Open your mind. LUT.
Lappeenranta University of Technology

TUOTANTOTALOUDEN KOULUTUSOHJELMA

Kustannusjohtaminen

Avoimen ja suljetun lähdekoodin data-analytiikkaohjelmistot kustannusnäkökulmasta

Open and closed source data analytics programs from a cost point of view

Kandidaatintyö

Markus Leppioja

Ilari Tuomela

TIIVISTELMÄ

Tekijät: Markus Leppioja ja Ilari Tuomela

Työn nimi: Avoimen ja suljetun lähdekoodin data-analytiikkaohjelmistot kustannusnäkökulmasta

Vuosi: 2017 **Paikka: Lappeenranta**

Kandidaatintyö. Lappeenrannan teknillinen yliopisto, tuotantotalous.
35 sivua, 3 taulukkoa
Tarkastaja(t): Tutkijatohtori, TkT Salla Marttonen-Arola

Hakusanat: Avoin lähdekoodi, suljettu lähdekoodi, data-analytiikka, data-analyysi, big data, ohjelmistojen kustannukset, data-analytiikan kustannukset
Keywords: Open source, closed source, data analysis, data analytics, big data, software costs, data analytics costs

Työn tavoitteena on vertailla avoimen ja suljetun lähdekoodin data-analytiikkaohjelmistoja kustannusnäkökulmasta. Työssä tuodaan myös esille, mistä komponenteista data-analytiikkaohjelmistojen kustannukset muodostuvat.
Työ on kirjallisuuskatsaus, ja sen lähteet painottuvat pääosin aiheeseen liittyvään kirjallisuuteen sekä tieteellisiin artikkeleihin. Työssä on käytetty myös kaupallisia lähteitä käytännönläheisyyden saavuttamiseksi.
Työssä huomattiin, että avoimen lähdekoodin data-analytiikkaohjelmistot ovat muuttumassa yhä potentiaalisemmiksi vaihtoehdoiksi perinteisiin suljetun lähdekoodin ohjelmistoihin verrattuna. Säästöt muun muassa lisenssi-, ylläpito- sekä konsultointikustannuksissa tekevät avoimen lähdekoodin ohjelmistosta houkuttelevan vaihtoehdon. Toisaalta yrityksille muutos avoimeen lähdekoodiin ei ole aina kannattava, sillä uusien moduulien ja ulkopuolisen konsultoinnin kustannukset voivat ylittää muutoksesta saadun hyödyn. Lisäksi avoimen lähdekoodin tietoturva aiheuttaa usein kysymyksiä yrityksissä siitä huolimatta, että avoimen lähdekoodin ohjelmistoissa tietoturva-aukot löydetään ja korjataan nopeammin.

SISÄLLYSLUETTELO

1 JOHDANTO

Avoimen lähdekoodin lisenssillä toimivat ohjelmistot ja ratkaisut ovat kasvattaneet suosiotaan nopeasti viime vuosina. Avoimen lähdekoodin ohjelmistoja käyttävät laajasti yksityishenkilöt, mutta niiden käyttö on yleistynyt myös yrityksissä huomattavasti (Tilastokeskus 2011). Suuryrityksetkin ovat havainneet avoimen lähdekoodin hyödyt ja mahdollisuudet ja monet näistä tukevatkin avoimen lähdekoodin ohjelmistojen kehitystä erilaisin keinoin. Monet perinteisetkin ohjelmistotalot ovat kehittäneet avoimen lähdekoodin lisenssiin perustuvia ohjelmistoja, esimerkiksi Microsoftin Visual Studio Code (Microsoft 2017).

Digitalisaation edetessä datan määrä kasvaa koko ajan erittäin nopeasti. Datan tallennusmahdollisuudet ovat kasvaneet valtavasti ja dataa voidaan kerätä yhä useammasta eri lähteestä monin eri tavoin (Elgendy & Elgral 2016, s. 1071-1072). Edellä mainittu teknologian kehitys on mahdollistanut suurten datamäärien yhä helpomman saatavuuden ja hyödynnettävyyden. Data-analytiikan tärkein käyttökohde on sen hyödyntäminen päätöksenteon tukena. Aikaisemmin data-analytiikka on ollut suurimmaksi osaksi historian tarkastelemista. Tämä on johtunut enimmäkseen siitä, että datan määrä ja lähteet ovat olleet rajallisia. Dataa ei ole ollut kovinkaan paljon saatavilla, ja sen keräämiseen tarkoitetut työvälineet eivät ole olleet yhtä kehittyneitä kuin nykyisin. Data-analytiikan painopiste on siirtynyt yhä enemmän tulevaisuuden ennustamiseen. Dataa voidaan käyttää uusien suuntausten ja trendien ennakoimiseksi. Data-analytiikkaan on olemassa useita laajoja ohjelmistokokonaisuuksia. Suljetun lähdekoodin ohjelmistoja ovat esimerkiksi SAS ja IBM SBSS. Vaihtoehtoja on myös avoimen lähdekoodin puolella, esimerkiksi R ja Python. (KDNuggets 2016).

Datan hyödyntämisen mahdollisuudet on huomattu myös yrityksissä; Gartner:in (2015, s. 3) tutkimuksen mukaan data-analytiikka on selvästi suurin teknologian investointikohde. Taulukossa 1 on esitettynä tutkimuksen tulokset vuodelta 2014 ja 2015. Taulukossa olevat luvut kertovat, kuinka monta prosenttia teknologiainvestoinneista ohjautuu tietylle osa-alueelle. Tämän perusteella voidaan sanoa, että investoinnit data-analytiikkaan ovat huomattavassa kasvussa.

Taulukko 1: *Teknologiainvestoinnit 2014 – 2015 (Gartner 2015, s. 3)*

Sija	Investoinnin kohde	2014	2015
1	Data-analytiikka	41 %	50 %
2	Infrastruktuuri ja palvelinkeskukset	31 %	37 %
3	Pilvipalvelut	27 %	32 %
4	Toiminnanohjausjärjestelmä	26 %	34 %
5	Mobiili	24 %	36 %
6	Digitalisaatio ja digitaalinen markkinointi	17 %	11 %
7	Turvallisuus	13 %	11 %
8	Verkkotyöskentely	12 %	12 %
9	Asiakassuhteet ja -kokemus	11 %	8 %
10	Toimialan erikoisohjelmistot	9 %	10 %
11	Vanhojen ohjelmistojen uudistaminen	7 %	7 %
12	Liiketoiminto-ohjelmistot	6 %	2 %
n=2,793			

Tämän työn tavoitteena on vertailla avoimen ja suljetun lähdekoodin data-analytiikkaohjelmistoja kustannusnäkökulmasta. Tavoitteena on tutkia, mistä data-analytiikkaohjelmistojen elinkaarikustannukset muodostuvat.

Työ toteutetaan kirjallisuuskatsauksena käyttäen kvalitatiivisia tutkimusmenetelmiä. Lähteinä käytetään alan kirjoja, artikkeleita ja tutkimuksia. Työssä käytetään myös lukuisia kaupallisia lähteitä käytännönläheisyyden saavuttamiseksi. Lähteissä pyritään painottamaan kaikkein tuoreimpia julkaisuja etenkin data-analytiikan osalta, sillä ala kehittyy erittäin nopeasti. Työssä keskitytään avoimen ja suljetun lähdekoodin data-analytiikkaohjelmistojen kustannuksiin. Työssä vertaillaan ohjelmistojen komponentteja ja ominaisuuksia, mutta se tehdään nimenomaan kustannusnäkökulmasta. Data-analytiikan hyödyntäminen ei ole työn keskiössä, mutta sitä sivutaan useaan otteeseen.

Työ on jaettu johdannon jälkeen neljään eri osaan: kahteen teoriapainotteiseen lukuun, vertailulukuun sekä johtopäätöksiin. Työ alkaa avoimen ja suljetun lähdekoodin käsitteiden selvittämisellä. Ideana on esitellä keskeisimmät erot näiden kahden eri ohjelmistotyypin välillä. Tämän jälkeen siirrytään tarkastelemaan data-analytiikkaohjelmistoja niiden hyödyntämisen ja kustannusrakenteen näkökulmasta. Luvussa neljä vertaillaan avoimen ja suljetun lähdekoodin data-analytiikkaohjelmistojen kustannuksia. Tässä luvussa pyritään ottamaan huomioon kaikki

näiden kustannuksiin vaikuttavat tekijät. Luvussa viisi esitellään johtopäätökset ja tiivistetään tämän työn tulokset.

2 AVOIN JA SULJETTU LÄHDEKOODI

Ohjelmistot voidaan luokitella kahteen eri luokkaan lähdekoodin avoimuuden perusteella; avoimeen ja suljettuun lähdekoodiin. Tässä luvussa määritellään sekä suljettu että avoin lähdekoodi ja tarkastellaan näiden kahden eri luokan ominaispiirteitä. Perinteisesti ohjelmistot, joita yritykset käyttävät, ovat suljetun lähdekoodin ohjelmistoja. Yritykset ovat kuitenkin yhä enemmän ottamassa käyttöön myös avoimeen lähdekoodiin perustuvia ratkaisuja.

2.1 Suljettu lähdekoodi

Suljetun lähdekoodin ohjelmisto tarkoittaa ohjelmistoa, jonka tekijänoikeudet ja lisenssin omistaa jokin tietty taho. Tämän vuoksi suljetun lähdekoodin ohjelmistosta käytetään myös termejä omisteinen ohjelmisto ja omisteinen lähdekoodi. Lähtökohtana on se, että lähdekoodi on tekijänsä omaisuutta (Tucker, Morelli & De Silva, 2011, s. 4). Ohjelmistoja kehitetään hyvin usein yrityksissä, jolloin ohjelmisto ja sen lähdekoodi ovat kyseisen yrityksen omistamia, ellei toisin ole sovittu. Suljetun lähdekoodin lisenssi estää usein käyttäjiä asentamasta ohjelmistoa useammalle kuin yhdelle päätelaitteelle. Lisenssi estää myös kopioimasta, muuttamasta ja jakamasta ohjelmistoa eteenpäin (Tucker et al., 2011, s. 4). Jokaisella yrityksellä on omat lisenssinsä käytössä, joten käyttöehdot riippuvat kuitenkin tapauskohtaisesti ohjelmistoista. On myös olemassa lisenssejä, jotka sallivat ohjelmiston asentamisen useammalle kuin yhdelle päätelaitteelle. Suljetun lähdekoodin ohjelmistot ovat usein maksullisia. Vaikka lisenssi kieltääkin ohjelmiston kopioimisen, usein ohjelmistoista on saatavilla laittomia kopioita.

Suljetun lähdekoodin ohjelmistot ovat niin sanottuja perinteisiä ohjelmistoja. Kun yritykset kehittävät sisäisesti uusia ohjelmistotuotteitaan myytäväksi, on kyseessä suljetun lähdekoodin ohjelmisto. Käyttäjällä ei ole pääsyä lähdekoodiin, eikä käyttäjä voi tehdä muutoksia tai korjauksia ohjelmistoon. Ohjelmiston virheistä on ilmoitettava lisenssin omistavalle yritykselle ja odotettava virheenkorjauspäivitystä. Esimerkiksi kriittiset tietoturva-aukot korjataan usein nopeasti, mutta muiden virheiden korjaamiseen saattaa kulua kauan. Suljetun lähdekoodin ohjelmistojen kehittäjillä on aina jonkinlainen vastuu ohjelmiston virheistä ja sen aiheuttamista ongelmista. Vastuiden jakautuminen on mainittu ohjelmiston lisenssissä ja käyttöehdoissa.

Käyttäjät, varsinkin yritysmaailmassa, toivovat usein ohjelmistoilta räätälöityjä ratkaisuja omiin tarpeisiinsa. Ohjelmistokehittäjillä ei ole välttämättä resursseja kehittää kaikkia käyttäjien toivomia ominaisuuksia ja lisäosia. Suljetun lähdekoodin ohjelmistoihin voidaan tällaisissa tilanteissa avata rajapintoja ulkopuolisille kehittäjille. Rajapintojen avulla kolmannet osapuolet voivat kehittää käyttäjien toivomia komponentteja ja lisäosia alkuperäiseen ohjelmistoon. Näin käyttäjien tarpeet tulevat paremmin tyydytetyiksi ja ohjelmistosta tulee enemmänkin ekosysteemin kaltainen alusta, jota kolmannet osapuolet voivat käyttää hyödyksi. Esimerkiksi Autodesk on avannut AutoCad ohjelmistonsa ulkopuolisille kehittäjille erittäin hyvin tuloksin. Tulevaisuudessa yhä useammat suljetun lähdekoodin ohjelmistot tulevat avaamaan rajapintoja ja muuttuvat enemmän ekosysteemeiksi. (Bosch 2009, s. 1-4).

2.2 Avoin lähdekoodi

Avoimen lähdekoodin ohjelmisto jaetaan ja lisensoidaan yhdessä sen lähdekoodin kanssa (Tucker et al. 2011, s. 4). Se, että ohjelmisto on ilmaiseksi ja vapaasti saatavilla jossakin, ei tee siitä avoimen lähdekoodin ohjelmistoa. Ohjelmistolla tulee olla avoimen lähdekoodin lisenssi, sillä ilman lisenssiä ohjelmistoa koskevat normaalit tekijänoikeuskäytännöt (Laakkonen 2013). Avoimen lähdekoodin ohjelmistoissa vapaaehtoiset korjaavat virheitä ja kehittävät ohjelmistoa. Näin ollen kehittäjien yhteisö on hyvin tärkeässä roolissa avoimen lähdekoodin ohjelmistojen kehityksessä. (Yu 2013, s. 46).

