Open Source -tuotteiden vaikutus tietojärjestelmän kokonaiskustannuksiin

Tapio Seppä-Lassila
pro gradu -tutkielma
Tampereen yliopisto
Tietojenkäsittelytieteiden laitos
Huhtikuu 2002

SISÄLTÖ

1 JOHDANTO

Oikeilla ratkaisuilla voi tietohallintojohto parhaimmillaan samalla sekä keventää IT-toimintojen kulurakennetta että parantaa palvelutasoa. Tavoitteeseen ei päästä niinkään hankinnoista tinkimällä vaan pikemminkin tulossa olevia kustannuksia eliminoimalla. Etukäteisvarautumisessa ennustamiskyvyt olisivat korvaamaton apu, mutta sen puuttuessa voidaan päätöksenteon vahvistajana käyttää joitakin systemaattisia menetelmiä. Menetelmien lähtökohtana on tunnistaa eri kustannustekijät ja saada selville niiden vaikutus kokonaiskustannuksiin. Tyypillisesti kustannusten hallintaan käytetään kahta menetelmää: keskittäminen ja standardisointi [David et. al. 2002]. Harkitenkin toteutettuna näiden menetelmien kääntöpuolena on usein IT-palvelutason heikentyminen. Tässä tutkielmassa kustannusten kertymistä peilataan läpi tietosysteemin elinkaaren [Järvinen ja Järvinen, 2000].

Nopeasti ajateltuna voi kuvitella, että nykyaika on ohjelmapakettien ja valmisohjelmistojen aikaa, ja ettei ohjelmistoprojekteja juurikaan tehdä omina tai teetettyinä hankkeina. Joltain osin ajatus pitää paikkansa, mutta helposti unohtuvat viime vuosien aikana suosioon noussut internet ja sen tuomat sovellusmahdollisuudet. Unohduksen tekee vielä merkittävämmäksi seikka, että yrityksissä on jo vuosia käytetty rahaa erilaisiin www-hankkeisiin, jotka ovat periaatteessa mitä tyypillisimpiä ohjelmiston kehitysprojekteja.

Open Source -mallinen ohjelmistonkehitys ja -jakelu ovat saaneet viime vuosina runsaasti julkisuutta varsinkin Linux-käyttöjärjestelmän ansiosta. Open Source -ohjelmat ovat maksuttomasti saatavilla internet-verkon kautta; myös niiden lähdekoodi on vapaasti saatavissa. Avoimien lähdekoodien ja avoimien standardien käyttämisellä voidaan hakea säästöjen lisäksi myös valinnan vapautta toimittajan valintaan. Valinnan vapauden taustalla ovat edelleen kustannustekijät: tietohallinto haluaa välttää lukittumistilannetta, jossa uusi järjestelmä olisi vanhaa käyttökelpoisempi ja edullisempi, mutta vaihtamisesta koituvat kustannukset olisivat vaihdon esteenä.

Open Source -mallin mukaisten sovellusten, kehitysympäristöjen ja muiden työvälineohjelmistojen hankinta ja päivitykset ovat maksuttomia. Järjestelmien rakentaminen ja ylläpito ei kuitenkaan muutu ilmaiseksi Open Source -tuotteidenkaan myötä, vaikka osan ohjelmointityöstäkin voi saada teetettyä vapaaehtoisvoimin. Laitteisto tarvitaan joka tapauksessa, ja määrittelytyö, rakennus ja ylläpito edellyttävät asiantuntemusta, jota ei ole ilmaiseksi tarjolla.

Viime aikoina on julkaistu joitakin tutkimuksia Open Source -tuotteiden käyttöön ottamisesta. Näissä on usein pitäydytty tutkimaan Open Source -ohjelmistojen soveltuvuutta työasemalaitteiden käyttöjärjestelminä ja toimisto-ohjelmina. Open Source -tuotteiden vahvuudet ovat toistaiseksi olleet kuitenkin palvelin- ja ohjelmankehityssektorilla, ja tutkielmassa tarkastellaan myös palvelinohjelmistojen kustannusvertailua. Tässä tutkielmassa lähestytään aihetta myös rakentamalla tietokantapohjainen www-palvelu Open Source -välineillä.

Open Source -tuotteet ovat kiinnostaneet erityisesti julkishallinnon yksiköitä, niin kuntia kuin valtioitakin, nimenomaan kustannussyistä, vaikka tuotteiden ympärillä käydään myös erilaisia ideologisia keskusteluja. Varsinkin julkishallinnossa tietotekniikka on tukitoiminto, jonka avulla kansalaisille tuotetaan palveluja. Myös yrityksessä tukitoimintoluonne säilyy, ohjelmistotalot poislukien, vaikka se olisi keskeinen toiminnan mahdollistaja. Voittoa tavoittelevassa liikeyrityksessä ei työkaluohjelmistojen valintaa pitäisi tehdä pelkästään ideologisin tai muin tunnepitoisin perustein. Tässä tutkielmassa tutkitaan, onko järjestelmäinvestoinnissa mahdollista aikaansaada mitattavissa olevia kustannussäästöjä hyödyntämällä Open Source -mallin mukaisia tuotteita.

Tutkielman lähtökohtana ei ole vastustaa mitään kaupallisessa levityksessä olevaa ohjelmistoa tai käyttöjärjestelmää. Kun tekstissä mainitaan nimeltä jokin kaupallinen ohjelmisto tai ohjelmistotalo, käytetään nimikettä vain välttämättömänä verrokkina Open Source -tuotteeseen. Tutkielma on kirjoitettu IT-järjestelmien rakentamisesta ja hankinnasta päätöksiä tekevän tietohallinto- ja yritysjohdon luettavaksi. Tutkielman lukija voi mieltää itsensä lukemisen ajaksi kuvitteellisen yrityksen tietohallintojohtajaksi ja seurata päätöksentekoon liittyvien motiiviensa kehittymistä. Tutkielman yhtenä tarkoituksena on antaa eväitä Open Source -ohjelmistomallin hahmottamiseen, ja sen suhteuttamiseen osaksi omaa IT-kokonaisuutta sekä valottaa kokonaiskustannuslaskennan kompleksisuutta. Tutkielman tekemiseen tai ideointiin ei ole liittynyt minkäänlaista sponsoritoimintaa. Tutkielman case-esimerkkinä oleva verkkopalvelu suunniteltiin ja ohjelmoitiin työn aikana. Ohjelmointityön tarkoituksena oli simuloida käyttökelpoisen lisäpalvelun tuottamista talon sisäisenä projektina. Toimivan sovelluksen tuottaminen onnistui varsin luontevasti, mitä on pidettävä kannustimena yrityksille säilyttää IT-osaamista organisaation henkilöresursseissa.

Yhtenä tavoitteena oli hahmotella laskentakaava, jota soveltamalla voidaan tutkia erilaisten ohjelmistovalintojen vaikutusta järjestelmän kokonaiskustannuksiin. On teoriassa mahdollista kirjoittaa kaava, johon sijoittamalla järjestelmään ja organisaation infrastruktuuriin liittyviä lukuja saadaan tuloksia, joita voisi pitää kokonaiskustannuslaskennan tuloksina. Tietojärjestelmien suurimmat kustannukset

kertyvät kuitenkin käytönaikaisista menoista [Gillen et al. 2001]. Sovelluskohteen ominaispiirteet ja henkilöstön osaaminen vaihtelevat organisaatioiden välillä niin runsaasti, että kiinteiden kertoimien luominen näille tekijöille katsottiin mahdottomaksi. Käytännön valintatilanteissa tämä ongelma ratkaistaan tutkimalla konkreettisesti sovelluskohdetta ja tarvittavia palveluita sekä luomalla tapauskohtaiset painoarvot suuresti vaihteleville tekijöille.

Tutkielma jakautuu kahdeksaan lukuun. Luvussa kaksi käsitellään tietojärjestelmän ominaispiirteitä ja tekijöitä, joita punnitaan järjestelmien valinnassa. Luvussa kolme sukelletaan Open Source -maailmaan ja käydään läpi keskeiset Open Source -ohjelmistoihin liittyvät piirteet ja käsitteet sekä ongelmat ja mahdollisuudet. Luvussa neljä käsitellään kustannustekijöitä ja kustannusten laskentaa eri näkökulmista sekä tarkastellaan kustannusten laskemisen kompleksisuutta. Luvussa viisi perehdytään esimerkkiprojektiin ja perustellaan valittu toimintatapa. Luvussa kuusi käsitellään ohjelmistohankkeen toteutusta sekä arvioidaan työn onnistumista ja sen suhdetta reaalimaailmaan. Luvussa seitsemän tutustutaan joihinkin Open Source -mallin ominaispiirteisiin. Lopuksi luvussa kahdeksan pohditaan Open Source -ohjelmistomallin elinmahdollisuuksia vastaisuudessa, sekä mahdollisia tutkimussuuntia, joihin tämän tutkielman pohjalta voisi jatkaa.

2 JÄRJESTELMÄN HANKINNAN YLEISPERIAATTEITA

Tässä luvussa käsitellään tietojärjestelmän hankintaa ja hankinnan suunnittelua kohdassa 2.1. Järjestelmien kustannukset alkavat kertyä jo suunnitteluvaiheessa ja tämä tulee ottaa huomioon laskennassa. Järjestelmien elinkaari voidaan esitetään kohdassa 2.2 yksinkertaisella kaaviolla. Elinkaaren viimeisenä vaiheena on järjestelmän poistaminen käytöstä. Poistamisesta aiheutuu kustannuksia, joihin elinkaaren alkuvaiheessa ei välttämättä ole kiinnitetty huomiota. Käytännössä voikin olla vaikea erottaa, mitkä kustannukset sisältyvät vanhan järjestelmän poistamiseen ja mitkä kuuluvat uuden järjestelmän käyttöönottoon.

Ohjelmistojen hankintahintaan kiinnitetään paljon huomiota. Osaltaan tästä syystä maksuttomat Open Source -tuotteet herättävät maksajien kiinnostusta. Varsinaiset hankinta- ja lisenssimaksut ovat pieni osa [Valtiovarainministeriö 2001a] IT-kustannuksista, mutta ohjelmistojen valinnoilla voidaan vaikuttaa muidenkin menolajien kurissa pitämiseen. Kohdassa 2.3 käsitellään hankintahintoja.

Ohjelmistojen valinnassa ja hankinnassa, ostettuna tai itse tehtynä, on monta näkökulmaa ja arvioijan suhde tuotteeseen (esimerkiksi käyttäjän, maksajan tai ylläpitäjän) vaikuttaa tarkasteluperspektiiviin. Hankinnan eri näkökulmia tarkastellaan kohdassa 2.4. Kohdassa 2.5 todetaan että IT-asiantuntijoilla saattaa olla myös ennakkoasenteita tai tunnepitoisia syitä suhtautumissaan joihinkin vaihtoehtoihin. Näiden asenteiden motiiveihin ei tutkielmassa mainintaa enempää puututa.

2.1 Ennen hankintaa

Tietojärjestelmän hankinnan, käytön ja ylläpidon kustannukset riippuvat useista tekijöistä. Ratkaisun hankintakustannukset nähdään helposti kokonaiskustannuksina eivätkä erilaiset välilliset ja seurannaiskustannukset eivät hahmotu kovin helposti.

Tästä ei tarvitse ostajaa täysin syyttää, sillä myyntivaiheessa kokonaiskustannusten mahdollisimman laaja selvittäminen ei välttämättä ole ratkaisun toimittajan intresseissä päällimmäisenä.

Järjestelmän määrittelyvaiheessa tehdään useita keskeisiä linjauksia, joita noudatetaan sen suunnittelu- ja rakennusvaiheessa. Toimintaympäristö ja tarpeet, joita täyttämään

järjestelmä hankitaan, muodostavat pelikentän, jonka sisältä haetaan optimaalinen ratkaisu.

Valitut ratkaisut saattavat pohjautua valmiisiin pakettiratkaisuihin, jotka järjestelmän toimittaja räätälöi asiakkaan toiveiden mukaisiksi. Valmiit tai puolivalmiit ratkaisut ovat tyypillisesti kaupallisia ohjelmistoja, joissa kustannuksia syntyy useassa vaiheessa.

Määrittelyvaiheen kustannukset ovat suoraan verrannollisia järjestelmän laajuuteen. Määrittelytyö tehdään useimmiten tuntiveloituksena, ja laajemman kokonaisuuden selvittämiseen ja kuvaamiseen kuluu luonnollisesti pidempi aika kuin suppeamman.

Järjestelmän rakentamisen ja suunnittelun yleisiä periaatteita ja menetelmiä kuvataan tässä tutkielmassa kirjallisuuslähteitä hyödyntämällä. Tietojärjestelmän kustannustekijöiden kartoittamisessa viitataan aiheesta julkaistuihin artikkeleihin. Tutkielmassa perustellaan kustannustekijöiden motiivit tekijöiden soveltamiseen laskennassa, ja samalla koetetaan luodaan kustannusten laskentamalli. Jäljempänä tekstissä käytetään myös lyhennettä *tco* tarkoittamaan järjestelmän kokonaiskustannuksia (total cost of ownership). Vaikka vieraskielisessä termissä puhutaan omistamisesta, on kustannuslaskennan kannalta teknisesti samanarvoista, miten järjestelmän hankinta on rahoitettu. Osa laitteista voi olla ostettuja, osa leasing-kalustoa. Ohjelmistoissakin voidaan soveltaa monipuolisia rahoitusmalleja.

Open Source -tuotteiden käytettävyydestä puhuttaessa tulee erottaa vapaassa jakelussa olevien valmiiden tuotteiden ja komponenttien hyödyntäminen sekä varsinainen Open Source -mallinen ohjelmistonkehitys toisistaan. Hyödyntämisellä tarkoitetaan tässä tutkielmassa lähinnä valmiiden ohjelmien käyttöä kaupallisten ohjelmistojen sijaan, kun taas Open Source -mallisella ohjelmistokehityksellä tarkoitetaan omien tuotteiden luovuttamista vapaaseen käyttöön jonkin avoimen lähdekoodin lisenssin mukaisesti. Oma tuote voi olla ohjelmisto, funktio, tietokanta tai muu määrittely.

2.2 Systeemin elinjakson määrittäminen

Tietojärjestelmien elinajan laskentaan sisällytetään yleensä järjestelmän suunnitteluun ja määrittelyyn käytetty aika ja muut resurssit. Nämä kuuluvat järjestelmän rakennusvaiheeseen (r), joka on oleellinen osa järjestelmän elinkaarta. Loppuosan voidaan ajatella olevan ohjelmiston käyttöaikaa (k).

```
|rrrrrrrrr|kkkkkkkkkkkkkkkkkkkkkkkkkk|
```

Järvinen [2000] tuo käyttöajanjaksolle mukaan uuden tekijän, huoltojakson. Huoltojakson erottaminen muusta käyttöjaksosta on perusteltua, koska se toiminnallisesti eroaa järjestelmän varsinaisesta tuotantokäytöstä. Tällöin järjestelmää ei käytetä siihen tarkoitukseen, johon se on suunniteltu ja rakennettu, mutta järjestelmä on kuitenkin elossa ja jatkaa toimintaa huollon jälkeen. Käyttöjakso jakautuu siis tuotanjaksoon (t) ja huoltojaksoon (h).

```
|rrrrrrrrr|ttttt tttt ttttt tttt  tt tt|
                 h    h     h    hh  h
```

On kuitenkin olemassa järjestelmiä, joita ei voida ajaa alas kovinkaan usein tai käytännössä juuri koskaan. Jos huoltojaksoon sisällytetään myös järjestelmän käyttöjakson aikainen kehittämis- ja parantamistyö, huoltojakso saattaa venyä niin pitkäksi, ettei käyttöjakson aikana ole mahdollista pitää järjestelmää alhaalla yhtäjaksoisesti niin kauan.

Järjestelmiin tehdäänkin korjauksia usein samanaikaisesti, kun niitä käytetään. Parannukset otetaan käyttöön "lennossa", järjestelmää välillä pysäyttämättä ja tuotannon häiriintymättä taikka pysäyttämällä järjestelmä vain lyhyeksi ajaksi, jolloin uusi komponentti liitetään mukaan tai jokin osa järjestelmästä korvataan uudemmalla.

Kutsun näitä ennen järjestelmän pysäytystä tapahtuvia huolto- ja ylläpitotoimenpiteitä ennakoivaksi huolloksi (e). Uuden luokan perustaminen on mielestäni tarkoituksenmukaista, koska ennakoivan huollon ei voida ajatella kuuluvan järjestelmän normaaliin tuotantokäyttöön, ja koska toisaalta haluan erottaa sen niistä huoltojaksoista, jotka toiminnallisesti syrjäyttävät tuotantojakson. Erityisen merkittäväksi uusi luokka tulee, kun haluamme tarkastella järjestelmän kokonaiskustannuksia. Ennakoivaa huoltoa voidaan tehdä järjestelmän koko käyttöjakson ajan, myös huoltojaksot mukaan lukien.

```
               ee   eee       eee   ee
|rrrrrrrrr|ttttt tttt ttttt tttt  tt tt|
                 h    h     h    hh  h
```

Järjestelmän elinkaari ei kuitenkaan pääty, kun sen käytöstä luovutaan. Järjestelmän tiedot tulee mahdollisesti tallentaa arkistokäyttöön. Mahdollisesti myös laitteistoa vanhenee samanaikaisesti, ja siitä tulee päästä eroon. Ellei yritys lopeta toimintaansa tai vaihda toimialaansa täysin, on vanhat tiedot usein siirrettävä uuden järjestelmän käyttöön. Tässä liikumme hieman "harmaalla alueella": luetaanko kustannukset, jotka liittyvät vanhojen tietojen tallentamiseen, konvertointiin ja siirtämiseen uuteen

systeemiin, uuden järjestelmän rakennusvaiheen kuluihin vai kuuluvatko ne vielä vanhan systeemin kuluihin.

Järvinen [2000] ottaa käyttöön uuden luokan: käytöstä poistaminen ja uudelleen käyttöön/kiertoon palauttaminen (p). Nyt saatu kuvaus järjestelmän elinkaaresta on informatiivisempi ja ulottuu loogisesti järjestelmän synnystä sen hautaan. Myös kustannusten kohdistaminen tiettyihin jaksoihin on nyt suoraviivaisempaa.

```
                ee   eee       eee   ee
|rrrrrrrrr|ttttt tttt ttttt tttt  tt tt|ppppp|
                  h    h     h    hh  h
```

2.3 Ohjelmiston hankintahinnasta

Ohjelmistot voidaan jakaa kahteen luokkaan sijoittamalla ne joko valmisohjelmiin tai räätälöityihin ohjelmistoihin. Valmisohjelma on asennusvalmis paketti, jonka ohjelmistotalo on tuotteistanut todetessaan sen olevan kypsä markkinoille. Ohjelmiston monistamiskustannukset ovat lähes olemattomat verrattuna kehityskustannuksiin tai asiakkaan vaatimusten perusteella räätälöityyn versioon. Edullisempaa hintaa voi pitää myös asiakkaan etuna. Yksinkertaistetusti asiakas maksaa lisenssimaksun, saa ohjelman käyttöönsä ja käyttää ohjelmaa tarpeensa mukaisesti. Jatkokustannukset riippuvat lähinnä ohjelman lisenssiehdoista.

Valitessaan valmisohjelman asiakas pystyy ennustamaan tarkemmin tulossa olevat kiinteät ohjelman käyttöön liittyvät kustannukset. Määrittelyvaiheen kustannuksia ja asennukseen liittyviä rasitteita ei tässä oteta huomioon. Valmisohjelmat ovat tyypillisesti hankintahinnaltaan edullisempia kuin nimenomaan asiakkaan tarpeeseen räätälöidyt ohjelmistot, mutta kustannukset nousevat, kun valittu ohjelmisto ei enää täytä tarpeita ja sitä olisi aika päivittää.

Valmisohjelmistojen mukana ei juuri koskaan seuraa lähdekoodia, joten muutoksia ei kovin paljon tehdä muulloin kuin ohjelmistovalmistajan julkaistessa uusia päivityksiä. Räätälöidyn ohjelman mukana voidaan sopia saatavaksi myös lähdekoodi. Tämä saattaa maksaa, ja tämä maksu on ohjelmiston hankkijan kustannus ns. lock-in -tilannetta vastaan.

Lock-in -tilanne on yksinkertaisimmillaan sellainen, että asiakkaalla on käytössään esimerkiksi taloushallinto-ohjelmisto. Asiakas haluaisi ohjelmistoon muutoksen, jotta se palvelisi yrityksen muuttunutta liiketoimintaa paremmin. Ohjelman toimittanut

ohjelmistotalo ilmoittaa muutostöille kustannusarvion, joka on asiakkaan mielestä kohtuuttoman suuri. Jos asiakkaalla ei ole käytössään ohjelman lähdekoodia tai lupaa sen muuttamiseen, vaihtoehdot ovat tulla toimeen puutteellisella ohjelmistolla, hyväksyä ohjelmistotoimittajan kalliilta tuntuva tarjous tai aloittaa toimet uuteen taloushallinto-ohjelmaan siirtymiseksi.

Toinen tyyppiesimerkki lock-in tilanteesta on sellainen jossa asiakas haluaisi integroida taloushallinnon ohjelmansa muihin olemassa oleviin tai suunnitteilla oleviin ohjelmiin, kuten crm-järjestelmään (customer relationship management) tai varastonhallinta-ohjelmistoon. Mikäli asiakkaalla ei ole käytössään rajapintatietoja tietojen tuontiin ja vientiin taloushallintaohjelmasta, saattaa integrointi olla kohtuuttoman hankalaa tai kallista. Saattaa olla, että taloushallintaohjelman toimittaneen ohjelmatalon valikoimiin kuuluva crm-tuote on huomattavasti vaihtoehtoista kalliimpi, vaatimattomammin ominaisuuksin varustettu tai vanhanaikaisempi kuin kilpailijan tuote. Jälleen on asiakkaalla valittavanaan joko epämieluisa tai kallis vaihtoehto.

Ohjelmiston toimittajalle voidaan avoimen lähdekoodin saamisen sijaan esittää myös kevennetty vaatimus. Hankittavan ohjelmiston tulee vähintään noudattaa avoimia standardeja, jolloin käytön aikana kertyneiden tietojen siirto muihin järjestelmiin voidaan toteuttaa. Lisäksi ohjelmistoon voidaan teettää tarvittaessa muilla valmistajilla lisäosia tai laajennuksia, mikäli tiedostoformaatit ja tiedonsiirtoprotokollat perustuvat avoimiin standardeihin. Avoin lähdekoodi sekä avointen standardien käyttö eivät sinänsä takaa hinnan edullisuutta seuraavassa projektissa, mutta parantavat asiakkaan neuvotteluasemaa, kun keskustellaan jatkoprojekteista.

2.4 Hankinnoissa useita näkökulmia

Ohjelmiston hankintaan liittyy hankintahinnan lisäksi lukuisa joukko muitakin tekijöitä, ja näiden merkitys vain kasvaa hankittavan tuotteen laajuuden mukaan. Tutkimme hieman näitä näkökulmia ja tekijöitä Messerschmittin ja Szyperskin [Messerschmitt and Szyperski 2000] jaon mukaan.

		Technology	***Processes***	***Value***
Participants	***Needs(users)***	Flexibility	Security, privacy	Functionality, impact
	Design (software engineers)	Representation, languages, execution, portability, layering	Architecture, composition vs. decomposition, standardization	Requirements, functionality, quality, performance

		Technology	*Processes*	*Value*
Facilitators	***Roles (managers)***	Infrastructure	Development, provisioning, operations	Uses
	Legal & policy (lawyers, regulators)	Intellectual property (patent, copyright, trade secret)	Licensing, business process patents, antitrust	Ownership, trademark (brand)
	Industrial organization (owners)	Components, portability	License vs. subscribe, outsourcing	Software and content supply, outsourced development, system integration, service provision
Observers	***Economics (economists)***	Costs	Business relationships, terms and conditions	Supply, demand, pricing

Taulukko 1. Ohjelmiston hankintaan liittyvät näkökulmat ja muut tekijät

Kaikkien ohjelmien tärkein ominaisuus on palvella siinä tarkoituksessa, johon se on hankittu ja täyttää käyttäjän tarpeet mahdollisimman hyvin. Käyttäjä voi tässä tarkoittaa yksittäistä henkilöä, osastoa tai iso konsernia. Käyttäjälle kaikki teknologiset tekijät ovat toissijaisia, tärkeintä on mahdollisuus hoitaa mahdollisimman hyvin tehtävä, johon tietotekniikkaa tarvitaan ja usein myös mahdollisimman nopeasti. Käyttäjälle on samantekevää kulkeeko teksti tulostimelle TCP/IP -protokollan ja vai AppleTalkin kuljettamana, samoin käyttäjää ei kiinnosta tallennetaanko tiedot IDE- vai SCSI-kiintolevylle.

Ohjelmiston suunnittelu aloitetaan tarpeiden, haluttujen toimintojen ja sovellusympäristöön liittyvien muiden reaalimaailman tilojen määrittelyllä ja kuvaamisella. Loppukäyttäjän ja järjestelmän toimittajan suurin haaste on sisällyttää kaikki tarvittavat prosessit sekä reaalimaailman tilat tulevaan ohjelmistoon. Loppukäyttäjän kannalta on merkityksetöntä, tuotetaanko tämä tietotekninen reaalimaailman projektio Open Source -työkaluilla vai joillakin muilla menetelmillä.

Open Souce -mallin mukainen ohjelmistotuotanto, sekä siihen liittyvät työkalut ja menetelmät, ei ole ensisijaisesti loppukäyttäjän, vaan pikemmin ohjelmiston kehityksestä, ylläpidosta ja varsinkin kustannuksista vastaavien intresseissä. Tätä logiikkaa jatkamalla voidaan ajatella, ettei edes ohjelmiston kehitys- ja ylläpitohenkilöston tarvitsisi olla kiinnostunut siitä millaisin lisenssein varustetuilla työkaluilla he töitään tekevät. Periaatteessa riittäisi, kun taloudesta vastavat ihmiset

tutkisivat tarjolla olevat vaihtoehdot ja ilmoittaisivat valintansa, joiden mukaan jatkossa toimitaan.

Käytännössä tietojärjestelmien, ohjelmistotyökalujen ja ohjelmien valinta ei ylipäätään ole aivan näin yksinkertaista. Yritysten ja julkishallinnon yksiköiden talouspäättäjillä on harvoin kokemusta ja tietämystä tietotekniseltä alueelta niin paljon, että voisivat yksin tehdä lopulliset päätökset. Tietohallintoa todennäköisesti kiinnostavat paljonkin erilaiset lisenssityypit ja hinnoittelumallit. Tietohallinnolla on budjettinsa, jonka puitteissa se hoitaa sille määrätyt tehtävät. Mikäli tarjolla on vaihtoehto, jossa lisenssimaksujen säästämisestä huolimatta voidaan palvelut toteuttaa tasoa laskematta, on vaihtoehto tutkittava. Tietohallintoyksikön on otettava taulukon 1 kaikki osatekijät huomioon, katsottava vuoroin kokonaisuutta maksajan silmin ja vuoroin palvelujen käyttäjän näkökulmasta. Tehtävä ei ole helppo, eikä taloushallinnolla aina ole resursseja ymmärtää tehtyjä ratkaisuja. Vuonna 2000 yhdysvaltalaisten yritysten talousjohtajista 86 % oli sitä mieltä, etteivät yrityksen IT-kulut olleet järkevästi hallinnassa [Schneider 2000].

Arviointiin mukaan otettavien tuotteiden valinta sekä arvostelukriteerien määritteleminen edellyttää tietotekniikan tuntemista hinnoittelupolitiikkaa syvemmältä. Yksi tapa, jolla voidaan kiertää tietotekniikka-asiantuntijan käyttöä, on valita tuotteet mahdollisimman pitkälle markkinajohtajien tarjonnasta. Tällä varmistetaan, että ollaan samassa veneessä monen muun kanssa. Sitä ei tiedetä, oliko valittu ratkaisu paras tuote juuri omaan tarpeeseen, saatikka sitä, miten paljon kalliimmaksi tai halvemmaksi jokin muu ratkaisu olisi tullut.

Tyylipuhdasta valmisohjelmistoa, sellofaanisoftaa, hankittaessa on iso osa määrittelytöistä tehty ohjelmiston valmistajan toimesta. Tarjolla olevista ohjelmista voidaan valita ne, jotka täyttävät taulukon 1 vaatimukset *Participants*-osion osalta mahdollisimman täydellisesti. Seulan läpäisseiden tuotteiden osalta tutkitaan mikä vastaa *Observers*-osion kysymyksiin parhaiten (=pienimmin kustannuksin) ja valitaan se. *Facilitators*-osion tekijät seuraavat automaattisesti mukana, eikä niihin tarvitse kiinnittää erityistä huomiota, koska useimmiten niille ei kuitenkaan voi enää hankintavaiheessa tehdä mitään.

Mikäli ratkaisu päädytään hankkimaan ohjelmistotalolta, jossa myös varsinainen ohjelmointityö tehdään, ovat *Facilitators*-osion tekijät keskeisiä määrittely- ja hankintaneuvottelujen puheenaiheita. Ratkaistavia asioita ovat muun muassa valmiin tuotteen omistusoikeus, oikeus lähdekoodiin, sen muokkaamiseen ja edelleen luovuttamiseen sekä molemminpuoliset velvollisuudet ja oikeudet tuotteen luovuttamisen ja käyttöönoton jälkeen. Tässä vaiheessa syntyy sopimuksia hyvin vaihtelevin ehdoin, ja mikäli ratkaisu sisältää Open Source -tuotteita, se otetaan huomioon tässä vaiheessa.

Jos sitä vastoin ohjelmistotuotanto tehdään organisaation omana työnä omaan käyttöön, ovat taulukon 1 kaikki tekijät olemassa. Kaikkia ei tarvitse ratkaista neuvottelemalla, vaan osa voidaan päättää tilaajalle itselle sopivimmalla tavalla. Ensi silmäyksellä helpoin ratkaisu olisi vain suunnitella ja toteuttaa ohjelma, ja ottaa se tuotantoon. Käytännössä kannattaa kuitenkin käydä läpi, määritellä ja kirjata ylös myös kaikki *Facilitators*-osion kohdat, aivan kuin ulkopuoliselta toimittajalta hankittaessa. Lisenssien ja omistusoikeuksien määrittely saattaa tässä vaiheessa tuntua tarpeettomalta ja turhauttavalta, mutta helpottavana seikkana on, että neuvottelut käydään oman organisaation kanssa. Välttämättä mitään ei varsinaisesti neuvotella, vaan asiat vain kirjataan ylös. Varsinkin liike-elämässä saattaa tulla yllättäviäkin omistuspohjamuutoksia, osastoja saatetaan yhtiöittää, toimintoja voidaan myydä pois tai ulkoistetaa jne. Tietohallintojohdon elämää helpottaa, kun olemassa olevien ohjelmien juridinen status on valmiiksi määritelty, ja tietohallinto voi keskittyä muutoksen vaatimiin konkreettisiin toimiin.

Ohjelmiston hankinnassa päätökset on tehtävä parin perustavaa laatua olevan vaihtoehdon välillä: hankitaanko räätälöitävä vai valmis sovellus ja kuka tehtävän hoitaa. Tutkimuksen mukaan [Nelson et. al. 1996] teknisesti pitkälle vietyjen ja erikoistuneiden ohjelmistojen hankinta annetaan helpommin ulkopuolisen tehtäväksi, kun taas yksikertaisimmat sovellukset hankitaan itse ja useimmiten valmisohjelmina. Toisin kuin saattaisi olettaa, yritysten strategisia sovelluksia ei tehdä muita sovelluksia useammin talon sisäisenä työnä. Yritykset ovat halukkaita teettämään sovelluksia ostoprojekteina ja ulkoistamaan ylläpidon.

Tietotekniikan päätöksenteon katsotaan lähestyvän yrityksen strategista päätöksentekoa: näin myös tco-laskennan voidaan ajatella olevan työkalu koko yrityksen johtamisessa, eikä pelkästään IT-osaston valintojen onnistumisen arvioinnissa. Yhdysvaltalainen tutkimuslaitos Hackett Benchmarking & Research tutki vuosien 1998-2001 välillä tapahtuneita muutoksia yritysten organisaatioissa [Hackett 2001] ja totesi tietohallintojohtajien entistä useammin kuuluvan yritysten strategiseen johtoryhmään. Raportointisuhteet olivat myös muuttuneet. Tutkimuksen mukaan 29 % tietohallintojohtajista raportoi suoraan hallituksen puheenjohtajalle, kun kolme vuotta aiemmin luku oli 20 %. Aiemmin 44 % tietohallintojohtajista raportoi talousjohtajalle, vuonna 2001 enää 37 %.

