Tampereen ammattikorkeakoulu

Geneerinen prosessimalli sovellusten välisen datamigraatioprojektin toteutukseen

Reijo Huotari

OPINNÄYTETYÖ
Tammikuu 2021

Tietojärjestelmäosaamisen ylempi tutkinto-ohjelma

TIIVISTELMÄ

Tampereen ammattikorkeakoulu
Tietojärjestelmäosaamisen ylempi tutkinto-ohjelma

HUOTARI, REIJO:
Geneerinen prosessimalli sovellusten välisen datamigraatioprojektin toteutukseen

Opinnäytetyö 57 sivua, joista liitteitä 7 sivua
Tammikuu 2021

Datamigraation merkitys IT-teknologiassa on kasvamassa, koska tekniikan kehittyessä on tarve muuttaa vanhoja tietojärjestelmiä hyödyntämään uuden teknologian tarjoamia ominaisuuksia. Vanhojen tietojärjestelmien modernisointi on yksi haastavimmista ongelmista ohjelmistoteknologian alalla. Varsin vaikeaksi tehtäväksi on tunnustettu vanhojen sovellusten sisältämän datan siirtäminen uudella teknologisella alustalla toimiviin nykyaikaisiin sovelluksiin. Vain pieni osa migraatioprojekteista saadaan vietyä loppuun suunnitellussa aikataulussa ja budjetissa. Suurin syy aikataulun tai budjetin ylittymiseen on se, että puuttuu valmis menetelmä, jonka avulla voidaan hallita migraatiotehtävän monimutkaisuutta.

Opinnäytetyön tarkoituksena oli perehtyä sovellusten väliseen datamigraation ja kuvata yleiskäyttöinen prosessi, jota voidaan käyttää sellaisenaan tai muokattuna sovellusten välisissä migraatioprojekteissa. Opinnäytetyön tavoitteena oli kehittää menetelmä toimeksiantajan, Visma Consulting Oy:n, migraatioprojektien suunnitteluun ja toteutukseen. Menetelmässä kuvataan, mitä kaikkia toimenpiteitä datalle pitää tehdä, jotta se on mahdollista luotettavasti siirtää sovelluksesta toiseen ilman hävikkiä ja datan vääristymistä.

Datamigraatio on kertaluonteinen operaatio, jossa tietoa siirretään yhdestä tietojärjestelmästä toiseen. Lähde- ja kohdejärjestelmien arkkitehtuuri ja tietomalli poikkeavat yleensä toisistaan, joten datan lataaminen kohdejärjestelmään vaatii datan muokkaamista, rikastamista tai konvertointia esitysmuodosta toiseen, ennen kuin se voidaan ladata kohdejärjestelmään. Migraatioprosessissa on huolehdittava siirrettävän tiedon eheydestä eli tieto ei saa vääristyä eikä mitään saa hävitä.

Opinnäytetyössä tehdyn kirjallisen tutkimuksen mukaan migraatioprojekti voidaan jakaa neljään vaiheeseen: esiselvitys, data-analyysi, kehitystyö ja varsinainen migraatio. Varsinainen migraatio sisältää datan purkamisen lähdejärjestelmästä, muokkaamisen kohdejärjestelmän vaatimaan muotoon sekä lataamisen kohdejärjestelmään. Esiselvitysvaihe on kriittisen tärkeä, koska siinä päätetään migraatioprosessin laajuus ja sisältö. Kaikki myöhemmät vaiheet ja suunnitelmat perustuvat esiselvitysvaiheessa tehtyihin päätöksiin.

Asiasanat: data, migraatio, prosessi, tietojärjestelmä

ABSTRACT

Tampere University of Applied Sciences
Master's Degree Programme in Information Systems Competence

HUOTARI, REIJO:
A Generic Process Model for Data Migration Projects between Information Systems

Master's thesis 57 pages, appendices 7 pages
January 2021

The importance of data migration in IT technology is growing, because as technology develops, there is a need to enable the old information systems to take advantage of the features offered by the new technology. The modernization of old information systems is one of the most challenging problems in the field of software technology. One of the most difficult tasks is to transfer the data from old applications to modern applications which uses a new technological platform and new functions. Only a small number of the migration projects can be completed within the planned schedule and budget. The main reason for exceeding the schedule or budget is the lack of a ready-made methodology to manage the complexity of the migration task.

The purpose of the thesis was to study data migration between applications, and to describe a general-purpose process that can be used as such or modified in inter-application migration projects. The aim of the thesis was to develop a method for planning and implementing migration projects for the client, Visma Consulting Oy. The method describes what all steps need to be taken with the data in order to be able to reliably transfer it from one application to another without loss and data distortion.

The findings indicate that the migration project can be divided at into four phases: pre-feasibility study, data analysis, development work and the actual migration. The actual migration involves extracting data from the source system, modifying it to the format required by the target system, and uploading it to the target system. The pre-feasibility study phase is critical because it decides the scope and content of the migration process. All subsequent steps and plans are based on the decisions made in the pre-feasibility study phase.

Key words: information systems, data, migration, process

SISÄLLYS

LYHENTEET JA TERMIT

Datamigraatio	Kertaluonteinen operaatio, jossa tietoa siirretään yhdestä tietojärjestelmästä toiseen
ETL	Datan purkamista (Extract), muokkaamista (Transform) ja lataamista (Load) kuvaava prosessi
Rikastaminen	Puuttuvan metatiedon lisääminen kohdejärjestelmään ladattavaan dataan
Kohdistaminen	Lähdejärjestelmän metatietojen siirtäminen kohdejärjestelmän metatietomalliin
Konversio	Metatiedon tai asiakirjan muuntaminen muodosta toiseen
Puhdistaminen	Puutteellisen tai virheellisen metatiedon poistaminen tai korjaaminen
XHTML	Merkintäkieli, jolla kuvataan tekstin rakennetta ja esitystapaa.
JSON	Avoimen standardin tiedostomuoto tiedonvälitykseen
PDF/A	Asiakirjojen pitkäaikaissäilytykseen tarkoitettu tiedostomuoto
SFTP	Tiedostosiirtoprotokolla, joka tarjoaa tietoturvallisen tavan siirtää tiedostoja verkkoyhteyden yli
PGP	Ohjelmisto, jolla tiedostoja ja viestejä voi salata, purkaa salaus sekä liittää aineistoon digitaalinen allekirjoitus
CDA R2	Standardoitu rakenne terveydenhuollon asiakirjojen tallentamiseen
BASE64	Binäärisen datan esittäminen tiivistetyssä muodossa ascii-merkkeinä
Kanta-palvelut	Kelan tarjoamia sosiaali- ja terveydenhuollon digitaalisia palveluja

1 JOHDANTO

1.1 Työn tausta

Datamigraation merkitys IT-teknologiassa on kasvamassa, koska tekniikan kehittyessä on tarve muuttaa vanhoja tietojärjestelmiä hyödyntämään uuden teknologian tarjoamia ominaisuuksia. Uudet tietojärjestelmät sisältävät paljon työskentelyä ja ylläpitoa tehostavia ominaisuuksia verrattuna vanhalla teknologialla toteutettuihin järjestelmiin. (Pund & Jain 2016, 691.)

Syitä datamigraation tekemiseen on muitakin kuin teknologinen kehitys, esimerkiksi

- Organisaatioiden yhdistyminen yrityskaupan tai jonkin muun syyn takia, jonka seurauksena organisaatioiden tietojärjestelmiä yhtenäistetään (Fhl & Schulz 2011, 438)
- Yrityksen sisäisten tai ulkoisten toimintatapojen (prosessien) muuttaminen, jonka seurauksena myös tietojärjestelmiä joudutaan muuttamaan (Matthes & Schulz 2011, 438)
- Teknologinen muutos, jossa vanhaa teknologiaa edustava ohjelmisto tai ajoalusta korvataan uudemmalla teknologialla (Sarmah, 2018, 1)
- Lainsäädännön vaatimat toimenpiteet, kuten esimerkiksi julkishallinnon ja terveydenhuollon asiakirjojen arkistointivaatimus
- Strategian muutos, jonka vuoksi siirrytään asiakaskohtaisesta ratkaisusta pilvipalveluun
- Otetaan käyttöön uusi tietojärjestelmä ja järjestelmän käyttöönoton yhteydessä siirretään dataa jostain olemassa olevasta järjestelmästä. Esimerkkinä tällaisesta tapauksesta voisi mainita analytiikkaohjelmiston tai sähköisen arkiston käyttöönotto, johon käyttöönoton yhteydessä siirretään lähdejärjestelmästä kaikki data. Myöhemmässä vaiheessa järjestelmään voidaan siirtää vain muuttuneet tiedot eli näissä tapauksessa online-integraatiossa hyödynnetään aikaisemmin tehtyä migraatiotoiminnallisuutta.

Vanhojen tietojärjestelmien modernisointi on yksi haastavimmista ongelmista ohjelmistoteknologian alalla. Varsin vaikeaksi tehtäväksi on tunnustettu vanhojen sovellusten sisältämän datan siirtäminen (migraatio) uudella teknologisella alustalla oleviin ja uusia toimintoja sisältäviin nykyaikaisiin sovelluksiin. Tutkimusten mukaan vain 16 % migraatioprojekteista saadaan vietyä loppuun suunnitellussa aikataulussa ja budjetissa. Projekteista 64 % valmistui myöhässä ja 37 % ylitti budjetin. Suurin syy aikataulun tai budjetin ylittymiseen oli valmiin metodologian puuttuminen, jonka avulla olisi voitu hallita migraatiotehtävän monimutkaisuutta. (Thalheim & Wang 2012, 260.)

Metodologian puuttumisen lisäksi migraatioprojektien toteuttamista vaikeuttaa se, että tietojärjestelmät voivat olla hyvin monimutkaisia sisältäen monia erityyppisiä tietolähteitä. Lisäksi tietojärjestelmien sisältämä tieto voi olla epätarkkaa, puutteellista, päällekkäistä tai epäjohdonmukaista. Varsinkin vanhojen tietojärjestelmien sisältämien tietorakenteiden muuttaminen uusien tietojärjestelmien vaatimalle tasolle voi olla hyvin kallista ja aikaa vievää. Tutkimuksissa on havaittu, että jopa 62 % migraatioprojektissa mukana olleissa tietojärjestelmissä on laatuongelmia siirretyssä tiedoissa. Monet migraatioprojekteissa suoritettavat tehtävät kuten tietojen profilointi, validointi ja puhdistus suoritetaan iteratiivisesti ja hyvin monesti alkuperäisiin määrittelyihin joudutaan tekemään muutoksia. Arvioiden mukaan 90 % alkuperäisistä määrittelyistä muuttuu ja yli 25 % määrittelyistä muuttuu useammin kuin kerran migraatioprojektin aikana. Nämä seikat korostavatkin sellaisten menetelmien ja parhaiden käytäntöjen merkitystä, joita voidaan käyttää migraatioprojektin suunnitteluun ja suorittamisen. (Thalheim & Wang 2012, 260.)

Datamigraation on voitu ajatella olevan vain datan tekninen kopiointi vanhasta järjestelmästä uuteen ilman, että sitä joudutaan muuttamaan. Näin ei yleensä ole vaan datamigraatio voi olla hyvinkin monimutkainen operaatio, jossa lähdejärjestelmässä olevaa dataa voidaan joutua muokkaamaan sekä käsitteellisesti, että teknisesti ennen kuin se saadaan vietyä kohdejärjestelmään. Datamigraatio on yleensä osa suurempaa järjestelmäuudistusprojektia, jolloin koko projektin onnistuminen riippuu datamigraation onnistumisesta. (Thalheim & Wang 2012, 261.)

1.2 Tutkimuksen tavoite ja tarkoitus

Opinnäytetyö käsittelee datamigraatiota eri sovellusten välillä. Datamigraatiossa tietoa ei pelkästään siirretä muuttumattomana järjestelmästä toiseen, vaan sitä pitää mahdollisesti rikastaa tai muuttaa esitysmuodosta toiseen ennen kuin data voidaan ladata vastaanottavaan järjestelmään. Migraatioprosessissa on huolehdittava siirrettävän tiedon eheydestä eli tieto ei saa vääristyä eikä mitään saa hävitä. Lisäksi on otettava huomioon tietoturvavaatimukset erityisesti siirrettäessä sensitiivistä, esimerkiksi ihmisten terveyteen ja hyvinvointiin liittyvä tietoa. Migraatioprosessi eli tietojen poiminta lähdejärjestelmästä, siirrot palvelimilta toiselle, tietojen muokkaus ja lataus vastaanottavaan järjestelmään pyritään automatisoimaan mahdollisimman pitkälle siten, että operaatiota suorittavien henkilöiden ei tarvitse puuttua siihen suorituksen aikana esimerkiksi käynnistämällä ajoja tai manuaalisesti kopioimalla tiedostoja paikasta toiseen.

Opinnäytetyön tarkoituksena on perehtyä sovellusten väliseen datamigraation ja kuvata yleiskäyttöinen prosessi, jota voidaan käyttää sellaisenaan tai muokattuna sovellusten välisissä migraatioprojekteissa. Prosessissa kuvataan mitä kaikkia toimenpiteitä datalle pitää tehdä, jotta se on mahdollista luotettavasti siirtää sovelluksesta toiseen ilman hävikkiä ja datan vääristymistä.

Opinnäytetyön tavoitteena on kehittää menetelmä toimeksiantajan, Visma Consulting Oy:n, migraatioprojektien suunnitteluun ja läpivientiin sekä sitä hyödyntäen laajentaa toimeksiantajan käytössä olevaa nykyistä Kanta-välityspalvelua kattamaan eri muodoissa olevan lähdeaineiston transformaation Kanta-palveluun sopivaksi. Vienti Kanta-palveluun oli välityspalvelussa valmiina, joten sen toteuttaminen tai mahdollinen muuttaminen ei kuulunut tähän työhön. Lisäksi datan purkamisen lähdejärjestelmästä toteutti toinen toimittaja, joten sitäkään ei tässä työssä kuvata.

Opinnäytetyön tietoperustana on käytetty olemassa olevaa tutkimustietoa sovellusten välisestä datamigraatioista sekä toimeksiantajan migraatioprojektien käytäntöjä ja teknologioita. Tutkimusmenetelminä olivat kirjallinen tutkimus sekä Kanta-välityspalvelun laajennuksen käytännön toteutus.

2 DATAMIGRAATION TEORIAA

2.1 Mitä on datamigraatio

Datamigraatio on kertaluonteinen operaatio, jossa tietoa siirretään yhdestä tietojärjestelmästä toiseen. Lähde- ja kohdejärjestelmien arkkitehtuuri ja tietomalli poikkeavat yleensä toisistaan, joten datan lataaminen kohdejärjestelmään vaatii datan muokkaamista ennen kuin se voidaan ladata kohdejärjestelmään. (Pund & Jain 2018, 690.)

Lähde- ja kohdejärjestelmien sisältämä data voi erota sekä käsitteellisellä että teknisellä tasolla (Matthes & Schulz 2001, 438). Tieto voidaan siirtää järjestelmien välillä sellaisenaan muuttamatta sitä, mikäli kyseessä on migraatio saman sovelluksen vanhasta versiosta uudempaan tai sitten dataa voidaan joutua muokkaamaan kohdejärjestelmän vaatimaan muotoon, mikä yleensä on tilanne, jos dataa siirretään eri sovellusten välillä. Siirrettävää tieto voi olla pelkästään metatietoa tai rakenteisessa tai ei-rakenteisessa muodossa olevia asiakirjoja sekä niihin liittyvää metatietoa.

Karkealla tasolla migraatioprosessi voidaan jakaa kolmeen osaan: datan purkaminen lähdejärjestelmästä, datan muokkaaminen ja datan lataaminen kohdejärjestelmään. Prosessia kuvataan yleensä lyhenteellä ETL (Extract, Transform, Load). Termi kuvaa hyvin yleisellä tasolla prosessia, jossa tieto kerätään lähdejärjestelmistä, muokataan kohdejärjestelmän vaatimaan muotoon ja viedään kohdejärjestelmään.

Extract-vaiheessa haetaan tieto kohdejärjestelmistä sovitulla tavalla ja muodossa, sekä tallennetaan se sovittuun paikkaan jatkokäsittelyä varten. Lähdejärjestelmän sisältämä data voidaan purkaa kokonaisuudessaan tai sitten jo tässä vaiheessa voidaan jättää purkamatta sellainen data, jota ei ole tarkoitus viedä kohdejärjestelmään (Honkavaara 2013, 16). Extract vaiheessa dataa ei välttämättä muokata ollenkaan, vaan yleensä se puretaan lähdejärjestelmästä siinä muodossa kuin se on helpointa. Vaiheessa pitää kuitenkin varmistaa, että kaikki se data, mikä on tarkoitus siirtää lähdejärjestelmään, on saatu purettua.

Transform-vaiheessa tehdään datan muokkaus lähdejärjestelmän tuottamasta muodosta kohdejärjestelmän vaatimaan muotoon. Muokkaustoimenpiteet voivat olla hyvinkin monenlaisia riippuen siitä miten erilaisia lähde- ja kohdejärjestelmät ovat. Dataa voidaan esimerkiksi puhdistaa, rikastaa, pilkkoa osiin, yhdistellä tai konvertoida esitysmuodosta toiseen.

