



TAMPEREEN TEKNILLINEN YLIOPISTO  
TAMPERE UNIVERSITY OF TECHNOLOGY

**JONI LAMMASNIEMI  
OHJELMOITAVIEN KOMPONENTTIEN VERSIONHALLINTA  
TEOLLISUUSYMPÄRISTÖSSÄ**

Diplomityö

Tarkastaja: Prof. Pekka Ruuskanen  
Tarkastaja ja aihe hyväksytty  
Talouden ja rakentamisen tiedekunta-  
neuvoston  
kokouksessa 24.9.2018

# TIIVISTELMÄ

**JONI LAMMASNIEMI:** Ohjelmoitavien komponenttien versionhallinta teollisuusympäristössä

Tampereen teknillinen yliopisto

Diplomityö, 56 sivua

Lokakuu 2018

Tietotekniikan koulutusohjelma

Pääaine: Tietotekniikka

Tarkastaja: Prof. Pekka Ruuskanen

Avainsanat: versionhallinta, konfiguraationhallinta, ohjelmoitavat komponentit, versiondog, PLC, ohjelmoitavat logiikat, parametrien hallinta, ohjelmoitavat laitteet

Tässä diplomityössä käsitellään ohjelmoitavien komponenttien versionhallintaa teollisuusympäristössä. Ohjelmoitavien komponenttien merkitys kasvaa yrityksen tuotavuuden ja järjestelmien turvallisuuden näkökannalta, joten tulevaisuudessa yritysten tulee panostaa entistä enemmän myös niiden versionhallintaan. Tästä syystä työ koettiin ajankohtaiseksi. Nykypäivänä teollisuudessa dokumenttien versionhallinta on yleistynyt ja saatavilla on paljon kaupallisia ratkaisuja. Nämä eivät kuitenkaan sovellu sellaisenaan laitteiden ohjelmistojen konfiguraationhallintaan, joten tarvittavien määrittelyiden ja esiselvitysten näkökanta eroaa dokumenttien versionhallintaan tarkoitettusta ohjelmistosta.

Työssä käsitellään aiheeseen liittyviä taustoja ja velvoitteita sekä versionhallintaprojektin kulkua määrittelydokumentaatiosta esimerkkiprojektin yhteydessä esiintyneisiin ongelmiin. Työssä pyritään antamaan riittävästi alkutietoja projektin toteuttamiseksi. Jokaisen yrityksen tarve on kuitenkin yksilöllinen, joten työn tuloksia ei voida suoraan käyttää minkä tahansa yrityksen tarpeisiin.

Määrittelydokumentaation merkitys on suuri projektin onnistumisen kannalta, koska vajavaisesta tai väärästä määrittelystä johtuvaa virhettä on hankala korjata eivätkä projektin kustannukset pysy annetuissa rajoissa. Ohjelmoitaviin komponentteihin liittyy monia erityispiirteitä, jotka tulee ottaa huomioon määrittelydokumentaatioissa ja esimerkiksi verkkoratkaisuiden merkitys on suuri. Myös yrityksen toimintamallit vaikuttavat suuresti valittavan ohjelmiston vaatimuksiin, koska paikallisesti tai useassa toimipisteessä toimivien yritysten vaatimukset eroavat toisistaan.

Työn lopussa käsitellään versiondog-nimistä ohjelmoitavien komponenttien versionhallintaan tarkoitettua ohjelmistoa. Esimerkkiprojektissa käydään lävitse ohjelmiston valintaprosessia, ohjelmiston toimintaa ja ohjelmiston käyttöönotossa vastaan tulleita ongelmia ja niiden ratkaisuita. Tämän lisäksi käydään läpi versiondog:iin liittyvää ohjeistusta ja käyttöympäristöjen jaottelua.

## ABSTRACT

**JONI LAMMASNIEMI:** Version control of programmable devices in industrial environment

Tampere University of Technology

Diplomityö, 56 pages

October 2018

Master's Degree Programme in Pervasive Technology

Major:Pervasive Computing

Examiner: Prof. Pekka Ruuskanen

Keywords: version control, configuration management, programmable component, versiondog, PLC, programmable logic controller, parameter management, programmable devices

This master's thesis studies the version control of programmable devices in industrial environments. The significance of the programmable components for a company's productivity and use in safety related systems are steadily growing. This means that in the future the version control of these components is becoming more substantial for the companies. These are the main reasons this master's thesis was regarded as current. The configuration management of programmable devices has risen to the same significance as document management. But there are some key differences how to define and make a preliminary planning for acquiring a version control software for programmable devices as compared to similar software for documents.

This study discusses about the background of the subject and related obligations. Version control software project and necessary definition documents are studied in later parts of this master's thesis. Also encountered problems are discussed. This study gives good starting point to put similar projects into action, but the results cannot be used as a given due to different needs of the companies.

Preliminary documentation is a key factor of a successful project, because an insufficient or wrong definition can lead to mistakes which are hard to correct and will cause extra costs. Programmable devices have some special characteristics which needs to be taken into account in preliminary documentation. Company working environment has also affects for definition documentation, because there are differences in needs if company is located at one place compared to multiple working sites. Chapter six studies version control software called versiondog. This project gives example what shall be taken into account when choosing a software, how the software works, and problems risen during commissioning of the software. Also needed instructions and dividing to different use environments are discussed in this chapter.

## ALKUSANAT

Kiinnostuin aiheesta sen monipuolisuuden vuoksi ja siitä syystä, että aiheesta ei ole juurikaan julkaistu tutkimuksia tai diplomitöitä. Aihe on monipuolinen ja tarjosi mahdollisuuden käyttää omaa osaamistaan laaja-alaisesti. Ohjelmistoprojekteihin liittyvät opinnot olivat hyödyksi projektin aikana. Lisäksi yhteistyö ja ongelmien ratkominen eri alojen ihmisten kanssa toi uusia näkökantoja oman työni tekemiseen.

Projektissa oli mahdollista yhdistää omaa teknistä osaamista ja parantaa vuorovaikutustaitoja ihmisten kanssa. Näin laajan projektin toteuttaminen yksin ei olisi ollut mahdollista. Projektissa aktiivisesti mukana olleista henkilöistä tulikin työn edetessä enemmän kuin pelkkiä työkavereita. Heidän kanssaan tuli vietettyä aikaa myös vapaa-ajalla, vaikkakin keskustelu kääntyi useasti projektiin liittyviin asioihin myös työajan ulkopuolella.

Haluan kiittää professori Pekka Ruuskasta työn ohjaamisesta ja joustavuudesta aikataulujen suhteen. Suuri kiitos kuuluu myös työkollegoille, jotka ovat antaneet oman näkökantansa työssä käsiteltyihin aiheisiin. Erityisesti Minna Innalan apu oli työn viimeistelyn kannalta merkittävä. Kiitokset kuuluvat myös Novotek Oy:lle ja AUVESY GmbH:lle versiondog:n demolisenssin toimittamisesta. Viimeisimpänä muttei vähäisempänä kiitokset kuuluvat läheisille ja eritoten lapsilleni, jotka ovat jaksaneet katsella vieressä diplomityön etenemistä.

Pori, 30.10.2018

Joni Lammasniemi

# SISÄLLYS

1. Johdanto . . . . .	2
2. Taustaa . . . . .	4
2.1 Aiheesta aikaisemmin tehdyt tutkimukset . . . . .	5
2.2 Versionhallintaan liittyvä lainsäädäntö, velvoitteet ja standartit . . . . .	6
3. Versionhallintajärjestelmän määritelmä ja määrittely . . . . .	8
3.1 Tarvittavien lähtötietojen määrittely . . . . .	9
3.2 Versionhallinnassa olevien tietojen määrittely . . . . .	11
3.3 Tallennettavien tietojen vastaanotto . . . . .	12
4. Versionhallintajärjestelmän käyttöönotto . . . . .	14
4.1 Vastaanotettujen tietojen vienti versionhallinnan alle . . . . .	15
4.2 Versionhallintajärjestelmän tietorakenteen määrittely . . . . .	15
4.3 Versionhallintaan liittyvät verkkoratkaisut . . . . .	16
4.4 Käyttöönoton yhteydessä mahdollisesti esiintyviä ongelmia . . . . .	18
5. Versionhallinnan tehtävät . . . . .	20
5.1 Integrointi osaksi jo olemassa olevia tietojärjestelmiä . . . . .	20
5.2 Versionhallinnan tyypilliset käyttäjäryhmät . . . . .	23
5.3 Versionhallintaan liittyvä ohjeistus . . . . .	25
5.4 Versionhallinnassa olevien tietojen käsittely . . . . .	26
5.5 Versionhallinnan liittäminen toimintamalleihin . . . . .	28
6. Versionhallintaohjelmiston käyttöönottoprojekti ja ohjelmiston esittely . . . . .	29
6.1 Määrittely . . . . .	29
6.2 Ohjelmiston valinta ja testikäyttö . . . . .	30
6.3 Ohjelmiston esittely . . . . .	31
6.4 Ohjelmoitavien laitteiden ohjelmien siirto versionhallintaan . . . . .	35
6.5 Eroavaisuuksien esittäminen . . . . .	38
6.6 Vastaantulleita ongelmia ja niiden ratkaisuja . . . . .	39
6.7 Versiondog:n käyttöä varten luodut ympäristöt . . . . .	42

6.8	Versiondog:n kouluttaminen yrityksen sisällä . . . . .	43
7.	Yhteenveto . . . . .	44
	Lähteet . . . . .	47

## KUVALUETTELO

1	Versionhallintaan tallennettavien lähtötietojen määrittely. . . . .	10
2	Esimerkkejä versionhallintaan vietävistä tiedoista. . . . .	12
3	Tietojen tuonti versionhallintaan. . . . .	13
4	Esimerkki verkkoratkaisusta ja tietoturvasta. . . . .	17
5	Esimerkki RAID 1 kokoonpanosta. . . . .	17
6	Esimerkki integrointiprosessista. . . . .	22
7	Esimerkki käyttäjärühmistä ja niihin liittyvistä toiminnoista. . . . .	24
8	Tietojen ulosvientiprosessi. . . . .	27
9	WebClient käyttöliittymä. . . . .	33
10	EasyClient käyttöliittymä. . . . .	33
11	UserClient käyttöliittymä. . . . .	34
12	AdminClient käyttöliittymä. . . . .	34
13	ReportClientin käyttöliittymä. . . . .	35
14	Uuden komponentin luominen. . . . .	36
15	Komponentin tyyppin valinta ja metatietojen täyttö. . . . .	37
16	Komponentin paikalliskopio valmiina. . . . .	37
17	Komponentti kirjattu sisään järjestelmään. . . . .	38
18	Tarkistussumman ja kääntöajan eroavaisuuden esitys. . . . .	39
19	Tikapuuohjelman eroavaisuuden esitys. . . . .	39
20	Version muutokset komponentin sisällä. . . . .	40

## LYHENTEET JA MERKINNÄT

Active Directory	Microsoftin kehittämä hakemisto ja käyttäjätietokanta
ASCII	Yleisin tekstitiedostojen muoto, jossa jokainen merkki koostuu 7 bitistä (eng. American Standard Code for Information Interchange)
CODESYS	IEC 61131-3 mukainen ohjelmointiympäristö
csv	Tiedostomuoto, jossa tieto on erotettu pilkuilla (eng. Comma Separated Value)
GIT	Hajautetusti toimiva versionhallintajärjestelmä, jonka toiminnassa on pyritty tehokkuuteen.
HMI	Ihmisen ja koneen välinen käyttöliittymä (eng. Human Machine Interface)
IAEA	Kansainvälinen atomienergiajärjestö (eng. International Atomic Energy Agency)
image	Kiintolevykopio, joka on täydellinen kopio levyn tiedosta ja rakenteesta.
KKS	Voimalaitoksilla käytettävä kohdistus järjestelmä, joka jakaa järjestelmät prosessien perusteella. (saksaksi Kraftwerk-Kennzeichensystem)
metatieto	Ohjelmalle annettavat tunnisteet, joilla komponentti voidaan tunnistaa.
pdf	Tiedostomuoto joka sisältää kaiken tiedon kuten tulostetussa version on esitetty (eng. Portable Document Format)
PL-luokitus	EN ISO 13849-1 mukainen luokitus, jonka luokat kertovat vaarallisten vikaantumisten määrän tuntia kohden
PLC	Ohjelmoitava logiikka, joka on erikoistunut reaaliaikaisten automaatioprosessien ohjaukseen (eng. Programmable Logic Controller).
STUK	Säteilyturvakeskus
SVN	Ohjelmistojen lähdekoodin hallintaan tarkoitettu versionhallintajärjestelmä
versiondog	AUVESY GmbH:n versionhallintajärjestelmä
YVL	Säteilyturvakeskuksen julkaisemat ja ylläpitämän ydinturvallisuusohjeet
XML	Merkintäkieli standardi, jossa tieto ja sen rakenne on kuvattu merkittävän tiedon joukkoon (eng. eXtensible Markup Language)

# 1. JOHDANTO

Tässä diplomityössä käsitellään ohjelmitavien laitteiden ohjelmistojen versionhallintaa teollisuusympäristössä. Aihe tuli ajankohtaiseksi nykyisessä työssä olevan projektin myötä ja asian taustojen tutkimisen perusteella vastaavasta aiheesta ei ole tehty diplomitöitä. Työn tavoitteena on antaa yleiskuva ohjelmitavien komponenttien versionhallinnasta kokonaisprojektina alkaen taustatutkimuksista sekä aiheeseen liittyvästä lainsäädännöstä, velvoitteista ja standarteista.

Työ etenee ennen versionhallinnan hankintaa tarvittavilla määrittelyillä ja niihin liittyvillä pohdinnoilla. Seuraavassa osiossa edetään versionhallinnan käyttöönottoon liittyvään dokumentaatioon ja lopussa esitetään esimerkkiohjelmisto. Työssä pyritään antamaan riittävästi tietoa vastaavanlaisen projektin määrittelyyn, ohjelmiston valintaan, käyttöönottoon ja versionhallinnan integroimiseen osaksi yrityksen toimintaa.

Diplomityössä esitellään myös vastaantulleita ongelmia ja niiden ratkaisuita, mutta työn tarkoitus ei ole olla valmis pohja vastaavan projektin toteuttamiseen. Työ sisältää pohjatietoa ja havaintoja, joita on tullut vastaan projektin toteuttamisen aikana. Työn lukijalta odotetaan teknistä pohjatietoa, mutta ei alan asiantuntemusta. Työn lopussa esiteltävä esimerkki ohjelmiston demolisenssi saatiin AU-VESY:lta ja ohjelmiston käyttö vastasi todellisia tilanteita.

Pohjamateriaalina on esimerkiksi Jenkins, Arnaud, Thompson, Yau ja John Wright julkaisema artikkeli "Version Control and Patch Management of Protection and Automation Systems" [1], joka käsittelee suoja-releiden ohjelmistopäivitysten hallinnan merkitystä ja riskejä, mutta artikkeli ei suoranaisesti ota kantaa versionhallinnan määrittelyyn. Käyttöönotto-osion pohjamateriaali perustuu Outi Arosen diplomityöhön "Tietojärjestelmän käyttöönotto ja sen arviointi" [2], joka käsittelee käyttöönoton aikana tehtyjä havaintoja ja käyttäjäkokemuksia. Dokumenttien hallinnan tärkeyttä käsitellään John Heckmann kokoamassa julkaisussa "Why Document Management: a White Paper" [3], jossa esitetään useita artikkeleita dokumenttien hallinnan tärkeydestä. Näiden artikkeleiden sisältöä on sovellettu tämän diplomityön pohjatietona.

Aiheeseen liittyvää lainsäädäntöä ei ole julkaistu, mutta toimialakohtaisia vaatimuksia on olemassa. Esimerkiksi STUK:n julkaisema YVL ohje A.8 [4] ottaa kantaa

luvanhaltijan velvollisuuksiin laitososien ikääntymiseen liittyen ja samassa yhteydessä mainitaan, että viitetiedot tulee tallentaa. Ohjelmoitavien komponenttien ohjelmistoja voidaan pitää edellä mainittuina viitetietoina. Vielä tarkemmin asiaan otetaan kantaa turvastandardissa SSG-39 [5], joka ottaa kantaa, miten ydinlaitosten ohjelmistojen versioita tulee hallita osana eliniänhallintaa. Myös ISO 9001 [6] tekee vaatimuksia laadunhallinnan osalta, ratkaisut näihin vaatimuksiin esitetään ISO 10007 [7] kohdassa 8.5.2.

Ennen ohjelmiston valintaa tulee tehdä riittävän tarkka määrittelydokumentaatio, jotta voidaan määrittää ohjelmiston ja toimittajan vaatimukset. Nämä vaatimukset ovat yritys- ja projektikohtaisia, joten tässä työssä on esitelty vaatimukset yleisellä tasolla, mutta painotetaan ohjelmoitavien komponenttien versionhallinnan erityistarpeita. Näitä ovat esimerkiksi versionhallintaan vietävien tietojen määrittely, tietoturvaratkaisut, verkkoratkaisut ja miten versionhallinnan tulee liittyä osaksi yrityksen aikaisempia tietojärjestelmiä. Lisäksi versionhallinnan tehtävät tulee määrittää etukäteen, jotta ohjelmistoja valittaessa voidaan tarkistaa, että kyseiset vaatimukset voidaan täyttää ohjelmiston toimesta. Vaatimuksia voidaan täydentää ja tulee tarkentaa, kun ohjelmisto on valittu, mutta etukäteen tehdyillä määrittelyillä voidaan nopeuttaa valintaprosessia ja tehostaa ohjelmiston käyttöönottoa.

Määrittelyn jälkeen siirrytään käyttöönottoprosessin vaatimiin määrittelydokumentteihin, joita ovat esimerkiksi tietojen tuonti versionhallintaohjelmistoon, tiedoista vaadittavat etukäteistiedot ja liitántärajapintojen määrittelyt. Ennen käyttöönottoa on hyvä suorittaa koekäyttö sopiville ohjelmistoille riittävän laajan käyttäjäryhmän toimesta. Käyttöönottoprosessin yhteydessä määritellään tarvittava ohjeistus ja luodaan tarvittavat käyttäjäryhmät, sekä tarkennetaan aikaisemmin luotua dokumentaatiota ohjelmisto kohtaisilla määrittelyillä. Käyttöönottoprojektin aikataulutusta ja sen seuraaminen on oleellinen osa onnistunutta projektia, jolla varmistetaan ettei projektille varatut resurssit ylitä.

Diplomityön viimeisessä osassa esitellään versiondog-niminen versionhallintaohjelmisto, joka koostuu kuudesta ohjelmasta ja kymmenestä lisäosasta. Tässä työssä ei esitellä lisäosien toimintaa. Esittelyssä käydään läpi perustoiminnot pintapuolisesti ja tutustutaan hallittavan komponenttien ohjelmistojen väliseen vertailuun. Lopussa esitellään vastaantulleita ongelmia ja näiden ratkaisua. Ongelmia aiheuttaa esimerkiksi versiondog:n tapa käsitellä kaikkia ohjelmia, tiedostoja ja kokonaisia projekteja komponentteina, ottamatta kantaa mitä nämä sisältävät. Tästä syystä vain projektin yhden osan palauttaminen edelliseen versioon ei ole mahdollista ilman, että menetetään kaikki tämän jälkeen tehdyt muutokset.

## 2. TAUSTAA

Ohjelmoitavien laitteiden ja komponenttien käyttö on erittäin yleistä teollisuudessa. Tämän myötä dokumenttien versionhallinnan rinnalle on tullut tarve ohjelmoitavien komponenttien ohjelmistojen versionhallintaan. Versionhallinta on mahdollista toteuttaa monella eri tasolla, jos ohjelmistoja on käytössä muutamia niin versionhallinnaksi riittää sopiva kansiorakenne ja nimeämisperusteen luominen eri versioiden välillä. Suuremman mittakaavan teollisuusympäristössä kuten autoteollisuus, lääketeollisuus ja voimalaitokset, voi olla kymmeniä tuhansia ohjelmoitavia laitteita tai komponentteja. Näistä jokainen sisältää yhden tai useampia ohjelmia ja käytössä voi olla samaan aikaan useita eri versioita samanlaisen komponentin ohjelmistosta. Tästä syystä näillä teollisuuden aloilla on syytä harkita erillistä versionhallintaa ohjelmoitaville komponenteille.

Versionhallinta toimii osana muutosten jäljitettävyyttä ja sen tarkoitus on ylläpitää tietoa ohjelmoitavien laitteiden ohjelmista, jotta tiedetään kulloinkin asennettu ja käytössä oleva versio. Versionhallinnan isoimmat käyttäjäryhmät voidaan jakaa kolmeen eri tyyppiin operaattorit, kunnossapito ja kehitysosasto. Operaattorit käyttävät järjestelmiä ja laitteita, joiden osana on ohjelmoitavia komponentteja. Kunnossapito tarvitsee tietoa ohjelmoitavien komponenttien ohjelmistoista vianhaun ja rikkoontuneen komponentin vaihdon yhteydessä, jolloin on erittäin oleellista että järjestelmä tai laite voidaan palauttaa vikaa edeltävään tilaan. Kehitysosasto kehittää järjestelmään ja laitteiden sisältämää koodia ja heille versionhallinta on oleellista muutossuunnittelua varten.

