AVOIMEN LÄHDEKOODIN OHJELMISTOPROJEKTIT TERVEYDENHUOLLOSSA

Mikko Huovila

Pro gradu -tutkielma

Sosiaali- ja terveydenhuollon tietohallinto
Itä-Suomen yliopisto
Sosiaali- ja terveysjohtamisen laitos

Joulukuu 2010

ITÄ-SUOMEN YLIOPISTO, yhteiskuntatieteiden ja kauppatieteiden tiedekunta
sosiaali- ja terveysjohtamisen laitos, sosiaali-ja terveydenhuollon tietohallinto

HUOVILA MIKKO: Avoimen lähdekoodin ohjelmistoprojektit terveydenhuollossa

Pro gradu -tutkielma, 84 sivua, 1 liite (3 sivua)

Tutkielman ohjaajat: Professori Kaija Saranto
Lehtori Sirpa Kuusisto-Niemi

Syyskuu 2010
Avainsanat: avoin lähdekoodi, ohjelmistokehitys, terveydenhuolto, tietojärjestelmät

Tämän tutkielman tarkoituksena oli selvittää, miten avointa lähdekoodia on käytetty terveydenhuollon ohjelmistoprojekteissa. Varsinaisia tutkimuskysymyksiä oli kolme: 1) Miten avoimeen lähdekoodin perustuvan ohjelmistokehitysmallin valintaa on perusteltu terveydenhuollon ohjelmistoprojekteissa? 2) Miten kehittämistyö on organisoitu avoimen lähdekoodin ohjelmistoprojekteissa terveydenhuollossa? 3)Mitkä olivat keskeisimmät välineet ja toimintatavat ohjelmistojen kehittämisessä?

Tutkielman aineistona oli terveydenhuollon tiedonhallinnan tieteellisissä julkaisuissa julkaistuja artikkeleita, jotka hankittiin systemaattisen kirjallisuuskatsauksen periaatteita soveltaen Pubmed ja INSPEC -tietokannoista. Sisäänotto- ja poissulkukriteerit täytti 29 artikkelia. Artikkelien sisältö analysoitiin teoriasidonnaisesti teemoittelemalla aineisto kolmeen pääteemaan, jonka pohjalta syntyivät tutkielman pääluvut.

Keskeisimmät motiivit avoimen lähdekoodin käytölle aineistossa olivat pragmaattisia, kuten ohjelmiston jatkokehittämisen mahdollistaminen ja turvaaminen, yhteensopivuus, valmiin koodin uudelleenkäyttö ja kustannushyödyt. Tutkielman kohteena olleiden ohjelmistoprojektien organisoitumisessa keskeisessä roolissa olivat kehittäjäyhteisöt. Organisoituminen tapahtui pääasiassa internetpohjaisten työkalujen avulla. Keskeistä projektien toteuttamisessa olivat työn organisoiminen modulaarisesti, tukeutumien standardeihin ja valmiiden luottaviksi todettujen ohjelmistokomponenttien uudelleen käyttäminen.

Tutkielman tulosten pohjalta voidaan todeta, että avoin lähdekoodi ei ole itsessään sellainen ratkaisu, jolla kaikki terveydenhuollon tiedonhallinnan ongelmat ratkeaisivat. Avoin lähdekoodi muuttaa käyttäjäorganisaatioiden ja ohjelmistoyritysten suhteita ja siirtää päätösvaltaa ohjelmistokehityksestä käyttäjille. Ohjelmistoyritysten liiketoimintalogiikoiden on pakko mukautua tähän tilanteeseen. Terveydenhuollon sitoutuminen avoimiin ohjelmistoprojekteihin voi tapahtua joko yhteisöllisesti tai sopimuksellisesti. Organisaatioiden oma osallistuminen ja innovointi ohjelmistoprojekteihin voi vaihdella. Tutkielmassa tunnistettiin tämän jaottelun pohjalta neljä ideaalityyppiä terveydenhuollon organisaatioiden suhteesta avoimen lähdekoodin ohjelmistoprojekteihin. Näitä ovat käyttäjä-kehittäjämalli, teknologiamalli, puhdas business-malli ja kumppanuusmalli.

Jatkossa olisi syytä tutkia aihetta lisää selvittämällä terveydenhuollon avoimen lähdekoodin ohjelmistojen lokalisointimahdollisuuksia Suomeen ja ohjelmistoyritysten suhtautumista avoimeen lähdekoodiin.

UNIVERSITY OF EASTERN FINLAND, Faculty of Social Sciences and Business Studies, Department of Health Policy and Management, Health and Human Services Informatics

MIKKO HUOVILA: Open Source Software Projects in Health Informatics

Master's thesis, 84 pages, 1 appendices (3 pages)

Advisors: Professor Kaija Saranto
Lecturer Sirpa Kuusisto-Niemi

September 2010
Keywords: health informatics, informatics systems, open source, software projects,

The purpose of this study was to find out how the open source development model has been used in the health informatics. The research covered three questions: 1) How has the choice of using the open source method been argued in the open source software projects in health informatics? 2) How has the development been organized in the software projects? 3) Which methods and techniques were the most important ones in the development?

The research data for this study were scientific articles published in scientific journals. The articles were collected by applying systematic literature review methods. The electronic databases (PubMed, INSPEC) were used to search for articles. Include and exclude criteria were met in 29 articles. The articles were analyzed by using qualitative methods. Three main themes were identified.. These themes formed categories, which were used to form the main chapters for this study.

The most important motives to use open source were pragmatic, like enabling software development in the future, interoperability, reusing stable code and cost effectiveness.
The developer communities were in a central role in the organization of the software development was . The software projects used mainly Internet-based communication tools. The most important methods and techniques in software development were modularizing, standards and the reuse of the robust software components that already exist.

Open source is not all-in-one solution to all the challenges in health informatics. Using of open source method gives many possibilities and freedoms which are not possible in traditional closed software development. In open source situation the source code is public, so the power to make decisions shifts from the software companies towards the user-organizations. The software industry is forced to change their business logic. Health care organizations can commit to software projects two ways: by community or by contracts. Involvement and innovation of the health care organizations can vary. In this study were identified four ideal types how health care organizations can relate to the open source software projects. These types are user-developer model, technology model, pure business model and partnership model.

There seems to be further requirements for research in open source in health informatics. In the Finnish context, it would be important to know how internationally developed software can be localized and how the local software companies see open source in the field of health informatics.

SISÄLLYS

Kuvat

Taulukot

1 JOHDANTO

Terveydenhuollon tietotekniikan ongelmat ovat saaneet paljon julkisuutta niin meillä Suomessa kuin kansainvälisestikin. Tietojärjestelmien käyttäjät ovat tyytymättömiä järjestelmien käytettävyyteen, järjestelmät eivät keskustele keskenään ja tietojärjestelmien käyttöön kuluu liikaa työaikaa. Samaan aikaan kuitenkin tietojärjestelmien kehittämisen kautta uskotaan saatavan merkittäviä tuottavuushyötyjä sosiaali- ja terveydenhuollon toiminnoissa tulevina vuosina. (Kaplan & Harris. 2009, 291-292; Turkki 2009, 40; Finanssipolitiikan linjat -hanke 2010, 129-132 Heeks 2006, 127;)

Avointen standardien ja avoimen lähdekoodin käyttö on nostettu yhdeksi keskeiseksi tekijäksi, jolla julkisten tietojärjestelmien kehittämiseen saadaan lisää yhteensopivuutta ja riippumattomuutta suurista järjestelmätoimittajista. Avoin lähdekoodi on ohjelmistojen kehittämismallina osoittautunut tehokkaaksi. Tästä osoituksena on monien avoimen lähdekoodin ohjelmistojen, kuten Linux-käyttöjärjestelmän, Apache -webpalvelimen ja Firefox -internetselaimen laaja levinneisyys. (Ghosh 2005, 7-8; Weber 2004, 5)

Avoimen lähdekoodin mallissa keskeisenä lähtökohtana on tehdä ohjelmiston käytöstä, muokkaamisesta ja edelleen kehittämisestä mahdollisimman vapaata. Kuka tahansa voi käyttää, kopioida ja parannella avoimen lähdekoodin ohjelmistoja oman tarpeensa mukaisesti. Ohjelmistoteollisuuden keskeinen liikesalaisuus on pitkään ollut ohjelmiston lähdekoodi. Avoimen lähdekoodin mallissa tämä logiikka kääntyy ylösalaisin, sillä ohjelmiston lähdekoodi on kaikkien saatavilla oleva julkishyödyke. Tyypillistä tälle toimintamallille on valmiiden ohjelmistokomponenttien laaja käyttäminen, ohjelmiston räätälöinti omien tarpeiden mukaiseksi sekä kehitystyön tapahtuminen yhteisöllisesti kehittäjäverkostoissa. (Weber 2004, 4, 154, 192-192; Lessig 2006, 146, 151; von Hippel 2005a, 10, 85-89)

Pitkäaikaisena Linux-harrastajana minua kiinnosti heti opintojen alusta asti, miten avointa lähdekoodia on hyödynnetty sosiaali- ja terveydenhuollon tietohallinnossa. Monet sosiaali- ja terveydenhuollon tietojärjestelmien ongelmat olivat mielestäni sellaisia, että avoimemmalla kehittämistavalla ehkäistäisiin niiden syntyä. Sosiaali- ja terveydenhuollon tiedonhallinnan tarkasteleminen avoimen lähdekoodin mallin näkökulmasta

tuntui luontevalle. Näin mielekäs ja innostava aihe opinnäytetyölle löytyi helposti.

Kun aloin selvittämän, miten avointa lähdekoodia oli käsitelty sosiaali- ja terveydenhuollon tiedonhallinnan tutkimuksessa, tuli nopeasti selväksi, että keskustelu painottui kokonaan terveydenhuoltoon. Vaikka oma ammatillinen taustani onkin sosiaalialalla, tämä työ rajoittuu koskemaan vain terveydenhuoltoa. Sosiaali- ja terveydenhuollon tietojärjestelmien kehittämisessä on kuitenkin paljon samankaltaisia vaikuttavia tekijöitä, joten tämän työn johtopäätöksiä voidaan hyvin soveltaa kehitettäessä sosiaalihuollon ohjelmistoja.

Tutkimusta avoimen lähdekoodin käytöstä terveydenhuollossa on vielä varsin vähän. Tieteellisissä julkaisuissa julkaistuista artikkeleista moni käsittelee aihetta yleisellä tasolla. Näissä artikkeleissa esitellään avoimen lähdekoodin mallia uutena ilmiönä terveydenhuollossa ja perustellaan, miksi avoimen lähdekoodin käyttäminen olisi hyvä ratkaisu myös terveydenhuollon kontekstissa. Jonkin verran on myös raportinomaisia artikkeleita, joissa kuvataan avoimeen lähdekoodiin perustuvien tietojärjestelmien kehittämistä ja käyttöönottoa. Tämän tutkielman aineisto koostuu jälkimmäisistä artikkeleista.

Tutkielman tehtävänä on vastata seuraaviin kysymyksiin:

1. Miten avoimeen lähdekoodin perustuvan ohjelmistokehitysmallin valintaa on perusteltu terveydenhuollon ohjelmistoprojekteissa?
2. Miten kehittämistyö on organisoitu terveydenhuollon avoimen lähdekoodin ohjelmistoprojekteissa?
3. Mitkä olivat keskeisimmät välineet ja toimintatavat ohjelmistojen kehittämisessä?

Tässä tutkielmassa tutkimusaineistoksi valikoituneet tieteelliset artikkelit ymmärretään yhtenä mahdollisena kielellisenä kuvauksena siitä todellisuudesta, jota ne pyrkivät kuvaamaan. Teksti on aina vain yksi näkökulma käsiteltävään aiheeseen. Aina on mahdollista avata ja sulkea uusia näkökulmia ja kirjoittaa uusia versioita kuvattavasta kohteesta. Toisin sanoen tässä tutkielmassa perusajatuksena on, että kieli ei ole sosiaalisen todellisuuden neutraali heijastaja. Kieltä käytetään eri tavoin pyrittäessä

saavuttamaan erilaisia päämääriä. Tästä johtuen myöskään aineiston analyysia ei voida tehdä sellaisesta lähtökohdasta, että tutkimusaineistossa käytetty kieli kertoo vääristelemättä todellisuudesta. Näin ollen tässä tutkielmassa aineistoon suhtaudutaan suhteellisemmin ja aineiston teksti nähdään järjestyneeksi sellaiseksi, mitä kirjoittajat ovat sillä alunperin tavoitelleet. (Eskola & Suoranta 2003, 138-141) Tämän sekä analyysin luotettavuuden arvioinnin vuoksi lainaukset tutkimusaineistosta on raportoitu alkuperäisessä englanninkielisessä muodossa.

Eskola & Suoranta (2003, 243-244) kannustavat kirjoittamaan tutkimusraportin siten, että teoriaa, aikaisempia tutkimustuloksia, omia tuloksia ja pohdintaa ei eroteta omiksi luvuikseen. Tällöin tutkimusraportti kirjoitetaan siten, että alussa on lyhyt johdanto aiheeseen ja itse tutkimukseen. Tässä tutkielmassa johdannon jälkeen oleva luku 2 on taustaa tutkimukselle ja siinä esitellään terveydenhuolllon tiedonhallintaa ja avointa lähdekoodia ilmiöinä. Tutkimusprosessi eli aineiston hankinta ja analyysi kuvataan luvussa 3. Tämän jälkeen raportti etenee teemoittain siten, että niissä kaikissa pääluvuissa käsitellään teoria, aikaisemmat tutkimustulokset, omat tulokset ja pohdinta. Eskola kutsuu tätä raportointi tapaa "tuplasuppilomalliksi". Tavoitteena on saada eri ainekset keskusteleman keskenään sen sijaan, että ne jäisivät toisistaan irrallisiksi (Eskola 2007, 165). Päälukujen jälkeen on vielä pohdintaluku, jossa vedetään yhteen tutkielman päälukujen tulokset.

Pro gradu -seminaarissa minua kehotettiin aloittaa raportin tekeminen tämän suuntaisesti. Luin myös Jari Eskolan artikkelin (2007), jossa seikkaperäisesti käydään läpi tutkielman tekeminen tällä mallilla. Koska tässä tutkielmassa ei ole yhtä selkeää teoriaa tutkimuksen viitekehykseksi vaan ennemminkin hajanainen kokoelma aiempia tutkimustuloksia ja useita pieniä teorioita, niin "tuplasuppilo" malli soveltuu hyvin tähän työhön. Näin tutkielma etenee ilmiöpohjaisesti ja erilaiset teoriat, käsitteet ym. toimivat tutkimusaineiston tulkintakehyksenä. Raportointitavan valinnan etu on ennen kaikkea siinä, että se pakottaa pohtimaan tarkkaan, miten erilaiset asiat liittyvät toisiinsa. Näin empiria ja teoria saadaan nivottua toisiinsa paremmin kuin raportointitavassa, jossa ne irrotetaan toisistaan. Parhaimmillaan tällaisessa raportointitavassa tutkija keskustelee aikaisempien tutkimusten, teorian ja oman aineistonsa kanssa. Tähän myös tässä tutkielmassa on pyritty. (Eskola 2007, 164-165)

2 TERVEYDENHUOLLON TIEDONHALLINTA JA AVOIN LÄHDEKOODI

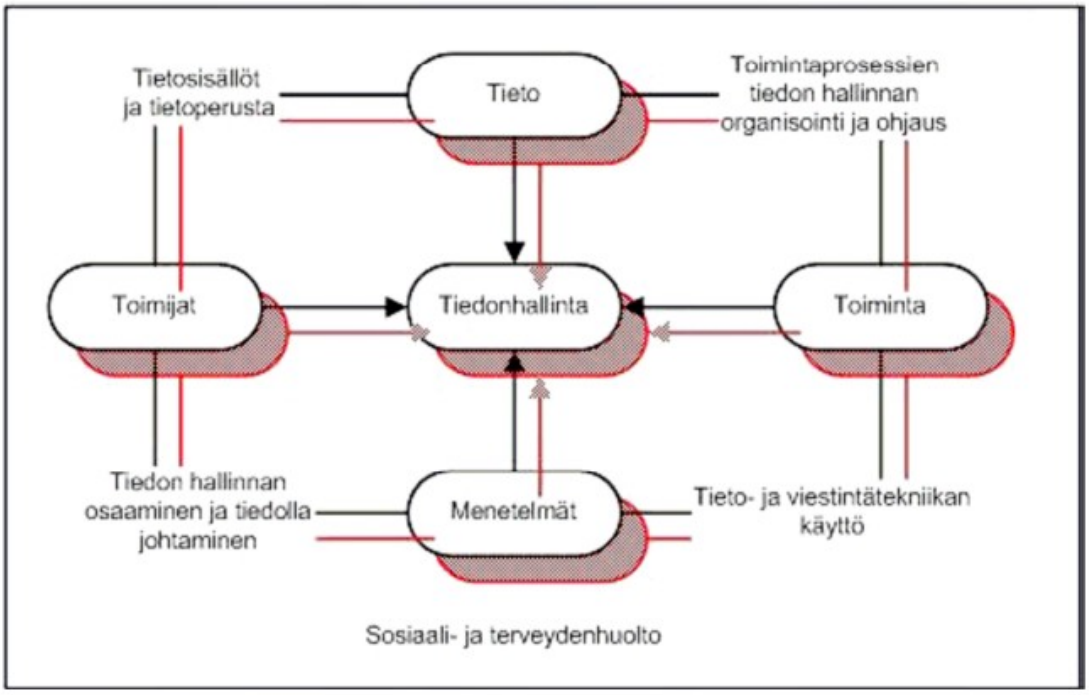
2.1 Terveydenhuollon tiedonhallinta tieteenä ja käytäntönä

2.1.1 Terveydenhuollon tietotekniikka ja tiedonhallinta

Terveydenhuollon tietotekniikkaa ja tiedonhallintaa on määritelty eri aikoina ja erilaisissa yhteyksissä milloin suppeammin, milloin taas laaja-alaisemmin. Eri määritelmissä painotetaan joko terveydenhuollon substanssia eli tietotekniikan hyödyntämisen tavoitteita tai vaihtoehtoisesti painopiste on ollut enemmän teknologiassa. Suomalaisessa kontekstissa Korpela & Saranto (1999, 19) ovat määritelleet terveydenhuollon tietotekniikan *tieto- ja viestintätekniikan soveltamiseksi terveydenhuollossa (tieteenalana ja käytännön toimintana).* Kun halutaan painottaa vain teknisiä ratkaisuja laajempaa näkökulmaa, tietotekniikka -termin sijaan voidaan käyttää termiä tiedonhallinta (Korpela & Saranto 1999, 19; Staggers & Thompson 2002, 255-261)

Kansainvälisessä keskustelussa käytettään yleisimmin käsitettä *Health Informatics.* Tällöin tarkoitetaan koko terveydenhuoltoa koskevaa tietotekniikkaa ja tiedonhallintaa. Varsinkin amerikkalaisissa yhteyksissä käytetään usein myös käsitettä *Medical Informatics.* Tämän taustana on, että tietotekniikkaa sovellettiin ensimmäisenä nimenomaan lääketieteellisiin tarkoituksiin. Terveydenhuolto ja terveystieteet eivät ole kuitenkaan pelkkää lääkärintyötä ja lääketiedettä. Terveydenhuollon tietotekniikka ja tiedonhallinta voidaankin jaotella ammattiryhmien näkökulmasta lääketieteellisen tietotekniikan lisäksi esimerkiksi hoitotyön tietotekniikkaan (nursing informatics) sekä laboratoriotyön ja kuvantamisen tietotekniikkaan. (Korpela & Saranto 1999, 24; Staggers & Thompson 2002, 255-261)

Itä-Suomen yliopiston sosiaali- ja terveydenhuollon tietohallinnon oppiaineessa on luotu tiedonhallinnan tutkimuksen viitekehys. Tässä mallissa toimijoilla tarkoitetaan sosiaali- ja terveydenhuollon palveluja käyttäviä tai tuottavia henkilöitä tai yhteisöjä. Tieto ymmärretään arvoketjuna datasta viisauteen. Toimintaa ovat palvelujen suunnittelu, toteutus, käyttö ja arviointi. Menetelmillä tarkoitetaan niitä teknisiä tai sosiaalisia toimintatapoja, joilla toiminnassa syntynyttä tietoa käsitellään, tallennetaan ja välitetään. (Kuusisto-Niemi & Saranto 2009, 22)



Kuva 1: Tiedonhallinnan tutkimuksen paradigma

Tiedonhallinnan tutkimus ei kohdistu vain näihin neljään entiteettiin, vaan myös niiden välisiin suhteisiin. Tiedon ja toiminnan yhdistämisessä ollaan kiinnostuneita toiminta-prosessien tiedonhallinnasta. Toiminnan ja menetelmien yhdistäminen on ensisijaisesti tieto- ja viestintätekniikan arviointia ja kehittämistä. Menetelmien ja toimijoiden tutkimisessa kohteena ovat tiedon hallinnan osaaminen ja tiedolla johtaminen. Toimijoiden ja tiedon yhdistelmässä taas ollaan kiinnostuneita tietosisältöjen ja tietoperustan kehittämisestä. (Kuusisto-Niemi & Saranto 2009, 22)

Tämän tutkielman kiinnostuksen kohde on ennen kaikkea menetelmissä, joilla terveydenhuollon tietojärjestelmiä kehitetään. Itä-Suomen yliopiston tietohallinnon tutkimuksen viitekehyksessä tutkielman kohde on toiminnan ja menetelmien välisissä suhteissa eli tieto- ja viestintätekniikan kehittämisessä.

2.1.2 Ohjelmistot ja tietojärjestelmät terveydenhuollossa

On arvioitu, että suomalaisessa terveydenhuollossa olisi käytössä yhteensä 4000 erilaista tietojärjestelmää (Finanssipolitiikan linjat -hanke 2010, 130). Näillä järjestelmillä on

hyvin erilaisia tehtäviä, ja terveydenhuollon monimutkaista tietojärjestelmäkokonaisuutta on pyritty usein hahmottamaan erilaisten jaotteluiden avulla. (mm. Turunen 2001; 54-62; Saarelma 1992; Korpela 1994,; Van der Loo 1995; Suomi 2000; Suomi 2001) Korpelan (1994, 102-106) mukaan jaottelun lähestymistapoja voivat olla mm. tiedon sisältö (potilastieto, muu tieto), ammattiryhmät (esim. johtajat, lääkärit, hoitajat, sihteerit, laboratorionhoitajat), tekninen arkkitehtuuri, pitkän tähtäimen tavoitteet ja tutkimusparadigmat (esim. lääketieteellinen päätöksenteko).

Tässä tutkielmassa terveydenhuollon tietojärjestelmät kategorisoidaan Mäkelän (2006, 35) esittämän luokittelun pohjalta. Mäkelän luokittelu on moniin muihin jaotteluihin verrattuna melko karkea, mutta samalla selkeydessään tämän tutkielman tarpeisiin sopiva. Mäkelä jakaa tieto- ja viestintätekniikan soveltamisen terveydenhuollossa neljään eri järjestelmätyyppiin.

Potilasjärjestelmien peruskäyttökohde on potilaan terveyteen, hoitoon ja terveydentilaan liittyvän tiedon tallentaminen. Potilastietojärjestelmä yhdistää potilaaseen liittyvän tiedot muihin terveydenhuollossa käytettäviin tietoihin. Näin syntyy potilaskertomus. Järjestelmät jakautuvat kahteen toiminnalliseen osioon: dokumentteihin ja viesteihin. Dokumentit ovat kokoelma potilastietoa, kuten potilaan käyntiin ja hänen terveydentilaansa liittyvää tietoa. Viestejä välitetään organisaatioiden sisällä tai organisaatioiden välillä. Viesti voi olla toimenpiteitä aiheuttava dokumentti, kuten pyyntö jonkin toimenpiteen suorittamiseksi tai vastaus aiempaan pyyntöön. Viesteinä voidaan myös lähettää tiedotteita tai potilaaseen liittyvän toimenpiteen tuloksia. (Mäkelä 2006, 36-38; Tolppanen 1999, 241-246; Turunen 2001, 61-62)

Hallintojärjestelmiä käytetään terveydenhuollon organisaatioiden hallinnollisen tiedon käsittelyyn. Potilashallintoon liittyy paljon potilaan kutsumiseen, ajanvarauksiin, tilastointiin ja taustatietoihin liittyvää tietoa. Potilashallintoon kuuluu myös maksusitoumukseen ja laskutukseen liittyvää tietoa. Hallintojärjestelmillä hallitaan näitä tietoja. Osaa hallintojärjestelemistä ollaan nykyisin korvaamassa potilastietojärjestelmiin integroituneilla järjestelmillä. Hallintojärjestelmien merkitys terveyspalvelujen johtamisessa on kasvanut merkittävästi viime aikoina. (Mäkelä 2006, 40-41, Tolppanen 1999; 248-249; Lauharanta & Kekomäki 1999; 296-311; Saranto 2005, 305-308; Turunen 2001, 61-62)

Kuvantamisjärjestelmiä ja kuva-arkistoja käytetään digitaalisilla kuvauslaitteilla otettujen lääketieteellisten kuvien tallennukseen ja käsittelyyn. Digitaalisen kuvantamisjärjestelmän perustoiminnot ovat digitaalinen kuvaus, kuvan esikatseluja ja käsittely, kuvan arkistointi ja katselu erillisellä PACS-kuva-arkistojärjestelmällä, sekä kuvien sekä kuviin liittyvien potilastietojen hallinnointi RIS-tietokantaohjelmistolla tai osana potilastietojärjestelmää. (Mäkelä 2006, 41-45; Kiuru 1999, 278- 294)

Erillisjärjestelmiä käytetään potilaiden etäseurantaan, diagnostiikkaan, valvontaan ja hoivaan, mutta ne eivät sisälly edellä mainittuihin kolmeen järjestelmätyyppiin. Sairaaloissa onkin käytössä kymmeniä tai jopa satoja erilaisia järjestelmiä. Sairaaloiden lisäksi erillisjärjestelmiä käytetään myös muualla terveydenhuollossa, kuten kotihoidossa, ambulansseissa tai neuvoloissa. Erillisjärjestelmiä käytetään mm. seuraavissa tehtävissä:

- Laboratoriojärjestelmät
- Mittaus ja monitorointi
- Etäseuranta ja monitorointi
- Erikoisalajärjestelmät
- Kotihoitojärjestelmät
- Konsultaatiojärjestelmät
- Ammattilaisten hoito-ohjeistukset
- Asiakasjärjestelmät

  (Mäkelä 2006, 45-49; Mäkinen & Soini 1999, 254-275; Saranto & Kouri 1999, 334-355; Turunen 2001, 61-62)

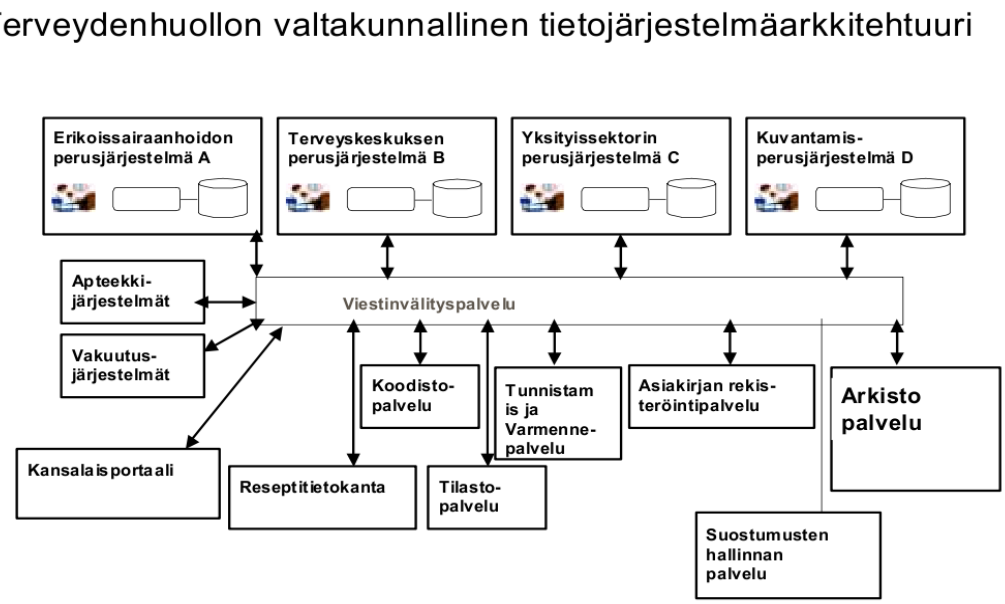
2.1.3 Terveydenhuollon kansallinen arkkitehtuuri

Valtioneuvosto teki vuonna 2002 periaatepäätöksen, jonka pohjalta käynnistettiin kansallinen terveyshanke. Sen yhtenä tavoitteena oli rakentaa valtakunnallisesti yhteentoimiva sähköinen potilasasiakirjajärjestelmä. Valmistelutyön aikana valtion tietoyhteiskuntaohjelman myötä nousi esiin ajatus valtakunnallisesta terveydenhuollon sähköisestä arkistosta. Tähän liittyen käytiin keskustelua, jossa korostettiin järjestelmän valtakunnallista yhteentoimivutta, potilaskertomusten sähköistä arkistointia, tiedon valtakunnal-

lista saatavuutta, tietosuoja- ja tietoturvakysymyksiä, tietojärjestelmien skaalaetujen hyödyntämistä ja ratkaisujen taloudellisuutta sekä valtakunnallisesti keskitettyjä palveluja (Iivari & Ruotsalainen 2006, 13-15)

Keskustelun myötä painopiste siirtyi alueellisista tietojärjestelmäratkaisuista valtakunnallisiin. Tähän vaikutti voimakkaasti tarve liikutella potilastietoja yli organisaatiorajojen sekä yleiset teknologiset kehitystrendit, jotka mahdollistavat valtakunnan laajuisten tietojärjestelmien toteuttamisen. (Iivari & Ruotsalainen 2006, 15) Näiden muutosten myötä syntyi terveydenhuollon valtakunnallinen tietojärjestelmäarkkitehtuuri, joka sisältää tiedon keräämiseen periaatteet, tiedon tallentamisen ja säilyttämisen (arkistoinnin) periaatteet, tiedon välityksen ja jakamisen periaatteet sekä tietosuojan ja tietoturvan arkkitehtuurin. Valtakunnallisen tietojärjestelmäarkkitehtuurin tavoitetilan ratkaisulla mahdollistetaan digitaalisten potilastietojen pitkäaikaisarkistointi ja potilastietojen valtakunnallinen tietoturvallinen saatavuus sekä terveydenhuollon toimijoille, potilaalle että muille toimijoille lakiin perustuva tiedonsaantioikeus.