Load-vaiheessa muokattu data ladataan kohdejärjestelmään (Honkavaara 2016, 17). Kohdejärjestelmä voi olla tietokanta, ohjelmisto tai fyysinen tallennuslaite. Lataus voidaan tehdä suoraan tietokantaan tai sovelluksen ollessa kyseessä sen tarjoamien rajapintojen kautta tai ohjelmistorobotiikkaa hyödyntäen käyttöliittymän kautta.

Koko migraatioprosessin kesto voi olla tapauksesta riippuen muutamasta tunnista jopa kuukausiin. Suurten migraatioiden kyseessä ollessa prosessi pitäisi suunnitella sellaiseksi, että ongelmien sattuessa se voidaan tarvittaessa keskeyttää ja ongelmien ratkettua jatkaa keskeytyneestä kohdasta. Kaikki ETL-prosessit pitää kuitenkin suunnitella aina tapauskohtaisesti ja yleensä kolmeosainen ETL-prosessi voidaan jakaa paljon useampaan osaan.

2.2 Datamigraation tyypit

Sovellusten sisältämää tietoa säilytetään erilaisissa medioissa tiedostoissa ja tietokannoissa. Sovellukset taas ovat osa yritysten ja yhteisöjen liiketoimintaprosesseja. Kun yritykset haluavat kehittää toimintojaan, niin hyvin monesti se vaatii muutoksia myös yritysten liiketoimintaprosesseihin ja niiden käyttämiin sovelluksiin. Muutokset voivat olla pieniä ja vaativat ehkä vain tietojen siirtämistä medialta toiselle tai sovelluksesta toiseen. Toisinaan muutokset voivat olla hyvinkin suuria, joissa liiketoimintaprosesseja muokataan paremmin sopivaksi yrityksen uuteen toimintamalliin. Myös datamigraatioita on erityyppisiä. Accelarion (2020) mukaan datamigraatiot voidaan jakaa neljään tyyppiin.

- **Tietokantamigraatio**, jossa data siirretään saman tietokantasovelluksen eri instanssien välillä tai eri tietokantasovellusten välillä. Tässä siirrossa

dataan ei yleensä tehdä muita muutoksia, kun korkeintaan vastaanottavan järjestelmän toiminnan kannalta oleellisia teknisiä muutoksia.

- **Datavaraston migraatio**, jossa data siirretään fyysiseltä tallennuslaitteelta toiselle. Tämä migraatiotyyppi ei yleensä sisällä tai sisältää hyvin vähän datan rakenteellisia muutoksia. Syynä datavaraston migraatioon voi olla esimerkin tallennuslaitteella olevan aineiston säilyttäminen käyttökuntoisena ja data siirretään vanhentuneelta medialta uudempaan. Puhutaan aineiston virkistämisestä.

- **Sovellusmigraatio**, jossa dataa siirretään saman sovelluksen eri versioiden välillä tai eri sovellustoimittajien samassa käyttötarkoituksessa olevien sovellusten välillä, jos organisaatio on vaihtamassa sovellustoimittajaa. Organisaatiossa voidaan myös ottaa käyttöön uusi sovellus, esimerkiksi analytiikkaohjelmisto tai sähköinen arkisto, johon operatiivisen järjestelmän sisältämä data siirretään sovelluksen käyttöönottovaiheessa. Sovellusmigraatiossa dataa voidaan joutua muokkaamaan hyvinkin paljon, jotta se saadaan vastaanottavan järjestelmän vaatimaan muotoon.

- **Liiketoimintaprosessien migraatio**, jossa organisaation olemassa olevia prosesseja muokataan, otetaan käyttöön uusi prosessi tai kahden eri organisaation prosesseja sovitetaan yhteen. Tämä on yleensä hyvin monimutkainen operaatio ja voi pitää sisällään kaikkia kolmea edellä mainittua migraatiotyyppiä.

2.3 Datamigraatioprojektien erityispiirteitä

Datamigraatioprojektit ovat kertaluonteisia tapahtumia. Niillä on selkeä alku- ja loppupäivä. Projektit vaativat erityisosaamista ja työvälineitä, joita yrityksillä ei yleensä ole, joten migraatioita tekevät niihin erikoistuneet yritykset ja henkilöt. Monesti, esimerkiksi sovellusten versiovaihdon yhteydessä, migraatioprojektit nähdään vain pakollisena kulueränä, koska sillä ei välttämättä kehitetä yrityksen toimintaa, vaan pelkästään turvataan sen jatkuminen vähintään samalla tasolla kuin ennen migraatiota. Näin ollen yritykset eivät välttämättä panosta riittävästi

migraatioprojektien suunnitteluun ja toteutukseen. Oletetaan, että se on vain yksinkertainen tekninen suoritus. (Matthes & Schulz, 2011.)

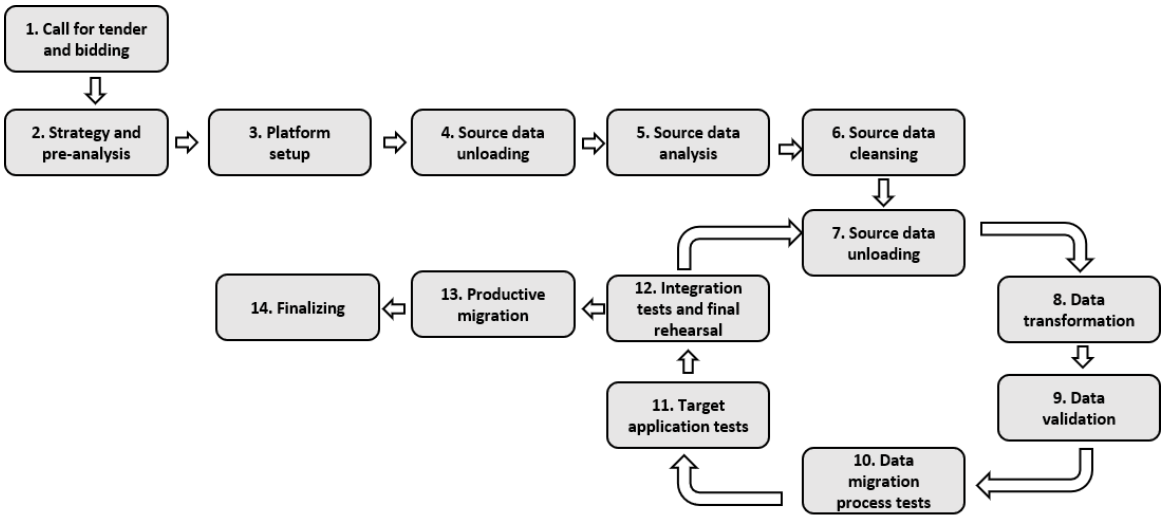
Data on yrityksen tärkeimpiä pääomia. Yritykset tuottavat koko ajan uutta dataa ja hyödyntävät toiminnassaan aikaisemmin tuotettua. Näin ollen kaikki muutokset ja operaatiot, jotka kohdistuvat dataan ovat yrityksen toiminnalle erityisen riskialtista. Parhaimmillaan kukaan ei huomaa mitään, jos tiedot siirretään onnistuneesti uuteen sovellusversioon. Pahimmillaan epäonnistunut migraatioprojekti voi lamauttaa koko yrityksen toiminnan.

3 DATAMIGRAATION TOTEUTUS

3.1 Migraatioprosessi

Tietojärjestelmät ovat nykyään monimutkaisia, joten migraatiot niiden välillä voivat myös olla hyvin monimutkaisia. Siksi ei voida antaa mitään yksityiskohtaista kaiken kattavaa prosessia, jolla voitaisiin hoitaa kaikki migraatiot. Voidaan kuitenkin kuvata yleinen prosessi, jota voidaan muokata kuhunkin migraatioprosessiin sopivaksi. (Eng, 7.)

Matthes ja Schulz (2011) ovat tutkineet datamigraatiota kirjallisuuslähteistä ja haastattelemalla monimutkaisissa datamigraatioprojekteissa mukana olleita ammattilaisia ja laatineet 14 vaiheisen migraatioprosessikuvauksen (KUVIO 1).



KUVIO 1. Migraatioprosessi (Matthes & Schulz 2011, 18)

Mallissa annetaan sääntöjä, ohjeita ja malleja, joita noudattamalla lähdejärjestelmän sisältämä data saadaan siirrettyä halutulla tavalla kohdejärjestelmään. Prosessin jokaisesta vaiheesta on kuvattu siinä suoritettavat tehtävät ja vaiheen tuottamat tuotokset.

Mallin osat 1-6 ovat migraation analyysi ja määrittelyvaihetta, jossa suunnitellaan projektia, perehdytään lähde- ja kohdejärjestelmään, valitaan migraatiostrategia,

käytettävä teknologia ja valmistellaan lähdejärjestelmän tuottamaan dataa migraatiota varten. Migraation tekninen toteutus ja testaus tehdään vaiheissa 7-12, joita käydään läpi niin monta kertaa, jotta päästään haluttuun lopputulokseen. Vaiheissa 13-14 tehdään varsinainen migraatio tuotannossa, sen validointi ja projektin hyväksyminen ja päättäminen.

Malli ei ole jäykkä ja ehdoton totuus, jonka mukaan migraatioprojektit pitäisi tehdä. Siinä olevia vaiheita voi jättää pois tai siihen voi lisätä uusia. Vaiheita tai niiden osia voidaan tehdä eri järjestyksessä kuin mitä mallissa on kuvattu. Esimerkiksi datan puhdistusta voidaan tehdä lähdejärjestelmässä ennen datan purkamista, purkamisvaiheessa, transformaatiovaiheessa, lataamisvaiheessa tai vasta kohdejärjestelmässä. Samoin osa migraatiossa käytetyistä komponenteista voidaan tarvittaessa tehdä jo ennen teknistä toteutus- ja testausvaihetta, mikäli esimerkiksi data purkamiseen tai puhdistamiseen ei ole olemassa tarvittavia työvälineitä.

Toisena esimerkkinä datamigraatioprosessista voidaan mainita Federal Student Aid -organisaation kehittämä datamigraation prosessimalli. Siinä prosessi on jaettuna neljään osaan: suunnittelu, analysointi, määrittely sekä toteutus ja sulkeminen (KUVIO 2).

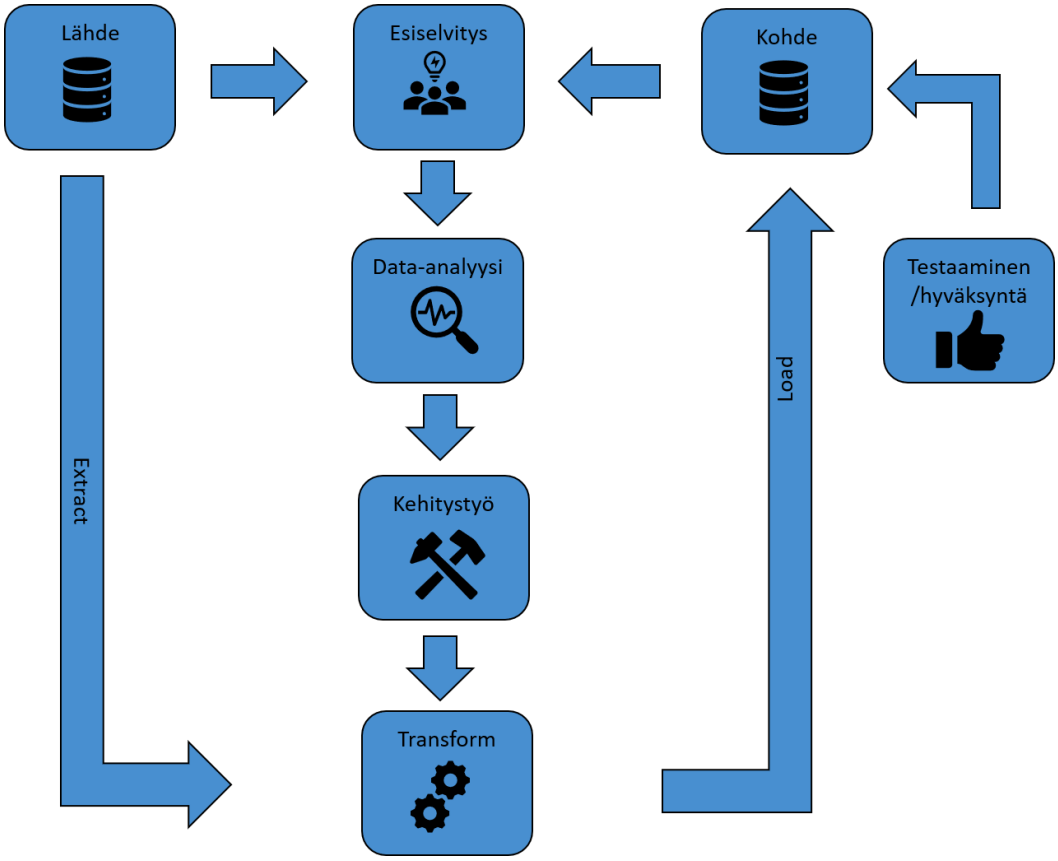
DATA MIGRATION PLANNING PHASE	DATA MIGRATION ANALYSIS & DESIGN PHASE	DATA MIGRATION IMPLEMENTATION PHASE	DATA MIGRATION CLOSEOUT PHASE
Plan Data Migration Project	Analyze Assessment Results	Develop Procedures	Document Data Migration Results
Determine Data Migration Requirements	Define Security Controls	Stage Data	Document Lessons Learned
Assess Current Environment	Design Data Environment	Cleanse Data	Perform Knowledge Transfer
Develop Data Migration Plan	Design Migration Procedures	Convert Transform Data (as needed)	Communicate Data Migration Results
Define and Assign Team Roles and Responsibilities	Validate Data Quality	Migrate Data (trial/deployment)	
		Validate Migration Results (iterative)	
		Validate Post-Migration Results	

KUVIO 2. Migraatioprojektin vaiheistus (Federal Student Aid 2019, 6-7).

Kuvassa on esitettynä kunkin vaiheen tehtävät. Kukin tehtävä jakautuu vielä useisiin alatehtäviin, joita en tässä työssä tarkemmin kuvaa. Mallissa on kuvattuna

yksityiskohtaisesti mitä kussakin alatehtävässä tehdään ja niissä syntyvät tuotokset. Lisäksi siinä on kuvattu mitä ongelmia ja riskejä migraatioprojektissa voi olla. Malli on sinänsä aivan käyttökelpoinen, mutta ehkä hiukan liian raskas käytettäväksi toimeksiantajani migraatioprojekteissa, jotka ovat pääsääntöisesti arkistointiprojekteja, joissa dataa siirretään erilaisista lähdejärjestelmistä toimeksiantajani sähköiseen arkistoon.

Kappaleessa 2.1 kuvatun ETL-prosessin ja edellä mainittujen migraatioprosessikuvasten pohjalta olen laatinut toimeksiantajani tarkoitukseen mielestäni sopivan version prosessista (KUVIO 3). Prosessi jakautuu neljään vaiheeseen: esiselvitys, data-analyysi, kehitystyö ja varsinainen migraatio. Kukin vaihe on kuvattuna tarkemmin seuraavissa kappaleissa.



KUVIO 3. Geneerinen migraatioprosessi.

3.2 Esiselvitys

3.2.1 Lähde- ja kohdejärjestelmien analysointi

Esiselvitysvaiheessa kartoitetaan lähdejärjestelmä(t) ja kohdejärjestelmä(t), joiden välillä migraatio toteutetaan. Selvitetään ja kuvataan lähde- ja kohdejärjestelmien metatiedot ja tietorakenteet. Tällä annetaan migraatiota suorittavalle projektille edellytykset ymmärtää miten järjestelmät toimivat ja miten migraatio niiden välillä olisi paras toteuttaa. Kartoitus on perusta työmäärän, riskien ja metatietojen kohdistukselle sekä muotokonversioiden ja rikastamisen tarpeelle. Vaiheessa tehdään alustava data-analyysi sekä liiketoiminta- että teknisellä tasolla. Selvitetään mitä ollaan siirtämässä ja miten se tehdään. Vaiheessa käsitellään tulevalle projektille asetettavia reunaehtoja organisatorisella, teknisellä, juridisella ja taloudellisella tasolla. Määrittelyvaiheen selvityksen perusteella voidaan myös päättää migraation toteutusstrategia. (Matthes & Schulz 2011, 19.)

Hyvin usein analysointivaihe kannattaa tehdä jo ennen kuin varsinaisen migraatioprojektin käynnistämistä. Analyysissa tehtyjen havaintojen perusteella voidaan kysyä tarjoukset migraatiopalveluita tarjoavilta yrityksiltä. Aikaisempi kokemus ja valmiit ohjelmistot kyseessä olevien sovellusten välisen migraation toteutukseen vähentää riskiä projektin epäonnistumisesta.

Lähde- ja kohdejärjestelmien analysointivaihe on kriittisen tärkeä, koska vaiheessa päätetään migraatioprosessin laajuus ja sisältö. Kaikki myöhemmät vaiheet ja suunnitelmat perustuvat tässä vaiheessa tehtyihin päätöksiin.

3.2.2 Migraatiostrategian valinta

Ennen kuin migraatioprojekti aletaan suunnitella yksityiskohtaisesti, niin päätetään migraatiostrategiasta eli miten migraatio tehdään. Voidaanko vai pitääkö kaikki lähdejärjestelmän sisältämä data siirtää samalla kertaa vai onko migraatio

mahdollista tehdä useammassa erässä? Suoritustavan mukaan migraatiot voidaan jakaa kahteen eri tyyppiin, joista toisessa tehdään koko migraatio yhdellä kertaa, toisessa se tehdään osissa (Oracle 2011, 3). Mikäli migraatiostrategiaa ei ole valittu ennen migraatioprojektin alkamista, niin päätös voidaan tehdä esimerkiksi analysointivaiheessa, kun on tutustuttu lähde- ja kohdejärjestelmiin.