2.2.1 Vapaa ohjelmisto ja avoin lähdekoodi

Avoimen lähdekoodin taustalla on kaksi eri suuntausta; *vapaa ohjelmisto* (free software) ja *avoin lähdekoodi* (open source). Vapaa ohjelmisto -aatteen aloitti Richard Stallman vuonna 1983, jolloin hän aloitti GNU -projektin, jonka tavoitteena oli luoda täysin vapaa ja avoin käyttöjärjestelmä. Hän loi vapaa ohjelmisto-määritelmän 1985 perustaessaan Free Software Foundationin. (Tucker et al. 2011, s. 4-5; Free Software Foundation 2017a). Vapaa ohjelmisto tarkoittaa ohjelmistoa, joka pyrkii kunnioittamaan vapautta ja ohjelmistoon liittyvän yhteisön oikeuksia. Tämä tarkoittaa, että käyttäjillä on vapaus käyttää, kopioida, jakaa ja kehittää ohjelmistoa niin kuin itse haluavat. Se, että ohjelmisto voidaan luokitella vapaaksi ohjelmistoksi tarkoittaa, että käyttäjät voivat:

1. käyttää ohjelmistoa tarkoituksesta tai päämäärästä riippumatta
2. muuttaa ohjelmiston lähdekoodia
3. jakaa ohjelmistoa vapaasti kenelle vain
4. kehittää ohjelmistoa ja jakaa nämä parannukset yhteisön käyttöön. (Tucker et al. 2011, s. 12; Free Software Foundation 2017b).

Vapaa ohjelmisto -aatteessa tärkeintä on käyttäjien vapaus, ei hinta, vaikkakin suurin osa näistä ohjelmistoista on saatavilla veloituksetta. Aatteen kannattajat kutsuvat muita ohjelmistoja ei-vapaiksi. Näissä ei-vapaissa ohjelmistoissa on aina joku taho, joka kontrolloi ohjelmistoa ja sen käyttöä. Vapaiden ohjelmistojen aatteeseen kuuluu, että käyttäjät kontrolloivat ohjelmistoa. (Free Software Foundation 2017b).

Avoin lähdekoodi -suuntaus lähti liikkeelle vuonna 1998, kun osa vapaiden ohjelmistojen kannattajista päätyi ajatukseen käyttää kaikista tämän suuntauksen ohjelmistoista nimitystä avoin lähdekoodi. Vapaa ohjelmisto -liikkeelle ohjelmistojen vapaus on poliittinen ja filosofinen asia, kun taas avoin lähdekoodi -suuntauksen kannattajat korostavat enemmänkin käytännön hyötyjä avoimen lähdekoodin ohjelmistojen kehityksessä. (Open Source Initiative 2017a). Edelleenkin nämä kaksi erilaista suuntausta ovat esillä ja varsinkin vapaa ohjelmisto -suuntaus korostaa kotisivuillaan aatteen poliittisia motiiveja ja eroa avoin lähdekoodi -suuntaukseen. Käytännössä nämä kaksi suuntausta tarkoittavat samaa asiaa ja englanniksi niistä käytetäänkin yhteisnimitystä **FOSS** (free and open software). Edellä mainitut neljä vapaan ohjelmiston vaatimusta toteutuvat käytännössä myös avoimen lähdekoodin ohjelmistojen kohdalla. Tässä työssä käytetään selvyyden vuoksi termiä **avoin lähdekoodi** kuvaamaan molempia suuntauksia.

2.2.2 Avoin lähdekoodi käsitteenä

Avoimen lähdekoodin ohjelmistot ovat usein jonkin yhteisön tuottamia. Myös yritykset voivat käynnistää omia avoimen lähdekoodin projektejaan, mutta hyvin monet ovat lähteneet liikkeelle muuta kautta (The R Foundation 2017). Avoimen lähdekoodin ohjelmistoissa lähdekoodi on avointa, eli kaikkien saatavilla (Prusansky 2011, s. 65). Avoimen lähdekoodin lisenssillä varustetut ohjelmistot ovat lähes poikkeuksetta saatavilla maksutta. Tämän vuoksi avoimen lähdekoodin ohjelmistot sopivat hyvin sellaisille markkinoille, joilla omistetun

lisenssin (suljetun lähdekoodin) ohjelmistojen ei ole kiinnostavaa toimia. Esimerkkejä tällaisista markkinoista ovat kehittyvät markkinat ja voittoa tavoittelemattomat organisaatiot. (Tucker et al. 2011, s. 4). Avoimen lähdekoodin ohjelmistoja käyttävät paljon myös yksityishenkilöt. Avoin lähdekoodi on kasvattanut suosiotaan yritysmaailmassa ja Tilastokeskuksen tutkimuksen mukaan (2011) lähes 80 prosentissa yli 10 hengen yrityksiä käytetään avoimen lähdekoodin ohjelmistoja. Conry-Murray toteaakin artikkelissaan (2006, s. 51), että nykyisin avoimen lähdekoodin ohjelmistot ovat täysin kilpailukykyinen vaihtoehto suljetun lähdekoodin ohjelmistoille monilla eri aloilla.

Avoin lähdekoodi ei tarkoita ainoastaan sitä, että käyttäjällä on pääsy ohjelmiston lähdekoodiin. Avoimen lähdekoodin ohjelmistoissa tulee täyttyä seuraavat kriteerit:

1. ohjelmisto on vapaasti jaettavissa ja uudelleen jaettavissa
2. lähdekoodi tulee jaella ohjelmiston mukana tai sen tulee olla helposti saatavilla
3. johdettuja teoksia tulee voida jaella ja käyttää samoin ehdoin kuin alkuperäistäkin ohjelmistoa
4. lisenssi voi rajoittaa johdettujen teosten jakelua vain, jos lisenssi sallii erillisten korjaustiedostojen leviämisen
5. lisenssi ei saa syrjiä ihmisiä tai ihmisryhmiä
6. lisenssi ei saa syrjiä mitään tiettyjä aloja tai pyrkimyksiä
7. ohjelmiston oikeudet pysyvät myös kaikilla joille ohjelmisto uudelleen jaellaan
8. lisenssioikeudet eivät saa olla riippuvaisia tuotteesta, jonka kanssa sitä jaellaan
9. lisenssi ei saa aiheuttaa rajoituksia muille ohjelmistoille
10. lisenssiin ei tule asettaa mitään varaumia minkään teknologian tai muun sellaisen vuoksi. (Open Source Initiative 2017b; Välimäki 2009, s. 201-202).

2.2.3 Avoimen lähdekoodin lisenssit

Avoimen lähdekoodin ohjelmistot on aina varustettu jollakin lisenssillä (Wormser 2010, s. 22). Tämä johtuu siitä, että vaikka ohjelmisto olisikin vapaasti saatavilla sitä koskevat normaalit tekijänoikeudet, mikäli sitä ei erikseen ole lisensoitu avoimeen käyttöön (Laakkonen 2013). Avoimen lähdekoodin lisenssejä on paljon ja ne sisältävät erilaisia rajoituksia uusiokäyttöön ja jakeluun liittyen (Wormser 2010, s. 22-23). David Wormser (2010) kritisoi artikkelissaan

avoimen lähdekoodin ohjelmistojen lisenssejä. Hänen mukaansa ne ovat hyvin usein huonosti laadittuja ja sisältävät hyötykäyttöä haittaavia tekijöitä. Lisensseissä on paljon tulkinnanvaraa, mikä voi aiheuttaa ongelmia tietyissä tilanteissa. Jos suljetun lähdekoodin ohjelmistoon lisää avoimen lähdekoodin komponentin, joidenkin lisenssien mukaan koko ohjelmiston lähdekoodin tulisi tällöin olla julkista. Ideana tässä on se, että ohjelmistoon tehtävät ”parannukset” ja uudet ominaisuudet palautuvat koko yhteisön käyttöön. (Wormser 2010, s. 23). Tämä kuitenkin luonnollisesti hankaloittaa kaupallisten toimijoiden mahdollisuuksia hyödyntää avoimen lähdekoodin komponentteja, jos he haluavat pitää lähdekoodin suljettuna. Edellä mainitun kaltaisista epäselvyyksistä on käyty selvittelyä jopa oikeudessa asti (Rubens 2015). Suljetun lähdekoodin ohjelmistoissa ei tällaisia ongelmia ole, sillä yritysten on taloudellisten intressien vuoksi kehitettävä käyttöehtojaan paremmiksi (Wormser 2010, s. 23).

Avoimen lähdekoodin lisenssejä on olemassa kahta päätyyppiä: *copyleft lisenssit* ja *ei-copyleft lisenssit*. Copyleft-lisenssejä voidaan myös kutsua vastavuoroisuutta vaativiksi lisensseiksi ja ei-copyleft lisenssejä salliviksi lisensseiksi. (Välimäki 2009, s. 205-206). Vastavuoroisuutta vaativissa lisensseissä ideana on, että kukaan ei voi ryhtyä ohjelmiston omistajaksi tekemällä siihen muutoksia (Free Software Foundation 2017c; Laakkonen 2013). Vastavuoroisuutta vaativissa lisensseissä yksikin tällä lisenssillä varustettu ohjelmiston komponentti muuttaa koko ohjelmiston lisensoitavaksi samalla lisenssillä. Vastavuoroisuutta vaativiin lisensseihin voi kuulua myös sellaisia lisenssejä, jotka eivät aseta edellä mainittua vaatimusta. (Välimäki, 2009, s. 206-207). Esimerkkejä vastavuoroisuutta vaativista lisensseistä ovat **GNU-lisenssit**, joita ovat muun muassa GNU General Public License (GPL) ja GNU Lesser General Public License (LGPL). Näistä varsinkin ensimmäinen on erittäin suosittu lisenssi, josta on julkaistu useita eri versioita. Muita suosittuja lisenssejä ovat MIT License ja Apache License, jotka ovat sallivia lisenssejä. Komponentteja yhdistellessä tulee ottaa huomioon eri lisenssien yhteensopivuus; kaikki avoimen lähdekoodin lisenssit eivät ole yhteensopivia keskenään. Vaikka lisenssejä on olemassa paljon, viisi suosituinta kattaa 75% kaikista avoimen lähdekoodin lisensseistä (Black Duck Software 2017). Viisi suosituinta lisenssiä ovat:

1. MIT License
2. GNU General Public License (GPL) 2.0
3. Apache License 2.0
4. GNU General Public License (GPL) 3.0

5. BSD License 2.0 (3-clause, New or Revised). License (Black Duck Software 2017).

2.2.4 Avoimen lähdekoodin kritiikki

Avoimessa lähdekoodissa käyttäjillä on pääsy ohjelmiston lähdekoodiin, jolloin he voivat korjata virheitä, lisätä rajapintoja, ratkaista ongelmia ja käyttää sitä ilmaiseksi. Muutokset joita käyttäjät tekevät, palautetaan alkuperäiseen lähdekoodiin, jotta ne olisivat kaikkien saatavilla. Näin avoimen lähdekoodin ohjelmisto kehittyy. Avoin lähdekoodi luottaa jatkuvaan kehitykseen lyhyissä sykleissä. (Raghunathan, Prasad, Mishra & Chang 2005, s. 903-904). Avoimen lähdekoodin ohjelmistojen kehityksessä on siis sama idea kuin ketterissä menetelmissä. Lisäksi avoin lähdekoodi luottaa avoimiin standardeihin, eli se käyttää avoimen lähdekoodin lisenssejä. Avoimen lähdekoodin kannattaja sanovat, että ohjelmistot ovat julkisia hyödykkeitä, joiden pitäisi olla vapaasti jaettavissa, käytettävissä ja jatkokehitettävissä. (Raghunathan et al. 2005, s. 903-905). Kannattajien mukaan tässä piilee myös avoimen lähdekoodin suurin vahvuus; muutokset ovat pieniä ja kehittäjiä on parhaimmillaan tuhansia ympäri maailman.

Avoimen lähdekoodin ratkaisut ovat saaneet myös kritiikkiä. Kriitikoiden mukaan avoimen lähdekoodin ohjelmiston laatu kärsii vapaamatkustajista. Kaikki olettavat muiden kehittävän ohjelmistoa, mutta eivät ole itse siihen valmiita. Todellisuudessa vain pieni, aktiivinen joukko kehittää ohjelmistoa, jolloin koko potentiaalia ei saavuteta. Vaikka ohjelmistoa kehitetäänkin yhteisön hyväksi ja kaikkien käyttöön, on sanottu myös, että kehittäjien tekemä työ ei ole täysin pyyteetöntä. Kehittäjät pyrkivät saamaan mainetta yhteisön sisällä ja esittelevät taitojaan tulevina työntekijöinä. Tällainen vapaanehtoinen kehitystyö voi myös muuten edistää kehittäjien omaa uraa. (Raghunathan et al. 2005, s. 903). Ohjelmiston kehityksen kannalta ei ole kovinkaan suurta merkitystä, mitkä ovat avoimen lähdekoodin kehittäjien ja käyttäjien todelliset motiivit.

Niin kuin edellisessä kappaleessa mainittiin, avoimen lähdekoodin ohjelmistoissa luotetaan jatkuvaan kehitykseen lyhyissä sykleissä. Lisäksi kehittäjät voivat asua ympäri maailman. Tämä aiheuttaa suuria haasteita projektinhallinnalle. Kriitikoiden mukaan projektinhallinnan puute huonontaa ohjelmiston laatua. Kannattajien mukaan juuri tämä on avoimen lähdekoodin parhaita puolia; kun lähdekoodi on avointa, käyttäjät ympäri maailman testaavat ja uudelleen

testaavat koodia, jolloin se kehittyy. Kannattajien mukaan avoimen lähdekoodin ohjelmistojen laatu ei synny projektin johtamisesta, vaan avoimuudesta. (Raghunathan et al. 2005, s. 903-910). Kirjallisuudessa on myös kuvailtu avointa lähdekoodia termillä ”Bazaar style”, jolla viitataan rönsyilevään, päällekkäiseen, mutta ketterään kehitysmenetelmään, joka nimenomaan juontuu lähdekoodin avoimuudesta (Raymond 1999).

3 DATA-ANALYTIIKKA

Data-analytiikan rooli nyky-yhteiskunnassa kasvaa jatkuvasti. Yksilöt omistavat yhä enemmän laitteita ja teknologiaa, jotka taas tuottavat ja tallentavat dataa monessa eri muodossa (Elgendy & Elgral 2016, s. 1072). Tutkimusten mukaan yrityksissä kiertävän datan määrä kasvaa keskimäärin 200% joka vuosi, ja vuonna 2015 yrityksien kokonaisdatan määrän ennustettiin kasvavan 800% viiden vuoden sisällä (Kościelniak & Puto 2015, s. 1054). Datan varastointikapasiteetti on viime vuosikymmenellä kasvanut valtavasti, joka taas on mahdollistanut suurien datamäärien varastoinnin kohtuulliseen hintaan (Elgendy & Elgral 2016, s. 1072). Lisäksi datan keräyksen keinot ovat muuttuneet ja kehittyneet, jonka tuloksena valtavaan datan määrään on nykyään helppo päästä käsiksi (Elgendy & Elragal 2016, s. 1072). Kyseistä valtavasta datan määrästä käytetään käsitettä ”*big data*”.