2.5 Ohjelmistot kiistakapulana

Tietokoneohjelmistoja voidaan vertailla monin eri menetelmin. Kriteereinä voivat olla erilaiset kustannustekijät, ohjelmiston soveltuvuus suunniteltuun käyttötarkoitukseen, ohjelmiston soveltuvuus vanhoihin järjestelmiin tai ohjelmistojen mahdolliset tulevaisuuden näkymät, kuten laajennettavuus sekä tuen ja ylläpidon jatkuvuus.

Nämä ovat useimmiten varsin hyvin mitattavissa olevia kriteereitä ja niiden pohjalta voidaan tehdä päätöksiä hankinnoista. Päätösten taustalla saattaa kuitenkin olla muitakin kriteereitä, joita ei ole helppo formuloida, eikä kovin helppoa erotella omiksi tekijöikseen.

Yksi tällainen tekijä on järjestelmäasiantuntijoiden ja ylläpitäjien muutosvastarinta, toinen on haluttomuus usean järjestelmän opiskelemiseen. Muutosvastarinta voi kohdistua tiettyyn toimintamalliin, tiettyyn ohjelma- tai laitevalmistajaan tai uuteen järjestelmään sinänsä, tuotteista tai tekijöistä riippumatta. Syitä ammattilaisiakin vaivaavaan muutosvastarintaan on lukuisia. Toisinaan syyt ovat enemmän todellisia ja konkreettisia, joskus varsinainen kompastuskivi on vain uuden järjestelmän tuntemattomuus.

Muutosvastarintafaktori, sen voittaminen ja muut tekijät jätetään tämän tutkielman ulkopuolelle, vaikka aihe itsessään on varmasti useankin tutkimustyön arvoinen. Tämän tutkielman näkökulmasta tietotekniikan valintoihin liittyvät käyttökelpoiset ja hyväksyttävät perustelut on pystyttävä esittämään taloudellisin argumentein.

3 MITÄ ON OPEN SOURCE

Tietotekniikkakontekstissa käytettynä *Open Source* -termi merkitsee ohjelman lähdekielisen koodin vapaata saatavuutta, sekä oikeutta soveltaa koodia muissa ohjelmissa suoraan tai modifioituna. Open Source on terminä varsin vakiintunut kansainvälisessä alan kirjallisuudessa. Suomen kielisessä tekstissä käytetään myös *avoin lähdekoodi* -ilmausta samassa tarkoituksessa.

Ohjelmien lähdekoodin luovuttaminen ohjelman mukana on peräisin yliopisto- ja tutkimusympäristöistä, joissa lähdekoodi katsottiin osaksi tutkimustuloksia, ja joka siten siten julkaistavissa muiden tulosten yhteydessä.

Jonkinlaisena Open Source -mallin syntysijana pidetään MIT:iä (Massachusetts Institute of Technology), jonka eräs ohjelmoija oli Richard Stallmann. Stallman on ollut erittäin aktiivinen Open Source -käsitteen levittäjä ja kehittäjä sekä projektityöskentelyn että julkisen esiintymisen kautta. Kohdassa 3.1 valotetaan Open Source -ohjelmistojen historiaa.

Open Source -projektien onnistuminen vaatii taitavaa projektinhallintaa samoin kuin kaikki muutkin ohjelmistoprojektit. Vapaaehtoinen ohjelmoija- ja testaajaryhmä on hankkeiden keskeinen voimavara. Tämän joukon menestyksellinen hallinta, koordinointi ja motivointi ilman taloudellisia kannustimia vaatii projektin johdolta sekä suunnitteluosaamista että sosiaalisia taitoja. Avoimen lähdekoodin käyttö tarjoaa myös yrityksille mahdollisuuksia esimerkiksi käyttäjäkunnan kasvattamiseksi sekä sidosryhmien sitouttamiseksi. Ohjelmistotalot voivat julkaista tuotteidensa lähdekoodin, jolloin talon ulkopuoliset, tavallisesti vapaaehtoiset, ohjelmoijat voivat tutkia koodia, tehdä parannusehdotuksia sekä ohjelmoida julkaistuun koodiin yhteensopivia lisätuotteita. Ohjelmistotalon tavoitteena on saada kansainvälinen vapaaehtoinen ohjelmoijajoukko ja julkaistujen tuotteiden määrä riittävän laajaksi, jolloin ohjelmistotalo koodin alkuperäisenä julkaisijana pääsee vaikuttamaan alan standardien kehittymiseen.Open Source -projektien osallistujia ja projektinhallintaa tarkastellaan kohdassa 3.2.

Kohdassa 3.3 esitellään Open Source Initiativen yleinen Open Source -määritelmä. Open Source -mallin lisenssejä käsitellään kohdassa 3.4. Ohjelmistojen juridiikka käsitellään alakohdassa kohdassa 3.4.1 ja kohdassa 3.4.2 pohditaan ohjelmistopatenttien merkitystä. Kaikki maksuttomat ohjelmat eivät Open Souce -tuotteita. Näitä tarkastellaan kohdassa 3.5. Kohdassa 3.6 tutustutaan Open Source -ohjelmistokehitysprojektin kulkuun.

3.1 Open Source historiasta lyhyesti

Ohjelmakoodin kierrätystä on harrastettu yliopistoissa ja tutkimuspiireissä todennäköisesti yhtä kauan kuin ohjelmointia ylipäätään. 1980-luvulla ohjelmien vaihto ja asiasta keskustelu lisääntyi. Tallennusvälineiden ja tietoliikennetekniikan kehittyessä maksuttomien tuotteiden jakelusta tuli yhä tavanomaisempaa. Richard Stallman, joka on ohjelmoinut muun muassa Emacs -editorin ja GCC-kääntäjän sekä koordinoinut lukuisten yhä käytössä olevien ohjelmien tuotantoa, otti käyttöön termin *vapaa ohjelma* (Free Software). Stallmann ei hyväksynyt ohjelmien lisensointia, koska hän näki lisensointitoiminnan rajoittavan vakavasti ohjelmistojen kehitystä.

Stallman perusti 1985 Free Software Foundation -nimisen säätiön edistämään vapaiden ohjelmien kehitystä. Säätiö lanseerasi termin *copyleft* ikään kuin vastalauseena lisenssisopimusten copyright -termille. Stallmannin GNU-projekti kehitti varusohjelmia Unixin kaltaiseen käyttöjärjestelmään, joka oli tarkoitus julkaista maksutta sekä vapaana kopiontirajoitteista. GNU-ohjelmiston tekijänoikeuksien suojelemiseksi luotiin uusi lisensointimalli, GNU General Public Lisence, GPL, jota käytetään edelleen myös uusissa ohjelmistoissa.

GNU-projektilla oli mittava määrä käyttöjärjestelmän varusohjelmia valmiina, mutta käyttöjärjestelmän ytimen, HURD:n, kehitystyö ei edennyt vastaavassa tahdissa. Varusohjelmia käytettiin vuosia eri Unix-versioiden yhteydessä, kunnes ne 1990-luvun alkupuolella paketoitiin Linus Torvaldsin ohjelmoiman käyttöjärjestelmän ytimen ympärille. Tällöin sai alkunsa mittava Linux-käyttöjärjestelmän kehitystyö, joka on tunnetuin Open Source -projekti tänä päivänä.

Richard Stallmannia kiinnosti kuitenkin enemmän vapaan ohjelmistojen oikeuksien puolustaminen kuin kollektiivisen ohjelmointityön hyödyntäminen myös liiketaloudellisesti. Netscape (Netscape Communications Corporation) tammikuussa 1998 ilmoitti julkaisevansa selaimensa lähdekoodin voittaakseen lähdekoodin avoimuuden myötä käyttäjiä takaisin, jotka se oli menettänyt Microsoftin selaimelle. Netscapen ilmoituksen mukaan pontta päätökseen antoi Eric Raymondin vuotta aiemmin julkaisema teos The Cathedral and the Bazaar. Kirjassa analysoidaan ohjelmistoprojektin dynamiikkaa, kun projekti toteutetaan vapaaehtoisten testaajien ja koodaajien voimin. Projektit noudattavat hyvin paljon samoja lainalaisuuksia kuin palkalliset hankkeet,

mutta taitavasti johdettuina tulokset ovat ratkaisevasti parempia, sekä ennen kaikkea kustannukset ovat huomattavasti vähäisemmät.

Helmikuussa 1998 Todd Anderson, Chris Peterson, John Hall, Larry Augustin, Sam Ockman ja Eric Raymond kokoontuivat miettimään keinoja vapaiden ohjelmien imagon parantamiseksi, sillä Stallmannin kiihkeä ideologinen linja ohjelmistojen vapauden puolesta sai yritykset ja sponsorit vierastamaan vapaita ohjelmistoja. Näin syntyi uusi järjestö Open Source Initiative, joka keskittyi lähdekoodin julkaisemisen pragmaattisiin puoliin. *Vapaat ohjelmistot* -termi korvattiin *Open Source* -termillä, joka on aatteellisesti neutraalimpi, ja korostaa ohjelmien lähdekoodin avoimuutta. Open Source -termistä käytetään myös käännösmuotoa *avoin lähdekoodi*.

3.2 Ketkä Open Source -projekteihin osallistuvat?

Open Source -ohjelmiston tuottamiseen saattaa osallistua satoja ohjelmoijia usealta mantereelta. Perinteisesti tällaisen projektiryhmän työskentelyä ei pidetä tehokkaana verrattuna kuluihin, jotka se aiheuttaa. Open Source -projektissa tilanne on toinen, koska kuluja ei juurikaan ole. Valta-osa ohjelmoijista tuottaa koodia maksutta. Heitä motivoi kiinnostus projektin kohteena olevaan tuotteeseen, mielenkiinto osallistua projektiin kaltaistensa kanssa, meriitit jotka lankeavat menestyksekkäille osallistujille tai oman nimen saaminen valmiin tuotteen credits -osioon. Todennäköisimmin vain projektin johtohenkilöt saavat palkkaa kyseisen tuotteen kehittämisestä. Taitavat johtajat saavat muodostettua jopa tuntemattomista vapaaehtoisohjelmoijista tehokkaasti toimivan projektiryhmän, kun ryhmää osataan kohdella oikein.

Tyypillisesti ohjelmistoprojektissa ryhmän tuottavuus laskee, kun osallistujamäärää kasvatetaan [Pressman 1997]. Ohjelmistojen ominaisuuksien lisääntyessä ja projektien kasvaessa ohjelmoijia kuitenkin tarvitaan kuitenkin yhä enemmän. Yksi projektinhallinnallinen keino jarruttaa tehokkuuden heikkenemistä on muodostaa useita rinnakkaisia kehitysryhmiä. Juuri näin Linux-ytimen kehitys toteutetaan nykyään. Projektin perustaja Linus Torvalds ei itse enää juurikaan ohjelmoi uusia osia, vaan keskittyy seulomaan uusista tuotoksista ne, joita tullaan hyödyntämään ytimen tulevissa versioissa. Kehitysryhmien koordinaattorit tekevät suurimman osan esivalinnoista ja Torvaldsille toimitetaan kunkin ryhmän käyttökelpoisimmat uudistukset.

3.2.1 Vapaaehtoisten kehittäjien johtaminen

Maksuttoman ohjelmoija- ja testausorganisaation mukaan saaminen on elinehto Open Source -projektin onnistumiselle. Eric S. Raymondin mukaan onnistunut projektin avaus on keskeistä koodaajien saamiseksi [Raymond 2000]. Ihmiset haluavat osallistua projekteihin, joista he ovat henkilökohtaisesti kiinnostuneita tai jotka he kokevat jostain syystä erityisen tärkeäksi. Raymond on menestyksellisesti koordinoinut useita Open Source -projekteja, kuten Emacs VC, Emacs editorin versionhallinta, ja Fetchmail -sähköpostiasiakasohjelma.

Ohjelman yleinen tarve ei kuitenkaan ole tae hankkeen etenemisestä. Richard Stallmannin GNU-projekti tuotti ison osan varusohjelmia, jotka yhdessä Linux-ytimen kanssa muodostavat Linux-käyttöjärjestelmän rungon. GNU-projektin eräänä tavoitteena oli tuottaa vapaa taulukkolaskentaohjelma, mutta syystä tai toisesta tämä projektin haara ei koskaan lähtenyt käyntiin toivotulla tavalla, vaikka taulukkolaskentaohjelmalla voidaan ajatella olevan suurikin yleinen merkitys. GNOME-projekti, jossa kehitetään Linuxin kanssa käytettävää graafista työpöytäohjelmistoa, kehitti myöhemmin erillisellä projektilla toimivan ja käyttökelpoisen Open Source -taulukolaskentaohjelman.

Raymond korostaa kirjassaan The Cathedral and Bazaar, että projektin käynnistäjältä vaaditaan hyviä johtajan ominaisuuksia ja karismaattista luonnetta. Tämä on helppo ymmärtää, mikäli ajatellaan projektin käynnistämistä eräänlaisena markkinointitapahtumana. Onnistuneen startin jälkeen projekti lähtee lumipallon tavoin kasvamaan ja pitää ohjelmoijat kiinnostuneina hankkeesta, koska päivityksiä ja uudistuksia tulee usein. Sen sijaan liian rauhallinen aloitus antaa vaikutelman, ettei projekti ole kiinnostava, joten ohjelmoijat vähentävät vierailujaan projektisivuilla, ja hanke saattaa hiljaisuudessa kuihtua kokoon. Ohjelmoijat, älykkäimmätkin, ovat ihmisiä ja useimpiin pätevät samat markkinoinnin lainalaisuudet kuin muihinkin. Useimmat haluavat työskennellä menestyksellisten projektien parissa, joissa on vetäjänä pätevät koordinaattorit, eikä työskentely hiljakseen etenevien ehkä marginaaliseen käyttöön jäävien tuotteiden parissa ole läheskään yhtä houkuttelevaa.

Nopealla syklillä tapahtuvat julkistukset toimivat sekä projektin elinkelpoisuuden indikaattorina että palkintoina kehittäjille. Koodaajat saavat nimensä heti muiden nähtäville, ja toistuvat maininnat tuovat ohjelmoijille meriittiä. Tällaisen ohjelmoijan on helpompi saada vastaisuudessa ideoitaan hyväksytyksi yhteisössä sekä mahdollisesti kerätä oma koodaajaryhmä itse perustamansa projektin ympärille.

Raymondin mukaan [Raymond 2000] projektin johtajan sosiaaliset taidot voivat olla ratkaiseva tekijä työn etenemisen kannalta. Käyttäjien kohtelu kehittäjäkollegoina on avain nopeaan koodin tuottamiseen ja virheiden jäljittämiseen. Kun käyttäjiä ja betatestaajia kohdellaan kuin he olisivat arvokkain kuviteltavissa oleva voimavara, he

alkavat vastaavasti toimia niin että nopeasti he ovatkin projektin johtajan arvokkain resurssi. Puolivalmiin ja virheitä sisältävän koodin julkaisemista ei tarvitse venyttää sen häpeän pelossa, että projekti tai koordinaattori joutuisivat naurunalaiseksi, sillä projektissa mukana olevat ohjelmoijat eivät tutki koodia paljastaakseen ohjelmiston kehnoksi, vaan päästäkseen parantamaan sitä.

Hyvin alkanut projekti saattaa aika ajoin osoittaa hiipumisen merkkejä. Potentiaalinen mahdollisuus on esimerkiksi silloin, kun ohjelmasta on julkaistu kohtuullisesti toimiva beta-versio. Suurimmat haasteet on voitettu ja jäljellä on entistä tarkempi testausvaihe ja virheiden paikallistaminen ennen kuin ohjelmasta voidaan julkistaa vakaa versio. Aktiivinen projektikoordinaattori julkaisee tässä vaiheessa tehtävälistoja ja määräaikoja niiden valmistumiselle. Vapaaehtoiset ohjelmoijat saattavat nyt mieluummin jättää loput työt muiden tehtäväksi kuin ottaa itselleen aikataulupaineita ilmaisprojektista.

Yksi keino tästä tilanteesta selviytymiseen on asettaa julkistamisajankohta tietylle päivälle ja ottaa silloin julkistettavaan versioon mukaan ne korjaukset, jotka ovat ehtineet valmiiksi. Näin tekee esimerkiksi Alan Cox, joka vastaa Linux-ytimen vakaiden versioiden julkistamisesta. Hän tekee julkistuksia melko säännöllisin väliajoin, mutta välttää lupaamasta etukäteen, mitkä virheet on korjattu tai mitä uusia ominaisuuksia on otettu mukaan seuraavassa versiossa.

Toinen tapa on lyödä lukkoon tehtävälista, mutta jättää sen julkistusajankohta auki. Tätä menetelmää käytetään tyypillisesti ytimen kehityksessä niin sanottujen experimental -versioiden julkistuksessa. Tutkittaessa projekteja [DeMarco and Lister 1987], joissa on käytetty tätä tapaa aikatalutuksessa, on todettu, että tulokset ovat sekä korkealaatuisia, että ne valmistuvat lyhyemmässä ajassa, kuin vastaavissa projekteissa, jotka on aikataulutettu joko "realistisesti" tai "aggressiivisesti".

3.2.2 Palkkakoodaajat Open Source -projektissa

Kaikki Open Source -tuotteet eivät ole vain vapaaehtoistyövoiman varassa. Tästä löytyy tuore esimerkki myös suomalaisesta IT-historiasta. Wapit Oy alkoi keväällä 1999 valmistella SMS/WAP gateway -ohjelmistoa. Gateway-ohjelmiston koodaus tulisi sitomaan useita kokopäiväisiä työntekijöitä kuukausien ajoiksi, vaikka projektin toteutus pohjautuisi Open Source -malliin.

Miksi liiketaloudellisin perustein toimiva yritys käyttäisi satoja tuhansia markkoja, ellei miljoonia, kehittääkseen tuotteen, joka valmistuttuaan annetaan ilmaiseksi pois? Yhtiön

perustaja ja ensimmäinen toimitusjohtaja Mato Valtonen [Valtonen 2001] perustelee asiaa kirjassaan, joka käsittelee yhtiön toimintaa.

Wapit Oy, 1998-2001, tuotti matkapuhelimiin liittyviä lisäarvotuotteita, erityisesti WAP-tekniikkaan pohjautuen (Wireless Application Protocol). Asiakkaita, potentiaalisia tai olemassaolevia, olivat matkapuhelinoperaattorit sekä mahdollisesti muut yritykset, jotka halusivat tarjota omille asiakkailleen sellaisia matkapuhelintekniikkaan perustuvia palveluita, joista asiakasyritys saa kilpailuetua omalla liiketoiminta-alueellaan. Wapit toimi koordinaattorina projektissa, jonka tavoite oli aikaansaada toimiva WAP/SMS Gateway-ohjelmisto (sms = short message system, perinteinen tekstiviestipalvelu gsm-puhelimissa).

Wapitin Kannel-projektin tarkoituksena oli julkaista gateway-ohjelmisto, joka olisi laadukas, monipuolinen ja ennenkaikkea maksuton. Kun matkapuhelinoperaattorit ottaisivat käyttöönsä Kannel-gatewayn, ainakin kaikki Wapit Oy:n tuottamat palvelut toimisivat näiden operaattorien verkossa. Ja koska lähdekoodi olisi vapaassa jaossa, kaikki lisäpalveluita tuottavat yritykset voisivat hyödyntää Wapitin kehittämää teknologiaa. Näin Wapit pääsisi ikäänkuin määrittelemään standardeja omalla toiminta-alueellaan.

Ohjelmistoa kehitettäisiin kuten basaarityyppisiä Open Source -ohjelmistoja yleensäkin, jolloin Wapitin ei tarvitsisi enää vastata kaikista ohjelmoijien palkkakuluista. Yritys saisi kuitenkin olla koordinaattorina linjaamassa kehityksen suuntaa. Näin olisi pienelläkin yrityksellä mahdollisuus nousta eturivin vaikuttajaksi, tosin muiden suostumuksella ja avustuksella.

Suurikin yritys voi hakea lisätehoa vapaaehtoisista ohjelmoijista. Nokia julkaisi helmikuussa 2002 osan tuotekehitysympäristönsä lähdekoodista. Koodi on kehitetty Nokian kustannuksella maksullisena ohjelmointityönä. Nyt Nokia vapautti koodin tarkoituksenaan kasvattaa sellaisten koodaajien määrää, jotka ohjelmoivat Nokian matkapuhelimien kanssa käytettäviä sovelluksia ja palveluita. Parhaimmillaan toimiessaan menetelmällä saadaan sidottua ohjelmoijia ympäri maailmaa Nokian standardeihin. Kääntöpuolella on koodin ajautuminen myös kilpailijoiden analysoitavaksi, mutta tästä koituvat haitat on ilmeisesti arvioitu pienemmiksi kuin saavutettavissa olevat edut.

Näiden esimerkkien tarkoitus oli kuvata, kuinka lähdekoodi vapauttamalla voidaan suunnitellusti hakea parempaa asemaa kaupallisilla markkinoilla. Sanapari -Open Source ei siis aina merkitse puoliboheemien koodaajien harrasteprojetia.

3.3 Open Source - yleinen määritelmä

Vaikka Open Source merkitsee avointa ohjelmakoodia ja vapauksia koodin uudelleen käyttämisessä, vaalitaan näitä vapauksia melko tiukoilla määritelmillä. Virallinen Open Source -ohjelmien määritelmä on saatavissa verkosta Open Source Initiativen sivuilta [Perens 2001]. Alla on Mikko Välimäen suomennos määritelmästä. Yritykset ja ohjelmistokehitysyhteisöt voivat luoda oman Open Source -lisenssointisopimuksen, jos katsovat sen sopivan kehitettävän tuotteen ominaisuuksiin paremmin kuin tarjolla olevat noin 30 sopimusmallia. Open Source Initiative tarkastaa ja antaa OSI -hyväksynnän (tässä Open Source Initiative) lisenssille, mikäli vastaavaa sopimusmallia ei ole ennestään olemassa eivätkä tarjotun lisenssin ehdot ole miltään osin ristiriidassa yleisen määritelmän kanssa.

Johdanto

Avoin lähdekoodi ei tarkoita vain lähdekoodin saatavuutta. Avoimen lähdekoodin ohjelmien täytyy myös täyttää seuraavat levitysehdot:

1. Vapaa levitysoikeus

Lisenssi ei saa estää ketään myymästä tai lahjoittamasta ohjelmaa osana yhdisteltyä ohjelmistoa, joka on koottu useista eri lähteistä saaduista ohjelmista. Lisenssissä ei saa määrätä ohjelman myymisen ehdoksi tälläisessä tapauksessa rojaltia tai muuta maksua.

> ***Perustelu**: Rajoittamalla lisenssiä siten, että vaaditaan vapaa levitysoikeus elinoidaan houkutus saada lyhyessä ajassa hieman myyntituloja monien pidemmän aikavälin etujen uhalla. Jos näin ei vaadittaisi, silloin yhteistyöhankkeilla olisi suuri vaara epäonnistua.*

2. Lähdekoodi

Ohjelman täytyy sisältää lähdekoodi ja ohjelman levityksen täytyy olla sallittu sekä lähdekoodina että käännetyssä muodossa. Jos jotakin osaa ohjelmasta levitetään ilman lähdekoodia, tällöin on selkeästi tiedotettava, miten lähdekoodi on saatavissa kohtuullisin kopiointikustannuksin - mieluiten Internetin kautta ilmaiseksi. Suositeltavin levitysmuoto on lähdekoodi, jota ohjelmoija voi muuttaa. Tahallisesti epäselvä lähdekoodi ei ole sallittu. Välimuodot kuten esiprosessorin tai kielen tulkin tulos eivät ole sallittuja.

***Perustelu**: Selkeän lähdekoodin saatavuus vaaditaan siksi, että ohjelmia ei voi kehittää ilman niiden muuttelemista. Koska tarkoitus on tehdä kehitystyö helpoksi, vaaditaan muutosten tekeminen helpoksi.*

3. Johdannaiset teokset

Lisenssin on sallittava muutosten tekeminen ja johdannaisten teosten luominen. Näitä on saatava levittää samoilla lisenssiehdoilla kuin alkuperäistä ohjelmaa.

***Perustelu**: Pelkkä mahdollisuus lukea lähdekoodia ei vielä riitä tukemaan ohjelman hajautettua arviointiprosessia ja nopeita kehitysvalintoja. Jotta kehitys olisi nopeaa, muutoksia on saatava kokeilla ja levittää.*

4. Lähdekoodin yhteenkuuluvuus

Lisenssi voi rajoittaa muutellun lähdekoodin levittämistä vain siinä tapauksessa, että lisenssi sallii korjaustiedostojen (patch) ja niiden lähdekoodin levittämisen. Korjaustiedostojen tarkoituksena on ohjelman muuttaminen, kun sitä käännetään. Lisenssin on nimenomaisesti sallittava muutetusta lähdekoodista käännettyjen ohjelmien levittäminen. Lisenssi voi edellyttää, että johdannaisissa teoksissa käytetään erilaista nimeä tai versionumeroa kuin alkuperäisessä ohjelmassa.

***Perustelu**:Vaikka rohkaiseminen parannusten tekemiseen on hyvä asia, käyttäjillä on oikeus tietää kuka on vastuussa ohjelmistosta. Vastaavasti tekijöillä ja ylläpitäjillä on oikeus tietää mihin heiltä pyydetään tukea sekä oikeus suojata heidän mainettaan.*

Niinpä avoimen lähdekoodin lisenssin täytyy taata, että lähdekoodi on helposti saatavilla, mutta se voi vaatia, että lähdekoodi tulee levittää alkuperäisenä muuttamattomana lähdekoodina ja korjaustiedostoina. Tällä tavalla "epäviralliset" muutokset voidaan tarjota saataville selvästi erotettuina niiden perustana olevasta lähdekoodista.

5. Henkilöiden ja ryhmien syrjinnän kielto

Lisenssi ei saa syrjiä ketään henkilöä tai henkilöryhmää.

***Perustelu**: Jotta avoimen lähdekoodin kehitysprosessista saataisiin suurin mahdollinen hyöty, tulisi siihen hyväksyä osallistumaan mahdollisimman monipuolisilta taustoilta tulevia henkilöitä ja henkilöryhmiä. Siksi kielletään sulkemasta ketään pois kehitysprosessista.*

Joillakin mailla, mukaanlukien Yhdysvallat, on vientirajoituksia tietyn tyyppisille ohjelmistoille. Avoimen lähdekoodin määritelmän mukainen lisenssi voi varoittaa lisenssin saajaa soveltuvista rajoituksista ja muistuttaa heitä velvollisuudesta noudattaa lakia; kuitenkaan itse lisenssiin ei saa kirjoittaa sellaisia rajoituksia.

6. Toimialojen syrjinnän kielto

Lisenssi ei saa syrjiä ketään käyttämästä ohjelmaa tietyllä toimialalla. On esimerkiksi kiellettyä rajoittaa ohjelman käyttöä liiketoiminnassa tai genetiikan tutkimuksessa.

***Perustelu**: Tämän kohdan tarkoituksena on kieltää lisenssiansat, joilla estetään avoimen lähdekoodin kaupallinen käyttö. Kaupalliset käyttäjät halutaan saada liittymään yhteisöön, ei tuntemaan olevansa siitä poissuljettuja.*

7. Lisenssin levittäminen

Ohjelmaan kuuluvien oikeuksien on sovelluttava suoraan kaikille niille, joille ohjelma on levitetty ilman, että heidän tulisi ottaa käyttöön myös jokin uusi lisenssi.

***Perustelu**: Tämän kohdan tarkoituksena on kieltää ohjelmiston sulkeminen epäsuorin keinoin kuten vaatimalla salassapitosopimusta.*

8. Lisenssi ei saa olla tuotekohtainen

Ohjelmaan kuuluvat oikeudet eivät saa riippua siitä, että ohjelma on osana jotakin tiettyä ohjelmistopakettia. Jos ohjelma erotetaan ohjelmistopaketista ja sen jälkeen sitä käytetään tai levitetään ohjelman lisenssillä, tällöin kaikkien niiden, joille ohjelma levitetään, tulee saada samat oikeudet kuin alkuperäisessä ohjelmistopaketissa.

***Perustelu**: Tämäkin kohta poistaa lisenssiansoja.*

9. Lisenssi ei saa rajoittaa muita ohjelmia

Lisenssi ei saa asettaa rajoituksia muille ohjelmille, joita levitetään lisensoidun ohjelman mukana. Lisenssi ei saa esimerkiksi vaatia, että kaikki muut ohjelmat, joita levitetään samalla tallennusvälineellä, olisivat avoimen lähdekoodin ohjelmia.

> ***Perustelu**: Avoimen lähdekoodin ohjelmistojen jakelijoilla on oikeus itse päättää omista ohjelmistaan.*
>
> *GPL-lisenssi on yhteensopiva tämän kohdan kanssa. Ohjelma joka on linkitetty GPL:n alla oleviin ohjelmakirjastoihin perii GPL:n vain, jos se muodostaa yhden teoksen eikä ole ainoastaan levitetty GPL-kirjastojen kanssa.*

3.4 Open Source -lisenssit

Erilaisia Open Source -lisenssejä on vähintään 30 erilaista, joille kaikille yleinen Open Source määritelmä asettaa reunaehdot esittämällä ohjelmistoilta edellytettävät yleiset vapaudet ja velvoitteet.

Open Source -lisenssit ovat vastakohta kaupallisten ohjelmien lisensseille ohjelmien käytön sallimisessa. Kaupallisten lisenssiehdojen tarkoitus on turvata ohjelmistojen valmistajalle kuuluvat tulot vastaamaan käytössä olevien ohjelmien määrää, ja tämä tehdään rajoittamalla ohjelmistojen luvatonta kopiointia ja levitystä. Open Source -lisensseillä vastaavasti pyritään mahdollistamaan ohjelmistojen tehokas leviäminen rajoittamalla ohjelmista perittävää maksua. Tietyin ehdoin molempia ohjelmatyyppejä voidaan käyttää sujuvasti jopa samoissa projekteissa. Tällöin Open Source -lisenssi on valittava tarkkaan, ettei avoin lähdekoodiosuus vie pohjaa koko projektin taloudelta.

Keskeiset eroavuudet erilaisten Open Source -lisenssien välillä ovat useimmiten tavassa, jolla kukin lisenssi määrittelee lisenssin alaisten ohjelmistojen yhdistämisen kaupallisiin tuotteisiin, sekä ehdoissa, jotka määrittävät lähdekoodiin tehtyjen muutoksien julkisuutta.

Open Source -mallisen ohjelmistotuotannon vahvuus on perustunut pitkälti kehittäjäyhteisön saamaan julkisuuteen tuotteiden ja korjausten julkistuksessa, eikä niinkään taloudellisiin etuihin. Käyttäjäkunnan intresseissä saattaa kuitenkin olla taloudellisen hyödyn saavuttaminen. Tähän käyttäjäkuntaan voi kuulua niin

yksityishenkilöitä, oppilaitoksia, eri kokoisia yrityksiä kuin myös julkisen hallinnon yksiköitä.

Avoimeen lähdekoodiin perustuvien ohjelmien kaupallista käyttöä ei suinkaan haluta tyrmätä. Kaupallisten käyttäjien toivotaan tuntevan olevansa oikeassa paikassa, eikä missään harmailla ohjelmistomarkkinoilla. Eri lisenssivaihtoehtojen lisäksi tätä tavoitetta palvelee yleisen määritelmän kohta 6.

Open Source -lisenssien vapauden vastakohtana voi pitää ohjelmistopatentteja. Patentoitua algoritmia, funktiota tai muuta ohjelmaa ei ole lupa käyttää omassa tuotannossa, vaikka lähdekielisen ohjelmakoodin saisi selville. Yritykset hakevat patentteja ohjelmilleen sekä suojatakseen tuotekehittylynsä tuloksia sekä estääkseen kilpailijaa hyödyntämästä samaa menetelmää. Yhdysvalloissa viranomaisten suhtautuminen ohjelmistopatenttihakemuksiin on ollut perinteisesti Eurooppalaista käytäntöä liberaalimpi, mutta viime vuosina on sama löyhä ohjelmistopatenttisuuntaus ottanut jo ensiaskeleensa Euroopassakin. Patentit hyödyntävät yrityksiä, joiden tuotteet saavat patenttisuojan. Muille ohjelmistojen kehittäjille kustakin myönnetystä ohjelmistopatentista on lähinnä haittaa.

3.4.1 Juridisia näkökulmia Open Source -lisensseistä

Eräs Open Source -malliin liittyvä tekijä on niin kutsuttu virusvaikutus (eng. virus effect). Termi on todennäköisesti otettu käyttöön, koska virusvaikutus -tilanteessa Open Source -ohjelman pätkä toimii kuin *troijan hevonen*. Termi, virus vaikutus, ei ole aivan täsmällinen, sillä troijan hevoseksi kutsutaan ohjelmaa, joka on tietoisesti hankittu, mutta pääasiallisen käyttötarkoituksensa lisäksi se sisältää ominaisuuksia tai toimintoja joista käyttäjä ei ole tietoinen, ja jotka toiminnot saattavat olla vahingollisia käyttäjän tietotekniikalle. Tämän määritelmän mukaan myös ohjelmointivirheet tekisivät ohjelmasta troijan hevosen, mutta yleensä troijan hevosista yhteydessä tarkoitetaan tarkoituksellisesti lisättyjä piilotoimintoja.

Vaikutus syntyy, kun avointa ohjelmakoodia tai sen muunneltua versiota hyödynnetään osana uutta ohjelmaa, ja näin tulee tämän uudenkin ohjelman lisensoinnin noudattaa avoimen lähdekoodin lisenssin ehtoja. Tässä vaiheessa syntyy tilanne, että avointa lähdekoodia hyödyntänyt taho saattaa joutua luovuttamaan omia immateriaalioikeuksiaan ja arvokasta lähdekoodiaan vapaaseen jakeluun korvauksetta. Muussa tapauksessa uuden ohjelman kehittäjätaho saattaa syyllistyä lisenssisopimuksen ehtojen ja aiempien tekijöiden oikeuksien loukkaukseen.

Ohjelmistojen vapauden vaalimisessa GPL-lisenssi (Gnu General Public Lisence) on eräs tiukimmista. Tyypillinen tämän lisenssin tuoma ongelma liittyy kaupallisen kirjaston käyttämiseen GPL-lisensoidun ohjelman yhteydessä, jolloin ohjelman levitys binaarimuodossa muuttuu kielletyksi. Tämän ongelman voittamiseksi kehitettiin LGPL-lisenssi (Gnu Library General Public Lisence tai nykyisin myös Gnu Lesser General Public Lisence), joka sallii kaupallisten kirjastojen linkittämisen Open Source -ohjelmaan.

Toinen käytännön ongelma saattaa olla työajan ja vapaa-ajan ohjelmoinnin erottaminen toisistaan. Valtaosa avointa lähdekoodia kehittävistä ohjelmoijista on ammattilaisia, jotka varsinaisessa palkkatyössään käsittelevät ja kehittävät samantyyppisiä ohjelmia, joihin he vapaa-ajallaan tuottavat ilmaiseksi julkisesta koodia. Tällöin saattaa syntyä tilanteita, jolloin työnantajan tuotteisiin liittyviä liikesalaisuuksia leviää kuin huomaamatta ulkopuolisten saataville.

Tämä tilanne on mahdollinen usealla eri tietotekniikan osa-alueella ja liittyy keskeisesti Open Source -mallin toimintatapaan. Esimerkiksi uuden oheislaitteen tullessa markkinoille laitteen mukana on tyypillisesti ajuriohjelmisto joillekin yleisimmin käytetyille käyttöjärjestelmille. Mikäli esimerkiksi Open Source -tyyppinen Linux, ei kuulu laitevalmistajan repertoaariin, saattavat innokkaat ohjelmoijat alkaa koodata Linux-yhteensopivaa ajuriohjelmaa.

Joskus laitevalmistajat julkistavat laitteesta ajuriohjelman kannalta keskeiset tiedot varmistaakseen laitteen mahdollisimman laajan leviämisen, kun omat resurssit eivät riittäisi ohjelmalliseen tukeen useammalle järjestelmälle. Näin päästäänkin usein hyvään lopputulokseen. Toimiva ajuriohjelma syntyy todennäköisesti nopeasti ja tarvittaessa siitä julkaistaan kehittyneempiä versioita, koska laitetta koskavat tiedot ovat useampien saatavilla ja näin myös ohjelmakoodi on helpommin tarkistettavissa.

Myös eri valmistajien ohjelmien yhteensovittamisessa tarvitaan rajapintatietoja. Ohjelmistovalmistajat voivat antaa rajapintatiedot jopa pahimmalle kilpailijalleen, jotta ohjelmien toiminnallisuus voitaisiin taata, ja näin samalla tuetaan oman ohjelman imagoa laadukkaana ja monipuolisesti yhteensopivana.

Jos valmistajalta ei saada laitteesta myyntipakkausta tarkempia tietoja, ajuriohjelman tai minkä tahansa muun ohjelman rajapintoja voidaan etsiä ns. reverse engineering -menetelmällä. Reverse engineering -menetelmässä ohjelman konekielisestä, ajettavasta versiosta rekonstruoidaan lähdekoodi. Reverse engineering -menetelmällä voidaan päästä kohtuullisen hyviinkin tuloksiin, esim. Intel-yhteensopivat prosessorit on alunperin kehitetty reverse engineering -menetelmällä. Työn onnistuminen ei kuitenkaan ole

varmaa ja menetelmä saattaa viedä kohtuuttomasti aikaa ja voimavaroja niin laitteiston kuin henkilöstön osalta.

Valmisohjelmien lisenssiehdot useimmiten kieltävät ohjelmien takaisin kääntämisen tai muun muodon muuttamisen. Tätä onkin noudatettava, mikäli tarkoituksena olisi ohjelman käyttäminen uuden tekijänoikeudellisesti itsenäisen saati sitten kilpailevan tuotteen kehittämiseen. Jos tarkoituksena on itsenäisen ohjelman ja muiden ohjelmien välisen yhteistoiminnan parantaminen, on takaisinkääntäminen sallittua. Edellä oleva on varsin yksiselitteisesti luettavissa tekijänoikeuslaista, mutta tuotteiden jatkokäsitttely ja niiden mukaanotto osaksi Open Source -ohjelmaa ei ole välttämättä ole itsestään selvää.

Tutkitaan hieman tekijänoikeuslakia [Tekijänoikeuslaki 1993].

> 11 b §
>
> Joka on laillisesti hankkinut tietokoneohjelman, saa valmistaa ohjelmasta sellaiset kappaleet ja tehdä ohjelmaan sellaisia muutoksia, jotka ovat tarpeen ohjelman käyttämiseksi aiottuun tarkoitukseen. Tämä koskee myös virheiden korjaamista.
>
> Se, jolla on oikeus käyttää tietokoneohjelmaa, saa valmistaa ohjelmasta varmuuskappaleen, jos se on tarpeen ohjelman käytön kannalta.
>
> Se, jolla on oikeus käyttää tietokoneohjelmaa, saa tarkastella, tutkia tai kokeilla tietokoneohjelman toimintaa niiden ideoitten ja periaatteiden selvittämiseksi, jotka ovat ohjelman osan perustana, jos hän tekee sen ohjelman tietokoneen muistiin lukemisen tai ohjelman näyttämisen, ajamisen, siirtämisen tai tallentamisen yhteydessä.
>
> Sopimuksen ehto, jolla rajoitetaan 2 ja 3 momentin mukaista tietokoneohjelman käyttöä, on tehoton.

Tässä sallitaan ohjelmien tutkiminen ja tarvittavien muutosten tekeminen, mikäli muutosten avulla ohjelmaa voidaan käyttää aiottuun tarkoitukseen. Tämän pykälän mukaan esimerkiksi omien lokalisointien teko on täysin sallittua.

> 11 c §

> Ohjelman koodin kopioiminen ja sen muodon kääntäminen on sallittua, jos nämä toimenpiteet ovat välttämättömiä sellaisten tietojen hankkimiseksi, joiden avulla voidaan saavuttaa yhteentoimivuus itsenäisesti luodun tietokoneohjelman ja muiden ohjelmien välillä, ja seuraavat edellytykset täyttyvät:
>
> 1) nämä toimenpiteet suorittaa lisenssinhaltija tai muu henkilö, jolla on oikeus käyttää ohjelman kappaletta, taikka heidän lukuunsa henkilö, jolla on siihen oikeus;
>
> 2) yhteentoimivuuden saavuttamisen kannalta tarpeellinen tieto ei aikaisemmin ole ollut helposti ja nopeasti 1 kohdassa tarkoitettujen henkilöiden saatavilla; sekä
>
> 3) nämä toimenpiteet rajoittuvat niihin alkuperäisen ohjelman osiin, jotka ovat yhteentoimivuuden saavuttamisen kannalta tarpeen.

Eli reverse engineering on erikseen mainiten sallittu nimenomaan yhteesopivuuden saavuttamiseksi muiden ohjelmien kanssa. Esimerkiksi tekstinkäsittelyohjelman tallennusformaatin selvittäminen on sallittua, jotta ohjelmalla tallennettuja asiakirjoja voidaan käyttää muussa ohjelmassa. Sen sijaan esimerkiksi ohjelman valikoiden rakentamiseen liittyviä rutiineja ei ole lupa tutkia.

> Tietoja, jotka on saatu 1 momentin säännösten nojalla, ei saa näiden säännösten nojalla:
>
> 1) käyttää muuhun tarkoitukseen kuin itsenäisesti luodun tietokoneohjelman yhteentoimivuuden aikaansaamiseen;
>
> 2) antaa muille, ellei se ole tarpeen itsenäisesti luodun tietokoneohjelman yhteentoimivuuden kannalta; eikä
>
> 3) käyttää ilmaisumuodoltaan huomattavassa määrin samanlaisen tietokoneohjelman kehittämiseen, valmistamiseen tai markkinoille saattamiseen taikka muuhun tekijänoikeutta loukkaavaan toimeen.

> Sopimuksen ehto, jolla rajoitetaan tämän pykälän mukaista tietokoneohjelman käyttöä, on tehoton.

Tämän perusteella selville saatuja tietoja voidaan hyödyntää oman tekstinkäsittelyohjelman tekemisessä niin, että oma tekstinkäsittelyohjelma kykenee lukemaan ja tallentamaan tutkitun ohjelman formaatissa.

Seuraava kappale, kappale 2), estäisi oman ohjelman julkaisemisen verkossa, mutta tallennusformaatin tunteminen on yhteensopivuuden kannalta ratkaiseva ominaisuus ja siksi ohjelman saa julkaista.

Kolmas kohta tuntuisi asettavan lopullisen esteen oman tekstinkäsittelyohjelman julkaisemiselle kaikkine tallennusformaatteineen, koska kyseessä on ilmaisumuodoltaan samanlaisen tietokoneohjelman kehittäminen eli ilmiselvästi kilpailevan tuotteen tekeminen. Tämä lainkohta ei kuitenkaan mielestäni ole välttämättä este tekstinkäsittelyohjelman julkistamiselle eikä ohjelmakoodin avoimelle julkaisemiselle Open Source -mallin mukaisesti.

Julkistaminen on tietojen antamista muille, joka on kiellettyä. Mikäli voidaan katsoa, että tekstinkäsittelyohjelma valikkoineen, komentoineen ja kaikkine muine ominaisuuksineen olisi kyetty tekemään ilman toisen ohjelman tutkimista ja tutkimalla saatujen tietojen käyttöä, voidaan katsoa että tallennusformaatin selvittäminen ja ominaisuuksien lisääminen itsenäiseen ohjelmaan ovat vain yhteensopivuuden kannalta merkittäviä, eivät ratkaisevia koko ohjelman valmistumisen kannalta. Näin selville saadut tiedot voidaan siis käyttää omassa tekstinkäsittelyohjelmassa ja tätä ohjelmaa voidaan kehittää Open Source -mallin mukaan.

Lisenssin haltija saa etsiä yhteensopivuuden mahdollistavia rajapintoja, mikäli niitä ei muuten ole saatavissa helposti ja nopeasti. Tätä laissa määriteltyä oikeutta ei voida sopimuksin rajoittaa, vaikka ohjelmiston valmistaja sitä yrittäisikin.

Mikäli takaisinkääntäminen ei onnistu, siihen tarvittavaa ohjelmistoa ei ole saatavilla tai edes olemassa eikä laitevalmistaja ole luovuttanut rajapintatietoja ulkopuolisten käyttöön, ohjelmoija voi ottaa lähtökohdaksi jonkin muun laitteen ajuriohjelman, johon lähdekoodi on saatavissa, ja pyrkiä tätä koodia muokkaamalla saamaan aikaan toimivan ratkaisun.

Tälläkin tavalla voidaan onnistua, mutta aina laitteiden toiminnot eivät arvaamalla aukene. Tässä vaiheessa saattaa joku laitevalmistajan henkilökunnasta tai lähipiiristä olla avulias tai muuten halukas pätevöitymään koodauspiireissä ja toimittaa kriittisen

koodinpätkän muiden saataville. Kilpailevan laitevalmistajan ohjelmistoasiantujalle jo muutaman parametritiedon näkeminen saattaa olla kullanarvoinen vinkki.

3.4.2 Patentit -lisenssoinnin toinen ääripää

Lähdekoodiin kohdistuvien immateriaalioikeuksien jakaantuminen ja mahdolliset työsuhdekeksintöihin liittyvät tulkintakysymykset voivat olla erityisen vaikeita avoimen lähdekoodin yhteydessä. Nämä epävarmuustekijät vaikuttavat muun muassa lisenssisopimusten pätevyyteen ja luovat siten riskejä myös kyseisen lähdekoodin lisenssinsaajille.

Tietokoneohjelma voidaan nykyään suojata hakemalla sille patentti. Patenttivirastot kieltäytyivät pitkään, aina 1980-luvun lopulle asti, hyväksymästä ohjelmapatentteja. Käytäntö muuttui enlantilaisen Vicom -tapauksen myötä [UK PatentOffice], jossa todettiin, että keksinnössä esitetty ohjelma olisi voitu toteuttaa myös loogisin piirein. Silloin se olisi kuulunut selvästi teknologian alaan ja ollut patentoitavissa. Kun se nyt oli toteuttu koodilla, ei ollut syytä evätä patenttisuojaa. Tämä tapahtui siitä huolimatta, että patenttilain 1 § sisältää nimenomaisen kiellon, jonka mukaan keksinnöksi ei katsota pelkästään löytöä, tieteellistä teoriaa tai matemaattista menetelmää, taiteellista luomusta, suunnitelmaa, sääntöä tai menetelmää älyllistä toimintaa, peliä tai liiketoimintaa varten taikka tietokoneohjelmaa, eikä tietojen esittämistä.

Universaalikoneesta lähtevän ajatuskuvion mukaan tämä sama koskee periaatteessa kaikkia tietokoneohjelmia, vaikka useissa tapauksissa fyysinen toteutus olisi käytännössä kuitenkin mahdoton. Silti Suomen, Euroopan ja Yhdysvaltain patenttivirastot myöntävät nykyisin rutiininomaisesti suojaa esim. tiedostojenhallinta- ja muistin allokointiohjelmille. Niiden määrä on kasvanut etenkin Yhdysvalloissa räjähdysmäisesti [Kemppinen 2001].

Tämä käytäntö on luonut tilanteen, jossa saattaa usein olla vaikea selvittää, mikä on suojattua ja mikä ei. Joskus yksinkertaisen tuntuinen ohjelma on saanut patentin kun taas hyvinkin omaperäinen ratkaisu on katsottu kuuluvaksi yleisesti tunnetun tekniikan tasoon. Vuonna 1999 Yhdysvalloissa myönnettiin 22 000 uutta ohjelmistopatenttia vuodessa ja patentteja kaikkiaan 150 000 vuodessa [Kemppinen 2001]. Hakijoina ovat kolmessa tapauksessa neljästä suuryhtiöt IBM, Fujitsu, Canon ja Microsoft, joista IBM lähes yhden kolmasosan osuudella kaikista.

Julkisuudessa käydyn keskustelun perusteella niin alan tutkijat kuin ohjelmistoteollisuus pitävät näiden patenttien uutuustutkimusta ylimalkaisena ja niiden olemassaoloa melko

jyräävänä. Ohjelmistopatenttien lukumäärän kasvusta huolissaan olevien mielestä lähes jokaisesta uudesta ohjelmistosta tai sovelluksesta löytyy jotain, mitä IBM, Microsoft ja kumppanit väittävät omakseen. Ajatus on hyvin vaikea todistaa oikeaksi, mutta se kertoo pienempien ohjelmistotalojen näkemyksen ohjelmistopatenttien kääntöpuolesta.

Perinteisemmiltä tekniikan aloilta tuttu menetelmä "blocking patent" on adoptoitu myös ohjelmistoteollisuuteen. Patentteja haetaan kilpailijoiden sovellusten käytön estämiseksi. Tämä ilmiö on erittäin ongelmallinen biotekniikassa mutta hyvin tuttu tietotekniikassakin.

Kemppinen täsmentää yleisen patenttisuojan ehtoja [Kemppinen 2001]:
"Patenttisuojan saamisen normaalit ehdot ovat keksinnön uutuus, keksinnöllisyys (non-obviousness) ja tekninen teho. Uutta ei ole mikään, mikä on aikaisemmin esitetty julkisesti tai tullut tunnetuksi esimerkiksi kirjallisuudessa. Keksinnöllinen ei ole sellainen laite, menetelmä tai käyttö, joka on olemassa olevaan tiedon tasoon (state of art) tukeutuvalle alan keskiverto ammattimiehelle itsestään selvä. Keksintö on täsmennettävä patenttivaatimuksissa, joita perinteisesti on tulkittu hyvin kirjaimellisesti ja ahtaasti."

Suuri riski liittyy mielestäni mahdolliseen kolmansien osapuolien immateriaalioikeuksien loukkaukseen. Mikäli avoimen lähdekoodin ohjelmaan on tarkoituksella tai epähuomiossa otettu osia kolmannen tahon suojatusta ohjelmasta, ovat vaaravyöhykkeessä kaikki, jotka kyseistä ohjelmaa käyttävät, jakelevat tai muokkaavat. Ohjelmistopatenttien määrän sekä ennen kaikkea niiden valvonnan ja hyödyntämisen kasvaessa riski on todellinen.

3.5 Maksuttomat ohjelmat verkossa

Internetin kautta voidaan maksuttomasti ladata hyvin paljon erilaisia ohjelmia ja ohjelmistotyökaluja. Kaikki nämä eivät kuitenkaan ole Open Source -tuotteita. Esimerkiksi suosittu internet- ja tietoliikenneohjelmien latausportaali Tucows luokittelee jakelemansa ohjelmistot seuraavasti:

Shareware – verkossa on ladattavissa ohjelman toimiva ja käyttökelpoinen versio. Ohjelmaa saa useimmiten koekäyttää maksutta rajatun ajan, jonka jälkeen se tulee rekisteröidä, mikäli haluaa jatkaa ohjelman käyttöä. Lisenssimaksun maksamalla ja rekisteröitymällä saa mahdollisesti jonkin lisäominaisuuden toiminnan aktivoitua, saa jatkossa tietoa ohjelman kehittymisestä ja uusista versioista ja ennen kaikkea käyttäjällä on nyt laillinen oikeus ohjelman käyttöön. Shareware-ohjelmien lisenssimaksut ovat tleensä joitakin kymmeniä euroja.

Freeware – näiden ohjelmien oikeudet ja toiminnat ovat shareware-ohjelmien kanssa muuten lähes vastaavat, mutta rahaa ei liikuteta mihinkään suuntaan. Tämän ryhmittelyn osalta Freeware-ohjelmat muistuttavat luonteeltaan eniten tutkielman kohteena olevien Open Source -ohjelmien lisenssipolitiikkaa. Freeware ohjelmien lähdekoodi ei yleensä ole saatavilla, mikä on ehdoton edellytys Open Source -ohjelmalle.

Demo – Myös kaupallisia ohjelmia voi ladata verkosta maksutta. Demoversioista puuttuu jokin keskeinen ominaisuus, kuten tallentaminen tai tulostaminen. Tulosteissa saattaa olla selvästi esillä oleva vähintäänkin häiritsevä merkintä, joka kertoo ohjelman olevan rekisteröimätön. On myös demoversioita, joiden kaikki toiminnot ja ominaisuudet ovat käytössä, mutta joiden käyttöä on rajoitettu joko aika- tai käyttökertaperusteisesti. Ohjelmaan ja sen toimintaan pääsee tutustumaan vapaasti ilman ostovelvoitetta.

Kun käyttäjä on rekisteröinyt ohjelman ja tarvittaessa maksanut lisenssimaksun, hän saa yleensä lisenssinumeron tai jonkin muun tunnuksen, jolla aktivoidaan ohjelman täydet käyttöoikeudet. Ohjelmaa ei yleensä enää asenneta tässä vaiheessa uudestaan koneelle.

Yllä mainitut maksuttomassa jakelussa olevat ohjelmatyypit eivät yleensä kuulu varsinaisen Open Source -mallin piiriin, mutta ne on esitelty tässä, koska niihin lukeutuu muun muassa erilaisia lisäosia (plug-in, add on), laiteohjaimia, ns. bugipäivityksiä käyttöjärjestelmiin ja muihin kaupallisiin ohjelmiin, sekä paljon muita pienehköjä huvi- ja hyötykäyttöön tarkoitettuja ohjelmia, joita monet tietokoneen käyttäjät maksutta lataavat verkosta koneelleen.

Kiteytetysti voidaan sanoa Open Source tarkoittavan ohjelmien luotettavuuden ja laadun kehittämistä, joka tapahtuu tukemalla riippumattomien tahojen mahdollisuudella tarkastaa lähdekoodia sekä lähdekoodin nopealla kehityksellä. Jotta ohjelma tulisi OSI-sertifioiduksi, sitä tulee jakaa sellaisin oikeuksin, jotka sallivat sen vapaan lähdekoodin lukemisen, levityksen, muokkauksen ja käyttämisen. Tässä OSI tulee sanoista Open Source Initiative [Perens 2001]. Se on yhteisö, joka toimii Open Source -määrittelyjen koordinoijana. Lyhenteen käyttäminen ei ole suositeltavaa, koska tietotekniikan sanastossa OSI on totuttu yhdistämään tietoliikenneprotokollien luokitteluun.

3.6 Open Source -ohjelmistokehitys

Open Source -projekti käynnistyy, kun projektin vetäjä julkaisee verkossa ohjelmistoprojektin perustiedot, siihen asti valmistuneen ohjelmakoodin, tietokantamäärittelyt ja muut tarpeelliset määrittelytiedot. Ohjelmasta kiinnostuneet

verkonkäyttäjät lataavat asennusmodulit omille työasemilleen, testaavat ohjelmaa ja mahdollisesti tutustuvat lähdekoodiin.

Tästä eteenpäin projekti elää, kun projektin vetäjät jatkavat ohjelman kehitystä kirjoittamalla lisää lisää lähdekoodia, integroimalla koodiin käyttäjiltä saatuja muutosehdotuksia sekä korjauksia, jotka tehdään saatujen virheraporttien mukaan.

Projektin kasvaessa vetäjät eivät välttämättä enää itse konkreettisesti ohjelmoi, vaan toimivat hankkeessa eräänlaisina ohjaajina tai koordinoijina. Johtaja-termi ei sovi heille luontevasti, koska projektiryhmän jäsenten työ perustuu vapaaehtoisuuteen eikä projektin perustajilla ole ohjelmoijiin määräysvaltaa.

Esimerkiksi Linux-käyttöjärjestelmän ytimen kehittämisessä, Linus Torvalds ei juurikaan itse tee ohjelmointia, vaan koodi on peräisin useammasta kehitysryhmästä, joiden toimintaa Torvalds koordinoi. Jos kehitysryhmillä tulee ristiriitaisia näkemyksiä päämääristä ja projektin tavoitteista, tällöin koordinaattorin tehtävä on toimia tulipalon sammuttajana ja tehdä linjaratkaisut. Myös Linuxin kehitystyön aikana kehittäjien näkemyserot ovat kuumentaneet ohjelmoijien tunteita niin, että projektin keskustelualueilla on nähty varsin aggressiivistakin dialogia. Ongelmat on ainakin toistaiseksi kyetty ratkaisemaan ja ytimen kehitys on jatkunut. Aina yksimielisyyttä ei löydy, ja tällöin kehitysryhmä saattaa jakaantua useampaan kehityshaaraan. Näin kävi esimerkiksi Bill Jolitzinin BSD-koodin pohjalta kehittämälle 386BSD-projektille, joka jakaantui FreeBSD-, OpenBSD- sekä NetBSD-projekteiksi kehitysryhmien erimielisyyksien vuoksi.

Projektien vetäjät päättävät, milloin muutoksin korjattu ohjelmaversio julkaistaan. Kehitystyön käydessä kiivaimpana uusia versiojulkistuksia tulee päivittäin, parhaimpina useita. Tämä on luonteenomainen ero Open Source -ohjelmien ja kaupallisten ohjelmien välillä. Open Source -ohjelmiin ilmestyy päivityksiä usein ja nopeasti, kun kaupallisten ohjelmien päivityssykli on useimmiten kuukausia. Ero näkyy selvimmin tietoturvan parantamiseen liittyvissä korjauksissa. Turvallisuusaukon löytymisen ja siitä tiedottamisen jälkeen korjaus on tavallisesti verkosta saatavissa jo muutamien tuntien tai vuorokausien kuluttua. Kaupallisten ohjelmien korjausversioita saa normaalisti odotella useita viikkoja tai kuukausia. Poikkeus tästä ovat virustorjuntaan erikoistuneet ohjelmistotalot, koska ne julkistavat uusia viruskuvaustiedostoja heti uusien virusten ilmestyttyä.

Tavallisen käyttäjän mahdollisuudet kaupallisten ohjelmien kehittämiseen ovat yleensä marginaaliset. Virheraportteja ja muutosehdotuksia voi toki esittää, mutta matka

toiveiden toteuttamiseen on huomattavasti pidempi kuin Open Source -mallissa, jossa on mahdollista tarjota konkreettisia korjauksia lähdekoodin avoimuuden ansiosta.

Projektien vetäjät hallinnoivat projektiin kuuluvaa materiaalia erilaisilla työkaluilla, jotka on tuotettu pääsääntöisesti Open Source -mallin mukaisesti. Todennäköisesti käytetyin näistä on CVS (Concurrent Versioning System). CVS on versionhallintajärjestelmä, joka toimii sekä lähdekoodin varastona että useiden samanaikaisten versioiden kehitysalustana. CVS:n avulla kehitystyö voidaan hajauttaa ympäri verkkoa. Rinnakkaisten kehitysvaiheiden tulokset yhdistetään palvelimelle, jossa tehdään samalla tarkastukset päällekkäisyyksien hoitamiseksi. Virheraporttien hallintaan ja virheiden seurantaan käytetään esimerkiksi Bugzilla ja GNATS -ohjelmia, jotka ovat syntyneet eri projektien yhteydessä ja sittemmin otettu laajempaan käyttöön.

Linux-ytimen kehitystyötä varten kehitettiin LXR-ohjelma, jossa hyödynnetään hypertekstimenetelmiä uusien kehittäjien kouluttamisessa ja projektiin tutustuttamisessa. LXR muuntaa Linux-ytimen lähdekoodin HTML-muotoon (Hypertext Markup Language) niin, että esimerkiksi funktioiden nimet toimivat myös linkkeinä funktioiden määritelmiin.

Open Source -ohjelmistoille on tyypillistä, että ne ovat käytettävissä useissa eri käyttöjärjestelmäympäristöissä. Tämä on mahdollista monipuolisen käännösympäristö-ohjelmiston ansiosta. Tällaisia ohjelmia ovat muun muassa autoconf, autoheader ja automake. Niiden avulla hallitaan ohjelmia, jotka voidaan kääntää useampaan eri ympäristöön.

Projektin koordinaarit julkaisevat ohjelmat useimmiten useassa eri muodossa. Käyttäjälle helpoin tapa ovat erilaiset asennusmodulit, jotka käyttäjä lataa omalle koneelleen, tekee helpon asennuksen ja alkaa käyttää ohjelmaa. Käyttäjälle työläämpi, mutta ohjelman levityksen kannalta yleiskäyttöisempi on lähdekoodimuotoinen levitys, jossa käyttäjä lataa projektin verkkosivuilta tarvittavat lähdekieliset modulit ja kääntää ne omassa laiteympäristössään. Eri käyttöjärjestelmä- ja laiteympäristöjen käännösohjelmat saattavat tehdä pieniä eroja lähdekoodin tulkinnassa ja siksi kääntämisen onnistumista varmistetaan usein attribuuteilla, jotka annetaan kääntäjälle käännöskäskyn yhteydessä. Kääntäjäohjelmien käyttäminen ja niiden määritteiden hyödyntäminen vaatii tietotekniikan tuntemista enemmän kuin keskimääräiseltä tietokoneen käyttäjältä voidaan edellyttää. Ohjelmistoprojektien osallistujille ja muille ohjelmointia enemmän tehneille käyttäjille ei lähdekielisten koodien kääntäminen ole vaikeaa.

4 KOKONAISKUSTANNUSTEN LASKENNASTA

Tässä luvussa kokonaiskustannuslaskentaa tarkastellaan useasta näkökulmasta. Kohdassa 4.1 tarkastellaan laskennan lähtökohtia.

Kaikki tietotekniikan päätökset tulisi kyetä perustelemaan taloudellisin argumentein. Osa valinnoista näyttäisi ensi tarkastelussa perustuvan inhimillisiin tekijöihin tai muihin tunnepitoisiin vaikuttimiin, mutta näidenkin ratkaisujen takaa on löydettävissä taloudelliset motiivit. Kohdassa 4.2 todetaan taloudellisuuden olevan tärkein päätöksentekoperuste hankinnoissa. Alakohdissa 4.2.1-4.2.4 ohjelmistoihin liittyviä taloudellisia aspekteja ja aloitetaan muodostaa yleistä kaavaa kokonaiskustannusten laskemiseksi.

Kohdassa 4.3 tutustutaan syvällisemmin kustannustekijöihin ja tarkastellaan miten tekijät sijoittuvat ohjelmiston elinjakson eri vaiheisiin. Kohdassa 4.4 ja alakohdissa 4.4.1 - 4.4.4 tarkastellaan kustannusten laskentaa vaikeuttavia ilmiöitä. Kohdassa 4.5 ja sen alakohdissa 4.5.1 ja 4.5.2 tarkastellaan aiempia tutkimuksia Open Source -tuotteiden kokonaiskustannuksista sekä esitellään eräs laskentamalli karkealla tasolla.

Suurin osa tietojärjestelmän kokonaiskustannuksista muodostuu käytönaikaisista, systeemin elikaaren tuotantovaiheen (t) kuluista. Näiden tunnistaminen ei ole vaikeaa, mutta niiden sovittaminen yleiseen laskentakaavaan on mutkikasta. Kohdassa 4.6 pohditaan käytönaikaisten kustannusten sovittamista kaavaan, jonka luominen käynnistyi kohdassa 4.2. Kohdassa 4.6 todetaan yleispätevän kaavan luominen mahdottomaksi ja pohditaan miten laskenta voidaan konkreettisessa ympäristössä tehdä.

4.1 Laskennan lähtökohdat vaihtelevat

Kokonaiskustannusten laskentaan ei toistaiseksi ole luotu yleispätevää mittaristoa tai laskentametodia, vaan laskentatapa vaihtelee tutkijoittain ja jonkin verran tutkimuskohteittain. Laskentamallien pohjana tulisi aina olla huolellinen esivalmistelu ja sovellusalueen määrittely, jotta voitaisiin päästä kokonaiskustannusten laskennassa mahdollisimman tarkoituksen mukaisiin ja paikkansapitäviin tuloksiin.

Laitteiden ja ohjelmistojen hankintahinnat, leasingkulut ja toistuvat lisenssimaksut on helppo sijoittaa kustannuslaskentakaavoihin, mutta arvioitaessa järjestelmän koko elinkaaren aikana kertyneita kustannuksia joudutaan ennustamaan tulevia tapahtumia.

Nämä ennakkoarviointiin pohjautuvat faktorit saavat aikaan vaihtelua eri laskentamalleissa. Joissakin laskelmissa lähtökohtana ovat esimrkiksi järjestelmän käytön aikana organisaatiosta ulos suuntautuvat rahavirrat, toiset laskelmat perustuvat tuottavuuden kasvuun ja sen aikaansaamiin säästöihin tai peräti tuottoihin. Varsinkin jälkimmäinen malli on liikeyrityksessä varsin vaikea määritellä, laskea ja todentaa aukottomasti oikeaksi useamman vuoden jaksolla eteenpäin katsottuna. Hintojen ja maksujen laskeminen ja laskentamallien tarkistaminen helpompaa kuin työn tehokkuuden arviointi, joka on kustannuslaskuissa vaikeimmin ennustettava tekijä [Kemerer 1987].

Kumpikaan edellä mainituista laskentatavoista ei ole lähtökohtaisesti väärä, mutta eri painotuksilla voivat tulokset olla hyvinkin eriävät samankaltaisista lähtöarvoista. Painotuksien mukaan vaihtelee esimerkiksi sen tulkinta, mitkä tulevat laitehankinnat katsotaan johtuvan uusien ohjelmistojen käyttöönotosta, mitkä hankinnat tai investoinnit taas luetaan välttämättömiin perusparannuksiin. Esimerkiksi syksyn 2001 aikana Microsoft laski kahden suomalaisen kaupungin tulevia tietotekniikkakuluja [Microsoft 2001a ja 2001b] ja päätyi esittämään molemmille kaupungeille edullisimpana ratkaisuna oman tuotevalikoimansa tuotteita. Ongelmalliseksi näiden laskelmien luotettavuuden arvioinnin tekee kuitenkin laskentamallien pysyminen pimennossa. Siksi on vaikea varmistua laskennan tulosten paikkansapitävyydestä.

Jos halutaan välttää ylimääräisiä spekulaatioita jälkikäteen, kaikki laskentaan liittyvät parametrit ja kertoimet tulisi julkistaa ja perustella.

4.2 Kaikkien IT-ratkaisujen pohjana taloudellisuus

Karrikoidusti sanottuna tietotekniikkaratkaisut ja hankinnat ovat useimmissa yrityksissä ja julkishallinnon yksiköissä vain "välttämätön paha" ja kustannuserä. IT-infrastruktuuri nopeuttaa useiden työtehtävien suoritusta, ja joissakin tehtävissä tietotekniikan käyttö ainoa kustannustehokas tapa hoitaa ne. Vaikka tietotekniikasta olisi tullut keskeinen toimintaedellytys ja tietotekniikan päätöksenteko lähenee yrityksen strategista johtamista, yksiköiden ydintoiminta voi olla peräisin satojen vuosien takaa, esimerkkeinä kirjastolaitos, verohallinto, sanomalehtien toimitukset tai pörssitoiminta. Näissä tietotekniikalla on huomattava mahdollistajan rooli, mutta edelleen se pysyy teknisenä tukitoimintona.

Kehittyneillä ratkaisuilla voidaan tuottaa asiakkaille palveluja nopeammin, monipuolisemmin tai tarkemmin kuin alkeellisilla työvälineillä, mutta toiminnan

lähtökohta on muu kuin tietotekniikan kehittäminen ja ylläpito. Toisaalta tietokoneohjelmien kehitystyötä tutkinut Roger S. Pressman esittää, että monilla aloilla ohjelmistoista on tullut kehitystä eteenpäin ajava voima [Pressman 1997]. Ohjelmistot ja tietotekniikan kehittyminen ovat toki edistäneet monien alojen kehittymistä ja uusien palvelujen syntymistä, mutta samaa voi sanoa myös autoista, lentokoneista, puhelimista tai telefax-laitteista.

Palvelujen käyttäjille tai asiakkaille on enemmän merkitystä sillä, mistä uutistoimistosta lehti saa juttuaiheita, ja miten niitä käsitellään, tai sillä millaisia yrityksiä hyväksytään noteerattavaksi julkisessa pörssissä ja mitä tietoja yritysten toiminnasta on saatavilla. Laitteistot, joiden läpi nämä tiedot ykkösinä ja nollina kulkevat, ovat hyvinkin toimiessaan teknisesti tärkeitä, mutta ainakin teoriassa täysin korvattavissa toisen valmistajan vastaavilla tuotteilla.

Yksiköiden sisällä saattaa työntekijöillä olla vapauksia päättää käytössään olevista työvälineistä. Työntekijällä voi olla mahdollisuus valita mieltymystensä mukaan eri laitevalmistajien tuotteita, hän voi valita ottaako kannettavan vai kiinteän työaseman, ja jopa käyttöjärjestelmän tai tekstinkäsittelyohjelman valinta voidaan joskus jättää käyttäjän tehtäväksi. Tällöinkään näiden vapauksien ei tulisi perustua työnantajan filantrooppiseen asenteeseen vaan edelleen kustannustekijöihin.

Vaikka yritykselle saattaa koitua ylläpidollisia lisäkuluja useamman eri ohjelmaversion käyttämisestä, työntekijän valitsema kannettava tietokone on todennäköisesti kalliimpi kuin vastaavan tehoinen pöytäkone, lasku voi silti jäädä pienemmäksi kuin siinä tapauksessa, että vapauksia ei myönnettäisi. Erilaiset työssä viihtymiseen ja työntekijöiden motivointiin liittyvät menetelmät ovat työnantajille arkipäivää varsinkin aloilla, joissa kärsitään ammattitaitoisen työvoiman vähyydestä. Kalliimmilla täsmäinvestoinnilla työntekijään pyrkii työnantaja ehkäisemään henkilöstön nopeasta vaihtumisesta seuraavia rekrytointikuluja ja tuotanto- tai palvelutason laskuja, jotka johtuvat osaavan työvoiman puutteesta.

Näiden esimerkkien tarkoitus on tuoda esiin, että yrityksen tai yksikön toiminnan kannalta IT-ratkaisujen tulisi olla arvovapaita. Yksilötasolla niillä saattaa olla muuta merkitystä työntekijälle, jotka merkitykset voivat muuntua taloudellisiksi jälleen yritykselle.

Yritysten mahdollisuudet tarjota valinnanvapauksia vaihtelevat suuresti. Osittain rajoittavana tekijänä ovat nimenomaan muut valitut tietotekniset ratkaisut, jolloin valinnan mahdollistaminen tuo kuluja, jotka katsotaan liian suuriksi suhteessa saavutettuun hyötyyn. Toisaalta jo ennen tietokoneiden tuloa työpaikoille, työntekijöillä

on ollut erilaisia toiveita, eikä jokaiselle kulmahuonetta haluavalle sellaista mitenkään voida osoittaa.

Myös useat yhteensopivuustekijät ovat derivoitavissa kustannustekijöiksi. Yhteensopivuus esimerkiksi tavarantoimittajan varastojärjestelmän kanssa tuottaa todennäköisesti positiivisia arvoja kustannuslaskennassa, jos tällä yhteensopivuudella saadaan tietokoneet keskustelemaan keskenään ja jollei tuotteiden saatavuuksien tarkistamiseen tai tilauksen tekemiseen tarvita ihmistyövoimaa molemmissa päissä. Myös erilaisten tiedostomuotojen yhteensopivuus todennäköisesti vähentää kokonaiskustannuksia, koska silloin ei tarvitse varautua henkilö- eikä teknisin resurssein erilaisiin konversioihin.

4.2.1 Yhteensopivuus kustannustekijänä

Yhteensopivuuden merkitystä arvioitiin yllä termillä *todennäköisesti*, koska yhteensopivuus ei ole tietotekniikan terminä eksakti. Mitä erilaisimmat elektroniset laitteet ja ohjelmat voidaan saattaa toimimaan keskenään vuorovaikutteisesti, jolloin niiden voisi sanoa olevan yhteensopivia. Kustannukset tosin voivat nousta kohtuuttoman suuriksi, jolloin käytännössä jokin muu ratkaisu on parempi.

Toisaalta esimerkiksi monilla työasemakäytössä olevilla toimistosovelluksilla voidaan asiakirjoja tallentaa useampaan tiedostomuotoon, vaikka oletustallennusmuoto olisi kussakin omansa. Tosin aina ei ohjelmalla A voida tallentaa suoraan ohjelman B oletusmuotoon, mutta A:lla voidaan tallentaa muotoon C, jota ohjelma B myös ymmärtää. Joskus välitallennusmuoto C on alkeellisempi formaatti kuin A ja B, jolloin asiakirjasta jää joitakin tarpeellisia ominaisuuksia pois. Tällöin voitaisiin sanoa, etteivät A ja B ole keskenään yhteensopivia ohjelmia. Jossain ympäristössä näitä pois jääneitä ominaisuuksia ei tarvita välttämättä koskaan eikä puutteen olemassaoloa ole siksi tiedostettu. Siinä ympäristössä ohjelmat A ja B ovat täysin yhteensopivia.

Edellä olevaa esimerkkiä voidaan tutkia matemattisesti. Kun käytetään ainoastaan yhtä toimistosovellusta, tämän valinnan tuoma komponetti kokonaiskustannuksiin on seuraava:
ts = toimistosovellus,
hh = hankintahinta,

$\text{kustannus}(ts_A) = \text{käyttäjät} * hh_A$

tai

kustannus(ts_B) = käyttäjät * hh_B .

Mikäli molemmat ohjelmat ovat käytössä ja A:n käyttäjiä on enemmän (esimerkiksi koska se on halvempi, laskennan kannalta sillä ei ole merkitystä) lasketaan toimistosovelluksen kustannusvaikutus näin:

kustannus(ts_{AB}) = käyttäjät * hh_A + $käyttäjät_B$ * (hh_B-hh_A)

Voidaanko puhua ohjelmien A ja B yhteensopivuudesta, mikäli on olemassa ohjelma D, jonka avulla voidaan tehdä täydellinen tiedostomuunnos A:n ja B:n välillä, jonka hinta on hh_D ja joka ohjelma tarvitaan jokaiselle konvertointeja tekevälle käyttäjälle. Koska A on standardiksi valittu tiedostomuoto, tarvitaan ohjelma D jokaiselle ohjelman B käyttäjälle. Näin yhteensopivuus saavutetaan hinnalla:

kustannus(ts_{ABD}) = käyttäjät * hh_A + $käyttäjät_B$ * (hh_B-hh_A+ hh_D)

Nyt tiedämme mitkä ovat kustannukset jompaa kumpaa tai molempia toimistosovelluksia käytettäessä. Näin voisimme käydä koko sovellusympäristön läpi ja laskea kaikkien tarvittavien sovellusten ja niiden vaihtoehtojen ja yhteensovittamisen kustannukset. Laskentaan voidaan lisätä myös kaikki laitteistot. Kunkin tuotteen hankintahinnat voidaan syöttää laskennan pohjaksi, ja komponentti D:n hinta annetaan, kun kuvitteellinen laskentaohjelma sitä kysyy.

Lopputuloksena saisimme edullisimman ja kalleimman mahdollisen yhdistelmän, sekä joukon muita vaihtoehtoisia yhdistelmiä, joiden hinta asettuu ääripäiden väliin. Hankintahintojen perusteella laskennallisesti saatu edullisin yhdistelmä saattaa olla myös kokonaiskustannuksiltaan edullisin, mutta ei välttämättä.

4.2.2 Tarkastelujakson pituus ja lisenssimaksut

Edellisessä kappaleessa selvitettiin minkälaiset ovat kunkin ohjelmistokombinaation kustannukset hankintavaiheessa. Kun järjestelmien kokonaisedullisuutta tutkitaan, tulee meidän tietää myös tulevien vuosien kustannuksista jotain. Vuosittaisia vaihteluita tuovat esimerkiksi ohjelmien käyttökustannukset, sillä lisenssipolitiikat vaihtelevat jonkin verran.

Kertamaksu

Joidenkin ohjelmien lisenssimaksua voisi verrata kirjan hankkimiseen. Tuotteita ostetaan n kappaletta, ja niistä maksetaan ostohetkellä n * ostohinta. Hankittaessa useita saman valmistajan ohjelmia hankintahintaan sovelletaan mahdollisesti määräalennuksia, jolloin voidaan käyttää myös *volyymilisenssi*-termiä. Hinta määräytyy käyttäjämäärän mukaan, eikä ohjelman käyttämiselle yleensä aseteta muita rajoituksia. Vastedes muita kustannuksia ei ole, ennen kuin ohjelmaan hankitaan maksullinen päivitys. Päivityksen hinta on yleensä p % ostohinnasta, p < 100. Lisenssissä ei ole sisäänrakennettuja velvoitteita päivitysten hankkimiseen. Tämä on valmisohjelmien perinteisin lisenssimaksumenetelmä.

Kertamaksu + vuosimaksu

Tätä lisenssikäytäntöä voisi verrata esimerkiksi golf-seuran jäsenmaksuun. Hankittaessa maksetaan lisenssimaksu ja vuosittain maksetaan vuosimaksu. Hinnoittelu perustuu tavallisimmin käyttäjien lukumäärään. Vuosimaksu, suuruudeltaan tässäkin p % lisenssimaksusta, sisältää yleensä päivitykset ja korjausversiot, jotka maksukauden aikana julkaistaan. Tunnetuin tähän malliin liitettävä ohjelmistotoimittaja on Microsoft. Microsoftilla on vuosien ajan ollut erilaisia hinnoittelutapoja ja lisenssivaihtoehtoja, jotka ovat perustuneet edellisen kappaleen mukaisiin yksittäislisensseihin sekä monipuoliseen valikoimaan volyymilisenssejä. Toukokuussa 2001 Microsoft ilmoitti uudesta hinnoittelutavasta, joka tulisi käyttöön lokakuussa 2001. Uusi malli yksinkertaisti kirjavaa volyymihinnoittelua ja oli luonteeltaan tyylipuhdas kertamaksu + vuosimaksu. Malli on hyvin selkeä, mutta suurimmat yritysasiakkaat esittivät tyytymättömyytensä kustannuksiin sekä vaihtoehtojen vähyyteen. Yksinkertaistettuna ehdot olivat seuraavat: vanhan volyymilisenssiasiakkaan tulee ensin päivittää olemassaolevat Microsoft-tuotteensa uusimpiin versioihin, minkä jälkeen sopimus voidaan tehdä. Vuosimaksun suuruus oli 29 % työasemasovelluksen ja 25 % palvelinsovelluksen alkuperäisestä lisenssimaksusta. Ne asiakkaat, jotka eivät tee sopimusta, menettävät ohjelmien päivitysoikeuden, ja maksavat ohjelmista täyden hinnan, kun haluavat niitä seuraavan kerran päivittää. Uuden mallin käyttöönotto on siirretty vuoden 2002 elokuuhun, joten sen toimintaa ei vielä voida arvioida.

Sovellusten ulkoistaminen

Asp-mallissa (Application Service Provider) asiakas maksaa ohjelmistosta sen käyttömäärän mukaan, kuten autoa vuokratessa tai sähköä käytettäessä, tai hän maksaa kiinteää kuukausi- tai vuosimaksua riippumatta käytön määrästä. Oleellista asp-mallissa on, että ohjelmien lisenssit säilyvät palvelun tarjoajan hallussa, toisin kuin *sovellushallintamallissa*, jossa ohjelmien hallinta on myös ulkoistettu, mutta palvelun ostaja omistaa lisenssit. Asp-mallista on viime vuosina keskusteltu runsaasti alan

julkaisuissa, mutta käyttöön malli ei ole levinnyt kovin laajasti. Markkinointitutkimusyhtiö IDC:n selvityksen mukaan sovellusten ulkoistamismarkkinat olivat vuonna 2001 Suomessa 80 miljoonan euron arvoiset, joista asp-mallin mukaisia sopimuksia oli 13 miljoonan euron arvosta [Lagus 2002].

Edellä mainittujen mallien lisäksi käytössä voi olla näiden erilaisia yhdistelmiä sekä joitain harvinaisempia lisenssimaksutapoja.

Aiemmin aloitettuun kustannuslaskentakaavaan voidaan lisätä muuttujan vh (vuosimaksu) jolla huomioidaan lisenssimaksujen muuttuminen. Aiemmin olemme laskeneet systeemin elinkaaren (r) -luokkaan kuuluvaa kustannusta, nyt otamme mukaan (t)-luokan kustannukset. Muuttuja hh edustaa edelleen hankintahintaa, joka joissakin lisenssimalleissa voi olla nimellä kiinteä aloitusmaksu.

Nyt kahden ohjelman yhteiskäytön kustannukset lasketaan seuraavasti:

$t\_kustannus(ts_{ABD}) =$ käyttäjät * hh_A + $käyttäjät_B$ * (hh_B-hh_A+ hh_D) + ($käyttäjät_A$*vh_A + $käyttäjät_B$*vh_B + $käyttäjät_D$*vh_D) * t ,

jossa t on tarkasteluaika vuosina.

Seuraavaksi otetaan laskennassa huomioon ohjelmien A ja B mahdolliset olemassa olevat lisenssit. Jotta vanhat lisenssit voidaan erottaa uusista hankittavista lisensseistä, otetaan käyttöön uusi muuttuja *uk* (uudet käyttäjät), joka kertoo uusien lisenssien määrän. Aiemmin käytetty muuttuja, käyttäjät, kertoo kaikkien tämän tuotteen käyttäjien määrän, uusien ja vanhojen. Käyttäjät-muuttuja kirjoitetaan jatkossa kaavaan lyhenteellä *kä*. Kaava on nyt muodossa

t_kustannus(ABD) = ukA*hhA + ukB * (hhB-hhA+hhD) + ($kä_A$*vh_A + $kä_B$*vh_B + $kä_D$*vh_D) * t .

Kaavan muodostaminen aloitettiin tutkittaessa kahden kilpailevan sovelluksen yhteiskäytön kustannuksia, ja tähän tarkoitukseen laajennettukin kaava toimii. Pienin muutoksin voimme soveltaa sitä minkä tahansa kahden ohjelman tai ohjelman ja laitteen yhteiskäytön kustannusten laskentaan, esimerkiksi käyttöjärjestelmän ja tekstinkäsittelyohjelman tai palvelinohjelman ja kytkimen tai reitittimen laskentaan;

yhteiskäyttö(ABD) = ukA*hhA + ukB * (hhB-hhA+hhD) + ($kä_A$*vh_A + $kä_B$*vh_B + $kä_D$*vh_D) * t .

Edellä on käsitelty kahden tuotteen yhteiskäytön kustannusten laskentaa. Myös yksittäisen ohjelmiston kustannuksia voidaan myös laskea, lähinnä hankintahintojen mukaan, sillä käytännössä ohjelmat ovat osa järjestelmäkokonaisuutta ja aina suhteessa muihin komponentteihin. Samoja kaavoja voidaan käyttää yksittäisenkin ohjelman kustannusten arviointiin, jolloin muuttujan B ja D kertoimet jäävät nolliksi.

Kokonaiskustannusten selvittämisessä järjestelmän osien yhteensopiminen ja yhteistoiminta muodostavat verkon. Laskennan tekijä etsii siitä heikkoja lenkkejä ja kohtia, joihin rasitus lähitulevaisuudessa keskittyy. Käyttökelpoisten tulosten saamiseksi on järjestelmäkomponenttien keskinäisiä kustannusvaikutuksia joka tapauksessa tutkittava, joten yhteistoimintanäkökulma voidaan ottaa heti alusta lähtien laskennan lähtökohdaksi.

Tarkastelujaksolla on oma merkityksensä myös käytönaikaisten kustannusten arvioinnissa. Laitteistoille, ja varsinkin niiden liikkuville mekaanisille osille on syytä ennustaa jonkinlaisia huolto- tai ylläpitokustannuksia. Kiintolevyjen valmistajat ilmoittavat levyilleen MTBF -luvun (Mean Time Between Failure). Luku kertoo kuinka pitkään saman malliset kiintolevyt valmistajan testien mukaan keskimäärin toimivat moitteettomasti. Luku on suuntaa antava, koska se on laskennallisen arvioinnin tulos. Se on keskiarvo joka perustuu todellista käyttöikää paljon lyhyempiin testijaksoihin.

Yksittäisen työasemakoneen käyttäjän näkökulmasta kiintolevyt kestävät usein riittävän monta vuotta: tietokone ehtii vanhentua ennen kiintolevyn vikaantumista, ja kiintolevyjen rikkoontumista suurempi ongelma on niiden hidastuminen ja täyttyminen. Palvelinkäytössä olevien koneiden kiintolevyt ovat raskaammassa käytössä kuin yksittäisen käyttäjän koneen kiintolevyt. Levyihin kohdistetaan luku- ja kirjoituspyyntöjä mahdollisesti taukoamatta läpi vuorokauden. Palvelimissa on usein RAID-levyjärjestelmä, jolloin levyjä on lukumääräisesti useita samassa koneessa. RAID-levyjärjestelmiä (Redundant Array of Inexpensive Disks) käytetään järjestelmän korkean käytettävyyden ylläpitämiseksi tai tehokkaampien levyoperaatioiden aikaansaamiseksi.

Kiintolevyt ovat käytännössä kulutustavaraa ja niiden vikaantuminen on arkipäivää järjestelmän ylläpitohenkilöstölle. Palvelinkäytössä on syytä varmistaa järjestelmän vikasietoisuus kiintolevyn rikkoontumisen varalta sekä se, että mahdollisen alasajon jälkeen järjestelmä kyetään palauttamaan tietosisällöltään alasajoa edeltävään tilaan.

Palvelinkäytössä olevien kiintolevyjen ikä tulee otta huomioon kustannuslaskennassa niin, että levyn käyttöiän pitkittyessä käyttökustannukset todennäköisesti kasvavat.

4.2.3 Palveluluokat laskennan perustaksi

Kaikkia mahdollisia tai olemassa olevia järjestelmän osia ei ole tarpeen eikä hedelmällistä ottaa keskinäiseen vertailuun tai laskentaan. Esimerkiksi kaupungin taloushallinnon ohjelmistolla ei välttämättä ole kovin paljon yhteistä koulujen opetusohjelmien kanssa, tai kaupunginkirjaston palvelutasolla ei ole yhteyttä terveyskeskuksen ohjelmistoratkaisuihin

Yrityksen, kaupungin tai valtionviraston toiminnat tulee jakaa palveluluokiksi, jolloin matalan tason laskenta eli sovelluskohtainen laskenta tehdään ensisijaisesti palveluluokan sisällä. Seuraavalla tasolla lasketaan eri palveluluokkien yhteistoiminnasta koituvat kustannukset.

Palveluluokat saadaan analysoimalla koko toimintaympäristö ja kuvaamalla palvelutarpeet ja muodostamalla tuloksista palveluluokat. Palvelutarpeella tarkoitetaan sitä tarvetta, joka osastoilla ja yksittäisillä työntekijöillä on suhteessa koko IT-järjestelmään. Esimerkiksi toimistotyöntekijän voisi ajatella tarvitsevan tekstinkäsittelypalvelua, taulukolaskentapalvelua, mahdollisuutta olla sähköpostiyhteydessä kollegojen ja asiakkaiden kanssa, mutta todennäköisesti hän tarvitsee myös tulostuspalveluja ja tehtävän mukaisia arkistopalveluita.

Ajatusta IT -infrastruktuurin jakamiseen laskennassa pienempiin osiin ja osien välisien kustannusten selvittämistä tukevat Rain [Rai et. al. 1997] tutkimukset. Hänen mukaansa mittauksissa saadaan huomattavia etuja jaottelemalla IT-kustannukset pienempiin toimintoihin ja sovelluksiin. Tosin tällöin IT-kulujen suhde kokonaisuutena yrityksen muihin kuluihin ei tällöin hahmotu yhtä selkeästi kuin jos IT-kustannuksia käsiteltäisiin isompina kokonaisuuksina. Rai tutki IT-panostusten suhdetta yrityksen tuottoihin ja niissä tutkimuksissa IT-kustannuksia voidaan käsitellä myös isompana kokonaisuutena. Kun tutkitaan ainoastaan IT-kustannuksia sinänsä, eikä niiden suhdetta organisaation budjettiin tai tuottojen kehittymiseen, tarkka jako ei aiheuta vaaraa perspektiivin hämärtymisestä.

4.2.4 Peruspalvelut tarvitsevat tukiresursseja

Peruspalvelut mahdollistetaan ohjelmisto- ja laitteistopohjaisin ratkaisuin. Työntekijälle todennäköisesti hankitaan työasema, joka on mikrotietokone tai päätelaite. Työasemat oletetaan olevan yhden työntekijän käytössä; yhteiskäyttötapauksia ei tässä oteta huomioon. Yhteiskäyttö on mahdollista, mutta marginaalista. Esimerkiksi valtion virastoissa ja laitoksissa oli vuoden 2000 lopussa 147 383 henkilökohtaista työasemaa.

Työasemien määrä suhteessa henkilöstöön oli 1,2 työasemaa/henkilö [Valtiovarainministeriö 2001a].

Tarvittavat ohjelmistot asennetaan joko työaseman kiintolevylle tai ne sijaitsevat palvelimella, josta ne tarvittaessa latautuvat työaseman muistiin. Mikäli valitaan palvelinpohjainen ratkaisu, työasema tarvitsee myös lähiverkkoliitännän. Lähiverkkoa tarvitaan myös tulostukseen, ellei jokaiseen työasemaan liitetä omaa tulostinta suoraan kiinni. Viimeistään sähköpostipalveluiden järjestäminen vaatii lähiverkoa. Teoriassa vaihtoehtoinen ratkaisuja on, esimerkiksi jokaiseen työasemaan asennettava suora yhteys ulkopuoliseen sähköpostipalvelun tarjoajaan, mutta tämä vaihtoehto ei ole realistinen kompleksiivisen rakenteensa sekä alhaisen hinta/tehokkuus suhteen vuoksi.

Lähiverkkoa siis tarvitaan tai ainakin voidaan hyödyntää monen palvelutarpeen tyydyttämisessä. Näin lähiverkkopalvelusta tulee palvelut mahdollistava jaettu resurssi. Palveluja mahdollistavia jaettuja resursseja on muitakin, kuten erilaiset sovellus-, tietokanta-, hakemisto- ja muut palvelimet ja tietoliikenteen struktuurit.

Nämä tukiresurssit ja erityistarpeisiin hankittavat sovellukset ja palvelut muodostavat sen IT-järjestelmän, jonka kokonaiskustannuksia tässä tutkielmassa pyritään hahmottamaan. Tukiresurssit ja erityistarpeiden sovellukset muodostavat perinteisen muna-kana -ilmiön. Järjestelmiä päästään harvoin rakentamaan täysin nollapisteestä, vaan yleensä on otettava huomioon olemassa olevien ratkaisujen rajoitteet ja mahdollisuudet. Tukiresurssit luovat puitteet, joiden mukaan suunnitellaan hankittavat sovellukset ja muut uudistukset ja laajennukset. Toisaalta tukiresurssit on hankittu ja hankitaan tarpeiden mukaan, nykyisten ja tulevien. Palvelujen määrä ja laatu luovat siis tarpeet tukiresursseille.

Tukiresurssien oikea mitoittaminen on tietohallinnon haasteellinen tehtävä. On ymmärrettävä toimintaympäristön palvelutarpeet, ja investointien tekniset ja taloudelliset reunaehdot. On myös pystyttävä jossain määrin ennustamaan tulevaa. Esimerkiksi erilaiset hajautetut tai keskitetyt ratkaisut, ja niiden toimivuus ja taloudellisuus näyttävät vuonna 2002 varmasti erilaiselta kuin kymmenen tai 15 vuotta aiemmin. Tietoliikennekapasiteetit ovat kasvaneet samaan aikaan kun tiedonsiirron kustannukset ovat tulleet alas. Sama kehitys on tapahtunut tallennusvälineille sekä suorittimille ja työmuistille. Ohjelmistojen hinnat ja henkilöstökulut sen sijaan eivät ole kehittyneet samansuuntaisesti.

1992		***2001***	
Kiintolevy AT-IDE 426 Mt	1908 euroa	Kiintolevy ATA 80 Gt	370 euroa
Ethernet RJ-45 keskitin, 9 porttia	1181 euroa	Ethernet RJ-45 keskitin, 8 porttia	59 euroa

1992		*2001*	
Microsoft Windows 3.11 lisenssi + levykkeet	103 euroa	Microsoft Windows XP Pro	217 euroa

Taulukko 2. Muutamien tietotekniikkatuotteiden hintakehitys vuosina 1992-2001

Taulukon 2 esimerkkiin on valittu kolme tuotetta, jotka edustavat vertailun molempina vuosina käytössä olleita tietotekniikan artikkeleita. Tuotteiden hinnat ovat peräisin tuona aikana ilmestyneistä julkaisuista ([Mikrolog1992] ja [Tietokone2001]). Hinnat ovat niissä julkisesti esillä, ne eivät ole tarjousten perusteella saatuja. Kaikki hinnat sisältävät arvonlisäveron 22 %, ja markkahinnat on muunnettu euroiksi kertoimella 5.94573 ilman indeksikorotuksia.

Kiintolevyjä käytetään kaikissa palvelimissa ja useimmissa työasemissa. Vertailussa oleva keskitin on 10 Mbit/s nopeudella toimivan IEEE 802.3 -standardin ethernet-lähiverkon komponentti. Ethernet-tyyppinen verkko on kirjoitushetkellä käytetyin lähiverkkoratkaisu. Mikäli uusissa myytävissä tietokoneissa mainitaan olevan verkkosovitin, se tarkoittaa käytännössä aina RJ-45 -liitännällä olevaa ethernet-verkkosovitinta. Tietotekniikan vähittäismyyntihinnastoissa ei muihin lähiverkkoratkaisuihin juurikaan enää tarjota tuotteita. Ethernet-verkosta on nykyään nopeampiakin vaihtoehtoja, mutta 10 Mbit siirtonopeuden lähiverkkoja käytetään edelleen. Microsoft Windows on 90-luvulla vakiinnuttanut asemansa työasemien graafisena käyttöliittymänä. Esimerkiksi vuonna 2000 oli valtionhallinnossa 148 000 henkilökohtaista työasemaa, ja näistä 88 prosentissa oli jokin Microsoftin graafinen käyttöjärjestelmä [Valtiovarainministeriö 2001a].

Tässä valossa on täysin ymmärrettävää, että kilpailevia tuotteita halvemman tai ilmaisen komponentin hyödyntämismahdollisuudet on tutkittava.

Tässä katsannossa on lähes paradoksaalista, että käytännössä yritysten arvo nousee niiden hankkiman IT-pääoman mukaan. Yhdysvalloissa tutkittiin [Brynjolfsson and Yang 1997] 1000 yrityksen pörssiarvon kehitystä kahdeksan vuoden aikana. Tutkimus osoitti, että jokainen yrityksen IT-omaisuuteen sijoitettu dollari nosti yrityksen arvoa neljä kertaa enemmän kuin perinteiseen omaisuuteen sijoitettuna. Tämän arvioidaan johtuvan sijoittajien uskosta yrityksen tuottavuuden parantumiseen kehittyneen tietotekniikan avulla, sekä joukosta erillisiä tekijöitä, jotka tietotekniikan yhdistämänä synergisesti kohottavat yritysten kilpailukykyä ja osaamista.

4.3 Kokonaiskustannuslaskennan menetelmistä ja tekijöistä

David ja kumppanit [David et. al. 2002] jakaa järjestelmän tco-laskennan tekijät varsin käytännön läheisesti, joskin painottaen hallintakustannuksia, jotka on vielä jaettu kahteen lohkoon, ylläpito- ja käyttökustannukset. Tätä jakoa käydään tässä tutkielmassa yksityiskohtaisemmin läpi, koska varsinkin hallintakustannusten osuus tuo esiin tekijöitä, jotka konkreettisesti aiheuttavat kustannuksia pääosin systeemin elinkaaren (t), (e), (h) ja (p) -luokissa. Näin luokiteltuna laskennan lähtökohtana on järjestelmän käyttäminen ja ylläpito sinänsä sekä kustannukset, jotka elinkaaren aikana syntyvät. Jako on tasapuolinen, koska se ei sisällä komponentteihin liittyviä oletuksia. Pienemmissä tai hyvin spesifisissä hankkeissa, kuten tämän tutkielman esimerkkiprojektissa, voitaisiin joitain lähtöoletuksia tehdä. Tämä voidaan katsoa sallituksi, jos hanke selvästi sijoittuu jonkin sellaisen palveluluokan sisään, joka on tunnistettu koko järjestelmän määrittelyssä.

Taulukossa 3 on eritelty kokonaiskustannuksiin vaikuttavia tekijöitä sekä esimerkkejä niiden kustannuksista.