Ohjelmistokoodien versionhallintaan on kehitetty useita avoimia ja kaupallisia sovelluksia. Nämä sovellukset on suunnattu pääosin tekstimuotoisen lähdekoodin hallintaan, eivätkä täten välttämättä sovellu suoraan ohjelmoitavien komponenttien versionhallintaan. Tämä johtuu ohjelmien erilaisesta rakenteesta, joissa ohjelmien toiminnallisuus ei ole suoraan luettavissa lähdekoodista. Toiminnallisen eron määrittäminen vaatii lähdekoodin tulkitsemista sopivalla ohjelmistolla, jotka ovat esimerkiksi Siemensin osalta suljettuja ohjelmistoja. Tämä tarkoittaa, että ohjelmistoa ei ole vapaasti saatavilla.

## 2.1 Aiheesta aikaisemmin tehdyt tutkimukset

Version Control and Patch Management of Protection and Automation Systems [1] julkaisussa käsitellään automaatiojärjestelmien, erityisesti suojareleiden, automaation tuomia haasteita. Johtuen sähköisten komponenttien lisääntymisestä laitteiden elinkaariajattelu on muuttunut ja tarve ohjelmistojen päivitykseen laitteen fyysisen eliniän aikana on noussut. Julkaisussa käsitellään lyhyesti päivitysten riskiarviointia, testaamista, asennusten ajoitusta ja mahdollisia riskejä. Heidän mukaansa: “In this paper the authors have outlined the major reasons why version control and patch management is an important requirement for modern substation protection and automation systems, and how this can be implemented with effective processes and technology.” Ohjelmistojen ja päivitysten versionhallinnan merkitys on tärkeässä asemassa, mutta myös tästä johtuvat riskit tulee tunnistaa sekä pyrkiä hallitsemaan siltä osin kuin se on mahdollista.

Dokumenttien hallinnan merkitystä yritykselle ja sen tuomia etuja käsitellään John Heckmannin [3] kokoamassa julkaisussa. Hänen mukaansa tyypillinen takaisinmaksuaika dokumenttien hallintajärjestelmälle on maksimissaan kuusi kuukautta. Tämä koostuu versionhallinnan tuomasta työtuntien säästöstä, kun kaikki dokumentit löytyvät yhdestä paikasta ja eri versioiden väliset sekaannukset eivät aiheuta taloudellisia menetyksiä. Heckmannin julkaisun mukaan dokumenttien versionhallinnan hankinnan yhteydessä on tärkeää listata halutut ominaisuudet ja suunnitella versionhallinnan käyttö jo etukäteen.

Versionhallintaan sijoitetun pääoman tuottoastetta käsitellään Bendix:n ja Borraccio:n [8] julkaisussa. Julkaisussa versionhallinnan kustannukset ja hyödyt sekä haitat jaetaan mitattaviin, osittain mitattaviin ja ei mitattavaviin kustannuksiin sekä hyötyihin. Julkaisun mukaan versionhallintajärjestelmän ROI-arvon laskennan ongelmana on saatujen hyötyjen arvottaminen johtuen ei-mitattavien hyötyjen suuresta määrästä. Kustannukset ovat pääosin laskettavissa, mutta palautusarvon laskemiseksi pitää pystyä muodostamaan numeerisia arvoja hyödyille. Heidän mukaansa: “We have not stated an explicit formula to calculate the ROI in this paper, but have left it at the model showing the parameters. Such a formula can be made, but it becomes very complex and is probably of little use. What complicates matters the most is that some costs and benefits are one-time (like buying the tool) or once a year (like licenses), whereas others are daily (like the support for parallel work). Yet others are not linear (low benefit until you get to know the tool/process). Therefore it is difficult to make a precise calculation that takes into account all these factors.” ROI-arvon laskentaan vaikuttaa useat parametrit, joten laskentamallista muodostuu monimutkainen eikä sitä tulla käyttämään useimmissa tapauksissa.

Donald Reifer [9] käsittelee julkaisussaan 14 ohjelmistoprojektin hallintaperiaatetta.

Nämä periaatteet jakaantuvat viiteen osa-alueeseen: suunnittelu, organisointi, henkilöstö, johtajuus ja hallinta. Hänen mukaansa projektia johtavien henkilöiden tulisi pyrkiä tekemään strategisia suunnitelmia lyhyen ajan taktiikoiden sijaan. Vaikka projektin aikataulut ja budjetointi eivät välttämättä suosi strategista suunnittelua, tämä edesauttaa projektin loppuun saamisessa. Projektin ongelmat pyritään ratkaisemaan mahdollisimman aikaisessa vaiheessa ja suunnittelu huomioi myös ohjelmiston myöhemmät käyttövaiheet sekä organisaation muut toimintatavat.

Versionhallintajärjestelmien historiaa, käyttötarkoitusta ja erityyppisiä muutoksenhallintamenetelmiä käsitellään Ali Koc ja Abdullah Uz Tansel:n artikkelissa “A Survey of Version Control Systems” [10], jossa käsitellään myös matemaattisesti eri rakennetyypeillä olevien versiohallinta järjestelmien tilantarvetta ja rakennetta. He mainitsevat kolme eri tyyppistä muutoksen hallintaa “snapshot” (tallennettu tilannekuva), “forward delta” (muutosten seuraaminen lähtötilanteesta eteenpäin) ja “backward delta” (muutosten seuraaminen nykytilasta taaksepäin), joita voidaan käyttää versionhallinnan pohjalla. Artikkelisi esittelee myös versionhallintajärjestelmien kehitystä ja taustoja.

## 2.2 Versionhallintaan liittyvä lainsäädäntö, velvoitteet ja standartit

Ohjelmoitavien komponenttien versionhallintaa koskevaa lainsäädäntöä suoranaisesti ole. Velvoitteita käsitellään esimerkiksi epäsuorasti YVL ohjeessa A.8 [4], jossa on maininta: “Luvanhaltijan on tallennettava ja ylläpidettävä laitosisien ikääntymisen hallinnassa tarvittavia viitetietoja. Tällaisia ovat suunnitteluasiakirjat (suunnitteluperusteet, tekniset määrittelyt, rakennemateriaalit, piirustukset, toimintakuvaukset) sekä kelpoistustiedot, valmistuksen tulosaineistot ja muutostyöaineistot.” Ohjelmoitavien komponenttien ohjelmistoa voidaan pitää tässä ohjeessa mainittuna viitetietona. IAEA turvastandarti SSG-39 [5] ottaa kantaa miten nykyaikaisten ydinlaitosten ohjelmistojen versioita tulee hallita osana ydinlaitosten eliniän hallintaa. Viestintävirasto on julkaissut ohjeen lokitietojen keräyksestä ja käytöstä [11]. Versionhallinnan yksi osa-alue on seurantatietojen keruu, jotta sisällön muuttumista voidaan seurata. Näissä velvoitteissa ja ohjeissa ei oteta kantaa miten versionhallinta tulee toteuttaa, joten lopullinen ratkaisu on loppukäyttäjältä riippuvainen.

Versionhallintaa käsitellään standartissa ISO 10007 [7], joka käsittelee laadunhallintaa konfiguraatiohallinnan näkökannasta. Kyseisen standartissa on kuvattu versionhallinnan vaatimuksia ja velvoitteita. Standartissa määritellään myös miten versionhallinnalla voidaan toteuttaa ISO 9001 [6] Laadunhallintajärjestelmiä vaatimuksia käsittelevän standartin kohta 8.5.2. Kohdassa 8.5.2 “Tunnistettavuus ja jäljitettävyys” on mainittu: “os tuotteen edellytetään olevan jäljitettävissä, organisaation

on hallittava yksittäisten tuotosten tunnistettavuutta ja säilytettävä jäljitettävyyden mahdollistavaa dokumentoitua tietoa.”

Muita versionhallintaan osittain viittavia standarteja ovat esimerkiksi ISO/IEC TR24748-2 [12], IEC 61508-3 [13] ja IEC 61131-1 [14] . Nämä standartit eivät suoraan ota kantaa versionhallinnan tehtäviin eikä käyttöön mutta asettavat velvoitteita koskien tuotteiden riskienhallintaa, ohjelmistojen elinkaareen, ohjelmoitavien elektronisten laitteiden turvallisuuteen, määrittelyyn ja tunnistukseen.

### 3. VERSIONHALLINTAJÄRJESTELMÄN MÄÄRITELMÄ JA MÄÄRITTELY

ISO 10007 [7] mukaan konfiguraatiolla tarkoitetaan tuotteen toiminnallisia ja fyysisiä ominaisuuksia, jotka liittyvät toisiinsa. Nämä määritellään ja kuvataan konfiguraatitietojen avulla. Konfiguraatitiedot puolestaan kuvaavat vaatimukset tuotteen suunnittelulle, toteutukselle ja verifiointille sekä mahdollistavat sen käytön ja ylläpidon koko elinkaarensa ajan. U.S Department of energy määrittää julkaisun SQAS20.01.00 - 2000 [15] luvussa 1: "Configuration Management (CM) is the process of identifying and defining the configuration items in a system, controlling the release and change of these items throughout the system lifecycle, recording and reporting the status of configuration items and change requests, and verifying the completeness and correctness of configuration items." Vapaasti suomennettuna: "Konfiguraationhallinta on prosessi, joka tunnistaa ja määrittää järjestelmän eri osien konfiguraation sekä hallinnoi näiden osien julkaisua ja muutosta koko järjestelmän eliniän ajan. Konfiguraationhallinta ylläpitää tietoa osien tilasta ja muutospyyntöistä sekä varmistaa konfiguraation yhteneväisyyden ja oikeellisuuden."

Versionhallintajärjestelmän määrittelyssä on oleellista määrittää, mitä tietoa tulee versionhallinnan alle ja miten järjestelmä integroituu osaksi teollisuusympäristöä ja aikaisempia tietojärjestelmiä. Aluksi on hyvä päättää, mille tasolle ohjelmistojen versionhallinta halutaan viedä ja kuinka tarkasti versionhallintajärjestelmän määrittely on hyödyllistä tehdä. Tämä on kompromissi käytetyn ajan ja rahan sekä halutun tuloksen välillä. Usein on mahdollista käyttää hyväksi aikaisempien projektien määrittelyä, mutta tässä pitää ottaa huomioon teknologian kehittyminen ja mahdollisesti muuttuneet standartit ja vaatimukset. Lisäksi määrittelyn kohteena olevan projektin käyttökohde tulee ottaa huomioon, vaikkakin versionhallintajärjestelmien käyttötarkoitus on usein sama.

Määrittelyn tarkoitus on muodostaa käsitys, mitä vaatimuksia versionhallinnan tulee täyttää ja mitä tehtäviä versionhallinnan on pystyttävä tekemään. Lisäksi määrittelydokumentaatio saattaa sisältää vaatimuksia ohjelmiston toimittajalle ja ohjelmiston eliniälle sekä tuotetuen vasteajalle. Esimerkiksi käytettävyyssaste saattaa olla yksi määrittelyn osa. Määrittelydokumentaatio saattaa sisältää useita osioita ja projektin eri vaiheille voidaan laatia omat määrittelydokumentit, jotta vaatimusten

täyttymistä on helpompi seurata ja seuraavan vaiheen vaatimuksia voidaan tehdä samalla aikaa.

### 3.1 Tarvittavien lähtötietojen määrittely

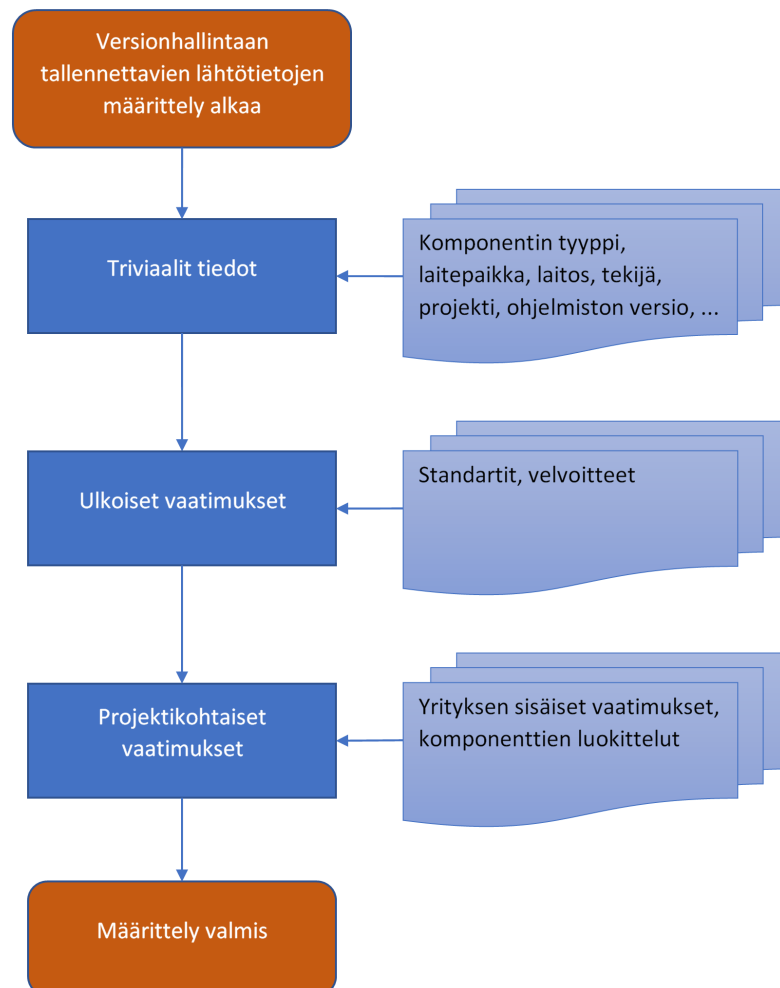
Lähtötietojen määrittelyn tulisi vastata kysymykseen: “Mitä kaikkea alkutietoja tulee tietää siirrettävästä ohjelmistosta ennen sen siirtoa versionhallintaan?” Osa näistä tiedoista voi olla hyvinkin triviaaleja, osa tulee mahdollisesti vaatimuksina käytössä olevista standarteista ja loput ovat projektikohtaisia tietoja. Lähtötietojen määrittäminen ei välttämättä ole mahdollista tehdä ennen versionhallintajärjestelmän valintaa, koska tallennettavien tietojen metatiedon tallennusmahdollisuudet poikkeavat toisistaan. Tämä tulee ottaa huomioon myös versionhallintajärjestelmän määrittelyn yhteydessä.

Tyypillisiä triviaaleja tietoja on esimerkiksi ohjelmoitavan komponentin tyyppi, laitepaikka, tekijä, laitos, projekti, ohjelmiston versio, alkuperäinen tekijä ja viimeisin muokkaaja sekä ohjelmiston tilatieto. Useimmat versionhallintajärjestelmät ylläpitävät automaattisesti useita näistä tiedoista, joten niiden syöttäminen järjestelmään on työnä kertaluontoista ja tehdään usein ohjelmistojen siirron yhteydessä. Nämä tiedot ovat oleellisia versionhallintajärjestelmän käytön kannalta, ja niiden määrittelyyn tulee suhtautua siihen kuuluvalla vakavuudella. Lähtötietoja on yleensä mahdollista muuttaa jälkikäteen, mutta se hankaloittaa ohjelmistojen hakua ja käyttöä loppukäyttäjän kannalta. Tämä hankaluus saattaa aiheuttaa vastarintaa, joka ei johdu varsinaisesti versionhallintajärjestelmän huonoudesta vaan sinne syötetyn tiedon vajavaisuudesta.

Esimerkiksi IEC 61508-1 [16] määrittää kohdassa 5.2.6, että “Dokumentaation on oltava tarkkaa ja suppeaa. Helposti ymmärrettävää niille henkilöille, joiden pitää hyödyntää sitä. Sopivaa siihen tarkoitukseen, johon se on aiottu sekä helposti käsille saatavissa ja ylläpidettävissä”. Ja mainitsee liitteessä A.4: “Dokumenttien luetteloon sisältyy tyypillisesti seuraava informaatio: piirustuksen tai dokumentin numero, revisiohakemisto, dokumentin nimikoodi, otsikko, muutoksen päivämäärä ja tietoväline”. Nämä määrittäykset kohdistuvat ohjelmoitavien elektronisten turvallisuuteen liittyvien järjestelmien toiminnalliseen turvallisuuteen, jotka ovat yksi mahdollinen käyttökohde ohjelmistojen versionhallinnalle. Tyypillinen versionhallintajärjestelmä luo automaattisesti aikaleimat ulos- ja sisäänkirjauksen yhteydessä, aikaleimoja voidaan seurata jälkikäteen lokikirjan avulla.

EN 82045-2 [17] käsittelee dokumenttien hallinnan metaelementtejä ja informaation viitemalleja. Tästä standartista on mahdollista hakea viitemalleja ohjelmistojen metatietojen määrittämiseen, mutta dokumentti on suunnattu hallintajärjestelmien

loppukäyttäjille ja tarkoitettu dokumenttien hallinnan perusapuvälineeksi. Standartia on mahdollista käyttää apuna lähtötietojen määrittelyssä, mutta oman arvioni mukaan dokumentti on enemmänkin avustava kuin tarkkaa määrittelyä sisältävä. Projektikohtaisia tietoja on esimerkiksi järjestelmien ja niihin liittyvien komponenttien jaottelu turvallisuus- ja tärkeysluokkiin. Esimerkiksi IEC 61140 [18] määrittää sähköisten laitteiden suojausluokitukset. Muita esimerkkejä jaottelun perusteena käytettävistä turvallisuuden liittyvistä luokitteluista ovat IEC 62061 [19] mainittu SIL-luokitus, STUK:n ohjeen YVL B.2 [20] kohdan 3.2 “Turvallisuustoimintoihin liittyvät luokitusperusteet” ja ISO 13849-1 [21] mainittu PL-luokitus. Tärkeysluokittelun pohjana voidaan käyttää esimerkiksi varaosien saatavuutta, tuotantomenetyksiä, vaikutusta hankintaketjuun ja komponentin merkitystä koko järjestelmän toimivuuden kannalta.



*Kuva 1* Versionhallintaan tallennettavien lähtötietojen määrittely.

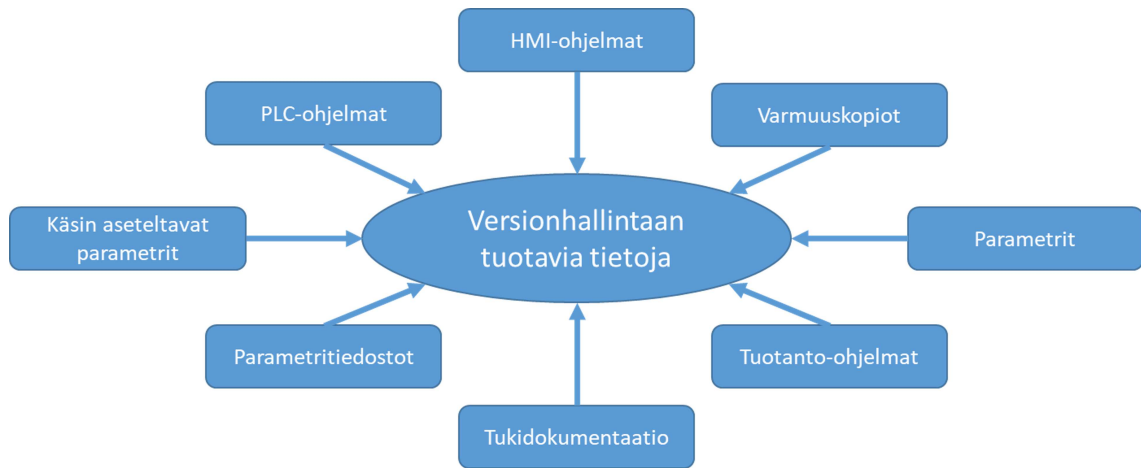
## 3.2 Versionhallinnassa olevien tietojen määrittely

Tietojärjestelmä on juuri niin hyvä kuin sen sisältämä tieto. Ohjelmoitavien komponenttien versionhallintajärjestelmän sisältöön kuuluu perinteisen dokumentaationhallinnan lisäksi myös komponenttien ohjelmat. SQAS20.01.00 - 2000 [15] julkaisussa määritetään ohjelmistoihin liittyvä dokumentaatio osaksi ohjelmistoa. Dokumenttien vienti osaksi ohjelmistojen versionhallintaa riippuu siitä, miten yritys on määrittänyt dokumenttien hallinnan. Osassa tapauksista on mielekästä viedä dokumentit osaksi ohjelmistojen versiohallintajärjestelmää, tapauksesta riippuen voidaan käyttää erillistä dokumenttien versionhallintaa ohjelmistojen versionhallinnan rinnalla. Toteutukseen ei suoraan oteta kantaa esimerkiksi standardissa IEC 82045-1 [22], joten toteutukset ovat projektikohtaisia.

Versionhallinnan olla olevia ohjelmoitavien komponenttien ohjelmistot ovat esimerkiksi PLC projekteja, käyttöliittymien asettelu- ja sisältötiedostoja, taajuusmuuntajien parametrien asettelutiedot, ohjelmoitavien antureiden ohjelmistot, ohjausjärjestelmien projektit ja image-kopiot, reittimien asetustiedostot ja muuntimien asettelutiedot. Tiedostotyyppiä tai sisältöä ei varsinaisesti ole määritetty, IEC 61508-1 [16] määrittää kohdassa 5.2.6 dokumentaation sisällön, jota voidaan soveltaa tässä kohdassa.