3.1 Big data

Big datalla viitataan yleisesti dataan, jonka *volyymi*, *vaihtelevuus*, *nopeus* ja *arvo* ovat niin suuria, että sitä on vaikea hallita tämän hetkisillä analysointityökaluilla (Kościelniak & Puto 2015, s.1053). Näin ollen big datan kuvaamiseen käytetään usein niin sanottua neljää V:tä, jotka ovat englanniksi: *volume* (suuri datan määrä), *variety* (monenlaista dataa tulee erilaisista lähteistä), *velocity* (nopeus, jolla dataa tuotetaan) ja *veracity* (vaihteleva tarkkuus) (Elgendy & Elgral 2016, s. 1072). Nämä neljä V:tä ovat big datan ominaispiirteitä, johon koko big data ja siihen liittyvä ideologia perustuu. Toisaalta Boyd ja Crawford (2012, s. 663-664) määrittelivät big datan *kulttuuriin*, *teknologiaan* ja *teoriaan* perustuvaksi ilmiöksi, joka perustuu seuraavan kolmen tekijän vuorovaikutukseen:

1. teknologia: parhaan mahdollisen laskennallisen suorituskyvyn yhdistäminen tarkkojen algoritmien kanssa, joiden avulla kerätään, analysoidaan, yhdistetään ja vertaillaan suuria datamääriä
2. analyysi: suuntausten tunnistaminen datasta, joiden avulla voidaan tukea ekonomisia, sosiaalisia, teknillisiä ja juridisia väitteitä
3. mytologia: usko siihen, että suuret määrät dataa tarjoavat parempaa tietoa.

Toisin sanoen big data perustuu siihen uskoon, että suuri datan määrä monesta eri lähteestä yhdistettynä nykyteknologiaan ja -algoritmeihin mahdollistaa parempien, suurempien ja tarkempien päätösten tekemisen.

Moderni data-analytiikka on tiede, jossa kehittyneitä analyyttisiä menetelmiä käytetään suurien datajoukkojen analysointiin. Arviolta noin 80% yritysten datasta on epäjärjestyksessä, mikä vaikeuttaa datan prosessointia ja analysointia perinteisillä menetelmillä kuten taulukkolaskentaohjelmistoilla (Kościelniak & Puto 2015, s. 1054). Big data -analyysin avulla tämä suuri datan määrä pyritään muuttamaan kuvaavampaan muotoon. Analyysista saadun raportin tulisi olla helposti luettavissa ja looginen. Sen antamien arvioiden tulisi olla johdonmukaisia, ja sen pitäisi helpottaa tulevaisuuden tilanteen ja päätöksenteon ennustamista. (Janssen, van Der Voort & Wahyudi 2017, s. 338-339).

Big data -analytiikan avulla voidaan parantaa huomattavasti päätöksentekoa, minimoida riskejä ja löytää datasta arvokkaita oivalluksia, joita ei muuten olisi mahdollista löytää (Elgendy & Elgral 2016, s. 1072). Esimerkiksi markkinointiyritykset voivat mainostaa asiakkailleen asioita, joita he haluavat ja tarvitsevat ilman, että asiakkaat ärsyyntyvät. Parhaimmillaan big data -analytiikan avulla voidaan tulevaisuudessa ennustaa esimerkiksi epidemioita, tulipaloja tai parhaan ajan ostaa lentolippu. (Jobs, Gilfoil & Aukers 2016, s. 19-20).

3.2 Data-analytiikkaohjelmistot

Yleisesti data-analytiikkaohjelmistojen tarkoituksena on antaa käyttäjälle mahdollisuus vastaanottaa data, muuntaa data haluttuun muotoon ja esittää se graafisesti käyttäjille. Data-analytiikkaohjelmistot pyrkivät siis helpottamaan datan jalostamista ja sen esittämistä. Lisäksi ne antavat usein mahdollisuuden kutsua muita ohjelmointikieliä, jolloin ohjelmistoon voi lisätä ominaisuuksia kohtuullisen helposti.

Data-analytiikkaohjelmistoja on useita erilaisia; markkinoilta löytyy sekä avoimen lähdekoodin että suljetun lähdekoodin vaihtoehtoja. Tässä osiossa vertaillaan kahden eri data-analytiikkaohjelmiston ominaisuuksia: avoimen lähdekoodiin perustuvaa R -ohjelmistoa sekä suljetun lähdekoodin SAS -ohjelmistoa. SAS on valittu kyseiseen osioon, koska se on eräs suosituimmista suljetun lähdekoodin data-analytiikkaohjelmistoista, ja se tarjoaa kattavan

näkökulman kaupallisista ohjelmistoista. R –ohjelmisto on valittu osioon, koska se on tällä hetkellä markkinoiden suosituin data-analytiikkaohjelmisto, ja se on avoimen lähdekoodin ohjelmisto (KDnuggets 2016). Lisäksi vertailtavat ohjelmistot pyrittiin valitsemaan siten, että ne vastaisivat käyttötarkoitukseltaan toisiaan mahdollisimman hyvin.

R tarjoaa käyttäjälleen laajan valikoiman statistisia ja graafisia työkaluja datan analysointiin. R-ohjelmiston kieli perustuu usein tilastollisessa tutkimuksessa käytettyyn kieleen nimeltä S. Eräitä R:n vahvuuksista on sen helppous tuottaa hyvin suunniteltuja kuvaajia ja sen laajennettavuuden potentiaali. R on suunniteltu olemaan oikea ohjelmointikieli. Tämä tarkoittaa sitä, että se on toimiva ja koherentti järjestelmä, eikä vain yhdistelmä spesifejä ja jäykkiä työkaluja, toisin kuin monet muut data-analytiikkaohjelmistot. R antaa käyttäjälleen mahdollisuuden luoda lisää toimintoja määrittelemällä uusia funktioita, joka taas parantaa ohjelmiston laajennettavuutta. Ohjelmiston toiminnallisuuksia on helppo lisätä ulkopuolisten pakettien avulla, ja se voi kutsua useita eri ohjelmointikielisiä komentoja. Lisäksi R antaa käyttäjälleen mahdollisuuden jalostaa sen sisäisiä olioita käyttäen C-kieltä. (The R Foundation 2017).

SAS Analytics Platform on R:lle kaupallinen vaihtoehto. SAS tarjoaa käyttäjilleen työkaluja dataan ja data-analytiikkaan liittyviin haasteisiin. Se tarjoaa valikoiman skaalautuvia metodeja, joiden avulla voidaan ratkaista erikokoisia ja -tyyppisiä ongelmia. Lisäksi SAS tarjoaa tuen kaikille suurille ohjelmointikielille sekä metodeille ja standardeille data-analytiikkaan liittyen. SAS tarkastaa analyysien tarkkuuden ja oikeellisuuden vertaamalla testejä viitekehyksiin ja toimialan yleisiin käytänteisiin. Ei-teknisille käyttäjille SAS tarjoaa visuaalisen käyttöliittymän, jonka avulla käyttäjät voivat suorittaa analyyseja ilman vahvaa teknistä taustaa. Ohjelmisto tarjoaa myös pilvipalveluvaihtoehdon sekä mahdollisuuden käyttää datan salausta ja turvallisia verkkoyhteyksiä. (SAS 2017).

Avoimen ja suljetun lähdekoodin data-analytiikkaohjelmistojen suurimmat erot tulevat niiden periaatteellisista eroavaisuuksista. Avoimen lähdekoodin ratkaisu antaa käyttäjilleen mahdollisuuden tehdä ja muokata käytännössä kaikkia ohjelmiston toiminnallisuuksia. Esimerkiksi käyttäjille annetaan mahdollisuus muokata R:n olioita ja toiminnallisuuksia käyttäen C-ohjelmointikieltä. Lisäksi käyttäjät voivat helposti ladata ja hyödyntää muiden tekemiä uusia toiminallisuuksia ja algoritmeja lataamalla paketteja ohjelmistoon. Toisaalta

suljetun lähdekoodin ratkaisu pyrkii antamaan käyttäjilleen paketin, joka on heti valmis käytettäväksi. SAS tarjoaa käyttäjilleen visuaalisen käyttöliittymän, joka mahdollistaa ohjelmiston käytön ilman teknistä taustatietoa. Ohjelmisto tarjoaa valmiit ratkaisut pilvipalveluihin ja datan salaukseen. Lisäksi SAS tarjoaa esimerkiksi valmiita datalouhintaohjelmistoja, joita on erittäin helppo yhdistellä SAS:in muihin ohjelmistoihin. Toisin sanoen R antaa käyttäjilleen enemmän vapauksia, kun taas SAS tarjoaa helpon mutta vähemmän muokattavan ratkaisun suurimpaan osaan käytännön ongelmista.

4 DATA-ANALYTIIKKAOHJELMISTOJEN KUSTANNUSVERTAILU

Uuden ohjelmiston käyttöönotto yrityksissä on usein monimutkainen prosessi, koska siihen sisältyy monia eri vaiheita, tasoja sekä tahoja. Näistä syistä se on usein hyvin kallis prosessi, ja se on suunniteltava tarkasti etukäteen, jotta se voidaan budjetoida oikein. Yleisesti ohjelmistojen käyttöönottoon voidaan ajatella kuuluvan seuraavat kuusi tehtävää:

1. rajapintojen suunnittelu
2. integraation muihin tietojärjestelmiin
3. datan muuntaminen
4. uuden laitteiston hankinta
5. ohjelmiston testaus
6. ohjelmiston asentaminen tarvittaville laitteille. (Yale 2001).

Tässä luvussa tarkastellaan avoimen ja suljetun lähdekoodin data-analytiikkaohjelmistojen kustannuksia. Tarkoituksena on vertailla ohjelmiston elinkaaren eri vaiheissa aiheutuvia kustannuksia. Lisäksi luvussa arvioidaan avoimen ja suljetun lähdekoodin ohjelmistojen eri osa-alueiden riskejä ja niistä aiheutuvia kustannuksia.

4.1 Ohjelmiston hankinta ja käyttöönotto

Ohjelmiston hankintaan ja käyttöönottoon liittyy paljon erilaisia kustannuksia. Varsinainen käyttöönottoprosessi voi olla hyvinkin pitkä ja kallis, sillä kustannuksia muodostuu muustakin kuin varsinaisesta ohjelmiston hankinnasta. Myös erilaiset muutoksista aiheutuvat kustannukset, kuten työntekijöiden koulutus, integraatiot ja varsinaisen tuottavan työn katkokset tulee ottaa huomioon kustannuksia arvioitaessa. Koko käyttöönottoprojekti on mahdollista ulkoistaa niin suljetun kuin avoimen lähdekoodinkin ohjelmistoissa (Nagy, Yassin & Bhattacherjee 2010, s. 149-150). Tämä aiheuttaa lisää kustannuksia, mutta omien työntekijöiden tuottava työ ei keskeydy.

4.1.1 Lisenssimaksut

Lisenssimaksut ovat yksi osa ohjelmistosta aiheutuvaa kokonaiskustannusta. Ohjelmistoon on yleensä hankittava käyttölisenssi, jotta yritys voi kyseistä ohjelmistoa käyttää. Suljetun lähdekoodin data-analytiikkaohjelmistoissa käyttölisenssi on maksullinen. Yksi vaihtoehto on ostaa myös ohjelmiston käyttö palveluna (Software-as-a-Service). Tällöin varsinaista lisenssimaksua ei ole, vaan ohjelmistosta veloitetaan esimerkiksi kuukausittain. Tällöin palvelu voi sisältää esimerkiksi käyttäjätuen sekä muita lisäpalveluita. (Sääksjärvi, Lassila & Nordström 2005, s. 177-178).

Avoimen lähdekoodin ohjelmistoissa lisenssimaksuja ei ole ja tätä korostaa myös avoimen lähdekoodin ratkaisuja ja palveluja tarjoava ActiveState (2016, s. 2). Ohjelmistot, kuten R, ovat ladattavissa maksutta niin yrityksille kuin yksityishenkilöillekin. Avoimen lähdekoodin ohjelmistojakin voidaan tarjota palveluna, jolloin siihen sisältyy ohjelmiston lisäksi erilaisia palveluja. Se, että ohjelmistossa ei ole lisenssimaksua, alentaa kustannuksia verrattuna lisenssimaksullisiin suljetun lähdekoodin ohjelmistoihin. Varsinkin suurissa ohjelmistoratkaisuissa (jollainen data-analytiikkaohjelmiston hankinta isolle yritykselle on) lisenssimaksut voivat olla hyvinkin korkeita. Tällöin avoimen lähdekoodin ratkaisu voi säästää huomattaviakin summia lisenssimaksuista. Lisenssimaksu on kuitenkin prosentuaalisesti melko pieni osa ohjelmistosta aiheutuvaa kokonaiskustannusta (Arnold 2011, s. 44). Tämän vuoksi lisenssimaksuttomuudelle ei kannata antaa liikaa painoarvoa kustannusten osalta. Toisaalta lisenssimaksujen puuttuminen voi alentaa yritysten kynnystä pilotoida erilaisia ratkaisuja ennen varsinaisen ostopäätöksen tekemistä. Tämä voi olla yrityksille erittäin hyödyllistä, jos vastaavan ohjelmiston hankinnasta ei ole aikaisempaa kokemusta.

4.1.2 Käyttöönotto

Ohjelmiston kokonaiskustannuksiin vaikuttaa myös se, kuinka pitkä aika ohjelmiston käyttöönottoon kuluu. Käyttöönotto vaatii työntekijöiltä resursseja sekä mahdollisesti ulkopuolista konsultointia. Data-analytiikkaohjelmiston vaihdon tai hankinnan yhteydessä täytyy huomata, että niin kauan, kun ohjelmisto ei ole yrityksessä käytössä, esimerkiksi päätöksentekoon ei saada tarvittavaa tukea data-analytiikasta. Tämä voi pahimmillaan huonontaa päätöksentekoa, jolla voi olla suuriakin kustannusvaikutuksia. Käyttöönottoon

kuluva aika riippuu myös siitä, tehdäänkö käyttöönotto itse, vai onko mukana ulkopuolisia konsultteja. Tässä yhteydessä yrityksen tulee huomioida myös vaihtoehtokustannukset; onko järkevämpää ostaa käyttöönottoprojekti ulkopuoliselta vai käyttää oman henkilöstön työaikaa käyttöönottoon?

