Eettisesti valveutunut ohjelmistokehitysmalli

Iikku Mattila

Tampereen yliopisto
Tietojenkäsittelytieteiden laitos
Tietojenkäsittelyoppi
Pro gradu -tutkielma
2010

Tampereen yliopisto
Tietojenkäsittelytieteiden laitos
Tietojenkäsittelyoppi
Iikku Mattila: Eettisesti valveutunut ohjelmistokehitysmalli
Pro gradu -tutkielma, 66 sivua, 6 liitesivua
Toukokuu 2010

Tässä tutkielmassa luon ehdotelman eettisesti valveutuneeksi ohjelmistokehitysmalliksi. Rakennan kehitysmallin huomioimaan ohjelmistoalalle tyypillisiä eettisiä kysymyksiä, jotka keräsin erilaisista tietotekniikka-alan ammattilaisille suunnatuista eettisistä ohjeistoista, organisaatioille suunnatuista eettistä toimintaa tukevista toimintakehoituksista, erilaisten kehitysmallien kehittäjien ilmoittamista tavoitteista ja yhdestä esimerkkiohjelmistoprojektista.

Tutkielmassa vertailen erilaisia eettisiä ohjeistoja toisiinsa ja joihinkin historiallisiin etiikkakäsityksiin. Kritisoin ohjeistojen toimivuutta keinona eettisesti valveutuneeseen ohjelmistokehitykseen. Tarkastelen organisaatioille suunnattuja kehoituksia luottamuksen käsitteen kautta ja havaitsen, että eettisestä toiminnasta seuraa kaupallisesti merkittävä luottamussuhde asiakkaaseen.

Ohjelmistokehitysmallivalinnan ja joidenkin projektissa kohdattavien eettisten ongelmien välillä ilmenee riippuvuussuhde. Luodessani ehdotelmaa eettisesti valveutuneeksi ohjelmistokehitysmalliksi huomioin erityisesti ketterien menetelmien reagoinnin muiden mallien ongelmiin. Ehdotettu kehitysmalli huomioi myös esimerkkiohjelmistoprojektin kehittäjien haastattelu- ja lomakekysymyksien analyysissä havaitsemiani yritys-asiakas -suhteeseen liittyviä kysymyksiä.

Avainsanat ja -sanonnat: etiikka, moraali, eettiset ohjeistot, luottamus, ohjelmistokehitysmallit, ketterät menetelmät, ohjelmistotuotanto

Sisällysluettelo

1 Johdanto

Uusi teknologia, olkoonkin se ydinvoimaa, geenimanipulaatiota tai tietotekniikkaa, mahdollistaa uusia asioita ja herättää siten uusia moraalisia kysymyksiä. Tämä tutkielma käsittelee ohjelmistoalaa koskevia moraalisia kysymyksiä ja erityisesti ohjelmistojen ja niiden kehittämisen etiikkaa.

Onko ohjelmiston eettisyydellä edes väliä? Eettisyyden huomioiminen vie väistämättä resursseja, joita voitaisiin käyttää kenties hyödyllisemminkin. Esimerkiksi Sammon konsernijohtaja Björn Wahlroos [Talouselämä, 2008] on sanonut, ettei yrityksillä voi olla moraalia, ja että niiden ainoa tehtävä on tuottaa mahdollisimman paljon varallisuutta omistajilleen. Onko eettisen ohjelmiston tuottaminen ristiriidassa tämän tehtävän kanssa?

Eettinen valveutuneisuus saattaa olla myös kaupallinen menestystekijä. Ruotsalainen tutkija David Steinholtz toteaa, että: ”Eettiset tekijät ovat merkittävämpi kilpailutekijä kuin ympäristökysymykset, jotka otetaan nykyään jo itsestään selvinä. Etiikasta tulee samanlainen tekijä kuin ympäristökysymykset ovat olleet viimeisten kymmenen vuoden aikana.” [Aaltonen ja Junkkari, 2003, s. 33].

Lähivuosien ja -vuosikymmenten teknologinen kehitys on mahdollistanut yksittäiselle ihmiselle yhä suuremman vaikutusmahdollisuuden ympäristöönsä, ympäröivään yhteiskuntaan ja luontoon. Kehittyvän tekniikan käyttöön liittyvien eettisten ongelmien pohdinta ei ole kuitenkaan edennyt yhtä nopeasti. Lisäksi tietojenkäsittelyn historia on sangen lyhyt muihin tieteenaloihin verrattuna, joten eettisten ohjeistojen kehittyminen on tapahtunut lyhyen ajan sisällä. [Hirvonen, 2000] Toisin kuin tietojenkäsittelyllä, monilla muilla tieteen ja ammattien aloilla on ollut vuosisatoja aikaa kehittää käytäntöjä eettisten ongelmien käsittelyyn [Berleur and Brunnstein, 1996].

Tietokoneiden ja ohjelmistojen rooli nyky-yhteiskunnassa on keskeinen ja kasvava kaikilla elämän osa-alueilla. Tämän tietokoneiden valtaisan tärkeän roolin vuoksi ohjelmistokehittäjillä on yhtä valtaisat mahdollisuudet aiheuttaa haittaa tai hyötyä lukuisten, jopa kaikkien, ihmisten elämiin. Kehitettävät ohjelmistot saattavat myös välillisesti mahdollistaa muutoksen, haitan tai hyödyn, aiheuttamisen muille [ACM/IEEE-CS, 1999]. Koska potentiaalisen muutoksen piirissä on erittäin suuri määrä ihmisiä, rahaa ja muita resursseja, on ilmeisen tärkeää tarkastella päätösten eettisiä vaikutuksia. Eettisiin ongelmiin ei aina ole ilmeistä ratkaisua, ja siksi onkin pohdittava, millainen ohjelmisto on eettinen ja kuinka sellainen tehdään. Eettisten näkökulmien

olemassaolon hyväksymistä ja niiden huomiointia kutsutaan tässä tutkielmassa eettiseksi valveutuneisuudeksi.

Tutkielman tavoitteena on selvittää, mistä on kyse, kun puhutaan ohjelmiston eettisyydestä. Asiaa lähestytään vastaamalla seuraaviin tutkimuskysymyksiin:

- Mitä etiikka ja moraali ovat yleensä ja erityisesti tietojenkäsittelytieteen kontekstissa?
- Kuinka ohjelmistokehittäjiä on ohjeistettu eettiseen valveutuneisuuteen?
- Onko ohjeistajilla konsensus siitä, mikä on eettisesti oikein?
- Kuinka käyttäjä saadaan luottamaan ohjelmistoon ja sen tuottaneeseen yritykseen?
- Millainen ohjelmistonkehitysprosessi on eettisesti valveutunut?

Tutkielma keskittyy kaupalliseen ohjelmistontuotantoon ja sivuuttaa vapaan lähdekoodin ohjelmistot niiden erityisen luonteensa vuoksi.

Tutkielma on jaettu seitsemään lukuun. Johdannon jälkeinen luku käsittelee etiikkaa ja ohjelmistoalan eettisiä ohjeistoja. Luvussa esitellään joitain ohjeistoja ja arvioidaan niiden toimivuutta eettisesti valveutuneen ohjelmiston luomisessa. Kolmas luku käsittelee luottamusta, sen olemusta ja keinoja rakentaa sitä asiakkaan ja yrityksen välille. Luvussa myös pohditaan luottamuksen suhdetta etiikkaan. Luvussa 4 käsitellään ohjelmistokehitysmalleja ja sitä, miksi niitä käytetään ja kuinka ne vaikuttavat ohjelmiston eettisyyteen. Viidennessä luvussa nostetaan esimerkiksi todellinen ohjelmistoprojekti: kirjastolainauskorvausjärjestelmän kehittäminen. Luvussa tarkastellaan, millä tavoin eettisyys on huomioitu kyseisen ohjelmiston kehityksessä ja millaisia yleistyksiä projektissa tehdyssä havainnoista voi tehdä. Kuudennessa luvussa sidotaan yhteen aiemmat luvut, etsitään vaatimuksia eettisesti valistuneelle ohjelmistokehitysmallille ja esitellään ehdotus tällaiseksi malliksi. Viimeinen, seitsemäs, luku on yhteenveto tutkielmasta ja sen tuloksista.

2 Eettiset ohjeistot

Moraalisesti kestävää ohjelmistontuotantoa on tavallisesti lähestytty muodostamalla eettisiä ohjeistoja. Ohjelmistoalalla kohdatut eettiset ongelmat ovat myös luoneet tarvetta valmiille eettisille toimintaohjeistoille. Toisaalta eettisiä kysymyksiä on pohdittu jo pitkään ja saavutetut tulokset, eettiset kannanotot, on haluttu muotoilla

jäsennetyksi informaatioksi. Eettisten ohjeistojen olemassaoloa puoltaa myös se, että niiden avulla voidaan lisätä alan ammattilaisten tietoisuutta eettisistä tavoitteista ja ihanteista. Alan uudet toimijat eivät välttämättä tunne työelämässä vastaantulevia eettistä harkintaa vaativia tilanteita ja alalla pitkään toimineet ovat saattaneet rutinoitua harkitsemattomiin toimintamalleihin.

Ohjeistoja voidaan myös käyttää keskustelupohjana, kun ohjelmistonkehityksessä kohdataan eettinen ongelma, jota halutaan analysoida. Lisäksi ohjeistot välittävät julkisuuteen mielikuvaa eettisesti sitoutuneesta ammattikunnasta [Hirvonen, 2000]. Näiltä osin tietotekniikan alan ohjeistoja voidaan verrata Hippokrateen valaan, jonka tuntevat monet, joilla ei lääketieteellistä koulutusta olekaan.

Vaikka ohjeistot käsittelevät uudehkoja asioita, kuten teknologiaa ja tietojärjestelmiä, ne käsittelevät myös etiikan peruskysymyksiä: yksityisyyttä, rehellisyyttä, hyvyyttä ja auktoriteettia. Ohjelmistoalan moraaliset ongelmat eivät siis ole täysin uusia ja uniikkeja, vaan ne ovat eettisten kysymysten yksi ilmenemismuoto, toki omine painotuksineen ja uusine ympäristöineen. Siksi on hyödyllistä lähestyä ohjelmistoalan ongelmia ensin yleisesti etiikan näkökulmasta.

2.1 Etiikka

Etiikka on filosofian osa-alue, joka tutkii oikeaa ja väärää, oikeudenmukaisuutta ja velvollisuutta ja muita näihin liittyviä käsitteitä. Se tutkii myös ihmisten tekemien valintojen ymmärrettävyyttä ja hyväksyttävyyttä ja erittelee hyvän elämän perusteita. [Saarinen, 1994, s. 252] Etiikan tutkimuksella on monituhatvuotinen historia, mutta se on silti ollut filosofian suuntauksista nopeimmin kehittyvä ja tuloksiltaan kiinnostavin 1970-luvulta lähtien [Airaksinen, 1995, s. 120].

Tukiainen [1999] huomauttaa sanan ”eettinen” kahdesta merkityksestä: yhtäältä se on tapa luonnehtia etiikkaa käsittelevää tutkimus- tai ajattelutapaa, mutta toisaalta sen merkitys on ”eettisesti arvokas” tai ”eettisesti jalo”, eli se ilmaisee hyväksyntää ja ihailua. Tukiaisen mukaan etiikan ja moraalin käsitteitä käytetään suurin piirtein samoissa yhteyksissä; joskin moraali on suppeampi käsite, siinä missä etiikka kattaa jopa kaikkien ihmiselämien kokonaisuuden. Hänen mukaansa ei kuitenkaan ole olemassa terävää rajaa moraalisten ja ei-moraalisessa mielessä eettisten kysymysten välillä. Niinpä tässä tutkielmassa käytetään termejä ”eettinen” ja ”moraalinen” synonyymeina tarkoittamaan ihailtavaa ja tavoiteltavaa toimintaa.

Nykyistä länsimaista etiikkakäsitystä ei voi ymmärtää ilman käsitystä antiikin Kreikan ajattelusta, joka edustaa niin kutsuttua hyve-etiikkaa. Sen mukaan etiikka koostuu hyveistä, jotka perustuvat ajatukselle hyvän elämän saavuttamisesta. Teorioita siitä, mitä hyveet varsinaisesti ovat, on useita ja ne ovat keskenään ristiriitaisia. Mainitsen näistä teorioista esimerkkeinä kolme. Vanhin hyveteoria on Homeroksen teoria, jonka mukaan *hyve on sitä, että täyttää paikkansa maailmassa ja yhteiskunnassa*. Tämä käsitys kehottaa siis elämään perinteen ja sosiaalisen tavan mukaisesti. [Airaksinen, 1995, s. 69] Heikkouksistaan ja uudistumisvastaisuudestaan huolimatta Homeroksen ajatus toistuu, kenties hieman eri tavoin muotoiltuna, myös kohdassa 2.2 tarkastelluissa ohjelmistoalan ohjeistoissa.

Homeroksen jälkeen seurasi Sokrateen ajatus hyveestä. Hän esitti, että *hyve on tietoa.* Sokrateen mukaan ihmiset haluavat toimia hyveellisesti. Kun he saavat tietoa siitä, mikä on hyveellistä, he myös toimivat tiedon mukaisesti. Tätä ajatusta ei ole yleisesti hyväksytty, sillä ihmisen toimintaa ja luonnetta määrää tiedon lisäksi moni muukin asia. [Airaksinen, 1995, s. 71] Eettisten ohjeistojen tiedotusfunktion kannalta tämä on kuitenkin kiintoisa hyveteoria.

Hyveajattelu jatkui kauan antiikin jälkeen ja kolmas esimerkki hyveteoriasta on työnteon ja ammattinäkökulman kautta ohjelmistonkehitykseen liittyvä protestanttinen hyvekäsitys, jonka mukaan työnteko itsessään on hyve ja työ eettinen velvollisuus. Saksalainen yhteiskuntatieteilijä Max Weber esittää, että tämä hyveteoria selittää monien protestanttisten yhteiskuntien nousun taloudelliseen vaurauteen 1600-luvulta lähtien. [Airaksinen, 1995, s. 83]

Vuosien saatossa hyve-etiikan rinnalle ja tilalle muodostui uusia tapoja ajatella etiikkaa, jolloin deontologinen- eli velvollisuusetiikka ja teleologinen- eli seuraamusetiikka eriytyivät. Deontologisen etiikkakäsityksen mukaan teon oikeellisuus ei riipu sen seuraamuksista, vaan periaatteista, joihin nojautuen teko on tehty. Tällaista ajattelua edustaa muun muassa Immanuel Kant, jonka muotoileman kategorisen imperatiivin mukaan tulee toimia vain sellaisen periaatteen mukaisesti, jonka voi tahtoa tulevan yleiseksi laiksi [Saarinen, 1994, s. 161]. Deontologista käsitystä onkin kritisoitu juuri siitä, ettei ilmeisiäkään negatiivisia seuraamuksia oteta huomioon teon oikeellisuutta puntaroidessa. Teleologinen etiikkakäsitys sen sijaan painottaa juuri tekojen seurauksia: myönteiset seuraamukset tekevät itse teostakin eettisesti oikean. [Saarinen, 1994, s. 253]. Teleologisia teorioita ovat esittäneet esimerkiksi Jeremy Bentham [1789/2005, s. 310], jonka mukaan etiikka on mahdollisimman suuren onnellisuuden määrän

tuottamista mahdollisimman monelle, ja John Stuart Mill [1864, 14-16], joka jatkoi Benthamin teoriaa korostamalla henkisten onnellisuudenlajien olevan ruumillisia nautintoja parempia.