Terveydenhuollon valtakunnallisen tietojärjestelmäarkkitehtuurin toteuttamiseksi eduskunta hyväksyi vuonna 2007 lain sosiaali- ja terveydenhuollon asiakastietojen sähköisestä käsittelystä (159/2007). Laki tekee Kansaneläkelaitoksesta viranomaisen, jonka vastuulle valtakunnallisen arkistopalvelun ja yhteisten tietojärjestelmäpalveluiden järjestäminen tulee. Lain mukaan terveydenhuollon organisaatioilla on velvollisuus liittyä yhteisiin palveluihin kevääseen 2011 mennessä. Tätä tutkielmaa tehtäessä hallitus on kuitenkin antanut esityksen (HE 176/2010), jolla liittymisvelvoite siirtyy vuoteen 2014.



Kuva 2: Terveydenhuollon valtakunnallinen tietojärjestelmäarkkitehtuuri (Iivari & Ruotsalainen 2006)

2.1.4 Tietotekniikan käyttöönoton haasteet terveydenhuollossa

Tietotekniikan laajalle käyttöönotolle terveydenhuollossa on asetettu suuria odotuksia ympäri maailmaa. Tietotekniikan odotetaan nostavan terveydenhuollon laatua ja tuovan säästöjä. (Kaplan & Harris 2009, 291)

Viime aikaisessa keskustelussa Suomessakin terveydenhuollon tietotekniikalle on asetettu merkittäviä odotuksia erityisesti tuottavuuden parantamiseksi. Hyvin toimiva sähköinen tiedonhallinta nähdään välttämättömyytenä terveydenhuollon kustannustenhallinnassa ja potilasprosessien tehostamisessa (Turkki 2009, 40). Julkisten palveluiden tuottaminen on erittäin työvoimaintensiivistä, joten tietotekniikan hyödyntämiseltä odotetaan hyvinkin merkittäviä tuottavuusvaikutuksia. Erityisesti terveydenhuollossa ongelmana ovat hyvin hajanaiset ja yhteensopimattomat tietojärjestelmät. Näiden käyttäminen vie suhteettoman paljon henkilökunnan työaikaa. (Finanssipolitiikan linjat -hanke 2010, 129-132)

Tähän asti tietotekniikan tuominen terveydenhuoltoon ei ole kuitenkaan ollut suuri menestys. Valitettavan usein ohjelmistojen käyttöönotot eivät ole onnistuneet odotetusti.

On esitetty erilaisia arvioita IT-projektien epäonnistumisista. Joidenkin arvioiden mukaan vähintään 40 % IT-projekteista jää kesken tai ne eivät täytä niille asetettuja vaatimuksia ja vähemmän kuin 40 % toimittajilta tilatuista suurista järjestelmistä saavuttaa tavoitteet (Booth 2000; ITCortex). Joissain arvioissa on esitetty, että jopa 70 % tietojärjestelmähankkeista epäonnistuu muodossa tai toisessa (Kaplan & Harris 2009. 291).

IT-projektien epäonnistumiset ovat yleisiä myös terveydenhuollossa. On jopa esitetty sellaisia arvioita, että useimmat terveydenhuollon tietojärjestelmät epäonnistuvat jollain tavalla (Heeks 2006, 127; Kaplan & Haris 2009, 292).

Terveydenhuollon tietojärjestelmähankkeiden ongelmat ovat yleisiä myös Suomessa. Vuodesta 2003 sosiaali- ja terveydenhuollon tietojärjestelmiin on käytetty noin 100 miljoonaa euroa valtion varoja. Valtiontalouden tarkastusvirasto on kuitenkin kiinnittänyt huomiota siihen, että ei ole täysin selvää, mitä kaikkea tuolla summalla on saatu aikaan. Kenelläkään ei ole selkeää kokonaisnäkemystä aikaansaannoksista. Haasteita on mm. eri osapuolien välisessä yhteistyössä, tilaajien asiantuntemattomuudessa ja kilpailutuksen puutteessa. Erittäin suuri ongelma on järjestelmien välinen yhteensopimattomuus, joka johtuu järjestelmien yhteensopimattomista rajapinnoista. (Puustinen 2008)

Erityisesti potilastietojärjestelmien käytettävyyttä on kritisoitu paljon. Järjestelmien ongelmina on pidetty mm. sitä, että tiedot ovat ohjelmistoissa hajanaisesti, potilaan perustietoja ei saada näkyville yhdellä silmäyksellä, ohjelmat eivät anna muistutuksia, lääkitysten kirjaus on monimutkaista ja järjestelmät ovat hitaita. Terveydenhuollon henkilöstö onkin kokenut, että heidän mielipiteitään ei juurikaan ole kuunneltu ohjelmistoja kehitettäessä. (Pihlava 2009, 16-17; Lahti 2008, 5-6; Lääveri 2008a, 3; Lääveri 2008b, 33)

Terveydenhuollon tietojärjestelmäprojektien epäonnistumisista ollaan oltu kiinnostuneita pitkään. Arvioita käyttöönotoista ja suosituksia hyviksi käytännöiksi on luotu jo 1970-luvulta alkaen (Kaplan & Harris 2009, 292). Heeks (2006, 127) kuitenkin huomauttaa, että aiemmassa tutkimuksessa on kahdenlaisia ongelmia.

Ensimmäinen ongelma liittyy yleistettävyyteen. Monet tutkimuksista ovat erityisen spesifejä. Yksittäiseen epäonnistumiseen keskittyminen tekee tulosten yleistämisen vai-

keaksi. Osa tutkimuksista on hyvin yleisellä tasolla ja pyrkii olemaan sovellettavissa kaikkiin mahdollisiin tilanteisiin. Molemmissa tapauksissa tutkimukset epäonnistuvat tunnistamaan tilannekohtaiset tekijät, jotka määrittävät onnistumisen tai epäonnistumisen. (mt., 127)

Toinen ongelma liittyy käsitteellistämiseen. Osa tutkimuksista tarjoaa käyttökelpoisia käytännöllisiä ratkaisuja, mutta niissä ei ole selkeää käsitteellistä mallia perustanaan. Toisaalta toiset taas ovat hyvin käsitteellisiä, mutta ne tarjoavat rajoitetusti käytännön ratkaisuja. (mt., 127)

Millainen tietojärjestelmäprojekti on sitten onnistunut? Onnistumista voidaan tarkastella useista eri näkökulmista. Näitä voivat olla esimerkiksi tehokkuus, vaikuttavuus, organisaation asenteet ja sitoutuminen, työntekijöiden tyytyväisyys tai potilaiden tyytyväisyys. Eri osapuolilla ei välttämättä kuitenkaan ole yhteistä näkemystä siitä, mikä näistä olisi tärkeintä. Onnistumisen määritteleminen ei olekaan yksiselitteistä ja staattista. Onnistuminen on dynaaminen käsite, jonka määritteleminen riippuu tilanteesta, ajasta ja määritellyistä kriteereistä. Eri projekteissa erilaisissa tilanteissa ja eri aikoina voi olla toisistaan poikkeavia odotuksia sille, mitä pidetään onnistumisena. Yhden osapuolen kokema onnistuminen voi olla epäonnistuminen toiselle. Tämän vuoksi usein on hyödyllistä luoda onnistumisen arvioinnille mahdollisimman objektiivisia mittareita. (Heeks 2006, 126; Berg 2001, 144-145)

Heeksin (2006, 126) mukaan edellä esitettyjä määrittelyn ongelmia voidaan osittain ratkoa epäonnistumisen kolmivaiheisella jaottelulla:

- *Kokonainen epäonnistuminen:* tietojärjestelmää ei koskaan oteta käyttöön tai se otetaan käyttöön, mutta hylätään välittömästi.
- *Osittainen epäonnistuminen:* hankkeen keskeisiä tavoitteita ei saavuteta tai syntyy epätoivottavia seurauksia.
- *Onnistuminen:* hankkeen useimmat osapuolet saavuttavat keskeisimmät tavoitteensa eivätkä koe merkittäviä epätoivottavia seurauksia.

Kaplan & Harris:in (2009, 292) mukaan useissa tutkimuksissa tarkemmasta määritel-

mästä riippumatta epäonnistumiseen liittyy se, että IT-projektia ei toimitettu niin kuin olisi pitänyt, projekti ylitti talousarvionsa tai se oli myöhässä.

Terveydenhuollon tietojärjestelmien arviointia varten Heeks on kehittänyt mallin, jolla hankkeiden onnistumisia voidaan jäsentää ja arvioida. Hänen mukaansa keskeistä terveydenhuollon tietojärjestelmien onnistumisissa ja epäonnistumisissa on jännite, joka syntyy tämänhetkisten realiteettien ja tietojärjestelmän asettamien odotusten välillä. Se, "missä olemme nyt", ja se, "minne tietojärjestelmä haluaa meidät viedä", voivat erota hyvinkin paljon toisistaan. Onnistuminen ja epäonnistuminen riippuvat tämän kuilun suuruudesta. (Heeks 2006, 128)

Mallin pelkistäminen kahteen osaan, suunnitelmiin ja todellisuuteen, edellyttää yksinkertaistamista. Mallissa lähtökohtana on, että suunnittelusta vastaavat *suunnittelijat* ja paikallista todellisuutta edustavat *käyttäjät*. Heeksin mukaan nämä ryhmät ovat keskeisessä asemassa, kun pyritään ymmärtämään terveydenhuollon tietojärjestelmien ongelmia. Niin suunnittelijoilla kuin käyttäjilläkin on omat kulttuuriset arvonsa, tavoitteensa ja olettamuksensa. (mt., 128)

Suunnittelijoiden ja käyttäjien välisen kuilun ymmärtämiseksi Heeks on aiemman tutkimuksen pohjalta erotellut seitsemän ulottuvuutta:

- Informaatio (tietovarastot, tietovirrat)
- Teknologia (laitteisto ja ohjelmistot)
- Prosessit (käyttäjien ja muiden toimijoiden aktiviteetit)
- Tavoitteet ja arvot (avain ulottuvuus, kulttuuri, politiikat)
- Työvoima ja taidot (osaamisen laadulliset ja määrälliset osa-alueet)
- Hallintojärjestelmät ja rakenteet
- Muut resurssit (erityisesti aika ja rahat)

Myös Lehenkarin (2003, 17) mukaan ohjelmistojen tuottamisen ja käyttöönoton ongelmat liittyvät usein järjestelmien tuottajien ja käyttäjien välisiin ongelmiin. Terveydenhuollon organisoituminen on monimutkaista ja ulkopuoliselle vaikeasti ymmärrettävää.

Esimerkiksi teknologian käyttäjä, hankintapäätöksen tekijä ja maksaja voivat sijaita eri organisaatioissa. Oppiminen ja vuoropuhelu on osoittautunut vaikeaksi sekä terveydenhuollon toimijoiden kesken, että yritysten kanssa tehtävässä yhteistyössä. Ongelmana on myös ollut tuotekehittäjien teknologiakeskeisyys ja puutteellinen tietämys kliinisen käyttötoiminnan vaatimuksista. Lehenkari näkee ongelmana myös yritysten ja julkisten terveydenhoito-organisaatioiden toiminnan vaikuttimien, aikajänteiden ja sitoutumisen erilaisuuden.

2.2 Avoimen lähdekoodin periaatteet ja niiden muotoutuminen

2.2.1 Lähdekoodi ohjelmiston "reseptinä"

Tietokoneen toiminta perustuu prosessorille syötettyihin yksinkertaisiin konekielisiin käskyihin. Konekielen käskyillä voidaan kuvata vain hyvin pieniä algoritmin osia, joten konekieli on ihmisen näkökulmasta hyvin kömpelöä. Tämän vuoksi tietokoneen ohjelmoinnin helpottamiseksi on kehitetty ns. korkean tason kieliä eli lausekieliä. Ohjelmointi tapahtuukin nykyisin lähes aina ihmiselle paremmin ymmärrettävillä lausekielillä. Tällaisia lausekieliä ovat mm. Basic, Java ja C++. (Boberg 2005, 10-11; Wikla 2003, 3)

Tietokoneen prosessori ei ymmärrä lausekieliä suoraan, joten korkean tason kielellä kirjoitettu ohjelma tulee ennen sen suorittamista muuntaa tietokoneen ymmärtämään konekieliseen muotoon. Lausekielisen ohjelman muokkaamista lausekieleltä konekielelle kutsutaan kääntämiseksi. Tässä käytetään apuna joko kääntäjää tai tulkkia. Kääntäjä muuttaa lausekielellä tehdyn ohjelman kerralla konekieliseen muotoon, kun taas tulkin avulla lausekielistä ohjelmaa käännetään lause kerrallaan suorituksen aikana. (Boberg 2005, 10-11; Wikla 2003, 3-4)

Lausekielellä kirjoitettua ohjelmaa kutsutaan myös lähdekoodiksi. Lähdekoodi on joukko ohjelmoijan tietokoneelle kirjoittamia ohjeita, jotka ovat ihmiselle ymmärrettävässä lausekielisessä muodossa. Ohjelmiston kehittäminen, korjaaminen ja muokkaaminen edellyttävät pääsyä lähdekoodiin. (Weber 2004, 4).

Weber vertaa lähdekoodia reseptiin ja käyttää analogiana virvoitusjuoma Coca-Colaa. Coca-Cola Company:n tärkein liikesalaisuus on juoman salainen resepti. Tätä yhtiö ei paljasta kuluttajille ja kilpailijoilleen missään tilanteessa. Kuka tahansa voi ostaa pullon Coca-Colaa ja juoda sen, mutta hän ei kykene ymmärtämään tarkalleen, miten juoma on valmistettu ja missä suhteessa vettä, sokeria ja muita aineksia on sekoitettu keskenään. (mt., 3-5)

Ohjelmistoteollisuuden ansaintalogiikka on perustunut pitkälti vastaavaan taloudelliseen logiikkaan kuin Coca-Cola:n oikeudet omistavalla yrityksellä. Yritykset suojaavat immateriaalista omaisuuttaan mm. tekijäinoikeuksin, patentein ja erilaisin lisensointikäytännöin. Ohjelmistoteollisuudessa tämä on tarkoittanut, että ohjelmistojen reseptiä eli lähdekoodia ei toimiteta ohjelmien mukana. Tämä on ohjelmistoyrityksille tehokas tapa kontrolloida sitä, miten ohjelmaa voidaan käyttää. Näin ohjelman muokkaaminen, parantaminen ja uuden version levittäminen on tehty mahdottomaksi. Ohjelmia, joiden lähdekoodi ei ole vapaasti saatavilla, kutsutaan suljetuiksi tai omisteisiksi (proprietary). (mt., 4; Lessig 2006, 146)

Avoimen lähdekoodin ohjelmistot kääntävät tämän logiikan kokonaan ylösalaisin. Lähtökohtana on, että ohjelmistojen lähdekoodi on vapaasti kenen tahansa käytettävissä eli se on avointa, julkista ja ei-omisteista. Open Source Initiativen puolivirallisessa avoimen lähdekoodin määritelmässä on kolme perustavaa lähtökohtaa tälle uudelle mallille:

- Lähdekoodi tulee jakaa ohjelmiston mukana tai antaa saataville joko ilmaiseksi tai korkeintaan luovuttamiskulujen hinnalla
- Kuka tahansa saa jakaa ohjelmaa eteenpäin vapaasti ilman rojalteja tai lisenssimaksuja tekijälle
- Kuka tahansa voi muokata ohjelmaa tai johtaa siitä uusia ohjelmia ja sen jälkeen jakaa muokattua ohjelmia samoin ehdoin.
  (Open Source Initiative, Weber 2004, 5)

Omisteisen ohjelmistoteollisuuden näkökulmasta tällaisin ehdoin toteutettuja avoimen lähdekoodin ohjelmistoja ei pitäisi olla mahdollista olla lainkaan olemassa tai niiden tulisi olla hyvin marginaalisia. Todellisuudessa kuitenkin avoimen lähdekoodin ohjelmis-

toista on tullut merkittävä osa ohjelmistoteollisuutta ja ne ovat joillain aloilla saavuttaneet suuriakin markkinaosuuksia 1990-luvun ja 2000-luvun aikana. (Weber 2004, 5)

2.2.2 Avoimen lähdekoodin liikkeen kehitysvaiheet ja periaatteiden muotoutuminen

Ohjelmistojen avoin kehittäminen liittyy läheisesti Unix-käyttöjärjestelmän syntyyn. Teleyhtiö AT&T:n tutkijat kehittivät Unixin vuonna 1969. Unixista ei kuitenkaan voitu tehdä kaupallista tuotetta, sillä Yhdysvaltain hallitus oli jo aiemmin kieltänyt suuryritystä laajentamasta toimintaansa puhelinmarkkinoiden ulkopuolelle. Tästä johtuen yhtiö päätti lisensoida käyttöjärjestelmän yliopistoille muutaman sadan euron nimellistä maksua vastaan. Lisenssi koski myös Unixin lähdekoodia, mikä mahdollisti koodin käyttämisen opetustarkoituksessa. Yliopiston opiskelijat ja tutkijat muokkasivat ja paranteli vat lähdekoodia lisäten Unixiin monenlaisia uusia toimintoja. (Miettinen 2006b 67-68; Weber 2004, 25-29; Moody 2001)

Unixin leviämisen myötä syntyi kokonaan uudenlainen käsitys ohjelmoinnista. Käyttöjärjestelmän kehittäminen tapahtui vähitellen osissa siten, että monet ihmiset ohjelmoivat sitä samanaikaisesti. Vuonna 1975 Unixia käytettiinkin jo yli neljässäkymmenessä yhdysvaltalaisessa yliopistossa ja tutkimuslaitoksessa. (Miettinen ym. 2006b, 68; Weber 2004, 29; Moody 2001)

Berkeleyn yliopistolla syntynyt tutkija-kehittäjien verkosto muodostui suurimmaksi ja tehokkaimmaksi Unixin jakelukanavaksi. Verkoston jäsenet halusivat jakaa Unixista oman versionsa ilmaiseksi kaikille sitä haluaville. Alunperin AT&T:n ja Berkeleyn tutkijoiden välillä oli epämuodollinen sopimus, joka hyödytti molempia osapuolia. AT&T sai käyttöönsä tutkijoiden parantamaa koodia ja nämä taas saivat käyttöjärjestelmän käyttöönsä. AT&T alkoi kuitenkin vaatia tutkijoilta salassapitosopimuksia, mikä oli tutkija-kehittäjille epäedullista. Tästä johtuen AT&T ja tutkijoiden verkoston välit kriisiytyivät. Berkeleyssä perustettiin yritys, joka alkoi myydä Berkeleyn tutkijoiden kehittämää versiota Unixista. AT&T oli laittanut teknologian kehittämiseen miljoonia dollareita, ja tämä nyt tämä oli vaarassa levitä ulkopuolisten hyödyntäjien käsiin. AT&T nostikin syytteen Berkeleyn jakeluverkostoa vastaan, mikä johti vuosia kestäneeseen oikeustais-

teluun. Oikeuden päätös saatiin kuitenkin vasta 1994. Tähän mennessä Berkeleyn tutkija-kehittäjät olivat kirjoittaneet AT&T:n alunperin kehittämän Unix-koodin kokonaan uudelleen. (Miettinen ym. 2006b, 68-69; Weber 2004, 29-35; Moody 2001).

Berkeleyn vaiheen ristiriita avoimen ja kaupallisen ohjelmistokehityksen välillä raivasi tietä avoimelle kehittämiselle Unix-koodin uudelleen kirjoittamisen kautta. Vapaa lähdekoodi ja avoin kehittämismalli syntyivät käsitteellisesti vuonna 1984. Tuolloin MIT:ssa työskennellyt Richard M. Stallman esitteli termin "Free Software". Tällä käsitteellä Stallman tarkoitti sitä, että ohjelmakoodin on säilyttävä vapaasti jaettavana eikä sana "free" siis tarkoittanut ohjelman maksuttomuutta. (Weber 2004, 48; Moody 2001)

Stallmanille vapaus tarkoittaa neljänlaista vapautta:

1. vapautta käyttää ohjelmaa mihin tahansa tarkoitukseen
2. vapautta tutkia ohjelman toimintaa ja muokata sitä omiin tarpeisiin,
3. vapautta levittää kopiota ohjelmasta eteenpäin ja
4. vapautta parantaa ohjelmaa ja jakaa parannukset yhteisöllisesti kaikkien hyödyksi.

   (Stallman 2002, 43; Weber 2004; 5, 48)

Näiden neljän vapauden toteutuminen käytännössä edellyttää mahdollisuutta päästä käsiksi ohjelmiston lähdekoodiin. Stallman kuitenkin oivalsi, että koodin säilyminen vapaana edellyttää rajoituksia. Mikäli koodi jaetaan täysin vapaasti ns. public domainina, voi kuka tahansa ottaa koodin ja käyttää sitä osana omisteista tuotetta. Näin seuraavalla kehittäjäsukupolvella ei ole mahdollisuuksia sellaiseen vapauteen, joita Stallman määritelmässään halusi. (Weber 2004, 48, Stallman 2002; Moody 2001)

Näitä vapauksia suojatakseen Stallman kirjoitti General Public License -käyttöoikeussopimuksen (GPL) ja perusti The Free Software Foundation -nimisen etujärjestön. Näiden tarkoituksena oli vaalia ohjelmistojen vapautta ja sitä, että alkuperäisestä koodista johdetut ohjelmat pysyvät edelleen vapaina. Tämän varmistamiseksi GPL-lisenssi estää muita käyttäjiä lisäämästä rajoituksia, jotka estäisivät vapauksien toteutumisen jatkossa (Weber 2004, 48; Moody 2001)

GPL-lisensoiduista ohjelmista ei voi tehdä ei-vapaita omisteisia eikä niissä käytettyä koodi saa käyttää osana omisteisia ohjelmistoja. Mikäli koodia yhdistetään osaksi ei-vapaata ohjelmaa tulee koko yhdistelmä julkaista vapaana ohjelmistona GPL-lisensoituna. Tämän vuoksi GPL-lisenssiä pidetään "tarttuvana" (viral). (Välimäki 2009, 205-206, 210). GPL-lisenssi oli merkittävä innovaatio avoimen lähdekoodin kehityksessä, ja se loi selkeät periaatteet ja normit, jotka määrittävät vapaita ohjelmistoja.(Weber 2004, 48-49; Moody 2001)

1990-luvun alussa suomalainen tietojenkäsittelytieteen opiskelija Linus Torvalds kyllästyi siihen, että PC-tietokoneille saatavat käyttöjärjestelmät olivat epävakaita ja Unix-tyyppiset järjestelmät olivat kohtuuttoman kalliita opiskelijalle. Ainoa mahdollinen PC-tietokoneessa toimiva Unix-tyyppinen järjestelmä oli Minix. Sen lähdekoodi oli saatavilla, mutta muutokset piti hyväksyttää järjestelmän kehittäneellä yhdysvaltalaisella professorilla. Minixin muokkaamisen hankaluus sekä se, ettei Minixillä pystynyt ottamaan yhteyttä Helsingin yliopiston Unix-koneisiin, sai Torvaldsin aloittamaan uuden toimivan ja luotettavan käyttöjärjestelmän kehittämisen (Miettinen ym. 2006b, 69-70, Moody 2001, 31-54)

Minixiin keskittyvällä postituslistalla Torvalds kertoi aikomuksestaan kehittää uusi käyttöjärjestelmä. Hänen viestinsä herätti kiinnostusta ympäri maailmaa, ja monet olivat halukkaita testaamaan Torvaldsin kehittämää koodia. Pari viikkoa myöhemmin ensimmäinen versio Linux-käyttöjärjestelmästä oli ladattavissa Helsingin yliopiston palvelimelta. (Miettinen ym. 2006b, 70; Moody 2001)

Torvalds päätti julkaista Linuxin koodin GPL-lisenssin alaisuudessa. Hän koki GPL-lisenssin ehtojen vastaavan hyvin aloittamansa projektin tarpeisiin. Ohjelmistojen vapaus Stallmanin tarkoittamassa mielessä oli hänelle toisarvoista. (Miettinen ym. 2006b, 70; Weber 2004, 54-55; Torvalds & Diamond 2001; 114-115)

Linuxin kehittämistyö jatkui perustuen sähköpostilistaan ja versionhallintajärjestelmään. Hanke oli lähtenyt Torvaldsin omista tarpeista, mutta se kasvoi nopeasti maailmanlaajuiseksi vapaaehtoisvoimin toimivaksi projektiksi. Yliopisto-opiskelijat, professorit ja itseoppineet ohjelmoijat muodostivat nopeasti käyttäjäkehittäjien yhteistyöverkoston. Ke-

hittäjien keskuudessa syntyi kilpailua siitä, kuka luo parhainta koodia ja pääsee näin osaksi uutta nopeasti kehittyvää käyttöjärjestelmää. Alussa Torvalds päätti itse, mitkä muutokset ja lisäykset otettiin mukaan Linuxin seuraavaan versioon. Yhteisö organisoitui kuitenkin pian kahdelle kehälle. Torvalds oli keskuksessa johtohahmona, ja hänen ympärillään koodiehdotuksia suodatti joukko luotto-ohjelmoijia. Ulkokehällä oli suuri joukko muutos- ja parannusehdotuksia tekeviä ohjelmoijia ja vioista raportoivia käyttäjiä. Torvalds ei tavannut luotto-ohjelmoijiaan moneen vuoteen. Koodin laatu näyttää olleen ratkaisevaa luottamuksen synnyssä. Linux-kehittyi nopeasti, ja Torvalds julkaisi ensimmäisen virallisen 1.0 version vuonna 1994. Kehitystahti kiihtyi edelleen 1990-luvun aikana. (Miettinen ym. 2006b, 70 ; Moody 2001, Torvalds & Diamond 2001)

Modulaarisuus eli järjestelmän muodostuminen toisistaan irrallaan ohjelmoitavista, mutta yhteen sovitettavista osista oli ratkaisevaa Linuxin kehittämisessä. Modulaarinen arkkitehtuuri ja internet mahdollistavat sijainnista ja ajasta riippumattoman, useiden ihmisten samanaikaisen työskentelyn. Modulaarisuus myös loi hankkeelle hierarkkisuutta ja työnjakoa. Jokaiselle moduulille nimettiin vastuuhenkilö, joka vastasi kyseistä moduulia koskevasta päätöksenteosta ja kommunikoinnista kehittäjäryhmän ulkopuolisten kanssa. (Miettinen ym. 2006b, 70)

Linuxin kehittäjäyhteisön organisoitumista pidetään usein avoimen kehittämismallin malliesimerkkinä. Siinä on keskeistä internetin välityksellä tapahtuva maantieteellisesti hajautunut toiminta, jossa kehittämisen kannalta olennainen lähdekoodi on kaikkien ohjelmointia osaavien saatavilla, muokattavissa ja uudelleen jaettavissa. Tällä periaatteella toimivien projektien lähtökohtana ja edellytyksenä on pidetty kehittäjien vapaaehtoista osallistumista ja tehtävien vapaaehtoista valintaa. (Miettinen ym. 2006b, 71; Weber 2004, 62)

Linuxin ohella yksi merkittävä virstanpylväs avoimessa kehittämisessä oli internetselain Netscapen lähdekoodin avaaminen. Netscape yhtiö teki tämän strategisen valintansa liiketoiminnallisin perustein sen sijaan, että Free Software Foundationin vapauden periaatteet olisivat olleet ensisijaisia. Tämän tapahtuman johdosta kehittäjäyhteisöissä käytiin kiivasta keskustelua siitä, millä käsitteellä avointa kehittämistapaa pitäisi kutsua. "Free software" -käsitettä pidettiin liian rajoittuneena ja harhaan johtavana. Tilalle lanseerat-

tiinkin käsite "Open source", jonka katsottiin kuvaavan paremmin sellaista avointa kehittämistapaa, joka ottaa huomioon myös liiketoiminnan lähtökohdat. Usein avoimesta kehittämisestä käytetään myös lyhennettä FOSS, joka tulee sanoista Free and Open Source Software. (Moody 2001, 267-283; Raymond 1998)

Avoimen lähdekoodin ohjelmistojen menestys on saanut 2000-luvulla ohjelmistoyritykset huomaamaan niiden kaupallisen potentiaalin. Omisteisten ja avoimien ohjelmien rinnalle syntyi hybridihankkeita, joissa yritykset pyrkivät hyötymään avoimen lähdekoodin eduista, kuten kehittäjäyhteisön tehokkaasta työskentelytavasta ja ilmaisesta panoksesta. Toisaalta yritykset myös osallistuvat avoimen lähdekoodin ohjelmistojen kehitykseen oman henkilökuntansa työpanoksella. Yhä useampaan avoimen lähdekoodin projektiin osallistuu palkallisia ohjelmoijia sekä markkinoinnin ja myynnin osaajia. Avoimen lähdekoodin mukaisesta ohjelmistokehityksestä onkin tullut osa ohjelmistoteollisuutta. (Miettinen ym. 2006b, 71-72; Weber 2004; 197-204)

Fitzgeraldin (2006, 587-588) mukaan käsitys siitä, että avoimen lähdekoodin ohjelmistokehitys perustuisi siihen, että erityisen lahjakkaat ohjelmoijat vapaaehtoisesti tarjoutuisivat tekemään huippulaadukasta kehitystyötä, on enemmän myyttistä kuvausta kuin todellisuutta. Avoimen lähdekoodin kehitys on muuntunut FOSS juuriltaan valtavirtaan paremmin sopivaan ja kaupallisesti elinkelpoisempaan suuntaan. Fitzgerald kutsuu tätä uutta tilannetta nimellä OSS 2.0. Muutos ilmenee viidellä osa-alueella.

Ensinnäkin ohjelmistojen kehitysprosessit muuttuvat entistä paremmin ohjatuiksi. Laajasti vapaaehtoisuuteen perustuvassa kehitystyössä strategista suunnittelua ei tehnyt juuri kukaan. FOSS-ohjelmistot ovat olleet pitkälti horisontaalisia, useilla aloilla käytettäviä, loppukäyttäjille näkymättömiä tuotteita, kuten esimerkiksi käyttöjärjestelmiä, palvelimia, tietokantaohjelmistoja ja kääntäjiä. Käsitys siitä, miten tällaisten ohjelmistojen tulisi toimia on, perustunut käyttäjä-kehittäjien kokemuksen kautta saatuun viisauteen. OSS 2.0 -tilanteessa sen sijaan yritykset punnitsevat tarkkaan, miten saavat parhaiten kilpailuetua hyödyntämällä avointa lähdekoodia. Näin avoimeen lähdekoodin perustuva toimintatapa tulee osaksi ohjelmistoyritysten arkkitehtuureja ja ohjelmistokehitystä. (mt., 591)

Toiseksi OSS 2.0 siirtää ohjelmistokehityksen painopistettä horisontaalisista palveluista tiettyihin aloihin erikoistuneisiin vertikaalisiin palveluihin. Tällaiset ohjelmistot ovat usein myös huomattavasti näkyvämpiä kuin taustalla toimivat horisontaaliset palvelut. Käytännössä tällainen ohjelmistokehitys voi näkyä mm. siten, että jokin tietty organisaatio antaa erikoisosaamistaan omaa alaansa palvelevan ohjelmistoprojektin käyttöön (mt., 591)

Kolmanneksi avoimen lähdekoodin ansaintalogiikat kehittyvät OSS 2.0 -tilanteessa liiketoimintaa paremmin tukeviksi. (mt., 591-592). Liiketoimintamalleihin palataan tarkemmin luvussa 6.3.

Neljänneksi ohjelmistotuotteiden tukipalvelut muuttuvat ammattimaisemmiksi. Sen sijaan, että asiakkaat etsisivät tukea ohjelmistoihinsa kehittäjäyhteisöltä, kuten internetin keskustelualueilta, he ovat valmiita maksamaan, tuesta, koulutuksista ja sertifioinnista. Yritykset vastaavat tähän kysyntään. Useat eri yritykset voivat antaa tukipalveluja yhden tuotteen ympärille. (mt., 590, 592-593)

Viidenneksi lisensointikäytännöt muuttuvat aikaisempaa myönteisemmiksi liiketoimintaa kohtaan. GPL-lisenssiä pidetään usein liian rajoittavana, sillä sen mukaan yhteisön etu menee yritysten liiketoimintaintressien edelle. (mt., 593-594). Tarkemmin lisensointikäytäntöjä käydään läpi seuraavassa alaluvussa.

2.2.3 Lisensointikäytännöt avoimen lähdekoodin ohjelmistojen oikeudellisena perustana

Avoimen lähdekoodin syntyhistoriaan liittyy paljon ideologista keskustelua ohjelmistojen luonteesta ja lähdekoodin vapaudesta. Unixin synty ja kehittäminen nostivat esiin kaupallisten toimijoiden ja ohjelmoijien välisen ristiriidan. Stallmanin määritelmä ohjelmistojen vapaudesta ja GPL-lisenssi syntyivät vastareaktiona omisteisten ohjelmistojen laajalle leviämiselle (Miettinen ym. 2006b, 69; Moody 2001). Toisaalta läheskään kaikki avoimen lähdekoodin ohjelmistokehitykseen osallistuvat eivät suhtaudu ohjelmistojen vapauteen yhtä tiukasti kuin Stallman ja hänen perustamansa Free Software Foundation. Monille avoimen lähdekoodin liikkeessä toimiville avoin kehittäminen nähdään

ensisijaisesti välineenä luoda hyvälaatuisia ohjelmistoja. (Moody 2001)

Tiukat vaatimukset asettavan GPL-lisensin lisäksi on olemassa myös suuri joukko muita lisenssejä, joita pidetään avoimen lähdekoodin lisensseinä. Open Source Initiativen mukaan avoin lähdekoodi ei tarkoita vain lähdekoodin saatavuutta. Ohjelmiston lisenssiä voidaan kutsua open source -lisensiksi, mikäli se täyttää taulukossa 2.1. kuvatut ehdot:

Open Source -määritelmä on paljon sallivampi ja väljempi kuin GPL-lisenssi ja Free Softwaren määritelmä. Lisenssit voidaankin jakaa karkeasti kolmeen eri ryhmään:

1. Vahvaa vastavuoroisuutta edellyttävät lisenssit (tarttuvat)
2. Vastavuoroisuutta edellyttävät lisenssit
3. Sallivat lisenssit (ei-tarttuvat)
   (JHS 169 2009, 25; Välimäki 2009, 203- 208)

Vahvaa vastavuoroisuutta edellyttävistä lisensseistä tyypillisin on Stallmanin luoma GPL-lisenssi ja sen versiot 2 ja 3. Kuten aiemmin on todettu, näitä lisenssejä yhdistää niiden tarttuva luonne eli muokattu koodi tulee edelleen julkaista samalla lisenssillä ja mikäli koodia käytetään osana muuten lisensoitua ohjelmaa, koko kokonaisuus tulee julkaista tarttuvan lisensin alaisuudessa. Tätä vaatimusta Stallman kuvaa käsitteellä "copyleft". (Lee 2006, 51-52; JHS 169 2009, 26; Välimäki 2009, 205-206)

Taulukko 2.1:Open Source -lisenssien ehdot

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.
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.
3. Johdannaiset teokset
Lisenssin on sallittava muutosten tekeminen ja johdannaisten teosten luominen. Näitä on saatava levittää samoilla lisenssiehdoilla kuin alkuperäistä ohjelmaa.
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.
5. Henkilöiden ja ryhmien syrjinnän kielto
Lisenssi ei saa syrjiä ketään henkilöä tai henkilöryhmää.
6. Toimialojen syrjinnän kielto
Lisenssi ei saa syrjiä ketään käyttämästä ohjelmaa tietyllä toimialalla. On esimerkiksi kiellettyä rajoittaa ohjelman käyttöä liiketoiminnassa tai genetiikan tutkimuksessa.
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.
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.
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.

(Open Source Initiative)

Sallivat lisenssit sen sijaan eivät pidä sisällään "copyleft" -vaatimusta. Näiden lisenssien alla julkaistut ohjelmistot sallivat koodin muokkaamisen ilman, että muokattua ohjelmistoa tulisi julkaista samojen lisenssiehtojen alla. Lisenssiehdot eivät siis ole tarttuvia, ja näin koodia voi käyttää haluamallaan tavalla myös omisteisissa ohjelmistoissa ilman

vaatimusta koodin julkaisemisesta. (Lee 2006, 52; JHS 169 2009, 25; Välimäki 2009; 208-210)

Esimerkkejä sallivista lisensseistä ovat mm. Apache-lisenssi ja Berkeley Software Distribution (BSD) -lisenssi. Apache on laajimmin käytössä oleva web-palvelinohjelmisto. Salliva lisenssi mahdollistaa Apachen koodin käyttämisen vapaasti osana kaupallisia tuotteita. BSD-lisenssi taas on luultavasti vanhin avoimen lähdekoodin lisenssi. Se mahdollistaa lähdekoodin salassa pitämisen ja sopii näin usein GPL-lisenssiä paremmin kaupallisiin intresseihin. BSD-lisensiin perustuvaa koodia onkin käytetty mm. Mac OS X -käyttöjärjestelmän perustana. Samoin Microsoft on käyttänyt BSD-lisensoitua koodia osana käyttöjärjestelmiään. (Lee 2006, 52-53; Välimäki 2009; 208-210)

Vahvaa vastavuoroisuutta edellyttävien ja sallivien lisenssien väliin asettuvat vastavuoroisuutta edellyttävät lisenssit. Useimmiten näissä lisensseissä edellytetään koodimuutosten julkistamista, mutta ohjelmistoja saa vapaammin yhdistellä muilla lisensseillä tehtyihin ohjelmistoihin. Tällaisista lisensseistä yleisimpiä ovat mm. Eclipse Public License (EPL), Mozilla Public License (MPL) ja GNU Lesser General Public License (LGPL). (JHS 169 2009, 26; Välimäki 2009, 217- 221)

Eri lisenssityyppien eroja havainnollistetaan taulukossa 2.2.:

Taulukko 2.2: Avoimen lähdekoodin lisenssityypit

	Vapaa levitys	**Vapaa käyttö**	**Vapaa lähdekoodi**	**Vastavuoroisuus**	**Vahva vastavuoroisuus**	**Esimerkkejä ohjelmista**
Suljettu omisteinen ohjelma						Microsoft Windows ja Office
BSD, Apache, MIT...	**x**	**x**	**x**			Apache, Apple Darwin, Google Android, FreeBSD
LGPL, MPL, EPL...	**x**	**x**	**x**	**x**		Firefox, OpenOffice.org Eclipse,
GPL 2, GPL 3, CPL...	**x**	**x**	**x**	**x**	**x**	Linux, Java, MySQL, Moodle, Gnome

(Muokattu JHS 169 2009, 26; Välimäki 2009; 207)

2.3 Aiempi tutkimus avoimen lähdekoodin käytöstä terveydenhuollon tietojärjestelmissä

Sosiaali- ja terveydenhuollon tietohallinnon tutkimuksessa avointa lähdekoodia on käsitelty varsin vähän. Alan julkaisuista löytyy joitain artikkeleita, joissa on lähinnä esitelty avoimen lähdekoodin mahdollisuuksia terveydenhuollon tietohallinnon ohjelmistojen kehittämiseen.

Carnall (2000, 976) nosti esiin avoimen lähdekoodin kehittämistavan mahdollisuudet terveydenhuollon tietohallinnossa British Medical Journalin pääkirjoituksessa lokakuussa 2000. Carnallin mukaan terveydenhuollon tietohallinnossa yksi merkittävimmistä ongelmista on riippuvuus yksittäisistä järjestelmän toimittajista. Asiakkaat joutuvat maksamaan suuriakin lisenssimaksuja ohjelmistoyrityksille, ja usein siirtyminen toiseen järjestelmätoimittajaan tulee vielä kalliimmaksi. Carnallin mukaan avoimen lähdekoodin ohjelmat ovat yksi ratkaisu saavuttaa riippumattomuutta järjestelmätoimittajista: ohjelmat ovat vapaasti käytettävissä ja muokattavissa ilman lisenssimaksuja. Vapaa muokattavuus on myös yksi avoimen lähdekoodin ohjelmien keskeisimpiä etuja, sillä ohjelmoijat voivat keskittyä muokkaamaan ohjelmiston osia paremmin organisaation tarpeita palvelevaksi.

Muina avoimen lähdekoodin etuina Carnall mainitsee luotettavuuden ja turvallisuuden. Koska ohjelmien lähdekoodi on kenen tahansa tarkastettavissa, se voidaan käydä läpi ennen käyttöä varten kääntämistä. Avoimuus myös takaa sen, että ohjelmistojen ylläpitäminen on mahdollista myös muilta kuin sen alkuperäisiltä tekijöiltä. Näin myös aiemmin tuotettua ohjelmakoodia voidaan käyttää uusien ohjelmaprojektien pohjana. Ohjelmistojen kehittämisessä voidaankin keskittyä uusien innovaation kehittämiseen sen sijasta, että samoja ongelmia ratkotaan moneen kertaan. (Carnall 2000, 976)

Shaw, Pepper, Cook, Houwink, Jain & Bainbridge (2002) ovat esittäneet mahdollisia suuntauksia, kuinka avoimen lähdekoodin ohjelmistot tulevat vaikuttamaan terveydenhuollon tietohallintoon. He esittävät avoimen lähdekoodin eduiksi ohjelmistojen laadukkuuden, alentuneet kehityskulut, asiakastyytyväisyyden, suuremman markkinaosuuden, pienemmän riskin joutua oikeudellisiin ongelmiin sekä parantuneen turvallisuuden. Kir-

joittajien mukaan suurimmat syyt avoimen lähdekoodin hitaaseen yleistymiseen terveydenhuollon tietohallinnossa ovat kriittisten tietojärjestelmien turvallisuusvaatimukset, terveydenhuollon henkilöstön alhainen tietotekninen osaaminen, tiukat vaatimukset tietoturvalle sekä lainsäädännön ja standardien vuoksi tarvittavat säännölliset parannukset ohjelmistoihin.

Artikkelin kirjoittajat antavat neljä suositusta, jotka edistäisivät avoimen lähdekoodin käyttöönottoa kansainvälisesti terveydenhuollon tietohallinnossa. Kirjoittajien mukaan tarvittaisiin kansainvälinen yhteisö, joka ottaisi vastuun kehittämisestä ja terveydenhuollon tietohallinnon tarpeita varten tulisi kehittää oma järjestelmänydin. Keskeistä olisi kehittää kaupallisia palveluja avoimen lähdekoodin tietohallinnon tarpeisiin ja panostaa riittävästi kansainvälisten standardien kehittämisen. (Shaw ym. 2002, 41-42)

Kirjoittajien mukaan avoin lähdekoodi ei automaattisesti ole ratkaisu terveydenhuollon tietohallinnon kaikkiin ongelmiin. Ennen kaikkea avoin lähdekoodi kuitenkin antaa mahdollisuuksia kehittää yhteistyötä ja ottaa paremmin huomioon käyttäjien tarpeita ohjelmistojen kehittämisessä. (Shaw ym. 2002, 43)

McDonald, Schadow, Barnes, Dexter, Overhage, Mamlin & McCoy (2003, 175-183) ovat esitelleet avoimen lähdekoodin kehittämismallia ja keskeisimpiä ohjelmistoprojekteja terveydenhuollon tietohallinnon alalla. Kirjoittajat näkevät avoimen lähdekoodin mahdollistavan uudenlaisen yhteyden tietohallinnon tutkimuksen ja käytössä olevien tietojärjestelmien välillä. Avoin lähdekoodi mahdollistaa tutkijoiden kehittämien uusien innovaatioiden kokeilemisen tietojärjestelmissä nykyistä helpommin. Kirjoittajien mukaan tämä vapauttaisi luovuutta ja nopeuttaisi kehitystä. Laaja avoimen lähdekoodin käyttö hyödyttäisi myös kaupallisia toimijoita vähentämällä kehitystyöhön liittyviä riskejä. Kaupalliset toimijat voisivat siirtää voimavarojaan erityisosaamista vaativiin palveluihin, kuten käyttöönottoon, käyttötukeen ja järjestelmien integrointiin. Tällaisessa tilanteessa kehitystyö olisi nykyistä edullisempaa ja innovaatioita tapahtuisi nopeammin. (McDonald ym. 2003, 179)

Artikkelin kirjoittajat painottavat standardien (mm. HL 7, DICOM, SQL, ODBC, PDF) merkitystä avoimen lähdekoodin käyttöönoton leviämisessä. Terveydenhuollon toimijat

tulevat varmasti jatkossakin käyttämään kaupallisesti tuotettuja ohjelmistoja, mutta standardien käyttäminen varmistaisi sen, että avoimeen lähdekoodin perustuvat ohjelmistomoduulit kykenevät kommunikoimaan suljettujen ohjelmien kanssa. (McDonald ym. 2003, 180)

Artikkelin mukaan avoimen lähdekoodin yleistymistä edistäisi parhaiten se, että julkisella rahoituksella kehitetyt ohjelmistot julkaistaisiin aina avoimen lähdekoodin lisenssin alaisuudessa. Toinen merkittävä edistävä tekijä olisi standardien käyttäminen, jota tulisi aina vaatia julkista rahoitusta jaettaessa. Kolmanneksi jo olemassa olevien avoimen lähdekoodin ohjelmistojen käyttöä tulisi suosia uusien hankkeiden perustana. Kirjoittajat pitävät kuitenkin tärkeimpänä tekijänä tiedon jakamista avoimen lähdekoodin mahdollisuuksista. (McDonald ym. 2003, 182)

Paré, Wybo & Delannoy (2009) haastattelivat kanadalaisia terveydenhuollon IT-johtajia selvittääkseen, mitkä asiat ovat keskeisimpiä avoimen lähdekoodin ratkaisujen käyttöönoton esteitä terveydenhuollossa. Haastatteluissa ilmeni seitsemän keskeistä estettä. Ensinnäkin terveydenhuollon organisaatioissa on nykyisin hyvin vähän resursseja ja osaamista, jotta ne voisivat ottaa vastuuta myös ohjelmistojen kehittämisestä ja ylläpidosta. Toiseksi poliittiset paineet voivat ohjata toimintaa tiettyyn suuntaan, kuten tiettyjen ohjelmistotoimittajien tuotteiden suosimiseen. Kolmanneksi avoimeen lähdekoodiin perustuvista terveydenhuollon ohjelmistoista on hankala saada luotettavaa tietoa. Neljäntenä tekijänä tunnistettiin terveydenhuollolle tyypillinen konservatiivinen suhtautuminen uudistuksiin. Viidentenä asiana haastateltavat nostivat esiin avoimen lähdekoodin ohjelmistoista kokonaisvastuun ottavien toimittajien puutteen. Tarvittaisiin toimittajia, jotka voitaisiin sopimuksellisesti sitouttaa tuki- ja ylläpitotoimintoihin ja näin varmistettaisiin jatkuvuus ja tuotteen kehittäminen. Kuudentena esteenä pidettiin organisaatioiden keskinäistä kilpailevaa kulttuuria. Viimeisenä, seitsemäntenä, esteenä pidettiin avoimen lähdekoodin tuotteiden piilokuluja. Vaikka itse ohjelmistosta ei tarvitsisi maksaa mitään, niin se tarvitsee ylläpitoa, korjauksia, päivityksiä ja koulutusta. Nämä kaikki maksavat. (Pare ym. 2009, 1-5)

Tutkimuksen tekijät tulivat siihen johtopäätökseen, että merkittävimmät esteet liittyvät poliittisiin paineisiin ja tiedonpuutteeseen avoimen lähdekoodin perustuvista kaupalli-

sista ratkaisuista. Haastatteluissa ilmeni, että paikallinen ministeriö toimi tiiviissä yhteistyössä yhden ohjelmistotoimittajan kanssa ja siksi ministeriön linjavalinnasta poikkeaminen oli hyvin vaikeaa. Jotta avoimen lähdekoodin tuotteet voisivat kilpailla omisteisten ohjelmistojen kanssa, tulisi niitä tarjoavien yritysten pystyä osoittamaan olevansa kaupallisesti uskottava vaihtoehto ohjelmistokehityksessä ja tukitoiminnoissa nykyisille ratkaisuille. Tämä ei todennäköisesti ole mahdollista ilman avoimen lähdekoodin yritysten markkinointi- ja myyntitoiminnan kehittymistä. (Pare ym. 2009, 3, 7)

Janamanchi, Katsamagas, Rahgupati & Gao (2009, 457, 459) kävivät läpi 174 avoimeen lähdekoodin perustuvaa terveydenhuollon tietojärjestelmäprojektia. Projektit haettiin Sourceforge:sta, joka on suurin avoimen lähdekoodin kehitysprojekteille suunnattu verkkopalvelu. Tutkimuksessa selvitettiin avoimen lähdekoodin ohjelmistojen laajuutta ja valikoimaa terveydenhuollossa. Terveydenhuollon organisaatiot osoittivat kasvavaa kiinnostusta moniin ohjelmistoprojekteihin ja toimivat niiden taloudellisina tukijoina. Taloudellinen tuki vaikutti positiivisesti projektiin osoittaen organisaatioiden sitoutumista, vetäen puoleensa kehittäjiä sekä antamalla tarvittavia resursseja ja tukea projektin onnistumiseksi. Kirjoittajat kehottavatkin terveydenhuollon organisaatioita lähtemään projektien tukijoiksi, sillä ne itsekin hyötyvät projekteissa kehitettyjen ohjelmistojen parantumisesta.

Kaksi kolmasosaa ohjelmistoprojekteista oli valinnut vastavuoroisuutta edellyttävän lisenssin. Tällä estetään koodin käyttäminen kaupallisissa ohjelmistoissa. Tutkimuksen mukaan tätä arvostettiin terveydenhuollon organisaatioissa, sillä lisensointimalli takaa tuotosten pysymisen kaikkien hyödynnettävinä. Tämä myös kannustaa kehittäjiä, sillä heidän työtään ei voida tulevaisuudessa käyttää epätoivotulla tavalla. (Janamanchi ym. 2009, 470)

3 TUTKIMUSPROSESSI

3.1 Järjestelmällinen kirjallisuuskatsaus tieteellisenä menetelmänä

Tutkielman aineiston hankintamenetelmänä oli tieteellisiin tietokantoihin tehtävä kirjallisuuskatsaus. Aineiston hankkiminen pyrittiin tekemään mahdollisimman systemaattisesti. Aineiston hankintatapa tuleekin lähelle järjestelmällistä kirjallisuuskatsausta, joita kutsutaan usein myös systemaattisiksi kirjallisuuskatsauksiksi, systemaattisiksi tutkimuskatsauksiksi tai systemaattisiksi tietokatsauksiksi. (Varonen, Semberg & Teikari 1999) Tieteellisen tiedon lisääntymisen vuoksi tarvitaan keinoja, joilla voidaan koota ja tiivistää tutkimustietoa paremmin hyödynnettävään muotoon. Järjestelmällinen kirjallisuuskatsaus on tieteellinen menetelmä, jonka avulla tätä tiivistämistä voidaan tehdä. (Kääriäinen & Lahtinen 2006, 37; Koivisto & Haverinen 2006, 112). Nostamalla esiin aikaisempia tutkimuksia pyritään selvittämään, onko aiheesta jo olemassa tutkimustietoa ja miten ja mistä näkökulmista ilmiötä on aiemmin tutkittu. Järjestelmällisiä kirjallisuuskatsauksia pidetään luonteeltaan teoreettisina tutkimuksina (Tuomi & Sarajärvi 2002, 110-120; Varonen ym. 1999).

Järjestelmällisen kirjallisuuskatsauksen ydin on sen suunnitelmallisuudessa, toistettavuudessa ja harhattomuudessa. (Johansson 2007, 4) Järjestelmällinen kirjallisuuskatsaus voidaan toteuttaa hieman toisistaan poikkeavin tavoin. Erityisesti lääketieteessä sitä on käytetty näyttöön perustuvaan tutkimukseen ja vaikuttavuustutkimuksiin. Tutkimustuloksia yhdistetään käyttäen kvantitatiivisia menetelmiä. Tällöin tällaista katsausta kutsutaan meta-analyysiksi. Laadullisessa tutkimuksessa taas käytetään kvalitatiivisia menetelmiä aikaisempien tutkimustulosten yhdistämisessä ja niiden uudelleen analysoinnissa. (Käärinen & Lahtinen 2006, 38; Virtanen & Salanterä 2007, 72-85)

3.2 Aineiston haku ja valintaprosessi

Järjestelmällisissä kirjallisuuskatsauksissa tutkimusprosessi tulee kuvata ja suunnitella siten, että lukija pystyy seuraamaan prosessin etenemistä ja pystyy ymmärtämään, miten tulokset on saatu ja miten tutkimus voitaisiin toistaa. (Koivisto & Haverinen 2006, 31; Hirsjärvi, Remes & Sajavaara 2004, 217). Järjestelmällisen kirjallisuuskatsauksen vai-

heet ovat seuraavat (Kääriäinen & Lahtinen 2006: 39-43; Johansson 2007, 5):

- Tutkimussuunnitelman laatiminen
- Tutkimuskysymysten määrittäminen
- Aineiston haku ja valintaprosessi
- Laadun arviointi
- Tulosten analysointi ja raportointi

Aineistonhakustrategiassa määritellään tietokannat, rajaukset ja hakusanat. Keskeistä on valintakriteerien määrittely, joka tarkoittaa aineiston sisäänotto- ja poissulkukriteerien määrittämistä. Nämä toimivat aineiston hyväksymis- ja hylkäämiskriteereinä. (Stolt & Routasalo 2007, 59). Saadut hakutulokset käydään läpi otsikon, asiasanojen tiivistelmän ja artikkelin perusteella. (Koivisto & Haverinen 2006, 108-126; Kääriäinen & Lahtinen 2006, 39-43).

Kirjallisuuskatsauksen toteuttaminen aloitettiin koehakujen tekemisellä. Hakuja tehtiin eri tietokantoihin ja samalla kokeiltiin erilaisia hakulausekkeita. Tässä vaiheessa käytettiin mm. seuraavia hakusanojen yhdistelmiä: *"open source health informatics", "open source medical informatics", "open source health record"*. Hakusanoja yhdistelemällä syntyi alustava kokonaiskuva siitä, millaisia artikkeleita tutkimusaiheesta on julkaistu.

Parhaiten kirjallisuuskatsauksen hakuja vastaavia tuloksia saatiin terveydenhuollon merkittävimmästä tietokannasta PubMedista. Hakujen tekemistä varten konsultoitiin Kuopion yliopiston kirjaston informaatikkoa. Hän suositteli laajentamaan hakuja myös tietojenkäsittelyä koskeviin tietokantoihin. Niistä parhaiten tutkimusta vastaavia hakutuloksia antoi INSPEC, joka on merkittävin tietojenkäsittely- ja tietojärjestelmätieteen artikkeleita sisältävään tekniikan alan tietokanta.

Koehakujen aikana muodostettiin myös tutkimusten valintakriteerit. Mukaan analysoitaviksi päätettiin hyväksyä artikkelit, jotka täyttivät seuraavat kriteerit:

- artikkelin tulee käsitellä terveydenhuollon ohjelmistoja ja avointa lähdekoodia
- artikkeli tulee olla julkaistu tieteellisessä journal-lehdessä

- artikkeli tulee olla saatavissa Kuopion yliopiston kirjaston käyttöoikeuksin

Poissulkemiskriteereiksi asetettiin seuraavat tekijät:

- artikkeli käsittelee bioinformatiikkaa
- artikkeli käsittelee lääkekehitystä
- artikkeli ei ole varsinainen tieteellinen artikkeli vaan esimerkiksi konferenssipaperi

Varsinainen haku tehtiin molempiin tietokantoihin seuraavanlaisella hakusanojen yhdistelmällä: *open source AND (health OR medical) AND (informatics OR record)*. Hakua rajattiin valitsemalla PubMed -tietokannassa rajoitus *limits: only items with link to full text.* Tällä haulla PubMed -tietokanta antoi 181 tulosta. Hakutuloksen antamat artikkelit käytiin läpi yksitellen lukemalla otsikko ja artikkelin abstrakti. Sisäänotto- ja ulossulkukriteerien jälkeen analysoitavaksi jäi 64 artikkelia.

Haku tehtiin samalla lausekkeella INSPECiin. Tuloksia tuli 74. Sisäänotto- ja ulossulkukriteerien jälkeen analyysiin jäi yhteensä yhdeksän artikkelia. Näistä seitsemän oli kuitenkin jo löytynyt PubMed -tietokannan kautta.