Big-Bang -tyyppisessä migraatiossa lähdejärjestelmän sisältämä data siirretään kokonaisuudessaan samalla kertaa kohdejärjestelmään. Yleensä tämä vaatii katkon tuotantokäytössä olevaan sovelluksen käyttöön siksi aikaa, kun data puretaan lähdejärjestelmästä ja viedään kohdejärjestelmään. Käyttökatkon jälkeen tuotantokäyttö jatkuu uudessa sovelluksessa, mikäli migraatio on suoritettu onnistuneesti. Mikäli migraatiossa havaitaan ongelmia, voidaan tuotantokäyttöä jatkaa vanhassa sovelluksessa. (Oracle 2011, 3.)

Big-Bang -tyyppisen migraatiostrategian hyvä puoli on, että onnistuessaan koko migraatioprosessi on viety läpi lyhyessä ajassa. Vanha sovellus voidaan aja alas heti onnistuneen migraation jälkeen. Huonona puolena on, että mikäli jotain odottamatonta tapahtuu, niin tuotantokäytön katko voi venyä ja samalla lamauttaa yrityksen toimintaa. Migraatio pitäisi suunnitella ja aikatauluttaa hyvin huolellisesti ja mahdollisesti testata kokonaisuudessaan ennen kuin ne toteutetaan oikeasti. Suunnitelmassa pitäisi olla varauduttu migraation perumiseen eli maininta siitä, milloin ja miten palataan vanhan järjestelmän käyttöön, mikäli migraation aikana tai sen jälkeisessä testissä havaitaan ongelmia. (Oracle 2011, 3.)

Vaiheittaisessa migraatiossa lähdejärjestelmän sisältämä data siirretään kohdejärjestelmään osissa. Molemmat järjestelmät ovat käytössä samaan aikaan, joten tuotantokatkon pituus jää lyhyeksi tai sellaista ei ole ollenkaan. Vaiheittaisen migraation hallinta on hiukan monimutkaisempaa kuin Big-Bang -tyyppisen, koska pitää tietää mitä dataa on jo siirretty uuteen järjestelmään ja mitä on vielä siirtämättä. Pitää myös suunnitella tarkkaan, miten datan siirto voidaan vaiheistaa. Samassa vaiheessa voidaan siirtää vain sellaista dataa, jolle ei jää riippuvuuksia vanhaan järjestelmään vielä jäävään dataan. (Oracle 2011, 4.)

Käyttäjille vaiheittainen migraatio voi näkyä siten, että osa toiminnallisuudesta on vanhassa sovelluksessa ja osa uudessa. On myös mahdollista, että käyttäjät jatkavat pelkästään vanhan sovelluksen käyttöä ja sinne tehdyt muutokset ohjataan automaattisesti ilman käyttäjän toimenpiteitä myös uuteen sovellukseen, mikäli muutokset kohdistuvat jo siirrettyyn dataan. (Oracle 2011, 4.)

Vaiheittainen migraatio ei ole niin riskialtis kuin Big-Bang -tyyppinen. Epäonnistuessaan migraatio ei pysäytä koko yrityksen toimintaa, koska vanha sovellus on edelleen käytössä. Toisaalta tämä on yritykselle myös kalliimpi, koska joudutaan suunnittelemaan ja toteuttamaan useita migraatioita. Lisäksi käytössä on koko ajan kaksi sovellusta, joita joudutaan ylläpitämään.

Ei ole selvää sääntöä, joka kertoisi kumpi edellä mainituista migraatiostrategioista pitäisi valita tai kumpi on parempi. Mikäli migraatioprojekti pystytään suunnittelemaan hyvin ennakolta, eikä sitä voida pilkkoa osiin tai sillä on ennalta määrätty ajankohta, jolloin migraatio pitäisi tehdä, niin silloin Big-Bang -tyyppinen sopii paremmin tai voi jopa olla ainut vaihtoehto. Jos taas migraation määrittely on vaikea, esimerkiksi monimutkaisten rakenteiden tai riippuvuuksien takia ja mikäli projekti on helposti pilkottavissa osaprojekteiksi, niin silloin vaiheittainen sopii paremmin. Tällöin voidaan esimerkiksi siirtää helpommat osiot ensin ja kokemusten karttuessa vaikeammat osiot myöhemmin. (CloverDX, 7.)

3.2.3 Projektisuunnitelman teko

Suunnitellaan projektin resurssointi ja kartoitetaan sidosryhmät. Tehdään projektin riskikartoitus, budjetti ja aikataulutus. Vaiheen lopputuloksena saadaan tarkka migraatioprojektin projektisuunnitelma siitä mitä ollaan tekemässä. Projektisuunnitelmassa on kuvattuna muun muassa projektin kattavuus, migraatiostrategia, tehtävät, kustannukset sekä liiketoiminnan ja teknologian asettamat reunaehdot. (Matthes & Schulz 2011, 28.)

Projektisuunnitelmassa pitää kuvata myös projektiorganisaatio. Datamigraatioprojekti vaatii hyvin monenlaista osaamista. Vaaditaan sekä liiketoimintaosaamista, että teknistä osaamista. Projektiorganisaatiossa pitää olla ihmisiä, jotka

tuntevat lähde- ja kohdejärjestelmät, migraatioteknologioiden osaamista, tietoverkko- ja palvelinosaamista sekä testausosaamista. Projektiorganisaatio voi olla hyvinkin suuri. Matthes & Schulzin (2011) mukaan migraatioprojektissa pitäisi olla edustettuina seuraavat roolit:

- **Liiketoiminta-alueen omistaja**. Projektin tilaajan edustaja. Toimii projektin johtoryhmän puheenjohtajana.

- **Migraatioprojektin projektipäällikkö**. Yleensä tilaajaorganisaatiosta. Päävastuussa projektin teknisestä toteutuksesta. Ohjaa projektiryhmien päivittäistä työskentelyä yhdessä ryhmien projektipäälliköiden kanssa.

- **Lähdejärjestelmän projektiryhmä**. Koostuu projektipäälliköstä ja lähdejärjestelmän tuntevista liiketoiminta- ja teknisistä asiantuntijoista.

- **Kohdejärjestelmän projektiryhmä**. Koostuu projektipäälliköstä ja kohdejärjestelmän tuntevista liiketoiminta- ja teknisistä asiantuntijoista.

- **Tekninen migraatiotiimi**. Vastaa migraation teknisestä toteutuksesta. Tekninen projektipäällikkö vastaa migraatio-ohjelmistojen teknisestä toteutuksesta yhdessä arkkitehdin ja migraatioasiantuntijoiden kanssa.

- **Testaustiimi**. Vastaa migraation testauksesta yhdessä lähde- ja kohdejärjestelmätiimien sekä teknisten testausasiantuntijoiden kanssa. Testaustiimin projektipäällikkö koordinoi testaussuunnitelman ja testauksessa tarvittavien skriptien toteutuksen.

- **Tietoliikennetiimi**. Vastaa migraatioympäristön tietoliikenteen ja palvelinympäristön pystyttämisestä ja toimivuudesta. Tiimissä on oma projektipäällikkö ja tietoverkko- sekä palvelinosaajia.

Projektiryhmän koostumus voi vaihdella migraatioprojektin tyypistä ja laajuudesta riippuen.

3.3 Data-analyysi

3.3.1 Datan profilointi

Tässä vaiheessa pyritään selvittämään millaista tietoa lähdejärjestelmä sisältää ja miten hyvälaatuista tieto on. Vaihe on tärkeä, mutta aikaa vievä, pitkäveteinen ja tarkkuutta vaativa osuus migraatioprosessin suunnittelussa. Mitä enemmän käytetään aikaan lähdedatan analysointiin, sitä enemmän se helpottaa migraatioprosessin myöhempien vaiheiden toteutusta. Tässä vaiheessa pyritään kuvaamaan lähdejärjestelmästä poimittu data niin hyvin, että sen perusteella voidaan tehdä suunnitelmat ja toteutus sen muokkaamiseksi kohdejärjestelmän vaatimaan muotoon. Kuvataan datan virheet, puutteet, epäjohdonmukaisuudet ja epätarkkuudet ja näin saadaan käsitys sen laadusta. Tämän vaiheen löydöillä on suora vaikutus siihen millaisia toiminnallisuuksia migraatio-ohjelmistoon pitää toteuttaa. Lisäksi löydöksiä voidaan hyödyntää testitapausten suunnittelussa ja toteutuksessa. (Matthes & Schulz 2011, 35-36.)

Data-analyysivaiheen lopputuloksena syntyy migraation määrittelydokumentaatio, jonka pohjalta lähdetään tekemään migraation teknistä toteutusta. Määrittelydokumentissa kuvataan metatiedoille migraatiossa tehtävät toimenpiteet sekä metatietojen kohdistus kohdejärjestelmän metatietomalliin. Lisäksi tässä vaiheessa tehdään alustava testaussuunnitelma. Määrittelyt käydään läpi teknisestä toteutuksesta vastaavien ja liiketoiminnasta vastaavien henkilöiden kesken ja hyväksytään ennen kuin teknistä toteutusta aletaan tekemään. (Oracle 2011, 11.)

Seuraavissa kappaleissa on kuvattuna datalle yleisimmin migraatiossa tehtävät muokkaustoimenpiteet. Jokaisessa migraatiossa ei välttämättä kaikkia toimenpiteitä tehdä.

3.3.2 Datan puhdistaminen

Tiedon laatu on tärkeää sovellusten ja sitä kautta yritysten toiminnalle. Sovelluksissa oleva virheellinen tai puutteellinen tieto voi johtaa myös virheellisiin päätöksiin. Tiedon laatu voidaan määritellä usean eri kriteerin perusteella. Taulukossa 1 on Loshin (2009) listaamat mielestään tärkeimmät kriteerit.

TAULUKKO 1. Tiedon laatukriteerit (Losh 2009)

Kriteeri	**Määritelmä**
Ainutkertaisuus (Uniqueness)	Tiedosta on tallennettu tietojärjestelmään vain yksi instanssi.
Johdonmukaisuus (Consistency)	Tieto pitää aina esittää samassa muodossa.
Oikea-aikaisuus (Timeliness)	Tieto pitää olla saatavilla silloin, kun sitä tarvitaan.
Oikeamuotoisuus (Validity)	Tiedon pitää olla tyyppimääritysten mukaisessa muodossa.
Täsmällisyys (Accuracy)	Tiedon pitää olla oikein.
Täydellisyys (Completeness).	Tiedon pitää olla täydellistä.

Sovellusten tietovarastoihin kertyy ajan mittaan vanhentunutta, puutteellista tai virheellistä dataa joko ohjelmistovirheiden, puutteellisen ylläpidon tai muun syyn takia. Jotta tätä puutteellista tietoa ei siirretä kohdejärjestelmään, pyritään se puhdistamaan eli korjaamaan siinä esiintyvät virheet ja puutteet.

Datan analysointivaiheen havaintojen perusteella voidaan lähdejärjestelmässä tehdä alustavaa tiedon puhdistamista joko manuaalisesti tai ohjelmallisesti. Voidaan korjata virheelliset tiedot, jotta ne eivät aiheuta ongelmia varsinaisessa migraatiossa. Sellaiset tiedot, joita ei ole tarkoitus siirtää kohdejärjestelmään voidaan poistaa tässä vaiheessa tai sitten se voidaan tehdä migraation muissa vaiheissa.

Matthes & Schulzin (2011) mukaan datan puhdistamista voidaan tehdä migraatioprosessin monessa eri vaiheessa

- Lähdejärjestelmässä
- Datan purkamisen aikana
- Transformaationprosessin aikana
- Kohdejärjestelmään latauksen aikana
- Kohdejärjestelmässä migraation jälkeen.

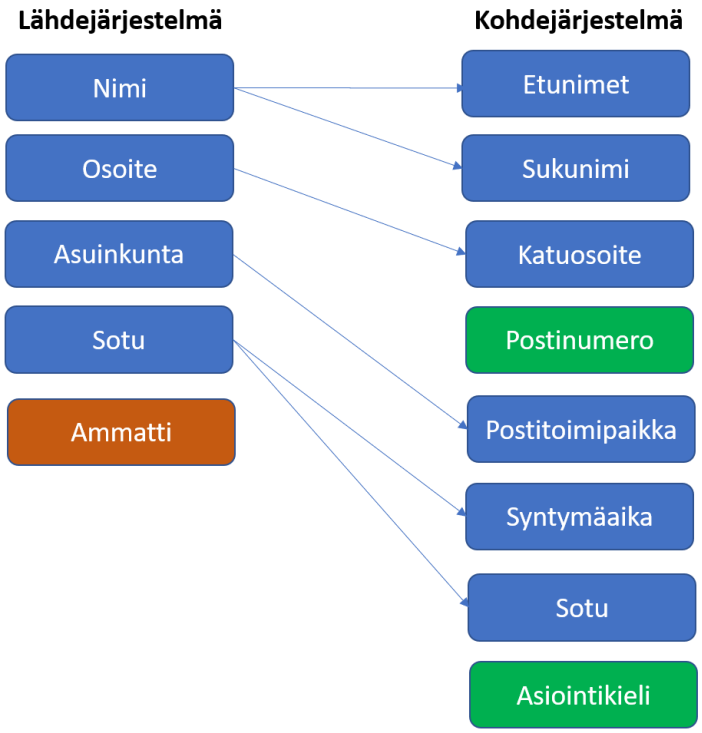
Täydellisesti datan puhdistamista ei ole mahdollista eikä tarkoituksen mukaistakaan tehdä, mutta pyritään korjaamaan sellaiset virheet ja puutteet, jotka voisivat aiheuttaa ongelmia migraatiossa tai kohdejärjestelmän toimivuudessa. Tällaisia ovat esimerkiksi duplikaatit ja väärässä muodossa olevat tieto tai kohdejärjestelmän kannalta pakolliset tiedot.

3.3.3 Metatietojen kohdistaminen

Kohdistetaan lähdejärjestelmän metatiedot kohdejärjestelmän vastaaviin metatietoihin. Esiselvitys-vaiheessa tehty metatietorakenteiden analyysi on ehdoton edellytys tämä vaiheen onnistumiseen. Metatietojen kohdistaminen voi olla hyvin mutkikas prosessi riippuen siitä miten paljon lähde- ja kohdejärjestelmien tietorakenteet eroavat toisistaan. Tämä on yleensä migraatioprosessin yksi haastavimmista ja aikaa vievimmistä tehtävistä. Epätarkkuudet ja virheelliset kohdistukset tässä vaiheessa vaikuttavat hyvin paljon tietojen oikeellisuuteen migraation jälkeen ja voivat pahimmassa tapauksessa johtaa koko siirtoprosessin epäonnistumiseen. Lähde- ja kohdejärjestelmän metatiedot voivat poiketa toisistaan hyvinkin paljon. Sama metatieto voi toisessa järjestelmässä olla nimetty toisin, metatieto on voitu jakaa useammaksi metatiedoksi tai metatieto voi puuttua kokonaan. Nämä tapaukset pitää käydä läpi ja kuvata käsittelysäännöt, miten niitä migraatiossa käsitellään.

Kuviossa 4 on kahden eri järjestelmän asiakkaan metatiedot. Lähdejärjestelmässä koko asiakkaan Nimi on yhdessä kentässä. Kohdejärjestelmässä se on pilkottu siten, että Etunimet ovat eri kentässä kuin Sukunimi. Jopa tämä voi aiheuttaa haasteita migraatiossa, jos asiakkaan sukunimi on moniosainen vierasperäinen kuten esimerkiksi von Hertzen tai van Gilse van Der Pals. Lähdejärjestelmässä ei ole Syntymäaika-tietoa, joten kohdejärjestelmään sellainen voidaan

muodostaa Sotu-kentästä. Kohdejärjestelmässä ei ole Ammatti-kenttää ollenkaan, joten pitää päättää mitä migraatiossa sille tiedolle tehdään. Jätetäänkö se kokonaan pois vai lisätäänkö tieto myös kohdejärjestelmän tietomalliin. Jos lisätään, niin se vaatii tietysti muutoksia moneen paikkaan kohdejärjestelmään, jotta lisättyä tietoa voidaan hyödyntää.



KUVIO 4. Esimerkki lähde- ja kohdejärjestelmän erilaisista tietorakenteista.

3.3.4 Metatietojen rikastaminen

Kohdejärjestelmässä voi olla sellaisia järjestelmän toiminnan kannalta olennaisia metatietoja, joita lähdejärjestelmästä ei löydy. Joko metatietokenttä puuttuu kokonaan lähdejärjestelmän tietomallista tai sitten metatiedon arvo puuttuu joltakin siirrettävältä instanssilta. Mikäli metatietoon ei saada arvoa, niin lähdejärjestelmän tietoja ei voida viedä kohdejärjestelmään. Tällaisissa tapauksissa pitää selvittää mistä ja miten puuttuvat metatiedot saadaan. Metatiedon arvoksi voidaan esimerkiksi viedä kaikkiin instansseihin sama arvo tai sitten metatiedon arvo pyritään päättelemään jostakin. Tieto voidaan muodostaa vaikkapa lähdejärjestel-

mässä jo olemassa olevista kentistä. Kuviossa 4 kuvatussa esimerkissä kohdejärjestelmän vaatima Syntymäaika voidaan muodostaa lähdejärjestelmässä puretussa tietorakenteessa olevasta Sotu-kentästä.