Seurausetiikan mukaan teon moraalista oikeutta voidaan arvioida esimerkiksi sen mukaan, lisääkö se tietoa, valtaa tai nautintoa. Ongelmana on, kenen näkökulmasta seuraamushyötyä tulee arvioida. Etiikka voidaan ymmärtää opiksi, joka kertoo, kuinka suurimmalle mahdolliselle ihmisjoukolle taattaisiin mahdollisimman suuri määrä hedonistisia arvoja: mielihyvää ja nautintoa [Airaksinen, 1995, s. 29].

Täysin toisenlaista ajattelua, eettistä egoismia, edustavat Epikuros ja Nietzsche. Heidän mukaansa seuraamuksia tulee arvioida vain teon tekijän näkökulmasta [Saarinen, 1994, s. 254]. Lisäksi Nietzsche hylkää koko ajatuksen etiikasta ja väittää, ettei voi olla olemassa yleispätevää etiikkaa. Samoin Karl Marx ajatteli etiikan olevan mahdottomuus, keino estää työväestön pyrkimyksiä valtaan [Airaksinen, 1995, s. 114].

Riippumatta siitä, mitä etiikka oikeastaan on, sitä on aina sovellettu ammatinharjoittamiseen. Vaikkei soveltava etiikka tarjoaisikaan yksimielisiä tuloksia, se mahdollistaa ristiriitaisten kannanottojen karsimisen ja auttaa selkiyttämään itse kunkin mielipiteitä. Ammattietiikan tehtävänä on määritellä arvot, joita ammattilaisen tulee työssään noudattaa. Alakohtainen etiikantutkimus pohtii vaaroja, joihin kyseistä ammattia harjoitettaessa yleisimmin langetaan. [Airaksinen, 1995, s. 127]

2.2 Olemassa olevia ohjeistoja

Ohjelmistoalaa ja sillä työskentelyä koskevia eettisiä ohjeistoja on tehty useita. Osa ohjeistoista on tietojenkäsittelijöiden liittojen kokoamia, osa yritysten sisäisiä, osa taas riippumattomien tutkijoiden koostamia. Eri kokoajat luonnollisesti korostavat eri piirteitä, vaikka pyrkivätkin kokonaisvaltaiseen näkökulmien huomioimiseen. Esimerkiksi The Institute for the Management of Information Systems -järjestön eettisen ohjeiston esipuheessa eri näkökulmat huomioidaan mainitsemalla, että kokonaisuudesta irrotettua yksittäistä ohjetta ei tule käyttää epäeettisten tekojen oikeutukseen, eikä tilanteeseen soveltuvan ohjeen puuttuminen oikeuta eettisen pohdinnan sivuuttamiseen [Rogerson et al., 2001]. Tarkkaa määritelmää siitä, mikä on eettinen ohjeisto, on vaikea tehdä juuri ohjeistojen runsaslukuisuuden ja kattavuusvaihteluiden vuoksi. Niinpä lähes mitä tahansa toiminta- tai käytösohjekokoelmaa voitaisiin pitää jonkinasteisena eettisenä ohjeistona.

Seuraavassa esitellään neljä ohjeistoa, jotka edustavat eettisten ohjeistojen kirjoa. Ensimmäisen ohjeiston takana on Association for Computing Machinery, joka on maailman vanhin tietotekniikan alan tieteellinen järjestö. Toisen esitellyn ohjeiston on koonnut Tietotekniikan Liitto, joka on suomalainen alunperin vuonna 1953 perustettu tietotekniikka-alan järjestö. Tutkijoiden kokoamia eettisiä ohjeistoja edustavat Rogersonin ja Gotterbarnin periaatteisto ja viimeisenä esitelty Bernard Gertin moraalisäännöstö, joka on myös esimerkki siitä, kuinka abstrakteja ohjeistot voivat olla.

2.2.1 Association for Computing Machineryn ohjeisto

Association for Computing Machinery (myöhemmin: ACM) perustettiin vuonna 1947. Vuonna 1992 se julkaisi eettisen ohjeiston omille jäsenilleen ja myöhemmin, vuonna 1999, se julkaisi ohjeiston kaikille ohjelmistotuotannon saralla työskenteleville ja sitä opettaville [ACM/IEEE-CS, 1999]. Se, että järjestöllä kesti noin 50 vuotta ennen kuin ohjeistot luotiin, kertoo osaltaan siitä, että eettinen pohdinta on verrattain tuore ilmiö tietotekniikka-alalla.

Vuoden 1992 ohjeisto on neliosainen, ja siihen kuuluvat seuraavat osiot:

1. yleiset moraaliset vaatimukset
2. tarkemmat ammatilliset vastuut
3. organisaation johtajuuteen liittyvät vaatimukset
4. ohjeiston noudattaminen.

Osiot kattavat yhteensä 24 tarkahkoa toimintaohjetta tai -sääntöä perusteluineen. Käyn seuraavaksi lyhyesti läpi osioiden sisällöt.

Yleiset moraaliset vaatimukset ovat nimensä mukaisesti yleisiä: ne käsittelevät yksittäisen ihmisen roolia yhteiskunnassa ja suhteessa muihin ihmisiin. Vaatimukset vastaavat yleistä konsensusta siitä, mikä on oikein: älä vahingoita muita, ole luotettava ja reilu, edistä yhteiskunnan hyvinvointia.

Tarkemmat ammatilliset vastuut ovat selkeämmin tietojenkäsittelyyn liittyviä. Näiltä osin ohjeisto vaatii kenties yllättäviäkin asioita: muun muassa mahdollisimman korkeaa ohjelmiston laatua, ammatillista kehittymistä ja suuren yleisön tietojenkäsittelyä koskevan tietoisuuden parantamista. Toki tarkempiin ammatillisiin vastuisiin kuuluu myös muita seikkoja, kuten ohjelmistoalaa koskevan lainsäädännön tunteminen ja noudattaminen, sopimusten kunnioittaminen ja tietojärjestelmien käyttö vain oikeutettuna.

Organisaation johtajuuteen liittyvät vaatimukset ovat osin ohjelmistoalalle spesifejä, osin yleisesti johtajuuteen liittyviä. Ensinnäkin johtajan tulisi selkeästi ilmoittaa alaisilleen näiden yhteiskunnalliset vastuut organisaation osana ja kannustaa näiden vastuiden kantamiseen. Tässä heijastuu aiemmin mainitsemani Homeroksen hyveteoria: hyveellisyys on sitä, että täyttää paikkansa yhteiskunnassa.

Lisäksi johtajalta edellytetään muun muassa taidokasta resurssienjakoa, kehottamista tietojärjestelmien oikeanlaiseen käyttöön ja ammatillisen kehittymisen mahdollisuuksien luomista organisaation jäsenille. Erityisesti maininnan arvoinen on ACM:n ohjeiston kohta 3.4, joka liittyy ohjelmiston vaikutuksen arviointiin. Kohdan mukaan jokaisen, johon luotava järjestelmä vaikuttaa, tulisi saada kertoa omat tarpeensa vaatimusmäärittelyvaiheessa ja nämä tarpeet tulisi huomioida järjestelmän toteutuksessa.

Ohjeiston viimeinen kohta kehottaa noudattamaan ohjeiston tavoitteita ja edistämään niiden yleistä noudattamista. Koko ohjeiston viimeinen alakohta on tärkeä – sen mukaan ohjeiston rikkominen on ristiriidassa ACM:n jäsenyyden kanssa ja törkeää rikettä saattaakin seurata erottaminen järjestöstä.

Uudemmassa, vuoden 1999 ohjeistossa, joka on suunnattu kaikille ohjelmistoalalla työskenteleville, luetellaan kahdeksan periaatetta. Jokaisella niistä on viidestä viiteentoista alakohtaa, yhteensä 80 sääntöä. [ACM/IEEE-CS, 1999]

ACM:n ohjelmistonkehittäjille neuvomat periaatteet ovat [ACM/IEEE-CS, 1999]:

1. Ohjelmistonkehittäjän tulee toimia yhdenmukaisesti yhteiskunnan edun kanssa.
2. Ohjelmistonkehittäjän tulee toimia asiakkaansa ja työnantajansa parhaaksi johdonmukaisesti yhteiskunnan edun kanssa.
3. Ohjelmistonkehittäjän tulee taata tuotteidensa korkein mahdollinen ammatillinen taso.
4. Ohjelmistonkehittäjän tulee säilyttää koskemattomuutensa ja itsenäisyytensä ammatillisessa päätännässä.
5. Johtavassa tai hallinnoivassa asemassa olevien ohjelmistonkehittäjien tulee toimia ja kehottaa muita toimimaan eettisesti ohjelmiston kehityksessä ja ylläpidossa.

6. Ohjelmistonkehittäjän tulee edistää alan eheyttä ja mainetta yhteiskunnan edun mukaisesti.
7. Ohjelmistonkehittäjien tulee olla reiluja ja avustavia kollegoitaan kohtaan.
8. Ohjelmistonkehittäjien tulee harjaantua ammatillisessa osaamisessa läpi elämän ja edistää eettistä lähestymistapaa alan harjoittamiseen.

Tämä ohjeisto on monella tapaa laajennos aiemmasta ohjeistosta. Ohjeet ovat tarkempia ja yksityiskohtaisempia kuin aiemmassa ohjeistossa, ja vaikuttavat kenties siksi suoraselkäisiltä ja ehdottomilta. ACM kuitenkin tiedostaa periaatteiden soveltamisen vaikeuden käytännössä. Siksi se huomauttaa, ettei periaatteisto ole valmis algoritmi, jota käyttäen syntyisi eettisiä ratkaisuja [ACM/IEEE-CS, 1999]. Perusteluksi tälle lievennyslausekkeelle annetaan tietyissä tilanteissa mahdollisesti esiin nouseva periaatteiden keskinäinen jännite, mutta tällainen perustelu arveluttaa. Mikäli ACM:n periaatteistoa käytettäisiin todellisessa elämässä ohjelmistoteollisuudessa tehtyjen päätösten arviointiin, kuten esimerkiksi Julkisen sanan neuvosto käyttää journalistin ohjeita [Journalistiliitto] päätösten arviointiin, olisiko ohjelmistonkehittäjän tai ohjelmistotalon liian helppoa vedota tähän lievennyslausekkeeseen?

2.2.2 Tietotekniikan liiton ohjeisto

Lähin suomalainen vastine ACM:n ohjeistolle on Tietotekniikan liiton (myöhemmin: TTL) tarjoama etiikan ohjeisto, joka vuonna 2002 päivitettiin kolmanteen versioonsa.

ACM:n ja TTL:n ohjeistot ovat laajuudeltaan samaa luokkaa: ACM:n ohjeistossa on kahdeksan pääkohtaa, TTL:n ohjeistossa vain seitsemän. Kohdat eivät kuitenkaan ole ohjeistoissa keskenään täysin samoja. TTL:n ohjeiston pääkohdat ovat seuraavat:

1. valta ja vastuu
2. tieto ja kokemus
3. asenne
4. viestintä
5. työn vaikutukset
6. muut ihmiset
7. eettisyyden kasvu.

Kiintoisaa on se, ettei TTL tarjoa pääkohtiensa alle varsinaisia alakohtia, vaan yleisen selitteen tai tarkenteen. Ensimmäisen kohdalla alla on esimerkiksi tarkennus:

”[Tietotekniikan ammattilaisen] on kannettava vastuunsa, joka näkyy tekoina ja toimina”, mutta myös selite: ”Tieto on valtaa ja tiedon käyttäminen vaatii viisautta, kuten muukin vallankäyttö.”

Tästä seuraa, että TTL:n ohjeisto on hieman abstraktimpi kuin ACM:n. Ohjeiston esipuheessa mainitaankin, että ”Koska eettisiin ongelmiin ei voida ennalta antaa täydellisiä ohjeita, tämän ohjeiston kohtia ei tule lukea ehdottomina totuuksina vaan suunnan näyttäjinä.” Ohjeisto ei esitä yksityiskohtaisia huomioita, mutta tuo esiin eri näkökulmia kuhunkin pääkohtaan.

Ohjeistossa huomautetaan mahdollisuudesta ottaa yhteyttä ohjeiston luoneeseen, TTL:n alaiseen etiikan työryhmään ja kysyä neuvoja eettisen ongelman ratkaisuksi. Tämä on hieno tapa parantaa ohjeiston hyödyllisyyttä – tarkoituksenahan on auttaa alan ammattilaisia ratkomaan eettisiä ongelmia. Palvelua ei kuitenkaan ole hyödynnetty: ohjeiston koonneen eettisen työryhmän edustajan Kari Kaipaisen mukaan viime vuosina ei ole tullut yhteydenottoja.

2.2.3 Rogerson ja Gotterbarnin periaatteet

Rogerson ja Gotterbarn [1997] esittelevät kahdeksan eettistä periaatetta, joita voi hyödyntää tehtäessä projektiin liittyviä päätöksiä, analysoitaessa tilanteita ja tiedotettaessa projektiin liittyvistä seikoista eri sidosryhmille. Nämä periaatteet ovat:

1. kunnia
2. rehellisyys
3. vinouma
4. ammatillinen kyky
5. huolellisuus
6. reiluus
7. sosiaalisen hinnan arviointi
8. tehokas toiminta.

Periaatteiden lyhyet nimet kätkevät taakseen suurempia merkityksiä. ”Kunnia” kysyy, koetaanko teko häpeälliseksi. ”Rehellisyys” varmistaa, ettei teko riko eksplisiittisiä tai implisiittisiä sopimuksia tai luottamusta. ”Vinouma” kehottaa huomioimaan ulkopuolelta tulevan paineen, joka vaikuttaa päätöksiin. ”Ammatillinen kyky” varmistaa, että ammatillinen osaaminen riittää aiottuun tehtävään.