Jotta versionhallinnan käyttö olisi mahdollisimman kannattavaa tulisi tiedostotyypit valita siten, että niiden hyödyntäminen olisi mahdollisimman helppoa ja tehokasta. Tiedostotyyppien käyttöä saattaa osaksi ohjata valitun versionhallintajärjestelmän tuetut tiedostotyypit on asia, johon tulisi ottaa kantaa versionhallintajärjestelmää valittaessa. Ohjelmoitavien komponenttien ohjelmointiympäristöt voidaan tällä hetkellä jakaa kahteen luokkaan, IEC 61131-3 [23] pohjaiseen CODESYS-ympäristöihin ja valmistajien itse määrittelemiін ohjelmointiympäristöihin. Versionhallinnan tuemat tiedostotyypit vaihtelevat suuresti, tosin yleisemmät tiedostotyypit ovat yleensä tuettuina. Näitä on esimerkiksi, ASCII-teksti, pdf, csv, excel, XML ja binäärimuotoiset tiedostot. Tuetulle tiedostotyyppille pystytään yleensä tekemään vertailu suoraan versionhallintajärjestelmän sisällä.

Versionhallinnassa olevat tiedot voivat esimerkiksi sisältää IEC 62708 [24] mukaiset instrumenttidatalehti, ihmisen ja koneen välisen käyttöliittymän määrittelyn ja toimintojen lohko-kaavion. Käytännössä tiedot ovat PLC-ohjelmia, ylemmän tason automaatio-ohjelmia, käyttöliittymien projektitiedostoja, parametrisoitavien laitteiden parametritiedostoja ja asetteluarvoja käsin aseteltavien laitteiden osalta. Sekä oleellinen osa versionhallintaa on varmuuskopioiden hallinta, jokaisesta järjestelmästä tulisi olla täydellinen varmuuskopio, jotta vikaantunut laite voidaan vaihtaa uuteen ja ohjelmisto palauttaa takaisin viimeiseen tunnettuun tilaan.



*Kuva 2 Esimerkkejä versionhallintaan vietävistä tiedoista.*

### 3.3 Tallennettävien tietojen vastaanotto

Tiedostojen vastaanotossa on oleellista määritellä vastaanottoprosessin eri vaiheet. Vaihteita voi olla esimerkiksi median oikeellisuuden tarkistus, tiedostojen oikeellisuuden tarkistus, komponenttien kohdistuksen tarkistus ja sisällön oikeellisuuden tarkistus.

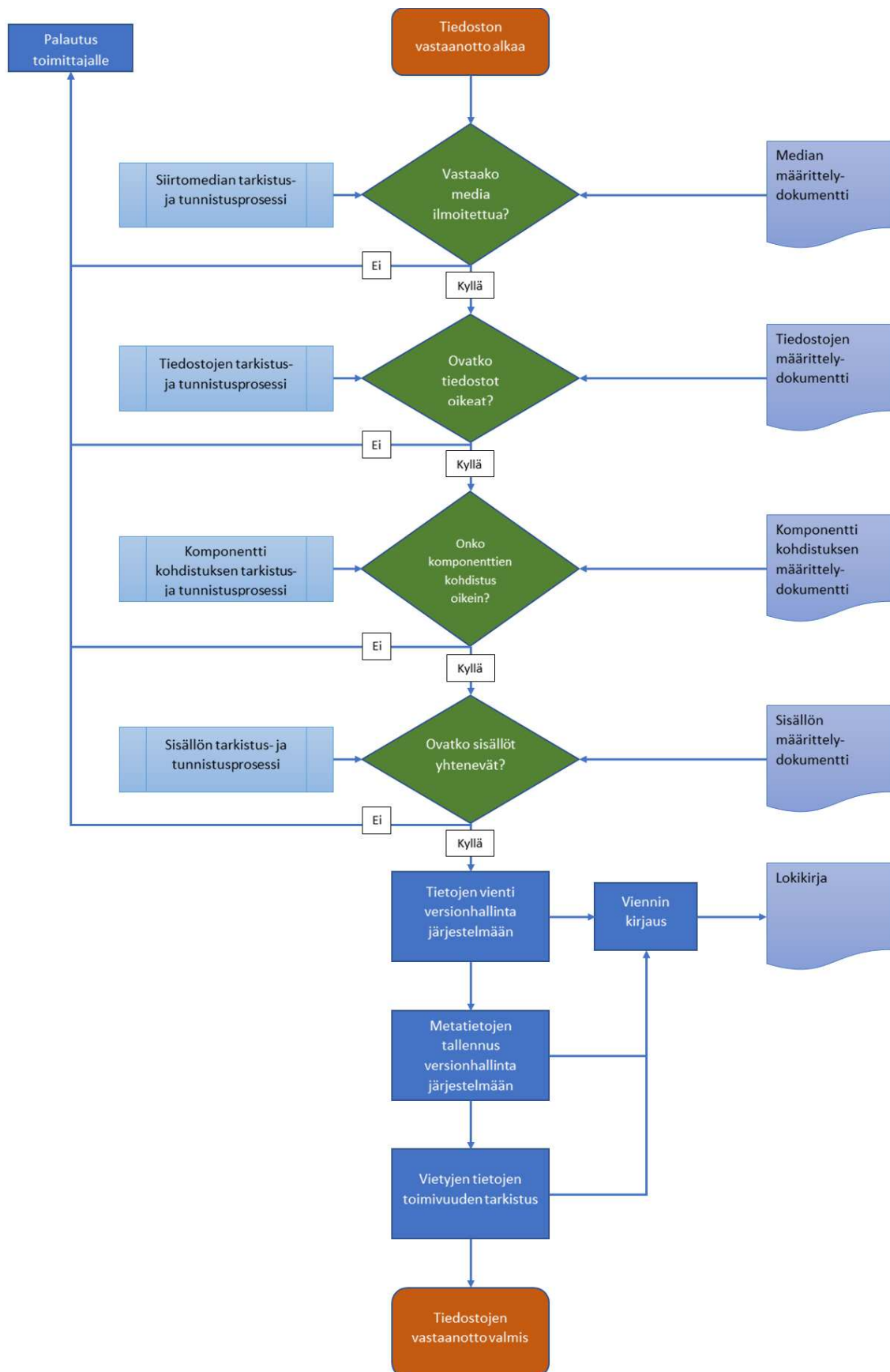
Komponenttien kohdistuksen oikeellisuudessa tarkistuksen tarkoituksena on tallennettävien tietojen kohdistus oikeisiin fyysisiin komponentteihin, jotta tieto ja fyysisen komponentti täsmäävät toisiinsa.

Viimeisenä osaprosessina on sisällön oikeellisuuden tarkistus, jossa varmistetaan että tiedot ovat yhtenevät oikean komponentin tiedon kanssa. Näin varmistetaan tietojen ajantasaisuus ja käyttökelpoisuus.

Prosessin kolme viimeistä vaihetta käsittelevät tietojen vieniä versionhallintajärjestelmään, tietoihin liittyvien metatietojen tallennusta ja toimivuuden varmistamista. Näiden toimintojen jälkeen vastaanotto on valmis ja tiedot ovat tämän prosessin osalta käsitelty.

Näiden toimenpiteiden jälkeen tiedot vietään versionhallintajärjestelmään ja kirjataan vieni lokikirjaan. Viimeisinä vaiheina on metatietojen tallennus versionhallintajärjestelmään ja tallennetun tiedon toimivuuden tarkistus. Kuvassa 3 on esitetty edellä mainittujen vaiheiden pohjalta tietojen vastaanoton vuokaavio, jossa oletuksena on, että tiedot ovat valmiina ja tuodaan suoraan versionhallinnan alle.

Tietojen vastaanottoprosessi on yritys- ja projektikohtaista, joten on myös täysin mahdollista että tallennettavat tiedot luodaan suoraan versionhallinnan alle ja ainoastaan muokkaus tehdään erillisessä ohjelmointiympäristössä. Tällöin on hyvä määrittää uuden versionluonnin pelisäännöt, jotta jokainen pienikin muutos ei luo uutta versiosta ohjelmasta, ellei näin haluta tarkoituksella toimia.



*Kuva 3 Tietojen tuonti versionhallintaan.*

## 4. VERSIONHALLINTAJÄRJESTELMÄN KÄYTTÖÖNOTTO

Outi Aronen [2] käsittelee diplomityössään tietojärjestelmän käyttöönottoa ja sen arviointia. Hän käsittelee kattavasti ohjelmiston käyttöönottoprosessia ja sen vaiheita. Uuden tietojärjestelmän käyttöönoton määrittely on vahvasti riippuvainen näkökannasta. Teknisesti käsiteltynä se on uuden järjestelmän asentamista, yksilöimistä parametrisoinnin avulla ja mahdollista tietojen siirtoa vanhasta järjestelmästä. Toinen näkökanta on, että tietokonejärjestelmä ja toiminta yhdistetään siten, että tavoitetaan hankinnanperusteena olevat tulokset. Käyttöönoton aikana tapahtuu siirtyminen määrittelystä käyttöön, jolloin aikaisemmin tehty määrittely on pohjatietona.

Käyttöönoton aikana järjestelmästä etsitään mahdollisia ohjelmointivirheitä ja varmistetaan, että järjestelmä on määrittelyn mukainen. Samalla testataan verkkoratkasuiden toimivuutta ja luodaan käyttäjäryhmät. Hyvin tehty käyttöönotto on perusta myöhemmälle käytölle. Käyttöönoton aikana uusi järjestelmä saattaa kohdata vastarintaa varsinkin, jos uusi järjestelmä korvaa vanhan järjestelmän. Uuden järjestelmän koulutus käyttäjille ja implementointi yrityksen nykyisiin toimintamalleihin vie aikaa, jonka jälkeen vasta on mahdollista saavuttaa kaikki hyöty järjestelmästä. Käyttäjien riittävä huomioonottaminen ja koulutus, sekä ohjelmiston toimittajan tekninen tuki ovat ensisijaisen tärkeitä.

Arosen ja Walfordin [25] mukaan yleensä paras menetelmä järjestelmän käyttöönottoon on ajallinen porrastus. Turbanin, McLeanin ja Wetherben [26] mukaan useimmiten on kannattavaa käyttää pilottiryhmää, jotta havaitaan virheet ja muutostarpeet ennen järjestelmän kokonaisvaltaista käyttöä. Tämän pilottiryhmän tulisi sisältää laaja otanta eri loppukäyttäjiryhmiä. Käytäntö on kuitenkin osoittanut, että jokainen käyttäjäryhmä näkee asiat omasta perspektiivistä ja keskittyvät useimmiten heille tärkeisiin asioihin. Oleellista on osoittaa käyttäjille versionhallinnan tuoma lisäarvo omaan työhön ja järjestelmän tuomat edut esimerkiksi konfiguraationhallinnan osalta.

Pitkittynyt käyttöönotto vähentää käyttäjien mielenkiintoa ohjelmistoa kohtaan, joten projektin tulee pysyä aikataulussa. Käyttöönoton aikana tulleet ongelmat ja niiden ratkaisut tulee kerätä ylös mahdollista myöhempää tarvetta varten. Samalla

voidaan tehdä huomioita järjestelmän toimivuudesta ja toimittajan palveluntasosta. Pitää kuitenkin ottaa huomioon, että jokaisella projektilla varsinkin erityisemmilla aloilla toimivilla yrityksillä saattaa olla hyvinkin spesifisiä vaatimuksia, joten asiakkaiden kaikkiin vaatimuksiin vastaaminen saattaa olla hankalaa.

## 4.1 Vastaanotettujen tietojen vienti versionhallinnan alle

Tietojen vastaanotossa on oleellista, että vastaanotettu tieto täsmää määritellyn dokumentaation ja todellisen tilanteen kanssa. Toisin sanoen tietojen on oltava yhteneviä versionhallintajärjestelmässä ja tehtaassa. Riittävän tarkka etukäteismäärittely ja tietojen tallentajan riittävä ammattitaito ja ohjeistus on ensiarvoisen tärkeää. Näiden lisäksi on oleellista, että tiedostoja järjestelmään tallentava henkilö tuntee versionhallintajärjestelmäksi valitun ohjelmiston ja yrityksen toimintatavat. Nämä edellytykset täyttämällä on mahdollista saavuttaa tehokas ja tarkka prosessi, joka täyttää annetut vaatimukset ja tuottaa halutun lopputuloksen.

Tietojen vientiin liittyen on tärkeää määritellä tietorakenne, jonka määrittelystä kerrotaan enemmän seuraavassa kappaleessa. Muita oleellisia asioita on haluttujen komponenttityyppien määrittely ja niiden noudattaminen sekä riittävä ohjeistus viennin toteuttamisesta. Vienti on joissain järjestelmissä mahdollista suorittaa massa-ajona, jolloin siirretään suuri määrä tietoa kerralla. Tällöin menetetään mahdollisuus tarkistaa tietojen oikeellisuus viennin yhteydessä.

Vienti koostuu vietävien tietojen tunnistamisesta, tallennuskohteen tunnistamisesta, komponenttityypin määrittelystä (jos tarpeellista), tiedostojen viennistä versionhallintajärjestelmään ja tarvittavien metatietojen kirjaamisesta viennin jälkeen. Oleellista on myös tarkistaa viedyn tiedon oikeellisuus viennin jälkeen ja riittävä kirjaaminen, jotta saavutetaan jäljettävyys jo viennistä alkaen.

## 4.2 Versionhallinnanjärjestelmän tietorakenteen määrittely

Versionhallinnan toimivuus, tietojen käytettävyys, riippuu vahvasti tietorakenteen toimivuudesta. Tästä syystä ennen tietojen vientiä järjestelmään on oleellista määrittää tietorakenne. Useimmiten yrityksellä on jo käytössä laite- ja komponenttikohdainen kohdistusjärjestelmä, jonka pohjalta tietojärjestelmä voidaan rakentaa. Versionhallintajärjestelmän kannalta komponentit tulee yksilöidä, jotta tieto osataan kohdistaa oikeaan komponenttiin. Eri teollisuuden aloilla on käytössä omat komponenttien nimeämiskäytäntönsä.

Esimerkiksi voimalaitosteollisuus käyttää laajalti saksalaista KKS-järjestelmää, joka jakaa järjestelmän pääprosessipiirien mukaan. KKS järjestelmää on käsitelty esimerkiksi Samu Syväsen opinnäytetyössä "KKS-koodausjärjestelmän soveltaminen

Alfa Laval Aalborg Oy:n PI-kaavioihin” [27] kappaleessa 3. Muita mahdollisia nimeämiskäytäntöjä on ISO 4157-1, -2 ja -3 [28][29][30] mukainen huonekäytäntö, joka voidaan yhdistää IEC 81346-1 [31] mukaiseen komponenttien nimeämiseen.

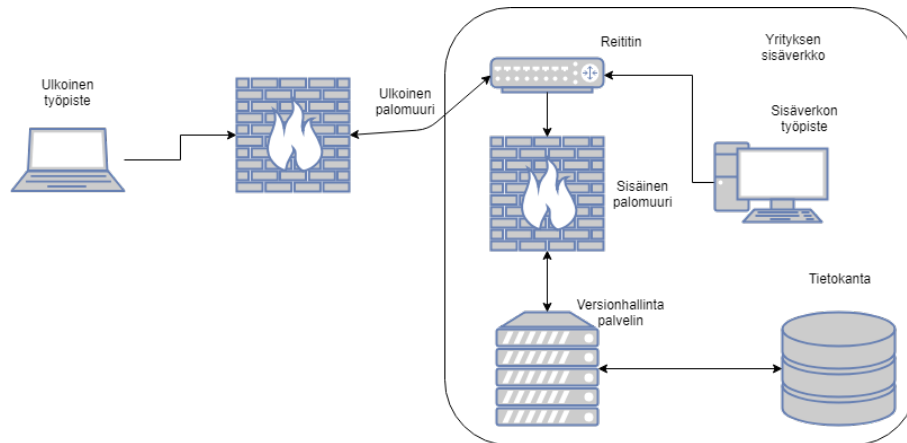
Tietorakenteen määrittely voidaan tehdä pilottikäytön jälkeen, mutta tulee määrittää ennen tietojen siirtämistä versionhallintajärjestelmään. Tietorakenteen määrittelyssä tulee ottaa kantaa laitteiden ja komponenttien luokitteluun sekä komponentin sisällä olevan datan tietorakenteeseen. Määrittelyn laajuuden tarve riippuu versionhallintajärjestelmän alle tulevan tiedon määrästä.

Tietorakenteeseen vaikuttaa myös käytettävän ohjelmiston tapa käsitellä tietorakennetta ja eri versioita. Varsinkin tietojen palauttamisen kannalta on oleellista tarkistaa miten versionhallintaohjelmisto käsittelee eri versioiden välisiä muutoksia. Palauttamisen kannalta oleellista on että jokainen versio voidaan palauttaa erillisenä selkokieleisenä tietona. Vaikkakin tämän tyyppisessä hallinnassa on heikkoutena suuri tilan tarve, kuten Koc ja Tansel [10] esittävät artikkelissaan.

### 4.3 Versionhallintaan liittyvät verkkoratkaisut

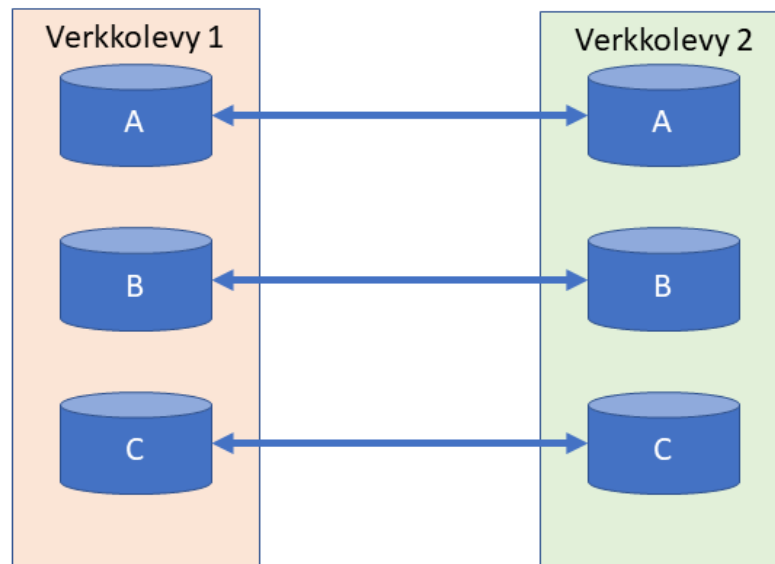
Versionhallintaan liittyvä verkkoratkaisu riippuu vahvasti yrityksen omista verkkoratkaisuista. Versionhallintajärjestelmän tiedot voidaan käsitellä keskitetysti palvelimella tai käyttäjien koneella. Jos tiedot säilytetään ja käsitellään keskitetysti yhdellä palvelimella, on mahdollista toteuttaa tietoturva ja varmuuskopionti keskitetysti, mutta tässä ratkaisussa kuormitetaan tiedonsiirtoverkkoa ja toiminta on vahvasti riippuvainen verkkoyhteyksien toiminnasta. Tämän tyyppinen ratkaisu sopii hyvin yrityksen sisäverkossa toimivaan versionhallintajärjestelmään, jossa tarve ulkopuolisiin yhteyksiin on vähäinen tai ei ole käytössä lainkaan. Käyttäjien koneella tehtävä tietojen käsittely ja tallennus sopii yrityksille ja projekteille, joissa verkkoyhteydet ovat heikot tai ei saatavilla. Tyypillinen esimerkki on asiakkaan luona tehtävä ohjelmointityö, jossa tehdyt ohjelmat ja muutokset synkronoidaan palvelimelle kun verkkoyhteys on saatavilla.

Versionhallintaan liittyy vahvasti myös tietoturva ja turvatut tiedonsiirtoyhteydet. Tietoturvan toteuttaminen on vahvasti riippuvainen yrityksen omasta osaamisesta ja tietoturvakäytännöistä. Luvussa 2.2 käsitellyt standardit antavat velvoitteita tiedon turvaamisesta ja seurannasta, mutta eivät suoraan käsittele tietoturvan tasoa. Kuvassa 4 on esitetty yksi esimerkki miten tietoturva voidaan toteuttaa tilanteessa, jossa versionhallintajärjestelmän tiedot ovat palvelimella.



**Kuva 4** Esimerkki verkkoratkaisusta ja tietoturvasta.

Varmuuskopiointi on oleellinen osa versionhallintaa, jotta järjestelmään tallennetut tiedot eivät häviä esimerkiksi kiintolevyn hajoamisen yhteydessä. Yksi mahdollinen vaihtoehto on RAID 1 järjestely, joka on esitetty kuvassa 5. Tässä tapauksessa kaikki tiedot ovat täysin redundanttisia, eikä toisen tallennusmedian rikoontuminen aiheuta tietojen menetystä. Muita mahdollisia varmuuskopiointi menetelmiä on virtuaalipalvelinten luonti ja monistus, sekä varmuuskopion ottaminen fyysiselle medialle. Fyysinen media voidaan arkistoida yrityksen käytäntöjen mukaan määritettyyn paikkaan, jolloin saadaan myös fyysinen varmuus esimerkiksi tulipalojen varalta.