Kustannuskategoria	*Kustannustekijä*	*Esimerkki*
Hankintakustannus	Laitteisto	Näytöt, keskusyksiköt, palvelimet
	Ohjelmat	Käyttöjärjestelmät, tietokannan hallintaohjelmat, toimistosovellukset
Ylläpitokustannukset	Keskittäminen	Erityislaitteistot (kuten älykkäät itsediagnosoivat komponetit, jotka ongelmatilanteessa välittävät tiedon verkon ylläpidolle) ja ohjelmistot (kuten hakemistopalvelut ja työpöydän hallinta järjestelmät) joita tarvitaan, jotta voidaan saada aikaan ja ylläpitää keskitetty järjestelmä. Tukihenkilöstö on koulutettava näiden järjestelmien käyttöön.
	Standardointi	Lähtökohtaisesti, heterogeeninen laitteisto ja ohjelmisto tulee korvata laitteilla ja ohjelmilla, jotka noudattavat valittuja standardeja. Käyttäjät saatetaan joutua kouluttamaan standardiohjelmistojen käyttöä, ja standardilaitteet saattavat maksaa ei-standardeja enemmän.
Käyttökustannukset	Tuki	Tukihenkilöstö joko omasta talosta tai tukisopimus tarvitaan huolehtimaan esiin tulevista laitteisto- ja ohjelmisto-ongelmista

Kustannuskategoria	*Kustannustekijä*	*Esimerkki*
	Evaluointi	Uusia versioita ja päivityksiä julkaistaan jatkuvasti niin ohjelmista kuin laitteista. Ennen uutuuksien hankintaa ja asennusta on tutkittava, toimivatko ne oletetusti ja ovatko ne yhteensopivia olemassa olevan IT-ympäristön kanssa.
	Asennus / päivitys	Kun uusi teknologia on tutkittu, se on asennettava ja siihen on tehtävä päivitykset. Laitteisto- ja ohjelmistopäivitykset liittyvät usein toisiinsa; uusi ohjelmisto vaatii yleisesti tehokkaampia laitteita pakottaen laitepäivityksiin.
	Koulutus	Koulutus auttaa käyttäjiä saamaan enemmän irti työasemistaan. Koulutus voidaan toteuttaa kahdella tavalla: luokkahuonekoulutuksena tai uuden ohjelman itseopiskeluna. Ohjelmisto- ja laitteistopäivitykset edellyttävät normaalisti jossain määrin loppukäyttäjien koulutusta.
	Ei käytössä -aika	Ei käytössä -aikoja tulee ohjelmisto- ja laitevirhetilanteiden vuoksi, mutta myös ohjelmistojen ja laitteistojen päivitysten aikana. Kun järjestelmä kaatuu, organisaation kustannukseksi tulevat toimimaton järjestelmä, toimeton työntekijä sekä ne kaikki tarvittavat toimet, jotka toiminnan palauttamiseksi.
	Hukkakäyttö	Gartner Groupin Bill Kirwin määrittelee termin hukkakäyttö (futz factor) seuraavasti: "yhtiön tekniikkaa käytetään henkilön omiin tarpeisiin". Tämä kustannus ei ole seurausta järjestelmästä sinänsä, vaan ajasta, jona työntekijä käyttää järjestelmää työhön liittymättömien asioiden hoitoon.
	Seuranta	Tämä on kustannus, kun pidetään kirjaa organisaation teknisestä omaisuudesta. Tietokoneet liikkuvat paljon, erityisesti isoissa yhtiöissä. Jotta voidaan määritellä, mikä omaisuus millekin osastolle kuuluu, on jonkinlaista kirjanpitoa pidettävä.
	Virukset	Virukset kasvattavat kokonaiskustannuksia kahdella tavalla: ne voivat tuhota tietoja, joiden uudelleen luominen on kallista tai ne voivat aikaansaada tietokoneen täydellisen romahtamisen tuottaen *ei käytössä -aikaa*.
	Sähkön kulutus	Joidenkin arvioiden mukaan yhden työaseman kustannus sähkön kulutuksessa on 240 dollaria vuodessa. Lisäksi tietokoneet kehittävät lämpöä, mikä voi lisätä ilmastoinnista aiheutuvia kustannuksia.

Taulukko 3. Esimerkkejä kokonaiskustannuksiin vaikuttavista tekijöistä

Taulukossa on useita kustannustekijöitä, jotka voidaan sijoittaa kahteen tai jopa kolmeen elinkaariluokkaan. *Keskittäminen*, *standardointi* ja *evaluointi* ovat työvaiheita, jotka on ainakin osittain toteutettava jo vaiheessa (r). Tämä koskee sekä räätälöityjä ohjelmistoprojekteja että valmisohjelmien hankintaa. Vaiheen (t) kustannuksia kertyy kohdista *tuki*, *asennus/päivitys*, *koulutus*, *seuranta* ja *sähkön kulutus*. Huoltojaksokson (h) kustannuksiksi voidaan lukea puhtaasti *ei käytössä - aika* sekä osittain

asennus/päivitys osioiden kustannukset. Ennakoivan huollon (e) luokkaan voi lukea kuuluviksi kohdat *keskittäminen*, *standardointi* ja *evaluointi*. Poisto-luokkaan (p) voidaan katsoa kuuluvaksi *standardointi* sekä *evaluointi*. Viruksiin liittyvää tekijää ei sijoiteta mihinkään systeemin elinkaariluokista, sillä taulukossa 3. Davidin ja kumppanien kirjaaman kuvauksen mukaan se sopii pikemminkin *tuki*- tai *ei käytössä aika* osioiden yhdeksi alakohdaksi. Mikäli tuotannon keskeyttämistekijöitä haluttaisiin listata, muita olisivat esimerkiksi tietoliikennekatkot, sähkökatkot tai muut puutteellisesta tietoturvasta johtuvat katkot kuin virukset. Nämä ovat kuitenkin normaalin tuotantotoiminnan uhkiin kuuluvia tekijöitä, joihin on *tuki*-osiossa varauduttava.

Hukkakäyttö -tekijän (futz factor) olemassaolo tunnustetaan yleisesti, mutta käsitykset sen merkityksestä järjestelmän kokonaiskustannuksiin vaihtelevat. Gartner Group laskee omassa tco-laskentamallissaan jokaisen työntekijän tuottavan tunnin viikossa hukka-aikaa, mikä korottaa järjestelmän kokonaiskustannuksia [Dryden 1998]. Tuon ajan työntekijä käyttää työnantajan tekniikkaa omiin tarkoituksiinsa, esimerkiksi pelatessaan peliä, suunnitellessaan perheen aikatauluja tai säätäessään tietokoneen työpöydän asetuksia. American Airline lentoyhtiön teknologisen suunnitteluyksikön toimitusjohtaja Jim Fitzgerald on sitä mieltä, että työntekijät käyttävät yhtiön tietokoneita osaltaan tuottamattomasti, mutta työnantajan aikaa voidaan hukata ilman tietokoneitakin. Fitzgerald sanoo hukka-ajan aiheuttamien lisäkustannusten olevan seurausta pikemmin työnjohtamiseen liittyvistä tekijöistä kuin tietojärjestelmistä. Yhdysvaltalainen vakuutusyhtiö General Accident Insurance Co. tekee tco-laskelmansa sekä hukkakäytön huomioiden että ilman hukkakäyttöfaktoria, koska he eivät ole vakuuttuneita tekijän merkittävyydestä [Dryden 1998]. Gartner Groupin tutkija Bill Kirwin korostaa, että tco-laskenta on yritysjohdon työkalu, eikä pelkästään IT-yksikköön kuuluva asia, ja siksi hukkakäyttö tulee aina huomioida.

Kirwin on varmasti oikeassa sanoessaan tco-laskennan kuuluvan yritysjohdon intresseihin, mutta toisaalta tco-laskennalla ei ole tarkoitus tutkia organisaatioiden yleistä tuottavuutta, vaan keskittyä IT-ratkaisujen kustannuksiin. Hukkakäyttö-tekijän mukaan ottaminen laskelmiin olisi helpompaa, mikäli olisi käytössä tutkimustietoa ohjelmien tai järjestelmien sellaisista piirteistä, jotka nimenomaisesti houkuttelevat ylimääräisten asioiden hoitoon työnantajan laitteilla. Internet-liittymän voidaan ajatella olevan tällainen huomionvarastaja, mutta on hyvin vaikea arvioida mikä painoarvo sille voidaan antaa.

Taulukko 3 toimii hyvänä pohjana, kun halutaan laskea käytönaikaisia kustannuksia. Kokonaisvaltaisemmassa kustannusten laskennassa tulisi hankintakustannuksia avata enemmän. Tietotekniikan hankkiminen tai hallinta voidaan nykyään rahoittaa hyvin monin vaihtoehdoin. Perinteisin on tuotteiden maksaminen suoraan kassasta toimittajan

laskun mukaan. Tällöin hankintakustannukset painottuvat ensimmäiselle vuodelle. Seuraavina vuosina kustannukset kertyvät käytöstä ja ylläpidosta.

Voidaan kysyä, mitkä ovat hankintakustannukset, kun järjestelmän maksamista varten otetaan laina, joka maksetaan takaisin joidenkin vuosien kuluessa tai kun koko järjestelmä tai sen osa hankitaan leasing-rahoituksella. Yrityksen kirjanpidossa nämä ensimmäisen vuoden ja seuraavien vuosien kustannukset on helpompi sijoittaa oikealle momentille, mutta mihin ne kuuluvat kokonaiskustannuslaskelmissa.

Ovatko vuosittaiset kulut nyt käyttökustannuksia vai tulisiko kaikki hankintaan liittyvät kustannukset merkata ensimmäisen vuoden hankintakustannuksiksi? Kumpikaan mainituista toimintatavoista ei palvele laskentaa kustannusten eriyttämisessä ja selkeyttämisessä. Kumpikin menetelmä saattaa antaa oikeita lukuja kokonaiskustannuksista usean vuoden tarkastelujaksolla, mutta vuositason kustannusjakauma on vääristynyt. Tästä voi päätellä, että myös hankintakustannukset on kyettävä tarvittaessa jakamaan kaikkien niiden vuosien rasitteeksi, joihin niillä todellisuudessa on vaikutusta.

4.4 Kokonaiskustannuslaskennan yleiset heikkoudet

Välineet IT-kustannusten laskemiseksi ovat tärkeitä kaikkien organisaatioiden taloushallinto- sekä tietohallintojohdolle. Ilman erityisiä laskentavälineitä voidaan ehkä selvittää erilaisin tyytyväisyyskyselyin jotakin tuotettujen palvelujen laadusta, mutta ilman vertailevaa kustannuslaskentaa jää arvailujen varaan olisiko samat palvelut voitu toteuttaa edullisemmin.

Laskentamallin valinta ja oikeiden parametrien valinta ja arvottaminen edellyttävät asiantuntevan henkilöresurssin käyttämistä jo suunnittelu- ja määrittelyvaiheessa. Näiden vaiheiden ylimalkainen käsittely kostautuu laskennan lopputulosten käyttöarvon heikentymisenä.

Vaikka määrittelyt tehtäisiin erittäin tarkasti ja huolellisesti, ovat laskennan lopputulokset enemmän tai vähemmän arvioihin perustuvia lukuja. Tietotekniikan kehityksen myötä tietotekninen infrastruktuuri muuttuu, ja uusia ennustamattomia kuluja tulee budjetin rasitteeksi. Toisaalta joillakin tietotekniikan osa-alueilla hintakehitys ollut laskeva jo vuosia, mikä on kompensoinut uusien innovaatioiden vaatimia lisäresursseja.

4.4.1 Ihminen on koneiston heikko lenkki

Hankintahinnat ja rahoituskulut voidaan laskea mekaanisesti, mutta inhimillisten komponenttien vaikutus tuo laskelmiin epävarmuutta. Laskennan pohjatyöksi on sovellusympäristö huolellisesti analysoitava ja kuvattava formaaliin muotoon, jotta laskenta olisi mahdollisimman kattava.

Määrittelyssä tulee ottaa huomioon käytettävät tarvittavat palvelut ja niiden tuottamiseen käytettävät sovellukset sekä laitteet ja käyttäjät. Käyttäjien valmiuksien kartoitus ja huomioon ottaminen laskelmissa muodostavat suuren haasteen.

On inhimillistä, että IT-tukiorganisaatio yliarvioi kykynsä vastata uusien järjestelmäkomponenttien asennuksesta, ylläpidosta ja koulutuksesta. Myös kykyjen aliarviointi on mahdollista, mikäli ei selkeästi nähdä uusien tuotteiden tuomaa etua kokonaisuudessa. Tällöin muutokset ja lisäykset kasvattavat työtaakkaa.

Perustoimistosovelluksen käyttäjän kyky siirtyä uuden sovelluksen käyttäjäksi saatetaan aliarvioida. Esimerkiksi tekstinkäsittelyohjelman vaihtaminen toiseen voidaan nähdä liian suurena harppauksena. Tekstinkäsittelyohjelmat ovat kymmenen viime vuotta olleet perustoiminnoiltaan varsin samankaltaisia. Kaikissa on graafinen käyttöliittymä, toiminnot valitaan alasvetovalikoita tai painikkeita käyttämällä, valitaan muotoiltava tekstialue ja valitaan haluttu muokkauskomento, kaikki yleisimmät tekstinkäsittelyohjelmat kykenevät esittämään muokattavan tekstin näytöllä jo tulostusasussa jne. Kun ohjelman vaihdon aiheuttamasta alkusäikähdyksestä on selvitty, eivät varsinaisessa käytössä vastaantulevat ongelmat ole välttämättä sen suurempia kuin saman valmistajan ohjelmistopäivityksestä olisi koitunut.

4.4.2 Arvioiden tekemistä ei voida välttää

Edellisissä kohdissa etsittiin kaavan avulla hinnoiltaan edullisinta ohjelmisto- ja laiteyhdistelmää. Tässä vaiheessa tiedetä kuitenkaan mitään yhdistelmän tehokkuudesta, laajennettavuudesta tai joustavuudesta. Käsittelemättä ovat myös kustannukset, jotka liittyvät yhdistelmien ylläpitoon, kehitykseen, käyttäjien koulutukseen ja yhdistelmän käytettävyyteen ylipäätään.

Nämä luvut eivät yleensä saatavilla järjestelmää hankittaessa. Tekniikan kehittymisen tuomat uusien innovaatioiden implementoinnit, niiden kustannukset tai tarpeellisuus on jollain tavoin arvioitava. Myös teknisten komponenttien hintakehitykseen on otettava

kantaa, sillä keskenään kilpailevat ratkaisut saattavat perustua erilaisiin teknisiin perusratkaisuihin, jolloin joudutaan arvioimaan vaikkapa edullisuuden pysyvyyttä.

Koulutuksen vaikutuksen arviointi on hyvin vaikeaa. Koulutus rasittaa organisaatioita sekä taloudellisesti että toiminnallisesti. Koulutuksen tarvetta ja kustannuksia ei voida tarkasti määritellä sen paremmin IT-henkilöstön kuin käyttäjienkään osalta kovin kauas etukäteen. Organisaation palvelutaso oletettavasti laskee koulutuksen ajaksi, ainakin mikäli koulutus tapahtuu työaikana. Lisäksi olisi arvioitava kuinka pian uuden järjestelmän käyttöönoton jälkeen palvelutaso on palannut entiselleen tai missä vaiheessa ennen uudistusta vallinnut palvelutaso ylitetään.

4.4.3 Laskentakaava voidaan kalibroida

Kaikkien muuttujien suorien ja välillisten kustannusten toteutumista ei todennäköisesti pystytä ennustamaan täsmällisesti edes yhden vuoden jaksolla, saati sitten pidemmällä aikavälillä. Kustannustekijöiden painotuksia ja kertoimia arvioitaessa nousee huolellisen pohjatyön merkitys jälleen esiin. Kohdealueen tunteminen ja erityispiirteiden huomioon ottaminen tuovat oikeaa perspektiiviä arviointiin, kun asetetaan arvoja arvioperustaisille muuttujille.

Kohdealueen erityispiirteiden huomioon ottamisen merkitystä tukee Kemererin [Kemerer 1987] tutkimus ohjelmistoprojektien kustannuslaskentamallien toimivuudesta. Neljän laskentamallin keskimääräinen virhe oli 500-600 % toteutuneiden projektien vaatimista työtunneista. Kohdealueen mukaan kalibroituna parhaat mallit kykenivät kuvaamaan 88 prosentin tarkkuudella projektin resurssivaatimukset.

Arvioinnissa tulisi käyttää muista vastaavista organisaatioista kerättyä tietoa ja yhdistämällä se kohdealueen tietoihin. Näin malli saadaan kalibroitua kohdealueeseen eikä yleistä laskentamallia sovelleta sokeasti. Ottamalla laskentaan mukaan usean organisaation tiedot voidaan esimerkiksi ohjelmistoprojektien kustannukset ennustaa luotettavammin, kuin pitäytymällä vain tutkimuksen kohteesta kerättävän tietoon [Briand et. al. 1999].

4.4.4 Kustannusten jakautuminen ei ole yksiselitteistä

Tässä kappaleessa tarkastelemme valtionhallinnon tietotekniikkakulujen kehittymistä vuosien 1995 ja 2000 välisenä aikana ja teemme joitakin tulkintoja kustannusten taustoista. Tässä ei ole tarkoitus etsiä virheitä kulujen kirjaamisesta tai henkilöstön

toiminnasta, vaan käyttää raporttia esimerkkinä siitä, millaisia ongelmia on kulurakenteiden syiden ja seurausten selvittämisessä jälkikäteen, jos käytössä on suurpiirteiset tiedot. Esimerkin läpikäyminen on hyödyllistä, sillä luotaessa kustannusten ennakointimallia, saatetaan kohdealueen edellisten vuosien toteutuneita lukuja ainakin jossain määrin hyödyntää parametrien valinnassa ja täsmentämisessä. Summat on muunnettu muuntokertoimella 5.94573, raportin [Valtiovarainministeriö 2001a] alkuperäisistä markkamääräisistä hinnoista euroiksi. Raportin tiedot perustuvat vastauksiin, jotka valtion virastot ja laitokset antoivat valtiovarainministeriön maaliskuussa 2001 toimeenpanemaan kyselyyn. Lukuihin ei sisälly ministeriön alaisuudessa toimivien valtion liikelaitoslain mukaisten yksiköiden tietoja.

Kyselyyn vastanneiden virastojen ja laitosten palveluksessa oli 31.12.2000 yhteensä 119 986 henkilöä 2 505 toimipisteessä. Tiedot näistä on saatu 131 kohdeorganisaatiolta. Kyselyyn vastasivat kaikki ministeriöt sekä käytännössä kaikki valtion virastot ja laitokset. Vuonna 2000 kyselyyn jätti vastaamatta yksi organisaatio valtionhallinnosta.

	Miljoonaa euroa					
	1995	1996	1997	1998	1999	2000
Palkat ja palkkiot	67.1	74.8	78.5	113.3	119.4	125.1
Laitevuokrat ja leasingmaksut	3.6	1.2	2	4.4	5.5	9.5
Palvelujen ostot	66.4	81.3	110.2	134.9	148.5	164.6
Tiedonsiirtopalvelut	22.6	25	26.1	29.9	32.9	38.6
Laiteostot	63.4	75	81.8	78.1	89.7	78.4
Ohjelmistomenot	23	25.5	29.5	26.5	27.8	27.4
Muut menot	11.1	12.8	12	11.1	12	14.6
Yhteensä	**257.2**	**295.6**	**340.1**	**398.2**	**435.8**	**458.2**

Taulukko 4. Valtionhallinnon tietotekniikkatoiminnan menot menolajeittain vuosina 1995-2000

Taulukosta 4 voidaan nähdä, että osa kustannuksista on useita vuosia ollut vakiintuneella tasolla euromääräisesti, kuten *laiteostot* ja *ohjelmistomenot*. Näiden suhteellinen osuus kokonaiskustannuksista on kuitenkin tarkasteluvuosien aikana laskenut. Laiteostojen suhteellinen osuus on supistunut eniten, 7.5 prosenttiyksikköä kuuden vuoden aikana.

Tarkastelujakson 1995-2000 aikana internetin käyttö ja internet-valmiudet ovat kasvaneet kaikkialla. Tämä selittänee osaltaan kasvaneita *tiedonsiirtopalvelut*-kustannuksia. Tiedonsiirtoon käytettävän tekniikan hinta, samoin teleoperaattorien veloitukset tiedonsiirtoyhteyksistä, ovat yleisesti laskeneet samanaikaisesti, niin Suomessa kuin

ulkomaillakin. Tiedonsiirtopalvelujen kustannukset kasvoivat noin 170 % kuudessa vuodessa. Oletettavasti samasssa ajassa ovat vielä enemmän lisääntyneet ne käyttäjät, jotka palveluja kykenevät hyödyntämään sekä palvelujen laatu, etenkin yhteysnopeuksien kehittyminen. Vuoden 2000 lopussa valtiohallinnon työntekijöistä yli 80 prosentilla oli käytössään sekä sähköposti- että www-palvelut. Käytettävissä ei ollut vertailua varten vuoden 1995 tietoja tietoliikenneohjelmista ja palveluista, joita käyttäjillä oli.

Laiteostoihin on käytetty tarkastelujakson aikaan joka vuosi lähes sama rahasumma. Tämän voi tulkita johtuvan siitä, että tietotekniikkalaitteiden hinnat ovat koko 1990-luvun laskeneet. Samalla rahalla on siis saatu hankittua enemmän tietotekniikkaa eli valtionhallinnon laitekanta on kehittynyt koko 1990-luvun loppupuolen ajan. Henkilökohtaisten työasemien määrä on lisääntynyt 79 700 laitteesta 147 000 laitteeseen ja palvelinten määrä 3 700 koneesta 6 800 koneeseen.

Sen sijaan ohjelmistokustannusten pysyminen 1990-luvun puolenvälin tasolla on hieman yllättävää, koska ohjelmistojen hinnat eivät ole laskeneet. Lukujen valossa voisi tulkita, että joko ohjelmistopalveluiden tarjonta on heikentynyt vuosi vuodelta tai valtio on onnistunut hankkimaan tarvittavan ohjelmiston vuosittain entistä edullisemmin ehdoin. Lisäksi on selvää, että ohjelmistopäivitykset menevät *ohjelmistomenot*-momentille, mutta raportista ei käy selville, onko mikrotietokoneiden mukana tulleet ohjelmistot laskettu ulos laitekustannuksista ja kirjattu ohjelmistomenoihin vai onko paketissa mukana tulleiden ohjelmien hinnat laiteostot -rivillä.

Kustannuslajin *palvelujen ostot* -kustannukset ovat kasvaneet eniten niin euroissa kuin suhteellisesti. Tähän kustannusluokkaan menevät todennäköisesti henkilökunnalle ostetut ulkopuoliset IT-koulutuspalvelut, asiantuntija-arviot ja -selvitykset sekä tilapäisen henkilöstön henkilöstön vuokraukset. Tiedossa ei ole, sisältyykö tähän ulkoa ostettuja tukipalveluita tai sellaisia määrittely- tai laskentatöitä, joita tarvitaan esimerkiksi uuden tietojärjestelmän pohjatiedoiksi. Mikäli tällaisia kustannuksia on sisällytetty *palvelujen ostot* -kohtaan, voidaan miettiä mitkä kuluista voisi sijoittaa paremminkin *ohjelmistomenot* -otsikon alle.

Vuonna 2000 organisatioilta tiedusteltiin arvioita kustannuksista, joita euroon siirtymisen vaatimat tietojärjestelmämuutokset aiheuttaisivat. Vastauksen antaneiden yksikköjen kulujen yhteissumma oli 31,3 miljoonaa euroa. Tietojärjestelmiin tarvittavat muutokset ovat suurelta osin todennäköisesti ohjelmistomuutoksia, -päivityksiä ja -varmistuksia. Mikäli kaikki arvioidut kustannukset ovat toteutuneet arvioidusti vuonna 2001 ja ne oli merkitty ohjelmistoja koskeviksi kuluiksi, ja muut ohjelmistoa koskevat kulut pysyisivät edellisen vuoden tasolla, kaksinkertaistuisi *ohjelmistomenot* -momentti yhden vuoden

aikana pysyttyään vuosia vakiintuneesti huomattavasti alemmalla tasolla. Jos valuuttamuutoksen aiheuttamat kustannukset on sijoitettu *palvelujen ostot* -kohtaan, on tulkittava, että raportissa ja sitä edeltävässä tutkimuksessa tähän kohtaan on sijoittu yleisestikin ohjelmistotyöt, jotka tekee jokin ulkopuolinen maksullinen taho.

Raportissa tuli esiin myös, että 125 organisaatiolla oli olemassa verkkosivut ja viidellä sivut olivat tekeillä. Jonkinlainen verkkolomake oli 84 organisaatiolla käytössä ja 22:lla lomake oli valmisteilla. Varsinaista asiakaspalvelua verkossa tarjosi 39 yksikköä ja valmisteilla olivat 33 yksikön asiointipalvelut. 31 vastaajan maksullisia tuotteita tai palveluita oli mahdollista hankkia internetin kautta ja 18 vastaajaa valmisteli myynnin aloittamista. Näistä hankkeista on kertynyt määrittely- ja suunnitteluvaiheessa kustannuksia, myös toteutus ja palveluiden ylläpito on aiheuttanut kustannuksia. Luontevia kululajikohteita mainituille kustannuksille ovat siis ainakin *palkat ja palkkiot*, mikäli omaa henkilökuntaa on osallistunut johonkin edellä mainituista vaiheista. *Ostettuihin palveluihin* menee helposti kaikkien vaiheiden kustannuksia, jos ulkopuolista apua on käytetty. *Laitevuokrat ja leasingmaksut* tai *laiteostot* imaisevat palvelinten hinnat, mikäli omia palvelimia on tarkoitukseen hankittu. Vaikka kyse on mitä suurimmassa määrin ohjelmistoprojekteista, ei *ohjelmistomenoihin* ole välttämättä kohdistettu mitään kuluja, - silti ei ole rikottu kirjanpitosäännöksiä.

Jos halutaan tutkia aiempien ohjelmistovalintojen välillisiä kustannuksia, ei tässä käytetyn kululajijaon perusteella voi tehdä juuri mitään päätelmiä.

Edellä esitetty kulujen kirjaamistapa ei ole menetelmänä moitittava, mutta huomataan, että *palvelujen ostot* on kustannuslajin otsikkona liian yleinen. Informatiivisempi ja myöhempää päätöksentekoa paremmin tukeva tapa olisi jakaa *palvelujen ostot* kolmeksi tai neljäksi uudeksi momentiksi.

4.5 Aiempia tutkimuksia Open Sourcen taloudellisuudesta

Kokonaiskustannuslaskelmien lopputuloksiin vaikuttaa keskeisesti organisaation olemassa oleva tietotekniikkainfrastrukstuuri, järjestelmien ikä sekä palveluiden uusimistarve. Usein laskennassa tehdään vertailuja kahden perusratkaisun ja niistä aiheutuvien kustannusten kesken, esim. Windows vs. Open Source tai Linux vs. Unix. Seuraavassa kohdassa on tutustutaan laskelmiin, joissa on vertailtu Windows-ja Open Source pohjaisten ratkaisujen kustannuksia kunnan tietotekniikkatarpeisiin. Kohdassa 3.5.2 perehdytään tutkimukseen, jossa verrattiin Linux- ja Unix -käyttäjärjestelmien taloudellisuutta palvelinkäytössä.

4.5.1 Suomessa ristiriitaisia laskelmia

Vuoden 2001 aikana Suomessa keskusteltiin julkisuudessa ainakin kahdesta selvityksestä, joissa tutkittiin Open Source -ohjelmistojen hintavaikutusta IT-kustannuksiin.

Turun kaupunki kiinnostui Open Source -ohjelmista Microsoftin toimistosovellusten vaihtoehtona. Kaupungin kiinnostus heräsi kun oli Microsoft aiemmin samana vuonna esitteli uuden lisenssimaksumallinsa [Turun kaupunki 2001]. Ehdotuksessaan kaupunginhallitukselle tietohallinnon johtoryhmä korosti lisenssimaksujen merkitystä ohjelmistokustannuksissa.

Tietohallinnon laskelmien mukaan nykyisten ohjelmien käyttö tulisi edullisimmillaan maksamaan 2,3 miljoonaa euroa seuraavan neljän vuoden aikana, päivitysten jälkeen kalleimmillaan 4,1 miljoonaa euroa samassa ajassa. Tietohallinto ennustaa Microsoftin käytön edellyttävän myös laitepäivityksiä, jotta valmistajan tuoreimmat ohjelmat toimisivat halutulla tavalla. Microsoftin ohjelmia käytettäes tulisi neljän vuoden aikana päivittää kaikkien henkilökohtaisten työasemien ohjelmat, mikä maksaisi 3,2 miljoonaa euroa.

Vertailussa olleet Open Source -pohjainen käyttöjärjestelmä ja toimistosovellukset maksaisivat 84 euroa asennuspakettia kohti. Niitä arvioitiin tarvittavan muutamia. Open Source -ohjelmien kustannusten laskettiin tulevan jakelumedioiden valmistuksesta ja postituksista.

Hankintahintojen perusteella valinta kallistuisi selvästi kääntyvän maksuttomien ohjelmien puolelle. Selvityksessä todetaan ympäristön muutosten vaatimusten vaatimien asennuskustannusten olevan samaa luokkaa kuin koko kaupungin henkilöstön sovellusten päivittäminen Microsoftin tuoreisiin versioihin. Tämä päivitys on kaupungissa jo suunnitteilla, joten Open Source -tuotteiden asennuksista ei ennusteta seuraavan korkeampia kustannuksia kuin kaupallisten ohjelmien käytön jatkamisesta.

Muutoksesta ehkä koituvien ongelmien kartoittamiseksi kaupunginhallitus päätti teettää selvityksen vaihtoehtoisten ohjelmien soveltuvuudesta kaupungin työasemastandardiksi. Selvityksen [Onnela 2001] lähtökohdat ovat enemmän pragmaattiset kuin taloudelliset. Selvityksessä käydään perusteellisesti läpi kunkin ohjelmakomponentin toiminta toivotussa ympäristössä, mutta toteutuksen kustannuksiin ei oteta kantaa. Koulutusta mainitaan tarvittavan, mutta tarvittavan koulutuksen laajuuteen ei tarkemmin puututa.

Selvityksen päätelmisssä suositellaan portaittaista siirtymistä maksuttomien ohjelmien käyttöön. Lyhyen aikavälin (vuoden 2003 loppuun mennessä) tavoitteena on siirtyä Open Source -toimistovellusten käyttöön maksullisen käyttöjärjestelmän päällä. Pidemmän tähtäyksen suunnitelma on vaihtaa myös käyttöjärjestelmät maksuttomiin tuotteisiin. Konkreettisina käytännön toimina ehdotetaan, että 200 pilottikäyttäjää aloittaisi uusien ohjelmien käytön vuoden 2002 syksyllä ja että pilottiprojektin tulosten perusteella vuoden 2003 puolella implementoitaisiin Open Source -työasemia asiainhallinnan tuotantokäyttöön. Selvityksessä on tehty tietohallinnon johtoryhmän näkemyksen kanssa yhtenevä päätelmä, ettei laitekantaa tarvitse Open Source -ympäristössä uusia yhtä usein kuin Microsoft-ohjelmia käytettäessä. Perusteluita väitteen tueksi ei esitetty.

Microsoft teki Turun selvityksen kanssa samoihin aikoihin, loppuvuodesta 2001, omat laskelmansa kahden kaupungin, Vaasan [Microsoft 2001a] ja Lappeenrannan, [Microsoft 2001b] tietohallinnon kustannuksissa. Laskelmien lähtökohtana oli saattaa molempien kaupunkien tietohallinto mahdollisimman vakioiduksi ja keskitetysti hallituksi. Raporteissa käydään monipuolisesti läpi tekijöitä, jotka lisäävät tai vähentävät kokonaiskustannuksia. Vaasan selvityksessä on eroteltu välittömät ja välilliset kustannukset, ja välillisten kustannusten merkitys on todettu kokonaisedullisuuden kannalta tärkeämmäksi. Tätä ajatusta tukevat myös Valtiovarainministeriön tilastot [Valtiovarainministeriö 2001a] Suomen valtiohallinnon tietotekniikkakuluista viime vuosikymmenen loppupuolelta.

Vaasan kaupunkia koskevan laskelman keskeinen lopputulos on, että ympäristön pitkälle viedyllä vakioinnilla ja keskitetyllä hallinnalla saavutetaan kustannussäästöjä verrattuna tilanteeseen, jossa tietohallinnassa ei tehdä mitään muutoksia. Numeraalisesti kiinnostavaksi nousee päätelmä, jonka mukaan Open Source -tuotteilla halutun ympäristön rakentaminen tuottaisi vuosittain 4,7 miljoonaa euroa korkeammat käyttökustannukset kuin maksullisilla ohjelmilla. Luvun kerrotaan syntyvän hallintakustannuksista, mutta niiden syntyä ei selvennetä. Mittavien käytönaikaisten kustannusten avoin jäsentely olisi erittäin mielenkiintoista. Kustannustekijöiden analysointi saattaisi estää muiden kaupunkien ajautumista vahingossa merkittävään kustannusloukkuun.