”Huolellisuus” tarkoittaa parhaan mahdollisen laadunvalvonnan takaamista, ”reiluus” puolestaan kaikkien osapuolten kuulemista ja näkemysten huomioimista. ”Sosiaalisen hinnan arviointi” kysyy, pystytäänkö hyväksymään tehtävästä päätöksestä johtuvat seuraamukset ja vastuu. ”Tehokas toiminta” käskee varmistamaan, että aiottu teko vaikuttaa tilanteeseen toivotusti ja kuluttaa mahdollisimman vähän resursseja.

Nämä periaatteet ovat huomattavan paljon abstraktimpia kuin ACM:n tai TTL:n ohjeistoissaan esittämät vaatimukset. Siinä, missä ACM:n ohjeet ovat ohjelmistokehittäjän työkalu, eräällä tapaa imperatiivinen toimintaohjeisto, ovat Rogersonin ja Gotterbarnin esittämät periaatteet ennemminkin tavoitteita, joiden saavuttamiseksi voi toimia monin eri tavoin. Tästä seuraa, että tämä ohjeisto on yleispätevämpi – lähes mitä tahansa työstettävää ongelmaa voidaan analysoida näiden periaatteiden kautta.

Rogerson ja Gotterbarn huomauttavat, että on epäkäytännöllistä arvioida jokaista pientäkin seikkaa yksityiskohtaisesti jokaisen kahdeksan periaatteen näkökulmasta. Kuinka sitten voi tietää, missä tilanteissa tulisi suorittaa tarkempi tilanneanalyysi? Tähän heidän esseensä ei anna vastausta, vaan huomautus ainoastaan ilmaisee, että kompromissit ovat välttämättömiä.

Näistä periaatteista paistaa kuitenkin ensisijaisesti läpi vaatimus siitä, että ohjelmiston kehittäjän tulisi sitoutua ohjelmistoon koko sen elinkaaren ajan ja kantaa ammatillinen vastuu ohjelmiston laadusta. Vaatimuksissa on tietty ehdottomuuden tuntu, mutta se on ymmärrettävää, sillä ne ovat yleisiä periaatteita, eikä kaikkia poikkeustapauksia ja -tilanteita voi listata. Toisaalta tämä saattaa aiheuttaa tilanteita, joissa periaatteet ovat keskenään ristiriidassa. Joissain tapauksissa voisi esimerkiksi olla häpeällisempää olla yrittämättä jotakin, johon ammatillinen kyky ei välttämättä riitä, kuin tehdä se ja epäonnistua.

Rogersonin ja Gotterbarnin abstraktit periaatteet jättävät lopullisen pohdinnan tilanteesta ohjelmistokehittäjän harteille. Kun esimerkiksi ACM:n vuoden 1999 ohjeisto spesifisti käskee ohjelmoijaa kehittämään taitojaan tarkan, avuliaan ja hyvin kirjoitetun dokumentaation kirjoittamisessa, on Rogersonin ja Gotterbarnin periaatteistossa vain maininta mahdollisimman korkeasta ohjelmiston laadusta.

2.2.4 Bernard Gertin moraalisäännöt

Rogerson ja Gotterbarn [1997] puhuvat ohjelmiston sidosryhmien tunnistamisesta, joka heidän mukaansa on olennainen osa ohjelmistonkehityksen eettisten vaikutusten

arviointia. Näiden ryhmien tunnistamiseen he suosittelevat työkaluksi Bernard Gertin kymmentä moraalista sääntöä, joita voi hyödyntää muissakin ohjelmistokehitysprosessin vaiheissa. Gertin säännöt ovat:

1. älä tapa
2. älä aiheuta kipua
3. älä estä toimimasta
4. älä vie vapautta
5. älä vie nautintoa
6. älä petä
7. pidä lupauksesi
8. älä huijaa
9. noudata lakia
10. tee tehtäväsi.

Gertin säännöt ovat abstrakteja, joissain tapauksissa monitulkintaisia ja ne voivat olla keskenään ristiriidassa. Tästä huolimatta niiden käyttö sidosryhmien tunnistamisessa vaikuttaa toimivalta työkalulta. Yleensä ongelma lienee ennemmin se, ettei kaikkia sidosryhmiä onnistuta tunnistamaan, kuin se, että tunnistettaisiin virheellisesti sidosryhmä, johon sovellus ei vaikutakaan. Kaikki mahdollinen apu sidosryhmien löytämiseen on siis arvokasta.

Säännöt sopivat tienviitoiksi eettisten ongelmien pohdintaan siinä missä Rogersonin ja Gotterbarnin ehdottamat periaatteetkin. Koska Gertin säännöt ovat esittelemistäni ohjeistoista abstrakteimmat, ne soveltuvat varmimmin mihin tahansa tilanteeseen, mutta eivät toisaalta selkeästi yhteenkään. Abstraktiudessaan säännöt myös herättävät kysymyksiä: viittaako esimerkiksi tappamisen kieltäminen vain ihmisiin vai myös eläimiin, luontoon, kulttuuriin tai kilpaileviin ohjelmistoalan yrityksiin?

2.3 Yhteisiä piirteitä

Vaikka jotkut edellä kuvatuista ohjeistoista ovat yksityiskohtaisia ja toiset abstrakteja, löytyy niistä myös paljon yhdenmukaisuutta ohjeistajasta riippumatta. Yhteenvetona eri mielipiteistä voisi sanoa, että ohjelmistoalan yrityksen ja yksittäisen ohjelmoijan tulee sovelluksen kehitysprosessissa huomioida seuraavat tavoitteet:

- Yhteiskunnan osana toimiminen: Lain tunteminen ja noudattaminen, sosiaalisen vastuun kanto ja yhteiskunnan edun tavoitteleminen.
- Rehellisyys ja reiluus: Lupausten pitäminen, sopimuksien noudattaminen, asiakkaalle kertominen myös seikoista, joita tämä ei osaa kysyä, ohjelmistokehittäjien kohteleminen tasavertaisesti osaamistasonsa perusteella ja kaikkien osapuolten huomioiminen vaatimusmäärittelyssä.
- Vastuu päätöksistä: Taipumattomuus ulkopuolisen painostuksen alla, johtavassa roolissa olevan johdonmukainen alaisten kohtelu, sosiaalisen vastuun kantaminen, lupausten pitäminen, toiminnan hyödyllisyys ja tehokkuus sekä vastuullinen tietojärjestelmien käyttö.
- Laadukas ohjelmisto: Kaikkien osapuolten huomioiminen vaatimusmäärittelyssä, asiakkaan edun mukainen toiminta, korkein mahdollinen ohjelmiston taso ja paras mahdollinen laadunvalvonta.
- Tietojenkäsittelytieteen edistäminen: Ammatillisen kyvyn tiedostaminen, harjaantuminen ammatillisessa osaamisessa, oppimismahdollisuuksien takaaminen alaisille, kollegoiden avustaminen, eettisten toimintatapojen edistäminen sekä ohjelmistoalan maineen ja tunnettavuuden parantaminen.

2.4 Ohjeistojen kritiikkiä

Eettisten ohjeistojen olemassaoloa ja oikeutusta on kritisoitu runsaasti. Esittelen seuraavassa tärkeimpiä ohjeistoja vastustavia argumentteja.

Thomson ja Schmoldt [2001] väittävät, että eettiset koodistot eivät itsessään muuta ohjelmistokehittäjien käyttäytymistä. Heidän mukaansa eettisen ohjelmistokehitysmallin käyttöönotto on kuitenkin perusteltavissa, mutta syynä on ennemminkin ohjelmiston onnistumistodennäköisyyden huomattava paraneminen. Thomson ja Schmoldt perustelevat väitteen paremmasta onnistumisesta seuraavasti: ohjelmistoprojekteista arvioidaan epäonnistuvan noin 70% ja usein syynä on kyvyttömyys tai vastentahtoisuus ymmärtää ihmiskontekstia. Täten ihmisten hyvä- ja pahakäsitysten tarkempi analyysi (ts. etiikka) tuottaisi varmemmin onnistuvia ohjelmistoja. Onnistuneet ohjelmistot puolestaan merkitsevät kaupallista menestystä tai käyttäjätyytyväisyyttä, jotka ovat yritykselle suotuisia asioita. Malli on teoreettinen, eivätkä Thomson ja Schmold osoita sille empiiristä todistusaineistoa.

Tutkimustietoa tarjoaa sen sijaan ruotsalainen ”FörebildsFöretaget konsument” -tutkimus, jonka mukaan 99,4% ihmisistä on ainakin joskus hylännyt yrityksen tai tuotteen sen takana olleiden epäeettisten arvojen takia [Aaltonen ja Junkkari, 2003, s. 32]. Tämä huikean korkea luku kertoo eettisten arvojen merkityksestä kuluttajalle, mutta ei ota kantaa siihen, kuinka vakavat rikkeet ovat kyllin vakavia vaikuttamaan ihmisten kulutuspäätöksiin.

Thomson ja Schmoldt [2001] esittävät muutakin kritiikkiä: eettiset ohjeistot ovat suunnattu ihmisyksilöille, joten ne pitävät eettisistä epäonnistumisista syyllisinä yksittäisiä henkilöitä, ei koko organisaatiota tai yritystä. Voiko yritystä kuitenkaan pitää eettisesti vastuullisena? Esimerkiksi Björn Wahlroos sanoo Talouselämän haastattelussa, ettei yrityksillä voi olla moraalia: “Yritykset ajavat omaa asiaansa, ja valtio kantaa yhteiskunnallisen vastuun. Näin toimii markkinatalous.” [Talouselämä, 2008] Katoaako Wahlroosin edustamassa ajattelussa yksittäisen ihmisen vastuunkantovelvollisuus, jollei organisaation tule kokonaisuutenakaan kantaa tuota yhteiskunnallista vastuuta? Esillä on vahva näkemysten ristiriita yrityksen roolista yhteiskunnassa. Entä kuinka Thomsonin ja Schmoldtin esittämä ajatus eettisestä toiminnasta kumpuavasta kaupallisesta menestyksestä asettuu kuvioon?

Päivi Hirvonen [2000] teki kyselytutkimuksen suomalaisille tietotekniikka-alan esimiehille aiheenaan eettisten ohjeistojen tunnettuus ja ohjeistuksen tarpeellisuus. Tulosten mukaan 86% esimiehistä pitää eettisten asioiden ohjaamista tarpeellisena, mutta vain 36% on tietoinen ohjeistojen olemassaolosta. Vaikka vastausprosentti oli melko alhainen 38%, kyseenalaistaa tämä tulos osaltaan ohjeistojen tarpeellisuuden: niitä sanotaan kaivattavan, mutta olemassa olevia ohjeistoja ei kuitenkaan huomioida. Tästä vähäisestä tarpeesta kielii myös se, ettei Tietotekniikan liiton tarjoamaa eettisen pohdinnan avunantopalvelua käytetä.

Ovatko eettiset ohjeistot tehneet yhdestäkään ohjelmistoprojektista eettisesti onnistuneempaa? Thomson ja Schmoldtin [2001] mukaan ohjeistot itsessään eivät tee ihmisistä eettisempiä. Tälle suorasukaiselle väitteelle he eivät kuitenkaan anna perustetta. Ehkä ohjeistojen lukeminen kuitenkin auttaa ihmisiä huomioimaan uusia eettisiä näkökulmia. Tätä mieltä on nimittäin Gotterbarn [1992], joka väittää, että ihmiset haluavat oletusarvoisesti toimia oikein. Sopivan koulutuksen ja kannustuksen avulla heillä olisi siihen mahdollisuus. Gotterbarn myös viittaa tutkimuksiin, joiden mukaan jatkuva keskustelu eettisistä kysymyksistä on tehokkain tapa opettaa etiikkaa.

Tässä on havaittavissa selkeä yhteys Sokrateen ajatukselle siitä, että hyve on tietoa ja ihmiset toimisivat hyveellisesti, kunhan vain tietäisivät kuinka.

Gotterbarn [1992] kuitenkin kritisoi yrityksiä, jotka huomioivat eettisyyden vain selkeäsanaisella eettisellä ohjeistolla. Hänen mukaansa tämä aiheuttaa “tarkistuslista”-tyyppistä lähestymistapaa etiikkaan. Työntekijä saattaa käydä läpi kirjallisen ohjeiston kohdat, todeta että asiat ovat kunnossa ja jättää varsinaisen pohdinnan tekemättä. Tästä voi seurata minimivaatimuskäyttäytyminen: ohjeiston kieltämät asiat ovat kiellettyjä ja kaikki muu onkin sitten sallittua [Mäkinen, 2006, s. 210]. Pelkkä ohjeisto saattaa myös jättää työntekijän hämmentyneeksi tilanteessa, jota ohjeisto ei kata. Tähän liittyy eräs ohjeistojen ongelma: tasapaino yksityiskohtaisuuden ja yleisyyden välillä. Yksityiskohtainenkaan ohjeisto ei pysty kattamaan kaikkia tapauksia, kun taas liian yleisestä ohjeistosta ei välttämättä ole apua yksittäisiä tapauksia pohdittaessa. Niin kauan kuin ohjeistojen kohteena on epämääräinen ”ohjelmistoalan ammattilainen”, pysyvät ohjeistot väkisinkin yleisinä. Ohjelmistoalalla työskentelee paitsi ohjelmoijia, myös testaajia, myyjiä, dokumentoijia ja asiakasneuvojia. Työtehtävien erilaisuus hankaloittaa yhteisten ohjeistojen luontia.

Ohjeistoihin sitoutuminen on yleensä vapaaehtoista, eikä niiden rikkomisesta ole seuraamuksia, sikäli kun pysytään lain sallimissa rajoissa. Gotterbarn [1992] kritisoi tältä osin Yhdysvaltain lainsäädäntöä. Hänen mukaansa huolimatonta tai tietoisen huonolaatuista jälkeä tuottavalle rakennusalan yritykselle voidaan määrätä sanktioita, mutta ohjelmistoalan yritystä ei voida vastaavasti rangaista kiireessä suoritetusta testauksesta – silloinkaan, kun kyseessä on esimerkiksi sydämentahdistimet. Mikäli onnettomuuksia sattuu, tilivelvollinen varmasti löytyy, mutta tämä ei ole sama kuin ennakoiva, onnettomuuksia vähentävä lainsäädäntö.

ACM:n jäsenistölleen suuntaama ohjeisto esittää rangaistuksen rikkomuksista: erotuksen yhdistyksestä. Tämä on varmastikin ainoa rangaistus, jonka yhdistys voi määrätä, mutta edistääkö se ohjeiston noudattamista? Käytännössä ohjeistot ovat kuitenkin suunnattu niille ammattilaisille, jotka kokevat tarvitsevansa neuvoja mahdollisimman eettiseen toimintaan. Heitä, jotka ovat päättäneet toimia epäeettisesti, eivät ohjeistot estä.

