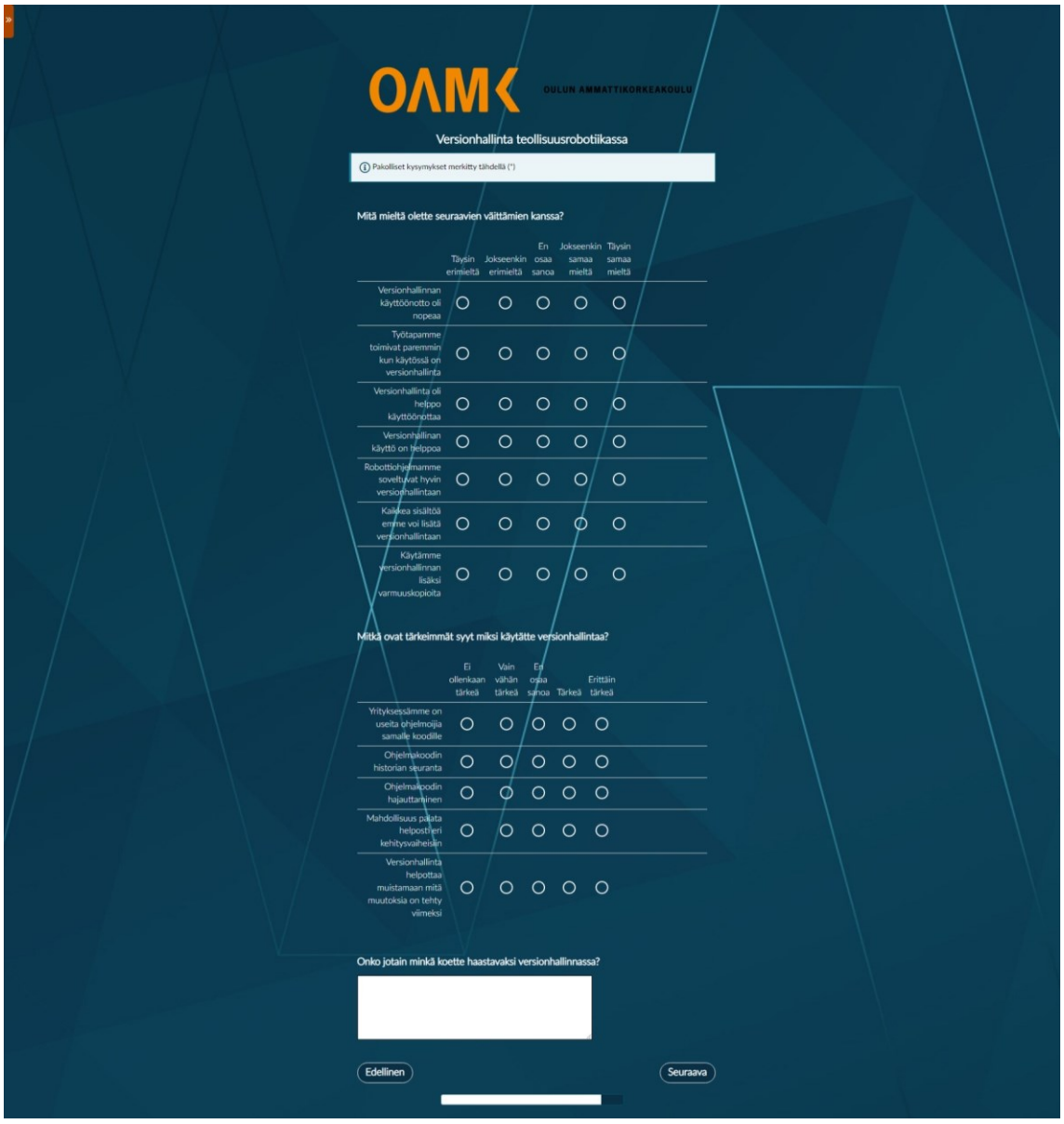
Seuraava



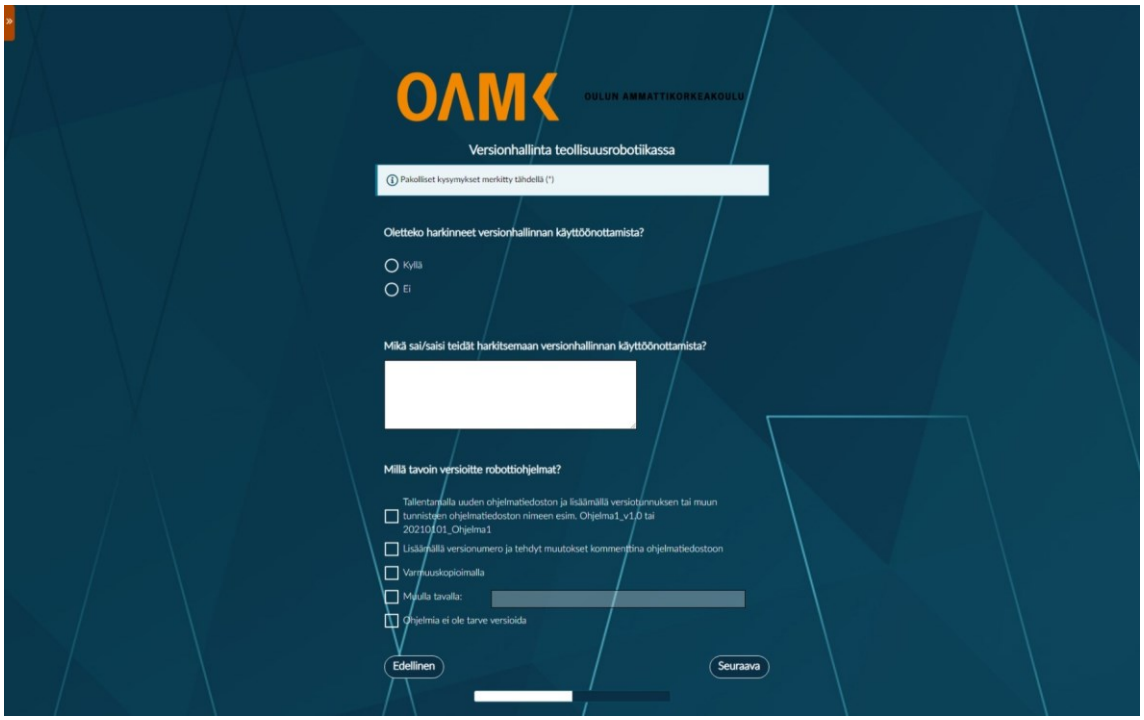


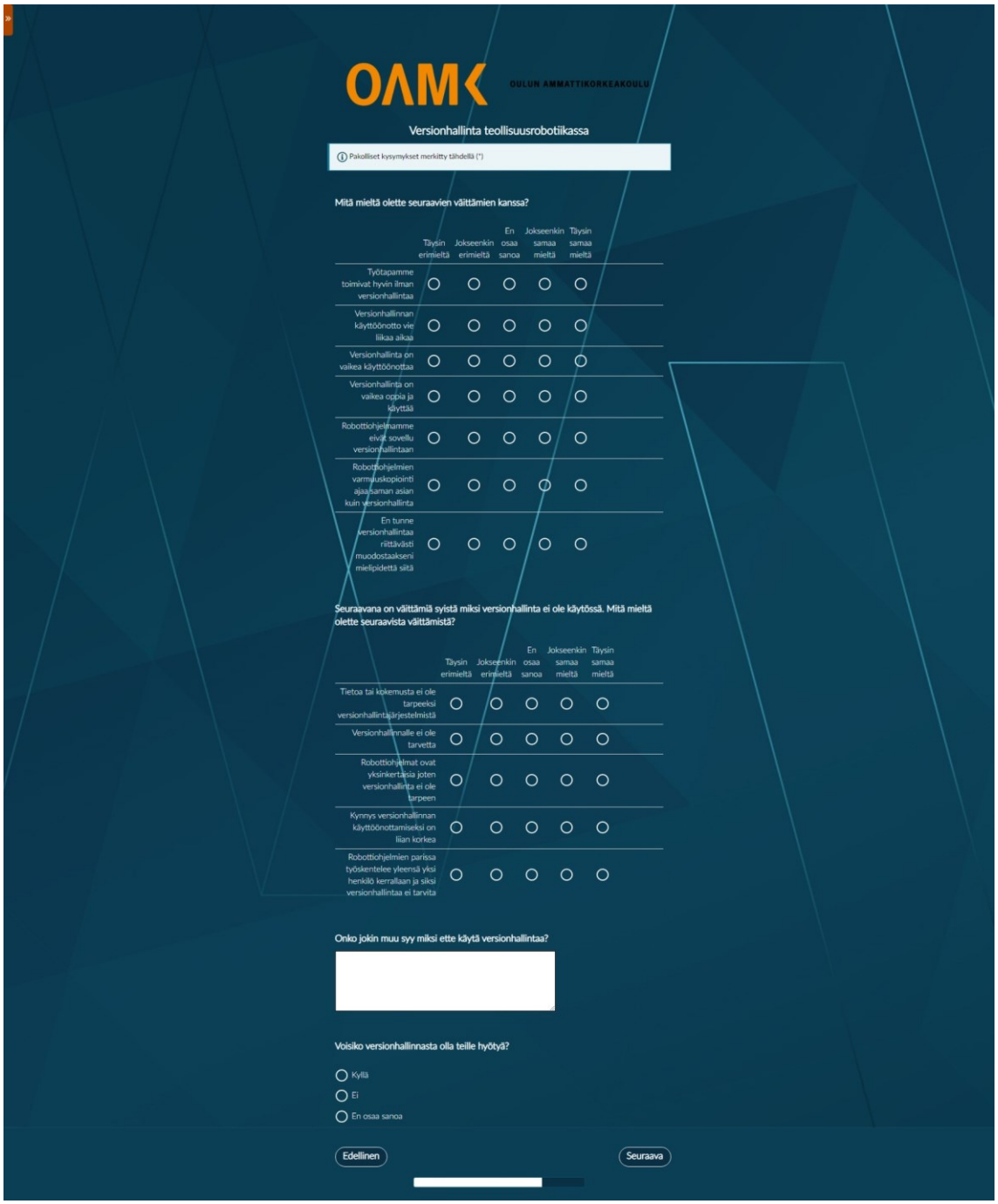
Jos vastaus kysymykseen “Onko teillä jo olemassa jokin versionhallinta järjestelmä robottien suoritettaville ohjelmille” oli kyllä:


