EMULOINTI RATKAISUNA DIGITAALISEN AINEISTON PITKÄ-AIKAISSÄILYTYKSEN ONGELMAAN

Samuel Ranta

Tampereen yliopisto
Informaatiotieteiden yksikkö
Informaatiotutkimus ja interaktiivinen media
Pro gradu -tutkielma
Toukokuu 2011

TAMPEREEN YLIOPISTO, Informaatiotieteiden yksikkö
Informaatiotutkimus ja interaktiivinen media
RANTA, SAMUEL: Emulointi ratkaisuna digitaalisen aineiston pitkäaikaissäilytyksen ongelmaan
Pro gradu -tutkielma, 141 s.
Toukokuu 2011

Digitaalisten aineistojen pitkäaikaissäilytys on ongelmana akuutti ja toistaiseksi vailla muistiorganisaatioiden yhteiseksi omaksumaa ratkaisua. Erityisiä vaikeuksia aiheuttavat multimediaa ja interaktiivisia toimintoja tarjoavat aineistot, kuten ohjelmistot. Suomalaisissa muistiorganisaatioissa käytössä olevat säilytysmenetelmät eivät kykene säilyttämään tällaisia aineistoja lainkaan.

Tässä tutkimuksessa selvitettiin emuloinniksi kutsutun menetelmän vahvuuksia sovellettaessa sitä digitaalisen aineiston pitkäaikaissäilytykseen. Tutkimusmenetelmänä käytettiin kirjallisuuskatsausta, johon yhdistettiin emulointikokeita. Tutkimusaineisto kerättiin kolmen vuoden aikana internetin hakukoneista Googlesta, Yahoosta, Altavistasta. Tietokantoina käytettiin EBSCOhost Academic Search Premier-, LISA: Library and Information Science Abstracts-, Emerald- ja Communication & Mass Media Complete (Ebsco) -hakutietokantoja. Hankintaehdotuksia tehtiin Tampereen yliopiston kirjastoon kirjojen loppuun painettujen mainosten perusteella. Tuloksena oli kattava valikoima alan luotettavinta tutkimuskirjallisuutta ja -artikkeleita.

Tutkimuksessa osoitettiin, että emulointi on välttämätön osa tietokoneilla tuotetun aineiston pitkäaikaissäilytystä. Jos emulointi ei ole käytössä muistiorganisaatiossa, jää silloin säilytyksen ulkopuolelle merkittävä osa aikamme tuottamaa aineistoa. Tutkimus osoitti myös, että suomalaisessa arkistointikulttuurissa emulointia ei ole kunnolla tutkittu, ja se on ohitettu säilytysmenetelmänä kevyin ja pinnallisin perustein. Emuloinnin suurin etu on sen kyky säilyttää aineisto kaikilla alkuperäisillä ominaisuuksillaan varustettuna. Tähän ei kykene mikään muu menetelmä. Emuloinnin historian tutkiminen osoitti emuloinnin olevan tietokoneteollisuudessa erittäin hyvin tunnettu tekniikka. Väistämätön johtopäätös tästä oli, että emulointi toimii myös pitkäaikaissäilytyksessä, jos sen kehittämiseksi pitkäaikaissäilytyksen tarpeisiin kohdistetaan tarpeeksi resursseja. Tutkimuksessa tarkastellaan myös emuloinnin ja virtualisoinnin käsitteitä, niihin liittyviä tulkintaepäselvyyksiä ja niiden käyttöä kirjallisuudessa.

Avainsanat: emulointi, emulaattori, emulaatio, pitkäaikaissäilytys, virtualisointi

Huomautuksia lukijalle

Tässä työssä olen käyttänyt pohjana OAIS-mallin suomennosta standardia SFS 9742. Olen kuitenkin lisännyt viitemallin eri osien eteen sanan 'moduuli'. Tämän olen tehnyt siksi, ettei viitemallin säilytys- ja muita toimintoja suorittavat osiot sekaantuisi järjestelmään talletettavaan aineistoon, tietopaketteihin, joista tuo aineisto koostuu. Olen myös vaihtanut joidenkin osien nimiä mielestäni paremmiksi suomennoksiksi. 'Access' on standardissa suomennettu 'pääsyksi' ja 'saatavuudeksi', mutta mielestäni 'portti' kuvaa paremmin tuon moduulin roolia OAIS-mallissa, sillä sen kautta tapahtuu järjestelmän kaikki käyttö aineistoja etsivän asiakkaan taholta.

Komponenttiemulointi tunnetaan lähdekirjallisuudessa moduuliemulointina, mutta olen käyttänyt moduuliemuloinnin sijasta käsitettä 'komponenttiemulointi'. Näin siksi, että komponenttiemulointi kuvaa mielestäni paremmin tuon emulointimenetelmän keskeistä ajatusta, eli yksittäisten tietokonekomponenttien emulointia.

Tietopakettien alkuperäiset, englanninkieliset nimet olen kuitenkin säilyttänyt, sillä niiden suomennoksesta seuraisi väistämättä se, että olisi myös käytettävä suomennoksesta johdettuja lyhenteitä. Katson että tämä saattaisi sekoittaa lukijan lopullisesti, varsinkin jos tämä haluaa tarkistella myös alkuperäisiä lähteitä, joiden pohjalta työ on kirjoitettu joissa sitten on taas käytössä toiset lyhenteet. Lukemisen helpottamiseksi olen laatinut työn loppuun keskeisistä käsitteistä sanaston.

Kiitokset

Kiitos Herra Jeesus siitä, että annoit voimaa tämän työn loppuun saattamiseen. Sinussa on kaikki ja Sinä olet kaikki. Ilman Sinua minä en mihinkään pysty.

Kiitos rakastaville vanhemmilleni, jotka ovat väsymättä tukeneet tämän työn kanssa henkisesti ja taloudellisesti!

Sisällysluettelo

1. JOHDANTO

Tietoteknistä kehitystä on aina leimannut lyhytnäköisyys. Tämän näimme viimeksi vuosituhannen vaihtumisen yhteydessä, jolloin maailma heräsi siihen, että edellisellä vuosituhannella oli asennoituminen ollut hyvin lyhytjänteistä. Sovellukset oli ohjelmoitu ottamaan huomioon vain kaksi viimeisintä numeroa vuosiluvuissa aiheuttaen sen, että ne olivat kykenemättömiä ymmärtämään vuosilukua 2000. Asian paljastuminen aiheutti suurta tyrmistystä ja ihmetystä ohjelmointi-insinöörien lyhytnäköisestä lähestymistavasta. 70- ja 80-luvuilla oli ajateltu vuoden 2000 olevan niin kaukana ettei sitä tarvitse ottaa millään tavalla huomioon. Jos ongelmia tästä linjauksesta aiheutuisikin, luotettiin tulevaisuuden insinöörien kehittyneempiin kykyihin ratkaista käsillä olevat ongelmat. Suhtautumistapa kostautui myöhemmin moninkertaisesti tulevaisuuteen työnnettyjen ongelmien äkkiä kaatuessa päälle. Yhteiskunnan tietotekninen kehitys on saavuttanut pisteen jossa sen tulee jo katsella taaksepäin ja analysoida menneisyyttään, sen saavutuksia ja virheitä ja suunnitella tulevaisuuttaan.

Nyt digitaalisten aineistojen suhteen on saavuttu tilanteeseen, joka muistuttaa vuosituhannen vaihtumisen yhteydessä esiin tulleita ongelmia. Tällä kertaa ongelma ei tosin koske vain vuosilukuja, vaan itse aineistojen säilymistä tulevaisuuden ihmisille. Jokainen kansalainen, elipä hän sitten missä maassa hyvänsä, on väistämättä jossain määrin riippuvainen digitaalisen aineiston pitkäaikaissäilytyksestä. Lähes kaikki nykyisin luotava tietoaines laaditaan ensin digitaaliseen muotoon. Yhä suurempi osa tuosta tiedosta jää digitaaliseen muotoon, saamatta koskaan fyysistä muotoa. (Gladney 2007, 3.) Ongelma on siis vakavuudeltaan kokonaan toista kertaluokkaa.

Millenniumohjelmointivirhettä voisi verrata tihkusateeseen, jonka rinnalla tulevan katastrofin uhka on kuin kaatosade. Jos digitaalisten aineistojen pitkäaikaissäilytys laiminlyödään, on mahdollista että aikakaudestamme ei jää jälkipolville mitään kirjallisia todisteita. Tästä uhkakuvasta tulee enenevässä määrin todellisuutta julkishallinnon siirtyessä kohti paperitonta toimistoa. Paperin etu verrattuna sähköisiin aineistoihin on se, että se säilyttää asiakirjalliset ominaisuutensa vähäisilläkin säilytystoimilla. Riittää, että paperia varjellaan luonnon tuhoavilta voimilta. Siksi suomessakin arkistolaitoksen kanta oli pitkään kaiken aineiston tulostaminen ja mappeihin vieminen. Jos paperiasiakirjoja

säilytetään ihanteellisissa olosuhteissa, sellaisissa jotka vallitsevat arkistojen pitkäaikaissäilytysvarastoissa, on todennäköistä että paperi säilyy lukukelpoisena satoja vuosia. Toisin sanoen niin pitkään, että mahdolliset pelastustoimet uhanalaisen aineiston pelastamiseksi ehditään hyvissä ajoin suunnitella ja toteuttaa.

Digitaalisen aineiston kanssa sen sijaan on jopa kiire suorittaa säilyttämistoimenpiteitä jotta ne säilyttäisivät todistusvoimaisuutensa. Jos mitään ei tehdä, aineisto muuttuu itsestään suhteellisen lyhyen ajan sisällä käyttökelvottomaksi. Arviot siitä kuinka nopeasti teknologia ja siihen kytkettyjen aineistojen lukumahdollisuudet vanhentuvat, jos oletamme aineistojen suhteen vallitsevan täydellisen passiivisuuden, vaihtelevat suuresti. Rusbridge (2006, 3) on sitä mieltä, että aineisto pysyy lukukelpoisena 10-15 vuotta ilman säilytystoimenpiteitä. Rothenbergin (1997, tässä Rothenberg 1999,2) mukaan tämä aika on niinkin lyhyt kuin viisi vuotta. Tämä pitää yhtä kiintolevyvalmistajien tuotteilleen lupaaman korkeintaan 5 vuoden takuun kanssa (Hämäläinen 2011, 4).

Mutta vaikka aineisto itsessään säilyisikin lukukelpoisena, muuttuu se joka tapauksessa käyttökelvottomaksi tarvittavien lukulaitteiden kadotessa markkinoilta ja mennessä epäkuntoon. Tätä kirjoittaessa on Suomessa menossa massiivinen VHS-aineistojen siirtäminen dvd-levyille kotitalouksissa. Ilman tätä migraatioksi kutsuttua siirtoa VHS-nauhat muuttuvat arvottomiksi, sillä niiden katselemiseen suunnitellut laitteistot menevät väistämättä vanhuuttaan epäkuntoon, eikä uusia enää valmisteta. Jos katselulaite, tässä tapauksessa VHS-nauhuri, on vaikeasti saatavilla, on hyvin todennäköistä että kaikki vähempiarvoiseksi katsottu aineisto tulee hylätyksi. Emme voi kuitenkaan tietää mitkä aineistot meidän ajastamme nähdään tulevaisuudessa merkittävinä ja siksi perintömme on sitä suuremmassa vaarassa, mitä huolettomammin suhtaudumme pitkäaikaissäilytyksen kysymyksiin. Jos riittäviin toimenpiteisiin ei tarpeeksi ajoissa ryhdytä, aika hävittää lopulta tasapuolisesti kaiken tuottamamme aineiston.

Digitaalisen aineiston kannalta ongelmallisinta on, ettei pelkkä aineiston fyysinen säilyttäminen tarkoita vielä säilyttämistä laisinkaan. Kaikki sähköiset aineistot on laadittu käyttäen jotakin ohjelmistoa. Tiedoston sisältö on vain tuon ohjelmiston tai samaa tiedostomuotoa tukevan ohjelmiston ymmärrettävissä. Ilman ohjelmistoa emme pääse aineistoon käsiksi. Tilannetta helpottamaan on kehitetty erilaisia katseluohjelmia. Nämä ovat kuitenkin ikään kuin vain aspiriini päänsärkyyn. Särky kyllä poistuu, mutta ongel-

man varsinainen aiheuttaja säilyy. Tässä tapauksessa "säryn", ongelman, aiheuttaa se, että tiedostot ovat pohjimmiltaan bittiröykkiöitä, jonoja ykkösiä ja nollia, jotka eivät sellaisenaan tarkoita yhtään mitään. Merkityksensä nämä tiedostot saavat vasta kun ne kulkevat tulkitsijan läpi. Tässä tapauksessa tuo tulkitsija on ohjelmisto, jolla asiakirja on laadittu.

Sama lähestymistapa, joka toimii esimerkiksi VHS-nauhoissa, ei siis toimi tietokoneella laadituissa aineistoissa,. Tästä on aiheutunut pulma, jota on alettu kutsumaan digitaaliseksi dilemmaksi. Näyttää vahvasti siltä, että digitaalista aineistoa ei voi millään tavoin säilyttää oikein. Varsinkin, jos säilytysmenetelmäksi valitaan vain yksittäinen tai hyvin kapea joukko erilaisia säilytysmenetelmiä. Sujuva, toimiva ja kaiken aineiston kattava säilytysmenetelmä edellyttää useiden toistensa kanssa saumattomasti yhteen toimivien menetelmien yhdistämistä.

Digitaalisten aineistojen arkistoiminen tarkoittaa siis käytännössä sitä, että elektroniselle julkaisulle annetaan tunnisteet, joiden perusteella ne voidaan löytää ja tuotetaan niihin kuvailevaa metadataa. Näiden lisäksi on digitaalinen aineisto vietävä kontrolloituun säilytysympäristöön. (Diessen & Steenbergen 2002d).

Tässä työssä keskityn tutkimaan emuloinniksi kutsuttua säilytysmenetelmää. Emuloinnissa toisen tietokoneen toimintaa jäljitellään ohjelmallisesti siten, että tämän emulaattoriohjelmiston kautta kyetään ajamaan jäljitellyille eli emuloiduille laitteistolle kirjoitettuja ohjelmistoja. Jäljittelyä voidaan tehdä myös laitteiston kautta, mutta tämä työ keskittyy pääasiassa ohjelmallisesti tapahtuvaan jäljittelyyn. Emulointi on monelle tuttu Windows-käyttöjärjestelmään sisäänrakennetusta DOS-emulaattorista, jonka avulla kykenee käyttämään monia vanhoja ohjelmistoja. Tietotekniikkateollisuus on pakostakin joutunut turvautumaan emulointiin koko olemassaolonsa ajan. Ilman emulointia kuluttajat olisi saatettu tietokonemarkkinoilla sietämättömään päivityskierteeseen. Uudet laitteistosukupolvet olisivat aina olleet yhteensopimattomia vanhojen ohjelmistojen ja vanhat ohjelmistot uusien laitteistojen kanssa. (Rothenberg 1999, 25.)

Emulointia on myös käytetty uusien laitteistojen testaamiseen ennen näiden rakentamista. Emuloinnilla vasta suunnitteilla oleva tietokoneen komponentti tai kokonainen jär-

jestelmä voidaan luoda ohjelmallisesti ja testata sen toimintaa ennen varsinaisen fyysisen tuotannon käynnistämistä. (Rothenberg 1999, 25.)

Tuttu jokapäiväinen esimerkki emuloinnista on myös vanhojen 80-luvun pelien pelaaminen uusissa tietokoneissa emulaattoreita käyttämällä. Tämä harrastus on saavuttanut valtavan suosion. Se antaa hyvin vahvan näytön emuloinnin kyvystä herättää henkiin kymmeniä vuosia vanhoja ohjelmistoja. Emuloinnin vastaansanomaton etu muihin säilytysmenetelmiin verrattuna on sen kyky tuoda aineisto tarkasteltavaksi juuri sellaisilla ulkonäöllisillä ja toiminnallisilla ominaisuuksilla, jotka sille sen laatimishetkellä asetettiin. Monet pitkäaikaissäilytysorganisaatiot ovat hylänneet emuloinnin, sillä ne pitävät sitä teknisesti liian vaativana, virheherkkänä ja kalliina (Hoeven & Wijngaarden 2005, 1). On kuitenkin olemassa aineistoja, joita ei kyetä säilyttämään ilman emulointia siten, että niiden kaikki tietosisältö säilyisi. Tällaisia ovat mm. multimediaa sisältävät aineistot, hypermedia ja kaikki tietokonepelit. Siksi pitkäaikaissäilytysjärjestelmästä puuttuu jotakin hyvin oleellista ilman emulointia ja on huolestuttavaa, että sen rooli on hyvin pieni kotimaisten pitkäaikaissäilytysorganisaatioiden strategioissa tai se on jätetty kokonaan pois.

Kysymys ei kuitenkaan ole vain tekninen, sillä digitaalisen aineiston pitkäaikaissäilytykseen liittyy tekniikan lisäksi myös laillisia, poliittisia, organisaationaalisia, hallinnollisia ja koulutuksellisia näkökulmia. Laillisista kysymyksistä suurin on tekijänoikeuskysymysten ratkaiseminen kaikki osapuolia tyydyttävällä tavalla. Tietokonelaitteistojen ja -ohjelmistojen suhteen tulisi laatia oma lainsäädäntönsä, joka mahdollistaisi laillisen emuloinnin ja sen kehittämisen. Hyvin suuri osa emuloinnista maailmassa tapahtuu nyt laittomasti, varsinkin vanhojen tietokonepelien alueella. Koska vanhoilla peleillä ei ole markkina-arvoa, ei tilanteeseen ohjelmistojen tuottajien taholta juurikaan puututa ja siksi niin kutsuttua retropelaamista katsotaan organisaationaalisella tasolla läpi sormien. Aineistojen systemaattisessa säilytyksessä tällainen ei ole mahdollista ja siksi on välttämätöntä muuttaa lainsäädäntöä siten, että se mahdollistaa emuloinnin kehittämisen samalla kun uusia laitteistoja tulee markkinoille lakia rikkomatta.

Tämän työn tarkoitus on omalta pieneltä osaltaan edistää tietoisuutta digitaalisen aineiston säilyttämisen mahdollisuudesta emulointia käyttäen. Kulttuuriperintömme on tilanteessa, jossa aineistoa tuotetaan valtavat määrät lisää päivittäin, mutta jossa puuttuu var-

muus siitä jääkö tuosta perinnöstä jälkipolville mitään perittäväksi. Asiaa on laiminlyöty samankaltaisten ajatuskulkujen johdattelemana, jotka johtivat esimerkiksi millennium-bugiksi kutsuttuun ongelmaan. Laiminlyönti aiheutti organisaatioille ja yksilöille huomattavasti suuremmat kustannukset kuin jos ongelmaan olisi reagoitu hyvissä ajoin. Jos digitaalisen aineiston pitkäaikaissäilytyksen ongelmakohdat jäävät ilman riittävää huomiota, tulevat kustannukset ja vahingot, niin aineelliset kuin aineettomatkin, olemaan suurempia kuin on koskaan nähty minkään toisen asiakirjoihin liittyvän asian suhteen ja niiden vaikutukset tulevat heijastumaan kaikkiin yhteiskunnan osa-alueisiin. Tämä antaa meille syyn toimia ennen kuin on liian myöhäistä. Myöhäistä on siinä vaiheessa, kun tietotaito vanhojen laitteistojen ja ohjelmistojen suhteen on kadonnut ja jäljellä on vain tallenteita, joita ei enää pystytä millään laitteistoilla lukemaan. Jos pystytäänkin, on aineisto tiedostomuodoissa, joiden tuki on ajat sitten poistunut tietotekniikan hektisiltä markkinoilta. On jo olemassa useita esimerkkejä, joissa tietoa on peruuttamattomasti menetetty, mutta suurin katastrofi tekee vasta tuloaan, johtuen siitä että olemme vasta siirtymässä kokonaan digitaaliseen aineistontuotantoon. Suunta on kuitenkin selkeä, ja muutaman vuoden sisällä kaikki suomalaiset muistiorganisaatiot ovat siirtyneet kokonaan digitaaliseen arkistonmuodostukseen.

2. ONGELMAN YDIN JA RATKAISUMALLIT.

Tässä luvussa perehdytän lukijan siihen, miksi digitaalinen aikakausi tuottaa pitkäaikaissäilytyksen kannalta ihan uudenlaisia ongelmia. Sellaisia, joita ei ollut olemassa silloin kun arkistotieteen perusteita laskettiin. Aliluvussa 2.1 keskityn digitaalisten ja perinteisten aineistojen välisiin eroihin. Aliluvussa 2.2 käyn läpi autenttisuuden käsitettä digitaalisessa maailmassa. Aliluvussa 2.3 selitän miksi tietokonemuseot ovat hyvä olla olemassa, mutta eivät sovellu pitkäaikaissäilytykseen. Aliluvussa 2.4 kerron uudistamistoiminnan välttämättömyydestä kaikissa säilytysmenetelmissä ja samalla tähdennän, ettei uudistamistoiminta vielä itsessään ole säilytysmenetelmä. Aliluvussa 2.5 tarkastelen tällä hetkellä suosituinta menetelmää migraatiota, ja sitä miksi se ei itsessään vielä riitä säilyttämään digitaalisia aineistoja. Aliluvussa 2.6 tarkastelen syitä siihen, miksi standardoimalla erilaisia tiedostomuotoja ei digitaalisen aineiston tulevaisuutta ole turvattu. Aliluvussa 2.7 esittelen emulointimenetelmää hyväksi käyttävän ratkaisun digitaalisen aineiston säilyttämisen ongelmaan.

2.1 Miksi digitaalisten aineistojen säilyttäminen on niin vaikeaa?

Digitaalisten aineistojen säilyttämisen ongelman ydin on siinä, että ne eivät ole suoraan ihmisen luettavissa ilman niiden lukemiseen suunniteltuja välineitä. Tämä keskeinen piirre erottaa digitaalisen aineiston perinteisistä aineistoista. Vaikka luolamaalaukset ovat tuhansia vuosia vanhoja tarvitsemme niiden tarkasteluun vain silmämme, mutta yhtäkään digitaalista tekstitiedostoa emme pysty lukemaan ilman sopivaa laitteistoa ja sopivaa ohjelmistoa. (Borghoff, Rödig, Scheffczyk & Schimtz 2006, 4.) Digitaalisen tekstin tarkastelemiseen tarvitaan laitteisto, jossa voidaan ajaa ohjelmisto, joka kykenee purkamaan tuossa aineistossa olevan koodauksen siten, että teksti saadaan näkyviin ihmiselle luettavissa olevassa muodossa. Kaikessa digitaalisessa aineistossa on mukana välttämättömänä elementtinä tietty skeema, tietomalli, jonka ehdoilla aineisto on jäsennelty. Jos skeemaa ei tunneta, on aineiston oikea lukeminen mahdotonta. Oikea tarkoittaa tässä sitä, että aineisto nähdään sellaisessa muodossa, jossa sen tekijä/tekijät halusivat sen näkyvän. Ilman skeemaa kaikki digitaaliset tiedostot ovat vain käsittämättömiä nollista ja ykkösistä koostuvia rivistöjä.

Aikaamme ylistetään teknisestä näppäryydestä ja käytössä olevasta tietotekniikasta, sillä se mahdollistaa sellaisten aineistojen tuottamisen, jollaisia ei aikaisemmin voinut kuvitella olevan olemassakaan. Se myös mahdollistaa aineiston tuottamisen vauhdilla, joka ei aikaisemmin olisi mitenkään ollut mahdollista. Asialla on kuitenkin kääntöpuolensa. Digitaalista aineistoa koskee kaksi olennaista kipukohtaa: sen luonne ja sen eksponentiaalisesti lisääntyvä määrä. Digitaalinen aineisto on perustavanlaatuiselta olemukseltaan hyvin yksinkertaista, se koostuu kahdesta erilaisesta sähköisestä tilasta. Käyttämällä näitä kahta sähköistä tilaa pystytään muodostamaan loputon määrä erilaisia bittijonoja, joista muodostuu lukemattomia erilaisia tiedostomuotoja ja standardeja. Kaikki digitaaliset aineistot ovat ajettavia ohjelmia siinä mielessä, ettei aineistoa voida tarkastella ilman ohjelmistoja, jotka kykenevät tulkitsemaan tuon aineiston sisällön ja kääntämään sen tietokoneen keskusyksikön ymmärtämiksi käskyiksi. (Rothenberg 2000, 14).

Kaikki tieto tietokoneissa on siis olemassa nollien ja ykkösien rivistöinä, niin kutsuttuina bittivirtoina. Ylemmällä tasolla niitä voi kutsua myös digitaalisten merkkien virroiksi, sillä kun nousemme nollien ja ykkösten tasosta ylöspäin, voimme käyttää kehittyneempää merkkijoukkoa (jotka nekin pohjimmiltaan koostuvat nollista ja ykkösistä) kuvaamaan aineistojamme (Borghoff ym. 2006, 5.)

Myöskin aineistojen vaatima tilamäärä on hyvin erilainen, muodostaen omalta osaltaan houkutuksia sen suhteen millaisen materiaalin säilyttämiseen kannattaa panostaa, vähiten tilaa vievien asettuessa helposti etusijalle. (Deegan & Tanner, 16.)

Diessenin mukaan digitaalisten aineistojen säilyttämistä tulee lähestyä vähintään kolmesta eri näkökulmasta. Ensimmäiseksi on otettava huomioon tallennusvälineiden arkistoiminen, toiseksi teknologian arkistoiminen ja kolmanneksi intellektuaalinen arkistoiminen. Tallennusvälineet menettävät ajan myötä niille valmistushetkellä luvattuja ominaisuuksia ja lopulta väistämättä tuhoutuvat. Siksi digitaalisia aineistoja on jatkuvasti uudistettava (refreshing) eli siirrettävä tallennusvälineeltä toiselle pakoon vanhentumisen vaikutuksia. (Diessen & Rijnsover 2002b, 1.)

Arkiajattelussa tätä pidetään yleensä digitaalisen aineiston suurimpana ongelmana, mutta lähemmässä tarkastelussa se osoittautuukin niistä pienimmäksi. Vaikka digitaalinen aineisto saataisiin säilymään siten, että sen bittivirta on varmassa tallessa, on aineisto

täysin arvotonta, jos ei ole enää olemassa laitteistoja, joiden avulla tuota aineistoa voidaan toistaa ja tarkastella. Laitteistojen ja ohjelmistojen jatkuva, kiihtyvä kehitys ja vanhan tekniikan nopea hylkääminen johtavat siihen, että vaikka aineistot olisivat täydellisesti arkistoidut, ei niitä pystytä hyödyntämään, jos tekniikka niiden ajamiseen puuttuu.

Intellektuaalinen arkistoiminen on näistä kolmesta kysymyksestä kaikkein vaikein. Digitaalisesta aineistosta pystyy tekemään kopioita todella helposti ja niitä pystyy muokkaamaan siten, että käsitys alkuperäisyydestä hämärtyy. Aineistoihin voidaan tehdä muutoksia siten, että jälkeenpäin on lähes mahdotonta tietää olivatko nämä muutokset läsnä jo alkuperäisessä aineistossa (Diessen & Werf-Davelaar 2002a, 2). Tämä on kysymys, johon ei teknologisilla keinoilla kyetä vastaamaan. Se on kysymys, joka informaatiotieteilijöiden on ratkaistava.

Kysymyksestä tekee hankalan se, että ennen digitaalisten aineistojen aikakautta kaikilla aineistoilla oli oma selkeä fyysinen olomuoto ja tuon fyysisen olomuodon yksityiskohdat määrittelivät aineiston autenttisuuden luotettavuuden. Kysymyksen pohdiskelu johtaa lopulta väistämättä siihen, että koko autenttisuudenkäsite on digitaalisten aineistojen kohdalla määriteltävä uudelleen, sillä aineistoja sisältävistä fyysisistä objekteista, kuten muistitikuista tai kiintolevyistä, ei pystytä päättelemään mitään itse aineistojen autenttisuudesta. Siksi Diessen ja Werf-Davelaar määrittelevät muutaman seikan, joilla digitaalisen aineiston autenttisuutta voidaan punnita. Heidän mukaansa aineistolla tulisi olla selkeä seurattavissa oleva elinkaari aineiston luomisesta sen tämän hetkiseen tilaan, sekä tekniikoita, joiden avulla voidaan havaita aineistoon mahdollisesti tehdyt muutokset ja pitää huolta aineiston koskemattomuudesta. Jos näistä on huolehdittu, ja niiden lisäksi aineisto palautetaan menetelmällä, joka täyttää pitkäaikaissäilytysorganisaation kriteerit aineiston alkuperäisyydestä, silloin voidaan sanoa aineiston olevan autenttinen. (Diessen & Werf-Davelaar 2002a, 1.)

Voidaan siis todeta että ensimmäinen ongelma digitaalisen aineiston kanssa on sen digitaalinen luonne. Digitaalisessa muodossa olevaa aineistoa ei voida säilyttää samoilla periaatteilla kuin perinteistä aineistoa. Kirjaston ei tarvitse varustaa kirjoja oheismateriaalilla, jota ilman kirja on käyttökelvoton. Mutta jos se laittaa hyllyyn kovalevyn täynnä tiedostoja, sen on pakko pakata tiedostojen oheen myös ohjelmisto, jolla nuo tiedostot voi ajaa, sekä vielä laitteiston, johon tuon kovalevyn voi liittää — koko ympäristön, jos-

sa kovalevyllä olevia tiedostoja pääsee tarkastelemaan. Muutoin heidän kovalevynsä on pikemmin suolaa haavoihin tiedonnälkäisille menneisyyden tutkijoille, joilla on suuria vaikeuksia saada mitään tolkkua kovalevylle tallennetuista tiedostoista. Rothenbergin mukaan tämä on rinnastettavissa siihen että laitetaan hyllyyn hieroglyfeillä kirjoitettu teos ilman Rosettan kiveä (Rothenberg 2000, 5). Vaikka sisältö on tarjolla, se on käytännössä täysin saavuttamattomissa.

Toinen suuri ongelma on digitaalisen aineiston eksponentiaalisesti kasvava määrä, jonka kasvuvauhti vain kiihtyy. Meillä ei ainoastaan ole luonteensa puolesta hankalaa aineistoa säilytettävänä vaan sen lisäksi sitä on moninkertainen määrä perinteisiin aineistoihin verrattuna, vaikka elämme vasta digitaalisen aikakauden alkupuolta. Maailmassa on tiedon tuotannon suhteen tapahtunut räjähdysmäinen kasvu. Vuonna 2002 maailmassa tuotettiin arviolta 5 eksatavua (5 miljardia gigatavua) tietoa eli noin 5,5 triljoonan kirjan verran tallennettuna paperille, filmille, magneettisille ja optisille tallenteille. (Lyman & Varian 2003, 1-112 tässä AMPAS 2007, 3.) Mutta koska digitaalista tietoa päästään tarkastelemaan ja käyttämään vain teknologian kautta, kun taas perinteisen tiedon käyttöön riittävät pelkät silmät, tuo tiedonräjähdys mukanaan myös vaaran suuren tietomäärän häviämisestä. Tämä johtuu siitä että digitaalisen tiedon käyttö maksaa. Mitä enemmän tuota tietoa on, sitä enemmän kustannuksia syntyy digitaalisen tiedon käyttämisestä. Käytännössä tämä lopulta nielee joko suuret määrät rahaa tai tietoa. Digitointiprojekteista voikin tulla tiedon mustia aukkoja, sillä organisaatiot saattavat rahapulassa jättää kalliit konvertointi- ja migrointitoimenpiteet tekemättä. (AMPAS 2007, 4.)

2.2 Digitaalisuus ja alkuperäisyys

Digitaalisen aineiston säilytystä käsittelevässä tutkimuskirjallisuudessa tulee usein vastaan käsite digitaalioriginaali. Sillä tarkoitetaan sitä alkuperäistä tiedostoa, jonka tiedoston luoja on koneelleen tallentanut. Tätä digitaalioriginaalia voimme pitää autenttisena aineistona, joka on varmasti sitä mitä se väittääkin olevansa. Mutta ongelmana on, kuinka pystymme tulevaisuudessa tarkastelemaan tuota aineistoa sellaisessa muodossa, että sen sisältö on todella identtinen sen kanssa, jonka aineiston luoja näki omalta näyttöpäätteeltään? Pitkäaikaissäilytysratkaisuja tarkasteltaessa on myönnettävä, että vaikka niissä kaikissa on vilpitön pyrkimys pitää kiinni digitaalioriginaalin esittämisestä sen al-

kuperäiseksi tarkoitetussa muodossaan, on tämän jalon tavoitteen saavuttaminen loppujen lopuksi mahdotonta. Tuosta tavoitteesta on tinkimättömästi pidettävä kiinni ja siihen on kaikella säilytystoiminnalla pyrittävä, mutta samalla on tiedostettava että sen toteutuminen täydellisenä on kaikkien ratkaisuvaihtoehtojen kohdalla mahdotonta. Tämän tosiasian hyväksymisen tulee olla kaiken digitaalisen aineiston pitkäaikaissäilytyksen ongelmia ratkovan toiminnan pohjalla. Vaikka emulaattorimme toimisi virheettömästi, ei emuloinnin kautta välity kaikkea, mikä välittyi niille, jotka tarkastelivat aineistoa alkuperäislaitteiston kautta.

Samoin on laita lähestymistavan, jossa pyrkimyksenä on laatia emulaatiospesifikaatiodokumentti jokaiselle maailmassa koskaan tuotetulle tietokonekomponentille. Se on yhtä lailla tuhoon tuomittu hanke kuin kirjastotieteen perustajahahmojen yritys koota kaikki maailman tieto yksien kansien sisälle. Laajuudessaan tehtävä on mahdoton, jos ei itse laajuutensa tähden niin resurssien puuttumisen tähden, ainakin jos lähtökohtana on komponenttispesifikaatioiden laatiminen kullekin komponentille jälkeenpäin. Asia kuitenkin muuttuisi jos kansallinen lainsäädäntö velvoittaisi komponenttivalmistajia toimittamaan spesifikaatiot valmistamistaan komponenteista digitaalisen aineiston pitkäaikaissäilytyksestä huolehtiville organisaatioille. Organisaatioilla olisi ehdoton velvollisuus pitää nämä spesifikaatiodokumentit vain emulointikehityksen käytössä ja niiden tarkastelua jopa tuon organisaation sisällä valvottaisiin hyvin tarkasti. Ehdottoman varmasti pidettäisiin huolta siitä etteivät spesifikaatiodokumentit pääse arkistosta leviämään ulkomaailmaan ja arkiston asiakas, emuloidun aineiston käyttäjä, näkisi vain emuloinnin lopputuloksen. Hän ei näkisi spesifikaatioita komponenteista, jotka muodostavat yhdessä emuloitavan laitteiston. Arkiston asiakkaalle voitaisiin luovuttaa Luovutustietopaketti[1], joka sisältää sekä aineiston että emulaattorin, mutta ei niitä määrittelydokumentteja joiden pohjalta tuo emulaattori on asiakkaan tarpeisiin generoitu. Komponenttimäärittelyjä säilyttämään ja niiden käyttöä hallinnoimaan perustettaisiin maailmanlaajuinen järjestelmä, joka jossain määrin muistuttaisi tekijänoikeusjärjestöjen kansainvälistä toimintaa. Tätä kaikkea varten voitaisiin EU:ssa perustaa ihan oma virastonsa, elintarviketutkimuslaboratorioiden yms. mallin mukaisesti.

1Dissemination Information Packet

Kaikki hankkeet, joiden on ollut tarkoitus tutkia emulointia säilytysmenetelmänä, ovat tähän mennessä olleet lähinnä tiedustelu- ja tunnusteluprojekteja. Ne ovat kyllä tuottaneet korvaamatonta tietoa, mutta eivät ole vielä saaneet isoja koneistoja toimimaan emuloinnin hyväksi. Digitaalisen aineiston pitkäaikaissäilytyksen ongelmien laajuutta ajatellen myös niihin käytetyt voimavarat, niin aineelliset kuin aineettomatkin, ovat olleet riittämättömiä. Tämä riittämättömyys näkyy siinäkin, että Kansallisen Digitaalisen Kirjaston (KDK) raporteissa emulointi on sivuosassa ja vain kahdesti mainittuna. Saattaa peräti olla, että yleiset mielikuvat emulaattoreista vanhojen tietokonepelien moottoreina, jonkinlaisina leluina, heijastelee KDK:n päättäjiin saakka. Tekniikkaa saatetaan pitää myös epävarmana ja epäluotettavana, mutta tällaisen käsityksen takana on tietämättömyys emuloinnin nykytilanteesta. Markkinoilla on nyt useita toimivia ja luotettavia emulaattoreita, joilla on suuri käyttäjäkunta.

2.3 Ratkaisu teknologian säilyttämisestä?

Kaikkien laitteistojen säilyttäminen alkuperäisessä muodossaan kaukaiseen tulevaisuuteen on ainut säilytysvaihtoehto, josta vallitsee lähes täydellinen konsensus ongelman kanssa työskentelevien parissa. Se on niin ilmiselvästi mahdotonta toteuttaa, ettei teknologian systemaattista säilyttämistä harjoiteta tiettävästi missään päin maailmaa. Yksittäisiä tietokonemuseoita toki on olemassa, mutta ei ainuttakaan organisaatiota, joka olisi asettanut tehtäväkseen säilyttää kaiken mahdollisen tietotekniikan. Vaikka syyt ovat ilmeiset, on ne silti syytä käydä läpi.

Teknologian säilyttäminen digitaalisten aineistojen turvaamiseksi on mahdotonta sillä 1) Laitteistoja on aivan liikaa, jotta ne mahtuisivat mihinkään kuviteltavissa olevaan tilaan. 2) Laitteistojen oheislaitteita on niin paljon, ettei ihminen kykene rakentamaan tarpeeksi isoa rakennusta pitämään niitä kaikkia sisällään. 3) Laitteistoja olisi erittäin vaikea huoltaa. Varaosia ei olisi saatavilla ja käytännössä organisaatiolla tulisi olla laitteistojen lisäksi niiden täydelliset piirustukset ja tietotaito valmistaa vanhaa elektroniikkaa. (Borghoff ym. 2006, 14-16; Deegan & Tanner 2006, 17.) 4) Koska tieto ei säily medioilla loputtomiin, olisi aineistot uudistettava säännöllisin väliajoin uusille tallennusvälineille.

Mutta koska vanhoille tietokoneille ei ole olemassa niiden kanssa yhteensopivia laitteita uusien tallennusvälineiden lukemiseen, pitäisi sellaiset rakentaa ja ohjelmoida niille ajurit, joiden avulla vanha laitteisto kykenisi kommunikoimaan näiden uusien laitteiden kanssa. (Rothenberg 1999, 12-13.)

Menetelmä voi toimia väliaikaisena säilytysmenetelmänä, jos parempaa ei ole, mutta pysyväissäilytykseen siitä ei ole sillä tietokoneiden elektroniikan elinikä on lyhyt, jopa ihanneolosuhteissa huomattavasti aikaisemmin oletettua lyhyempi. Elektroniikka muuttuu pikkuhiljaa toimintakyvyttömäksi pidemmän ajan kanssa tapahtuvien fysikaalisten prosessien seurauksena. Uusien täysien samanlaisten valmistaminen ei ole suoranainen teknologinen mahdottomuus, mutta käytännössä yhteiskunnan resurssit ovat riittämättömät tuottamaan niitä. Ongelmana myös on, että koska itse digitaalisista entiteeteistä joudutaan joka tapauksessa ottamaan kopioita uudemmille medioille, on ratkaistava kysymys siitä, kuinka nämä uusille medioille säilötyt tiedostot kyetään siirtämään vanhoihin koneisiin. Tämä vaatisi sellaista tietotaitoa, joka on todennäköisesti sen tarvitsemishetkellä jo ajat sitten maailmasta kokonaan hävinnyt. Tästä vallitsee ainakin tällä hetkellä konsensus asiaa tutkivien joukossa, sillä kukaan ei ole vielä vakavissaan ehdottanut tietokonemuseoita säilytysmenetelmäksi. (Deegan & Tanner, 17; Rothenberg 2000.)

Kuitenkin Rauber (2004, 20) on sitä mieltä, että tietokonemuseot ovat jossain määrin välttämättömyys. Onhan mahdotonta verrata emuloidun aineiston ja alkuperäisessä laitteistossa ajetun aineiston vastaavuutta ilman alkuperäistä laitteistoa. Jossakin määrin tietokoneiden säilyttämistä on pakkokin harjoittaa. Pelkät ohjelmistot eivät yksinomaan riitä antamaan kokonaiskuvaa ajastamme tulevaisuuden ihmisille.

Suomessa tietokonemuseolähestymistapa on viety pisimmälle Jyväskylässä, jossa kanavuoren uumenissa Puolustusvoimien varikkoluolassa on tallessa Suomen ensimmäisiä tietokoneita, mm. MIR-2, jota Teknillisessä korkeakoulussa käytettiin 1970-luvulla ja Saksalainen ZuseZ23, jolla laskettiin metrolinjojen ratoja 1960-luvulla. Näillä laitteistoilla on saatu konkreettisesti todeta miten korvaamatonta tietoa esim. reikäkorteilla on edelleen, joita ei millään nykyisillä laitteilla pystytä lukemaan. (HS 2001.) Juha Hakalan (2001) mukaan vanhojen tietokoneiden toimintakykyisenä pitämisen raja kulkee 20 vuoden kohdalla.

Borghoffin, Rödigin, Scheffczykin ja Schmitzin (2006, 16) mukaan vaarana on myös se, ettei tulevaisuudessa enää ymmärretä miten meidän aikanamme valmistettiin tietokoneet teollisten prosessien muuttuessa aikojen kuluessa niin radikaalisti. Emmehän mekään enää täysin pysty jäljentämään kaikkia muinaisia valmistusmenetelmiä, emmekä sen vuoksi pysty tuottamaan ihan samanlaisia metalliesineitä ja vaatteita kuin kauan aikaa sitten.

2.4 Ratkaisu uudistamisesta?

Uudistamistoiminta on välttämättömänä elementtinä läsnä jokaisessa säilytysmenetelmässä sillä kaikkia digitaalisen aineiston tallennusmedioita vaivaa lyhytikäisyyden ongelma (Deegan & Tanner 2006, 18). Aineistoa on siis siirrettävä tallennusvälineeltä toiselle tietyllä aikataululla käytetyistä tallennusmedioista riippuen. Jos vain aineiston tarkasta verifioimisesta pidetään huolta, on uudistaminen silloin turvalliseksi katsottua toimintaa, jossa on vain hyvin pieni riski että tietoja kadotettaisiin. Uudistaminen ei yksistään takaa aineiston säilymistä käyttökelpoisena, mutta ilman sitä ei säilytysjärjestelmä voi toimia kun kyse on digitaalisesta aineistosta.

Research Libraries Groupin ja Comission for Preservation and Accessin mukaan uudistaminen on laajempi ja rikkaampi käsite kuin migraatio. Siinä missä uudistaminen tarkoittaa vain tiedon mekaanista siirtämistä vanhemmalta tallenteelta uudemmalle, migraatio merkitsee tämän lisäksi myös sitä että säilytysprosessi koostuu toimintojen joukosta, jolla pidetään huolta siitä, että loppukäyttäjällä on säilytysketjun toisessa päässä käytössään käyttökelpoinen aineisto. (Commission on Preservation and Access 1996, tässä Russell 1999.)

Uudistaminen ei siis itsessään ratkaise digitaalisen aineiston pitkäaikaissäilytyksen ongelmia millään lailla, mutta uudistamista on pakko jokaisen pitkäaikaissäilytysorganisaation harjoittaa. On kuitenkin tähdennettävä, ettei se itsessään ole säilytysmenetelmä, sillä tällaiseen ajatteluun törmää melko usein, eivätkä ihmiset tavallisesti pyri ajattele-

maan muutamaa vuotta pidemmälle. Siinä ajassa ei ongelmia välttämättä vielä esiinny vaikka pelkkää uudistamista harjoitettaisiinkin.

2.5 Ratkaisu migraatiosta?

Tämä on tällä hetkellä yleisimmässä käytössä oleva säilytysmenetelmä. Sen ovat suomessa ottaneet käyttöön kaikki julkishallinnon tahot paremman ratkaisun vielä toistaiseksi puuttuessa. Siinä aineisto muutetaan toiseen muotoon siten ettei tietohäviötä pitäisi syntyä. Migraatioon (käytetään myös termiä konvertointi) liittyy samankaltaista käsitediskurssia kuin emulointiin. Tutkijat eivät ole yksimielisiä siitä, mitkä ovat ne yhteiset tekijät jotka erottavat migraation ja uudistamisen toisistaan. Hedström (1998) luokittelee kopioinnin, migraation ja uudistamisen kaikki migraatio-nimikkeen alle siten, että uudistaminen ja kopioiminen ovat osa migraatiota. Pitkäaikaissäilytys on vaiheessa, jossa se hakee muotoaan. Se on havaittavissa tutkimuskirjallisuudessa termien rajojen häilyvyytenä, määritelminä jotka ovat eri aineistojen kesken ristiriitaisia. Toisinaan taas käsitteet esitetään utuisina ja monitulkintaisina.

Borghoff ym. (2006, 14) huomauttaa, että koska migraatio tiedostomuodoista toiseen on niin yleinen tietotekniikassa käytetty tekniikka, voidaan ainakin olla varmoja siitä, että migraation takana on vankka tietotaito IT-ammattilaisten keskuudessa. Mutta Borghoff ym. myöntävät, että jokainen migraatiosukupolvi tuottaa muutoksia aineistoon ja nämä muutokset kasaantuvat siten, että voidaan epäillä niiden lopulta uhkaavan aineiston autenttisuutta. Lisäksi Borghoff ym. toteavat migraation osoittautuneen vaikeasti automatisoitavaksi. Vain yksinkertaisimmat migraatiotoimenpiteet voidaan kokonaan hoitaa automatiikalla. Mutta niissäkin tulokset on tarkistettava käsin siltä varalta, että automaattisessa migraatiossa on mennyt jotakin pieleen. Migraatio on monien aineistojen kohdalla monivaiheinen prosessi, varsinkin multimedia-aineistojen, jotka koostuvat useista eri tiedostomuodoista. (Borghoff et al. 2006, 14)

Bearmanin (1999, 2) mukaan digitaalisten aineistojen systemaattinen migrointi on ylenkatsottu alue, jota tulisi tutkia enemmän. Bearmanin mielestä migraatiossa ei myöskään välttämättä menetetä mitään, jos vain migraatiota kehitetään tarpeeksi pitkälle. Ongelmat migraation suhteen johtuvatkin hänen mukaansa enemmän siitä, ettei sen kehittämi-

seen ole suunnattu riittävästi voimavaroja, eivätkä siitä, että se olisi säilytysmenetelmänä soveltamiskelvoton. (Bearman 1999, 2.)