Suljetun lähdekoodin tapauksessa käyttöönoton voi yleensä ostaa ohjelmistoa tarjoavalta yritykseltä. Tässäkin tilanteessa käyttöönottoprojekti vaatii jonkinlaista työpanosta myös omalta henkilökunnalta, vaikka muutoin käyttöönotto olisikin ulkoistettu. Suljetun lähdekoodin ohjelmiston käyttöönottoon saattaa kulua pitkiäkin aikoja, jos integraatiot nykyisiin järjestelmiin tuottavat paljon työtä. Avoimen lähdekoodin tapauksessa tilanne on hyvin samankaltainen. Palvelun voi yleensä ostaa ulkopuolelta, ja avoimen lähdekoodin tapauksessa yritys voi kilpailuttaa käyttöönottoprojektin. Tämä ei yleensä suljetun lähdekoodin ohjelmistoissa ole mahdollista, vaan käyttöönotto joudutaan ostamaan ohjelmistoa tarjoavalta yritykseltä tai sen yhteistyökumppanilta. Avoimessa lähdekoodissa yritys ei ole ”naimisissa” yhden ohjelmistotoimittajan kanssa, vaan kaikki palvelut voidaan kilpailuttaa tapauskohtaisesti. Kustannusnäkökulmasta tämä alentaa hieman ulkopuolisen käyttöönoton kustannuksia avoimen lähdekoodin ohjelmistoissa. On kuitenkin sanottu, että avoimen lähdekoodin ohjelmistojen osaajia voi olla hankalampi löytää ja heistä aiheutuvat kustannukset voivat olla korkeampia (Conry-Murray 2006, s. 54).

Käyttöönottoprojektit avoimella ja suljetulla lähdekoodilla ovat hyvin samanmittaisia. Avoimen lähdekoodin ohjelmistoissa käyttöönottoa nopeuttaa koodin avoimuus ja sitä kautta integraatioiden tekeminen voi olla nopeampaa. Tämä on kuitenkin hyvin tapauskohtaista. Isoilla yrityksillä, kuten SAS, on kokemusta kaikenlaisista käyttöönottoprojekteista, joten vastaavanlainen käyttöönotto on voitu tehdä jo aikaisemmin. Tällöin ongelmia on vähemmän ja käyttöönotto voi sujua nopeastikin.

4.1.3 Työntekijöiden koulutus

Uusien ohjelmistojen käyttöönoton yhteydessä ohjelmistoa käyttävät työntekijät tulee kouluttaa ohjelmiston käyttöön (Sommerville 2005, s. 613). Koulutus kannattaa aloittaa jo ohjelmiston käyttöönottovaiheessa, sillä käyttäjien mukaan ottaminen parantaa heidän tyytyväisyyttään huomattavasti (Leonard-Barton & Kraus 1985). Data-analytiikkaohjelmiston käyttöönotossa

koulutettavia työntekijöitä ovat ainakin analyytikot, mutta myös muut, jotka tarvitsevat ohjelmistoa esimerkiksi raporttien tarkasteluun. Uuden ohjelmiston käyttöönotto, tai vanhan ohjelmiston korvaaminen uudella, aiheuttaa usein vastarintaa työntekijöiden keskuudessa (Leonard-Barton & Kraus 1985). Työntekijöiden osallistuttaminen käyttöönottoprojektiin on tärkeää, jotta muutosvastarintaa olisi mahdollisimman vähän. Työntekijöiden koulutus aiheuttaa kustannuksia yrityksille, mutta jos koulutusta ei tehdä kunnolla, ei työntekijä välttämättä osaa käyttää ohjelmistoa parhaalla mahdollisella tavalla. Tällöin kustannusvaikutus on olla huomattavasti suurempi tuottavuuden ja työmotivaation laskun vuoksi.

Avoimen lähdekoodin ohjelmistot ovat yleensä vapaasti kaikkien saatavilla. Kun käytännössä kaikilla on pääsy ohjelmistoon, käyttäjien lukumäärä voi olla hyvinkin suuri. Tästä on etua yritykselle, joka valitsee avoimen lähdekoodin data-analytiikkaohjelmiston, esimerkiksi R:n. R:ään on olemassa useita verkkokursseja, jotka ovat vapaasti ja veloituksetta kaikkien käytettävissä. Osa nykyisistä ja tulevista työntekijöistä on saattanut käyttää ohjelmistoa jo ennakolta. Tästä on huomattavaa apua ohjelmiston käyttöönotossa ja työntekijöiden koulutuksessa. Se, että ohjelmisto on osalle tuttu, voi myös helpottaa muiden työntekijöiden sopeutumista uuteen ohjelmistoon. Tämä voi vähentää muutosvastarintaa ja koko koulutusprojektin kustannuksia. Joskus voi olla hyödyllistä palkata henkilöitä, jotka osaavat ottaa uutta teknologiaa käyttöön. Tämäkin voi osaltaan auttaa työntekijöiden sopeutumista uuteen teknologiaan. (Nagy et al. 2010, s. 149-150).

Suljetun lähdekoodin data-analytiikkaohjelmistot ovat yleensä laajoja ohjelmistoja, joista ihmisillä ei välttämättä ole kovinkaan paljon kokemusta. SAS on melko vanha ohjelmisto, joten sen eri versiota monet ovat käyttäneet, mutta tilanne ei ole yhtä hyvä kaikkien suljetun lähdekoodin data-analytiikkaohjelmistojen kohdalla. Suljetun lähdekoodin ohjelmistoihin ei yleensä ole saatavilla ilmaisia koulutusmateriaaleja (ei ainakaan yhtä kattavasti kuin avoimeen lähdekoodiin). Näin ollen työntekijöille ohjelmisto ei välttämättä ole etukäteen tuttu ja koulutusta joudutaan tarjoamaan enemmän. Tilanne on sama tulevien työntekijöiden kohdalla. Avoimen lähdekoodin ohjelmistoissa on etuna se, että käyttäjäkunta on usein huomattavasti laajempi kuin vastaavilla suljetun lähdekoodin ratkaisuilla. Tämä pätee melko hyvin myös data-analytiikkaohjelmistojen kohdalla.

Työntekijöiden kannalta tärkeää on ohjelmiston käytettävyys. Niin kuin luvussa 3.2 mainittiin suljetun lähdekoodin data-analytiikkaohjelmistot, kuten SAS, tarjoavat intuitiivisen, helppokäyttöisen ja visuaalisen käyttöliittymän. Tämä helpottaa huomattavasti käyttäjien uuden ohjelmiston omaksumista. Käyttäjät, joilla ei ole teknistä taustaa, eikä kokemusta esimerkiksi ohjelmoinnista, oppivat nopeasti ohjelmiston käytön. Avoimen lähdekoodin ohjelmistoissa tilanne ei välttämättä ole samanlainen. Myös R:ään on kehitetty visuaalinen käyttöliittymä R Studio, mutta lähestymiskulma on kuitenkin teknisempi, kuin SAS:in kohdalla. Tällöin käyttäjien, joilla ei teknistä taustaa ole, voi olla hankalampi oppia käyttämään R:ää. Tämä voi vaatia enemmän koulutusta ja tukea varsinkin alkuvaiheessa, joka puolestaan lisää työntekijöiden koulutuksesta aiheutuvia kustannuksia.

4.1.4 Integraatiot ja testaus

Kun yrityksessä otetaan uusi ohjelmisto käyttöön, se on integroitava yhteensopivaksi muiden ohjelmistojen ja tietojärjestelmien kanssa. Uusien ohjelmistojen rajapinnat sopivat hyvin harvoin yhteen yrityksien vanhojen ohjelmistojen rajapintoihin (Yale 2001). Data-analytiikkaohjelmiston tapauksessa ohjelmistoon tuodaan dataa monesta eri lähteistä, jolloin datan tulee olla oikeassa muodossa, jotta ohjelmisto pystyy käyttämään dataa hyödykseen. Integrointi toisiin järjestelmiin tapahtuu ohjelmiston rajapintojen kautta. Uusi ohjelmisto voi sisältää valmiita rajapintoja, joita kannattaa integroinnissa käyttää hyödyksi. Jos rajapintoja ei ole valmiina, tai ne eivät ole yhteensopivia vanhojen järjestelmien kanssa, on rajapinnat suunniteltava ja kehitettävä ennen kuin ohjelmisto on käyttövalmis. Rajapinnat ovat harvoin täysin yhteensopivia vanhojen järjestelmien kanssa (Yale 2001), joten integraatio voi olla työmäärältään suuri osa koko ohjelmiston käyttöönottoprojektia. Rajapintojen luominen voi aiheuttaa suuriakin kustannuksia yritykselle, varsinkin jos rajapinnat täytyy luoda alusta loppuun. Jos joudutaan käyttämään ulkopuolista konsultointia, kustannukset nousevat edelleen. On sanottu, että ohjelmiston kustomointi kaikin tavoin nostaa ohjelmiston käyttöönottoprojektin kustannuksia (Raths 2013).

Data-analytiikkaohjelmistojen kohdalla integraatioiden merkitys korostuu, sillä dataa kerätään yrityksissä monesta eri lähteestä ja monessa eri muodossa. Käyttöönotossa kannattaa huomioida myös mahdolliset tulevaisuuden tarpeet, jotta uusia ohjelmistoja hankkiessa rajapinnat data-analytiikkaohjelmistoon olisivat jo valmiina etukäteen. Joissakin tilanteissa

dataa joudutaan siirtämään manuaalisesti järjestelmästä toiseen, mutta tämä nostaa ohjelmistosta aiheutuvaa kokonaiskustannusta. Ideaalitilanne olisi, että datan siirtäminen ja muuntaminen oikeaan muotoon tapahtuisi täysin automaattisesti, eikä minkäänlaista manuaalista työtä tarvittaisiin. Rajanpintoja kehitettäessä täytyy siis huomioida sekä datan siirtäminen että muuntaminen.

Nagy, Yassin ja Bhattacherjee kertovat artikkelissaan (2010, s. 150), että suljetun ja avoimen lähdekoodin ohjelmistojen integrointi vanhoihin ohjelmistoihin on hankalaa. Heidän mukaansa tällaisissa tilanteissa yritykset vaikuttavat karttavan avoimen lähdekoodin ratkaisuja. Rajapintoja on kuitenkin tarjolla ja artikkelin mukaan helpommin rajapintoja on saatavilla avoimen lähdekoodin ohjelmistoihin. He korostavat avoimen lähdekoodin kohdalla ulkopuolisen konsultoinnin mahdollisuuksia; vaikka ohjelmisto pohjautuu avoimeen lähdekoodiin, konsultointia on saatavilla ja kaikkea ei tarvitse tehdä itse. Tämän perusteella avoimen lähdekoodin ohjelmistot ovat helpompia ja sitä kautta myös kustannuksiltaan alhaisempia integroida vanhoihin järjestelmiin, kuin suljetun lähdekoodin ohjelmistot. Data-analytiikan kohdalla tämä korostuu entisestään rajapintojen kriittisyyden vuoksi.

Rajapintojen luomisen jälkeen niiden toimivuus tulee testata. Testaamisessa varmistetaan rajapintojen toimivuus ja se, että ohjelmisto toimii moitteettomasti uudessa ympäristössä (Rajlich & Bennett 2000, s. 66). Jos tässä vaiheessa ilmenee ongelmia, ne on helpompi korjata ennen kuin ohjelmistoa käytetään koko yrityksen laajuisesti. Testauksessa tulee kiinnittää huomiota myös suorituskykyasioihin ja varmistua myös siitä, että laitteet toimivat oikein ohjelmiston kanssa. Testausvaiheen jälkeen mahdolliset virheet korjataan, jonka jälkeen testataan uudelleen. Tätä iteraatiota jatketaan, kunnes virheet on korjattu ja ohjelmisto on valmis käytettäväksi koko yrityksen laajuisesti. Testausvaiheessa on hyvä pitää loppukäyttäjät jollakin tasolla mukana, jotta yleisimpien käyttötapausten ohjelmistovirheet tulevat korjatuiksi (Leonard-Barton & Kraus 1985).

4.1.5 Laitteisto ja pilvipalvelut

Siinä vaiheessa, kun uusi ohjelmisto otetaan käyttöön, on syytä pohtia myös IT-laitteiston tilannetta. Uusi ohjelmisto voi vaatia laitteistolta enemmän tehoa ja kaikkia ohjelmiston ominaisuuksia ei välttämättä saada täysipainoisesti käyttöön, jos laitteiston kapasiteetti on

riittämätön. Näiden syiden vuoksi laitteistoa tulisi tarkastella ohjelmistohankinnan yhteydessä. Data-analytiikkaohjelmistojen kohdalla tärkeitä ovat laskentateho, sekä tietokannat. Data-analytiikkaohjelmistot vaativat verrattain paljon laskentatehoa, riippuen käsiteltävän datan laadusta ja määrästä.

Yleisesti ottaen yritykset pyrkivät käyttämään jatkuvasti vähemmän rahaa laitteistoon. Sen sijaan, että yritykset ostavat uutta laitteistoa, he haluavat ohjelmistoja, jotka vähentävät kustannuksia, parantavat turvallisuutta ja integroituvat helposti jo olemassa oleviin ohjelmistoihin. (Economist 2002, s. 58-59). Toisin sanoen yritykset ostavat uutta laitteistoa vain, jos on pakko. Näin ollen valitun ohjelmiston käyttämien algoritmien tehokkuus ja rajapintojen integraatioprosessi vaikuttavat paljon yrityksen päätökseen valitessa uutta ohjelmistoa käytettäväksi. Toisaalta data-analytiikkaohjelmistojen tapauksessa kerätyn ja prosessoidun datan varastointiin vaaditaan valtavat määrät kovalevymuistia, jota yrityksillä ei usein valmiiksi ole.