**Kuva 5** Esimerkki RAID 1 kokoonpanosta.

## 4.4 Käyttöönoton yhteydessä mahdollisesti esiintyviä ongelmia

Ongelmia esiintyy pääosin käyttöönoton alkuvaiheessa. Ongelmat voivat johtua esimerkiksi puutteellisista määrittelyistä, käyttäjien erilaisesta näkökannasta, puutteellisista ohjeistuksesta ja valitun ohjelmiston ongelmista. Tämän lisäksi tulevat projekteille tyypilliset ongelmat, joita ovat budjetissa ja aikataulussa pysyminen. Ongelmia voidaan vähentää riittävällä määrittelyllä, joissa pyritään huomioimaan kaikki näkökannat riittävän laajasti. Jos muiden ohjelmistoa käyttävien yritysten auditointi on mahdollista, niin se tulisi tehdä. Vähintäänkin toimittajan auditointi ja referenssien läpikäynti on suotavaa.

Puutteellisen tai väärän määrittelyn tyypillisesti aiheuttama ongelma on ohjelmiston epäsopivuus käyttötarkoitukseen. Myös saavutettu hyöty saattaa jäädä alunperin määritettyä pienemmäksi. Samoin nykyisten tietojärjestelmien integrointi osaksi käyttöönnettävää versionhallintajärjestelmää saattaa osoittautua mahdottomaksi. Määrittelyyn käytettävä pohjatieto voi muuttua kesken projektin. Tämä voi johtua yrityksen muuttuneista käytännöistä tai standardimuutoksista. Näiden muutosten ennaltahuomioonottaminen on yleensä vaikeaa, mutta määrittelyprojektin aikaisella yhteistyöllä muiden yrityksen tahojen kanssa voidaan minimoida muutosten aiheuttamat ongelmat.

Käyttäjien erilaiseen näkökantaan ei juurikaan voida vaikuttaa. Kuten Aronen [2] toteaa diplomityössään, on uuden tietojärjestelmän lähtötaso aina aikaisempaa tietojärjestelmää matalampi käyttäjän näkökannasta. Käyttäjien riittävällä huomioonotamisella ja opastuksella voidaan madaltaa lähtökynnystä ja vähentää muutosvastarinnan syntyä. Käyttäjään vaikuttaa myös yrityksen edelliset tietojärjestelmien käyttöönotoista jääneet kokemukset, joihin nykyinen projekti ei voi vaikuttaa.

Ohjeistukseen liittyvät ongelmat voivat johtua määrittelyn tai ohjeenkirjoittajan ja ohjeenkäyttäjän erilaisista näkökulmista. Ohjeistuksen ongelmat eivät yleensä ole peruuttamattomia mutta saattavat aiheuttaa ylimääräistä työtä. Tyypillisiä ohjeistukseen liittyviä ongelmia ovat käyttöönottoprosessiin liittyvien prosessien vaillinaisen kuvaaminen, ohjeistuksen terminologian väärä käyttö, epäselvät ohjeet ja ongelmatilanteiden ratkaisuun tarvittavien ohjeiden puuttuminen.

Ohjelmistoon liittyvät ongelmat saattavat tulla esille vasta käyttöönoton yhteydessä johtuen todellisen verkkoympäristön käytöstä ja todellisen tiedon eroavaisuudesta testiympäristöön verrattuna. Testiympäristössä tietoturvan ja käyttäjäoikeuksien rajoitukset eivät, parhaan kokemuksen saamiseksi, ole yleensä yhtä rajoitetut kuin varsinaisessa versionhallinnan työympäristössä. Muita ohjelmistoon liittyviä ongelmia saattaa tulla vastaan testilisenssien vaihdossa käyttölisenssiin, joilloin ohjelmiston ominaisuudet saattavat poiketa testatusta. Tyypillisiä ongelmia on verkkoympäristön rajoitukset, mistä johtuen versionhallintaan käytettävän verkkoym-

päristön resurssit ovat alhaisemmat kuin alunperin oli suunniteltu. Tästä johtuen käyttäjien kokema suorituskkyky on testiympäristöä heikompi ja saattaa korostaa käyttäjän kokemia negatiivisia kokemuksia versionhallintaohjelmistosta.

Versionhallinnan käyttöönnotossa esiintyy myös perinteiset projektien ongelmat. Näitä voivat olla budjetin ylittyminen, toimittajan kanssa tulevat erimielisyydet, kesken projektin tulevat henkilöstövaihdokset sekä yrityksen sisäiset muutokset. Ohjelmistojen hallintaan liittyviä ongelmia on mainittu ISO 21503 [32] standardissa, jonka mukaan hankaluuksia voivat aiheuttaa esimerkiksi saavutettavien tavoitteiden epävarmuus tai epäselvyys, teknologiset näkökannat, logistiset ongelmat ja sidosryhmien erilaiset näkökannat ja odotukset.

## 5. VERSIONHALLINNAN TEHTÄVÄT

Versionhallinnan pääasiallinen tehtävä on ylläpitää tietokantaa tallennettavasta tiedosta. Näin voidaan varmistua, että käytettävä tieto on ajantasaista, saatavilla ja muutokset ovat jäljitettävissä. Versionhallinnan tyypillisiä tehtäviä teollisuudessa on toimia ohjelmoitavien komponenttien ohjelmistojen tallennuspaikkana, jolloin ohjelmistoihin voidaan tehdä muutoksia keskitetysti ja voidaan hallita kentällä olevia versioita. Samalla ylläpidetään ajantasaista tietokantaa mitä laitteita on käytössä ja mitkä ohjelmistot niihin on asennettu.

Versionhallinnan tehtävät riippuvat ohjelmiston määrittelystä, jolloin pelkän versionhallinnan rinnalla tehtävinä voi olla varmuuskopiointi, saatavuus eri toimiaosa-  
puolien välillä tai tuotteiden ja toiminnallisuuksien elinkaarenhallinta. Kaikkien näiden tehtävien tarkoitus on tuottaa lisäarvoa versionhallintaohjelmistolle, joilloin voidaan perustella järjestelmään käytetyt resurssit.

Myös laatustandardtien [6] ja [7] sekä vaatimusten, esimerkiksi YVL A.8 [4] täyttäminen on osa versionhallinnan tehtävää, vaikkakin nämä standardit ja vaatimukset voidaan täyttää myös muilla tavoilla. Tehtävien määrittely on tapauskohtaista, ne ovat riippuvaisia ohjelmiston ominaisuuksista sekä projektin määrittelystä. Tehtävien kuvaus voi muuttua projektin aikana ja ohjelmiston elinkaaren aikana johtuen käyttöympäristön tai ulkopuolisten vaatimusten muuttumisesta.

Versionhallinnan tehtävät eivät ole yksiselitteisiä ja parhaan hyödyn saamiseksi tulee tehtäviä käsitellä usealta eri näkökannalta, sekä käyttää määrittelyyn riittävästi resursseja. Näin voidaan välttyä väärän ohjelmiston valinnalta ja välttyään mahdollisilta lisähankinnoilta myöhemmässä vaiheessa. Myös yrityksen sisäinen rakenne ja eritoten tietotekniset ratkaisut ja valmiudet vaikuttavat suuresti tehtävien määrittelyyn.

### 5.1 Integrointi osaksi jo olemassa olevia tietojärjestelmiä

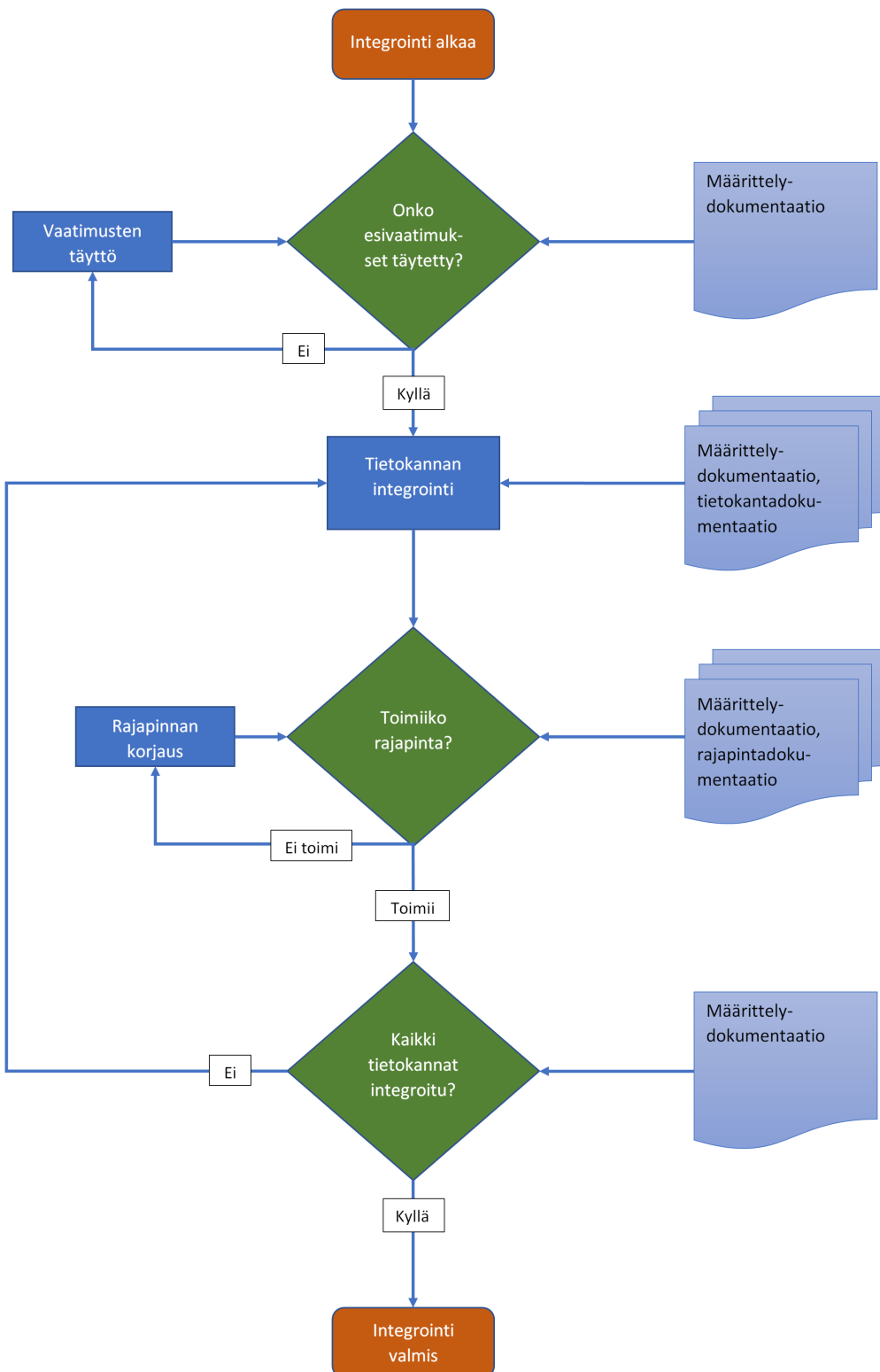
Integroinnin pääasiallinen tarkoitus on yhdistää versionhallinta osaksi yrityksen nykyisiä tietojärjestelmiä. Käytössä olevat tietokannat, versionhallinnan tehtävät osana kokonaisuutta ja liityntärajapinnat on määritelty osana määrittelydokumentaatiota. Integraation laajuus ja vaativuus riippuvat aikaisempien tietojärjestelmien

toteutuksesta ja laajuudesta. Eri toimialojen välinen yhteistyö on vahvasti läsnä integraation tekemisessä, koska integraatio harvoin käsittää pelkästään tietojen vieniä versionhallintaan, jolloin eri toimialojen ydinosaamista voidaan hyödyntää ja integrointi on tehokasta ja vastaantulevat ongelmakohdat voidaan ratkaista ilman viiveitä.

Integrointiprosessi aloitetaan versionhallintaohjelmiston asennuksen ja tarvittavien aloitustoimenpiteiden jälkeen. Näitä toimenpiteitä ovat esimerkiksi tarvittavien käyttäjien luonti, verkkoyhteyksien avaaminen ja konfigurointi sekä tallennustilan varaaminen ohjelmiston käyttöön. Integrointi tulisi suorittaa etukäteen määrittelyssä järjestyksessä, tarvittaessa järjestys määritetään määrittelydokumentaatiossa, jotta integrointiprosessi pysyy tehokkaana. Integroitavia tietojärjestelmiä on esimerkiksi käyttäjätietokanta, joka sisältää myös käyttäjäkohtaiset oikeudet.

Tietojärjestelmän integroinnin yhteydessä tarkistetaan liityntärajapintojen toimivuus, jotta voidaan varmistua eri tietokantojen välisen yhteyden toimivuudesta. Rajapinnat ovat määritetty jo projektin aikaisemmassa vaiheessa, mutta joissain tapauksissa tarvitsee käyttää kolmatta ohjelmaa. Tämän ohjelman tehtävä on toimia rajapintamuuntimena kahden eri järjestelmän välillä.

Yleisimmät tietokannat joita käytetään rajapinnan kautta ovat Active Directory [33] ja XML-pohjaiset [34] tietokannat. Active Directory on Microsoftin kehittämä hakemistopalvelu, jota käytetään tyypillisesti esimerkiksi käyttäjätietokannoissa. XML-tietokantaa käytetään tiedon kuvaamiseen, tietokantaa voi käyttää esimerkiksi erilaisten ohjelmoitavien komponenttien ja ohjelmistojen tyyppien hallintaan.



*Kuva 6* Esimerkki integrointiprosessista.

## 5.2 Versionhallinnan tyypilliset käyttäjäryhmät

Versionhallinnan tyypillisiä käyttäjäryhmiä ovat ylläpitäjät, kunnossapito, tutkimus- ja kehitysosasto sekä operaattorit. Käyttäjäryhmien oikeudet, toimintatavat ja versionhallinnan käytön tarve poikkeavat toisistaan. Näiden lisäksi saattaa olla omat käyttäjäryhmänsä testi- ja käyttöönottoprosessien aikana. Käyttäjäryhmien jaottelu on mahdollisesti suoritettu jo yrityksen aikaisempien tietojärjestelmien perusteella. Myös määrittelydokumentaatioissa viitataan käyttäjäryhmiin. Pienissä yrityksissä ei usein katsota tarpeelliseksi jakaa käyttäjiä eri käyttäjäryhmiin, poislukien ylläpitäjät. Yrityksissä joissa käytetään paljon ulkopuolista työvoimaa saatetaan luoda oma käyttäjäryhmänsä vierailijoille, jolloin oikeudet rajataan vain heitä koskeviin komponentteihin. Yksi henkilö saattaa kuulua useaan eri käyttäjäryhmään ja käyttäjäryhmä muuttua työtehtävien mukana. Tarvittaessa yhdelle käyttäjälle saatetaan luoda useampi käyttäjätunnus eri toimenpiteitä varten, jos esimerkiksi tietoturvamääritykset näin vaativat. Esimerkkinä voidaan mainita, että ylläpitäjien tekemät toimenpiteet tulee tehdä aina yrityksen sisäverkosta.

Ylläpitäjien tyypilliset tehtävät ovat tietokantojen ylläpito, muutosten tekeminen versionhallinnan asetuksiin, varmuuskopioiden ottaminen ja ongelmatilanteiden ratkominen. He eivät tyypillisesti käytä versionhallinnan tietoja työssään, vaan tukevat muiden käyttäjäryhmien toimintaa ja varmistavat versionhallintaohjelmiston käytökelpoisuuden ottamatta kantaa sen varsinaiseen sisältöön. Ylläpitäjän rooli on olla järjestelmän asiantuntija ja käyttäjäryhmällä on kaikki käyttöoikeudet. Jos yrityksen tietokannassa on turvallisuuteen liittyviä tietoja, jotka ovat salassapidettäviä, ylläpitäjät voidaan kouluttaa näitä tietoja käsittelevien henkilöiden joukosta. Versionhallintaohjelmiston käyttöliittymä ja käytettävät ohjelmat saattavat erota muista käyttäjäryhmistä riippuen miten ohjelman käyttöliittymä on toteutettu.

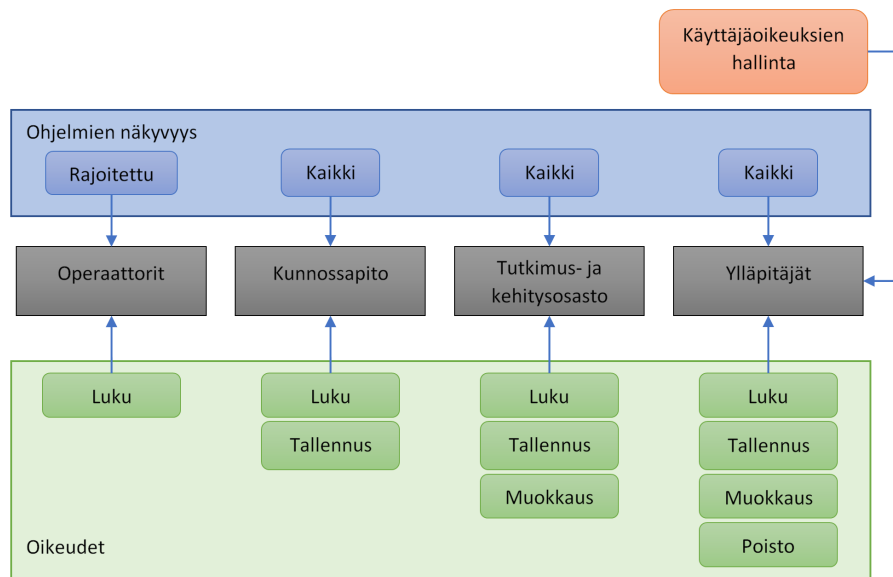
Kunnossapidon tyypilliset toimenpiteet versionhallintaan liittyen ovat komponenttien ohjelmistojen siirto uusiin asennettaviin komponentteihin. Kunnossapito käyttää järjestelmässä olevia tietoja myös vianhaun yhteydessä, ilman fyysistä yhteyttä kentällä olevaan laitteeseen. Komponenttien tai parametrien muuttuessa kunnossapito päivittää tietokantaan oikeat tiedot, riippuen yrityksen toimintatavoista. Kunnossapidolla tulee olla vahva yhteys tutkimus- ja kehitysosastoon sekä operaattoreihin, jotta vikatilanteissa selvittää nopeasti ja tehokkaasti. Kunnossapidon käyttäjäryhmä voidaan vielä jakaa erikseen asentajiin ja asiantuntijoihin, joista asentajilla on tyypillisesti vain lukuoikeudet ja asiantuntijoilla tämän lisäksi myös muutosoikeudet. Kunnossapidon kannalta versionhallintaohjelmisto on tietopankki, josta kunnossapitäjä löytää ajantasaisen tilanteen laitoksen tilasta.

Tutkimus- ja kehitysosasto käyttää versionhallintaa suunnitellessa tulevia muutoksia komponentteihin ja järjestelmiin. Tällä käyttäjäryhmällä on myös oikeudet luo-

da uusia komponentteja ja kokonaisuuksia versionhallintaan sekä lukita ohjelmistoja, jotka ovat käyttökielossa. Tutkimus- ja kehitysosasto käyttää versionhallintaa myös parametrien muutosten seuraamiseen, jolloin voidaan havaita epätavallisesti toimivia komponentteja tai ohjelmia. Versionhallinnan kannalta tämä käyttäjäryhmä on vaativin, koska samaa ohjelmistoa saattaa muokata ja käyttää useampi käyttäjä samanaikaisesti. Tämä tulee ottaa huomioon testikäytön yhteydessä, jotta vältetään myöhemmiltä ongelmilta.

Operaattoreiden näkökannalta versionhallinta on työkalu, jota saatetaan käyttää päivittäin töiden suorittamiseen. Tämä riippuu vahvasti yrityksen tyypistä, esimerkiksi erilaisia tuotantolinjoja saatetaan ajaa pienissä erissä ja ohjelmistoja vaihdetaan päivittäin. Tällön korostuu versionhallinnan käyttöliittymän selkeys ja helpokäyttöisyys, jotta ohjelmien vaihdossa ei tapahdu virheitä eikä vaadi ylimääräisiä toimenpiteitä operaattoreilta. Operaattorit käyttävät tyypillisesti versionhallinnan perusominaisuuksia, jotka eivät ole vaativia versionhallinnan kannalta mutta vaikuttavat yrityksen toiminnan tehokkuuteen.

Yhteenvetona käyttäjäryhmistä voidaan sanoa, että ne ovat tapauskohtaisia vaikkakin niistä voidaan tunnistaa neljä perustyyppiä, jotka eroavat toisistaan versionhallinnan käytön kannalta. Käyttäjäryhmistä vaativimpana voidaan pitää tutkimus- ja kehitysosastoa, jossa useat käyttäjät voivat tehdä muutoksia samoihin ohjelmistoihin. Kunnossapidolle ja operaattoreille versionhallinta on työkalu, jonka tehtävä on auttaa jokapäiväistä työtä. Ylläpitäjille versionhallinta on yksi ohjelmisto muiden joukossa, jossa saattaa kuitenkin olla useita rajapintoja sekä ohjelmiston toimimattomuus saattaa aiheuttaa suuria tuotannon menetyksiä.