Hieman samoilla linjoilla autenttisuuden suhteen on Rusbridge (2006, 4-5) antaessaan esimerkkinä autenttisuuden muuttumattomuudesta migroinnissa tapahtuneista muutoksista huolimatta sir Walter Scottin kirjoittaman "Keniworth"-teoksen. Alkuperäinen oli ladottu painokoneeseen käsinkirjoitetusta käsikirjoituksesta. Lopputuloksena oli hieno nahkakantinen artefakti, jolla on suuri arvo Scottin perikunnalle ja historian tutkijoille, mutta jonka lukeminen muille on tuskaista, sillä se on täynnä painovirheitä. Teoksesta on kuitenkin otettu uusia painoksia, joissa painovirheet on korjattu ja joissa teoksen ulkoasu on täysin erilainen kuin mitä alkuperäisteoksen. Silti emme aseta tämän uusintapainoksen autenttisuutta kyseenalaiseksi. Suurelle yleisölle teoksen tarina on sen tärkein pääoma. Tutkijalle alkuperäinen ulkonäköineen ja painovirheineen on oleellinen. Rusbridge yrittää siis sanoa etteivät akateemikot voi määritellä kaikkien puolesta sitä, mikä on riittävän alkuperäistä. (Rusbridge 2006, 4-5.)

Russellin mukaan liiallinen luottamus ja panostus migraatioon saattaa koitua organisaatioille lopulta kalliimmaksi kuin aloitushinnaltaan kalliimmat ratkaisut kuten emulointi (Russell 1999). Migraation näennäinen helppous ja halpuus houkuttelevat organisaatioita panostamaan siihen. Se myös tuottaa välittömiä tuloksia, jotka organisaation työntekijät kokevat tyydyttäviksi. Tämä saattaa houkutella ajattelemaan, että ongelma on ratkaistu ja vasta tulevaisuudessa he havahtuvat siihen, etteivät enää kykene avaamaan vanhoja aineistoja. Silloin on hyvin todennäköisesti enää myöhäistä puuttua ongelmaan. On nimittäin muistettava, että migraatio on monien aineistojen kohdalla monivaiheinen prosessi, varsinkin multimedia-aineistojen, jotka koostuvat useista toisiinsa linkitetyistä tiedostomuodoista. Pelkkä aineiston migroiminen, varsinkin jos mitään muita toimenpiteitä ei tehdä, voi johtaa korruptoitumisen jatkuvaan lisääntymiseen aineistossa. Se mikä tuottaa välittömästi konkreettisen tuloksen ja tyydyttää hetken tarpeen, voi pidemmällä tähtäimellä jopa haitata itse pitkäaikaissäilytystä. Jos aineistoon kohdistetaan muita säilytystoimenpiteitä vasta useamman migraatiosukupolven jälkeen, saattaa korruptoituneisuus olla jo niin pitkällä, että tulee jo vastaan kysymys siitä onko se enää autenttista aineistoa laisinkaan. Vaikka ratkaisu löytyisikin myöhemmin suurella rahalla, ovat muistiorganisaatiot ja suurin osa yksittäisistä henkilöistä niin varattomia, että käytännössä he

hylkäävät aineiston, jota eivät pysty enää käyttämään ja jonka palauttamiseen heillä ei ole varaa. Näin tullaan väistämättä menettämään jotakin korvaamatonta, jonka arvo havaitaan vasta menetyksen jälkeen.

Migraatiossa aineistojen koskemattomuuden kannalta on suuria ongelmia, sillä migroitujen tiedostojen käyttö johtaa väistämättä loputtomaan spekulointiin siitä, millainen alkuperäinen asiakirja todella oli, ja näin migroidun aineiston todistusvoima voidaan aina asettaa kyseenalaiseksi vetoamalla mahdollisesti migroinnissa tapahtuneisiin tietojen katoamisiin tai vääristymisiin. Tämä on digitaalisen asiakirjan kannalta todella vakava asia, varsinkin jos alkuperäiset on kokonaan hävitetty. On luultavaa, että monessa organisaatiossa migroinnin jälkeen tarpeettomaksi jäänyt alkuperäinen tiedosto lopulta hävitetään, sillä tarpeetonta päällekkäisyyttä on vältettävä kaikissa organisaatioissa sekaannuksien välttämiseksi ja tilan säästämiseksi.

Rothenbergin (1999, 13-14) näkemykset migraatiosta ovat jyrkimpiä. Hän näkee migraation menetelmänä, jonka organisaatiot ovat ottaneet käyttöön lähinnä siksi, ettei sille ole ollut tarjolla todellista vaihtoehtoa, jota olisi voinut alkaa käytännössä harjoittamaan. Rothenbergin mukaan aineistojen pakottaminen uusiin tiedostomuotoihin kussakin migraatiosukupolvessa johtaa väistämättä aineistojen korruptoitumiseen ja tietohäviöihin. Asian tekee ongelmallisemmaksi vielä se, että jokainen uusi tiedostomuoto vaatii uuden, omanlaisensa ratkaisun kehittämisen vanhojen aineistojen muuntamiseen. Aiemmista migraatiokierroksista ei välttämättä hyödytä kovinkaan paljon, koska jokainen migraatiokierros tuo mukanaan omat, uudenlaiset ongelmansa. Näin jokainen migraatiokierros on Rothenbergin mukaan yhtä virhealtis, ongelmallinen ja kallis. (Rothenberg 1999, 13-14.)

Tämän lisäksi Rothenberg muistuttaa, että migraatiota ajaa eteenpäin jatkuva, hellittämätön pakottava kiire, sillä tiedostomuodot muuttuvat yllättäen ja vaihtelevilla nopeuksilla. Vaikka aineisto olisi huolellisesti tallennettu, ei siitä ole apua jos muuntoa uuteen tiedostoformaattiin ei suoriteta riittävän ajoissa. On mahdollista, että aineiston lukemiseen tarvittava ohjelmisto ja tuon ohjelmiston ajamiseen tarvittavat laitteistot ehtivät vanhentumaan käyttökelvottomiksi siten, että aineiston käyttömahdollisuudet ovat peruuttamattomasti menetetyt. Eikä tämä vaara ole kertaluontainen, vaan tulee aina uudestaan eteen jokaisen migraatiokierroksen lähestyessä. Rothenberg myös huomauttaa, että

hyvin suuri osa yksittäisistäkin aineistoista koostuu yhä lisääntyvässä määrin useiden eri tiedostomuotojen muodostamista kokonaisuuksista. Näin on todennäköistä, että jotkin tiedostot aineistokokonaisuudessa vaativat migroimista aikaisemmin kuin toiset. Migraatioprosessista tulee siis hyvin monimutkainen sillä aineistoon tulisi kohdistaa migraatioita hyvin erilaisilla aikatauluilla. (Rothenberg 1999, 14-15.)

Tästä seuraa johdonmukaisesti vaara, että osa aineistokokonaisuuden tiedostoista putoaa migraation ulkopuolelle tai tulee migroitua väärään aikaan. Vaaraksi muodostuu siis se, että osa aineistokokonaisuuksista menetetään pysyvästi niiden mahdollisesti muuttuessa toimimattomiksi vaihtelevien migraatioaaltojen kohdistuessa niihin. (Rothenberg 1999, 15-16.)

Rothenberg muistuttaa myös, että migraatio hyötyy vain vähän ajan kanssa lisääntyvästä laskentatehosta ja muusta teknisestä kehityksestä. Laskentatehon kasvu voi migraation tapauksessa jopa kääntyä pitkäaikaissäilytystä vastaan, koska laskentatehon kasvu tehostaa migraatiota siten, että automaattinen migraatio voi tuhota organisaation tärkeimmän pääoman salakavalasti aiheuttamalla korruptoitumista aineistoihin. (Rothenberg 1999, 15.)

Rothenberg tiivistää väitteensä esittämällä, että migraatio tarjoaa vain illuusion ratkaisusta, mutta ei todellista ratkaisua. Koska yksi migraation keskeisistä peruselementeistä on Rothenbergin mukaan ennalta-arvaamattomuus, perustuu migraatio toiveajatteluun, jossa toivo asetetaan tietotekniikka-asiantuntijoiden kykyyn ratkaista tulevaisuudessa eteen tulevat ongelmat. (Rothenberg 1999, 15.)

2.6 Ratkaisu standardoiduista tiedostomuodoista?

Eri ohjelmistovalmistajien tuotteet käyttävät erilaisia rakenteita aineistojen tallentamisessa ja näyttämisessä. Näitä erilaisia tietorakenteita kutsutaan tiedostomuodoiksi (Borghoff ym. 2006, 5). Monet arkistonhoitajat ajattelevat tällä hetkellä tietyn, yhteiseksi sovitun tiedostomuodon olevan selkeä ja mutkattomin ratkaisu digitaalisen aineiston pitkäaikaissäilytyksen ongelmaan. Kaikkein suosituin näistä tiedostomuodoista vaikuttaa olevan XML. XML ei kuitenkaan sovellu kaikkien lukuisien erilaisten aineistomuotojen säilyttämiseen (Asproth 2005).

Nykyään on muodostunut käytännöksi, että ohjelmistot varustetaan tuontisuodattimilla[2]. Tuontisuodattimien avulla eri ohjelmistojen tiedostomuodot ovat vaihtokelpoisia keskenään. Tuontisuodatin toimii siten, että se jättää vierasta tiedostomuotoa ohjelmistoon tuotaessa pois ne merkit, joita toinen ohjelmisto ei ymmärrä tai jotka se mahdollisesti tulkitsisi väärin (Jaakkohuhta 2003, 255).

Yleisessä käytössä ovat myös siirtosuodattimet[3], joiden avulla harvinaisemmissa sovelluksissa luodut aineistot saadaan muutettua yleisessä käytössä olevien sovellusten käyttämiin tiedostomuotoihin (Jaakkohuhta 2003, 191). Borghoffin ym. (2006, 6) mukaan tiedostomuotojen suuri määrä antaa viitteitä siihen suuntaan, että olisi mielekästä laatia tiedostomuotostandardeja, joiden avulla kaikissa sovelluksissa laadittu aineisto olisi vaihtokelpoista keskenään.

Rothenberg (1999, 10) taasen tyrmää idean yhteisistä standardeista pitkäaikaissäilytyksen perustana täysin. Hänen mukaansa yhteiset standardit eivät takaa aineistojen säilyvyyttä, sillä ohjelmistojen valmistajat haluavat varmistaa oman markkina-asemansa laajentamalla standardia lisäten siihen ylimääräisiä ominaisuuksia. Koska standardi ei tue näitä ylimääräisiä ominaisuuksia, joita ohjelmistovalmistajien on pakko kehittää erottuakseen markkinoilla, eivät näitä erikoisominaisuuksia käyttävät aineistot tule säilymään, kun aineisto muutetaan yhteisstandardimuotoon.

---

2 import filter

3 export filter

Meille ovat tuttuja sovellusten ilmoitukset siitä, että "tässä muodossa tallennettaessa dokumentin kaikki ominaisuudet eivät välttämättä näy oikein". Tiedämme kukin omasta kokemuksestamme, että toisessa sovelluksessa laadittu aineisto ei välttämättä näy oikein toisessa tai jotain jää uupumaan siitä. Rothenberg ottaa esille Coddin 1982 esittämän väitteen siitä, että minkä tahansa alun perin RDB-muodossa olevan tietokannan voi muuttaa häviöttömästi RDBS-muotoon. Väite perustuu siihen, että kaikki relaatiotietokannat perustuvat samaan perustason toiminnallisuuteen. Vaikka tämä pitääkin paikkansa, Rothenberg huomauttaa, että RDBMS on erittäin harvinainen poikkeus standardien joukossa pysyväisyydessään muiden standardien jatkaessa muutosprosessiaan. Lisäksi juuri se, että standardien tehtävänä on rajoittaa ilmaisunvapautta aineistoissa saa väistämättä aikaan sen, että ohjelmistovalmistajat lähtevät kehittämään niitä eteenpäin pystyäkseen tarjoamaan ohjelmistojensa kautta asiakkaille ominaisuuksia, joita muiden ohjelmistoissa ei ole. Rothenbergin mukaan tällaisen aineiston pakottaminen standardoituun formaattiin väistämättä hävittää jotakin siten, että lopputulos ei ole uutta formaattia, mutta ei vanhaakaan, vaan jonkinlainen välimuoto näiden kahden väliltä. (Rothenberg 1999, 10-11.)

Vielä Rothenberg kommentoi standardiratkaisun puolestapuhujien ehdotusta siitä, että kaikkien aikaisempien tiedostomuotojen muuntaminen uudempien standardien mukaisiin tiedostomuotoihin tehtäisiin pakolliseksi. Rothenbergin mukaan tämä on rinnasteista sille, että muinaisenglannilla kirjoitettu teos pakotetaan jokaisen muinais- ja nykyenglannin välissä olleen kielen lävitse. Näinhän kääntäjät eivät koskaan tee, vaan eri käännökset tehdään aina alkuteoksen pohjalta. (Rothenberg 1999, 10-11.) Rothenbergin mukaan on yksiselitteistä, että standardien varaan ei digitaalisen tiedoston pitkäaikaissäilytystä voi laskea. Hänen mielestään oiva osoitus tästä on merkistökoodauksien muuttuminen. ASCII-standardi on saanut hiljalleen väistyä uudemman Unicoden tieltä. (Rothenberg 1999, 11.)

2.7 Ratkaisu emuloinnista?

Emulointiin panostaminen vaatii organisaatiolta työtä, jonka hedelmät näkyvät todennäköisesti vasta pidemmän ajan päästä. Näin organisaation panostaminen emulaatioon vaatii tietynlaista optimistista asennetta ja riskinottoa, joka toisaalta, pieleen mennessään voi aiheuttaa organisaatioille taloudellisia tappioita, joista nämä eivät kykene toipumaan. Siksi on hyvin ymmärrettävää, että useat organisaatiot saattavat kokevat emulaation liian suureksi riskiksi ja panostavat uudistamiseen ja migraatioon, joista he saavat välittömän hyödyn. Voidaan kuitenkin esittää epäilys siitä, että tämä lyhytnäköinen toiminta saattaa kostautua organisaatioille ja maksaa lopulta enemmän kuin panostaminen vaihtoehtoiseen ratkaisuun (kuten emulointiin), heti alusta saakka.

Emulointiratkaisun aloituskustannukset ovat huomattavasti suuremmat kuin muiden ratkaisumallien ja siksi sen markkinointi organisaatioille ja julkishallinnolle on vaikeaa. On kuitenkin mielekkäämpää panostaa hankalasti aloitettavaan ratkaisuun, joka tuottaa toimivan lopputuloksen aineistojen eheyden ja autenttisuuden siitä kärsimättä, kuin mennä siitä mistä aita on matalampi ja havahtua tulevaisuudessa ylitsepääsemättömiin ongelmiin.

3. TUTKIMUSKYSYMYKSET

Tämän työn tarkoitus on kartoittaa emuloinnin vahvuuksia ja heikkouksia ja ottaa selvää sen soveltamiskelpoisuudesta pitkäaikaissäilytysorganisaatioiden säilytysohjelmiin. Hypoteesini on, että emulointi on ainut todellinen vastaus digitaalisten aineistojen mahdollisimman autenttiseen esittämiseen tulevaisuudessa ja muut ratkaisut tarjoavat jotain, mikä on parempi kuin ei mitään ratkaisua ollenkaan, mutta eivät kuitenkaan täytä niitä vaatimuksia, joita tulevaisuudessa tullaan olettamaan että säilytysratkaisumme on täyttänyt.

Aineistojeni ja niistä havaitsemieni puutteiden perusteella olen muotoillut seuraavat tutkimuskysymykset:

-Mitkä ovat emuloinnin vahvuudet pitkäaikaissäilytyksessä?

-Millainen on emuloinnin asema pitkäaikaissäilytysprojekteissa?

-Mitä voidaan päätellä emuloinnin suhteen suoritetuista tapaustutkimuksista?

-Onko emuloinnilla olennaista roolia tietokoneteollisuudessa?

-Ovatko emuloinnin ja virtualisoinnin käsitteet erotettavissa toisistaan?

-Asettavatko laillisuuskysymykset ylivoimaisia esteitä emuloinnin käytölle pitkäaikaissäilytyksessä?

-Johtuuko emulointiratkaisujen kehittämisen pysähtyneisyys arkistoalalla siitä ettei emuloinnin kehitykseen kohdisteta tarpeeksi resursseja?

4. TUTKIMUSMENETELMÄ JA AINEISTON ESITTELY

Tässä luvussa ja sen aliluvuissa esittelen käytetyn tutkimusmenetelmän ja tutustutan lukijan aineiston alkuperään.

4.1 Tutkimusmenetelmä

Tutkimus on kirjallisuuteen perustuva katsaus emuloinnista. Lisäksi aineistoa on täydennetty emulointikokeilla. Käsittääkseni juuri tällaista tutkimusmenetelmienkombinaatiota ei ole aiemmin käytetty.

Tieteellinen tieto tunnistetaan ominaisuuksistaan. Näitä ominaisuuksia ovat tieteellisen tiedon julkisuus ja avoimuus kaikille. Tieteellisen tiedon tulee olla kaikkien arvioitavissa ja siitä tulee käydä yksiselitteisesti ilmi se, mistä tiedot on saatu ja mikä on kirjoittajan omaa ja mikä lähteistä peräisin olevaa. Kirjallisuuskatsauksen tarkoituksena on vastata tutkimuskysymykseen rajattua lähdemateriaalia käyttämällä. Kirjallisuuskatsauksen laatimisen motivaationa on se, että se tarjoaisi uusien tutkimuksien tekijöille ja tutkimusalan ammattilaisille tutkimustietoa koosteena ilman, että heidän tarvitsee kajota alkuperäislähteisiin. Tämä säästää heiltä paljon aikaa ja antaa siten arvokkaan panoksen tieteelliseen keskusteluun. (Leino-Kilpi 2007, 2.)

Näiden tavoitteiden täyttämisen lisäksi olen halunnut yhdistää kirjallisuuskatsaukseen Hirsjärven, Remeksen ja Sajavaaran (2009, 260-263) mainitseman 'kertomuksen etsimisestä'. Tarkoitukseni on johdattaa lukija esiin tuomieni pohjatietojen lävitse kohti itse tekemiäni havaintoja.

Kirjallisuuskatsauksen voi tehdä vain jos tutkittavalta alueelta löytyy tarpeeksi tutkimustietoa (Leino-Kilpi 2007, 2). Emuloinnin osalta tämä ehto täyttyi moninkertaisesti sillä tutkimustietoa löytyi niin paljon, että sen rajaamisessa oli suuria ongelmia. Se myös aiheutti ongelmia sen suhteen, mitä tutkimuksia valitsin mukaan katsaukseeni. Jos nimittäin tutkimuksien tulokset ovat kovin samankaltaisia, silloin niiden pohjalta on hy-

vin vaikea tehdä uutta tietoa luovia johtopäätöksiä (Axelin, Johansson & Ääri 2007, 92). Siksi otin mukaan myös emulointiin kriittisesti suhtautuvien tutkijoiden kirjoituksia.

Kirjallisuuskatsauksia on olemassa useita eri lajeja ja jokaisen katsauksen tekijän on valittava hänen katsaukseensa parhaiten sopiva laji. Yritin muotoilla katsaukseni sellaiseksi, että se sopisi rakenteeltaan parhaiten aineistoni ja aiheeni luonteeseen. Näin sain monimuotoisen tutkimuskirjallisuuteni mahdutettua saman katsauksen alle. Kirjallisuuskatsauksessani tarkoituksena on, että tiedon kertymisestä muodostuu kumulatiivista ja eri tutkimuksien jättämistä palasista saadaan muodostettua ehyt kokonaiskuva. (Flinkman & Salanterä 2007, 84.)

Kirjallisuuskatsauksen tehtävänä on tutkimustiedon kerääminen yhteen siten, että yhteen kootun tiedon perusteella voidaan tehdä johtopäätöksiä. Yhdistellään tutkimuksia joiden pohjalla on samansuuntainen kysymyksenasettelu ja tehdään niistä yhteenvetoa. (Polit & Beck 2004, 721 tässä Flinkman & Salanterä 2007, 85.) Emuloinnin havaitsin siinä mielessä mielenkiintoiseksi aiheeksi, että se näyttää kuljettavan mukanaan tiettyjä tutkimuskysymyksiä tehden siten helpoksi samansuuntaisia tutkimuskysymyksiä asettavien tutkimuksien löytymisen.

Kirjallisuuskatsaus voi täyttää useita erilaisia tehtäviä paljastamalla aukkoja aikaisemmista tutkimuksista. Näistä aukoista voidaan löytää lisätutkimuksien aiheita. Aukot voivat myös paljastaa puutteita aiemmin tehdyissä tutkimuksissa. (Russell 2005, tässä Flinkman & Salanterä 2007, 86.)

Katsausta laadittaessa tulisi ensin päättää tutkimuskysymykset ja vasta sitten aloittaa suunnittelemaan aineiston keruuta. Tähän ohjeeseen nähden menettelin tässä työssä joidenkin tutkimuskysymysten osalta nurinkurisessa järjestyksessä. Aineiston keruun minua kiinnostavasta aiheesta olin aloittanut jo vuosia aikaisemmin. Prosessia ohjasi tuolloin alustava tutkimuskysymys "onko emulaatio käyttökelpoinen säilytysmenetelmä digitaalisen aineiston pitkäaikaissäilytykseen"'. Myöhemmin tämä kysymys sai seminaarissa saadun palautteen myötä muodokseen "mitkä ovat emuloinnin vahvuudet pitkäaikaissäilytyksessä?".

Kaiken kaikkiaan katsauksen teossa oli yhteensä 5 eri vaihetta: tutkimusongelman muotoilu, aineiston kerääminen, aineiston arviointi, aineiston analyysi, aineiston tulkinta ja tulosten esittäminen (Cooper 1989, 15 tässä Flinkman & Salanterä 2007, 88).

Tutkimuskysymysten selkeä asettelu asettaa katsaukselle rajat ja tarjoaa sille selkeän suunnan. Ilman huolellisesti aseteltuja tutkimuskysymyksiä jää epäselväksi mikä on tutkimuksen tarkoitus ja mitkä ovat sen keskeisiä käsitteitä. Mitä enemmän aiempaa tutkimusta aiheesta on, sitä enemmän on tutkimuskysymystä rajattava. (Polit & Beck 2004, tässä Flinkman & Salanterä 2007, 88.) Jos taas aiempaa tutkimusta ei juurikaan ole, on silloin lavennettava tutkimuskysymystä, jotta siihen saadaan mukaan riittävästi tutkittavaa aineistoa, josta muodostuu mielekäs kokonaisuus (Russell 2005, tässä Flinkman & Salanterä 2007, 88).

Vaikka kirjallisuuskatsaus suoritetaan jo valmiiksi tehtyjen tutkimusten pohjalta, vaatii katsauksen teko silti tekijältään luovuutta tutkimuksien tuloksien yhdistämisessä. Eri tutkimuksien väliset käsitteiden määrittelyiden erot aiheuttavat sen, että käsitteet saavat lopulliset muotonsa integroituun katsaukseen sitä tehtäessä. (Cooper 1989, tässä Flinkman & Salanterä 2007, 90.)

Jos aineistoon valittujen tutkimusten sisältö on hyvin samankaltaista ja niissä asetetut aineiston sisällyttämis- ja poissulkukriteerit eivät toimi, on niiden pohjalta lähes mahdotonta tehdä kovin vahvoja tai merkittäviä johtopäätöksiä. Eri metodeilla tehtyjen tutkimusten arvon ja luotettavuuden arviointi on haastavaa. (Flinkman & Salanterä 2007, 92-93.)

Empiiristen tutkimusten laadun arviointia voidaan kirjoittaa auki esimerkiksi aineiston toisteisuuden, yleistettävyyden ja aineiston edustavuuden mukaan. Kvantitatiivisia tutkimuksia voidaan arvioida esimerkiksi määrittämällä ne kriteerit, joilla katsaukseen mukaan otettuja tutkimuksia arvioidaan. (Flinkman & Salanterä 2007, 93.)

Katsauksissa voidaan käyttää äänestysmetodia siten, että tutkimuksia tarkastellaan sen valossa, kuinka paljon tutkimukset ovat saaneet empiiristä tukea. Sillä tarkoitetaan tutkimustuloksia ja niiden esiintyvyyttä. Siinä paljastuu kunkin tutkimuksen nauttima empiirinen tuki, eli se mitkä tulokset ovat saaneet tutkimuksista eniten tukea. (Polit & Beck 2004, 696 tässä Flinkman & Salanterä 2007, 96.)

Tulosten esittäminen on katsauksen viimeisin vaihe. Julkaistu katsaus lisää tiedeyhteisön pääomaa ja osallistuu tieteelliseen keskusteluun. Johtopäätöksiä esitettäessä on lukijalle näytettävä päätelmäketjut, jotka ovat johtaneet esitettyihin johtopäätöksiin. Työelämässä olevat ihmiset eivät usein kiireidensä vuoksi ehdi tutustua viimeisimpiin tutkimuksiin ja siksi kirjallisuuskatsaukset ovat heille oiva tapa saada uusinta tietoa koostettuna. (Flinkman & Salanterä 2007, 98).

Erilaiset katsaukset ovat ikään kuin tutkimuksia tutkimuksista ja niitä koskevat samat luotettavuuden kriteerit kuin alkuperäistutkimuksiakin (Kirkevold 1997, tässä Flinkman & Salanterä 2007, 98).

4.2 Aineiston esittely

Tutkimuksessa käytetty aineisto pyrkii lähtökohtaisesti olemaan mahdollisimman tasapuolinen valikoima erilaisia emulointiin digitaalisen aineiston pitkäaikaissäilytyksessä liittyviä julkaisuja. Useat tutkimuksessa käytetyt julkaisut eivät kuitenkaan välttämättä käsittele pelkästään emulointia säilytysmenetelmänä, mutta olivat kuitenkin tutkimuksen kannalta relevantteja, käsitellen emulointia jossakin määrin, tai sisältäen jonkin aiheeseen liittyvän idean tai näkökulman jollaista muissa julkaisuissa ei ollut.

Jos julkaisu keskittyisi pelkästään emulointiin, se olisi silloin väistämättä niin tekninen, ettei se enää käsittelisi digitaalisten aineistojen pitkäaikaissäilyttämisen ongelmaa yleisemmin. Sellaiset julkaisut jäisivät joka tapauksessa tutkimukseni ulkopuolelle, sillä pyrinhän nimenomaan omalta osaltani auttamaan kokonaisvaltaisen ratkaisun löytymistä digitaalisen aineiston pitkäaikaissäilyttämisen ongelmaan.

Tasapuolisuuteen pyrkimisestä huolimatta on mahdoton välttää aiheesta eniten julkaisseiden tahojen voittopuolista asemaa aineiston julkaisijoina. Emuloinnin suhteen kaikkein tuotteliain on ollut Alankomaiden kansalliskirjasto Koninklijke Bibliotheek. Toki muiden julkaisemia tutkimuksia on, mutta niissä usein viitataan runsaimmin juuri Koninjklijke Bibliotheekin julkaisuihin. Tästä seuraa väistämättä se, että on mahdotonta

välttää yhden julkaisijan massiivisen volyymin vaikutusta tutkimuksen tekoon ja tästä tutkimuksesta vedettävien johtopäätösten syntymiseen.

Olen kuitenkin pyrkinyt huomioimaan tämän KB:n monopoliaseman emulointitutkimuksessa antamalla enemmän painoarvoa niille julkaisuille, jotka ovat tulleet KB:n ulkopuolelta. On selvää, että KB:llä on emulointitutkimusten julkaisijana oma lehmä ojassa, sillä he ovat laittaneet tutkimuksiinsa paljon rahaa, eivätkä enää voi kovin kunniakkaasti sanoutua irti aikaisemmista emulointiin positiivisesti suhtautuvista tutkimustuloksistaan. Siksi melkein väistämätöntä on, että heidän uudemmat julkaisunsa aina vahvistavat sitä, mitä ovat aiemmissa julkaisuissaan esittäneet.

Näkökulmat emulointiin ja siihen liittyvät tekniset ratkaisut ovat kuitenkin vuosien varrella muuttuneet, ja jo yksistään näiden saman julkaisijan julkaisujen sisällä tapahtuneita muutoksia havainnoimalla voidaan löytää huomionarvoisia seikkoja. Nämä muutokset antavat myös vihjeitä siitä, miten asioiden voidaan olettaa kehittyvän tulevaisuudessa.

Aineiston keruun olen suorittanut kolmen vuoden aikana lähteinäni internetin hakukoneet Google, Yahoo ja Altavista ja Tampereen Yliopiston kirjaston Nelliportaalin kautta EBSCOhost Academic Search Premier-, LISA: Library and Information Science Abstracts-, Emerald- ja Communication & Mass Media Complete (Ebsco) -hakutietokannat. Lisäksi olen etsinyt aiheeseen liittyvää kirjallisuutta amazon.com-kirjakaupasta, josta löytää teoksia kaikilta aloilta. Useiden teoksien viimeisille sivuille on painettu mainoksia muista samalta kustantajalta tulleista alaan liittyvistä kustannuksista ja näistä mainosesittelyistä olen saanut nimikkeitä, joiden perusteella olen tehnyt hankintaehdotuksia, ja näin aineisto on laajentunut. Tampereen Yliopiston kirjasto on hankkinut kaiken ehdottamani aineiston, josta suuri kiitos heille.

Aivan ensimmäinen siemen, josta aineistonkeruu lähti liikkeelle, oli Kirsti Kekkin ja Olli Salmisen toimittama "Digimaan kartta". Sen emulointia ja muita säilytysmenetelmiä käsittelevän osan lopussa oli lähdeluettelo, jonka pohjalta lähdin tekemään hakuja Googleen ja tietokantoihin. Näiden hakujen pohjalta löytyi aineistoja, joiden lähdeluettelot johtivat taas uusiin lähteisiin, joissa oli lisää lähdeluetteloja. Tämä osoittautuikin parhaimmaksi aineistonkeruumenetelmäksi jo silloin kun valmistelin kandidaatintyötäni samasta aiheesta. Tampereen Yliopiston kirjaston keskitetyllä tietokantahaulla ei tämän

tutkimustyön aloittamisen aikana löytynyt aihetta käsittelevää materiaalia käyttämällä "emulation", tai suomenkielistä "emulaatio" -sanaa, joten materiaalin jäljille oli ensin päästävä jostain muualta, tässä tapauksessa Googlesta, Yahoosta ja Altavistasta. Tätä työtä viimeistellessä on tilanne jo muuttunut ja nyt Nelli-portaalin pikahaku alueelta "Viestintä, tietojenkäsittely, informaatiotieteet" antaa "emulation"-sanalla haettaessa 3798 hakutulosta, jossa on mukana monia tässä työssä käytettyjä artikkeleita. Tätä työtä valmistellessa hain nuo artikkelit kuitenkin Googlesta, Yahoosta ja Altavistasta.

Aluksi otin aineistoa mukaan melko laveasti, mutta lopulta rajasin pois kaiken sellaisen aineiston, jossa emulointia ei laisinkaan mainittu. Lisäksi piti rajata pois aineisto, jossa esiintyi sana "emuloida", mutta jossa emulointi tarkoitti jotain muuta kuin emulointia tietokoneissa. Sanana emulointi (jäljittely, matkiminen) on tunnettu kaikilla tieteenaloilla, joten pelkän emuloinnin etsiminen tuottaa aineistoa kaikilta tieteenaloilta. Tietokoneisiin liittymättömät artikkelit on helppoa karsia joukosta pois. Webster's Online Dictionaryn mukaan emulointi tarkoittaa myös kadehtimista ja intohimoista kilpailua, joten hakutulokset voivat olla hyvin kirjavia.

Aineiston löytämistä helpotti, mutta toisaalta sen rajausta vaikeutti näiden kolmen vuoden aikana tapahtunut voimakas emulointitutkimuksen lisääntyminen. Hakutulokset ovat nyt toisenlaisia, kuin mitä ne olivat kandidaatintyötä tehdessäni. Aikaisemmin hakutuloksia piti selata löytääkseen jotain relevanttia, mutta nyt hakulauseke "long-term preservation emulation", tuottaa ensimmäisten hakutuloksien joukkoon pitkän listauksen tässäkin työssä käytettyjä dokumentteja.

Kieliksi rajasin äidinkieleni lisäksi englannin. Muilla kielillä kirjoitettujen tutkimusten tarkasteluun ei olisi ollut mitään mahdollisuuksia. Lisäksi emuloinnista ei juuri muilla kielillä tutkimusta löydykään kuin englanniksi, julkaisumaasta riippumatta. Ilmeisesti tutkimusalue on katsottu niin kapeaksi ja potentiaalinen lukijaryhmä muutoinkin niin pieneksi, ettei pelkästään omalla kielellä julkaisemiseen ole katsottu olevan varaa.

Patentteja koskevat haut tein pääasiassa www.freepatentsonline.com -palvelussa. Vuosikymmenkohtaisessa tarkastelussa käytin hakutermeinä yrityksen nimeä ja vuosilukuja, esimerkiksi "ibm" ja "1960-1970". Tämän jälkeen saatuani kaikki tuon firman tuona ajanjaksona hakemat patentit, lisäsin hakutermeihin sanan "emulator", saadakseni jään-

nökseksi vain ne jotka käsittelevät emulointia. Tämän lisäksi kävin niitä vielä manuaalisesti läpi varmistaakseni, että ne todella liittyvät emulointiin, eikä niissä käytetä tuota termiä jossain muussa kuin tämän työn kontekstissa tarkoitetussa merkityksessä.

Patenttihaku oli välttämätöntä rajata kattamaan jonkin tietyn yrityksen, tämän työn tapauksessa IBM:än, hakemat patentit. Emulointi on terminä, varsinkin tekniikan alueella, niin yleinen että hakutulokset jakautuvat holtittomasti kaikille mahdollisille alueille ilman jämäkkää rajausta. Firma on rajaavana tekijänä emuloinnin suhteen paras, sillä suurin osa emulointiin liittyvistä patentoiduista keksinnöistä on tehty jonkin yrityksen toimeksiannosta.

5. EMULOINTI

Tässä luvussa vien lukijan itse asiaan eli emuloinniksi kutsuttuun menetelmään. Aliluvussa 5.1 ja sen aliluvuissa 5.1.1 ja 5.1.2 käyn läpi emuloinnin historiaa niin tarkasti kuin olen kyennyt sen rekonstruoimaan sirpaleisista lähteistä. Aliluvussa 5.2 käyn läpi emulointikäsitteeseen liittyviä tulkintaepäselvyyksiä. Aliluvussa 5.3 perehdytän lyhyesti lukijan emulointiin säilytysmenetelmänä. Luvussa 5.4 ja sen aliluvuissa 5.4.1, 5.4.2 ja 5.4.3 käyn läpi emuloinnin eri variaatiot. Luvussa 5.5 ja sen aliluvuissa käyn lävitse käytettävyyskysymyksiä, joita emuloinnin käyttöönotto pitkäaikaissäilytyksessä väistämättä aiheuttaa. Luvussa 5.6 tutustutan lukijan näkemyksiin emuloinnista, jotka David Bearman on tehnyt tunnetuksi. Luvussa 5.7 luon silmäyksen ongelmiin, joita emuloinnin käytöstä koituu nykyisen lainsäädännön kanssa. Viimein luvussa 5.8 tuon esille ohjelmistoarkistojen puuttumisen ongelman.

5.1 Emuloinnin historiaa

Tietokonepeleillä on ollut keskeinen rooli emulointitekniikoiden kehittymisessä. Voidaan sanoa, että emulaattori todistavat toimivuutensa ajamalla onnistuneesti tietokonepeliä. Tämä johtuu siitä että tietokonepeli käyttää tietokonelaitteiston kaikkia komponentteja täydellä teholla, ja siten paljastaa mahdolliset heikkoudet emuloinnissa. Taulukkolaskentasovellus on huomattavasti ongelmattomampi emuloitava kuin grafiikka- ja äänipiirejä käyttävä tietokonepeli. Siksi Windows-käyttöjärjestelmän mukana tuleva komentoriviohjelma kykenee ajamaan esimerkiksi OlympiaStat-ohjelmistoa vuodelta 1993 ongelmitta. Sen sijaan grafiikkaa ja ääntä sisältävien demo-ohjelmien ajaminen Windowsin komentorivistä tuottaa usein jo ongelmia.

Emuloinnin historiassa voidaan nähdä kaksi kehityshaaraa, teollinen ja harrastelijamainen. Meidän on tehtävä ero näiden kahden välille, jos haluamme säilyttää historiikkimme johdonmukaisena. Teollinen emulointi juontaa juurensa tietokoneteollisuuden alkumetreille, kuten olemme jo todenneet, mutta harrastelijapiireissä tapahtunut emulointiliikehdintä alkoi vasta myöhemmin. Ilmeisesti sille ei ollut vielä tietokonekauden alkuvai-

heissa tarvetta. Tarve lienee syntynyt siitä, kun kotitietokoneet tulivat suuren yleisön kukkarolle sopiviksi. Siten yhteensopivuusongelmat eri tietokonelaitteistojen välillä alkoivat olemaan yhä enemmän arkipäivää.

Emuloinnin historiasta on kirjoitettu hyvin vähän. Sekin mitä on kirjoitettu, on hyvin pirstaleista, eikä kata johdonmukaisesti koko tietokoneiden aikakautta. Siksi emuloinnin historiaa on kirjoitettava yhdistelemällä eri lähteistä hankittuja palasia toisiinsa. Tehtävä on hankala, sillä monet tiedot vanhoista laitteistoista, esimerkiksi niiden julkaisuvuosista vaihtelevat jonkin verran. Useat emulaattorit ovat vuorostaan saaneet alkunsa harrasteprojekteina, eikä emulaattorin tekijä ole välttämättä panostanut hengentuotteensa kehityksen dokumentointiin. Emulaattoreiden kehityksen painetta tietokoneteollisuuden ulkopuolella on ajanut harrastajien saama voimakas tyydytyksen tunne "koneen voittamisesta". Tämä on otaksuttavasti edelleenkin kaikkein suurin motivaattori, sillä emulaattoreilla ei tiettävästi ole kukaan vielä tehnyt omaisuuksia.

5.1.1 Ei mitään uutta auringon alla

Vaikka emulointi on monille uusi ilmiö ja käsite, ei tietotekniikkateollisuuden läpimurto olisi itse asiassa ollut koskaan mahdollista ilman emulointia. IBM huomasi tämän jo valmistaessaan ensimmäisiä kaupallisia tietokoneitaan. 1964 IBM aloitti System/360-sarjan tietokoneiden valmistuksen (Poulsen 2001, 1). Tätä ennen IBM oli julkaissut 7090/94-sarjan tietokoneita, jotka olivat 360-sarjan edeltäjiä. IBM:än asiakkaiden olisi ollut mahdotonta investoida uusiin ohjelmistoihin ja siksi IBM oli pakotettu varustamaan System/360-tietokoneet vanhempia koneita jäljittelevällä toiminnallisuudella saavuttaakseen mahdollisimman suuren yhteensopivuuden vanhojen ohjelmistojen ja uusien laitteistojen välillä. Jopa nykyisissä zSeries-sarjan koneissa pystyy ajamaan S/360:lle kirjoitettuja ohjelmistoja. IBM on siis tuonut yhteensopivuuden tietokoneissaan näihin päiviin saakka.

Emulointitaipaleen alussa oli siis IBM ja sen suuryrityksille tarkoittamat tietokoneet. Voimme siis karkeasti laskea teollisen emuloinnin alkaneen 60-luvulla pakon sanelemana.

Seuraava merkittävä askel emuloinnissa oli in-circuit emulation -tekniikan käyttöönotto 70-luvun alussa. Tämä ICE-tekniikka on edelleen tehokkain tapa ohjelmointivirheiden

etsimiseen. Tekniikka toimii siten, että laitteistosta poistetaan varsinainen prosessori ja se korvataan ICE-laitteen omalla prosessorilla, joka emuloi poistetun prosessorin toimintaa. Tämä originaalin suorittimen jäljittely antaa mahdollisuuden luoda keskeytyspisteitä, joissa voidaan pysähtyä tarkastelemaan jotakin yksittäistä suorittimeen menossa olevaa käskyä, tai vastaavasti suorittimesta ulos tulevaa liikennettä. ICE-tekniikan avulla järjestelmän toimintaa tutkiva henkilö voi asettaa ennakkoehdon, jonka täyttyessä suoritus keskeytyy. ICE mahdollistaa myös ohjelmien ajamisen siten, että ohjelmaa ajetaan askelletusti käsky käskyltä mahdollisten ohjelmointivirheiden havaitsemiseksi.

Tietokoneteollisuuden kannalta tärkein ominaisuus ICE-tekniikassa on kuitenkin ehkä se, että sen avulla voidaan testata vasta suunnitteluvaiheessa oleville tietokonekomponenteille kirjoitettua ohjelmakoodia. Näin ohjelmistojen kehittäjien ei tarvitse odottaa varsinaisen fyysisen laitteen valmistumista, vaan he voivat emuloida sitä ICE-laitteiston avulla. Näin ohjelmistoja voidaan tehdä valmiiksi laitteistoille, joista on olemassa vasta piirustukset, mutta ei vielä edes prototyyppiä.

Lähteistäni olen vetänyt sen johtopäätöksen, että ensimmäisen harrastelijaemulointiohjelmiston kirjoitti Michael Day. Projekti sai alkunsa vedonlyönnistä. Vetoa lyötiin siitä että pystyttäisiin ohjelmoimaan emulaattori, joka emuloi Intelin 8080-suoritinta Lomas 8086-suorittimessa ja kykenisi ajamaan Microsoftin CP/M-käyttöjärjestelmää. Vetoon sisältyi ehto, että ohjelma olisi kirjoitettava yhdessä viikossa. Michael Day tarttui vetoon ja sai viikossa aikaan emulaattorin, joka melkein toimi. Se kykeni kyllä emuloimaan 8080 suoritinta, mutta kaatui heti jos siinä yritti ajaa jotakin ohjelmistoa. Tämän jälkeen Day unohti kirjoittamansa ohjelman muutamaksi vuodeksi. Vanha ohjelma tuli taas ajankohtaiseksi kun Day yritti konvertoida 8080:lle kirjoitettua ohjelmistoa IBM PC:lle. Day tuli siihen tulokseen, että oli löydettävä jokin muu ratkaisu kuin konvertointi ja muisti silloin vuosia sitten ohjelmoimansa emulaattorin. Tämä tapahtumakulku johti siihen, että Day ohjelmoi IBM PC:lle CP/M-emulaattorin. Myöhemmin hän teki tästä myös Z80-suorittimille sopivan version, sillä muutkin halusivat kyetä käyttämään CP/M-käyttöjärjestelmää ilman että heidän tarvitsisi hankkia uusia laitteistoja. Näin siis ensimmäinen kaupallinen emulaattoriohjelmisto syntyi MS-DOS-käyttöjärjestelmää käyttävien ihmisten toivomuksista päästä käyttämään samalla laitteistolla myös CP/M-käyttöjärjestelmää. Valmiin ohjelmiston nimeksi tuli CPeMulator. Koska sen kehitystyö

alkoi 1981, näyttää ilmeiseltä, että se oli ensimmäinen harrastelijapiireistä liikkeelle lähtenyt kaupallinen emulointiohjelmisto, jossa emulointi tapahtui pelkästään ohjelmistolla ilman lisälaitteita. (Day, S.a.)

Ilman yhteensopivuutta aikaisempien malliensa kanssa tuskin yksikään tuntemamme tietokonevalmistaja olisi edes tullut tietoisuuteemme. Kuluttajat eivät voi hyväksyä tilannetta, jossa heidän olisi joko luovuttava kaikista vanhoista ohjelmistoistaan tai ostettava ne uusiksi uudelle laitteistoille käännettyinä. Tästä syystä tietokoneissa, niin kotikuin teollisuuskäytössäkin olleissa on aina käytetty jonkinlaista tekniikkaa takaamaan yhteensopivuus aikaisemmin julkaistujen mallien kanssa. Teollista emulointia on siis aina käytetty.

Tästä syystä Commodore 128 -tietokone varustettiin lähes 100% yhteensopivuudella aikaisemman Commodore 64:n kanssa (Datasalen, S.a). Commodore 128:n tapauksessa on tosin hieman epäselvää missä määrin kyse on emuloinnista, ilmeisesti emulointia ei tuossa laitteistossa paljoakaan tapahdu, vaan kyse on ennemmin lähinnä rakenteellisista yhteneväisyyksistä, jotka mahdollistavat yhteensopivuuden. Itse asiassa Commodore 128:ssa oli kolme tietokonetta yksissä kuorissa (Kirps 2007). Koneen pystyi käynnistämään erilaisiin tiloihin, joista yksi oli Commodore 128:n edeltäjä Commodore 64. Commodore-tietokoneiden vaikutus vallitsevassa tietokonekulttuurissamme on ohittamaton. Se on myös nykyään eniten ja kattavimmin emuloitu tietokone. Commodore 64 -emulaattoreita löytyy kaikille käyttöjärjestelmille ja Mac-koneille. Se on yksi vahvimpia evidenssejä siitä, että myös meidän tällä hetkellä käyttämiämme laitteistoja pystytään varmasti tulevaisuudessa moitteettomasti emuloimaan, jos emuloinnin kehitykseen vain panostetaan.

Commodoren tavoin teollisesta emuloinnista hyvä esimerkki on Atarin toiminta. 1977 Atari julkaisi Atari 2600-pelikonsolin. 1982 julkaistiin Atari 5200, jonka myynti jäi kuitenkin heikoksi lukuisten suunnitteluvirheiden takia. Huomionarvoinen yksityiskohta on, että 5200-konsolin ensimmäinen versio oli täysin yhteensopimaton 2600-konsolin kanssa. Yleisön painostuksesta 5200-konsolista julkaistiin myöhemmin versio, joka oli yhteensopiva 2600-konsolille julkaistujen pelien kanssa. Mutta 1984 julkaistu Atari 7800 suunniteltiin yhteensopivaksi kaikkien aikaisempien Atari-pelikonsoleiden kanssa (Atari 1984). Samoin kuin IBM:n System/360 kanssa tehtyjen ratkaisujen suhteen, täs-

säkin oli kyse pakon sanelemasta ratkaisusta. Kuluttajat eivät olisi hyväksyneet täydellistä yhteensopimattomuutta omistamiensa vanhemmille konsoleille kirjoitettujen pelien kanssa. Emulointiohjelmisto sille ilmestyi verrattain myöhään, koska Atari 7800 ei koskaan ollut markkinamenestys. Se on osoitus siitä, että vaikka jokin laitteisto olisi huonosti tunnettu suuren yleisön keskuudessa, pystytään sitä kuitenkin tarvittaessa emuloimaan, jos vain täsmälliset faktat sen teknisestä rakenteesta ovat tiedossa.