Pilvipalveluiden suosio on jatkuvasti kasvussa. Pilvipalvelut antavat mahdollisuuden varastoida dataa toisen yrityksen tarjoamassa palvelussa, jossa kerätty data tallennetaan toisen yrityksen kovalevyille. Pilvipalvelut mahdollistavat tiedon jakamisen ja saatavuuden sijainnista huolimatta ja helpon skaalattavuuden. (Bojanova 2013, s. 12-13). Pilvipalveluiden käyttöönotto on helppo prosessi, ja yritys voi ostaa lisää tallennustilaa tarpeen mukaan. Näin ollen yrityksen ei tarvitse ostaa itse kovalevyjä ja ylläpitää suurta datan määrää, vaan sen hoitaa ulkoinen toimija. Data-analytiikan kohdalla tämä on erittäin käytännöllinen palvelu, koska datan määrä kasvaa usein hyvin nopeasti, kun dataa aloitetaan keräämään useammasta lähteestä. Pilvipalveluihin usein liittyy myös mahdollisuus ulkoistaa datan prosessointi.

Pilvipalveluiden tarjoama datan prosessointipalvelu mahdollistaa datan prosessoinnin ja tallentamisen samassa lähteessä. Sen ydinidea on käyttää mahdollisimman suuria prosessointiresursseja datan rinnakkaiseen analysoimiseen nopeuttamaan prosessia. (Chen, Mao, Zhang & Leung 2014, s. 12). Pilvipalveluiden yhdistäminen data-analytiikkaohjelmiston prosessointipalveluihin varmistaa, että yrityksellä on tarpeeksi suuri prosessointiteho datan analysoimiseen. Yrityksen ei ole pakko ostaa lisää fyysistä prosessointitehoa, vaan se voi ostaa ulkoisen toimijan tarjoamaa prosessointitehoa. Pilvipalveluissa datan tallentamisen ja prosessoinnin yhdistäminen yleistyy jatkuvasti, ja tulevaisuudessa pilvipalvelujen tarjoajat

panostavat entistä enemmän turvallisuuteen, datan visualisointiin ja rinnakkaisprosessoinnin optimointiin (Bojanova 2013, s. 13).

Yrityksen on kaikista kustannustehokkainta ottaa käyttöön pilvipalvelut data-analytiikkaohjelmiston rinnalle, jos sillä ei ole valmiiksi jo tarpeeksi prosessointitehoa tai tallennustilaa datan käsittelemistä varten. Pilvipalvelut ovat halvempi vaihtoehto prosessointitehon ja datan varastointikapasiteetin lisäämiseen. Avoimen lähdekoodin ohjelmistojen etuna on, että se saattaa olla hieman helpompi integroida pilvipalveluihin. Toisaalta monet suljetun lähdekoodin ohjelmistot kuten SAS tarjoavat jo valmiin pilvipalvelun lisämaksusta. Näin ollen avoimen lähdekoodin ja suljetun lähdekoodin ohjelmistojen välillä ei ole suurta kustannuseroa laitteiston näkökulmasta.

4.2 Ylläpito ja tuki

Ohjelmistojen tarve ylläpidolle kasvaa jatkuvasti, mikä taas aiheuttaa lisäkustannuksia yrityksille. Tällä hetkellä yli 50% kaikista asiakkaista haluaa uusia toiminnallisuuksia heillä jo käytössä oleviin ohjelmistoihin sen sijaan, että he ottaisivat käyttöön uusien tarpeiden mukaisia täysin uusia ohjelmistoja. Lisäksi suuret maailmanlaajuiset muutokset kuten Euroopan Unionin yhteisen rahavaluutan euron muodostaminen aiheutti päivittämisen tarpeen noin 85% ohjelmistoista. (Jones 2006, s. 2-3). Ylläpito- ja tukimaksuja maksetaan yrityksille palkkiona ohjelmistojen ylläpidosta. Ylläpito- ja tukikustannuksista veloitetaan vuosittain yleensä 20-25% ohjelmiston lisenssimaksusta. Näin ollen ylläpito- ja tukikustannuksista muodostuu ajan myötä kaikista suurin kulu uuden ohjelmiston käyttöönotossa. (Activestate 2016, s. 2; Laukkanen 2013). Yritysten on siis panostettava entistä enemmän ohjelmistojen ylläpitoon, joka taas voi aiheuttaa huomattavia työvoimakustannuksia, sekä suljetun lähdekoodin ohjelmistoille päivityskustannuksia.

Kasvanut ylläpidon tarve on muokannut ohjelmistoyritysten rakennetta huomattavasti. Yrityksissä on yhä enemmän ohjelmistojen ylläpitoon erikoistuneita työntekijöitä verrattuna ohjelmiston kehittäjiin. Ylläpitoon erikoistuneet työntekijät eroavat esimerkiksi kehittäjistä siten, että he eivät kehitä uusia ohjelmistoja, vaan he korjaavat ja lisäävät toiminnallisuuksia vanhoihin ohjelmistoihin. Ylläpidolla tarkoitetaan yleensä sekä ohjelmistojen toiminnallisuuksien tai osien lisäämistä että ohjelmistovirheiden korjaamista. Ylläpidon tarve

on kasvanut pääosin, koska vanhojen ohjelmistojen päivittäminen on erittäin aikaa vievää sekä työllistävää. Ohjelmiston ylläpitovaihe on yleisesti hyväksytty ohjelmiston kaikista kalleimmaksi vaiheeksi sekä usein myös pisimmäksi vaiheeksi. (Jones 2006, s. 3-4; Ahmed 2006, s. 450).

Ajan myötä, kun ohjelmistoon on tehty monia pieniä korjauksia sekä lisätty ominaisuuksia, ohjelmiston laatu sekä arkkitehtuuri kärsivät huomattavasti. Ohjelmiston laadun kärsimisen myötä ohjelmistovirheitä ilmenee jatkuvasti enemmän, ja niiden korjaaminen aiheuttaa huomattavia ylläpitokustannuksia (Jones 2006, s. 13). Tämä aiheuttaa erityisen paljon kustannuksia suljetun lähdekoodin ohjelmistoissa, koska yritysten on jatkuvasti käytettävä konsultointipalveluita, kun taas avoimen lähdekoodin tapauksessa yrityksen sisäinen asiantuntija voi päivittää ja muokata ohjelmistoa. Lisäksi esimerkiksi rajapintojen muuttuessa avoimen lähdekoodin ohjelmistoihin voidaan usein asentaa yhteisön tekemiä integraatioratkaisuja, kun taas suljetun lähdekoodin ohjelmistoissa on jälleen kerran luotettava ulkopuoliseen konsultointiin.

Yksi avoimen lähdekoodin eduista on sen ympärille rakentuneen yhteisön tuki. Ohjelmistojen ylläpitokustannukset voivat laskea huomattavasti, koska ohjelmistojen ylläpitäjät voivat pyytää apua yhteisöltä ongelman ratkaisemiseksi. Avoimen lähdekoodin arkkitehtuuri ja lähdekoodi ovat kaikkien nähtävissä, jolloin yrityksen on myös helpompi arvioida sen toiminnallisuuksia ja integraatiomahdollisuuksia tehokkaammin kuin suljetun lähdekoodin tapauksessa. Lisäksi esimerkiksi ohjelmistojen päivitykset ilmestyvät keskimäärin kaksi kertaa nopeammin avoimen lähdekoodin ohjelmistoille kuin suljetun lähdekoodin ohjelmistoille (Hoepman & Jacobs 2007, s. 83). Näin ollen ohjelmistojen potentiaalinen häiriöaika, jolloin ohjelmisto ei ole käytössä, pienenee.

Kaikista tehokkaimman ylläpitotuloksen saa, kun yritys käyttää ylläpitoon erikoistuneita asiantuntijoita. Monessa yrityksessä kehittäjät myös ylläpitävät ohjelmistoja. Tämä voi olla huono kustannusnäkökulmasta, sillä asiantuntijat voivat olla jopa kaksi kertaa tehokkaampia kuin kehittäjät. Koska kaikki voivat käyttää avoimen lähdekoodin ohjelmistoa, asiantuntijoita ja osaajia on potentiaalisesti enemmän. Potentiaalisia osaajia on helpompi löytää ja palkata, ja yritys ei ole välttämättä sitoutunut vain yhden yrityksen asiantuntijuuteen. (Jones 2006, s. 11; Ahmed 2006, s. 449). Lisäksi yritys voi käyttää yhteisön tekemiä valmiita ratkaisuja ja

lisätoiminnallisuuksia, jolloin yritys säästyy ylimääräiseltä ohjelmoinnilta. Toisaalta, jos yrityksellä ei ole omia ylläpitoon erikoistuneita asiantuntijoita, ulkoistamalla osan palveluista voi saada aikaan huomattavia kustannussäästöjä.

Monet yritykset käyttävät ulkoisia palveluita ohjelmistokehityksen eri vaiheessa. Esimerkiksi yli 70% Fortune 500 yrityksistä ulkoisti ainakin osan IT-toiminnoistaan. Ulkoistamisen yleisiä etuja ovat työvoiman lisäsaanti, erikoisosaaminen sekä alhaiset hinnat. Ulkoistamisen myötä yritys voi keskittyä pääliiketoimintaansa, ja potentiaalisesti säästää aikaa sekä rahaa. (Ahmed 2006, s. 449-453). Data-analytiikkaohjelmistojen tapauksessa esimerkiksi ohjelmistojen välinen integraatio ja lisätoiminnallisuuksien kehittäminen voitaisiin ulkoistaa asiantuntevalle yritykselle, ja itse datan kerääminen ja jalostaminen suoritetaan yrityksen sisäisesti. Toisaalta data-analytiikkaohjelmistojen ylläpitopalveluiden ulkoistaminen saattaa aiheuttaa moraalisia ongelmia, jos ylläpitoa suorittava yritys voi päästä kerättyyn dataan käsiksi.

4.3 Koodin laatu ja tietoturva

Avoimien ohjelmistojen suosion kasvaessa, ajankohtaiseksi tulevia kysymyksiä ovat: ”ovatko avoimen lähdekoodin ohjelmistot laadullisesti verrattavissa suljetun lähdekoodin ohjelmistoihin?” ja ”voiko avoimen lähdekoodin tietoturvaan luottaa, jos sen turvajärjestelmät ovat kaikkien nähtävissä?” Avoimen lähdekoodin ohjelmistoille ei aina laadita sääntöjä, jotka määrittelevät ketkä valvovat ohjelmointiprosessia tai kuka saa luoda ja muokata ohjelmistojen ominaisuuksia (Schryren 2011, s. 132). Tämä aiheuttaa ymmärrettäviä huolen aiheita avoimen lähdekoodin käyttäjille, erityisesti jos he käsittelevät yksityishenkilöiden dataa, kuten data-analytiikkayritykset usein tekevät. Tässä osiossa vertaillaan avointa lähdekoodia ja suljettua lähdekoodia koodin laadun, tietoturvan, ja niistä aiheutuvien kustannusten näkökulmasta.

4.3.1 Koodin laatu

Yleisesti avoimen lähdekoodin ongelmana on, että ohjelmistoilla on paljon käyttäjiä jotka haluavat käyttää ohjelmistoa, mutta eivät halua olla mukana kehittämässä ohjelmistoa. Kyseiset käyttäjät ovat niin sanottuja ”free rider” -käyttäjiä, joiden kontribuutio ohjelmistolle on käytännössä olematon. Jos ohjelmiston käyttäjät eivät ole kiinnostuneita parantamaan ohjelmiston ominaisuuksia sekä laatua, sen käyttö lopetetaan aikanaan. Avoimen lähdekoodin

ohjelmistot vaativat vankan käyttäjäpohjan, jolloin ohjelmiston suosion noustessa, myös potentiaalisten tukijoiden määrä kasvaa. (Raghunathan et al. 2005, s. 903). Suljetun lähdekoodin ohjelmistojen etuna on, että koodia parannetaan ja virheitä korjataan jatkuvasti, koska ohjelmistolla on oltava luotettava tukipalvelu. Näin ollen suljetun lähdekoodin ohjelmiston käyttäjät voivat maksua vastaan varmistaa, että ohjelmistovirheet korjataan aikanaan, ja ohjelmistoa tuetaan myös tulevaisuudessa.

Koodin laadun tutkimiseen on monta eri menetelmää. Yksi näistä on esimerkiksi virheiden summan vertaaminen joka tuhanteen riviin kirjoitettua koodia. Vuonna 2014 tehdyn Coverity:n tutkimuksen mukaan vuonna 2013 avoimen lähdekoodin ohjelmistojen laatu meni suljetun lähdekoodin edelle ensimmäistä kertaa. Toisin sanoen avoimessa lähdekoodissa löydettiin keskimäärin vähemmän virheitä verrattuna kirjoitetun koodin määrään, ja ohjelmistovirheet korjattiin nopeammin.

Eräs selitys suljetun lähdekoodin ohjelmistovirheiden suhteelliselle paljoudelle voi olla yritysten mentaliteetti, jossa uhrataan koodin laatua, jotta ohjelmisto saadaan julkaistua tavoiteaikaan mennessä. Yrityksiä painostetaan saamaan toimiva versio ohjelmistosta asiakkaille käytettäväksi ja korjaamaan ohjelmistovirheet myöhemmin. (Raghunathan et al. 2005, s. 905). Avoimen lähdekoodin tapauksessa käyttäjät voivat ilmoittaa ohjelmistovirheistä tehokkaammin kuin suljetussa lähdekoodissa. Käyttäjät voivat esimerkiksi kertoa tarkalleen, missä kohtaa lähdekoodia virhe on, tai missä käyttötapauksissa virhe tapahtuu (Hoepman & Jacobs 2007, s. 82). Lisäksi korjaajia on useampi, jolloin ratkaisuja ongelmiin tulee määrällisesti enemmän ja nopeammin. Toisaalta suljetun lähdekoodin ohjelmistojen etuna on, että virheiden korjaajat ovat aina alan ammattilaisia. Näin ollen korjaukset ja uudet ratkaisut ovat usein tarkasti harkittuja kokonaisuutta ajatellen, jolloin ne aiheuttavat harvemmin uusia virheitä ohjelmistossa, kuten esimerkiksi integraatio-ongelmia.