Mikä siis lopulta on ohjeistojen tarkoitus, jos niitä ei tunneta, jos niitä ei voida soveltaa käytännön tilanteissa, jos ne eivät vaikuta ohjeistojen lukijaan, eikä niiden rikkomisestakaan seuraa merkittävää rangaistusta? Itsessään ne eivät ole riittävä keino moraalisesti kestävään ohjelmistokehitykseen.

2.5 Ympäristö

On vielä eräs seikka, johon edellä esittämäni periaatteistot eivät eksplisiittisesti puutu, nimittäin ympäristöystävällisyys. Tässä tutkielmassa ei tarkemmin pohdita ilmastonmuutoksen syitä ja seurauksia, mutta oletetaan ympäristöuhan olevan todellinen. Ympäristönsuojelunäkökulmaa käsitellään edellä mainituissa periaatekokoelmissa yllättävän vähän. Rogerson ja Gotterbarn mainitsevat ”luonnollisen ympäristön” yhtenä minkä tahansa projektin sidosryhmänä, mikä tuntuu irralliselta huomiolta ja on vailla tarkempaa analyysia. Myös Tietotekniikan liitto sivuuttaa aiheen mainitsemalla että ”[Ammattilaisen on] otettava huomioon esimerkiksi ihmisoikeudet, ympäristön suojelu, lainsäädäntö ja tekijänoikeudet.” Kenties ympäristönsuojelun rooli eettisessä keskustelussa on kasvanut vasta viime vuosina, ohjeistojen kirjoittamisen jälkeen?

Mitkä ovat ympäristönsuojelun kannalta oikeita yritysratkaisuja ja missä määrin ohjelmistoalan yritys eroaa ympäristönsuojelullisilta näkökulmiltaan muiden alojen yrityksistä? Ensimmäiseen kysymykseen vastauksia voi etsiä ympäristönsuojelulain ensimmäisessä pykälässä [Ympäristönsuojelulaki, 1§] esitetyistä tavoitteista:

1. ehkäistä ympäristön pilaantumista sekä poistaa ja vähentää pilaantumisesta aiheutuvia vahinkoja
2. turvata terveellinen ja viihtyisä sekä luonnontaloudellisesti kestävä ja monimuotoinen ympäristö
3. ehkäistä jätteiden syntyä ja haitallisia vaikutuksia
4. tehostaa ympäristöä pilaavan toiminnan vaikutusten arviointia ja huomioon ottamista kokonaisuutena
5. parantaa kansalaisten mahdollisuuksia vaikuttaa ympäristöä koskevaan päätöksentekoon
6. edistää luonnonvarojen kestävää käyttöä
7. torjua ilmastonmuutosta ja tukea muuten kestävää kehitystä.

Ohjelmistoalan yrityksillä on tyypillisesti hyvät lähtökohdat edellä kuvattujen tavoitteiden mukaiseen ympäristöystävälliseen toimintaan: työ on pääosin palvelu- ja tietotyötä, siinä ei käsitellä suuria määriä raaka-aineita tai myrkyllisiä kemikaaleja, eivätkä käytetyt koneet yleensä ole tietokoneita suurempia. Alan yrityksien ympäristövaikutukset ovatkin jokseenkin välillisiä – sähkön ja toimistotarvikkeiden

kulutusta, tietokone- ja muun laitteiston valmistuksesta aiheutuvia vaikutuksia sekä esimerkiksi palavereista johtuvaan matkustamiseen liittyviä vaikutuksia.

Ohjelmistoalalla sähkönkulutus on kenties selkeimmin havaittavissa oleva ympäristörasite, joten sitä on helpoin tarkastella: vähäisempi kulutus aiheuttaa vähäisempiä haittavaikutuksia. Steven T. Moeller [2002] esittelee kaksi erilaista tapaa energiansäästöön: energiatehokkuuden *(energy efficiency)* ja energiankäytön vähentämisen *(energy curtailment)*. Tehokkuuden hän määrittelee energiansäästöksi, joka ei vähennä energialla tuotettavan hyödykkeen tai palvelun määrää, kun taas energiankäytön vähentäminen viittaa toimintaan, joka vähentää saatua hyödykettä – ajetaan vähemmän autoa, säädetään termostaattia viileämmälle tai tulostetaan dokumentteja valikoidummin.

Dan Pritchett, entinen eBayn työntekijä, on sanonut, että tehottomat ohjelmointiratkaisut lisäävät eBayn energiankulutusta 25-30%. [Bate, 2007] Projektinjohtaja voikin parantaa kehitettävän sovelluksen energiatehokkuutta sijoittamalla ohjelmointiresursseja koodin tehokkuuden optimointiin. Tämä päätös on kuitenkin erikseen tehtävä, ja mikäli vaatimusmäärittelyssä ei ole erikseen mainittu energiatehokkuutta, on kynnys näennäisen ylimääräiseen resurssienkäyttöön suuri tai jopa ylitsepääsemätön, sillä se voitaneen jopa tulkita sopimuksenvastaiseksi, “turhaksi”, toiminnaksi.

On kuitenkin huomattava, että pelkästään jo uuden ohjelmiston kehitys vanhojen järjestelmien tilalle saattaa parantaa tehtävien prosessien energiatehokkuutta. Energiatehokkuuden mahdollinen paraneminen uudella sovelluksella on kuitenkin tapauskohtaista, eikä vielä itsessään takaa uuden sovelluksen olevan mahdollisimman energiatehokas ratkaisu kyseessä olevien prosessien suorittamiseen.

Muitakin ympäristönsuojelulaissa mainittuja tavoitteita voidaan huomioida: jätteen syntyyn ja luonnonvarojen kestävään käyttöön voidaan varmasti vaikuttaa yritystason päätöksillä. Useimmat tällaiset tavoitteiden saavuttamiseksi käytettävät keinot ovat kuitenkin yhteisiä kaikkien alojen yrityksille, eivätkä siis spesifejä ohjelmistoalalle, toisin kuin ohjelmistojen algoritminen tehokkuus. Yleisiä keinoja toimistojen energiansäästöön on esittänyt esimerkiksi Maailman Luonnonsäätiö, joka on kehittänyt aihetta käsittelevän Green Office -järjestelmän. [WWF]

3 Luottamus

Eettiset ohjeistot lähestyvät ohjelmiston eettisyyttä ohjelmistokehittäjän näkökulmasta, mutta ohjelmistoa voidaan lähestyä myös sen käyttäjän näkökulmasta. Koska ohjelmistot luodaan käyttäjilleen, on tämä ohjelmisto-käyttäjä -suhde määrittelevä tekijä projektin onnistumisessa. Käyttäjä ei aina ole totuttu työpöytätietokoneen ääressä istuva työläinen, vaan esimerkiksi ohjelmistoa välillisesti suurten automatisoitujen koneiden tai järjestelmien kautta hyödyntävä henkilö. Tämä riippuu ympäröivän systeemin ohjelmistolle asettamista tavoitteista. Ohjelmistoprojektin eettinen onnistuminen on kyseenalaista, jos käyttäjä epäilee ohjelmiston toimivan väärin, vahingollisesti tai muutoin vastoin käyttäjän tahtoa. Eettiset ohjeistot kehottavat korkeaan sovellustekniseen laatuun, joka ei kuitenkaan itsessään riitä, jos käyttäjän todella halutaan luottavan ohjelmistoon.

Ohjelmistot ovat viime vuosina siirtyneet yhä suuremmassa määrin Internetiin selainkäyttöisiksi sovelluksiksi. Verkossa koetussa ohjelmisto-käyttäjä -suhteessa ei ole sellaisia välikäsiä kuin fyysinen kauppa tai opetusinstituutio, jotka väistämättä vaikuttaisivat ohjelmiston uskottavuuteen ja luottamussuhteeseen. Tässä tutkielmassa on siis perusteltua rajata luottamuksen tutkiminen vain online-ympäristöön, jossa ohjelmistoyritys on selkeimmin itse vastuussa luottamussuhteen luomisesta, eikä luottamussuhdetta ole implisiittisesti rakentamassa kolmansia osapuolia.

Yrityksen ja asiakkaan suhde on Internetin käytön yleistymisen myötä muuttunut voimakkaasti. Asioiminen ei rajoitu tiettyihin liikkeisiin maantieteellisen sijainnin perusteella, joten asiakkaalle on auennut monia uusia kauppapaikkoja. Yrityksille tämä näkyy monina mahdollisina asiakkaina, jotka tosin täytyy saada asiakkaiksi eri keinoin kuin fyysisessä maailmassa.

Tämä muutos on nostanut esille myös sellaisia luottamuksen kysymyksiä, joiden pohtimiseen ei aiemmin ole ollut tarvetta. Erityisesti kun asiakkaan ja yrityksen välinen kommunikaatio kaventuu pelkäksi verkkosivukontaktiksi, moni asiakas joutuu pohtimaan riittäviä ehtoja luottamukselle. Kenelle voi luovuttaa henkilötietonsa? Entä luottokortin numeron?

Corritoren ja muiden [2001] mukaan tähän asti tehty tutkimus luottamuksesta koskee ennen kaikkea offline-luottamusta, joten online-luottamusta pohdittaessa voimme käyttää tehtyjä tutkimuksia vain suuntaviivoina. Myskja [2008] väittää, että kun kommunikaatiosta puuttuu fyysinen ulottuvuus, suorat henkilöiden väliset kontaktit ja

elekieli, meistä tulee aiempaa haavoittuvaisempia ja alttiimpia huijaukselle. Kehollisen läheisyyden puuttuessa ihmiset ovat myös epäröivämpiä luottamaan kommunikaation toiseen osapuoleen.

Mitä luottamus oikeastaan on? Joidenkin mukaan se on asenne, ominaisuus tai suhde, joidenkin mielestä luonnollinen tunne tai uskomus, joidenkin mielestä tietoinen valinta. Jos se onkin tietoinen valinta, niin siitä ei ole yksimielisyyttä, onko se johdonmukainen päätelmä saatavilla olevasta aineistosta vai kenties toimintamalli, joka tyystin sivuttaa järkeilyn. [Koehn, 2003].

Luottamuksen koetaan ilmentävän tiettyä varmuutta toisen tahon toiminnasta, mutta se sisältää enemmänkin kuin pelkän varmuuden. Koehn antaa esimerkiksi auringon: vaikka meillä onkin syytä uskoa sen nousevan jälleen huomenna, olisi väkinäistä väittää, että luotamme aurinkoon. Kyse ei siten ole vain siitä, että auringonnousu on vääjäämätön fyysinen ilmiö; voimme myös olla vakuuttuneita, että pimeällä kujalla kohtaamamme vihainen rikollisjoukkio aiheuttaa ongelmia, muttemme missään nimessä luota heihin. Luottamuksessa on kyse ennemminkin tietynlaisesta varmuudesta luottamuksen kohteen hyvästä tahdosta luottajaa kohtaan.

Tuo varmuus saattaa pohjautua aiempiin kokemuksiin luotetusta, mutta Koehn kiistää ajatuksen siitä, että luottajan ja luotettavan välillä tulisi välttämättä olla jonkinlainen valmis hyvistä kokemuksista muodostunut suhde tai kontakti, sillä joskus luotamme nimenomaan luodaksemme tuon suhteen.

Yksi luottamuksen tarkoituksista on mahdollistaa monimutkaisuuden vähentäminen ihmismielestä. Luottamalla toiseen tahoon, esimerkiksi yritykseen, asiakkaan ei täydy eksplisiittisesti pohtia kaikkea kommunikoimaansa informaatiota. [Corritore et al., 2001]. Ohjelmistokontekstissa tällainen kognitiivisen kuorman vähentäminen tekee ohjelmiston käyttökokemuksesta miellyttävämmän, mikä tarkoittaa tuotteen kohonnutta laatua. Luottamus myös yksinkertaistaa vaihtoehtojen määrää toiminnassamme ja ilman luottamusta yhteiskunnalle koituisikin valtavia sosiaalisia ja taloudellisia kustannuksia [Myskja, 2008]. Eräs kiintoisa piirre luottamuksessa on sen sidonnaisuus aikaan ja paikkaan. Mikäli voisimme jatkuvasti tarkkailla toisen toimia, ei luottamukselle olisi tarvetta. Samoin meidän ei tarvitsisi luottaa järjestelmään, jonka toimintaperiaate olisi täysin tunnettu ja ymmärretty. [Giddens, 1991]

3.1 Luottamuksen tyypit

Koska on vaikeaa yleisesti määritellä luottamusta, Koehn [2003] määrittelee neljä erilaista luottamuksen tyyppiä. Ne ovat

1. päämääräkeskeinen
2. laskelmoiva
3. tietopohjainen
4. kunnioituspohjainen.

Luottamustyypit kuvaavat luottamusta eri näkökulmista ja toisaalta luottamusta sen kehittyessä ajallisesti.

Kun kaksi henkilöä kokee jakavansa yhteisen päämäärän, ilmenee päämääräkeskeistä luottamusta. Päämäärä itse saattaa olla eettisesti hyvä tai paha, ja Koehn esittää terroristien tai huligaanienkin luottavan toisiinsa. Osapuolilla ei ole kiinnostusta toistensa luonteisiin tai muihin piirteisiin, vaan heille päämäärä on tärkeintä. Keskinäistä valehtelua tai manipulointia ei koeta vääräksi, jos se edistää päämäärää. Tällä luottamuksella on suuri tunneperäinen komponentti, joka ilmenee retoriikkana tai propagandana.

Laskelmoiva luottamus pyrkii arvioimaan etukäteen, kuinka luotettu tulee toimimaan. Onko osapuolella näyttöä lupausten pitämisestä tai muuten hyvä maine? Luottaja laskee mahdolliset hyödyt ja riskit, jotka luottamuksesta aiheutuu, ja mikäli hyötyä on enemmän kuin haittaa, luottaja päättää luottaa.

Tällaista luottamusta esiintyy ennen kaikkea yritysten välillä, tai kun osapuolet ovat toisilleen tuntemattomia. Mukana luottamussuhteessa saattaa olla sopimuksia, jolloin luottamus osapuolten välillä välittyy lainsäädännön ja oikeuslaitoksen kautta. Kun osapuolet luottavat toisiinsa ulkopuolisen järjestelmän kautta, luottamusta voidaan pitää vähäisempänä. Keskinäinen luottamus on kuitenkin olemassa ja ilmenee tapauksissa, joita ei täysin eksplisiittisesti ole määritelty sopimuksessa.

Kun ihmiset tuntevat toisensa ja kommunikoivat usein, nousee esiin tietopohjainen luottamus. Luottamus rakentuu kun osapuolet viihtyvät toistensa seurassa, eikä tällaiseen luottamukseen sisälly oletusta minkäänlaisesta hyödyn keräämisestä. Kyseessä onkin enemmän ystävyys- kuin hyötysuhde, mutta luottamustyyppi saattaa silti ilmetä myös yritysmaailmassa, jossa eri yritykset tekevät yhteistyötä useiden

vuosien ajan. Tietopohjainen luottamus on pitkäjänteisempää kuin laskelmoiva luottamus ja kestää tätä suurempia muutoksia osapuolten asemassa.