**Kuva 7** Esimerkki käyttäjäryhmistä ja niihin liittyvistä toiminnoista.

## 5.3 Versionhallintaan liittyvä ohjeistus

Versionhallintaan liittyvä ohjeistus koostuu eri käyttäjäryhmille luotavista ohjeista sekä yleisistä ohjeista. Ohjeistuksen perustana käytetään usein versionhallinnan toimittajan luomaa ohjeistusta, josta luodaan yrityskohtaiset ohjeet. Ohjeistuksen tulee käsittää peruskäytön lisäksi harvemmin käytetyt toiminnot sekä määrittelydokumentaatiossa mainittavien metatietojen käytön ohjeistus. Riittävän kattavilla ohjeilla varmistetaan versionhallinnan tehokas käyttö ja yhtenäinen linja koko yrityksen toiminnassa. Luomalla riittävä sisäinen ohjeistus voidaan välttää ulkopuolisten konsulttien käyttöä, jolloin säästetään yrityksen resursseja. Ohjeistus voidaan määrittää käyttäjäryhmäkohtaisesti kyseisen käyttäjäryhmän tarpeet ja näkökanta huomioon ottaen. Kirjallisen ohjeistuksen tueksi voidaan luoda video-ohjeistuksia sekä koulutusmateriaalia, joilla tuetaan käyttäjiä versionhallintaohjelmiston käytön aloituksessa.

Käyttäjäryhmäkohtaisissa ohjeissa kuvataan versionhallintaohjelmiston liittyminen työn kannalta tärkeisiin työvaiheisiin, esimerkiksi uuden käyttäjän luominen ja käyttöoikeuksien asetus. Ohjeissa tulee olla kuvattuna tarvittavat esitietovaatimukset, mitkä muut yrityksen ohjeet liittyvät kyseiseen toimintoon ja mitä tietoja tulee viedä myös muihin tietojärjestelmiin. Yksinkertaisimmillaan ohjeena voi olla viittaus toimittajan ohjeen kohtaan tai toimittajan ohje muunnettuna yrityksen sisäisen ohjeen muotoon. Ohjeistuksen yksityiskohtaisuuteen vaikuttaa käyttäjäryhmä ja tekninen suuntautuneisuus, jolloin jokaisen asian yksityiskohtainen selittäminen ei välttämättä ole tarpeellista.

Yleisiin ohjeisiin kuuluvat esimerkiksi ohjelmiston käynnistys, tietojen tuonti ja vienti sekä tietorakenteen käyttö. Nämä ohjeet kuvaavat määrittelydokumentaatiossa määritettyjen asioiden tekemistä ohjelmistolla. Yleisissä ohjeissa ei oleteta ohjeen lukijalta järjestelmän tai sen käytön syvällisempää tuntemusta, jolloin ohjeen tulee olla yksityiskohtainen ja jokainen tarvittava toimenpide kuvattuna yksiselitteisesti. Yleiset ohjeet noudattavat yrityksen sisäisiä toimintatapoja, joita täydennetään versionhallintaohjelmiston käyttöohjeilla. Yleisiä ohjeita voidaan luoda jo testikäytön ja käyttöönoton aikana, jolloin toimintatavat saadaan yhdenmukaistettua aikaisessa vaiheessa ja ohjeita voidaan testata riittävästi ennen varsinaista julkaisua. Julkaisemalla ohjeet riittävän aikaisessa vaiheessa vältetään eri toimintatavoista johtuvista ristiriitaisuuksista ohjelmiston käytössä. Toimintatapojen ja ohjeiden välinen eroavaisuus tulee pitää mahdollisimman pienenä tai tarpeen vaatiessa ohjetta tulee päivittää vastaamaan toimintatapoja.

Ohjeistus on osa versionhallinnan dokumentaatiota, jota tulee päivittää säännöllisesti. Esimerkiksi versionhallintaan tulevat uudet ominaisuudet saattavat vaatia ohjeistuksen päivitystä. Myös ohjelmistoon liittyvien tietojärjestelmien muut-

tuminen tai lakimuutokset aiheuttavat päivitystarpeen. Tästä syystä ohjeita tulee katselmoida riittävän usein ja arvoida päivitystarve. Katselmointia varten tulee nimetä tarvittava määrä ohjevastaavia, joiden tehtävänä on määrittää ohjeistuksen päivitystarve ja tehdä mahdolliset muutokset. Isoissa yrityksissä tämä on vakiintunut toimenpide, jolloin ohjeella on määritetty vastaava henkilö ja muutokset hyväksytään esimiehen tai työryhmän toimesta.

## 5.4 Versionhallinnassa olevien tietojen käsittely

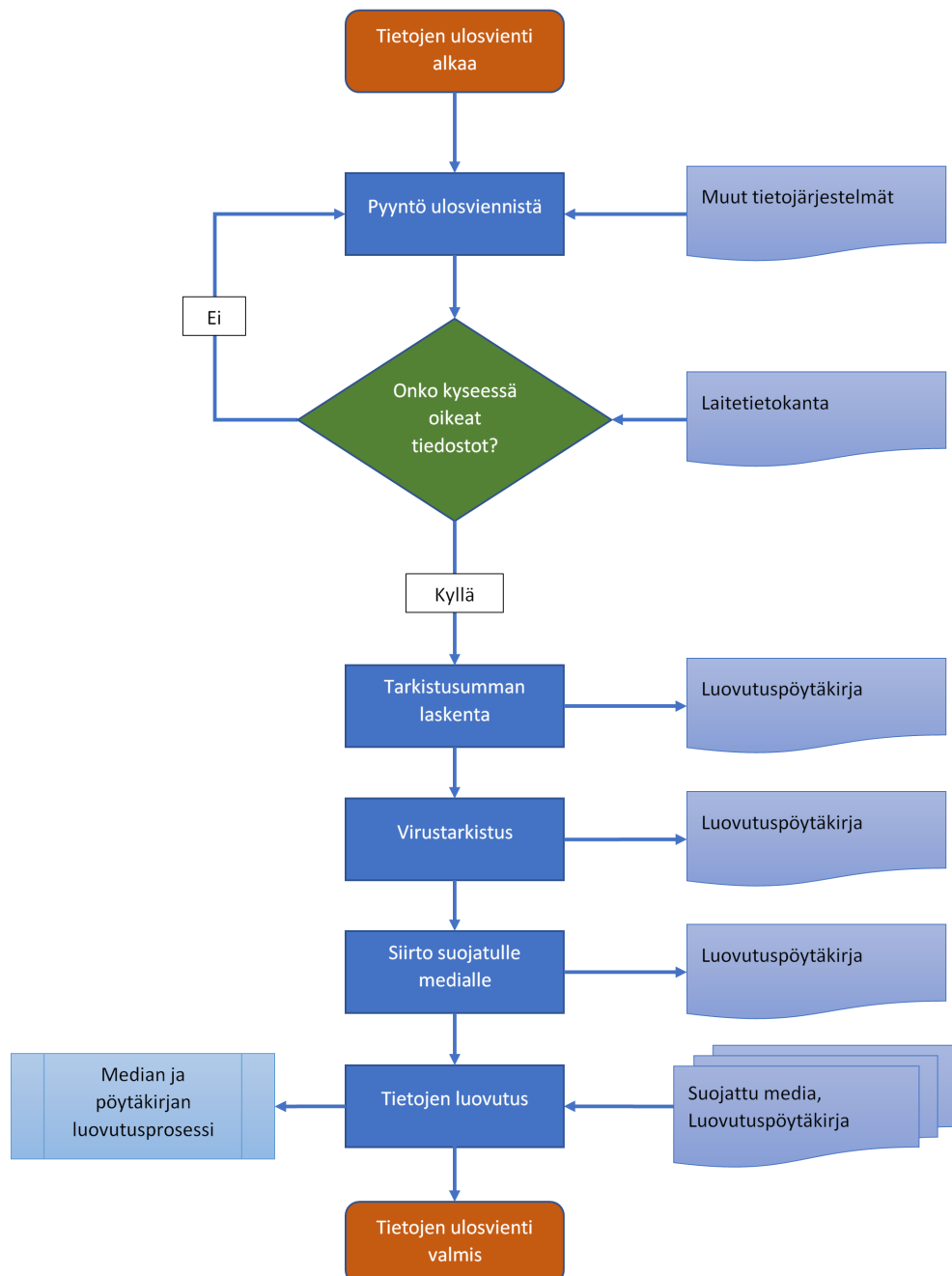
Versionhallinnassa olevien tietojen käsittely voidaan jakaa kahteen osaan, tietojen fyysiseen käsittelyyn sekä tietojen tilan käsittelyyn. Fyysisellä käsittelyllä tarkoitetaan siirtoa eri laitteiden välillä siirtomedian välityksellä, jolloin tiedostoa liikutetaan kahden fyysisesti erillisen laitteen välillä. Tietojen tilan käsittely sisältää tiedon missä tilassa kyseinen ohjelmisto on ja kenen käsittelyssä kyseinen ohjelma on. Tietojen käsittelyn periaatteet ovat yrityskohtaisia ja ovat riippuvaisia yrityksen toimialasta. Tietojen käsittelyyn voidaan ottaa kantaa jo määrittelyvaiheessa, varsinkin tilojen käsittelyn osalta. Mutta viimeistään ohjeistuksen teon yhteydessä nämä asiat tulee käsitellä ja ohjeistaa.

Ohjelmoitavien komponenttien käyttöön liittyy usein valmistajakohtaisten ohjelmistojen tarve, jolloin versionhallintaohjelmiston lisäksi tarvitsee olla käytössä mahdollisesti useita erillisiä ohjelmistoja. Joissakin tapauksissa tiedot tulee viedä erilliseen laite- tai valmistajakohtaiseen ohjelmointilaitteeseen, jolloin tulee määrittää tietojen käsittelyyn liittyvät toiminnot. Tietojen siirrossa kohdelaitteen ja versionhallinnan välillä tulee ottaa huomioon tietojen eheyden säilyminen, johon liittyy oleellisena osana myös tietoturva. Lisäksi käsiteltävät tiedot saattavat sisältää taloudellisesti merkittävää tietoa, jolloin tietojen salaaminen ja tietoturvalliset siirtomediat ovat osa käsittelyketjua.

Tietojen fyysistä käsittelyä ohjaavat yrityksen tietoturvakäytännöt ja ohjelmistojen merkityksellisyys yritykselle. Tietojen ulosvienti saattaa olla nimetty vain tietyille henkilöille ja tietokoneille. Sisäisesti käsittelyä ei ole välttämättä tarvetta rajata, jos tietojen käsittelyyn on riittävä ohjeistus. Ulosvientiä varten on hyvä luoda selkeä prosessi, jonka mukaan toimitaan, jotta tietojen oikeellisuus ja tietoturvallisuus säilytetään. Kuvassa 8 esitetään esimerkkiprosessi, joka ottaa kantaa tiedon eheyteen ja tietoturvaan.

Käsittelyprosessiin liittyy oleellisena osana myös tietojen lukitus ja tilamerkinnot, jolloin muut ohjelmistoa käsittelevät henkilöt ovat tietoisia kyseisen ohjelman tilasta. Tilojen määrä riippuu käytettävästä ohjelmistosta ja käyttötarkoitus määritetään yrityskohtaisesti, useimmiten ohjelmiston tekijä on määrittänyt omat ajatuksensa

tilojen määrittämisestä. Tietojen lukituksella estetään tahaton käyttö ja kesken-eräisten tai kiellettyjen ohjelmistojen vienti tehdasympäristöön. Tilamerkinnoilla määritetään kyseisen ohjelmistoversion tila, joita voivat olla esimerkiksi “vain testi-käyttöön”, “työn alla” ja “alkuperäinen”. Tiloja käytetään havainnoillistamaan sekä selkeyttämään versionhallinnan käyttöä sekä tarvittaessa eri tiloja voidaan yhdistää komponenttien lukituksiin.



*Kuva 8 Tietojen ulosvientiprosessi.*

## 5.5 **Versionhallinnan liittäminen toimintamalleihin**

Jotta versionhallinta saadaan tehokkaasti osaksi yrityksen aikaisempia toimintamalleja, tulee tuntea aikaisemmat toimintamallit ja tunnistaa toimintamallit, joissa on liityntä versionhallintaan. Jos yrityksellä ei aikasemmin ole ollut versionhallintaa, tulee toimintamallit tarkistaa ja pyrkiä löytämään mahdolliset liityntäpisteet versionhallintaan. Uusia toimintamalleja tulee luoda tarpeen mukaan ottaen huomioon päällekkäisyydet vanhoihin toimintamalleihin. Ohjelmitavien komponenttien versionhallinta liittyy laaja-alaisesti monen eri osa-alueen toimintamalleihin, jos liitettäviä toimintamalleja havaitaan vain muutama tulee arviointi tehdä uudelleen. Versionhallinta on sekä tietoa tuottava että vastaanottava toiminto, joten molemmat näkökannat tulee huomioida. Perusteluna ja pohjatietona tälle toiminnalle voidaan pitää standartia ISO 15489-1 [35], joka käsittelee asiakirjahallinnon perusteita ja käsitteitä. Tämän standardin pohjalta on mahdollista kehittää toimintamallit asiakirjojen laadintaan, talteenottoon ja hallintaan.

Toimintamallien ja versionhallinnan välinen yhteys voidaan jakaa tietoa tuottaviin ja tietoa tarvitseviin toimintoihin, osassa toimintamalleja tiedonsiirto voi toimia molempiin suuntiin. Esimerkkinä kaksisuuntaisesta toimintamallista on ohjelmiston vianhaku, jossa ohjelmisto noudetaan versionhallinnasta ja korjattu ohjelmisto viedään takaisin versionhallintaan.

## 6. VERSIONHALLINTAOHJELMISTON KÄYTTÖÖNOTTOPROJEKTI JA OHJELMISTON ESITTELY

Johtuen työn julkisuudesta esimerkissä käsitellään versionhallintaohjelmiston käyttöönottoprojektin kulkua, vastaantulleita haasteita ja niiden ratkaisuja sekä itse ohjelmistoa yleisellä tasolla.

Projekti etenee pääosin tämän diplomityön aikaisemmissa osissa läpikäytyin vaiheiden alkaen versionhallintaohjelmiston määrittelystä. Seuraavassa vaiheessa määritetään kriteerit täyttävät ohjelmistot ja valitaan testivaiheeseen pääsevä ohjelmisto. Testivaiheessa ohjelmistoa testataan eri käyttäjäryhmien välillä ja kerätään esille nousseet asiat, joihin tarvitaan tarkennusta toimittajalta. Kun tarvittavat tarkennukset on tehty ja ongelmat on ratkaistu, tehdään virallinen tarjouspyyntö ohjelmistosta. Tarjouksen hyväksymisen jälkeen varsinaisen ohjelmiston toimitus voi alkaa ja käyttöönottoprojekti aloitetaan.

Käyttöönottoprojekti alkaa määrittelydokumentaatiossa esitellyn verkkoratkaisun luonnilla, ohjelmiston asennuksella, pääkäyttäjän luonnilla ja asetusten tekemisellä. Samalla luodaan projektin aikainen ”ydinryhmä”, joka vastaan käyttöönoton läpimenosta ja varmistaa että projekti etenee sovitussa aikataulussa. Samalla he toimivat yhteyshenkilöinä projektin muiden sidosryhmien välillä.

Versionhallinta ohjelmiston esimerkissä käsitellään AUVESY:n kehittämää versiondog versionhallintaohjelmistoa, joka on suunnattu ohjelmitavien komponenttien versionhallintaan. Versiondog:n vahvuutena on tulkita useiden PLC, robotiikka ja käyttöliittymien ohjelmia ilman kyseisen valmistajan ohjelmiston käyttöä. Myös yleisemmät tiedostotyypit ovat tuettuina ja versionhallinta voidaan toteuttaa binäärikoodi tasolla mille tahansa tiedostotyyppille. AUVESY esittelee referenssiasiakanaan [36] muun muassa STIHL:n, Nestle:n ja CERN:n.

### 6.1 Määrittely

Versionhallinnan määrittelyn pohjana käytettiin aikaisempia dokumenttienhallintaan tarkoitettujen ohjelmistojen määrittelyä, joista kerättiin yleiset vaatimukset esimer-

kiksi toimittajan laatuvaatimusten osalta. Vaatimuksina oli esimerkiksi ISO 9001 [6] täyttäminen ja uskottavuus yrityksen toiminnan jatkuvuuden osalta. Tämä määrittely ei sulkenut pois avoimen lähdekoodin ohjelmistoja tai yritykselle varta vasten toteuttavaa yksilöllistä versionhallintaohjelmistoa.

Koska versionhallintaohjelmiston tuli huomioida ohjelmoitavien komponenttien hallinta, dokumenttien versionhallintaan käytettävän ohjelmiston ollessa valmiina, määrittelyn pääpaino oli ohjelmoitavien komponenttien hallinnan erikoispiirteillä. Näitä piirteitä oli eri tyyppisten komponenttien tuki, versioiden välisten eroavaisuuksien selkeä esitys, ohjelmien tilojen hallinta ja useiden eri käyttäjäryhmien käyttötarpeiden huomioon ottaminen.

Näiden pohjalta muodostettiin tehtävienmäärittely, joka valittavan versionhallintaohjelmiston tulee pystyä toteuttamaan. Määrittelyssä käytettiin alkutietoina yrityksessä käytössä olevien komponenttien listaa. Tästä listauksesta tunnistettiin komponentit, joiden tuki on erityisen tärkeää tai ei ole hallittavissa komponenttien ohjelmointiympäristön kautta.

Johtuen tallennettavien ohjelmien merkityksestä yrityksen toimintaan tietoturva- ja käyttöäjoikeuksienhallinta ovat erittäin merkityksellisiä, joten näiden määrittely tehtiin erillään muista määrittelyistä. Huomionarvoisia asioita ovat tietokannan varmuuskopioinnin ratkaisut, mahdollisuus eristää versionhallintaohjelmisto muista yrityksen ohjelmistoista ja mahdollisuus käyttää ohjelmistoa ilman yhteyttä ulkopuoliseen verkkoon. Käyttäjienhallinnan näkökannasta merkityksellisiä ovat ohjelmien näkyvyyden rajaaminen käyttäjäkohtaisesti, tietojen tarkoituksellisen tuhoamisen estäminen ja mahdollisuus seurata käyttäjien toimintaa kirjanpidon avulla. Nämä viimeksi mainitut osiat voidaan pitää myös osana tietoturvaa ja käytettävyyttä.

Lopullinen määrittelydokumentaatio valmistui määrittelyssä aikataulussa, mutta kaikkien määrittelyiden tarkkaa kuvaamista ei ollut mahdollista tehdä ennen kuin käytettävä ohjelmisto oli valittu. Tarkennettavia määrittelyksiä oli esimerkiksi tietokantarakenne, käyttöäjoikeuksien hallintametodit ja osa tietoverkkoratkaisuista. Nämä määrittelyt tarkennettiin projektin edetessä kun valitun ohjelmiston mahdollisuudet ja rajoitteet oli tiedossa.

## 6.2 Ohjelmiston valinta ja testikäyttö

Versionhallintaohjelmistojen kartoitus aloitettiin arvioimalla käytössä olevat versionhallintaohjelmistot, mutta niiden sopivuuden ohjelmoitavien komponenttien hallintaan ei koettu olevan riittävä. Tästä syystä kartoitettiin markkinoilla olevia vaihtoehtoja, esimerkiksi avoimen lähdekoodin GIT ja SVN. Myös näiden sopivuus ohjelmoitavien komponenttien versionhallintaan todettiin riittämättömiksi ja suurin osa

kaupallisestikin saatavilla olevia ohjelmia on tarkoitettu rivikoodia sisältävien ohjelmointikielien käyttöön. Ohjelmoitavien komponenttien versionhallintaan erikoistuneita ohjelmistoja ei juurikaan löydy markkinoilta ja taustatutkimuksen jälkeen versiondog vaikutti mahdolliselta ratkaisulta versionhallintaan.

Ennen testikäyttöä arvioitiin testiryhmään tarvittava henkilöiden määrä ja testikäyttöön osallistuvat henkilöt, otannaksi päätettiin noin 25 henkilöä. Henkilöt valittiin loppukäyttöä ajatellen useasta eri käyttäjäryhmästä ja kokemus vaihteli lähes juuri työt aloittaneesta kokeneeseen henkilöön. Pidempään yrityksessä työskennelleet osaavat yrityksen toimintatavat valmiiksi ja pystyvät vertaamaan toimintaa työtapoihin liittyen. Testikäyttö toteutettiin kolmessa osassa, aluksi pidettiin päivän mittainen esittely ja alkukoulutustilaisuus, tämän jälkeen vapaata käyttöä kolmen viikon ajan ja loppuyhteenveto toimittajan kanssa.

Esittelyssä käytiin läpi ohjelmiston tehnyttä yritystä ja ohjelmiston käyttötarkoitusta toimittajan näkökannasta. Myös ohjelman eri osia esiteltiin toimittajan toimesta. Alkukoulutus toteutettiin interaktiivisesti, jossa toimittajan edustaja kävi ohjelman rakennetta läpi yhdessä käyttäjien kanssa, luotiin käyttäjätunnukset ja kokeiltiin normaalin työkierron mukainen käyttö. Tämän aikana edustaja kävi lävitse heti vastaantulleita ongelmatilanteita ja luovutti pääkäyttäjälle oikeudet projektipäällikölle.

Vapaan käytön aikana ohjelmistoa testattiin eri tyyppisillä ohjelmoitavilla komponenteilla ja luotiin esimerkkikäyttötapauksia eri käyttäjäryhmien kesken. Lisäksi osalle testikäyttäjistä annettiin tehtäväksi yrittää erottaa versioiden välinen suhde tai saada järjestelmä sekaisin omalla toiminnalla, jotta järjestelmän toimintaa voitiin testata myös ei ohjeiden mukaisen toiminnan aikana. Samalla vastaantulleita havaintoja kerättiin valmiiksi loppuyhteenvetoa varten.