Avoimessa lähdekoodissa sitä ylläpitävä yhteisö on keskiössä. Tekijät haluavat tunnustusta teoistaan yhteisön sisällä, ja osoittaa osaamistaan yrityksille. (Raghunathan et al. 2005, s. 903). Tämä taas saa yhteisön jäsenet kirjoittamaan laadukasta koodia, jotta heidän uskottavuutensa ja kunnioituksensa muiden käyttäjien kesken pysyy yllä. Ilmiö saa yhteisön jäsenet tekemään enemmän taustatutkimusta, joka taas saa aikaan ohjelmistolle uusia paranneltuja työkaluja ohjelmistojen kehitystä, testausta ja arvostelua varten. (Hoepman & Jacobs 2007, s. 83).

Suljetun lähdekoodin ongelmana on, että ratkaisut ovat usein yhtenäisiä ja jäykkiä (Neumann 1999, s. 128). Koska suljetulla lähdekoodilla ei ole yhteisöä arvioimassa koodin laatua, samat virheet usein toistuvat ja uudet innovatiiviset ratkaisut jäävät usein kokonaan käyttämättä. Tämä taas huonontaa koodin laatua, ja saattaa esimerkiksi hidastaa ohjelmiston toimintaa. Ohjelmiston toiminta on erityisen tärkeää data-analytiikkaohjelmistoille, koska ne käsittelevät suuria määriä dataa lyhyessä ajassa. Tällöin ohjelmiston algoritmien on oltava tehokkaita, jotta sitä käyttävät tietokoneet voivat prosessoida ja käsitellä datan mahdollisimman vähillä resursseilla.

Ohjelmistovirheet voivat olla yrityksille kalliita. Yrityksille aiheutuu virheistä usein häiriöaika, jolloin ohjelmistoa ei voida käyttää. Tämä aiheuttaa kustannuksia esimerkiksi estämällä työntekijöitä tekemästä työtään, ja myöhästyttämällä projekteja määräajasta (Raghunathan et al. 2005, s. 906). Data-analytiikan kannalta ohjelmistovirheet voivat aiheuttaa suuria ongelmia esimerkiksi korruptoimalla tietokannan, jolloin yritys voi menettää useamman vuoden ajalta kerätyn datan. Avoimen lähdekoodin tapauksessa virheistä vastuussa on käyttäjä, kun taas suljetun lähdekoodin tapauksessa ohjelmiston myyjä on usein vastuussa aiheutuneista vahingoista. R on erittäin suosittu avoimen lähdekoodin ohjelmisto, jolla on hyvin paljon käyttäjiä, joten todennäköistä on, että kaikki toimii oikein. Yritys ei kuitenkaan voi siirtää virheistä aiheutuvaa riskiään avoimen lähdekoodin tapauksessa, vaan se joutuu kantamaan kyseisen riskin itse. Suljetun lähdekoodin tapauksessa käyttäjä on suojattu ainakin osittain aiheutuneista vahingoista. Toisaalta, koska avoimen lähdekoodin ohjelmistoissa on keskimäärin vähemmän ohjelmistovirheitä, yritykset voivat käyttää ohjelmistoja esimerkiksi korvaamattoman datan varastointiin ja jalostamiseen suuremmalla luottamuksella.

4.3.2 Tietoturva

Tietoturvan arvo ohjelmistoissa kasvaa jatkuvasti. Datan määrä kasvaessa, tietovuodot ovat yhä suurempi riskitekijä, ja potentiaaliset ekonomiset vahingot voivat olla tuhoisia. Erityisesti data-analytiikkaohjelmistojen tapauksessa, joissa käsitelty data voi paljastaa yksityishenkilöiden luonteesta ja kiinnostuksista arkaluontoista tietoa, tietoturvan on oltava taattu.

Yksi suljetun lähdekoodin suurimmista eduista on sen koodin salassapito. Hyökkääjän on vaikeampi etsiä ohjelmistosta tietoa, koska järjestelmän lähdekoodi sekä arkkitehtuuri ovat

salaisia. Avoimen lähdekoodin tapauksessa hyökkääjät voivat etsiä ohjelmistosta sekä rakenteellisia heikkouksia että koodauksen heikkouksia ja käyttää näitä hyväksi. Toisaalta tapauksissa, joissa suljetun lähdekoodin ohjelmistojen rakenne tai koodi on vuotanut kaikille nähtäväksi, on usein löydetty paljon tietoturva-aukkoja. Lisäksi vaikka ohjelmistot ovat suljettuja, se ei estä hyökkääjiä käyttämästä esimerkiksi virheenjäljittäjiä heikkouksien löytämiseksi. (Hoepman & Jacobs 2007, s. 79-82). Tämä tarkoittaa sitä, että sekä avoimen lähdekoodin ohjelmistot että suljetun lähdekoodin ohjelmistot ovat haavoittuvaisia tietoturvauhkille. Avoimen lähdekoodin tapauksessa kaikki voivat arvioida tietoturvan toimivuutta, kun taas suljetun lähdekoodin ohjelmistossa tietoturva-aukot eivät ole kaikille nähtävissä. Toisaalta tietoturva-aukot löydetään myös suljetuista ohjelmistoista melkein aina ajan myötä.

Avoimen lähdekoodin projektit perustuvat luottamukseen kehittäjiin. Koska projekteissa on harvoin valintaprosessi, myös koodin tietoturvasta herää kysymyksiä. Koodin laatu saattaa olla huonoa, tai koodissa saattaa olla pahoja rakenteellisia virheitä, joka saattaa asettaa ohjelmiston alttiiksi tietoturva-aukoille (Schryren 2011, s. 140). Suljetun lähdekoodin kehittäjät ovat usein tunnettuja ja luotettavia, jolloin voidaan olettaa kehittäjien huomioivan ohjelmiston turvallisuuden jo suunnitteluvaiheessa.

Avoimessa lähdekoodissa kehittäjien taidon lisäksi kyseenalaiseksi tulevat heidän motiivinsa kehittää lisää toiminnallisuuksia tai korjata vanhoja. He voivat piilottaa halutessaan vaarattoman näköisiin ohjelmiin takaovia, joiden avulla he voivat päästä käsiksi esimerkiksi käyttäjien tietoihin. (Hoepman & Jacobs 2007, s. 81-82). Huonolaatuinen koodi tai piilotetut tietoturva-aukot saattavat aiheuttaa huomattavaa vahinkoa yrityksille. Esimerkiksi hyökkääjä voi käyttää kaikkien kyseistä ohjelmistoa käyttävien yritysten keräämää käyttäjien dataa tai analyysin tuloksia kiristyksen kohteena, tai myydä sitä eteenpäin kolmannelle osapuolelle. Tällöin yritys voi joutua maksamaan asiakkailleen tietovuodosta aiheutuneita vahingonkorvauksia.

Vaikka avoin lähdekoodi aiheuttaa tiettyjä tietoturvariskejä, sen myötä tulee myös monta mahdollisuutta. Koska lähdekoodi on avointa, kaikki käyttäjät voivat etsiä ohjelmistosta

virheitä ja ilmoittaa niistä tehokkaasti. Näin ollen kehittäjät tai virheen löytäjät voivat korjata aukon mahdollisimman nopeasti, jonka seurauksena avoimen lähdekoodin ohjelmistojen virheet korjataan keskimäärin huomattavasti nopeammin (Schryen 2011, s. 140). Koska määrätietoinen hyökkääjä löytää melkein aina heikkouden ohjelmistosta, myös suljetun lähdekoodin ohjelmistojen heikkouksia voidaan käyttää hyväksi. Tästä johtuen moni tietoturva-aukko suljetuissa ohjelmistoissa ei tule julkisuuteen koskaan, vaikka niitä löytyisikin, koska sitä hyväksi käyttävät haluavat pitää sen salassa. (Hoepman & Jacobs 2007, s. 82-83). Näin ollen avoimen lähdekoodin ohjelmistot ovat suhteessa turvallisimpia, koska tietoturva-aukot korjataan nopeammin yhteisön toiminnasta. Lisäksi yritys ei joudu maksamaan päivityksistä, jolloin he myös säästävät kustannuksissa.

Käyttäjillä on mahdollisuus lisätä tai poistaa ominaisuuksia avoimen lähdekoodin ohjelmistoissa. Käyttäjien on helppo lisätä valmiita lisäturvajärjestelmiä ohjelmistoihin, tai he voivat poistaa ohjelmistosta ylimääräisiä osia, jonka myötä myös tietoturva-aukkojen määrä vähenee. (Hoepman & Jacobs 2007, s. 83). Avoimen lähdekoodin käyttäjät voivat siis parantaa ohjelmistonsa tietoturvallisuutta lisäämällä siihen osia, jotka ovat usein yhteisön tekemiä ja ilmaisia. Suljetun lähdekoodin jäykkyys voi aiheuttaa käyttäjille tietoturvaongelmia, koska käyttäjä ei voi muokata tai parantaa ohjelmiston tietoturvaominaisuuksia.

Avoimen lähdekoodin tietoturva aiheuttaa edelleenkin kysymyksiä käyttäjien kesken. Järjestelmän lähdekoodi ja arkkitehtuuri ovat kaikille nähtävissä, jolloin kuka vain voi hyökätä milloin ja mihin vain järjestelmän osaan. Toisaalta, koska käyttäjä voi nähdä koko järjestelmän toiminnan, hän voi itse arvioida sen toimivuuden ja luotettavuuden, toisin kuin suljetun lähdekoodin tapauksessa, jolloin hänen on vain luotettava yrityksen tarjoamaan tietoturvaan. Alla olevaan taulukkoon on koottu avoimen ja suljetun lähdekoodin vahvuudet ja heikkoudet koodin laadun ja tietoturvan näkökulmasta.

Taulukko 2: *Avoin ja suljettu lähdekoodi koodin laadun ja tietoturvan näkökulmasta*

	Avoin lähdekoodi	Suljettu lähdekoodi
Puolesta (+)	*Yhteisö voi löytää ja korjata virheitä *Kaikki näkevät lähdekoodin ja ohjelmiston arkkitehtuurin *Innovatiiviset ratkaisut *Koodin laatu	*Lähdekoodi salainen *Korjaukset tekevät luotettavat ammattilaiset *Ohjelmistovirheet korjataan ajan myötä *Vastuu ohjelmistoyrityksellä
Vastaan (-)	*Lähdekoodi kaikille nähtävissä *Luottamus kehittäjiin *Vaatii uskollisen käyttäjäkunnan	*Lähdekoodin pitäminen salassa hankalaa *Yrityksen mentaliteetti *Vain kehittäjät korjaamassa ja löytämässä virheitä

4.4 Lisenssiehdot ja -kustannukset

Suljetun lähdekoodin ohjelmistot on suojattu aina ohjelmistoa tuottavan yrityksen omilla lisensseillä. Käyttäjän tulee hyväksyä nämä käyttöehdot ennen kuin ohjelmiston käyttäminen on mahdollista. Käyttöehdot voivat rajoittaa esimerkiksi ohjelmiston käyttötarkoitusta ja uudelleen jakelua. Niin kuin luvussa 1 mainittiin, myös avoimen lähdekoodin ohjelmistoilla on jonkinlainen lisenssi, joka määrittää kyseisen ohjelmiston käyttöehdot. David Wormser (2010) kritisoi artikkelissaan avoimen lähdekoodin ohjelmistojen lisenssejä muun muassa tulkinnanvaraisuudesta ja epäselvistä ehdoista. Lisäksi erilaisten lisenssien suuri lukumäärä aiheuttaa ongelmia kehittäjille ja käyttäjille.

Suljetun lähdekoodin ohjelmistoissa lisenssit ovat yleensä selkeitä ja hyvin ymmärrettävissä (Wormser 2010, s. 23). Suljetun lähdekoodin ohjelmistoissa lisenssien määrittämät käyttöehdot ovat hyvin tarkkaan määriteltyjä ja ehdot ovat verrattain tiukkoja. Lisenssi rajoittaa yleensä esimerkiksi ohjelmiston käyttäjien tai päätelaitteiden määrää. Tällöin yrityksissä tulee tarkkaan harkita, keille kaikille henkilöille ohjelmiston käyttöoikeudet kannattaa sallia. Data-analytiikkaohjelmistoa voi käyttää yrityksessä useampi henkilö ja vielä useampi saattaa tarvita ohjelmistoa esimerkiksi raporttien tarkasteluun. Kustannusnäkökulmasta tällöin tulee miettiä, keille ohjelmiston käyttöoikeus on oikeasti tarpeellinen; voidaanko esimerkiksi raportointi hoitaa jollakin toisella tavalla? Kustannusten puolesta olisi parasta, että varsinaista ohjelmistolisenssiä maksetaan vain niiden käyttäjien osalta, joiden työnkuvan kannalta ohjelmiston käyttö on todella tarpeellista.

Avoimen lähdekoodin ohjelmistojen lisenssit ovat lähes poikkeuksetta ilmaisia, jolloin kustannusnäkökulmasta käyttäjämäärä ei ole rajoittava tekijä. Avoimen lähdekoodin ohjelmistoissa lisenssikustannuksia ei synny, jos kaikki lisenssiehtoja noudatetaan tarkasti. Tämä voi olla usein hankalaa tulkinnanvaraisuuden ja epäselvyyden vuoksi. Kustannuksia avoimen lähdekoodin ohjelmistoissa voi syntyä lisenssirikkomuksista ja mahdollisista oikeustoimista. Lisenssiasioissa tulee olla erityisen tarkkana, jos yritys myy jonkinlaista avoimen lähdekoodin ratkaisua (esimerkiksi kehittää lisäosia tai kirjastoja R:ään) tai haluaa käyttää jotakin avoimen lähdekoodin komponenttia osana suljetun lähdekoodin ohjelmistoaan.