Darek Mihocka perusti emulointiin omistautuneen yrityksensä Branch Always Softwaren 1988. Yhtiö tunnetaan nykyään nimellä Emulators inc. Yrityksensä alkuvaiheista saakka Mihockalla oli yksi visio: että kaikkeen tarvittaisiin vain yksi tietokone. Sen tunnetuimpia tuotteita ovat SoftMac, Gemulator, Xformer ja Fusion PC. Näillä ohjelmistoilla saa PC-koneen emuloimaan Macintosh-koneita. Tämän vuosituhannen puolella on Mac-koneiden emulointi PC-koneissa helpottunut sen johdosta, että Apple siirtyi käyttämään Intel-suorittimia. 1994 Emulators inc. julkaisi ensimmäisen version Xformeristaan, joka emuloi Atari ST:tä. (Stevens 2008, 77-78; 96.)

1989 Gadgets by Small -niminen yhtiö julkaisi Spectre GCR -nimisen lisälaitteen, jonka avulla Atari ST:n sai emuloimaan Applea. Lisälaitteen suurin yhteensovitustehtävä oli saada Atari ST lukemaan Macintoshin diskettejä, joissa käytettiin liukuvaa kierrosnopeutta, toisin kuin Atari ST:ssä. Lakikysymykset oli vältetty siten, että yksin tuon laitteen ostamisella ei emulointi onnistunut, vaan käyttäjän oli ostettava erikseen Applelta Macintoshin käyttöjärjestelmälevykkeet ja ROM-tiedostot. Myöhemmin 1990 Talon technologies julkaisi SuperCharger-lisälaitteen, jolla Atari ST:n sai emuloimaan myös PC:tä. (Plotkin 1990, 33.) Näin Atarin käyttäjä sai lisälaitteiden avulla ulottuvilleen kolme erilaista tietokonealustaa.

1990-luvulle tultaessa emulointi alkoi olemaan kova sana. 80-luvun 8-bittisiä tietokoneita alettiin emuloimaan uusilla tehokkaammilla 16-bittisillä koneilla. Ne eivät kyenneet yhtä täydelliseen emulointiin kuin nykyään, mutta ovat hyvä osoitus siitä, että erilaisten järjestelmien väliset raja-aidat on aina haluttu, yritetty ja kyetty kaatamaan käyttäjäkunnan toimesta. Suuri yleisö on myös aina ymmärtänyt tietotekniikan verrattain lyhyen eliniän, huolimatta varsinkin 80-luvulla alan julkaisuissa mantran lailla hoetusta "elektroniikka on käytännöllisesti katsoen ikuista"-uskontunnustustuksesta. Lisäksi emulointikehitystä on ajanut voimakkaasti eteenpäin oman aseman nostaminen emu-

lointiscenessä. Vaikka olisikin ollut yleisessä tiedossa että tietokone X ei ole riittävän tehokas emuloimaan konetta Y, on emulaattori haluttu silti koodata osoitukseksi omista ohjelmointitaidoista.

Jeff Vavasour julkaisi 1990 emulointiohjelmiston TRS-80 -tietokoneelle. TRS-80:llä oli merkittävä markkinaosuus kotitietokoneiden joukossa 70-luvun lopulla. Vavasour kertoo, että siirtyessään työskentelemään PC-koneilla, hän jäi kaipaamaan tunnetta siitä että laitteisto on läpikotaisin hänen kontrollinsa alla. Tunne, joka hänellä oli ollut TRS-80:n kanssa. 1989 hän selaili vanhoja tietokonelehtiä ja törmäsi artikkeliin, jossa väitettiin että TRS-80:n mikroprosessori Z-80 kykenisi jäljittelemään Applen 6502-suoritinta. Tästä hän sai kipinän kirjoittaa emulaatio-ohjelmistoa, joka emuloisi TRS-80 tietokonetta PC-konetta. Vavasourin tarinasta käy ilmi se motivaatio, joka on ajanut eteenpäin kaikkea emulointikehitystä. "Haasteen ja tekemisen hauskuuden vuoksi", on hänen vastauksensa kysymykseen siitä, miksi hän koko projektiin ryhtyi. (Vavasouri 2000.)

Miha Peternl ohjelmoi todennäköisesti ensimmäisen Commodore 64-laitteistoa emuloivan ohjelmiston PC-tietokoneille 1991. Ohjelman nimeksi tuli C64S. Luultavasti tämän ohjelmiston myötä alkoi C-64 -tiedostojen siirtäminen PC-koneisiin ja myöhemmin Internetiin kaikkien ladattavaksi. Ohjelma nimittäin sisälsi kaiken tarvittavan tiedostonsiirtoon C-64:n ja PC:n välillä. 1991 C-64 oli vielä tuotannossa ja myynnissä, joten halukkaat pioneerit saivat niin laitteet kuin levykkeille tallennetut ohjelmistotkin vaivatta käyttöönsä. Ainut ylimääräinen fyysinen osa joka tiedostonsiirtoon tarvittiin oli erikoiskaapeli, jollaisen elektroniikkataitoinen harrastelija kykeni vähäisellä vaivalla rakentamaan itsekin. Emulaattori oli kirjoitettu kokonaan Assembly-kielellä. Voidaan otaksua, että tuon ajan PC-koneilla ei jouheva emulointi olisi muilla kielillä ohjelmoiden ollut lainkaan mahdollista, sillä jotta koneesta saatiin kaikki irti oli sitä ohjelmoitava mahdollisimman ruohonjuuritasolla, jolloin korkeamman tason ohjelmointikielet eivät tulleet kysymykseen, sillä ne vaativat kääntäjän.

1990-luku, Windows 3.1 ja multimediastandardin julkistaminen PC-koneille olivat ratkaisevia tekijöitä emulointiharrastuksen suosion nostamisessa. Nämä uudet ominaisuudet PC-koneissa tekivät mahdollisiksi vanhojen pelikoneiden, kuten Nintendo Entertainment Systemin emuloimisen PC-koneissa. Tämä myös johti siihen, että emulaatio-ohjelmistojen graafiseen käyttöliittymään alettiin panostamaan ja näin emuloinnista tuli käyt-

täjäystävällisempää. Tämä on myös Stevensin mukaan syy siihen miksi emulointiscene alkoi niin vahvasti kehittymään PC-koneiden puolella (Stevens 2008, 67-69.)

Emuloinnin historiaa on mahdotonta tutkia törmäämättä Maratz Fayzulliin. Hän tarttui ensimmäisenä haasteeseen iNES-pelikonsolin emuloimisesta. Emulaattorin levisi nopeasti ympäri maailman juuri tuolloin voimakkaasti yleistyvien internet-yhteyksien ansiosta. Leviämisen varjopuolena oli se seikka, että emulointi muodostui eräänlaiseksi synonyymiksi piratismille ja kopiosuojauksien murtamiselle. Tämä ei ollut Maratz Fayzullin kaltaisten pioneerien tarkoitus. Myös Fayzullin emulaattori tuli lopulta piratisoiduksi, mikä vähensi monien emuloinnista kiinnostuneiden ohjelmoijien mielenkiintoa emulointia kohtaan. (Stevens 2008, 69-73.)

1990-luvun alkupuolella esiintyi siis voimakasta kiinnostusta nimenomaan pelikonsolien emuloimista kohtaan. 90-luvun alkupuoli antoi emuloinnille pitkäksi aikaa ikävän leiman piratismin kanavana.

Vuosituhannen vaihteen häämöttäessä alkoi yleistymään idea siitä, että olisi olemassa yksi ohjelmisto, joka emuloisi kaikkia aikaisempia laitteistoja. Tämä tarkoitti lähinnä pelikonsoleja. Ohjelmisto sai nimekseen Multiple Arcade Machine Emulator ja se kykenee emuloimaan useita tuhansia konsoli- ja kolikkopelejä. Jos tämä on mahdollista pelien kohdalla, herättää se väistämättä kysymyksen siitä, miksei tämä olisi mahdollista kaikkien tietokoneohjelmien kohdalla? MAME sai myöhemmin runsaasti sisarprojekteja ja muita seuraajia. MAME ja sen sisar- ja oheisprojektit ovat merkittäviä tämän työn kannalta sillä niiden pyrkimys on nimenomaan emuloida alkuperäistä laitteistoa mahdollisimman uskollisesti, kaikkine virheineen päivineen. Tämä lähestymistapa on juuri se, jota tavoitellaan kun emulointia ollaan käyttämässä digitaalisen aineiston pitkäaikaissäilytykseen.

1999 emulointirintamalla kohistiin kun tammikuun 28. päivä julkaistiin internetissä UltraHLE-emulaattori. Nintendo 64-pelikonsolin emuloimista tuon ajan PC-koneilla oli aikaisemmin pidetty mahdottomana. UltraHLE esitteli kokonaan uudenlaisen konseptin emulointiin. Sen sijaan että siinä olisi pyritty emuloimaan laitteistoa mahdollisimman ruohonjuuritasolta, niin kuin emulaattoreissa aiemmin, tässä emulaattorissa ei pyritty-kään emuloimaan Nintendo 64-pelikonsolin jokaista komponenttia. UltraHLE:ssä emu-

lointiohjelmiston tehtävänä olikin tunnistaa mitä ohjelmisto haluaa tehdä ja sitten tehdä se käytössä olevan laitteiston kautta, ei luomalla noita komponentteja virtuaalisesti tietokoneen muistiin, kuten aikaisemmin oli ollut lähtökohtana. Tästä tulivat ohjelmiston kolme viimeistä kirjainta, HLE, High Level Emulation. Tämä oli myös ensimmäinen kerta emuloinnin historiassa, kun emuloitiin sellaista laitteistoa, jolla oli vielä markkina-arvoa. UltraHLE ilmestyi vain kolme vuotta Nintendo 64-pelikonsolin markkinoille tulon jälkeen. Nintendo ja peliohjelmistojen valmistajat uhkasivat emulaattorin tekijöitä niin kovalla lakipaineella, että nämä jättivät projektin kokonaan sikseen. Tapahtuma on kuitenkin merkittävä, koska se todisti että laitteiston ei tarvitse olla vanha, jotta sitä pystytään emuloimaan PC-koneilla.

Stevens (2008, 100) tuo esiin unelman yhdestä ainoasta tietokoneesta, joka kykenisi ajamaan minkä tahansa tietokoneohjelman ja emuloimaan mitä tahansa laitteistoa. Tämä muistuttaa suuresti Lorien laatimaa Universal Virtual Computerin (UVC) ideaa. Siihen palaamme myöhemmin.

Eräs mainitsemisen arvoinen laiteperhe emuloinnin historiassa on Acornin julkaisema Archimedes-tietokoneiden sarja. Archimedes-tietokonepaketti varustettiin emulointiohjelmistolla, joka toimitettiin mukana tulevalla levykkeellä. Emulaattorilla pystyi muuntamaan uuden Archimedes-tietokoneensa vanhemmaksi BBC Micro -tietokoneeksi ja siten ajamaan vanhoja BBC Microlle kirjoitettuja ohjelmistoja. Näin Archimedes tietokone pystyi viemään käyttäjänsä vuodesta 1987 vuoteen 1981. Kaikki vanhat ohjelmistot toimivat uudemmassa koneessa moitteettomasti. (Acorn 1987.)

Commodoren menestystarina jatkui myöhemmin Amiga-tietokoneperheen julkaisemisella. Amiga 500 saavutti valtavan suosion pelikoneena, johtuen siitä että suuri määrä C-64 käyttäjiä päivitti koneensa Amigaan. Kuitenkin, myöhemmin julkaistut Amiga-mallit eivät enää menestyneet samoilla tavoin ja voidaankin otaksua, että juuri niiden keskinäinen yhteensopimattomuus oli yksi suurimpia tekijöitä, joka vauhditti PC-teknologian läpimurtoa tietokoneen synonyymiksi. Tämän työn kirjoittajalla on omakohtainen katkerahko kokemus siitä, kuinka päivittäessään laitteistoaan seuraavaan sukupolven Amigaan, kaikki vanhat ohjelmistot muuttuivat arvottomiksi ja koko varustelun joutui aloittamaan ikään kuin tyhjästä.

Kuitenkin Amigalle ilmestyi lukuisia C-64 -emulaattoreita, kuten Go64, MagiC64 ja A64 (Goodwin 1996. 24). Myös kilpailevan PC-järjestelmän yhteensopivuuden merkittävyys ymmärrettiin ja Amigaan oli saatavilla Sidecar-niminen tuote, jolloin pystyi käyttämään Amigaa ja PC:tä rinnakkain (Bolsø et al. S.a.). Amiga pystyi myös emuloimaan oman menestyksensä huipulla ehkä pahinta kilpailijaansa AtariST:tä Amtari ja Medusa-nimisten emulaattoreiden avulla (Gruehn a, S.a; Gruehn b, S.a). Atarin emuloiminen Amigalla oli siinä mielessä helppoa, että sekä Atarissa että Amigassa oli sama Motorola 68000-suoritin.

Amiga pystyi myös emuloimaan Mac-tietokonetta A-max- ja Emplant -emulaattoroiden avulla. Emplant piirilevyyn oli jopa kirjoitettu se ydinajatus, joka tekee emuloinnista niin tärkeän tekniikan tietotekniikassa. Siinä luki: "Sinä pidät kädessäsi laitetta, joka pystyy emuloimaan käytännöllisesti katsoen mitä tahansa tietokonetta mitä maailmassa on koskaan julkaistu". (Block et al. 2006; NicDouille, 2006.)

Myös PC-laitteistoille oli saatavissa jo 80-luvun puolella ohjelmistoja, joilla pystyi emuloimaan muiden tietokoneiden toimintaa. Yksi tällainen oli The Apple Emulator APL2EM, joka kykeni emuloimaan Mac-tietokonetta jo Intelin 80286-pohjaisissa laitteistoissa, mikä on oiva osoitus siitä että emulointi ei välttämättä vaadi valtavasti laskentatehoa, jos ohjelmoija osaa kirjoittaa koodinsa siten, että laitteiston ominaisuudet käytetään monipuolisesti ja tehokkaasti hyväksi. (Stevens 2008, 60-61.)

Näiden esimerkkien tarkoitus on osoittaa, että emulointi on ollut ja on edelleen välttämätön ja hyvin tunnettu alue tietokoneteollisuudessa. Siksi meillä ei ole painavia perusteluja epäillä sen suoriutumisesta myös digitaalisen kulttuuriperinnön saavutettavuuden turvaamisesta pitkälle tulevaisuuteen.

5.1.2 Patenttien kertomaa

Tutkimusta tehdessäni havaitsin, että seuraamalla patenttitietokantoihin rekisteröityjä emuloimiseen liittyviä keksintöjä voidaan hahmotella aikajanaa emuloinnin kehitykselle. Patenttitietokannasta löytyvät kaikki merkittävät keksinnöt, myös tietotekniset. On hyvin epätodennäköistä että uutta tietoteknistä kojetta, menetelmää tai mallia ei olisi patentoitu, sillä vain keksinnön patentoinnilla voidaan varmistaa, että siitä saadaan maksi-

maalinen taloudellinen hyöty. Näin patenttitietokannat muodostavat luotettavan ja kronologisen tietokannan tekniikan kehityksen seurantaan.

Koska IBM aloitti ensimmäisenä tuottamaan tietokoneita sarjatuotannolla julkiseen käyttöön ja he tuon kehityspaineen pakottamana myös mitä ilmeisemmin ensimmäisinä käyttivät emulointia tietokoneissaan (ainakin tämän työn kontekstin tarkoittamassa merkityksessä), oli siitä syystä odotuksenani että he olisivat patentoineet emulointiin liittyviä menetelmiä ja tekniikoita jo 1960-luvulla. Tämä ennakko-oletukseni osoittautui kuitenkin vääräksi. 1950-1960 IBM on hakenut vain kahdelle tietotekniselle keksinnölle patenttia, eikä niistä kumpikaan liity emulointiin. 1960-1970 tapahtuu räjähdysmäinen kasvu, patentteja on haettu jo 724 kpl. Näistä 3 liittyy emulointiin, mutta niistäkään ei yksikään tämän työn emuloinnille antaman merkityksen piirissä. Seuraavalle vuosikymmenelle 1970-1980 siirryttäessä IBM on hakenut patentteja yhteensä 4716 kpl ja näistä 37 liittyy emulointiin. 1980-1990 IBM on hakenut patentteja 5709 kpl ja näistä emulointiin liittyviä on 20 kpl. 1990-2000 IBM on hakenut patentteja 23943 kpl, joista emulointiin liittyviä on 1572 kpl. 2000-2010 IBM on hakenut patentteja 29846 kpl, joista emulointiin liittyviä 1467 kpl.

Huomaamme siis, että tietotekniikan yleistyessä ja siihen liittyvien patenttien määrän lisääntyessä tapahtuu väistämättä myös emulointitekniikkaan liittyvien patenttien lisääntymistä. Tämä on vahva osoitus siitä, että tietotekniikka ei olisi käyttökelpoista ilman emulointia tai ainakin sen käyttö olisi niin hankalaa ettei se olisi koskaan yleistynyt nykyisessä määrin.

Hakuni rajasin koskemaan ainoastaan IBM:än hakemia patentteja koska muutoin hakualue laajenee kaikkien patenttien alueille, johtuen emulointi-sanan yleisyydestä ja monimerkityksellisyydestä. IBM:än valitsin siksi, että se on ollut ja on edelleen merkittävin tietokonelaitteistojen valmistaja, ja nykyisin yleisimmässä käytössä olevan PC-arkkitehtuurin takana.

Tyypillinen patentti sisältää tiivistelmän, keksinnön kaavakuvat ja keksinnön taustan. Patenttidokumentissa käydään läpi miten aikaisemmin on ratkaistu jokin tekninen ongelma, ja sitten esitetään väitteitä uudesta keksinnöstä, joiden paikkansapitävyys kaavakuvia apuna käyttämällä osoitetaan.

Esimerkiksi Danny B. Ballardin Motorolalle 1985 hakemassa patentissa esitellään tekniikka, joka lisää uusien ja vanhojen tietokonelaitteistojen yhteensopivuutta siten, ettei uudemman laitteiston tarvitse yhteensopivuuden takia luopua kehittyneempien ominaisuuksiensa käytöstä. Patentissa esitellyn menetelmän avulla saadaan siis uudempi laitteisto yhteensopivaksi vanhemmille laitteistoille kirjoitettujen ohjelmistojen kanssa siten, ettei uudempien laitteistojen ominaisuuksista tarvitse yhteensopivuustilassa luopua. (Ballard 1985, 4.)

Nopeus on aina ollut emuloinnissa oleellinen asia. Esimerkiksi 1990 patentoidussa keksinnössä esitellään tekniikkaa, jonka avulla emulointia voidaan huomattavasti nopeuttaa. Nopeutus perustuu siihen, että järjestelmä arvaa mikä on seuraava suoritettava toiminto, joka on emuloitava. Järjestelmä esiaktivoi tuohon toimintoon liittyvät komponentit. Lopputuloksena on emuloinnin nopeutuminen. (Ammann et al., 1990, 1-6.)

Potter patentoi 1998 emulointitekniikan, joka vähentää ongelmia erilaisilla käskykannoilla varustettujen suorittimien välillä. Tuohon aikaan oli yleistä, että suoritinvalmistajat varustivat suorittimiaan käskykannoilla, joita muissa suorittimissa ei ollut tehostaakseen markkina-asemaansa. Tämä aiheutti paljon ongelmia 90-luvulla, jolloin ohjelmistovalmistajat halusivat käyttää tiettyjen suorittimien tukemia tehokkuutta nostavia käskykantoja, mutta samalla hintana oli yhteensopimattomuus monen muun suorittimen kanssa. Potter esittelee patentissaan suorittimen alijärjestelmän, johon lähetetään käsky, jota suoritin ei ymmärrä. Alijärjestelmä asettaa tuon käskyn sijalle vastaavan käskyn, jonka suoritin kykenee ymmärtämään ja lähettää sen takaisin suorittimelle. Näin eri käskykannoilla varustetut suorittimet saadaan keskenään yhteensopiviksi. (Potter 1998, 5.)

Kuijsten esitteli 1999 emulointijärjestelmän, jossa yhtä emuloitua kellojaksoa kohti kyetään emuloimaan useita kellojaksoja. Tämä mahdollistaa monimutkaisten piirien nopeamman emuloinnin käyttämällä itse fyysistä laitteistoa vähemmän. (Kuijsten 1999, 24.)

Giles patentoi grafiikkaprosessorinemulointijärjestelmän ja menetelmän emuloidun ajan ja todellisen ajan synkronointiin (Giles 2000, 2). Patentti on merkittävä koska siinä esitelty tekniikka mahdollisti Playstation-pelikonsolin pelien sulavan emuloimisen PC-koneissa Virtual Game Station -emulaattorilla. Sitä ennen oli Playstationin emuloimista

PC-koneissa pidetty jokseenkin mahdottomana. Sony reagoi asiaan haastamalla Gilesin firmoineen ensin oikeuteen ja hävittyään osti Virtual Game Stationin ja lopetti sen jakelun saadakseen sen pois vähentämästä Playstation-pelikonsolien myyntiä. (Giles, S.a.)

2000 Lorie on patentoinut tässä työssä esitetyn Universal Virtual Computerin (UVC) konseptin. Siinä esitetään, että jos patentissa esitetystä teknologiasta tulisi yleinen standardi, se velvoittaisi kaikki tietokonelaitteistojen valmistajat kirjoittamaan laitteistoilleen UVC-vastineen. Lorien näkee tämän ainoana keinona säilyttää ohjelmistojen ja niitä ajavien laitteistojen toiminnallisuus riittävän pitkälle tulevaisuuteen. (Lorie 2000, 1.)

Tässä työssä pyrin osoittamaan, että lukemattomista emulointitekniikoihin liittyvistä patenteista tämä viimeisin on kaikkein merkittävin, sillä se pyrkii yhdistämään kaikki tietokonealustat yhdeksi alustaksi, ratkaisten siten kaikki yhteensopivuusongelmat.

Diagrammi 1. Teollisen ja harrastelijamaisen emuloinnin sijoittuminen aikajanalle.

5.2 Emulointi ja virtualisointi -käsitteiden rajaamisesta

Emulointi ja virtualisointi ovat käsitteinä monitulkintaisia, ja lähteitä tutkimalla näyttää aluksi aivan toivottomalta saada asiaan lopullista selvyyttä. Emulointia ja virtualisointia käytetään lähes synonyymeina, ja vain harva lähde rajaa ne tiukasti erilleen toisistaan. Lisäksi arkikielessä niitä käytetään hyvin huolimattomasti ja päällekkäin. Tutkimusprosessini edetessä tulin siihen johtopäätökseen, että jotta voin tutkia emulointia, on minun ensin osattava määritellä ja rajata tutkittava kohde, emulointi ja eroteltava se muista kohteista. Tehtävä ei ollut helppo sillä asiasta on saatavilla paljon tietoa ja osa tiedoista on selvästi virheellisiä. Esim. tietokonekansan parissa jokseenkin korkeaa arvostusta nauttiva MikroPC-lehti käyttää näitä käsitteitä lähes mielivaltaisesti vaihtaen yhtäkkiä (jopa samasta kohteesta puhuttaessa) kesken kappaleen käsitteen virtualisoinnista emuloimiseen ja toisin päin (Pulkkinen 2007, 45; Pulkkinen 2008, 26). Voidaan siis sanoa, että asiasta on saatavilla paljon yleisesti luotettavina pidettyjä lähteitä, jotka nauttimastaan arvostuksesta huolimatta käyttävät näitä kahta käsitettä huolimattomasti aiheuttaen sekaannusta.

Voidaan kysyä onko koko käsite-epäselvyydellä mitään merkitystä, sillä sekä emuloinnin että virtualisoinnin tuottama lopputulos näyttää loppukäyttäjälle usein ulkoisesti täysin samanlaiselta, ja siksi väärinkäsityksen vaara on ilmeinen. Jos siis vaikutus on sama, mitä merkitystä on sillä mitä nimeä tuosta vaikutuksesta käytetään? Sillä on digitaalisen aineiston pitkäaikaissäilytyksen kannalta erittäin suuri merkitys, sillä virtualisoimalla ei voida tulevaisuudessa käsitellä meidän aikamme digitaalista aineistoa.

Virtualisoinnissa ei käännetä suorittimien käskyjä toisen suorittimen kielelle ja juuri tätä on tulevaisuudessa väistämättä tehtävä ja siksi tarvitsemme emulointia. Digitaalisten aineistojen pitkäaikaissäilytysprojektien kannalta on haitallista, jos projekteissa työskentelevät ihmiset eivät tunne käsitteiden välisiä eroja. Käsitteet menevät sekaisin myös useissa tämän työn lähteenä käytetyissä julkaisuissa, ja siksi onkin aihetta ehdottaa että käsitteisiin kiinnitettäisiin aiempaa enemmän huomiota.

Asiassa on kuitenkin tapahtunut viime vuosina huomattavaa edistystä, sillä ympäri maailmaa on alettu kyselemään, mitä eroa näillä kahdella käsitteellä on, mitä ne pitävät sisällään ja mitä eivät, ja millä perustein sama valmistaja on luokitellut toisen tuotteensa

emulaattoriksi ja toisen virtualisointiohjelmistoksi. Käsitekysymyksessä on nyt siis huomattavasti enemmän lähteitä, ja siten parempi lähtökohta selvitystyölle kuin tehdessäni samasta aiheesta kandintyötä 2009-2010.

CAMiLEON-projekti toteutettiin 1999-2003. Siinä emulointi määriteltiin toiminnaksi, jossa luodaan uudelleen digitaalisten objektien tarkasteluun tarvittava tekninen ympäristö.[4] Tämä määritelmä sallii kaikki keinot tuon menneisyyden teknisen ympäristön toteuttamiseen, ja siten levittää työn liian laajalle. Raymond Lorie (2002, 35) määrittelee emuloinnin toiminnaksi jossa binääriohjelmaa ajetaan uudessa järjestelmässä siten, että jokainen komento systemaattisesti simuloidaan.[5] Tämä on jokseenkin epäselvä määritelmä ja sikäli outo, että samalta julkaisijalta (KB:ltä) on tullut ulos tutkimuspapereita joissa emulaatio määritellään paljon täsmällisemmin, selvemmin ja kansantajuisemmin. Tarvitsemme siis selvemmän määritelmän ja jatkamme etsimistä.

Rothenberg (2000) kuvailee emulaattorin virtuaalikoneeksi, joka vastaa emuloitua tietokonetta ja joka ajetaan emuloivassa koneessa.[6] Hän jatkaa että vastavuoroisesti mikä tahansa virtuaalikoneohjelma emuloi tuota virtuaalikonetta isäntätietokoneessa.[7] Paras analogia arkielämästä emulaattorille on hänen mukaansa imitaattori. Runoilijaa imitoivalta henkilöltä odottaisimme, että hän kykenee kirjoittamaan runon imitoimansa henkilön mukaisesti, mutta runoilijaa esittävältä näyttelijältä emme sitä odottaisi.

Jaakkohuhdan (2003) mukaan virtuaalikone tarkoittaa näennäiskonetta. Wikipedian kuvauksen mukaan näennäiskone on ”on ohjelmallisesti toteutettu tietokone, jossa voidaan ajaa ohjelmia kuin aidossa koneessa.” Näin siis päädymme siihen, että jokaista virtuaali-

---

4 ”the re-creation on current hardware of the technical environment required to view and use digital objects from earlier times”.

5 ”the process of executing a binary program on a new system by systematically simulating the behavior of every instruction”.

6 ”Since an emulator is a program that makes one computer act like a different computer, an emulator can be thought of as implementing a virtual machine that corresponds to the emulated computer and runs on the emulating computer”.

7 ”Conversely, any implementation of a virtual machine is a program that emulates that virtual machine on some host computer ”.

konetta esim. Java Virtual Machinea voi sanoa emulaattoriksi, sillä erolla että ne eivät jäljittele jotain fyysistä tietokonetta, vaan luovat tietokoneen muistiin koneen, jota ei ole fyysisesti koskaan ollut olemassa. Huomaamme kuitenkin välittömän ristiriidan tämän kuvauksen ja sen kanssa, mitä Javan kehittänyt Sun Microsystems sanoo omasta tuotteestaan. Heidän mukaansa Java on ohjelmointiympäristö, joka toimii omassa hiekkalaatikossaan, virtuaalikoneessa. Emuloinnista ei heidän mukaansa ole kysymys. Huomaamme siis, että koska emulointi tarkoittaa väljästi tulkiten matkimista ja jäljittelyä, on loppujen lopuksi kunkin raportin kirjoittajan omassa vallassa se, mitä hän nimittää emuloinniksi.

Jatkamme ja puramme käsitteet nyt osiin: emuloinniksi kutsutaan sellaisen sovelluksen toimintaa, joka saa ohjelmallisesti aikaan saman vaikutuksen kuin alkuperäinen laitteisto ja virtualisoimiseksi kutsutaan tekniikkaa, jossa järjestelmän resurssien tekniset piirteet piilotetaan järjestelmältä siten, että yksi fyysinen resurssi kykenee toimimaan useana loogisena resurssina. Siten koneessa voi toimia päällekkäin kaksi eri käyttöjärjestelmää yhden suorittimen toimiessa jaetusti kahtena näiden käyttöjärjestelmien välillä.

Näiden kuvausten perusteella asia näyttääkin jokseenkin selvältä, mutta muuttuu mutkikkaammaksi kun havaitaan että nimenomaan myös emulointiohjelmistolta piilotetaan järjestelmän todelliset tekniset piirteet, ja huijataan ohjelmistoa toimimaan vääränlaisen laitteiston päällä. Vielä sekavammaksi tilanne muuttuu ohjelmistojen toimittajien kuvauksia lukiessa. Esim. Kevin Lawtonin kehittämän, alun perin kaupallisen, mutta myöhemmin vapaaseen jakeluun lasketun BOCHS-ohjelmiston ilmoitetaan olevan ”emulaattori, joka tarjoaa virtualisointiympäristön, jossa voidaan ajaa toista käyttöjärjestelmää emuloidun alustan päällä toisessa käyttöjärjestelmässä”.[8] Tuollaisen kuvauksen jälkeen selvää on ainakin se ettei asia ole selvä. Jonesin (2011, 2) mukaan BOCHS on pikemmin simulaattori kuin virtualisointiohjelmisto, koska se emuloi isäntäkoneelle vierasta laitteistoa. Tämä toteamus muuttaa asian vieläkin kulmikkaammaksi, sillä Verdegemin (2007) mukaan simulointi on aivan eri toimintaa kuin emulointi. Esimerkkinä hän sanoo ettei lentosimulaattori lennä, siinä missä taas emulointiohjelma ei pelkästään vain matki

8 ”Bochs, like QEMU, is a portable emulator that provides a virtualization environment in which to run an operating system using an emulated platform in the context of another operating system.” (IBM/BOCHS)

alkuperäistä ohjelmaa vaan todella on se. Armstrong (2004) arvelee, että emuloinnin ja virtualisoinnin välinen käsite-epäselvyys johtuu siitä että sanaa ”virtuaalinen” käytetään niin monissa yhteyksissä. Vaikka Armstrong puhuukin englanninkielestä, pitää tämä toteamus hyvin paikkansa myös suomenkielen kohdalla. Arkielämässä saatamme todeta että olemme ”virtuaalisesti läsnä” jossakin jne.

Armstrong, joka ohjelmoi työkseen virtualisointiohjelmistoja, toteaa useiden virtuaalikoneiden olevankin tosiasiassa emuloituja. Armstrongin mukaan emulaattorin erottaa virtuaaliohjelmistosta se, että emulaattori toteuttaa kohdeprosessorin toiminnan kokonaan ohjelmallisesti. Armstrong jatkaa että virtualisoinnissa otetaan kohteeksi fyysinen prosessori, ja jaetaan se useiden eri käyttäjien kesken valvojan[9] johdolla. Eri käyttäjät, esim. Macintoshin OS/X ja sen päällä virtualisointiohjelmiston kautta ajettava Windows käyttävät siis samaa prosessoria samalla käskykannalla, josta johtuu nopeuden suuri ero virtualisointi- ja emulointiohjelmistojen välillä. Näin lopputuloksena on että virtualisointi on eri asia kuin emulointi, mutta usein nämä käsitteet menevät sekaisin korkeallakin tietotekniikkateollisuuden komentoketjussa ja aiheuttavat siten paljon sekaannusta.

Armstrong kertoo motivaatiokseen kirjoittaa aiheesta sen, että saa työnsä puolesta usein pyyntöjä, joissa pyydetään että VirtualPC-ohjelmisto koodattaisiin emuloimaan jotain toista prosessoria kuin mikä omassa koneessa on. Pyynnöt osoittavat, ettei käsitteitä hallita kunnolla, sillä virtualisointiohjelmiston lähtökohtana on aina se, että se käyttää koneen omaa fyysistä prosessoria, ei ohjelmallisesti luotua, emuloitua prosessoria. Käsitteiden jatkuva sekoittuminen toisiinsa tulee esiin virtualizationadmin.com palvelussa siten, että VirtualPC kutsutaan ”emulointipaketiksi”.[10] Tämän Armstrong (2004) jyrkästi kieltää. Myös ammattikorkeakoulujen tutkintotöissä käsitteet menevät sekaisin. Purho(2007) toteaa VMWaren tuotteiden mahdollistavan eri käyttöjärjestelmien emuloimisen rinnakkain. Kuitenkin VMWare itse pitää tuotteensa jyrkästi erossa emuloinnista.

Termit sekoittuvat myös KB:n julkaisuissa. Tilanne on outo ja kertoo jotain siitä, kuinka uutta vielä on virtualisoinnin ja emuloinnin hyödyntäminen päivittäisessä tietokoneen-

9 Hypervisor

10 ”This is used by emulation packages like VMware Server, Workstation, Virtual PC, and Virtual Server.” (virtualizationadmin.com/faq)

käytössä. Kun nämä tekniikat yleistyvät, aletaan myös käsitteiden huolimattomaan käyttämiseen kiinnittämään todennäköisesti huomiota.

Hyvä esimerkki ohjelmistosta, jota lähdemateriaalissa usein osittain nimitetään virheellisesti emulaattoriksi on Wine-niminen sovellus, jolla voi ajaa Windowsille kehitettyjä sovelluksia Unix-pohjaisissa käyttöjärjestelmissä. Tekijät ilmoittavat ettei se ole emuloi suoritinta, sillä se ei käännä suorittimelle kirjoitettuja käskyjä, eikä sen tarvitsekaan, sillä Windows ja Unix toimivat samalla suorittimella. (Winewiki 2010.) Esimerkki lähdeaineistossa olevasta virheellisestä nimityksestä löytyy The Digital Preservation Testbedin (DPT) laatimasta raportista ”Emulation: Context and Current Status”, jossa VirtualPC-ohjelmistoa kutsutaan emulaattoriksi vaikka todellisuudessa tämä virtualisoi kuten Armstrong (2004), voimakkaasti argumentoikin. Myös Winestä sama raportti esittää täysin eriävän näkemyksen kuin mitä Winen kehittäjien omat sivut. DPT-ryhmän mukaan Wine on ilmiselvästi emulaattori, mutta sellainen emulaattori, joka ei emuloi laitteistoa vaan käyttöjärjestelmää (DPT 2003, 21).

Kumpaan suuntaan vaakakuppi kallistuu? Onko Winen kehittäjillä liian suppea käsitys emuloinnista, eivätkä he tunne toisen ohjelman ohjelmallista emulointia, vai onko DPT-ryhmällä puolestaan liian laaja näkemys emuloinnista? Onko DPT:n näkemys emuloinnista niin lavea, että sen kautta kaikki tietokoneen toiminta voidaan lukea emuloinniksi tai emuloinnin ulkopuolelle, jos niin tahdotaan? DPT:n julkaisussa emulointi-käsitteen kattama alue määritellään hyvin laajaksi. Sen mukaan emulointia on mikä tahansa teknine ratkaisu, jota käytetään jonkin toisen teknisen ratkaisun paikalla saavuttamaan sama vaikutus kuin alkuperäisellä teknologialla (DPT 2003, 17). Hannu Jaakkohuhdan (2003, 177) kirjoittaman IT-ensyklopedian mukaan emulointi tarkoittaa ”laitteen tai ohjelman kykyä matkia – emuloida – toista vastaavaa toimintoa” ja emulointiohjelma on ”ohjelma, jonka avulla laitteella jäljitellään toista laitetta tai ohjelmaa”. Tästä määrittelystä huomaamme, että emuloinniksi voidaan nimittää loppujen lopuksi jokseenkin mielivaltaisesti melkein mitä tahansa toimintoa. Mikäpä ohjelmisto ei loppujen lopuksi jäljittelisi tai matkisi jotain toista vastaavaa toimintoa? Jopa jäljittelyn ja matkimisen määrittelyn rajat tulevat lopulta vastaan.

Emuloinnille on asetettava sellaiset kriteerit, joiden kanssa tutkimustyötä kyetään mielekkäästi jatkamaan kohteen ollessa tarkoin rajattu. Ilman tarkkaa rajaamista tutkimus

jatkaisi monien sen lähteiden viitoittamaa käsite-epäselvyyttä. Näin on siis perusteltua jatkaa erilaisten määritelmien tutkiskelua siten, että löydetään lopuksi tutkimuksen tarpeita vastaava määrittely.

Määrittelyä rajattaessa tullaan kuitenkin lopulta huomaamaan, että päädyttyämme lopulliseen määrittelyyn se sulkee joissakin lähteissä emulaattoreiksi kutsuttuja ohjelmistoja itsensä ulkopuolelle. Mitä ilmeisimmin on pakon edessä alistuttava siihen, ettei täydellistä kaikkia tarpeita tyydyttävää määritelmää ole löydettävissä, vaan on tyydyttävä siihen, joka sopii oman tutkimuksen tarpeisiin.

DPT (2003, 17) jatkaa emuloinnin määritelmän tarkentamista siten, että se ilmoittaa emulaattorin olevan ”ohjelmisto, joka ajetaan tietokoneessa (isäntäkoneessa), ja joka saa tuon tietokoneen käyttäytymään kuin jokin toisenlainen tietokone (kohdekone)”[11]. Ongelmana tässä määritelmässä on kuitenkin se, että jokaisesta ohjelmistosta voidaan ajatella siten, että se saa tietokoneen käyttäytymään kuin jokin toinen kone. Määritelmä siis sulkisi sisäänsä kaikki ohjelmistot. Esimerkkeinä tästä voisivat olla esimerkiksi tietokoneen kello- ja tekstinkäsittelyohjelmistot. Oleellista onkin kiinnittää huomio sanaan ”tietokone”. Kello- ja tekstinkäsittelyohjelmistot muuntavat kyllä tietokoneen väliaikaisesti kelloiksi ja tekstinkäsittelykoneiksi, mutta ne eivät muuta tietokonetta toiseksi tietokoneeksi. Mutta tässäkin meillä on ongelma, sillä eihän kone fyysisesti muutu toiseksi koneeksi. Niinpä DPT (2003, 18) muotoileekin määritelmän uudelleen muotoon ”emulaattori on ohjelmisto, jota ajetaan tietokoneessa (isäntäkone), ja joka siten virtuaalisesti luo toisen tietokoneen (kohdekoneen)”.[12] Kun tässä virtuaalisesti luodussa tietokoneessa ajetaan jokin ohjelma, tuo ajo on ohjelman ajoa emulaation alaisuudessa.

Ennen lopullista käsitteen lukkoon lyömistä tarkastamme vielä muutaman määritelmän. Gummeruksen Suuren sivistyssanakirjan (Nurmi et al., 2003) mukaan emulaattori on ”ohjelma, jonka avulla tietokone tai oheislaite jäljittelee jotakin toista laitetta, ja on siten käytännössä sen kanssa yhteensopiva”. Emulointi tarkoittaa heidän mukaansa jäljittelyä

---

11 ” An emulator is a program that runs on one computer (the emulator's "host" system) and makes that computer behave like a different computer (the emulator's 'target' system). ”

12 ”An emulator is a program that runs on one computer (the emulator's "host" system) and thereby virtually recreates a different computer (the emulator's "target" system)”.

ja mukauttamista. On mielenkiintoista huomata, että kaikelle kansalle suunnatun sivistyssanakirjan määritelmät ovat yksinkertaisuudestaan huolimatta huomattavasti selkeyttävämpiä kuin ammattikirjallisuuden määritelmät samoille termeille.

Webster's Online Dictionaryn mukaan emuloinniksi kutsutaan tapahtumaa, jossa toinen järjestelmä toimii samalla tavoin kuin toinen.[13] Sanakirja lisää vielä mukaan imitoinnin, vahvistaen näin Rothenbergin selostuksen imitaatiosta emuloinnin parhaimpana esimerkkinä.[14]

MOT Tietotekniikan liiton ATK-sanakirja 5.0 selittää termin näin: ”emuloida = jäljitellä laitteiston toimintaa toisella siten, että toinen laitteisto suorittaa samat ohjelmat, ja tuottaa samat tulokset kuin jäljitelty laitteisto.” Näin tulemme siihen lopputulokseen, että jos samat resurssit (suorittimet, keskusmuistit, jne.) jaetaan useiden käyttäjien (kuten esim. käyttöjärjestelmien) kesken, silloin kyseessä on virtualisointi. Jos taas laitteiston toimintaa jäljitellään ja matkitaan, on kysymyksessä emulointi. Tämän työn sisällä rajaamme kuitenkin emuloinnin koskemaan toimintaa, jossa ohjelmaa ajetaan toisella suorittimella kuin sillä, jolle se on alun perin kirjoitettu. Emulointi tässä työssä tarkoittaa siis suorittimelle x kirjoitettujen käskyjen kääntämistä suorittimelle y, ohjelmiston ajamista 'väärän' suorittimen päällä. Mitä tahansa jäljittelyä ja matkimista emme siis luokittele nyt emuloinniksi, silloin käsitemäärittelymme laajenisi liikaa, ja paisuttaisi työn mittasuhteet yhdelle tutkijalle mahdottomiin mittasuhteisiin.

On myös tärkeää huomata, että emulointia ja virtualisointia ei voi kilpailuttaa keskenään ennen kuin tiedetään mihin tarkoitukseen niitä aiotaan käyttää. Macissä toimii vaivattomasti Windows virtualisointiohjelmiston kautta, sillä nykyisissä Maceissä on täysin sama Intel-prosessori, ja sama käskykanta kuin PC-koneissakin. Näin virtualisointiohjelmisto pystyy suorittamaan Windowsin ajamisen Macissä nopeiten ja tehokkaimmin, ja emuloinnin käyttäminen olisi tuossa tehtävässä hidasta ja kömpelöä. Toisaalta taas, jos halutaan luoda järjestelmä, joka voidaan täydellisesti eristää siitä laitteistosta jossa

13 ”One system is said to emulate another when it performs in exactly the same way, though perhaps not at the same speed.”

14 ”Emulation, imitation of behavior of a computer or other electronic system with the help of another type of computer/system.”

sitä ajetaan, on käytettävä emulaattoria. Jones huomauttaakin juuri tästä syystä emulaattoreita käytettävän usein laitteistoarkkitehtuurien testaamisen välineinä, sillä niiden avulla voidaan testata laitteistoja, joita ei fyysisesti olla vielä valmistettu.

Emulointi on siis hidasta ja syö paljon koneen laskentatehoa, mutta se on ainut tapa käynnistää tulevaisuuden tietokoneessa nykyisen tietokoneen käyttöjärjestelmä ohjelmistoineen. Koska emuloinnin luoma laitteisto on muusta koneesta tiukasti eristetty, ei ole vaaraa koneen muun toiminnan häiriintymisestä emulaattorin takia. Tietokoneohjelmoijat puhuvatkin ”hiekkalaatikosta”, jossa emulaattori toimii (JPC 2007). Emuloinnin suurin etu on, että emulaattoria pyörittävä laitteisto voi olla millainen tahansa, kunhan se on laskentateholtaan tehokkaampi kuin emuloitu laitteisto.

Pitkäaikaissäilytys on sellaisessa vaiheessa, jossa se hakee muotoaan ja se on havaittavissa tutkimuskirjallisuudessa termien rajojen häilyvyytenä. Toisissa kirjoituksissa käsitteet on ehdottomaan sävyyn asetettu tiettyyn kehikkoon, ja tuon kehikon ulkopuolelle meneviä määritelmiä ei hyväksytä. Toisissa taas käsitteet esitetään utuisina ja monitulkintaisina. Omalta osaltani haluan luoda tällä työllä edes jonkinlaista tolkkua tähän käsitteiden sekamelskaan, joka verrattain nuoressa digitaalisen aineiston pitkäaikaissäilytyksessä vielä vallitsee.

5.3 Emulointi säilytysmenetelmänä

Emuloinnin suurimpia ongelmia aineiston säilyttämisen kannalta on, ettei surrogaatti voi koskaan täysin korvata alkuperäistä, eikä siihen kyetä tallentamaan kaikkea alkuperäisen objektin tietosisältöä. Nopeasti ajatellen se voisi olla mahdollista, mutta jatkamalla ajatuskulkua huomaa pian, että olemme hyvin tottuneita lähestymään digitaalisia objekteja fyysisten esineiden kautta. Emuloitu kasettipeli on toki pelinä riittävän samankaltainen vastatakseen alkuperäistä, mutta tuohon peliin on alkuperäisessä kontekstissa liittynyt paljon muutakin kuin vain näytöllä näkyvä pelitoiminta. Pelikasetissa on ollut omat painatuksensa, ja kasettikotelossa kansitaiteensa. Alkuperäisyyttä koemme vasta kun meillä on tuo alkuperäinen kasetti kasettikoteloineen käsissämme, ja suoritamme pelin ajamisen latauksineen siten kuin se tapahtui 80-luvulla. (Deegan & Tanner 2006,

10.) Myös Granger (2000, 2) huomauttaa että maailmassa on paljon aineistoja, joissa aineiston ulkonäkö ja sen välittämä tuntuma saattavat olla yhtä tärkeitä kuin itse tietosisältö.

Meistä tuntuu käsittämättömältä, että menneisyydessä on hävitetty kirjoja, jotka meidän ajassamme olisivat jo rahalliselta arvoltaan korvaamattomia. Tilanne on kuitenkin nyt täysin samankaltainen mitä se oli tuolloin. Tuhoamme paljon korvaamatonta aineistoa, jota tulevaisuudessa tullaan haikailemaan, mutta muutakaan emme voi. Kaikkea ei voi säilyttää, kuten kirjastonhoitajat tietävät. Tuo asia ei muutu digitaalisessa maailmassa miksikään, vaikka tallennuskapasiteetit jatkuvasti moninkertaistuvat. Digitaalisen aineiston kanssa suurimpia ongelmia on se, että arkistonhoitajalla on suuri houkutus säästää sellainen aineisto joka on helposti säilytettävää (Deegan & Tanner 2006, 16).