Tieto voidaan myös päätellä jonkun toisen tiedon perusteella tai sitten kaikille viedään sama tieto vakiona. Migraatio-ohjelmisto voi hakea tarvitsemansa tiedon parametritiedostosta tai jostain toisesta sovelluksesta, esimerkiksi tiedonohjausjärjestelmästä tai koodistopalvelusta. Kuviossa 4 olevassa esimerkissä lähdejärjestelmässä ei ollut vastaavuutta Asiointikieli-metatiedolle. Koska tietoa ei oikein voida päätellä mistään, niin näissä tapauksissa voidaan kaikkiin instansseihin viedä vakioarvo ja päivittää oikea tieto kohdejärjestelmään myöhemmin, kun sellainen saadaan esimerkiksi asiakkaalta itseltään. Lisäksi kuvion 4 esimerkissä lähdejärjestelmästä ei löytynyt vastaavuutta kentälle Postinumero. Sille voidaan päätellä arvo kenttien Asuinkunta ja Osoite perusteella. Tieto voidaan hakea niiden avulla esimerkiksi migraation parametreista tai sitten jostain tarjolla olevasta julkisesta palvelusta.

3.3.5 Metatietokonversiot

Tietojärjestelmissä voidaan sama metatieto tallentaa monessa eri muodossa. Paras esimerkki tällaisesta on ajan esittäminen. Suomalainen tapa esittää päivämäärä käyttöliittymällä on yleensä muodossa PP.KK.VVVV. Päivä ja kuukausi yleensä ilman etunollia, mutta monesti myös etunollien kanssa. Amerikkalainen tapa esittää päiväys on muodossa KK/PP/VVVV. Amerikkalaisessa muodossa kuukausi voidaan esittää numeron sijasta myös kirjoitetussa muodossa esimerkiksi October 22, 2020. ISO 8601 -standardin mukaan päivämäärä merkitään muodossa VVVV-KK-PP. Asia monimutkaistuu vielä tästä lisää, jos samassa kentässä säilytetään myös kellonaika ja aikavyöhyke, jolloin päiväys voi näyttää esimerkiksi tällaiselta 2020-10-22T15:45:232+02:00.

Se miten aikatieto on tallennettu tietokantaan voi poiketa siitä, miten se näytetään käyttöliittymällä. Tietokannassa aika voi olla tallennettuna, tietokannasta riippuen joko DATE, TIME, DATETIME, TIMESTAMP (aikavyöhykkeen kanssa tai ilman) muodossa. Onpa sellainenkin ratkaisu mahdollista, että päivämäärä tallennetaan

LONG-tyyppisessä kentässä Unix Epoch time -muodossa, joka kertoo ajan millisekunteina 1.1.1970 lähtien.

Mikäli lähdejärjestelmän ja kohdejärjestelmän välillä on eroavaisuuksia ajan esittämisessä, migraatiossa pitää tehdä sääntö, miten lähdejärjestelmän esittämä aika muunnetaan kohdejärjestelmän vaatimaan muotoon. Mikäli molemmissa järjestelmissä on kenttään tallennettuna pelkästään päivämäärä, niin sen konvertoiminen esimerkiksi muodosta PP.KK.VVVV muotoon VVVV-KK-PP on yksinkertaista. Mikäli toisessa järjestelmässä on mukana myös kellonaika tai toisessa järjestelmässä kellonaika on tallennettuna erilliseen TIME-kenttään niin sitten asia mutkistuu vähän.

3.3.6 Koodistokonversiot

Käyttöliittymällä selväkielisenä näytetty tieto tallennetaan monesti tietojärjestelmään koodina. Tämä voidaan tehdä tilan säästämiseksi tai vaikkapa helpottamaan tiedon esittämistä eri kielillä. Jos koodistoja ei ole kansallisesti standardoitu, niin hyvin harvoin tietojärjestelmissä on käytössä samanlainen kooditus. Toisessa voi olla numeeriset koodit, toisessa tekstimuotoiset. Vaikka molemmissa olisikin numeeriset koodit, niin sama koodi yhdessä järjestelmässä voi tarkoittaa ihan toista toisessa järjestelmässä. Tietojärjestelmien välisessä migraatiossa lähdejärjestelmässä käytetty kooditus muunnetaan kohdejärjestelmän käyttämään kooditukseen. Tämä voidaan tehdä käyttäen apuna kääntötauluja, joihin lähde- ja kohdejärjestelmien käyttämät koodistot kuvataan ja niitä hyödyntäen muunnetaan migraatio-ohjelmiston käyttämään tekniseen muotoon. Kääntötaulu voi olla esimerkiksi tietokanta, tekstimuotoinen parametritiedostot tai vaikka xml- tai xsl-tiedosto.

Esimerkiksi terveydenhuollon AR/YDIN-näkymät -luokitusta käytetään terveydenhuollon potilastietojärjestelmissä sitomaan potilaskertomukseen kirjattavaa tietoa tiettyyn sisältö- tai hoitokokonaisuuteen kuten lääketieteen erikoisalaan, ammattialaan tai palveluun liittyviin hoitoihin (Kela). Taulukossa 2 on esimerkki kääntötaulun määrittelystä, johon on kuvattu AR/YDIN-näkymän selväkielinen nimi sekä sen kooditus lähde- ja kohdejärjestelmissä. Mikäli lähdejärjestelmän

koodille ei löydy suoraan vastinetta kohdejärjestelmästä, niin sitten pitää päättää mitä niissä tapauksissa tehdään. Tapauksesta riippuen voidaan joko luoda kohdejärjestelmään uusi koodi tai sitten kohdistetaan se johonkin olemassa olevaan koodiin.

TAULUKKO 2. Koodistokonversio.

Näkymän nimi	Lähdejärjestelmä	Kohdejärjestelmä
Sisätaudit	100	SIS
Kuntoutus	105	KUN
Radiologia	106	RTG
Sosiaalityö	117	SOS
Lääkehoito	126	LÄÄ

3.3.7 Asiakirjojen konversiot

Migraatiossa voidaan metatietojen lisäksi siirtää myös asiakirjoja. Esimerkiksi asianhallintajärjestelmään tallennetaan hyvinkin paljon monenlaisia asiakirjoja ja monessa eri muodossa. Siellä voi olla esimerkiksi Microsoft Office -ohjelmistolla tehtyjä asiakirjoja, kuvia, videoita, sähköposteja tai pdf-muotoisia asiakirjoja. Mikäli kohdejärjestelmä ei tue kaikkia lähdejärjestelmässä olevia asiakirjatyyppejä, tai jostain muusta syystä, voidaan osa tai kaikki lähdejärjestelmän sisältämät asiakirjat joutua muuntamaan kohdejärjestelmän vaatimaan muotoon. Muu syy voi olla esimerkiksi pitkäaikaissäilytykseen määrättävien asiakirjojen arkistointi, jolloin ne pitää muuntaa pitkäaikaissäilytykseen sopivaan muotoon tai potilasasiakirjojen muuntaminen Kanta-järjestelmän Potilastiedon arkiston vaatimaan CDA R2 -muotoon.

Tiedostojen konvertointi muodosta toiseen on hyvin riskialtista, koska aina vaarana, että osa tiedoista katoaa. Näin voi käydä esimerkiksi silloin, jos lähdedokumentissa on käytetty sellaisia fontteja, joita konversiota suorittava ohjelmisto ei tunnista. Konvertoidut tiedostot tulisikin tarkistaa hyvin aina konversion jälkeen. Tämä ei kuitenkaan ole täysin aukoton, jos konvertoitavia tiedostoja on hyvin paljon. Millään ei kaikkia pystytä tarkistamaan.

3.4 Kehitystyö

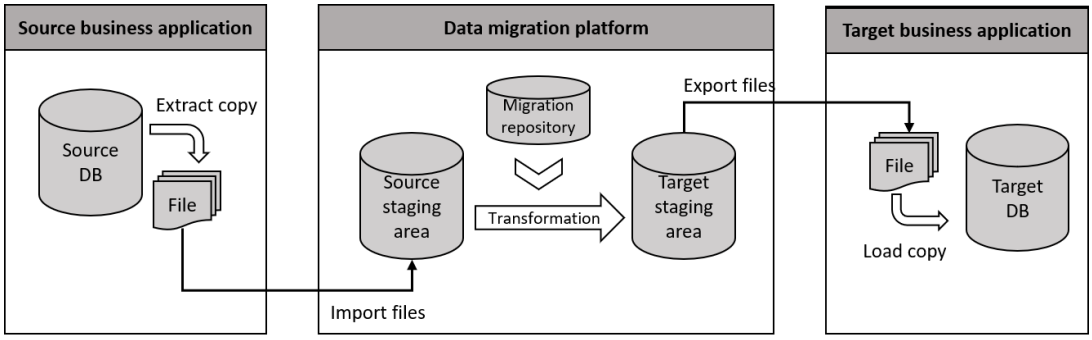
3.4.1 Teknisen arkkitehtuurin suunnittelu

Esiselvitys- ja data-analyysivaiheissa tehtyjen havaintojen ja suunnitelmien perusteella aletaan suunnittelemaan migraation teknistä toteutusta. Ensimmäisessä vaiheessa suunnitellaan migraation toteutuksen tekninen arkkitehtuuri, kuten tarvittavat palvelimet, tietoverkot, tietoliikenneyhteydet ja -avaukset. Kuvataan, miten tietoa siirretään eri migraatiovaiheissa tarvittavien palvelinten välillä sekä miten huolehditaan tietoturvasta, mikäli siirrettävä tieto on sensitiivistä. Selvitetään, onko olemassa valmista migraatio-ohjelmistoa, jota voitaisiin käyttää vai pitääkö sellainen tai osia siitä tehdä projektissa.

Määritellään keinot ja työvälineet, joilla tiedot saadaan purettua lähdejärjestelmästä. Selvitetään, onko järjestelmässä toiminnallisuus, jonka kautta sen sisältämät tiedot saadaan purettua levyjärjestelmään, vai voidaanko tiedot hakea jonkin rajapinnan kautta tai suoraan tietokannasta. Ohjelmistorobotiikkaa hyödyntämällä tiedot saadaan purettua myös käyttöliittymän kautta. Tämä voi monesti olla ainut tapa, jos esimerkiksi lähdejärjestelmä on jo vanha, eikä sen alkuperäistä toimittajaa syystä tai toisesta ole enää olemassa. Suunnitellaan missä muodossa data puretaan ja miten ne siirretään migraation seuraavan vaiheen saataville. Mikäli lähdejärjestelmässä on toiminnallisuus datan purkamiseen, niin se voidaan tehdä tässä vaiheessa, vaikka muu osa migraatio-ohjelmistosta on vielä tekemättä. Näin päästään heti analysoimaan dataa ja suunnittelemaan sen jatkokäsittelyä.

Kohdejärjestelmän osalta selvitetään, miten tiedot saadaan ladattua sinne. Tarjoaako se rajapintoja, joita voidaan käyttää, vai viedäänkö tiedot suoraan tietokantaan. Ohjelmistorobotiikkaan on mahdollisuus hyödyntää myös tietojen latukseen vastaanottavaan järjestelmään. Robotti voi ne syöttää esimerkiksi käyttöliittymän kautta, mikäli sopivaa rajapintaa ei ole käytettävissä, eikä tietoja voida viedä suoraan tietokantaankaan.

Teknisen suunnittelun lopputuloksena saadaan kuvaus migraatiossa käytettävästä teknologiasta ja migraatioalustan kokonaisarkkitehtuurista. Matthew & Schulz (2011) ovat kuvanneet yleisen arkkitehtuurimallin, jossa lähdejärjestelmästä purettu data siirretään erilliseen migraatioalustaan muokattavaksi (KUVIO 5).



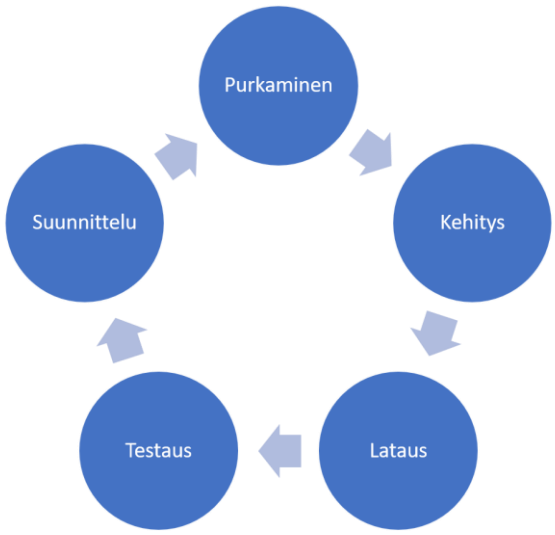
KUVIO 5: Migraatioalustan yleiskuvaus (Matthes & Schulz 2011, 29)

Migraatioalusta muodostuu lähde- ja kohdejärjestelmien datan kokoamisalueista sekä säilytyspaikasta, joka pitää sisällään migraation suorittamisessa tarvittavat ohjelmistot, skriptit ja parametrit. Kuvassa kokoamisalueet ovat tietokantoja, mutta mikäli migraatio tehdään sovellusten rajapintoja hyödyntäen, ne voivat myös olla tiedostoja tai sovelluksia. Lähdejärjestelmästä data on voitu purkaa tiedostoihin, migraatioalustalla se muokataan kohdejärjestelmän vaatimaan muotoon ja kirjoitetaan tiedostoihin, jotka migraatio-ohjelmisto lataa kohdesovellukseen.

3.4.2 Migraatiologiikan tekninen toteutus

Tässä vaiheessa rakennetaan migraatiotyössä käytettävät ohjelmistot ja skriptit, mikäli niitä ei ole vielä valmiina. Hyvin usein myös valmiita, mahdollisesti aikaisemmin toteutetuissa vastaavissa migraatioissa tehtyjä komponentteja joudutaan muokkaamaan, koska jokainen migraatio on erilainen. Ohjelmistoihin rakennetaan tarvittavat toiminnallisuudet, joilla data saadaan muokattua kohdejärjestelmän vaatimaan muotoon. Lisäksi tarvittaessa rakennetaan datan purkamisessa ja lataamisessa käytettävät komponentit.

Tekninen toteutus on eri vaiheista koostuva iteratiivinen prosessi (KUVIO 6), jossa muokataan migraatiologiikkaa ja testataan se lähdejärjestelmästä puretulla pienellä testiaineistolla.



KUVIO 6. Migraatiologiikan kehitysprosessi.

Suunnitteluvaiheessa tarkennetaan esiselvitysvaiheessa lähde- ja kohdejärjestelmien analysoinnin havaintojen pohjalta migraatio-ohjelmiston logiikkaa. Kehitysvaiheessa toteutetaan datan purkamiseen, transformointiin ja lataukseen käytettävät ohjelmistot. Ohjelmistot testataan. Mikäli havaitaan virheitä, tehdään tarvittavat korjaukset. Kehitysvaiheessa ohjelmistoja ei testata koko lähdejärjestelmästä löytyvällä datamassalla vaan pienellä aineistolla kerrallaan. Tarvittaessa testiaineistoa voidaan purkaa useampaan kertaan. Mikäli mahdollista, niin jo kehitysvaiheessa kannattaa testata myös aineiston lataus kohdejärjestelmään.

3.4.3 Validointi ja testaus

Ennen kuin migraatio toteutetaan tuotannossa, se testataan testiympäristössä. Testaus on kaikista kattavin, mikäli se on mahdollista toteuttaa koko aineistolla tuotantoa vastaavassa ympäristössä. Tähän on kuitenkin harvoin mahdollisuutta, joten testiin pyritään poimimaan kattava joukko mahdollisimman erilaisia tapauksia, jotka ajetaan koko migraatioprosessin läpi.

Migraatioprosessin testauksen tavoitteet (Data Migration Testing Tutorial):

- Pyritään havaitsemaan ja minimoimaan mahdollisimman aikaisessa vaiheessa käyttäjille migraatiosta aiheutuvat haitat, kuten järjestelmien käyttökatkot
- Varmistetaan, että siirrettyä tietoa pystytään ja osataan hyödyntää kohdejärjestelmässä siinä käyttötarkoituksessa mihin se on tarkoitettu
- Varmistamaan, että tietoa ei häviä eikä monistu migraation aikana
- Varmistetaan, että tieto pysyy merkitykseltään muuttumattomana migraation aikana
- Varmistetaan, että migraatio pystytään viemään läpi sinne suunnitellussa aikataulussa.

Testausta tehdään migraatioprosessin jokaisessa vaiheessa. Testataan datan oikeellisuus prosessin eri vaiheissa, migraatioprosessin suoritus ja kohdejärjestelmään latauksen jälkeen kohdejärjestelmän toimivuus sekä sen liitynnät muihin järjestelmiin. Lisäksi pitää varautua migraation epäonnistumiseen ja testata myös palautusprosessi, jossa kohdejärjestelmä palautetaan alkuperäiseen tilaansa. (Data Migration Testing Tutorial).

Datan purkamisen jälkeen varmistetaan, että kaikki kohdejärjestelmään siirrettäväksi tarkoitettu data on purettu lähdejärjestelmästä ja validoidaan sen oikeellisuus ja eheys. Mahdolliset puutteet ja poikkeavuudet pyritään löytämään, jotta voidaan suunnitella tarvittavat puhdistamistoimenpiteet. Datan validointi tapahtuu parhaiten siihen tarkoitukseen kehitetyillä skripteillä.