Avoimen lähdekoodin lisenssien kanssa toimiessa lisenssiehdot kannattaa ottaa tarkasti huomioon. Isojakin yrityksiä, kuten Microsoftia, Samsungia ja JVC:tä, ovat syytetty lisenssiehtojen rikkomisesta (Wormser 2009, s. 22). Avoimen lähdekoodin lisensseissä on myös yhteensopivuusongelmia; kaikki avoimen lähdekoodin lisenssit eivät ole yhteensopivia toistensa kanssa. Tämä johtuu lisenssiehtojen erilaisuudesta ja varsinkin siitä, että osa lisensseistä on vastavuoroisuutta vaativia ja osa niin sanottuja sallivia lisenssejä. Yhteensopivuusongelmat ja vastavuoroisuutta vaativien lisenssien erikoispiirteet voivat tulla tärkeäksi siinä vaiheessa, kun suljetun lähdekoodin ohjelmistoon tai avoimen lähdekoodin data-analytiikkaohjelmistoon tilataan kolmannelta osapuolelta jotakin lisäosaa tai komponenttia. Tällöin myös tilaajan tulee varmistua siitä, että kolmas osapuoli on noudattanut tarkasti sekä alkuperäisen ohjelmiston että uuden komponentin lisenssiehtoja. Jos lisenssiehtoja ei ole noudatettu, vastuussa on ainakin komponentin toimittaja, mutta riski voi realisoitua myös tilaajalle. Kaiken kaikkiaan lisenssiehtoja tarkasti noudattamalla voidaan minimoida kaikki ylimääräiset kustannukset, joita ehtojen noudattamatta jättämisestä voisi syntyä. Riskin realisoituessa kustannuksia data-analytiikkaohjelmiston kohdalla aiheutuu ainakin oikeudenkäynnistä, vahingonkorvauksista, työajan menetyksestä, datan hyödyntämättömyydestä ja mahdollisesti uuden komponentin tilaamisesta.

Edellä mainittujen asioiden pohjalta voidaan sanoa, että avoimen lähdekoodin lisenssit aiheuttavat suurempia riskejä yrityksille, kuin suljetun lähdekoodin ohjelmistot. Riskit eivät välttämättä kaikkien yritysten kohdalla realisoidu, mutta riskin realisoituessa kustannukset voivat nousta kalliiden ja pitkien oikeusprosessien vuoksi hyvinkin korkeiksi. Suljetun lähdekoodin ohjelmistoissa tulee ottaa huomioon lisenssien käyttörajoitukset, jotka puolestaan voivat lisätä kustannuksia esimerkiksi rajoitettujen käyttäjämäärien vuoksi. Nämä kustannukset ovat huomattavasti helpommin hallittavissa, kuin avoimen lähdekoodin vastaavat. Yrityksen, joka on ottamassa avoimen lähdekoodin data-analytiikkaohjelmiston käyttöön, olisi syytä hankkia apua taholta, jolle avoimen lähdekoodin lisenssiasiat ovat tuttuja. Jos osaamista ei löydy yrityksen sisältä, lakimiehen palkkaaminen tai konsultointipalvelut ovat hyviä vaihtoehtoja. Tämä nostaa data-analytiikkaohjelmiston kustannuksia, mutta pienentää huomattavasti riskiä joutua ongelmiin lisenssien kanssa.

4.5 Ohjelmiston vaihto ja alasajo

Yleisesti ohjelmiston alasajo aiheuttaa huomattavia kustannuksia. Yritykset sijoittavat usein ohjelmistoihin valtavia määriä resursseja sekä varoja. Näin ollen yritykset eivät kovin mielellään vaihda ohjelmistosta toiseen, koska yritys mieltää ohjelmistoon sijoitetut varat hukkaan menneeksi investoinniksi. (Nagy et al. 2010, s. 150).

Yrityksen on otettava jo alkuvaiheessa huomioon työntekijöiden edeltävä kokemus käyttöönotettavasta ohjelmistosta. Suljetun lähdekoodin ohjelmistoja pidetään yleisesti tunnetumpina kuin avoimen lähdekoodin ohjelmistot, jolloin yritys voi säästää huomattavia summia koulutuskustannuksissa. Toisaalta avoimen lähdekoodin ohjelmistot ovat usein muilta elinkaarikustannuksiltaan halvempi vaihtoehto. (Vonfange & Lavigne 2011, s. 7*)*.

Elinkaarikustannusten laskeminen on erittäin oleellinen tekijä tutkiessa kannattaako vaihtaa suljetun lähdekoodin ohjelmistosta avoimen lähdekoodin ohjelmistoon. Vaikka vaihto suljetusta lähdekoodista avoimeen lähdekoodiin saattaa olla suuri investointi, avoimen lähdekoodin ohjelmisto voi tulla jo kohtuullisella aikavälillä halvemmaksi vaihtoehdoksi. Jos vaihto on kannattava yritykselle, se kannattaa tehdä, vaikka suljetun lähdekoodin ohjelmisto ei olisi vielä käyttöikänsä päässä. (Nagy et al. 2010, s. 151). Toisaalta on otettava huomioon myös ohjelmiston käyttötarkoitus. Jos esimerkiksi data-analytiikkaohjelmistoa käytetään vain yksinkertaisten analyysien tekemiseen, yrityksen ei usein kannata vaihtaa ohjelmistoa. Tällöin yritys ei usein vaadi erityistä konsultointia ohjelmiston tuottajalta ja ohjelmiston perusversiot riittävät, jolloin avoimen lähdekoodin ohjelmiston rakentamiseen vaatima konsultointi ja ylläpito voivat aiheuttaa huomattavia kustannuksia. Toisaalta, jos yritys vaatii harvinaisempia ratkaisuja ja jatkuvaa ohjelmiston kehittämistä, avoimen lähdekoodin ratkaisu on usein parempi pitkällä aikavälillä. Tällöin yritys voi joko kilpailuttaa konsultointipalveluita, tai palkata omia työntekijöitä kehittämään ohjelmistoa.

Eräs suljetun lähdekoodin ohjelmistojen piirteistä on sen toimintamalli, jossa asiakkaat pyritään tekemään riippuvaiseksi ohjelmistoon. Ohjelmistoa myyvä yritys voi julkaista uusia versioita samasta ohjelmistosta, ja veloittaa asiakkailta huomattavia summia pitkällä aikavälillä (Economist 2002, s. 54). Tästä syystä suljetun lähdekoodin ohjelmistojen tuottajat pyrkivät usein tekemään ohjelmistoistaan mahdollisimman hankalia vaihtaa toiseen (Sabhlok 2013). Jos

yritys päättää vaihtaa ohjelmiston suljetusta lähdekoodista avoimeen lähdekoodiin, he pääsevät eroon myös tästä riippuvuudesta. Toisaalta, vaikka yritys ei ole enää riippuvainen ohjelmiston tuottajasta, he ovat silti riippuvaisia valitsemastaan ohjelmistosta (Caulkins, Feichtinger, Grass, Hartl, Kort & Seidl 2013, s. 1183). Yrityksen on kuitenkin huomattavasti helpompi vaihtaa avoimen lähdekoodin ohjelmisto toiseen ohjelmistoon, koska avoimen lähdekoodin ohjelmistoihin ei yleensä rakenneta esteitä, jotka vaikeuttaisivat ohjelmiston vaihtoa (Sabhlok 2013). Data-analytiikkaohjelmistoissa avoimen lähdekoodin ratkaisut ovat erittäin hyviä tulevaisuutta ajatellen, sillä avoimen lähdekoodin helpottavat yrityksen toimintaa tulevaisuudessa. Avoimen lähdekoodin ratkaisuissa ohjelmistojen integraatiota muihin data-analytiikkaan vaatimiin ohjelmistoihin helpotetaan jatkuvasti, joten yritys voi vaihtaa toiseen avoimen lähdekoodin ratkaisuun kohtuullisen vähällä työllä. Näin ollen pidemmällä aikavälillä yritys voi säästää ohjelmistojen päivityskustannuksissa sekä integraation vaatimissa työvoimakustannuksissa huomattavia summia vaihtamalla suljetun lähdekoodin ohjelmistosta avoimen lähdekoodin ohjelmistoon.

5 JOHTOPÄÄTÖKSET

Työn tavoitteena oli vertailla avoimen ja suljetun lähdekoodin data-analytiikkaohjelmistoja kustannusnäkökulmasta. Työmme päätutkimuskysymys oli:

Kannattaako yrityksen ottaa käyttöönsä avoimen vai suljetun lähdekoodin data-analytiikkaohjelmisto?

Päätutkimuskysymys jakautui seuraaviin osatutkimuskysymyksiin:

1. Mistä komponenteista avoimen ja suljetun data-analytiikkaohjelmistojen kustannukset muodostuvat?
2. Mitä eroja avoimen ja suljetun lähdekoodin data-analytiikkaohjelmistoissa on?

Vastauksena päätutkimuskysymykseen voidaan sanoa, että yrityksen on järkevämpää ottaa avoimen lähdekoodin kuin suljetun lähdekoodin data-analytiikkaohjelmisto käyttöön. Taulukossa 3 on esiteltynä tärkeimmät avoimen ja suljetun lähdekoodin data-analytiikkaohjelmistojen ominaisuudet ja eroavaisuudet.

Taulukko 3 *Avoimen ja suljetun lähdekoodin data-analytiikkaohjelmistojen ominaisuudet ja eroavaisuudet*

	Avoin lähdekoodi	Suljettu lähdekoodi
Puolesta (+)	* Ei lisenssimaksua * Kilpailuttaminen * Työn ulkoistamismahdollisuudet * Helpompi integroida * Yhteisön tuki * Avoimen lähdekoodin ohjelmistojen tunnettuus * Päivitysnopeus * Lähdekoodi näkyvissä * Laadukkaampi koodi * Parempi tietoturva * Innovatiiviset ratkaisut	* Visuaalinen käyttöliittymä * Käyttöönottoprojektin läpiviennin varmuus * Käyttäjätuki * Ammattimainen ylläpito- ja tukipalvelu * Vastuu ongelmista ohjelmistoimittajalla * Selkeät lisenssiehdot * Vähemmän riskejä
Vastaan (-)	* Voiko yhteisöön luottaa täysin? * Vaatii uskollisen käyttäjäkunnan * Lähdekoodi näkyvissä * Ongelmat lisenssien kanssa	* Lisenssimaksut * Integraatiovaikeudet * "Naimisissa" yhden toimittajan kanssa * Ohjelmistotoimittajan suhtautuminen koodin laatuun * Enemmän virheitä koodissa * Ohjelmiston vaihtaminen * Jäykät ratkaisut

Työssä huomattiin avoimen lähdekoodin olevan parempi vaihtoehto ylläpidon näkökulmasta. Koska ohjelmistojen ylläpitokustannukset kasvavat jatkuvasti, avoimen lähdekoodin suurin etu on sen kilpailukyvykkyys. Yritykset voivat kilpailuttaa ulkopuolisia konsultteja ohjelmiston ylläpitoon, ja näin ollen saada kilpailukykyisimmän hinnan. Avoimen lähdekoodin ratkaisuissa yritys ei ole riippuvainen käytettävästä ohjelmistosta, vaan se voi tulevaisuudessa vaihtaa parempaan ohjelmistoon verrattain pienellä vaivalla.

Avoimen lähdekoodin ohjelmistojen yhteisöt korjaavat ja parantavat ohjelmistoja. Yhteisö korjaa ohjelmistovirheet nopeammin ja kehittää uusia ilmaisia toiminnallisuuksia useammin kuin suljetun lähdekoodin ohjelmistojen tuottajat. Avoimen lähdekoodin ohjelmistoja käyttävät yritykset voivat etsiä ratkaisuja ja pyytää yhteisöltä apua ylläpitoon ilmaiseksi, kun taas suljetun lähdekoodin ratkaisuissa kaikki lisätoiminallisuudet ja ylläpito ovat maksullisia. Toisaalta suljetun lähdekoodin ohjelmistot tarjoavat aina asiantuntevaa tukea ja asiakaspalvelua, kun taas avoimen lähdekoodin ratkaisuissa yritykset voivat joutua luottamaan ainoastaan yhteisön tukeen.

Data-analytiikkaohjelmisto on yhdistettävä jollakin tavalla yrityksen muihin tietojärjestelmiin. Integraatio voi olla hyvinkin suuri osa koko ohjelmiston käyttöönottoprojektin kustannusta. Integraatiot ovat yleensä helpompia tehdä avoimen lähdekoodin ohjelmistoihin. Varsinkin vanhojen järjestelmien integrointi on suljetun lähdekoodin ohjelmistoihin vaikeampaa kuin vastaaviin avoimen lähdekoodin ratkaisuihin. Avoimen lähdekoodin menestyminen ja helpompi integraatio on johtanut siihen, että myös suljetun lähdekoodin ohjelmistokehittäjät avaavat yhä enemmän rajapintoja ohjelmistoihinsa. Näin ollen integraatiot voivat tulevaisuudessa helpottua ja kolmannet osapuolet voivat kehittää ohjelmistoja, jotka toimivat yhteen suljetun lähdekoodin ohjelmistojen kanssa. Vielä tällä hetkellä avoimen lähdekoodin ohjelmistojen integraatio muihin järjestelmiin on yksinkertaisempaa ja sitä kautta edullisempaa kuin suljetun lähdekoodin ohjelmistoissa.

Data-analytiikkaohjelmistoissa koodin laatu sekä ratkaisujen tehokkuus ovat merkittäviä tekijöitä. Kyseiset ohjelmistot käsittelevät ja analysoivat valtavia datamääriä, jolloin ne vaativat paljon prosessointitehoa sekä muistia. Ohjelmistossa käytetyt ratkaisut vaikuttavat laitteistolta vaadittavaan prosessointitehoon sekä analysoitavan datan määrän ylärajaan. Yritys voi ostaa lisää prosessointitehoa sekä datan tallennustilaa uuden laitteiston tai pilvipalveluiden

muodossa, mutta tuottavan ohjelmiston käyttäminen on huomattavasti kustannustehokkaampaa.

Yleisesti avoimen lähdekoodin ohjelmistoissa koodin laatu on parempaa. Koodissa on vähemmän virheitä, ja ohjelmistojen ominaisuudet ovat innovatiivisempia. Avoimen lähdekoodin kehittäjiä on paljon, jolloin virheet koodissa korjataan nopeammin ja uusia teknisiä ratkaisuja kehitetään jatkuvasti. Data-analytiikkaohjelmistoissa tämä tarkoittaa sitä, että avoimen lähdekoodin ohjelmistot tarjoavat yleensä teknisesti parempia ratkaisuja sekä paljon valmiita ilmaisia toiminnallisuuksia. Toisaalta avoimen lähdekoodin ohjelmistojen kehittäjien luotettavuus on usein kiistanalainen aihe yrityksissä.