Loppuunyhentevedossa havainnot esiteltiin testikäyttöön osallistuneille henkilöille ja vastaantulleista ongelmista keskusteltiin toimittajan edustajan kanssa. Osa ongelmista liittyi selkeästi yrityksen vakiintuneisiin käytäntöihin, joiden sovittaminen ohjelmiston toimintaan ei ollut kaikilta osin mahdollista. Loppuyhteenvedon havaintojen perusteella ohjelmisto todettiin käyttötarkoitukseensa sopivaksi ja täyttävän aikaisemmin laaditut määrittelyt. Näin ollen voidaan edetä suoraan vaatimusten täyttävän ohjelmiston testauksesta eteenpäin.

### 6.3 Ohjelmiston esittely

Versiondog on kolmen henkilön start-up yrityksen lopputulos, jonka kehitys on aloitettu vuonna 2007. AUVESY:n kotisivujen mukaan ohjelmiston kehittelyyn käytettiin aikaa kaksi vuotta ennen markkinoille julkaisua. Tämän jälkeen kasvu on ollut tasaista ja nykyään yrityksessä työskentelee 65 henkilöä. Versiondog toimii

palvelin-asiakaspohjaisesti, joten yhteyden palvelimeen on oltava aktiivinen ohjelman käytön ajan. Asiakasohjelmisto koostuu kuudesta asiakasohjelmasta, joiden toiminnallisuus on osaksi päällekkäistä, mutta on suunnattu eri käyttötarkoituksiin. Näiden lisäksi on saatavilla kymmenen lisäosaa, jotka voidaan tilata erikseen. Palvelin- ja asiakasohjelmisto on mahdollista asentaa eri tietokoneille mutta myös samalle tietokoneelle tehtävät asennukset ovat mahdollisia. Versionhallinta perustuu komponentteihin, jotka voivat sisältää yhden tiedoston tai koko projektin ohjelmiston ja koko komponentin versio muuttuu samalla kertaa.

Asiakasohjelmisto koostuu seuraavista osista: UserClient, EasyClient, AdminClient, BackupClient, ReportClient ja WebClient, joiden lisäksi voidaan yrityksen tarpeiden mukaan tilata myös lisäosia. Perusohjelmista UserClient ja EasyClient on tarkoitettu päivittäisten versionhallintaan liittyvien toimintojen tekemiseen. EasyClient on nimensä mukaan yksinkertaistettu käyttöliittymä, jossa käyttäjän valinnat on tehty helpoiksi yksinkertaistamalla kuvakkeita ja karsimalla esillä olevia toimintoja. AdminClient on tarkoitettu pääkäyttäjän käyttöön, sillä voidaan esimerkiksi hallita käyttäjätunnuksia ja komponenttien näkyvyyttä. ReportClient on tarkoitettu keskitettyyn raporttien tekoon ja WebClient on verkkopohjainen käyttöliittymä komponenttien hallintaan.

Versiondog:ssa jokaiselle erikseen hallittavalle osalle luodaan komponentti, jolle määritetään komponenttityyppi luonnin yhteydessä. Komponentti itsessään ei ota kantaa mitä tietoja se sisältää, mutta komponentin tyyppillä määritetään miten versiondog tulkitsee ohjelmaa. Jos komponentin tyyppi eroaa todellisesta tyyppistä esimerkiksi eroavaisuuksien tunnistus ei toimi oikein. Tällä tavoin yksi ohjelmiston ominaisuuksista jää käyttämättä, sillä ohjelma pystyy esittämään tuetut komponentit kuten ne on esitetty natiiviohjelmassa ja visualisoimaan eroavaisuudet tuetuissa ohjelmistoissa. Komponenttien hallinta luo omat rajoitteensa versionhallinnassa, tähän tapaukseen palataan tarkemmin tämän diplomityön kappaleessa 6.6.

UserClient ja EasyClient ohjelmissa on ohjelman käyttöä tukeva työkierto kuvattu vasemmalta oikealle, jolloin käyttäjän on helpompi seurata seuraavaa toimenpidettä. Työkierron mukaan komponentti otetaan käyttöön toiminnolla "Check-Out", jossa komponentti siirretään käyttäjän tietokoneelle. Tämän jälkeen komponentti avataan muokkausta varten "Open with editor" ja tehdään tarvittavat muutokset. Seuraavassa vaiheessa luodaan uusi versio "Create new version", jolloin versiondog tarkistaa versioiden väliset eroavaisuudet. Viimeisessä vaiheessa komponentti siirretään takaisin versionhallintaan "Check-In" toiminnolla, jolloin työkierto on valmis. Muokkauksen suorittamisen ajaksi komponentti on mahdoliista lukita, jolloin muut käyttäjät eivät voi luoda uusia versioita. Lukitus voidaan suorittaa mille tahansa komponentille ja avaus voidaan tehdä vain lukitsijan tai pääkäyttäjän toimesta.

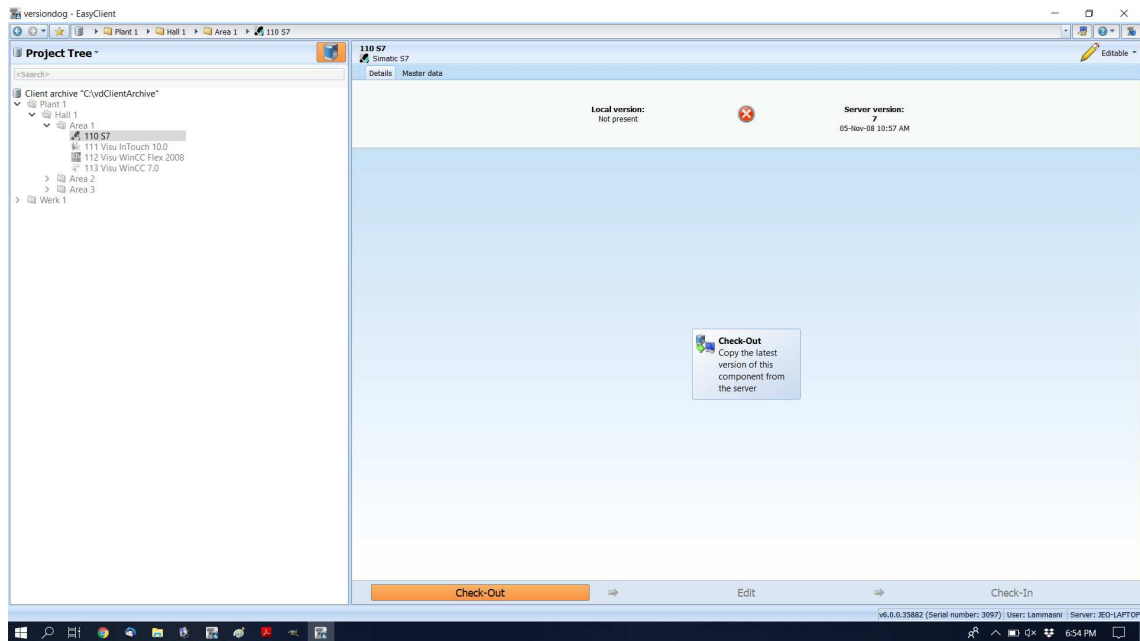
versiondog Logged in as: Lammasni

Component log - Filter...

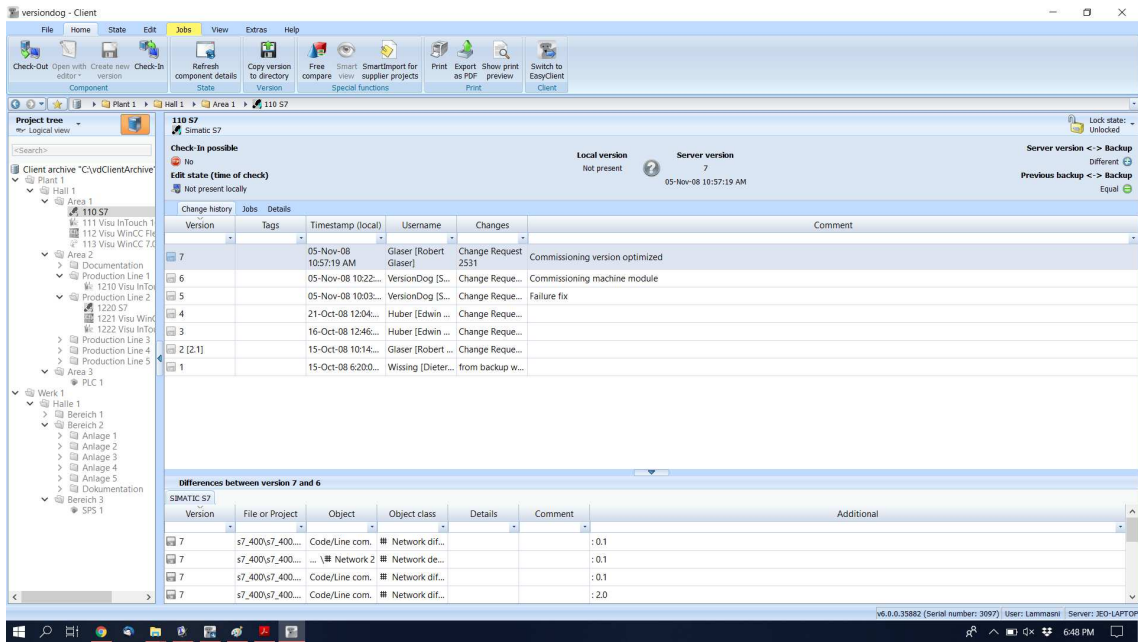
Timestamp	Directory	Component n...	Component t...	Command	Username	Computer name	Version
05/01/2011 15:23:29	Werk 1Halle 1Berel...	112 Visu WinCC Flex...	WinCC flexible	Check-Out	VersionDog [Superad...	VMWARE-100-006-02	2
05/01/2011 15:21:53	Plant 1Hall 1Area 1	112 Visu WinCC Flex...	WinCC flexible	Check-Out	VersionDog [Superad...	VMWARE-100-006-02	2
27/01/2010 12:03:28	Werk 1Halle 1Berel...	Checkliste Installation	Word	Check-In	VersionDog [Superad...	VMWARE-VD-SERVER 1	23/0
27/01/2010 12:03:28	Werk 1Halle 1Berel...	SPS 1	CODESYS	Check-In	VersionDog [Superad...	VMWARE-VD-SERVER 1	01/1
27/01/2010 12:03:26	Werk 1Halle 1Berel...	CAD-Plan	PDF	Check-In	VersionDog [Superad...	VMWARE-VD-SERVER 2	23/1
27/01/2010 12:03:25	Werk 1Halle 1Berel...	Buecherliste	Excel	Check-In	VersionDog [Superad...	VMWARE-VD-SERVER 1	23/0
27/01/2010 12:03:23	Werk 1Halle 1Berel...	PCS7 V7.1	Simatic PCS 7	Check-In	VersionDog [Superad...	VMWARE-VD-SERVER 1	21/0
27/01/2010 12:03:21	Werk 1Halle 1Berel...	S7-400	Simatic S7	Check-In	VersionDog [Superad...	VMWARE-VD-SERVER 1	21/0
27/01/2010 12:03:20	Werk 1Halle 1Berel...	S7-300	Simatic S7	Check-In	VersionDog [Superad...	VMWARE-VD-SERVER 1	21/0
27/01/2010 12:03:18	Werk 1Halle 1Berel...	MultiPrjLib	Simatic S7 Library	Check-In	VersionDog [Superad...	VMWARE-VD-SERVER 1	21/0
27/01/2010 12:03:17	Werk 1Halle 1Berel...	MultiPrj	Simatic S7 Multipr...	Check-In	VersionDog [Superad...	VMWARE-VD-SERVER 1	22/0
27/01/2010 12:03:16	Werk 1Halle 1Berel...	1230 RSLogix500	RSLogix500	Check-In	VersionDog [Superad...	VMWARE-VD-SERVER 1	21/0
27/01/2010 12:03:15	Werk 1Halle 1Berel...	1222 Visu InTouch 10.0	InTouch	Check-In	VersionDog [Superad...	VMWARE-VD-SERVER 1	16/1
27/01/2010 12:03:14	Werk 1Halle 1Berel...	1221 Visu WinCC Fle...	WinCC flexible	Check-In	VersionDog [Superad...	VMWARE-VD-SERVER 1	16/1
27/01/2010 12:03:12	Werk 1Halle 1Berel...	1220 S7	Simatic S7	Check-In	VersionDog [Superad...	VMWARE-VD-SERVER 1	16/1
27/01/2010 12:03:10	Werk 1Halle 1Berel...	1210 Visu InTouch 10.0	InTouch	Check-In	VersionDog [Superad...	VMWARE-VD-SERVER 1	16/1
27/01/2010 12:03:09	Werk 1Halle 1Berel...	113 Visu WinCC 7.0	WinCC	Check-In	VersionDog [Superad...	VMWARE-VD-SERVER 2	16/1
27/01/2010 12:03:08	Werk 1Halle 1Berel...	112 Visu WinCC Flex...	WinCC flexible	Check-In	VersionDog [Superad...	VMWARE-VD-SERVER 2	15/1
27/01/2010 12:03:07	Werk 1Halle 1Berel...	111 Visu InTouch 10.0	InTouch	Check-In	VersionDog [Superad...	VMWARE-VD-SERVER 1	15/1
27/01/2010 12:03:05	Werk 1Halle 1Berel...	110 S7	Simatic S7	Check-In	VersionDog [Superad...	VMWARE-VD-SERVER 7	05/1
27/01/2010 12:03:04	Plant 1Hall 1Area 3	PLC 1	CODESYS	Check-In	VersionDog [Superad...	VMWARE-VD-SERVER 1	01/1
27/01/2010 12:03:03	Plant 1Hall 1Area 2...	PCS7 V7.1	Simatic PCS 7	Check-In	VersionDog [Superad...	VMWARE-VD-SERVER 1	21/0
27/01/2010 12:03:02	Plant 1Hall 1Area 2...	S7-400	Simatic S7	Check-In	VersionDog [Superad...	VMWARE-VD-SERVER 1	21/0
27/01/2010 12:03:01	Plant 1Hall 1Area 2...	S7-300	Simatic S7	Check-In	VersionDog [Superad...	VMWARE-VD-SERVER 1	21/0

38 rows

*Kuva 9 WebClient käyttöliittymä.*

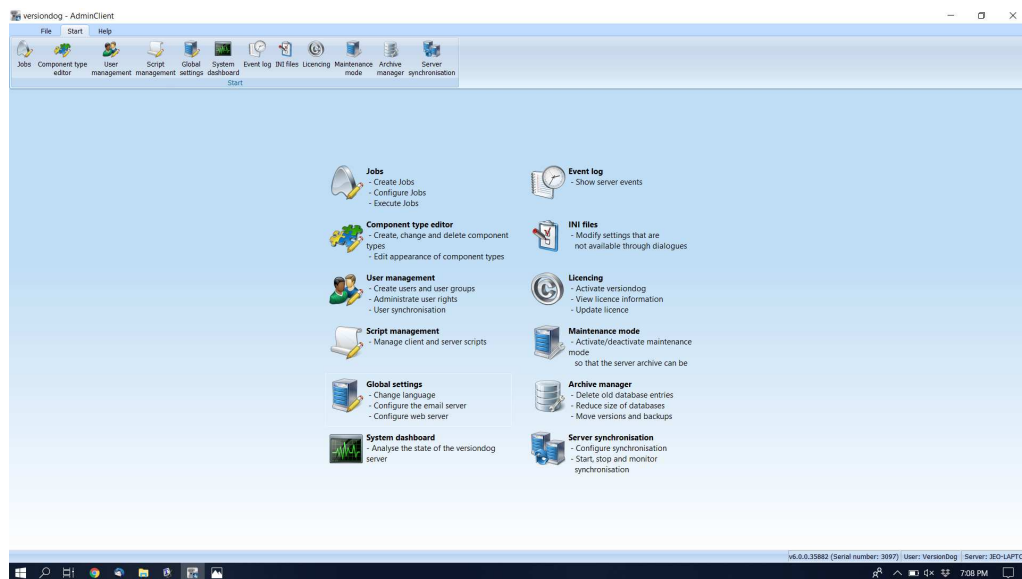


*Kuva 10 EasyClient käyttöliittymä.*



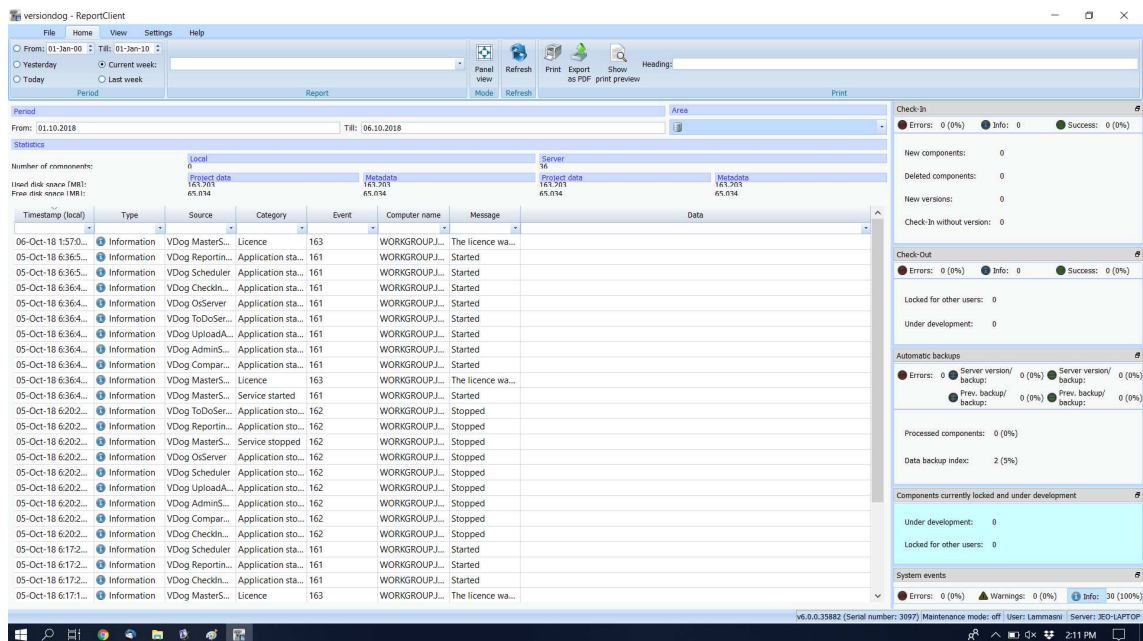
*Kuva 11 UserClient käyttöliittymä.*

AdminClient sisältää versiondog:n kaikki pääkäyttäjän toiminnot esimerkiksi käyttäjienhallinnan, ohjelmiston asetukset, komponenttien ja komponenttityyppien asetukset, lisenssien hallinnan ja palvelimen varmuuskopioiden hallinnan. Käyttäjähallinnassa määritetään käyttäjätunnukset, komponenttien ja kansioden näkyvyydet sekä esimerkiksi käyttäjien oikeudet. Oikeuksia voidaan hallita komponenttikohtaisesti, jolloin voidaan rajoittaa tehokkaasti eri käyttäjien ja käyttäjäryhmien toiminnallisuutta.



*Kuva 12 AdminClient käyttöliittymä.*

ReportClient on tarkoitettu ainoastaan keskitettyyn raporttien luontiin, ohjelmassa on mahdollista tarkistaa kaikkien komponenttien tilanne samanaikaisesti. Ja luoda valmiit pohjat raporttien luontiin sekä tarkistaa eri käyttäjien tekemät toimenpiteet. Ohjelmassa on myös mahdollista nähdä varmuuskopioiden viimeisimmän tilanteen, jos automaattinen varmuuskopiointi on käytössä, ja varmistua että versionhallinnassa on viimeisimmät versiot laitteiden ohjelmistoista. ReportClient ohjelma pitää myös kirjaa koko versiondog:n käytöstä.