Tässä vaiheessa kirjallisuushaun kriteerit täyttäviä artikkeleita oli yhteensä 66 kappaletta. Artikkeleihin tutustumisen myötä niiden määrä tuntui suurelle. Tämän vuoksi sekä sisäänotto- että ulossulkukriteerejä tiukennettiin. Sisäänottokriteeriksi lisättiin:

- Artikkelin tulee käsitellä konkreettista terveydenhuollon ohjelmistoprojektia

Ulossulkukriteereiksi lisättiin:

- artikkeli on yleinen kuvaus avoimen lähdekoodin periaatteista tai mahdollisuuksista terveydenhuollon tietohallinnossa
- artikkeli käsittelee verkko-oppimista
- artikkeli käsittelee infektioiden leviämisen ennustamiseen liittyviä ohjelmistoja
- artikkeli käsittelee laboratorionäytteiden analysointiohjelmistoja

Näillä lisäkriteereillä artikkelien joukko rajautui selvästi pienemmäksi. Yhteensä artikkeleita otettiin analyysiin 29 kappaletta. Hakujen kautta saaduista artikkeleista, erityisesti avoimen lähdekoodin yleisiä mahdollisuuksia käsitteleviä artikkeleita pystyttiin hyödyntämään tutkielmassa aiempia tutkimustuloksia käsittelevässä luvussa.

Tässä tutkielmassa on syytä huomata, että tutkimusaineisto ei täytä täysin yleisesti järjestelmällisille kirjallisuuskatsauksille vaadittavia kriteereitä. Tutkimusaineisto on julkaistu tieteellisissä julkaisuissa, mutta useimmat analyysiin valitut artikkelit eivät ole varsinaisesti tutkimuksia, vaan ennemminkin ohjelmistoprojektia kuvailevia artikkeleita. Näissä artikkeleissa kuvataan ja pyritään ymmärtämään projektin toimintaa ja tavoitteita joko ohjelmistoprojektin sisältä tai jonkin ulkopuolisen kuvaamana. Näin ollen tässä tutkielmassa ei varsinaisesti tehdä järjestelmällisille kirjallisuuskatsauksille tyypillistä ns. toisen asteen tutkimusta, joka kohdistuu jo olemassa olevaan tutkimustietoon (Pekkala 2001, 59; Koivisto & Haverinen 2006, 112). Pikemminkin kysymys on laadullisin menetelmin tehdystä teoreettisesta tutkimuksesta, jossa aineiston haku, valintaprosessi ja laadun arviointi tehdään järjestelmällisen kirjallisuuskatsauksen menetelmin. Tätä tutkielmaa voikin luonnehtia laadulliseksi teoreettiseksi tutkimukseksi, jossa aineisto kerätään järjestelmällisen tiedonhankinnan menetelmin.

3.3 Aineiston analyysi

Laadullisen aineiston analyysin voidaan jakaa aineistolähtöiseen, teoriasidonnaiseen ja teorialähtöiseen. Aineistolähtöisessä analyysissä pyritään luomaan teoreettinen kokonaisuus tutkimusaineistosta käsin. Analyysiyksiköt eivät ole etukäteen sovittuja tai harkittuja. Teoriasidonnaisessa analyysissä analyysiyksiköt valitaan aineistosta, mutta aikaisempi tieto ohjaa ja auttaa analyysin etenemisessä. Analyysistä on tunnistettavissa aikaisemman tiedon vaikutus, mutta sen merkitys ei ole teoriaa testaava vaan pikemminkin uusia ajatusuria aukova. (Tuomi & Sarajärvi 2004, 97-99)

Teorialähtöisessä analyysissä nojaudutaan johonkin tiettyyn teoriaan, malliin tai jonkin auktoriteetin esittämään ajatteluun. Tutkimuksessa kuvataan tämä malli ja sen pohjalta määritellään mm. tutkimuksessa kiinnostavat käsitteet. Tutkittava ilmiö määritellään

jonkin aiemmin tunnetun mukaisesti ja aineiston analyysiä ohjaa valmis aiemman tiedon perusteella luotu kehys. Tämäntyyppistä analyysiä käytetään usein aikaisemman tiedon testaamiseen uudessa kontekstissa. (mt., 99-100)

Tässä tutkielmassa aineisto analysoitiin teoriasidonnaisesti. Ennen aineiston hakua tietokannoista luettiin aiempaa avointa lähdekoodia käsittelevää tutkimusta, jotta aihepiiri tulisi tutuksi ja tutkimuskysymykset pystyttäisiin muodostamaan. Tämän tutkielman kannalta hyödyllisimmiksi teoreettisiksi lähtökohdiksi osoittautuivat Steven Weberin (2004), Eric von Hippelin (2005) ja Reijo Miettisen tutkimusryhmän (2006) teokset. Tutkimuskysymysten pohjalta luotiin apukysymyksiä aineiston analyysia varten. Nämä olivat:

- Miten avoimen lähdekoodin käytön puolesta argumentoidaan?
- Miten kehittäjäyhteisöt toimivat?
- Miten modulaarisuus näkyy ohjelmistoprojekteissa?
- Millaisilla liiketaloudellisilla malleilla yritykset toimivat?
- Miten käyttäjien ja valmistajien välistä suhdetta kuvataan?

Näiden teemojen tunnistamisen jälkeen luotiin matriisi analyysin pohjaksi (mukaillen Tuomi & Sarajarvi 2004, 116-117) . Jokainen analyysiin valittu artikkeli luettiin läpi siten, että matriisiin kerättiin teemoja koskevat ilmaisut.

Taulukko 3.1: Aineiston analyysimatriisi

						Pääteemat					
Nro.	Kirjoittajat	Artikkelin nimi	Keskeinen sisältö	Millainen tietojärjestelmä	Lisenssityyppi	Miksi open source?	Kehittäjäyhteisöil-maisut	Modulaarisuus	Liiketoimintamalli	Käyttäjät ja valmis-tajat	Muuta huomioitavaa

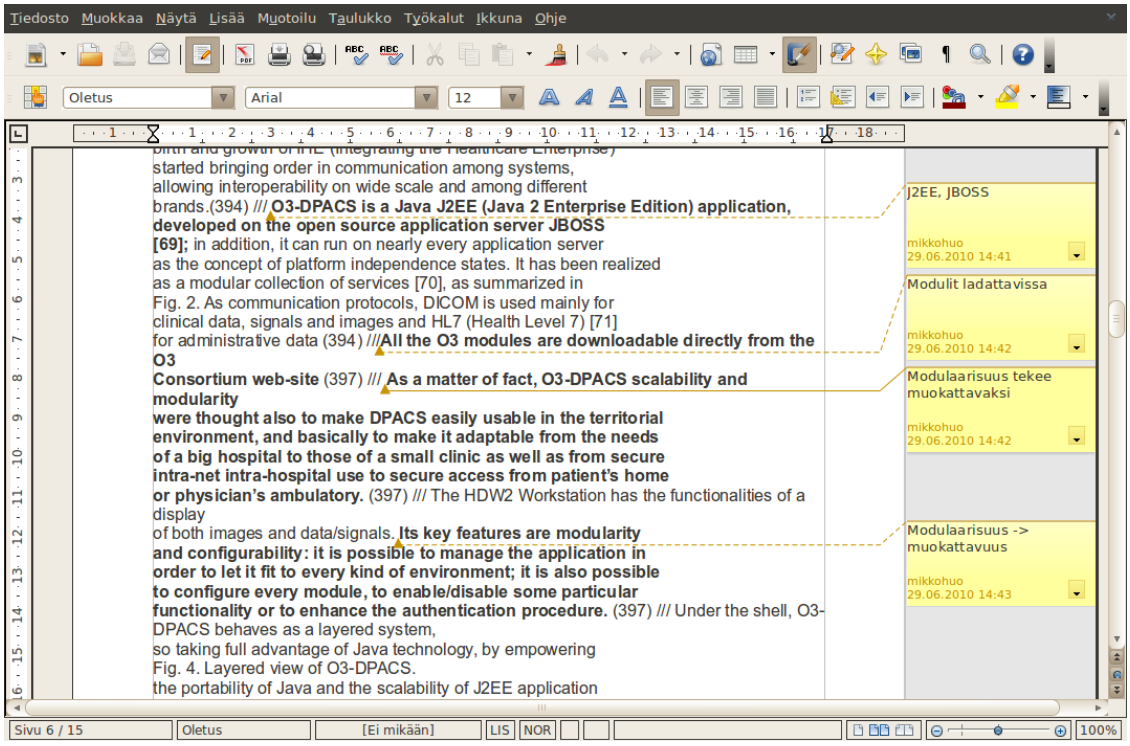
Aineiston analyysin ensimmäinen vaihe syntyi matriisin avulla. Käytännössä kysymys oli teemoittelusta, jossa aineiston teksti järjestettiin uudelleen teemojen mukaisesti. (Eskola & Suoranta 2003, 174-175; Saaranen-Kauppinen & Puusniekka 2006; Pyörälä 1995, 15-16) Taulukkolaskentaohjelmaan tehdyn matriisin solut purettiin teemoittain

omiksi tekstinkäsittelytiedostoiksi. Tässä vaiheessa huomattiin, että erityisesti kehittäjäyhteisöjä koskevat ilmaisut sekä käyttäjien ja valmistajien välistä suhdetta koskevat ilmaisut käsittelivät hyvin samanlaisia asioita. Näin ollen oli luontevaa yhdistää nämä teemat. Liiketoimintamalleja käsitteleviä ilmaisuja löytyi aineistosta varsin vähän ja nekin käsittelivät pääasiassa yritysten roolia kehittäjäyhteisön osana. Nämä ilmaisut päätettiin liittää myös saman kehittäjäyhteisön organisoitumista käsittelevän pääteeman alle. Näin tutkielman rakenne alkoi muotoutua kolmen pääteeman, motiivien, kehittäjäyhteisön organisoitumisen, sekä avoimelle lähdekoodille tyypillisten välineiden ja tekniikoiden, kautta.

Kun pääteemoja koskevia ilmaisuja käytiin läpi niissä selvästi toistui samoja asioita. Näin aineiston järjestäminen edelleen alateemoiksi tuntui mielekkäälle (Saaranen-Kauppinen & Puusniekka 2006). Alateemojen muodostamisessa käytettiin tekstinkäsittelyohjelman kommentointitoimintoa. Teemoittain järjestettyjä ilmaisuja luettaessa tehtiin kommentointitoiminnolla muistiinpanoja (Eskola 2007, 171). Näiden merkintöjen pohjalta alateemojen luominen helpottui huomattavasti. Tässä vaiheessa myös siirrettiin joitain ilmaisuja pääteemasta toiseen.

Tutkielman päälukujen rakenne perustuu aineiston teemoittelulle. Kuvaus pääteemoista, alateemoista ja niihin liittyvistä ilmaisuista on taulukoitu jokaisen pääluvun loppuun.

Eskola & Suoranta (2003, 175, 179) esittävät, että teemoittelu analyysimenetelmänä edellyttää tiivistä empirian ja teorian vuoropuhelua. Teorian ja empirian välinen yhteys jää tutkimusraporttia kirjoitettaessa kuitenkin helposti ohueksi. Tämä vaara on erityisesti laadullisessa tutkimuksessa, sillä usein tutkimuksen alkupuolella esitetty teorialuku jää kokoelmaksi erilaista luettua materiaalia, jota on vaikea hyödyntää empirialuvuissa ja työn päättävissä pohdinnoissa. (Eskola 2007, 163) Aineiston analyysissa tätä pyrittiin estämään sillä, että aiempi tutkimus ohjasi etenemistä vahvasti analyysimatriisin kautta. Tutkielman raportoinnilla ns. "tuplasuppilomallin" mukaisesti pyrittiin myös saamaan tiivistä yhteyttä teorian, aiemman tutkimuksen ja empirian välille.



Kuva 3: Teemoittelu tekstinkäsittelyohjelmassa

4 AVOIMEN LÄHDEKOODIN KÄYTÖN MOTIIVIT

4.1 Avoimen lähdekoodin käytön hyödyt

Avoin lähdekoodi on herättänyt paljon kiinnostusta julkisissa toimijoissa ympäri maailmaa. Euroopassa ilmiöstä on käyty keskustelua, niin paikallisella tasolla, kansallisesti kuin Euroopan Unionin instituutioissakin. Ghosh (2005, 7-10) on analysoinut tätä poliittista keskustelua ja jakanut keskustelussa esitetyt argumentit kahteen kategoriaan:

4.1.1 Yhteiskunnalliset hyödyt

Euroopan unionissa on käytetty käsitettä "eInclusion" kuvaamaan *kansalaisten oikeutta päästä osalliseksi julkisesta tiedosta ja palveluista*. Avoin lähdekoodi on nähty mahdollisuutena tuoda näitä sähköisiä palveluja myös niiden kansalaisten ulottuville, joiden taloudelliset edellytykset ovat heikot tai joilla on vammaisuudesta johtuvia rajoitteita. (Ghosh 2005, 7-8)

Toisena yhteiskunnallisena hyötynä Ghosh (2005, 8) esittää *itsenäisyyden ohjelmistotoimittajista, mahdollisuuden muokata ohjelmistoa ja tämän myötä tulevat paikalliset taloudelliset hyödyt*. Avoin lähdekoodi antaa paikallisille toimijoille mahdollisuuden itsenäisesti räätälöidä käyttämistään ohjelmistoista tarpeisiinsa sopivia. Riippumattomuus yksittäisistä isoista ohjelmistotoimittajista antaa mahdollisuuksia paikallisille ohjelmistoyrityksille osallistua ohjelmistojen räätälöintiin. Tämä mahdollisuus voi antaa merkittäviä sysäyksiä paikalliselle ohjelmistoihin liittyvälle liiketoiminnalle ja näin tukea työllisyyttä. Monissa Euroopan maissa onkin tehty päätöksiä, joilla on haluttu edistää avoimen lähdekoodin käyttöä. Valmiiden ohjelmistojen räätälöinnissä on paikallisilla yrityksillä ollut merkittävä rooli (Ghosh 2005, 8, 10-11; Grassmuck 2005; 14-35)

Avoimella lähdekoodilla on myös mahdollista parantaa tietoyhteiskuntakehityksessä jälkeen jääneitä alueita nousemaan parempaan asemaan, sillä muualla kehitettyjä ohjelmistoja ja teknologisia ratkaisuja voidaan ottaa käyttöön ilman lisenssikuluja. Tästä on käytännössä hyviä kokemuksia niin EU:n sisällä kuin kehitysmaissakin. (mm. Ghosh 2005, 11-13; Kagai & Kimolo 2005, Kim 2005; Coleman 2005)

Kolmas yhteiskunnallinen hyöty on *demokraattisen läpinäkyvyyden parantuminen*. Demokraattisissa hallintojärjestelmissä kansalaisilla on oikeus tietoon ja viranomaisten toiminnassa syntyvät asiakirjat ovat lähtökohtaisesti julkisia. Julkisuusperiaatteen juuret voidaan nähdä jo Jeremy Benthamin ajattelussa. Hänen mukaansa edustuksellinen demokratia ja parlamentarismi tarjoavat kansalaisille mahdollisuuden valvoa edustajiaan niin, että edustajien toimet edistävät kaikkien hyvinvointia. Kansalaisten luottamus saadaan avoimuudella, jota odotetaan sekä lainsäädännöltä että hallinnolta. (Setälä 2003, 133-134)

Stanfordin yliopiston oikeustieteen professori Lawrence Lessig on kiinnittänyt huomiota ohjelmistojen koodin vaikutuksiin kansalaisten elämään.. Lessig käsittelee teoksessaan Code 2.0 (2006, 138-153) myös julkisten tietojärjestelmien lähdekoodin merkitystä toiminnan avoimuudelle. Lessigin mukaan lähdekoodi tulisi olla avointa, jotta kansalaisilla olisi mahdollisuus tarkistaa sen toiminta. Kansalaisten oikeuksia ei tule rajata vain tiedon saamiseen, vaan kansalaisten kontrollin tulee ulottua myös siihen, miten julkista valtaa käyttävä viranomainen käsittelee tietoa. Ilman pääsyä ohjelmistojen lähdekoodiin on mahdotonta selvittää, miten jokin prosessi etenee tietojärjestelmissä. (Ghosh 2005, 8; Berry & Moss 2006, 26; Lessig 2006)

L Jean Camp (2006) on kiinnittänyt huomiota siihen, että avoin koodi ei itsessään vielä takaa toimintojen läpinäkyvyyttä. Avoin koodi on välttämätön, mutta se ei itsessään riitä, sillä kaikkien prosessienkin tulee olla läpinäkyviä. Erityisesti sähköisen äänestyksen ympärillä on käyty keskustelua lähdekoodin avoimuuden merkityksestä prosessien läpinäkyvyydelle (mm. Lessig 2006, 140-143; Berry & Moss 2006;Kitcat 2004).

4.1.2 Käytännölliset hyödyt

Käytännöllisistä hyödyistä yhtenä keskeisimmistä pidetään avoimen lähdekoodin mahdollistamaa *yhteensopivuuden parantumista ja riippumattomuutta ohjelmistotoimittajista*.

Ohjelmistoteollisuuden perinteinen liiketoimintalogiikka on ollut yksinkertainen. Yritys

myy asiakkailleen oikeuksia käyttää ohjelmistoa, mutta tuotteen täysi omistajuus ei siirry asiakkaalle. Tässä keskeistä on, että yritykselle jää valta päättää, mitä asiakas on oikeutettu ohjelmalla tekemään. Lisenssissä voidaan määritellä mihin tarkoitukseen tai kuka ohjelmaa saa käyttää. Käyttöoikeuksien lisäksi yritykset saattavat myydä tuotteisiin liittyviä palveluita, kuten tukea, konsultointia tai järjestelmien integrointia vastatakseen asiakkaan tarpeisiin paremmin. (Weber 2004, 191; Lessig 2006; 146, 151)

Weberin (2004, 191-192) mukaan tällainen liiketoimintalogiikka synnyttää markkinat, jossa valta on siirtynyt enemmän tietotekniikkapalveluja tuottaville yrityksille kuin asiakkaille. Asiakkaat haluavat ratkaisuja käytännöllisiin ongelmiinsa, ja he ilmaisevat tarpeitaan epätäydellisesti. Ohjelmistoyritykset toteuttavat periaatteessa ratkaisuja asiakkaiden tarpeisiin. Asiakkaiden tarpeet ovat kuitenkin hyvin moninaisia, vaihtelevia ja ainutlaatuisia. Käytännössä ohjelmistoyritykset luovat tuotteita, jotka pyrkivät tyydyttämään suurimpien asiakkaiden tarpeet. Näin syntyy tuotteita, jotka ovat riittävän hyviä palvellakseen laajoja käyttäjäjoukkoja, mutta ilman, että niitä on optimoitu kenenkään yksittäisen käyttäjän tarpeisiin. Asiakkaat joutuvat näin ongelmiin yrittäessään käyttää ja integroida näitä tuotteita siten, mikä vastaisi heidän todellisia tarpeitaan. Tuotteiden muokkaaminen itselle sopivaksi on hyvin kallista ja käytännössä usein mahdotonta ilman pääsyä lähdekoodiin.

Avoin lähdekoodi haastaa edellä kuvatun liiketoimintamallin täysin päinvastaisella lähtökohdalla. Lähdekoodi vapautetaan kontrollista, mikä synnyttää tilanteen, jossa yritykset eivät voi tehdä merkittävää voittoa sillä, että ne myisivät kopioita avoimen lähdekoodin ohjelmista. Keskeisintä kuitenkin on, että valta markkinoilla siirtyy ohjelmistojen tuottajilta asiakkaille ja ohjelmistojen käyttäjille. (Weber 2004, 192; Lessig 2006, 147-151)

Päästessään käsiksi lähdekoodiin käyttäjät voivat valita, mitä toimintoja ohjelmistoonsa haluavat, muokata ominaisuuksia vastaamaan paremmin omin tarpeitaan ja ratkaista aiempaa helpommin yhteensopivuuteen liittyviä ongelmia. Asiakkaat tulevat myös riippumattomiksi ohjelmiston valmistajasta, sillä he voivat joko itse tai jonkun kolmannen osapuolen avulla tehdä muutoksia lähdekoodin. (Weber 2004, 192-193; Krishnamurthy 2005, 288) Avoin lähdekoodi ja avoimet standardit mahdollistavat erilaisten

integrointiratkaisujen toteuttamisen ja ohjelmistojen omatoimisen kehittämisen. Vaikka käytännössä vain harvat organisaatiot lähtevät tekemään muutoksia lähdekoodiin, riippumattomuus ohjelmistotoimittajista on silti yksi tärkeimpiä syitä avoimen lähdekoodin ohjelmistojen valintaan. (Karjalainen 2010,18; Glott & Ghosh 2005, 22; Ghosh 2005, 8-9; Berry & Moss 2006, 26; Woods & Guliani 2005; West & Dedrick 2008)

Turvallisuuden parantuminen on yleinen argumentti avoimen lähdekoodin puolesta. Onkin näyttöä siitä, että ainakin merkittävimmissä avoimen lähdekoodin ohjelmistoissa, kuten Apache ja Linux-ydin, on vähemmän virheitä ja haavoittuvuuksia kuin niiden kaupallisissa vastineissa. Selitys tälle on siinä, että lähdekoodin ollessa avointa, turvallisuutta voi tarkastella laaja joukko ohjelmoijia. Eric Raymondin (1999, 41) tunnettu selitys on ”Given enough eyeballs, all bugs are shallow”.(Ghosh 2005, 8-9; Berry & Moss 2006, 26,)

Hoepman & Jacobs (2007, 79) käyttävät analogiana lukkosepän työtä. Lukkoseppä voi pitää tiedon lukkojensa toteutustavasta salaisena tai sitten hän voi antaa työnsä kenen tahansa, myös varkaiden, arvioitavaksi. Kirjoittajat kysyvät kumman lukkosepän työhön olisi perustellumpaa luottaa?

Lähdekoodin avoimuus mahdollistaa sen, että kuka tahansa voi löytää ohjelmistohaavoittuvuuksia. Näin ohjelmiston käytöstä kiinnostuneet voivat joko itse tai jonkin kolmannen osapuolen avulla tehdä arvioinnin ohjelmiston toiminnasta tai tehdä sen koodin itse parannuksia. Kun koodin virheistä tulee julkisia, ne myös korjataan nopeasti. (Hoepman & Jacobs 2007, 83; Hietanen 2001)

Avoimen lähdekoodin ohjelmistojen turvallisuuden paremmuudesta verrattuna suljetun lähdekoodin ohjelmistoihin on käyty paljon keskustelua. Jotta koodin "vertaisarvioinnin" kautta ohjelmistojen laadun parannusta voi käytännössä tapahtua, tulee ohjelmistosta kiinnostuneita ohjelmoijia olla paljon. Käytännössä useimmat avoimen lähdekoodin ohjelmistot eivät saa riittävän paljon kiinnostuneita kehittäjiä tutkimaan ja parantamaan koodia. Usein ollaankin tilanteessa, että myös avoimen lähdekoodin ohjelmistoja käytettäessä joudutaan toimimaan samalla tavalla luottamuksen varassa kuin omisteisten

ohjelmistojenkin kanssa. (mm. Hoepman & Jacobs 2007; Fugetta 2003, 83-84; Kitcat 2004 65-67; Mercuri 2005 15-19; Camp 2006; Hietanen 2001)

Kolmas keskeinen pragmaattinen argumentti avoimen lähdekoodin puolesta on sen tuomat *kustannushyödyt*. Säästöjä haetaan erityisesti ohjelmien hankintaan ja ylläpitoon. Joidenkin tutkimusten mukaan erityisesti kokonaiskustannukset (total cost of ownership) olisivat avoimen lähdekoodin ratkaisuissa merkittävästi edullisempia. Toisaalta myös vastakkaisia näkemyksiä on esitetty. Joka tapauksessa selvää on, että suljetuissa ohjelmissa kustannukset tulevat lisenssimaksuista, kun taas avoimen lähdekoodin ohjelmissa ei näitä kuluja ole. Useissa tutkimuksissa useimmin mainittu motiivi avoimen lähdekoodin käytölle on nimenomaan kustannushyödyt (Karjalainen 2010, 18; West & Dedrick 2008; Ghosh 2006;. Glott & Ghosh 2005). On kuitenkin syytä painottaa, että myös avoimen lähdekoodin kohdalla erilaisista tukipalveluista, kuten koulutuksesta, käyttöönotoista, ylläpidosta ja laitteistoista tulee edelleen kustannuksia, vaikka itse ohjelmistoista ei tarvitsekaan maksaa lisenssimaksuja. (Ghosh 2005, 8-9; Berry & Moss 2006, 25-26)

Suomessa avoimen lähdekoodin ohjelmistoja käytetään valtaosassa kunnista. Kunnista 65 prosenttia käyttää Linuxia joissain tehtävissä. Suurimmat syyt avoimen lähdekoodin käyttämiselle ovat ohjelmistojen hinta ja kokonaiskustannukset. Muita merkittäviä tekijöitä ovat tietoturva ja helppo lisenssien hallinta. (Välimäki, Oksanen & Laine 2005, 514-519)

4.2 Avoimen lähdekoodin käytön motiivit tutkimusaineistossa

Tutkimusaineistossa kuvattiin laajasti syitä, miksi artikkeleissa kuvatuissa ohjelmistoissa oli päädytty avoimen lähdekoodin käyttöön.

Artikkeleissa nostettiin esiin kenen tahansa mahdollisuus päästä käsiksi lähdekoodiin, joka tekee mahdolliseksi ymmärtää ohjelman toimintaa ja niin haluttaessa kehittää sitä eteenpäin. Koodin avoimuudella halutaan kannustaa osallistumaan ohjelmiston edelleen kehittämiseen:

> *"An open-source project has been established to promote continued development of the system." [24]*

Aineistossa esitettiinkin olettamus, että koodin avoimuus nostaisi ohjelmistojen laatua:

> *"In a sense, the open source philosophy toward software development is no different than the concept of peer review in scientific research. Namely, peer review and open sharing of information leads to more robust outcomes " [14]*

Koodin "vertaisarvioinnin" lisäksi hajautetun, usealle toimijalle jakautuvan kehitystyön etuna pidettiin myös sitä, että ohjelmistovirheiden korjaaminen ja päivitykset tulevat saataville nopeasti.

Yhtenä keskeisimmistä avoimen lähdekoodin käyttämisen motiiveista pidettiin sitä, että ohjelmiston jatkokehitys on mahdollista pitkällä tähtäimellä riippumatta toimittajasta:

> *"open-source assures stability in the service use of the products, being the user – the health-care system in the general sense – able to use it with long-term continuity and also to improve, update, modify and integrate it as much as necessary for his scopes, even in case the developer or the vendor disappears from the market." [13]*

Esimerkkinä mainittiin tilanne, jossa ohjelmiston kehitys on riippuvaista yhdestä yrityksestä tai jopa yksittäisistä ohjelmoijista. Näin ollen kehitystyö on helposti hyvin haavoittuvaista näistä avaintoimijoista. Mikäli ohjelmiston lähdekoodiin ei ole pääsyä, tulee jatkokehittämisestä mahdotonta.

Riippuvuutta yhden toimittajan omisteisista tuotteista pidettiinkin yhtenä suurimmista terveydenhuollon tietojärjestelmien ongelmista:

> *"On the one hand, most of the commercially available EPR systems are proprietary and as a consequence, incompatible with other types of complementary platforms." [20]*

Avoimen lähdekoodin käytöllä haluttiinkin parantaa erityisesti yhteensopivuutta, sillä kokemuksia huonosti yhteensopivista suljetuista järjestelmistä oli paljon:

> *"The development of EMR systems in the US over the last two decades has been dogged by problems of closed, proprietary and incompatible systems" [11]*

Dokumentaation ja lähdekoodin avoimella jakamisella on ratkaiseva merkitys yhteensopivuuden kehittämiselle. Näin ohjelmistojen suunnittelu- ja toimintaperiaatteiden ymmärtäminen sekä yhteensopivuuden toteuttaminen mahdollistuvat aiempaa paremmin.

> *"All Indivo technical documents, including design concepts and source code, are freely available and accessible on the Internet. Critical to interoperability and adoption, all design concepts, application programming interfaces and document formats are open and public." [17]*

> *"Open source technology was chosen for prototype implementation to facilitate interoperability with other complementary platforms, and to increase the flexibility of the developed application, while reducing development costs" [20]*

Yhteensopivuutta edistää myös valmiin koodin uudelleenkäyttö. Useammassakin artikkelissa pidettiin tärkeänä sitä, että valmiita komponentteja voidaan ottaa suoraan käyttöön ja "pyörää ei tarvitse keksiä uudestaan". Näin kehitystyö nopeutuu huomattavasti.

> *"Access to a means of avoiding the reinvention of existing tools and facilitating the identificatio and reuse of existing software modules are two key elements in the rapid development of applications used to process medical images." [10]*

Aineistossa esitettiin näkemys, että avoimen lähdekoodin ohjelmistot olisivat vakaampia kuin omisteiset suljetut ohjelmistot. Näin voitaisiin valmiiden avoimeen lähdekoodiin perustuvien ohjelmistokomponenttien kohdalla välttyä työläältä virheiden testaamiselta.

> *"Because software development has a significant component of trial and error, reuse of well-tested components helps reduce technical problems, especially for complex functions like order entry. " [11]*

Valmiiden komponenttien käytöllä on yhteys myös neljänteen avoimen lähdekoodin suosimisen motiiviin. Valmiit avoimen lähdekoodin komponentit ovat ilmaiseksi saatavilla ja näin laskevat kokonaiskustannuksia.