Transformaatiovaiheen jälkeen testataan, että kaikki datalle suoritettavat muokkaustoimenpiteet ovat tuottaneet halutun lopputuloksen eikä dataa häviä näiden muokkaustoimenpiteiden aikana. Transformaationprosessin lopputuloksena datan, esimerkiksi lähdejärjestelmästä purettujen tiedostojen, lukumäärä ei välttämättä ole sama kuin syöttödatan johtuen siitä, että dataa voidaan pilkkoa tai yhdistellä prosessin aikana. Pitäisi kuitenkin varmistaa, että käsitteellisellä tasolla dataa ei ole hävinnyt. Esimerkiksi, kun siirretään asiakastietoja, on varmistettava, että kaikki asiakastiedot todellakin on siirretty. Testauksessa voidaan käyttää apuna skriptejä, joilla voidaan laskea metatietojen lukumääriä ennen ja jälkeen

prosessin, tutkia että pakolliset kentät on täytetty ja metatiedot ovat muodollisesti oikeanlaisia. Lisäksi testauksessa voidaan numeerisista metatiedoista laskea tarkistussummia ennen ja jälkeen transformaatioprosessin ja niitä vertailemalla varmistaa, että mitään ei ole hävinnyt tai muuttunut prosessin aikana (Matthes & Schulz & Haller 2011, 444).

Migraatioprosessin testauksen tarkoituksena on varmistaa koko prosessin toimivuus lähdejärjestelmän datan poiminnasta sen lataukseen kohdejärjestelmään. Tässä vaiheessa testataan, että migraatioprosessi voidaan suorittaa onnistuneesti alusta loppuun. Varmistetaan, että kaikki migraatioon osallistuvat ihmiset tietävät omat tehtävänsä ja aikataulun mihin aikaan mikin vaihe pitää tehdä. Samalla saadaan arvokasta tietoa prosessin suorittamiseen kuluvasta ajasta ja sitä voidaan käyttää apuna suunnitellessa varsinaista tuotantomigraatiota. (Matthes & Schulz 2011, 50.)

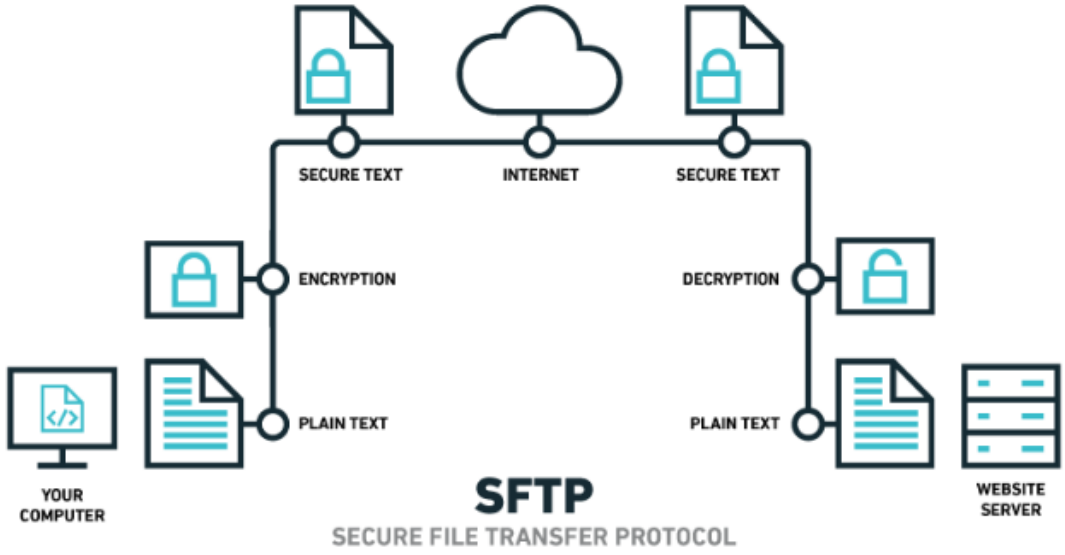
Migraatiossa siirretään tietoa eri tietoverkkojen ja palvelimien välillä. Testeillä pitää varmistua, että tarvittavat tietoliikenneyhteydet on avattu. Varmistetaan, että esimerkiksi palomuurit on oikein konfiguroitu ja kaikki tarvittavat portit on avattu.

Testataan kohdejärjestelmän toiminta datan lataamisen jälkeen. Varmistetaan, että sinne ladattu data on virheetöntä eikä se aiheuta häiriöitä sovelluksen toimintaan. Varmistetaan sovelluksen suorituskyky ja käytettävyys vastaa sille asetettuja tavoitteita. Testataan myös sovelluksen yhteistoiminta siihen liittyvien muiden sovellusten ja laitteistojen esimerkiksi tulostimien kanssa. (Matthes & Schulz 2011, 51.)

3.4.4 Tietoturva

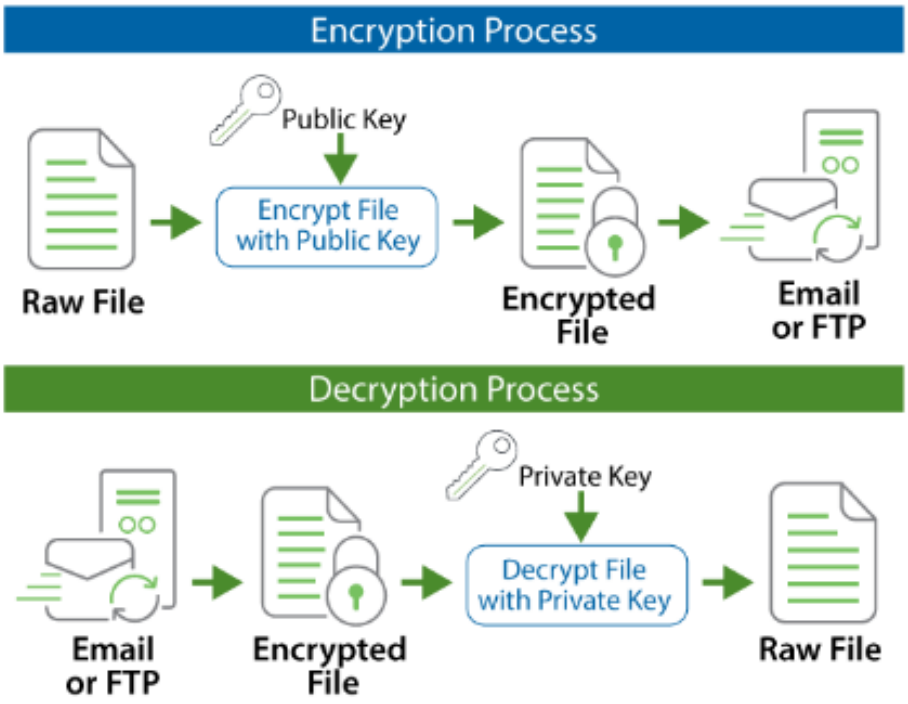
Migraatioprosessissa dataa siirrellään mahdollisesti eri tietoverkkojen ja palvelinten välillä. Mikäli tieto on hyvin arkaluontoista, se pitäisi suojata mahdollisen tietovuodon varalta. Käytetään siirroissa SSL suojattua tiedonsiirtoprotokollaa SFTP:tä. SFTP suojaa dataa siirron aikana, mutta ei enää sen jälkeen, kun se on kirjoitettu tiedostopalvelimelle. Kuviossa 7 on kuvaus SFTP-siirrosta. SFTP-asia-

kaskohjelmisto salaa tiedostot ennen niiden lähettämistä tietoverkkoon ja vastaanottavalla palvelimelle oleva ohjelmisto purkaa salauksen ennen tiedostojen kirjoittamista levyjärjestelmään. Siirron aikana tiedot on salatussa muodossa.



KUVIO 7. Datan suojaus siirron aikana (Hughes, 2018)

Sen lisäksi, että data suojataan siirron ajaksi, voidaan suojata myös levylle kirjoitetut datatiedostot. Varsinkin, mikäli siirroissa käytetään yrityksessä yleisesti käytössä olevaa tiedostopalvelinta ja palvelimelle on pääsy sellaisilla henkilöillä, joiden ei välttämättä tarvitse tai saa päästä näkemään siirrettyjä tiedostoja, kannattaa ne suojata salaamalla. Lähetettävä data voidaan salata esimerkiksi käyttäen PGP (Pretty Good Privacy) -ohjelmistoa (KUVIO 8). Lähettäjä salaa datatiedostot käyttäen vastaanottajan julkista avainta. Sen jälkeen tiedostot lähetetään vastaanottajalle käyttäen SFTP-protokollaa. Vastaanottaja siirtää tiedostot tiedostopalvelimelta tietoturvalliseen paikkaan ja purkaa tiedostojen salauksen käyttäen omaa salaista avaintaan. PGP:llä salaamalla varmistetaan myös, ettei tieto ole muuttunut siirron aikana, koska jos näin on tapahtunut salauksen purkaminen ei enää onnistu.



KUVIO 8. Siirrettävän datan suojaus PGP:llä (Raicea, 2017)

Myös migraatiossa käytettävät palvelimet pitää suojata. Palvelimelle saa olla pääsy vain migraatioprojektissa työskenteleville tai palvelimen ylläpidosta vastaaville henkilöille. Palvelimelle asennetaan vain migraatiossa tarvittavat ohjelmistot ja kaikki ylimääräiset sisään tai ulos menevät portit suljetaan. Lisäksi palvelimet tulee olla suojattuna haittaohjelmilta. Palvelimien lokien avulla pystytään jälkikäteen seuraamaan, kuka ja milloin tietoja on katsellut tai muuttanut.

3.5 Migraation toteutus

Ennen migraation aloitusta estetään muutosten tekeminen lähdejärjestelmässä oleville tiedoille sekä uusien tietojen lisääminen. Toteutetaan migraation eri vaiheet sekä niihin suunnitellut testit. Testataan kohdejärjestelmän ja sen liittymien toiminta migraation päätyttyä. Mikäli havaitaan virheitä, niin selvitetään, voidaanko virheet korjata kohdejärjestelmässä, vai pitääkö kohdejärjestelmä palauttaa alkuperäiseen tilaansa. Aina paluu alkuperäiseen ei välttämättä ole mahdollista. Näissä tapauksissa myös tuotantomigraatiota pitäisi harjoitella tuotantoympäristöä vastaavassa ympäristössä, jotta mahdolliset ongelmat havaitaan ja ne voidaan korjata ennen varsinaista tuotantomigraatiota.

4 ESIMERKKI TOTEUTETUSTA MIGRAATIOSTA

4.1 Taustaa

Opinnäytetyön toimeksiantaja tarjoaa terveydenhuollon organisaatioille vanhojen potilastietojen arkistointia. Yleensä kyseessä on vanha, aktiivikäytöstä poistunut järjestelmä, jonka tietosisältö pitää arkistoida tai aktiivikäytössä oleva järjestelmä, jossa asiakirja on laadittu ennen kuin palvelunjärjestäjä on alkanut tallentamaan asiakirjoja reaaliaikaisesti Kanta-palveluun. Potilasasiakirjoille on lainsäädännössä määritelty verrattain pitkät säilytysajat, eikä tietosisältöä voi hävittää järjestelmän käytöstä poistamisen yhteydessä. Arkistolaitoksen sekä Terveyden ja hyvinvointilaitoksen määräysten mukaan ainoa hyväksytty loppusijoituspaikka pysyväissäilytettävälle potilastiedoille on Kelan ylläpitämä Kanta-palvelun Potilastiedon arkisto (Arkistolaitos 2014). Potilastiedon arkisto on keskitetty kansallinen potilas- ja asiakastiedon arkisto, josta terveydenhuollon ammattilaiset voivat hakea potilastietoja, jotka on kirjattu eri terveydenhuollon organisaatioissa

Toimeksiantajan Kanta-välityspalvelu vastaanottaa usealta eri toimittajalta ja useassa eri terveyden- tai sosiaalihuollon tietojärjestelmässä tuotettuja asiakirjoja. Tietojärjestelmien toimittajien kanssa on sovittu missä muodossa tiedot pitää välityspalveluun toimittaa. Palvelu ei muokkaa tai rikasta saatua aineisto miten-kään vaan pelkästään allekirjoittaa asiakirjat sähköisesti, paketoi ne Kanta-palvelun vaatimaan muotoon ja lähettää ne Kanta-palvelun Potilastiedon arkistoon.

Terveydenhuollon organisaatioilla on vanhoja käytöstä poistettuja järjestelmiä, joille ei enää jostain syystä ole toimittajaa. Asiakirjoja ei voida näihinkään järjestelmiin jättää, vaan ne pitää sieltä saada tavalla tai toisella purettua ja vietyä pitkäaikaissäilytykseen. Lisäksi tietoja pitää mahdollisesti rikastaa tai konvertoida esitysmuodosta toiseen ennen kuin ne voidaan viedä Kanta-palveluun. On ilmennytkin tarvetta sellaiselle Kanta-välityspalvelulle, joka pystyy rikastamaan ja muokkaamaan vastaanottamaansa aineistoa Kanta-palvelun vaatimusten mukaiseksi. Potilastietojärjestelmissä tuotettujen asiakirjojen sisältöä ei ole tarkoitus eikä saakaan muuttaa, vaan pelkästään tuottaa puuttuvia vaadittuja metatietoja

ja mahdollisesti konvertoida asiakirja esitysmuodosta toiseen. Esimerkiksi tekstimuotoinen asiakirja PDF/A-muotoiseksi ja ei-rakenteisessa muodossa olevat asiakirjat rakenteiseen muotoon.

Opinnäytetyössä kuvattu migraatioprosessi koe ponnistettiin Visman Consulting Oy:n sisäisellä kehitysprojektilla, jossa sosiaalihuollon vanhojen asiakastietojen arkistointiprosessia laajennettiin siten, että ei-Kanta yhteensopivassa muodossa poimittu data muokataan Kanta-palvelun vaatimusten mukaiseksi ja arkistoidaan Kanta-palveluun.

4.2 Esiselvitysvaiheen havainnot

4.2.1 Lähdejärjestelmä

Tilaajaorganisaatiolla on käytössä sosiaalihuollon asiakastietojärjestelmä, joka on tullut elinkaarensa päähän. Organisaatio on luopunut sen käytöstä, mutta siellä olleita päättyneitä asiakkuuksia ei ole siirretty uuteen käyttöön otettuun järjestelmään. Koska tietojärjestelmä on poistettu käytöstä, eikä sitä ole koskaan liitetty Kanta-palveluun, siellä säilytyksessä olevat asiakirjat ovat niin sanottuja vanhoja potilastietoja. Vanhoja potilastietoja ovat potilasasiakirjat, jotka on tallennettu terveydenhuollon tietojärjestelmiin tai arkistoitu paperimuodossa ennen Kanta-palvelun Potilastiedon arkiston käyttöönottoa. (Vuollet & Nurmi & Välkkilä 2020, 5.)

Alkuperäistä tietojärjestelmän toimittajaa ei enää ole olemassakaan, joten tietojen poimiminen sieltä arkistointiin vientiä varten on haastavaa. Järjestelmässä ei ole dokumentoitua rajapintaa, jonka kautta tiedot voitaisiin hakea eikä niitä saada haettua luotettavasti myöskään suoraan tietokannasta tietokannan rakenteen monimutkaisuuden vuoksi. Tietojen haussa päätettiin hyödyntää ohjelmistorobotiikkaa.

4.2.2 Kanta-palvelut / Sosiaalihuollon asiakastiedon arkisto

Sosiaalihuollon asiakastiedon arkisto on osa Kelan Kanta-palveluita ja se mahdollistaa keskitetyn sähköisten sosiaalihuollon asiakastietojen aktiivisen käytön ja pysyvän säilyttämisen. Sosiaalihuollon asiakastiedon arkistoon viedään kaikki sosiaalipalveluissa syntyvät, asiakasta koskevat asiakasasiakirjat sekä vanhat potilasasiakirjat, jotka syntyneet ennen Kanta-arkiston käyttöönottoa. (Vuollet & Nurmi & Välkkilä 2020, 5.)

Rekisterinpitäjä on vastuussa tietojen toimittamisesta Kanta-palveluun sekä niiden muuntamisesta Kanta-palvelun vaatimaan muotoon. Kanta-palvelu tarjoaa rajapinnan, jonka kautta sosiaalihuollon asiakirjat voidaan viedä Kanta-palveluun. Data-analyysivaiheessa perehdyimme tarkemmin mihin muotoon data pitää muuntaa ennen sen viemistä Kanta-palveluun.

4.3 Data-analyysi

4.3.1 Lähdejärjestelmästä poimittu data

Tiedot lähdejärjestelmästä poimitaan ohjelmistorobotiikkaa hyödyntäen käyttöliittymän kautta tekstimuotoisiin tiedostoihin Päätösluettelo (Liite 2) ja Asiakaskertomus (Liite 3). Poimittavien asiakkaiden tiedot robotti saa projektin tilaajan asiakastietojärjestelmästä tuotetulta raportilta Perusturva-asiakasluettelo (Liite 1). Data-analysointivaiheessa huomasimme, että lähdejärjestelmästä tuotetuista raporteista ei löydy kaikkea Kanta-palvelun tarvitsemaa dataa, joten poimittua aineistoa joudutaan rikastamaan migraatiovaiheessa.