*Kuva 13 ReportClientin käyttöliittymä.*

## 6.4 Ohjelmoitavien laitteiden ohjelmien siirto versionhallintaan

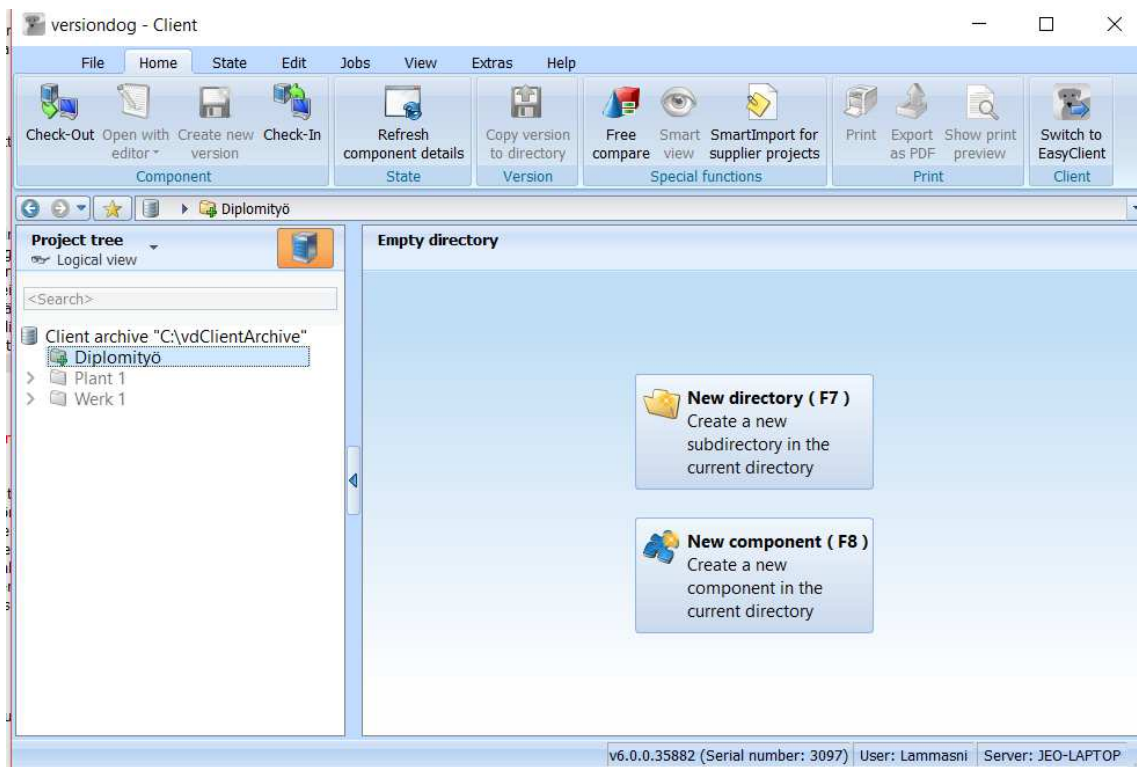
Ohjelmoitavien komponenttien ohjelmistojen vienti versiondog:iin toteutettiin vaiheittain, yksi kokonaisuus kerrallaan. Samalla tarkistettiin tallennettavan ohjelmiston ja parametrien ajantasaisuus. Ohjelmistojen vientiä varten luotiin prosessi, jota seurataan jokaisen versionhallintaan vietävän komponentin kanssa. Prosessin mukaan tarkistetaan, että järjestelmän kuvausdokumentissa on mainittu kaikki järjestelmän sisältämät versionhallintaan vietävät laitteet ja niiden laitetunnukset ovat oikein sekä vietävät tiedot vastaavat todellisuutta.

Jos kerralla viedään useita kokonaisuuksia versionhallintaan, on tärkeää priorisoida järjestys, jotta yrityksen kannalta merkitykselliset kokonaisuudet tulee vietyä ensin versiondog:iin. Samaa priorisointia käytetään myös tarkistuksen laajuuteen. Kaikkia komponentteja ei vertailla asennettuun versioon vaan luotetaan viimeisimpään tallessa olevaan tietoon. Tallennetun tiedon aikamäärä ja järjestelmään tehdyt

muutokset ovat valmiiksi tiedossa, joten näitä vertaamalla voidaan luoda arvio tiedon oikeellisuudesta.

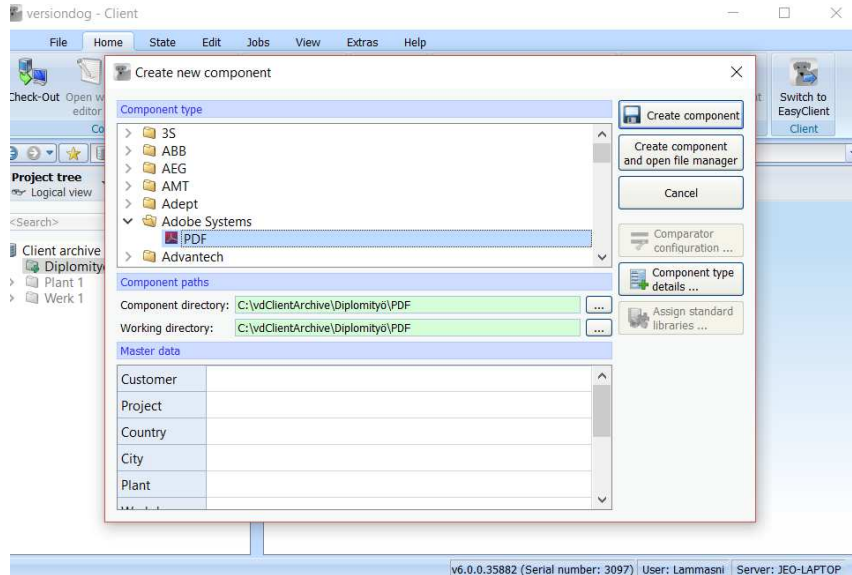
Tietojen tallentaminen versiondog:iin tarkastusprosessin jälkeen oli varsin yksinkertainen toimenpide: tarvittava tietorakenne oli luotu jo valmiiksi, luotiin uusi komponentti metatietoineen ja vietiin tallennettavan laitekomponentin tiedot tämän komponentin alle. Komponentin tyyppi on etukäteen määritetty tallennettavan laitteen mukaan. Lopuksi vietiin tiedot palvelimelle ja kuitattiin komponentti valmiiksi käyttöä varten.

Kuvassa 14 on esitetty uuden kansion luonnin jälkeinen tilanne. Valitsemalla "New component" aloitetaan uuden komponentin luonti. Versiondog:n asetuksista voidaan valita, onko samassa hakemistossa vain komponentteja vai onko mahdollista luoda alikansioita samalle tasolle komponenttien kanssa.



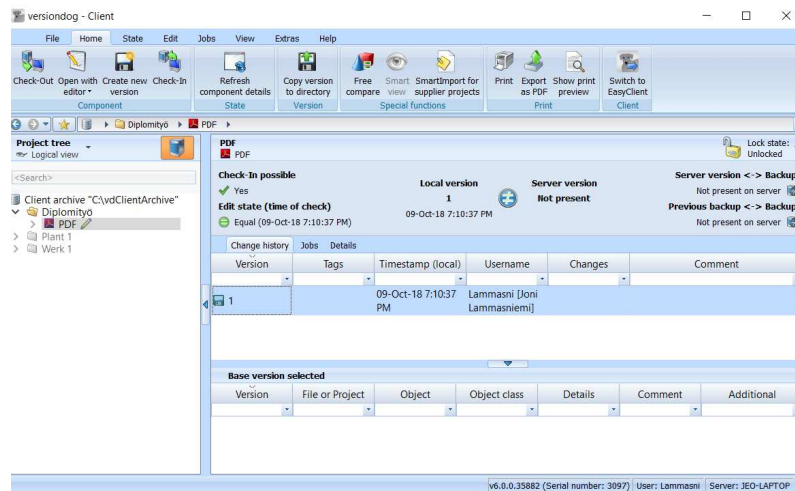
*Kuva 14 Uuden komponentin luominen.*

Kuvassa 15 esitetään komponentin tyyppin valinta, tässä tapauksessa kyseessä on diplomityön kehitysversio, joka on PDF-tiedosto. Komponenttityypit on jaoiteltu ohjelmiston tuottaneen yrityksen mukaisesti osiin. Samalla on mahdollista täyttää tarvittavat metatiedot, joiden tekstejä voidaan muokata AdminClient ohjelmalla. Samassa yhteydessä annetaan komponentin nimi, joka on muutettavissa myös jälkikäteen.



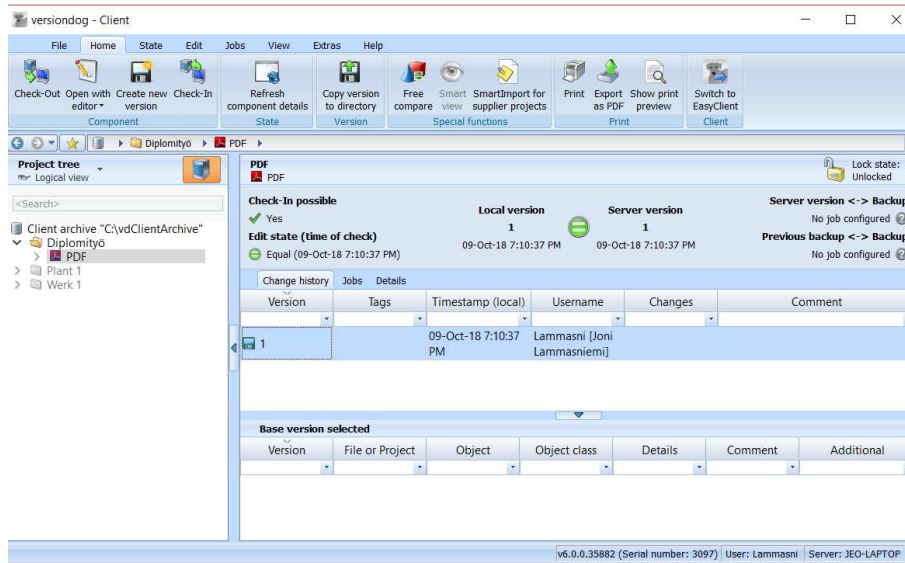
*Kuva 15 Komponentin tyyppin valinta ja metatietojen täyttö.*

Kuvassa 16 on komponentti luotu ja valittu haluttu tiedosto tai projekti komponentille. Valinta tapahtuu vetämällä haluttu ohjelma komponentin päälle. Samalla versiondog luo paikalliskopion kyseisestä ohjelmasta.



*Kuva 16 Komponentin paikalliskopio valmiina.*

Kuvassa 17 komponentti on kirjattu sisään palvelimelle ja siitä on luotu versio 1. Samassa ikkunassa näkyy että paikallinen ja palvelimella oleva versio ovat samalla numerolla ja komponentti on muiden käyttäjien käytettävissä.



*Kuva 17 Komponentti kirjattu sisään järjestelmään.*

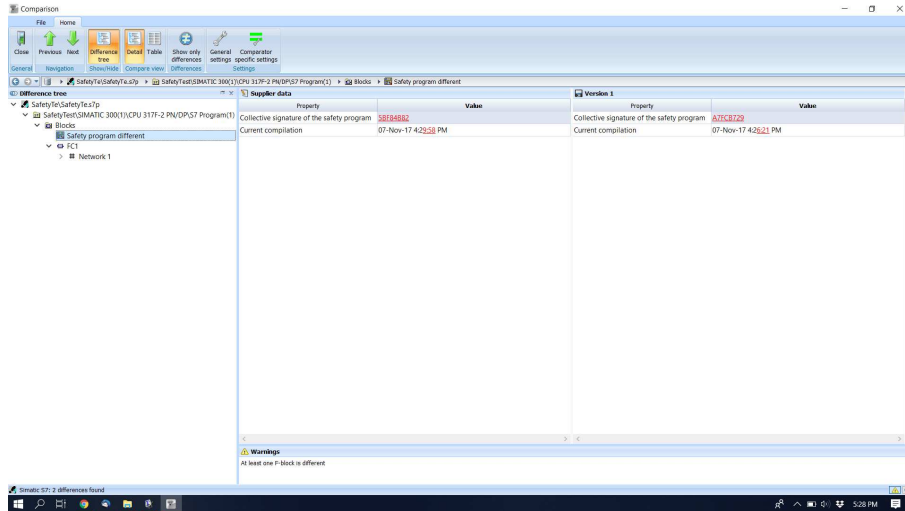
## 6.5 Eroavaisuuksien esittäminen

Versiondog pystyy esittämään tuettujen laitteiden ohjelmistojen eri versioiden väliset eroavaisuudet visuaalisesti eri väreillä. Kuvaustapa vastaa alkuperäistä esitysmuotoa. Kaikki eroavaisuudet listataan ohjelman rakenteen mukaisessa järjestyksessä. Tämän lisäksi ohjelma ilmoittaa muuttuneet tarkistussummat ja päivämäärät. Eroavaisuuksien havainnointiin ei tarvita ulkopuolista ohjelmaa, jos vertailu suoritetaan versiondog:n tukemaan laitteistoon tai tiedostotyyppiin. Mutta muokkauksia ei ole mahdollista tehdä versiondog:n avulla.

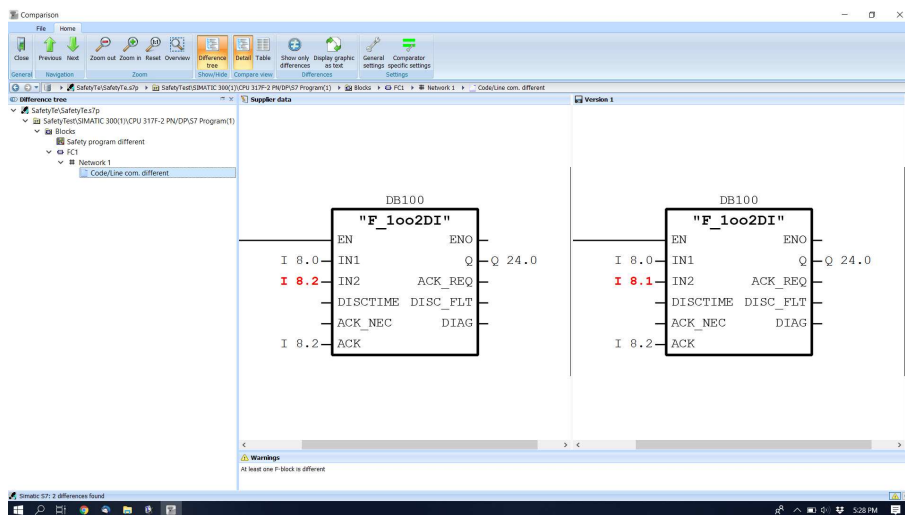
Muutosten visuaalinen esittäminen helpottaa eroavaisuuksien havainnointia, jolloin muutosten arviointi on helpompaa käyttäjän kannalta. Vertailtavat versiot voidaan valita vapaasti ja vertailu voidaan suorittaa myös ulkopuolisella medially olevaan dataan nähden. Näin ollen eri versioiden ja työnalla olevien ohjelmia vertailua voidaan suorittaa vapaasti ilman, että ohjelmia tarvitsee viedä versiondog-järjestelmään.

Kuvissa 18 ja 19 on esitetty S7F-300 turvalogiikkaan tehty muutos, jossa Siemens:n tekemään sisääntulojen lukemiseen tarkoitettuun funktioon on vaihdettu toisen sisäänmenon osoite. Samasta syystä turvaohjelman tarkistussumma ja ohjelman käynnön aika on muuttunut. Versiondog esittämä tarkustussumman ja aikaeroavaisuus

on esitetty kuvassa 18 ja tikapuuohjelman muutoksen graafinen esitys on esitetty kuvassa 19.



*Kuva 18 Tarkistussumman ja kääntöajan eroavaisuuden esitys.*

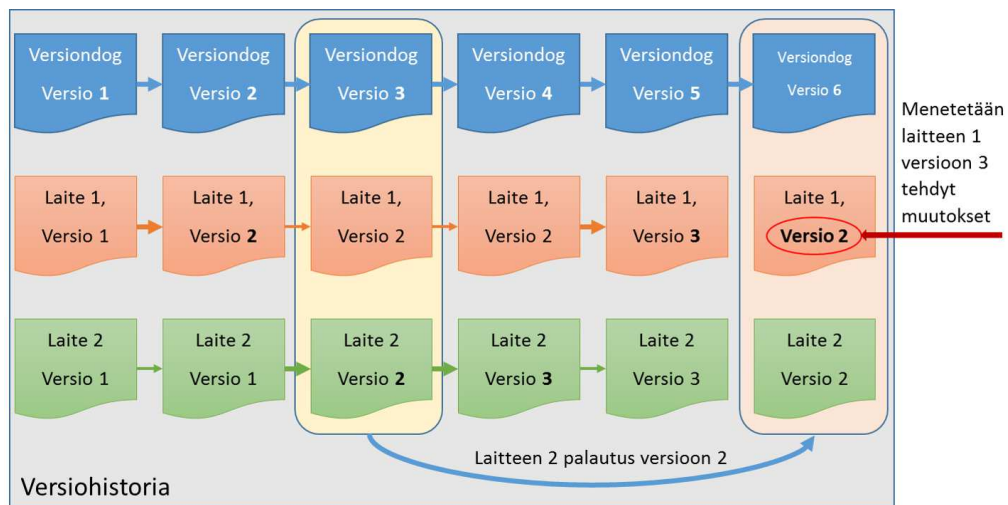


*Kuva 19 Tikapuuohjelman eroavaisuuden esitys.*

## 6.6 Vastaantulleita ongelmia ja niiden ratkaisuja

Versiondog:n käyttöönoton aikana vastaantulleet isoimmat ongelmat liittyivät testi-asennuksen ja todellisen asennuksen toteutuksen eroihin, vaikka määrittely oli tehty valmiiksi, niin testiympäristö toteutettiin matalammilla vaatimuksilla kuin varsinainen asennus. Tästä johtuen havaittiin asiakas-palvelin yhteydessä ongelmia, joita ei esiintynyt varsinaisen testikäytön aikana.

Versiondog:n tavassa käsitellä kaikkia ohjelmia komponentteina havaittiin komponenttien versionhallinnassa ominaisuus, jota ei oltu osattu ottaa huomioon määrittelydokumentaatioissa. Ongelma esiintyy kun samassa versionhallintakomponentissa on useampia fyysisiä komponentteja, joiden välillä ei ole pakollista versioriippuvuutta. Tällöin minkä tahansa fyysisen komponentin ohjelmiston päivitys uuteen versioon nostaa koko versionhallintakomponentin versiota, jolloin kahden eri fyysisen komponentin ohjelmiston yhden version nousu tarkoittaa kahden version nousua versionhallinnassa. Ongelma ilmenee kun pitäisi palauttaa yhden fyysisen komponentin ohjelma edelliseen versioon, jolloin koko versionhallintakomponentissa paluu saattaa olla useita versioita ja samalla menetetään välissä olevien versioiden muutokset. Kuvassa 20 on esitetty tilanne, jossa versionhallintakomponentin sisällä on kaksi fyysistä laitetta, esimerkiksi PLC ja HMI, joiden välillä ei ole pakollisia riippuvuuksia. Laitteiden ohjelmistoja päivitetään eri tahtiin, jolloin versionhallintakomponentin numero muuttuu jokaisella päivityksellä. Esimerkiksi johtuen laitteen 2 uusimman ohjelman virheestä päätetään palauttaa laitteen ohjelmisto takaisin versioon 2. Samalla palataan versionhallintakomponentin versioon 3, jossa laitteen 1 ohjelmiston versio on 2. Näin ollen palautuksessa menetetään laitteen 1 ohjelmistoversion 2 jälkeiset muutokset, eikä laitteen 1 versio ole enää viimeisin versio.



*Kuva 20* Version muutokset komponentin sisällä.

Tästä tilanteesta johtuen yrityksen toimintamalleihin tuli tehdä uusi määrittely, jonka mukaan jokainen fyysisen laitteen ohjelmisto tulee toimittaa erillisenä ohjelmistona. Poikkeuksena ovat kokonaisuudet, joiden välillä on pakollinen riippuvuus versioiden välillä. Mutta tämä poikkeus on mahdollista jättää pois määrittelyistä, jolloin tulkinta on yksiselitteistä ja selkeämpää toimittajan suuntaan. Ongelmana on jo valmiiksi tehdyt ohjelmistot, jotka joudutaan jakamaan osiin erikseen tai luomaan

useampia erillisiä versionhallintakomponentteja. Jos luodaan useampia erillisiä versionhallintakomponentteja on mahdollista asentaa väärä versio komponentteihin, jos ohjelmointiliitännät ja -ympäristö ovat samat.

Samantyyppinen ongelma esiintyy tapauksissa, joissa ohjelma sisältää usean eri fyysisen komponentin tietoja. Jos versionhallintakomponentiksi on määritetty fyysinen laite, esimerkiksi laitepaikkatunnuksen mukaan, versionhallintakomponenttien luonti on yksiselitteistä. Tällöin ohjelma tulee kopioida jokaiselle fyysiselle laitteelle erikseen. Lisäksi on luotava toimintamalli, miten ja missä aikamääreessä se on purettava vain kyseessä olevan laitteen tietoja sisältäväksi paketiksi. Jos tietoja siirretään vähän kerrallaan tämä voidaan tehdä versionhallintaan viennin yhteydessä, mutta jos vietävä tietomäärä on suuri niin työn tekeminen saattaa olla mielekäästä siirtää myöhemmäksi.

Johtuen monesta eri käytössä olevasta komponenttityypistä on mahdollista käyttää samantyyppiseen tiedostoon useita eri komponenttityyppejä, jotka toimivat oikein mutta aiheuttavat ristiriitoja komponenttityyppien määrän vertailun osalta. Ratkaisuna on määrittää komponenttityypit etukäteen ja tarvittaessa päivittää toimintamallia, jos parempi komponenttityyppi ilmenee myöhemmin ja uusia komponentteja lisätään versiondog:iin päivitysten yhteydessä. Versiondog mahdollistaa komponenttityypin vaihtamisen ilman, että versionhistoriaa menetetään muutoksen aikana.