Kaikki suljetun lähdekoodin kehittäjät ovat lähtökohtaisesti aina ammattilaisia. Näin ollen heidän kehittämät ratkaisut voidaan olettaa olevan toimivia sekä turvallisia. Avoimen lähdekoodin tapauksessa kehittäjät ovat usein tuntemattomia, jolloin heidän tekninen taustasta ei ole tietoa. Lisäksi kehittäjien motivaatiosta osallistua projektiin ei ole tietoa, jolloin he voivat jättää ohjelmistoon tietoturva-aukkoja. Suljetun lähdekoodin ohjelmistoissa vastuu tietoturvaongelmista, aiheutuvista vahingoista ja niiden korvaamisesta on usein ohjelmistoa tuottavalla yrityksellä, kun taas avoimen lähdekoodin tapauksessa vastuu on käyttäjällä.

Suljetun lähdekoodin ohjelmistojen tietoturva perustuu ohjelmiston rakenteen ja lähdekoodin salassapitoon. Avoimen lähdekoodin ohjelmistoissa tietoturva perustuu toimivaan ohjelmiston arkkitehtuuriin sekä koodin laatuun. Koska kaikki käyttäjät näkevät ohjelmiston rakenteen sekä lähdekoodin, he voivat itse arvioida sitä ja ilmoittaa sen puutteista yhteisölle. Näin ollen avoimen lähdekoodin ohjelmistoissa tietoturva-aukot huomataan ja korjataan usein nopeammin.

Suljetun lähdekoodin ohjelmiston käyttöönoton yhteydessä tulee maksaa ohjelmiston käyttölisenssi. Riippuen lisenssiehdoista se voi olla joko käyttäjä –tai yrityskohtainen. Avoimen lähdekoodin ohjelmistoissa lisenssimaksuja ei ole, joten tämän osalta avoimen lähdekoodin ohjelmistojen kustannukset ovat pienemmät. Lisenssimaksut ovat kuitenkin vain pieni osa ohjelmistosta aiheutuvaa kokonaiskustannusta, joten niille ei tule antaa liian suurta painoarvoa. Avoimen lähdekoodin ohjelmistoissa tulee huomioida avointen lisenssien aiheuttamat rajoitukset. Lisenssien rikkomisesta voi syntyä huomattaviakin kustannuksia. Monissa

tilanteissa on hyvä palkata osaava henkilö, jonka vastuulla lisenssiasiat ovat. Edellä mainittua ongelmaa ei ole suljetun lähdekoodin ohjelmistoissa.

Ohjelmiston vaihtaminen on usein yritykselle hankala prosessi. Tämä johtuu siitä, että yritys on investoinut ohjelmistoon, ja jos ohjelmisto ei ole vielä tullut käyttöikänsä päähän, vaihtamiseen ei kovin helposti lähdetä. Suljetun lähdekoodin ohjelmistoissa asiakas on riippuvainen ohjelmistotoimittajasta ja ratkaisut on usein rakennettu niin, että vaihtaminen toiseen ohjelmistoon on mahdollisimman hankalaa. Ohjelmisto kannattaa kuitenkin vaihtaa, jos sen avulla voidaan säästää tulevaisuuden elinkaarikustannuksissa. Jos data-analytiikan käyttö on yksinkertaista, vaihtaminen ei välttämättä ole järkevää. Avoimen lähdekoodin ohjelmistot ovat monesti elinkaarikustannuksiltaan edullisempia (Arnold 2011, s. 44), joten jos data-analytiikka on tärkeässä osassa yrityksen toimintaa, vaihtaminen kannattaa. Tällöin saadaan hyödynnettyä myös muut avoimen lähdekoodin edut.

Data-analytiikkaohjelmistojen elinkaarikustannukset muodostuvat monesta tekijästä. Kustannukset ovat aina yrityskohtaisia ja muun muassa integraatioiden laajuus voi vaikuttaa hyvin merkittävästi koko projektin kustannuksiin. Kustannuksiin vaikuttaa, onko ohjelmisto avoimen vai suljetun lähdekoodin, mutta monilta osin kustannukset ovat samansuuruisia. Työssä kuitenkin havaittiin, että pääosin avoimen lähdekoodin data-analytiikkaohjelmistojen kustannukset ovat pienempiä kuin suljetun lähdekoodin vastaavien. Tätä tukee myös kirjallisuus, jossa on esitetty useita perusteluja ja esimerkkejä tukemaan avoimen lähdekoodin etuja ja tämän työn johtopäätöstä (Arnold 2011, s. 44; Nagy et al. 2010, s. 148; Vonfange & Lavigne 2011, s. 7-10; Yang 2016, s. 681-682). Jatkotutkimusta ajatellen tämä työ toimii kvalitatiivisena pohjana aiheen kvantitatiiviselle tutkimukselle. Jatkotutkimuksessa olisi hyödyllistä tutkia yritysten toteutuneita elinkaarikustannuksia avoimen ja suljetun lähdekoodin data-analytiikkaohjelmistojen osalta.

6 LÄHTEET

Ahmed, R. E. 2006. Software maintenance outsourcing: Issues and strategies. *Computers and Electrical Engineering, 32*(6), s. 449-453.

Arnold, S. 2011. Rob ROI: Open Source and Cost of Technology. *Online, 35*(4), s. 42-44.

Bojanova, I. 2013. Cloud Computing. *IT Professional, 15*(2), s. 12-14.

Bosch, J. 2009. From Software Product Lines to Software Ecosystems. *International Software Product line Conference,* s. 1-10.

Boyd, D. & Crawford, K. 2012. Critical Questions For Big Data: Provocations for a cultural, technological, and scholarly phenomenon. *Information, Communication & Society, 15*(5), s. 662-679.

Caulkins, J. P., Feichtinger, G., Grass, D., Hartl, R., Kort, P. & Seidl, A. 2013. When to make proprietary software open source. *Journal of Economic Dynamics and Control, 37*(6), s. 1182-1194.

Chen, M., Mao, S., Zhang, Y. & Leung, V. 2014. Big Data: Related Technologies, Challenges and Future Prospects. *Springer eBooks*. 89s.

Conry-Murray, A. 2006. The Open Source Leap Of Faith. *InformationWeek, 1103*, s. 51-52,54-55.

Coverity 2014. Coverity Scan Report Finds Open Source Software Quality Outpaces Proprietary Code for the First Time. [verkkoraportti] [Viitattu: 15.3.2017] Saatavissa: http://www.coverity.com/press-releases/coverity-scan-report-finds-open-source-software-quality-outpaces-proprietary-code-for-the-first-time/

Economist. 2002. IT grows up. *Economist, 364*(8287), s. 53-54.

Elgendy, N. & Elgral, A. (2016). Big Data Analytics in Support of the Decision Making Process. *Procedia Computer Science, 100*, s. 1071-1084.

Hoepman, J., Jacobs, B. 2007. Increased security through open source. *Communications of the ACM, 50*(1), s. 79-83.

Janssen, M., van Der Voort, H. & Wahyudi, A. 2017. Factors influencing big data decision-making quality. *Journal of Business Research, 70*, s. 338-345.

Jobs, C., Gilfoil, D. & Aukers, S. 2016. How Marketing Organizations Can Benefit from Big Data Advertising Analytics. *Academy of Marketing Studies Journal, 20*(1), s. 18-35.

Jones C. 2006. The Economics of Software Maintenance in the Twenty First Century. [verkkoraportti] [Viitattu: 24.2.2017] Saatavissa: http://www.compaid.com/caiinternet/ezine/capersjones-maintenance.pdf

Kościelniak, H. & Puto, A. 2015. Big Data in Decision Making Processes of Enterprises. *Procedia Computer Science, 65*, s. 1052-1058.

Leonard-Barton D. & Kraus W. 1985. Implementing New Technology. [verkkoartikkeli] [Viitattu 23.2.2017] Saatavissa: https://hbr.org/1985/11/implementing-new-technology

Nagy, D., Yassin, A. & Bhattacherjee, A. 2010. Organizational adoption of open source software: Barriers and remedies. *Communications of the ACM, 53*(3), s. 148-151.

Neumann, P. 1999. Robust open-source software. *Association for Computing Machinery. Communications of the ACM, 42*(2), s. 128.

Prusansky, C. 2011. The Lower-Cost Open-Source Software Alternative. *Fire Engineering, 164*(5), s. 65-67.

Raghunathan, S., Prasad, A., Mishra, B., Chang H. 2005. Open source versus closed source: Software quality in monopoly and competitive markets. *Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on, 35*(6), s. 903-918.

Rajlich, V., Bennett, K. 2000. A staged model for the software life cycle. *Computer, 33*(7), s. 66-71.

Raymond, Eric S. 1999. *The Cathedral and the Bazaar: Musings on Linux and Open Source by an Accidental Revolutionary*. Sebastopol, CA. O'Reilly Media. 241s.

Sääksjärvi M., Lassila A. & Nordström H. 2005. Evaluating the software as a service business model: from CPU time-sharing to online innovation sharing. *Helsinki School of Economics, Department of Business Technology*. s. 177-186.

Schryen, G. 2011. Is open source security a myth? *Communications of the ACM, 54*(5), s. 130-140.

Sommerville I. 2005. Software Cost Estimation. [e-kirja] [Viitattu: 2.3.2017] Saatavissa: https://ifs.host.cs.st-andrews.ac.uk/Books/SE7/SampleChapters/ch26.pdf

Tilastokeskus. 2011. Tietotekniikan käyttö yrityksissä 2011. [tutkimusraportti] [Viitattu 25.2.2017] Saatavissa: https://www.stat.fi/til/icte/2011/icte_2011_2011-11-24_fi.pdf

Tucker, A., Morelli, R., & De Silva, C. 2011. Software development: An open source approach. *Boca Raton FL, CRC Press*. 370s.

Välimäki, M. 2009. *Oikeudet tietokoneohjelmistoihin* 2., uud. p. Helsinki, Talentum. 282s.

Vonfange, M. & Lavigne, D. 2011. IT Cost Optimization Through Open Source Software. *The Open Source Business Resource,* s. 5-10.

Wormser, D. 2010. Open-Source Software: The Value of "Free". *Intellectual Property & Technology Law Journal, 22*(5), s. 22-26.

Yale. 2001. 4203 Accounting for Software Costs, Computer Systems, and Business Process Reengineering. [verkkoartikkeli] [Viitattu: 26.2.2017] Saatavissa: https://your.yale.edu/policies-procedures/policies/4203-accounting-software-costs-computer-systems-and-business-process#4203.3

Yang, H. 2016. Total Cost of Ownership for Application Replatform by Open-source SW. *Procedia Computer Science, 91*, s. 677-682.

Yu, L. 2013. The Market-Driven Software Ecosystem. *IT Professional, 15*(5), s. 46-50.

KAUPALLISET LÄHTEET

Activestate. 2016. True Cost of Open Source Software. [verkkoraportti] [Viitattu: 23.2.2017] Saatavissa: https://www.activestate.com/sites/default/files/pdfwp/whitepaper-true-cost-opensource.pdf

Black Duck Software. 2017. Top Open Source Licenses. [verkkojulkaisu] [Viitattu 10.3.2017] Saatavissa: https://www.blackducksoftware.com/top-open-source-licenses

Free Software Foundation. 2017a. About the GNU Operating System. [verkkoartikkeli] [Viitattu: 20.2.2017] Saatavissa: https://www.gnu.org/gnu/about-gnu.html

Free Software Foundation. 2017b. What is free software? [verkkoartikkeli] [Viitattu: 20.2.2017] Saatavissa: https://www.gnu.org/philosophy/free-sw.html

Free Software Foundation. 2017c. Copyleft: Pragmatic Idealism. [verkkoartikkeli] [Viitattu: 20.2.2017] Saatavissa: https://www.gnu.org/philosophy/pragmatic.html

Gartner. 2015. Flipping to Digital Leadership: Insights from the 2015 Gartner CIO Agenda Report. [tutkimusraportti] [Viitattu: 22.3.2017] Saatavissa: https://www.gartner.com/imagesrv/cio/pdf/cio_agenda_insights2015.pdf

KDNuggets. 2016. R, Python Duel As Top Analytics, Data Science software – KDnuggets 2016 Software Poll Results. [tutkimusraportti] [Viitattu: 22.3.2017] Saatavissa: http://www.kdnuggets.com/2016/06/r-python-top-analytics-data-mining-data-science-software.html/2

Laakkonen, C. 2013. Avoin lähdekoodi – Tiedätkö mitä rajoituksia se asettaa ohjelmistosi käytölle? [verkkoartikkeli] [Viitattu 24.2.2017] Saatavissa: https://www.sofokus.com/blogi/avoin-lahdekoodi/

Laukkanen M. 2013. Liiketoiminta-analytiikan ohjelmistot, osa 1. [verkkoraportti] [Viitattu: 23.2.2017] Saatavissa: http://www.louhia.fi/2013/01/27/liiketoiminta-analytiikan-ohjelmistot-osa-1/

Microsoft. 2017. Open Source at Microsoft. [verkkosivusto] [Viitattu: 22.3.2017] Saatavissa: https://opensource.microsoft.com

Open Source Initiative. 2017a. History of the OSI. [verkkoartikkeli] [Viitattu 21.2.2017] Saatavissa: https://opensource.org/history

Open Source Initiative. 2017b. The Open Source Definition. [verkkoartikkeli] [Viitattu 21.2.2017] Saatavissa: https://opensource.org/osd

Raths, D. 2013. The True Cost of Open Source. [verkkoartikkeli] [Viitattu 10.3.2017] Saatavissa: https://campustechnology.com/Articles/2013/09/19/The-True-Cost-of-Open-Source.aspx?Page=1

Rubens, P. 2015. How 2 Legal Cases May Decide the Future of Open Source Software. [verkkoartikkeli] [Viitattu 24.2.2017] Saatavissa: http://www.cio.com/article/2893738/open-source-development/how-2-legal-cases-may-decide-the-future-of-open-source-software.html

Sabhlok R. 2013. Open Source Software: The Hidden Cost of Free. [verkkoartikkeli] [Viitattu: 19.3.2017]. Saatavissa: https://www.forbes.com/sites/rajsabhlok/2013/07/18/open-source-software-the-hidden-cost-of-free/#10d276da4001

SAS. 2017. Analytics Platform. [verkkosivusto] [Viitattu: 1.3.2017] Saatavissa: https://www.sas.com/en_us/software/platform.html

The R Foundation. 2017. R: What is R? [verkkosivusto] [Viitattu: 1.3.2017] Saatavissa: https://www.r-project.org/about.html