4.3.2 Sosiaalihuollon asiakirjat

Sosiaalihuollon asiakastiedon arkistoon tallennetaan neljää eri tyyppiä olevia asiakirjoja:

- Asiakkuusasiakirja

- Asia-asiakirja
- Asiakasasiakirja
- Liiteasiakirja.

Asiakkuusasiakirjaan tallennetaan sosiaalihuollon asiakkaan perustietoja sekä sosiaalihuollon asiakkuuden hallinnassa tarvittavia tietoja. Asia-asiakirjaan tallennetaan sosiaalihuollossa käsiteltävänä olevan asian tiedot. Asiakasasiakirjaan tallennetaan asiakkaan hoitoon liittyvää tietoa.

Asiakkuuden hoitoon liittyvät asiakirjat linkitetään toisiinsa. Asiakkuusasiakirjaan voi liittyä yksi tai useampi Asia-asiakirja. Asia-asiakirjan voi liittyä yksi tai useampi Asiakassiakirja. Asiakasasiakirjaan voi liittyä nolla tai useampi liiteasiakirja. (Kanta, 2019a). Taulukossa 3 on kuvattuna eri asiakirjatyyppien pakollisuus sekä rakenne missä muodossa asiakirjat pitää tallentaa Kanta-palveluun.

TAULUKKO 3. Sosiaalihuollon asiakirjatyypit (Kanta 2019b, 3)

Asiakirjatyyppi	Pakollinen	Näyttömuoto	Rakenteinen muoto
Asiakkuusasiakirja	Kyllä	XHTML	JSON
Asia-asiakirja	Kyllä	XHTML	JSON
Asiakasasiakirja	Kyllä	XHTML tai PDF/A	-
Liiteasiakirja	Ei	XHTML tai PDF/A	-

Kappaleissa 4.3.3 ja 4.3.4 on kuvattuna XHTML- ja JSON-muotoisten asiakirjojen rakenne.

4.3.3 Näyttömuotoinen XHTML-asiakirja

Näyttömuotoinen XHTML-asiakirja koostuu fhir-xhtml.xsd (HL7) XML-skeemassa määritellyistä elementeistä. Näyttömuoto sisältää vain asiakirjan varsinaisen sisällön ilman normaalisti XHTML-dokumenttiin kuuluvia html- ja head-elementtejä, jotta asiakirja voidaan asiakastiedon arkistossa sisällyttää näytettäväksi osana toista verkkosivua. Sosiaalihuollon asiakirjoille ei anneta CSS-tyylimäärityksiä, sen sijaan XHTML-muodon elementeille määritellään class-attribuutit, joiden

avulla asiakirjan ulkoasu voidaan määritellä asiakirjan näyttävässä järjestelmässä asiakirjan ulkopuolisella CSS-tiedostolla. Kuviossa 9 on kuvattuna XHTML-muotoiselle asiakirjalle sallitut elementit. (Kanta, 2019b.)

class-attribuutin arvo	Elementin käyttötarkoitus
soc-document	asiakirjan perusasettelu, kuten palstan leveys ja marginaalit
soc-header	ylätunniste
soc-logo	palveluntuottajan logon formatointia varten varattu luokka
soc-hdr1, soc-hdr2, soc-hdr3	ylätunnisteen sarakkeet
soc-main-wrapper	luokka, jonka sisälle tulee varsinainen asiakirjan sisältö
soc-meta	luokka asiakirjan metatietoja varten
soc-list	luokka, jonka sisälle sijoitetaan asiakirjan kohdat. Jokaisessa kohdassa on otsikko ja siihen kuuluva sisältö (pl. soc-fulltitle ja soc-fullcontent)
soc-item	luokka, jolla esitetään yksi asiakirjan kohta. Yksittäisen kohdan otsikko näytetään palstan vasemmalla puolella ja kohdan sisältö sen rinnalla oikealla puolella. Pitkät otsikot sijoittuvat kuitenkin omalle rivilleen ennen sisältöä.
soc-title	asiakirjan kohdan otsikko
soc-content	asiakirjan kohdan sisältö
soc-label	yksittäisen kentän nimike
soc-field	yksittäisen kentän arvo
soc-fulltitle	otsikko, jonka leveys on sama kuin palstan leveys
soc-fullcontent	kohdan sisältö, joka tulee koko palstan leveydelle
soc-footer	asiakirjan alatunniste
soc-ftr1, soc-ftr2, soc-ftr3	alatunnisteen sarakkeet

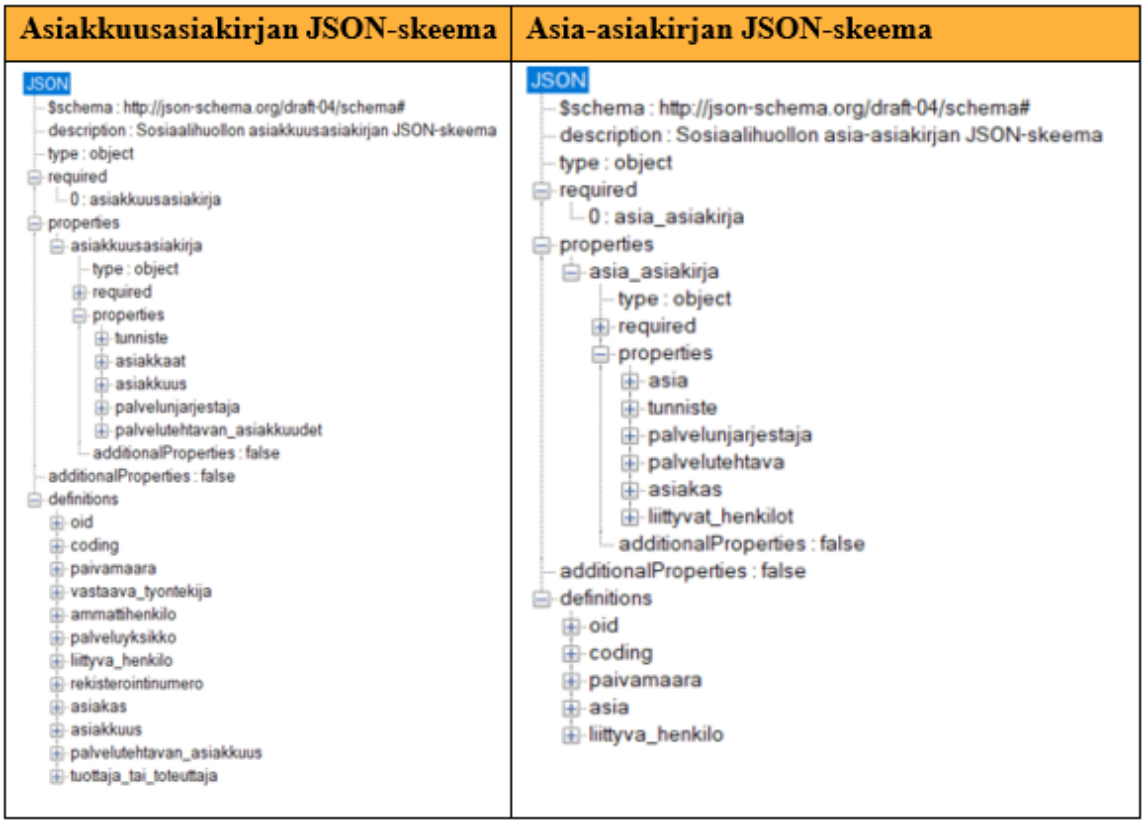
KUVIO 9. XHTML-asiakirjan sallitut elementit (Kanta 2019b, 6).

4.3.4 Rakenteinen JSON-asiakirja

Asiakkuus- ja asia-asiakirjat viedään arkistoon XHTML-muodon lisäksi JSON-muodossa, jotta niitä voidaan käsitellä myös koneellisesti. Asiakasasiakirjoista ei ole pakko muodostaa JSON-muotoista asiakirjaa, joten sitä ei tässä ratkaisussa

tehty. Kaikkien asiakirjatyyppien rakenteisissa esitysmuodoissa käytetään pääasiassa kaikille asiakirjoille yhteisiä JHS 170 -suosituksen mukaisia tietotyyppejä.

Asia-asiakirjalle sekä Asiakkuusasiakirjalle on kummallekin oma JSON-skeema (KUVIO 10), jonka mukaan JSON-asiakirjat muodostetaan. Skeemassa kuvataan asiakirjatyypille pakolliset ja ei-pakolliset elementit sekä niiden tietotyypit.



KUVIO 10. Asiakkuusasiakirjan ja Asia-asiakirjan JSON-skeemat.

4.3.5 PDF-muotoinen asiakirja

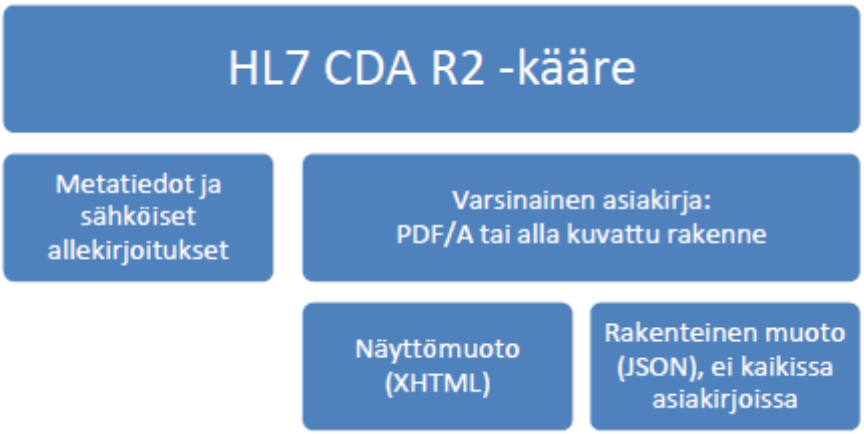
Asiakasasiakirja ja Liiteasiakirja voidaan XHTML-muodon sijaan arkistoida PDF-muotoisena. Tällöin PDF-muotoinen asiakirja pitää muuntaa pitkäaikaissäilytykseen tarkoitettuun PDF/A-1b -muotoon. (Kanta 2019b). Migraatiossa lähdejärjestelmästä poimittu tekstimuotoinen asiakirja voidaan konvertoida sellaisenaan ilman mitään muutoksia PDF/A-1b -muotoon.

4.3.6 HL7 CDA R2 -kääre

Sosiaalihuollon asiakirjastandardi määrittelee yhtenäisen tiedostotason esitysmuodon sosiaalihuollon asiakirjojen arkistointiin ja siirtämiseen. Sosiaalihuollon asiakirjastandardin mukaan

”Sosiaalihuollon asiakirjat esitetään Health Level 7 Clinical Document Architecture, Release 2 (HL7 CDA R2) -standardin mukaisessa muodossa. CDA-muotoinen kääre sisältää asiakirjan metatiedot ja itse asiakirjan sisällön joko sekä rakenteisessa muodossa että näyttömuodossa tai ainoastaan näyttömuodossa.” (Kanta 2019b, 2.)

Kuviossa 11 on esitetty CDA-kääreen ja sen sisältämien komponenttien rakenne. CDA-kääre noudattaa rakenteeltaan kansainvälistä CDA R2 -skeemaa, mutta sitä on Kanta-palvelun toimesta laajennettu mahdollistamaan asiakirjan rakenteisen muodon tallennus. Näyttömuotoinen (XHTML tai PDF/A) sekä rakenteinen muoto (JSON) tallennetaan CDA-kääreessä Base64-koodatussa muodossa omiin elementteihinsä.



KUVIO 11. Sosiaalihuollon asiakirja metatietoineen CDA-kääreessä (Kanta 2019b)

Kuviossa 12 tiivistettynä valmis arkistoitava XML-muotoinen CDA-kääre. Näyttömuotoinen XHTML-asiakirja on viety nonXMLBody-osioon text-elementtiin. Text-elementtiin viedään myös PDF/A-1b -muotoinen asiakirja. Rakenteinen JSON-asiakirja viedään nonXMLBody-osioon hl7fi:json-elementtiin. Sosiaalihuollon

asiakirjat allekirjoitetaan sähköisesti ja XML-muotoinen allekirjoitus liitetään mukaan CDA-kääreeseen. Allekirjoitus kohdistetaan CDA-kääreen nonXMLBody-rakenteeseen eli yhdellä allekirjoituksella allekirjoitetaan kaikki CDA-kääreeseen pakatut asiakirjat.

```
<?xml version="1.0" encoding="UTF-8"?>
<ClinicalDocument xmlns="urn:hl7-org:v3" xmlns:hl7fi="urn:hl7finland">
    ...
    <hl7fi:localSocialHeader>
        ....
        <hl7fi:signatureCollection>
            <hl7fi:signature ID="signature1">
                <hl7fi:signatureDescription code="3" codeSystem="1.2.246.537.5.40127.2006" codeSystemName=
                "KanTa-palvelut - Sähköisen allekirjoituksen tyyppi" displayName="Järjestelmäallekirjoitus /
                perusjärjestelmä"/>
                <hl7fi:signatureTimestamp ID="timestamp1">2020-05-15T13:37:08Z</hl7fi:signatureTimestamp>
                <ds:Signature xmlns:ds="http://www.w3.org/2000/09/xmldsig#" Id="signature1">
                    <ds:SignedInfo>
                        <ds:CanonicalizationMethod Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"/>
                        <ds:SignatureMethod Algorithm="http://www.w3.org/2001/04/xmldsig-more#rsa-sha256"/>
                        <ds:Reference URI="">
                        <ds:Reference URI="">
                    </ds:SignedInfo>
                    <ds:SignatureValue>
                    Rh9AQOm2MnVFlY56vUJrZ15ftd95W0MeGTkS3C7Ogtmi9o/fC3CvDx6cp6hQPO0E+7f+RnNxvS52r7QzVPPS0nE0Bp9Hay41iz17ZY
                    93jzJOsUJHQbj4hV3MwyMISaKGEBCah4IhQEqvj2fFUpX6i8ydw3V6bBYvDYBFhS3ibHwW8g4ft9G6zhmg94JznrCLGBqLl+QCRpXb
                    rlZnz4OO+QgdicCKc0QrxkI2UFc153ZrposXKGdZPFa9DLgTkzH5hV/08KjwPjNxFq/IZKNZ8umizEjjh4VPNnN/1irRDTQ3whg70T
                    YKorN/sBjADeUxQzM8Y9Qq7/ATKB8pIa3I1w==</ds:SignatureValue>
                    <ds:KeyInfo>
                </ds:Signature>
            </hl7fi:signature>
        </hl7fi:signatureCollection>
    </hl7fi:localSocialHeader>
    <component>
        <nonXMLBody ID="1.2.246.10.1387809.10.10.95.2.998877663.1">
            <text mediaType="application/xhtml+xml" representation="B64">
            PD94bWwgdmVyc2lvbj0iMS4wIiBlbmNvZGluZz0iVVRGLTgiPz48ZGl2IHhtbG5zPSJodHRw0i8vd3d3LnczLm9yZy8xOTk5L3hodG1sIiBjbG
            Fzcz0ic29jLWRvY3VtZW50Ij48ZGl2IGNsYXNzPSJzb2MtaGVhZGVyIi8+PGJyLz48ZGl2IGNsYXNzPSJzb2MtbWFpbi13cmFwcGVyIj48aDEg
            Y2xhc3M9InNvY1mdWxsdGl0bGUiPkFzaWFra3V1c2FzaWFraXJqYTwvaDE+PHA+QXNpYWtpcmphbiB0dW5uaXN0ZTogMS4yLjI0Ni4xMC4xMz
            g3ODA5LjEwLjEwLjk1LjIuOTk4ODc3NjYzPC9wPjxoMiBjbGFzcz0ic29jLXRpdGxlIj5Bc2lha2thYW4gdGllZG90PC9oMj48ZGl2IGNsYXNz
            PSJzb2MtbGlzdCI+PGRpdiBjbGFzcz0ic29jLWl0ZW0iPjxkaXYgY2xhc3M9InNvYy1jb250ZW50Ij48cD5Bc2lha2thYW4gc3VrdW5pbWk6IF
            Rlc3RpbmVuPGJyLz5Bc2lha2thYW4gZXRlbmltZXQ6IEthbGxlIEFsdmFyaTxici8+QXNpYWtrYWFuIGhlbmtpbMO2dHVubnVzOiAyNjA0Nzct
            OTMzWTxici8+QXNpYWtrYWFuIGtvdGlrdW50YTogVE9ETzxici8+PC9wPjwvZGl2PjwvZGl2PjxoMiBj</text>
            <hl7fi:json>
            eyJhc2lha2t1dXNhc2lha2lyamEiOnsicGFsdmVsdW5qYXJqZXN0YWphIjp7InBhbHZlbHV5a3Npa290IjpbeyJhc2lha2t1dXMiOnsiYWxrYW
            1pc3B2bSI6IjE5OTIwMTAxIn0sInRlbm5pc3RlIjp7InZhbHVlIjoiMS4yLjI0Ni4xMC4xMzg3ODA5LjEwLjEwIn19XX0sImFzaWFra2FhdCI6
            W3siaGVua2lsb3R1bm51cyI6eyJzeXN0ZW0iOiJ1cm46b2lkOjEuMi4yNDYuMjEiLCJ2YWx1ZSI6IjI2MDQ3Ny05MzNZIn0sInN1a3VuaW1pIj
            oiVGVzdGluZW4iLCJldHVuaW1ldCI6WyJLYWxsZSIsIkFsdmFyaSJdfV0sImFzaWFra3VlcyI6eyJhbGthbWlzcHZtIjoiMTk5MjAxMDEiLCJw
            YWF0dHltaXNwdm0iOiIxOTkyMTIzMSIsInZhbmhhX2FzaWFra3V1cyI6dHJ1ZX0sInRlbm5pc3RlIjp7InZhbHVlIjoiMS4yLjI0Ni4xMC4xMz
            g3ODA5LjEwLjEwLjk1LjIuOTk4ODc3NjYzIn0sInBhbHZlbHV0ZWh0YXZhbl9hc2lha2t1dWRldCI6W3sicGFsdmVsdXRlaHRhdmEiOnsic3lz
            dGVtIjoidXJuOm9pZDoxLjIuMjQ2LjUzNy42LjEyMjEuMjAxNjAxIiwiY29kZSI6IjIwIiwiZGlzcGxheSI6IlZhbW1haXNwYWx2ZWx1dCJ9LC
            JhbGthbWlzcHZtIjoiMTk5MjAxMDEiLCJha3RpaXZpbmVuIjpmYWxzZX1dfX0=</hl7fi:json>
        </nonXMLBody>
    </component>
</ClinicalDocument>
```


KUVIO 12. Arkistointivalmis CDA-kääre.