Versiondog:n tapa käsitellä muuttuneita tiedoston nimiä ja tyypppejä on yksiselitteinen, jos tiedoston nimi tai -tyyppi muuttuu välissä niin versiondog luo kaksi rinnakkaista tiedostoa saman versionhallintakomponentin alle. Käytön yhteydessä pitää määrittää, mitä tiedostoista käytetään. Tällöin on mahdollista, että versionhallinta menee sekaisin. Varsinkin jos muutoksia tehdään ristiin tiedostojen välillä. Tämä pätee erityisesti yleistyyppisiin tiedostoihin. Tästä syystä on oltava erityisen tarkka tiedostojen takaisintuonnin yhteydessä, että tiedostojen nimet ja tyypit eivät ole muuttuneet. Ongelma korostuu yrityksen ulkopuolella tehtävissä muutoksissa. Tämän johdosta toimintamalleja tulee päivittää tiedostojen osalta, jolloin tiedostojen nimeämiskäytännöt on määritetty valmiiksi.

Ongelmaa aiheutti myös versionhallintaan vietävän tiedon vertailu todelliseen laitteessa olevaan tietoon johtuen siitä, että monessa laitteessa on ajoparametreja. Nämä parametrit muuttuvat laitteen käytön aikana ja ovat enemmänkin tilatietoja kuin parametreja. Tästä syystä versiondog havaitsi useita eroavaisuuksia verratessa valmiiksi tallennettua ja asennettua versiota. Tämän jälkeen havaitut eroavaisuudet tuli käydä yksitellen läpi ja määritellä mitkä eroavaisuudet ovat merkityksellisiä. Näiden tulkinta ei ole aina yksiselitteistä ja ilman kyseisen laitteen syvällistä tuntemusta tulkinta vie aikaa. Ratkaisuna oli määrittää yrityksen sisältä osaamisalueet ja vastuuhenkilöt eri tyyppisille laitteille, jolloin henkilöiden osaamista käytetään hyväksi eroavaisuuksien tulkinnessa. Näin saavutetaan tehokkuutta, kun vain on-

gelmatapauksissa tarvitsee ottaa yhteyttä muihin henkilöihin, ja nämä henkilöt on nimetty etukäteen. Myös versiondog:iin tietoa tallentavaan työryhmään pyritiin valitsemaan henkiöitä, jotka osaavat kyseisen järjestelmän tai kokonaisuuden ohjelmistojen käytön ja pystyvät tulkitsemaan erovaihteluita tallennuksen yhteydessä. Syy miksi ei suoraan haluttu tallentaa asennettua versiota oli, että laitteesta haetussa versiossa puuttuu usein kommentointi, jolloin menetetään osa ohjelman sisältämää informaatiosta.

## 6.7 **Versiondog:n käyttöä varten luodut ympäristöt**

Versiondog:n käyttöympäristö jaettiin yrityksessä kahteen osaan, varsinaiseen käyttöympäristöön sekä testi- ja koulutusympäristöön. Jakamalla ympäristö kahteen eri osaan voidaan versiondog:ia kouluttaa ja testata eri toiminnallisuuksia ilman, että varsinaisen käyttöympäristön eheys vaarantuu. Käyttöympäristöihin jako on myös osa tietoturvaluottua, ympäristöt eivät ole yhteyksissä toisiinsa, jolloin käyttöoikeuksia voidaan jakaa vapaammin.

Varsinainen käyttöympäristö luotiin yritysten toimintamallien mukaan ja se täyttää yrityksen sisäiset tietoturvaluottimukset, jotta tallennettavien tietojen tietoturvaluottuus voidaan varmistaa ja tiedon eheyteen voidaan luottaa. Käyttäjätunnusten jako tehdään pyyntöjen mukaan ja ne vaativat hyväksynnän useammalta taholta. Jokaisen käyttäjän näkyvyys- ja käyttöoikeudet määritetään yksilöllisesti työtehtävien mukaan, jotta tietoturvaluottimukset täyttyvät.

Testi- ja koulutusympäristö toteutettiin kevennetyillä tietoturvaluottimuksilla, resurssien käytön tehostamiseksi ja ympäristö on mahdollista palauttaa alkutilanteeseen yksinkertaisin toimenpitein. Näin käyttöympäristö voidaan pitää puhtaana koulutus- ja testikäyttöä varten. Toinen tietoturvaluottava asia on uusien versiondog-päivitysten testaaminen tässä ympäristössä ennen asennusta varsinaiseen käyttöympäristöön, jotta käyttöympäristön toiminnallisuus voidaan taata myös päivityksen jälkeen. Testi- ja koulutusympäristön käyttäjätunnuksia luodaan pyynnöstä pääkäyttäjän toimesta, eikä erillistä hyväksymisprosessia vaadita.

Useiden ympäristöjen luonti vaatii useamman versiondog-lisenssin, joten hankinta- ja ylläpitokustannukset nousevat verrattuna yhden ympäristön käyttöön. Päätöksen pohjalla ovat usein yrityksen tarpeet ja tallennettavan tiedon merkitys, käyttäjähallinnalla voidaan myös estää tahattomat pääsyt ja pitää tuotantoympäristön toimintavarmuus yllä ilman erillistä kokeilu- ja testiympäristöä.

## 6.8 **Versiondog:n kouluttaminen yrityksen sisällä**

Versiondog:n toiminta liittyy vahvasti konfiguraationhallintaan, joten versiondog:n esittely yhdistettiin osaksi konfiguraationhallintakoulutusta. Yleisesittely käsitteli lyhyesti versiondog:n taustoja, valintaperusteita ja taustalla olevaa yritystä. Samassa yhteydessä kerrottiin, miten versiondog yhdistyy jo olemassa oleviin toimintamalleihin, mitä ohjeistusta versiondog:iin liittyy ja ketkä ovat vastuuhenkilöt eri toimintoaloilla.

Versiondog:n varsinainen koulutus koostuu kahdesta eri osasta, ensimmäinen osa koskee kaikkia versiondog:n käyttäjiä. Ensimmäisessä osiossa luodaan käyttäjätunnukset ja käydään läpi eri ohjelmien avaaminen ja käyttö pintapuolisesti. Pääpaino on UserClient:n käytöllä ja komponenttien uloskirjaamisella tutkimista varten. Tämän koulutuksen tarkoituksena on luoda käyttäjälle perusedellytykset versiondog:n käyttöön ja tallennettujen ohjelmien lukemiseen. Koulutus on suunnattu peruskäyttäjälle, joiden ei tarvitse tehdä muutoksia laitteiden ohjelmiin.

Koulutuksen toinen osio on suunnattu käyttäjille, joiden työtehtäviin kuuluu myös ohjelmien muutokset ja takaisinventi versiondog:iin. Koulutuksessa käydään läpi uuden komponentin luonti, komponentin uloskuittaus, muokkaus ja takaisinventi. Lisäksi luodaan uusi versio komponentista ja testataan vertailuominaisuuden käyttöä sekä versiondog:ssa valmiina olevaan ohjelmaa vasten että ulkopuolisella medially tuotua ohjelmaa vasten. Tämän koulutuksen lopussa koulutettavat tekevät itselleen kansiot, kopioivat olemassa olevan komponentin kansioon ja luovat siitä uuden version.

Näiden kahden koulutuksen lisäksi yrityksessä järjestetään osastoittain koulutuksia, jotka keskittyvät osaston tarvitsemiin toimintoihin. Koulutuksessa käydään läpi todellisia tilanteita, joissa edellytetään versiondog:n käyttöä. Osastoilla on nimetyt vastuuhenkilöt, jotka toimivat kouluttajina ja heillä on riittävät oikeudet koulutus- ja testiympäristössä uusien käyttäjien luomiseen. He myös osallistuvat aktiivisesti ohjeiden luontiin ja päivitykseen sekä toimintamallien päivittämiseen.

Koulutusten merkitys varsinkin versiondog:n alussa on suuri ja mahdollistaa versiondog:in tehokkaan käytön osana yrityksen muita toimintamalleja. Myös uusien ominaisuuksien koulutus ja kertaukset ovat osa koulutussuunnitelmaa. Suunnitelmaa tulee päivittää tarpeen mukaan. Myös uusien työntekijöiden koulutus tulee ottaa huomioon koulutussuunnittelussa ja osana uusien työntekijöiden perehdytysuunnitelmaa.

## 7. YHTEENVETO

Tämän diplomityön tavoitteena oli esitellä ohjelmoitavien komponenttien versionhallintaa teollisuusympäristössä. Työssä käytiin läpi aikaisempia tutkimuksia, joita ei kuitenkaan juurikaan ole samasta aiheesta tehty. Aihe koettiin ajankohtaiseksi koska ohjelmoitavien komponenttien käyttö on yleistynyt teollisuudessa viimeisten vuosikymmenien aikana. Ohjelmoitavilla komponenteilla on korvattu monimutkaiset relepohjaiset kytkenät ja ne ovat korvanneet myös kovalangoitettuja turvapiirejä monin paikoin. Komponenttien ohjelmistojen versionhallinta eroaa perinteisten dokumenttien versionhallinnasta niin vaatimusten kuin tietojärjestelmien osalta. Suurin osa ohjelmistojen versionhallintaohjelmistoista on tarkoitettu rivipohjaisen ohjelmien lähdekoodin hallintaan, joka ei kaikin osin sovellu suoraan laitteiden ohjelmien ja niihin liittyvien projektien versionhallintaan.

Yhtenä taustamateriaalina käytettiin Version Control and Patch Management of Protection and Automation Systems julkaisua, joka käsittelee suojareiden ohjelmistopäivityksiä releiden elinkaaren aikana. Julkaisussa mainitaan erityisesti ohjelmistopäivitysten aiheuttamat riskit ja niiden hallinta, mutta aihealue on tässä työssä laajennettu käsittämään kaikkia ohjelmoitavia laitteita. Muita taustamateriaaleina käytettiin ohjelmistoprojekteihin liittyviä julkaisuita, jotka käsittelevät projektien onnistumista ja takaisinmaksuaikaa. Aiheeseen ei ollut suoranaista taustamateriaalia, joten työssä on käytetty soveltuvin osin dokumentaation versionhallintaan liittyvää materiaalia.

Ohjelmoitavien komponenttien versionhallintaan liittyvää lainsäädäntöä ei ole julkaistu, mutta esimerkiksi STUK:n julkaisemat YVL-ohjeet velvoittavat epäsuorasti luvanhaltijan ylläpitämään laitteisiin liittyvää versiohistoriaa. Myös IAEA ottaa samaan asiaan kantaa omassa julkaisussaan, joten voimme olettaa että tulevaisuudessa ohjelmoitavien komponenttien vaatimukset tullaan mainitsemaan suoraan myös YVL-ohjeissa. Laadunhallinnan standardin ISO 9001 asettamat veloitteet ja vaatimukset voidaan täyttää jäljitettävyyden ja tunnistettavuuden osalta standardilla ISO 10007, jossa käsitellään konguraationhallintaa.

Taustamateriaalien ja versionhallintaan liittyvien standardien ja veloitteiden pohjalta luotiin soveltuvilta osin ohjelmoitavien komponenttien versionhallintaan liittyvä määritelmä ja määrittelydokumentaatio. Dokumentaatio käsittelee tämän lisäksi

tarvittavien lähtötietojen määrittelyä, versionhallinnassa olevien tietojen määrittelyä ja tallennettavien tietojen vastaanottoa. Sanallisten kuvauksien lisäksi työssä esitetään useat dokumentaatioprosessit kuvaajina helpottamaan lukemista. Prosessit esiteltiin yleisellä tasolla, joten jokainen prosessi tulee muokata yrityskohtaisiksi ennen käyttöä. Kuvaukset pyrittiin luomaan käytännönläheisiksi ja ottamaan huomioon ohjelmoitavien komponenttien versionhallinnan erityistarpeet. Ja tämän lisäksi määrittelyyn vaikuttaa suuresti yrityksen toimintaympäristö ja sisäiset vaatimukset, joilla puolestaan on suuri vaikutus ohjelmiston valintaan.

Versionhallinnan käyttöönotossa tulee ottaa huomioon käyttäjäryhmät sekä erityisesti tietoverkkojen ratkaisut, jotka vaikuttavat käytännön toteutukseen. Tämän lisäksi etukäteisselvityksen ja testikäytön suunnitelun merkitystä käyttöönotossa ei tule väheksyä. Onnistuneen käyttöönoton pohjana on riittävän tarkka dokumentaatio ja resurssien riittävä varaaminen, jotta vältetään perinteiset käyttöönoton ongelmat. Näitä ongelmia ovat esimerkiksi aikataulujen myöhästymisen ja asetettujen tavoitteiden eroaminen lopputuloksesta. Myös tietoturvaan liittyvät asiat on tärkeää ratkaista käyttöönoton alussa, jotta käyttöönottoa ei jouduta keskeyttämään jos vaatimukset eivät täyty.

Käyttöönottoprosessiin liittyy oleellisesti myös ohjelmoitavien komponenttien tuontiin liittyvä dokumentaatio, joka kuvaa mitä alkutietoja komponenteista tarvitaan ja miten vienti versionhallintaan tehdään. Lisäksi käyttöönoton yhteydessä luodaan tarvittavat käyttäjät ja käyttäjäryhmät. Käyttöönoton hallitsemiseksi on hyvä muodostaa pieni ryhmä, jossa on jäseniä yrityksen eri käyttäjäryhmistä, jotta kaikki versionhallintaa tarvitsevat käyttäjäryhmät ovat edustettuina. Jokaisella käyttäjäryhmällä on omat erityistarpeensa ja toimintamallinsa, jotka tulee ottaa huomioon versionhallinnassa. Käyttöönoton aikana tulee ylläpitää määrittelydokumentaatiota ja varmistua, että vaatimukset täyttyvät, jotta ohjelmiston lopullinen käyttö on mahdollista. Käyttöönotossa valittu versionhallintajärjestelmä liitetään osaksi yrityksen muita tietojärjestelmiä ja käyttöönoton rinnalla luodaan tarpeellinen ohjeistus ja liitetään versionhallinta osaksi yrityksen aikaisempia toimintamalleja.

Esimerkkihjelmistona käytettiin versiondog-ohjelmistoa, joka on tarkoitettu versionhallintaohjelmistoksi teollisuusympäristöön. Ohjelmisto tukee yleisempiä teollisuudessa käytettyjä komponentteja ja mahdollistaa eri versioiden välisten erojen esittämisen kuten ohjelma on alkuperäisesti esitetty. Ohjelmassa luodaan komponentteja, jotka voivat sisältää yksittäisen tiedoston tai kokonaisen projektin. Komponentille määritetään komponenttityyppi, jonka perusteella ohjelma tekee vertailun ja eroavaisuuksien esittelyn. Ohjelmisto koostuu kuudesta ohjelmasta ja tämän lisäksi on kymmenen erikseen hankittavissa olevaa lisäosaa. Ohjelmista kaksi on tarkoitettu versionhallinnan päivittäiseen käyttöön ja tämän lisäksi on verkkopohjainen hallinta. Raportoinnille, asetusten tekemiseen ja varmuuskopioiden hallintaan

ohjelmistossa on omat ohjelmansa. Ohjelmisto toimii asiakas-palvelin periaatteella, palvelimen tietokannasta tehdään paikalliskopio muokattavasta komponentista, johon tehdään muutokset ja päivitetty versio viedään takaisin palvelimelle. Viennin yhteydessä tehdään vertailu ja luodaan uusi versio komponentista.

Versionhallintaohjelmiston käyttöönottoprojektissa tulee ottaa huomioon projektin laajuus ja vaativuus. Kannattavan lopputuloksen saavuttaminen vaatii laajan taustatyön, jotta asetetut tavoitteet saavutetaan ja käyttöönottoprojekti onnistuu annettujen resurssien rajoissa. Yhtenä ongelmakohtana on ohjelmitavien komponenttien versionhallinnan ympäristön kompleksisuus, joka tulee ottaa huomioon lyhyessä ajassa taloudellisesti kannattavan lopputuloksen aikaansaamiseksi.

## LÄHTEET

- [1] Dylan Jenkins, Jerome Arnaud, Stephen Thompson, Matthew Yau, John Wright, Version Control and Patch Management of Protection and Automation Systems, IET konferenssi julkaisu, maaliskuu 2014
- [2] Outi Aronen, Tietojärjestelmän käyttöönotto ja sen arviointi, diplomityö 2010, Tampereen teknillinen yliopisto
- [3] John Heckman, Why Document Management: a White Paper, Syyskuu 2013. [Viitattu: 28.5.2018] <http://www.heckmanco.com/docs/DMWhitePaper.pdf>
- [4] Säteilyturvakeskus, Ydinlaitoksen ikääntymisen hallinta, 20.5.2014. [Viitattu: 17.8.2018] <https://www.stuklex.fi/fi/ohje/YVLA-8>
- [5] SSG-39 International Atomic Energy Agency, Design of Instrumentation and Control Systems for Nuclear Power Plants, Vienna, 2016, [Viitattu: 27.6.2018] [https://www-pub.iaea.org/MTCD/Publications/PDF/Pub1694\\_web.pdf](https://www-pub.iaea.org/MTCD/Publications/PDF/Pub1694_web.pdf)
- [6] ISO 9001:2015, Quality management systems: Requirements, julkaistu 2015.
- [7] ISO 10007:2017. Quality management – Guidelines for configuration management, julkaistu 2017
- [8] Lars Bendix, Lorenzo Borraccio, Towards a Suite of Software Configuration Management Metrics, Lund Institute of Technology, syyskuu 2005
- [9] Donald J. Reifer, Software Management, 7th ed., New Yor, John Wiley & Sons Inc, 2006
- [10] Ali Koc, Abdullah Uz Tansel, A Survey of Version Control Systems, artikkeli 2011
- [11] Viestintävirasto, Lokien keräys ja käyttö, 2006. [Viitattu:18.8.2018] <https://www.viestintavirasto.fi/attachments/tietoturva/Lokitusohje.pdf>
- [12] ISO/IEC TR24748-2:2011, Systems and software engineering Life cycle management Part 2: Guide to the application of ISO/IEC 15288 (System life cycle processes), 1st ed., julkaistu 2011.
- [13] IEC 61508-3:2010, Functional safety of electrical/electronic/programmable electronic safety-related systems - Part 3: Software requirements, julkaistu 2010.

- [14] IEC 61131-1:2003, CEN, Programmable controllers - Part 1: General information, julkaistu 2003.
- [15] Reference Document SQAS20.01.00 - 2000, Department of Energy Quality Managers Software Quality Assurance Subcommittee, Software Configuration Management (SCM) A Practical Guide , julkaisu 2000. [Viitattu: 18.8.2018] <https://www.energy.gov/sites/prod/files/cioprod/documents/scmguide.pdf>
- [16] IEC 61508-1:2010, Functional safety of electrical/electronic/programmable electronic safety-related systems. Part 1: General requirements, julkaistu 2010.
- [17] IEC 82045:2005, Dokumenttien hallinta. Osa 2: Metadataelementit ja informaation viitemalli, julkaistu 2005
- [18] IEC 61140:2016, Protection against electric shock: Common aspects for installation and equipment, julkaistu 2016
- [19] IEC 62061:2005, Safety of machinery. Functional safety of safety-related electrical, electronic and programmable electronic control systems, julkaistu 2005
- [20] YVL B.2, Säteilyturvakeskus, Ydinlaitosten järjestelmien, rakenteiden ja laitteiden luokittelu, julkaistu 2013
- [21] ISO 13849-1:2015, Safety of machinery – Safety-related parts of control systems – Part 1: General principles for design, julkaistu 2015
- [22] EN 82045-1:2001, Document management – Part 1: Principles and methods, julkaistu 2001
- [23] IEC 61131-3:2013, Programmable controllers, part 3: Programming languages, julkaistu 2013
- [24] IEC 62708:2015 Document kinds for Electrical and Instrumentation Projects in the Process Industry, julkaistu 2015
- [25] Walford, Business Process Implementation for IT Professionals and Manager, Artech House, 1999
- [26] Turban, McLean, Wetherbe, Information Technology for management: Transforming business in the digital economy, John Wiley & Sons, 2002
- [27] Samu Syvänen, KKS-koodausjärjestelmän soveltaminen Alfa Laval Aalborg Oy:n PI-kaavioihin, opinnäytetyö 2014, Satakunnan ammattikorkeakoulu
- [28] ISO 4157-1:1998, Construction drawings. Designation systems. Part 1: Buildings and parts of buildings, julkaistu 1998

- [29] ISO 4157-2:1998, Construction drawings. Designation systems. Part 2: Room names and numbers, julkaistu 1998
- [30] ISO 4157-3:1998, Construction drawings. Designation systems. Part 3: Room identifiers, julkaistu 1998
- [31] IEC 81346-1:2009, Industrial systems, installations and equipment and industrial products - Structuring principles and reference designations - Part 1: Basic rules, julkaistu 2009
- [32] ISO 21503:2017, Projektin-, ohjelman- ja salkunhallinta. Ohjeita ohjelmanhallinnasta, julkaistu 2017
- [33] Active Directory Domain Services Overview, Microsoft 2017, [Viitattu: 20.9.2018] <https://docs.microsoft.com/en-us/windows-server/identity/ad-ds/get-started/virtual-dc/active-directory-domain-services-overview>
- [34] Introduction to XML, Doug Tidwell, IBM 2002, [Viitattu: 20.9.2018] <https://www.ibm.com/developerworks/xml/tutorials/xmlintro/xmlintro.html>
- [35] ISO 15489-1:2017, Tieto ja dokumentointi. Asiakirjahallinto. Osa 1: Käsitteet ja periaatteet, julkaistu 2017
- [36] AUVESY, referenssit, [Viitattu: 28.9.2018] [https://www.versiondog.com/versiondog\\_customers.html](https://www.versiondog.com/versiondog_customers.html)