Perinnejärjestelmien määritys ja niiden ongelmat organisaatioissa

Joonas Konttila

Kandidaatintutkielma

Tietojenkäsittelytieteiden tutkinto-ohjelma

Oulun yliopisto

2022

Abstrakti

Tietojärjestelmät ovat nykyaikaisten yritysten liiketoiminnan edellytys. Usein nämä liiketoiminnalle kriittiset järjestelmät on kehitetty kymmeniä vuosia sitten. Järjestelmille on tyypillistä niiden aiheuttamat ongelmat niin järjestelmän kehityksessä, ylläpidossa kuin itse organisaatioissa. Jotta järjestelmä vastaisi organisaation kasvavia liiketoimintavaatimuksia, on järjestelmää päivitettävä jatkuvasti. Nämä toistuvat muutokset kuitenkin lisäävät kumulatiivisesti järjestelmän monimutkaisuutta. Tällaisia järjestelmiä kutsutaan perinnejärjestelmiksi. Tämän tutkielman tarkoituksena on selvittää, miten perinnejärjestelmä määritellään ja minkälaisia ongelmia se tuo eri organisaatioissa. Kirjallisuuskatsauksen pohjalta löydettiin neljä yhtenäistä perinnejärjestelmän ominaisuutta: Järjestelmä on joustamaton, kooltaan suuri, iältään vanha, ja sen täytyy olla liiketoiminnallisesti kriittinen järjestelmä.

Perinnejärjestelmiä on vaikea, ja joissain tapauksissa jopa mahdotonta jatkokehittää. Huono jatkokehitettävyys johtuu koodikannan laajuudesta ja huonosta dokumentaatiosta. Vuosien aikana järjestelmässä on työskennellyt usea eri ylläpitäjä käyttäen erilaisia ohjelmointimalleja ja käytänteitä. Kun tarvittavat yhdenmukaisuutta parantavat refaktoroinnit on jätetty tekemättä, on järjestelmässä ohjelmointikäytänteitä useilta eri vuosikymmeniltä. Dokumentaation puuttuessa tiettyjä toiminnallisuuksia on saatettu toistaa useisiin eri kohtiin lähdekoodissa, mikä entisestään huonontaa jatkokehitettävyyttä. Tutkielma on toteutettu systemaattisena kirjallisuuskatsauksena.

Asiasanat
Perinnejärjestelmät, Perinnejärjestelmien ongelmat, Perinnejärjestelmän määritys

Ohjaaja
Tohtori, Pertti Seppänen

Taulukot

Sisällysluettelo

1. JOHDANTO

Kuvittele tilanne, jossa työskentelet maanlaajuisessa huonekaluja myyvässä yrityksessä IT-ammattilaisena. Esimiehesi, joka on ollut yrityksessä koko työuransa, jää eläkkeelle ja sinut ylennetään esimiesrooliin. Ylennys ei tarkoita ainoastaan palkankorotusta, vaan koska olet töissä IT-hallinnossa, olet nyt vastuussa yrityksen kymmeniä vuosia vanhasta IT-järjestelmästä. Järjestelmä on hyvin perinteinen: Se on kehitetty IT-kuplan aikaan 1990-luvun lopussa, siihen ei juurikaan ole koskettu, mutta se siitä huolimatta palvelee yritystä kohtuullisen hienosti. Ei siis mitään hätää! Eräänä kauniina päivänä yrityksen käyttöön tulee kuitenkin uusi viivakoodeja lukeva varastointilaite, ja sen integrointi järjestelmään tuntuu täysin mahdottomalta. Projektin budjetti paisuu, aikataulut venyvät eikä kenelläkään tuntuisi riittävän hermoja järjestelmän kanssa työskentelyyn.

Tilanne on hyvin tuttu jokaiselle perinnejärjestelmien kanssa työskentelevälle. Ongelmia onkin hyvin vaikea välttää, sillä liiketoiminnan kehittyessä myös järjestelmien kehittyminen on välttämätöntä. Liiketoiminnan kehittyessä myös pohjalla olevan tietojärjestelmän tulee mukautua kasvaviin vaatimuksiin. Tämän takia tietojärjestelmää tulee jatkuvasti ylläpitää uusien ominaisuuksien ja tarvittavien korjausten vuoksi (Lehman, 1980).

Bernhard Rieder toteaa Stevensonin ja Helmondin (2020) artikkelissa, kuinka olemme jatkuvasti tekemisissä aikaisemman työn perinnön kanssa, sillä uudet toiminnallisuudet rakennetaan vanhojen toiminnallisuuksien päälle. Kun seuraava kehitystyö joutuu mukautumaan edellisten kehitystöiden päälle, syntyy tietojärjestelmään helposti dokumentoimattomia aukkoja sekä arkkitehtuurista tulee joustamaton ja epäselvä. Järjestelmä heikkenee, ellei sitä ylläpidetä tiukasti ja mukauteta toimintaympäristön muutoksiin (Lehman, 1980).

Vaikka tutkijat ja perinnejärjestelmien kanssa työskentelevät ammattilaiset jakavat yhteiset mielipiteet perinnejärjestelmien ongelmista, on perinnejärjestelmiä vielä silti valtavasti eri organisaatioissa. Vuonna 2022 teetetyn kyselyn mukaan, Micro Focus arvioi perinnejärjestelmissä yleisesti käytössä olevan ohjelmointikielen, COBOLin, rivilukumääräksi 800 miljardia (Micro Focus, 2022). Useimmat yritykset haluavat kehittää sovelluksiaan vastaamaan uusia liiketoimintavaatimuksia, mutta perinnejärjestelmien ongelmat vaikeuttavat jatkokehitystä ja modernisointia (Erlikh, 2000).

1.1. Tausta ja motivaatio

Tämän tutkielman tarkoituksena on esitellä eri ongelmia, joita on todettu ilmenevän perinnejärjestelmissä. Ongelmalla tarkoitetaan perinnejärjestelmässä ilmenevää ominaisuutta, mikä vaikeuttaa sen jatkokehitettävyyttä lyhentäen perinnejärjestelmän elinkaarta. Lisäksi tutkielman tarkoituksena on löytää yhtenäinen määritelmä perinnejärjestelmälle.

Aihe on mielenkiintoinen, sillä teen itse töitä perinnejärjestelmien parissa ja olen usein törmännyt tilanteisiin, joissa usealla ihmisellä on eri käsitys perinnejärjestelmän määrityksistä ja ominaisuuksista. Lisäksi täsmällistä perinnejärjestelmien ongelmia käsittelevää kirjallisuuskatsausta ei ole löytynyt, vaan jokainen tutkija on erikseen tehnyt havaintoja ongelmista ja raportoinut niistä lyhyesti omiin alaa käsitteleviin tutkimuksiinsa. Tästä syystä tähän tutkimuskatsaukseen valitut lähteet edustavatkin laajasti koko perinnejärjestelmän tutkimuksen aihepiiriä.

Viime aikoina perinnejärjestelmien tutkimus on keskittynyt erilaisten modernisointihankkeiden tutkimukseen. Erilaisia strategioita ja teorioita on esitetty paljon, ja syystä. Perinnejärjestelmien ongelmat ovat organisaatioille valtavan kallis päänvaiva. Ainoa tapa päästä eroon ongelmista on joko kehittää kokonaan uusi järjestelmä, tai modernisoida edellinen eri metodeja käyttäen. Näitä strategioita on tutkittu paljon, mutta useat modernisointia käsittelevät tutkimukset ovat ristiriidassa keskenään. Ristiriitojen aiheita ovat juurikin perinnejärjestelmän määritys ja ongelmien tärkeysjärjestykseen laittaminen. On sanomattakin selvää, että modernisoinnissa kannattaa keskittyä eniten ongelmia synnyttävien ominaisuuksien korjaamiseen pienten semanttisten yksityiskohtien sijaan. Tämä kirjallisuuskatsaus helpottaa ongelmien ja niiden vaikutusten käsittämistä perinnejärjestelmien kontekstissa.