4.4 Kehitysvaihe

4.4.1 Prosessin yleiskuvaus

Sosiaalihuollon asiakirjastandardin mukaisesti arkistoitava kokonaisuus muodostui tässä ratkaisussa seuraavista asiakirjatasoista:

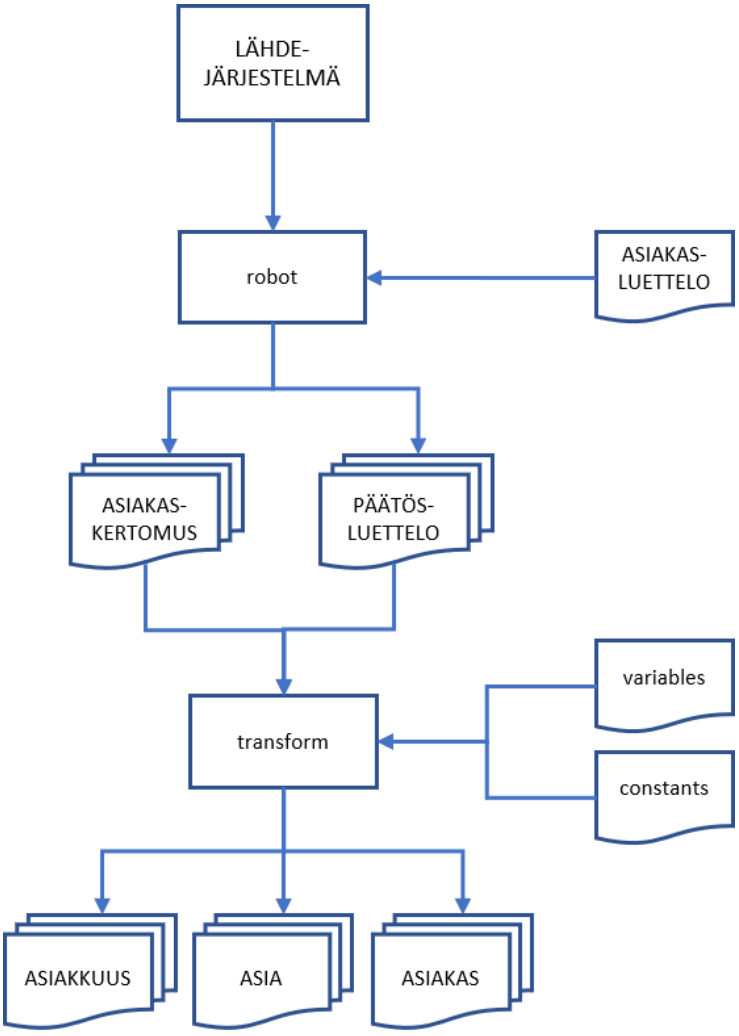
- Asiakkuus-asiakirja (vähintään yksi asiakkuus per henkilö)
- Asia-asiakirja (vähintään yksi asia per asiakkuus)

- Asiakas-asiakirja (1-n kappaletta asiakasasiakirjoja per asia, sisältää varsinaisen asiakaskohtaiset sosiaalihuollon merkinnät).

Kehitysvaiheessa ohjelmoitiin ja konfiguroitiin ohjelmistorobotti poimimaan tarvittavat datat lähdejärjestelmästä. Lisäksi rakennettiin ohjelmistot asiakirjojen muokkaamiseen ja paketoimiseen CDA R2 -kääreeseen. Kuviossa 13 on kehitysvaiheessa rakennetun migraatioprosessin kuvaus, joka karkealla tasolla on seuraavanlainen:

- Ohjelmistorobotti lukee poimittavien asiakkaiden sosiaaliturvatunnukset Perusturva-asiakasluettelo -nimisestä txt-muotoisesta tiedostosta, hakee asiakasta koskevat tiedot lähdejärjestelmästä ja kirjoittaa kaikki asiakasta koskevat päätökset Päätösluettelo-nimiseen txt-muotoiseen tiedostoon. Lisäksi jokaisesta päätöksestä tuotetaan oma Asiakaskertomus-niminen txt-muotoinen tiedosto.
- Koska lähdejärjestelmästä ei saada kaikkia Kanta-palvelun vaatimia metatietoja niin migraatiossa metatietoja pitää rikastaa. Asiakas on tuottanut rikastettavat metatiedot Excel-muotoisessa tiedostossa, josta kaikille asiakirjoille yhteiset metatiedot viedään constants-nimiseen parametritiedostoon ja asiakirjatyyppikohtaiset metatiedot variables-nimiseen parametritiedostoon.
- Päätösluettelo-tiedostoista sekä parametrina saaduista metatiedoista muodostetaan Asiakkuus- ja Asiakirjoille XHTML- ja JSON-muotoiset sisällöt (esimerkit näistä liitteissä 4-7).
- Tekstimuotoinen Asiakaskertomus konvertoidaan PDF/A-1b muotoon.
- Muodostetaan CDA R2 -muotoiset Asiakkuus-, Asia- ja Asiakasasiakirjat. PDFA-1b-, JSON- ja XHTML-muotoisille asiakirjoille tehdään BASE64-koodaus ennen niiden paketoimista CDA R2 -kääreeseen.
- CDA-rakenteessa olevat asiakirjat allekirjoitetaan sähköisesti ja lisätään XML-muotoinen allekirjoitus CDA-rakenteeseen.

Migraatioprosessin lopputuloksena saadaan arkistointivalmiita CDA R2 -muotoisia asiakirjoja. Prosessin aikana tarkistetaan näyttömuotoisten ja rakenteisten asiakirjojen oikeellisuus XHTML- ja JSON-skeemoja vastaan. Lisäksi validoidaan prosessin lopputuloksena syntyneet CDA R2 -kääreet CDA-skeemaa vastaan.



KUVIO 13. Sosiaalihuollon asiakastietojärjestelmän Kanta-migraatioprosessi.

4.4.2 Testaus

Asiakirjat validoidaan teknisesti ennen kuin ne viedään sähköiseen arkistoon. Validoinnissa tarkistetaan, että asiakirjat täyttävät niille asetetut rakenteelliset tekniset vaatimukset. Sosiaalihuollon asiakirjastandardin (Kanta 2019b) mukaan teknisessä validoinnissa varmistetaan, että:

- XHTML-muotoinen asiakirja on muodostettu XHTML-FHIR -skeeman mukaisesti
- PDF/A-muotoinen asiakirja on vaatimusten mukainen
- JSON-asiakirja on JSON-syntaksin mukainen

- CDA-asiakirja on skeeman mukainen
- Asiakirjoissa on käytetty vain kansallisesti määriteltyjä koodistoja
- Asiakirjat on allekirjoitettu ja allekirjoitus on validi

Kun asiakirjat läpäisivät teknisen validoinnin, ne ladattiin Visma Consulting Oy:n sähköiseen arkistoon käyttäjien tekemää sisältötarkistusta varten. Testauksessa varmistuttiin, että kaikki lähdejärjestelmästä puretut asiakirjat on viety sähköiseen arkistoon ja niiden sisältö vastaa lähdejärjestelmässä olevien asiakirjojen sisältöä.

5 POHDINTA

Opinnäytetyön tavoitteena oli selvittää millainen olisi yleisluontoinen malli sovellusten välisen migraatioprojektin läpivientiin. Datamigraatioiden merkitys IT-teknologiassa on kasvamassa, koska vanhoja, jopa kymmeniä vuosia käytössä olevia sovelluksia modernisoidaan, jotta niiden käytössä voidaan hyödyntää uuden teknologian tarjoamia ominaisuuksia. Vaikka migraatioprojektit periaatteessa ovat samantyyppisiä, siirretään dataa sovelluksesta toiseen, niin niissä kaikissa on kumminkin omat erityispiirteensä ja haasteensa. Sovellukset poikkeavat toisistaan sekä liiketoimintaobjektien, että teknisten objektien tasolla. Jopa samassa käyttötarkoituksessa olevat, mutta eri toimittajien sovellukset voivat poiketa huomattavastikin toisistaan sovelluksen kehittämisvaiheessa tehtyjen teknisten ratkaisujen vuoksi. Dataa ei pelkästään siirretä vaan sille joutuu tekemään erilaisia muokkaustoimen piteitä, jotta se saadaan vietyä lähdejärjestelmään.

Migraatioprojekti voidaan jakaa neljään eri vaiheeseen: esiselvitys, data-analyysi, kehitystyö ja varsinainen migraatio. Varsinainen migraatio pitää sisällään tiedon purkamisen lähdejärjestelmästä sekä sen muokkaamisen ja lataamisen kohdejärjestelmään.

Migraatioprojektin teknistä toteutusta edeltävät esiselvitys- ja analyysivaiheet ovat äärimmäisen tärkeitä koko projektin onnistumisen kannalta. Mikäli niissä vaiheissa tehdään virheitä tai jätetään jotain huomioimatta, niin se kostautuu projektin myöhemmissä vaiheissa ylimääräisenä työnä ja kustannuksena. Vaiheissa perehdytään sekä kohde- että lähdejärjestelmien tekniseen arkkitehtuuriin, tietomalleihin, rajapintoihin, dataan sekä sovelluksen toimintaympäristöön. Näiden perusteella tehdään suunnitelma migraation toteuttamiseksi.

Migraationprojektin teknisessä toteuttamisvaiheessa tehdään tarvittavat ohjelmistot ja skriptit, jotka muokkaavat lähdejärjestelmästä poimitun datan kohdejärjestelmän vaatimaan muotoon. Toteuttamisvaihe on iteratiivinen prosessi, jossa testataan datan purkamiseen, muokkaamiseen ja lataamisen käytetyt ohjelmistot. Lisäksi siinä vaiheessa testataan koko migraatioprosessi alusta loppuun saakka, jotta voidaan todentaa sen toimivuus.

Varsinainen tuotantomigraatio voidaan tehdä joko yhdellä kertaa tai vaiheittain. Yhdellä kertaa tehtävä migraatio on riskialttiimpi, koska siinä lähdejärjestelmä poistuu käytöstä ja siirrytään käyttämään kohdejärjestelmää. Migraatio pitää suunnitella, aikatauluttaa ja testata äärimmäisen tarkasti ja varautua myös palaamaan vanhan sovelluksen käyttöön, mikäli migraatiossa jokin menee pahasti pieleen. Onnistuessaan kertaluonteinen migraatio säästää aikaa ja rahaa, koska vanhan sovelluksen käytöstä voidaan heti luopua. Vaiheittaisessa migraatiossa lähdejärjestelmän sisältämä data siirretään osissa kohdejärjestelmään. Tämä voi kestää pitkäänkin, jopa useita viikkoja tai kuukausia. Tällöin käytössä onko koko ajan kaksi järjestelmää, mikä lisää kustannuksia ja on käyttäjillekin hankalaa. Toisaalta vaiheittainen migraatio on turvallisempi, koska migraation epäonnistuessa ei tarvitse tehdä palautustoimenpiteitä.

Tämän opinnäytetyön yhtenä osa-alueena oli toimeksiantajayritykselle tehtävä kehittämistehtävä. Tehtävänä oli laajentaa toimeksiantajan Kanta-välityspalvelua siten, että sillä voidaan vastaanottaa usealta eri toimittajalta ja useassa eri terveyden- tai sosiaalihuollon tietojärjestelmässä tuotettuja asiakirjoja ja muokata niitä Kanta-palveluun sopivaan muotoon. Kehittämistehtävässä lähdejärjestelmänä oli käytöstä poistettu sosiaalihuollon tietojärjestelmä, josta data piti saada purettua ja muokattua. Esiselvitysvaiheessa kävi ilmi, että tietojärjestelmän toimittajaa ei enää ole olemassakaan, eikä sovelluksen dokumentaatiokaan ollut kovin kattavaa. Haasteeksi tulikin miten tiedot saadaan sovelluksesta purettua, koska sillä ei ollut avoimia rajapintoja, joiden kautta ne olisi voitu hakea ja toisaalta tietokannan rakennekin oli hyvin monimutkainen. Päätimmekin hakea tiedot käyttöliittymän kautta ohjelmistorobotiikan avulla. Näin tiedot saatiin purettua txt-muotoisiin tiedostoihin, jotka sitten muokattiin Kanta-yhteensopivaksi.

Lähteitä etsiessä kävi ilmi, että kovin paljon datamigraatiosta ei löydy tieteellisiä julkaisuja. Florian Matthes, Christopher Schulz ja Klaus Haller ovat tutkineet aihetta yhdessä ja erikseen ja kirjoittaneet siitä useampiakin julkaisuja. Niistä sainkin paljon arvokasta tietoa omaan tutkimukseeni. Tieteellisten julkaisujen lisäksi lähteenä olen käyttänyt migraatiopalveluja tarjoavien yritysten kirjoittamia artikkeleita, joista mielestäni löytyi viimeisintä tietoa migraatioissa käytettävistä ohjelmistoista ja menetelmistä. Toisaalta niihin piti suhtautua varauksella, koska niissä artikkeleissa yrityksen mainostavat omia tuotteitaan ja palveluitaan.

Opinnäytetyö saavutti sille asetetut tavoitteet. Tässä migraatioprosessi on kuvattuna vielä hyvin karkealla tasolla. Jatkossa sitä voisi tarkentaa esimerkiksi kuvaamalla kustakin vaiheesta syntyvät tuotokset. Tässä työssä ei ole otettu kantaa migraatioissa käyttäviin ohjelmistoihin ja apuvälineisiin. Yhtenä kehittämistehtävänä voisikin olla selvittää millaisia työvälineitä on tarjolla esimerkiksi tietokantojen välillä tehtävään migraatioon. Ohjelmistorobotiikka on nopeasti kasvava IT-teknologian osa-alue. Sen hyödyntämistä voisi myös selvittää tarkemmin. Kehittämistehtävässä hyödynsimme robotiikkaa tietojen purkamisessa, mutta yhtä hyvin sitä voisi käyttää myös tietojen muokkaamiseen tai vientiin kohdejärjestelmään.

LÄHTEET

Accelario. 2020. What are the different types of data migration. Luettu 15.11.2020. https://accelario.com/articles/what-are-the-different-types-of-database-migration/

Arkistolaitos. 2014. Sosiaalihuollon asiakasasiakirjojen ja -tietojen pysyvä säilytys. AL/20064/07.01.01.03.01/2014.

CloverDX. Everything You Need to Know About Data Migrations. Luettu 9.11.2020. https://www.cloverdx.com/explore/data-migration

Eng, D. Process evaluation of general data migration guidelines – A comparative study. Linköping Universitet.

Federal Student Aid. 2019. Data Migration Roadmap Guidance. Luettu 07.12.2020 https://studentaid.gov/sites/default/files/fsawg/static/gw/docs/ciolibrary/Data_Migration_Roadmap_Guidance.pdf

HL7. FHIR XHTML Schema. Luettu 15.11.2020. http://www.hl7.org/implement/standards/fhir/fhir-xhtml.xsd

Honkavaara, J-P. 2013. ETL-prosessin parhaita käytäntöjä tietovaraston rakentamisessa. Pro gradu -tutkielma. Tampereen yliopisto.

Hughes, J. 2018. An Introduction to Secure Shell Access and Secure File Transfer Protocol. Luettu 28.12.2020. https://torquemag.io/2018/02/introduction-to-ssh-and-sftp/

Kanta. 2019a. Vanha asiakastieto Sosiaalihuollon asiakastiedon arkistoon.

Kanta. 2019b. Sosiaalihuollon asiakirjastandardi. Versio 2.6.

Kela. Kansallinen koodistopalvelu. AR/YDIN – Näkymät.

Loshin, D. 2009. Master Data Management. The MK/OMG Press.

Morris, J. Practical Data Migration, 2012. Second edition. BCS, The Chartered Institute for IT. (https://www-proquest-com.libproxy.tuni.fi/publication/76016?OpenUrlRefId=info:xri/sid:primo)

Matthes, F. Schulz, C. 2011. Towards an integrated data migration process model. State of the art & literature overview. Technische Universität Munchen.

Matthes, F. Schulz, C, Haller K. 2011. Testing & Quality Assurance in Data Migration Projects.

Oracle. 2011. Successful Data Migration. An Oracle White Paper.