Kunnioituspohjaista luottamusta muodostuu osapuolten jakaessa samanlaisen mielenkiinnon viisauteen ja hyvyyteen ja kun on halua keskustella ja ymmärtää toista. Osapuolet kunnioittavat toisiaan, eikä heillä ole tarvetta hyödyntää toisiaan. Ajan myötä osapuolten mielipiteet saattavat jopa alkaa muistuttaa toisiaan. Tällainen luottamuksen tyyppi on yleistä hyvien ystävien kesken. Osapuolilla ei ole tarvetta kyseenalaistaa toistensa ajattelua, vaan he voivat luottaa siihen, että ristiriitatilanteissa kumpikin hakee yhteistä etua. Tällainen luottamus sallii jopa tehtyjen sopimusten rikkomisen – luottaja pystyy uskomaan, että luotettu on harkinnut sopimuksen rikkomista kyllin tarkasti ja toiminut niin vain koska se on ollut välttämätöntä yhteisen edun kannalta. Niinpä emme koe luottamusta rikotun, jos ystävä myöhästyy sovitusta tapaamisesta hyvän syyn takia. Verkkokauppaympäristössä tällainen luottamus saattaisi mahdollistaa korvaavan tuotteen lähettämisen asiakkaalle tilatun asemesta.

Koska kunnioituspohjainen luottamus on monella tapaa kestävintä, Koehn [2003] suosittelee verkossa toimivien yritysten toimivan tavoilla, jotka rakentavat tämän tyyppistä luottamusta.

3.2 Luottamusta online-sovellukseen

Koehn [2003] luettelee useita keinoja luottamuksen rakentamiseen verkkopalveluissa, vaikkakin useimmat hänen mainitsemista keinoista soveltuvat lähinnä verkkokauppoihin. Muutoinkin verkkoluottamusta käsittelevä kirjallisuus keskittyy ensisijaisesti verkkokauppoihin, mutta myös terveysjärjestelmiin. Tämä on sikäli ymmärrettävää, että edellinen on helppo esimerkki tilanteesta, jonka useimmat tuntevat, ja jälkimmäinen taasen esimerkki aiheesta, jossa luottamus järjestelmiin on kaikki kaikessa – kyseessä on käyttäjien henki.

Koehn mainitsee verkkokaupan selkeimmät vaatimukset, kuten että kaupattujen tuotteiden kuvat ja kuvaukset vastaavat todellisuutta, ja että kieliasu ja oikeinkirjoitus ovat kunnossa. Hän myös esittää joitain sosiaaliseen mediaan liitettyjä ajatuksia, kuten että palvelun käyttäjien luoma sisältö, esimerkiksi tuotteiden arvostelut, parantavat palvelun luotettavuutta.

Muita Koehnin esittämiä, erityisesti kunnioituspohjaista luottamusta parantavia, tekijöitä ovat nopeasti toimiva sivusto, hillitty sähköpostien lähettäminen asiakkaalle, kolmannen, jo luotetun tahon, sisällyttäminen sivuille, sivuston automaattinen

muokkaaminen käyttäjän mieltymysten mukaan, kirjoitusvirheiden huomaaminen käyttäjän syötteistä, mahdollisuus tuotepalautuksiin, yrityksen yhteystietojen esittely ja kaikin puolin hyvä palvelu myös verkkosivuston ulkopuolella. Koehn korostaa luottamuksen ihmistenvälisyyttä: käyttäjä haluaa luoda suhteen ihmisiin, ei verkkosivustoon. [Koehn, 2003]

Nämä ovat kaikki pieniä yksityiskohtia, jotka kuitenkin väärin tehtyinä saavat käyttäjän epäileväksi.

3.3 Eettiseen organisaatioon luottaminen

Siinä missä ohjelmiston herättämä luottamus voidaan tulkita osatekijäksi eettisessä onnistumisessa, toimii luottamuksen ja etiikan suhde myös vastakkaiseen suuntaan. Yksityiskohtien lisäksi suuremmatkin linjat vaikuttavat sovelluksen luotettavuuteen. Kyse on siitä, että verkkosivuston taakse halutaan yleisemminkin nähdä: moni tutkija on identifioinut palveluntarjoajan maineen tärkeäksi luottamusta rakentavaksi tekijäksi. Yang ja muut [2007] väittävät, että yrityksen mainetta voidaan parantaa terveellä organisaatiokulttuurilla. Tämä tarkoittaa siis eettisesti oikean toiminnan muuntumista luottamukseksi yhtiöön ja sen tuottamiin palveluihin.

Eettisesti oikeaa toimintaa seuraava kaupallinen menestys vastaa kohdassa 2.4 esiteltyä Thomsonin ja Schmoldtin [2001] ajatusta siitä, että eettinen ohjelmistoprojekti onnistuu suuremmalla todennäköisyydellä kuin epäeettinen. Samoilla linjoilla on merkittävä yritysetiikan tutkija Manuel Velasquez: ”Näyttää siltä, että etiikka ei heikennä tulosta vaan pikemminkin vahvistaa sitä”. [Aaltonen ja Junkkari, 2003, s. 49]

Organisaatiokulttuuria voidaan yrittää muokata paremmin eettistä toimintaa tukevaksi esimerkiksi työntekijöiden käyttäytymistä ohjaavilla ohjeilla, joiden mahdollista toimivuutta on käsitelty kohdassa 2.4. Toisena esimerkkikeinona yhtiön maineen parantamiseksi Yang ja muut [2007] esittelevät tavan tehdä henkilöstölle tausta- ja henkilöarvionnit ja korottaa heistä osa 'luotetun henkilöstön' asemaan. Tälle osalle henkilöstöstä annetaan suuremmat valtuudet katsoa ja muuttaa yhtiön hallussa olevaa informaatiota. Yang ja muut kuitenkin huomauttavat, että juuri nämä 'luotettuun henkilöstöön' kuuluvat ovat ajoittain olleet osallisina suuriin tietoturva- ja yksityisyysrikkomuksiin. Tämä indikoi heidän mielestään poikkeamaa yrityksen ja yksilön arvojen välillä. Vaikka he eivät tämän pohjalta kritisoi 'luotettu henkilöstö' -ajattelua, on tämä ongelma nähtävissä ristiriitana, joka tekee henkilöstön erottelun

luotettuun ja epäluotettuun hyödyttömäksi. Parempina keinoina hyvän organisaatiokulttuurin rakentamiseen he ehdottavat seuraavia:

- Johtoryhmien rakentamista organisaation eri tasoille. Kukin ryhmä pyrkii muuntamaan organisaation arvot päivittäiseksi toiminnaksi.
- Sessioita, joissa kerrotaan ja muistutetaan organisaation arvoista uusille ja vanhoille työntekijöille.
- Organisaation nykytilan selvittämistä ja erinomaisten työntekijöiden löytämistä ja kouluttamista.
- Roolimallikoulutusta korkeimmalle johdolle, jotta he voivat johtaa tarvittavat muutokset organisaatioon.

Nämä neuvot eivät ole kovinkaan suoraviivaisia, mutta ehkä juuri siinä on niiden voima. Niitä hyödyntäessä vaaditaan todellista harkintaa ja työtä, eikä organisaatiokulttuurin muuttaminen terveemmäksi voinekaan olla helppoa.

Millaista yrityskulttuuria itseasiassa haetaan, kun haetaan ”tervettä organisaatiokulttuuria”? Yhdenlaisen vastauksen tarjoavat Aaltonen ja Junkkari [2001, s. 115], jotka listaavat yhdeksän yritysorganisaation piirrettä, jotka tukevat eettistä toimintaa:

- organisaation, vastuiden ja roolien selkeä määrittely
- avoimen keskustelun mahdollisuus
- haastavat tavoitteet, jotka on mahdollista saavuttaa
- kahdensuuntainen viestintä organisaation kaikilla tasoilla
- lupa esittää poikkeavatkin kannanotot ja ideat, varsinkin omaa työtä koskevat
- käytössä riittävät tiedot ja muut resurssit
- jatkuva oppiminen ja koulutus
- oikeudenmukaiset palkkio- ja mittausjärjestelmät, joissa arvostetaan pitkän aikavälin tuloksia
- vastuun ottamisen kulttuuri.

Aaltonen ja Junkkari eivät kerro yksityiskohtaisia tapoja näiden kulttuuritavoitteiden saavuttamiseksi. Monet niistä ovat suoraviivaisesti siirrettävissä ohjelmistoprojektin

aikaisiksi projektiryhmän toimintatavoiksi, mutta osaan on puututtava organisaatiotasolla.

4 Ohjelmistokehitysmallit

Ratkaistaessa kysymystä siitä, kuinka ohjelmistoja voidaan tehdä eettisesti valveutuneesti, on tarkasteltava, kuinka ohjelmistoja yleensä tehdään. Ohjelmiston matka ideasta valmiiksi tuotteeksi on pitkä ja moniosainen prosessi. Tähän prosessiin luovat vakautta, kontrollia ja järjestystä erilaiset kehitysmallit. Ilman kehitysmallia ohjelmiston kehitys saattaa käydä kaoottiseksi ja ohjelmistoprojektin epäonnistumisen riski kasvaa. Toimiva kehitysmalli varmistaa ohjelmiston korkean laadun ja aikataulussa pysymisen sekä mahdollistaa ohjelmiston pitkäikäisyyden ylläpidon ja jatkokehityksen näkökulmasta. [Pressman, 2001, p. 19]

Kullakin ohjelmistoprojektilla on toki omat erityispiirteensä, eikä yhtä kehitysprosessia voi toistaa täysin samanlaisena projektista toiseen. Jokaisessa prosessissa täytyy kuitenkin vastata tiettyihin, aina toistuviin ongelmiin, jotka voidaan jakaa kolmeen osaan [Pressman, 2001, p. 22]:

1. määrittelyvaihe
2. toteutusvaihe
3. ylläpitovaihe.

Ensimmäisessä vaiheessa määritellään, mitä oikeastaan ollaan tekemässä: tunnistetaan käsiteltävä informaatio, etsitään ohjelmiston tärkeimmät toiminnallisuudet, huomioidaan liittymät muihin järjestelmiin, arvioidaan suunnitelurajoitteet ja määritellään, millainen ohjelmisto on onnistunut.

Kehitysmallin toteutusvaihe ratkaisee, kuinka ohjelmisto varsinaisesti toteutetaan, lähtien tietomalleista ja algoritmisista yksityiskohdista siihen, kuinka ohjelmisto testataan. Tämä vaihe pitää sisällään ohjelmiston teknisen suunnittelun, lähdekoodin tuottamisen ja kokonaisuuden testauksen.

Ylläpitovaihe keskittyy muutoksenhallintaan: virheiden korjaukseen, muuttuvaan ympäristöön mukautumiseen ja uusien ominaisuuksien lisäämiseen. Tämän vaiheen sisällä toistuu määrittely- ja toteutusvaiheet, mutta lähtökohta on olemassaoleva ohjelmisto. Ylläpitoon kuuluu muutakin kuin ohjelmointia: esimerkiksi asiakkaille tarjottu tekninen tuki on osa laadukasta ohjelmistokokonaisuutta.

Näiden vaiheiden lisäksi on joukko tehtäviä, joita suoritetaan jatkuvasti, projektin vaiheesta riippumatta. Tällaisia ovat esimerkiksi muodolliset katselmoinnit, laadunvarmistus, dokumentointi ja riskinhallinta.

Vuosien saatossa on suunniteltu monenlaisia kehitysmalleja. Saadut kokemukset ovat ohjanneet kehitysprosesseja jähmeästi käyttäytyvistä yhä joustavampaan suuntaan. Esittelen seuraavaksi joitain tärkeimpiä malleja ja niiden mahdollista vaikutusta ohjelmistoprojektin aikana kohdattaviin eettisiin ongelmiin.

4.1 Vesiputousmalli

Vesiputousmalli on saanut vertauskuvallisen nimensä prosessin yksisuuntaisuudesta: malli etenee vaihe kerrallaan ja kehityksen tulos on valmis prosessin lopussa. Vesiputousmalli on tietyllä tapaa kehitysmallien kantaisä, yksinkertainen, mutta sinänsä toimiva malli, jota myöhemmin kehitetyt mallit pyrkivät parantamaan.

Mallista on esitetty joitain keskenään hieman poikkeavia versioita, mutta esimerkiksi Pressmanin [2001] esittelemässä versiossa on viisi vaihetta:

1. vaatimusmäärittely
2. suunnittelu
3. koodintuotanto
4. testaus
5. tuki.

Vaatimusmäärittely on luvun alussa esitellyn määrittelyvaiheen ilmentymä. Se on prosessi, jossa mahdollisimman tarkasti hahmotetaan ratkaistava ongelma ja kerätään lista ohjelmiston toiminnoista, käyttäytymissäännöistä, tehokkuudesta ja käyttöliittymäelementeistä.

Toteutusvaihe on tässä mallissa jaettu kolmeen osaan: suunnittelu-, koodintuotanto-, ja testausvaiheeseen. Suunnitteluvaiheessa luodaan tekninen arkkitehtuuri, tietorakenteet, käyttöliittymäsuunnitelmat ja algoritmiset yksityiskohdat, joita voidaan arvioida ja muokata ennen varsinaista koodintuotannon alkamista. Kun suunnitelmaan ollaan tyytyväisiä, voidaan siirtyä koodintuotantoon, joka merkitsee suunnitelman muuntamista tietokoneen ymmärtämään muotoon. Koska suunnitelma on tarkasti luotu, tämä vaihe on teoreettisesti hyvinkin nopea ja mekaaninen. Testausvaiheessa varmistetaan, että tuotettu ohjelmisto toimii vaatimusmäärittelyn mukaisella tavalla.

Tuki pitää sisällään paitsi asiakastuen, myös ohjelmiston päivittämisen ja ylläpidon. Mahdollisia muutoksia tehdessä käydään läpi kaikki kehitysvaiheet.