1.2. Tutkimuskysymykset ja -menetelmä

Tämä tutkielma toteutetaan systemaattisena kirjallisuuskatsauksena, jossa kootaan yhteen eri tutkimuksissa ilmenneitä perinnejärjestelmien ongelmia. Munn ym. (2018) mukaan kirjallisuuskatsaus soveltuu erityisesti sellaisten tutkimusten suorittamiseen, joiden yhteydessä ei olla vielä esitetty tarkentavia kysymyksiä. Kirjallisuuskatsauksen avulla voidaan tunnistaa aiheeseen liittyviä keskeisiä osa-alueita, tekijöitä ja käsitteistöä. Tutkimusmenetelmäksi valikoitui kvalitatiivinen menetelmä. Kvalitatiivisella tutkimusmenetelmällä eli laadullisella tutkimuksella tarkoitetaan sellaista tutkimusta,

jossa pyritään kokonaisvaltaisesti ymmärtämään aihetta sen ominaisuuksien sekä laadun avulla. (Munn ym., 2018). Tutkimuskysymyksenä ja -apukysymyksenä kirjallisuuskatsaukselle toimivat seuraavat kysymykset:

- Miten perinnejärjestelmä määritellään?
- Millaisia ongelmia perinnejärjestelmien olemassaolo tuo organisaatioille?
    - Miten tunnistettuja ongelmia voidaan välttää?

Tutkielma keskittyy tutkimaan perinnejärjestelmien ongelmia, joten ulkopuolelle jäävät muun muassa perinnejärjestelmien modernisointi, modernisoinnin strategiat ja perinnejärjestelmien migraatiot. Lisäksi erilaisten ongelmien lievitys- ja parannusstrategioihin ei perehdytä sen tarkemmin. On tärkeää ensiksi käsitellä perinnejärjestelmien määritys, sillä sen avulla saadaan kartoitettua perinnejärjestelmien kokonaiskuvaa paremmin ennen niiden ongelmien käsittelyä.

Lähteitä haettiin seuraavista tietokannoista: IEEE Xplore ja Google Scholar. Rajauksia tuloksille tehtiin seuraavia hakusanoja käyttämällä ja yhdistelemällä: ’legacy’, ’system’, ’issues’, ’problems’ , ’definition’, ’software’, ’modernization’. Lähteet löydettiin käyttäen näitä hakusanoja. Lisäksi hyödynnettiin hyväksi havaittujen lähteiden lähdeluetteloita. Perinnejärjestelmän ominaisuudet ja toimintatavat eroavat paljon perinteisistä järjestelmistä, joten lähteiden haku keskittyi ’legacy’ -hakusanan käyttöön. Perinnejärjestelmien ongelmia ja määritystä on eniten käsitelty 1990- ja 2000-luvun taitteessa. Siksi suurin osa lähdemateriaalista on tältä aikakaudelta. Tutkimuskatsaukseen valitut lähteet on valittu laajasti eri perinnejärjestelmiä käsittelevistä tutkimuksista. Perinnejärjestelmän määritelmän löytämiseen käytettiin yhdeksän lähdettä. Ongelmien, niiden vaikutusten ja niitä välttävien toimenpiteiden käsittelyyn käytettiin 26 lähdettä. Useita lähteitä on hyödynnetty molempien tutkimuskysymysten käsittelyssä.

1.3. Tutkimuksen rakenne

Toisessa luvussa esitellään perinnejärjestelmät ja niille yleisesti hyväksytty määritelmä. Lisäksi luvussa käsitellään lyhyesti perinnejärjestelmien tutkimuksen historiaa. Kolmannessa luvussa käsitellään perinnejärjestelmien tunnistettuja ongelmia ja syitä sille miksi niitä syntyy. Lisäksi esitellään keinoja, joilla ongelmia voidaan vähentää jatkossa. Viimeisessä luvussa käsitellään tutkielman tuloksia yleisesti ja tehdään niistä yhteenveto. Lisäksi esitetään pohdintaa sekä aiheen ympäriltä, että mahdollisista

jatkotutkimusvaihtoehdoista. Viimeisenä kappaleena löytyy tutkimuksessa käytettyjen lähteiden luettelo.

2. PERINNEJÄRJESTELMÄN MÄÄRITYS

Perinnejärjestelmän määritystä on tutkittu useissa eri tutkimuksissa. M'bayan, Lavalin, ja Moallan (2017) mukaan vakiomääritelmää perinnejärjestelmälle ei ole kuitenkaan olemassa, vaikkakin lukuisat tutkijat ovat antaneet oman näkemyksensä ja määritelmänsä perinnejärjestelmästä. Samaan tulokseen päätyivät myös Brooke ja Ramage (2001). Useat tutkimukset jakavat kuitenkin yleiskäsityksen Bennettin (1995), jo 1990-luvun alussa tekemän määritelmän kanssa. Bennett (1995) luonnehtii perinnejärjestelmiä suuriksi järjestelmiksi, joiden kanssa ei pärjätä, mutta ne ovat elintärkeitä niitä omaaville organisaatioille. Samana vuonna Brodie ja Stonebraker (1995) määrittelevät perinnejärjestelmän tietojärjestelmäksi, joka vastustaa merkittävästi muokkaamista ja kehittymistä uusien sekä jatkuvasti muuttuvien liiketoimintavaatimusten täyttämiseksi. Kolme vuotta myöhemmin Gold (1998) kuvailee perinnejärjestelmää edelleen käytössä olevaksi tietojärjestelmäksi tai sovellusohjelmaksi, joka on kilpailukyvyltään ja yhteensopivuudeltaan heikko. Järjestelmälle on myös tyypillistä, että sen korvaaminen tai uudelleensuunnittelu on liian kallista. Tämä tarkoittaa, että järjestelmä on suuri, monoliittinen ja vaikeasti muokattava.

Khadka, Batlajery, Saeid, Jansen, ja Hagen (2014) teetättivät tutkimuksessaan kyselyn perinnejärjestelmien asiantuntijoille selvittääkseen, miten he määrittävät perinnejärjestelmät. 76,7 % asiantuntijoista olivat sitä mieltä, että perinnejärjestelmät ovat organisaation ydinjärjestelmiä, jotka ovat kymmeniä vuosia vanhoja. Noin puolet kyselyyn vastanneista olivat sitä mieltä, että käytetty ohjelmointikieli ei tee järjestelmästä perinnejärjestelmää, vaan sen määrittävät muut tekijät. Kyselyssä nousi esiin myös uusia perinnejärjestelmän tunnusmerkkejä, joita muissa lähteissä ei todeta. Näitä ovat perinnejärjestelmien toimivuus, varmuus ja tehokkuus. Järjestelmät ovat pyörineet tuotannossa vuosikymmeniä, ja ovat täten osoittaneet luotettavuutensa organisaatiossa.

Bennettin määritelmää jatkavat myös De Lucia, Fasolino, ja Pompelle (2001). Heidän mukaansa järjestelmästä tulee perinnejärjestelmä, kun organisaatio on liiketoiminnallaan riippuvainen järjestelmän olemassaolosta, järjestelmä ei täytä organisaation liiketoiminnallisia vaatimuksia, ja tarvittavat muutokset järjestelmään vaatimusten täyttämiselle vaatisivat liian suuren pääomallisen sijoituksen. Lisäksi perinnejärjestelmät kuluttavat paljon rahaa ylläpitoon ja pienetkin muutokset saattavat aiheuttaa ongelmia ylläpitäjille.

Edwards, Mallalieu ja Thompson (1999) jatkavat määrittelytyötä ottamalla huomioon laajemman kontekstin, johon kuuluu järjestelmän lisäksi myös sen käyttäjät, johto, ylläpitäjät ja johtajat. Heidän määrittelynsä perustuu järjestelmien vaikutusten kontekstualisointiin ohjelmiston ominaisuuksien kuvaamisen sijaan. Tämä tarkoittaa sitä, että perinnejärjestelmää kuvailtaisiin keskenään vuorovaikutuksessa oleviksi elementeiksi, jotka muodostavat yhdessä kokonaisuuden. Yhden elementin muutos vaikuttaa tämän kokonaisuuden kautta useaan elementtiin. Perinnejärjestelmä on siis vaikeasti muokattava ja joustamaton, sillä ainoastaan yhtä komponenttia on vaikea muuttaa, ilman että sillä on vaikutusta muihin ympärillä oleviin komponenttiin tai niiden osiin. Samaa ajatukseen perinnejärjestelmän laajemman kontekstin huomioimisesta yhtyvät myös Brooke ja Ramage (2001), joiden mukaan perinneohjelmistot ja perinnejärjestelmät tulee erottaa toisistaan. Tämä tarkoittaa sitä, että perinnejärjestelmiin kuuluu ohjelmiston lisäksi myös järjestelmän muita osia. Näitä ovat ihmiset, asiantuntemus, laitteisto, tiedot, liiketoimintaprosessit, liiketoimintaympäristöt sekä lähestymistavat ohjelmistojen ylläpitoon ja kehittämiseen. Kaikki nämä asiat - ja erityisesti niiden keskinäinen vuorovaikutus - muodostavat perinnejärjestelmän. Lisäksi he toteavat, kuinka perinnejärjestelmän ohjelmistoa kuvataan usein joksikin tai kaikiksi seuraavista ominaisuuksista: suuret, vanhat, voimakkaasti muutetut, vaikeasti ylläpidettävät, vanhanaikaiset (Brooke & Ramage, 2001).

Matthiesen ja Bjørnin (2015) artikkeli tiivistää kaikki edellä mainitut määritelmät yhteen. Artikkelissa he siteeraavat Bennettin luomaa määritelmää perinnejärjestelmien joustamattomuudesta, suuresta iästä ja koosta. He tunnistavat myös aiemmin mainitut ominaisuudet perinnejärjestelmän ulkopuolisista tekijöistä. Eri lähteistä löytyneet tulokset perinnejärjestelmien ominaisuuksista on listattu taulukkoon 1.

Taulukko 1 Perinnejärjestelmien ominaisuudet eri julkaisuissa.

Lähde	Suuri koko	Joustamattomuus	Liiketoiminnalle kriittinen	Kallis	Vanha järjestelmä	Vanha koodikieli	Vanha laitteisto	Ei vastaa vaatimuksia	Vaikea ylläpitää
Matthiesen & Bjørn (2015)	X	X	X	X	X	X			
Khadka ym. (2014)			X		X				
De Lucia ym. (2001)	X	X	X	X	X			X	X
Brooke & Ramage (2001)	X				X	X		X	X
Bisbal ym. (1999)		X	X				X		
Edwards ym. (1999)		X							
Gold (1998)	X	X		X					
Bennett (1995)	X	X	X		X	X			
Brodie & Stonebraker (1995)		X							

Taulukon perusteella voimme todeta, että lähes kaikki tutkimukset jakavat Bennettin 1990-luvun puolivälissä luoman tulkinnan perinnejärjestelmistä. Taulukon perusteella voimme määritellä perinnejärjestelmän yleisimmin tunnistetut ominaisuudet:

- Vanha järjestelmä
- Liiketoiminnalle kriittinen
- Joustamattomuus
- Suuri koko

Lähdemateriaalista tehdyt havainnot osoittavat, että perinnejärjestelmän määrittely pelkästään ohjelmistotasolla on rajallinen. Perinnejärjestelmä koostuu ohjelmiston lisäksi myös ympäröivistä järjestelmän osista, ja ne vaikuttavat samaan tapaan ohjelmiston tapaan järjestelmän määrittelyyn perinnejärjestelmäksi. Tutkimuksissa mainittiin eri järjestelmän osien keskinäisen vuorovaikutuksen vaikuttavan erityisesti perinnejärjestelmän määritykseen. Näiden osien roolia tai vaikutustapoja järjestelmään ei uusissa tutkimuksissa todettu tai huomioitu. Uuden tutkimuksen perusteella perinnejärjestelmän ominaisuuksiin kuuluu myös muita järjestelmän osia, kuten käyttäjät, asiantuntemus, laitteisto, johto ja ylläpito.

3. PERINNEJÄRJESTELMIEN ONGELMAT

Kuten johdannossa jo mainittiin, ongelma perinnejärjestelmässä määritellään ominaisuudeksi, mikä vaikeuttaa sen jatkokehitettävyyttä täten lyhentäen sen elinkaarta. Ongelmia halutaan aina välttää. Tämä ei kuitenkaan tapahdu itsestään, vaan se vaatii aktiivista työtä järjestelmän jokapäiväisessä kehityksessä. Tällaisen välttämistyön edellytyksenä on laaja-alainen ymmärrys mahdollisista ongelmista ja niiden synnystä.

Perinnejärjestelmien ongelmia on dokumentoitu erilaisissa tutkimuksissa paljon, ja niitä esiintyy laajasti koko perinnejärjestelmien tutkimuksen saralla. Ongelmien tunnistaminen ja ymmärtäminen auttaa välttämään niiden syntyä. Tämä osio identifioi ongelmat, listaa ne ja vastaa kysymyksiin:

- Mitä ongelma tarkoittaa?
- Mistä ongelma johtuu?
- Miten ongelmaa vältetään?

3.1. Joustamattomuus

Aiemmassa kappaleessa tutkimme perinnejärjestelmien yhteisiä tunnusmerkkejä, ja yhdeksi näistä tunnusmerkeistä nousi joustamattomuus, jota voidaan kutsua myös muokkaamattomuudeksi. Joustamaton perinnejärjestelmä vastustaa muutosta (Brodie & Stonebraker, 1995) ja joissain tapauksissa on kykenemätön muuttumaan lainkaan (Bisbal ym., 1999) (Erlikh, 2000).

Nykyisiä perinnejärjestelmiä ei koskaan suunniteltu vastaamaan nykyisiä liiketoimintavaatimuksia. Monet vanhat järjestelmät on kehitetty ennen strukturoidun ohjelmoinnin yleistymistä. Tästä syystä prosessimallit ja perusperiaatteet, kuten modulaarisuus, sidonnaisuus, johdonmukaisuus ja hyvät ohjelmointikäytännöt, otettiin käyttöön liian myöhään. Näitä järjestelmiä kehitettiin tapauskohtaisten menetelmien mukaisesti, ja niissä käytettiin usein ohjelmointitekniikoita, jotka eivät sovellu suurten järjestelmien kehittämiseen (Warren, 1999). Warren listaa kolme syytä, miksi moni perinnejärjestelmä on tahattomasti luotu joustamattomaksi:

- Järjestelmän lyhyt elinajanodote. Järjestelmien käyttöönottohetkellä ei osattu odottaa, että nykyiset perinnejärjestelmät olisivat käytössä vielä vuosikymmeniä myöhemmin.
- Ohjelmistosuunnittelussa ei osattu huomioida järjestelmän evoluutiota.

- Järjestelmän kehitysvaiheessa jouduttiin huomioimaan sen aikaiset tekniset rajoitukset, kuten muisti ja prosessointiteho. Näiden teknisten vaatimusten täyttämiseksi järjestelmän ylläpidettävyydestä tingittiin. Pitkäkestoisissa liiketoimintajärjestelmissä ylläpidettävyys on järjestelmän laadun tärkein mittari.

Perinnejärjestelmän muokattavuutta tulisi lisätä, jotta järjestelmä kykenisi vastaamaan nopeasti muuttuvia liiketoimintavaatimuksia. Muokattavuuden lisäämiselle on olemassa hyvin vähän helppoja keinoja, eikä kirjallisuuskatsauksen yhteydessä löydetty vaihtoehtoja ongelman ratkaisemiseksi. Khadka, Batlajery, Saeid, Jansen, ja Hagen (2014) teetättivät tutkimuksessaan kyselyn perinnejärjestelmien asiantuntijoille selvittääkseen, mitkä perinnejärjestelmien ongelmat ajavat järjestelmän modernisoitavaksi. 85 % vastanneista asiantuntijoista vastasivat muokattavuuden lisäämisen syyksi modernisointityön aloittamiselle. Modernisointi voitaisiin siis nähdä yhtenä toimenpiteenä, jolla voidaan lisätä joustavuutta perinnejärjestelmään.

3.2. Dokumentoinnin- ja asiantuntijoiden puute

Jo vuonna 1996 Cook ja Visconti (1996) totesivat, että epätarkka, puuttuva, epätäydellinen tai puutteellinen dokumentaatio johtaa ohjelmiston laadun heikkenemiseen. Heidän mukaansa empiiriset tiedot osoittavat, että ohjelmiston dokumentointituotteet ja -prosessit ovat ohjelmistojen laadun keskeisiä osatekijöitä (Cook & Visconti, 1996). Siksi onkin merkittävää huomata, kuinka lukuisissa lähteissä mainitaan dokumentaation puute osana perinnejärjestelmien ongelmia.

Perinnejärjestelmissä on tyypillistä, että dokumentaatio katoaa (Matthiesen & Bjørn, 2015). Ylläpitäjien ja jatkokehittäjien on usein lähes mahdoton tietää, mitä dokumentteja on kateissa. Tämä johtuu siitä, että harvoin löytyy dokumentointia siitä, mitä järjestelmästä on dokumentoitu. Annett (2019) listaa yleisimpiä syitä dokumentoinnin puuttumiselle:

- Tukijärjestelmät voidaan poistaa käytöstä ja tiedot poistaa. Näin voi tapahtua silloin, kun dokumentaatiosta ei ole selvää yhteyttä järjestelmään, jota se koskee.
- Dokumentaatio on edelleen olemassa, mutta kukaan ei tiedä missä se on. Jos dokumentaatio on siirretty, niin linkit eivät ole enää voimassa. Tämä on yleinen ongelma, kun asiakirjahallintajärjestelmiä siirretään.
- Dokumentaatio ovat harvinaisessa tai tuntemattomassa muodossa, eikä niitä osata avata tai käsitellä.

- Dokumentaatio on tallennettu vain paikallisesti, tai niitä ei ole tallennettu lainkaan. Jos tiedot on tallennettu vain yhden koneen muistiin ja tietokone hajoaa, on riskinä, että dokumentaatio menetetään.

Perinnejärjestelmille on myös tyypillistä, että dokumentaatio ei ole ajan tasalla tai se on tehty huonosti (Matthiesen & Bjørn, 2015). Tällöin ainoana luotettava totuuden lähde on perinnejärjestelmän lähdekoodi (Bennett, 1995) (Matthiesen & Bjørn, 2015). Annett (2019) mainitsee, kuinka puutteellinen dokumentaatio on hyvin usein tarkoituksellista. Perinnejärjestelmän ylläpitäjät ja kehittäjät ovat vuosien mittaan kasvaneet osaksi järjestelmää, joten he ovat tärkeitä tekijöitä ylläpidossa. He ymmärtävät asemansa, ja jättävät tarkoituksenmukaisesti asioita dokumentoimatta suojatakseen näin oman työpaikkansa: Jos olet ainoa henkilö, joka osaa uudelleen käynnistää järjestelmän sen kaatuessa, tai luoda tietyn raportin, sinua on lähes mahdoton korvata. Tieto siis ei ole hukassa, vaan se on jätetty tarkoituksella dokumentoimatta (Annett, 2019) (Veerman, 2003).

Annett (2019) mainitsee, kuinka toimiva ohjelmisto on tärkeämpää kuin kattava ja yksityiskohtainen dokumentaatio. Hän jatkaa kertomalla, kuinka perinteisen ohjelmisto- ja järjestelmädokumentaation lisäksi perinnejärjestelmässä on paljon myös muuta kriittistä metadataa, mikä usein unohtuu dokumentoida. Tähän sisältyvät tukidokumentit, kuten:

- Käyttäjätunnukset ja salasanat: Järjestelmän käyttäjät ja järjestelmävalvojat varmasti tietävät omat tunnuksensa, mutta onko tietokantapalvelimen tunnukset tallennettu jonnekin?
- Ohjeet tuotantoon viemisestä: Perinnejärjestelmän lähdekoodi on tallessa, mutta tietääkö kukaan, miten lähdekoodi käännetään ja uusi versio julkaistaan?
- Versiohallinnan julkaisuhaara: Jos versiohallinnassa on useita haaroja ja versioita, kuka tietää mikä haara on tuotannossa?
- Kommunikaatioprotokollat ja ulkopuoliset järjestelmät: Onko järjestelmien välistä kommunikaatiota dokumentoitu? Mitkä protokollat ovat käytössä? Mitä viestit sisältävät? Mitkä järjestelmät ylipäätään kommunikoivat keskenään? Kuinka usein? Jos kommunikaatiodokumentaatio puuttuu, joutuu ylläpitäjä usein selvittämään kaiken tämän manuaalisesti testaamalla.

Perinnejärjestelmien asiantuntijoilla teetätetty kysely paljastaa, että yli 90 % asiantuntijoista olivat sitä mieltä, että merkittävimmät esteet järjestelmien modernisoinnissa ovat (Khadka ym. 2014):

- tietämyksen puute, erityisesti asiantuntijoiden vähyys.
- (ajantasaisen) dokumentaation puute.

Tutkimus vahvistaa aiemmista lähteistä löydetyn tiedon: Järjestelmään on hankala tehdä muutostöitä, jos dokumentaatio on puutteellista tai sitä ei ole ollenkaan. Tutkimuksessa mainitaan myös asiantuntijoiden vähyys esteenä perinnejärjestelmän modernisoinnille. Asiantuntijoiden puute onkin yleinen perinnejärjestelmien ongelma. Perinnejärjestelmien ylläpitoon tarvitaan järjestelmän ymmärryksen lisäksi myös taitoa vanhan teknologian osaamisesta (Sandborn & Prabhakar, 2015). Dokumentaation ollessa puutteellista järjestelmän osaaminen on muutaman asiantuntijan varassa, jotka ovat oppineet perinnejärjestelmän erityisosaajiksi. Järjestelmiä hallitsevien asiantuntijoiden palkkaaminen on käynyt yhä vaikeammaksi, joten yritykset saattavat joutua luopumaan muuten tyydyttävistä teknologioista osaamisen puutteen vuoksi (Hainaut ym., 2008) (Erlikh, 2000).

Yksi ratkaisu dokumentointiongelmaan on dokumentointiprosessin parantaminen (Cook & Visconti, 1996). Dokumentointiprosessin parantaminen ei kuitenkaan paranna vanhaa dokumentaatiota, vaan ainoastaan tehostaa dokumentoinnin tasoa jatkossa. Vanhan dokumentoinnin tasoa ehostetaan uudelleendokumentoimalla järjestelmää käyttäen eri menetelmiä (Sugumaran & Ibrahim, 2011).

3.3. Kallis ylläpidettävyys

Perinnejärjestelmien ylläpito on kallista (Bisbal., 1999) (Srinivas ym., 2015). Dwidedi, Henriksen, Wastell ja De' (2013) käsittelevät artikkelissaan perinnejärjestelmien ylläpidettävyyden kalleutta. Artikkelissa osoitetaan, miten ohjelmistojen ylläpidon suhteelliset kustannukset ovat kasvaneet aina 2000-luvulle saakka. Taulukossa 2 on kuvattu tutkimuksissa havaittujen ohjelmistojen ylläpitokustannuksien osuutta koko IT-budjetista vuosien 1980–2000 välisenä aikana (Dwivedi ym., 2013).

Vaikka Dwidedin ym. (2013) käsittelemät ylläpitojen kustannukset ovat vanhoista tutkimuksista, eivät korkeat ylläpidon kustannusten arviot ole juurikaan muuttuneet tuoreissa tutkimuksissa. Crotty ja Horrocks (2016) arvioivat 75 prosenttia pankkien ja vakuutusyhtiöiden IT-budjeteista kuluvan nykyisten järjestelmien ylläpitoon. Lisäksi

Yhdysvaltojen tietotekniikkaan osoittamasta budjetista arviolta 80 % kuluu järjestelmien ylläpitoon ja jokapäiväiseen operointiin (United States Government Accountability Office, 2019).

Taulukko 2 Ohjelmistojen ylläpitokustannukset eri julkaisuissa (Dwivedi ym. 2013)

Tutkimus	**ohjelmistojen ylläpitokustannukset**
Erikh (2000)	> 90 %
Eastwood (1993)	75 %
Port (1998)	60 % - 70 %
Lientz and Swanson (1980)	> 50 %

Kaikki nämä arviot osoittavat, että perinnejärjestelmien ylläpitoon on aina kulunut suurin osa IT-kustannuksista. On kuitenkin huomattava, että nämä kaikki luvut ovat arvioita, sillä tutkimusta perinnejärjestelmien aiheuttamista kuluista on tehty hyvin niukasti. Perinnejärjestelmän ylläpidon ja innovoinnin välinen epätasapaino IT-budjetissa painottuu voimakkaasti ylläpidon suuntaan. Tämä on valtava perinnejärjestelmiin liittyvä ongelma, sillä budjetin kuluminen perinnejärjestelmien ylläpitoon ja järjestelmätukeen on pois uusien innovaatioiden kehittämiseen kohdennetuista resursseista (Dwivedi ym., 2013).

Kallis ylläpidettävyys on seuraus järjestelmän muista ongelmista, kuten joustamattomuudesta, dokumentaation puutteesta ja kalliista osaamisesta. Ihanteellinen ratkaisu on muuttaa perinnejärjestelmät uudempiin ja tuottavampiin sovellusalustoihin, jotta yritykset voivat hyödyntää nopeampia ja halvempia kehitysmenetelmiä (Erlikh, 2000). Esimerkiksi Sneedin ja Verhoefin (2020) tutkimuksessa yksittäisen perinnejärjestelmän modernisoimisella saatiin vuosittaiset kulut laskemaan yhteensä miljoonalla eurolla. Khadkan ym. (2014) teettämässä tutkimuksessa 80,4 % asiantuntijoista totesivat, että perinnejärjestelmien korkeat ylläpitokustannukset ovat yksi merkittävimmistä perinnejärjestelmien modernisoinnin syistä.

3.4. Johdonmukaisuuden puute

Carlo mainitsee kirjoituksessaan (2021), kuinka perinnejärjestelmien ohjelmistokoodi on harvoin johdonmukaista. Mitä vanhemmaksi järjestelmän ohjelmistokoodi muuttuu, sitä enemmän erilaisia ohjelmistomalleja koodikantaan syntyy. Vuosien saatossa useat

ylläpitäjät ovat muokanneet lähdekoodia. Järjestelmän elinkaaren aikana tehdyt useat muutokset ovat johtaneet siihen, että lähdekoodi on jäsentymätöntä, mikä tekee jatkokehityksestä vaikeaa ja kallista (Veerman, 2003).

Suurin osa perinnejärjestelmistä ovat saaneet alkunsa vuosia sitten pieninä, yhden toiminnon järjestelminä. Uudet vaatimukset ovat johtaneet siihen, että perinnejärjestelmään on tehty tiheästi muutoksia ja ylläpitoa. Tämä on johtanut jäsentymättömään lähdekoodiin, jota on vaikea ylläpitää (Khadka ym., 2014). Pitkäikäisillä järjestelmillä on tyypillistä, että ne ovat kehittyneet ja kasvaneet vapaasti ilman ulkopuolisia rajoitteita. Kun tarkat määritelmät tai ohjelmistokäytännöt eivät ole rajoittamassa ohjelmiston kehitystä, se voi ajan kuluessa muuttua suureksi ja monimutkaiseksi järjestelmäksi. Tällaisesta järjestelmästä puuttuu johdonmukaisuus (Annett 2019).

Johdonmukaisuuden puute hankaloittaa valtavasti järjestelmän jatkokehitystä. Annett (2019) listaa tyypillisiä epäjohdonmukaisuuden tunnusmerkkejä:

- Liiketoimintalogiikan sijaitseminen väärässä paikassa. Järjestelmän kehittyessä tarvittavat arkkitehtuuriset muutokset on jätetty tekemättä.
- Ohjelmistokoodissa on useita työkaluja saman tehtävän suorittamiseen. Tämä johtuu siitä, että uusi kehittäjä lisää ominaisuuden käyttämällä tuntemiaan työkaluja perehtymättä järjestelmässä valmiina olleeseen toteutukseen.
- Ulkopuolisia kirjastoja käytetään tehtäviin, joihin niitä ei ole koskaan tarkoitettu.
- Äkilliset muutokset koodin tyylissä – mallien käytöstä formatointiin.

Tieteellistä tutkimusta johdonmukaisuuden parantamiseen perinnejärjestelmissä ei tätä tutkielmaa varten löytynyt. Carlo kertoo kuitenkin kirjoituksessaan (2021) muutamia tapoja, joilla johdonmukaisuutta voidaan parantaa. Hänen ratkaisunsa johdonmukaisuuden parantamiseen on jatkuva refaktorointi. Jokaisen ylläpitotyön yhteydessä lähdekoodi tulee jättää selkeämmäksi ja johdonmukaisemmaksi, kuin se oli ennen työn aloittamista. Uudet ohjelmointimallit tulee hyväksyä osaksi kehitystä, ja luottaa siihen, että jatkuvan refaktoroinnin kautta perinnejärjestelmästä tulee johdonmukaisempi. Carlo painottaa muutosten dokumentoinnin tärkeyttä, ja kaikkien automatisoitavissa olevien työvaiheiden automatisointia.