Emulaation kannalta ongelmallista on se, että siinä on lähes täydellisen säilytysratkaisun ydin, mutta tekijänoikeusongelmat tekevät sen laajamittaisesta, virallisesta kenttätutkimuksesta hyvin vaikeaa tai ainakin suppeaa. Ihanteellisintahan olisi, jos emulaattorit ja niihin tallennetut aineistot voitaisiin luovuttaa vapaaseen jakeluun siten, että ne olisivat vapaassa levityksessä kaikkien saatavilla. Tämä on kuitenkin mahdotonta, sillä vain harva aineisto on vapautettu tekijänoikeuksista. Tosin vapautettujenkin ohjelmistojen määrä on suuri, mutta suhteessa lukkojen takana oleviin määrä on vain pisara valtamerestä. Juuri tämän kysymyksen pikaiseen ratkaisuun olisi suunnattava voimavaroja kaikkien arkistokentällä vaikuttavien instituutioiden ja organisaatioiden taholta. Nyt on oikea hetki siihen. KDK:n loppuraportissa todetaan että emulaation asema migraation rinnalla selviää ajan kanssa hankkeen jatkovaiheissa. Tuohon hankkeeseen on vaikutettava siten, että emulaatio otetaan siinä vakavasti huomioon säilytysmenetelmänä.

5.4 Emulointimenetelmät

5.4.1 Pinottu emulointi

Pitkäaikaissäilytyksen vaatimukset aiheuttavat ongelmia kaikissa ratkaisumalleissa. Tulevaisuutta on mahdotonta ennustaa ja kaikki lupaukset jonkin järjestelmän täysin varmasta toimivuudesta paljastuvat lähemmässä tarkastelussa katteettomiksi. Myös emulointiratkaisun kehitteleminen sisälsi useita välivaiheita. Kun Rothenberg ensimmäisen kerran kirjoitti aiheesta 1995, puhuttiin yleisellä tasolla emulaattoreista menemättä sen kummemmin niiden teknisiin yksityiskohtiin. Asiaa lisää tutkittaessa paljastui kuitenkin se väistämätön tosiseikka, että olisi mahdotonta ohjelmoida erillinen, itsenäinen emulaattori jokaiselle maailmassa käytössä olleelle/olevalle tietokonealustalle. Ei mahdotonta teknisesti, mutta mahdotonta käytännössä, sillä se vaatisi rajattomat resurssit, lukemattoman määrän insinöörityötunteja ja saumatonta yhteistyötä kaikkien tietokonevalmistajien taholta.

Siksi seuraava askel kehityksessä oli ketjutettu- tai pinottuemulointi, jossa alustalle A kirjoitetaan emulaattori A, joka on tehty ajettavaksi alustassa B. Tämä tapahtuu alustan A käyttöiän päättymisen ollessa lähellä. Käyttöön otetaan alusta B. Kun alustan B vanhentuminen alkaa olla käsillä, kirjoitetaan silloin emulaattori B alustalle B, joka on tehty ajettavaksi alustassa C. Käyttöön otetaan alusta C. Kun palautetaan alustassa A ajetussa ohjelmissa luotu aineisto, käynnistetään alustassa C emulaattori B, josta käsin taas käynnistetään emulaattori A, jossa voidaan ajaa alkuperäinen ohjelmisto, jolla aineisto on luotu ja jolla aineisto päästään palauttamaan. Tämä on todistetusti toimiva vaihtoehto, mutta se on myös todella tehoja syövä ratkaisumalli tehden emuloinnista lopulta jokseenkin tehotonta verrattuna käytössä olevaan laskentatehoon. (Rothenberg, 2000, 40.) Siksi tämä lähestymistapa on syytä heti hylätä kannattamattomana.

Tästä seuraava askel emuloinnin kehittämisessä oli emulaatiospesifikaatioiden kirjoittaminen. Ydinajatuksena on, että meillä on jokaista laitteistoa kohden dokumentoituna emulointispesifikaatio, jonka pohjalta emulaattoriohjelma voidaan tulevaisuudessa kirjoittaa sitä tarvittaessa. Tässä lähestymistavassa emme ole sidotut mihinkään tiettyyn ohjelmointikieleen, vaan voimme antaa ohjelmoijalle vapaat kädet valita tulevaisuudessa kullekin laitteistolle sopivan ohjelmointikielen, joka toimii tuossa alustassa tehokkaimmin. Lisäksi spesifikaatioita olisi helpompi kirjoittaa kuin kokonaisia emulaattoreita ja mikä tärkeintä, ei tarvitsisi turvautua paljon laskentatehoa vievään ketjutettuun emulointiin. Mutta vaikka tämä lähestymistapa nopeasti ajateltuna kuulostaa hyvältä, se tarkemmin tarkasteltuna paljastuu tarvittavan työn moninkertaistajaksi ja myös eräänlaiseksi vastuunpakoilemiseksi. Ongelman ratkaisemisen sijaan olisimme vain siirtäneet tuon ongelman tulevaisuuteen ilman mitään takeita siitä, että nyt kirjoittamamme spesifikaatio on tulevaisuudessa riittävä ja mahdollistaa emulaattorin kirjoittamisen. (Rothenberg, 2000, 41.) Työn määrä lisääntyy, sillä emulaatiospesifikaatioiden lisäksi täytyy tulevaisuudessa silti kirjoittaa jokaista uutta tietokonesukupolvea kohden uusi valikoima erilaisia emulaattoreita eri laitteistokokoonpanoja varten. Koska haluamme löytää ratkaisun, joka vaatii mahdollisimman vähän työtä ja maksaa mahdollisimman vähän silti täyttäen kaikki pitkäaikaissäilytystä koskevat vaatimuksemme, hylkäämme tämänkin lähestymistavan.

Rothenberg (2000, 41) kehittää tätä ajatusta eteenpäin ja esittelee meille idean siitä, että olisi olemassa jonkinlainen emulaatiotulkkiohjelma, joka kykenisi tulkitsemaan tiettyä maailmanlaajuisesti sovittua emulaatiospesifikaatiokieltä. Näin riittäisi se, että tuo tulkkiohjelmisto ajoittain päivitetään uudempaan ja modernisoidaan kunkin aikakauden teknologisten vaatimusten mukaisesti. Itse emulaatiospesifikaatiot pysyisivät pätevinä hamaan tulevaisuuteen ja ainut insinöörityötä vaativa kohde olisi itse järjestelmän sydän, emulaatiotulkkiohjelma.

Vielä Rothenberg (2000, 41) kuljettaa ideaansa ja päätyy lopulta emulaatiovirtuaalikoneen konseptiin. Emulaatiovirtuaalikone (EVM) olisi virtuaalikone, joka kyettäisiin melko vähäisellä työllä, muokkaamalla joitakin ohjelmakomponentteja, saamaan helposti yhteensopivaksi uuden alustan kanssa. Rothenberg (2000, 41) esittää että emulaatiospesifikaatiokielen kehittyminen ajan kuluessa ei muodostaisi mitään ongelmaa tälle

ratkaisumallille, sillä tätä uutta kieltä varten kehitettäisiin uusi emulaatiospesifikaatiotulkki, joka toimisi sen aikaisessa emulaatiovirtuaalikoneessa. Uusi tulkki osaisi tulkita myös vanhempaa spesifikaatiokieltä. Kun emulaatiovirtuaalikone itse joutuisi muutoksen alle, tämä uudempi emulaatiovirtuaalikone tehtäisiin sellaiseksi, että se puolestaan osaisi emuloida vanhempaa emulaatiovirtuaalikonetta.

Tässä kohden Rothenbergin tapa ratkaista ongelma vaikuttaa jonkinlaiselta silmänkääntötempulta. Loppujen lopuksihan hän vaihtaa ketjutetun emuloinnin virtuaalikoneiden ketjutettuun emulointiin, olkoonkin että se ehkä veisi vähemmän tehoja kuin toisiaan emuloivat emulaattorit, olisi asetelma kuitenkin aikaa myöden lineaarisesti tehontarvetta kasvattava. Menetelmässä olisi sama heikkous kuin ketjutetussa emuloinnissakin, tuo heikkous vain liikkuisi hieman eri tasolla.

			Sovellus
			Käyttöjärjestelmä A
		Sovellus	Emulaattori 2005
		Käyttöjärjestelmä A	Käyttöjärjestelmä B
	Sovellus	Emulaattori 2005	Emulaattori 2010
	Käyttöjärjestelmä A	Käyttöjärjestelmä B	Käyttöjärjestelmä C
Sovellus	Emulaattori 2005	Emulaattori 2010	Emulaattori 2015
Käyttöjärjestelmä A	Käyttöjärjestelmä B	Käyttöjärjestelmä C	Käyttöjärjestelmä D
Laitteisto 2005	Laitteisto 2010	Laitteisto 2015	Laitteisto 2020

Taulukko 1. Pinottu emulointi. Lähde: Hoeven, Slats,Verdegem & Wijngaarden. 2005.

5.4.2 Komponenttiemulointi

Rothenberg (2000, 44) täydentää emulaatiovirtuaalikoneen konseptia ns. komponenttiemuloinnilla, jossa jokaista tietokoneen komponenttia kohden on yksi emulaattori. Nämä emulaattorit voidaan kytkeä toisiinsa samalla tavoin kuin tietokoneiden komponentit olivat toisiinsa kytkettyinä fyysisessä tietokoneessa. Komponenttiemuloinnissa tuota fyysisten komponenttien keskeistä vuorovaikutusta jäljiteltäisiin ohjelmallisesti siten, että näiden ohjelmallisten komponenttien yhteisvaikutuksen summana olisi samanlainen vaikutus kuin alkuperäisellä fyysisellä laitteistolla.

Tavoitteena on kehittää järjestelmä, joka vaatii mahdollisimman vähän työtä emuloinnin käyttämiseksi pitkäaikaissäilytyksessä. Rothenbergin mukaan tämä varmistetaan siten, että edeltävät emulaatiovirtuaalikoneet ja emulaatiospesifikaatiokielet ovat aina sen jälkeen kehitettyjen emulaatiovirtuaalikoneiden ja emulaatiospesifikaatiokielten tulkittavissa jotta emulointi saadaan suoritettua. Näin jo kirjoitettuja emulaatiovirtuaalikoneita tai emulaatiospesifikaatioita ei tarvitse jälkeenpäin kirjoittaa uudelleen, vaan kehitys kohdistuu aina uusimpaan sukupolveen.(Rothenberg 2000, 8.)

Komponenttiemulaattori toteutui myöhemmin Dioscuri-nimisessä hankkeessa, jonka kehitystä johti 2006-2008 Tessella Support Services Inc. ja siitä eteenpäin KB yhdessä eurooppalaisten pitkäaikaissäilytysprojektien, PLANETSin ja KEEPin kanssa. Ensimmäinen versio tästä komponenttiemulaattorista laitettiin yleiseen jakeluun heinäkuussa 2007 ja nyt tätä kirjoittaessa on projekti edennyt versioon 0.7.0. Dioscuri on julkaistu avoimen lähdekoodin ohjelmistona, ja se on vapaasti kaikkien ladattavissa (Verdegem 2008).

Realisoituneessa tuotteessa on emulointivirtuaalikonetta kehitetty siten, että itse aineistopaketille suoritetun arkistointiprosessin jälkeen ja tuon aineistopaketin ajamiseen käytetyn laitteiston komponenttien emulaattoreiden kehittämisen jälkeen pitkäaikaisorganisaation työ keskittyy emulointivirtuaalikoneen huoltamiseen ajan myötä siten, että se toimii aina uusimmilla alustoilla. Emulointivirtuaalikoneita ei emuloida ketjutetusti, vaan insinöörityö keskittyy kulloisenkin emulointivirtuaalikoneen pitämiseen yhteensopivana aikansa tietokoneiden kanssa. Varsinaisen emuloinnin suorittavat komponentit pysyvät muuttumattomina. Näin ketjutuksen heikkoudesta on ainakin alustavasti päästy

eroon, mutta on muistettava että ratkaisun kestävyys tulee ilmi vasta pidemmän ajan kuluessa. Pitkäaikaissäilytysorganisaatioiden on pysyttävä jatkuvasti tarkkoina ja ajan hermolla, jotta he eivät havaitse jotakin tietotekniikkamaailmassa tapahtunutta muutosta liian myöhään ja siten joudu oman arkistoimisstrategiansa kanssa umpikujaan.

Digitaalisen dokumentin palauttamiseksi tulevaisuudessa Rothenbergin säilytysskenaarion mukaan tulisi seuraavien ehtojen täyttyä: 1) Aineiston bittivirta on säilytetty. 2) Aineiston tarkasteluun tarvittavat ohjelmistot on säilytetty. 3) Emulaatiospesifikaatiot kaikille aineiston tarkasteluun tarvittaville laitteistolle on säilytetty, sekä bittivirtana että varmistuksena myös ihmisen luettavassa olevassa muodossa[15]. 4) Emulaatiospesifikaatiotulkkiohjelma, joka ajetaan emulaatiovirtuaalikoneessa ja joka kääntää kohdassa 3 tallennetun emulaatiospesifikaation, on säilytetty. 5) Emulaatiospesifikaatio jokaiselle emulaatiovirtuaalikoneelle on säilytetty sekä bittivirtana että varmistukseksi myös ihmisen luettavissa olevassa muodossa. Tämä mahdollistaa sen, että uudempi emulaatiovirtuaalikone kykenee emuloimaan vanhempaa. 6) Semanttiset ja syntaksiset spesifikaatiot kutakin emulaatiospesifikaatiokieltä kohden on säilytetty ihmisen luettavissa olevassa muodossa niin kauan kuin tuo spesifikaatiokieli pysyy käytössä olevana. Tämä mahdollistaa emulaatiospesifikaatioiden kirjoittamisen uusille laitealustoille. 7) Kulloinkin käytössä olevan emulaatiovirtuaalikonesukupolven spesifikaatiot on säilytetty ihmisen luettavissa olevassa muodossa. Tämä mahdollistaa emulaatiospesifikaatiotulkkien ajamisen tässä virtuaalikoneessa, ja virtuaalikoneen itsensä siirtämisen uusille laitealustoille. (Rothenberg 2000, 10-11.)

On myönnettävä, että Rothenbergin yksinkertaisena mainostama emulointiratkaisu ei kaikkien hänen julkaisemiensa tehtävälistojensa valossa välttämättä näytä olevan sieltä yksinkertaisimmasta päästä. Hänen ajatuksenjuoksunsa seuraaminen vaatii paikoitellen pinnistelyä, ja silloin herää väistämättä lukijalle myös epäilyksiä hänen ehdottamansa säilytysratkaisun toimivuudesta. Mutta on tärkeää muistaa, että loppujen lopuksi yksin-

15Rothenberg selittää käyttämänsä termin ”ihmisen luettavissa oleva”(human-readable) siten että se tarkoittaa mitä tahansa informaatiota, jonka ihminen pystyy tulkitsemaan ja ymmärtämään minimaalisella vaivalla tietyn ajan kuluessa. Rothenberg toteaa määritelmänsä olevan tahallisen epätarkka ja pitävän sisällään vain ne digitaaliset muodot jotka perustuvat standardeihin määrittelyihin ja sulkevan ulkopuolelleen sellaiset digitaalisen tiedon muodot, jotka perustuvat vanhentuneisiin määrittelyihin, siitä huolimatta että ne olivat joskus standardeja. (Rothenberg 2000, 11.)

kertaista pitkäaikaissäilyttämistä ei ole olemassakaan. Digitaalisen aineiston luonne itsessään on sellainen, että sen säilyttäminen vaatii monivaiheisen säilytysprosessin. Monimutkainen aineisto, jota kaikki digitaalinen aineisto laitteisto- ja ohjelmistoriippuvaisuutensa tähden on, vaatii väistämättä monimutkaisen säilytysratkaisun. Rothenberg (2000) selostaa yleistajuisesti kohta kohdalta kuinka emulointiin perustuva säilytysratkaisu toimisi. Ratkaisu jakaantuu 9 erilliseen vaiheeseen, joista jokainen täydentää edellistä kerryttävästi siten, että jos jossakin vaiheessa havaitaan virhe, voidaan palata takaisin.

1) Huolehditaan siitä, että alkuperäinen aineisto tarkasteluunsa tarvittavine ohjelmistoineen ja niiden spesifikaatioineen kopioidaan uudelle medialle suojaan korruptoitumiselta.

2) Huolehditaan siitä, että kaikki aineiston luonnetta selittävä metadata pysyy ymmärrettävänä. Tosiasia on, että metadata saattaa aikojen kuluessa vaatia konvertointia. Ainakin metadatasta tulisi konvertoida sen ylimmät kerrokset siten, että ylempien kerroksien jälkeen alla olevat kerrokset säilyvät ymmärrettävinä. Vielä vaatimuksena on, että näiden konvertointien tulisi olla peruutettavissa siten, ettei metadataa kyetä konvertointitoimenpiteellä pysyvästi korruptoimaan.

3) Huolehditaan siitä, että aineiston, emulaattorispesifikaatioiden, emulaattorispesifikaatiokielten, emulaattorispesifikaatiotulkkien ja emulaattorivirtuaalikoneenspesifikaatioiden välillä olevat linkitykset on määritelty oikein ja että ne toimivat oikein.

4) Tallennetaan kaikki emulaatiospesifikaatiotulkit kaikille emulaatiospesifikaatiokielille, joita on käytetty DSEP:n kokoelmissa olevien aineistopakettien emulointiin tarvittavien emulaattoreiden määrittelyyn. Tämän otaksutaan olevan jokseenkin kevyt vaihe, sillä spesifikaatiokielet tuskin vaihtuvat kovin usein. (Rothenberg 2000, 13.)

5) Huolehditaan siitä, että kun uusi emulaattorispesifikaatiokieli kehitetään, varmistetaan että emulaattorispesifikaatiotulkki kehitetään sille samalla. Tämä emulaattorispesifikaatiotulkki viedään AIP:nä DSEP:hen. (Rothenberg 2000, 13.)

6) Huolehditaan siitä, että kun uusi emulaatiovirtuaalikone kehitetään, aikaisemmasta emulaatiovirtuaalikoneesta kirjoitetaan emulaattorispesifikaatio, jotta uudempi emulaa-

tiovirtuaalikone kykenee ajamaan kaikki aikaisemmat emulaatiovirtuaalikoneet. Tämäkin emulaattorispesifikaatio viedään AIP:nä DSEP:hin. (Rothenberg 2000, 13.)

7) Huolehditaan siitä, että sekä emulaattorispesifikaatiokielestä ja emulaatiovirtuaalikoneesta on olemassa ihmiselle luettavissa olevassa muodossa dokumentaatio (Rothenberg 2000, 13).

8) Varmistetaan, että emulaatiovirtuaalikone on aina ajettavissa jollakin alustalla, joka on saatavilla DSEP:issä (Rothenberg 2000, 13).

9) Kirjoitetaan ihmiselle luettavissa olevaan muotoon ohjeet siitä, kuinka emulointia käytetään vanhojen aineistojen palauttamiseen ja kuinka emulointi-infrastruktuurista pidetään huolta, jotta se pysyy käyttökelpoisena myös tulevaisuudessa. (Rothenberg 2000, 13.)

Digitaalisen aineiston säilyttämiseen liittyy paljon erilaisia näkökulmia, johtuen siihen liittyvän teknologian ja itse digitaalisen aineiston monimutkaisuudesta. Digitaalisen aineiston säilyttämisen suhteen tapahtui 90-luvun ja 2000-luvun alun voimakkaiden erimielisyyksien jälkeen yhteensulautumista. Emuloinnin kiihkeimmät puolestapuhujat joutuivat myöntämään, että ilman migraatiota ei digitaalisen aineiston kanssa voida vakavissaan puhua säilyttämisestä johtuen digitaalisten tiedostomuotojen ja tallennusmedioiden jatkuvasta kehittymisestä. Migraatiota on pakko suorittaa digitaalisen aineiston kanssa tai muuten sitä ei kerta kaikkiaan pysty arkistoimaan.

Mutta myös emulointiin alun perin halveksivan sävyisesti suhtautuneet tahot joutuivat myöntämään, että on olemassa lukuisia aineistoja, joita ei voi mielekkäästi säilyttää lainkaan ilman emulointia. Aineistoja, joiden kohdalla voidaan sanoa että pelkkien merkkijonojen arkistoiminen ei todellisuudessa säilytä yhtään mitään. Tällaisia ovat esim. tietokonepelit ja muu multimedia-aineisto, jossa emulointi on ainut tapa saada aineisto näkyviin siinä muodossa, jossa tuon ajan ihmiset sen näkivät.

5.4.3 Universal Virtual Computer

Kärjistetyt näkemykset emuloinnista ja migraatiosta tiivistyivät lopulta Raymond Lorien 2002 julkaisemaan UVC:n (Universal Virtual Computer) konseptiin. UVC:n ydinajatus on kehittää täysin laitteistoista riippumaton käyttöjärjestelmä, joka toimii millä tahansa tietokonealustalla ja pystyy ajamaan kaikkia tietokoneohjelmia mitä koskaan on kirjoitettu. Idea kuulostaa nopeasti ajateltuna utopistiselta ja peräti mahdottomalta. Kun puramme idean osiin, alkaa vaikutelma olemaan toisenlainen.

UVC perustuu tietokonearkkitehtuurin osiin, jotka ovat olleet olemassa ensimmäisistä tietokoneista lähtien. Niistä voidaan suurella varmuudella olettaa, että ne tulevat pysymään muuttumattomina vastaisuudessakin. Meidän on itse asiassa vaikea ymmärtää, kuinka tietokoneita voisi olla ilman näitä tietokoneajan alusta saakka käytössä olleita tietokonearkkitehtuurin osia. Tällainen perusarkkitehtuuri on esim. von Neumann-arkkitehtuuri, jonka mukaelmia kaikki käyttämämme tietokoneet ovat.

UVC:n käyttämiseen pohjautuvan arkistointiratkaisun ytimessä on ajatus aineistojen dekoodaamisesta uuteen loogiseen formaattiin, joka on tulevaisuuden ohjelmoijille helposti ymmärrettävissä (Oltmans &Wijngaarden, 2004, 24). Aineistoa ei siis esikäsittelytoimenpiteenä pelkästään migroida, vaan se myös konvertoidaan. Rothenbergin henkilökohtainen voimakas panostus UVC:hen kertoo paljon hänen omasta paradigman muutoksestaan. Vielä muutama vuosi ennen UVC-konseptin esittelyä oli Rothenbergin julkaisuissa havaittavissa ehdottoman kieltävää suhtautumista migroimiseen ja konvertoimiseen. Nyt hän on kuitenkin valmis hyväksymään ne molemmat osaksi arkistointiratkaisuaan. Samoin on havaittavissa käsitysten loiventumista emulointia kohtaan migraatioon perustuvien arkistointiratkaisujen kannattajien julkaisuissa. Puolin ja toisin on siis jouduttu myöntämään se tosiasia, ettei ilman toisiaan täydentäviä menetelmiä voida digitaalisia aineistoja koko laajana kirjonaan luotettavasti ja kattavasti arkistoida.

Ydinajatuksena UVC-konseptissa on se, että arkistoimme haluamamme aineiston kanssa ohjelman P. Ohjelma P dekoodaa tuon aineiston loogiseen muotoon tulevaisuuden asiakasohjelmalle. Samaan pakettiin arkistoidaan myös loogisen muodon määritelmä yhdessä ohjelma Q:n kanssa. Ohjelman Q tarkoitus on palauttaa tuon loogisen muodon

määritelmä, johon aineisto on dekoodattu. Se on loogisen muodon looginen muoto, metatiedon metatieto.

Varsinaisen työn hoitaa UVC, jolle ohjelmat P ja Q on kirjoitettu. Tulevaisuuden tietokone emuloi UVC:tä, joka ajaa ohjelmat P ja Q ja palauttaa niiden avulla aineistosta helposti luettavan loogisen muodon. Tämä muoto muistuttaa hyvin paljon XML-muotoa. (Lorie 2002, 1-5.)

Ensimmäiseksi UVC:n kokeiluaineistomuodoksi valittiin PDF sen staattisen olemuksen vuoksi. On oleellista huomata, että ajettavien aineistojen ongelmaa ei ole ratkaistu, eikä UVC:tä voi vielä suositella arkistointimekanismina organisaatioille ja instituutioille yleisesti. Toistaiseksi UVC on siis kyvytön arkistoimaan edes kaikkein yksinkertaisinta tietokoneohjelmaa. UVC-emulaattori pitää nykyisessä versiossa vain huolen siitä, että staattisista aineistoista kuten PDF-dokumenteista ja kuvista pystytään tulevaisuudessa palauttamaan helposti luettavissa ja tulkittavissa oleva looginen muoto. (Lorie 2002, 3.)

Näihin staattisiin tiedostoihin, kuten PDF- ja JPEG-formaateissa oleviin tiedostoihin, UVC kohdistaa konversio-ohjelman, joka dekoodaa alkuperäisen datan loogiseen formaattiin. Tämä looginen formaatti on tulevaisuudessa huomattavasti helpommin ymmärrettävissä kuin alkuperäinen muoto. Tuo konversio-ohjelma kirjoitetaan tänään, mutta ajetaan vasta tulevaisuudessa UVC-emulaattorissa, joka ohjelmoidaan silloin käytössä oleville laitteistoille. Jos arkistoitu aineisto on sovellus (joita UVC ei vielä kykene arkistoimaan), nykyiselle tietokoneelle kirjoitetaan UVC-emulaattori nyt ja tulevaisuudessa kirjoitetaan toinen UVC-emulaattori sen hetkiselle tietokoneelle ja näin menneisyydessä kirjoitettu emulaattori saadaan ajettua tulevaisuudessa. Suurena etuna tässä on se, ettei tulevaisuudessa tarvitse kirjoittaa emulaattoreita menneisyyden laitteistoille. Siten säästytään paljon resursseja vaativalta takaperoiselta insinöörityöltä. (Lorie 2002, 6.)

Eri arkistointiratkaisujen mahdollisen menestyksen arvioimisessa pitkäaikaissäilytyksen onnistumisen suhteen oleellista on se, millaisia toimenpiteitä eri arkistointiratkaisut vaativat meiltä nyt. Mitä vähemmän ne vaativat, sitä parempi. Toisaalta, jos arkistointiratkaisut perustuvat siihen, että ongelmat siirretään tulevaisuuteen, silloin ne ovat selkeästi

kääntyneet itseään vastaan. Se tarkoittaisi käytännössä sitä, että olemme vain siirtäneet ongelmat tulevaisuuden ihmisten harteille.

UVC:hen perustuva arkistointiratkaisu vaatii meiltä arkistointihetkellä tiettyjä toimenpiteitä. Meidän on määriteltävä säilytettäville aineistoille loogiset skeemat, jotka sopivat noille aineistolle parhaiten. Ohjelma, joka suorittaa aineiston dekoodauksen tekee sen skeeman perusteella. Dekoodauksen tuloksena syntyy hyvin XML:n kaltaista tunnisteisiin perustuvaa tekstiä, jonka aineiston tarkastelijan tulevaisuudessa on todennäköisesti helppo ymmärtää. Tämä tosin koskee vain sellaisia aineistoja, joiden rakenne on muutoinkin yleisesti tunnettu. Tilanne on kokonaan toisenlainen, jos aineistoluokka on sellainen, ettei siitä ole yleistä tuntemusta suuren yleisön keskuudessa. Koska tämä tilanne on väistämätön monien aineistojen edessä, aineistojen käyttäjä tulee tarvitsemaan ohjausta loogisten rakenteiden merkityksien tulkitsemisen kanssa. (Lorie 2002, 8.)

Tarvitaan myös kuvaus aineiston sisäisestä rakenteesta, jotta dekoodausohjelma osaa dekoodata aineiston uuteen loogiseen formaattiin. Muutoin dekoodaus olisi mahdotonta ja tieto siten menetettäisiin. Dekoodausohjelman täytyy siis tuntea sen materiaalin luonne jota se dekoodaa ja siksi se pitää varustaa tuolla tiedolla. Dekoodausohjelma on se ohjelma, jonka UVC-emulaattori tulevaisuudessa ajaa. Meidän on ohjelmoitava se ja testattava huolellisesti sen toiminta valmistamassamme UVC-emulaattorissa, jonka kirjoitamme emuloimaan käyttämäämme laitteistoa.

Ennen kuin menemme syvemmälle UVC:n toimintaan ja rakenteeseen, on syytä tehdä selväksi kahden toisiinsa hyvin helposti sekoittuvan käsitteen ero. UVC:tä käsittelevässä tutkimuskirjallisuudessa toistuu lyhenne LDV. On tärkeää huomata, että tämä lyhenne voi tarkoittaa kahta asiaa, jotka eivät saa sekoittua toisiinsa tai muutoin UVC-arkkitehtuurista tulee täysin käsittämätön. UVC-arkkitehtuurissa on erikseen Logical Data View ja Logical Data Viewer. Ensimmäinen tarkoittaa uuteen formaattiin dekoodattua aineistoa ja jälkimmäinen tarkoittaa sovellusta, jolla tuota dekoodattua aineistoa tarkastellaan. Joissakin julkaisuissa jälkimmäisestä käytetään vain nimitystä viewer, jotta se ei sekoittuisi ensimmäiseen. (Diessen, Hoeven & Meer 2005.)

Suomenkielen etuna on, että se mahdollistaisi helpon termien erottamisen toisistaan. Ainakaan toistaiseksi ei UVC-arkkitehtuuria ole kuitenkaan vielä suomennettu, eikä muutoinkaan suomalaisissa organisaatioissa tai pitkäaikaisselvityksissä huomioitu.

Käsittelen asiaa käänteisesti, ensin tulevaisuudesta käsin. Tämä käänteinen järjestys auttaa paremmin hahmottamaan mihin kutakin UVC:n moduulia tarvitaan, ja mitkä niiden roolit ovat pidemmällä tähtäimellä. Tulevaisuudessa tietokoneohjelmoijalla on ensimmäisenä tehtävänä ohjelmoida UVC (Universal Virtual Computer), universaalivirtuaalitietokone, tuona aikana käytössä olevalle tietokonealustalle (Diessen ym. 2005, 199).

UVC:n spesifikaatio, rakennuspiirustukset, löytyvät tallennettuina useisiin eri digitaalisiin muotoihin ja näiden tallenteiden pettämisen varalta myös mikrofilmille ja metallilevylle kaiverrettuna. Tätä metallilevyä voisi verrata Rosettan kiveen, jonka ranskalainen Bouchard löysi Rosettan kaupungista (nyk. Rashid) (The British Museum 2011). Tämän kiven avulla aikaisemmin käsittämättömät hieroglyfit saivat nopeasti merkityksensä. Rothenberg onkin ehdottanut UVC:n spesifikaation tallentamista digitaalisten medioiden lisäksi myös metallilevyille kaiverrettuina siltä varalta, että tulevaisuudessa kaikki tallennusvälineemme pettävät. Tärkeää on pitää huolta, että tieto siitä miten UVC rakennetaan säilyy aivan varmasti tulevaisuuteen.

UVC:n rakentamisen jälkeen on ohjelmoijan kirjoitettava vielä toinen sovellus, Logical Data Viewer. LDV:n rooli aineiston palautusprosessissa on hyvin merkittävä sillä se ohjaa koko tuota prosessia. LDV:n tarkkaa ohjelmakooditason rakennetta emme pysty sanomaan, sen määrittelee tulevaisuudessa käytössä oleva laitteisto. LDV kontrolloi kaikkea UVC:n ja sen moduulien välillä tapahtuvaa tietoliikennettä. LDV käynnistää UVC:n, lähettää enkoodatun datan ja formaattidekooderin sille ja vastaanottaa takaisin Logical Data View -tunnisteet. Näiden tunnisteiden ja loogisessa data skeemassa määriteltyjen tunnisteiden merkityksien avulla pystyy LDV rekonstruoimaan arkistoidun aineiston. (Diessen ym. 2005, 199.)

Palaamme nyt ajassa takaisinpäin siihen hetkeen, jolloin aineistopaketit kapseloitiin arkistoitavaksi. Mitä aineistokapseliin sisällytettiin? Kapseliin sisällytettiin kolme aineistopakettia: spesifikaatio UVC:n rakentamiseen, looginen tietomalli ja UVC dekooderi. Looginen tietomalli on kuvaus siitä millaisista tunnisteista LDV koostuu ja mitkä ovat

niiden keskinäiset suhteet. Ilman skeemaa ei LDV (viewer) pysty rakentamaan digitaalisen objektin rekonstruktiota, sillä se ei tiedä mitä LDV:ssä esitetyt tiedot merkitsevät.

UVC-dekooderi voidaan ja se tuleekin kirjoittaa aineiston arkistoimisen yhteydessä. Tätä tehtävää ei voida jättää tulevaisuuteen, sillä tulevaisuudessa, ainakin tarpeeksi pitkälle mentäessä, syntyy vaikeuksia ymmärtää alkuperäisten digitaalisten objektien välittämiä merkityksiä. Tärkein metatieto, joka arkistoidun aineiston mukana toimitetaan, on dekooderi. Dekooderi kääntää aineistokapselissa olevat aineistopaketit LDV-muotoon. Luotettavan arkistoinnin vaatimuksena on, että dekooderia testataan ennen aineistopaketin siirtämistä pitkäaikaissäilytykseen siten, että käytössä olevalle tietokonealustalle ohjelmoidaan UVC. Sitten aineistopaketti ja dekooderi ajetaan UVC:llä ja todetaan toimivuus. Jos UVC-emulaattori rakennetaan tulevaisuudessa aineistossa toimitettujen spesifikaatioiden mukaisesti, se suurella varmuudella antaa tulokseksi samanlaisen LDV:n kuin tänäänkin (Oltmans & Wijngaarden 2004, 3-4).

Pitkäaikaissäilytysjärjestelmien keskeisenä tehtävänä näiden toimenpiteiden jälkeen on säännöllisesti tarkistaa aineistokapseleissa olevat aineistopaketit eheyden ja korruptoimattomuuden varmistamiseksi. Sen tulee myös suorittaa laadunvalvontaa pistokokeiden muodossa siten, että kapseloiduista aineistopaketeista tehdään välillä palautusajoja varmistukseksi arkistoimisen onnistumisesta.

UVC:n ulkopuolelle jää toistaiseksi kaikki se aineisto, jonka kohdalla Rothenberg kuulutti emuloinnin välttämättömyyttä. Tämä on merkittävä puute. UVC ei siis vielä kykene palauttamaan yksinkertaisintakaan tietokoneohjelmaa. Voidaan sanoa, ettei se vielä kykene emuloimaan. Ei ole kuitenkaan nähty syitä siihen, miksi se ei siihen pidemmän kehitystyön jälkeen kykenisi. Vasta sitten UVC:tä voi alkaa markkinoimaan pitkäaikaissäilytysratkaisuna, kun se todella kykenee siihen, mistä emulointia alunperinkin mainostettiin. Siihen, että se pystyy säilyttämään enemmän ja paremmin kuin esimerkiksi pelkkä migraatio. Muutoin sen suosimisessa ohi jonkin toisen ratkaisun olisi kyse vain tiettyjen teknisten mieltymysten suosimisesta.

5.5 Emuloinnin käytettävyys tulevaisuudessa

Keskeinen huoli digitaalisen aineiston palauttamisessa emuloimalla on se kuinka varmistamme, että tulevaisuuden ihmiset osaavat käyttää ohjelmistojamme, joissa käyttäjää saattaa tervehtiä pelkkä kehotteen edessä vilkkuva kursori. Meillä vielä on kokemusta MS-DOSista ja muista vanhoista järjestelmistä, mutta meidän mukanamme häviää myös tietotaito käyttää niitä. Vastaus saattaa löytyä tekniikasta jota on toimisto-ohjelmistoissa käytetty jo pitkään, makronauhotteista.

5.5.1 Makronauhoitukset

Cornellin (2011, 1) mukaan makronauhoittaja on ohjelmisto joka ottaa talteen joukon käyttäjän suorittamia toimintoja siten, että ne voidaan toistaa myöhemmin. Ajatuksena on että me, jotka arkistoimme aineistoa tulevaisuuden sukupolville, täydennämme niiden metatietoja makronauhotteilla, jolloin heidän ei tarvitsisi ohjelmaa käyttääkseen muuta kuin käynnistää ohjelmiston mukana tulleita makronauhotteita. Hoevenin, Rechertin, Schröderin ja Suchodoletzin (2010) mukaan markkinoilla olevien makronauhoittajien ongelmana on se, etteivät ne tee keskenään vaihtokelpoista standardoitua jälkeä. Pitkäaikaissäilytyksen kannalta ensimmäinen ja merkittävin askel olisi luoda yhteinen kaava, jota kaikki tallennetut makrot noudattaisivat. Näin ne olisivat maailmanlaajuisesti vaihtokelpoisia. Toisena suurena ongelmana on, että makronauhoitusohjelmat ovat jokseenkin tietoisia ympäristöstä jossa ne toimivat, mutta emulaattoreiden suhteen näin ei ole. Hoevenin ym. ehdotus asiaan on, että makronauhoitteissa käytettäisiin hyväksi eräänlaisia visuaalisia synkronointikohtia, esimerkiksi ruudunkaappauksia hiiren osoittimen ympäriltä ennen ja jälkeen suoritetun toiminnon (Hoeven ym. 2010, 5). Makroa toistettaessa emulaattori vertaisi näitä otoksia suoritettavaan prosessiin.

Valmiiden makronauhoitusohjelmien kanssa on kuitenkin sellainen ongelma, etteivät ne ole tietoisia emulaattorin sisäisestä toiminnasta. Ne tallentavat vain koneellisesti käyttäjän antamat komennot, hiiren liikkeet ja muut toiminnot. Tästä saattaa seurata se, että makronauhoite toimii väärin. Asiaan on esitetty ratkaisuksi aikaan sidottua toimintojen ohjausta siten, että mitataan aika joka kuluu eri toimintojen väliseen käynnistämiseen, nauhoitetaan se makroksi ja toimitaan sen mukaisesti.

5.5.2 Näkymäpolut

Tosiasia on että digitaaliset objektit eivät koskaan ole olemassa tyhjiössä. Kaikilla ohjelmistoilla on riippuvuuksia ja kytköksiä, joita ei itse ohjelmakoodia tutkimalla pystytä kovin helposti päättelemään. Ainakin sellainen päättelytyö vaatisi työmäärän johon millään yhteiskunnalla/instituutiolla ei ole varaa. Pohtiessaan digitaalisten aineistojen kontekstisuuden ongelmaa Lorie (2002) piirsi alustavasti suuntaviivoja, joista PLANETS-projektissa työskennelleet Hoeven ja Suchodoletz (2009) myöhemmin hahmottelivat näkymäpolkujen käsitteen.

PLM:n pohjalta voidaan tehdä erilaisia kombinaatioita, joita Hoeven ja Suchodoletz (2010) kutsuvat näkymäpoluiksi. Näkymäpolku on Hoevenille ja Suchodoletzille digitaalisen objektin tiedostoformaatista piirretty viiva ohjelmistoihin ja laitteistoihin, joita objektin tarkastelemiseen tarvitaan. Se on yksinkertaistettu malli arkistoidun aineiston riippuvuussuhteista. Ilman tällaisia malleja on olemassa mahdollisuus että aineisto jää kellumaan tyhjiöön, jolloin tulevaisuudessa olisi pakko turvautua kalliiseen ja paljon työtä vaativaan salapoliisityöhön, jossa ohjelmiston riippuvaisuuksia pyrittäisiin selvittämään tutkimalla koodia itsessään. Jokainen sovellus tarvitsee tietyn käyttöjärjestelmän toimiakseen ja jokainen käyttöjärjestelmä taas toimii jonkinlaisen laitteiston päällä.

View Path perustuu Diessenin hahmottelemaan PLM:ään. PLM:n tehtävänä on kuvata graafisesti millainen suhde samankaltaisilla objekteilla, esimerkiksi tietyssä formaattimuodossa olevilla tiedostoilla, on ympäristöönsä. Hoeven ja Suchodoletz hioivat tuota mallia 2009 ja nyt PLM-mallissa on kolme tasoa: sovellustaso, käyttöjärjestelmätaso ja laitteistotaso. PLM-malli ei ole kuitenkaan lukittu näihin tasoihin ja tarpeen vaatiessa, tai niin hyväksi nähdessään, voi kukin organisaatio muokata siitä omia tarpeitaan vastaavan.

Näkymäpolut ovat hyvin yksinkertaisia, mutta ajan myötä riippuvaisuudet muuttuvat laitteistojen ja ohjelmistojen vanhentuessa. Näin näkymäpolkua on päivitettävä siten, että vanhentuneen elementin korvaa jokin toinen saman asian hoitava elementti. Suchodoletzin ja Hoevenin2009, 148) mukaan on kolmenlaisia tilanteita, joissa näkymäpolut ovat ahtaalla. Tietyssä ajantaitteessa saattaa jollekin objektille olla olemassa vain yksi toimiva näkymäpolku, ja toisena taas saattaa olla useita vaihtoehtoisia näkymäpolkuja.

Saattaa myös käydä niin, että objekti ajautuu palauttamistoiminnan ulkopuolelle, sillä kaikki näkymäpolut ovat lakanneet olemasta valideja. Tässä kohdin tarvitaan emulaattoreita pitämään näkymäpolut toiminnallisina. Jos näkymäpolkuja on vain yksi, on tapaus silloin selvä. Ongelmallisin näistä kolmesta on se että näkymäpolkuja on useita yhtä objektia kohden. Silloin nimittäin säilytysjärjestelmään väistämättä tarvitaan jonkinlainen valintaproseduuri, jonka tehtävänä on löytää kaikkein käyttökelpoisin ja mielekkäin näkymäpolku. Hoevenin ja Suchodoletzin (2009, 149) vastaus tähän on varustaa PLM mittalaajennoksella. Mittalaajennos laskee näkymäpoluille painoja riippuen siitä kuinka onnistuneesti kukin niistä tuo esille objektin autenttisuuden, ja kuinka tarkan esityksen alkuperäisestä se tuottaa. Näkymäpolku varustettaisiin siis metatietokerroksella, joka sisältäisi tiedot siitä miten ja millä perustein näkymäpolkuja on painotettu. Näin näkymäpolut mahdollistaisivat eri näkymäpolkujen vertailun valittujen kriteereiden perusteella. Tämä mahdollistaisi näkymäpolkujen arvioimisen siitä näkökulmasta mitkä niistä ovat kustannustehokkaimpia, helppokäyttöisimpiä, alkuperäiselle aineistolle uskollisimpia ja ehkäpä vielä että mitkä näkymäpolut ottavat huomioon käyttäjän mieltymykset. Vielä Hoeven ja Suchodoletz ehdottavat että näkymäpolut voisi altistaa käyttäjien arvioinnin alle siten, että nämä voisivat kirjoittaa niistä omia mielipiteitään ja kokemuksiaan, arvostelujaan. Näin näkymäpolkuja voisi avainsanoittaa samoin kuin nykyään pystytään tekemään kirjastojen nettisivuilla eri kirjoille.

Koska kunkin näkymäpolun pitäminen toiminnallisena aiheuttaa väistämättä kustannuksia, sillä järjestelmässä joudutaan pitämään huolta siitä että kukin näkymäpolku komponentteineen on varmasti toimiva, saattaa näkymäpolkujen testaamisesta olla hyötyä niiden ylläpitämisen mielekkyyden arvioimisessa. Joissakin tapauksissa saattaa olla kustannustehokkainta yhdistää useampi näkymäpolku yhdeksi, jos tämä ei tee aineiston esittämiselle väkivaltaa. (Hoeven & Suchodoletz 2009, 150.) Hoeven & Suchodoletz (2009, 152) ehdottavat että toimiviksi ja kustannustehokkaiksi havaittuja näkymäpolkuja voitaisiin vaihtaa muistiorganisaatioiden kesken.

Digitaalinen arkisto saattaa joutua hylkäämään arkistoitavaksi tarjotun aineiston vastaanottomoduulissa, jos se ei pysty takaamaan tälle varmasti toimivaa näkymäpolkua. Vastaanotossa on suoritettava aineistolle identifiointia ja luokittelua. Tähän on olemassa työkaluja, mutta ei vielä sellaisia jotka ottaisivat huomioon kaikki tietotekniikkaan liit-

tyvät riippuvaisuudet, ne eivät siis vielä analysoi aineistoja emuloinnin näkökulmasta. Näitä työkaluja tulisi laajentaa ja täydentää PLM:llä. (Hoeven & Suchodoletz 2009, 150.)

5.5.3 Emulaatiota selaimen kautta

Hoeven ja Suchodoletz (2009, 152) ehdottavat, että eri aineistojen emulointiin erikoistuisivat ihan omat tahonsa, jolloin tavallisten käyttäjien ei tarvitsisi niinkään tietää emuloinnin teknisistä yksityiskohdista, emulaattoreiden asetuksista ja muista vanhaan tekniikkaan liittyvistä detaljeista. Aineistoja selaava näkisi emuloinnin lopputulokset virtuaalipäätteen kautta omalta tietokoneeltaan, ihanteellisimmassa tapauksessa selaimensa kautta. Näin aineistojen käytöstä tulisi sijainnista riippumatonta eikä emulointi jäisi vain arkistojen tutkijansaleihin. Käyttäjä ei tarvitsisi mitään erityisiä ohjelmistoja tai muuta järjestelyjä. Tämä mahdollistaisi myös emulointipalvelujen keskittämisen siten, että eri muistiorganisaatiot voisivat tämän keskittymisen avulla jakaa työkuormaansa ja omaa erityisosaamistaan eri alueilla. Selvää on että aineistojen palauttaminen emuloimalla johtaisi siihen, että eri muistiorganisaatioilla olisi eri aineistoalueita, joiden emuloimisesta juuri heillä on erityisesti kokemusta. Kaiken tämän tietotaidon kerääminen yhden keskitetyn tahon alle olisi järkevää ja hyödyttäisi mahdollisimman paljon muita. Lisäksi vältyttäisiin laillisuuskysymysten aiheuttamilta ongelmilta, sillä aineistoja ei tarvitsisi ladata käyttäjien koneille, vaan ne pysyisivät tämän tahon palvelimilla ja käyttäjä näkisi vain lopputuloksen. (Hoeven & Suchodoletz 2009, 152.)

Freiburgin yliopistolla ja Koninklijke Bibliotheekilla onkin meneillään yhteinen projekti etäemulointipalvelujen kehittämiseksi.