Vesiputousmallia pidetään usein vanhanaikaisena, mutta se on varteenotettava vaihtoehto, kun ohjelmiston vaatimukset ovat kyllin selkeät. Käytännössä näin ei kuitenkaan yleensä ole, ja vaatimusmäärittelyvaiheessa tehdyt virhearviot saatetaan huomata vasta myöhäisessä vaiheessa ohjelmiston valmistuttua. Vaikka malliin kuuluukin mahdollisuus uusien versioiden tekemiseen, on tämä mahdollisuus epäsuora keino iteroida kohti parempaa tuotetta. Ongelmat johtuvat pääosin siitä, että asiakkaan tai ohjelmistotuottajan on usein erittäin vaikeaa määritellä kaikkia vaatimuksia heti projektin alussa, vaikka tämä malli sitä vaatiikin. Heikkouksistaan huolimatta tämäkin kehitysmalli on kuitenkin parempi vaihtoehto kuin suunnittelematon ohjelmistokehitys.[Pressman, 2001, pp. 28-30]

4.2 Prototyyppimalli

Koska asiakas tietää projektin alussa yleensä vain suurpiirteiset vaatimukset ohjelmistolle, on kehitetty tapoja mahdollistaa vaatimusten tarkentaminen projektin edetessä. Yksi tällainen tapa on prototyyppimalli, joka alkaa samoin kuin vesiputousmalli: vaatimusmäärittelyllä. Nyt ei kuitenkaan pyritä tarkasti keräämään kaikkia vaatimuksia, vaan tunnistamaan alueet, joilla tarvitaan lisää tarkennusta. Näitä alueita voi olla esimerkiksi käyttöliittymä, algoritminen tehokkuus tai integraatio tekniseen alustaan.

Lisätyötä tarvitsevien alueiden tunnistamisen jälkeen tehdään nopea suunnittelu, joka kattaa vain tarpeellisimmat asiat ongelma-alueita esittelevän prototyypin valmistamiseksi. Prototyyppi luodaan ja esitellään asiakkaalle, joka arvioi sen toimivuutta ja tarkentaa vaatimuksia. Tarkennusten mukaan luodaan uusia prototyyppejä, jotka paremmin toteuttavat asiakkaan toiveen ja samalla auttavat ohjelmistokehittäjiä hahmottamaan, kuinka ohjelman sisäinen logiikka tulisi parhaiten toteuttaa. Kun vaatimukset ovat prototyypin avulla tarkentuneet, voidaan aloittaa varsinaisen ohjelmiston toteuttaminen.

Tässä tulee vastaan prototyyppimallin suurin ongelma: voidaanko prototyyppiä käyttää lopullisen ohjelmiston luomiseen? Se on usein tehty mahdollisimman nopeasti ja vähäisellä suunnittelulla, kenties käyttäen työkaluja, jotka eivät sovellu tuotantokäyttöön. Täten se ei välttämättä sovellu jatkokehitettäväksi valmiiksi ohjelmistoksi. Asiakas kuitenkin helposti näkee sen toimivana, lähes valmiina,

sovelluksena ja saattaa vaatia “pieniä korjauksia”, joilla prototyypistä saataisiin nopeasti valmis tuote.

Jos prototyypin tarkoituksesta kuitenkin sovitaan kyllin selkeästi ennen kehityksen aloittamista, ei tällaisia ongelmia pitäisi ilmetä, ja prototyyppi auttaa luomaan laadukkaamman ja toivotummanlaisen ohjelmiston. On myös tapauksia, joissa prototyyppiä voidaan onnistuneesti jatkokehittää valmiiksi ohjelmistoksi, tai vähintäänkin hyödyntää joitain osia sitä varten kirjoitetusta lähdekoodista. [Pressman, 2001, s. 31-32]

4.3 Evolutionääriset mallit

Prototyyppimallissa käytetään verrattain paljon aikaa prototyypin luomiseen, ottaen huomioon, että se joudutaan kuitenkin heittämään pois [Gilb, 1981]. Toisaalta vesiputousmallissa saatetaan usein havaita, ettei lopullinen tuote toimi parhaalla mahdollisella tavalla, vaikka toki vaatimusmäärittelyn ehdot täyttääkin. Usein ohjelmistolle on myös niin kiireellinen tarve, että on parempi julkaista toiminnallisuudeltaan rajoitettu versio kuin odottaa täyttä valmistumista ja siirtää julkaisua pitkälle tulevaisuuteen.

Näihin ongelmiin vastaavat evolutionääriset mallit kuten inkrementaalinen malli. Siinä ohjelmisto jaetaan pienemmiksi kokonaisuuksiksi eli inkrementeiksi, jotka toteutetaan peräkkäin. Ensimmäiseen inkrementtiin kuuluu ohjelmiston ydin ja seuraaviin lisäominaisuuksia ja uusia toiminnallisuuksia. Kunkin inkrementin valmistuttua ohjelmisto on tietyllä tapaa valmis, esimerkiksi testausta varten. Ryhdyttäessä toteuttamaan kutakin uutta inkrementtiä voidaan siihen liittyviä vaatimuksia tarkentaa tai muuttaa sen kokemuksen pohjalta, jota edellisen inkrementin lopputuloksen testaamisesta ollaan saatu. [Pressman, 2001, p. 34]

Kokonaisuuteen verrattuna inkrementit ovat yksinkertaisempia kontrolloida ja hallita pienemmästä koostaan johtuen. Myös ohjelmistokehittäjät ovat motivoituneempia, kun projekti etenee tasaisin väliajoin ja jo aikaisessa vaiheessa muodostuva esiversio tyydyttää asiakkaankin seurannanhalua. Asiakas ei myöskään projektin alussa vaadi näennäisen oleellisia, mutta todellisuudessa turhiksi osoittautuvia ominaisuuksia, tietäessään, että nämä ominaisuudet voidaan kyllä lisätä myöhäisemmässä vaiheessa projektia.

Monista hyvistä puolista huolimatta inkrementaalisessa mallissa on ongelmansakin. Jos ohjelmiston jako inkrementteihin on ollut huolimatonta tai kokonaisuus muuten paisuu,

voivat varhaisessa vaiheessa toteutetut ohjelmistotekniset ratkaisut osoittautua riittämättömiksi esimerkiksi skaalautuvuudeltaan tai uudelleenkäytettävyydeltään. Tästä seuraa hidasta ohjelmiston uudelleenstrukturointia ja -ohjelmointia, joka voi olla välttämätöntä myös silloin, kun myöhemmät inkrementit ovat ristiriidassa aikaisempien inkrementtien vaatimusten kanssa [Hughes and Cotterell, 2006, s. 83]. Tämä kaikki voi johtua eräänlaisesta kokonaiskuvan katoamisesta: ohjelmistokehittäjillä voi kulua resursseja sellaiseen väliversioiden valmisteluun, joista ei lopputuloksen kannalta olekaan hyötyä.

4.4 Ketterät menetelmät

Aiemmin kuvaamani kehitysmallit ovat verrattain vanhoja ja koeteltuja. Uusia malleja on kuitenkin kehitetty, ja tällä hetkellä kehitysmallien kehitys etenee suuntaan, joka korostaa yhä enemmän joustavuutta ja tiedostaa, että usein suunnitelmat muuttuvat projektin edetessä. On syntynyt joukko kehitysmalleja, joita kutsutaan yhteisnimellä ”ketterät menetelmät” (agile methods), ja jollaisia ovat esimerkiksi Extreme Programming [Beck, 1999], Scrum [Schwaber and Sutherland, 2010], Feature Driven Development [Palmer ja Felsing, 2002], The Rational Unified Process [Rational, 1998] ja Adaptive Software Development [Highsmith, 2000]. Useiden ketterien kehitysmallien edustajat järjestivät yhteisen konferenssin vuonna 2001 ja julkaisivat manifestin, jossa he julistivat yhteistä ideologiaansa. Manifesti esittää neljä vastakkainasettelua ja ketterien menetelmien kannattajien preferenssit niiden suhteen [Agile Manifesto, 2001]:

1. yksilöt ja vuorovaikutus ovat tärkeämpiä kuin prosessit ja työkalut
2. toimiva ohjelmisto on tärkeämpää kuin kattava dokumentaatio
3. asiakasyhteistyö on tärkeämpää kuin sopimusneuvottelu
4. muutokseen reagointi on tärkeämpää kuin suunnitelman seuraaminen.

Laajemmiksi avattuina iskulauseet merkitsevät ensinnäkin yhteisöllisen tunnelman rakentamista ohjelmistokehittäjien välille: tiiviitä ryhmiä, lähekkäisiä työympäristöjä ja ryhmähenkeä. Toisekseen korostetaan jatkuvaa toimivaksi testatun ohjelmiston julkaisemista tasaisin väliajoin. Tavoitteena on niin korkeatasoinen ohjelmakoodi, että tarve resursseja kuluttavalle dokumentoinnille vähenee. Kolmanneksi, asiakkuussuhteissa suositaan sopimuksia, joissa korostuu yhteistyö kehittäjien ja asiakkaan välillä. Välittömästi alkava ohjelmistoversioiden tuottaminen vähentää asiakkaan riskiä jäädä ilman toimitettua ohjelmistoa. Neljänneksi, ohjelmistokehittäjien

ja asiakkaan muodostaman kehitysryhmän tulee olla kykenevä, oikeutettu ja valmis tekemään muutoksia aiemmin tehtyihin päätöksiin kehitysprosessin edetessä.

Näiden kehittäjien itse julistamien tavoitteiden tai lisäksi ketteristä menetelmistä on löydetty muitakin yhteisiä piirteitä. Abrahamssonin ja muiden [2002] mukaan ketterät menetelmät on suunniteltu:

1. tuottamaan nopeasti ensimmäisen julkaistavan version, jolloin kehitysryhmä saa välitöntä palautetta varhaisessa vaiheessa
2. kehittämään yksinkertaisia ratkaisuja, jolloin on vähemmän muutettavaa ja muutosten teko on nopeampaa
3. parantamaan ohjelmistosuunnittelun tasoa jatkuvasti, jolloin seuraavan iteraation suoritus on yhä halvempaa
4. testaamaan jatkuvasti, jolloin virheelliset toiminnot löydetään varhaisemmassa, halvemmalla korjattavassa vaiheessa.

Yhdistävinä tekijöinä ketterillä menetelmillä on myös nopea inkrementaalinen sykli, yhden syklin ollessa esimerkiksi kahdesta kuuteen viikkoa, menetelmien soveltuvuus erityisesti pieniin, alle 10 hengen, kehitysryhmiin ja kehitysryhmän sisäisen vuorovaikutuksen tärkeyden korostaminen. [Abrahamsson et al., 2002; Agile Manifesto, 2001].

Eksplisiittinen keskustelu kuuluu kiinteästi esimerkiksi Scrum-malliin, jossa ohjelmistokehittäjien muodostama työryhmä pitää päivittäin noin viidentoista minuutin mittaisen aamupalaverin. Palaverissa jokainen osallistuja kertoo, mitä on tehnyt edellisen kokouksen jälkeen, mitä aikoo tehdä seuraavaan kokoukseen mennessä ja mikä estää tekemästä työtä mahdollisimman tehokkaasti. Malliin kuuluu myös inkrementtien, Scrum-termeillä sprinttien, välissä tapahtuvat keskustelut, joissa käydään läpi edellisen sprintin aikaiset onnistumiset ja pohditaan, mitä voitaisiin parantaa seuraavan sprintin aikana. [Schwaber, 2004]

Ketterien menetelmien tyypilliset ominaisuudet eivät ole pelkkiä vahvuuksia: kun manifesti asettaa toimivan ohjelmiston tärkeämmäksi kuin kattavan dokumentaation, tarkoittaa se, ettei resursseja käytetä dokumentaation kirjoittamiseen, jolloin siitä saattaa muodostua puutteellinen tai virheellinen. Tavoitteeksi asetettu korkeatasoinen ohjelmakoodi vaatii keskivertoa parempia ohjelmoijia, ja keskiverto-ohjelmoijat saattavat koodia paloittain uudelleenkirjoitettaessa heikentää sen kokonaisrakennetta ja

laatua. Jatkuva sopeutuminen vaatimusmuutoksiin projektin edetessä saattaa hankaloittaa projektin julistamista valmiiksi ja selkeästi määritellyn lopputuloksen puute saattaa ajaa asiakkaan loputtomaan maksukierteeseen. [Marovic et al., 2009]. Vaatimusten tarkentuessa varhaisen arkkitehtuurin muuttaminen nykyisiä vaatimuksia vastaavaksi saattaa olla erittäin aikaavievää [Elshamy and Elssamadisy, 2007]. Näiden ongelmien välttämiseksi uhat tulee huomioida jo asiakkaan ja yrityksen välistä sopimusta neuvoteltaessa.

5 Esimerkkinä lainauskorvausjärjestelmä

Teoreettisten mallien ja suositusten lisäksi on hyvä analysoida eettisten asioiden huomiointia myös käytännön ohjelmistoprojektissa ja pyrkiä löytämään joitain vielä mainitsemattomia näkökulmia eettisyyteen. Esimerkkinä reaalimaailman ohjelmistoprojektista toimii Sanasto ry:n [www.sanasto.fi] tarpeisiin kehitettävä lainauskorvausjärjestelmä. Sanasto on tilannut järjestelmän suomalaiselta ohjelmistoyhtiöltä Eduixilta, jonka työntekijänä olen itse ollut mukana ohjelmiston kehitysprosessissa.

Järjestelmää käytetään tekijänoikeuslain 19§:n 4 momentin [Tekijänoikeuslaki, 2006] mukaisten lainauskorvausten tilittämiseen tekijöille. Järjestelmään kerätään tietoja lainauksista useista suomalaisista kirjastoista ja siihen kuuluu selainkäyttöliittymä, jolla kirjojen tekijät erittelevät korvausosuudet kustakin teoksestaan. Ohjelmisto tuottaa listauksen, jota Sanasto käyttää referenssinä jakaessaan vuosittaiset korvaussummat tekijöille.

5.1 Kysymyksiä tekijöille

Maaliskuussa 2009 järjestettiin ryhmähaastattelu, johon osallistuivat Eduix oy:n toimitusjohtaja, teknologiajohtaja ja projektipäällikkö. Haastatteluajankohtana ohjelmistoprojekti oli vaatimusmäärittelyvaiheen loppupuolella. Haastattelun tavoitteena oli jo varhaisessa vaiheessa tunnistaa mahdollisia eettisiä kipukohtia, jo tapahtuneita tai ennakoituja. Erityisesti sellaiset näkökulmat, joita eettiset ohjeistot eivät huomioi, olisivat erityisen arvokkaita. Haastattelukysymykset ovat liitteenä 1.

Haastattelu alkoi kysymyksillä ohjelmiston tilaajista, käyttäjistä ja sidosryhmistä. Rogerson ja Gotterbarn [2001] korostavat sidosryhmien tunnistamista yhtenä tärkeimmistä keinoista parantaa ohjelman eettisyyttä. Tässä projektissa huomataan

kuitenkin käytännön ja teorian ero: osa vaatimuksista nousee lainsäädännöstä ja suurta osaa sidosryhmistä edustaa Sanaston henkilöstö. Lakia säädettäessä on tietysti ollut mukana suuri joukko kirjailijoiden, kääntäjien, valokuvaajien, taiteilijoiden ynnä muiden luovan alan ammattilaisten etujärjestöjä [HE 126/2006 vp]. Ohjelmiston vaatimusmäärittelyssä nousisi kuitenkin esiin toisenlaisia kysymyksiä kuin lainsäädännössä, joten ohjelmiston toimivuuden ja ongelmattomuuden kannalta on oleellista, että Sanaston henkilöstö onnistuu edustamaan loppukäyttäjiä kyllin tasapuolisesti.