3.5. Käyttämättömät- ja ylikäytetyt toiminnallisuudet

Perinnejärjestelmät ovat vuosien aikana keränneet valtavasti erilaisia toiminnallisuuksia. Osa toiminnallisuuksista on vuosien aikana täysin unohtunut parempien toimintatapojen korvatessa niiden alkuperäisen tarkoituksen, ja osaa toiminnallisuuksista ei käytä enää kukaan. Perinnejärjestelmä voidaan myös löytää tilanteesta, jossa se on jo osittain korvattu uudella järjestelmällä. Tällöin on mahdollista, että vanha toiminnallisuus jää kummittelemaan perinnejärjestelmään, vaikka sitä ei enää kukaan käytä (Annett, 2019).

Ylimääräisistä toiminnallisuuksista tulee ongelma silloin, kun niitä ei ole dokumentoitu. Ylläpitäjä kuluttaa turhaan aikaansa käyttämättömän toiminnallisuuden korjaamiseen ja huomioimiseen. Lisäksi ongelma kasvaa, jos järjestelmää ollaan modernisoimassa, ja turhaa aikaa käytetään käyttämättömän toiminnallisuuden modernisointiin. On myös huomattava, että käyttämätön toiminnallisuus ei automaattisesti tarkoita, että toiminnallisuus olisi turha. Tästä hyvänä esimerkkinä on ydinreaktorijärjestelmään lisätty toiminnallisuus ydinonnettomuuden varalta. Toiminnallisuutta ei ole käytetty vuosiin, mutta se ei silti tarkoita sitä, että toiminnallisuutta ei tarvittaisi (Annett, 2019).

Käyttämättömien toiminnallisuuksien lisäksi perinnejärjestelmissä saattaa myös olla ylikäytettyjä toiminnallisuuksia. Ylikäytetyistä toiminnallisuuksista tulee ongelma silloin, kun toiminnallisuus pitää siirtää uuteen järjestelmään esimerkiksi modernisointityön yhteydessä. Toiminnallisuuden kopiointi uuteen järjestelmään kaikkine yksityiskohtineen on haastavaa ja joissain tapauksissa jopa mahdotonta. Tämä saattaa aiheuttaa käyttäjissä kehitysvastarintaa (Sneed & Verhoef, 2019).

3.6. Hauraus & integrointivaikeudet

Kuten jo aikaisemminkin on todettu, perinnejärjestelmän toistuvalla muokkauksella on kasautuva vaikutus, joka lisää sen monimutkaisuutta. Monimutkaisuus luo järjestelmään haurautta, joka ilmenee vaikeana muokattavuutena (Mehta & Heineman, 2002). Hauraus saa järjestelmän käyttäytymään arvaamattomasti, jolloin yhteen osa-alueeseen on lähes mahdotonta tehdä muutoksia ilman, että jokin toinen osa-alue kaatuu tai menee virhetilaan. Hauras järjestelmä on ylläpitäjän painajainen (Annett, 2019).

Hauras järjestelmä ei kuitenkaan tarkoita samaa kuin epätasapainossa oleva järjestelmä. Haurauden ja epätasapainon eroa tietojärjestelmissä voisi kuvailla korttitalolla: Korttitalo on vakaa, mutta hauras järjestelmä. Ilman ylimääräisiä vaikutteita se pysyy pystyssä ja toteuttaa sille annettua tehtävää. Kun yrität tehdä korttitaloon muutoksia, tai siihen

vaikuttaa jokin ennalta arvaamaton ulkopuolinen voima kuten tuuli, se sortuu helposti (Annett, 2019).

Usein järjestelmän haurautta lisää sen kommunikaatio ulkopuolisten järjestelmien kanssa. Kasvavien vaatimusten takia järjestelmään on luotu kommunikaatiokyky ulkopuolisiin järjestelmiin. Esimerkiksi yrityksen tuntikirjausjärjestelmään on jossain kohtaa täytynyt lisätä kyky keskustella uuden, kolmannen osapuolen palkanmaksujärjestelmän kanssa. Nämä rajapinnat järjestelmien välillä ovat äärimmäisen hajoamisalttiita muutosten tapahtuessa. Tilannetta ei helpota se, että lähes poikkeuksetta tarvittava dokumentaatio puuttuu tai ei ole ajan tasalla (Michiels ym. 2003).

3.7. Lisenssit & lainsäädäntö

Ohjelmistolisenssi on oikeudellinen asiakirja, jolla säännellään tekijänoikeudella suojattujen ohjelmistojen käyttöä tai uudelleenjakelua. Suuriin järjestelmiin on vuosien aikana kertynyt valtavasti erilaisia ja erikokoisia lisenssejä eri tarkoituksiin (Tuunanen ym. 2009). Lisenssi voi koskea lähes mitä tahansa ohjelmiston osaa: käyttöliittymäkomponentti, ulkopuolinen data, tuotantopalvelin ja niin edelleen. Perinnejärjestelmä on jatkanut kehitystään, mutta lisenssit ovat pysyneet samoina. Tämä saattaa rajoittaa järjestelmän kehitystä. Kuvitellaan ohjelmistotuote, joka on lisensoitu vain yhdelle palvelimelle. Perinnejärjestelmän kehittyessä järjestelmän osat halutaan virtualisoida niin, että yhdestä palvelimesta tulee monta virtualisoitua osaa. Tällaisissa tapauksissa lisenssistä tulisi maksaa jokaiselta virtualisoidulta osalta erikseen, mikä kasvattaa kustannukset moninkertaisiksi (Annett, 2019).

Perinnejärjestelmiä käyttävissä organisaatioissa on tyypillistä, ettei tiedetä mihin lisenssit on aikoinaan tallennettu ja kuka niistä vastaa. Lisenssimaksujen ajatellaan olevan uponneita kustannuksia, ja täten lisenssit ovat alttiita uponneiden kustannusten harhakuvitelmalle. Tämä tarkoittaa sitä, että organisaatio on vuosikymmeniä maksanut lisenssistä kuluja, joita ei voi enää palauttaa. Organisaatio on näin sitoutunut lisenssin käyttöön. Yrityspoliittisesti lisenssin vaihtaminen halvempaan vaihtoehtoon voi olla hankalaa, sillä ihminen ajattelee aiempien lisenssimaksujen menevän näin täysin hukkaan (Annett, 2019).

Lisenssien tapaan myös lainsäädäntö luo järjestelmän ulkopuolisia sääntöjä perinnejärjestelmän toimintaan ja kehitykseen. Lisenssien ehdot muuttuvat hyvin harvoin

perinnejärjestelmän elinkaaren aikana. Lainsäädäntö taas saattaa muuttua ja pakottaa perinnejärjestelmän mukautumaan uusiin säädöksiin (Annett, 2019).

4. POHDINTA

Tutkimuksen tavoitteena oli tutkia perinnejärjestelmissä ilmeneviä ongelmia ja niistä aiheutuvia haittoja. Lisäksi tavoitteena oli selvittää, kuinka perinnejärjestelmä määritellään. Perinnejärjestelmät eivät tule koskaan katoamaan keskuudestamme, joten on ensisijaisen tärkeää ja mielenkiintoista tutkia niiden ominaisuuksia ja vaikutuksia niin järjestelmän sisällä, kuin sen ulkopuolella.