Pund, M. Jain, P. 2016. System and Process for Data Transformation and Migration from Libsys to Koha. International Research Journal of Engineering and Technology (IRJET).

Raicea R. 2017. How Pretty Good Privacy works, and how you can use it for secure communication. Luettu 03.01.2021. https://www.freecodecamp.org/news/how-does-pretty-good-privacy-work-3f5f75ecea97/

Russom, P. 2006. Best Practices in Data Migration. The Data Warehousing Institute.

Sarmah, S. Data Migration. 2018. Scientific & Academic Publishing.

SoftwareTestingHelp. Data Migration Testing Tutorial: A Complete Guide. Luettu 13.11.2020. https://www.softwaretestinghelp.com/data-migration-testing/

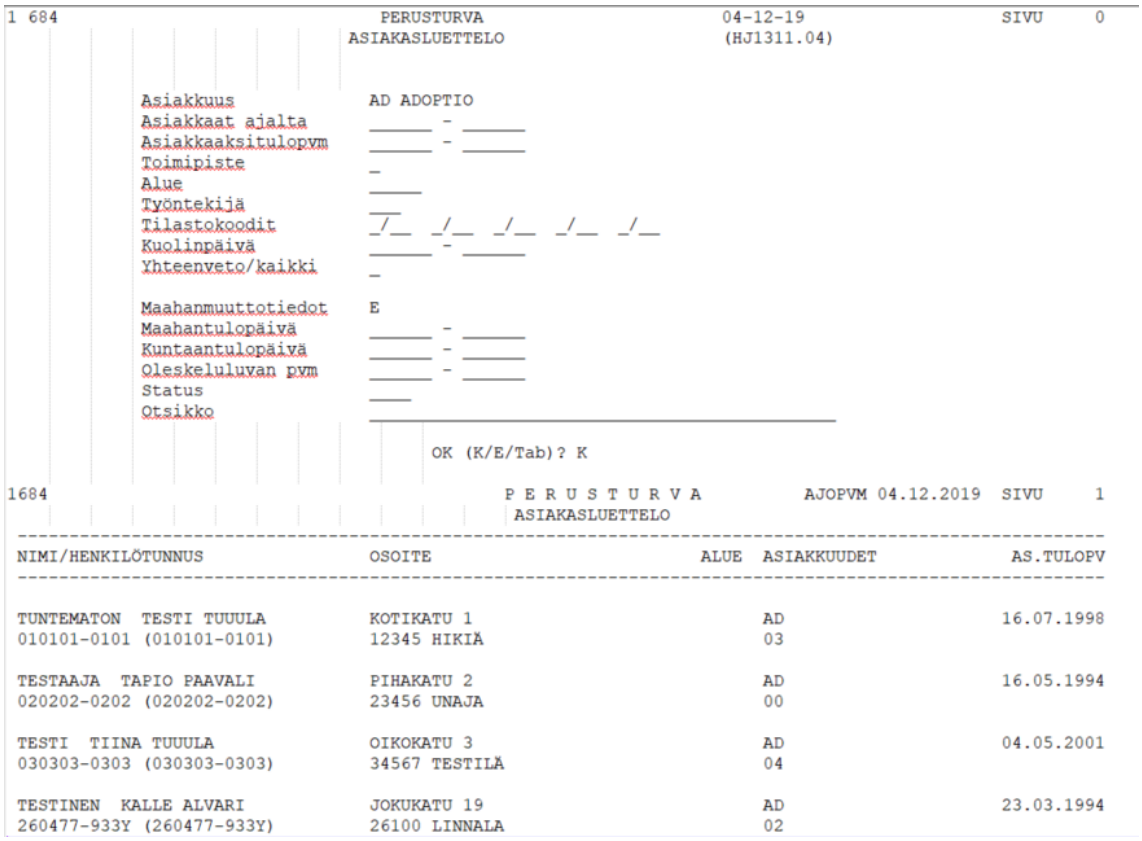
SoteDigi Oy. Luettu 17.11.2020. https://digifinland.fi/wp-content/uploads/2020/10/Vanhojen-tietojen-arkistointi_final_1_10.pdf

Thalheim, B. Wang, Q. 2012. Data migration: A theoretical perspective.

Vuollet, J. Nurmi, J. Välkkilä, L. 2020. Vanhojen tietojen arkistointi -raportti.

LIITTEET

Liite 1. Perusturva asiakasluettelo



```
1 684                          PERUSTURVA                    04-12-19                SIVU     0
                               ASIAKASLUETTELO               (HJ1311.04)

          Asiakkuus            AD ADOPTIO
          Asiakkaat ajalta     ______ - ______
          Asiakkaaksitulopvm   ______ - ______
          Toimipiste           _
          Alue                 _____
          Työntekijä           __
          Tilastokoodit        _/__  _/__  _/__  _/__  _/__
          Kuolinpäivä          ______ - ______
          Yhteenveto/kaikki    _

          Maahanmuuttotiedot   E
          Maahantulopäivä      ______ - ______
          Kuntaantulopäivä     ______ - ______
          Oleskeluluvan pvm    ______ - ______
          Status               ____
          Otsikko              ________________________________________

                               OK (K/E/Tab)? K

1684                                P E R U S T U R V A          AJOPVM 04.12.2019  SIVU     1
                                     ASIAKASLUETTELO
 ------------------------------------------------------------------------------------------------------
 NIMI/HENKILÖTUNNUS                  OSOITE                          ALUE  ASIAKKUUDET      AS.TULOPV
 ------------------------------------------------------------------------------------------------------

 TUNTEMATON  TESTI TUUULA            KOTIKATU 1                            AD               16.07.1998
 010101-0101 (010101-0101)           12345 HIKIÄ                           03

 TESTAAJA  TAPIO PAAVALI             PIHAKATU 2                            AD               16.05.1994
 020202-0202 (020202-0202)           23456 UNAJA                           00

 TESTI  TIINA TUUULA                 OIKOKATU 3                            AD               04.05.2001
 030303-0303 (030303-0303)           34567 TESTILÄ                         04

 TESTINEN  KALLE ALVARI              JOKUKATU 19                           AD               23.03.1994
 260477-933Y (260477-933Y)           26100 LINNALA                         02
```

Liite 2. Päätösluettelo

A-KLINIKKA

SÄVELTIE 112
26100 LINNALA

PÄÄTÖSOTE
A-KLINIKKA
TV /940692
27.04.1994

TESTINEN KALLE ALVARI
JOKUKATU 19
26100 LINNALA

260477-933Y

PÄÄTTÄJÄ NIPINII TEIJA
PUKARIN

SAIRAANHOITAJA

OIKEAKSI TODISTAMINEN

Asianmukaisesti allekirjoitetusta pöytäkirjasta kirjoitetun otteen oikeaksi todistaa:
22.04.1994

allekirjoitus

MUUTOKSENHAKU

Tähän viranhaltijan päätökseen tyytymättömällä on oikeus saattaa päätös sosiaalilautakunnan käsiteltäväksi. Muutoksenhakuaika on neljätoista (14) päivää siitä päivästä, jolloin asianomainen sai tiedon päätöksestä lukuunottamatta sitä päivää. Vaatimus päätöksen siirtämisestä sosiaalilautakunnan käsittelyyn tapahtuu jättämällä kirjallinen muutoksenhakuvaatimus sekä tämä päätös henkilökohtaisesti tai postitse seuraavaan osoitteeseen:
LINNALA KAUPUNGIN SOSIAALILAUTAKUNTA, käyntios. Kalliokatu 1, 26100 LINNALA - postios. PL 59, 26101 LINNALA.

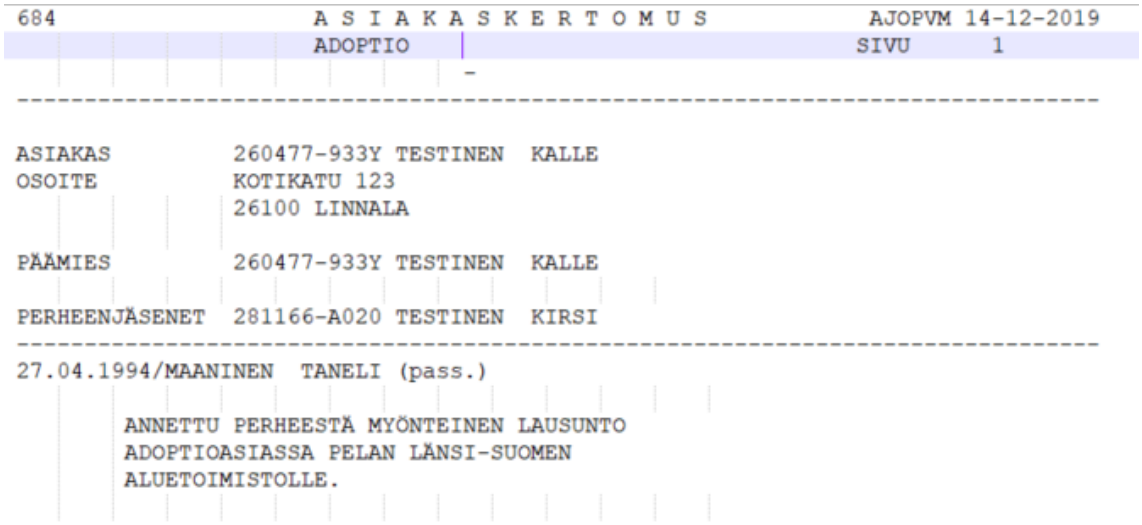
TIEDOKSIANTOTODISTUS

Tämä päätös on annettu tiedoksi 27.04.1994 postitse/valtuutetulle/asianomaiselle

Tiedoksiantaja

Tiedoksisaaja
(mikäli annettu henk.koht.)

Liite 3. Asiakaskertomus



```
684                    A S I A K A S K E R T O M U S          AJOPVM 14-12-2019
                       ADOPTIO                                SIVU      1
                                -
-------------------------------------------------------------------------------

ASIAKAS          260477-933Y TESTINEN  KALLE
OSOITE           KOTIKATU 123
                 26100 LINNALA

PÄÄMIES          260477-933Y TESTINEN  KALLE

PERHEENJÄSENET   281166-A020 TESTINEN  KIRSI
-------------------------------------------------------------------------------
27.04.1994/MAANINEN  TANELI (pass.)

        ANNETTU PERHEESTÄ MYÖNTEINEN LAUSUNTO
        ADOPTIOASIASSA PELAN LÄNSI-SUOMEN
        ALUETOIMISTOLLE.
```

Liite 4. XHTML-muotoinen Asiakkuusasiakirja

```
<?xml version="1.0" encoding="UTF-8"?>
<div xmlns="http://www.w3.org/1999/xhtml" class="soc-document">
    <div class="soc-header"/>
    <br/>
    <div class="soc-main-wrapper">
        <h1 class="soc-fulltitle">Asiakkuusasiakirja</h1>
        <p>Asiakirjan tunniste: 1.2.246.10.1387809.10.10.95.2.998877663</p>
        <h2 class="soc-title">Asiakkaan tiedot</h2>
        <div class="soc-list">
            <div class="soc-item">
                <div class="soc-content">
                    <p>Asiakkaan sukunimi: Testinen<br/>Asiakkaan etunimet: Kalle Alvari<br/>
                    Asiakkaan henkilötunnus: 260477-933Y<br/>Asiakkaan kotikunta: TODO<br/>
                    </p>
                </div>
            </div>
        </div>
        <h2 class="soc-fulltitle">Asiakkuuden tiedot</h2>
        <div class="soc-list">
            <div class="soc-item">
                <div class="soc-content">
                    <p>Vanha asiakkuus: True
                        <br/>Asiakkuuden alkamisaika: 01.01.1992<br/>Asiakkuuden päättymisaika:
                        31.12.1992<br/>
                    </p>
                </div>
            </div>
            <h2 class="soc-title">Palvelunjärjestäjän tiedot</h2>
            <div class="soc-item">
                <h3 class="soc-title">Palvelunjärjestäjän palveluyksikkö</h3>
                <div class="soc-content">
                    <p>Palveluyksikön organisaatiokoodi: 1.2.246.10.1387809.10.10<br/>Palveluyksikön
                    asiakkuuden alkamisaika: 01.01.1992<br/>Palveluyksikön asiakkuuden päättymisaika:
                    31.12.1992<br/>
                    </p>
                </div>
            </div>
            <h2 class="soc-title">Palveluntehtävän asiakkuuden tiedot</h2>
            <div class="soc-item">
                <div class="soc-content">
                    <p>Palvelutehtävä: 1.2.246.537.6.1221.201601.20
                        <br/>Palvelutehtävän asiakkuuden alkamisaika: 01.01.1992<br/>Palvelutehtävän
                        asiakkuuden päättymisaika: 31.12.1992<br/>Palvelutehtävän asiakkuuden tila:
                        False
                        <br/>
                    </p>
                </div>
            </div>
        </div>
    </div>
    <div class="soc-footer"/>
</div>
```

Liite 5. XHTML-muotoinen Asia-asiakirja

```
<?xml version="1.0" encoding="UTF-8"?>
<div xmlns="http://www.w3.org/1999/xhtml" class="soc-document">
    <div class="soc-header"/>
    <br/>
    <div class="soc-main-wrapper">
        <h1 class="soc-fulltitle">Asia-asiakirja</h1>
        <p>Asiakirjan yksilöintitunnus: 1.2.246.10.1387809.10.10.95.2.998877662</p>
        <h2 class="soc-title">Asiakkaan tiedot</h2>
        <div class="soc-list">
            <div class="soc-item">
                <div class="soc-content">
                    <p>Asiakkaan sukunimi: Testinen<br/>Asiakkaan etunimet: Kalle Alvari<br/>Asiakkaan syntymäaika: 26.04.1977<br/>
                    Asiakkaan henkilötunnus: 260477-933Y<br/>
                    </p>
                </div>
            </div>
        </div>
        <h2 class="soc-fulltitle">Asia</h2>
        <div class="soc-list">
            <h2 class="soc-title">Asian tiedot</h2>
            <div class="soc-item">
                <div class="soc-content">
                    <p>Asiatunnus: 1.2.246.10.1387809.10.10.95.2.998877665<br/>Asian nimi: Sosiaalihuolto - Sosiaalihuollon asia
                        <br/>Asian tila: Suljettu
                        <br/>Asian avauspäivä: 01.01.1992<br/>Asian päättymispäivä: 31.12.1992<br/>Palvelutehtävä: Vammaispalvelut
                        </p>
                </div>
            </div>
            <h2 class="soc-title">Palvelunjärjestäjän tiedot</h2>
            <div class="soc-item">
                <div class="soc-content">
                    <p>Palvelunjärjestäjän nimi: Testilän kuntayhtymä<br/>Palvelunjärjestäjän organisaatiokoodi:
                    1.2.246.10.1387809.10.10<br/>
                    </p>
                </div>
            </div>
        </div>
    </div>
    <div class="soc-footer"/>
</div>
```

Liite 6. JSON-muotoinen Asiakkuusasiakirja


```
{
    "asiakkuusasiakirja": {
        "palvelunjarjestaja": {
            "palveluyksikot": [
                {
                    "asiakkuus": {
                        "alkamispvm": "19920101"
                    },
                    "tunniste": {
                        "value": "1.2.246.10.1387809.10.10"
                    }
                }
            ]
        },
        "asiakkaat": [
            {
                "henkilotunnus": {
                    "system": "urn:oid:1.2.246.21",
                    "value": "260477-933Y"
                },
                "sukunimi": "Testinen",
                "etunimet": [
                    "Kalle",
                    "Alvari"
                ]
            }
        ],
        "asiakkuus": {
            "alkamispvm": "19920101",
            "paattymispvm": "19921231",
            "vanha_asiakkuus": true
        },
        "tunniste": {
            "value": "1.2.246.10.1387809.10.10.95.2.998877663"
        },
        "palvelutehtavan_asiakkuudet": [
            {
                "palvelutehtava": {
                    "system": "urn:oid:1.2.246.537.6.1221.201601",
                    "code": "20",
                    "display": "Vammaispalvelut"
                },
                "alkamispvm": "19920101",
                "aktiivinen": false
            }
        ]
    }
}
```

Liite 7. JSON-muotoinen Asia-asiakirja


```
{
    "asia_asiakirja": {
        "palvelunjarjestaja": {
            "nimi": "Testilän kuntayhtymä",
            "tunniste": {
                "value": "1.2.246.10.1387809.10.10"
            }
        },
        "asiakas": {
            "henkilotunnus": {
                "system": "urn:oid:1.2.246.21",
                "value": "260477-933Y"
            },
            "sukunimi": "Testinen",
            "etunimet": [
                "Kalle",
                "Alvari"
            ]
        },
        "asia": {
            "avauspvm": "19920101",
            "nimi": {
                "system": "urn:oid:1.2.246.537.6.1265.201701",
                "code": "7",
                "display": "Vammaispalvelujen asia"
            },
            "paattymispvm": "19921231",
            "tunniste": {
                "value": "1.2.246.10.1387809.10.10.95.2.998877665"
            },
            "tila": {
                "system": "urn:oid:1.2.246.537.6.1501.201601",
                "code": "2",
                "display": "Suljettu"
            }
        },
        "tunniste": {
            "value": "1.2.246.10.1387809.10.10.95.2.998877662"
        },
        "palvelutehtava": {
            "system": "urn:oid:1.2.246.537.6.1221.201601",
            "code": "20",
            "display": "Vammaispalvelut"
        }
    }
}
```