Lainsäädäntö pohjautuu toimitusjohtajan mielestään virheelliseen ajatukseen, jonka mukaan lainauksia korvataan, koska ne vähentävät myyntitilastoja. Hän huomauttaa, että suosittuja teoksia sekä lainataan että ostetaan runsaasti. Tämä asettaa ohjelmistokehittäjän vaikeaan tilanteeseen: kuinka eettistä on tehdä sovellus, jonka perusajatuksen uskoo itse olevan väärä? Koska ohjelmistoyhtiö voi vapaasti valita toteuttaako se tällaisen järjestelmän vai ei, kyse ei niinkään ole lain noudattamisesta tai rikkomisesta, jota esimerkiksi ACM:n eettisen ohjeiston kohta 2.3 käsittelee, vaan yleisestä yhteiskunnan edistämisestä sen sanelemilla säännöillä. Homeroksen hyveteoriaa soveltaen ohjelmistoyhtiö tekee oikein täyttäessään paikkansa yhteiskunnassa: tässä tapauksessa toteuttaessaan sen tietojärjestelmän, joka vaaditaan lain täysimääräiseen toteutumiseen. Myös protestanttinen työetiikka, jonka mukaan työ itsessään on hyve, puoltaa järjestelmän toteuttamista, sillä työntekoahan ohjelmistonkehitys on. Ohjelmistokonseptin valinta on kuitenkin eettinen ongelma, johon eettisesti valveutuneen kehitysmallin tulisi ottaa kantaa.

Vaikka osa sidosryhmistä on vain edustuksellisesti läsnä vaatimusmäärittelyssä, projektissa pyritään huomioimaan loppukäyttäjien antama palaute: toimitusjohtaja korostaa valitun kehitysmallin iteratiivisuutta sanoen, että ensimmäisen vuoden aikana toteutetaan ne välttämättömimmät ominaisuudet, jotka vaaditaan vuoden 2007 lainauksien korvaamiseen. Tämä on esimerkki kohdassa 4.3 kuvatusta kiireestä, joka puoltaa evolutionäärisen kehitysmallin valitsemista.

Haastattelussa käsiteltiin myös yksityisyydensuojaa, jota ACM:n ohjeiston kohta 1.7 kehottaa kunnioittamaan. Haastatellut vertasivat järjestelmää kirjastotietokantoihin: julkiselle yleisölle avointa tietoa olisivat kirjojen tekijät ja lainausmäärät, mutta henkilö- ja yhteystiedot olisivat suljettuja. Tämä on helposti perusteltava näkökanta, jolle on oikeastaan vaikeaa kuvitella vaihtoehtoa. Tiedon rajaamista ohjelmistokehittäjäryhmän sisällä haastatellut eivät maininneet, eikä sille perusteita olisikaan: kohdassa 3.3

esiteltiin Yangin ja muiden [2007] ajatus yrityksen ja yksilön välisistä näkemyseroista, joista saattaa seurata tietoturva- ja yksityisyysrikkomuksia yrityksen parhaista tavoitteista huolimatta.

Haastattelu jatkui kysymyksillä lähdekoodin avoimuudesta. Haastateltujen mukaan asiakas ei ollut vielä tehnyt päätöstä asiasta, mutta Eduixin ehdotus on täysin julkinen koodi, jota ulkopuoliset tahot voivat sopimuksen mukaan katselmoida. Ehdotusta perustellaan luottamuksen rakentamisella järjestelmää kohtaan: *”Mikään ei tietysti tee todennettavammaksi kuin se, että sieltä löytyy täsmälleen se, miten ne tilastot luetaan sisään ja saadaan lopputulos ulos.”*

Kohdassa 3.1 esiteltiin Koehnin [2003] määrittelemiä luottamuksen tyyppejä. Eduixin ehdottama lähdekoodin avoimuus rakentaa Koehnin määrittelemää laskelmoivaa luottamusta: käyttäjille annetaan näyttöä rahanjakoprosessin tasapuolisuudesta ja rehellisyydestä. Giddens [1991] jopa väitti, ettei järjestelmään, jonka toimintaperiaate olisi täysin tunnettu ja ymmärretty, tarvitsisi edes luottaa. Pelkkä ohjelmiston lähdekoodin avoimuus ei tietenkään riitä täyteen järjestelmän tunnetuksi tulemiseen, vaikka onkin askel sitä kohti.

Käytännössä mahdolliset katselmoinnit suorittanee jokin luotettu kansalaisjärjestö, jolloin rakentuu kunnioitukseen pohjaavaa luottamusta, jota Koehn kuvailee kestävimmäksi luottamuksen tyypiksi ja jollaista hän kehottaa rakennettavan verkkosovellukseen.

Lähdekoodin avoimuusehdotusta perustellessaan haastatellut eivät maininneet avoimuudella olevan vaikutusta koodin laatuun. Asiasta erikseen kysyttäessä he kuitenkin olivat vakuuttuneita, että mahdollisuus julkiseen katselmointiin vaikuttaa vain positiivisesti kirjoitettavan lähdekoodin tasoon. ACM:n ohjelmistonkehittäjille suuntaamaan eettisen ohjeiston kolmas kohtahan vaatii mahdollisimman korkean ammatillisen laadun takaamista ohjelmistotuotteeseen, ja julkisesti katselmoitava lähdekoodi saattaa olla merkittävän hyvä keino ohjelmiston laadun kohottamiseksi. Tätä ilmiötä kehuvat myös avoimen lähdekoodin kannattajat: Raymondin [2000] mukaan jokainen ohjelmointivirhe löytyy, kun kyllin moni silmäpari katselmoi koodia.

Korkealaatuisen lähdekoodin kirjoittaminen on hankalampaa kuin heikkolaatuisen kirjoittaminen, ja se vie siten enemmän resursseja. Tosin on myös niin, että lähdekoodin korkea laatu näkyy helpompana eli nopeampana jatkokehityksenä. Lähdekoodin avoimuuden vaikutuksesta resurssienkäyttöön haastatellut eivät maininneet mitään.

Myöskään eettisistä ohjeistoista ei löydy selkeää vastausta: ne kehottavat sekä korkeaan laatuun että kustannustehokkaaseen ohjelmistotuotantoon. Tavoitteet voi kenties yhdistää onnistuneesti, vaikka ne vaikuttavatkin ristiriitaisilta.

Kohdassa 2.4 esitettiin tutkimustulos, jonka mukaan vain 36% suomalaisista tietotekniikka-alan esimiehistä on tietoisia eettisten ohjeistojen olemassaolosta [Hirvonen, 2000]. Myöskään haastatellut eivät osanneet nimetä ainuttakaan tietotekniikka-alan eettistä ohjeistoa. Tämä vastaa edellä mainitun tutkimuksen tuloksia siitä, että eettistä pohdintaa pidetään tarpeellisena, vaikkei valmiita ohjeistoja tunneta.

Ohjeistojen tuntemisesta saattaisi olla hyötyä tässäkin projektissa: kysyessäni havaituista eettisistä ongelmista haastatellut viittasivat eri tavoin siihen, että ohjelmiston avulla suoritetun rahanjaon tulee olla mahdollisimman tasapuolinen, rehti ja taloudellisesti tehokas toimenpide. Tämä vastaus kattaa vain pienen osan esimerkiksi ACM:n ohjeiston käsittelemistä eettisistä kysymyksistä. On toki niin, että haastattelu suoritettiin ohjelmiston varhaisessa vaatimusmäärittelyvaiheessa, jossa ongelmia ei vielä välttämättä ollutkaan havaittaviksi tai kohdatut ongelmat oli koettu normaaleiksi ja rutiininomaisiksi vaiheiksi ohjelmistokehityksessä, ei niinkään ongelmiksi.

Kysymyksenasettelun laajentuessa tästä nimenomaisesta projektista kaikkiin eettisiin ongelmiin yleensä ohjelmistoprojekteissa, haastatellut yhä kertoivat yrityksen suhteesta asiakkaaseen ja korostivat asiakkaan vastuuta ohjelmiston käytössä. Tästä asiakkaan ja ohjelmistokehitysorganisaation vallanjaon suhteesta eettiset ohjeistot ovat yllättävän hiljaa. Vaikka esimerkiksi ACM:n ohjeisto kehottaakin ohjelmistokehittäjää toimimaan asiakkaan hyväksi ja säilyttämään itsenäisyytensä ammatillisten päätösten teossa, se ei esitä huomioita vastuunjakamisesta päätösten teossa.

Tässä yhteydessä ”eettinen ongelma” onkin kenties terminä huono, sillä se lähestyy pohdintaa ongelmien, eli jonkinlaisten vääryyksien tai rikkeiden, kautta. Eettiset ohjeistot kuitenkin kertovat oikeastaan siitä, kuinka ohjelmistoja luodaan mahdollisimman oikein tai hyvin. Algoritmien matemaattisen kauneuden näkeminen eettisesti tärkeänä kysymyksenä vaatii sellaista perehtymistä eettisen ohjelmistokehityksen diskurssiin, ettei sen voi olettaa olevan luontevaa yksinomaan yritysmaailmassa toimivalle taholle. Aaltonen ja Junkkari [2003, s. 140] kertovat saaneensa ohjeeksi etiikasta kertovan kirjansa kirjoittamiseen, että ”useimmat johtajat ja työelämän vaikuttajat ovat käytännön ihmisiä. Menkää suoraan asiaan älkääkä rasittako heitä Kanteilla ja muilla filosofeilla.” Olisikin kiintoisaa tietää, mitä 86%

yritysjohtajista oikeastaan tarkoitti, vastatessaan Hirvosen tutkimuksessa eettisten asioiden ohjaamisen olevan tarpeellista.

Hieman eri kysymyksenasettelulla haastatellut antoivat erilaisia vastauksia: kysyttäessä yrityksen yhteiskuntavastuusta ja pyydettäessä vapaata kommentointia haastatellut kertoivat kierrätyksestä, videoneuvottelusta matkaamisen sijaan, joukkoliikenteestä ja vähäisestä paperinkulutuksesta, eli asioista, jotka eivät ole niinkään ongelmia kuin valintoja. Toimitusjohtaja kuvaa yrityksen ja yhteiskunnan suhdetta sanoen, että ”*toimitaan niin kuin järkevä yhteiskunnan jäsen ja yksityishenkilö muutenkin toimisi*” ja että ”*vaikka meillä ei ole mitään julkilausumaa eikä mitään nokiatyyppistä hölynpölyä yhteiskuntavastuusta ja tällasesta, niin me vaan käytännössä noudatetaan sitä, toimitaan niin.*” Aaltonen ja Junkkari [2003, s. 140] muotoilevat tämän seuraavasti: ”Etiikka on tavallista elämää. Jokainen meistä tekee joka päivä satoja eettisiä ratkaisuja, useimmiten erittäin hyviä, emmekä edes huomaa niitä”. Saattaakin olla harhaanjohtavaa käyttää vain vahvasti latautunutta etiikka-termiä viittaamaan ohjelmistoalan moninaisiin näkökulmiin.

Yang ja muut [2001], Thomson ja Schmoldt [2001] tai Manuel Velasquez [Aaltonen ja Junkkari, 2003, s. 49] väittävät eettisen toiminnan parantavan kaupallista menestystä. Siihen nähden on kiintoisaa ja hieman yllättävää kuulla ohjelmistoalalla työskentelevien mielipide: epäeettinen toiminta ei ole ”*heilauttanut tai kaatanut yhtään toimijaa*”. Haastateltavien mukaan ”*suurimmat toimijat on tehnyt suurimmat kuprut*”, eli suuresti ylihintaisia, toimimattomia järjestelmiä toimittavat yritykset voivat jatkaa alalla työskentelyä suurina tekijöinä, jotka yhä saavat uusia tilauksia.

Käytännön esimerkkinä haastateltavien mielipiteistä voi esittää ongelmallisia järjestelmiä tuottaneen Tiedon, entisen TietoEnatorin. Se on 16 000 työntekijällään yksi Pohjoismaiden merkittävimmistä tietotekniikan yrityksistä [Tieto]. Valtiontalouden tarkastusvirasto teetti arvioinnin alueellisten tietoyhteiskuntahankkeiden toteutuksesta [VTV, 2008], jossa todetaan muun muassa seuraavaa Pohjois-Lapin sähköisten hallintapalvelujen kehittämisestä:

> *Hankkeen toteutuksen aikana syntyi ongelmia, koska valitun IT-toimittajan, TietoEnator Oyj:n, toimittama taloushallinnon ohjelmisto Economa Intime Plus paljastui käyttötarkoitukseensa nähden keskeneräiseksi. Ohjelmistossa esiintyneisiin ongelmiin jouduttiin etsimään ratkaisuja vielä hankkeessa järjestetyn koulutuksen aikana,*

> *joten TietoEnator Oyj joutui kehittämään valmisohjelmistona myymäänsä tuotetta vielä käyttöönottovaiheen aikana.*

Tarkastusviraston arviointi kertoo myös Oulun seudun päivähoidon verkkopalveluiden kehittämisestä. Projekti päättyi, vaikka palvelun kehittäminen oli kesken. Arviointi antaa epäonnistumisen syyksi Tiedon ongelmat [VTV, 2008]:

> *Projektin keskeneräisyys johtui pääosin TietoEnator Oyj:n vaikeuksista toteuttaa ja ottaa käyttöön sosiaalihuollon tietojärjestelmän integraatiorajapinta Hämeenlinnan seudulla käytössä olevan sähköisen asioinnin alustan kanssa.*

Tapahtuneista toteutusongelmista huolimatta yritys on juuri tehnyt uusia sopimuksia esimerkiksi Elisan ja Itellan kanssa [Tieto, Q2]. Esitellyt tapaukset ovat tietysti vain osa Tiedon tuoteportfoliosta, eivätkä kaikki sen tuottamat järjestelmät ole epäonnistuneet.