5.6 David Bearmanin näkemyksistä

Emuloinnin ympärillä on alusta alkaen käyty ajoittain kiivasluonteistakin keskustelua sen tarkoitusperistä. David Bearman (1999) otti voimakkaasti kantaa Rothenbergin emulointivisioihin sen jälkeen kun tämä oli julkaissut aihetta käsittelevän artikkelinsa

1999. Koska Bearman on suomalaisessa(kin) informaatiotutkimuksessa jokseenkin korkealle arvostettu kirjoittaja ja vaikuttaja, katson tarpeelliseksi tuoda esille oman näkemykseni Bearmanin emulointiin liittyvistä kannanotoista.

Bearmania siteerataan paljon suomalaisen informaatiotutkimuksessa ja hänen lausahduksiaan näkee usein luennoitsijoiden esityksissä, varsinkin hänen kuuluisinta väitettään siitä, että Rothenberg pyrkii säilyttämään digitaalisesta aineistosta väärät asiat keskittyessään niiden toiminnallisuuteen itse sisällön sijasta (Bearman 1999, 1). Koska emulointitutkimus on edennyt valtavasti vuodesta 1999, jolloin Bearman otti vielä kesken olevaan asiaan voimakkaasti kantaa, on siinäkin mielestäni syytä kyseenalaistaa hänen emuloinnista kirjoittamansa asiat, sillä ne ovat sisäisessä ristiriidassa ja ohittavat monia tosiasioita, kuten aion nyt osoittaa.

Bearmanin mukaan Rothenberg keskittyy säilyttämään väärää asiaa keskittyessään informaatioteknologian toiminnallisuuden säilyttämiseen itse elektronisen aineiston sijasta. Lisäksi hän antaa ymmärtää, että Rothenberg tarjoaisi jonkinlaista "yksi lääke kaikkiin vaivoihin"-ratkaisua. Rothenbergin julkaisuista käy selvästi ilmi, että hän tiedostaa tämän tulkintavaaran ja siksi onkin omituista, että David Bearman syyttää häntä viisasten kiven etsimisestä ja tarjoamisesta digitaalisten aineistojen pitkäaikaissäilytyksen ratkaisuksi (Bearman 1999, 1). Syynä saattaa olla se, että kun Rothenberg julkaisi raporttinsa ”Avoiding Technological Quicksand”, oli Rothenberginkin käsitys emuloinnista pitkäaikaissäilytysongelman ratkaisuna vielä lapsenkengissään, eikä hänen silloin julkistamansa raportti tuonut esiin emuloinnin kaikki ongelmia.

Silti ei voida perustellusti väittää että Rothenberg olisi jotenkin kaunistellut tai pyöristellyt ongelmaa, saatikka ehdottanut siihen ratkaisuksi mitään sellaista, joka voidaan tulkita jonkinlaiseksi helppoheikin myymäksi ihmelääkkeeksi. Bearmanin mukaan Rothenbergin tarjoama emulointi ei riittävästi ota huomioon digitaalisten aineistojen ongelmia, eikä voi toimia. Lisäksi hän toteaa että koko emulointiratkaisun tarjoaminen saattaa olla kulttuuriperinnöllemme jopa haitallista, sillä se voi rohkaista päättävässä asemassa olevia henkilöitä kannattamaan sen käyttöönottoa ja siten rohkaisemaan vaarallista toiveajattelua.

Bearman aloittaa väittämällä että Rothenbergin julkaisujen johdosta monet organisaatiot olisivat tulkinneet tilanteen siten, että emulointi on nyt todistetusti toimivaksi osoitettu menetelmä ja siksi heidän ei enää tarvitse kantaa huolta organisaationsa tuottamien digitaalisten aineistojen suhteen. Emulointi pitää niistä huolen. Kuitenkaan Bearman ei näytä tällaisesta suhtautumistavasta ainoatakaan esimerkkiä eikä viitettä yhteenkään julkaisuun, josta tällainen asenne kävisi ilmi. Vielä hämmentävämmäksi tilanteen tekee se, että Bearman heti tämän todettuaan myöntää ettei Rothenbergkaan 1999 julkaisussaan esitä, että emulointi olisi vielä käyttökelpoisessa kehitysvaiheessa pitkäaikaissäilytyksen suhteen. Hän täydentää tätä vielä toteamalla, että jos digitaalisia aineistoja ei siirretä pois laitteisto- ja ohjelmistoympäristöstään, ne tulevat vanhentumaan ja siten lopulta tuhoutumaan niiden mukana. On kuitenkin hyvä huomata että Rothenberg on tästä täysin samaa mieltä omassa julkaisussaan (Rothenberg 1999, 4).

Bearman jatkaa tuomalla esiin, että monet tietokoneasiantuntijat ovat sanoneet ettei emulointi tule koskaan toimimaan pitkäaikaissäilytyksen tarpeisiin kelvollisella tavalla (Bearman 1999, 1). Mutta kuten väitteessään organisaatioista, jotka ovat jättäytyneet toiveajattelun vaarallisiin pauloihin, hän ei taaskaan viittaa yhteenkään julkaisuun, jossa tietokoneasiantuntija esittäisi emuloinnista tällaisia väitteitä. Hän myös esittää omana mielipiteenään, että hänestä kaikki esimerkit emuloinnin toteuttamiskelpoisuudesta todistavat sen toimimattomuudesta pikemmin kuin sen toimivuudesta, mutta ei perustele tätäkään väitettään millään tavalla (Bearman 1999, 1).

Vielä hän jatkaa Rothenbergin moittimista ja heittää kehään kuuluisan väitteensä, että Rothenberg pyrkii säilyttämään väärän asian keskittyessään tietojärjestelmien toiminnallisuuteen itse aineistojen sijasta (Bearman 1999, 1). Tästä kaikesta hän tekee sen villin johtopäätöksen, että vaikka emulointi toimisikin, se ei onnistuisi säilyttämään digitaalisten asiakirjojen todistusarvoa, ja samalla se jättää säilyttämättä ne digitaaliset asiakirjat, joissa todistusarvolla ei ole niinkään merkitystä. (Bearman 1999. 1)

Hoeven, Slats, Wijngaarden ja Verdegem (2005, 3) myöntävät, että emulointi keskittyy säilytettävän tietoaineksen sijaan ympäristöön jossa tuo tietoaineisto toimii. On kuitenkin tärkeää huomata, että itse tietoaineksen säilyttämiseen ei suuremmin tarvitse uhrata voimavaroja, jos tuon tietoaineksen toiminnallinen ympäristö on säilytetty. Jos siis ym-

päristö on onnistuneesti säilytetty, on sen mukana melkein automaattisesti säilytetty itse aineistokin.

Bearman ei kuitenkaan pysähdy tähänkään vaan jatkaa, että Rothenbergillä on lähtökohtaisesti väärä tavoite, kun tämä toistuvasti puhuu siitä että digitaalisten asiakirjojen säilyttäminen lukukelpoisina on merkittävä asia. Bearman esittää että digitaalisen aineiston pitkäaikaissäilytyksen kannalta sen lukukelpoisuus tulevaisuudessa ei ole ainoa eikä riittävä toiminnallinen vaatimus. Vielä hän syyttää Rothenbergiä siitä, ettei tämä itse asiassa ymmärrä mikä tekee digitaalisesta asiakirjasta todistusarvoisen pidemmällä aikatähtäimellä (Bearman 1999, 1). Tämä väite on ristiriitainen Bearmanin omien digitaalisten asiakirjojen säilyttämisvaatimusten kannalta. Vaikka hän sanoo, ettei kyky lukea digitaalista aineistoa ole riittävä säilyttääkseen digitaalisen asiakirjan todistusvoimaisuuden, hän kuitenkin toteaa, että riittävä todiste menneisyyden toiminnoista on järjestelmään syötetty ja sieltä ulos lähetetty bittivirta, joka sitten säilytetään yhdessä metatietojen ja kaikkien todistusarvoon vaikuttavien tekijöiden kanssa. Bearman on siis säilyttämässä *pelkästään* asiakirjojen luettavuuden ja kaiken lisäksi vielä muodossa joka ei enää vastaa alkuperäistä (Bearman 1999, 2).

Bearmanin kritiikissä on joitakin merkillisiä lapsuksia, joita ei ole huomioitu aiemmin. Hän esittää esimerkkinä emulaation toimimattomuudesta skenaarion, jossa tietokannan tila tietyllä hetkellä on tallennettu emulointia varten ja joka sitten avataan emuloimalla. Bearmanin mukaan meillä ei tällöin ole mitään asiakirjoja tuosta tietokannasta, vaan ainoastaan tietokannan toiminnallisuus ja teoreettinen mahdollisuus luoda asiakirjoja, jotka näyttävät siltä, että ne ovat syntyneet menneisyydessä. Tässä Bearman itse asiassa vihjaa että emulointi avaisi tien mahdollisuuteen väärentää tulevaisuudessa digitaalisia asiakirjoja siten, että ne näyttäisivät olevan peräisin meidän ajaltamme. Kaikkein suurimman huomion ansaitsee nyt kuitenkin Bearmanin virheellinen oletus, että Rothenberg olisi ikään kuin säilyttämässä vain tietokantoja toiminnallisuuksineen laiminlyöden kokonaan sinne syötetyn aineiston.

Rothenberg on alusta lähtien kirjoittanut digitaalisten aineistojen säilyttämisestä erillään ohjelmistoista, joilla ne on luotu. Hän ehdottaa ratkaisuna järjestelmää, joka aineiston palautushetkellä yhdistää nämä erikseen tallennetut datapaketit toimivaksi kokonaisuudeksi. Siinä on mukana sekä tietokantaan syötetty ja sieltä ulos lähetetty bittivirta, että

itse tietokannan toiminnallisuus ja alkuperäinen ympäristö. Tämä mahdollistaa tietokantaan syötetyn ja sieltä ulos lähetetyn bittivirran tarkastelemisen alkuperäisessä ympäristössä, sellaisena kuin tuon bittivirran syöttäjät sen aikoinaan näkivät. (Rothenberg 1999, 17.)

Bearmanin väite siitä, että Rothenberg laiminlöisi itse aineiston, on perätön. Kuitenkin Bearmankin toteaa myönnytyksenä Rothenbergille, että sellainen aineisto, joka itsessään on ajettavaa koodia, on väistämättä emuloitava (Bearman 1999, 2). Tässä kohden Bearman antaa ymmärtää, että itsessään ajettavaa koodia edustava aineisto olisi jotenkin vähemmistönä digitaalisten aineistojen joukossa, vaikka tosiasiassa suurin osa digitaalisesta aineistosta on ajettavaa koodia, joka saa järjellisen merkityksen vain siinä laitteisto ja ohjelmistoympäristössä, johon se alun perin kirjoitettiin ajettavaksi.

Bearman syyttää Rothenbergiä paperitiikereiden luomisesta, kun hän asettaa emuloinnille vaihtoehtoja, jotka sitten toteaa pidemmällä tähtäimellä toimimattomiksi ja näin tulee siihen tulokseen, että emulointi on ainut pitkäaikaissäilytyksessä toimiva vaihtoehto (Bearman 1999, 2). Bearman siis antaa ymmärtää että nämä vaihtoehdot ovat ainakin osittain Rothenbergin omien virheellisten käsityksien värittämiä, ja siten kaikin puolin jo lähtökohtaisesti väärin aseteltuja. Hätkähdyttävintä on kuitenkin Bearmanin näiden väitteiden jälkeen asettama oma vaihtoehto: digitaalisten aineistojen systemaattinen migrointi. Tässä hän väittää, että Rothenberg on jättänyt tämän vaihtoehdon täysin huomiotta. Jos luemme Rothenbergin julkaisua tarkkaan, huomaamme että hänen ehdottamaansa ratkaisuun nimenomaan sisältyy digitaalisten aineistojen systemaattinen migrointi (Rothenberg 1999, 20).

Näin Bearmanin syytös paljastaa siis sen, ettei hän itse ole kovin hyvin lukenut Rothenbergin kirjoituksia. Lisäksi Bearmanin argumentointi systemaattisen migroimisen puolesta on puutteellista. Vaikka hän itsekin toteaa, ettei standardeihin voi perustaa pitkäaikaissäilytystä, hän jättää ottamatta huomioon, että nimenomaan systemaattinen migrointi vaatii tiukat standardit, joita noudatetaan. Muutoinhan ei voitaisi systemaattisesti migroida. Näin Bearmanin ehdottama ratkaisu systemaattisestia migroinnista on ristiriidassa hänen toteamuksensa kanssa siitä, että luottamus standardeihin on naivia (Bearman 1999, 1).

Bearman myös syyttää Rothenbergiä siitä, että tämä tuomitsee migroinnin toimimattomaksi viittaamatta ainoaankaan aiheesta tehtyyn tutkimukseen. Samalla hän kuitenkin itse tuomitsee emuloinnin toimimattomaksi viittaamatta ainoaankaan emuloinnista tehtyyn tutkimukseen (Bearman 1999, 1-2). Bearmanin mukaan migrointiin pohjautuva säilytysstrategia olisi joutunut ylenkatsottuun asemaan. Todellisuudessa emulointi ja sen tutkimus on pitkäaikaissäilytyskentällä ollut aina altavastaajan asemassa, ja joutunut taistelemaan siitä, että se otetaan vakavasti muiden säilytysmenetelmien rinnalla. Migraatio on useimpien muistiorganisaatioiden ensisijainen menetelmä ja emulointia suoritetaan vain tarvittaessa.

Tässä työssä esille tuodut esimerkit tukevat sitä, että emulointi on menetelmä, joka on aina toiminut, jos sen toimintakykyiseksi saattamiseen on vain riittävästi panostettu. Juuri siksi Bearmanin väite siitä, että esimerkit emuloinnin toiminnallisuudesta ovat pikemmin todisteita sen toimimattomuudesta, on perin merkillinen.

5.7 Emulointi ja laki

Emuloinnin kehittäjillä ja lainsäädännöllä on aina ollut ongelmia toistensa kanssa. Emulointi ei itsessään riko mitään lakia, mutta siihen väistämättä liittyy aina jokin elementti joka liikkuu lain harmaalla alueella. Tällä lain harmaalla alueella tietokoneteollisuuden edustajat pyrkivät loppuun saakka kynsin ja hampain suojelemaan asiakkaidensa etuja. Vanhojen ohjelmistojen kohdalla kyse ei ole rahasta vaan yhtiöiden hallussa olevasta intellektuaalisesta pääomasta. Tämä intellektuaalinen pääoma voi usein olla paljon arvokkaampaa kuin tuotteista aikoinaan saadut rahatulot. Mitä ilmeisimmin moni ohjelmistoyhtiö käyttää vanhoja ohjelmistojaan ja niissä kehittäneitä ongelmanratkaisutapojaan hyväksi uudemmissa tuotteissaan, eikä siksi halua vapauttaa näitä tuotteita, ainakaan niiden lähdekoodia, vapaaseen jakeluun. Vaikka kyse on vanhoista laitteistoista ja ohjelmistoista, joilla ei enää ole markkina-arvoa, niiden tekijänoikeudet eivät ole missään vaiheessa rauenneet.

Osa ohjelmistoyhtiöistä on vapaaehtoisesti päästänyt vanhoja ohjelmistojaan vapaaseen jakeluun. Tällaisia ohjelmistoja kutsutaan abandonwareksi, hylätyksi tavaraksi. Valitet-

tavasti kuitenkin vain pieni murto-osa abandonwaresta on ohjelmistontuottajien itse vapaaseen levitykseen päästämää ja suurin osa laitonta. Osa yhtiöistä pitää tiukasti kiinni intellektuaalisesta omaisuudestaan, eikä salli jakelua siitä huolimatta, ettei jaetulla materiaalilla ole enää rahallista arvoa. Tällainen tilanne on kuluttajan kannalta hyvin valitettava, sillä kuluttajalla ei ole mitään laillista keinoa saada ohjelmistoa laillisesti käyttöönsä. Hänen on pakko rikkoa lakia, jopa siinäkin tapauksessa että on joskus menneisyydessä ostanut ohjelmiston. Ohjelmisto on medialla, jota uudet lukulaitteet eivät enää tue. Kuluttajan on joko migroitava median sisältö uudelle medialle, jolloin hän samalla joutuu purkamaan kopiointisuojauksen, joka sekin on rikollista, ja hänen on myös hankittava laitteistoja, joilla siirto onnistuu. Käytännössä alkuperäisen tuotteen omistaja tällaisessa tilanteessa kirjoittaa julkaisun nimen hakukoneeseen ja imuroi ohjelmiston joltakin laittomalta sivustolta. Lopputuloksena meillä on siis aina lain rikkominen.

Rothenberg nostaa lakikysymykset myös esiin ja tiedostaa niiden merkittävän aseman pitkäaikaissäilytyksen tulevaisuudessa. Koska pitkäaikaissäilytysorganisaatio ei voi rikkoa lakia vähimmässäkään määrin, voi koko emuloinnin käyttäminen pitkäaikaissäilytyksessä kaatua lakikysymyksiin. Rothenberg ei nähdäkseni anna lakikysymykselle riittävästi painoa. Hänelle lakikysymys vaikuttaa olevan jotain sellaista, joka voidaan hoitaa neuvotteluilla emulointiratkaisun kehittämisen ohella. (Rothenberg 1999a, 21-22; Rothenberg 1999b, 14; Rothenberg 2000a, 45; Rothenberg 2000b, 46.)

Tiedämme kuitenkin, että useat toimivat keksinnöt ja menetelmät ovat jääneet marginaali-ilmiöiksi, koska lakikysymyksistä ei ole päästy kaikkien oikeuksien omistajien kanssa yhteisymmärrykseen. Jotta emulointiratkaisun kehittämistyö ei valuisi hukkaan, tulisi lakikysymykset hoitaa mielestäni kuntoon ennen kuin emulointia suurimittaisesti kehitetään pitkäaikaissäilytysorganisaatioille sopivaksi ratkaisuksi.

5.8 Ohjelmistoarkistoista

Hoeven ja Suchodoletz (2009, 151) nostavat esiin yhden huolestuttavan huomion tämän hetkisestä tilanteesta digitaaliarkistojen kentällä. Varsinaista standardisoitua ohjelmistojen arkistointia ei tällä hetkellä vielä tapahdu. Ilman tallennettuja ohjelmistoja emulaat-

torit ovat hyödyttömiä (Reichherzer & Brown, 2006 tässä Hoeven & Suchodoletz 2009, 151).

Kirjastoissa ohjelmistoja kohdellaan kuten kirjojakin, ne laitetaan pakkauksineen hyllyyn odottamaan sitä aikaa, jolloin yhdelläkään asiakkaalla ei enää ole edes laitteistoja joilla noita tallenteita lukea. Syy siihen miksi tähän on tyydytty johtuu siitä että ohjelmistot ovat hyvin vaikeita talletettavia juurikin näkymäpoluissa esiin tulleiden riippuvuuksien vuoksi. Jos ohjelmisto halutaan tallettaa, sen voidaan katsoa olevan talletettu vasta sitten kun kaikki siihen liittyvät riippuvaisuudet on myöskin talletettu. Windowsin päällä pyörivän ohjelmiston toiminta perustuu useiden ohjelmistojen yhteistyöhön, ja jos tuo sovellus halutaan tallettaa, on silloin pakko tallettaa myös kaikki ne ohjelmistot joiden yhteistyöhön tuon sovelluksen kokonaistoiminta perustuu. Tämä nostaa meille valtavasti lainsäädäntöön ja tekijänoikeussuojaan liittyviä kysymyksiä ratkaistavaksi. (Hoeven & Suchodoletz 2009, 151.)

Monet ohjelmistot on varustettu kopiointisuojauksilla ja vaikka ne pystytäänkin kiertämään, ei digitaalinen arkisto voi lähteä lakia rikkomaan. Niinpä onkin ehdotettu, että ohjelmistoarkistot voisivat tehdä ohjelmistojenvalmistajien kanssa sopimuksen siitä että heille toimitetaan kopioita ohjelmistoista ilman suojauksia. Tällainen toiminta vaatii kuitenkin ihan uudenlaisen, valtion tukeman organisaation rakentamista. Valtioilla on useita arkistoja, mutta ohjelmistoarkistoja sillä ei vielä ole ainuttakaan. Luotettava pitkäaikaissäilytys edellyttää ohjelmistoarkiston olemassaoloa ja tämän seikan Hoeven ja Suchodoletz (2009, 151) tiedostavat.

Ohjelmistojen arkistoimisessa tulee vastaan useita esteitä. Yksi niistä on se, että ohjelmistoista on valmistettu useita erilaisia versioita eri alustoja käyttävien käyttäjien tarpeisiin ja ne on varustettu erilaisilla kielillä ja kalenterisyntakseilla. Nämä vaihtelevat maanosan ja kulttuurin mukaan. Kaikki nämä eri jakelumuodot tulisi ottaa huomioon ohjelmistojen arkistoimisessa. (Hoeven & Suchodoletz 2009, 151.)

Ohjelmistoarkisto toimisi samalla periaatteella kuin tallettaminen Kansallisessa Audiovisuaalisessa Arkistossakin eli talletettu aineisto on tallettajan omaisuutta ja hän voi sen koska tahansa noutaa pois. Näin ihmiset voisivat hyvinkin mielellään käydä tallettamassa vanhat koneensa ja ohjelmistonsa arkistoon, sillä heille itselleen säilytys tuottaa on-

gelmia ja arkistossa ne ovat paljon paremmassa tallessa kuin missään muualla. Suomessa tulisi siis nähdäkseni viipymättä aloittaa valmistelut kansallisen ohjelmistoarkiston perustamiseksi.

6. OAIS-MALLI

Jotta olisi mahdollista ymmärtää pitkäaikaissäilytyksen ongelman ratkaisuksi kehitettyjä emulointia käyttäviä digitaalisen aineiston talletusjärjestelmiä, on sisäistettävä niiden rakenne. Näiden rakenteiden omaksuminen ei ole mahdollista ilman, että tuntee Open Archival Information System-mallin, ja sen historian, jonka asettaman viitekehyksen päälle eri säilytysjärjestelmät rakentuvat. Tähän mennessä OAIS-mallin ovat ottaneet käyttöön kaikki suuret pitkäaikaissäilytykseen erikoistuneet organisaatiot. Käytännön toteutus kuitenkin vaihtelee organisaatioittain, ja kukin organisaatio on muokannut OAIS-mallin tarjoaman viitekehyksen itselleen sopivaksi. Juuri muokattavaksi OAIS-malli on suunniteltukin. OAIS-malli itsessään on yleispätevä kaikkeen arkistointikäyttöön ja sitä voidaan käyttää muissakin kuin digitaalista aineistoa käsittelevissä järjestelmissä. Emulointia käyttävistä säilytysjärjestelmistä esimerkiksi IBM:n ja Alankomaiden kansalliskirjaston yhdessä rakentama Digital Information Archiving System (DIAS), sen alle rakennettu palvelu e-depot, ja KB:n yhdessä Networked European Deposit Library (NEDLIB)-hankkeen kanssa rakentamat Deposit System for Electronic Publications (DSEP)-järjestelmät käyttävät kaikki rakenteensa ja toimintansa pohjana OAIS-mallia.

Aliluvussa 5.1 vien lukijan OAIS-mallin syntysijoille ja paljastan, miksi OAIS-malli on niin merkittävä digitaalisten aineistojen pitkäaikaissäilytyksessä. Aliluvussa 5.2 esitellen OAIS-mallin keskeiset toiminnot. Aliluvun 5.3 aliluvuissa 5.3.1, 5.3.2 ja 5.3.3 käyn läpi OAIS-mallin roolia emulaatiota käyttävissä pitkäaikaissäilytysprojekteissa.

6.1 OAIS-mallin historia

OAIS-mallin tarina alkoi 1995 kun Consultative Committee for Space Data Systemsin pelko tuottamansa digitaalisen aineiston tulevaisuuden kohtalosta oli johtanut pisteeseen, jossa oli lähdettävä etsimään jonkinlaista toimintamallia digitaalisille aineistoille soveltuvalle tietojen pitkäaikaissäilytysjärjestelmälle (Lavoie 2004, 1-2). Huomattiin kuitenkin pian, ettei sellaista ollut, ja siksi heidän ensimmäiseksi tehtäväkseen tuli sellaisen kehittäminen. Jo kehitystyönsä alkuvaiheissa havaittiin, että aloitetun kehitystyön mahdollinen lopputulos tulisi kattamaan paljon laajemmankin alan kuin vain avaruustut-

kimuksen tuottaman digitaalisen aineiston. Ymmärrettiin, että mikäli viitemalli saataisiin valmiiksi, se todennäköisesti olisi käyttökelpoinen myös muissa organisaatioissa erimuotoisten aineistojen arkistoimiseen. OAIS-malli on nimittäin sovellettavissa erilaisten organisaatioiden käyttöön, ja myös muuhun kuin digitaaliseen aineistoon. Projekti näytti tuottavan vastauksia keskeisiin pitkäaikaissäilytyskysymyksiin, ja CCSDS päätti lopulta avata kehitystyön kaikille halukkaille. Näin luonnostelun alla oleva viitemalli sai tuekseen useiden organisaatioiden yhteistyön. Viitemallin ensimmäinen luonnos tuli valmiiksi 1997, ja seuraava kahden vuoden päästä 1999. ISO-standardin luonnokseksi OAIS-malli hyväksyttiin 2000, ja lopullisen hyväksynnän se sai 2002 kansainvälisenä ISO-standardina nro 14721. (Lavoie 2004, 2.)

Ennen OAIS-mallia ei ollut olemassa viitekehystä, jota olisi voinut käyttää digitaalisille aineistoille tarkoitetun pitkäaikaissäilytysjärjestelmän hahmottelemiseen. Jotta digitaalista aineistoa voitaisiin saattaa pitkäaikaissäilytykseen kelpaavaan muotoon, tuli ensin määritellä tarkoin, mitkä ovat tuossa aineistossa keskeiset yhdistävät tekijät, millainen yhteinen terminologia tarvitaan ajatuksien vaihdon tueksi eri organisaatioiden välille. Lisäksi oli selvitettävä mitkä ovat digitaalisen aineiston pitkäaikaissäilytysjärjestelmän keskeiset toiminnot, joita ilman pitkäaikaissäilytystä ei voida luotettavasti toteuttaa. Näihin kysymyksiin oli löydettävä vastauksia, joiden suhteen päästäisiin yhteisymmärrykseen digitaalisia aineistoja hallussaan pitävien organisaatioiden kanssa. (Lavoie 2004, 2.)

Rothenberg (2000, 4) antaa OAIS-mallille hieman risuja siitä, että se puhuu arkistoinnista vaikka keskittyy enemmän kuvaamaan aineiston paketointia, kuvailua ja manipulointia käsittelemättä aineistolle ajan aiheuttamia ongelmia. Lisäksi OAIS kutsuu emulointia vakavaksi teknologiseksi ja taloudelliseksi riskiksi, ja ottaa migroimisen vastaan kaikkein loogisimpana arkistointimenetelmänä. Tästä Rothenberg(2000, 5) on hyvin närkästynyt ja syyttää OAIS-mallia emulointiratkaisun laiminlyömisestä. Hänen mielestään se on malli, johon ei ole jätetty tilaa emuloinnille, ja joka on piirretty vain migraatiota ajatellen. Näin hän tulee lopulta siihen tulokseen, että OAIS-mallin ironia on siinä, että se on suunniteltu säilytysmalliksi, mutta loppujen lopuksi sanoo itse säilyttämisestä vain hyvin vähän. OAIS-mallin suosion syy on Rothenbergin mukaan siinä, että siihen

on onnistuttu kiteyttämään useita olennaisia ominaisuuksia, joita organisaatiolla tulee olla, jotta tämä voisi aineistoja järjestelmällisesti säilyttää. (Rothenberg 2000, 5.)

Digitaalisen aineiston pitkäaikaissäilytystä käsittelevissä julkaisuissa esiintyy kaksi erehdyttävästi toisiaan muistuttavaa käsitettä, OAIS ja OAI. Kyse ei kuitenkaan ole samasta lyhenteestä, joka ajoittain kirjoitettaisiin väärin. OAI (Open Archives Initiative) on organisaatio, jonka tehtävänä on kehittää ja edistää tiedonjakamiseen liittyviä yhteisiä standardeja. OAI-malli keskittyy digitaalisten arkistojen välisen tiedonvaihdon standardeihin, niiden kehittämiseen ja käyttöön oton edistämiseen. Sen kehitystä eteenpäin ajavana voimana on ollut tarve jakaa tehokkaasti tieteellisten artikkeleiden sähköisiä versioita eri tiedekuntien ja yliopistojen välillä. "Arkisto" tässä lyhenteessä ei tarkoita arkistoa pitkäaikaissäilytysnäkökulmasta, vaan sillä viitataan e-tulosteita paljon käyttävien tahojen tapaan kutsua e-tulosteiden säilytyspaikkaa arkistoksi. Steering Committee, joka hallinnoi OAI:tä pyytää että arkistoalalla työskentelevät olisivat suopeita heidän tapaansa käyttää ”arkisto”-sanaa kohtaan. OAI:llä ei myöskään ole mitään tekemistä emuloinnin kanssa. (OAI 2011.)

OAI-mallissa arkisto-termi ymmärretään hyvin eri tavalla kuin muissa malleissa. Siinä 'arkisto' tarkoittaa esimerkiksi palvelinta, jossa aineisto on. Standardi keskittyy siihen, että aineiston on oltava vapaasti saatavilla kaikille, ja että sitä koskevien metatietojen tulee olla sellaisessa muodossa, että ne on palautettavissa kenen tahansa käyttöön standardoiduilla protokolloilla. Tähän tarkoitukseen OAI-mallissa on kehitetty OAI-PMH-protokolla. Vuodesta 2002 protokolla on ollut vakaa ja useiden organisaatioiden käytössä. (OAI 2011.)

Termien maailma on monimutkainen pitkäaikaissäilytyksen alueella. Ne menevät usein lomittain tai jopa päällekkäin, ja näitä päällekkäisyyksiä pyritään poistamaan. On ehdotettu että OAIS ja OAI- malli sulautettaisiin yhteen (Hirtle 2001, 2). Näin ne palvelisivat tiedeyhteisöä tehokkaimmalla mahdollisella tavalla.

6.2 OAIS-mallin toiminnallisuus

OAIS-malliin perustuvassa säilytysjärjestelmässä on vuorovaikutussuhde useiden ulkoisien toimijoiden kesken. OAIS-mallin avulla pyritään tunnistamaan nämä toimijat, ja määrittelemään heidän käyttämänsä OAIS-malliin suuntautuvat käyttöliittymät. Mallin mukaan nämä toimijat ovat hallinto, tuottaja ja kuluttaja. OAIS-malli ei siis ikinä toimi tyhjiössä vaan yhteistyössä toimijoidensa kanssa. (Lavoie 2004, 5.)

Hallinnon tehtävänä on huolehtia säilytysjärjestelmän säilytysstrategian ajankohtaisuudesta, sen soveltuvuudesta järjestelmän kulloiseenkin tilanteeseen. Hallinnon on reagoitava muuttuneeseen tilanteeseen säilytysjärjestelmässä ottamalla tarpeen vaatiessa käyttöön väliaikaisia vaihtoehtoisia ratkaisuja. OAIS-mallin on kyettävä pitämään tuottajalle antamansa lupaus tietoaineksen säilyttämisestä poikkeustilanteesta huolimatta. Hallinto pitää huolta siitä, että säilytysjärjestelmällä on resursseja korjata mahdollisia virheitä. Huomatessaan epäkohtia sen on muutettava säilytysjärjestelmän toimintapolitiikkaa. Hallinto ei kuitenkaan ole vastuussa järjestelmän päivittäisistä toiminnoista siinä mielessä, että vastuu päivittäisten toimintojen oikeellisuudesta ja niiden suorittamisesta olisi hallinnolla. Tästä pitää huolen oma moduulinsa arkiston sisällä, hallinnointimoduuli. (Lavoie 2004, 5.)

Tuottaja voi olla järjestelmä, organisaatio tai yksilö, joka tuo tietoainesta säilytysjärjestelmään pitkäaikaissäilytykseen. Tiedon tuominen järjestelmään tapahtuu vastaanottomoduulin kautta, joka tarkistaa saapuvan tietoaineksen metatietoineen ja valmistelee sen arkistoimista varten. Jokaista aineistolajia kohden on omat metatietovaatimukset, joiden on täytyttävä, jotta tietoa voidaan ottaa säilytykseen. (Lavoie 2004, 6.) Kuluttajat ovat yksilöitä, organisaatioita tai järjestelmiä, jotka lopulta käyttävät tietoa. Tiedon tuottajista tulee näin usein myös sen kuluttajia, sillä ulkopuolisten tahojen lisäksi myös tuottajat käyttävät itse arkistoon viemäänsä, ja myös muiden organisaatioiden arkistoon viemää aineistoa. OAIS-viitemalli osoittaa kuluttajien joukosta erityisen kohderyhmän, josta se käyttää käsitettä *nimetty yhteisö*. Sillä tarkoitetaan sitä kuluttajien luokkaa, jonka voidaan otaksua ymmärtävän arkistoidun tietosisällön itsenäisesti ilman ulkopuolista neuvoa. Yksi OAIS-mallin pakollisista vaatimuksista on, että sen tulee tallettaa tietoa sellaisessa muodossa, joka on arkiston suurimman käyttäjäryhmän ymmärrettävissä ja käytet-

tävissä. On tärkeää huomata, että OAIS-mallin on tarkoitus palvella kaikkia, mutta nimetty yhteisö on se, jonka oletetaan ymmärtävän ja osaavan käyttää aineistoja hyväkseen ilman asiantuntijan opastusta.

Nimetyn yhteisön laajuutta ei määritellä OAIS-mallin ja sen arkistoimien tietoaineistojen pohjalta, vaan nimetty yhteisö on se, joka määrittelee missä muodossa tietoaines talletetaan, mitkä komponentit OAIS-mallista ovat käytössä ja miten ne toimivat vuorovaikutuksessa keskenään. Näin siis nimetty yhteisö on se jolla on eniten vaikutusvaltaa arkiston toiminnan ja arkistoitavan tietosisällön suhteen. Nimetty yhteisö ei ole kattavuudeltaan millään lailla lukittu, vaan voi vaihdella ajan vaatimusten mukaisesti. Nimetyn yhteisön dynaamisuus pitää sisällään myös sen oman laajuuden, ja nimetty yhteisö voi halutessaan kaventaa tai laajentaa sitä käyttäjäryhmää, joka luetaan kuuluvaksi nimettyyn yhteisöön. Nimetyn yhteisön laajuuden määritteleminen on olennaista, sillä sen mukaan katsotaan mitä metatietoja aineiston mukana on välttämätöntä toimittaa, jotta aineisto täyttäisi ymmärrettävyyden vaatimukset. Nimetty yhteisö voi laajimmillaan kattaa kaikki käyttäjät, joka vaatimusten suhteen tarkoittaa sitä, ettei mitään erikoisvaatimuksia aineiston ymmärtämisen ole asetettu. Tällöin myös metatietojen liittäminen aineistoon muuttuu huomattavasti vaikeammaksi, sillä aineistojen toimittaminen kaikille halukkaille nostaa aineistoon liitettävän metatiedon määrää huomattavasti. (Lavoie 2004, 6.)

OAIS-mallin toimijoiden ja moduulien kanssa on muistettava, ettei kyse ole näiden tahojen fyysisestä erosta, tai maantieteellisesti erillään olevasta sijainnista. OAIS-mallissa on haluttu erottaa toisistaan loogisella tasolla moduulit ja toimijat siten, että näiden asemaa ja roolia mallissa voidaan tarkastella. OAIS-mallilla on kaksoisrooli. Se sekä säilyttää aineistoa, että tarjoaa aineistoa käyttöön. Tämän kaksoisroolinsa se täyttää kuudella erilaisella moduulilla.

Vastaanottomoduuli (Ingest) huolehtii tuottajalta tulevan aineiston kulkeutumisesta muihin moduuleihin. Tämän moduulin kanssa aineiston tuottajat ovat eniten tekemisissä. Aineiston tuottaja lähettää säilytyksensuunnittelumoduulille kuitin vastaanottomoduuliin lähettämästään tietopaketeista, jotta uusista tietopaketeista ollaan tuossa moduulissa tietoisia. Tietoaines hyväksytään tarkastamalla sen korruptoitumattomuus ja eheys, sen muoto muunnetaan pitkäaikaissäilytykseen sopivaksi, siitä uutetaan tai sille luodaan

metatietueet helpottamaan aineiston löytymistä ja oikeaa käyttöä. Lopuksi se lähetetään datanhallintamoduuliin pitkäaikaissäilytettäväksi. (Lavoie 2004, 8.)

Datanhallintamoduulissa (Archival Storage) tapahtuvat pitkäaikaissäilytyksen kannalta keskeisimmät toiminnot. OAIS-mallin suosion ja maailmanlaajuisen leviämisen syy on juuri siinä, että se on hyvin helposti mukautettavissa kunkin organisaation tarpeisiin, ja siihen voidaan vaivattomasti lisätä tai poistaa moduuleja tarpeen mukaan. Olennainen osa datanhallintamoduulia on säännöllisesti tapahtuva aineiston eheyden tarkistus (Lavoie 2004, 8). Ilman tätä toimintoa saattaisi arkistoon vähitellen keräytyä virheitä, jotka havaittaisiin vasta aineistoa palautettaessa sitä kysyvälle asiakkaalle. Tällöin saattaisi olla jo liian myöhäistä korjata aiheutunut vahinko, varsinkaan jos aineiston alkuperäiseltä toimittajalta ei enää kyettäisi aineiston eheyttä varmistamaan. Koska digitaalisen aineiston arkistosäilytyksen luonne on se, että arkistoon tuotu uusi aineisto on osittain riippuvainen sinne jo kertaalleen tuodusta aineistosta, virheiden vaikutus kasvaa ajan kanssa eksponentiaalisesti. Näin virheet perustavissa aineistoissa voivat johtaa koko pinon kaatumiseen (van der Werf 2000, 34; 58).[16] Tällaisesta voidaan mainita esimerkkinä arkistoitu verkkolehti. Verkkolehden portaali, sen valikoissa oleva grafiikka ja muu oheismateriaali on kertaalleen talletettu ja tuon lehden yksittäiset numerot viittaavat tuohon jo talletettuun portaaliin. Jos siis talletetussa portaalissa on virheitä, heijastuu niiden vaikutus myös numeroihin ja artikkeleihin joita numeroissa on. Näissä taas on edelleen linkkejä, joiden toiminta perustuu talletettuun portaaliin, ja näin virheiden vaikutus moninkertaistuu jokaisessa taitekohdassa uhaten lopulta koko talletetun aineiston käyttökelpoisuutta.

Datanhallintamoduulin keskeisin tehtävä järjestelmän käyttäjän kannalta on aineiston palauttaminen sitä pyytävälle siten, että hän ymmärtää tuon aineiston asiayhteyden ja kykenee käyttämään sitä niin kuin sen tekijät tarkoittivat sitä käytettävän. On tärkeää

16 The system disk image of the reference platform is also packaged in a separate AIP, because it is used to run more than one publication. When such a new reference platform has been created, all new deposit publications that run on it carry a reference to this reference platform in their technical metadata .

huomata, että tämä toteutuu sanatarkasti *vain* niissä säilytysratkaisuissa, joissa emulaatio on mukana palauttamismekanismeissa.

Tiedonhallintamoduulin (Data Management) vastuulla on aineistoja kuvailevien metatietojen identifioiminen, tallettaminen, ja päivittäminen. Ilman metatietoja koko arkiston sisältö on hyödytöntä, joten tiedonhallintamoduulin toimivuudesta huolehtiminen on koko säilytysjärjestelmän kannalta hyvin keskeistä. Tiedonhallintamoduuli huolehtii myös järjestelmän toimintojen tuottamien tilasto- ja muiden käyttötietojen säilyttämisestä. Tämä tieto on järjestelmän sisällä välttämätöntä, jotta järjestelmän johto pystyy reagoimaan toimenpiteitä vaativiin asioihin, esim. parantamaan jonkin tietyn aineistoryhmän saatavuutta. Jos tiettyä aineistoa kysytään erityisen paljon jonakin tiettynä vuodenaikana, kuten tilanne usein on esim. kirjastoissa, voidaan aineiston saatavuutta helpottaa tuon sesonkikauden ajaksi. (Lavoie 2004, 9.)

Tiedonhallintamoduuli tuottaa raportteja muiden moduulien suorittamista kyselyistä OAIS:än sisällä, ja hoitaa omiin tietokantoihinsa kohdistuvien kyselyihin vastaamisen. Voidaan sanoa, että tiedonhallintamoduuli on eräänlainen OAIS:n informaatikko, jonka tehtävänä on tallettaa tietoa säilytysjärjestelmästä ja välittää sitä eteenpäin pyydettäessä. (Lavoie 2004, 9.)

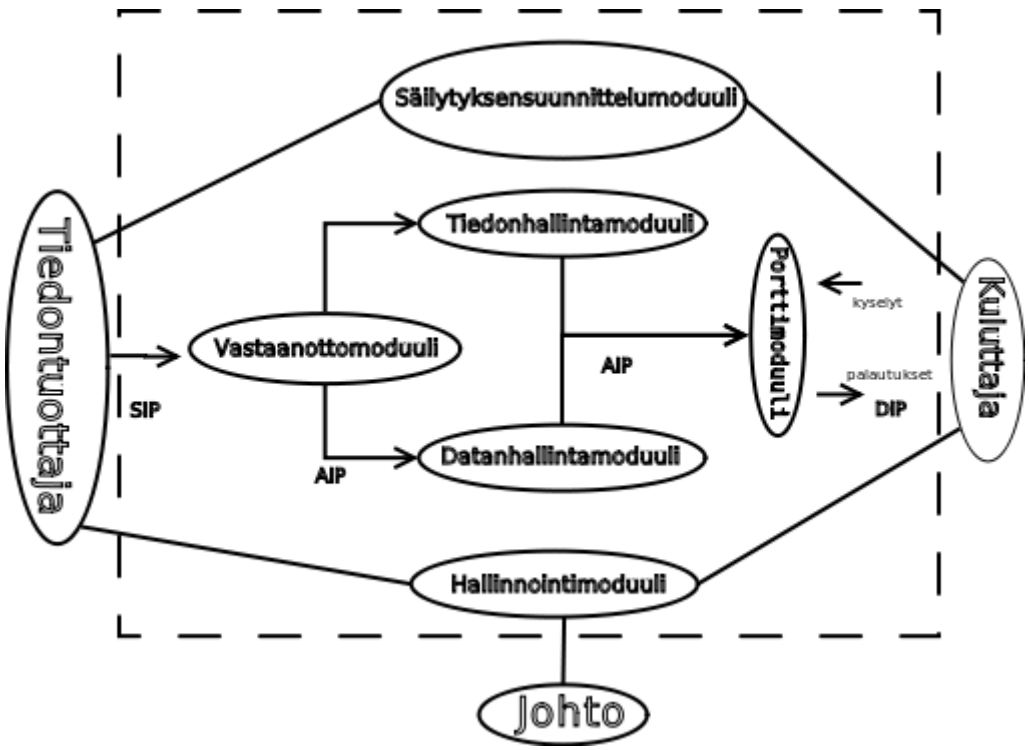
Säilytyksensuunnittelumoduuli (Preservation Planning), OAIS-mallin neljäs toiminnallinen moduuli, toimii vartijana muun maailman ja OAIS-arkkitehtuurin välissä. Sen tehtävänä on pysyä ajan tasalla teknologian kehityksessä ja hankkia tietoa uusista teknologisista innovaatioista. Se laatii suosituksia säilytysjärjestelmää tehostavien teknologioiden käyttöönotosta ja suojelee säilytysjärjestelmää sitä uhkaavilta mahdollisilta uhilta. Sellaisia ovat esimerkiksi säilytysjärjestelmän vanhentumassa olevat teknologiset ratkaisut, jotka on pikaisesti vaihdettava uudempiin. (Lavoie 2004, 9.) Säilytyksensuunnittelumoduuli tiedustelee säännöllisesti nimetyn yhteisön toiveita saadakseen kaivetuksi esille mahdollisesti laiminlyödyt alueet säilytysjärjestelmässä. Se kehittää strategioita, joilla se kykenee vastaamaan muutoksien aiheuttamiin haasteisiin. (Lavoie 2004, 9.)

Porttimoduuli (Access) huolehtii toiminnoista, joiden kautta asiakkaat, etusijassa nimetty yhteisö, käyttävät arkiston aineistoja. Tämän moduulin kautta säilytysjärjestelmä joko lunastaa tai pettää kaikki sen asiakkailleen antamansa lupaukset (Lavoie 2004, 9 - 10).

Porttimoduuli joutuu tulikokeeseen jo siinä vaiheessa kun käyttäjät tekevät hakuja aineistotietokantaan. Kuinka nopeasti porttimoduuli toimittaa kattavan tuloslistauksen on yksi mittari sille, kuinka hyvin koko järjestelmä toimii. Sen perusteella käyttäjät myös muodostavat ensimmäisen vaikutelmansa. Siksi siihen kannattaa panostaa paljon, etenkin nopeuteen. Porttimoduuli ei kuitenkaan ole vain sisällöntuoja, vaan jos siihen on tarvetta, se suorittaa erilaisia muuntotoimenpiteitä aineistolle ennen sen luovuttamista asiakkaalle. Jos aineisto sitä vaatii, porttimoduuli varustaa aineiston mahdollisilla kopiointisuojauksilla tai tarkastelua rajoittavilla suojauksilla. Muuntotoimenpiteet voi käsittää esim. liiallisen, käyttäjän kannalta tarpeettoman, ehkä jopa haitallisen metatiedon karsimista aineistosta. (Lavoie 2004, 9 – 10.)

Viimeinen moduuli OAIS-mallissa on hallinnointimoduuli (Administration). Hallinnointimoduuli on vastuussa kaikkien muiden moduulien toiminnasta ja toimii siten niiden esimiehenä. Hallinnointimoduuli kommunikoi kaikkien muiden moduulien kanssa ja toimii niiden pääarkkitehtinä. Se on vuorovaikutuksessa myös asiakkaiden kanssa, niin aineistontuottajien, kuin sen kuluttajienkin kanssa. Se siis huolehtii asiakaspalvelusta ja neuvottelee aineistontuottajien kanssa siitä, mitkä ovat mielekkäimpiä tiedon toimitusmuotoja, eli mitä SIP-pakettien (Submission Information Package) tulee pitää sisällään. (Lavoie 2004, 10.)

Yhteensä OAIS-mallissa on siis kuusi moduulia, joiden yhteinen tarkoitus on palvella nimettyä yhteisöä tuomalla sen saataville säilytettyä digitaalista aineistoa siten, että nimetylle yhteisölle koituu siitä mahdollisimman pieni vaiva. Mutta kahden moduulin kautta säilytysjärjestelmä näyttäytyy asiakkailleen, vastaanottomoduulin ja porttimoduulin. Niiden kautta heijastuu asiakkaille se, millainen toiminnallisuus vallitsee järjestelmän sisällä moduulien kesken.