Selvityksen mukaan Vaasan kaupungille ainoa taloudellisesti mielekäs ratkaisu on siirtyä tehokkaammin keskitettyyn tietohallintajärjestelmään, ja toteuttaa se Microsoftin tarjoamilla tuotteilla. Raportissa todetaan kaupungin käyttävän yhteensä yli 200:aa sovellusta, joista osa on kaupungin toimintojen kannalta keskeisiä. Näistä sovelluksista saatavien palvelujen katsotaan olevan kohtuuttoman kalliita toteuttaa muussa kuin nykyisessä ympäristössään. Perusteluja väitteelle ei esitetä, mutta toisaalla todetaan

nykyistenkin sovellusten vaativan päivityksiä toimiakseen jatkossa Microsoftin uudemmissa ympäristöissä.

Vaasan ja Lappeenrannan laskelmissa ei ole otettu huomioon käyttöympäristön portaittaisen vaihtamisen mahdollisuuksia, eikä ole otettu kantaa pilottihankkeisiin. Vaasan selvityksen perusteella voi päätellä, että kaupunki on joltain osin lukittunut (lock-in -tilanne) jo käytössä olevaan Microsoft-ympäristöön, eikä sen vaihtamiseen ole suuria mahdollisuuksia. Laskelmien mukaan Microsoft-ratkaisu on myös edullisin, joten maksullisista ohjelmista ei edes ole tarvetta hankkiutua eroon.

Ohjelmien valmistajan ei tule ollakaan se peikko, josta pyritään eroon, vaan valinta- ja vaihtoperusteiden tulee löytyä tarvittavasta tekniikasta. Vaasan tapauksessa sovellukset eivät ilmeisesti perustu avoimiin standardeihin, joten vaihtoehtoisten ohjelmien tuottaminen maksullisena tai maksuttomana ohjemointihankkeena on vaikeaa ellei mahdotonta.

4.5.2 Linux palvelinkäytössä edullisempi kuin Unix

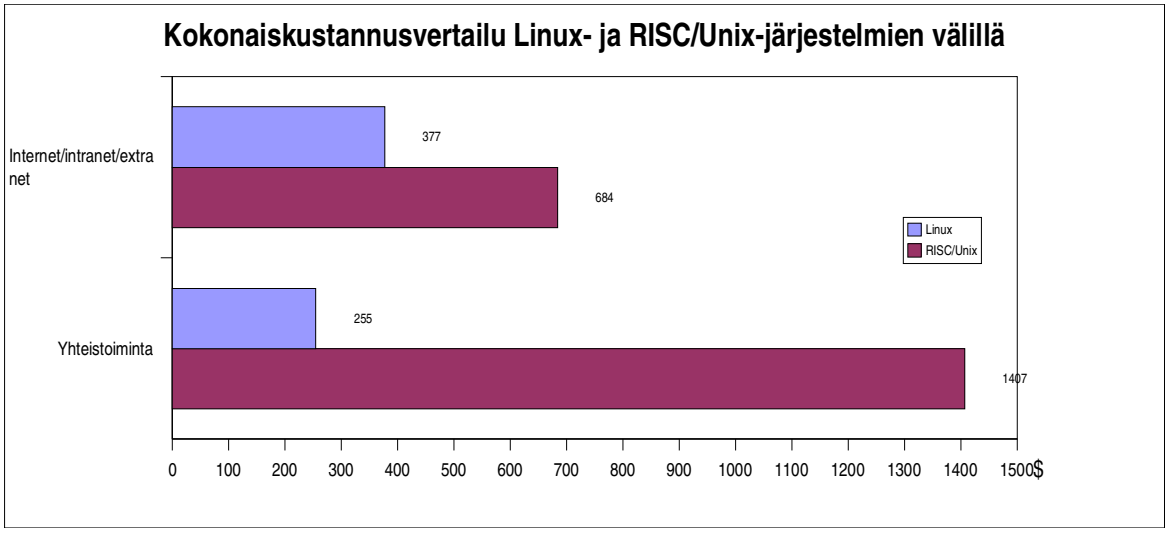
IDC:n selvityksen [Gillen et al. 2001] mukaan Linux-käyttöjärjestelmä asennettuna Intel-pohjaisiin tietokoneisiin on varteenotettava vaihtoehto RISC/Unix-yhdistelmille yrityskäytössä. Linux tarjoaa merkittävät mahdollisuudet vähentää käytönaikaisia kustannuksia. Potentiaalisia hyötyjiä ovat yritykset jonka työntekijöillä on riittävät valmiudet ryhtyä hyödyntämään Open Source -tuotteita. Väitteensä tueksi IDC esittää tuloksia, jotka se sai selvittäessään tietojenkäsittelykustannuksia joko Linux/Intel- tai Unix/RISC -alustoille rakennetuissa ympäristöissä.

Intel-suorittimet ovat arkkitehtuuriltaan CISC-pohjaisia (complex instruction set computer). Unix-koneissa taas RISC-prosessorit (reduced instruction set computer) ovat suosiossa. CISC-suorittimien käskykanta sisältää vaihtelevan pituisia sekä suoritusajaltaan erilaisia konekäskyjä. RISC-suorittimien käskyt ovat lyhyitä ja keskenään samanpituisia ja käskyjä voidaan suorittaa samanaikaisesti rinnakkain. Käskyjen rinnakkaisella suorituksella pyritään mahdollisimman suureen suoritustehoon [ATK-sanakirja 1996]. Kustannussyistä Linux suunniteltiin alunperin Intelin CISC-prosessoria käyttäville koneille ja on edelleen hyvin suosittu nimenomaan Intel-pohjaisissa koneissa. Myöhemmin Linuxista on ilmestynyt versioita myös muiden valmistajien CICS- ja RISC-suorittmia varten.

Saavuttaakseen Linuxin edut yrityksen on aloitettava pilottiprojektista ja kehitettävä pitemmän jakson suunnitelma, jonka mukaan Linuxin käyttöä yrityksessä laajennetaan.

Laajennus voi tapahtua korvaamalla nykyisiä palveluja Linux-pohjaisilla tuotteilla tai jakamalla nykyisten sovellusten kuormaa osittain Linux-koneille [Gillen et al. 2001].

IDC:n mukaan keskeisiä huomioon otettavia seikkoja Linuxin täysipainoiseksi käyttöön valjastamiseksi on muutamia. Mahdolliset sovelluskohteet tulee arvioida huolellisesti ja suunnittelijoiden tulee ymmärtää, miksi valittuihin kohteisiin voi tai kannattaa harkita Linux-vaihtoehtoa. Johdon odotukset on oltava selvillä ja testihankkeiden tuloksia tulee kyetä monitoroimaan. Hankkeen menestyminen edellyttää tasapainoa kuormitusvaatimusten, IT-yksikön valmiuksien ja Linux-järjestelmän ominaisuuksien kesken. IDC listaa ominaisuudet seuraavasti: kustannus, suorituskyky, luotettavuuus, hallittavuus, sovellusten saatavuus ja toimittajatuki. Ottamalla huomioon edellä mainitut tekijät saadaan lupaavia tuloksia, kun tutkitaan internet/intranet/extranet -palveluja sekä yhteistoiminnallisuuteen liittyviä tehtäviä. Tulokset on esitetty kaaviossa 1.



Kaavio 1. Linux- ja RISC/Unix-järjestelmien vuositason kustannukset tuhatta käyttäjää kohden. Hinnat ovat Yhdysvaltain dollareissa.

Internet/intranet/extranet-osuuteen kuuluivat kaikki sisäisiin ja ulkoisiin palomuureihin liittyvät sovellukset ja tehtävät, web-palvelut ja -välimuistitoiminnot, sekä web-palvelut yhteistyökumppanien ja asiakkaiden tarpeisiin.

I/I/E -osio jakaantuu kahteen pääkohtaan. Ensimmäinen on 12-osainen *käyttökulut*-kohta, jossa kustannukset ovat kohtuullisen tasoissa: 266 USD/1000 käyttäjää Linuxin osalla, 341 USD/1000 käyttäjää RISC/Unix puolella. Toinen kohta koskee palvelimien laitteisto- ja ohjelmistokustannuksia. Näistä muodostuvat luvut ovat vastaavasti 111 ja 343.

Yhteistoiminta-osuuteen on tässä yhteydessä sisällytetty sovellukset, joiden avulla käyttäjät voivat tehdä yhteistyötä jakamalla tietoja ja toimintoja. Näitä ovat muun muassa elektroniset kalenterit, jaetut hakemistot ja tietokannat, keskustelualustat, muokattavien ohjelmien kehitysympäristöt sekä muut viestintään liittyvät sovellukset.

Yhteiskäyttöpalvelut jaetaan myös käytönaikaisiin kustannuksiin sekä laitteisto-, ohjelmisto- ja asennuskustannuksiin. Yhteiskäytössä käytönaikaiset kustannukset ovat selvästi Linuxin eduksi 220 USD/1000 käyttäjää, RISC/Unix ratkaisuilla vastaava luku on 1176. Hankintakustannusten osalta suhteellinen ero on dramaattisempi, tosin rahassa pienempi: Linuxilla 35 USD/1000 käyttäjää, RISC/Unixilla 231.

IDC:n laskentamalli kokonaiskustannuksien selvittämiseksi on yksinkertaistettuna kahdeksanportainen:

1. Määritellään järjestelmän kunkin palvelun tukitehtäviä hoitavan henkilöstön lukumäärä.

2. Määritetään työajasta prosentuaalinen osuus, jonka IT-henkilöstö käyttää edellisen kohdan tehtävien tukipalveluiden hoitoon.

3. Lasketaan viikottain käytetty aika tunteina.

4. Muunnetaan edellisen kohdan tulos vuosittaisiksi työtunneiksi.

5. Muunnetaan kustannukset dollareiksi (tai euroiksi, punniksi tai muuksi valuutaksi).

6. Normalisoidaan käyttäjäkohtaiset kustannukset koskemaan tuhatta tuettua käyttäjää.

7. Määritellään laitteistokustannukset ja ohjelmistojen lisenssimaksut, ja normalisoidaan ne koskemaan tuhatta tuettua käyttäjää.

8. Yhdistetään kaikkien palvelujen käyttökustannukset (tuki- ja hallintakustannukset) ja laitteisto-, ohjelmisto- ja asennuskustannukset, jolloin saadaan selville valitun järjestelmän kokonaiskustannukset.

Linuxin pienempiä kustannuksia selitetään Linux-palvelimina käytettyjen Intelin 32 bittisten CISC-arkkitehtuurin laitteiden edullisuudella verrattuna Unix-koneiden RISC-arkkitehtuurin koneisiin. Intel-pohjaisia laitteita tulkittiin voitavan käyttää pidemmän

ajan ja niitä voidaan kierrättää tehtävästä seuraavaan. Lisäksi Intel-koneiden yksinkertaisempi rakenne tuo säästöjä palvelimien poistovaiheessa ohjelmistollisissa purku- ja siirtotehtävissä sekä laitteistojen vaihdossa.

IDC:n näkemyksen mukaan Linuxin runsaan oheisohjelmiston täysipainoinen hyödyntäminen alentaa merkittävästi käytönaikaisia kustannuksia. Toinen tietokantoihin liittyvä kustannustekijä liittyy sähköpostipalveluihin. Linux-ympäristössä sähköpostin välittäjänä toimii useimmiten Sendmail-ohjelma, kun taas Unix-käyttöjärjestelmän kanssa saatetaan käyttää esimerkiksi Lotuksen Notes- tai Domino-ohjelmistoja. Lotus-vaihtoehto luo lisätarpeen uuden tietokantaympäristön hallintaan ja lisää siten käytönaikaisia kustannuksia. Tämä tulos antaa viitteitä siitä, että Open Source -tuotteilla voidaan saavuttaa todellisia säästöjä palvelutasoa kuitenkaan alentamatta.

IDC:n tutkimukseen oli kerätty tietoja 142 yhdysvaltalaisesta yrityksestä, joiden yhteisliikevaihto oli 2,4 miljardia dollaria. Yrityksistä 80 edusti Linux-leiriä, kun 62 yrityksessä oli Unix-ympäristö. Laskentaan vaikuttavat tekijät oli raportissa monipuolisesti ja loogisesti eritelty ja niiden käyttäminen laskennan pohjana oli perusteltu tekijäkohtaisesti. Pois jätettyjä kustannustekijöitä käsiteltiin myös, samoin kuin syitä niiden jättämiseen tco-laskelman ulkopuolelle. Raportin mukaan Linux-ympäristön edut voidaan hyödyntää tehokkaimmin kun IT-henkilöstöllä on riittävät edellytykset vastata palvelujen järjestämisestä sekä ylläpidosta. Raportin pohjalta jää avoimeksi, miten riittävä ammattitaito voidaan arvioida ja mitata, ja millaisin kustannuksin henkilöstön osaaminen voidaan nostaa riittävälle tasolle.

Selvityksessä kerrottiin käytetyn jonkinlaista tutkimustyökalua, jonka avulla kerätyt tiedot syötettiin laskentaan, mutta laskentamallia ei yksityiskohtaisesti esitelty. Toisaalta laskentaperiaatteet esiteltiin varsin tarkkaan ja selvitys perustui pitkälti olemassaolevien yritysten toteutuneisiin kustannuksiin, joihin myös ennusteet tulevista kustannuksista pohjautuivat.

Kustannusten yhteismitallisuutta parantaa varmasti niiden normalisoiminen tiettyä käyttäjämäärää kohti, mutta yritysten yhteisen liikevaihdon perusteella useimmissa tutkimukseen osallistuneista yrityksistä on todennäköisesti vähemmän kuin 1000 työntekijää. Normalisointi 1000 työntekijään tuo mielikuvan tutkittujen yritysten koosta, ja vastaavasti siitä millaisiin yrityksiin selvityksen tuloksia on luontevaa käyttää. Kustannusten normalisointi sataa tai viittä sataa työntekijää kohti olisi ehkä ollut luontevampaa.

Edellä käsiteltyjä tutkimuksia järjestelmien kokonaiskustannusten laskennasta yhdistää lopullisten laskentakaavojen jääminen ainoastaan tutkimusryhmän käyttöön. Tämä

vaikeuttaa tulosten oikeellisuuden arviointia sekä mahdollisten kehityskohteiden paikallistamisen laskentamenetelmissä. Osasyy kaavojen salaamiseen juontuu varmasti laskelmat tehneiden organisaatioiden liiketoiminnallisesta luonteesta. Molemmat tutkimusyritykset Gartner Group sekä IDC, joka on osa IDG-konsernia, perustavat toimintansa maksullisiin tutkimuksiin, joita ne toteuttavat liike-elämän ja julkishallinnon tarpeisiin. Vaikka tutkimukset ovat maksullisia, ei niiden metodeja ja tuloksia tule tästä syystä kyseenalaistaa. Tulokset tosin ovat tutkijan kannalta kiusallisia, koska niiden pohjalle ei voi rakentaa teoriaa, ennen kuin tulosten syntyhistoria on tarkemmin tiedossa ja arvioitavissa.

4.6 Käytönaikaisia kuluja ei voi yleistää

Ohjelmistojen hankinta- ja lisenssimaksujen pohjalta on luontevaa muodostaa yleinen kaava, jota sovelletaan kustannuslaskennassa. Edellä esitetyn valossa voidaan todeta, että käytönaikaisten kustannusten yhtä suoraviivainen sijoittaminen kaavaan ei ole viisasta. Vaikka käytönaikaisesta kustannuskertymästä voidaan eriyttää useita tekijöitä, tekijöiden painoarvojen määräämiseen ei voida osoittaa yleispätevää menetelmää. Mekaanisen kaavan luominen aiemmin aloitetulla tavalla on tullut tiensä päähän.

Edellä on tarkasteltu useasta eri näkökulmasta järjestelmän elinkaaren (t), (e), (h) ja (p) -vaiheiden kustannustekijöitä. Tarkastelun perusteella ei voida löytää näihin luokkiin kuuluvia kustannuksia, joiden kertyminen korreloisi suoraan valitun ohjelmiston lisenssiperiaatteiden kanssa. Mikäli käytönaikaisia kustannuseroja syntyy, ne ovat seurausta ohjelmistojen arkkitehtuurista, tietorakenteista ja muista teknisistä ominaisuuksista sekä käyttäjien valmiuksista tuotteiden käyttöön.

Sovelluskohteen yksityiskohtaista analyysia kustannuslaskentaa voi verrata normaaliin, ennen ohjelmistoprojekteja tehtävään vaatimusmäärittelyyn. Toiminnot, niiden vaatimukset ja riippuvuussuhteet käydään läpi ja mallinnetaan sopivaa menetelmää käyttäen. Määrittelyn aikana saadaan selville, millaisin painotuksin edellä esitellyt kustannustekijät tulee ottaa huomioon laskennassa. Määrittelyn aikana myös selvitetään kirjaamiskäytäntö, jota organisaatiossa on käytetty IT-budjettien valmistelussa ja toteutumisen seurannassa.

Luotettavan vertailun tekemiseksi tulee myös palveluiden toteuttamiseen tarvittavat ohjelmistot ja niiden laitteistovaatimukset määritellä konkreettisten esimerkkien kautta. Esimerkiksi IDC:n [Gillen et al. 2001] tutkimuksessa todettiin Unix/Risc ympäristössä tarvittavan samojen palvelujen tuottamiseksi useampia palvelinlaitteita kuin Linux-ympäristössä.

Mikäli tehdään kustannuslaskentavertailua kahden kilpailevan järjestelmän välillä, joista toinen pohjautuu organisaatiossa jo käytössä olevaan tekniikkaan tai ohjelmistoon, on toteutuneiden kustannusten analyysi erityisen merkittävä. Välttämättä pelkkä IT-osaston budjettianalyysi ei tuo riittävästi tietoja, vaan organisation kustannusrakennetta tulee tutkia laajemmin kirjaamislogiikan selvittämiseksi.

Kappaleessa 4.5.2 esitelty IDC:n noudattama laskentamalli on sovellettavissa palvelinvertailua yleisempäänkin käyttöön. Mallissa osakustannukset on normalisoitu koskemaan tuhatta käyttäjää. Yhden organisaation kustannuksia laskettaessa ei normalisoinnilla ole samaa merkitystä kuin tutkimuksessa, jossa otetaan huomioon useamman yrityksen kustannukset. Normalisoiminen on hyvä tapa saattaa eri asiakkaiden kustannukset yhteismitalliseksi, jos on kyse laajemmasta tilastointitarpeesta tai jos yrityksen toimialaan kuuluu kokonaiskustannusten laskeminen.

Laskelmien lisäksi tai niistä huolimatta uusia menetelmiä voidaan useimmissa organisaatioissa testata pilottiprojekteilla, ennen suurien tietoteknisten linjausten tekemistä. Pilottiprojekti kannattaa määritellä siten, että se on selkeästi rajattu osa laajempaa kokonaisuutta, tai miniatyyrimalli isommasta ratkaisusta.

Vaikka pilottiprojektissa on lopulliseen tuotantoympäristöön verrattuna puutteita ja rajoituksia, käytännön testien kautta saadaan arvokasta kokemusta. Tämän kokemuksen tulisi auttaa arvioimaan kustannuslaskennan parametreja ja varsinkin niitä, joissa tehdään arvioita uuden järjestelmän tulevista kustannustarpeista tai -hyödyistä.

5 CASE -ESIMERKKINÄ SEMINAARIJÄRJESTELMÄ

Tutkielman ohjelmointityön tarkoituksena oli simuloida kuvitteellisen yrityksen tietohallintojohdon asettamaa pilottiprojektia, jossa tutkitaan oheispalveluiden tuottamista valituilla työkaluilla. Kohdassa 5.1 kuvataan esimerkkisovelluksen toiminta. Kohdassa 5.2 pohditaan onko valittu sovellus onnistunut esimerkkisovelluksena. Kohdassa 5.3 tarkastellaan lähestymistapaa ohjelmointityöhön.

5.1 Esimerkkisovelluksen kuvaus

Tutkielman case-esimerkkinä on tuote, jolla seminaarinjärjestäjä voi suhteellisen automatisoidusti hallita seminaarin ilmoittautumisia sekä markkinointia. Seminaarin sijaan tapahtuma voi olla mikä tahansa muukin yleisö- tai yksityistilaisuus, jonka osallistujatietoja halutaan hallita. Tässä kuvattava järjestelmä on valittu kohteeksi, koska lähes kaikki organisaatiot järjestävät joskus tilaisuuksia, joiden osallistujista tai ainakin heidän lukumäärästäään on hyvä olla etukäteen perillä. Osallistujien ilmottautumisten hallinta on melko rutiiniluontoista työtä, ja se vie työaikaa. Siihen liittyvät rutiinit sopivat loistavasti koneen hoidettaviksi.

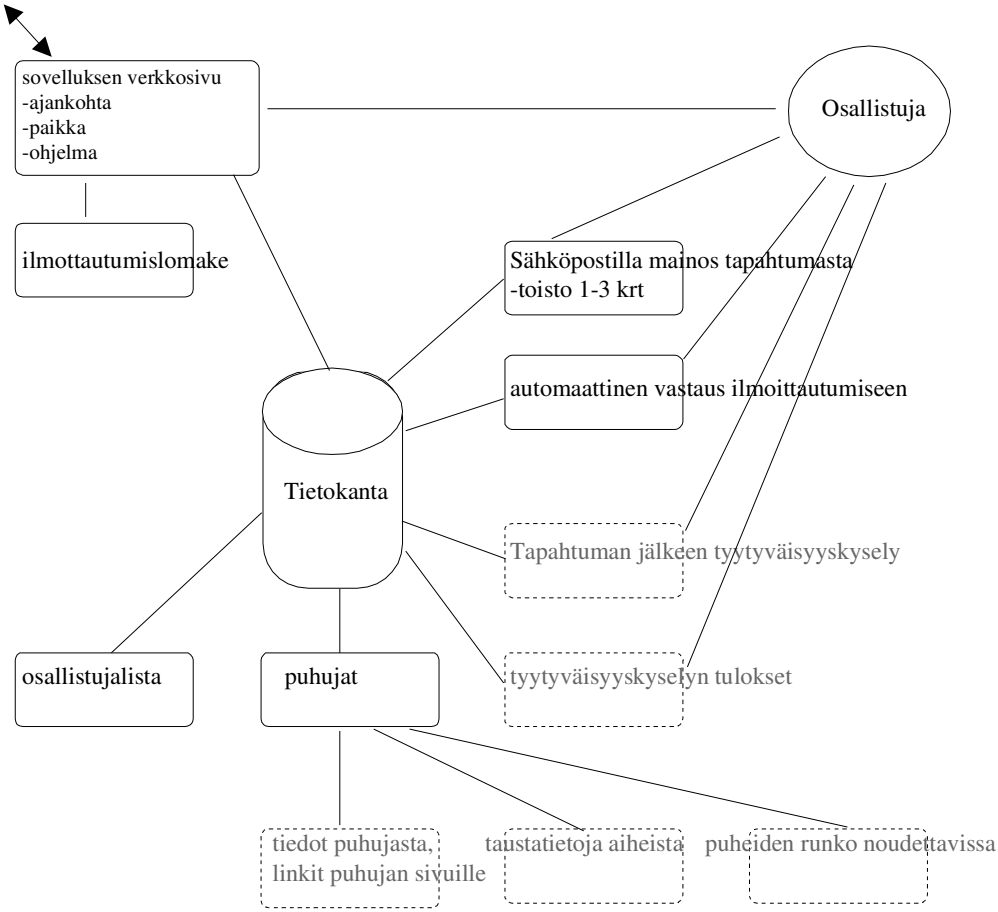
Sovellus integroidaan tapahtuman järjestäjän verkkosivuille, ja jos tapahtumien järjestäminen on satunnaista, sovellus voi käytön väliaikoina olla passiivisena. Tarpeen tullessa lisätään sopiville verkkosivuille linkit, joilta pääsee ilmottautumissovelluksen etusivulle.

Tämän sovelluksen lähtöajatus on, että käyttäjä on saatu jollain markkinoinnin keinolla vierailemaan sovellussivuille. Impulssi vierailuun voi syntyä monin eri tavoin, eikä niiden keskinäiseen paremmuuteen oteta tässä tutkielmassa kantaa. Näitä menetelmiä ovat esimerkiksi suoramarkkinointi paperi- tai sähköpostilla tai tekstiviestillä, lehti-, tv-, radioilmoitukset, banneri-ilmoitukset ja muut linkit internet-verkossa.

Tuotteen painopiste on vieraslistan hallinnassa. Markkinointiosuutta ovat lähinnä lyhyet kuvaukset tulevista tapahtumista sekä mahdolliset linkit lisätietosivuille. Järjestäjä voi pitää yllä osallistuja-ja esiintyjälistoja eri tapahtumiin, hakea kunkin osallistujan historia- sekä taustatiedot, sekä lähettää sähköpostia ja tekstiviestejä niille vieraille, jotka ovat sen sallineet. Syntyvää osallistujalistaa voi jatkossa käyttää myös markkinoinnin tietokantana. Järjestelmään voidaan tuoda markkinointitarkoituksia varten henkilötietoja

myös muista tietokannoista, esimerkiksi järjestäjän omasta crm-järjestelmästä (customer relationships management) tai ostetuista rekistereistä.

Seminaarivieraan on mahdollista tutustua järjestäjän tapahtumiin sekä ilmoittautua niihin.



Kuva 1. Esimerkki sovelluksen toiminnalliset osat

Kuvassa 1 on karkeasti esitetty sovelluksen toimintaa. Katkoviivalla piirretyt osat ovat jatkokehitysmahdollisuuksia ja niitä ei ole ohjelmoitu toimiviksi.

5.2 Onko valittu CASE-esimerkki hyvä tämän tutkielman osana?

Tässä tutkielmassa käsitellään sekä ohjelmistotuotannon tekijöitä että useita Open Source -maailman ilmiöitä. Erityisesti lukijalle pyritään tuomaan eväitä, joilla hän voi hahmottaa avoin lähdekoodi -termiä, sekä esittelemään vaikutuksia joita maksuttomien ohjelmien integroinnilla on järjestelmän kustannuksiin ja toimintaan. Tutkielman tarkoituksena ei

niinkään ole löytää vastausta kysmykseen "onko Open Source hyvä vai huono?" kuin arkipäiväistää avoimen lähdekoodin käsitettä esittelemällä mallin käyttäytymistä.

Työn osana toteutettu ohjelmistoprojekti tuo tutkittavaan aiheeseen konkreettisen otteen. Tämän kautta teemme pienimuotoisen tutkimusretken Open Source -maailmaan ja integroimme löydöksen reaalimaailmaan. Tuloksista hyödytään kahdella tavalla. Ensinnäkin tutkielmassa toteutettu ilmottautumisjärjestelmä voidaan liittää osaksi yrityksen tai yhteisön olemassa olevaa verkkosivustoa. Toiseksi tutkitaan mitä kustannustekijöihin liittyviä ilmiöitä tällaisen projektin aikana tulee huomioida. Nämä kustannustekijät muodostavat yhdessä järjestelmän kokonaiskustannukset.

Esimerkkinä oleva ilmottautumisjärjestelmä on ohjelmistona useimmille yrityksille toimintaa tukeva järjestelmä, ei välttämätön hankinta. Tuotetta voidaan hyödyntää useimmissa yrityksissä ja muissa yhteisöissä. Ilmottautumisjärjestelmä vapauttaa henkilöresursseja rutiinitehtävästä muuhun työhön, koska osallistujan kirjaaminen siirretään hänen omaksi tehtäväkseen. Järjestelmä lähettää myös automaattisesti viestejä osallistujille.

Osallistujan tiedot kirjautuvat tietokantaan, jossa ne säilyvät vielä tapahtuman jälkeenkin. Järjestäjä voi milloin tahansa ottaa tietokannasta raportin, josta nähdään mihin tilaisuuksiin tietty osallistuja on ottanut osaa. Samoin voidaan tapahtumakohtaisesti hakea tiedot tapahtumien osallistujamääristä.

Järjestelmässä on käytetty standardiratkaisuja tietokannan määrittelyssä ja käsittelyssä, joten esimerkiksi osallistujatietojen vieminen järjestäjän asiakashallintajärjestelmään tai tietojen hakeminen sieltä onnistuu varsin yksinkertaisesti. Jos taas esimerkkijärjestelmän käyttäjällä ei ole olemassa varsinaista asiakashallintajärjestelmää, tämä ilmottautumisjärjestelmä tietokantoineen voi auttaa hahmottamaan mihin sellaista järjestelmää voi käyttää.

Esimerkkisovelluksessa on toteutettu keskeiset perustoiminnot, kuten sovelluksen yhteydenotto tietokantaan, tietueiden haut, muutokset, lisäykset ja poistot. Esimerkin raportit tulostetaan näytölle, mutta ohjelmakoodia muuntamalla ne voisi ajaa vaikka tekstitiedostoon ja jatkaa niiden käsittelyä tarvittaessa tekstinkäsittely- tai taulukkolaskentaohjelmalla.

Tässä ohjelmaversiossa ei ole huomioitu esimerkiksi tapahtumien maksullisuutta. Ohjelmiston ylläpitäjän on mahdollista lisätä hintakentät tietokannan kuvaukseen, ja lisätä tarvittavat yhteenlaskut ja tulostukset php-kielisiin ohjelmamoduleihin. Näin voisi

esimerkiksi seurata, ketkä asiakkaat käyttävät eniten rahaa osallistumismaksuihin ja millaisista tilaisuuksista he ovat kiinnostuneita.

Open Source -mallin kannalta esimerkkiprojektin puute on, ettei ohjelmointityötä ole julkistettu projekti ja eikä siinä siten ole hyödynnetty muita ohjelmoijia. Tämä johtui muun muassa siitä, että järjestelmän ohjelmointi- ja suunnittelu oli kohtuullisella työmäärällä yhden ihmisen toteutettavissa. Tutkielman kirjoittajana halusin lisäksi perehtyä käytettäviin työkaluihin ja niiden ominaisuuksiin, ja samalla muodostaa näkemystä ohjelmien käytettävyydestä esimerkin kaltaisiin projekteihin. Mikäli sovellus julkaistaisiin esimerkiksi GPL-lisenssin alaisuudessa, voisi kuka tahansa jatkaa ohjelman kehitystä tai vaikkapa perustaa Open Source -kehitysprojektin tuottaakseen täydellisemmän ilmottautumis- tai asiakashallintajärjestelmän.

Kaikki tutkielman valmistamiseen käytetyt ohjelmat ovat jonkin Open Source -lisenssin alaisia ja maksuttomasti saatavilla lähdekoodeineen.

5.3 Lähestymistavan valinta

Sovellusesimerkissä toteutetaan eräänlainen tietotekninen innovaatio. Innovaatio yleisesti tarkoittaa jonkinlaista uudistusta, jolla pyritään tuottamaan hyötyä käyttäjälleen [Järvinen ja Järvinen, 2000]. Konstruktiivisen tutkimuksen luonteeseen kuuluu uuden todellisuuden rakentaminen olemassaolevan tiedon mukaan.

Tutkielman yksi tavoite oli rakentaa Open Source -työkaluilla verkkosovellus. Toteutuksen lähtötilana oli tietyn palvelun puuttuminen fiktiivisestä yrityksestä. Palvelun konsepti oli kirjoittajan näkemys palvelun tarpeellisuudesta, idea tuotannossa olevan palvelun toiminnasta ja siihen tarvittavista ohjelmoitavista komponenteista.

Unix-ja Linux-käyttöjärjestelmien yksi ominaispiirre on, että ne sisältävät hyvin paljon pieniä ohjelmatiedostoja, jotka tekevät vain yhden tai hyvin rajatun toiminnon, mutta tekevät sen hyvin [Frisch 1997]. Pieniä komponentteja on helpompi käydä läpi yksi kerrallaan, ja kehittää niitä nopeammiksi ja virheettömämmiksi kuin lähteä parsimaan ja jäljittämään pidempiä kokonaisuuksia. Tämä on mahdollistanut järjestelmien jatkuvan kehityksen ilman suuria hyppäyksiä ja kuiluja uusien ja vanhojen versioiden välillä.

Ohjelmointityö tehtiin tietokoneella, jonka käyttöjärjestelmänä on Linux. Tietokone oli varustettu sekä työasema- että palvelinohjelmilla, joten kehityksen aikaiset testit oli hyvin nopea toteuttaa, jopa ilman aktiivista internet-yhteyttä. Näin toteutustavaksi sopii luontevasti vaihejako.

Järjestelmä koostuu relaatiotietokannasta sekä erillisistä suhteellisen itsenäisistä ohjelmaosista, joilla kullakin on oma varsin spesifinen roolinsa kokonaisuudessa. Näin projektin varsinainen tavoitetila on heti alussa selkeästi määritettävissä ja kaikki toimet tähtäävät jo määritellyn lopputilan saavuttamiseksi. Sovellus ohjelmoidaan ja testataan komponetti kerrallaan, jotka heti valmistuttuaan ovat osa kokonaisuutta.

Sovelluksesta saatiin valmiiksi käyttökelpoinen versio nopeasti, kolmessa viikossa. Vaikka tutkielman ohjelmointityötä ei tehty Open Source -mallin mukaisesti julkistamalla koodi ja pyrkimällä keräämään ohjelmointiryhmä, haluttiin julkaisupolitiikassa noudattaa Open Source -mallin yleistä periaatetta, eli versioita julkaistaan nopeasti ja usein.

Toinen mahdollinen toteutusmalli olisi ollut kirjoittaa sovellus mahdollisimman valmiiksi ja virheettömäksi, ja julkaista se vasta sitten. Koska tässä tapauksessa versioiden julkaisu merkitsi käytännössä kirjoittajan vuorottelua ohjelmointityön ja tutkielman teoriaosan välillä, versioiden "julkaiseminen" loi luonnollisen vaihtelun työskentelyssä. Julkaisu on tässä lähinnä teoreettinen ja hieman liioitteleva mutta sellaisena silti käyttökelpoinen termi: kirjoittajahan oli työn aikana ainoa, joka tutki koodia tai sovelluksen toimintaa.

6 PROJEKTIN TOTEUTUS JA ARVIOINTI

Esimerkkisovelluksen tuottamiseksi tutkielman kirjoittaja tutustui tarjolla oleviin avoimen lähdekoodin mallin mukaisiin työkaluohjelmistoihin. Työkaluista oli valittavana useita vaihtoehtoja, eikä tässä käytetty kokoelma ole ainoa sopiva yhdistelmä. Projektin ohjelmointitavoite onnistui ja toimiva verkkopalvelu tietokantoineen syntyi. Työkalut olivat ohjelmina vakaita eikä häiriöitä esiintynyt. Työkaluihin tutustutaan kohdassa 6.1 ja alakohdissa 6.1.1-6.1.4.

Yksittäisprojektina verkkosovelluksen luominen uusilla työkaluilla olisi ehkä hieman hintava projekti, mutta tässä suhteessa se ei eroa maksullisten välineiden käytöstä. Pilottiprojektia tuloksia voi pitää onnistuneena kokeiluna, eikä mikään estäisi käyttämästä samoja välineitä laajempienkin projektien toteuttamiseen. Samoja välineitä voi hyödyntää myös varsinaiseen Open Source -malliseen ohjelmantuotantoon, jolloin päästään konkreettisesti hyötymään mallin maksuttomasta ohjelmoijareservistä. Kohdassa 6.2 arvioidaan projektia eri näkulmista sekä projektin onnistumista.

6.1 Työkaluohjelmistojen valinnasta

Tutkielmassa rakennettu palvelinjärjestelmä on toteutettu maksuttomia ohjelmallisia välineitä käyttämällä. Työkalujen valintaperusteina olivat niiden arvioitu soveltuvuus hankkeeseen sekä niiden käytön yleisyys. Valinta tehtiin tutustumalla työkalujen ominaisuuksiin sekä sellaisiin sovelluksiin joissa työkaluja oli käytetty. Aineistoa tähän löytyi internet-verkosta ja alan painettujen julkaisujen artikkeleista. Linux-käyttöjärjestelmästä on saatavilla lukuisia eri jakelijoiden paketteja. Ohjelmointikielenä olisi voinut olla PHP:n sijaan Perl, Python tai joku muukin. Tietokantavalinta olisi voinut osua MySQL:n sijasta PostgreSQL:ään. Ainoastaan www-palvelin Apache oli lähes itsestään selvä valinta. Lopullisen työkalukokoelman ratkaisi niiden arvioitu keskinäinen yhteistoiminta.

Koska muilla työkaluilla ei tehty edes koeversioita, ei käytettyä valikoimaa voida todeta parhaaksi mahdolliseksi yhdistelmäksi, mutta onnistuneeksi kylläkin. Valinta olisi sama, mikäli tämän kokemuksen jälkeen tulisi koota paletti uudestaan seuraavaa projektia varten. Muita kandidaatteja voisi kokeilla esimerkiksi vaihtamalla niitä yksitellen tuotantovalmiin sovelluksen komponenteiksi. Uuden työkalun vaatiman muokkauksen yhteydessä erot tulisivat välittömästi esiin ja näin saataisiin kerättyä kokemusperäistä tietoa välineiden soveltuvuudesta.

6.1.1 Käyttöjärjestelmä - Linux

Palvelimen käyttöjärjestelmä on Linux. Linux on suomalaisen Linus Torvaldsin koordinoima ja johtama OpenSource -projekti. Torvalds käynnisti projektin opiskellessaan Helsingin yliopistossa. Linuxin kehittely aloitettiin vuonna 1991 ja versio 1.0 julkaistiin keväällä 1994. Sitä ennen oli liikkeellä runsaasti evoluutioversioita, jotka olivat numeroinniltaan muotoa 0.xx.

Linux muistuttaa läheisesti Unix-käyttöjärjestelmää ja sen eri murteita. Torvalds ohjelmoi Linuxin ytimen, kernelin, yhdisti siihen muun muassa GNU-projektin tuottamia varusohjelmia ja julkaisi paketin verkossa Linux-nimellä. Kernel sisältää muistin, tiedostojen, avoinna olevien ohjelmien, tietoliikenneyhteyksien ja erilaisten laitteiden hallitsemiseksi tarvittavan ohjelmiston.

Ilmestyessään Linux herätti kiinnostusta, koska käyttöjärjestelmä toimi Intel 386- ja 486-suorittimia käyttävissä mikrotietokoneissa, koska se oli maksuton ja koska siinä oli kehittyneitä ominaisuuksia, jotka puuttuivat muista sen ajan mikrotietokoneiden käyttöjärjestelmistä. Aito moniajo (useamman ohjelman yhtäaikainen käyttäminen), monen käyttäjän samanaikainen tuki ja sivuttava virtuaalimuisti olivat siihen saakka olleet tuhansien eurojen hintaisten käyttöjärjestelmien ominaisuuksia. Linuxin ensimmäiset hyötykäyttäjät olivat yliopistojen ja korkeakoulujen henkilökuntaa ja opiskelijoita. Heillä oli mielenkiintoa ja taitoa asentaa ja ylläpitää monimutkaista käyttöjärjestelmää. Yritysmaailmassa internet-palveluntarjoajat löysivät Linuxista edullisen ja suorituskykyisen alustan palvelimilleen. Nykyään Linuxia käytetään työasemissa, palvelimissa, palomuureissa ja erilaisissa sulautetuissa järjestelmissä.

Aluksi Linux julkaistiin erittäin tiukoin lisenssiehdoin, joiden mukaan koodista ja sen levittämisestä ei ollut mahdollista periä maksua missään tilanteessa. Järjestelmän suosion kasvaessa ehdot olivat muodostua kasvun pullonkaulaksi. Järjestelmä oli maksuttomasti saatavissa verkosta, mutta paketit olivat paisuneet kookkaiksi, jolloin niiden lataaminen modeemin kautta kesti kohtuuttoman kauan. Käyttäjät, joilla oli nopea internet-yhteys ja kirjoittava cd rom -asema käytössään, olisivat lisenssiehtojen mukaan voineet toimittaa kaikille halukkaille ohjelman kopioita, mutta se olisi ollut tehtävä ilman mitään vaivan palkkaa. Tämän ongelman poistamiseksi lisenssiehtoja muutettiin niin, että ohjelman levittämisestä ja tallennusmateriaaleista sai periä maksun, ei itse ohjelmakoodista. Muutos loi edellytykset Linuxin kaupallisille jakelumarkkinoille: syntyi nopeasti useita Linuxin jakeluun ja muihin lisäarvopalveluihin, kuten koulutukseen ja kirjallisuuteen

erikoistuneita yhtiöitä. Suurimmat Linux jakeluyhtiöt ovat listautuneet Yhdysvaltain pörssiin.

6.1.2 www-palvelinohjelmisto - Apache

Apache (A PAtCHy server) on maksuton www-palvelinohjelmisto, ja se on käytetyin www-palvelinohjelmisto maailmassa [www.netcraft.com]. Ohjelmiston suosio perustuu sen maksuttomuuteen, hyvään suorituskykyyn sekä ohjelman aktiivisen kehittämiseen. Taulukossa 5 on tilastotietoja www-palvelinohjelmistojen jakautumisesta vuoden 2001 kahdelta viimeiseltä kuukaudelta.

Developer	November 2001	Percent	December 2001	Percent	Change
Apache	7750275	61.88	8588323	63.34	1.46
Microsoft	3307207	26.40	3609428	26.62	0.22
iPlanet	431935	3.45	383078	2.83	-0.62
Zeus	174052	1.39	172352	1.27	-0.12

Taulukko 5. Käytetyimmät www-palvelinohjelmistot vuoden 2001 lopulla.

6.1.3 Ohjelmointikieli - PHP

CGI (Common Gateway Interface) on määritys, jonka pohjalta www-palvelimilla voidaan ajaa erillisiä sovelluksia. CGI määrittää kaksi keskeistä parametritekijää: sen missä muodossa html-lomakkeelta viedään tiedot palvelimella olevalle ulkoiselle sovellukselle ja sen miten tulokset toimitetaan takaisin html-selaimelle.

PHP (PHP: Hypertext Preprocessor) on kehittynyt CGI-ohjelmien tekemiseen tarkoitetusta komentokokoelmasta täysipainoiseksi ohjelmointikieleksi sekä sovelluspalvelinympäristöksi. PHP -kirjainyhdistelmän etymologia on hieman rekursiivinen. Tämä ei ole esimerkki hyvästä lyhenteiden luomisesta, mutta jostain syystä samankaltaisia lyhenteitä on muitakin: GNU tulee sanoista Gnu's Not Unix. GNU on Richard Stallmannin projekti Open Source -käyttöjärjestelmän luomiseksi.

PHP on tulkattava kieli. PHP-ohjelmat ajetaan aina, kun palvelin toimittaa www-sivun selaimelle. PHP:n käyttöä puolustaa pari merkittävää nopeuteen liittyvää tekijää. Palvelin tekee tulkkauksen ja lähettää tulkatun tiedon selaimelle. Käytännössä Apache-palvelimeen asennetaan palvelimen kiinteäksi osaksi PHP-moduli; aina kun Apache on käynnissä, myös PHP on käynnissä. Tällöin tulkkaus on huomattavasti nopeampaa

verrattuna siihen, että joka kerta kun tulkattavaa on, käynnistettäisiin erillinen tulkkiohjelma, kuten esim. Perl-ohjelmien kanssa tehdään.

PHP voidaan asentaa toimimaan myös perinteisen CGI-sovelluksen tavoin. Tällöin menetetään huomattava nopeusetu vaihdettaessa tietoja tietokantojen kanssa. Mikäli PHP asennetaan moduuliasennuksena kiinteäksi osaksi Apachea, voidaan tietokantayhteyksiä pitää jatkuvasti auki. CGI-tyyppissä asennuksessa jokaisen tietokantaa käsittelevän PHP-sivun alussa tietokantayhteys avataan erikseen ja sivun lopussa suljetaan. Tämä hidastava tekijä korostuu, mitä enemmän palvelulla on yhtäaikaisia käyttäjiä.

PHP -kieliset CGI-sovellukset eroavat HTML-koodin ja ohjelmakoodin välisen liikkumisen osalta joistakin muista, esimerkiksi Perl- tai C-kielisistä CGI-sovelluksista. Sen sijaan, että PHP-koodiin kirjoitettaisiin useita komentoja, joilla tuotetaan HTML:ää, voidaan PHP-koodi upottaa normaalin HTML-koodin sekaan.

```
<html>
     <head>
          <title>example</title>
     </head>
     <body>

          <?php
          echo "Hello World!";
          ?>

     </body>
</html>
```

Kuva 2. Esimerkki PHP-koodista upotettuna HTML:n sisään

HTML-palvelin tunnistaa PHP-koodin sen alku -ja loppumerkeistä, tägeistä, ja osaa näin tulkata oikeat osiot. PHP-kieli soveltuu hyvin käytettäväksi XHTML- (The Extensible HyperText Markup Language) ja XML -dokumenttien (Extensible Markup Language) yhteydessä. PHP:tä käytettäessä voidaan varsin joustavasti liikkua PHP:n ja HTML:n välillä, esimerkiksi alla olevan esimerkin mukaisesti.

```
<?php

     if ( boolean-expression )
     {
        ?>
        <strong>This is true.</strong>
        <?php
     }
     else
     {
        ?>
        <strong>This is false.</strong>
        <?php
     }
```

Kuva 3. PHP sallii koodista poistumisen jopa keskellä if..else -rakennetta

Kuvien 2 ja 3 esimerkit ovat PHP-käyttöohjeista. Palvelimen tekemä tulkkaus on keskeinen ero PHP:n ja Javascriptin välillä. Javascriptiä käytetään myös www-sivujen toiminnallisuuden toteuttamisessa. Käytettäessä PHP:tä palvelin tulkitsee koodin ja lähettää asiakassovellukselle vain tulokset. Näin vältytään ylimääräisen datan siirroilta ja saadaan haluttaessa suorittava koodi pidettyä näkymättömänä ulkopuolisilta tarkkailijoilta.

PHP toimii oliohjelmointikielenä, eli sillä voidaan määritellä olioluokkia ja niiden jäsenfunktioita ja ominaisuuksia. Myös luokkien ominaisuuksien periytyminen toimii.

Esimerkkisovelluksessa on käytetty MySQL tietokantaa, mutta se ei ole ainoa tietokanta, jonka yhteydessä PHP:tä voidaan käyttää. PHP toimii muun muassa Adabas D, Ingres, Oracle (OCI7 and OCI8), dBase, PostgreSQL, MS-SQL, Sybase, IBM DB2, Informix, ja Unix dbm -tietokantojen kanssa, eikä käyttö ole rajoittunut mihinkään tiettyyn käyttöjärjestelmä- tai tietokantaympäristöön. PHP:n kanssa on luontevaa käyttää MySQL tietokantaa, koska PHP:n perusversiossa on siihen sisäänrakennettuna täysi tuki. Useimpien muiden tietokantaohjelmistojen kanssa käytetään joko ODBC-yhteyskäytäntöä (Open DataBase Connectivity) tai nimenomaisen tietokannan omia rajapintoja.

6.1.4 Tietokannaksi MySQL

MySQL on maksuton relaatiotietokanta, joka on ruotsalaisen T.c.X Ab:n (nykyisin MySQL Ab) aluperin omaan käyttöönsä kehittämä tietokantasovellus. MySQL-ohjelmistolla on kaksi erilaista lisenssivaihtoehtoa. Tietokantaa voi käyttää maksuttaa GPL-lisenssin alaisena (Gnu General Public Lisence) tai kaupallisiin tarkoituksiin voi lunastaa MySQL AB:lta maksullisen lisenssin, jolloin GPL:n rajoitukset eivät rajoita maksullisten toteutusten levitystä.

MySQL -nimen SQL merkitsee, että tietokannan käsittelyyn käytetään SQL-kieltä (Structured Query Language), joka on standardoitu kyselykieli tietokantojen hallintaan. MySQL:n käyttämä SQL on tosin hieman laajennettu versio standardi-SQL:stä, mikä tulee ottaa huomioon vaihdettaessa tietokantaa muuhun SQL-pohjaiseen tietokantaan.

MySQL:n toteutus on luonteeltaan monisäikeinen. Säikeistyksen etuna ovat tehokkaammat operaatiot yhdelläkin suorittimella varustetuissa palvelimissa, sekä aito tuki usean suorittimen järjestelmiin. Vaikka MySQL on nopea kannan perustoiminnoissa [MySQL 2001], ei suuri tehokkuus johdu ainoastaan säikeistyksestä, sillä MySQL ei ole ainoa tietokanta, joka on ohjelmoitu hyödyntämään säikeistystä. Hyvän suorituskyvyn tavoittamiseksi on ohjelmakoodin oltava tehokasta ja järkevästi toteutettu. MySQL on C- ja C++ -kielillä ohjelmoitu Open Source -tuote, joten sen lähdekoodi on kaikkien kiinnostuneiden ohjelmoijien saatavissa ja heillä on mahdollisuus tehdä parannusehdotuksia koodiin. Siitä ei ole tietoa kuinka paljon MySQL sisältää Open Source -yhteisön kirjoittamaa koodia ja mikä on MySQL Ab:n osuus, joten MySQL:n suorituskyvyn ja ohjelman kehitysmenetelmän suhteesta ei voi tehdä päätelmiä.

Taulukoiden 6 testit on toteutettu samalla Windows NT tietokoneella. Kyselyn monimutkaistuessa luku- ja kirjoitusajat pitenevät, mutta vaativammista kyselyistä ei ole saatavilla valmiita benchmark-tuloksia. Tästä syystä jää avoimeksi, miten käyttäytyminen muuttuu suhteessa muihin tietokantoihin.

Reading 2000000 rows by index	***Seconds***
mysql	367
mysql odbc	464
db2 odbc	1206
informix odbc	121126
ms-sql odbc	1634
oracle odbc	20800
solid odbc	877
sybase odbc	17614

Inserting (350768) rows	***Seconds***
mysql	381
mysql odbc	619
db2 odbc	3460
informix odbc	2692
ms-sql odbc	4012
oracle odbc	11291
solid odbc	1801
sybase odbc	4802

Taulukko 6. MySQL on nopea perusoperaatioissa.

MySQL-tietokantaa käytettäessä eivät sovellusohjelmat keskustele suoraan tietokannan kanssa, vaan palvelinohjelman kautta. Palvelinohjelma voi hoitaa samanaikaisesti

useampia laajoja tietokantoja. Rajoitukset tulevat käyössä olevan levytilan ja käyttöjärjestelmän myötä. Tietokannat koostuvat tauluista. Taulujen maksimikoko vaihtelee kahdesta gigatavusta kahdeksaan teratavuun sen mukaan, mikä on suurin yksittäisen tiedoston koko, jota palvelimen käyttöjärjestelmä osaa käsitellä. Taulujen määrää tietokannassa ei ole rajoitettu.

6.2 Projektin arviointi

Tutkielman teoriaosassa tutkitaan Open Source -tuotteiden merkitystä tietojärjestelmän kokonaiskustannuksiin. Kokonaiskustannuslaskennan helpoin osa on laskea erilaisten lisenssi- ja hankinhintojen vaikutukset. Nämä alkuvaiheen kustannukset muodostavat kuitenkin vain pienen osan järjestelmän kaikista kustannuksista. Suurimmat erot eri vaihtoehtojen välillä tulevat käytönaikaisista kustannuksista. Käytönaikaisten kustannusten muodostumisesta on erittäin vaikea luoda yleispätevää määrittelyä tai kaavaa. Useimmissa laskentamalleissa todetaan IT-henkilöstön ammattitaito yhdeksi tekijäksi, joka vaikuttaa järjestelmän käytönaikaisiin kustannuksiin. Henkilökunnan tehtävänä on kehittää ja ylläpitää järjestelmiä yksin tai yhdessä palveluntarjoajien kanssa.

Ohjelmointiprojektin tarkoitus oli simuloida tilannetta, jossa IT-henkilöstö tuottaa organisaatiolleen lisäpalveluita Open Source -tuotteilla talon sisäisenä projektina. Osaava IT-henkilöstö kykenee toimimaan myös organisaation toiminnan kehittäjänä, eikä ainoastaan ongelmatilanteiden ratkaisijana. Mikäli IT-henkilöresurssit ovat pysyvästi alimitoitetut, on kehitystyön tekeminen luonnollisesti vaikeaa tai mahdotonta, niin maksullisilla kuin maksuttomilla ohjelmilla. Kokeiluprojektin kuvitteellisessa organisaatiossa oletetaan henkilöstöresurssit kehitystyön kannalta riittäviksi.

Projektin ohjelmistokustannukset olivat 300 euroa, joka oli Red Hat Professional-Linux jakeluversion vähittäismyyntihinta. Paketti sisälsi kaikki tarvittavat työasema- ja palvelinohjelmistot. Samat tuotteet on saatavissa maksutta internet-verkosta, mutta paketoidun jakeluversion valmiit asennus-CD:t sekä fyysiset käyttöohjekirjat olivat projektinhallinnan kannalta riittävä peruste maksaa tämä hinta. Usean käyttäjän ympäristössä kustannukset eivät juurikaan tästä nouse, sillä ohjelmistopakettia voi käyttää rajoituksetta kuinka monta käyttäjää tahansa.

Esimerkkinä testiprojekti on toimiva, koska siinä käytetään sekä palvelin- että työasemaohjelmistoja -niillä voidaan toteuttaa huomattavasti laajempia ja yritystoiminnan kannalta kriittisempiäkin sovelluksia. Tuore esimerkki on suomalaisen Jarmo Rosenqvistin toteuttama ostoreskontraohjelmistoprojekti. Rosenqvist on lähes yksin ohjelmoinut sähköisen ostoreskontrajärjestelmän. Ohjelmiston avulla yrityksen

laskujen hyväksyminen, maksaminen ja vienti kirjanpitoon sekä tilintarkastus yksinkertaistuvat ja nopeutuvat merkittävästi. Ostoreskontra on mielenkiintoinen sovellusalue, koska siihen sisältyy useimmissa yrityksissä runsaasti manuaalista rutiinityötä, joka voitaisiin hoitaa huomattavasti elegantimmin sähköisessä muodossa. Ostoreskonta-ohjelmisto on suomenkielinen GPL-lisenssin alainen Open Source -ohjelmisto, joka on ladattavissa www.pupesoft.com -osoitteesta internetistä. Ohjelmiston tuottamiseen on käytetty samoja työkaluja kuin tutkielman esimerkkiprojektissa.

Tutkielman projektissa ohjelmoitiin sovelluksen käyttöliittymät, perustoiminnot sekä liitännät tietokantaan valmiiksi. Käyttöliittymät jätettiin varsin karkealle tasolle, koska ulkoasun viimeistelyn ei katsottu sisältävän erityisiä ongelmia ratkaistavaksi. Sovellus on toimiva muilta osin paitsi tekstiviestin lähettämisen osalta. Tekstiviestitoiminto puuttuu, koska kirjoittajalla ei ollut käytössään maksuttomia gateway-palveluita. Tämän toiminnon lisääminen tarvittaessa myöhemmin ei ole vaikeaa.

Sovellukseen ei myöskään ohjelmoitu kaikkia varmistuksia virheellisten asiatietojen tarkistamista varten. Tällaisia funktioita, kuten sähköpostiosoitteen syntaksin oikeellisuuden tarkistus, on runsaasti saatavissa verkosta maksutta. Open Source -projekteihin keskittyneillä verkkosivuilla, kuten freshmeat.net tai sourceforge.net on avoimien projektien lisäksi mittava määrä kierrätettäviä algoritmejä ja funktioita eri tarkoituksiin.

Sovelluksen ohjelmointiin kului aikaa noin sata työtuntia. Tämä aika sisältää myös käyttöjärjestelmän, www-palvelimen, tietokantapalvelimen ja php-modulin asentamisen. Näihin asennuksiin kokeneelta ylläpitäjältä menee tuskin kokonaista työpäivää. Kirjoittaja tutustui ensimmäistä kertaa kuhunkin näistä työvälineistä vasta valmistautuessaan asennuksiin, joten valmistelujen ja konfigurointien kanssa työympäristö oli käyttövalmis kolmessa työpäivässä. Tietokannan ja sovelluksen rakenteen suunnitteluun ei juuri aikaa tarvinnut käyttää, koska niitä oli mietitty jo kuukausia ennen toteutusta. Käytettävä tietokantaohjelmisto sallii melko suuretkin muutokset olemassa olevaan kantaan, joten uusia tauluja ja kenttiä voitiin lisätä aina tarvittaessa. Kokeneempi ohjelmoija olisi todennäköisesti kirjoittanut saman php-koodin alle viikossa. Toisaalta ilman käsitystä tietokantojen toimintaperiaatteista ja kokemusta kannan määrittelystä olisi peruskeskustelunkin luominen ohjelmakoodin ja tietokannan välillä lähes ylivoimainen tehtävä. Tämän projektin kustannukset voi laskea joko nyt käytettyjen työtuntien perusteella tai arvioimalla oman organisaation osaamistaso ja suhteuttamalla työmäärä siihen.

Kokemattoman ohjelmoijan yksittäisprojektina sovellukselle tulisi varsin kallis hinta, mutta näin ei tällaisia projekteja tulisikaan hoitaa. Parempi menettely olisi sellainen, jossa kokenut ohjelmoija tekisi verkkoon nopeasti yksinkertaisen version, ja sen jatkokehittelyyn ja testaukseen osallistuisi useampi koodaaja. Lisäksi tulee muistaa, että tämä on vain yksi sovellus.

Tällaisen toimivan, mutta vielä hieman robustin sovellusversion julkaiseminen asiakaskäyttöön ei ole suositeltavaa. Kehitystyötä voisi helposti jatkaa esimerkiksi organisaation sisäisenä Open Source -projektina. Vaikka käytössä ei olisi kovin monta ohjelmoijaa, useimmat työntekijät voivat toimia betatestaajina, ja antaa kehitysehdotuksia ja virheraportteja. Yrityksen sisällä voi samaan aikaan olla käynnissä useitakin työkaluohjelmistoprojekteja, joiden edistäminen perustuu vapaaehtoisuuteen. Projektinhallinnallisesti kannattaa miettiä vapauksien antamista joko tehtävälistan tai aikataulun osalta.

Tämän kaltaisen pilottiprojektin toteuttamisen jälkeen myös kuvitteellisessa yrityksessä olisi paremmat valmiudet osallistua Open Source -kehitykseen. Toiminnassa olevien Open Source -verkkosivustojen kautta kynnys osallistumiseen on hyvin matala, sillä mihinkään ei ole pakko sitoutua ja ohjelmia voi ladata vapaasti käyttöön. Esimerkiksi sourceforge.net -sivustolla oli huhtikuussa 2002 yli 37 000 avointa projektia. Joku niistä saattaisi hyvinkin olla crm-ohjelma, joka sopisi esimerkkiprojektin yhteydessä käytettäväksi.

Projektin tavoitteena olevan innovaation toteutus onnistui valituilla työvälineillä. Aikaisempia innovaatioita saattaa olla olemassa mutta niistä ei ollut tietoa projektin alkaessa. Mikäli vastaava kaupallinen tuote on olemassa, voidaan sen olettaa olevan ulkoasultaan viimeistellympi ja ominaisuuksiltaan monipuolisempi kuin tämän. Nyt toteutetun innovaation etu kaupallisiin kilpailijoihin nähden, on tavassa jolla sitä voidaan kehittää. Avoin lähdekoodi mahdollistaa tuotteen kehittämisen käyttäjäorganisaation toiveiden mukaisesti. Kehitys tapahtuu parhaassa tapauksessa maksuttomasti.

7 OPEN SOURCE -MALLIN SOVELTAMINEN KÄYTÄNTÖÖN

Open Source -malli ei ole joko hyvä tai huono. Menetelmään liittyy sekä heikkouksia että mahdollisuuksia. Tutkielmassa on jo aiemmin käsitelty lisensseihin ja juridiikkaan liittyviä sudenkuoppia. Tässä luvussa käsitellään kohdassa 7.1 vaikeuksia, joita liittyy vapaaehtoistyövoiman käyttöön sekä uskomusta jonka mukaan määrällä saadaan myös laatua. Myöhemmin kohdassa 7.2 käsitellään arkisemmalla tasolla tekijöitä, joita voi pitää mallin konkreettisina etuina myös suomalaisen julkishallinnon IT-yksiköille.

7.1 Open Source -mallin rakenteellisia riskejä

Tässä tutkielmassa on tutkittu Open Source -ohjelmistojen vaikutusta tietojärjestelmän kokonaiskustannuksiin. Empiirisesti tutkittiin valmiiden avoimeen lähdekoodiin perustuvien ohjelmistojen soveltuvuutta uuden ohjelmiston rakentamiseen. Tämän kokeilun lopputuloksena saatua ohjelmistoa voisi edelleen kehittää ja käyttää vaikkapa Open Source -menetelmää soveltaen.

Vaikka julkaistun koodin lukija ottaisi sen tarkasteluun tarkoituksenaan hyödyntää siitä osia, tämäkin voi poikia hyödyllisiä korjausehdotuksia tai joillekin rutiineille parempia vaihtoehtoja. Harvoin, ellei kyse ole varsinaisista liikesalaisuuksista, on muutoin mahdollista saada ulkopuolista ammattilaista kiinnostuneena käymään valmista ohjelmakoodia läpi.

Kehitysideoiden tulvaa odotellessa on kuitenkin syytä pitää jalat maan pinnalla, sillä tämän "ilmaisen reservin" käyttöön liittyy tekijöitä, jotka eivät kaikki ole projektin omistajan hallinnassa.

7.1.1 Vapaaehtoisia ei voi pakottaa

Käytettävissä on periaatteessa maailmanlaajuinen ohjelmoijayhteisö omine kokemuksineen ja verkostoineen. Ohjelmointiyhteisön voi tietyin varauksin rinnastaa äänestäjäjoukkoon. Molemmissa ryhmissä innostukset nousevat ja laskevat, ja kiinnostuksen painopiste eri asioihin vaihtelee ajanjaksottain. Ohjelmoijayhteisöön ei tule suhtautua ilmaisena työvoimana. Kyse on kriittisestä ja käyttäytymiseltään vaikeasti ennustettavasta joukosta, jonka työpanos perustuu vapaaehtoisuuteen ja ohjelmoinnin tuottamaan tyydytykseen. Yhteisö osallistuu sellaisiin ohjelmointihankkeisiin, jotka se

näkee riittävän haasteellisina ja hyödyllisinä. Yhteisö voi olla lähes rajaton voimavara, mutta sen sääteleminen voi osoittautua mahdottomaksi. Projektin eteneminen saattaa pysähtyä, tai aktiiviset osallistujat saattavat yrittää kaapata projektin itselleen ja alkaa linjata sitä mieleiseensä suuntaan. Projektin kaappaamista on mahdoton täysin estää, sillä kyse on vapaasti kopioitavista ja kehiteltävistä ohjelmista. "Kapinajohtajan" persoonasta ja ideoista riippuu, kumpaan projektiin muut ohjelmoijat haluavat osallistua.

Lähdekoodin avoimuus ei yksin ratkaise vaikean projektin ohjelmointiongelmia. Vaikka lähdekoodin avoimuus mahdollistaakin ohjelman ominaisuuksien arvioimisen, ei arviointitehtävä ole missään tapauksessa läpihuutoasia. Ohjelmat voivat olla laajoja ja niiden dokumentointi on saatettu laiminlyödä. Ohjelmien syvällinen analysointi turvallisuusominaisuuksien osalta onnistuu vain osalta ohjelmoijista, eikä se ole yhtä yksinkertaista kuin ohjelman loogisuuden ja toiminnan tarkastaminen. Näin ollen monimutkaisten ohjelmien turvallisuuden arviointia ei aina voi luotettavasti toteuttaa ilmaiseksi. Useimmat ohjelmoijat eivät tiedä paljon turvallisuusseikoista ellei heitä ole siihen erityisesti koulutettu.

Yhteisön jäsenet tekevät muutoksia tai lisäyksiä koodiin useimmiten silloin, kun he löytävät siitä puutteita. Open Source -mallin yhteydessä usein toistettu mantra kuuluu: "Miljoona silmäparia saa kaikki virheet häviämään". Lukuisat silmäparit tutkivat tosin vain niitä moduuleita, joihin heidän kiinnostuksensa kohdistuu. Kokonaisuuden hahmottaminen jää vähemmälle huomiolle. Ohjelmistoissa kokonaisuuden toimintaperiaatteiden suunnittelemista ja toteuttamista pidetään usein vaikeimpana tehtävänä. Punaisen langan ja jatkuvuuden puuttuminen prosessista saattaa muodostua riskiksi ohjelman turvallisuudelle sekä aiheuttaa umpikujia ohjelman jatkokehityksessä.

Ensimmäiset versiot ovat usein epävakaita ja virheitä täynnä. Tarvitaan useita versioita ennen kuin koodin laatu on kaupallisten ohjelmien tasolla. Tosin versioiden julkaisemisen välillä voi kulua ehkä vain tunteja, mikäli ohjelman yhteisö on laaja ja aktiivinen. Vaikka väitetään, että koodin tarkastelu johtaa koodin korkeaan laatuun, myös laajalle levinneistä avoimen lähdekoodin ohjelmista löydetään jatkuvasti merkittäviä virheominaisuuksia, jotka ovat olleet siinä jo vuosia havaitsemattomina. Näin tapahtui esimerkiksi nimipalvelinohjelma *Bind*issa sekä sähköpostipalvelin *Sendmail*issa. Virheiden löytyminen vasta vuosien päästä voi johtua myös siitä, että aiemmin lähdekoodin lukeminen ei ole ollut erityisen suosittua. Silmäparien lisääntyessä virheitä löydetään myös vanhoista ohjelmista.

7.2 Julkishallinnolle useita etuja Open Source -mallin avulla

Laadullisesti Open Source -ohjelmistojen ei pitäisi olla julkishallinnolle erityinen riski. Open Source -projektimallilla kehitetyt ohjelmat ovat osoittautuneet tutkimuksissa laadukkaammiksi kuin vastaavat kaupalliset ohjelmat [Miller et al.1989, 1998, Forrester and Miller 2000]. Näissä tutkimuksissa havaittiin, että vapaan lähdekoodimallin tuottamat käyttöjärjestelmät ja niiden perustyökalut sisälsivät vähemmän virheitä, ja niiden luotettavuus oli merkittävästi parempi kuin kaupallisten Unixien vastaavat ohjelmat.

Linux -käyttöjärjestelmää ei lueta varsinaisiin *ix -käyttöjärjestelmiin, kuten Unix:n eri versiot ja IBM:n AIX. Sillä ei tämän tutkielman eikä luotettavuustestien tulosten käsittelyn kannalta ole suurta merkitystä. Kaikkia näitä käyttöjärjestelmiä voidaan käyttää tutkielman kannalta samoissa tehtävissä, joten ne asettuvat arvioitaessa samalle viivalle.

7.2.1 Kustannussäästöt helpoin hahmottaa

Valtion ja kuntien virastot, yhtiöt sekä muut verovaroin toimivat yhteisöt pitäisi ehkä jopa velvoittaa tukemaan Open Source -tuotteita ja käyttämään mahdollisimman leveällä rintamalla avoimen lähdekoodin ja avointen standardien mukaisia ohjelmia. Valtionyksiköiden laitteisto-, ohjelmisto- ja palveluhankintoja säätelevät jo nykyisin ohjeet kilpailuttamisesta ja kokonaiskustannusten huomioon ottamisesta.

Kuntien infrastruktuureissa on runsaasti samankaltaisuuksia ja mikäli toiminnoissa päästään hyödyntämään synergiaetuja, taloudelliset säästöt saattavat muotoutua mittaviksi. Yleisimmät ohjelmistotarpeet kuten yritysten ja julkishallinnon tukitoiminnot ovat varsin samankaltaisia.

Ohjelmistot tulisi kehittää modulaarisesti, jotta pienet ja isot kunnat voivat hyödyntää periaatteessa samoja ohjelmia, ja erityistarpeisiin voidaan ladata lisäoptioita. Erityistarpeita muodostavat esimerkiksi maantieteelliset erikoisalueet, kuten saaristo, Lapin pohjoisimmat ja harvaanasutuimmat alueet, tai alueet joilla poikkeuksellinen suuret ammatinharjoittamiseen liittyvät painotukset. Elektronisten kassapäätteiden alkuaikoina myyjät torjuivat asiakkaiden tinkimisyrityksiä vetoamalla väitteeseen, että tuotteiden hinnat oli ohjelmoitu tietokoneelle. Vaikka väite saattoi joskus olla tottakin, tällaisia väitteitä eivät asiakkaat enää hyväksy. Myöskään julkishallinnossa tietotekniikka ei saa määrätä tai rajoittaa asioiden hoitoa. Mikäli ohjelmissa havaitaan puutteita tai

ohjelman toimintalogiikka on käytännön elämälle vieras, tarpeellliset muutokset on pystyttävä toteuttamaan kohtuullisessa ajassa kohtuullisin kustannuksin.

7.2.2 Kuntien vapaus it-ohjelmistojen valinnassa lisääntyy

Open Source-malli ja siihen olennaisena käsitteenä liittyvä modulaarisuus tuo julkisyhteisöihin lisää vapautta ohjelmistojen ja laitteistojen valinnassa. Palvelinpohjaiset ratkaisut toimivat TCP/IP-protokollan kautta ja käyttöliittymänä toimii verkkoselain. Työasemana toimii mikä tahansa päätelaite, jolla on mahdollisuus liittyä kunnan intranet-verkkoon. Ohjelmia ei tarvitse pitää erityisinä internet-ohjelmina, vaan normaaleina asiakas-palvelin -sovelluksina. Internetin yleistyminen on tosin tuonut runsaasti valinnanvaraa ja mahdollisuuksia toteuttaa myös lähiverkon ohjelmia.

Kuntien IT-hallinnolla olisi avoimen lähdekoodin aikana mahdollisuus valita, millä ohjelmilla kunnan palvelimet kuormitetaan (mitkä optiot ovat käytössä) sekä mitkä versiot ohjelmista otetaan käyttöön. Lainsäädännön muuttuessa osa vanhoista jää ohjelmaversioista käyttökelvottomiksi. Näin tapahtunee tulevaisuudessakin, mutta Open Source -mallissa vanha ohjelmisto ei välttämättä muutu käyttökelvottomaksi. Riittää, kun joku jossakin (tässä tapauksessa todennäköisesti Suomessa, johtuen kielestä) internet-verkon vaikutuspiirissä ottaa käsittelyynsä vanhan ohjelman lähdekoodin, tekee siihen tarvittavat muutokset ja lähettää uuden version muiden saataville.

8 YHTEENVETO

Tässä luvussa tehdään yhteeveto tutkielman tuloksista ja havainnoista. Tutkielmassa tehtyjä havaintoja käsitellään tarkemmin kohdassa 8.1. Kohdassa 8.2 selvitetään mitä lukijan tulee ottaa huomioon pohtiessaan tutkielman tuloksia. Kohdassa 8.3 käsitellään tuloksia yhdistettynä ne muutamiin käytännön suosituksiin. Kohta 8.4 ja sen alakohdat 8.4.1-8.4.3 esittelevät Open Source -ohjelmistojen jatkotutkimukselle kolme vaihtoehtoista suuntaa.

8.1 Havainnot

Open Source -malli on tullut jäädäkseen osaksi informatioteknologiaa. Mallia ei tarvita kaikissa ohjelmistoprojekteissa, mutta sen kategorinen sivuuttaminenkaan tietotekniikkaa käyttävissä yhteisöissä ei olisi viisasta. Open Source -ohjelmistojen käyttö tuottaa välittömän kustannussäästön elinkaarensa alussa, koska käytöstä ei synny hankinta- eikä lisenssimaksuja. Suurin osa ohjelmistojen kokonaiskustannuksista koostuu käytönaikaisista kustannuksista. Käytönaikaisten kustannusten sovittaminen yleiskäyttöiseen kustannuslaskentakaavaan on ongelmallista sovelluskohteiden infrastrukstuurin suuren vaihtelun vuoksi.

Kustannukset voidaan selvittää analysoimalla organisaatiota ja hankkimalla sitä kautta kertoimet käytönaikaisten kustannustekijöiden arvioimiseen. Käytönaikaisten kustannussäästöjen mahdollisuus syntyy Open Source-tuotteiden teknisistä ja ohjelmallisista ratkaisuista, ei siitä, että niillä ei ole varsinaista hankintahintaa. Lähdekoodin avoimuus mahdollistaa ohjelmien sopeuttamisen toimintaympäristön muutoksiin myös käytön aikana, mutta muutostyöt edellyttävät ammattitaitoista henkilökuntaa.

Käytettäessä avoimen lähdekoodin ohjelmistoja IT-henkilöstö voi hyödyntää toimivia Open Source -portaaleja, joissa on runsaasti valmiita ohjelmia, algoritmeja ja apuohjelmia. Ne voidaan integroida saumattomasti muihin vastaavilla työkaluilla toteutettuihin ohjelmiin.

Avoimen lähdekoodin käyttö vähentää tehokkaasti riskiä lukittua tiettyyn ohjelmistoon tai toimittajaan. Lähdekoodi voidaan saada käyttöön myös maksullisten ohjelmaprojektien myötä, mutta valmisohjelmat eivät juuri koskaan tätä mahdollisuutta tarjoa.

Open Source -mallin mukainen ohjelmistokehitys ei tule olemaan ainoa oikea vaihtoehto ohjelmistotuotannon tulevaisuudessa. Lähdekoodin uudelleenkäyttöä eri muodoissa on ollut todennäköisesti yhtä kauan kuin tietokoneohjelmia on kirjoitettu. Kymmenen viime vuoden aikana tietotekniikan kehitys on mahdollistanut koodin jakamisen entistä helpommin entistä laajempien ryhmien käyttöön entistä nopeammin. Projekteihin voi osallistua valtava joukko vapaaehtoisia ammattilaisia ja joistakin yhteishankkeina tuotetuista ohjelmista tulee pysyvä osa ajan tietotekniikkaa. Vastaisuudessakin tulee olemaan ohjelmistotaloja, joiden tuotteet perustuvat maksulliseen suljettuun koodiin ja joilla riittää maksavia asiakkaita. Todenn tullaan myös huomaamaan lähdekoodin avoimuuden ja tietojärjestelmiin avoimen pääsyn ero. Vaikka yrityksen laskutusjärjestelmä toteutetaan ilmaisen tietokantaohjelmiston avulla, se ei merkitse että tuhannet tai miljoonat uteliaat pääsevät avoimesti tietoihin käsiksi, sillä samat tuhannet ja miljoonat silmäparit ovat tehokkaasti tukkineet vapaan lähdekoodin ohjelmistosta tunkeutujan mentävät aukot. Huomattavasti yksinkertaisempaa on murtautua henkilöautoon tai omakotitaloon.

Ohjelmistolisenssit muodostavat prosentuaalisesti varsin pienen osan ohjelmistotuotannon kuluista. Aika on resurssi, jota eniten tarvitaan. Projektille voidaan joko antaa pitkä toteutusaikataulu tai kiinnittää useampia suunnittelijoita tekemään työ lyhyemmässä ajassa. Ensimmäinen vaihtoehto antaa mahdollisuuden jakaa suunnittelijan palkkakustannukset pidemmälle ajanjaksolle, mutta sisäänrakennettu heikkous on ohjelman käytöstä saatavien hyötyjen ja ansainnan siirtyminen kauemmaksi tulevaisuuteen. Valmistuvan sovelluksen luonne ratkaisee, mikä painaa vaakakupeissa eniten. Open Source -mallissa jaetaan ohjelmointityö useammalle koodaajalle, ja samalla jaetaan myös tarvittavaa aikakuormaa.

Open Source ohjelmistojen helpoimmin mitattava kustannusetu on elikaaren alkupään hankintakustannusten puuttuminen. Yleensä avoimen lähdekoodin ohjelmat ovat ilmaisia, mutta myös kaupallisten projektien lähdekoodi voidaan luovuttaa asiakkaalle. Lähdekoodin ja käytettyjen tietorakenteiden haltuunsaaminen on asiakkaan valttikortti mietittäessä ohjelmiston tai toimittajan vaihtamista. Avoimen lähdekoodin -mallin mukaiset projektit sekä niiden tuotteena syntyvät ohjelmistot ovat vähemmän haavoittuvia organisaation sisäisille sekä ulkopuolisille muutoksille, koska nämä ohjelmat voidaan mukauttaa olosuhteiden muutoksiin vaivattomammin kuin suljetut järjestelmät.

Avoimen lähdekoodin käyttöön usein liittyy myös avointen standardien hyödyntäminen. Avoimet standardit ovat ohjelmiston hankkijalle pienimuotoinen vakuutus siitä, ettei kaikkea tarvitse tehdä alusta alkaen uudestaan, mikäli jossain vaiheessa halutaan vaihtaa

ohjelmistotoimittajaa tai ylläpitopalveluita toimittavaa kumppania. Avointen standardien käytön voidaan ajatella turvaavan myös ohjelmistotoimittajan selustaa, vaikka usein toisin ajatellaan ja päinvastaisesta käytännöstä on esimerkkejä. Julkistaessaan käyttämänsä formaatit ja protokollat voi ohjelmiston kehittäjä tavoitella laajempaa tukea tuotteilleen ja hakea vahvempia asemia markkinoilla.

Hankinta- ja lisenssimaksujen puuttuminen on Open Source -ohjelmien ilmeisin kustannusetu, joka on myös helppo todentaa laskelmalla. Suurin osa tietojärjestelmien kustannuksista muodostuu kuitenkin käytönaikaisista kuluista. Nämä kustannukset koostuvat useista eri tekijöistä, joiden painoarvo vaihtelee suuresti riippuen sovelluskohteesta. Käytönaikaiset kustannukset voidaan luotettavasti laskea vasta kohteen infrastruktuurin huolellisen analyysin jälkeen, eikä yleiskäyttöisen kaavan tuottamia tuloksia voi pitää vakuuttavina.

Avoimen lähdekoodin kustannusedut eivät tule käyttäjille automaattisesti. Parhaan edun Open Source -ohjelmista saavat organisaatiot, joissa on IT-osaamista ja omaksumiskykyä omasta takaa. Tällöin voidaan hankintatilanteessa toimia analyyttisemmin ja ratkaisuvaihtoehtoja voidaan löytää muualtakin kuin vanhan toimittajan valikoimista.

8.2 Tutkielman rajoitukset

Vasta tutkielman kirjoittamisen aikana selvisi, kuinka suuren osan tietojärjestelmän kokonaiskustannuksista käytönaikaiset kulut muodostavat. Mikäli tästä olisi tutkielmaa suunniteltaessa ollut selkeämpi käsitys, olisi lähtökohta muodostunut hieman toiseksi.

Tutkimuksen kohteeksi olisi ollut mahdollista määritellä fiktiivinen käyttöympäristö, jonne olisi määritelty palvelutarpeet. Näiden tarpeiden toteuttamisen vertailu Open Source -tuotteilla ja kaupallisilla tuotteilla olisi tuottanut konkreettisempia tuloksia käytönaikaisista kustannuksista. Fiktiivinen ympäristö olisi voinut olla oikeakin yritys tai julkishallinnon yksikkö, jolloin henkilöstöresurssin osaamiseen ja koulutustarpeeseen olisi voinut ottaa kantaa.

Erillisen ohjelmistoprojektin painoarvo tutkielmassa jäi ehkä hieman kevyeksi. Välttämättä kokonaiskustannuksia tutkittaessa ei tarvittaisi esimerkin kaltaista ohjelmointityötä, ellei tuotetta verrata esimerkiksi vastaavaan kaupalliseen tuotteeseen. Käytännön tasolla projektista oli suuresti hyötyä. Ilman konkreettista ohjelmointityötä, palvelimien asennusta sekä tiedonhakumatkoja Open Source -yhteisöjen ja työkaluohjelmistojen verkkosivuille kirjoittajan käsitys Open Source-mallista ja -projektien toiminnasta olisi jäänyt huomattavasti ohuemmaksi.

Tutkielmassa puhuttiin tietojärjestelmistä, ohjelmista ja ohjelmistoista. Joissakin kohdissa saattoi jäädä lukijan vastuulle päätellä, kuinka laajaa ohjelmakokonaisuutta tekstissä kulloinkin käsiteltiin. Tutkielman termeihin kuului myös *ohjelmistoprojekti*. Aina ennen ohjelmistoprojektien alkua on tehtävä perusteelliset määrittelyt lopputuotteesta. Tämä kuuluu järjestelmän elinkaaren luokkaan (r). Rakennusvaiheen kustannukset jäivät minimaaliselle huomiolle tutkielmassa, eikä tutkielma siltä osin anna oikeaa kuvaa kustannusten kertymisestä. Eräs syy tapahtuneeseen saattaa olla valitun materiaalin vaikutus, sillä käytetyt esimerkit ja taulukot keskittyivät valmisohjelmistojen kustannustekijöihin.

8.3 Käytännön suosituksia

Tutkielmassa tarkasteltiin ohjelmiston kustannuksia koko ohjelman elinkaaren ajalta alkaen hankinnasta ja päätyen ohjelman poistamiseen käytöstä. Ohjelmiston tuotantojakson, ennakoivan huollon ja poiston aikaiset kustannukset menevät osittain päällekkäin, ja organisaation tarpeet ja omat resurssit vaikuttavat siihen kuinka paljon synergiaetuja valitut ratkaisut mahdollistavat. Tämän vuoksi yleiskäyttöistä kaavaa ei käytönaikaisten kustannusten laskemiseksi katsottu järkeväksi tämän tutkielman yhteydessä muodostaa. Kaavojen tuottamiseksi tulisi tutkia joitain esimerkkiyrityksiä, määritellä palvelut jotka halutaan toteuttaa ja arvioida henkilöstön osaaminen sekä käyttöönoton että tulevien järjestelmien käytön osalta. Muita analysoitavia tekijöitä ovat muun muassa vanhojen työasemien ja palvelimien suorityskyky ja uusimistarve, säilytettävien ohjelmien yhteistoiminta uusien kanssa, ylläpito- ja tukitarve, ylläpidon automatisointi mahdollisuudet. Tällaisen työn voi arvioida olevan laajuudeltaan samaa luokkaa tämän tutkielman kanssa.

Erilaisten palveluiden ja organisaation välttämättömienkin tukitoimintojen ulkoistamisen pohtiminen on ikuisuuskysymykseltä vaikuttava kuuma peruna. Ulkoistamisen hyödyt ja haitat, samoin kuin perusteet, joilla päätökset ulkoistamisesta tehdään, ovat kovin erilaiset yritysmaailmassa ja julkishallinnossa. Tällä hetkellä ajan henki pikemminkin suosii informaatioteknologian palveluiden ja tuotteiden ostamista, kuin niiden suunnittelua ja tuottamista organisaation sisällä.

Hankkiessaan talon ulkopuolisin voimin tuotetun ohjelmiston ostaja ei periaatteessa hyödy lisenssimaksujen poisjäännistä, vaikka tuote olisi toteutettu Open Source-työkaluilla. Ohjelman ostaja maksaa työajasta, joka tuotteen valmistamiseen on käytetty. Edun saa tällöin ohjelmiston valmistanut ohjelmistoyritys, mikäli se käyttää avoimen lähdekoodin piirissä olevia työkaluja ja tuotteita. Edullisemman valmistushinnan pitäisi

periaatteessa näkyä lopputuotteen hinnassa, mutta näkyykö se siinä, on eri asia. Tällä ei ohjelmiston hankkivan yrityksen näkökulmasta ole suurta merkitystä, sillä ohjelmiston hankintahinta hankintakuluihin on vain pieni osa ohjelmiston kokonaiskustannuksista.

Kun IT-osaaminen on tarpeeksi pitkälle ulkoistettu, ei asiakasorganisaatiossa ole omaa henkilökuntaa koordinoimaan projekteja ja hyödyntämään Open Source -mallin etuja. Kyse ei ole niinkään osaavista ohjelmoijista vaan pikemminkin päätöksentekijöistä ja linjaratkaisuista. Tämä ei kuitenkaan koske vain Open Source -projekteja, vaan kaikkia organisaatioiden it-hankintoja ja -valintoja.

Tietohallintopäättäjä, jolla on vaatimaton kokemus ja vain vähän näkemystä alan kehityksestä, saattaa tehdä toimiviakin hankintoja, jos hän käyttää osaavaa konsulttia projektien valmistelussa ja valvonnassa. Tästä ei suoraan seuraa, että näin saadut ratkaisut olisivat kokonaiskustannuksiltaan edullisimmat tai edes toimivimmat tai kuka tälläisessä kuviossa oikeastaan on organisaation IT-päättäjä.

On myös muistettava, että vaikka ohjelmistotoimittaja käyttäisi avoimen lähdekoodin työkaluja, niillä valmistetut lopputuotteet eivät automaattisesti ole avoimia. Avoimuus lopputuotteen lähdekoodissa ja standardeissa pienentää lukittumisriskiä. Ohjelmakoodin saatavuus ja standardiratkaisujen käyttäminen esimerkiksi tietokantaratkaisuissa ovat etuja, jotka ostaja saa riippumatta siitä osaako tai haluaako hän niitä myöhemmin käyttää.

8.4 Jatkotutkimusmahdollisuudet

Open Source -tuotteet ja niiden soveltuvuuden arviointi kuhunkin organisaatioon tulisi mielestäni jakaa kolmeen lohkoon. Samaa jakoa voi noudattaa niin yksityissektorilla kuin julkishallinnossa.

8.4.1 Toimistosovellukset

Toimistosovelluksiin luen perinteiset office-paketit sekä sähköposti- ja internet-selainohjelmat. Näiden valinnassa keskeisempää kuin lähdekoodin vapaa saatavuus on varsinaisten ohjelmien maksuttomuus. Näillä voidaan hakea lyhyenkin tähtäyksen kustannussäästöjä ohjelmistolisenssimaksuissa. Open Source -toimistosovellusten käyttöönotto- ja oppimiskynnyksen ei pitäisi olla suurempi kuin vastaavan kaupallisen ohjelman uuden päivitysversion opiskelu. Open Source -toimistosovelluksissa on graafinen käyttöliittymä ja komentojen haku ja asetusten teko noudattavat samoja peruslogiikoita kuin graafisilla käyttöliittymillä varustetut ohjelmat ylipäätään. Erilaiset

kansalliset oikoluvut ja sanastot ovat Open Source -ohjelmien akilleen kantapää, sillä ne eivät yleensä valmistu ohjelmistojen kanssa samaan tahtiin.

8.4.2 Palvelin- ja tietokantatuotteet

Palvelin- ja tietokantaohjelmistot ovat väylä, jota pitkin tähän mennessä avoimen lähdekoodin tuotteet ovat levinneet tehokkaimmin yritysten ja organisaatioiden konehuoneisiin. Useimmin nämä ovat tulleet ikään kuin pakon sanelemana keittiön kautta; on tarvittu jokin palvelu, mutta määrärahoja ei ole ollut sen hankkimiseen. Open Source -projektein tuotettuja ohjelmistoja on ollut maksuttomasti tai hyvin edullisesti saatavissa, ja nämä tuotteet ovat ratkaisseet ongelman.

Avoin lähdekoodi on toiminut ikään kuin etukäteisvakuutuksena ohjelmien vaarattomuudesta jo olemassaolevaan ympäristöön nähden. Lukuisat silmäparit ovat käyneet ohjelmat jo moneen kertaan läpi, joten todennäköiset turvallisuusaukot on havaittu ja paikattu. Avoimen lähdekoodin ohjelmissa on noudatettu voimassa olevia standardeja mahdollisimman pitkälle, jotta mahdollisimman laaja leviäminen ja ohjelmien yhteensopiminen muiden saman aikakauden ohjelmien kanssa on voitu varmistaa.

Ohjelmien lähdekoodin avoimuus tietoliikenteen suurten yhetysnopeuksien aikana antaa kuitenkin mahdollisuuden myös pahantahtoisten kokeilijoiden ja soveltajien häiriköinnille. Pessimistisesti ajatellen voi pitää vaarana sitä, että yhä kasvava häirikköjoukko kykenee entistä lyhyemmässä ajassa toteuttamaan massiivisia binaarihyökkäyksiä asiallisesti toimivaa it-yhteisöä kohtaan.

Onneksi ohjelmistojen ja tekniikan kehitys tukee vastaavassa määrin myös erilaisten tietoturvallisuuteen liittyvien tuotteiden evoluutiota. Tietoturvatekijöihin liittyvät projektit saattavatkin vastaisuudessa nousta yhä keskeisemmiksi niin olemassa olevien ohjelmistotuotteiden piirissä kuin uusina hankkeina.

8.4.3 Ohjelmistotuotanto

Ohjelmistotalon ei tarvitse lopettaa omien kaupallisten ohjelmien myymistä tai kehittämistä, vaikka se ottaisi valikoimiinsa Open Source -tuotteita. Open Source -mallin mukaan tuotetut ohjelmat voivat toimia vanhan tarjonnan laajennuksina tai lisäpalveluina. Yhteensopivuusongelmia ei ainakaan pitäisi esiintyä, koska vanhojen ohjelmistojen kehittäjä toimii lisäosien Open Source -koordinaattorina. Vaarana tässä on,

että Open Source-osiin tutustumalla kilpailijoiden on mahdollista hahmotella kaupallisen ohjelman tietorakenteita. On mielipidekysymys, onko tämä minkäänlainen vaara ensinkään - käytetyissä tietorakenteissa ei välttämättä ole mitään niin mullistavaa, jota kilpailija ei voisi omatoimisesti kehittää. Mikäli suljetut tiedostomuodot tai muut rakenteen peittämiseen perustuvat tekijät ovat syitä, jonka vuoksi ohjelmistotalon asiakkuudet pysyvät samalla toimittajalla, on aikapommi jo alkanut tikittää. Mikäli asiakkaan uskollisuus ohjelmistotoimittajalle on jokin muu kuin ohjelman hyvät ominaisuudet, hinta tai sen soveltuvuus hänen tarpeisiinsa, on vain ajan kysymys milloin asiakas törmää kilpailijan tuotteeseen, joka hurmaa ostajan. Paluuta vanhaan suhteeseen ei silloin enää ole.

LÄHTEET

[ATK-sanakirja 1996] Tietotekniikan liitto ry:n sanastotoimikunta, *ATK-sanakirja*. Suomen ATK-kustannus, Espoo, 1996.

[Briand et. al. 1999] Lionel C. Briand, Khaled El Emam, Dagmar Surmann, Isabella Wieczorek and Katrina D. Maxwell, *An assessment and comparison of common software cost estimation modeling techniques*. Proceedings of the 1999 international conference on Software engineering. IEEE Computer Society Press, Los Alamitos, CA, USA 1999 .

[Brynjolfsson and Yang 1997] Erik Brynjolfsson and Shinkyu Yang, *The intangible benefits and costs of investments: evidence from financial markets*. Proceedings of the eighteenth international conference on Information systems. 1997, Atlanta, United States

[David et. al. 2002] Julie Smith David, David Schuff and Robert St. Louis, *Managing Your IT Total Cost of Ownership*. ACM 45, No 1, 101-106.

[DeMarco and Lister 1987] Tom DeMarco and Timothy Lister, *Peopleware: Productive Projects and Teams*. Dorset House Publication, New York, 1987.

[Dryden 1998] Patrick Dryden, *'Futz factor' measurement tough to pin down in TCO*. Computerworld, April 13. 1998. saatavilla: www.computerworld.com/cwi/story/0,1199,NAV47_STO30535,00.html

[Forrester and Miller 2000] Justin E. Forrester and Barton B. Miller, *An Empirical Study of the Robustness of Windows NT Applications Using Random Testing*. University of Wisconsin 2000, ftp://grilled.cs.wisc.edu/technical_papers/fuzz-nt.pdf

[Frisch 1997] Æleen Frisch, *UNIX - Tehokäyttäjän opas*. Suomen Atk-kustannus. Jyväskylä 1997.

[Gillen et al. 2001] Al Gillen, Dan Kusnetzky and Scott McLarnon, *The Role of Linux in Reducing the Cost of Enterprise Computing*. IDC, Framingham, Massachusetts, 2001.

[Hackett 2001] Hackett Benchmarking & Research, *Chief Findings Of Multi-Year Study of Information Technology at Global 2000 Companies: Worries About Prioritization, Increases in Complexity And Difficulty Recruiting Top Talent*. Tiivistelmä tutkimuksesta. Linkki tarkistettu 27.3.2002. Saatavilla: http://www.prnewswire.com/cgi-bin/stories.pl?ACCT=104&STORY=/www/story/03-07-2001/0001443064&EDATE=

[Järvinen ja Järvinen 2000] Pertti Järvinen ja Annikki Järvinen, *Tutkimustyön metodeista.* Opinpajan kirja, Tampere, 2000.

[Kemerer 1987] Chris F Kemerer, *An empirical validation of software cost estimation models*. Communications of the ACM, Volume 30, Issue 5, Pages: 416 - 429.

[Kemppinen 2001] Jukka Kemppinen, *Tietokoneohjelman oikeussuoja*. Helsingin teknillisen korkeakoulun julkaisu. Helsinki 2001.

[Lagus 2002] Antti J. Lagus, *Asp-markkinat vielä lähtökuopissaan*. Tietokone-lehti 3/2002. Helsinki Media Oy. Helsinki 2002.

[Messerschmitt and Szyperski 2000] Messerschmitt, D.G., Szyperski, C. *Industrial and Economic Properties of Software: Technology, Processes, and Value.* University of California at Berkeley Computer Science Division Technical Report, Berkeley, 2000.

[Microsoft 2001a] Microsoftin tekemä TCO-laskelma Vaasan kaupungin tietotekniikkakuluista, valmistumisajankohta 22.11.2001. Saatavilla: http://www.microsoft.com/finland/business/tco/vaasa.pdf

[Microsoft 2001b] Tiivistelmä Microsoftin tekemästä TCO-laskelmasta Lappeenrannan kaupungin tietotekniikkakuluista, valmistumisajankohta tammikuu 2002. Saatavilla: http://www.microsoft.com/finland/business/tco/lpr.pdf

[Mikrolog 1992] Mikrolog Oy, *Tuoteluettelo ja hinnasto 2/92*. Oy Mikrolog Ltd, Keuruu 1992.

[Miller et al. 1989] Miller, B. Fredriksen, B. So: *An Empirical Study of the The Reliability of UNIX Utilities* 1989, ftp://grilled.cs.wisc.edu/technical_papers/fuzz.pdf

[Miller et al. 1998] Miller B., Koski D., Lee C. P., Maganty V., Murthy R., Natarajan A., Steidl J.: Fuzz Revisited : *A Re-examination of the Reliability of UNIX Utilities and Services*. University of Wisconsin 1998, ftp://grilled.cs.wisc.edu/technical_papers/fuzz-revisited.pdf

[MySQL 2001] MySQL Technical Reference for Version 4.0.1-alpha. Saatavilla mm. http://www.mysql.com/documentation/

[Nelson et. al. 1996] Paul Nelson, William Richmond, Abraham Seidmann, *Two Dimensions of Software Acquisition*. Communications of the ACM ,Vol. 30, No. 7, p.29-35.

[Onnela 2001] Eija Onnela, *Loppuraportti OpenOffice-työasemaohjelmiston ja Linux-käyttöjärjestelmän soveltuvuudesta kaupungin työasemastandardiksi*. Turun kaupungin tietotekniikkaosasto 17.12.2001.

[Peeling et. al. 2001] www.govtalk.gov.uk/documents/QinetiQ_OSS_rep.pdf

[Perens 2001] B. Perens: Open Source Definition Version 1.9, 22. Helmikuuta 2002 http://www.opensource.org/osd.html

[Pressman 1997] Roger S. Pressman, *Software Engineering - A Practitioner's Approach*. McGraw-Hill, 1997.

[Raymond 2000] Eric S. Raymond, *The Cathedral and the Bazaar-Musings on Linux and Open Source by an Accidental Revolutionary*. O'Reilly & Associates, Inc. Sebastopol, California, 2000. saatavilla myös: http://www.tuxedo.org/~esr/writings/cathedral-bazaar/

[Schneider 2000] C. Schneider, *Hold on to your wallet! Are chief information officers ruining CFOs' budgets?* CFO Magazine. November 22, 2000.

[Rai et. al. 1997] Arun Rai, Ravi Patnayakuni, and Nainika Patnayakuni, *Technology Investment and Business Performance*. Communications of the ACM ,Vol. 40, No. 7, p.89-97.

[Tekijänoikeuslaki 1993] Tekijänoikeuslaki 11b § ja 11c §. Helsingissä 7 päivänä toukokuuta 1993.

[Tietokone 2001] Tietokone -lehti 12/2001, *Computeria Oy-tuotehinnasto*. Helsinki Media Oy, Helsinki 2001.

[Turun kaupunki 2001] Turun kaupunki, *Selvitystyön tilaaminen OpenOffice työasemaohjelmiston ja Linux käyttöjärjestelmän soveltuvuudesta kaupungin työasemastandardiksi*. Turun kaupungin tietohallinnon johtoryhmä § 30, 5.9.2001 sekä Turun kaupunginhallitus § 906, 10.9.2001.

[UK PatentOffice] Ison-Britannian patenttivirasto, 22.2.2002, www.patent.gov.uk/patent/notices/practice/computer.htm

[Valtiovarainministeriö 2001a] Valtiovarainministeriön yhteenveto, *Tietoja valtion tietohallinnosta ja tietotekniikasta 2000*. Valtiovarainministeriö, Hallinnon kehittämisosasto, Helsinki 2001

[Valtiovarainministeriö 2001b] Valtiovarainministeriön työryhmämuistioita 27/2001, *Valtiontietotekniikan rajapintasuosituksia.* Valtiovarainministeriö, Hallinnon kehittämisosasto, Helsinki 2001

[Valtonen 2001] Mato Valtonen: *Noh, sano naakka ku nokka katkes.* Tammi, Helsinki 2001.

[www.netcraft.com] Netcraft Web Server Survey – Active Sites,22.1.2002, http://www.netcraft.com/Survey/