> *"In the end, the overall cost–benefit-analysis led to the decision to use MySQL as a relational DB (DD is innately provided) and a DBMS based on the internet technologies HTML and PHP scripting languages. All of these are open source solutions that are freely available, eliminating software costs." [26]*

Keskeiset avoimen lähdekoodin ohjelmistot ovat myös hyvin dokumentoituja, helposti

hallinnoitavissa ja niille on saatavilla tukea internetistä:

> *"For larger installations open source software can be cheaper and more 'future proof', also Linux is very easy to manage and support over the internet. Good documentation and easy-to-configure systems are important." [11]*

Muina taloudellisesti huomion arvoisina tekijöinä pidettiin terveydenhuollon resurssien kohdentamista varsinaiseen hoitotyöhön tietojärjestelmien sijaan sekä sitä, että avoin lähdekoodi mahdollistaa koodaustyön teettämisen paikallisilla toimijoilla. Tällä on merkittäviä työllisyysvaikutuksia.

4.3 Yhteenveto ja pohdinta

Tutkimusaineistossa esiin nousseita avoimen lähdekoodin käyttämisen motiiveja terveydenhuollon tietojärjestelmissä voidaan hyvin tarkastella Ghoshin kategorisoinnin kautta. Tutkimusaineiston argumenteissa painottuvat prgamaattiset hyödyt useammin kuin yhteiskunnalliset, vaikka myös yhteiskunnallisia hyötyjä painottavia argumenttejakin oli mukana. Artikkeleissa avointa lähdekoodia tarkasteltiin pitkälti ohjelmistotuotantotapana eikä niinkään laajempana yhteiskunnallisena ilmiönä, kuten poliittisena tai ideologisena kysymyksenä. Ghosh kuitenkin huomauttaa, että prgamaattiset hyödyt voidaan nähdä myös yhteiskunnallisina hyötyinä. Tällainen tiukka kategorisointi ei luonnollisesti ole aina selkeärajainen tai yksioikoinen. Näin aineistossa osoitetuilla pragmaattisilla hyödyillä on selkeä laajempi yhteiskunnallinen vaikutus, vaikka artikkelien kirjoittajat eivät sitä ole itse tuoneetkaan esiin artikkelia kirjoittaessaan.

Tutkimusaineistosta esiin nousseet syyt avoimen lähdekoodin käyttöön terveydenhuollon tietojärjestelmissä on tiivistetty seuraavaan taulukkoon:

Pääteema	***Avoimen lähdekoodin käytön motiivit***	
Alateema	*Alateeman ilmaisut*	*Hyöty (Yhteiskunnallinen / Pragmaattinen)*
Jatkokehittämisen mahdollistaminen ja turvaaminen	Pääsy lähdekoodiin Kannustaminen edelleen kehittämiseen Kehittäjien yhteistyö ja "vertaisarvioitu" koodi Riippumattomuus ohjelmistotoimittajista	Y, P Y, P P Y, P

Yhteensopivuus	Koodin avoimuus mahdollistaa yhteensopivuuden toteuttamisen	P
Valmiin koodin uudelleenkäyttö	"Pyörää ei tarvitse keksiä uudelleen" Nopeuttaa kehitystyötä Valmiiden komponenttien vakaus	Y, P P P
Kustannushyödyt	Valmiit komponentit saatavilla ilmaiseksi Ei lisenssikustannuksia Tukea mahdollista saada internetistä Terveydenhuollon rajallisten resurssien kohdentaminen hoitotyöhön Koodaustyö työllistää paikallisesti	Y, P P P Y Y

5 KEHITTÄJÄYHTEISÖJEN ORGANISOITUMINEN

> *"While it is not completely understood why some open source projects succeed and some fail, community building is a core component of a successful open source project." [23]*

Steven Weberin mukaan keskeinen ero perinteisen omisteisen kehittämistavan ja avointen kehittämisyhteisöjen välillä on niiden erilainen organisoituminen. Ensimmäisissä ohjelmistoja kehitetään hierarkkisissa organisaatioissa, kun taas avoimet kehittämisyhteisöt toimivat verkostomaisesti. (Weber 2004, 154; Miettinen ym. 2006b, 29)

Weberin (2004, 154) mukaan avoimet ohjelmistot ovat ei-kilpailullisia julkishyödykkeitä. Tällaisille tuotteille ovat tyypillisiä myönteiset verkostovaikutukset: mitä useampi käyttää tuotetta, sen arvokkaammaksi se tulee. Tämä johtuu ohjelmistotuotteen ilmaisen jakelun ja kopioinnin lisäksi ennen kaikkea siitä, että käyttäjät hyötyvät muiden käyttäjien huomaamista virheiden korjaamisista, parannusehdotuksista ja käyttökokemuksista. Innovaation jakaminen johtaa usein sekä innovaation luojan että yhteiseen hyötyyn. Jotta kehittäminen onnistuisi jaetusti ja kollektiivisesti, on koodin oltava vapaasti saatavilla. (von Hippel 2005a, 10, 85-89; Weber 2004, 154; Miettinen ym. 2006b, 29)

Von Hippelin mukaan innovaation jakaminen ja vapauttaminen kenen tahansa käyttöön on usein paras ja käytännöllisin vaihtoehto myös innovaation luojalle. Innovaation suojaaminen liikesalaisuutena ei todennäköisesti ole menestyksekästä kovinkaan pitkään. Monet tietävät samanlaisia asioita, ja usein innovaatioon liittyvän tiedon jakaminen on innovaation luojalle vain vähäinen menetys tai ei menetys lainkaan. (von Hippel 2005a, 10, 81-82)

Verkostomaista innovaatiotapaa voidaan tarkastella myös innovaatioiden leviämisen näkökulmasta. Rogers (2003, 394-399) on jakanut innovaatioiden leviämisen keskitettyyn ja hajautettuun malliin. Keskitetyssä mallissa innovaatiot ovat lähtöisin asiantuntijoilta, kuten jonkin yrityksen tutkimus- ja kehitysosastolta, ja niitä levitetään mahdollisille omaksujille, jotka hyväksyvät tai hylkäävät innovaation. Innovaation omaksuja nähdään tässä mallissa vain passiivisena vastaanottajana.

Hajautetussa mallissa innovaatiot leviävät horisontaalisissa verkostoissa vertaistoimijoiden kesken. Innovaatiot ovat paikallisten johtavien käyttäjien luomia, ja muut omaksujat saattavat kehitellä niitä edelleen. Keskitetty ja hajautettu malli ovat ideaalityyppejä, jotka esitellään taulukossa 5.1.

Taulukko 5.1: Innovaatioiden diffuusio

	Keskitetty innovaatioiden diffuusio	**Hajautettu innovaatioiden diffuusio**
Keskittymisen aste päätöksenteossa ja vallassa	Yleinen päätöstenteon kontrolli kansallisella hallinnolla ja teknisten alojen asiantuntijoilla	Vallan ja kontrollin laaja jakaminen diffuusiosysteemin jäsenten kesken. diffuusiota tapahtuu paljon spontaanisti ja suunnitelemattomasti
Diffuusion suunta	Ylhäältä-alas diffuusio asiantuntijoilta paikallisille innovaatioiden käyttäjille	Innnovaatioden diffuusio vertaistasolla horisontaalisissa verkostoissa
Innovaation lähteet	Muodollinen T&K, teknisteen asioiden ekspertit	Innovaatiot tulevat kokemuksen pohjalta ei-eksperteiltä, jotka ovat usein käyttäjiä
Kuka päättää, mitä innovaatioita levitetään?	Korkeat hallintoviranomaiset ja teknisten asioiden ekspertit	Paikalliset yksiköt päättävät epämuodollisen arvioinnin pohjalta, mitä ovat ne innovaatiot joiden tulisi levitä
Asiakkaiden tarpeiden tärkeys edistettäessä diffuusioita	Innovaatiokeskeinen lähestyminen: teknologia ajaa eteenpäin, painottaen innovaatioin luomia tarpeita	Ongelmalähtöinen lähestyminen: teknologia vedetään mukaan, luodaan paikallisiin tarpeisiin ja ongelmiin
Uudelleeninnovoinnin määrä	Pieni määrä paikallista sopeuttamista ja uudelleen innovointia kun innovaatio levitetään omaksujille	Paljon paikallista sopeuttamista kun innovaatio levitetään omaksujille

(Rogers 2003, 396)

5.1 Käyttäjälähtöisyys

Useissa aineiston artikkeleissa nousi esiin tyytymättömyys sellaista ohjelmistokehitystä kohtaan, jossa käyttäjien mahdollisuudet vaikuttaa lopputulokseen ovat vähäiset. Tämän vastakohtana esiin nousi avoimelle lähdekoodille tyypillinen lähtökohta, jossa käyttäjät itse ryhtyvät myös kehittäjiksi.

> *"In general, there are two types of users of terminologies: terminology developers and terminology consumers. The developers of RadLex are in the former group, and those wishing to use RadLex or to create applications enabled by RadLex are in the latter group. Terminology*

> *developers need a tool that meets the functional requirements of terminology management." [21]*

Kun käyttäjät toimivat myös kehittäjinä, heillä on mahdollisuus muokata ohjelmiston toiminnasta juuri sellainen, kun he parhaaksi näkevät. Lisensointimalleissa, jossa lähdekoodi on omisteista ja se ei ole käyttäjien saatavilla, mahdollisuudet tehdä haluttuja muutoksia ovat hyvin rajattuja tai niitä ei ole laisinkaan.

> *"A second concern was expressed by researchers who were using proprietary database implementations. These participants wanted the freedom to add [proprietary] data elements to TMA exchange documents without violating the specification. They also indicated that they required a loose data structure that could be easily re-constructed from their own databases." [4]*

Eräässä artikkelissa alleviivattiin erityisesti sitä, että ohjelmistoarkkitehtuurin suunnittelulla voidaan kannustaa siihen, että käyttäjät voivat räätälöidä ohjelmiston käyttöliittymästä omiin tarpeisiinsa sopivan:

> *"By designing the graphical user interface as a client that uses the standard API, the Indivo approach encourages a distributed process of application development that supports creation of "home grown" as well as commercial modules." [17]*

Tutkimusaineistossa esiin noussutta käyttäjäkeskeisyyttä on mielenkiintoista tarkastella innovaatiotutkija Eric von Hippelin teorioita vasten. Hän esittää teoksessaan Democratizing Innovations (2005a), että tuotteiden ja palveluiden käyttäjät, niin yritykset kuin yksittäiset kuluttajatkin, ottavat vastuuta innovoinnista ja tuotekehittelystä entistä enemmän. Von Hippelin mukaan käyttäjät, jotka innovoivat itse, voivat kehittää juuri sellaisia tuotteita kuin haluavat ja näin vähentää riippuvuutta valmistajista. Yksittäisten käyttäjien ei kuitenkaan tarvitse itse kehittää kaikkea, vaan he voivat hyötyä innovaatioista, joita toiset käyttäjät ovat kehittäneet ja asettaneet vapaasti käytettäväksi. (von Hippel 2005a, 1)

Von Hippelin (1998) uraa uurtava havainto on, että yhä useammissa tapauksissa käyttäjät itse muokkasivat ja kehittelivät käyttämiään tuotteitaan. Käyttäjien, niin yksittäisten henkilöiden kuin yritystenkin, joukossa on toimijoita, jotka pystyvät osoittamaan ja ennakoimaan sellaisia tarpeita ja ratkaisuja, joihin valtaosa käyttäjistä ja valmistajista ei osaa kiinnittää huomiota. Nämä ns. johtavat käyttäjät pystyvät ennakoimaan tulevia

markkinatrendejä. (von Hippel 2005a, 22-23)

Käyttäjäkeskeiset innovaatioprosessit ovat lähes vastakkaisia perinteisille suljetuille innovaatiomalleille, joissa valmistajat pyrkivät estämään innovaatioidensa kopiointia mm. patentein ja tekijänoikeuksin. Käyttäjän rooli suunnittelussa jää hyvin vähäiseksi. Von Hippelin mukaan empiiriset tutkimukset kuitenkin osoittavat, että käyttäjien rooli kehittäjinä on monissa teollisissa ja kulutustuotteissa hyvin merkittävä. Tuotteiden kehittämisen painopiste on siirtymässä yhä enemmän valmistajilta käyttäjille, mikä on vaikeaa monille valmistajille. Avoin, hajautettu innovaatiomalli haastaa voimakkaasti vallitsevaa käsitystä työnjaosta. Monien yritysten ja teollisuudenalojen onkin arvioitava uudelleen liiketoimintamallejaan. (von Hippel 2005a, 2)

Käyttäjien ja valmistajien tavat kehittää tuotteita ja palveluita eroavat toisistaan, sillä käyttäjät ja valmistajat tietävät yleensä eri asioita. Käyttäjillä on taipumusta kehittää innovaatioita, jotka ovat toiminnallisesti uudenlaisia. Tällaiset innovaatiot tarvitsevat runsaasti informaatiota käyttäjien tarpeista ja kontekstista. Vastaavasti valmistajien innovaatiot ovat yleensä parannuksia laajasti tunnettuihin tarpeisiin ja edellyttävät laajaa ymmärtämystä kehittämisen ratkaisuista. (von Hippel 2005a, 8)

5.2 Yhteisön organisoituminen

Von Hippelin mukaan innovaatioiden kehittäminen tehostuu merkittävästi, mikäli käyttäjät jakavat tietojaan keskenään jollakin tavalla. Näin käyttäjät eivät toisistaan tietämättä tee päällekkäistä työtä. Käyttäjien kehittämille innovaatioille on tyypillistä niiden vapauttaminen kaikkien käytettäväksi. Innovaation luoja siis luopuu tekijänoikeuksistaan ja antaa vapaaehtoisesti kenen tahansa kiinnostuneen päästä käsiksi innovaatioon liittyvään informaatioon. Innovaatiosta tulee näin julkishyödyke. (von Hippel 2005a, 9, 93-97)

Laajasta yhteistyöstä on hyötyä innovaatiota luoville kehittäjille. Heillä onkin tapana toimia erilaisissa yhteistyöverkostoissa. Innovaatioyhteisöt voivat nopeuttaa ja tehostaa kehitystyötä. Yhteisöissä käyttäjät ja valmistajat voivat kehittää, testata ja jakaa

innovaatioitaan. (von Hippel 2005a, 11. 93-97; Miettinen ym. 2006b, 42-43)

Avoimen lähdekoodin kehittäjäyhteisöjä pidetään usein tyyppiesimerkkinä käyttäjäkeskeisestä yhteiskehittelystä. (von Hippel 2005b; Weber 2004; Miettinen 2006a; Miettinen 2006b)

Tutkimusaineistoissa kuvattiin kehittäjäyhteisöjen toimintaa ja organisoitumista monipuolisesti. Artikkeleissa korostettiin muun muassa sitä, että lähdekoodin julkaisemisella haluttiin kannustaa mahdollisimman monia osallistumaan kehitystyöhön ja jakamaan tuotoksiaan avoimesti. Viime kädessä tästä hyötyvät koko yhteisön lisäksi myös yksittäiset kehittäjät.

> *"This special issue on segmentation and registration using ITK hopefully will encourage the medical image analysis community to develop and publish their software in ITK style. Not only the community as a whole would greatly profit from this, but also each individual researcher, since she/he will save a lot of time and probably derive results of a level that would not be achievable on her/his own." [29]*

Aineistossa kuvattiin myös kehittäjäyhteisön jäseniä ja sen organisoitumista. Usein kehitystyö oli lähtenyt jonkin yksittäisen toimijan tarpeista. Osassa artikkeleista yhteisö oli jo syntynyt ja sitä kuvattiin tarkemmin. Toisissa artikkeleissa sen sijaan toivottiin yhteisön syntymistä koodin avoimen julkaisemisen kautta. Kenties yksi syy artikkelin julkaisemiseen olikin nimenomaan siinä, että ohjelmistoprojektista haluttiin kertoa laajemmallekin yleisölle.

Terveydenhuollon toiminta perustuu varsinkin Euroopassa pitkälti julkiselle rahoitukselle. Siksi useissa tapauksissa kehittäjäyhteisön jäsenissäkin oli paljon julkisia toimijoita. Moni artikkeli kuvasikin korkeakouluissa käynnistettyjä projekteja.

> *"Moreover, extensive support is now provided by universities, federal agencies, and companies to the development of open tools, libraries, and software programs for biomedical research and innovation." [10]*

Mikään yhteisö ei voi toimia tehokkaasti ilman jonkinlaista organisoitumista. Siksi toiminnan ohjaamiseksi ja tavoitteiden saavuttamiseksi tarvitaan sitä, että jokin taho ottaa vastuun kokonaisuudesta. Suurimmassa osassa artikkeleista ei kuvattu tätä organisoitu-

mista mitenkään. On selvää, että pienemmissä projekteissa organisoituminen tapahtuu hyvin kevyesti, kun taas suuremmissa projekteissa vastuun jakaminen tapahtuu systemaattisemmin. Tällä pyritään siihen, että kehitystehtäviä jaetaan tarkoituksenmukaisella tavalla eri kehittäjätahojen kesken.

> *"DVT project has a steering committee with responsibility for guiding legal, technical, and commercial aspects. The steering committee meets every 6 months to discuss past progress, current issues, and future requirements. A project manager was elected by the steering committee to manage theDVT project on a daily basis and report back to the committee. Development tasks were divided up based on the available skills of developers who report to the project manager." [19]*

Pohjimmiltaan avoimen lähdekoodin kehittämistyö perustuu vapaaehtoisuuteen ja siihen, että kukin kehittäjäyhteisöön kuuluva toimija voi itse valita, mitä tekee. Tämän vuoksi suuri kysymys on, miten tällaiset löyhät verkostot voivat pysyä koossa, tuottaa laadukkaita tuotteita ja ratkoa ristiriitoja?

Miettinen ym. (2006b, 49) korostavat, että kaikkein keskeisin selittäjä avointen ohjelmistoprojektien organisoitumiselle on yhteinen kohde. Kaikkien tavoitteena on saada aikaan laadukas tuote, joka on mahdollista saada aikaan vain yhdistämällä kehittämistyöhön osallistuvien toisiaan täydentävä osallistuminen. Tämän tekee mahdolliseksi modularisointi. Toisin sanoen yhteisö helpottaa suuren kokonaisuuden kehittämistä jakamalla kehitystyön yhteen liitettäviin moduleihin. Modulien väliset yhteydet pidetään mahdollisimman vähäisinä, jolloin niiden hallitseminen on mahdollista. Modulaarisuus mahdollistaa yhden osan kehittämisen siten, että muiden modulien rakennetta ei tarvitse muuttaa. (Miettinen ym. 2006b, 49-60; von Hippel 2005a, 11. 93-97; Weber 2004, 88, 149)

Modulaarisuus luo myös perustan työnjaolle ja päätöksentekorakenteelle. Laajoissa avoimen lähdekoodin projekteissa kehittämisvastuuta on välttämätöntä jakaa useille toimijoille, ja kukaan yksittäinen henkilö ei voi olla kaikesta viime kätisessä vastuussa. (Miettinen ym. 2006b, 50; Weber 2004; 88-93) Modulaarisuuden merkitykseen tutkimusaineiston ohjelmistoprojekteissa palataan seuraavassa pääluvussa.

Linuxin kehitystyössä on alusta alkaen ollut selvää, että kehittäjät hyväksyvät Linus

Torvaldsin auktoriteetin ja viime kätisen oikeuden päättää, mitä koodia Linux-kerneliin otetaan mukaan. Käytännössä Linux on kasvanut jo niin suureksi projektiksi, että näin ei voida enää toimia, vaan koodin hyväksymisvastuuta on delegoitu Torvaldsin luottohenkilöille eli ns. sisäpiirille. Monet pienet avoimen lähdekoodin projektit voivat kuitenkin edelleen toimia tämän mallin mukaisesti. (Weber 2004; 88-93)

Linuxille tyypillisille luottohenkilöihin perustuvan mallin lisäksi, päätöksenteko voidaan organisoida myös kehällisiin ympyröihin. Pieni ydinjoukko päättää lopullisesta koodin mukaan ottamisesta. Toisella kehällä olevat henkilöt voivat muokata tehtyjä ehdotuksia ja kolmas kehä taas voi tehdä ehdotuksia uudesta koodista. Kolmas mahdollinen malli perustuu taas muodollisempaan päätöksentekoon. Esimerkiksi Apache:n kehittäjien eliitti muodostaa ydinryhmän, jossa äänestetään muutoksista. Ryhmällä on tarkat säädökset siitä, kuinka paljon hyväksyttyjä ääniä pitää olla, jotta jokin muutos hyväksytään. Mallissa myös kontrolloidaan tarkkaan, ketkä kehittäjistä pääsevät mukaan muutoksia käsittelevään ydinryhmään. (Weber 2004. 88-93)

Avoimen lähdekoodin kehittämishankkeissa on joukko erilaisia omistamista, päätöksentekoa ja ohjelmointikäytäntöjä koskevia sääntöjä. Näistä keskeisimpiä ovat avoimen lähdekoodin lisenssit, jotka muodostavat avoimen lähdekoodin keskeisimmän muodollisen sosiaalisen rakenteen. Lisenssejä voidaankin pitää eräänlaisina yhteiskuntasopimuksina, jotka luovat yhteisön identiteettiä ja jonka kaikki osallistujat hyväksyvät. Lisenssien keskeinen tehtävä on luoda sosiaalinen rakenne, joka kannustaa kehittämistyöhön varmistamalla pääsyn lähdekoodin. Lisenssit antavat käyttäjille oikeuden kopioida, levittää edelleen ja muokata koodia sekä levittää edelleen muokattua versiota ohjelmistosta. Lisenssit myös estävät käyttäjiä luomasta rajoituksia muille käyttäjille, jotka uhkaisivat projektin alkuperäisiä tavoitteita. (Miettinen ym. 2006b, 50; Weber 2004, 84-86, 161-166; Osterloh 2007, 167)

5.3 Yritysten rooli yhteisössä

Tutkimusaineistossa korostuu monessa kohtaa, että mahdollisuuksia vaikuttaa ohjelmistojen kehitykseen halutaan lisää. Voidaankin sanoa, että avoimen lähdekoodin myötä

valta innovaatioista ja ohjelmistojen kehittämisestä siirtyy ohjelmistoyrityksiltä niiden asiakkaille. Tämä haastaa ohjelmistotuotannon ansaintalogiikan, sillä suljetun toimintamallin perustana on ollut innovaatioiden ja oman henkisen omaisuuden tiukka kontrollointi. Käytännössä tämä on kulminoitunut yritysten hyvin itsenäisenä toimintana ja ohjelmistojen lähdekoodin pitämisenä suurimpana liikesalaisuutena.

Henry Chesbroughin mukaan yritykset ovat pakotettuja muuttamaan innovaatioperiaatteitaan. Hänen mukaansa on tapahtumassa innovaatioparadigman muutos. Suljetun innovaatioparadigman murtumisen aiheuttavat neljä tekijää. Ensimmäinen tekijä on kokeneen ja korkeasti osaavan työvoiman liikkuvuus. Siirtyessään yrityksestä toiseen työntekijät vievät mukanaan merkittävästi osaamista. Toinen tekijä on yksityisen pääomarahoituksen merkittävä lisääntyminen. Pääomarahoittajat tukevat pieniä kasvuyrityksiä, ja haluavat luoda näin uusia haastajia vanhoille markkinoita hallitseville yrityksille, jotka ovat aiemmin rahoittaneet alan tutkimus- ja kehitystoiminnan lähes kokonaan. (Chesbrough 2006, 34-41)

Kolmas suljetun innovaatioparadigman murtumisen aiheuttava tekijä on tuotteiden ja teknologian elinkaaren nopeutuminen. Yritysten on pystyttävä nopeuttamaan tapaa, jolla ne käsittelevät uutta tietoa. (Chesbrough 2006, 38-39) Neljäs tekijä on ulkopuolisten toimittajien lisääntynyt potentiaali. Aiemmin mainituista kolmesta tekijästä johtuen sekä koulutustason kohoamisen myötä kaiken kokoisilla yrityksillä on mahdollista saada koulutettua työvoimaa. Näin ne pystyvät toimittamaan yhtä hyvää tai jopa parempaa laatua kuin mitä yritys olisi pystynyt tuottamaan sisäisesti. (Chesbrough 2006, 39-40)

Edellä esitetyistä tekijöistä johtuen yritykset eivät voi enää estää uusien ideoiden leviämistä yrityksen ulkopuolelle. Samanaikaisesti yrityksille on tullut mahdolliseksi hyödyntää tutkimustietoutta useista eri ulkopuolisista lähteistä. Tätä yrityksen ulkopuolelta tulevaa tutkimusta voidaan tuoda yritykseen ja edelleen kehittää uusiksi tuotteiksi ja palveluiksi. (Chesbrough 2006, 40)

Chesbroughin mukaan avoin innovaatio kääntää monet suljetun innovaation perustavanlaatuiset oletukset ylösalaisin. Tässä uudessa innovaatioparadigmassa yritykset käyttävät kehitystyön lähteinä sisäisten innovaatioiden ohella myös ulkoisia innovaatioita.

Avoimessa innovaatiossa yhdistetään sisäisiä innovaatiota ulkopuolelta tuleviin innovaatioihin. Näin syntyy kaupallisia tuotteita tai palveluita. Avoimen ja suljetun innovaatioparadigman keskeiset peruserot on koottu taulukkoon 5.2.

Taulukko 5.2: Suljetun ja avoimen innovaation periaatteet

Suljetun innovaation periaatteet	**Avoimen innovaation periaatteet**
Alan huiput työskentelevät meillä	Kaikki alan huiput eivät työskentele meidän yrityksessämme. Meidän on työskenneltävä niin yrityksen sisältä kuin ulkoa olevien osaavien ihmisten kanssa
Hyötyäksemme T&K:stä meidän on keksittävä, kehitettävä ja siirrettävä innovaatiot itse.	Yrityksen ulkopuolinen T&K pystyy luomaan huomattavaa arvoa: sisäistä T&K:tä tarvitaan ottamaan itselle osuus tuosta arvosta.
Jos keksimme jotain, saamme sen markkinoille ensimmäisenä.	Meidän ei tarvitse olla tutkimuksen aloittaja hyötyäksemme siitä.
Se yritys, joka on markkinoilla ensimmäisenä, voittaa.	Paremman liiketoimintamallin rakentaminen on tärkeämpää kuin olla ensimmäisenä markkinoilla.
Voitamme kilpailun, jos luomme eniten alan parhaita ideoita.	Voitamme kilpailun, jos luomme parhaan mahdollisen yhdistelmän sisäisten ja ulkoisten ideoiden käytössä.
Meidän pitäisi kontrolloida henkistä omaisuuttamme (intellectual property), jotta kilpailijat eivät hyötyisi ideoistamme.	Meidän pitäisi luoda voittoa sillä, että muut käyttävät meidän henkistä omaisuuttamme ja meidän pitäisi ostaa sitä toisilta aina, kun se edistää liiketoimintaamme.

(Chesbrough 2006; Torkkeli ym. 2007, 28)

Weberin (2004, 193-194) mukaan tässä tilanteessa ohjelmistoyritysten ansaintamahdollisuudet voivat löytyä kahdesta suunnasta. Ensinnäkin liikevoitto voidaan saada keskittymällä suojaamaan juridisin rajoituksin sitä, mitä jää jäljelle, kun lähdekoodi on vapautettu. Esimerkiksi brändit ja tuotemerkit ovat merkittäviä tekijöitä monimutkaisilla markkinoilla, jossa asiakkaan on vaikea arvioida tuotteen tai palvelun laatua.

Toinen mahdollisuus on osaamisessa, joka mahdollistaa lähdekoodin muuttamisen käytännön sovellukseksi. Kilpailuvaltti syntyy lähdekoodin hyvästä tuntemisesta, sen muokkaamisesta sopivaksi useisiin eri tilanteisiin ja kyvystä ratkoa siihen liittyvä ongelmia. (mt., 194; Krishnamurthy 2005, 280)

Tämän kahtiajaon pohjalta Weber (2004, 195-204) erittelee avoimen lähdekoodin liiketoimintamallit kuuteen ideaalityyppiin. Ne on esitelty taulukossa 5.3.

Taulukko 5.3: Avoimen lähdekoodin liiketoimintamallit

Liiketoimintamalli	**Ansaintalogiikka**	**Esimerkki**
Support sellers	Myydään avoimen lähdekoodin tuotteita ja tarjotaan niihin tuki- ja räätälöintipalveluita.	Red Hat, Canonical, Novell, MySQL
Loss leaders	Avoimen ohjelmiston kautta luodaan kysyntää ja laajempia markkinoita kaupallisille tuotteille. Avoin tuote voi tuoda yritykselle mainetta tai tehdä kaupallisesta tuotteesta paremman.	Apple
Sell it, free it	Yritys myy ohjelmistoaan ensin kaupallisena tuotteena ja sopivaksi katsomassaan vaiheessa vapauttaa ohjelman koodin. Koodin avaaminen tehdään vaiheessa, jossa uskotaan avoimen kehittämisen uskotaan olevan kannattavampaa kuin ohjelmiston omisteisuuden kautta saatujen tulojen.	Netscape, SUN
Accessorizing	Yritys myy fyysisiä tuotteita, jotka auttavat avoimen lähdekoodin ohjelmistojen käyttämisessä. Tuotteena voi olla esimerkiksi ohjekirjat tai muu ohjelmistoon liittyvä dokumentaatio.	O'Reilly & Associates
Service Enabler	Yritys jakaa ja tukee avoimen lähdekoodin ohjelmistoja luodakseen kysyntää voittoa tuoville palveluille. Mikäli ohjelmisto menestyy hyvin se luo yrityksen muille tuotteille ja palveluille, jotka mahdollistavat avoimen ohjelmiston käytön.	Hewlett-Packard, IBM
Branding	Yritys myy tuotteita oman brändinsa maineen kautta. Vaikka se ei omista itse tuotetta eli avoimen lähdekoodin ohjelmaa, ovat asiakkaat valmiita maksamaan brändin takaamasta laadusta.	Red Hat, Novell

(muokattu Weber 2004 ja Krishnamurthy 2005 pohjalta)

Tutkimusaineistossa yritysten roolia osana kehittäjäyhteisöjä kuvattiin monin tavoin. Useissa projekteissa yrityksillä oli keskeinen rooli joko projektin aloittajana, ohjelmistoon liittyvien palveluiden tuottajana tai työpanoksen antajana projektin käyttöön.

> *"Originally developed by Agfa and Philips to test their products, DVTk has grown into a professionally managed, open-source, vendor neutral, DICOM Validation Framework." [19]*

Yritystoiminnalle nähtiin monenlaisia mahdollisuuksia kehittäjäyhteisöissä. Vaikka ohjelmistojen lähdekoodi onkin avointa ja näin ilmaiseksi saatavilla, tarvitaan käyttäjäorganisaatioissa tukea monissa eri tilanteissa. Näin yrityksen tarjoamat palvelut erityisesti käyttöönotoissa, ohjelmistojen kustomoinnissa ja integraatioissa ovat edelleen hyvin tärkeitä.

> *"Since this “service” work is growing a lot, O3 Enterprise, a spin-off from the University of Trieste, is now being constituted, to offer services of implementation, management, customization and integration to the healthcare enterprises in Italy and abroad. Other spin-off companies with*

> *Universities abroad (e.g. with the University of Maribor in Slovenia) are also under study." [13]*

Parhaimmillaan avoimen lähdekoodin projektit synnyttävät yritystoimintaa, josta hyötyy myös koko kehittäjäyhteisö.

> *"service-oriented commercial entities that are early adopters of openSourcePACS may become important in providing consulting and operational services, while still contributing back to the open source community as a whole." [9]*

Aineiston artikkeleissa esiintyy Weberin esittämistä liiiketoimintamalleista *support sellers, loss leaders ja service enabler.*

5.4 Käyttäjien kuuleminen ja osallistava suunnittelu

Monissa artikkeleissa kuvattiin käyttäjien kuulemisen merkitystä. Käytännössä suuri osa avoimen lähdekoodin hankkeista on nykyisin sellaisia, että käyttäjät eivät itse osaa ohjelmoida. Näin usein esitetty ideaalitilanne siitä, että käyttäjät itse kehittävät ohjelmistoja, ei voi käytännössä toteutua. Tässä tilanteessa käyttäjien rooli voi löytyä ideoijina ja palautteen antajina. Ideoija-käyttäjät osallistuvat ohjelmistojen suunnitteluun kertomalla omista henkilökohtaisista tai organisaationsa käyttötarpeista tietotekniikan asiantuntijoiden tehdessä varsinaisen ohjelmointityön. (Miettinen ym. 2006b, 76)

Aineistoissa nähtiin käyttäjien merkitys ohjelmiston kehittämiselle hyvin tärkeänä:

> *"Specifications are only adopted when they fill the needs of a heterogeneous user community" [4]*

Käyttäjien kuulemisessa tärkeinä välineinä nähtiin määrittelyvaiheessa mm. avoimet workshopit. Ohjelmistojen edelleen kehittämistä palautteen pohjalta painotettiin monessa artikkelissa. Avoimen rakenteen vuoksi osien uudelleen kirjoittaminen on mahdollista helpohkosti.

> *"In late 2001, after the reliability and functionality of the prototype application had been proven, it was decided to create a new application on the basis of what had been learned and what had been offered as suggestions and criticisms from our user base. The development team was*

> *expanded into an international group of programmers with varying specialties, and SQL Clinic was written from the ground up as a modern Web-based application that was built on the same open-source tools as its predecessor." [12]*

Linuxin kehitystä on pidetty avoimen kehittämismallin ja hajautetun kehitystyön ideaalimallina, mutta on kuitenkin kyseenalaista, onko tällainen kehitysmalli siirrettävissä sellaisenaan loppukäyttäjille suunniteltujen ohjelmistojen suunnitteluun. Avoimen lähdekoodin hankkeissa tarvittaisiinkin enemmän toimintatapoja, jotka tukisivat loppukäyttäjien parempaa huomioimista. Yhtenä mahdollisuutena on esitetty osallistavan suunnittelun periaatteiden tuomista osaksi avoimen lähdekoodin ohjelmistokehitystä. (Miettinen ym. 2006b, 77-78)

Osallistavan suunnittelun keskeisiä periaatteina pidetään sitä, että työntekijöillä täytyy olla pääsy relevanttiin informaatioon. Heillä tulee olla itsenäinen asema suhteessa kohdattaviin ongelmiin ja mahdollisuus osallistua päätöksentekoon. Tarjolla täytyy olla sopivia osallistavan suunnittelun menetelmiä ja työntekijöillä tulee olla mahdollisuus uusiin teknologisiin ja organisatorisiin ratkaisuihin. (mt., 74-75) Terveydenhuollon ohjelmistoprojekteissa osallistavan suunnittelunmenetelmiä onkin jo käytetty jonkin verran (esim. Häkkinen & Korpela 2007; Hyysalo 2006)

5.5 Yhteisön kommunikointivälineet

Kehittäjäyhteisöjen organisoitumisessa keskeistä on löytää kommunikaatiotavat, jotka mahdollistavat maantieteellisesti hajautetun yhteisön kommunikoinnin. Aineistoissa yleisimmin mainittuja välineitä olivat:

1. nettisivut
2. sähköpostilistat
3. keskustelufoorumit
4. wikit
5. lähdekoodin hallintajärjestelmä
6. dokumentaatio

Nämä välineet mahdollistavat yhteisön toiminnan ja madaltavat kynnystä kenen tahansa osallistumisen kehitystyöhön.

> *"The dvtk.org Web site is now the location for the latest downloads, defect tracking details, and forums on using DVTk. Interested individuals and companies may join the project through the Web site. A weekly project telephone conference coordinates the activities of the development team. " [19]*

Erityisesti käyttöönotoissa yhteisön tuki on usein hyvin tärkeää, ja tässä internet-pohjaiset kommunikaatiovälineet ovat keskeisessä osassa:

> *"The implementers list server has become a very popular site for discussion of implementation issues and general topics of particular interest to implementers, at times surpassing traffic on the developers list. Issues and questions pertaining specifically to OpenMRS implementers are posted to the implementers mailing list and/or forum and answered by implementers. If unresolved, issues are often answered by developers who are registered on the implementers list server. In general, the list servers have been fairly effective at dividing traffic into developer and implementer groups." [23]*

Kehittäjäyhteisöt voivat olla hyvinkin suuria ja aktiivisia. Tämän vuoksi toimivat työkalut ohjelmiston kehitystyöhön ovat erityisen tärkeitä.

> *"The community of registered Protégé users exceeds 70,000. In addition, the large and active Protégé user community is highly engaged in Protégé code development, regularly contributing enhancements to the software (http://protege.stanford.edu/community/wiki.html), as well as participating in online discussion groups devoted to modeling questions, technical-support issues, and requests for new features (http://protege.stanford.edu/community/archives.html)" [21]*

Aiemman tutkimuksen pohjalta tiedetään, että ilman uudenlaisia internet-pohjaisia välineitä ei nykyisenlaista avoimen lähdekoodin kehittämismallia olisi voinut syntyä. Internet on mahdollistanut lähdekoodin tuomisen kenen tahansa ulottuville sekä muutosten ja parannusten tekemisen siihen. Lisäksi se on luonut edellytykset käydä keskustelua hajautetusti toimivien kehittäjien kesken. Internet on se perusteknologia, joka mahdollistaa avoimen lähdekoodin yhteisöjen toiminnan. (Weber 2004, 83-84) Tässä tutkielmassa internet-pohjaisia välineitä käytettiinkin avoimen lähdekoodin ohjelmistoprojekteille hyvin tyypilliseen tapaan. (vrt. Fogel 2005)

5.6 Yhteenveto ja pohdinta

Aineiston analyysissa nousi esiin tarve saada sellaisia ohjelmistoja, jotka vastaavat käyttäjien tarpeeseen. Moni aineiston ohjelmistoprojekteista oli syntynyt nimenomaan tiettyyn tarpeeseen, jota saatavilla olevat ohjelmistotuotteet eivät pystyneet täyttämään. Avoimen lähdekoodin lisensointimallit antavat mahdollisuuksia muokata ja kehittää ohjelmistojen toiminallisuutta rajattomasti, mutta se ei yksinään riitä, vaan tarvitaan menetelmiä, joilla loppukäyttäjien näkökulma tulee huomioiduksi riittävästi. Avoimelle lähdekoodin alkuperäisille juurille tyypillinen käyttäjä-kehittäjiin perustuva toimintatapa ei ollut yksiselitteinen toimintatapa aineistossa. Terveydenhuollon ammattilaiset ovat ohjelmistojen loppukäyttäjiä, ja aineistossa heidän rooliaan käsiteltiin ennemminkin osallistavan suunnittelun mukaisina ideoija-käyttäjinä. Koska terveydenhuollon ammattilaisten joukossa on hyvin vähän ohjelmointitaitoisia henkilöitä, voisi osallistavan suunnittelun periaatteista ja menetelmistä syntyä toimiva linkki käyttäjien ja kehittäjien välille. Tällä olisi mahdollista pienentää ohjelmiston tekijöiden ja käyttäjien välistä kuilua. Terveydenhuollon organisaatiot ovat myös perinteisesti varsin hierarkisia (Itkonen 2005, 349, Miettinen M. 2005, 268-269), joten käyttäjäkeskeisessä ohjelmistokehitystyössä olisi helpompi ottaa ensimmäisiä askeleita ideoija-käyttäjä -typpisesti sen sijaan, että tavoiteltaisiin suoraan käyttäjä-kehittijen kautta tapahtuvaa osallistumista.

Tutkimusaineiston artikkeleissa kuvatut ohjelmistoprojektit olivat yhteisöiltään hyvin eri kokoisia. Siksi myös niiden tapa organisoida yhteisön toimintaa vaihteli paljon. Monissa artikkeleissa kuitenkin korostui julkisten toimijoiden panostus ohjelmistoprojektien alkuunpanijoina. Avoimen lähdekoodin ohjelmistojen synty tapahtuu usein korkeakouluissa, ja se näkyi tässäkin aineistossa. Suomalaisessa kontekstissa terveydenhuollon tietotekniikan kehittäminen voisi kuulua enemmänkin yliopistosairaaloiden tehtäviin. Tähän sopisi hyvin tieteellisestä toiminnasta tuttu avoin toimintamalli ja verkostoituminen. Monet terveydenhuollon tietotekniikan haasteista ovat hyvin samanlaisia ympäri maailmaa, joten liittyminen jo olemassa oleviin kehittäjäyhteisöihin voisi olla hedelmällistä suomalaisille terveydenhuollon toimijoille.

Tutkimusaineistossa yritysten panos nähtiin tärkeänä osana kehittäjäyhteisön toimintaa. Aineistossa yritysten paikka osana kehittäjäyhteisöä nähtiin hyvin samanlaisena kuin

mitä kirjallisuudessa on esitetty avoimen lähdekoodin liiketoimintalogiikasta. Ohjelmistoyritysten tehtävänä on toimia avoimen lähdekoodin ohjelmistoihin liittyvinä asiantuntijoina. Liikevoitto syntyy erilaisten tukipalveluiden myynnistä. Sellaisessa organisoitumismallissa, jossa kehittäjä-käyttäjien sijaan ideoija-käyttäjillä on suuri rooli, ohjelmistoyritysten tehtäväksi jäisi edelleen toteuttaa käytännön ohjelmointityö. Tässä tilanteessa kuitenkin ostaja maksaisi käyttöoikeuksien sijaan ohjelmointityöstä, mikäli lähdekoodi julkaistaisiin avoimella lisenssillä.

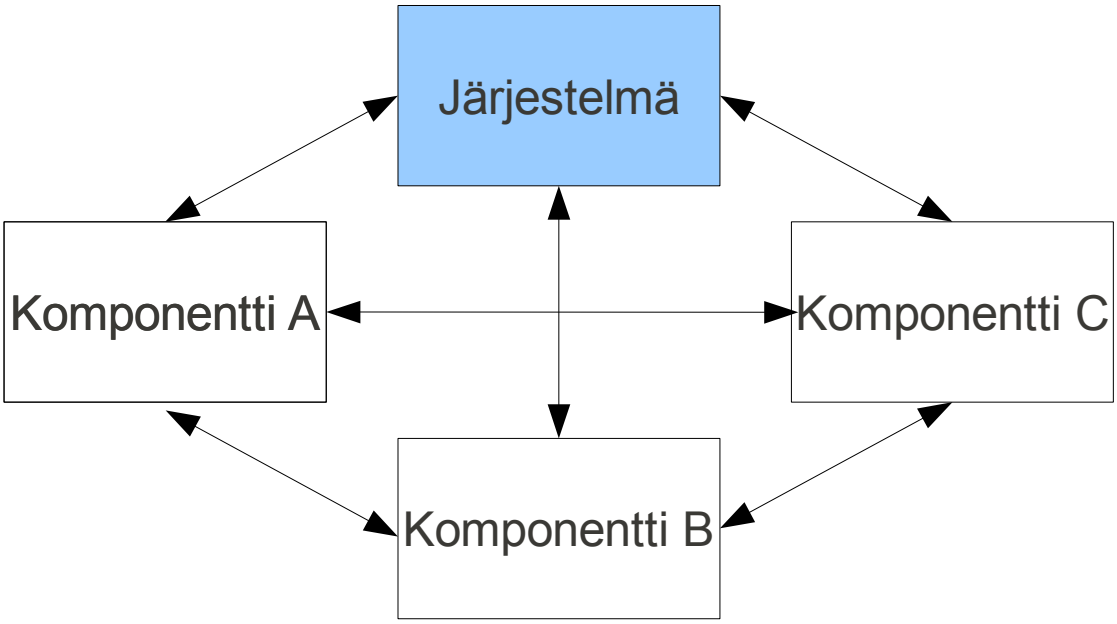
Tutkimusaineiston analyysin pohjalta esiin nousi viisi keskeistä ilmiötä, alateemaa, jotka kuvaavat tutkimusaineiston kehittäjäyhteisöjen tyypillisiä piirteitä. Nämä alateemat on esitelty seuraavassa taulukossa:

Pääteema	***Kehittäjäyhteisön organisoituminen***
Alateema	*Ilmaisut*
Käyttäjälähtöisyys	Ohjelmiston tekeminen omaan tarpeeseen Muokkaaminen oman näköiseksi Käyttäjä-kehittäjät
Organisoitumismallit	Kehittäjäyhteisöt Kannustaminen osallistumaan kehitystyöhön Julkisen rahoituksen ja korkeakoulujen keskeinen merkitys Organisoitumisen muodollisuus vaihtelee
Yritysten liiketoimintamallit	support sellers loss leaders service enablers
Käyttäjien kuuleminen	osallistavan suunnittelun mahdollisuudet
Organisoitumisvälineet	nettisivut sähköpostilistat keskustelufoorumit wikit lähdekoodin hallintajärjestelmä dokumentaatio

6 MENETELMÄT JA TEKNIIKAT AVOIMESSA KEHITYSTYÖSSÄ

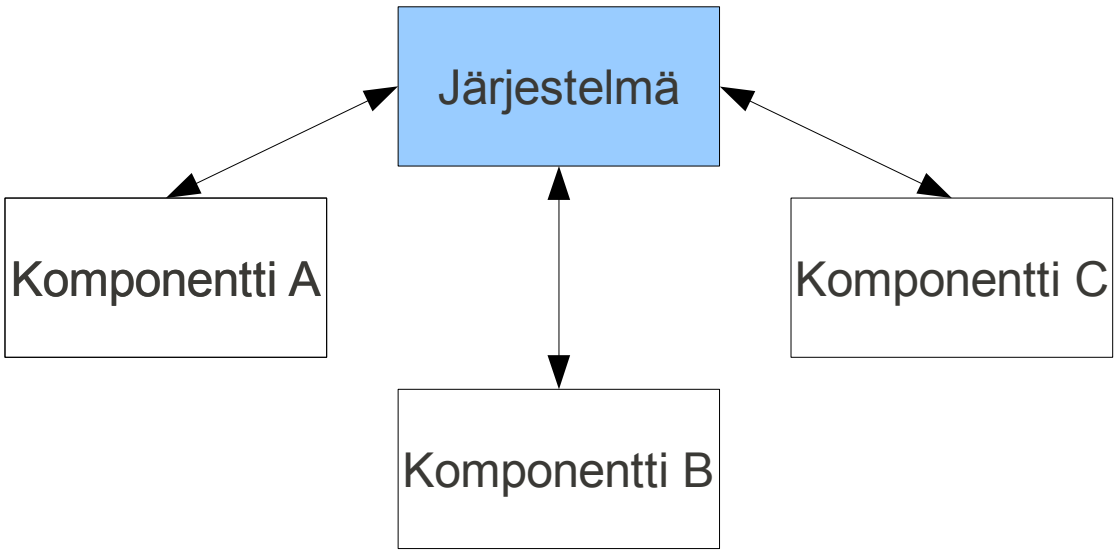
6.1 Modulaarisuus

Chessbrough kuvaa modulaarisuuden merkitystä kahdella arkkitehtuurikuvalla. Järjestelmä muodostuu komponenteista A, B ja C. Ensimmäisessä kuvassa jokaisen komponentin välillä on keskinäisiä yhteyksiä. Kun yhteen komponenttiin tehdään muutoksia, sillä on vaikutuksia koko järjestelmään. Käytännössä todellisissa järjestelmissä, joissa osia voi olla tuhansia, erilaisten yhteyksien määrä voi olla miljoonia. Tällaisen arkkitehtuurin hallitseminen on erittäin haastavaa. (Chesbrough 2006, 58-60)



Piirros 1: Arkkitehtuuri jossa paljon riippuvuuksia

Toisessa kuvassa kuvataan modulaarista arkkitehtuuria. Siinä lähtökohtana on pyrkiä ymmärtämään komponenttien välistä riippuvuutta mahdollisimman hyvin. Modulaarisessa arkkitehtuurissa muutos yhteen komponenttiin ei aiheuta muutoksia muissa komponenteissa, sillä komponenttien väliset riippuvuudet on minimoitu. Tällainen arkkitehtuuri mahdollistaa sen, että kehittäjät voivat tehdä muokkauksia yhteen komponenttiin ilman, että heidän tarvitsee huolehtia vaikutuksista muihin komponentteihin. Tämän seurauksena järjestelmän kehittäminen voidaan jakaa useille tahoille. Esimerkiksi yritysten välille voi syntyä kilpailua siitä, kuka toteuttaa parhaan komponentin A. (Chesbrough 2006, 61)



Piirros 2: Modulaarinen arkkitehtuuri

Modulaarisuutta pidetään avoimen lähdekoodin ohjelmistoprojektien työnjaon perustana. Ohjelmiston modularisointi suhteelliseen riippumattomiin osiin tekee kokonaisuudesta hallittavan. Kaikkein tärkeintä on, että moduulien tuotokset ovat selkeitä ja kommunikointi muiden moduulien kanssa tapahtuu standardoitujen rajapintojen kautta. Moduulien kehittäminen voi tapahtua hyvinkin itsenäisesti, kunhan kommunikointirajapinta toteutetaan oikein. (Miettinen ym. 2006b, 49; Weber 2004, 173)

Modulaarinen malli onkin itse asiassa edellytys hajautetulle kehittämiselle. Järjestelmää, joka rakentuu tiukasti toisistaan riippuvaisista osista, on käytännössä mahdotonta kehittää löyhässä hajautetussa kehittäjäyhteisössä. Modulaarisen mallin keskeinen etu on siinä, että kehittäjät pystyvät testaamaan omaa moduuliaan aiheuttamatta ongelmia toisten moduulien kanssa työskenteleville ohjelmoijille tai tarvitsematta ennakoida muiden tekemiä innovaatioita. (Weber 2004, 173)

Modulaarisuuden edut näkyvät erityisesti suurissa ohjelmistoissa. Yhtenä Linus Torvaldsin merkittävimmistä päätöksistä pidetään Linux-kernelin rakenteen uudelleen organisoimista itsenäisiin moduuleihin. Alunperin kernel oli suunniteltu rakenteeltaan monoliittiseksi, muuta sen saadessa yhä enemmän ja enemmän kiinnostuneita käyttäjiä ja kehittäjiä oli myös sen rakenteen muututtava. Kernelin rakenteen muuttaminen modulaariseksi mahdollisti kehitystyön jakamisen aivan toisella tavalla kuin aiemmassa rakenteessa. (mt., 173)

Aineistossa modulaarisuuden kuvaus oli hyvin yleistä ja sitä perusteltiin monin tavoin. Keskeisin peruste modulaarisuudelle oli ennen kaikkea sen tuomat mahdollisuudet muokata ohjelmiston toiminnallisuutta tarpeen mukaan.

> *"Its key features are modularity and configurability: it is possible to manage the application in order to let it fit to every kind of environment; it is also possible to configure every module, to enable/disable some particular functionality or to enhance the authentication procedure." [13]*

Muokattavuus taas mahdollistaa sen, että ohjelmistoa voidaan käyttää hyvinkin erilaisissa ympäristöissä.

> *"As a matter of fact, O3-DPACS scalability and modularity were thought also to make DPACS easily usable in the territorial environment, and basically to make it adaptable from the needs of a big hospital to those of a small clinic as well as from secure intra-net intra-hospital use to secure access from patient's home or physician's ambulatory." [13]*

Modulaarinen kehittäminen toteutuu hyvin myös sellaisissa ohjelmistoissa, joissa on varta vasten kehitetty rakenne, joka mahdollistaa helpon tavan tehdä siihen lisäominaisuuksia lisäosien avulla.

> *"In addition, Protégé has a plug-in architecture, permitting developers to extend Protégé's core functionality in many ways without needing to modify the Protégé source code. There are more than 90 Protégé plug-ins providing advanced capabilities such as import/export, validation, and visualization of large ontologies" [37]*

Aineistossa oli myös artikkeleita, jossa ko. ohjelmistoprojekti muokattiin lisäosaksi laajempaan järjestelmään.

> *"In our efforts to make these systems available, we are implementing the core functionality of e-Chasqui as a module in the OpenMRS system." [7]*

Modulaarisuus nähtiin myös keinona hallita arkkitehtuuria niin ohjelmiston tasolla kuin laajemminkin. Yksi esitetty tapa oli kerrosarkkitehtuuri, jossa jokainen kerros pyritään pitämään mahdollisimman erossa toisista.

> *"The middle tier consists of three parts: the presentation layer, the kernel and the middleware. We have tried to separate them as much as possible." [27]*

6.2 Standardit

Standardilla tarkoitetaan dokumenttia, jossa on määritelty yleistä ja toistuvaa käyttöä varten sääntöjä, ohjeita tai piirteitä tuotteille, prosesseille ja palveluille (Mykkänen, Korhonen, Porrasmaa, Tuomainen & Ensio 2005, 4; PMI 2000). Standardit ovat tärkeitä, sillä ne varmistavat tuotteiden ja palveluiden yhteensopivuuden ja laadukkuuden. Tekniset standardit vähentävät tuotteiden ja palveluiden kaupallisesti merkityksettömiä eroja ja mahdollistavat eri toimittajien tuotteiden, palveluiden, laitteistojen ja ohjelmistojen yhteensopivuuden. (Mykkänen ym. 2005, 4; Hoe 2006, 1)

Terveydenhuollossa standardoinnin lähtökohtana on ollut visio avoimemmasta järjestelmäarkkitehtuurista, johon on helppo kehittää uusia tuotteita ja palveluja. Standardien avulla pyritään saavuttamaan yhteensopivia järjestelmiä ja tietojen siirtymistä niiden välillä. Tavoitteena on resurssien parempi käyttö sekä mahdollisuus ottaa modulaarisesti käyttöön uusia palveluita. Standardit helpottavat järjestelmien liittämistä ja integrointia ja näin pienentävät kustannuksia. (Mykkänen ym. 2005, 5)

Standardit voidaan ryhmitellä karkeasti neljään luokkaan niiden avoimuuden pohjalta:

- *Omisteiset standardit* ovat usein suojattuja joko patentein tai tekijänoikeuden pohjalta.
    - Omisteiset standardit voivat olla *suljettuja*, jolloin niiden rakenne pidetään salaisuutena ja niiden käyttäminen edellyttää lisensointisopimusta standardin omistajan kanssa.
    - Omisteiset standardit voivat olla myös *julkisia*. Tällaiset standardit ovat usein jonkin yrityksen kehittämiä, mutta määrittelyt ovat julkisesti saatavilla. Standardien käyttö voi olla kuitenkin kiellettyä tai ne edellyttävät lisensointia, joko korvausta vastaan tai korvauksetta. Monet ns. laajassa käytössä olevat ns. de facto-standartit ovat omisteisia
- *Yksimieliset julkiset standardit* ovat jonkin suljetun tai kontrolloidun ryhmittymän määrittelemiä. Ryhmittymä kehittää standardia palautteen pohjalta.

- *Yksimieliset avoimet standardit* on määritelty avoimessa ryhmittymässä tai ryhmässä, johon kuuluu jäseniä esimerkiksi yrityksistä, yliopistoista ja tutkimuslaitoksista.
- *De-jure avoimet standardien* määrittelyt on tehty virallisissa kansallisissa tai kansainvälisissä standardointi organisaatioissa
  (Cerri & Fugetta 2007, 4)

Eri toimijoiden ja standardointiorganisaatioiden kesken on erilaisia määritelmiä siitä, miten avoimet standardit määritellään. Useimmissa avoimen standardin määritelmissä painotetaan avointa kehittämisprosessia, helppoa saatavuutta ja riippumattomuutta. (Hoe 2006, 1-4; Cerri & Fugetta 2007, 4-5; Berkman 2005, 6) Cerri & Fugetta listaavat viisi vaatimusta, jotka tulee täyttyä, jotta standardia voidaan pitää avoimena:

- standardin määrittelydokumentaatio tulee olla saatavilla julkisesti, joko ilmaiseksi tai muodollista korvausta vastaan
- standardin tulee olla virallisen standardointiorganisaation tai avoimen ryhmittymän omistuksessa ja hallinnassa: mikään yksittäinen osapuoli ei saa sitä omistaa tai kontrolloida eikä millään yksittäisellä osapuolella ole siihen erityisoikeuksia
- standardin määrittely- ja kehitystyö tulee tehdä avoimessa prosessissa: kaikilla kiinnostuneilla tulee olla mahdollisuus osallistua standardointiprosessiin, jossa päätöksenteko tapahtuu avoimesti ja konsensushakuisesti
- standardi tulee olla vapaasti kaikkien kiinnostuneiden käyttöönotettavissa ilman maksuja. Mahdolliset patentoidut teknologiat tulee lisensoida ei-syrjivästi ja maksuttomin ehdoin.
- standardin laajentamisen ja uudelleenkäytön tulee olla mahdollista muissa avoimissa standardeissa.
  (Cerri & Fugetta 2007, 4-5)

Avoimet standardit ja avoin lähdekoodi sotketaan usein keskenään. Ne eivät kuitenkaan tarkoita samaa asiaa eivätkä välttämättä edellytä toisiaan. Vaikka avoimen lähdekoodin ohjelmistotuotantotapaan kuuluu tiiviisti avoimien standardien noudattaminen, on olemassa avoimeen lähdekoodin perustuvia ohjelmia, jotka eivät noudata avoimia standar-

deja. Toisaalta myös omisteiset ohjelmat toimivat yhä useammin avointen standardien pohjalta. (Berkman 2005, 6; Cerri & Fugetta 2007, 5)

Avointen standardien ja avoimen lähdekoodin välisiä eroja ja yhtäläisyyksiä on kuvattu seuraavasta taulukossa:

Taulukko 6.1: Avointen standardien ja avoimen lähdekoodin erot

	Avoin standardi	Avoin lähdekoodi
Luonne	Kokoelma määrityksiä	Lähdekoodi
Rajapintojen avoimuus	Määritelmien kautta (by definition)	Suunnittelun kautta (by design)
Yhteensopivuus	Mahdollistettu	Ei voida olettaa
Lisensointi	Erilaisia tapoja	Erilaisia tapoja
Kehittämistapa	Yhteisöllinen	Yhteisöllinen

(Berkman 2005, 7)

Aineistossa standardoinnista oli jonkin verran mainintoja. Ennen kaikkea sen nähtiin mahdollistavan tiedon siirto järjestelmien ja organisaatioiden välillä.

> *"The advantage of using this tandardized format is that reports from a variety of sources can be transformed into this common format and then a single scrubber (and other tools such as an automated UMLS coder) can easily process reports from many different source institutions." [1]*

Aineistossa mainittiin usein terveydenhuollon perusstandardit, mikä kuvaa hyvin sitä, että avoimen lähdekoodin ohjelmistoissa usein sitoudutaan hyvin jo olemassa oleviin standardeihin (Weber 2004, 238).

> *"OpenMRS implements several standards to support its functioning as a client server and distributed application and to improve interoperability with other systems. Data and forms are represented in the Extensible Markup Language (XML) and the XForms (XML Forms) standards, respectively and data is communicated between the FormEntry module and the application layer packaged as Health Level Seven (HL7; www.hl7.org) messages. The RDBMS stores data according to an open relational model and can be manipulated and queried using standard Structured Query Language (SQL). [23]*

Yhdessä aineiston artikkelissa tiivistettiin hyvin standardoinnin merkitys ja sen tuomat mahdollisuudet. Tämä kaikki on kuitenkin mahdollista saavuttaa vain hyvän suunnittelun myötä, jotta kaikki eri osapuolet sitoutuvat standardointiin.

> *"Consequently, a significant boost in medical image analysis can be expected (1) if the source code is published together with the paper, allowing to compare different algorithms and avoiding re-implementation, (2) if the code is written in a standardized way, allowing to combine different algorithms/methods,(3) if standardized test data is used for evaluation, allowing to objectively judge the performance of algorithms, and (4) if the algorithms can be integrated in a clinical environment with reasonable effort, allowing to refine existing/develop new algorithms according to medical requirements, to prove their relevance, and, finally, to improve health care. Standardization is the key to all these issues. But standardization can only be successful with a thorough design, i.e., a consistent design suitable for all major applications of a domain and flexible enough to be extended to non-foreseen cases." [29]*

6.3 Valmiit komponentit

Niin aiemmassa tutkimuksessa kuin tämän tutkielman aineistossakin keskeiseksi avoimen lähdekoodin hyödyksi on nostettu valmiiden ohjelmistojen ja komponenttien uudelleenkäyttö. Näin voidaan estää "pyörän uudelleen keksiminen".

> *"Reuse of open source components greatly reduced the time necessaryto develop our system. They also helped avoiding reinventing the wheel, one of our goals." [27]*

Tärkeänä valmiiden avoimeen lähdekoodiin perustuvien komponenttien suosimisen syynä pidettiin myös sitä, että ne ovat usein hyvin testattuja ja vakaita.

> *"Because software development has a significant component of trial and error, reuse of well-tested components helps reduce technical problems, especially for complex functions like order entry." [11]*

> *"Additionally,we wanted to use open source components, not onlybecause of the possibilityto modify the software according to our needs, but also because open source components are believed to be more robust." [27]*

Tutkimusaineistossa artikkeleissa mainittiin useita erilaisia valmiita komponentteja, ohjelmistoja ja tekniikoita, joita oli hyödynnetty ohjelmiston kehittämisessä. Seuraavaan taulukkoon on koottu yhteen aineistossa yleisimmin mainitut valmiit komponentit ja tekniikat:

Taulukko 6.2: Aineistossa mainittuja komponentteja ja tekniikoita

Käyttöjärjestelmät	Linux Windows OS X
Palvelinohjelmistot ja ohjelmistokehitysalustat	Apache JBOSS J2EE LAMP (Linux, Apache, MySQL, PHP/PERl)
Ohjelmointikielet	C++ HTML Java Perl PHP Python Ruby
Tietokannat	MySQL SQL
Dokumenttistandardit	HTML XML XForms JDOM HL7
Internet-selaimet	Internet Explorer Mozilla Firefox Safari
Muut	FLTK ITK ITSTK JMS Jabber

6.4 Yhteenveto ja pohdinta

Modulaarisuus, standardit ja valmiiden komponenttien uudelleenkäyttö eivät ole ainoastaan avoimen lähdekoodin ohjelmistokehitykseen liittyviä asioita, vaan samat periaatteet ovat käytössä myös omisteisiin ohjelmistoihin perustuvassa ohjelmistotuotannossa. Niiden hyödyntäminen on kuitenkin keskeinen periaate avoimen lähdekoodin kehitystyössä. Ohjelmien ja järjestelmien modulaarinen rakenne on edellytys sille, että kehitystyö voi jakautua verkostomaisesti usealle taholle. Standardit taas huolehtivat moduulien keskinäisestä yhteensopivuudesta. Erityisesti terveydenhuollossa standardien merkitys on kasvanut koko ajan, sillä yksi suurimmista terveydenhuollon tietotekniikan ongelmista on ollut järjestelmien välinen yhteensopimattomuus. Sitovien standardien yleistyminen mahdollistaa sen, että vahvasti omisteisiin ohjelmistoihin perustuviin terveydenhuollon tietojärjestelmiin on mahdollista yhdistää myös avoimia ohjelmistoja.

Aineiston pohjalta tunnistettiin kolme keskeistä avoimen lähdekoodin ohjelmistoprojek-

tein tekniikkaa tai menetelmää. Nämä alateemat ja niitä koskevat pelkistetyt ilmaisut on kuvattu seuraavassa taulukossa:

Pääteema	***Menetelmät ja tekniikat kehitystyössä***
Alateema	*Ilmaisut*
Modulaarisuus	Mahdollisuudet muokata ohjelmiston toiminallisuutta Ohjelmistoa voidaan käyttää hyvinkin erilaisissa ympäristöissä Mahdollistaa lisäosat tai toimimisen lisäosana Mahdollistaa kerrosarkkitehtuurin
Standardit	Mahdollistaa tiedon siirron Tyypillistä sitoutuminen standardeihin Standardit edellyttävät hyvää suunnittelua
Valmiit komponentit ja koetellut tekniikat	Pyörän uudelleenkeksimisen välttäminen Laadukkaiden komponenttien uudelleenkäyttö

7 POHDINTA

7.1 Yhteenveto tutkielman tuloksista

Tutkimusaineiston artikkeleissa keskeisimmät motiivit avoimen lähdekoodin käytölle olivat pragmaattisia, kuten ohjelmiston jatkokehittämisen mahdollistaminen ja turvaaminen, ohjelmistojen välisen yhteensopivuuden varmistaminen, valmiin koodin uudelleenkäyttö sekä kustannushyödyt. Aineistossa avointa lähdekoodia käsiteltiin ohjelmistotuotantotapana, eikä juurikaan pohdittu ilmiön laajempia yhteiskunnallisia ulottuvuuksia, vaikka monet pragmaattiset hyödyt sisältävätkin laajempiakin yhteiskunnallisia vaikutuksia.

Aineiston ohjelmistoprojekteille oli tyypillistä syntyminen tiettyyn käyttäjien tarpeeseen, jota olemassa olevat ohjelmistotuotteet eivät pystyneet täyttämään. Tutkielman kohteena olleiden ohjelmistoprojektien organisoitumisessa keskeistä oli käyttäjälähtöisyys ja organisoituminen kehittäjäyhteisöihin. Käyttäjien osallistaminen ohjelmistosuunnitteluun nähtiin onnistuvan paremmin ideoija-käyttäjien kuin, että avoimelle lähdekoodille tyypillisten kehittäjä-käyttäjien kautta. Ohjelmistojen julkaisemisella avoimen lähdekoodin periaatteiden mukaisesti haluttiin kannustaa muitakin osallistumaan kehitystyöhön. Organisoituminen tapahtui pääasiassa internetpohjaisten työkalujen, kuten nettisivujen, sähköpostilistojen ja keskustelufoorumien avulla. Tutkielman aineistossa nostettiin esiin myös yritysten osuus kehittäjäyhteisyössä. Ohjelmistoyritysten rooli perustuisi asiantuntijuuteen ja tukipalveluiden myymiseen.

Keskeisimpiä menetelmiä ja tekniikoita projektien toteuttamisessa olivat työn organisoiminen modulaarisesti, tukeutumien standardeihin ja valmiiden luotettaviksi todettujen ohjelmistokomponenttien uudelleen käyttäminen. Näitä samoja periaatteita käytetään nykyisin myös omisteisuuteen perustuvassa ohjelmistotuotannossa, mutta nämä periaatteet ovat edellytys kehitystyön organisoimiseksi avoimelle lähdekoodille tyypillisiin verkostomaisiin yhteisöihin, joissa eri toimijat voivat tavoitella yhteistä päämäärää. Terveydenhuollon avointen standardien yleistyminen mahdollistaa nykyistä paremmin avoimen lähdekoodin ohjelmistojen mukaan ottamisen laajempiin järjestelmäkokonaisuuksiin.

Taulukko 7.1: Yhteenveto tutkielman tuloksista

Avoimen lähdekoodin käytön motiivit	
Jatkokehittämisen mahdollistaminen ja turvaaminen	Pääsy lähdekoodiin Kannustaminen edelleen kehittämiseen Kehittäjien yhteistyö ja "vertaisarvioitu" koodi Riippumattomuus ohjelmistotoimittajista
Yhteensopivuus	Koodin avoimuus mahdollistaa yhteensopivuuden toteuttamisen
Valmiin koodin uudelleenkäyttö	"Pyörää ei tarvitse keksiä uudelleen" Nopeuttaa kehitystyötä Valmiiden komponenttien vakaus
Kustannushyödyt	Valmiit komponentit saatavilla ilmaiseksi Ei lisenssikustannuksia Tukea mahdollista saada internetistä Terveydenhuollon rajallisten resurssien kohdentaminen hoitotyöhön Koodaustyö työllistää paikallisesti
Kehittäjäyhteisön organisoituminen	
Käyttäjälähtöisyys	Ohjelmiston tekeminen omaan tarpeeseen Muokkaaminen oman näköiseksi Käyttäjä-kehittäjät
Organisoitumismallit	Kehittäjäyhteisöt Kannustaminen osallistumaan kehitystyöhön Julkisen rahoituksen ja korkeakoulujen keskeinen merkitys Organisoitumisen muodollisuus vaihtelee
Yritysten liiketoimintamallit	support sellers loss leaders service enablers
Käyttäjien kuuleminen	osallistavan suunnittelun mahdollisuudet
Organisoitumisvälineet	nettisivut sähköpostilistat keskustelufoorumit wikit lähdekoodin hallintajärjestelmä dokumentaatio
Menetelmät ja tekniikat kehitystyössä	
Modulaarisuus	Mahdollisuudet muokata ohjelmiston toiminallisuutta Ohjelmistoa voidaan käyttää hyvinkin erilaisissa ympäristöissä Mahdollistaa lisäosat tai toimimisen lisäosana Mahdollistaa kerrosarkkitehtuurin
Standardit	Mahdollistaa tiedon siirron Tyypillistä sitoutuminen standardeihin Standardit edellyttävät hyvää suunnittelua
Valmiit komponentit ja koetellut tekniikat	Pyörän uudelleenkeksimisen välttäminen Laadukkaiden komponenttien uudelleenkäyttö

Mihin terveydenhuollon tietojärjestelmien haasteisiin avoin lähdekoodi voi vastata? Tämän tutkielman pohjalta voidaan sanoa, että avoin lähdekoodi ei ole mikään yksittäinen taikatemppu, jolla kaikki ongelmat korjautuvat. Pikemminkin on niin, että avoin lähde-

koodi antaa mahdollisuuksia tehdä tiettyjä asioita toisin. Luvussa 3.4 esitellyistä ongelmakohdista valtaosa on sellaisia, että ne eivät ole riippuvaisia koodin avoimuuteen liittyvistä asioista, vaikka joitain yhteyksiä näihin eri alueisiin löytyykin.

Tämän tutkielman keskeisenä tuloksena on, että avoimen lähdekoodin käyttämisessä terveydenhuollossa on keskeistä vallan siirtyminen ohjelmistoyrityksiltä järjestelmien käyttäjille ja kehittäjille. Lähdekoodin muuttuminen julkishyödykkeeksi tarkkaan varjellun liikesalaisuuden sijaan tarkoittaa suurta muutosta sen suhteen, kenellä on valta päättää siitä, mitä ohjelmistoon otetaan mukaan ja miten sitä kehitetään. Itse asiassa avoin lähdekoodi johtaa siihen, että kuka tahansa voi tehdä muutoksia ja kehittää koodia haluamaansa suuntaan. Tämä ei kuitenkaan tarkoita sitä, että koodista käännettyjä ohjelmia käytettäisiin terveydenhuollon organisaation sisällä kontrolloimattomasti. Käytännössä kehittämistavan muutos ei todennäköisesti näkyisi suurimmalle osalla loppukäyttäjistä juuri lainkaan. Kyse onkin ennen kaikkea siitä, että terveydenhuollon organisaatioiden valta ohjelmistojen kehittämisessä vahvistuu merkittävästi, ja ohjelmistoyritysten liiketoimintalogiikka joutuu väkisin mukautumaan tähän muutokseen. Omisteisessa mallissa käyttöoikeuksien myynti ja ohjelmistotuotteiden monistaminen ovat keskeiset liikevoiton lähteet. Avoimessa mallissa ohjelmistoyritysten rooli on myydä työpanosta ja erilaisia avoimen lähdekoodin tuotteiden ympärille luotuja palveluja terveydenhuollon organisaatioille.

Terveydenhuollon organisaatioiden suhdetta avoimiin ohjelmistoprojekteihin on tämän tutkielman pohjalta kuvattu nelikentässä (taulukko 7.2). Tässä tyypittelyssä vaaka-akselilla erotellaan, mitä kautta terveydenhuollon organisaatio sitoutuu käyttämänsä avoimen lähdekoodin ohjelmistoon. Yhteisöllisellä suhteella tarkoitetaan sitä, että organisaatio ottaa suoraan kehittäjäyhteisön luoman ohjelmiston käyttöönsä, vastaa itse ohjelmiston ylläpidosta, tukeutuu ongelmatilanteissa yhteisön asiantuntemukseen ja tehdessään muutoksia lähdekoodiin antaa ne koko yhteisön käyttöön.

Sopimuksellinen suhde taas tarkoittaa tilannetta, jossa terveydenhuollon organisaatio tekee sopimuksen avoimen lähdekoodin tuotteita tarjoavan yrityksen kanssa, joka vastaa siitä, että ohjelmisto toimii sopimuksessa luvatulla tavalla. Sopimuksessa voidaan tehdä sitoumuksia myös tukipalveluista, ylläpidosta, jatkokehittämisestä jne. Perusperiaate on,

että terveydenhuollon organisaatio voi ostaa avoimeen lähdekoodiin perustuvan ohjelmistotuotteen "avaimet käteen" -periaatteella.

Pystyakselilla taas erotellaan, miten terveydenhuollon organisaatio käyttää mahdollisuuttaan osallistua avoimen lähdekoodin ohjelmistoprojektin kehittämiseen omilla innovaatioillaan. Avoin koodi mahdollistaa ohjelmiston muokkaamisen omiin tarpeisiin sopivaksi ja tekemään siihen parannuksia. Näin organisaation sisältä nousevien kehittämisideoiden toteuttamiselle ohjelmistoissa ei ole ulkoisia esteitä. Kyse on ennen kaikkea organisaation omista asenteista, johtamisesta ja työkulttuurista.

Taulukko 7.2: Terveydenhuollon organisaation suhde avoimiin ohjelmistoprojekteihin

		Sitoutuminen ohjelmistoon	
		Yhteisöllinen	Sopimuksellinen
Innovointi	Suurta	1	2
	Pientä	3	4

Nelikentän kohdissa yksi ja kolme terveydenhuollon organisaatio valitsee omilla kriteereillään tarpeisiinsa sopivan avoimen lähdekoodin ohjelmiston. Organisaatio vastaa itse ohjelmiston käyttöönotosta ja sen sovittamisesta kokonaisarkkitehtuuriinsa. Näin ollen on keskeistä, että terveydenhuollon organisaation sisällä on riittävästi osaamista ja resursseja, jotta käyttöönotto, mahdolliset muokkaukset ja ohjelmiston ylläpito voidaan hoitaa itse. Keskeinen ero kohtien yksi ja kolme välillä on omien innovaatioiden toteuttamisessa. Kohdassa yksi eli *käyttäjä-kehittäjämallissa* terveydenhuollon organisaation sisällä tehdään ohjelmiston aktiivista kehittämistyötä ja annetaan kehittämistyön tulokset vastavuoroisesti myös muiden käyttöön. Kohdassa kolme eli *teknologiamallissa* taas ei juurikaan toteuteta omia muutoksia, joten käytettävät avoimen lähdekoodin ohjelmistot ovat usein valmiita vakaiksi ja laadukkaiksi todettuja tuotteita, kuten palvelinohjelmistoja ja tietokantoja. Samoja ohjelmistoja käytetään myös kohdassa yksi, mutta niitä voidaan myös kehittää edelleen tai niitä käytetään järjestelmien osina.

Tilanteen yksi vastakohtana taas voidaan pitää *puhdasta business-mallia* eli kohtaa neljä. Tämä vaihtoehto muistuttaa eniten perinteistä ohjelmistokauppaa. Ohjelmistotuotteen käytöstä sovitaan sopimuksellisesti, jolloin myyjä sitoutuu toimittamaan tuotteen ja ottaa sen toiminnasta täyden vastuun. Tuotteen tukipalvelut, ylläpito ja jatkokehitys ovat myös tuotteen myyjän vastuulla. Tässä tilanteessa terveydenhuollon organisaation omien ohjelmistoon kohdistuvien tarpeiden toteuttaminen tapahtuu sopimuksessa sovitulla tavalla eikä organisaation sisällä tehdä muutoksia ohjelmiston lähdekoodiin.

Kohdassa kaksi eli *kumppanuusmallissa* yhdistyvät terveydenhuollon organisaation ja yrityksen välinen sitoutuminen sopimuksilla sekä terveydenhuollon organisaatiossa tehtävä innovointi. Kuten tilanteessa neljä, vastuu ohjelmiston toiminnasta, tukipalveluista ja ylläpidosta on myyjällä, mutta ohjelmiston muokkaamisessa ja jatkokehittämisessä terveydenhuollon organisaatio ottaa merkittävän roolin ja antaa siihen omaa panostaan. Kumppanuusmallissa ohjelmiston kehittämistyö tapahtuu kehittäjäyhteisön, ohjelmistoon perustuvia tuotteita myyvän yrityksen sekä ohjelmiston käyttäjien yhteistyönä. Kaikki kolme osapuolta ovat riippuvaisia toisistaan.

Miten terveydenhuollossa voitaisiin kannustaa avoimeen lähdekoodiin keskeisenä osana kuuluvaan innovatiivisuuteen? Jotta käyttäjät voisivat todella osallistua terveydenhuollon tietojärjestelmien kehittämistyöhön joko kehittäjä-käyttäjinä tai ideoija-käyttäjinä, on innovaatioille luotava edellytykset työyhteisöissä. Terveydenhuollon yksiköissä on kannustettava innovatiiviseen ajattelutapaan, ja tarvitaan kommunikointikanavia, jota kautta työntekijöillä on mahdollisuus esittää muutoksia ohjelmistojen toimintaan, joko ehdotuksina uusista ominaisuuksista tai jopa suoraan koodina. Tämä edellyttää, että terveydenhuollon johtamisessa luodaan ympäristö, joka kannustaa innovatiivisuuteen.

Terveydenhuollon organisaatioille on ollut ominaista hierarkkisuus, joka on perustunut pitkälti käskyvaltasuhteisiin ja erikoistuneeseen työnjakoon. Tällaisessa asetelmassa terveydenhuollon työntekijä nähdään toimijana, joka toimii annettujen normien ja sääntöjen mukaan. (Itkonen 2005, 349). Käytännössä tämä näkyy johtamiskäytäntöinä, jotka eivät kannusta innovatiivisuuteen ja uuden kehittämiseen. Työkulttuuri pikemminkin estää innovatiivisuuden. Toisaalta on huomattava, että terveydenhuollon organisaatiot toimivat poliittishallinnollisen ohjauksen alla, mikä asettaa toiminnalle rajoituksia mm.

lainsäädännöllisin sekä ennen kaikkea taloudellisin reunaehdoin. Näin ollen terveydenhuollon organisaatioille ei ole tärkeintä optimoida toimintaansa kysynnän kasvattamiseksi, ja siksi joustava reagointi toimintaympäristön muutoksiin ei ole välttämätöntä. (Miettinen M. 2005, 268).

Organisaatioille, jotka ovat vähäisesti innovatiivisia, on tyypillistä visionäärisyyden ja visioiden puute tai aneemisuus. Terveydenhuollon organisaatioissa tämä näkyy muun muassa siten, että johtamisen ja henkilöstön välillä on luottamuksen puutetta. Toimintojen rutiininomaisuus ja työn kiireellisyys vähentävät innovatiivisuutta. Terveydenhuollossa on myös paljon reviiriajattelua ja organisaatioiden väliset tiukat toimialuerajat. Nämä kaikki vaikuttavat kielteisesti innovatiivisuuteen. (Miettinen M. 2005 269)

Innovatiivisille organisaatioille tyypillistä on oppiminen ruohonjuuritasolla. Tällaisissa organisaatioissa johtamisessa korostuu ihmisten ja tiedon johtaminen. Korkea innovatiivisuus edellyttää paljon demokratiaa ja vähän byrokratiaa. Työntekijöitä arvostetaan ja heihin luotetaan. Muutoksen johtamisessa arvioidaan aikaisempia prosesseja ja valitaan parhaat mahdolliset vaihtoehdot. (Miettinen M. 2005, 266)

Voidaan siis todeta, että avoimien innovaatioprosessien syntyminen terveydenhuollon organisaatioiden sisälle ei ole mahdollista, jos organisaatioiden sisällä ei tapahdu työkulttuurin ja johtamisen muutoksia. Pare ym (2009, 4) nostivat esiin terveydenhuollon tietotekniikan konservatiivisen luonteen. Tämän vuoksi on epätodennäköistä, että oman innovaation osuus ohjelmistojen kehittämisessä voisi nousta suureksi kovin nopeasti. Siksi avoimen lähdekoodin ohjelmistojen käyttöönotto ja kehittäminen terveydenhuollon organisaatioissa on sidoksissa ennen kaikkea tietohallinnosta vastaavien johtajien asenteisiin ja johtamiskulttuuriin. Organisaatioittain vaihtelevan tietohallintokulttuurin vuoksi eri organisaatioilla on erilaisia valmiuksia ottaa käyttöön avoimen lähdekoodin ohjelmistoja. On kuitenkin huomattava, että kaikkein konservatiivisimmatkin organisaatiot voivat ottaa käyttöön avoimen lähdekoodin ohjelmistoja, mikäli niitä tarjotaan perinteisten omisteisten ohjelmistojen tapaan vahvaan sopimuksellisuuteen perustuen.

7.2 Tutkimusprosessin arviointia

Laadullisen tutkimuksen luottavuuden arviointi ei ole yksiselitteistä eikä siihen ole olemassa selkeitä ohjeita. Näin ollen luotettavuutta on arvioitava kokonaisuutena, ja on huomioitava tutkimuksen kohde ja tarkoitus, tutkijan omat sitoumukset sekä tutkimuksen eettisyys. Keskeistä on tutkimuksen sisäinen johdonmukaisuus. (Tuomi & Sarajärvi 2004, 135-138) Koska yksiselitteisiä tutkimuksen arviointikriteereitä ei ole, keskeistä on tutkimusprosessin avoimuus. Tutkijan on selostettava kaikki tutkimuksen vaiheet, kuten aineiston keruuprosessi ja analyysin perusteet. (Tuomi & Sarajärvi 2004, 129; Hirsjärvi ym. 2004, 217)

Laadullisessa tutkimuksessa validiteetti voidaan ymmärtää pätevyytenä, uskottavuutena ja vakuuttavuutena: Onko tutkimus perusteellisesti tehty ja ovatko saadut tulokset ja tehdyt päätelmät "oikeita"? Toisin sanoen validiteetilla tarkoitetaan tutkijan kykyä rakentaa toimiva tutkimusasetelma sekä tutkijan tulkintojen paikkansapitävyyttä. (Eskola & Suoranta 2003, 219-222; Pyörälä 1995, 15; Saaranen-Kauppinen & Puusniekka 2006)

Laadullisessa tutkimuksessa reliabiliteetillä tarkoitettaan aineiston käsittelyn ja analyysin luotettavuutta. Tässä tärkeää on analyysin arvioitavuus ja uskottavuus. Analyysin arvioitavuutta auttaa, kun lukija pystyy seuraamaan ja toisaalta kritisoimaan tutkijan päättelyä. Uskottavuus taas tarkoittaa, että esitettyihin tulkintoihin päätyminen on tapahtunut uskottavasti. Analysoinnissa onkin tärkeää tehdä perusteltuja ja aukikirjoitettuja kategorisointeja ja koodauksia. (Pyörälä 1995, 16; Saaranen-Kauppinen & Puusniekka 2006)

Laadullisen tutkimuksen kohdalla esitetään usein, että aineistoa pitäisi kerätä niin paljon, että se kylläääntyy. Toisin sanoen samat asiat alkavat kertautua eikä lisäaineisto tuo enää tutkimusongelman kannalta uutta oleellista lisätietoa. Kirjallisuudessa esitetään yleensä, että noin 15 vastaajaa riittäisi aineiston kylläääntymiseen, mutta käytännössä on aina harkittava tutkimuskohtaisesti aineiston riittävyys . (Hirsjärvi ym. 2004, 171; Eskola & Suoranta 2003, 62-63)

Tässä tutkielmassa tutkimusprosessi on pyritty avaamaan niin yksityiskohtaisesti ja läpi-

näkyvästi, että joku toinen henkilö pystyisi toistamaan aineiston haun ja sen analyysin. Aineiston hakua varten testattiin eri hakusanoja, niiden lähikäsitteitä sekä erilaisia hakusanojen yhdistelmiä. Aineiston kerääminen PubMed-tietokannasta oli onnistunut valinta, sillä lisäaineistoa ei juurikaan saatu muista tietokannoista, joihin tehtiin täydentäviä hakuja. PubMed-tietokanta on merkittävin terveydenhuollon tietokanta. Tässä tutkielmassa tehdyn haun perusteella voidaan päätellä, että se kattaa hyvin myös terveydenhuollon tiedonhallintaa ja tietotekniikkaa käsittelevät julkaisut, sillä keskeisimpiin tietojenkäsittelytieteen tietokantoihin tehdyt haut eivät sisältäneet juurikaan tämän tutkielman kannalta merkittävää lisäaineistoa.

Hyväksyttävän aineiston sisäänotto- ja poissulkukriteerit oli suunniteltu etukäteen ja niiden avulla aineistoa saatiin rajattua hyvin. Hyväksyttävän aineiston laatuvaatimuksia jouduttiin kuitenkin tiukentamaan tulosten suuren määrän vuoksi. Tämä rajaaminen oli kuitenkin perusteltua.

Aineiston analyysin periaatteet on myös kuvattu. Analyysin avuksi tehty taulukko ja sen pohjalta syntyneiden pääteemojen muodostuminen auttaa lukijaa ymmärtämään vaiheita, joiden kautta teemoittelu eteni. Aineiston teemojen jakautuminen alateemoiksi on myös avattu. Tämä mahdollistaisi prosessin toistamisen.

Siinä vaiheessa, kun aineiston hakua tehtiin ja rajattiin, ei ollut vielä täysin selvää, mitä aineiston artikkelit kokonaisuutena sisälsivät. Käytännössä haasteeksi nousi se, että aineiston artikkelit ovat syntyneet omista lähtökohdistaan ja useimmissa niissä ei ensisijaisesti käsitelty niitä kysymyksiä, joista tässä tutkielmassa oltiin kiinnostuneita. Nyt aineiston kyllääntymistä on vaikea arvioida, sillä useissa kohdin aineiston ilmaukset olivat samansuuntaisia, mutta haastattelututkimuksissa käytetty viidentoista vastaajan kriteeri kyllääntymiselle ei täyty, sillä alateemoja muodostettiin huomattavasti alhaisemmillakin ilmaisujen määrillä, jotka käsittelevät samaa ilmiötä. Näin ollen laajempi aineisto olisi parantanut tutkielman luotettavuutta. Laadullisen tutkimuksen luonteeseen ei kuulu tulosten yleistettävyys, mutta mikäli tutkimusaineisto olisi ollut laajempi, se olisi voinut mahdollistaa menetelmätriangulaation, eli määrällisten tutkimusmenetelmien käytön laadullisen analyysin ohella. Toisaalta on myös todettava, että melkein kolmenkymmenen tieteellisen artikkelin läpikäyminen oli työlästä ja siksi aineiston rajaaminen

tutkielmassa tehdyllä tavalla oli perusteltua. Koska tutkielman aineisto perustui julkaistuihin tieteellisiin artikkeleihin, niin samanlaisia eettisiä kysymyksiä, joita esimerkiksi haastatteluaineistojen käsittelyssä joudutaan ottamaan huomioon, ei tässä tutkielmassa ole ollut tarpeen pohtia.

Tutkimuksen tekijän omat asenteet ja lähtökohdat vaikuttavat aina väistämättä tutkimusprosessiin. Erityisesti, jos aihe on läheinen, voi riittävän etäisyyden ottaminen ja objektiivisuuteen pyrkiminen olla vaikeaa. Tässä tutkielmassa oli prosessin pitkittymisestä tältä osin paljon hyötyä. Koska tutkielman tekemisessä oli pitkiäkin useiden kuukausien taukoja, se auttoi ottamaan etäisyyttä aiheeseen ja näkemään kokonaisuuden paremmin. Viime kädessä tämän tutkielman pätevyyden, uskottavuuden ja johdonmukaisuuden arviointi jää lukijalle, mutta tutkimusprosessin mahdollisimman avoimella kuvauksella on pyritty antamaan siihen mahdollisimman hyvät mahdollisuudet.

7.3 Jatkotutkimusaiheet

Tämän tutkielman tarkoituksena oli ymmärtää ja kuvata avoimeen lähdekoodin perustuvaa ohjelmistojen kehittämistapaa terveydenhuollon kontekstissa. Tämä tutkielma oli ensimmäinen suomalainen tutkimus avoimesta lähdekoodista terveydenhuollossa. Tutkittavaa siis riittää vielä paljon. Käytännön terveydenhuollon tietojärjestelmähankintoja tekevät päättäjät hyötyisivät selvityksistä, joissa käytäisiin läpi ohjelmistoprojekteja vaihtoehtoina kaupallisille tuotteille ja samalla selvitettäisiin niiden lokalisointimahdollisuuksia Suomeen. Samalla olisi myös hyödyllistä kartoittaa kaupallisten tietojärjestelmätoimittajien tarjoamia avoimen lähdekoodin tuotteita sekä niiden halua tarjota avoimen lähdekoodin tuotteiden tuki- ja ylläpitopalveluita. Paré ym. (2009) kaltainen tutkimus, jossa selvitettäisiin terveydenhuollon tietohallintopäätöksiä tekevien näkemyksiä avoimen lähdekoodin ohjelmistoista olisi myös hyödyllinen niin kotimaisessa kontekstisssa kuin laajemminkin eurooppalaisesti.

8 KIRJALLISUUS

Berg M. 2001. Implementing information systems in health care organizations: myths and challenges. International Journal of Medical Informatics 64 (2001) 143-156

Berkman Center for Internet and Society at Harvard Law School. 2005, Roadmap for Open ICT Ecosystems. Saatavilla osoitteesta: http://cyber.law.harvard.edu/epolicy/roadmap.pdf.

Berry, D M.& Moss, Giles. 2006. Free and Open-Source Software: Opening and Democratising e-Government's Black Box. Information Polity. Volume 11 (1). pp 21-34

Boberg Jorma. 2005. Johdatus tietojenkäsittelytieteeseen. Turun yliopisto.

Booth R. 2000. IT Project Failures Costly, TechRepublic/Gartner Study Finds. Saatavilla: http://articles.techrepublic.com.com/5100-10878_11-1062043.html Noudettu. 6.8.2009

Camp, L. Jean. 2006. Varieties of Software and their Implications for Effective Democratic Governmen" . Proceedings of the British Academy, Vol. 135, pp. 183-185, 2006 Available at SSRN: http://ssrn.com/abstract=905277

Carnall Douglas. 2002. Medical software's free future. BMJ. 2000. Oct 21;321(7267):976

Chesbrough Henry W. 2006. Open innovation : the new imperative for creating and profiting from technology. 2. painos. Boston (Mass.) : Harvard Business School Press.

Coleman Gabriella. 2005. NGO's in the Developing Worlds. Teoksessa Karaganis Joe & Latham Robert (toim). 2005. The Politics of Open Source Adoption. Version 1.0. Social Science Research Council.

Eskola Jari & Suoranta Juha. 2003. Johdatus laadulliseen tutkimukseen. 6. painos Vastapaino. Jyväskylä

Eskola Jari. 2007. Laadullisen tutkimuksen juhannustaiat. Laadullisen aineiston analysi vaihe vaiheelta. Teoksessa Aaltola Juhani & Valli Raine (toim.) Ikkunoita tutkimusmetodeihin II. Näkökulmia aloittelevalle tutkijalle tutkimuksen teoreettisiin lähtökohtiin ja analyysimenetelmiin. 2. korjattu ja täydennetty painos. Jyväskylä: PS-kustannus, 159-183.

Finanssipolitiikan linjat -hanke. 2010. Julkinen talous tienhaarassa. Finanssipolitiikan suunta 2010-luvulla. Valtiovarainministeriön julkaisuja 8/2010. Saatavilla sähköisessä muodossa osoitteesta www.vm.fi/julkaisut

Fitzgerald Brian. 2006. The Transformation of Open Source Software. MIS Quartely Vol. 30 Bo. 3. pp. 587-598 / September 2006

Fogel Karl. 2005. Producing Open Source Software: How to Run a Successful Free

Software Project. O'Reilly Media; 1 edition

Fuggetta Alfonso. 2003. Open source software – an evaluation. The Journal of Systems and Software 66(2003), 77-90.

Cerri, D & Fuggetta, A. 2007. Open standards, open formats, and open source. J. Syst. Software (2007),. doi:10.1016/j.jss.2007.01.048

Ghosh Rishab Aiyer. 2005. The European Politics of F/OSS Adoption. Teoksessa Karaganis & Latham (toim.). The Politics of Open Source Adoption. Version 1.0. Social Science Research Council.

Ghosh, Rishab Aiyer. 2006. Study on the Economic Impact of Open Source Software on Innovation and the Competitiveness of the Information and Communication Technologies (ICT) Sector in the EU. Contract number ENTR/04/112, UNU-MERIT, European Commission, November 20 2006,

Glott, R. and Ghosh, R. 2005. Usage of and Attitudes Towards Free / Libre and Open Source Software in European Governments. FLOSSPOLS: Government Survey Report, Contract number FP6-IST- 507524, MERIT, University of Maastricht, Netherlands, 31 March 2005,

Grassmuck Volker. 2005. Limux - Free Software for Munich. Teoksessa Karaganis & Latham (toim.). The Politics of Open Source Adoption. Version 1.0. Social Science Research Council.

Heeks Richard. 2006. Health Information Systems: Failure, Success And Improvisation. International Journal of Medical Informatics 75(2): 125-137

HE 176/2010. Hallituksen esitys eduskunnalle laeiksi sosiaali- ja terveydenhuollon asiakastietojen sähköisestä käsittelystä annetun lain sekä eräiden muiden lakien muuttamisesta.

Hietanen Herkko. 2001. Lähdekoodi ja huoltovarmuus, Helsinki Instititute for Information Technology,

Hoe Nah Soo. 2006. Free/Open Source Software – Open Standards. Asia-Pacific Development Information Programme. e-Primers on Free/Open Source Software. Elsevier. Saatavilla internetistä: http://www.iosn.net/open-standards/foss- open-standards-primer/foss-openstds-withcover.pdf

Hoepman Jaap-Henk & Jacobs Bart. 2007. Increased security through open source. Communications of the ACM. Volume 50 Issue 1

Hyysalo Sampsa. 2006. Käyttäjätieto ja käyttäjätutkimuksen menetelmät. Edita publishing.

Häkkinen Heidi & Korpela Mikko. 2007. A participatory assessment of IS integration needs in maternity clinics using activity theory. International Journal of Medical

Infomatics 76 (2007) 843-849

ITCortex. Failure rate: Statistics over IT projects failure rate. Saatavilla: http://www.it-cortex.com/Stat_Failure_Rate.htm Haettu: 6.8.2009

Iivari Anna-Kaisa & Ruotsalainen Pekka. 2006. Terveydenhuollon valtakunnallisen tietojärjestelmäarkkitehtuurin periaatteet. Alueellisista ratkaisuista kansalliseen kokonaisuuteen . Sosiaali- ja terveysministeriön selvityksiä 2006:8

Itkonen Pentti. 2005. Informaatioteknologian vaikutus strategisen johtamiseen ja toiminnan organisointiin. Teoksessa Vuori Jari (toim.) 2005. Terveys ja johtaminen. Terveyshallintotiede terveydenhuollon työyhteisöissä. WSOY

Janamanchi Balaji, Katsamakas Evangelos, Raghupathi Wullianallur , Gao Wei. (2009). The State and Profile of Open Source Software Projects in health and medical informatics. International Journal of Medical Informatics 78, 457-472

JHS 169 Avoimen lähdekoodin ohjelmien käyttö julkisessa hallinnossa. Saatavilla verkosta: http://www.jhs-suositukset.fi/suomi/jhs169

Johansson Kirsi, Axelin Anna, Stolt Minna & Ääri Riitta-Liisa (toim.). Systemaattinen kirjallisuuskatsaus ja sen tekeminen. Turun yliopisto. Hoitotieteen laitoksen julkaisuja. Tutkimuksia ja raportteja sarja A51

Kagai Bildad & Kimolo Nicolas. 2005. FOSSFA in Africa: Opening the Door to State ICT Development Agenda. A Kenya Case Study. Teoksessa Karaganis Joe & Latham Robert (toim). 2005. The Politics of Open Source Adoption. Version 1.0. Social Science Research Council.

Kaplan B & Harris Salamone K. 2009. Health IT Success and Failure: Recommendations from literature and an AMIA Workshop. Journal of the American Medical Informatics Association Volume 16, Number 3, 2009, 291-299

Kim Eugene. 2005. F/OSS Adoption in Brazil: the Growth of a National Strategy. Teoksessa Karaganis Joe & Latham Robert (toim). 2005. The Politics of Open Source Adoption. Version 1.0. Social Science Research Council

Kiuru Aaro. 1999. Tietotekniikan ja tietojärjestelmien käyttö radiologiassa ja kuvantamisessa. Teoksessa Tietotekniikka ja tiedonhallinta sosiaali- ja terveydenhuollosa (toim. Saranto & Korpela). WSOY, Porvoo

Karjalainen Martti. 2010. Large-scale migration to an open source office suite: An innovation adoption study in Finland. Tietojenkäsittelytieteiden laitoksen -sarja A-2010-4. Tampereen yliopisto. Saatavilla verkosta: http://acta.uta.fi/teos.php?id=11357

Karaganis Joe & Latham Robert (toim). 2005. The Politics of Open Source Adoption. Version 1.0. Social Science Research Council.

Kitcat Jason. 2004. Source Availability and E-Voting: An Advocate Recants.

Communications of the ACM. Vol 47. No. 10.

Koivisto Juha & Haverinen Riitta. 2006. Systemaattiset tutkimuskatsaukset vaikuttavuuden arvioinnin välineenä sosiaalialalla. Hallinnon tutkimus 25 (3), 108-126.

Korpela Mikko. 1994. Nigerian practice in computer systems development. A multidisciplinary theoretical framework, applied to health informatics. Helsinki University of Technology. Deparment of Computer Science Reports 1993: TKO-A31. Helsinki

Korpela Mikko & Saranto Kaija. 1999. Peruskäsitteet, osa-alueet ja toimijat. Teoksessa Tietotekniikka ja tiedonhallinta sosiaali- ja terveydenhuollosa (toim. Saranto & Korpela). WSOY, Porvoo

Krishnamurthy Sandeep. 2005. An Analysis of Open Source Business Models. Teoksessa toim. Feller ym. Perspectives on Open Source and Free Software, MIT Press,

Kuusisto-Niemi Sirpa & Saranto Kaija. 2009. Sosiaali- ja terveydenhuollon tiedonhallinta - Paradigma tieteenalan perustana. FinJeHew. Finnish Journal of eHealth and eWelfare. 2009. 1 (1) 19-23

Kääriäinen Maria & Lahtinen Mari. 2006. Systemaattinen kirjallisuuskatsaus tutkimustiedon jäsentäjänä. Hoitotiede 18 (1), 37-45

Lahti Jarmo. 2008. Kymmenen järjestelmää, viisi lääkäriä. Suomen Lääkärilehti 6/2007 vsk 63. Liite 6, 5-7

Laki sosiaali- ja terveydenhuollon asiakastietojen sähköisestä käsittelystä 9.2.2007/159

Lauharanta Jorma & Kekomäki Martti. 1999. Tietojärjestelmien käyttö terveyspalvelujen johtamisessa. Teoksessa Tietotekniikka ja tiedonhallinta sosiaali- ja terveydenhuollosa (toim. Saranto & Korpela). WSOY, Porvoo

Lee Jyh-An. 2006. New Perspectives on Public Goods Production: Policy Implications of Open Source Software. Vanderbilt J. of Entertainment and Tech. Law Vol 9:1:45:2006

Lehenkari Janne. 2003. Teknologisten innovaatioiden haaste terveydenhuollossa. Teoksessa Miettinen Reijo, Hyysalo Sampsa, Lehenkari Janne & Hasu Mervi. 2003. Tuotteesta työvälineeksi? Uudet teknologiat terveydenhuollossa. STAKES. Saarijärvi

Lessig Lawrence. 2006. Code. version 2.0. Basic Books. New York

Lääveri Tinja 2008a. Lääkärien mielipiteitä ei ole kuunneltu. Suomen Lääkärilehti 6/2008 vsk 63. Liite 6. 33

Lääveri Tinja 2008b. Kehitettävää riittää. Suomen Lääkärilehti 6/2008 vsk 63. Liite 6. 33

McDonald Clement J., Schadowab Gunther, Barnesab Michael, Dexterab Paul, Overhageab J.Marc, Mamlinab Burke, McCoyc J.Michael. 2004. Open Source software in medical informatics—why, how and what. International Journal of Medical Informatics. Vol. 69. Issue 2. 175-184

Mercuri Rebecca T. 2005. Trusting in Transparency. Communications of the ACM. Vol 48. No 5.

Miettinen Merja. 2005. Terveydenhuollon innovatiivisuuden esteitä, kannusteita ja mahdollisuuksia. Teoksessa Vuori Jari (toim.) 2005. Terveys ja johtaminen. Terveyshallintotiede terveydenhuollon työyhteisöissä. WSOY

Miettinen Reijo, Hyysalo Sampsa, Lehenkari Janne & Hasu Mervi. 2003. Tuotteesta työvälineeksi? Uudet teknologiat terveydenhuollossa. STAKES. Saarijärvi

Miettinen Reijo. 2006a. Hajautettu luominen ja tiedon omistusoikeudet. Tietoyhteiskunnan innovaatiopolitiikan perusteet. Tieteessä tapahtuu 4/2006

Miettinen Reijo, Toikka Kari, Tuunainen Juha, Lehenkari Janne & Freeman Stephanie. 2006b. Sosiaalinen pääoma ja luottamus innovaatioverkoissa. Helsingin yliopiston toiminnan teoria ja kehittävän työntutkimuksen yksikkö. Tutkimusraportteja 9. Helsinki

Moody Glyn. 2001. Kapinakoodi. Linus Torvalds ja vapaan ohjelmoinnin vallankumous. Tammi. Helsinki

Mykkänen J, Korhonen M, Porrasmaa J, Tuomainen T, & Ensio A. (2005). Tietojärjestelmien standardointityön organisointi ja kehittäminen terveydenhuollossa: nykytila ja toimenpide-ehdotukset . Standardointiselvitystyön loppuraportti . Osaavien keskusten verkoston julkaisuja 3/2005

Mäkela Kari. 2006. Terveydenhuollon tietotekniikka. Terveyden ja hyvinvoinnin sovellukset. Talentum. Tammer-Paino Oy

Mäkinen Kirsti & Soini Esa. 1999. Klinisen laboratorion tietojärjestelmät. Teoksessa Tietotekniikka ja tiedonhallinta sosiaali- ja terveydenhuollosa (toim. Saranto & Korpela). WSOY, Porvoo

Open Source Initiative. The Open Source Definition. Saatavilla verkosta: http://www.opensource.org/docs/osd Noudettu 15.5.2008

Osterloh M. & Rota S. 2007. Open source software development - Just another case of collective invention? Research Policy 36 157-171

Paré Guy, Wybo Michael D., Delannoy Charles. 2009. Barriers to Open Source Software Adoption in Quebec's Health Care Organizations. Journal of Medical Systems 33:1-7

Pekkala Eila. 2001. Systemaattiset kirjallisuuskatsaukset. Teoksessa Voutilainen Päivi, Leino-Kilpi Helena, Mikkola Taru & Peipponen Arja (toim.). 2001. Hoitotyön

vuosikirja 2001. Näyttöön perustuva hoitotyö. Kustannusosakeyhtiö Tammi. Tampere. 58-68.

Pihlava Minna. 2009. Potilastietojärjestelmä kehittyy hoitoprosessin kanssa. Mediuutiset Nro 21/2009. 22.5.2009

Puustinen Johanna. 2008. Mihin hukkui terveyshankkeiden 100 miljoonaa euroa? Tietoviikko 17.10.2008 Verkkolähde: http://www.tietoviikko.fi/kaikki_uutiset/article148618.ece Noudettu:10.8.2009

PMI. 2000. Guide to the Project Management Body of Knowledge (PMBOK Guide). Project Management Institute, 2000.

Pyörälä Eeva. 1995. Kvalitatiivisen tutkimuksen metodologiaa.. Teoksessa Jaakko Leskinen (toim.). Laadullisen tutkimuksen risteysasemalla. Helsinki. Kuluttajatutkimuskeskus

Raymond Eric. 1998. Goodbye, "free software"; hello, "open source" Verkkolähde: http://www.catb.org/~esr/open-source.html Noudettu 10.10.2010

Raymond Eric. 1999. The Cathedral & the Bazaar. Musings on Linux and Open Source by an Accidental Revolutionary. O'Reilly Media

Rogers Everett M. 2003. Diffusion of Innovations. New York. Free press

Saaranen-Kauppinen Anita & Puusniekka Anna. 2006. KvaliMOTV - Menetelmäopetuksen tietovaranto [verkkojulkaisu]. Tampere : Yhteiskuntatieteellinen tietoarkisto [ylläpitäjä ja tuottaja]. <http://www.fsd.uta.fi/menetelmaopetus/>. (Viitattu 13.07.2010.)

Saarelma Osmo. 1992. Perusterveydenhuollon tietojärjestelmien kehitys. Sosiaali- ja terveyshallitus raportteja 1992:49. VAPK-kustannus. Helsinki

Saranto Kaija & Kouri Pirkko. 1999. Tieto- ja viestintätekniikka kansalaisten ja ammatilaisten vuorovaikutuksen tiedonhankinnan välineenä. Teoksessa Tietotekniikka ja tiedonhallinta sosiaali- ja terveydenhuollossa (toim. Saranto & Korpela 1999). WSOY, Porvoo

Saranto Kaija. 2005. Tietojärjestelmät terveydenhuollon muutoksen johtamisessa. Teoksessa Vuori Jari (toim.). Terveys ja johtaminen. Terveyshallintotiede terveydenhuollon työyhteisöissä. WSOY

Setälä Maija. 2003. Demokratian arvo. Teoriat, käytännöt ja mahdollisuudet. Gaudeamus, Helsinki.

Shaw Nikki, Pepper David, Cook Tim, Houwink Pieter, Jan Nilesh & Bainbridge Mike. 2002. Open source and international health informatics: placebo or panacea? Informatics in Primary Care (2002) 10: 39-43

Staggers Nancy & Thompson Cheryl Bagley. 2002. The evolution of definitions for

nursing informatics: a critical analysis and revised definition. Journal of the American Medical Informatics Association : JAMIA 2002;9(3):255-61.

Stallman Richard. 2002. Free Software, Free Society: Selected Essays of Richard M. Stallman . GNU Press. Free Software Foundation. Boston. MA. Usa

Stolt Minna & Routasalo Pirkko. 2007. Tutkimusartikkelien valinta ja käsittely. Teoksessa Johansson Kirsi, Axelin Anna, Stolt Minna & Ääri Riitta-Liisa (toim.). Systemaattinen kirjallisuuskatsaus ja sen tekeminen. Turun yliopisto. Hoitotieteen laitoksen julkaisuja. Tutkimuksia ja raportteja sarja A51. Turku. 58-70

Suomi Reima. 2000. Leapfrogging for modern ICT usage in hte health care sector. Teoksessa. toim. Hansen, Bihler & Mahrer .Proceedngs of the 8th ECIS conference. Wien, 3-5 July 2000, 1269-1275

Suomi Reima. 2001. Streamlining Operatins in Health Care with ICT. Teoksessa toim. Spil & Stegwee. Strategies of Healthcare Information Systems. IDEA Group Publishing: Hershey, PA.

The Open Source Initiative. The Open Source Definition. http://www.opensource.org/docs/osd

Tuomi Jouni & Sarajärvi Anneli. 2002. Laadullinen tutkimus ja sisällönanalyysi. Helsinki: Tammi

Turkki Teppo. 2009. Nykyaikaa etsimässä. Suomen digitaalinen tulevaisuus. Elinkeinoelämän valtuuskunta. Helsinki

Turunen Pekka. 2001. Tietojärjestelmien arviointimenetelmien valinta terveydenhuolto-organisaatiossa - sidosryhmänäkökulma. Turun kauppakorkeakoulun julkaisuja. Sarja / Series A-5:2001, Turku.

Tolppanen Esa-Matti. 1999. Elektroninen potilaskertomus. Teoksessa toim. Saranto & Korpela. Tietotekniikka ja tiedonhallinta sosiaali- ja terveydenhuollosa. WSOY, Porvoo

Torkkeli Marko, Hilmola Olli-Pekka, Salmi Pekka, Viskari Sari, Käki Hannu, Ahonen Mikko & Inkinen Sam. 2007. Avoin innovaatio. Liiektoiminnan seitinohuet yhteistyörakenteet. Tutkimusraportti 190. Lappeenrannan teknillinen yliopisto. Kouvolan tutkimusyksikkö

Torvalds Linus & Diamond David. 2001. Just for Fun. Menestystarina. Schildts. Keuruu

van der Loo Ruud P. 1995. Overview of Published Asseessment and Ecaluation Studies. Teoksessa toim. van Gennip & Talmon. Assessment and Evaluation of Information Technolgy in Medicine. IOS Press. Amsterdam

Varonen Helena, Semberg Virpi & Teikari Matti (toim.). 1999. Tieteestä käytäntöön. Systemaattiset kirjallisuuskatsaukset terveydenhuollossa. FinOHTAn raportti 11. Saatavana URL: http://finohta.stakes.fi/FI/julkaisut/raportit/raportti11.htm

Weber Steven. 2004. The Success of Open Source. Harvard University Press.

West, J. and Dedrick, J. (2008). The Effect of Computerization Movements Upon Organizational Adoption of Open Source. In Kraemer, K. and Elliott, M. (eds.), Computerization Movements and Technology Diffusion: From Mainframes to Ubiquitous Computing, Information Today, Medford, NJ, USA.

Wikla Arto. 2003.Ohjelmoinnin perusteet Java-kielellä, 4. painos, OtaDATA.

Virtanen Heli & Salanterä Sanna. 2007. Laadullinen metayhteenveto – systemaattinen kirjallisuuskatsaus laadullisista tutkimuksista. Teoksessa Johansson Kirsi, Axelin Anna, Stolt Minna & Ääri Riitta-Liisa (toim.). Systemaattinen kirjallisuuskatsaus ja sen tekeminen. Turun yliopisto. Hoitotieteen laitoksen julkaisuja. Tutkimuksia ja raportteja sarja A51. Turku. 71-83.

Von Hippel Eric. 1998. The Sources of Innovation. Oxford University Press.

Von Hippel Eric. 2005a. Democratizing Innovation. Campridge, Mass. The MIT Press

Von Hippel Eric. 2005b. Open Source Software as ”User Innovation Networks”. Teoksessa Feller Joseph, Fitzgerald Brian, Hissam Scott A. & Lakhani Karim R. 2005 Perspectives on Free and Open Source Software. The MIT Press. Cambridge Massachusetts. London

Woods, D. & Guliani, G. (2005). Open Source for the Enterprise – Managing Risks, Reaping Rewards. O'Reilly, Sebastopol, CA, USA.

Välimäki Mikko, Oksanen Ville & Laine Juha. 2005. An Emprical Look at the Problems of Open Source Adoption in Finnish Municipalities

Välimäki Mikko. 2009. Oikeudet tietokoneohjelmistoihin. Talentum. Helsinki

LIITE 1: TUTKIMUSAINEISTO

	Artikkelin kirjoittajat	Artikkelin nimi	Julkaisu	Järjestelmätyyppi
1	Beckwith ym.	Development and evaluation of an open source software tool for deidentification of pathology reports	BMC Medical Informatics and Decision Making 2006, 6:12	Erillisjärjestelmä
2	Bellika ym.	Propagation of program control: A tool for distributed disease surveillance	International Journal of Medical Informatics 76 (2007) 313–329	Erillisjärjestelmä
3	Berman	Concept-Match Medical Data Scrubbing: How Pathology Text Can Be Used in Research	Arch Pathol Lab Med—Vol 127, June 2003	Erillisjärjestelmä
4	Berman ym.	The tissue microarray data exchange specification: A community-based, open source tool for sharing tissue microarray data	BMC Medical Informatics and Decision Making 2003, 3:5	Erillisjärjestelmä
5	Bernabe-Ortiz ym.	Handheld computers for self-administered sensitive data collection: A comparative study in Peru	BMC Medical Informatics and Decision Making 2008, 8:11	Erillisjärjestelmä
6	Bhupatiraju ym.	The MERG Suite: Tools for discovering competencies and associated learning resources	Source Code for Biology and Medicine 2008, 3:7	Erillisjärjestelmä
7	Blaya ym.	A web-based laboratory information system to improve quality of care of tuberculosis patients in Peru: functional requirements, implementation and usage statistics	BMC Medical Informatics and Decision Making 2007, 7:33	Laboratorojärjestelmä
8	Boulous ym.	Web GIS in practice IV: publishing your health maps and connecting to remote WMS sources using the Open Source UMN MapServer and DM Solutions MapLab	International Journal of Health Geographics 2006, 5:6	Erillisjärjestelmä
9	Bui ym.	openSourcePACS: An Extensible Infrastructure for Medical Image Management	IEEE Transactions on InformationTechnology in Biomedicine, Vol. 11, No. 1, January 2007	Kuvantamisjärjestelmä
10	Caban ym.	Rapid Development of Medical Imaging Tools with Open-Source Libraries	Journal of Digital Imaging, Vol 20, Suppl 1, 2007: pp 83Y93	Kuvantamisjärjestelmä
11	Fraser ym.	Implementing electronic medical record systems in developing countries	Informatics in Primary Care 2005;13:83–95	Potilaskertomus
12	Good ym.	SQL Clinic: The Open-Source Alternative for Electronic Medical Records	Psychiathric Services March 2005 Vol. 56 No. 3	Potilaskertomus

13	Inchingolo ym.	O3-DPACS Open-Source Image-Data Manager/Archiver and HDW2 Image-Data Display: An IHE-compliant project pushing the e-health integration in the world	Computerized Medical Imaging and Graphics 30 (2006) 391–406	Kuvantaminen
14	Langer	OpenRIMS: An Open Architecture Radiology Informatics Management System	Journal of Digital Imaging, Vol 15, No 2, (June), 2002: pp 91-97	Kuvantaminen
15	Los ym.	OpenSDE: A strategy for expressive and flexible structured data entry	International Journal of Medical Informatics (2005) 74, 481—490	Erillisjärjestelmä
16	Mandal ym.	Development of an electronic radiation oncology patient information management system	J Can Res Ther 2008;4:178-85	Potilaskertomus
17	Mandl ym.	Indivo: a personally controlled health record for health information exchange and communication	BMC Medical Informatics and Decision Making 2007, 7:25	Potilaskertomus
18	Neamatulla h ym.	Automated de-identification of free-text medical records	BMC Medical Informatics and Decision Making 2008, 8:32	Potilaskertomus
19	Potter ym.	Mastering DICOM with DVTk	Journal of Digital Imaging, Vol 20, Suppl 1, 2007: pp 47Y62	Kuvantaminen
20	Puentes ym.	Integrated multimedia electronic patient record and graph-based image information for cerebral tumors	Computers in Biology and Medicine 38 (2008) 425 – 437	Potilaskertomus
21	Rubin ym.	Protégé: A Tool for Managing and Using Terminology in Radiology	Journal of Digital Imaging, Vol 20, Suppl 1, 2007: pp 34Y46	Kuvantaminen
22	Schacht Hansen ym.	Wireless access to a pharmaceutical database: A demonstrator for data driven Wireless Application Protocol applications in medical information processing	J Med Internet Res. 2001 Jan–Mar; 3(1): e4	Erillisjärjestelmä
23	Seebregts ym.	The OpenMRS Implementers Network	Int. J. Med. Inform. (2009), doi:10.1016/j.ijmedinf.2008.09.005	Potilaskertomus
24	Stott ym.	OSPACS: Ultrasound image management system	Source Code for Biology and Medicine 2008, 3:11	Kuvantaminen
25	Sundvall ym.	Integration of tools for binding archetypes to SNOMED CT	BMC Medical Informatics and Decision Making 2008, 8(Suppl 1):S7	Erillisjärjetelmä
26	Tüttelman ym.	Optimising workflow in andrology: a new electronic patient record and database	Asian J Androl 2006; 8 (2): 235–241	Potilaskertomus

27	van der Linden ym.	PropeR visited	International Journal of Medical Informatics (2005) 74, 235—244	Potilaskertomus
28	Vasilescu ym.	WS/PIDS	IEEE Transactions on Information Technology in Biomedicine, Vol. 12, No. 1, January 2008	Potilaskertomus
29	Wolf ym.	The Medical Imaging Interaction Toolkit	Medical Image Analysis 9 (2005) 594–604	Kuvantaminen