Kuvio 1. OAIS-mallin toiminnallisuus. Lähde: Lavoie 2004.

6.3 OAIS-malli emulaatiota käyttävissä pitkäaikaissäilytysjärjestelmissä

6.3.1 DSEP ja OAIS

DSEP on NEDLIB-projektissa syntynyt digitaalisen aineiston pitkäaikaissäilytysjärjestelmä. NEDLIB-projektia ja DSEP-järjestelmän kehitystä johti KB. Projekti toteutettiin 1998-2000, ja siinä oli mukana kahdeksan kansalliskirjastoa, myös Suomen kansalliskirjasto. Projektin tarkoituksena oli kehittää yhteinen arkkitehtuurinen viitekehys ja tarvittavat työkalut digitaalisille aineistoille tarkoitettujen pitkäaikaissäilytysjärjestelmien rakentamiseen. Kohderyhmänä DSEP:iä kehitettäessä olivat erityisesti sellaiset muistiorganisaatiot, jotka ovat aikeissa laajentaa toimialuettaan digitaalisen aineiston puolelle, mutta joille on vielä epäselvää, kuinka he käytännössä saavat digitaaliset aineistot mahtumaan jo toiminnassa oleviin järjestelmiinsä. Juuri tämän ongelman äärellä projektiin osallistuneet kansalliskirjastotkin olivat. On kuitenkin huomattava, että digitaalisen aineiston pitkäaikaissäilytysjärjestelmä on eri asia kuin digitaalisen aineiston kirjasto.

Projektiin osallistuneilla kansalliskirjastoilla oli ongelmana digitaalisen aineiston pitkäaikaissäilytys, ei sen saattaminen yleiseen jakeluun. Siksi DSEP-mallista on jätetty pois joitakin OAIS-mallin moduuleja, joiden on katsottu liittyvän enemmän digitaalisen aineiston kirjastojakeluun. Vastaavasti siihen on lisätty joitakin moduuleja, jotka on katsottu alkuperäisen OAIS-mallin puutteiksi. (Werf-Davelaar 1999,1.)

OAIS-mallin etuna on sen muokattavuus organisaatioiden omiin tarpeisiin. DSEP-säilytysjärjestelmässä käytettiin OAIS-mallia ohjaavana tekijänä, mutta siihen tehtiin joitakin muutoksia, jotta malli sopisi paremmin suunniteltavaan järjestelmään. DSEP-järjestelmä koostuu 6 moduulista, joista 5 on OAIS-mallin mukaisia. Yksi ylimääräinen moduuli, säilytysmoduuli, lisättiin digitaalisen aineiston säilytystä varten. (Werf-Davelaar 1999, 4.)

Koska vastaavanlaiset projektit, kuten Curl Exemplars in Digital Archives (CEDARS), ja Preserving and Accessing Networked Documentary Resources of Australia (PANDORA), käyttivät OAIS-mallia, katsottiin että se on riittävän vahva näyttö OAIS-mallin soveltuvuudesta digitaalisen aineiston pitkäaikaissäilytysjärjestelmän kehityksen pohjaksi. (Werf-Davelaar 1999, 1.)

NEDLIB-projektin painotus oli asetettu digitaalisen aineiston pitkäaikaissäilytykseen liittyviin teknisiin kysymyksiin, erityisesti Rothenbergin ehdottaman emulointiratkaisun yksityiskohtiin, ja näiden yksityiskohtien sovittamiseksi rakenteilla olevaan järjestelmään. Yksi NEDLIB-projektin keskeisistä tavoitteista olikin OAIS-mallin kehittäminen siten, että se soveltuisi paremmin DSEP:in käyttöönottoon, ja sen myötä muiden digitaalisten pitkäaikaissäilytysjärjestelmien suunnitteluun. Suurin osa projektin kehitystyöstä oli sellaista, että tuon työn tuloksia voidaan yleisesti soveltaa digitaalisiin kirjastoihin, eikä vain DSEP:iin. (Werf-Davelaar 1999,2.)

DSEP:in prosessimallista nähdään, että siihen rakennetut toiminnot on sovitettu OAIS-mallin moduuleihin pohjautuen. DSEP:ssä on lähdetty siitä, että järjestelmä tarvitsee sisään syötettävää materiaalia varten saantomoduulin (acquisition), joka ottaa vastaan uuden aineiston nimikkeen ja metatiedot. Saantomoduuli välittää saamansa tiedot keräilymoduulille. Keräilymoduulin tehtävänä on hankkia varsinainen säilytykseen tuleva aineisto eli SIP-paketti. Keräily voi tapahtua sähköpostilla, webharavalla tai perinteisen

postin kautta. DSEP-työryhmä on katsonut, etteivät pitkäaikaissäilytysorganisaatiot voi sanella tallettajilleen millaisia formaatteja nämä saavat käyttää, vaan heillä pitää olla valmius ottaa vastaan kaikki tiedostomuodot. Pitkäaikaissäilytysorganisaation on neuvoteltava talletuksien tekijöiden kanssa tiedostomuodoista ja siitä toimitetaanko aineistosta kokotekstiversio luettelointia varten. Siksi tähän moduuliin keskittyy eniten kehitystyötä organisaatioissa, jotka ottavat käyttöön DSEP-järjestelmän. (Werf-Davelaar 1999, 4; 2000, 7.)

Vastaavasti DSEP:ssä on paketointi&toimitus-moduuli (delivery), joka välittää aineiston porttimoduuliin, johon asiakas on tehnyt aineistohakuja. Pitkäaikaissäilytysorganisaatiot eivät voi ennustaa kaikkia mahdollisia muutoksia ja vaatimuksia, joita tulevaisuudessa saattaa jakelupaketteihin kohdistua. Siksi paketointi&toimitus-moduulin on oltava joustava ja mukautettavissa uudenlaisiin ratkaisuihin, joita tulevaisuudessa mahdollisesti otetaan käyttöön. (Werf-Davelaar 1999, 4; 2000, 8.)

DSEP-mallissa porttimoduuli on huomattavasti rajoitetumpi kuin OAIS-mallissa. Tämä toimintojen rajoittaminen johtuu siitä, että monet toiminnot liittyvät laajemmin digitaaliseen kirjastoon, ja menevät siten DSEP:in toimialueen ohitse. Porttimoduuli on DSEP:stä aineistoa hakevalle kaikkein tärkein moduuli koko säilytysjärjestelmässä ja ainoa, jonka kanssa pelkästään muiden tallettamia aineistoja käyttävän on oltava tekemisissä. Porttimoduuli hakee käyttäjän pyynnöstä tietyn Archival Information Packagen (AIP), ja valmistelee siitä käyttäjän vaatimuksia vastaavan Dissemination Information Packagen (DIP). Sama aineisto voi siis tulla porttimoduulista ulos useissa eri muodoissa tilauksesta riippuen. Aineisto voidaan toimittaa kokoteksti-indeksoinnilla tai muunnettuna käyttäjän pyytämään tiedostomuotoon. Aineisto voidaan myös toimittaa varustettuna emulaattorilla. (Werf-Davelaar 1999, 5.)

Datanhallintamoduuli ei DSEP-järjestelmän rakentajien mielestä ollut sellaisenaan riittävä DSEP:iin. OAIS-mallin mukaisella datanhallintamoduulilla ei nimittäin ole, eikä edes tarvitse olla tietoa säilyttämänsä tiedon intellektuaalisesta sisällöstä. Niinpä he lisäsivät DSEP:iin säilytysmoduulin (preservation), koska OAIS-mallissa ei sellaista alun perin ollut. Jos säilytysmoduuli katsoo tarpeelliseksi migroida aineiston, silloin tuosta aineistosta syntyy migraation tuloksena uusi versio. Jotta vanhempi ei tuhoutuisi, ja siihen voitaisiin aina tarvittaessa palata, viedään tämä migroimisen seurauksena syntynyt

uusi tiedosto uutena aineistona saantomoduulin kautta järjestelmään sisään. Alkuperäisessä OAIS-mallissa migraatio oli sijoitettu datanhallintamoduuliin, mutta siinä oli silti epäselvää, missä nämä migraatiot tapahtuvat. Siksi DSEP-järjestelmän kehittäjien mielestä oli välttämätöntä lisätä järjestelmään säilytykselle omistettu moduuli, säilytysmoduuli. (Werf-Davelaar 1999, 5-6.)

DSEP-järjestelmässä lähdetään siitä, että järjestelmän on tiedostettava milloin säilytyksessä olevat aineistot ovat vaarassa muuttua saavuttamattomiksi järjestelmän ulkopuolella tapahtuvien muutosten takia.

DIP voi myös olla aineisto ja joukko emulaatiospesifikaatioita, jotka on kirjoitettu Universal Virtual Computeria (UVC) varten. Tällaisissa tapauksissa uusista spesifikaatioista luodaan metatietoa, joka viedään tiedonhallintamoduuliin. (Werf-Davelaar 1999, 5-6.)

DSEP:ssä liikkuvien tietopakettien malli on OAIS-viitemallin mukainen. Siinä mielessä DSEP:in tietopaketit eivät puhtaasti seuraa OAIS-mallin määritelmiä, että OAIS:ssä ne ovat loogisia objekteja, kun taas DSEP:ssä aineisto, siihen liittyvä metatieto ja sen ajamiseen tarvittavat ohjelmistot voivat kaikki olla erillisinä paketteina. Nämä erilliset paketit on linkitetty toisiinsa läpinäkyvän osoituksen (opaque identifiers, ks. "läpinäkyvä osoitus") mukaisia metatietotunnisteita apuna käyttäen. Tällainen menettely on mielekästä, sillä aineistoon liittyvää metatietoa joudutaan ajoittain päivittämään, varsinaisen aineiston pysyessä koskemattomana. Näin siis on välttämätöntäkin pitää aineisto ja siihen liittyvä metatieto erillään toisistaan, jottei varsinainen aineisto pääse korruptoitumaan metatietojen päivityksien yhteydessä. (Werf-Davelaar 1999, 6.)

DSEP-järjestelmään on omaksuttu OAIS-viitemallista tapa jakaa metatiedot kahteen eri ryhmään, esitysmetatietoon ja säilytyskuvausmetatietoon. Varsinkin, jos aineistoon kohdistetaan migraatiota, on säilytyskuvausmetatietoja täydennettävä aineistossa mahdollisesti tapahtuneiden tieto- ja toiminnallisuushäviöiden johdosta. Emulointia tarvitsevien aineistojen yhteyteen on liitettävä kaikki esitysmetatieto, mikä vaaditaan emuloinnin toteuttamiseen ja myös tarkat ohjeet siitä, miten digitaalioriginaalia käytetään emuloinnin alaisuudessa. On oltava yhteisymmärrys siitä, millainen autenttisuuden menetyksen aste on vielä hyväksyttävissä. (Werf-Davelaar 1999, 7.)

NEDLIB toi emuloinnin osaksi OAIS-järjestelmää DSEP:in kehittämisen kautta. Tämä vaatii varsinkin emulointiin liittyvien metatietomoduulien lisäämistä OAIS-malliin. (Werf-Davelaar 1999, 8.)

Rothenbergin visioiden toteuttaminen vaati OAIS-mallin laajentamista ja muokkaamista (Rothenberg 2000, 13). DSEP-mallissa OAIS-mallia on kehitelty eteenpäin siten, että se soveltuu paremmin pitkäaikaissäilytykseen. Rothenberg on sitä mieltä, että emulointi vaatii niin suuren joukon erilaisia koordinoituja toimintoja, ettei sitä voi puristaa yhteen moduuliin, vaan että se tulee jakaa OAIS-mallin esittelemiin erillisiin moduuleihin. Juuri näin DSEP-mallissa on tehtykin, ja Rothenberg muistuttaa meitä siitä, että OAIS on tarkoitettu viitemalliksi, ei järjestelmän suunnittelumalliksi. (Rothenberg 2000, 5.)

On syytä käydä läpi, mitä OAIS-mallin tietopaketit AIP, DIP ja SIP pitävät sisällään emulointiratkaisussa. Emulointia varten tehdyn AIP:n tulisi sisältää kaikki tieto ohjelmistoista ja niihin liittyvistä komponenteista, jotka tarvitaan tuon aineiston täydelliseen palauttamiseen. Niiden lisäksi sen tulisi sisältää kaikki tarvittava tieto koskien laitteisto-ympäristöä ja sen oikeaa konfiguraatiota aineiston palauttamisen kannalta. (Rothenberg 2000, 14.) DSEP:n kantavana ideana on että lähtökohtaisesti kaikki viedään AIP-paketteina järjestelmään, niin aineistot kuin niiden käyttöön tarvittavat ohjelmistokomponentitkin. Ne ovat kaikki tuolla tasolla samanlaisia bittivirtapaketteja, ja siinä mielessä yhdenvertaisia. Yhtenä vaihtoehtona Rothenberg (2000, 14) esittää myös sitä että sekä aineisto että sen tarvitsemat ohjelmistokomponentit talletetaan yhteen yhteiseen AIP:hen. Tämä tosin lisäisi arkiston aineistojen toisteisuutta huomattavasti, mutta voisi olla perusteltua joidenkin tärkeimmiksi katsottujen aineistojen kohdalla.

Ihanteellisessa tapauksessa kaikki tarvittava materiaali aineiston arkistoimiseen ja sen palauttamiseen tulee aineiston tallettajalta, mutta on selvää ettei tämä aina voi toteutua. Jos komponentteja puuttuu, aineistonvastaanottomoduuli voi pyytää niitä tallettajalta ennen kuin aineisto otetaan SIP:nä vastaan järjestelmään. Jos sitä ei kuitenkaan tallettajalta saada tai tallettajalla ei ole resursseja saattaa puuttuvia komponentteja säilytyskuntoon, saantomoduuli tai keräilymoduuli hankkii ne jostain tai käynnistää niiden kehitystyön. Kehitystyön käynnistäminen on vaihtoehdoista tietysti kaikkein kallein ja organisaatiolle raskain ja siksi myös viimeinen, johon käytännössä turvaudutaan. Mielekkäimmältä vaikuttaa vaihtoehto jossa tarvittavia komponentteja pyydetään tallettajalta, sillä

oletettavasti hänellä on oman aineistonsa palautusvaatimuksista paras tietotaito, ja siten pyytäminen tallettajalta noudattaa yleisesti järkeväksi hyväksyttyä vähimmän vaivan lakia. Mutta selvää on myös, että emulointiin perustuvaa pitkäaikaissäilytysjärjestelmää on turha perustaa, jos ei ole resursseja puuttuvien ohjelmamoduulien kehittämiseen. (Rothenberg 2000, 15.)

Rothenberg ehdottaa, että DSEP tarjoaisi tallettajille jonkinlaista validointipalvelua, jossa tallettajat voisivat käydä testaamassa aineistojaan nähdäkseen täyttävätkö nämä DSEP:in kriteerit sisäänotettavalle aineistolle. Validointipalvelu voisi sisältää myös testikentän, jossa tallettaja voi testata omien aineistojensa palautumista emuloinnin avulla ennen talletuksen tekemistä. Joissakin tapauksissa tallettaja voi tuoda aineistonsa keräilymoduuliin siten, että se sisältää ajettavan version emulaattorista tai emulaatiospesifikaation. Tämä on tarpeen varsinkin jos kyse on harvinaislaatuisemmasta aineistosta, jollaista arkistossa ei entuudestaan ole, tai jota on vain hyvin vähän. Tällaisessa tapauksessa keräilymoduuli voi erottaa emulaattorin muusta aineistosta (SIP:stä jonka tallettaja toimittaa), ja tallettaa tuon aineistosta uutetun emulaattorin sitten järjestelmään omana AIP:nä. Tallettaja voisi tässä tapauksessa toimittaa SIP:n sellaisessa muodossa, että se tukee emulaattorin ”uuttamista” irtonaiseksi muusta aineistosta. Arvoluokittelussa korkeimmalle sijoitettu aineisto tarkistettaisiin toimivuuden varalta huolellisemmin kuin muut aineistot. (Rothenberg 2000, 16.) Vaikka aineisto osoittaa todellisen arvonsa vasta pidemmän ajan päästä, on jokaisessa arkistojärjestelmässä välttämätöntä suorittaa arvonmääritystä.

Käytännössä DSEP:ssä aineisto, joka ei sisäänottohetkellä sisällä kaikkea tarvittavaa aineiston säilytykseen, tulee helposti hylätyksi. Näin tapahtuu varsinkin jos puuttuvaa komponenttia ei löydy muista organisaatioista. Tietysti voidaan myös käynnistää puuttuvan komponentin kehitystyö. Luultavasti kehitystyötä harjoitettaisiin vain äärimmäisen tärkeissä säilytyskohteissa, sillä kehitystyön käynnistäminen, ylläpitäminen ja loppuun vieminen ovat pitkäaikaissäilytysorganisaatiolle liian kallista usein suoritettavaksi. (Rothenberg 2000, 16.) Joka tapauksessa on selvää, että juuri tällaisten puutteiden löytäminen on keräilymoduulin tärkeimpiä tehtäviä, sillä puuttuvat komponentit ja niiden kehitystarve määrittelee säilytyksen luotettavuuden pitkällä tähtäimellä, ja siinä mielessä säilytysjärjestelmä ei saa säästää väärissä paikoissa. Standardeja tarvitaan DSEP-järjes-

telmässä siihen, että sovitaan yhteisesti ohjelmistojen paketoimiskäytännöistä ja ohjelmisto ja laitteistovaatimuksien nimeämiskäytännöistä. Tämä helpottaa aineiston sisäänottoa huomattavasti (Rothenberg 2000, 17).

Suurimpia etuja aineiston ja sen emulointiin tarvittavien komponenttien jakamisesta erillisiin paketteihin on se, että sitä voidaan päivittää ilman, että joudutaan millään lailla koskemaan itse aineiston sisältävään säilytyspakettiin. Tämän mahdollistaa se, että metatieto on aineistosta erillään. Näin itse aineisto on suojassa korruptoitumiselta. Metatietoa migroidaan tarpeen mukaan ja metadatatiedostojen sisältämiä linkityksiä päivitetään tarpeen mukaan. Rothenbergin mukaan OAIS-mallin heikkoutena emulointiratkaisun soveltamisen suhteen on, että se suhtautuu DIP-paketteihin staattista aineistoa sisältävinä paketteina, mutta emulointipaketeissa aineisto on useimmiten interaktiivista.

DSEP:in tapauksessa DIP on nähtävä pakettina, joka sisältää sekä interaktiivisen aineiston katseluvälineet, että itse ajettavan aineiston. Asiakkaan tarpeet huomioon ottaen järjestelmä voi tuottaa aineistosta DIP:n, joka sisältää aineiston ja sen ajamiseen tarvittavan emulaattorin. Emulointi voidaan suorittaa myös järjestelmän sisällä ja tuottaa tuon emuloinnin lopputuloksesta käyttökopio asiakkaalle. Käyttökopion ei tarvitse sisältää kaikkia alkuperäisen aineiston käyttöominaisuuksia, riittää kun se täyttää asiakkaan tarpeet. Tämä käyttökopio voidaan tuoda keräilykomponentin kautta SIP:nä järjestelmään sisälle tulevaisuuden vastaavaa tarvetta varten, jotta samanlaista käyttökopiota ei tarvitse tuottaa uudelleen. (Rothenberg 2000, 18.)

Emulointia vaativan aineiston palauttaminen sujuisi aina tässä järjestyksessä: 1) Käynnistetään emulaatiovirtuaalikone ja ladataan siihen emulaatiospesifikaatiotulkkiohjelma. 2) Uutetaan emulaatiospesifikaatio emulaatiospesifikaation sisältävästä AIP:stä ja ladataan tämä emulaatiospesifikaatio tulkkiohjelmaan. 3) Uutetaan käynnistysbittikarttakuva, joka sisältää sovellukset ohjelmistot sisältävistä AIP:stä ja ajetaan tämä emulaattorissa. 4) Uutetaan aineisto omasta AIP:stään ja avataan se sovelluksessa, joka on tuon aineiston käyttöön tarkoitettu. (Rothenberg 2000, 21.)

Asetelma muuttuu hieman monimutkaisemmaksi, jos emulaatiospesifikaatio onkin kirjoitettu vanhemmalla spesifikaatiokielellä, kuin sillä joka on tällä hetkellä käytössä. Silloin edellä lueteltuun palautusjärjestykseen tulisi yksi kohta lisää. Tuossa ylimääräisessä

kohdassa uutettaisiin emulointispesifikaatiodokumentti AIP:stä, ja sen jälkeen se ajettaisiin emulaatiospesifikaatiotulkkiohjelmassa, joka vuorostaan ajettaisiin emulaattorissa aikaisemmassa emulointivirtuaalikoneessa, ja tämä taas ajettaisiin uudemmassa emulointivirtuaalikoneessa. Jokainen sukupolvi aiheuttaa ketjuun yhden toisteisuuden lisää, ja näin mitä pidemmälle tulevaisuuteen mennään sitä enemmän emulaattoreita ajetaan sisäkkäin. (Rothenberg 2000, 21.)

Rothenberg asettaa ratkaisussaan paljon toivoa tietotekniikkateollisuuden mukanaoloon emulointisäilytyksessä. Silloin tietotekniikkateollisuus ja niiden palveluksessa oleva massiivinen koneisto pitäisivät huolen siitä, että emulointispesifikaatiokieliä ja emulointispesifikaatiotulkkeja kehitetään aktiivisesti. Kun tarkastelemme emulointiratkaisua kokonaisuudessaan huomaamme pian, että ilman tietotekniikkateollisuuden resursseja ei laajamittainen tukeutuminen emulointiin ole itse asiassa lainkaan mahdollista. Muutoin DSEP:in harteille tulee valtavan kehitys- ja päivitystyön ylläpito, josta väistämättä tulee lopulta niin kallista ettei se enää kykene rahoittamaan sitä. Näin koko talletusjärjestelmä ja kaikki sinne viety aineisto on vaarassa.

Ohjelmistojen- ja laitteistojen valmistajat voidaan velvoittaa kehittämään emulointiin perustuvan säilytysratkaisun komponentteja. Asiasta voidaan säätää samankaltaisia valmistajia velvoittavia määräyksiä, kuin on voimassa esimerkiksi tietyistä tyyppihyväksyntämerkeistä, joita ilman tuote ei pääse markkinoille. Tällaisten määräysten noudattaminen maksaa paljon laitteistojen ja ohjelmistojen valmistajille. Siksi on odotettavissa, että näihin määräyksiin kohdistuu aluksi suurta vastustusta valmistajien taholta. Positiivisena puolena asiassa on kuitenkin se, että emulaatiospesifikaatioiden, emulaatiospesifikaatiokielten, emulaatiospesifikaatiotulkkien ja emulointivirtuaalikoneiden kehittäminen on loppujen lopuksi hyvin pieni osa siitä työmäärästä, joka käytetään isoissa ohjelmisto- ja muissa tietotekniikka-alan yrityksissä uusien tuotteiden kehittämiseen. Voidaan siis otaksua, että uusien määräyksien täyttäminen ei painaisi resurssikelkassa kovinkaan paljon. Varsinkin kun otetaan huomioon, etteivät nämä määräykset tarkoita, että joutuisivat luovuttamaan yleiseen jakeluun jotakin markkina-asemaansa vaarantavaa.

Nykyisin kaikkien näyttölaitteiden on oltava esim. TCO-sertifikaatin mukaisia, ja tämän sertifikaatin vaatimusten täyttäminen lisäsi näyttöjen kustannuksia. Tämän sertifikaatin

täyttämisen vakuudeksi painettu merkki tuotteissa on mitä ilmeisemmin lisännyt kuluttajien luottamusta tuotteisiin ja siten tuotteiden myyntiä. Voidaan otaksua, että tavaran tuottajat suhtautuivat TCO-sertifikaattiin aluksi hyvin nihkeästi. Uudet asetukset voisi siis mahdollisesti kääntää valmistajien silmissä kalliista velvollisuudesta ja uudesta rajoitteesta kunniamerkiksi, jonka tavoitteleminen on kannattavaa. Luultavasti kuluttajat haluavat panostaa pitkäikäisyyteen kohdattuaan omassa arjessaan lyhytnäköisen suunnittelun haitat jo useaan kertaan. DSEP-arkisto olisi erikoisasemassa näiden spesifikaatioiden suhteen. Valmistajilla olisi velvollisuus lähettää ne DSEP-arkistoon heti niiden kehittämisen jälkeen. Näin varmistettaisiin, että pitkäaikaissäilytysarkistolla on hallussaan tarvittavat spesifikaatiot jo ennen kuin noita spesifikaatioita käyttävä aineisto alkaa saapumaan keräilymoduulin kautta DSEP:iin (Rothenberg 2000, 24).

DSEP:n ongelmana on se, kuinka metatieto itsessään saadaan säilymään lukukelpoisena aikojen saatossa. Olettamuksena on, että metatietoa migroidaan, eikä korruptoitumisella ole merkitystä, kunhan sitä ei tapahdu niin paljon, että tieto alkaa olemaan suorastaan virheellistä tai haitallista. Metatiedolla itsellään ei ole niinkään historiallista arvoa, joten se saa korruptoitua, eikä sen suhteen tarvitse säilyttää alkuperäistä ulkonäköä tms. seikkoja. (Rothenberg 2000, 27.)

Metatiedon Rothenberg jakaa kolmeen luokkaan niiden arvon mukaisesti. Informaatio, jolla on hallinnollinen arvo, informaatio, joka kuvaa yksittäisiä AIP:tä ja informaatio, joka tekee mahdolliseksi pääsyn AIP-paketteihin ja joka yhdistää AIP-paketteja toisiinsa. Näistä ensimmäinen informaatiolaji on Rothenbergin mukaan sellaista, jonka voi menettää ilman että pitkäaikaissäilytys siitä kärsisi. Tämä johtuu siitä, että hallinnollinen tieto voidaan luoda aina uudelleen tai hylätäkin jos sitä ei enää tarvita. Ainakaan sen alkuperäisen olemuksen säilyttämisestä ei tarvitse kantaa huolta. (Rothenberg 2000, 29.)

Kolmas tiedonlaji aiheuttaa järjestelmän suunnittelijoille ongelmia, joihin ei olekaan olemassa niin nopeaa ja helppoa ratkaisua kuin nopeasti ajatellen voisi kuvitella. Nimittäin metatiedot jotka linkittävät AIP:t toisiinsa, ovat verrattavissa käyttöjärjestelmien tiedostopolkuihin. Jos polku on väärä, ei palautus voi onnistua. Ongelmana tämän tiedon kanssa on se, että ajan myötä, myös AIP:den paikka DSEP:ssä voi muuttua, kuten tiedostopolut käyttöjärjestelmissä. Tällöin meidän pitäisi päästä muuttamaan metatietoa,

jonne tuo sijainti on merkitty. Tämä taas johtaisi siihen, että meidän täytyisi käydä läpi kaikki AIP:t joissa tuo sijaintitieto on, ja arkistolla olisi näin edessään valtava päivitysurakka jokainen kerta kun AIP:n sijainti jostain syystä muuttuu DSEP:ssä. Juuri tämän tähden DSEP-järjestelmän kohdalla on tehty päätös metatiedon irrottamisesta varsinaisesta aineistosta. Se säästää järjestelmän ylläpitäjät päivityspainajaiselta, joka olisi laajuudessaan mahdoton toteuttaa. (Rothenberg 2000, 30.)

Digitaalisia entiteettejä voidaan osoittaa useilla eri tavoilla, mutta nämä nimeämistavat voidaan jakaa kahteen luokkaan, semanttisiin ja läpinäkyviin. Semanttiset ovat sellaisia, jotka ovat vaivatta ihmisen ymmärrettävissä kuten osoitteet ja nimet. Ne luovat kuitenkin tulkintaongelman. Esimerkkinä voidaan käyttää vaikkapa katuosoitetta, joka perustuu tämän hetkiseen asuinpaikkaani. Katuosoitteeni ilmaisee tällä hetkellä hyvin tarkasti missä asun, niin tarkasti ettei siitä voi erehtyä. Mutta on hyvin todennäköistä, ettei se ilmaise sitä yhtä tarkasti enää pidemmän ajan päästä. Silloin on saattanut muuttua osoitteiden merkintätapa, ja myös tuon merkintätavan tulkitsemistapa. Sen lisäksi vielä ympäristö, jossa osoitteessa mainittu kohde sijaitsee, on saattanut muuttua siten ettei merkintä johdakaan enää oikeaan paikkaan. (Rothenberg 2000, 30.)

Läpinäkyvän osoituksen tarkoitus on määrittää AIP:lle ID, joka on vain joukko bittejä. Sitä ei tulkita mitään skeemaa käyttäen ja siten se nimeää tietyn AIP:n absoluuttisesti. Yksi bittijono käytetään läpinäkyvän osoituksen menetelmässä vain kerran. Linkki on olemassa kahden paketin välillä, jos niiden ID-kentästä löytyvä bittijono on identtinen. Siksi niitä ei myöskään koskaan tarvitse muokata millään tavalla. (Rothenberg 2000, 30-31.)

6.3.2 OAIS ja DIAS

DIAS (Digital Information Archiving System) on KB:n elektronisten aineistojen talletusjärjestelmä, joka on kehitetty yhteistyössä IBM:än kanssa. Niin kuin monet muutkin pitkäaikaissäilytysprojektit, tämäkin rakennettiin OAIS-mallia pohjana käyttäen. OAIS-mallin mukaisesti myös DIASissa on siis perusmoduulit: vastaanotto-, säilytys-, hallinta-, portti-, hallinto-, valvonta- ja kirjanpitomoduulit. OAIS-malliin on kohdistettu voimakkaita muokkaustoimenpiteitä, jotta se sopisi paremmin DIASin tavoitteisiin. Muokattavuus omia tarpeita vastaavaksi on yksi OAIS-mallin keskeisimmistä eduista.

Diessen (2002c, 3) esittelee säilytysalijärjestelmän, jonka tehtävänä on 1) Tunnistaa vaarassa olevat aineistot. Aineistot ovat vaarassa jos teknologian vanhentuminen uhkaa niitä. 2) Ottaa käyttöön tarvittavat aineistonpalauttamisstrategiat, esimerkiksi emulointi ja migraatio. 3) Rekisteröidä kaikki aineistojen käyttämiseen tarvittava metatieto, josta käy ilmi aineiston palauttamiseen tarvittavat laitteistot ja ohjelmistot. (Diessen 2002c, 3.)

Koska DIASissa säilytyskomponentteja on enemmän kuin alkuperäisessä OAIS-mallissa, digitaalisen aineiston erityisluonne ja varsinkin emuloinnin käyttö osana säilytysjärjestelmää vaatii omat erityisratkaisunsa. Siinä missä alkuperäisessä OAIS-mallissa on aloittavana moduulina lähetyksen vastaanotto, on DIASin versiossa ennen vastaanottoa jakelu/keräys-niminen moduuli, joka tekee talletettavalle aineistolle ennakkotarkistuksen. Ennakkotarkistus on emuloinnin kannalta tärkeämpää kuin monissa muissa aineistoissa, sillä onnistunut aineiston palauttaminen emuloinnin avulla vaatii lukuisten seikkojen huolellista tarkistusta, ja ennakkotarkistuksen tehtävänä on pitää huolta siitä, ettei järjestelmään pääse aineistoa, jota asiakas pyytäessään ei saakaan palautettuna takaisin. Pakkaus/jakelu-komponentin tehtävänä on toimittaa aineisto porttikomponentin kautta DIASista ulos. Se suorittaa aineistolle kaikki ne jälkitoimenpiteet, jotka saattavat olla tarpeen ennen sen luovuttamista aineiston tilanneelle asiakkaalle. (Diessen 2002c, 4.)

Suurin ero OAIS-mallin ja DIAS-järjestelmän mallin välillä on se, että OAIS-mallissa aineiston metatieto on sisällytetty samaan pakettiin aineiston kanssa. DIASissa nämä on erottu toisistaan, jotta talletetun aineiston metatietoja päästään muokkaamaan ilman, että siitä on vaaraa arkistoidun aineiston koskemattomuudelle. DIASissa jokaiselle objektille on varattu oma tiedostotyyppitunniste, joilla nämä erotetaan toisistaan. (Diessen 2002c, 5.)

On tärkeää huomata että DIAS-säilytysjärjestelmässä sekä migroimiselle että emuloinnilla on keskeinen rooli ja näiden kahden säilytysmenetelmän varassa ovat DIASin kaikki aineistot.

Tekninen metatieto jakautuu neljään kerrokseen DIASissa. Tiedostomuotokerrokseen, joka määrittää bittivirran rakenteen, sovelluskerrokseen, käyttöjärjestelmäkerroksen ja laitteistokerrokseen, jossa digitaalinen objekti muutetaan käyttäjälle tarkasteltavaan

muotoon. Tämä kerrostaminen helpottaa digitaalisten objektien riippuvaisuuksien tutkimista. Tätä kerrostettua mallia kutsutaan DIASissa säilytyksenkerrostumamalliksi (preservation layer model). Se on tärkeä työkalu, sillä siitä nähdään, jos jokin kerros on rikki ja uhkaa siten tehdä digitaalisen objektin käyttökelvottomaksi. Säilytyksenkerrostumamalli on pohja näkymäpolkujen rakentamiselle ja tarjoaa kehyksen, johon erilaisia säilytysstrategioita voidaan sovittaa. Säilytyksenkerrostumamalli on OAIS:n tavoin yleiskäyttöinen malli, jota voi hyödyntää digitaalisen aineiston pitkäaikaissäilytyksessä.

Riippuvuussuhteiden tutkinta paljasti, että mikään yksittäinen tietorakenne ei riitä kuvaamaan kaikkien aineistojen riippuvuussuhteita. Tämä antaa tukea DIAS:in päätökselle erottaa tekninen metatieto ja itse aineisto toisistaan. Jos metatieto ja aineisto olisivat samassa tallenteessa, riippuvuussuhteiden muuttuessa jouduttaisiin metatiedot muuttamaan jokaiseen aineistoon erikseen. Kun metatieto on aineistosta erillään, riittää kun riippuvuussuhteita korjataan kerran niiden muuttuessa. Säilytyksenkerrostumamalli auttaa riippuvuussuhteiden tunnistamisessa. (Diessen 2002c, 10.)

Säilytyksenkerrostumamallin laaja käyttöönotto DIAS:ssa kertoo siitä, miten hyvin sillä pystytään kartoittamaan aineistojen riippuvaisuussuhteita. Säilytyksenkerrostumamalli on juuri, josta voidaan johtaa näkymäpolkuja. Se mukailee karkeasti von Neumann-arkkitehtuuria, johon suurin osa nykyajan tietokoneista perustuu. von Neumannin tietokonearkkitehtuurissa on 6 kerrosta, joista jokaisen toiminnallisuus perustuu edelliseen kerrokseen. Jos vertaamme von Neumann arkkitehtuuria ja säilytyksenkerrostumamallia toisiinsa huomaamme, että niiden kerrostuminen noudattaa samaa logiikkaa.

Jokaisella aineistolla on vähintään yksi näkymäpolku, joka paljastaa kaikki tarvittavat tiedostot aineiston palauttamiseen. Mistä DIAS rakentaa sen? Aineisto viedään säilytysjärjestelmään AIP-paketteina. Jokainen AIP-paketti sisältää XML-muodossa olevan sisällysluettelon (table of contents), jossa on määritelty kaikki tiedostot, jotka tuohon AIP:hen kuuluvat. Sisällysluettelosta löytyvät tiedostotyyppitunnisteet, joiden avulla DIAS yhdistää objektin tiettyyn tekniseen metatietoon. Näin AIP-paketteja ei tarvitse enää muokata sen jälkeen kun ne on kerran tuotu järjestelmään.

IBM julkaisi DIAS:sta ensimmäisen version 2002. Alkuaan se tunnisti 30 erilaista tiedostotyyppiä, eli siinä oli tiedostotyyppitunnisteet 30:lle eri tiedostoformaatille. Kun

säilytysjärjestelmään tuodaan uusi tiedostomuoto, tehtävänä on luoda kaikki tunnisteet, jotka tarvitaan jotta tiedostomuoto olisi tuettu. DIAS:ssa se tapahtuu seuraavalla tavalla. Ensin julkaisija lähettää aineistonsa DIAS:iin. Sitten aineistonvastaanottajamoduuli suorittaa aineistolle kaikki tarvittavat valmistelut, jotta se on vastaanottokelpoinen DIAS-järjestelmään. Bibliografinen luokittelija syöttää bibliografiset tiedot aineistosta KB:n luetteloon. Seuraavaksi tekninen luetteloija valitsee aineistotyypille parhaiten sopivat näkymäpolut, joista menee tieto myös säilytysalijärjestelmään. Näkymäpolut ja aineistot yhdistetään toisiinsa tiedostotunnistetyyppien avulla. Jos aineisto on uudenlaisessa tiedostomuodossa, jollaista ei ole aikaisemmin talletettu DIAS:iin, silloin säilytyksenkerrostumamallin valvoja määrittelee uudet näkymäpolut ja säilytyksenkerrostumamallit. Säilytysvirkailija tarkkailee prosessia kokonaisuudessaan ja jos aineistot ovat vaarassa joutua palauttamisen ulottumattomiin, silloin hän päättää otetaanko käyttöön migraatio vai emulointi vaarassa olevien aineistojen kohdalla.

6.3.3 DSEP:n ja DIAS:n vertailua ja yhteenvetoa

DSEP ja DIAS ovat molemmat digitaalisten aineistojen pitkäaikaissäilytysjärjestelmiä. Digitaalisista kirjastoista ne eroavat oleellisesti siten, että niiden toiminnan pääpaino on aineistojen säilyttäminen eikä niiden tarjoaminen suuren yleisön saataville.

Sekä DSEP:ssä että DIAS:ssa on molemmissa muokattu alkuperäistä OAIS-mallia. Tämä on katsottu välttämättömäksi, jotta malli palvelisi rakennettuja pitkäaikaissäilytysjärjestelmiä paremmin. Moduuleja on joko lisätty tai poistettu säilytysjärjestelmän pääasiallisen käyttäjäryhmän etujen mukaisesti. Menettely on sopusoinnussa OAIS-mallin luonteen ja sen takana olevan muokattavuuden idean kanssa. DSEP:n ja DIAS:n lisäksi myös useat muut pitkäaikaissäilytysjärjestelmät ovat ottaneet toimintansa perustaksi OAIS-mallin ja OAIS-mallin käyttöön ottavien tahojen määrä on jatkuvassa kasvussa.

Emulointi on monimutkainen säilytysratkaisu ja vaatii siksi omat arkkitehtuuriset erityisratkaisunsa säilytysjärjestelmään ja siihen sisällettyihin moduuleihin. Juuri tästä syystä DIAS:ssa on ennen talletuksen vastaanottavaa moduulia jakelu/keräys-moduuli,

joka pitää huolta talletettavan aineiston ennakkotarkastuksesta. Tarkastuksen tehtävänä on pitää huolta siitä että talletettava aineisto pystytään palauttamaan emuloimalla.

Myös DSEP:ssä on huomioitu emulointi moduuleja kehitettäessä. DSEP:n erona DIAS:n voidaan mainita se, että siinä tiedostetaan voimakkaammin emuloinnin mutkikkuus palautusmenetelmänä ja siksi emulointi on DSEP:ssä jaettu kaikkien siinä olevien moduulien kesken. DSEP:n erikoisuutena on myös siihen suunniteltu validointipalvelu, jolla aineistojen tallettajat pääsisivät testaamaan palautuksen toimivuutta.

Aineistojen metatietojen suhteen sekä DSEP:ssä että DIAS:ssa on omaksuttu sama politiikka. Metatiedot erotetaan molemmissa järjestelmissä varsinaisesta aineistosta omiksi säilytyspaketeikseen. Näin pidetään huolta siitä ettei itse aineisto vahingoitu kohdistettaessa säilytystoimenpiteitä metatietoihin. Metatieto on myös katsottu sellaiseksi tiedoksi, joka tarpeen vaatiessa kyetään luomaan uudelleen, kun taas itse aineisto on ainutkertaista ja korvaamatonta.

Sekä DSEP:ssä että DIAS:ssa on molemmissa omat mekanisminsa aineistojen identifioimiseen ja riippuvuussuhteiden määrittelyyn. DSEP:ssä käytetään läpinäkyvää osoitusta ja DIAS:ssa säilytyksenkerrostumamallia säilyttämään tieto siitä mitkä tietopaketit kuuluvat yhteen.

7. TAPAUSTUTKIMUKSET

Tässä luvussa tuon hieman empiriaa muutoin kovin teoreettisen tutkimukseni tueksi. Aliluvussa 7.1 esittelen Woodsin tekemää tutkimustyötä. Aliluvussa 7.2 käyn läpi Hollywoodin tuottamaa raporttia, jossa tuotiin esille myös emuloinnin mahdollisuus materiaalin säilyttäjänä. Aliluvussa 7.3 selostan Koninklijke Bibliotheekin tekemiä emulointikokeita. Aliluvussa 7.4 on selostettu Seamus Rossin ja Ann Gowin koeasetelma ja sen lopputulos. Lopulta aliluvussa 7.5 on omakohtainen kokemukseni emuloinnin välttämättömyydestä.

7.1 Demoscenen kulttuuriperinnön säilyttämistutkimus

Kam Woods Indianan yliopistosta kävi läpi 250 Motorola 68000, MOS 6502 ja Intel x86 arkkitehtuureille kirjoitettua demoscenen tuotosta eri vuosikymmeniltä. Hän tutki emulointiajojen vastaavuutta alkuperäisiin. Tutkimuksessaan hän esittelee virhe- ja onnistumistilastoja emuloitavien aineistojen suhteen. Demoscenen tuotosten analysointi tarjoaa ihanteellisen koeasetelman arvioitaessa emuloinnin soveltuvuutta digitaalisen aineiston pitkäaikaissäilytykseen, sillä demoscene on olemassaolonsa aikana tehnyt demoja lukuisille eri laitteistoalustoille. Demot ovat ohjelmistoina luonteeltaan audiovisuaalisia, laitteiston kaikkia komponentteja käyttäviä ja ottavat suorittimista kaiken irti ruohonjuuritason käskytyksellään. Demo-ohjelmien audiovisuaalinen näyttävyys perustuu äänen ja kuvan täydellisen synkronoinnin vaatimukseen, ja siten ne eivät juuri anna anteeksi emulointiohjelmiston tekemiä mahdollisia virheitä. Lisäksi demoja ohjelmoineet ryhmät ovat usein lisänneet teoksiinsa ASCII-grafiikkaa, joka pelkässä binäärimuodossa on merkityksetöntä ja vaatii emulointia näkyäkseen oikein. Toisin kuin monen muun aineiston suhteen, demot ovat vapaasti jaossa tekijöidensä ylläpitämillä sivustoilla ja siksikin niitä on helppo testata laillisuuskysymysten aiheuttamatta haittaa koeasetelmien suorittamiselle.

Tutkimuksessaan Woods käytti avoimen lähdekoodin emulaattoreita, Wineä, E-UAE:ta, Hataria ja VICEä. Hänen pyrkimyksenään oli tutkia voidaanko emulaattoreilla saavuttaa aineistojen suhteen sellainen käytettävyysaste, että aineistoihin pääsee käsiksi yhdellä hiiren klikkauksella.

Woods valitsi Motorola 68000-, MOS 6502-, ja Intel x86 -arkkitehtuurit testiajojen kohteeksi, sillä niille on tehty eniten demoja. Niille on myös saatavilla tarkasti prosessoriarkkitehtuuria jäljitteleviä emulaattoreita. Emulaattoreiksi Woods kelpuutti vain sellaisia ohjelmistoja, jotka suorittavat yhtä monta käskyä kellojakson aikana kuin jäljiteltävä suoritin. Vaatimuksena oli myös, että emulaattorin pitää kyetä tuottamaan virhelokitiedosto, josta voidaan jälkeenpäin käydä läpi virheilmoitukset kohta kohdalta tarkemman kokonaiskuvan saamiseksi. Emulaattorin lähdekoodin tuli olla tutkittavissa, jotta siitä voitiin tarkistaa toteutuuko kellojaksotarkkuus.

Motorola 680xx-suoritinperhettä testattiin E-UAE -emulaattorilla, joka konfiguroitiin vastaamaan Amiga 500+ - ja Amiga 1200 -tietokoneita. Hatari-emulaattoria käytettiin Atari 1024ST-tietokoneen emuloimiseen, joka niin ikään kuuluu Motorola 680xx-suoritinperheeseen. MOS 6502-suoritinperhettä emuloitiin VICE-emulaattorilla, joka on tunnettu kellojaksotarkkuudestaan. Intel x86-perhettä emuloitiin WINE-emulaattorilla, jonka yhtenä valintaperusteena oli se, että sen avulla Windowsista riippuvaista materiaalia voitiin käyttää ilman varsinaista Windows-lisenssiä.

Tutkimuksessaan Woods havaitsi että vaikka emuloitavasta laitteistosta on olemassa tarkat piirustukset, emulaattori toimii aina hieman epätäydellisesti. Woodsin julkaisusta käy ilmi, että toimimisprosentti hänen koeasetelmassaan oli hyvin korkea. Kokeessa näkyy myös selkeä suhde toimivuuden ja emulaattorin kehittämiseen käytetyn panoksen välillä. Mitä kauemmin emulaattori on ollut kehityksen alla, sitä luotettavammin se kykenee ajamaan säilytetyt aineistot.

	E-UAE		HATARI		VICE	WINE
	Amiga 500+	Amiga 1200	Atari ST	Atari STe	C-64	Intel x86
Lataus epäonnistui	6	3	3	2	5	14
Ajo keskeytyi	2	0	0	1	0	1
Video OK Ääni OK	143	168	177	201	212	112
Video virheitä Ääni OK	91	66	41	37	24	31
Video OK Äänivirheitä	6	3	27	9	9	0
Videovirheitä Äänivirheitä	2	10	2	0	0	92

TAULUKKO 2. Woodsin kokeen tulokset. Lähde: Woods, K. 2008.

7.2 Digitaalinen dilemma

Hollywoodissakin on herätty digitaalisuuden aiheuttamiin ongelmiin, eikä syyttä suotta: uudet elokuvat sisältävät yhä enemmän digitaalista materiaalia tai ovat jo kokonaan digitaalisia. AMPAS (Academy of Motion Picture Arts and Sciences) käynnisti kaksivuotisen hankkeen jonka tarkoituksena oli saattaa yksien kansien väliin kaikki, mitä Hollywoodissa hankkeen teon aikana keskusteltiin digitaalisesti tuotettujen elokuvien tulevaisuudesta. Hankkeen tuloksena syntyi tähän mennessä ennenkuulumatonta itsekritiikkiä, jollaista Hollywoodin suunnalta ei ole aikaisemmin julkaistu. Raportin muodossa julkaistun hankkeen lopputulokset ovat kuin avunhuuto, joka esitetään julkisesti siinä toivossa että organisaatiot, joita asia koskee, vastaisivat tuohon huutoon keskittämällä resursseja digitaalisen materiaalin säilytysongelmien ratkaisuun (AMPAS 2007, 56).