OULUN AMMATTIKORKEAKOULU
Versionhallinta teollisuusrobotiikassa
Pakolliset kysymykset merkitty tähdellä (*)
Mitä järjestelmää käytätte?
Versioitteko lisäksi ohjelmia muulla tavalla? Voitte valita myös useita.
Varmuuskopioimalla
Lisäämällä versionumero ja tehdyt muutokset kommenttina ohjelmatiedostoon
Tallentamalla uuden ohjelmatiedoston ja lisäämällä versiotunnuksen tai muun tunnisteen ohjelmatiedoston nimeen esim. Ohjelma1_v1.0 tai 20210101_Ohjelma1
Muulla tavalla:
Ohjelmia ei versioida muulla tavalla
Edellinen
Seuraava


OULUN AMMATTIKORKEAKOULU
Versionhallinta teollisuusrobotiikassa
Pakolliset kysymykset merkitty tähdellä (*)
Mitä mieltä olette seuraavien väittämien kanssa?
Täysin erimieltä
Jokseenkin erimieltä
En osaa sanoa
Jokseenkin samaa mieltä
Täysin samaa mieltä
Versionhallinnan käyttöönotto oli nopeaa
Työtapamme toimivat paremmin kun käytössä on versionhallinta
Versionhallinta oli helppo käyttöönottaa
Versionhallinan käyttö on helppoa
Robottiohjelmamme soveltuvat hyvin versionhallintaan
Kaikkea sisältöä emme voi lisätä versionhallintaan
Käytämme versionhallinnan lisäksi varmuuskopioita
Mitkä ovat tärkeimmät syyt miksi käytätte versionhallintaa?
Ei ollenkaan tärkeä
Vain vähän tärkeä
En osaa sanoa
Tärkeä
Erittäin tärkeä
Yrityksessämme on useita ohjelmoijia samalle koodille
Ohjelmakoodin historian seuranta
Ohjelmakoodin hajauttaminen
Mahdollisuus palata helposti eri kehitysvaiheisiin
Versionhallinta helpottaa muistamaan mitä muutoksia on tehty viimeksi
Onko jotain minkä koette haastavaksi versionhallinnassa?
Edellinen
Seuraava

Jos vastaus kysymykseen “Onko teillä jo olemassa jokin versionhallinta järjestelmä robottin suoritettaville ohjelmille” oli ei:





OULUN AMMATTIKORKEAKOULU

Versionhallinta teollisuusrobotiikassa

Pakolliset kysymykset merkitty tähdellä (*)

Mitä mieltä olette seuraavien väittämien kanssa?

	Täysin erimieltä	Jokseenkin erimieltä	En osaa sanoa	Jokseenkin samaa mieltä	Täysin samaa mieltä
Työtapamme toimivat hyvin ilman versionhallintaa	○	○	○	○	○
Versionhallinnan käyttöönotto vie liikaa aikaa	○	○	○	○	○
Versionhallinta on vaikea käyttöönottaa	○	○	○	○	○
Versionhallinta on vaikea oppia ja käyttää	○	○	○	○	○
Robottiohjelmamme eivät sovellu versionhallintaan	○	○	○	○	○
Robottiohjelmien varmuuskopiointi ajaa saman asian kuin versionhallinta	○	○	○	○	○
En tunne versionhallintaa riittävästi muodostaakseni mielipidettä siitä	○	○	○	○	○

Seuraavana on väittämiä syistä miksi versionhallinta ei ole käytössä. Mitä mieltä olette seuraavista väittämistä?

	Täysin erimieltä	Jokseenkin erimieltä	En osaa sanoa	Jokseenkin samaa mieltä	Täysin samaa mieltä
Tietoa tai kokemusta ei ole tarpeeksi versionhallintajärjestelmistä	○	○	○	○	○
Versionhallinnalle ei ole tarvetta	○	○	○	○	○
Robottiohjelmat ovat yksinkertaisia joten versionhallinta ei ole tarpeen	○	○	○	○	○
Kynnys versionhallinnan käyttöönottamiseksi on liian korkea	○	○	○	○	○
Robottiohjelmien parissa työskentelee yleensä yksi henkilö kerrallaan ja siksi versionhallintaa ei tarvita	○	○	○	○	○

Onko jokin muu syy miksi ette käytä versionhallintaa?

Voisiko versionhallinnasta olla teille hyötyä?

- ○ Kyllä
- ○ Ei
- ○ En osaa sanoa

Edellinen Seuraava

Lopuksi kaikille yhteinen viimeinen kysymys: