Tampereen yliopisto

Niki Väänänen

PERINNEJÄRJESTELMIEN MERKITYS ORGANISAATIOSSA

Hyödyt, haitat ja mitä ottaa huomioon

Informaatioteknologian ja viestinnän tiedekunta
Kandidaattitutkielma
Toukokuu 2021

TIIVISTELMÄ

Niki Väänänen: Perinnejärjestelmien merkitys organisaatiossa - Hyödyt, haitat ja mitä ottaa huomioon
Kandidaattitutkielma
Tampereen yliopisto
Tietojenkäsittelytieteiden tutkinto-ohjelma
Toukokuu 2021

Perinnejärjestelmät (engl. legacy system) ovat yrityksessä tai organisaatiossa pitkään käytössä olleita tietojärjestelmiä. Nämä järjestelmät ovat liiketoiminnan kannalta kriittisiä, koska niihin on yleensä riippuvuuksia muista sisäisistä järjestelmistä. Niiden käyttö ja ylläpito koetaan usein kalliimmaksi ja työläämmäksi kuin olisi ihanteellista. Tämän tutkielman tavoitteena on ottaa selvää, millä tavoin perinnejärjestelmät vaikuttavat yrityksen tai organisaation toimintaan, ja selvittää, miksi niistä eroon pääseminen on niin vaikeaa. Työssä tehdään myös nopea katsaus siihen, millä tavoin näitä järjestelmiä voidaan modernisoida. Tarkoituksena on muodostaa hyvä kokonaiskäsitys perinnejärjestelmistä ja niihin kohdistuvasta päätöksenteosta.

Tämä tutkielma on toteutettu kirjallisuuskatsauksena. Lähteitä haettiin useista eri tietokannoista. Suurin osa lähteistä on peräisin IEExplore- ja ScienceDirect-tietokannoista. Näiden lisäksi mukana on yksittäisiä uutisartikkeleita. Tutkielma jakautuu kolmeen osioon. Ensimmäisessä osiossa tarkastellaan perinnejärjestelmiä ja niiden haittoja. Toisessa osiossa tarkastellaan, miksi perinnejärjestelmistä eroon pääseminen on hankalaa ja miten niiden syntymistä voidaan ennaltaehkäistä. Lopuksi kolmannessa osiossa otetaan tarkasteluun erilaiset perinnejärjestelmien modernisoimisen keinot.

Tutkimukset osoittavat perinnejärjestelmien rasittavan yritysten ja organisaatioiden IT-budjetteja merkittävästi, jopa seitsemänkymmenenviiden prosentin osuus budjetista käytetään pelkästään järjestelmien ylläpitoon. Tämän koetaan tukahduttavan merkittävästi uutta IT-alalla tapahtuvaa innovaatiota. Lisäksi COBOL ynnä muut yhtä vanhat ohjelmointikielet eivät houkuttele nuoria ohjelmoijia, minkä takia näiden järjestelmien ylläpito on vanhan sukupolven ohjelmistokehittäjien vastuulla. Perinnejärjestelmistä eroon pääsemistä vaikeuttaa erityisesti käsitys, että perinnejärjestelmät koetaan vakaiksi ja että uuteen järjestelmään siirtyminen on epävarmaa taloudellisuuden ja kannattavuuden kannalta. Perinnejärjestelmiä voidaan tarpeen tullen modernisoida, jossa tarkoituksena on palauttaa niiden käytettävyys ja yhteensopivuus vastaamaan yrityksen tai organisaation tarpeita.

Avainsanat: Perinnejärjestelmä, modernisaatio, uudelleensuunnittelu, käärintä

Sisällysluettelo

1 Johdanto

Perinnejärjestelmät tulevat esille säännöllisin väliajoin uutisotsikoissa, vaikka lukija ei välttämättä ensisilmäyksellä tajuaisikaan niistä olevan kyse. Kyseessä saattaa olla esimerkiksi jonkin suuren yrityksen tai organisaation elektronisen palvelun käyttökatko tai raportti, jossa käsitellään, millainen kuluerä vanha tietojärjestelmä on. Tällaiset ongelmat juontavat yleensä juurensa perinnejärjestelmiin.

Vanhoja tietojärjestelmiä kutsutaan perinnejärjestelmiksi. Tässä tutkielmassa on tarkoituksena tarkastella, mitä perinnejärjestelmät ovat, millä tavoin ne vaikuttavat yrityksiin ja organisaatioihin ja mitä niille voidaan tehdä. Perinnejärjestelmiä on tärkeää oppia ymmärtämään, koska teknologian kehitys kulkee varsinkin nykyään ohjelmistopuolella äärimmäisen nopeaa tahtia, mikä vääjäämättä johtaa siihen, että perinnejärjestelmiä tulee olemaan olemassa vielä myös tulevaisuudessa. Perinnejärjestelmien ymmärtäminen antaa siis yritykselle tai organisaatiolle kyvyn välttää niihin liittyviä sudenkuoppia ja näin ollen kehittää tietojärjestelmistään tehokkaampia ja käyttäjäystävällisempiä.

Aiheesta on tuotettu ymmärrettävästi paljon tutkimusmateriaalia. Varsinkin perinnejärjestelmien ongelmia on tutkittu paljon. Perinnejärjestelmät voidaan määritellä yrityksen tai organisaation toiminnan kannalta kriittisiksi järjestelmiksi, jotka vastustavat muokkausta. Näiden järjestelmien ylläpitoon käytetään yrityksissä ja organisaatioissa jopa seitsemänkymmentäviisi prosenttia IT-budjetista (Crotty & Horrocks, 2016). Ylläpitokustannukset ovat korkeita muun muassa vanhentuneiden ohjelmointikielten, kuten COBOLin käytön takia (Van Den Heuvel, 2009).

Perinnejärjestelmien haittoja ymmärretään oikein hyvin, mutta niistä eroon pääseminen tuottaa usein yritykselle tai organisaatiolle päänvaivaa. Mitä pidempään järjestelmä on ollut toiminnassa, sitä haluttomampia siitä ollaan päästämään irti (Sneed & Verhoef, 2020). Myöskään järjestelmän korvaamisesta tai modernisaatiosta saadut hyödyt jäävät usein tuntemattomiksi ennen tällaisen projektin valmistumista (Matthiesen & Bjørn, 2015).

Perinnejärjestelmiä voidaan myös siis modernisoida niiden korvaamisen sijaan. Modernisaatio on prosessi, johon ryhdytään, kun ylläpito ei riitä järjestelmän ajan tasalla pitämiseen (Comella-Dorda & muut, 2000). Modernisaation menetelmät jaetaan kahteen eri kategoriaan riippuen siitä, millaisia muutoksia järjestelmään tehdään (Van Den Heuvel, 2009).

Tämän tutkielman tarkoituksena on tarkastella kirjallisuuskatsauksen kautta, mitä perinnejärjestelmät tarkemmin ottaen ovat, ja miten niitä voidaan määritellä. Niiden aiheuttamat haitat ja tavat miten näitä ratkaistaan ovat myös erityisenä kiinnostuksen kohteena. Luvussa kaksi tehdään katsaus tämän tutkielman tutkimusmenetelmiin. Luvussa kolme määritellään termi perinnejärjestelmä ja sen ominaisuudet. Luvussa neljä määritellään perinnejärjestelmien yleisimmät haitat ja syyt niiden takana. Luvussa viisi tarkastellaan miksi kaikista haitoista huolimatta perinnejärjestelmistä ei tunnuta päästävän eroon. Luvussa kuusi käsitellään keinoja, joilla perinnejärjestelmien syntyä voidaan ennaltaehkäistä. Lopuksi tehdään vielä katsaus, miten modernisaation avulla voidaan tehdä perinnejärjestelmistä taas käyttökelpoisia.

2 Tutkimusmenetelmät

Tämä tutkielma on toteutettu kirjallisuuskatsauksena. Tässä tutkielmassa keskityn tarkastelemaan perinnejärjestelmiä, niihin liittyviä ongelmia, miksi niitä syntyy ja millaisia ratkaisuja näihin ongelmiin on esitetty. Tarkoituksena oli tehdä yleinen katsaus perinnejärjestelmiin, joten tämän tutkielman ulkopuolelle jätettiin esimerkiksi riskianalyysi, joka tuntui olevan usean tieteellisen julkaisun keskipisteenä. Perinnejärjestelmät eivät olleet käsitteenä etukäteen minulle kovin tuttuja, joten melkein kaikki tieto tuli minulle uutena. Tutkielmassa käyttämäni materiaalin hain ScienceDirect-, IEExplore-, ACMDL-, Springer- ja Andor-tietokannoista. Google Scholar auttoi myös muutaman lähteen jäljittämisessä.

Käyttämiäni hakusanoja olivat esimerkiksi ”legacy system”, ”reengineering”, ”decomission”, ”modernization”, ”wrapper” ja ”cost”. Näitä termejä yhdistelemällä hain siis lähteitä edellä mainituista tietokannoista. Löytämieni lähteiden lähdeluettelo osoittautui myös oikein oivaksi tavaksi löytää lisää lähteitä ja hakutermejä. Käytetyt lähteet olivat pääosin tieteellisiä artikkeleita, mutta erityisesti perinnejärjestelmien haittojen kohdalla jouduin turvautumaan myös uutisartikkeleihin.

Suurin osa tutkielmassani käyttämistäni lähteistä on julkaistu 90-luvun lopulla tai 2000-luvun alussa. Tälle ja viime vuosikymmenelle tultaessa perinnejärjestelmistä tuotetun tutkimusmateriaalin määrässä tapahtui huomattava pudotus. Perinnejärjestelmiin kohdistuvan tutkimuksen määrä näyttäisi siis pienentyneen viime vuosina huomattavasti. Käytetty termistö ja näiden järjestelmien takana vaikuttavat tekijät eivät kuitenkaan ole vaikuttaneet muuttuneen kovinkaan radikaalisti, joten vanhemmassakin lähdemateriaalissa esiintyviä ajatuksia ja ratkaisuja voidaan vieläkin soveltaa melko

hyvin nykypäiväänkin perinnejärjestelmiin. Tämä ei toisaalta ole yllättävää, koska perinnejärjestelmät, joiden ongelmista puhuttiin vuosituhannen vaihteessa, voivat olla hyvinkin samoja, joista puhutaan vielä tänäkin päivänä.

Perinnejärjestelmien kohtaamat kyberuhat vaikuttavat olevan melkoisen vähän tutkittu osa-alue. Uhista kirjoitettu materiaali oli äärimmäisen niukkaa ja tieteelliset lähteet, jotka edes mainitsivat asiasta, tuntuivat vain sivuavan tätä aihetta lyhyesti. Tästä johtuen jouduin tutkielman tällä osa-alueella turvautumaan uutisartikkeleihin, joissa analysoitiin näitä kyberuhkia. Olin oikeastaan hiukan pettynyt kyberuhkia käsittelevien lähteiden vähäisyyteen, koska tutkielman aihetta rajatessani tämä oli yksi perinnejärjestelmien osa-alue, josta olisin ollut erityisen kiinnostunut.

3 Perinnejärjestelmä

Tässä kappaleessa määritellään tarkemmin, mikä on perinnejärjestelmä ja mitkä ovat sen tunnusmerkit. Lisäksi alustetaan jo vähän mitä haittoja perinnejärjestelmien yhteydessä esiintyy. Aloitetaan siis määrittelemällä termi perinnejärjestelmä lyhyesti.

Perinnejärjestelmä (englanniksi legacy system) on vanhentunut tietojärjestelmä ja siihen liittyvä ohjelmisto ja laitteisto. Perinnejärjestelmät voidaan määritellä järjestelmiksi, jotka ovat olleet pitkään käytössä ja perustuvat vanhentuneeseen teknologiaan (Matthiesen & Bjørn, 2015). Tämän lisäksi Crotty ja Horrocks (2016) listasivat tutkimuksessaan muutamia lisämääritelmiä perinnejärjestelmälle:

1. mikä hyvänsä kriittinen ohjelmistojärjestelmä, joka vastustaa muokkausta ja jonka toiminnan pettämisellä on merkittävä vaikutus yrityksen tai organisaation operaatioihin
2. järjestelmä tai systeemi, joka pohjautuu vanhentuneeseen teknologiaan, mutta on kriittinen jokapäiväisessä toiminnassa.

Tässä määritelmässä näyttäisi korostuvan vahvasti näkemys perinnejärjestelmien kriittisyydestä yrityksen tai organisaation toiminnalle. Tämä johtuu siitä, että varsinkin erityisen vanhojen järjestelmien kohdalla on varsin tavallista, että systeemi on liitoksissa useamman uudemmankin järjestelmän kanssa organisaation tai yrityksen sisällä. Varsinkin perinnejärjestelmien tapauksessa on mahdollista, että pienikin muutos järjestelmässä voi aiheuttaa arvaamattoman ketjureaktion muissa järjestelmissä, joihin on tehty liitoksia vuosien saatossa (Gangadharan ja muut, 2013).

Perinnejärjestelmän tunnusmerkeiksi voidaan katsoa vanha ikä, vanhentunut koodikieli, huono dokumentaatio, puutteellinen datan ylläpito ja hajanainen rakenne

(Crotty & Horrocks, 2016). Vaikka nämä ovat perinnejärjestelmän tavanomaiset tunnusmerkit, ei järjestelmän tarvitse täyttää näistä kaikkia ollakseen perinnejärjestelmä. Jotkin näistä tunnusmerkeistä voivat jo yksinäänkin vaikuttaa järjestelmän toimintaan tarpeeksi negatiivisesti, että järjestelmä voidaan luokitella perinnejärjestelmäksi. Esimerkiksi vanhentunut koodikieli tekee järjestelmän muokkaamisesta paljon monimutkaisempaa ja kalliimpaa kuin se olisi, jos järjestelmä olisi toteutettu modernilla ohjelmointikielellä. Tästä syystä perinnejärjestelmiä usein vaivaavat erinäiset yhteensopivuusongelmat, koska nykyaikaiset ohjelmointikielet eivät yksinkertaisesti ole yhteensopivia vanhan koodikielen kanssa. Uuden toiminnallisuuden lisääminen perinnejärjestelmään on siis vaikeaa, mikä osaltaan kiihdyttää järjestelmän vanhentumista (Sneed & Verhoef, 2020). Perinnejärjestelmien haittoja tullaan tarkastelemaan vielä tarkemmin seuraavassa kappaleessa.

4 Perinnejärjestelmien haitat

Kuten jo edellisessä kappaleessa mainittiin, perinnejärjestelmät nähdään hyvin pitkälti yrityksen tai organisaation toimintaa haittaavana tekijänä. Miksi näin on? Tarkastellaan siis tässä kappaleessa tarkemmin, millaista haittaa perinnejärjestelmistä koituu.

Perinnejärjestelmät ovat keskimääräiseltä iältään jo hyvinkin vanhoja, yli kahdenkymmenenkin vuoden ikä on hyvin tavallinen näistä järjestelmistä puhuttaessa (Van Den Heuvel, 2009). Tämä on merkittävää varsinkin, koska Dedeken (2012) mukaan kymmenestä kahteenkymmeneen vuotta käytössä ollut ohjelmisto on viettänyt enemmän aikaa perinnejärjestelmänä kuin aktiivisessa kehityksessä olevana.

Koska perinnejärjestelmät ovat hyvinkin vanhoja, voidaan olettaa myös ohjelmointikielen, jolla ne on kirjoitettu olevan vanhaa. Perinnejärjestelmät on useimmiten kirjoitettu käyttäen COBOLia, PL/1:tä tai Assembly/370:ää ohjelmointikielenä, tosin käytetyt ohjelmointikielet eivät ole käytännössä rajoittuneet pelkästään näihin (Van Den Heuvel, 2009). Koska esimerkiksi juuri COBOL-ohjelmointikielen osaajista on nykyään paljon puutetta, ovat useat tähän ohjelmointikieleen pohjautuvat järjestelmät vaarassa jäädä puutteelliselle ylläpidolle tulevaisuudessa (Ihanus, 2006). Sneed ja Verhoef (2020) ovat tässä asiassa samaa mieltä ja toteavat tutkimuksessaan nuorten ohjelmoijien työskentelevän mieluummin uuden teknologian parissa. Näille nuoremmille ohjelmoijille ei enää opeteta COBOLin kaltaisia vanhentuneita kieliä. Vaikka yrityksen tai organisaation olisi mahdollista kouluttaa ohjelmoijiaan käyttämään vanhoja ohjelmointikieliä, tulee tässä yleensä vastaan

haluttomuutta investoida lisää resursseja perinnejärjestelmään. Koska nuorista osaajista on puutetta, nostaa tämä vanhojen ohjelmointikielten osaajien kysyntää. Heidän palkkaansa tämä kasvava kysyntä tulee suurella todennäköisyydellä vaikuttamaan positiivisesti, mutta yritysten ja organisaatioiden IT-budjettien kannalta tämä kehitys ei liene aivan yhtä toivottavaa.

Arviolta noin seitsemänkymmentäviisi prosenttia pankkien ja vakuutusyhtiöiden IT-budjetista kuluu olemassa olevien perinnejärjestelmien ylläpitoon (Crotty & Horrocks 2016). Ei ole siis ihme, että perinnejärjestelmien aiheuttamiin ongelmiin etsitään ratkaisua. Korkeiden ylläpitokustannusten kurissa pitämiselle on suuri tarve monissa organisaatioissa. Nämä korkeat ylläpitokustannukset syövät varoja, jotka voitaisiin muuten käyttää uusiin innovaatioihin. Kaiken lisäksi näiden kulujen osuus budjetissa kasvaa jatkuvasti. Ei ole tunnistettu yritystä tai organisaatiota, jonka laitteiston tai ohjelmiston ylläpitoon varattu budjetti olisi laskussa (Gangadharan ja muut, 2013). Kasvavat taloudelliset kustannukset, joita perinnejärjestelmän ylläpito tuo tullessaan, voi siis muodostaa merkittävän taloudellisen taakan yritykselle. Koska IT-budjetit ovat painotettu vahvasti pelkästään nykyisen järjestelmän ylläpitoon, ei tästä jää paljonkaan ylimääräisiä varoja uusien innovaatioiden tekemiseen. Tämä johtaa vääjäämättä innovaation laskuun, koska keskitytään vain nykyisen toiminnallisuuden säilyttämiseen.

Vaikka suuret kuluerät todennäköisesti painavat vaakakupissa yrityksen tai organisaation päätöksenteossa suhteellisen paljon, eivät kaikki perinnejärjestelmiin liittyvät ongelmat ole luonnoltaan puhtaasti taloudellisia. McKernan (2019) toteaa artikkelissaan perinnejärjestelmien olevan kompleksisuutensa vuoksi erittäin haavoittuvaisia erilaisille kyberhyökkäyksille. Perinnejärjestelmien tietoturva voi olla vaarantunut monesta eri syystä, esimerkiksi vanhentuneet järjestelmät eivät saa enää minkäänlaisia tietoturvapäivityksiä, jolloin tietoturvasta vastaa ainoastaan järjestelmän käyttäjä. Teknistä tukea on myös hankalaa saada, koska järjestelmän valmistanutta yritystä ei välttämättä ole enää olemassa. Kokonaisuutena perinnejärjestelmien tietoturva on kuitenkin melko vähän tutkittu osa-alue.

Heikentynyt tietoturva tuo mukanaan myös lainsäädännöllisiä ongelmia, jos perinnejärjestelmään tallennetun tiedon käyttö on jollain tavalla lainsäädännössä määrättyä. Jos perinnejärjestelmää käytetään esimerkiksi yrityksen asiakastietojen säilyttämiseen, tulee nämä tiedot säilöä oikealla tavalla lainsäädännön pykäliä noudattaen. Tietojen vääränlainen säilyttäminen tuottaa paljastuessaan yleensä sakkorangaistuksen. Euroopan Unionin sisällä on voimassa yleinen tietosuoja-asetus,

joka tunnettaan lyhyemmin nimellä *GDPR* (General Data Protection Regulation). Tällä asetuksella valvotaan juuri henkilötietojen tallentamista Euroopan unionin sisällä, joskin esimerkiksi yritysten ja organisaatioiden nettisivut EU:n ulkopuolellakin joutuvat noudattamaan tietosuoja-asetusta, jos ne ovat saatavilla EU:n sisäpuolella. Yleisen tietosuoja-asetuksen mukaan esimerkiksi puutteellisen tietoturvan perusteella yritystä tai organisaatiota voidaan sakottaa erittäin suurilla summilla, jopa 20 miljoonan euron verran, riippuen rikkomuksen laadusta ja vakavuudesta (GDPR). Varsinkin pienemmän kokoluokan yrityksillä tuskin on tarpeeksi varoja ottaa riskiä joutua maksamaan näin suuria summia vanhentuneen järjestelmän takia. Rikkomuksista seuraa yleensä myös paljon negatiivista julkisuutta, joka vaikuttaa yrityksen julkisuuskuvaan.

Tässä luvussa tehtiin lyhyt katsaus muutamaan yleisimpään haittaan, joita perinnejärjestelmien yhteydessä kohdataan. Heikko tietoturva ja suuret kuluerät yritykselle ovat jo yksinään vakuuttava syy pyrkiä hankkiutumaan eroon perinnejärjestelmästä. Kuitenkin monet yritykset ja organisaatiot päättävät jatkaa näiden järjestelmien käyttöä niistä luopumisen sijaan. Seuraavassa luvussa tarkastellaan, miksi perinnejärjestelmistä eroon pääseminen on hankalaa.

5 Miksi perinnejärjestelmistä on hankala päästä eroon?

Perinnejärjestelmistä on yleensä yrityksille ja organisaatioille enemmän haittaa kuin hyötyä. Miksi sitten niitä päästetään syntymään ja niiden käyttöä jatketaan tästä huolimatta? Tässä kappaleessa tarkastellaan hieman tarkemmin, mitkä tekijät vaikeuttavat perinnejärjestelmistä luopumista.

Matthiesen ja Bjørn (2015) toteavat, että usein ongelmat, joita kohdataan perinnejärjestelmien yhteydessä, eivät ole luonnoltaan vain teknisiä, vaan usein myös yritykseen ja organisaatioon liittyviä. Varsinkin bisnespuolella törmätään usein hankaluuksiin, koska myös ohjelmiston tai laitteiston käytöstä poistoon kuluu rahaa puhumattakaan tilanteesta, jossa nykyinen järjestelmä joudutaan korvaamaan.

Perinnejärjestelmästä luopuminen ja siirtyminen moderniin järjestelmään tuo aina mukanaan oman epävarmuutensa. Tämä epävarmuus saa organisaation johdon haluttomaksi tekemään radikaaleja muutoksia, varsinkin jos olemassa olevan järjestelmän toimintakyky on vielä toistaiseksi tyydyttävällä tasolla. Olemassa olevasta investoinnista halutaan pitää kiinni (Dedeke, A. 2012). Ajatellaan siis järjestelmän toimivan tarpeeksi hyvin nykytilanteessa, joten miksi siis haaskata aikaa ja resursseja järjestelmän korjaamiseen, jos se ei kerran nykyisellään tuota ongelmia yrityksen tai organisaation

toiminnalle. Lisäksi uuden järjestelmän käyttöönotto voi tuoda mukanaan vastarintaa työntekijöissä, koska mitä pidempään järjestelmä on käytössä, sitä haluttomampia työntekijät ovat päästämään siitä irti ja siirtymään käyttämään uutta järjestelmää (Sneed ja Verhoef, 2020). Uusi järjestelmä tuo myös mukanaan tilanteen, jossa työntekijät joutuvat kouluttautumaan uuden järjestelmän käyttöön, joka myös osaltaan lisää haluttomuutta.

Matthiesenin ja Bjørnin (2015) tutkimuksessa havaittiin, että vaikka perinnejärjestelmää haluttaisiin lähteä korvaamaan uudella järjestelmällä, kohtaavat tällaiset projektit usein ongelmia, joita on hankalaa ennustaa ennen projektin aloitusta. Esimerkiksi jos korvaavan järjestelmän hankinta ulkoistetaan, voidaan törmätä ongelmiin tiedonjaon kanssa, koska henkilötietoja sisältävää dataa ei yleensä saa helposti kuljettaa rajojen yli. Matthiesenin ja Bjørnin (2015) havainto on sinänsä melko yllättävä, koska Sneed ja Verhoef (2020) toteavat ulkoistamisen olevan suosittu ratkaisu perinnejärjestelmien ongelmien ratkaisemisessa.

Kuten jo aiemmin käsiteltiin, mainitsivat Crotty ja Horrocks (2016) tutkimuksessaan perinnejärjestelmien olevan yleensä kriittisiä yrityksen tai organisaation toiminnalle. Tämä on todennäköisesti perimmäinen syy, miksi perinnejärjestelmiä pääsee syntymään niin paljon. Koska kaikki yrityksen tai organisaation toiminta on kasattu perinnejärjestelmän varaan, koetaan se liian arvokkaaksi, jolloin siitä luopumista ei välttämättä edes harkitakaan.

Khadka ja muut (2015) onnistuivat tutkimuksessaan paikallistamaan myös muutaman hyödyn perinnejärjestelmistä. Perinnejärjestelmien todettiin joissain tapauksissa toimivan nopeammin ja tehokkaammin kuin modernit järjestelmät. Tämä johtui siitä, että uudempien järjestelmien koodi oli raskaampaa esimerkiksi palomuurien takia. Matthiesen ja Bjørn (2015) korostivat myös perinnejärjestelmien vakautta, koska vuosien toiminnan seurauksena järjestelmässä ei yleensä enää esiinny yllättäviä ongelmia. Varsinkin koettu vakaus voisi olla hyvä selittäjä perinnejärjestelmien yleisyydelle.

6 Miten ennaltaehkäistä perinnejärjestelmien syntymistä?

Koska perinnejärjestelmien syntyä voidaan nykyään pitää jo yleisenä ongelmana, on vuosien saatossa kehitetty useita sääntöjä ja suunnittelumenetelmiä, joiden avulla pyritään estämään perinnejärjestelmien synty alkuunsa. Tässä luvussa eritellään

muutamia keinoja, joilla perinnejärjestelmien ongelmiin varaudutaan jo järjestelmien normaalin elinkaaren aikana.

Monet nykyajan perinnejärjestelmistä on rakennettu aikana, jolloin tietojärjestelmien prosessointiteho ja muisti olivat paljon kalliimpia kuin nykyään. Tästä johtuen nämä järjestelmät rakennettiin priorisoimaan tehokkuutta, joka usein johti heikkoon ymmärrettävyyteen ja ylläpidettävyyteen (Crotty & Horrocks, 2016). Hyvät ohjelmointitavat, kuten hyvä dokumentaatio, koodin kommentointi ja versionhallinta voisivat auttaa ratkaisemaan tällaisia ongelmia, mutta niitä on hankala soveltaa enää valmiiseen järjestelmään.

Gangadharan ja muut (2013) nostivat esille ajoissa tehdyn käytöstä poiston edut. Erityisesti he korostivat ajatusta, että uutta laitteistoa tai ohjelmistoa hankittaessa, niiden tulisi korvata jotain vanhaa, vanhan järjestelmän päälle kasaamisen sijaan. Jokaisen uuden IT-hankinnan kohdalla poistettaisiin siis käytöstä jotain vanhaa. Näin mitään järjestelmän osaa ei päästettäisi ihanteellisessa tilanteessa niin vanhaksi, että siitä tulisi perinnejärjestelmä. Todettiin kuitenkin, että vaikka tämä toimintatapa parantaa uuden ja vanhan teknologian välistä tasapainoa, vain harvoilla yrityksillä tai organisaatioilla on kykenevyyttä tai halua noudattaa tätä. Käytöstä poistosta puhutaan vielä lisää myöhempänä.

Perinnejärjestelmistä tulee hankalia ylläpitää ja poistaa käytöstä sen mukaan kuinka paljon riippuvuuksia on muodostettu muihin järjestelmiin yrityksen tai organisaation sisällä (Gangadharan ja muut, 2013). Näitä riippuvuuksia voitaisiin välttää erottamalla yrityksen tai organisaation sisäisiä järjestelmiä toisistaan, jolloin ylläpidettävyyden pitäisi vastaavasti parantua.

Vaikka ennaltaehkäisevä toiminta on tärkeää tietojärjestelmien kohdalla, pääsee perinnejärjestelmiä tästä huolimatta syntymään. Seuraavassa luvussa tarkastellaan mitä toimia voidaan suorittaa jo olemassa olevan perinnejärjestelmän toiminnallisuuden parantamiseksi.

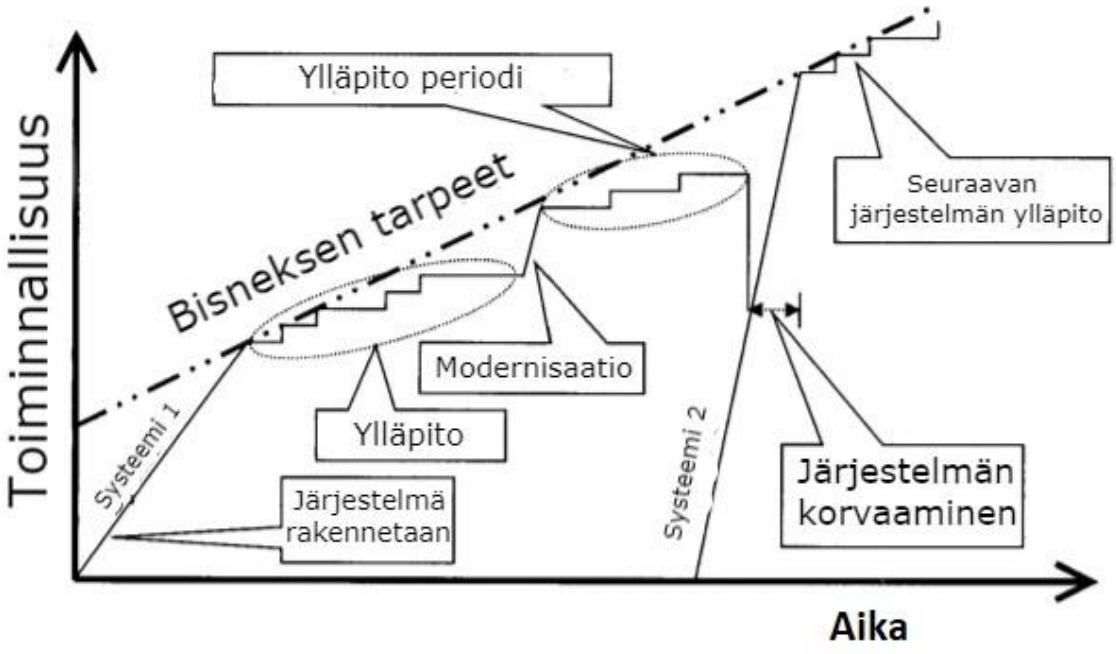
7 Mitä tehdä olemassa oleville perinnejärjestelmille?

Koska perinnejärjestelmät ovat usein kalliita ja organisaation toiminnan kannalta kriittisiä (Matthiesen & Bjørn, 2015), on niiden korjaamiseen ja modernisoimiseen kehitetty vuosien saatossa muutamiakin erilaisia lähestymistapoja. Vaikka termin ”perinnejärjestelmä” negatiivisesta konnotaatiosta johtuen voitaisiin olettaa käytöstä poistamisen olevan oikea lähestymistapa niihin liittyvien ongelmien ratkaisussa, voi

useissa tapauksissa tämän sijaan järjestelmän korjaaminen olla pidemmän päälle parempi ratkaisu. Toisaalta joissain tapauksissa oikein tehty ja ajoitettu hallittu käytöstä poisto voi pitkällä aikavälillä johtaa myös hyviin tuloksiin.

7.1 Modernisaatio

Modernisaatio (modernization) on kattotermi, joka pitää sisällään ne menetelmät, joiden avulla perinnejärjestelmiä voidaan lähteä korjaamaan, eli niin sanotusti tuomaan ne takaisin nykypäivään. Comella-Dorda ja muut (2000) määrittelivät modernisaation olevan prosessi, johon tulisi ryhtyä, kun pelkkä ylläpito ei riitä pitämään järjestelmää ajan tasalla. Modernisaatio tuo järjestelmän toiminnallisuuden vastaamaan yrityksen tai organisaation tarpeita (kuva 1). Modernisaation menetelmä valitaan tarkemmin riippuen siitä, millaisia tarpeita korjattavaan järjestelmään kohdistuu. Seuraavissa kappaleissa tarkastellaan muutamia modernisaation menetelmiä.

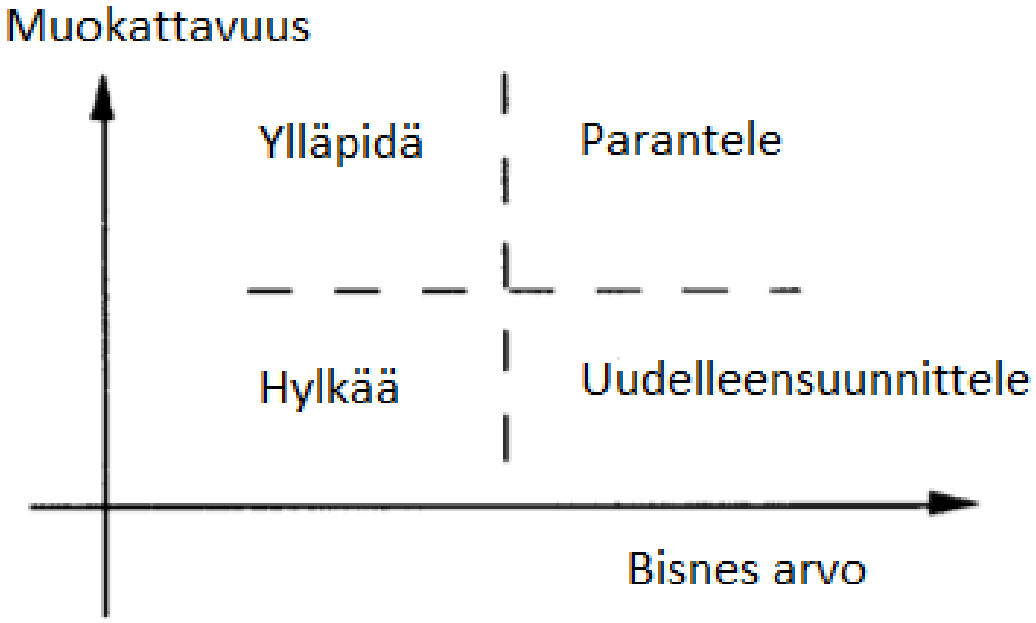


Kuva 1. *Tietojärjestelmän elinkaari (Comella-Dorda & muut, 2000).*

Perinnejärjestelmien ongelmien korjaamiseksi on esitetty useita erilaisia ratkaisuja koko järjestelmän uudelleen kirjoittamisesta käärintäratkaisuihin. Nämä ratkaisut voidaan karkeasti jakaa kahteen eri kategoriaan, jotka ovat, *valkoinen laatikko* (white-box) ja *musta laatikko* (black-box) (Van Den Heuvel, 2009).

Jacobson ja Lindstöm (1991) esittelivät tutkielmassaan päätöksentekomatriisin, joka kuvaa mitä perinnejärjestelmälle tulisi tehdä riippuen kyseisen järjestelmän

muokattavuudesta ja sen taloudellisesta arvosta (kuva 2). Korkean taloudellisen arvon omaavat järjestelmät uudelleensuunnitellaan tai parannellaan, kun taas matalan arvon järjestelmiä ylläpidetään tai niistä hankkiudutaan eroon.



Kuva 2. *Päätöksentekomatriisi, joka kuvaa mitä tehdä perinnejärjestelmälle (Jacobson & Lindstöm, 1991).*

Valkoinen laatikko -ratkaisut vaativat syvällistä ymmärrystä muokattavan järjestelmän toiminnasta, prosesseista ja komponenteista. Järjestelmää muokataan koodi- ja rakennemuutosten avulla haluttuun suuntaan, jotta jotain laadullista tekijää, kuten käytettävyyttä, voitaisiin parannella (Comella-Dorda ja muut, 2000). Valkoinen laatikko -kategoriaan kuuluvat menetelmät vaativat siis asiantuntevuutta toteutukselta ja ovat keskimäärin musta laatikko -menetelmiä työläämpiä toteuttaa.

Musta laatikko -kategoriaan kuuluvat ratkaisut pyrkivät myös parantamaan järjestelmän käytettävyyttä ja yhteensopivuutta, mutta toisin kuin valkoisen laatikon menetelmät, tämän kategorian menetelmät eivät yleensä kajoa perinnejärjestelmän koodiin tai sen rakenteeseen. Musta laatikko -menetelmissä tarkastellaan perinnejärjestelmän sisään- ja ulostulevia syötteitä, jotta voidaan muodostaa ymmärrys järjestelmän toiminnasta. Musta laatikko -kategorian menetelmät pyrkivät peittämään perinnejärjestelmän kompleksisuutta uudella koodilla, joka on rakennettu järjestelmän päälle (Comella-Dorda ja muut, 2000).

Tehdään seuraavissa luvuissa katsaus muutamaan menetelmään, joilla perinnejärjestelmiä voidaan lähteä korjaamaan niiden käyttöiän ja yhteensopivuuden

parantamiseksi. Lopuksi tehdään vielä nopea katsaus, mitä hyötyjä voidaan saavuttaa käytöstä poiston avulla.

7.2 Uudelleensuunnittelu

Uudelleensuunnittelu (reengineering) on perinnejärjestelmien korjaamisessa tavallista ylläpitoa askeleen verran pidemmälle menevä ratkaisu. Uudelleensuunnittelussa eheytetään järjestelmä vuosia jatkuneen ylläpidon seurauksena tapahtuneiden koodimuutosten jäljiltä. Uudelleensuunnittelu on kattotermi, joka pitää sisällään menetelmiä, jotka voitaisiin katsoa erillisiksi. Tässä tutkielmassa tarkastellaan kuitenkin uudelleensuunnittelua kokonaisuutena sen pienemmiksi osa-alueiksi pilkkomisen sijaan.

Järjestelmän *ylläpidossa* (maintenance) on kyse järjestelmän kunnostamisesta ilman perustavanlaatuisia muutoksia koodiin ja sen sisäistä arkkitehtuuria rikkomatta (Van Den Heuvel, 2009). Tämän täydellisenä vastakohtana on uudelleensuunnittelu, jossa tarkoituksena on juuri tehdä perustavanlaatuisia muutoksia koodiin sen lisäksi että järjestelmän arkkitehtuuri suunnitellaan kokonaan uudestaan. Uudelleensuunnittelu on prosessi, joka parantaa ohjelmiston ymmärrettävyyttä ja/tai valmistelee tai parantaa ohjelmistoa itsessään, jotta saavutettaisiin parantunut ylläpidettävyys, uudelleenkäytettävyys tai evoluutio (Ko & muut, 2007).

Uudelleensuunnittelu aloitetaan *takaisinmallinnuksella* (reverse engineering). Tämä tehdään yleensä lähdekoodia ja dokumentaatiota apuna käyttäen, mutta ei ole kuitenkaan harvinaista, että jompikumpi näistä tai molemmat puuttuvat kokonaan. Tässäkin tilanteessa perinnejärjestelmän koodi voidaan takaisinmallintaa, joskin se on tällöin työläämpää. Tähän on olemassa automaattisia työkaluja, mutta niiden tuottamat tulokset ovat virhealttiimpia kuin ihmisen tekemänä. Näin ollen tässä työvaiheessa ei voida nojata vain automaattisiin työkaluihin, vaan tarvitaan lisäksi ohjelmoijia täyttämään joitakin aukkoja (Chiang 2007). Vaaditun työmäärän takia uudelleensuunnittelu on yleensä kallis prosessi

7.3 Käärintä

Käärintä (wrapper) on perinnejärjestelmien korjausmenetelmä, jossa olemassa oleva järjestelmä hajotetaan komponenteiksi ja niiden ympärille luodaan uusi rajapinta, joka muuntaa kutsut vanhaan järjestelmään sopiviksi (Sneed & Verhoef, 2020). Vastaavasti rajapinta muuttaa perinnejärjestelmästä ulostulevan syötteen uudelle ohjelmistolle yhteensopivaksi. Näin pyritään piilottamaan vanhan järjestelmän kompleksisuus.

Käärintä on hyvin yleinen musta laatikko -kategorian menetelmä perinnejärjestelmien modernisoinnissa (Comella-Dorda ja muut, 2000). Käärinnässä perinnejärjestelmä jää kirjaimellisesti mustaksi laatikoksi, koska kaikki käyttäjän tekemä toiminta kohdistuu rajapintaan vanhan järjestelmän sijaan. Rajapinnan tekemä komentojen kääntäminen jää käyttäjältä kokonaan näkemättä päällepäin.

Perinnejärjestelmän komponenttien käärintä jättää vanhan koodin koskemattomaksi. Tämä on kaksiteräinen miekka, koska vaikka käärintä on käytännössä hyvin nopea toteuttaa, jää uusi järjestelmä komponenttien sisällä olevan vanhentuneen ohjelmointikielen takia yhä riippuvaiseksi tämän ohjelmointikielen osaajista, jos järjestelmän komponentteihin tarvitsee tulevaisuudessa minkäänlaisia muutoksia. Käärintä on kuitenkin suhteellisen nopea ja muihin vaihtoehtoihin nähden edullinen toteuttaa. Suurin osa käärinnän toteutuksen yhteydessä syntyvistä kuluista syntyy muodostetun uuden järjestelmän testaamisesta (Sneed & Verhoef, 2020).

Kuten edellisessä luvussa jo mainittiin, musta laatikko -kategoriaan kuuluvat menetelmät eivät yleensä koske perinnejärjestelmän lähdekoodiin ollenkaan, jos se vain on mahdollista. Näin ollen käärinnän avulla tuotettu uusi järjestelmä on huomattavasti heikommin muokattavissa kuin esimerkiksi uudelleensuunnittelun avulla modernisoitu järjestelmä. Käärintäratkaisussa on kuitenkin omia etuja, jotka voivat osoittautua yhtiölle tai organisaatiolle tehokkaammiksi tarkasteltavan järjestelmän iästä ja käytettävyyden tilanteesta riippuen. Käärinnän suhteellinen riskittömyys tekee siitä hyvän vaihtoehdon, koska perinnejärjestelmään tallennettu data ja toiminnallisuus säilyvät tismalleen samana kuin ennen käärintää (Comella-Dorda ja muut, 2000). Jos järjestelmää ei tarvitse muokata rajusti tulevaisuudessa kannattaa perinnejärjestelmän käärintää harkita dramaattisempien ratkaisujen sijaan.

7.4 Käytöstä poisto tai Korvaaminen

Joissakin tapauksissa perinnejärjestelmän käytöstä poistaminen on varteenotettava vaihtoehto korjaamisen sijaan. Perinnejärjestelmän käytöstä poistaminen on radikaali ratkaisu, jolla voidaan saavuttaa merkittäviä etuja, mutta se on samalla vaihtoehdoista kaikkein riskialttein (Van Den Heuvel, 2009). Tarkastellaan tässä samalla myös perinnejärjestelmän korvaamista, koska myös tässä tapauksessa vanha järjestelmä poistetaan käytöstä. Erona on vain se, että tilalle hankitaan uusi järjestelmä, joka toteuttaa saman toiminnallisuuden, sen sijaan että toiminnallisuus vain katoaisi.

Käytöstä poisto on jokaisen järjestelmän ja ohjelmiston elinkaaren luontainen päätepiste. Ei ole siis ihmekään, että sama kohtalo odottaa lopulta myös perinnejärjestelmiä. Perinnejärjestelmien käytöstä poistaminen on kuitenkin asteen verran monimutkaisempaa, kun otetaan huomioon kaikki liitokset, joita järjestelmään on tehty sen käyttöiän aikana. Tämä järjestelmän elinkaaren aikana muodostunut kompleksisuus tulee ottaa huomioon, kun käytöstä poistoa toteutetaan. Pelkkä sähköjohdon vetäminen seinästä tulee todennäköisesti aiheuttamaan käyttökatkon jossain toisessa organisaation järjestelmässä. Onkin siis tärkeää etukäteen ottaa selvää, miten perinnejärjestelmän käytöstä poisto vaikuttaa muiden organisaation sisäisten järjestelmien toimintaan (Gangadharan ja muut, 2013).

Tietyissä tapauksissa perinnejärjestelmän käytöstä poistaminen ei ole järkevää, koska järjestelmän tekemät tehtävät ovat liian kriittisiä yrityksen tai organisaation toiminnan kannalta. Näissä tapauksissa perinnejärjestelmä pyritään korvaamaan uudemmalla tietojärjestelmällä, joka toteuttaa tarpeista riippuen kaikki vanhan järjestelmän toiminnot tai vain osan niistä.

8 Keskustelu

Nyt olemme käyneet läpi mitä perinnejärjestelmät ovat, mitä haittaa niistä on, miksi niistä ei tunnuta pääsevän eroon, miten niitä ehkäistään ja miten niitä voidaan modernisoida. Perinnejärjestelmät tuottavat yrityksille ja organisaatioille päänvaivaa, mutta niistä ei kuitenkaan tunnuta luovuttavan niin usein kuin se olisi suositeltavaa.

Perinnejärjestelmien aiheuttamat ongelmat ovat paljon tutkittu aihe. Suurimmiksi näistä näyttivät korostuvan hankala ylläpidettävyys ja tästä seuraava IT-budjetin paisuminen. Perinnejärjestelmät ovat hankalia ylläpitää, koska hajanainen rakenne ja vanhat ohjelmointikielet takaavat, että todennäköisesti ainoastaan järjestelmän rakentamiseen osallistunut henkilö ymmärtää järjestelmän toimintaa tarpeeksi hyvin. Toisaalta tämäkään ei aina riitä, koska vuosien käytön aikana järjestelmään lisätyt ominaisuudet nostavat sen kompleksisuuden vielä korkeammalle tasolle. Loppujen lopuksi yrityksessä tai organisaatiossa ei ole enää välttämättä ketään, joka tuntisi kaikki perinnejärjestelmän riippuvuudet.

Perinnejärjestelmästä eroon pääseminen on usein hankalaa. Perinnejärjestelmät ovat usein kriittisiä yrityksen tai organisaation toiminnalle, joten niiden käyttöä jatketaan reippaasti yli niiden suositellun eliniän. Perinnejärjestelmät toimivat yleensä tarpeeksi vakaasti, joten yrityksen tai organisaation johto ei yleensä riskeeraa varoja niiden

parantelemiseen. Uuteen järjestelmään siirtymisestä syntyvät mahdolliset säästöt ovat myös epäselviä ennen modernisointi- tai korvaamisprojektin aloittamista. Perinnejärjestelmistä luopuminen vaatisi siis reipasta asenteenmuutosta toteutuakseen.

Pitäisikö perinnejärjestelmistä siis päästä kokonaan eroon? Perinnejärjestelmien hyödyistä ei ole tehty paljonkaan tutkimuksia. Toisaalta tutkimukset yrittävät yleensä ratkaista ongelmia, minkä takia hyödyt eivät yleensä ole pääosassa. Perinnejärjestelmien hyödyiksi on tutkimusten mukaan löydetty tehokkuus ja vakaus. Tehokkuudella tarkoitetaan tarkemmin toimintojen suoritusaikoja. Perinnejärjestelmä pystyy joissain tapauksissa suorittamaan toimintoja nopeammin kuin vastaava moderni järjestelmä. On totta, että vanhat ohjelmointikielet ovat yleensä nopeampia kuin uudet. Khadka ja muut (2015) mainitsivat kuitenkin palomuurin erityisesti tekijänä, joka lisäsi suoritusaikaa. Tämä viittaa siihen, että perinnejärjestelmässä tehokkuus saavutetaan uhraamalla nykyaikaiselle järjestelmälle elintärkeitä ominaisuuksia, kuten turvallisuudesta vastaavaa ohjelmistoa. Vakauskin tuntuu enemmän uskomukselta, että vuosikymmeniä toiminut järjestelmä ei kykene enää tekemään mitään yllättävää.

Perinnejärjestelmistä tehdyssä tutkimuksessa puhutaan usein korkeista ylläpitokustannuksista. Seitsemänkymmentäviisi prosenttia IT-budjetista kuulostaa kieltämättä erittäin korkealta pelkästään perinnejärjestelmän ylläpitoon käytettäväksi. Haittaa, joka näistä ylläpitokustannuksista seuraa innovaatiolle, voidaan kuitenkin pitää mielestäni vielä merkittävämpänä ongelmana, koska yrityksen tai organisaation teknologinen kehitys heikkenee merkittävästi, kun lähes kaikki resurssit käytetään nykyisen järjestelmän toiminnan ylläpitämiseen. Kuten jo aiemmin mainittiin teknologisesti vanhanaikaiset yritykset ja organisaatiot eivät houkuttele nuorta työvoimaa yhtä hyvin kuin sellaiset, joiden teknologia on ajan tasalla. Tästä johtuen perinnejärjestelmille ei ole tarpeeksi osaavia ylläpitäjiä, minkä takia niiden ylläpito hankaloituu.

Perinnejärjestelmien ongelmat voidaan siis kiteyttää ylläpidon ongelmiksi. Kun järjestelmän ylläpito ei kykene enää saamaan perinnejärjestelmän toiminnallisuutta vastaamaan bisneksen tarpeita, tulisi järjestelmälle suorittaa modernisaatio. Modernisaatiossa perinnejärjestelmän nykytilanne analysoidaan, jonka jälkeen valitaan menetelmä, jolla järjestelmän puutteita lähdetään korjaamaan. Nämä menetelmät jaetaan kahteen luokkaan: valkoiseen- ja mustaan laatikkoon. Valkoisen laatikon menetelmät ovat suuria projekteja, joissa koko perinnejärjestelmä rakennetaan käytännössä kokonaan uudestaan vastaamaan nykyisiä tarpeita. Valkoisen laatikon menetelmät tuottavat uuden

järjestelmän, joka ymmärrettävä ja ylläpidettävä. Tämä prosessi on kuitenkin aikaa vievä ja kallis. Mustan laatikon menetelmät ovat nopeita projekteja, joissa perinnejärjestelmä jaetaan komponentteihin, jotka kääritään uuden koodin muodostamaan rajapintaan. Tämä on nopea ja verrattain edullinen ratkaisu, mutta se vain peittää vanhan järjestelmän, joten muutokset näihin käärittyihin komponentteihin ovat yhä yhtä hankalia toteuttaa kuin ennen modernisaatioita. On siis tärkeää tehdä oikea päätös perinnejärjestelmän kanssa toimimiseen, jotta uudesta järjestelmästä ei tule liian kallis tai liian heikosti ylläpidettävä, jolloin joudutaan palaamaan takaisin lähtöruutuun.

9 Yhteenveto

Tässä tutkielmassa tarkasteltiin perinnejärjestelmiä, niiden haittoja, syitä miksi niistä ei päästä eroon ja erilaisia tapoja, joilla niitä voidaan modernisoida. Mitä siis opittiin? Perinnejärjestelmät ovat vakava haitta yritysten ja organisaatioiden IT-innovaatioille. IT-budjeteista suurin osa kuluu niiden ylläpitoon. Kuitenkin niistä luopuminen tai niiden modernisoiminen ei ole itsestäänselvyys, koska näin saavutetut edut tunnetaan yleensä hyvin huonosti. Modernisoinnin menetelmillä perinnejärjestelmien käytettävyyttä ja ylläpidettävyyttä saadaan nostettua merkittävästi.

Perinnejärjestelmät vievät rahoitusta uudelta innovaatiolta ja heikko ylläpidettävyys tulee korostumaan todennäköisesti tulevaisuudessa entistäkin enemmän, kun COBOLin ja muiden vanhojen ohjelmointikielten osaajat käyvät yhä harvinaisemmiksi. Perinnejärjestelmien korvaamiseen tai modernisoimiseen tulisi siis ryhtyä jo nyt, kun se on vielä mahdollista.

Yritysten ja organisaatioiden tulisikin tehdä aktiivista analyysiä käytössä olevista järjestelmistä, jotta innovaation lasku ei pääsisi etenemään. Perinnejärjestelmästä luopuminen tasapainottaisi IT-budjettia ja tekisi yrityksestä tai organisaatiota paljon houkuttelevamman uudelle työvoimalle. Perinnejärjestelmästä kiinni pitämisellä ei vain yksinkertaisesti saavuteta tarpeeksi merkittäviä hyötyjä, että se pidemmän päälle olisi kannattavaa.

Jatkossa olisi kiinnostavaa mennä hiukan syvemmälle perinnejärjestelmien tyyppeihin. Lähdemateriaaleissa perinnejärjestelmät tyypillisesti kuvattiin hyvin yleisellä tasolla, joten en halunnut tässä tutkielmassa keskittyä kovinkaan paljoa siihen mikä rooli perinnejärjestelmällä on yrityksen tai organisaation sisällä. Tähän olisi varmaan ollut helpompi vastata, jos olisi ollut aikaisempaa kosketuspintaa perinnejärjestelmiin. Kaduin myös hiukan, että aioin tutkia myös perinnejärjestelmien hyötyjä, koska tämä ei ollut

minkään tieteellisen julkaisun tutkimuskysymyksenä. Hyötyjä piti etsiä hiukan rivien välistä.

Lähdeluettelo

Chiang, C. Software Stability in Software Reengineering. (2007). *IEEE International Conference on Information Reuse and Integration, 2007, pp. 719-723*. doi: 10.1109/IRI.2007.4296705

Comella-Dorda, S., Wallnau, K., Seacord, R.C., Robert, J. (2000). A survey of black-box modernization approaches for information systems. *Proceedings 2000 International Conference on Software Maintenance, 2000, pp. 173-183*. doi: 10.1109/ICSM.2000.883039.

Crotty, J. & Horrocks, I. (2016). Managing legacy system costs: A case of a meta-assesment model to identify solutions in a large financial services company. *Applied Computing and Informatics, Volume 13, Issue 2, 2017.* (175-183) https://doi.org/10.1016/j.aci.2016.12.001.

Dedeke, A. (2012). Improving Legacy-System Sustainability: A Systematic Approach. *IT Professional, vol. 14, no. 1, pp. 38-43, Jan.-Feb. 2012*. doi: 10.1109/MITP.2012.10.

Gangadharan G.R., Kuiper E.J., Janssen M., Luttighuis P.O. (2013) IT Innovation Squeeze: Propositions and a Methodology for Deciding to Continue or Decommission Legacy Systems. *IFIP Advances in Information and Communication Technology, vol 402.* https://doi.org/10.1007/978-3-642-38862-0_30

GDPR viitattu: 13.4.2021. *https://gdpr.eu/fines/*

Ihanus, M. (2006). Suomea uhkaa pula järeiden järjestelmien taitajista. *Ilta-Sanomat 11.5.2006.* Viitattu 23.3.2021. *https://www.is.fi/tyoelama/art-2000001463107.html*

Jacobson, I. & Lindström, F. (1991). Reengineering of Old Systems to an Object-Oriented Architecture. *ACM SIGPLAN Notices, vol 26, issue 11. https://doi-org.libproxy.tuni.fi/10.1145/118014.117980*

Khadka, R., Shrestha, P., Klein, B., Saeidi, A., Hage, J., Jansen, S., Dis, E., Bruntink, M. (2015). Does software modernization deliver what it aimed for? A post modernization analysis of five software modernization case studies. *2015 IEEE International Conference on Software Maintenance and Evolution (ICSME), 2015, pp. 477-486*. doi: 10.1109/ICSM.2015.7332499.

Ko, H., Chang, G., Kim, K. (2007). A Reengineering Approach of the Legacy System in the Digital Media Domain. *International Conference on Software Engineering Advances (ICSEA 2007), 2007, pp. 76-76.* doi: 10.1109/ICSEA.2007.10.

Matthiesen, S. & Bjørn, P. (2015). Why Replacing Legacy Systems Is So Hard in Global Software Development: An Information Infrastructure Perspective. *Proceedings of the 18th ACM Conference on Computer Supported Cooperative Work & Social Computing (CSCW '15).* https://doi-org.libproxy.tuni.fi/10.1145/2675133.2675232

McKernan, D. (2019). Five major risks of legacy IT systems. *Tenfour 16.11.2017*. viitattu 12.4.2021. *https://tenfour.com/blogsource/five-major-risks-of-legacy-it-systems*

Sneed, H. & Verhoef, C. (2020). Cost-driven software migration: An experience report. *Journal of Software: Evolution and Process. https://www.researchgate.net/publication/339012487_Cost-driven_software_migration_An_experience_report*

van den Heuvel, W. (2009). Aligning Modern Business Processes and Legacy Systems: A Component-Based Perspective*. MIT Press*