Vaikka perinnejärjestelmien tutkimusta on tehty jo vuodesta 1990 alkaen, ei perinnejärjestelmälle ole löytynyt vieläkään yhteisymmärryksellistä määritystä ammattilaisten keskuudessa. Tutkimuksessa huomattiin, että eri tutkijat määrittelevät perinnejärjestelmät lähes poikkeuksetta järjestelmän ominaisuuksien mukaan. Näitä ominaisuuksia määrittelevät tutkimukset ajoittuvat 90-luvun loppuun ja vuosituhannen vaihteeseen. Perinnejärjestelmien tyypillisistä ominaisuuksista voidaan löytää selkeitä yhtymäkohtia eri tutkijoiden ja käytännön toimijoiden kesken, mutta erimielisyyksiä silti löytyy. Kirjallisuuskatsauksen avulla perinnejärjestelmät voidaan määritellä neljän ominaisuuden avulla:

- Vanha järjestelmä
- Liiketoiminnalle kriittinen
- Joustamattomuus
- Suuri koko

Kuitenkin pian huomattiin, että pelkkä perinnejärjestelmän ohjelmisto on yksinään puutteellinen määritelmä perinnejärjestelmälle. Täten määrittelyyn haluttiin ottaa mukaan myös muut järjestelmän osa-alueet, kuten laitteisto, käyttäjät, asiantuntemus, johto ja ylläpito. Siitä millaisia näiden osa-alueiden ominaisuudet perinnejärjestelmässä tarkalleen ovat, ei kirjallisuuskatsauksen yhteydessä löydetty tutkittua tietoa. Näiden ominaisuuksien, ja niiden luomien ongelmien tarkempi kuvailu ja määritelmä perinnejärjestelmien kontekstissa voisi olla mahdollinen jatkotutkimusaihe perinnejärjestelmien tutkimuksen saralla.

Perinnejärjestelmien ongelmia on käsitelty lähdeaineistossa 80-luvulta aina tähän päivään saakka. Suurin kontribuutio on kuitenkin tapahtunut 90-luvun lopussa. Ongelmia on tarkennettu sitä mukaa, kun tutkimusosaaminen on kehittynyt. Tutkimuksen aikana huomattiin, miten koottua, tieteellistä kirjallisuuskatsausta ei perinnejärjestelmien ongelmiin juuri löytynyt. Tämä edelleen korostaa kirjallisuuskatsauksen tärkeyttä.

Perinnejärjestelmiä on vaikea, ja joissain tapauksissa jopa mahdotonta jatkokehittää. Jatkokehityksen vaikeus tiivistääkin hyvin kaikki kirjallisuuskatsauksessa tehdyt havainnot perinnejärjestelmien ongelmista. Huono jatkokehitettävyys johtuu koodikannan laajuudesta ja huonosta dokumentaatiosta. Vuosien aikana järjestelmässä on työskennellyt usea eri ylläpitäjä käyttäen erilaisia ohjelmointimalleja ja -käytänteitä. Kun tarvittavat yhdenmukaisuutta parantavat refaktoroinnit on jätetty tekemättä, on järjestelmässä ohjelmointikäytänteitä useilta eri vuosikymmeniltä. Dokumentaation puuttuessa tiettyjä toiminnallisuuksia on saatettu toistaa useisiin eri kohtiin lähdekoodissa, mikä entisestään huonontaa jatkokehitettävyyttä. Voi jopa tuntua siltä, että lähdekoodia on tahallisesti yritetty syövyttää, jotta tulevaisuuden kehitys olisi mahdollisimman vaikeaa.

Perinnejärjestelmässä esiintyy riippuvuuksia niin järjestelmän sisällä, kuin järjestelmän ulkopuolella. Nämä erilaiset sidokset tekevät järjestelmästä hauraan ja joustamattoman. Tämä tarkoittaa, ettei mitään muutoksia voida tehdä ilman, että se vaikuttaa johonkin toiseen osa-alueeseen. Hauraus ja joustamattomuus tekevät perinnejärjestelmästä entistä huonommin jatkokehitettävän. Huonosta jatkokehitettävyydestä kärsii koko organisaatio, ja se näkyy perinnejärjestelmän kustannuksissa. Perinnejärjestelmän ylläpidon kalleus johtuu aiemmin listatuista ongelmista. Kun koodikanta on huonoa ja dokumentointi puutteellista, vaatii kaikki toimenpiteet ylimääräistä selvitysaikaa ja osaamista. Mikä tahansa muutos lähes aina vaikuttaa ympärillä oleviin toimintoihin, ja joissain tapauksissa jopa muihin järjestelmiin. Tällaisia riippuvaisuuksia on puutteellisen dokumentaation ja epäjohdonmukaisen koodikannan takia mahdotonta ennustaa etukäteen. Täten riippuvaisuudet saattavat aiheuttaa ennustamattomia virhetilanteita ja ongelmia, jotka vaativat aina lisäselvitysaikaa. Perinnejärjestelmät on usein kehitetty käyttäen vanhoja ohjelmointikieliä, joiden osaajia on saatavilla hyvin vähän. Olemassa olevat ylläpitäjät ovat keränneet vuosien aikana valtavan osaamispääoman järjestelmästä, ja ovat täten korvaamattomia.

Ongelmia on hankala välttää, sillä pitkäikäisillä järjestelmillä on tapana, että ne ovat kehittyneet ja kasvaneet vapaasti ilman ulkopuolisia rajoitteita. Perinnejärjestelmän kehitystä voisi verrata suureen kaupungin kehitykseen, joka on alun perin saanut alkunsa pienenä kylänä, mutta ajan kuluessa se on laajentunut sattumanvaraisesti jokaiseen suuntaan. Taloja on rakennettu mihin sattuu, tiet ovat muodostuneet sinne mistä ihmisten on kätevin kulkea sen sijaan, että teille olisi suunniteltu mahdollisimman tehokkaat ja hyvät kulkureitit. Kaupungin kasvaessa sen infrastruktuuri ei juurikaan ole mukautunut

kasvuun, ja suuria korjaustoimenpiteitä, kuten teiden leventämistä kasvavan liikennemäärän takia, ei ole tehty. Nykyään taloja ei voida enää siirtää pois edestä, joten infrastruktuurin on mukauduttava aikoja sitten tehtyjen päätösten mukaisesti. Suurta kaupunkia ei koskaan suunniteltu suureksi kaupungiksi. Samaan tapaan perinnejärjestelmääkään ei koskaan suunniteltu isoksi, liiketoiminnalle kriittiseksi järjestelmäksi. Kaupungin tapaan sekin on saanut alkunsa yksinkertaisena ja helppona kokonaisuutena, joka on vain sattunut kasvamaan valtavasti.

Kirjallisuushaun aikana huomattiin, kuinka perinnejärjestelmien ongelmien määritystä on tehty paljon 1990-luvun ja 2000-luvun taitteessa. Lisäksi lähes kaikki tuoreetkin tutkimukset käyttävät lähteinään ja teoriapohjanaan näitä artikkeleita ja teoksia vuosituhannen taitteesta. Tämä tarkoittaa sitä, että tuoreen tutkimuksen avulla voitaisiin selvittää nykyajan perinnejärjestelmien piirteitä, ja tutkia ovatko ne muuttuneet 1990-luvun järjestelmistä. Selvityksen kohteena voisi olla esimerkiksi uusien ongelmien ilmeneminen sekä vanhojen ja uusien ongelmien vertailu. Tällaisesta tutkimuksesta olisi hyötyä organisaatioille, jotka eivät tahdo ajautua perinnejärjestelmän ylläpitokierteeseen. Lisäksi kirjallisuuskatsauksessa löydettyjen ongelmien ymmärrys selkeyttää niiden vaikutusten käsittämistä. Tämä helpottaa perinnejärjestelmien modernisoinnin tutkimusta.

5. YHTEENVETO

Perinnejärjestelmät ovat haaste organisaatioille. Kasvavat ylläpitokulut ja liiketoimintavaatimukset yhdistettynä alati innovatiivisempaan yrityskenttään pakottavat organisaatiot tekemään ratkaisuja perinnejärjestelmiensä suhteen. Perinnejärjestelmistä ja niiden huonoista ominaisuuksista on puhuttu ja varoiteltu eri tahojen toimesta jo vuosikymmeniä, mutta silti niitä edelleen löytää lähes jokaisesta isommasta organisaatiosta. Se todistaa niistä eroon pääsemisen vaikeuden.