Elokuvatuotannossa on laajassa mitassa siirrytty digitaalisuuteen ymmärtämättä tämän teknisen muutoksen aiheuttamia uusia kysymyksiä ja haasteita. Digitaalisia järjestelmiä käyttävien elokuvastudioiden suurin pääoma ovat kaikki ne tuotokset, jotka noita järjes-

telmiä käyttäen on tehty. Jos nämä tuotokset häviävät, häviää niiden mukana myös kaikki pääoma, joka niiden tekemiseen, myyntiin ja markkinointiin on sijoitettu. Tilanne on muuttumassa ja toimiin on ryhdytty, mutta nämä toimet eivät kuitenkaan ole vielä tarpeeksi mittavia ja kattavia, jotta voitaisiin sanoa uhkien hävinneen.

Tutkimuksen avoimuuden mahdollisti se, että lähteitä ei paljasteta. Tämä katsottiin välttämättömäksi sillä monilla tutkimukseen osallistuneista saattaisi olla lausuntojensa takia työpaikka vaarassa. (AMPAS 2007, esipuhe).

Digitaalijärjestelmiä Hollywoodiin myyvä markkinointikoneisto ei ole kiinnostunut myymillään järjestelmillä tuotetun aineiston pitkäaikaissäilytyksestä, eikä ole tehnyt tarpeeksi kauas tähtääviä suunnitelmia sen varalta (AMPAS 2007, esipuhe). Ja miksipä olisikaan sillä pitkäaikaissäilytyksellä ei ole sellaista markkina-arvoa, jonka järjestelmiä tuottava yritys voisi hyödyntää järjestelmiä myydessään.

Aikaisemmin oli mahdollista yksinkertaisesti arkistoida kaikki raakafilmimateriaali. Tämän teki mahdolliseksi filmin hyvät säilyvyysominaisuudet ja säilytyksen helppous. Filmi vaatii säilyäkseen vain viileän ja kuivan tilan. Vaikkakin elokuvien digitaalimasterit talletetaan filmille, tuossa prosessissa varmistetaan vain yhden version arkistoiminen elokuvasta, ei suinkaan koko tuotantoprosessin aikaansaamaa materiaalia. (AMPAS 2007, 1.) Myöhemmin juuri tämä materiaali on se, jolla elokuva saadaan tuottamaan vielä teatterilevityksen loputtuakin. Teatteriversioista yli jäänyt materiaali on kysyttyä tavaraa DVD-levityksissä ja tuosta materiaalista tuotantoyhtiö voi myöhemmin tehdä mahdollisesti uuden version elokuvasta.

Materiaali oli analogisen elokuvatuotannon aikakaudella automaattisesti varmassa tallessa sillä se oli filmillä. Nyt se on tiedostoina, jotka on säännöllisesti migroitava ja uudistettava uusiin formaatteihin ja tallenteisiin tai muutoin tuo materiaali on vaarassa kadota. Tuon materiaalin rahallinen arvo on suunnaton. Jotkin maailmanlaajuisesti toimivat mediayhtiöt ovat myyneet filmiarkistonsa miljoonista ja joissakin tapauksissa jopa biljoonista dollareista (AMPAS 2007, 7).

Analogisen aikakauden etu Hollywoodille on se, että analogisten tallenteiden vanheneminen, hitaasti tapahtuva tallennuksen heikkeneminen, johtaa lopulta vain pohjakohinan ja muiden häiriöiden lisääntymiseen. Digitaalisen tallennuksen puolella sama heikkene-

minen johtaa materiaalin täydelliseen häviämiseen (AMPAS 2007, 9). Siksi tietoa on jatkuvasti migroitava uudemmille tallennusmedioille. Tallennusmedioista saatu tutkimustieto on kuitenkin huolestuttavaa. Esimerkiksi DVD-levyistä vain 50% on laskettu olevan käyttökelpoisia 15 vuoden kuluttua (The X lab 2007, tässä AMPAS 2007, 9).

Digitaalitekniikka on raakafilmikulujen poistumisen myötä antanut elokuvantekijöille mahdollisuuden pitää kameroita päällä ilman pelkoa lisäkustannuksista ja näyttelijöille se on suonut mahdollisuuden hioa suorituksiaan enemmän ilman, että se maksaisi tuotantoyhtiölle ylimääräistä. Kuitenkin on olemassa huoli, että tämä lähdemateriaalin lisääntyminen johtaa lopulta suurempiin yhteenlaskettuihin tuotantokustannuksiin, kun otetaan huomioon materiaalin jälkituotanto ja arkistointi (AMPAS 2007, 12).

On myös havaittu, että joissakin tuotannoissa lähdemateriaalia jää lopulta jäljelle vähemmän kuin analogisella aikakaudella johtuen siitä, että ohjaajat poistavat kuvauspaikoilla ottoja, joille eivät katso olevan käyttöä (Hurwitz 2007, tässä AMPAS 2007, 12). Aikaisemmin kaikki jäi talteen ja käyttökelpoisuusarviointi tapahtui vasta kun asiaan oli saatu enemmän perspektiiviä. Digitaalisuus voi siis tällä tavoin tehdä luovalle työlle myös vahinkoa, sillä aikaisemmin ainut tapa hävittää kuvattu kohtaus oli tuhota filmi, eikä siihen koskaan ryhdytty paikan päällä vaan vasta jälkikäteen. Kun kuvaustilanne on ohi ja ohjaaja on leikkaajan kanssa leikkauspöydän ääressä, saattaa hänen näkemyksensä otosten mahdollisesta turhuudesta olla jo aivan toisenlainen.

”Digital Dilemma”-raportissa tultiin lopulta siihen tulokseen että digitaalisen ja analogisen version arkistoimisen välillä samasta elokuvasta on 11-kertainen kustannusero. Digitaalinen master tulee lopulta maksamaan tuotantoyhtiölle niin paljon, että tästäkin syystä studiot pitävät kiinni filmillä säilyttämisestä niin pitkään kuin mahdollista. (AMPAS 2007, 43.)

Elokuvan arkistoimisessa filmille on useita etuja. Elokuva voidaan varastoida filmille ihanteellisiin olosuhteisiin ja siellä se pysyy käyttökelpoisena, vaikka sen tuottanut yhtiö hakeutuisi konkurssiin tai vaikka aineisto säilytys laiminlyötäisiin unohtamalla se varastoon. Digitaalisella elokuvateollisuudella ei toistaiseksi ole lainkaan arkistoa, vain kirjasto josta aineistoa käytetään. Kirjastolla ja arkistolla on elokuvateollisuudessa valtava ero, arkisto on suunniteltu pitkäaikaissäilytystä varten ja kirjasto taas väliaikaista

käyttöä varten. (AMPAS 2007, 1.) Koska erilaiset fotokemialliset tuotteet ovat maailmanlaajuisesti olleet käytössä jo yli sata vuotta ovat analogisen arkistoinnin yksityiskohdat eri puolilla maailmaa hyvin tunnettuja. Myöskään formaattisotaa ei ole sillä formaatiksi on vakiintunut 35mm filmi. (AMPAS 2007, 6.) Alkuperäiseen masterfilmiin tarvitsee kajota vain hyvin harvoin ja se saa levätä rauhassa arkistointiholvissaan.

Nykyisin elokuvat toimitetaan teattereihin filmiesityskopion lisäksi varustettuna DCP:llä (Digital Cinema Package). Suuri ero tämän filmiesityskopion ja DCP:n välillä on se, että jälkimmäisestä ei kukaan tällä hetkellä elokuvateollisuudessa tiedä, miten sen elinikä saataisiin samanlaiseksi kuin filmiesityskopion. On olemassa vaara, että myöhemmin tulevaisuudessa asiakkaalle toimitettu DCP ei enää toimi, sillä laitteita joilla sen voisi toistaa ei enää ole, ja uudemmat laitteet eivät tuota formaattia enää ymmärrä. Jos arkistointiholveihin viedään digitaalista materiaalia tulevaisuutta murehtimatta, saattaa tuotantoyhtiön kaikkein arvokkain pääoma, sen tuottamat elokuvat, verrattain nopeasti lakata olemasta olemassa.

Digitaalisen materiaalin suhteen on olemassa sellainen dilemma, että vaikka materiaali olisi, sitä ei kuitenkaan välttämättä ole, jos digitaalisen tiedon käyttökelpoisuudesta ei ole riittävässä määrin huolehdittu. Aineisto ehkä kyetään palauttamaan vaikka näinkin olisi päässyt käymään, mutta tuo palauttaminen on niin kallista ja hidasta, että käytännössä siihen ei kaikkien aineistojen kohdalla ryhdytä. Näin osa materiaalista yksinkertaisesti jätetään vanhentumaan entisestään, kunnes se jonkin omistajanvaihdoksen tms. yhteydessä todennäköisesti hävitetään tilaa viemästä.

Näin siis lopputuloksena on vaatimus siitä, että digitaaliselle elokuvamateriaalille on löydettävä arkistointiratkaisu, joka takaa sille säilyvyyden käyttökelpoisena vähintään yhtä kauan kuin filmimateriaalilta voidaan odottaa eli 100 vuotta. AMPASin suorittaman tutkimuksen hälyttävä tulos on, että tällaista arkistointiratkaisua ei tällä hetkellä ole olemassa missään päin maailmaa ja kaikissa alan organisaatioissa pähkäillään saman ongelman kanssa. Niinpä tämän hetken käytäntö onkin, että jokaisesta synnynnäisesti digitaalisesta produktiosta teetetään myös filminegatiivi, jonka varaan elokuvan pitkäaikaissäilytys lasketaan. (AMPAS 2007, 8.)

Elokuvan valmistuksen viimeinen vaihe on masternegatiivifilmin luominen. Kodak toi markkinoille 1992 Cineon-nimisen järjestelmän, joka kykeni muuntamaan analogisen filmin digitaaliseksi tiedoksi, prosessoimaan sen ja sitten siirtämään sen takaisin filmille. Yli puolet elokuvista masteroidaan käyttäen tätä digitaalista välivaihe menetelmää. (AMPAS 2007, 10-11.)

Tutkimustaan varten AMPAS haastatteli lukuisia elokuva-alan edustajia, jotka ovat olleet mukana elokuva-alalla ennen digitalisoitumista, sen aikana ja sen jälkeen. Digitaalisen teknologian käyttö ei sinänsä ole mitään uutta elokuvien tuotannossa, mutta se on, että itse elokuva on digitaalisessa muodossa, eikä filmiä enää käytetä. Digitaalisen teknologian käyttö elokuvan tuotannossa löi itsensä läpi Jurassic Park -elokuvan myötä. Alkuperäinen suunnitelma oli käyttää perinteisiä animointitekniikoita ja pienoismalleja, mutta kokeilut digitaalisten tekniikoiden parissa osoittautuivat lopulta niin lupaaviksi, että tehtiin päätös luoda dinosaurukset täysin digitaalisesti. Näin Jurassic Parkista tuli ensimmäinen elokuva, jossa digitaaliset hahmot olivat keskeisessä roolissa. Lopputulos oli hyvin onnistunut sillä elokuvaa katsoessaan suuri yleisö unohti katsovansa tietokoneella luotua lopputulosta. Tämän jälkeen ilmestyi Toy Story 1995, joka oli ensimmäinen täysin digitaalisesti tuotettu elokuva. Sen jälkeen elokuvissa on turvauduttu lähes yksinomaan digitaalitekniikkaan. (AMPAS 2007, 10.)

Digitaalisen materiaalin arkistoimisen tavoitteena on aineiston käyttö ilman rajoitteita ja säilytys ilman korruptoitumista. Nurinkurista on se, että analoginen aineisto säilyy korruptoitumattomana kunhan sen annetaan olla rauhassa säilytyspaikassaan, kun taas digitaalinen aineisto nimenomaan korruptoituu jos sen annetaan vain olla rauhassa säilytyspaikassaan! Maailmassa on tiedon tuotannon suhteen tapahtunut valtava eskalaatio. Vuonna 2002 maailmassa tuotettiin arviolta 5 eksatavua (5 miljardia gigatavua) tietoa eli noin 5,5 triljoonan kirjan verran tallennettuna paperille, filmille, magneettisille ja optisille tallenteille. (Lyman & Varian 2003, 1-112 tässä AMPAS 2007, 3.)

Koska digitaalisen tiedon varjopuolena on se, että sitä päästään tarkastelemaan ja käyttämään vain teknologian kautta, kun taas perinteisen tiedon käyttöön riittää pelkät silmät, tuo tiedonräjähdys mukanaan myös vaaran tiedon häviämisestä. Tämä johtuu siitä, että pääsy digitaaliseen tietoon maksaa. Mitä enemmän tuota tietoa on, sitä enemmän kustannuksia aiheutuu pääsyn järjestämisestä. Käytännössä tämä lopulta nielee joko

suuret määrät rahaa tai tietoa. Digitointiprojekteista voikin tulla tiedon mustia aukkoja, sillä organisaatiot saattavat rahapulassa jättää kalliit konvertointi- ja migrointitoimenpiteet tekemättä. Tällöin digitaaliseen muotoon muutettu analoginen tieto muuttuu ajan kuluessa käyttökelvottomaksi (AMPAS 2007, 4.)

Juuri tästä syystä esim. EROS (The Center for Earth Resources Observation and Science) joka kasvattaa hallussaan pitämäänsä tietomäärää päivittäin 2 teratavun vauhdilla, migroi hallussaan olevansa tiedon 3-5 vuoden välein. Aineisto koostuu maata kiertävien satelliittien välittämästä tiedosta maan kuorien liikkeistä ja erilaisista ilmakuvista. Heidän varmennuksensa on kolminkertainen: ensimmäinen kopio aineistosta on sijaitsee robottinauhavarmennusjärjestelmässä, toinen kopio kellareissa ja kolmas kopio ajomatkan päässä olevassa talletuspaikassa. Ajomatkan päässä siksi, että syyskuun 11.2001 sai heidät huomaamaan että tämä kolmas varasto kannattaa pitää ajoetäisyyden sisällä. Lentoliikenne saattaa häiriintyä pitkiksikin ajoiksi.

EROS pitää hallussaan laajaa kokoelmaa filmimateriaalia organisaation aikaisemmilta vuosilta ja on arkistoinut ne samoin kuin Hollywoodissa tehdään. Kysymys kuuluu, kun filmi tulee säilytystiensä päähän, tehdäänkö siitä silloin uusi filmikopio vai siirretäänkö aineisto pysyvästi digitaaliseen muotoon? Eroksen politiikka on skannata filmimateriaalia digitaaliseen muotoon tarvittaessa, mutta se ei ole halpaa toimintaa. Yhden filmiruudun skannaus tuottaa noin 800 mb tietoa ja maksaa Erokselle 20-30 dollaria. On tehty mielenkiintoinen havainto liittyen digitaalisen tiedon käyttämiseen. Digitaalisen aineiston käyttökustannukset ovat verrannollisia siihen kuinka monta kertaa tiedostoja magneettinauhoilta luetaan ja mitä useammin dataa nauhoilta luetaan, sitä suurempi riski on menettää sitä. (AMPAS 2007, 26-27.)

Onko digitaalisuuden hyvänä puolena parempi kuvanlaatu ja siten nautinnollisempi elokuvakokemus? Asia ei ole ihan näinkään yksiselitteinen. Nimittäin tosiasiassa suurimmassa osaa digitaalisissa elokuvateattereissa esitetään elokuvat 2K-formaatissa, joka on vain puolet siitä mitä 35mm filmi aikoinaan tarjosi kaikille katsojille kaikissa teattereissa. Lähimpänä 35mm:n filmiä on 4K:n digitaalimasteri, mutta sellaisen tekeminen on kalliimpaa, ja koska suuri yleisö on tyytynyt 2K-formaattiin, eikä edes ehkä osaa vaatia parempaa kun ei tiedä paremmasta, on 2K vakiintunut elokuvissa yleisesti käytettäväksi

formaatiksi. Suomessa tällä hetkellä ainut 4K:n elokuvateatteri on KAVA:n tiloissa, mutta yhtään sellaista ei ole kaupallisessa käytössä.

Kuitenkaan edes 4K ei vastaa tarkkuudeltaan 35 mm:n filmiä. Arkistokäytössä 35mm:n filmiä vastaavan digitaalikopion luomiseen tarvittaisiin 8K:n superskanneri, mutta näillä näkymin sellaista ei olekaan tulossa markkinoille, vaikka niin joskus puhuttiin. Jos filmikopioita ei arkistoihin enää tule, on arkistojen haltuun jäävä elokuvakopio huonompilaatuinen kuin mitä 35mm:n filmikopiot olivat ennen digitaaliaikaa 40 vuotta sitten ja sisältää vähemmän informaatiota (AMPAS 2007, 14). Asia muuttuu vielä hullunkurisemmaksi kun muistamme että digitaalimasterin arkistointi tulee 11 kertaa kalliimmaksi kuin analogisen (AMPAS 2007, 2).

Raportissa todetaan, että toistaiseksi filmiä vastaavaa pitkäaikaissäilytysratkaisua ei digitaalimastereille ole. Yleiseksi käytännöksi alalla onkin muodostunut, että mastereita pyritään säilyttämään niille mahdollisimman suotuisissa olosuhteissa siinä toivossa, että ratkaisu löydetään myöhemmin (AMPAS 2007, 51). Tallennusvälineitä tai ohjelmistoja ongelman ratkaisemiseen ei toistaiseksi kerta kaikkiaan ole. Monissa elokuvastudioissa ollaan sitä mieltä että ratkaisu digitaalisen aineiston pitkäaikaissäilytykseen on löydetty vasta kun on olemassa ratkaisu, joka sallii materiaalin varastoimisen ilman jatkotoimenpiteitä filmin tapaan.

Mielenkiintoista on myös, että audiovisuaalisella alalla tulee laiminlyötyä metatiedon luominen. Materiaali luodaan ensin ja metatieto kytketään siihen vasta jälkeenpäin. Terveydenhuollossa tilanne on juuri päinvastainen: kaikki potilaan tiedot syötetään ennen kuvien ottamista, ja siten metadata on niissä heti materiaalin syntymästä saakka mukana. Raportissa vihjataankin, että elokuva-ala voisi ottaa tässä oppia terveysalan käytännöistä. Toisaalta elokuva-alan on vältettävä se karikko, jossa terveysala digitaalisten materiaalien suhteen kävi 1970-luvulla, jolloin digitaaliset kuvauslaitteet ensimmäisen kerran rantautuivat terveydenhoitoalalle. Silloin kaikilla valmistajilla oli omat suljetut tiedostoformaattinsa, ja asiakkaat saatiin sitoutumaan tiettyihin järjestelmätoimittajiin. (AMPAS 2007, 25.)

Digitaalisten aineistojen suhteen Hollywoodissa tuotantoyhtiöt pitävät itsestäänselvyytenä sitä, että tulevaisuudessa aineiston digitaalinen luonne hyödyntyy nimenomaan niihin

kytketyn metatiedon kautta. Haaveillaan siitä, että organisaation sisäisestä verkosta voidaan elokuvan nimellä hakea kaikki siihen liittyvä materiaali ja ladata materiaalin tarkasteluun valtuutetun työntekijän tietokoneelle. (AMPAS 2007, 15.)

Eräs varoittavimpia esimerkkejä audiovisuaaliselta alalta on U-matic -nauhaformaatin esittely 70-luvulla. Joissakin tv-tuotantoyhtiöissä vastaanotto oli niin innostunutta, että alkuperäiset 16mm filmit hävitettiin turhaan tilaan viemästä. Myöhemmin tämä osoittautui pahaksi virheeksi, sillä U-maticille konvertoidut filmit eivät säilyneetkään formaatin osoittauduttua hyvin epäluotettavaksi. Näin uutistoimitusten arkistoihin syntyi aukkoja. (AMPAS 2007, 19.) Läksystä otettiin kyllä opiksi sillä 1956 2” nauhan markkinoille tulon jälkeen on nähty yli 60 videonauhaformaattia. Nyt kaikille on tullut selväksi ettei mikään näistä ole pysyvä tai luotettava tallennusväline. (AMPAS 2007, 19.)

Tutkimuksesta varten tehdyissä haastatteluissa öljy-yhtiöiden edustajien kanssa kävi ilmi, että geologisilla aloilla ja elokuvateollisuudella on paljon yhteistä datan tuottamisen alueella. Molemmilla aloilla tuotetaan suuria datamääriä, joilla itsessään sellaisenaan ei ole kovinkaan suurta arvoa, vaan ne saavat arvonsa vasta jatkokäsittelyn kautta. Öljyalalla geologiset skannaukset saavat suurimman arvonsa vasta jälkeenpäin, kun skannauksien prosessointitekniikoiden kehittyessä niistä saadaan uutettua enemmän ja tarkempaa tietoa. Öljy-yhtiöiden ongelmat ovat hyvin samankaltaisia elokuva-alan kanssa. Ne ovat myös riippuvaisia toimittajiin sidotuista ratkaisuista, tietyistä tiedostoformaateista eikä kenellekään ole suunnitelmaa siitä miten aineiston säilyminen tulevaisuudessa turvataan. (AMPAS 2007, 21.)

Emuloinnista raportissa annetaan myönteinen kuva, mutta sen varjopuoleksi mainitaan suuret kustannukset ja jatkuvan kehitystyön tarve, eli kaikki se mikä saa juuri elokuva-alan ihmiset pysymään filmikopioissa, ne kun eivät jatkotoimenpiteitä tarvitse säilyäkseen. Raportin mukaan emulointi vaatii jatkuvaa kehitystyötä, jotta se saadaan pidettyä toimintakuntoisena ja tämä tulee hyvin kalliiksi. Kuitenkin raportti antaa tunnustusta onnistuneille emulointiprojekteille kuten Camileonille ja UVC:lle, mutta toteaa, että arkistot yleisesti ovat pyrkineet kehittämään migraatiota. Raportissa todetaankin, että monien mielestä emulointi on vain migroinnin monimutkaisempi muoto. (AMPAS 2007, 38-39.) UVC:n kohdalla tämä pitää paikkansakin sillä siinä yhdistyvät sekä emulointi että migrointi yhdeksi tehokkaaksi kokonaisuudeksi.

Elokuva-alan tuottaman aineiston luonteen vuoksi voisi ajatella, ettei emuloinnilla ole sille paljon annettavaa. Tämä ei kuitenkaan pidä paikkaansa, sillä kaiken digitaalisen aineiston tarkasteluun tarvitaan aina katselinsovellus, joka tulkitsee digitaalisen tiedon. Siksi on väistämätöntä, että tuota katselinsovellusta on jossain vaiheessa emuloitava. Elokuvateattereissa käytetyissä digitaaliprojektoreissa on tietokoneet, joiden toimintaa on tulevaisuudessa emuloitava, sillä alkuperäiset laitteet menettävät pikkuhiljaa toimintakykynsä. Raportissa todetaankin että emulointi on jäänyt migroinnin alle ja sen tutkimus on laiminlyöty (AMPAS 2007, 39).

Näin voidaan todeta raportin vahvistavan olettamukseni siitä, että emuloinnin tutkimusta ja testaamista on laiminlyöty kuin myös sen, että vaikka emulointia ei olekaan pitkäaikaissäilytyksessä paljoa tutkittu, sen toimivuutta ei voida kiistää. Näin AMPASin julkaisema tutkimusraportti Hollywoodin digitaalisten aineistojen tilasta antaa syyn jatkotutkimukseen ja erityisesti sen seikan selvittämiseen, voisiko emuloimalla saada aikaan autenttisempia esityksiä vanhoista aineistoista kuin migroimalla?

7.3 Koninklijke Bibliotheekin koeasetelmat

Ehkä varhaisin virallinen tutkimus emuloinnin kyvyistä digitaalisen aineiston palauttamisen suhteen oli Koninklijke Bibliotheekin vuonna 2000 julkaisema tutkimus "An Experiment in Using Emulation to Preserve Digital Publications". Koninklijke Bibliotheek suoritti koesarjan, jonka se tarkoitti ensimmäisen vaiheen tutkimukseksi emuloinnin evaluoinnissa pitkäaikaissäilytysratkaisuna. Koska koesarja oli ns."ensimmäistä aaltoa", oli siihen saatavissa vain valmiita emulaattoreita, ei sellaisia emulointisovelluksia, jotka olisi nimenomaisesti suunniteltu pitkäaikaissäilytystä silmällä pitäen. Tarkoituksena olikin määrittää tuleville tutkimuksille sovellettava kehys, jonka pohjalta tutkimustyötä voitaisiin tehdä ja määrittää mitkä ovat ne seikat, joiden perusteella emulointitutkimuksen painoarvoa tutkimuskentällä voidaan arvioida. Toisin sanoen, se asetti pohjavaatimukset sille, millaista on sovelluskelpoisia tutkimustuloksia tuottava emulointitutkimus. (Rothenberg 2000.)

Emulointitutkimuksessa suosittiin Barry Boehmin (1988) esittelemää spiraaliprosessia, jossa jokaisen koekierroksen tavoitteet asetetaan edellisestä koekierroksesta saatujen tulosten perusteella. Näitä koekierroksia Boehmin mallissa kutsutaan spiraaleiksi ja tämä menettely on laajalti käytössä ohjelmistojen kehittelyssä. Spiraalimalli sopii oivasti emuloinnin tutkimukseen, sillä kukin spiraali paljastaa menettelytavoista heikkouksia, joiden korjaaminen asetetaan seuraavan spiraalin tavoitteeksi.

Rothenberg (2000, 74-75) määrittelee tietyn vaihteluvälin emulaation lopputuloksessa, joka on vielä hyväksyttävissä autenttisuuden ehdon täyttymiseksi. Tämä vaihteluväli on hänen mukaansa verrattavissa siihen vaihteluun, jonka aineisto sai alkuperäisissä laitteistoissaankin eri käyttäjillä. Emulointia ei siis heti suoralta kädeltä luokitella epäsopivaksi jonkin materiaalin suhteen, vaikka emuloinnin tuottama lopputulos ei näyttäisi/kuulostaisi täysin identtiseltä. Rothenberg nimittää tätä synnynnäiseksi vaihteluväliksi.[17] Synnynnäisen vaihteluvälin määritteleminen on välttämätöntä, jotta voidaan perustella ja dokumentoida, mitä emulointikokeissa saadut tulokset jatkossa tehtävien päätösten kannalta merkitsevät. (Rothenberg 2000, 74-75.)

Kaiken kaikkiaan koeasetelman tulokset olivat hyvin lupaavia, vaikka itse asetelma jäikin aiottua laihemmaksi resurssipulan takia (Rothenberg 2000, 77). Emulointi osoitti tuottavansa esityksiä alkuperäisistä aineistoista, jotka pysyttelivät hyväksyttävän vaihteluvälinsä rajoissa. Se oli jopa niin tarkkaa, että Macintoshissa ajettu Windows-emulaattori jumittui samoissa kohdissa, joissa Windows ilman emulointiakin jumittui. Joissakin tapauksissa emuloitu ympäristö osoittautui jopa vakaammaksi kuin alkuperäinen, ilman emulointia ajettu ohjelmisto. (Rothenberg 2000, 79.)

Rothenbergin johtaman testauksen tärkeintä ja rikkainta antia on emulointitestikehyksen laadinta. Se synnytti testausproseduurin, jota voidaan soveltaa tulevaisuudessa tehtäviin koesarjoihin. Rothenberg piti hyvin tärkeänä sitä, että laadittiin eräänlaisia ”tyypillisen käytön”-skenaarioita. Niiden määrittelemässä viitekehyksessä arvioitiin emuloinnin tuottaman esityksen vastaavuutta alkuperäisessä ympäristössä ajetun ohjelmiston tuottamaan esitykseen. (Rothenberg 2000, 80.) Yksi tällainen ”tyypillisen käytön”-skenaario

17 Native range of variability

voi olla pätevä useiden erilaisten aineistojen kohdalla. Toisaalta yksi aineisto voi vaatia useita ”tyypillisen käytön”-skenaarioita tullakseen edustetuksi tarpeeksi kattavasti.

Kattava emulointitestaus vaatii Rothenbergin mukaan seuraavat elementit: aineiston; joukon ”tyypillisen käytön”-skenaarioita; alkuperäisen ohjelmiston, jota käytettiin aineiston esittämiseen; alkuperäisen käyttöjärjestelmän, jossa aineiston esittämiseen käytetty ohjelmisto ajettiin; alkuperäisen laitteistokonfiguraation ja alkuperäisen laitteiston ohjelmistoineen testiaineistolle. Näiden avulla tehtiin vertailua siitä kuinka hyvin emuloitu esitys vastasi alkuperäisen kokoonpanon samasta aineistosta tuottamaa esitystä. (Rothenberg 2000, 81.)

Tärkeä osa vakavasti otettavaa emulointitestausta oli ns. ”tyhjien” testien suorittaminen muiden testien välissä. Näissä testeissä sekä kohde että isäntäalusta olivat samat. Aineiston käyttäytymistä alkuperäisellä alustalla verrattiin sen käyttäytymiseen toisella koneella alkuperäisellä alustalla ja aineistolla. Testissä täysin samanlaista esitystä verrattiin toiseen täysin samanlaiseen. Tällaisen toiminnan tarkoitus oli tutkia testiprosessin itsensä luotettavuutta ja sille oletettuja ”tyypillisen käytön”-skenaarioita. Kaikkein ihanteellisinta oli jos tämä saatiin suoritettua tuplasokkotestauksena siten, että testinsuorittajat ja sen valvojat olivat yhtä tietämättömiä siitä, että testi oli ns. ”tyhjä”. (Rothenberg 2000, 81.)

Itse koeasetelmien suhteen hankaluutena oli valita testijoukkoon mukaan aineistoja jotka edustaisivat mahdollisimman hyvin sellaista aineistoa, jotka olivat tärkeitä ja yleisiä säilytysjärjestelmissä. Kriteereiksi luetellaan raportissa seuraavat: Aineiston tulee edustaa kattavasti loogisia formaatteja, interaktiivisuutta ja standardeja, joita talletusjärjestelmä tukee; aineiston tulee edustaa hyvin muistiorganisaatioiden tyypillistä ja eniten käytettyä aineistoa. (Rothenberg 2000, 60-61.) Tällainen työ oli jo itsessään digitaalisten aineistojen kannalta merkittävää aikaisemman viitekehyksen puuttuessa ja siksikin siihen kannatti ryhtyä.

Oikeassa testissä isäntä-alusta oli erilainen kuin kohdealusta ja alkuperäisen ohjelmiston ajamiseen käytettiin emuloitua kohdealustaa. Ideaalitilanne oli sellainen, että koe saatiin järjestettyä siten, että kokeen tarkkailijat ja sitä suorittavat koehenkilöt olivat tietämättömiä siitä, oliko kyseessä emuloinnilla aikaansaatu aineiston esitys vai alkuperäisessä ko-

koonpanossa toimiva aineiston esitys. Vaikeutena tässä oli se, että jos sekä emuloiva kone, että alkuperäistä kokoonpanoa edustava kone olivat yhtä tehokkaita, silloin emuloiva kone paljasti itsensä emuloinniksi hitaudellaan. Jotenkin tuli siis järjestää se, että emuloitu esitys ja alkuperäinen esitys olisivat nopeudeltaan samanlaisia, eikä niitä siten kyennyt toisistaan erottamaan. Tähän voitiin vaikuttaa useilla eri tavoilla, joista yksi oli se että emuloiva kone oli aina sen verran tehokkaampi kuin alkuperäistä kokoonpanoa edustava, että emuloinnin aiheuttama hidastus tulisi siten kompensoitua. Vaihtoehtoisesti voitiin hidastaa alkuperäistä kokoonpanoa edustavaa laitteistoa ohjelmallisesti siten, ettei eroa voinut havaita. (Rothenberg 2000, 82.)

Kaikenkaikkiaan tutkimus tuotti vahvan empiirisen evidenssin emuloinnin soveltuvuudesta pitkäaikaissäilytyksen tarpeisiin. Osoittautui, että emulointitestaukseen käytetty viitekehys on käyttökelpoinen. Tutkimuksen tekijät yllättyivät siitä, että jopa valmiita emulaattoreita käyttämällä saatiin lopputulokseksi esitys, jota ei kyennyt erottamaan alkuperäisestä. Paikoitellen emuloinnin alla ajettu aineisto käyttäytyi jopa vakaammin kuin alkuperäinen (Rothenberg 2000, 79). Se on erittäin vahva osoitus siitä, että emuloinnissa on potentiaalia digitaalisen aineiston pitkäaikaissäilytyksen ongelmien ratkaisuun ja että lisätutkimukset ovat perusteltuja. (Rothenberg 2000, 6.)

7.4 Seamus Rossin ja Ann Gowin koeasetelma

Seamus Ross ja Ann Gow tekivät v.1999 kattavan tutkimuksen data-arkeologiasta. Yksi osa tutkimusta oli emulointikoeasetelma. He emuloivat PC:llä Sinclair Spectrum-tietokonetta. Kohdekoneena oleva PC oli 100mhz prosessorilla, 16MB RAM-muistilla, 1.5 GB kovalevyllä, SVGA-näytöllä ja Soundblaster-äänikortilla varustettu laitteisto, jossa käyttöjärjestelmänä oli Windows 95. Nettiyhteys otettiin kokeilun ajaksi pois päältä. (Gow & Ross 1999, 33.)

Gerard Sweeney, intohimoinen Sinclair-harrastaja ja harrastelijaemulointiohjelmistojen kirjoittaja, oli mukana koeasetelman tuottamisessa. Kokeeseen valittiin mukaan satun-

naisesti joitakin nauhoja Sweeneyn kokoelmista. Palautettava aineisto oli c-kaseteille tallennettua, osa tavallisille audionauhoille ja osa datanauhoille. Osaan kaseteista ei ollut merkitty mille Sinclair-tietokoneelle kirjoitettua aineistoa ne sisälsivät. Tässä Sweeney monen vuoden kokemuksesta harjaantunut kyky erottaa korvakuulolta, mille laitteistolle nauhalla oleva aineisto on kirjoitettu, osoittautui erittäin hyödylliseksi. Eri Sinclair-laitteistoilla on erilainen latausohjelma kasetin alussa, ja soittamalla kasettia kasettinauhurissa pystyy eron kuulemaan. (Gow & Ross 1999, 33.) Tämä on juuri sellaista tietoa, jollaista voi olla vain johonkin laitteistoon erikoistuneilla henkilöillä omakohtaisen kokemuksen kautta hankkineet. Tällainen tieto tulisi tallettaa tulevaisuudessa aineistoja palauttavien ihmisten käytettäviksi.

Jotta kaseteilla olevaan tietoon olisi päästy käsiksi, piti jotenkin saada johdettua kaseteilla oleva tieto PC-koneeseen. Tähän tarkoitukseen he valitsivat ihan tavallisen kasettinauhurin, joka yhdistettiin erikoiskaapelilla PC:n sarjaporttiin. (Gow & Ross 1999, 33.)

Varsinainen prosessi jakaantui kahteen vaiheeseen, jossa ensimmäisessä siirrettiin kaseteilla oleva tieto PC-koneeseen käyttäen hyväksi freeware-ohjelmistoja. Siirtoon käytettiin aivan tavallista jakkiliittimillä varustettua audiokaapelia, jonka toinen pää kytkettiin kasettinauhuriin ja toinen pää Soundblaster-äänikortin sisäänmenoliittimeen. (Gow & Ross 1999, 34.)

Kokeessa käytettiin kahta ohjelmistoa, joista toisen tehtävänä oli muuntaa kasetilta luettu tieto emulaattorille sopivaksi ja toisen ohjelmiston tarkoitus oli hoitaa itse emulointi. Näin aineistonpalautusprosessi jakautui kahteen vaiheeseen. Ensimmäisessä vaiheessa nauhalla oleva tieto siirrettiin Sounblaster-äänikortin kautta tietokoneeseen. Kaseteilla tieto oli lohkoissa, jotka Tape2Tap-ohjelmisto luki ensin yksittäisiksi lohkoiksi ja lopuksi kokosi yhdeksi .TAP-tiedostoksi. Ensimmäinen nauha oli vahingoittunut ja se oli vaihdettava toiseen. Toisella nauhalla siirto onnistui ja Tape2Tap-ohjelmisto loi Ballcraz.tap-nimisen tiedoston kovalevylle. Tape2Tap-ohjelmisto helpotti tiedon siirtoa ilmoittamalla värikoodeilla millaista tietoa on menossa. Punainen väri merkitsi, että luetaan otsaketta ja keltainen, että luetaan lohkoa. (Gow & Ross 1999, 34.)

Toisessa vaiheessa käynnistettiin itse emulointi. Z80-niminen emulaattoriohjelmisto valittiin tähän tehtävään. Erittäin helppokäyttöinen ohjelmisto mahdollisti BallCrazy-tieto-

konepelin käynnistämisen emulaattorissa muutaman klikkauksen jälkeen. Vaikka mukana oli Sinclair-tuntija Gerald Sweeney, ei tässä emulointikokeessa ollut mukana mitään sellaista elementtiä, joka estäisi ketä tahansa tekemästä samaa koetta omalla tietokoneellaan. (Gow & Ross 1999, 35.)

Läpikäyty lyhyt ja yksinkertainen koeasetelma osoittaa, että emulointitekniikka on nykyään hyvin kehittynyttä ja on vaikea ymmärtää, mitkä asiat voisivat estää käyttämästä sitä muistiorganisaatioissa.

7.5 Tapaus Frame Graphics

Työskennellessäni KAVA:ssa tätä gradua suunnitellessa tuli vastaan tapaus, joka vahvistaa käsitystä siitä, että ilman emulointia voidaan todella menettää tietoa. Frame Graphics-niminen mainoselokuvia valmistava yritys teki talletuksen KAVAan ja toimitti tietokannan johon oli viety tiedot tuotetuista elokuvista. Kävi nopeasti ilmi ettei tietokannan käyttäminen noin vain onnistukaan, tai ainakaan tietojen tuominen siitä sellaisenaan.

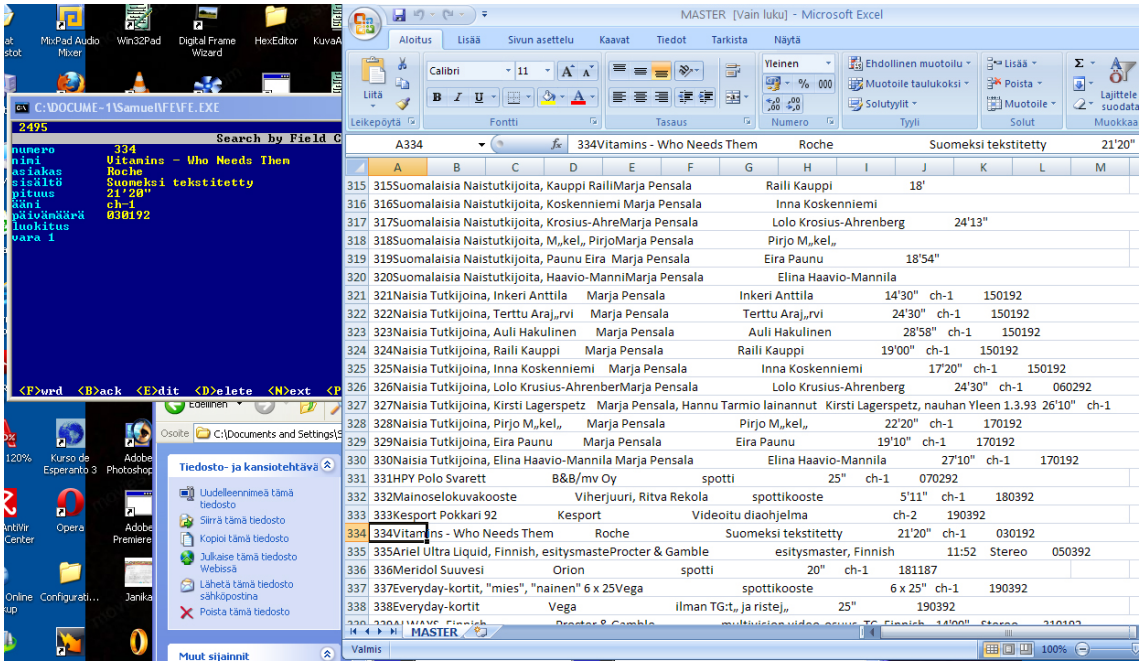
Ohjelmisto oli yhdysvaltalaisen Expressware Corporationin valmistama File Express v.4.18. DOS-ohjelmana se oli pakko ajaa emulaattorin kautta, jos sen ylipäänsä aikoi ajaa. Tässä tapauksessa emulointi onnistui hyvin Windowsin sisäisellä command prompt-ohjelmalla. Windowsin sisäinen command prompt-ohjelma on DOS-emulaattori. Microsoft on joutunut valmistamaan sen Windowsiin, jotta säilyisi asiakkaiden vaatima yhteensopivuus joidenkin vanhempien sovelluksien kanssa. Nykyisillä Windows-käyttöjärjestelmillä ei ole enää yhteyttä MS-DOS -käyttöjärjestelmään, joten emulointi on ainut vaihtoehto DOS-ohjelman ajamiseksi.

Avattuani tiedoston sekä alkuperäisessä ohjelmistossa että emuloinnin alla näytti ilmeiseltä, että jos haluaa tarkastella tietokantaa sellaisena kuin se todella aikoinaan oli ja millaiseksi se tarkoitettiin, on sen tarkasteluun käytettävä emulaattoria. On hyvin olen-

naista huomata että ilman Windowsin mukana toimitettua DOS-emulaatiota ei ohjelman ajaminen olisi lainkaan mahdollista. Silloin tarvittaisiin emulaattoriohjelmisto Windowsin ulkopuolelta. Nyt komentorivin kautta käynnistyvä emulointi mahdollisti sen, että tietokantaa kykeni tarkastelemaan sen alkuperäisessä ulkoasussaan ja alkuperäisellä toiminnallisuudellaan. Halutessani olisin voinut vaikka jatkaa tuon nyt vanhentuneen tietokannan ylläpitoa.

Sen sijaan migroimalla, siirtämällä File Express -ohjelmassa laadittu tiedosto pois alkuperäisestä ympäristöstään moderniin Exceliin, katoavat sarakeotsaketiedot. Alkuperäisessä tietokannassa oli merkitty kuka on mainoksen tilaaja, kuka on tuottaja, mikä on mainostettava tuote jne. Excelissä näkyvissä ovat enää vain tilaajat, tuottajat ja mainostettavat tuotteet. Toisin sanoen, alkuperäisessä oli tallella sekä sarakeotsikot että solujen attribuutit. Migroimisen jälkeen tallella olivat vain enää solujen attribuutit. Kun kyseessä on 90-luvulla tehty kirjanpito, on vielä helppoa löytää ihmisiä, jotka todennäköisesti tietävät mikä tieto tuossa Excel-näkymässä tarkoittaa mitäkin. Mutta kuka osaa kertoa sen sitten kun tämä ja sitäkin seuraava sukupolvi on poissa?

On loogista otaksua, että pidemmän ajan päästä ei kukaan enää ottaisi selvää, kuka oli tilaaja ja kuka tuottaja ja mitä tarkoitusta varten video tehtiin. Emme edes voi tietää käytetäänkö pidemmälle tulevaisuuteen mentäessä samoja nimityksiä samoista asioista. Kieli elää ja muuttuu koko ajan.



KUVA 1. Emuloitu ja migroitu aineisto vierekkäin.

Väistämättä migroitukin tiedostoformaatti tulee aikaa myöden vanhentuneeksi samoin kuin sovellus jolla sitä käytettiin (tässä tapauksessa Excel), ja se joudutaan siten migroimaan uudestaan. Tämän migroimiskierteen jatkuessa sen korruptoiva vaikutus aineistoon on kumuloituva. Jos jo ensimmäisessä migrointisukupolvessa menetetään solujen sisältämien tietojen sarakeotsikot, kuinka paljon häviääkään tietoja kymmenien vuosien migrointien seurauksena, jolloin pitkäaikaissäilytys voidaan katsoa vasta alkaneeksi?

Ilman emuloinnin tuottamaa eksaktia esitystä alkuperäisestä aineistoista on vaarana, että tuottamamme kirjanpidon lukeminen vaatii tulevaisuudessa asialle erikseen omistautuneet ammattilaiset, jotka hekään eivät kykene palauttamaan kaikkea. Sellaista ei voi palauttaa, mitä ei ole. Emulointikaan ei tosin kykene säilyttämään aivan kaikkea, kuten kokemusta siitä, millaista on pidellä käsissään 1990-luvun levykettä. Mutta on tärkeää huomata, että emulointi kykenee säilyttämään *eniten*.

"Digitaalisessa maailmassa on valittava mitä menetämme", todetaan Unescolle tuottamassa raportissa, joka koskee digitaalisen kulttuuriperinnön säilyttämistä (Lusenet 2002, 3). Järkevää toimintaa on se, että valitsemme vaihtoehdon, joka menettää alkuperäisestä aineistosta ja sen välittämästä vaikutelmasta vähiten.

8. MIKÄ PITKÄAIKAISSÄILYTYKSESSÄ MAKSAA?

Digitaalisten aineistojen pitkäaikaissäilytysten kustannusten laskeminen on siinä mielessä tyhjän päällä, että pystymme parhaimmillaankin laskemaan mitä säilytys maksaa nyt, mutta emme voi tietää pitävätkö laskemamme kustannukset ollenkaan paikkaansa tulevaisuudessa. Tarkoissakin laskemissamme saattaa olla jokin virheellinen ennakko-oletamus, joka vie laskumme täysin harhaan. Silti kustannuksia on mietittävä ja laskettava, sillä vaihtoehdot joihin organisaatioilla ei ole varaa, ovat automaattisesti poissuljettuja, olivat ne sitten miten hyvin toimivaksi todettuja tahansa.

Digital Preservation Testbed -työryhmä käy raportissaan lävitse erilaisia muuttujia, joiden voidaan olettaa todennäköisesti vaikuttavan aineistojen pitkäaikaissäilytyksen kustannuksiin. DPT:n (2005, 5) mukaan digitaalisen aineiston pitkäaikaissäilytyksessä kustannukset aiheutuvat pääasiassa: 1) Järjestelmän rakentamisesta eli siihen tarvittavien laitteistojen hankkimisesta ja ylläpidosta. 2) Henkilöstökuluista. 3) Järjestelmän vaatimien ohjelmistojen ja säilytysmenetelmien kehittämisestä. 4) Aineistojen säilytystoimenpiteistä ja asiakkaille toimittamisesta aiheutuvista kustannuksista. 5) Muista tekijöistä, jotka vaikuttavat kokonaiskustannuksiin. (DPT 2005, 5.) Seuraavassa tarkastellaan kohtien 1-5 kustannuksia.

Tarkemmin eriteltynä kohdat 1-5 pitävät yksityiskohtaisemmin sisällään seuraavanlaisia kustannuksiin vaikuttavia tekijöitä: 1) Lähdemme nyt liikkeelle siitä kiistämättömästä tosiasiasta, että digitaalisen arkistonkin on digitaalisuudestaan huolimatta sijaittava jossakin. Sitä varten tarvitaan siis kiinteistö. Helposti ajattelemme, että digitaalinen arkisto voisi digitaalisuutensa ansiosta ikään kuin olla olemassa jonkinlaisessa virtuaalitodellisuudessa, jonka ei tarvitse viedä mitään tilaa. Vaikka itse aineisto ei (varsinkaan tallennusmenetelmien jatkuvasti kehittyessä) kovasti tilaa tarvitsekaan, ja digitaalisen arkiston palvelimetkin voivat lopulta kutistua niin pieniksi etteivät ne juurikaan fyysistä tilaa vie, ihmiset eivät tule kutistumaan. Koska ihmisten tulee pystyä noita laitteita käyttämään ja hallinnoimaan ei niiden tilantarve tule koskaan häviämään olemattomiin. Lisäksi jos pitkäaikaissäilytysorganisaatio todella haluaa saavuttaa luotettavan säilyttäjän maineen, sen on pidettävä huolta siitä että aineistot ovat varmassa turvassa ulkopuolisilta uhkilta. Tällaisia uhkia ovat esim. ilkivalta ja luonnonmullistukset. Tietyillä aineis-

toilla voi olla myös suuri arvo pimeillä markkinoilla tai ne voivat olla vaarallisia vääriin käsiin joutuessaan, jolloin niille varatun säilytyspaikan on oltava murtovarma. Arkiston on säilytettävä aineistonsa useilla eri tallennusvälineillä rinnakkaisesti ja näiden tallennusvälineiden hankkimisesta muodostuu omat kustannuksensa. (DPT 2005, 5-7.)

Laitteistot, olivat ne sitten kuinka pieniä tahansa, tarvitsevat oman palvelinhuoneensa, joka on ilmastoitu. Digitaaliselle arkistolle on olennaista, että sen tietokoneet ovat yhteydessä ulkomaailmaan nopeimmilla saatavilla olevilla tietoverkkoyhteyksillä. Henkilöstö, joka työskentelee organisaatiossa tarvitsee toimistotilat, joihin kuuluu myös jonkinlainen palaveritila. Henkilöstöä varten täytyy olla myös keittiö ja saniteettitilat. Täysivaltaisesti toimiva, asiakkaita palveleva arkisto tarvitsee tietysti ennen pitkää toimivat asiakkaille tarkoitetut palvelutilat samaan tapaan kuin esim. KAVA:lla on katselupisteet ja tutkijahuoneet. Näistä tekijöistä muodostuu pitkäaikaissäilytysarkiston tilakustannukset. (DPT 2005, 5-7.)

2) Henkilöstökuluissa tapahtuu usein helposti aliarviointeja. DPT:n (2005, 7-8) mukaan henkilöstön ensimmäisenä tehtävänä on säilytysjärjestelmän suunnittelu ja rakentaminen. Arkistoalan ammattilaisilla on paljon kokemusta siitä, että näissä arvioinneissa voi helposti tapahtua pahoja virhearviointeja. Suunnittelu- ja rakentamisaikatauluilla on pahana tapana muodostua ylioptimistisiksi, jolloin niiden toteutumisen pitkittyminen nostaa kustannuksia huomattavasti. Järjestelmän rakentamisen arvioidaan DPT:n (2005, 7) raportissa vievän n. 1-2 työvuotta.

Järjestelmän rakentamisen jälkeen henkilöstöstä muodostuu juoksevia henkilöstökuluja. Näitä ovat esimerkiksi järjestelmän rahoituksen järjestämisestä, säilytysjärjestelmän vaatimien muutostöiden suorittamisesta, säilytysjärjestelmän hallinnoimisesta ja ylläpidosta, tietoturva-asioiden järjestämisestä ja laaduntarkkailusta kerääntyvät työtunnit. (DPT 2005, 8.)

3) Järjestelmän vaatimien ohjelmistojen ja säilytysmenetelmien kehittämisen aiheuttamat kustannukset ovat riippuvaisia esimerkiksi siitä, mitkä seikat määritellään välttämättömiksi aineiston autenttisena pitämisen kannalta. DPT:n mukaan (2005, 9) suuresti kustannuksiin vaikuttava tekijä on se kuinka suurina joukkoina digitaalisia asiakirjoja käsitellään. Digitaalisten asiakirjojen joukko muodostuu samalla sovelluksella laadituis-

ta digitaalisista asiakirjoista. Autenttisuuden kriteereiden määrittelyyn saatetaan joidenkin joukkojen kohdalla tarvita säilytysorganisaation ulkopuolista apua. Mitä enemmän autenttisuutta vahvistavia tekijöitä vaaditaan, sitä enemmän niiden määrittelystä muodostuu työtunteja ja sitä suuremmat ovat kustannukset. (DPT 2005, 9.)

4) Varsinaiset aineistoista aiheutuvat kulut ovat digitaalisen aineiston pitkäaikaissäilytysarkistossa vain pieni murto-osa säilytyksen kokonaiskustannuksista. Nämä kustannukset muodostuvat aineistojen migroimisesta säilytysjärjestelmää varten, aineistojen migroimisesta asiakkaita varten, ja emuloinnin käytöstä tiettyjen aineistojen kohdalla. Näihin kustannuksiin vaikuttaa paljon se kuinka suurina joukkoina aineistoja käsitellään. Oletettavasti eniten kustannuksia tulee aineistoista, joista puuttuu jotakin olennaista, kuten metadataa, ja silloin henkilökunta joutuu käsin lisäämään nuo puuttuvat tiedot. Kustannuksia aiheutuu myös paljon siitä, jos jokin aineistojoukko vaatii uuden emulaattorin kehittämistä ja testausta. (DPT 2005, 11.)

5) Muut kustannuksiin vaikuttavat tekijät ovat sellaisia, joihin pitkäaikaissäilytysorganisaatio ei pysty itse vaikuttamaan. Sellainen on esimerkiksi säilytyspalveluun kohdistuva rasitus. Mitä enemmän arkistolla on käyttäjiä, sitä suuremmat ovat sen käytöstä koituvat kustannukset. Toisaalta suurempi käyttäjämäärä tekee mahdolliseksi myös tarjota käyttäjille enemmän maksullisia lisäpalveluja. (DPT 2005, 12.)

Säilytystoimenpiteiden välissä oleva aika on merkittävä lisätekijä kustannuksiin. Mitä useammin aineistot migroidaan, sitä enemmän kustannuksia. Tämä johtuu siitä että mitä useammin aineistoihin kohdistetaan säilytystoimenpiteitä, sitä suurempi riski on vaikuttaa aineistojen autenttisuuteen ja eheyteen. Siten nousee myös tarve tehdä useampia autenttisuuden ja eheyden tarkistuksia, joka nostaa kustannuksia. Aineiston hajanaisuus vaikuttaa kustannuksiin huomattavasti enemmän kuin aineiston määrä, sillä jos aineisto on luonteeltaan sirpaleista, siihen joudutaan kohdistamaan useita eri säilytysstrategioita, josta aiheutuu huomattavasti enemmän työtä kuin suurien homogeenisten aineistomassojen prosessoimisesta. (DPT 2005, 12.)

Jokseenkin yksiselitteiseltä vaikuttaa se, että henkilöstökulut ovat suurimmat kustannuksien aiheuttajat, harjoitettiinpa pitkäaikaissäilytystä sitten missä hyvänsä ja millä menetelmällä hyvänsä. Tästä huolimatta, varsinkin suurempien organisaatioiden saattaa

olla kannattavaa panostaa ratkaisuun, joka laskuissa on hankintahinnoiltaan kalliimpi, jos sen ylläpitokustannukset ovat pidemmän ajan kuluessa pienemmät. Pienemmälle instituutiolle vaatimattomampi ratkaisu saattaa olla kannattavampi. Yleispätevänä sääntönä voidaan sanoa, että joka kerta kun organisaatiossa joudutaan pysähtymään tutkimaan jotain aineistoa erikseen, tulee tuosta aineistosta organisaatiolle hintavampi kuin sellaisesta, joka prosessoidaan rutiininomaisesti.

9. TUTKIMUKSEN JOHTOPÄÄTÖKSET JA SUOSITUKSET

Mitkä ovat emuloinnin vahvuuksia pitkäaikaissäilytyksessä? Läpikäymäni aineiston ja tapaustutkimusten perusteella voidaan sanoa että emuloinnin suurin vahvuus on sen kyvyssä säilyttää digitaalinen aineisto kaikkine ominaisuuksineen niin lähelle alkuperäistä kuin suinkin on mahdollista. Tässä suhteessa kaikki muut säilytysmenetelmät kalpenevat emuloinnin rinnalla.

Voimme verrata emulointia suhteessa muihin säilytysmenetelmiin siten, että emuloinnissa me pyrimme pitämään museoidun talon pystyssä loputtomiin ja ponnistelemme löytääksemme keinoja onnistuaksemme tässä säilytyksessä. Rakennamme jotakin uudestaan vain, jos se on ehdottoman välttämätöntä, ja suoritamme kaikki restauroinnit yms. säilytystoimenpiteet mitä suurimmalla varovaisuudella, ettemme muuta talon alkuperäistä ulkonäköä ja tunnelmaa.

Muissa säilytysmenetelmissä asenteemme on enemmän sen suuntainen, että säilytämme tietoja siitä millainen museoitu talo oli. Pidämme alkuperäistä ulkonäköä ja tunnelmaa lähinnä sivuseikkana ja olemme varmoja siitä, että säilyttämämme tiedot alkuperäisen talon rakenteesta mahdollistavat sen, että pystymme halutessamme rakentamaan siitä sellaisen kopion, joka on riittävän lähellä alkuperäistä. Lisäksi katsomme, etteivät alkuperäinen ulkonäkö ja tunnelma välitä säilytettävästä kohteesta mitään merkittävää.

Kaikissa allegorioissa on ongelmansa ja kömpelyytensä, varsinkin digitaaliseen aineistoon sovellettuna. Silti mielestäni jälkimmäinen kuvaus asenteesta, jossa talon alkuperäistä ulkonäköä ja tunnelmaa ei pidetä säilyttämisen arvoisena, vastaa aika pitkälle suomalaisten muistiorganisaatioiden asennetta digitaaliseen aineistoon. Me kaikki tiedämme, että näköispainos vanhasta kirjasta ei korvaa alkuperäistä. Se saattaa välittää meille kaiken sen tiedon jonka tarvitsemme, mutta suhtaudumme alkuperäiseen silti ihan eri tavalla. Ja jos haluamme tutkia tarkemmin esimerkiksi kauan aikaa sitten eläneiden oppineiden työskentelytapoja, ei näköispainos enää riitä meille vaan meidän on päästävä kääntelemään alkuperäistä isokokoista kirjaa saadaksemme elävän kokemuksen siitä, millaisia etuja ja rajoitteita heidän työskentelytavoissaan oli. Tämä saattaa muuttaa näkemyksiämme muinaisten oppineiden työskentelytavoista suuresti, jopa niin paljon, että

vasta tuon kokemuksen kautta voimme sanoa todella tietävämme millaista heidän työskentelynsä oli.

Millainen on emuloinnin asema pitkäaikaissäilytysprojekteissa? Kun vastaan kysymykseeni emuloinnin roolista pitkäaikaissäilytysprojekteissa on ensin todettava, että suomalaisissa pitkäaikaissäilytysprojekteissa emulointia on karsastettu sen näennäisen monimutkaisuuden ja kalliiden aloituskustannusten takia. Ulkomaisissa pitkäaikaissäilytysprojekteissa on sen sijaan otettu emulointi mukaan vakavasti otettavana vaihtoehtona. Otaksun, että tähän vaikuttaa paljon suomalainen arkistoperinne, joka vaikuttaa hyvin vanhoilliselta ja mielikuvituksellisia ratkaisuja kaihtavana. Ulkomailla osataan ehkä olla tässä suhteessa ennakkoluulottomampia. Emulointi vaikuttaa menetelmänä monen arkistonhoitajan silmissä ja korvissa monimutkaiselta rakettitieteeltä, mutta se ei ole sitä yhtään enempää tai vähempää kuin muukaan tietotekniikka.

Nähdäkseni todellinen syy emuloinnin käyttämättömyyteen pitkäaikaissäilytyksessä on se, että emulointi on arkistointialalla vielä suhteellisen tuntematonta aluetta ja arkistokäyttöön tarkoitetut emulaattorit vielä lapsenkengissään. Tämä saa monet tyytymään vähempään, mutta se ei tee oikeutta digitaaliselle kulttuuriperinnöllemme. Jos arkistokäyttöön tarkoitettujen emulointiratkaisujen kehittämiseen ohjataan tarpeeksi voimavaroja, tässä tutkimuksessa läpikäydyn aineiston perusteella ei ole syytä epäillä, etteikö lopputuloksena olisi toimiva ja luotettava säilytysjärjestelmä. Tämä johtopäätös voidaan vetää myös esitellyistä tapaustutkimuksista, vastaten siten kolmanteen tutkimuskysymykseeni ja samalla viimeiseen tutkimuskysymykseeni arkistoalan emulointiratkaisujen kehittyneisyyden pysähtyneisyydestä.

Neljäs tutkimuskysymykseni, onko emuloinnilla olennaista roolia tietokoneteollisuudessa, sai vastauksensa emuloinnin historian tutkimisesta. Emulointi on tietokoneajan alusta saakka hyvin tunnettu tekniikka, jonka etenemistä kohti kehittyneempiä ratkaisuja voimme seurata patenttitietokantaa tarkastelemalla. Emuloinnin historiassa on kehittynyt rinnakkain kaksi haaraa, jotka olen nimennyt teolliseksi ja harrastelijamaiseksi. Harrastelijamaisen emuloinnin piirissä aloitetut projektit ovat usein laajentuneet lopulta teollisen puolelle saavutettuaan tietyn tason. Mitä teollisen emuloinnin puolella ei ole emuloitu, sitä on emuloitu harrastelijamaisen emuloinnin puolella. Näin väistämätön

johtopäätökseni on, että Rothenberg ei ole liioitellut emuloinnin roolia tietokonemaailmassa.

Vastaus tutkimuskysymykseeni emulointiin liittyvistä käsite-epäselvyyksistä oli erityisen hankala selvittää. Emulointi ja virtualisointi ovat käsitteinä hankalia ja aiheuttavat paljon sekaannusta. Näennäisestä sekamelskasta huolimatta ne kyetään teknisesti erottamaan toisistaan selvärajaisesti. Näiden kahden käsitteen sekoittuminen toisiinsa tulee kuitenkin ottaa huomioon alan kirjallisuutta ja tutkimuksia lukiessa. On pidettävä varansa, sillä asiantuntijatkin käyttävät välillä huomaamattaan näitä käsitteitä toistensa synonyymeina tai muutoin väärissä yhteyksissä.

Tutkimuskysymys emuloinnin suhteesta lainsäädäntöön sai selkeän vastauksen, joka antaa aihetta jatkotutkimuksille. Emuloinnin suhde lainsäädäntöön on huolestuttava. Tulin siihen tulokseen että Rothenberg ei valitettavasti suhtaudu tähän kysymykseen riittävällä vakavuudella, vaan ohittaa sen julkaisuissaan melko kepeästi. Katson, että lakikysymykset on ratkaistava ennen kaikkia muita. Ilman lain voimaa ei emulointia voi käyttää yhteiskunnan ylläpitämissä pitkäaikaissäilytysorganisaatioissa. Ilman lain voimaa arkistointikäyttöön kehitetyt emulointiratkaisut raukeavat ainakin arkistojen osalta tyhjiin, vaikka löytäisivätkin käyttöä joltakin muulta alalta. On siis viipymättä aloitettava emulointiin liittyvien lakikysymysten selvittäminen. Tämän selvitystyön tulee tapahtua pitkäaikaissäilytysorganisaatioiden taholta, jolloin niillä on takanaan riittävä auktoriteetti.

Yksi pahimpia uhkakuvia digitaalisen aineiston säilyttämisen kannalta on mielestäni sellainen mahdollisuus, että muistiorganisaatio ostaa säilytyspalvelut joltain yksityiseltä yritykseltä ja jonkun ajan kuluttua tuo yritys menee nurin. Näin aineistot jäävät ilman huolenpitoa oman onnensa nojaan muistiorganisaation ollessa kykenemätön valjastamaan riittävästi oman talon väkeä aineistosta huolehtimiseen.

Eri säilytysmenetelmien yksityiskohtainen testaaminen on aloitettava nyt, sillä jokaisessa menetelmässä on omat vahvuutensa ja heikkoutensa, ja nämä ominaisuudet tulevat esille vasta eri menetelmien pitkäaikaisessa yhtä-aikaisessa käytössä. Tällaiseen testausjaksoon ei tällä hetkellä olla suuntaamassa voimavaroja suomalaisissa muistiorganisaatioissa. Katson, että Kansallisen Digitaalisen Kirjaston tulisi koordinoida eri menetelmien kokeiluhankkeita käytännössä suomalaisissa muistiorganisaatioissa. Näin testaus-

projektin työmäärä tulisi järkevästi jaettua suomalaisten pitkäaikaissäilytysorganisaatioiden kesken.

Open Archival Information System -mallin maailmanlaajuinen käyttöönotto antaa hyvän pohjan eri säilytysstrategioiden tasavertaiseen vertailuun ja testaamiseen. Yhteiseen konsensukseen on pitkäaikaissäilytyksen suhteen päädytty sen suhteen, että kaupallisia tiedostomuotoja on vältettävä kaikissa säilytysvaihtoehdoissa niiden markkinatilannesidonnaisuuden takia. Siksi pitkäaikaissäilytysorganisaatioissa on päädytty avoimen lähdekoodin tiedostomuotoihin. Tässä katson kehityskulun olleen pitkäaikaissäilytyksen kannalta toivottava.

Koska tietokoneiden ja ohjelmistojen valmistajat sen paremmin kuin pitkäaikaissäilytysorganisaatiotkaan eivät ole kyenneet tarjoamaan ratkaisua säilytyksen ongelmaan, ovat monet 70-80 -lukujen kasvatit ottaneet sydämen asiakseen säilyttää näihin laitteisiin liittyvää tietotaitoa ja ohjelmistoja. Tämä globaali liikehdintä on synnyttänyt useita säilytysprojekteja, jotka ovat yleensä keskittyneet johonkin tiettyyn laitteistoon. Tällaisia projekteja ovat esimerkiksi Microbee Software Preservation Project ja Amiga Music Preservation. Nimensä mukaisesti jälkimmäinen on keskittynyt yksinomaan musiikkiin. Jo pelkästään tuon musiikin säilyttämisessä on omat haasteensa, sillä musiikin toistaminen oikein vaatii sen laitteiston emuloimista, jossa toistettavaksi tuo musiikki on tarkoitettu.

Amiga-musiikin toistamista varten on kehitetty oma Unix Amiga Delitracker Emulator -niminen sovellus. Tämä emulaattori keskittyy varsinkin Paula-äänipiirin emuloimiseen. Tietysti itse lopputulos, laitteiston synnyttämä musiikki voidaan tallentaa laadukkaasti jollekin tallennusvälineelle ja säilyttää vain se, mutta Amigayhteisö on yksimielinen siitä, että samalla menetään jotain hyvin oleellista, jos vain lopputulos säilyy. Dokumentaatiota siitä miten tuohon lopputulokseen on päästy menetetään.

Jos esimerkiksi Suomessa olisi yksi keskitetty organisaatio digitaalisen kulttuuriperinnön säilyttämiselle, voisi olla mahdollista että he ostaisivat nämä yksityiset arkistointiprojektit ja keräisivät ne yhden katon alle. Näin myös harrastelijat voisivat saada korvauksen tekemästään työstä.

Kuluttajina meidän tulisi vaatia tietokoneteollisuudelta, ettemme hyväksy yhteensopimattomuutta uusien laitteistojen ja vanhempien digitaalisten aineistojen välillä. On lähetettävä vahva viesti ja pidettävä huolta siitä että se myös kuullaan. Tietokoneita tuottavat yhtiöt ovat osoittaneet että heillä on resursseja yhteensopivuuden kehittämiseen, jos sitä heiltä vain vaaditaan.

Emulointiratkaisua vastaan käytetään usein argumenttina erilaisten laitteistojen runsautta ja noiden lukuisten laitteistojen emuloimisen aiheuttamaa suunnatonta työmäärää. Kuitenkin, innokas emulointiyhteisö on osoittanut, että jos vain tahtoa riittää ja resursseja kohdistetaan, pystytään mitä tahansa menneisyyden laitteistoa emuloimaan. Tästä voidaan hyvänä esimerkkinä mainita Atari 2600, jota on erityisen vaikea emuloida siinä käytettyjen poikkeuksellisten ratkaisujen vuoksi.

Tietokoneteollisuudessa on 90-luvun puolivälistä lähtien ollut vallalla valitettava trendi, jossa ohjelmakoodin optimoimiseen ei panosteta samalla tavoin kuin 80-luvulla, jolloin laitteisto pysyi monilla tietokoneohjelmoijilla samana, mutta tuosta laitteistosta saatiin enemmän suorituskykyä irti ohjelmallisesti tehtyjen ratkaisujen avulla. Kun uutta laitteistoa ei ollut saatavilla tai siihen ei ollut varaa, oli pakko yrittää saada tehtyä enemmän samalla laitteistolla optimoimalla ohjelmakoodia. Tämän kehityspaineen poistuessa laitteistojen jatkuvasti halventuessa ja kehitykseen kiihtyessä, ei tällaiseen ohjelmointitaitojen huippuunsa hiomiseen ollut enää tarvetta, josta seurauksena oli nykyisin vallalla oleva laiska ohjelmointikulttuuri. Nykyisessä ohjelmointikulttuurissa pyrkimyksenä on tehdä vain toimivaa, kun aikaisemmassa kulttuurissa oli toimivuuden lisäksi pyrkimys tehdä ohjelma *mahdollisimman järkevästi*.

Kaikkien komponenttien moduulikirjasto saattaa olla yhtä iso hanke kuin ihmisen perimän kartoitus (human genome project), ja vaatii useiden yliopistojen maailmanlaajuista yhteistyötä. Projektin jossa työ on jaettu järkeviin kokonaisuuksiin useiden yliopistojen kesken. Tampereen Yliopiston Informaatiotieteiden yksikkö voisi ottaa tämän kartoitustyön suhteen pioneerin roolin aloittamalla sen, ja ehdottamalla työkuorman jakamista muiden yliopistojen informaatiotieteitä opettavien yksikköjen kanssa.

Kysymys digitaalisten asiakirjojen todistusvoimaisuuden säilyttämisestä on niin merkittävä, että tällaista kartoitusprojektia olisi viipymättä ryhdyttävä suunnittelemaan. Digi-

taalisen asiakirjan todistusvoimaisuus voi tulevaisuuden (ja nykyisissäkin) oikeudenkäynneissä ratkaista kuka on syyllinen ja kuka syytön. Esim. yhdysvalloissa se voi vaikuttaa jopa kuolemantuomion langettamiseen.

Tulevaisuuden arkistonhoitajat ovat eräänlaisia digitaalisen aineiston ”pappeja”. He pitävät huolta, että digitaalinen aineisto on kussakin maailmanajassa ymmärrettävässä muodossa tuolle kulloinkin vallitsevalle kulttuurille. He myös siirtävät tietotaitonsa seuraavalle sukupolvelle. Digitaalinen aineisto saattaa synnyttää oman ammattikunnan, joka on omistautunut vain sille, kuten entisajan maanviljelijät ja sepät olivat omistautuneita omille ammateilleen. Jossain vaiheessa tulevaisuutta saattaa tuo ammattikunta tulla sitten tarpeettomaksi.

Tietokonemuseoiden laitteistoja olisi järkevää käyttää emulaattoreiden testaamiseen. Pitkäaikaissäilytysorganisaatioiden ja tietokonemuseoiden tulisi tehdä testiajojen järjestämisten suhteen tiivistä yhteistyötä. Vain alkuperäisten laitteistojen tuottamaan esitykseen vertaamalla voidaan arvioida emuloinnin luotettavuus.

Lisäksi katson, että Tampereen Yliopistossa voitaisiin jokseenkin pienelläkin panostuksella helposti tehdä emulaattoreiden käytettävyys- ja luotettavuustutkimusta ilman että se vaatisi tuekseen suurimittaisia rahoituksia. Tieteen tulee rohkeasti etsiä tietoa uusista paikoista ja mennä sinne minne ei aikaisemmin ole menty!

SANASTO

Aineistokapseli = Aineistopaketeista koostuva säilytysjärjestelmään pitkäaikaissäilytystä varten talletettu kapseli.

Aineistopaketti = talletetun aineiston bittivirta tai sen metatiedot tai tuon bittivirran suorittamiseen tarvittavat ohjelmistot pitkäaikaissäilytykseen kelpaavina pakettina.

Aineiston palauttaminen, palauttaminen = Digitaalisen aineiston saattaminen ihmisen tarkasteltavissa olevaan muotoon.

AIP = Archival Information Package. Säilytyspaketti. OAIS-mallissa se tietoaines, joka on paketoitu pitkäaikaissäilytystä varten

ASCII = American Standard Code for Information Interchange. Lähinnä tekstimuotoiselle tiedolle tarkoitettu merkkien esitystapastandardi. ASCII järjestelmää käytetään kaikissa tietokoneissa.

Bochs = Avoimeen lähdekoodiin perustuva PC-emulaattori.

CAMILEON = Creative Archiving at Michigan & Leeds: Emulating the Old on the New. Michiganin ja Leedsin yliopistojen yhteinen emuloinnin tutkimiselle omistettu projekti.

CCSDS = Consultative Committee for Space Data Systems. Kansainvälinen avaruustutkimusjärjestöjen yhteinen foorumi, jonka tehtävänä on kehittää avaruustutkimuksen tiedonhallintaa tukevia standardeja.

CEDARS = Curl Exemplars in Digital ArchiveS. Joint Information Systems Committeen rahoittama digitaalisen aineiston pitkäaikaissäilytysprojekti.

CP/M = Control Program/Monitor. Digital Researchin Z80- ja Intel 8080- & 8085 -suorittimille kehittämä käyttöjärjestelmä.

Data-arkeologia = Toimintaa, jossa tietoja pyritään palauttamaan tallenteilta, joiden lukemiseen ei ole enää olemassa välineitä. Palauttamista voidaan tehdä esim. magneettimikroskoopin ja OIR:n avulla.

Demo-ohjelma = Tämän työn kontekstissa grafiikkaa ja ääntä sisältävä tietokoneohjelma, jonka tarkoitus on esitellä tekijöidensä ohjelmointitaitoja.

Demoscene = Alakulttuuri, jossa eri ryhmittyvät tekevät grafiikkaa ja ääntä sisältäviä esityksiä tietokoneella. Demoscenessä kilpaillaan siitä, kuka kykenee tekemään näyttävimmän esityksen jollakin tietyllä laitteistolla.

Nimetty yhteisö = Designated Community. OAIS-mallin määrittelemä pitkäaikaissäilytysjärjestelmän pääasiallinen kohderyhmä.

DIAS = Digital Information Archiving System. Koninklijke Bibliotheekin ja IBM:än yhdessä rakentama elektronisten aineistojen talletusjärjestelmä.

Dioscuri = Koninklijke Bibliotheekin kehittämä komponenttiemulaattori.

DIP = Dissemination Information Package. Jakelupaketti, joka koostuu yhdestä tai useammasta säilytyspaketista ja joka luovutetaan asiakkaalle tämän tehtyä aineistopyynnön säilytysjärjestelmään.

DOS = Disk Operating System. Yleisnimitys tietokoneissa levymuistien käyttöä tukeville käyttöjärjestelmille.

DSEP = Deposit System for Elektronic Publications. NEDLIB-projektin yhteydessä syntynyt digitaalisen aineiston pitkäaikaissäilytysjärjestelmä.

E-UAE = Amiga-emulaattori UNIX-käyttöjärjestelmille. Myöhemmin tunnettu nimellä UAE ja tukee nykyisin useita käyttöjärjestelmiä.

Emulaatiospesifikaatiotulkki = Tulkkiohjelma, joka kykenee luomaan emulaattorin alustoille, joista on kirjoitettu riittävä spesifikaatio.

Emulaatiovirtuaalikone = Virtuaalikone, jossa on emulaatiospesifikaatiotulkki ja jonne emulaatiospesifikaatiot ajetaan ja joka näistä kokoaa emuloinnin.

Emulaattori = Ohjelmisto, jota ajetaan tietokoneessa (isäntäkone), ja joka siten virtuaalisesti luo toisen tietokoneen (kohdekoneen).

EVM = Emulaatiovirtuaalikone (Emulation Virtual Machine). Ohjelmisto, joka kykenee kokoamaan emulaattorin erillisiä komponentteja emuloivista ohjelmistoista.

Hatari = Atari-emulaattori GNU/LINUX, BSD, Mac OS X ja Windows-käyttöjärjestelmille.

Konvertointi = katso migraatio

LDV = Logical Data View mutta myös Logical Data Viewer. Logical Data View on uuteen formaattiin dekoodattu säilytetty aineisto. Logical Data Viewer on sovellus joka lukee Logical Data Viewin.

Luotettava säilytysmenetelmä = Toimintojen joukko, joka varmistaa digitaalisen aineiston tarkastelun sen alkuperäisessä muodossa ilman informaatiohäviötä siten, että aineiston autenttisuus ja todistusvoima säilyvät, ilman että tuon menetelmän toteuttamisesta aiheutuu pitkäaikaissäilytysorganisaatiolle sen resurssit ylittäviä kustannuksia.

Läpinäkyvä osoitus = AIP:hen liitetty bittijono, joka toimii AIP:n tunnisteena.

Makronauhoittaja = Macro recorder. Ohjelmisto, joka ottaa talteen käyttäjän suorittamia toimintoja myöhempää suorittamista varten. Emuloinnissa tästä tekniikasta saattaisi olla apua tulevaisuuden käyttäjän aineiston käytön helpottamisessa.

Migraatio = Joukko organisoituja toimenpiteitä, joilla pidetään huolta digitaalisten aineistojen siirtämisestä vanhoista tallenteista/laitteistoista uusiin.

Komponenttiemulaatio = Emuloinnin muoto, jossa jokaista tietokonelaitteiston fyysistä komponenttia emuloi oma erillinen ohjelmallinen emulaattorinsa. Emulaattorit yhdistetään toisiinsa loogisella tasolla samoin kuin komponentit olivat kytkettyinä toisiinsa alkuperäisessä fyysisessä laitteistossa. Lopputulos vastaa alkuperäisen fyysisen laitteiston aikaansaamaa vaikutusta.

NEDLIB = Networked European Deposit Library. Koninklijke Bibliotheekin johtama projekti, johon ottavat osaa useat eurooppalaiset kansalliskirjastot, myös Suomen. Pro-

jektin tarkoituksena on kehittää yhteinen arkkitehtuuri kaikille elektronisten aineistojen talletusorganisaatioille.

OAI = Open Archives Initiative. Organisaatio, jonka tehtävänä on kehittää ja edistää tiedonjakamiseen liittyviä yhteisiä standardeja.

OAIS = (Open Archival Information System). Consultative Committee for Space Data Systemsin kehittämä suositusmalli, josta myöhemmin on tullut keskeinen rakennuspalikka digitaalisten aineistojen säilytysjärjestelmiin.

PANDORA = Preserving and Accessing Networked Documentary Resources of Australia. Australian kansalliskirjaston ylläpitämä verkossa oleva websivuarkisto.

Pitkäaikaissäilytys = Vähintään 100 vuotta siten, että aineisto on tarkasteltavissa kaikkien alkuperäisten ominaisuuksiensa kera.

PearPC = Arkkitehtuuririippumaton PowerPC-emulaattori, joka kykenee ajamaan suurinta osaa kaikista PowerPC-käyttöjärjestelmistä

Pinottu emulointi = Tällä hetkellä käytössä olevan laitteisto ollessa vanhentumisuhan alla, sille kirjoitetaan emulaattori, joka toimii tuota laitteistosukupolvea edeltäneessä emulaattorissa. Näin emulaattoreita voidaan kerrostaa loputtomiin ja laskentatehon eksponentiaalinen kasvu pitää huolen suorituskyvyn riittävyydestä.

PLM = Preservation Layer Model. PLM:n tehtävänä on kuvata graafisesti millainen suhde samankaltaisilla objekteilla, esim. tietyssä formaattimuodossa olevilla tiedostoilla, on ympäristöönsä.

PLM Administrator = Henkilö, joka tarvittaessa määrittelee uusia näkymäpolkuja ja PLM:iä. Tämä koskee tilannetta, jossa DIAS:iin tuodaan uudentyyppisiä objekteja tai jos tietokonearkkitehtuureissa tapahtuu muutoksia.

Publication Ingester = Henkilö, jonka vastuulla on digitaalisten objektien valmistelu vastaanottoa varten DIAS-järjestelmään.

QEMU = Avoimeen lähdekoodiin perustuva emulaattori ja virtualisoija.

RAM = Read Access Memory. Tietokonemuisti, johon voidaan sekä lukea, että kirjoittaa ja josta tiedot häviävät virran katkaisun yhteydessä.

Rosettan kivi = Rosettan kaupungista(nyk. Rashid) löytynyt kivi, joka toimi avaimena hieroglyfien ymmärtämiseen.

SIP = Submission Information Package. Tietopaketti, joka vastaanotetaan tiedon luovuttajalta säilytyspaketin rakentamiseen.

Skeema = tietomalli, kaava, jonka mukaan tiedot järjestetään. Esim. XML on skeema.

Säilytysvirkailija = Henkilö, joka tarkkailee teknologisia muutoksia ja noiden muutosten vaikutuksia DIAS:iin arkistoitujen aineistojen saatavuuteen. Jos aineistot ovat vaarassa muuttua palauttamattomiksi, silloin hänen täytyy päättää mikä säilytysstrategia, migraatio vai emulointi, otetaan käyttöön.

UAE = Unusable Amiga Emulator, myöhemmin Unix Amiga Emulator ja nykyisin vain UAE.

URI = Uniform Resource Identifier. URL:n ja URN:n yläluokka. Menettely, jolla voidaan yksilöidä IP-verkon resursseja. URI:lla voidaan yksilöidä myös resursseja jotka ovat verkon ulkopuolella.

URL = Uniform Resource Locator. Standardi, jolla määritellään resurssien sijainti verkossa. Se määrittää myös Internet-protokollan, esim. HTTP, FTP.

URN = Uniform Resource Name. Verkkojulkaisun ainutkertainen ja pysyvä tunniste. URN:ää voidaan käyttää samoin kuin kirjojen ISBN-numerointia.

UVC = (Universal Virtual Computer). Raymond A. Lorien (IBM) hahmottelema alustariippumaton virtuaalitietokonemalli.

VGA = Video Graphics Adapter

VICE = Commodore 64 emulaattori kaikille alustoille.

Näkymäpolku = Digitaalisen objektin ohjelmisto- ja laitteistoriippuvuuksia kuvaava piirros.

Visuaalinen synkronointikohta = Ruudunkaappauskuva hiiren osoittimen ympäriltä tietystä kohtaa toimintosarjaa.

WINE = Windows Emulator myös Wine Is Not an Emulator. UNIX-pohjaisissa käyttöjärjestelmissä toimiva Windows-emulaattori.

LÄHTEET

Ammann, L & Jackson, H & Johnson, C. 1990. Fast Emulator Using Slow Processor. United States Patent 4,975,869. Saatavilla osoitteessa: http://www.freepatentsonline.com/4975869.html. (käytetty 07.05.2011).

Asproth, Viveca. 2005. Information Technology Challenges for Long-Term Preservation of Electronic Information. International Journal of Public Information Systems vol.2005, issue 1. Östersund: Dept. of Information Technology and Media.

Atari. 1984. Atari unveils advanced video game that is expandable to introductory computer. Lehdistötiedote. New York.

Ballard, D. Emulating Computer. United States Patent 4,638,423. Saatavilla osoitteessa: http://www.freepatentsonline.com/4638423.html. (käytetty 07.05.2011).

Bearman, David. 1999. Reality and Chimeras in the Preservation of Electronic Records. D-Lib Magazine Volume 5, Number 4. Saatavilla: http://www.dlib.org/dlib/april99/bearman/04bearman.html. (käytetty 23.10.2010).

Block, W. & Bolsø, E. & Hall, S & Rateliff, A. 2006. The Big Book of Amiga Hardware. Utilities Unlimited: Emplant. Saatavilla osoitteessa: http://www.amiga-hardware.com/showhardware.cgi?HARDID=345. (käytetty 08.05.2011).

Bolsø, E. & Chris (CLS2086) & Hrothgar & Pedersen, J. 2006. The Big Book of Amiga Hardware. Commodore A1060. Saatavilla osoitteessa: http://www.amiga-hardware.com/showhardware.cgi?HARDID=327. (käytetty 08.05.2011).

Borghoff, U. & Rödig, P. & Scheffczyk, J. & Lothar, S. 2006. Long-Term Preservation of Digital Documents. Berlin: Springer.

Cornell, Paul. 2011. Using Macros to Speed Up Your Work. Microsoft. Saatavilla osoitteessa http://office.microsoft.com/en-us/support/using-macros-to-speed-up-your-work-HA001019230.aspx. (käytetty 09.05.2011).

Datasalen. S.a. Commodore 128D Saatavilla:http://www.datasalen.se/Utstallning/Data/CBM/commodore128deng.htm (käytetty 02.05.2011).

Day, Michael. S.a. A quick history of the CPeMulator. Saatavilla: http://www.programmersheaven.com/download/15547/0/ZipView.aspx (käytetty 02.05.2011).

Diessen, R. & Werf-Davelaar, T. (2002a). Authenticity in a Digital Environment. The Hague: IBM Netherlands.

Diessen, R. & Rijnsoever, B. (2002b). Managing Media Migration in a Deposit System. The Hague: IBM Netherlands.

Diessen, R. (2002c). Preservation Requirements in a Deposit System. The Hague: IBM Netherlands.

Diessen, R. & Steenbergen, J. (2002d). the Long-Term Preservation Study of the DNEP Project. Proof of Concept. The Hague: IBM Netherlands.

Digital Preservation Testbed Project. 2003. Emulation: Context and Current Status. Den Haag: Digital Preservation Testbed Project.

Flinkman, M. & Salanterä, S. 2007. Integroitu katsaus - eri metodeilla tehdyn tutkimuksen yhdistäminen katsauksessa. Turku: Turun Yliopisto.

Giles, A. 1998. Graphics Processor Emulation System And Method With Adaptive Frame Skipping To Maintain Synchronization Between Emulation Time And Real Time. United States Patent 6,115,054. Saatavilla: http://www.freepatentsonline.com/6115054.html. (käytetty 07.05.2011).

Giles, A. S.a. Aaron's Computing History. Saatavilla: http://www.aarongiles.com/history/. (käytetty 07.05.2011).

Gladney, H. M. 2007. Preserving Digital Information. Berlin: Springer.com.

Goodwin, S. 1996. Commodore 64 Emulators. Amiga Format. Issue 91, 24.

Granger, Stewart. 2000. Emulation as a Digital Preservation Strategy. D-Lib Magazine Volume 6, Number 10. Saatavilla: http://www.dlib.org/dlib/october00/granger/10granger.html. (käytetty 23.10.2010).

Gruehn, P. a. S.a. Expansion Cards. Amtari. Saatavilla osoitteessa: http://www.amiga.resource.cx.exp/amtari. (käytetty 08.05.2011).

Gruehn, P. b. S.a. Expansion Cards. Medusa. Saatavilla osoitteessa: http://www.amiga.resource.cx.exp/medusa. (käytetty 08.05.2011).

Hakala, Juha. Long-Term Preservation of Electronic Documents. Helsinki: The National Library of Finland.

Hirsjärvi, S., Remes, P., Sajavaara, P. 2007. Tutki ja kirjoita. Helsinki: Tammi.

Hoeven, J., & Wijngaarden, H. 2005. Modular emulation as a long-term preservation strategy for digital objects. Saatavilla osoitteessa: http://www.iwaw.net/05/papers/iwaw05-hoeven.pdf. (käytetty 15.05.2011). The Hague: Koninklijke Bibliotheek & National Library of the Netherlands.

Hoeven, J., Diessen, R. & Meer, K. 2005. Development of a Universal Virtual Computer (UVC) for long-term preservation of digital objects. Journal of Information Science, 31 (3), 196-208.

Hoeven, J., Slats, J., Verdegem, R. & Wijngaarden, H. 2005. Emulation - a viable preservation strategy. Saatavilla osoitteessa: www.kb.nl/hrd/dd/dd_projecten/Emulation_research_KB_NA_2005.pdf. (käytetty 13.05.2011).

Hoeven, J., Rechert, K., Schröder, J. & Suchodoletz, D. 2010. Seven steps for reliable emulation strategies solved problems and open issues. Saatavilla osoitteessa: www.ifs.tuwien.ac.at/dp/ipres2010/papers/vonsuchodoletz-53.pdf. (käytetty 14.05.2011).

Hoeven, J. & Suchodoletz, D. 2009. Emulation: From Digital Artefact to Remotely Rendered Environments. The International Journal of Digital Curation. Issue 3, Volume 4.

Johansson, Kirsi., Axelin, Anna., Stolt, M., Ääri, Riitta-Liisa. 2007. Systemaattinen kirjallisuuskatsaus ja sen tekeminen. Turku: Turun Yliopisto.

Jones, Tim. 2011. Platform emulation with Bochs. IBM Corporation 2011. Saatavilla: http://www.ibm.com/developerworks/linux/library/l-bochs/?ca=drs- (käytetty 02.05.2011).

Kirps, Jos. 2007. In Memory Of The Commodore 128. Saatavilla: http://www.kirps.com/web/main/_blog/all/in-memory-of-the-commodore-c128.shtml (käytetty 02.05.2011).

Kuijsten, H. 1996. Emulation Systems Having Multiple Emulator Clock Cycles Per Emulated Clock Cycle. United States Patent 5,920,712. Saatavilla osoitteessa: http://www.freepatentsonline.com/5920712.html. (käytetty 07.05.2011).

Lavoie, B. 2004. The Open Archival Information System Reference Model: Introductory Guide. Dublin: Online Computer Library Center, Inc.

Leino-Kilpi, H. 2007. Kirjallisuuskatsaus - tärkeää tiedon siirtoa. Turku: Turun Yliopisto.

Lorie, R. 2000. Long Term Archiving of Digital Information. United States Patent 6,691,309. Saatavilla osoitteessa: http://www.freepatentsonline.com/6691309.html. (käytetty 07.05.2011).

Lorie, R. 2002. The UVC: a method for preserving digital documents. Hague: Koninklijke Bibliotheek.

Lusenet, Yola de. 2002. Preservation of digital heritage. Draft discussion paper prepared for UNESCO. European Commission on Preservation and Access. Saatavilla: www.archivi.beniculturali.it/INTRANET/estero/Preservation_Access.pdf. (käytetty 07.05.2011).

NicDouille. 2006. The Big Book of Amiga Hardware. ReadySoft: A-Max. Saatavilla osoitteessa: http://www.amiga-hardware.com/showhardware.cgi?HARDID=340. (käytetty 08.05.2011).

OAI. Open Archives Initiative Frequently Asked Questions. Saatavilla:http://www.openarchives.org/documents/FAQ.html. (käytetty 25.02.2010).

Plotkin, D. 1990. New PC Emulator and Spectre GCR Hit the Streets. Start Contributing 4 (6), 33.

Potter, B. 1995. Emulation Techniques For Computer Systems Having Mixed Processor/Software Configurations. United States Patent 5,742,794. Saatavilla: http://www.freepatentsonline.com/5742794.html. (käytetty 07.05.2011).

Poulsen, L. 2001. IBM 360/370/3090/390. Saatavilla: http://www.beagle-ears.com/lars/engineer/comphist/ibm360.htm (käytetty 06.11.2009).

Pulkkinen, M. Linuxin ja Windowsin sekakäyttö säästää rahaa. MikroPC 4/2008, 26-28.

Pulkkinen, M. Ole realisti ja virtualisoi. MikroPC 1/2007, 44 - 47.

Purho, J. 2007. Työasemapalvelun kehittäminen virtualisoinnin avulla. Espoo: Teknillinen korkeakoulu.

Ross, S. & Gow, A. 1999. Digital Archaeology: Rescuing Neglected and Damaged Data Resources. London: Library Information Technology Centre.

Rothenberg, J. 1999a. Avoiding Technological Quicksand: Finding a Viable Technical Foundation for Digital Preservation. A Report to the Council on Library and Information Resources. Washington DC: Council on Library and Information Resources.

Rothenberg, J. 1999b. Ensuring the Longevity of Digital Information. Santa Monica: Council on Library and Information Resources. Saatavilla: http://clir.org/pubs/archives/ensuring.pdf. (käytetty 23.10.2010)

Rothenberg, Jeff. 2000a. An Experiment in Using Emulation to Preserve Digital Publications. Den Haag: The Koninklijke Bibliotheek.

Rothenberg, Jeff. 2000b. Using Emulation to Preserve Digital Documents. The Hague: Koninklijke Bibliotheek.

Stevens, K. L. 2008. The Emulation User's Guide. Lulu.com.

The British Museum. The Rosetta Stone. Saatavilla:http://www.britishmuseum.org/explore/highlights/highlight_objects/aes/t/the_rosetta_stone.aspx (käytetty 05.05.2011).

The JPC Project. What are the differences between Virtualisation and Emulation? Saatavilla: http://jpc.sourceforge.net/home_emulation.html (käytetty 03.05.2011)

Werf-Davelaar, T. 1999. Long-term preservation of electronic publications: the NEDLIB project. D-Lib Magazine. September 1999. Volume 5, Number 9. National Library of the Netherlands.

Verdegem, R. 2007. Back to the future - Dioscuri, the modular emulator for digital preservation.

Verdegem, R. 2008. Dioscuri: emulation in practice. Toulouse: European Comission. Den Haag: Nationaal Archief of the Netherlands.

Wijngaarden, H & Oltmans, E. 2004. Digital Preservation and Permanent Access: The UVC for Images. The Hague: Koninklijke Bibliotheek, National library of the Netherlands.

Winewiki. 2010. Debunking Wine Myths. Saatavilla osoitteessa http://wiki.winehq.org/Debunking_Wine_Myths. (käytetty 12.05.2011).

Woods, K. 2008. Virtualisation for Preservation of Executable Art. Bloomington: Indiana University.