Ongelmista puhuttaessa oletetaan, että uusi järjestelmä on aina parempi ja hienompi kuin perinnejärjestelmä. Modernit ohjelmointikielet ja työkalut ovat ohjelmoijien suosiossa, mutta uudet järjestelmät tuovat silti mukanaan omia tyypillisiä ongelmiaan. Onkin mielenkiintoista ajatella sitä kehityspolkua, mitä vasta valmistunut uusi järjestelmä kulkee vain muuttuakseen lopulta omaksi perinnejärjestelmäkseen. Olen sitä mieltä, että jos kenen tahansa annettaisiin valita varman mutta joustamattoman, ja epävarman sekä joustavan välillä, kaikki valitsisivat ensimmäisen vaihtoehdon tilanteesta riippumatta.

On mielenkiintoista huomata, kuinka sanat ”legacy” tai ”perinnejärjestelmä” saavat monissa aikaan pelkkiä negatiivisia ajatuksia. Sanaa voisi verrata muihin tieteen aloihin, missä sana ”perinne” tai ”perintö” liitetään pikemminkin johonkin arvokkaaseen kuin ongelmaan. Ehkäpä emme IT-maailmassa osaa arvostaa näitä järjestelmiä tarpeeksi. Ovathan ne kuitenkin lukemattomien vuosien ajan hoitaneet tehtävänsä moitteettomasti. Ehkä perinnejärjestelmiä tulisikin arvostaa ja vaalia samalla tavalla, kuin tavallisia perinteitä.

Lähteet

Annett, R. (2019). Working with Legacy Systems: A practical guide to looking after and maintaining the systems we inherit. Packt Publishing Ltd.

Bennett, K. (1995). Legacy systems: coping with success. IEEE Software, 12 (1), 19–23. doi:10.1109/52.363157

Bisbal, J., Lawless, D., Bing Wu, & Grimson, J. (1999). Legacy information systems: issues and directions. IEEE Software, 16(5), 103–111. doi:10.1109/52.795108

Brodie, M. L., & Stonebraker, M. (1995). Migrating legacy systems: gateways, interfaces & the incremental approach. Morgan Kaufmann Pub.

Brooke, C., & Ramage, M. (2001). Organisational scenarios and legacy systems. International Journal of Information Management, 21(5), 365-384. http://oro.open.ac.uk/2667/1/IJIM2001.pdf

Carlo, N., (2021), Should we stick to old patterns or sacrifice consistency?. Haettu osoitteesta: https://understandlegacycode.com/blog/consistency-or-progress/

Cook, C. R., & Visconti, M. (1996). New and improved documentation process model. Oregon State University, Department of Computer Science.

Crotty, J., & Horrocks, I. (2017). Managing legacy system costs: A case study of a meta-assessment model to identify solutions in a large financial services company. Applied computing and informatics, 13(2), 175-183.not

De Lucia, A., Fasolino, A. R., & Pompelle, E. (2001). A decisional framework for legacy system management. In Proceedings IEEE International Conference on Software Maintenance. ICSM 2001 (pp. 642-651). IEEE.

Dwivedi, Y. K., Henriksen, H. Z., Wastell, D., & De', R. (Eds.). (2013). Grand Successes and Failures in IT. Public and Private Sectors. IFIP Advances in Information and Communication Technology. doi:10.1007/978-3-642-38862-0

Eastwood, A. (1993). "Firm fires shots at legacy systems". Computing Canada 19 (2), p. 17.

Edwards, H. M., Mallalieu, G. M., & Thompson, J. B. (1999, October). Some insights into the maintenance of legacy systems within small manufacturing and distribution organisations in the UK. In Proceedings. Twenty-Third Annual International Computer Software and Applications Conference (pp. 13-20). IEEE.

Erlikh, L. (2000). Leveraging legacy system dollars for e-business. IT professional, 2(3), 17-23.

Gold, Nicolas (1998). "The Meaning of 'Legacy Systems'". Technical Report 7/98, Dept. of Computer Science, Durham University, UK.

Khadka, R., Batlajery, B. V., Saeidi, A. M., Jansen, S., & Hage, J. (2014). How do professionals perceive legacy systems and software modernization? Proceedings of the 36th International Conference on Software Engineering - ICSE 2014. doi:10.1145/2568225.2568318

Lehman, M. M. (1980). Programs, life cycles, and laws of software evolution. Proceedings of the IEEE, 68(9), 1060-1076.

Lientz, B.P. & Swanson, E. (1981). "Problems in application software maintenance". Communications of the ACM 24 (11), 763-769.

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

M'baya, A., Laval, J., & Moalla, N. (2017). An assessment conceptual framework for the modernization of legacy systems. 2017 11th International Conference on Software, Knowledge, Information Management and Applications (SKIMA). doi:10.1109/skima.2017.8294120

Mehta, A., & Heineman, G. T. (2002). Evolving legacy system features into fine-grained components. In Proceedings of the 24th international Conference on Software Engineering (pp. 417-427).

Michiels, I., Deridder, D., Tromp, H., & Zaidman, A. (2003). Identifying problems with legacy software; preliminary findings of the ARRIBA project. In Proceedings of

ELISA, International Workshop on Evolution of Large-scale industrial Software Applications.

Micro Focus 2022. COBOL Market Shown to be Three Times Larger than Previously Estimated in New Independent Survey. Haettu osoitteesta https://www.microfocus.com/en-us/press-room/press-releases/2022/cobol-market-shown-to-be-three-times-larger-than-previously-estimated-in-new-independent-survey

Port, O. (1988). “The software trap – automate or else”. Business Week 3051 (9), 142-154.

Sandborn, P. A., & Prabhakar, V. J. (2015). The Forecasting and Impact of theLoss of Critical Human Skills Necessary for Supporting Legacy Systems.IEEE Transactions on Engineering Management, 62(3), 361–371.https://doi.org/10.1109/tem.2015.2438820

Sneed, H., & Verhoef, C. (2019). Re-implementing a legacy system. Journal of Systems and Software, 155, 162-184.

Sneed, H. M., & Verhoef, C. (2020). Cost-driven software migration: An experience report. Journal of Software: Evolution and Process. 32(7), 1-26. https://doi.org/10.1002/smr.2236

Srinivas, M., Ramakrishna, G., Rao, K. R., & Babu, E. S. (2016). Analysis of legacy system in software application development: A comparative survey. International Journal of Electrical and Computer Engineering, 6(1), 292.

Stevenson, M., & Helmond, A. (2020). Legacy systems: internet histories of the abandoned, discontinued and forgotten. Internet Histories, 4(1), 1-5.

Sugumaran, N., & Ibrahim, S. (2011). An Evaluation on Software Redocumentation Approaches and Tools in Software Maintenance. Communications of the IBIMA, 2. doi:10.5171/2011.875759

Tuunanen, T., Koskinen, J., & Kärkkäinen, T. (2009). Automated software license analysis. Automated Software Engineering, 16(3), 455-490.

United States Government Accountability Office. (2019). Agencies Need to Develop Modernization Plans for Critical Legacy Systems (Raportti nro. GAO-19-471). Haettu osoitteesta https://www.gao.gov/assets/700/699616.pdf

Veerman, N. (2004). Revitalizing modifiability of legacy assets. Journal of Software Maintenance and Evolution: Research and Practice, 16(45), 219–254. doi:10.1002/smr.295

Warren, I. (1999). The renaissance of legacy systems: method support for software-system evolution. Springer Science & Business Media