Haastattelussa sivuttiin lyhyesti myös ympäristönäkökulmaa. Toteutettavan järjestelmän ympäristövaikutuksia ei ole erikseen arvioitu, eikä minkäänlaisia mittauksia ole suunnitteilla. Kohdassa 2.5 esitellyistä syistä johtuen ohjelmistoalan yrityksillä on yleensä hyvät lähtökohdat ympäristöystävälliseen toimintaan, ja käytetyt resurssit ovat lähinnä sähköä ja laitteistoja. Eräs haastatelluista vertasi lainauskorvausjärjestelmän toteuttamista tietojärjestelmänä ja lomakkeiden pohjalta toimivana paperiversiona ja huomautti, että ainakin paperinkulutus on sähköisessä järjestelmässä vähäisempää. Tässä ohjelmistoprojektissa ei ympäristönäkökulmasta ajateltuna olekaan mitään kovin poikkeavaa muihin projekteihin verrattuna.

Haastattelun perusteella ei ole syytä olettaa tämän lainauskorvausjärjestelmäprojektin kehitystyön aikana tehtävän eettisesti arveluttavia valintoja. Kipukohtia voi seurata sidosryhmien läsnäolosta vaatimusmäärittelyssä vain Sanaston edustamana tai ajankäytössä lähdekoodin valmistelussa kelvolliseksi avoimeen julkaisuun. Haastateltujen suurimpana eettisenä ongelmana pitämään rahanjaon tasapuolisuuteen on niin hyvin valmistauduttu, ettei siihen liittyen ole odotettavissa yllätyksiä. On kuitenkin huomioitava haastattelun luonne tutkimusmetodina: kerrottaessa omasta ja yrityksensä toiminnasta on jokaisella haastateltavalla tavoitteena antaa itsestään hyvä kuva, tässä yhteydessä siis eettisesti vastuuntuntoinen ja rehti. Tämän piilotavoitteen olemassaolosta kertoo myös se, etteivät haastatellut kertaakaan sanoneet tehneensä jotakin väärin.

Erityisen mielenkiintoisena löydöksenä haastattelusta nousi kysymys asiakkaan ja ohjelmistoyrityksen päätäntävallan suhteesta. Vaikka eettiset ohjeistot eivät suhdetta kommentoi, on se potentiaalinen eettisten ristiriitojen lähtökohta. Johnson [2001] esittää kolme erilaista mallia asiakkaan ja ammattilaisen väliseen vastuunjakoon:

1. agentti (Agency)
2. holhoava (Paternalism)
3. uskottu (Fiduciary).

Agenttisuhteessa yritys tekee juuri sen, mitä asiakas pyytää, vaikka yrityksen henkilöstö näkisi asiakkaan ideat huonoina. Esimerkkinä tästä Johnson esittää pörssimeklarin toimen. Holhoavassa suhteessa yritys voi asiantuntijuutensa varjolla päättää, kuinka toimitaan, ja asiakas tottelee. Lääkäri-potilas -suhde on tyypillisesti tällainen. Kolmas vastuunjaon malli antaa osavastuun päätöksenteosta kummallekin osapuolelle. Asiantuntija kertoo asiakkaalle rehellisesti, mitä ongelmatilanteessa voidaan tehdä ja mitä ei voida tehdä ja muutenkin pitää asiakkaan tilanteen tasalla. Myös asiakas kertoo kaiken oleellisen informaation, ja päätös voidaan tehdä yhdessä. Johnson suosittaa ohjelmistoalalle tätä kolmannentyyppistä päätöksentekosuhdetta ja eettisesti valistuneen kehitysmallin tulisikin rakentaa tällainen vastuunjako.

5.2 Järjestelmän käyttöönoton jälkeinen kysely

Lainauskorvausjärjestelmän ensimmäinen versio avattiin yleiseen käyttöön Helsingin Kirjamessuilla 22.10.2009. Noin kuukausi järjestelmän julkaisun jälkeen projektin toteuttaneen kehitysryhmän jäsenille tarjottiin verkossa täytettävä kyselylomake, jonka tarkoituksena oli arvioida projektin aikana tehtyjä eettisiä valintoja. Kyselylomake muodostettiin ensisijaisesti alakohdassa 2.2.1 esitellyn ACM:n eettisen ohjeiston pohjalta. Kyselylomake lähetettiin koko kehitysryhmälle eli kahdeksalle hengelle, joista viisi vastasi. Lomake on tutkielman liitteenä 2.

Kyselystä ja sen tuloksista ei voida tulkita yksiselitteistä eettistä onnistumista tai epäonnistumista, sillä kuten Gotterbarn [1992] varoittaa, tällainen ajattelu johtaisi tarkistuslista-ajatteluun, jonka uhkana on varsinaisen eettisen pohdinnan sivuuttaminen. Tuloksista voi kuitenkin löytää yksittäisiä onnistumisia tai epäonnistumisia, joiden huomiointi auttaa kehittämään käytettyä ohjelmistokehitysmallia. Yksittäistapauksena tästä kyselystä ei voida tehdä tulkintoja siitä, mitkä ovat yleisimmät ongelmakohdat ohjelmistokehityksessä, mutta se osoittaa, että tietyt ongelmat ovat ilmenneet ainakin

kerran. Luvussa 6 esitellään moraalikysymyksiä huomioiva ohjelmistokehitysmalli, ja olisi ongelmallista, mikäli malli ei vastaisi edes tämän ohjelmistoprojektin aikana kohdattuisiin haasteisiin. Toisaalta mahdolliset onnistumiset on hyvä huomioida käytännössä toteutuneina ilmiöinä ja pyrkiä löytämään keinoja niiden toistamiseen.

Projektin kokonaisonnistumista parhaiten arvioivat kysymykset kertovat hyvästä onnistumisesta: kaikkien kyselyyn vastanneiden mielestä projekti tuotti ohjelmiston, joka täyttää tarkoituksensa ja johon asiakas on tyytyväinen. Toinen onnistuminen on lopputuotteen helppokäyttöisyys: kaikkien vastaajien mielestä ohjelmisto on varsin tai täysin helppokäyttöinen. Nämä vastaukset yhdessä kertovat ohjelmistosta, joka lunastaa paikkansa ja on kelpo väline laissa määritellyn tarpeen toteuttamiseen.

Vastaajien suhde lainsäädäntöön näkyy kysymyksessä, joka arvioi järjestelmän vaikutusta yhteiskunnan oikeudenmukaisuuteen. Vastaukset vaihtelivat 'hieman' ja 'paljon' välillä. Kysymyksen suurempi merkitys on kuitenkin siinä, kuinka mielekkääksi työntekijä kokee työnsä. Tunne työn merkityksettömyydestä olisi epätoivottavaa, vaatiihan jopa ACM:n eettisen ohjeiston ensimmäinen kohta edistämään yhteiskunnan ja ihmisten hyvinvointia [ACM, 1992]. Siksi onkin hyvä, että kaikki kyselyyn vastanneet kokivat ohjelmistolla olevan jonkinlaisen oikeudenmukaisuutta lisäävän vaikutuksen. Työnteon mielekkyyteen vaikuttaa lopputuotteen merkityksellisyyden lisäksi myös kokemus työssäoppimisesta ja kaikki vastaajat kokivatkin oppimisen riittäväksi – tosin ohjelmistoalalla jo 14 vuotta työskennellyt vastaaja ei kokenut kehittyneensä ammatissaan lainkaan projektin aikana. Tämä johtunee kokemuksen kertymisestä, joka ilmenee nuorta ammattilaista matalampana kehittymisen tarpeena. Muut vastaajat kokivat kehittyneensä jonkin verran tai paljon.

ACM:n ohjeisto [ACM, 1992] määrittelee laadun kenties tärkeimmäksi pakotteeksi ammattilaiselle, ja siksi onkin huolestuttavaa, ettei yksikään vastaajista sanonut tehneensä todella laadukasta työtä. Kaksi vastaajista kertoi tehneensä hyvää ja kaksi keskinkertaista jälkeä ja yksi vastaajista jopa ilmoitti tehneensä huonoa laatua. Näihin vastauksiin vaikuttaa varmasti itsearvioinnin hankaluus tai jonkinlainen pelko itsekehusta, sillä muun kehitysryhmän työtä arvioidessa vastaukset jakaantuivat tasaisesti keskiverron ja todella hyvän välille. Jonkinlaisesta muiden ja oman työn vertailun vaikeudesta tai vähäisyydestä itse- ja vertaisarvioiden eroavaisuudet kuitenkin kertovat. Kun kaikkien vastaajien mielestä kollegiaalinen kommunikaatiokin oli kelvollista tai sujuvaa, on vaikeaa osoittaa, mistä arvioiden ristiriitaisuus johtuu. Huonosti kommunikoivassa työryhmässä saattaisi muodostua vääristyneitä kuvia

omasta työstä suhteessa muiden työhön, mutta sellaisesta ei vaikuta olevan kyse: kaikki vastanneet kokivat saaneensa muulta kehitysryhmältä jonkin verran tai runsaasti palautetta työstään. Yksi selitys ristiriidalle kuitenkin löytyy: kaksi vastaajista koki antaneensa vain niukasti palautetta muulle työryhmälle, siinä missä muut vastasivat antaneensa sitä jonkin verran tai runsaasti. Näiden kahden niukasti palautetta jakaneen vastaajan aktivoiminen olisi kenties tukenut mielestään huonoa laatua tehneen vastaajan työtä ja auttanut tätä tekemään laadukkaampaa jälkeä.

Saattaa myös olla, ettei työn vertaisarviointiin tuntunut riittävän resursseja: kolme viidestä vastaajasta sanoo, että omaan työntekoon oli hieman tai aivan liian vähän aikaa. Työn tarvitseman ajan ja saavutetun laadun suhdetta voi olla vaikea arvioida, sillä niistä kahdesta vastaajasta, jotka kokivat, että työhön oli aivan liian vähän aikaa, toinen vastasi tehneensä huonoa ja toinen hyvää laatua. Huomioitavaa kuitenkin on, että jo esihaastattelun tuloksista oli ennakoitavissa riski työajan vähäiseen riittävyyteen.

Toinen esihaastattelussa ilmennyt riski oli sidosryhmien edustus vaatimusmäärittelyvaiheessa vain välillisesti. Tämä ilmeni kyselyn tuloksissa siten, että kolmen vastaajan mielestä toteutunut järjestelmä oli keskimääräisesti suunnitellun kaltainen ja kahden mielestä vain hieman suunnitellun kaltainen. Tämä kertoo ohjelmiston vaatimusten muuntuneen projektin edetessä, kenties jopa hieman enemmän kuin keskiarvoprojektissa. Muutokset vaatimuksissa ovat kenties osaltaan vaikuttaneet kokemuksiin työajan vähäisestä riittävyydestä.

Kysely ei tuonut ilmi suuria moraalisia epäonnistumisia. Työn laatuun ja ajan vähäisen riittävyyden kautta työssäviihtymiseen liittyi ongelmia tai ainakin riskejä. Kolme kyselyyn vastanneista ilmoittikin käytetyn ajan ja laadun välisen suhteen yhdeksi yleisimmistä eettisistä kysymyksistä ohjelmistontuotannossa. Voittoa tavoittelevan yrityksen ei kuitenkaan ole välttämättä yksiselitteistä nostaa tuotekehitykseen käytettyjen resurssien määrää. Eettisesti valistuneen kehitysmallin tulisi siis erityisesti huomioida ajan ja laadun välinen ristiriita.

6 Eettistä pohdintaa ohjelmistokehitysmalliin

Kun tavoitteena on tehdä ”eettinen ohjelmisto”, on ensin kysyttävä, mitä se tarkoittaa. Ohjelmistolla on elinkaarensa konseptista tuotteeksi ja edelleen käyttötuen loppuun asti. Koko tämän elinkaaren aikana tuotteen valmistanut ohjelmistoyritys tekee väistämättä erinäisiä eettisiä valintoja. Nämä valinnat muodostavat kokonaisuuden, jota voitaneen

kutsua ohjelmiston kokonaiseettisyydeksi. Näin ajateltuna ”eettinen ohjelmisto” on siis sellainen ohjelmisto, jonka elinkaaren aikana tehdyt eettiset valinnat ovat kaikki onnistuneita. Suuri osa tätä elinkaarta on matka konseptista valmiiksi tuotteeksi, eli itse kehitysprosessi. Vaikka esimerkiksi valmiin tuotteen käyttäjätuki ja mainonta ovatkin tärkeä osa ohjelmiston elinkaarta, ei niihin liittyviä eettisiä valintoja tässä tutkielmassa käsitellä, sillä ne ovat teknisen kehitysprosessin jälkeen tapahtuvia asioita. Tämä rajaa tässä tutkielmassa seuraavaksi esitetyn ohjelmiston eettisyyden arvioinnin koskemaan vain osaa kokonaiseettisyydestä.

Jotta ohjelmiston eettisyyttä voi arvioida, sen tulee olla jonkinlainen suure, jota voisi kenties myös mitata. Väitteessä on osa totta: moraaliset ongelmat voi ratkaista paitsi väärin tai oikein, myös vähän sinne päin tai poikkeuksellisen hyvin. Lisäksi ohjelmistokehityksen aikana tehdään suuri joukko eettistä harkintaa vaativia ratkaisuja, ja näistä voidaan ratkoa onnistuneesti jokin tietty osa, olkoonkin se paljon tai vähän. Mahdollista on sekin, että muutoin onnistuneessa kehitysprosessissa jää jokin tietty eettinen osa-alue huomioimatta. Tämä kaikki osaltaan rohkaisee tavoittelemaan moraalisesti kestävää ohjelmistokehitystä – vaikka täydellistä ratkaisua ei löytäisikään, on jokainen askel kohti ratkaisua jo itsessään merkityksellinen. Ohjelmistokehityksessä kohdattavien ongelmien suuruusluokka-, laatu- ja vaikutuserojen vuoksi on kuitenkin vaikeaa, ellei mahdotonta, kehittää kvantitaviisia metriikoita ohjelmiston moraalisen onnistumisen arviointiin tai supistaa eettinen pohdinta jonkinlaisiksi eettisyyspisteiksi.

Laadullinen arviointi, joka pitää sisällään epäkohtien ja onnistumisten huomiointia vaikuttaa olevan ainoa tapa arvioida ohjelmiston eettisyyttä. Arvioitavien seikkojen monipuolisuuden vuoksi on harhaanjohtavaa luokitella ohjelmistot vain eettisiksi tai epäeettisiksi. Varsinkin edellisessä kappaleessa esitetty ajatus siitä, että ”eettinen ohjelmisto” voi olla vain sellainen, jonka kehitysprosessin aikana kaikki eettiset valinnat on tehty onnistuneesti, on erityisen hankalaa yhdistää kaksijakoiseen luokitteluun, sillä se ei tekisi erottelua ”epäeettisten ohjelmistojen” kesken, eikä osittaisella onnistumisella olisi merkitystä. Asia ei parane sillä, että kaikkien moraalivalintojen onnistumisen sijaan vaadittaisiin vain jonkin murto-osan, kuten puolien, onnistumista. Tehtyjen ja kohdattujen moraalivalintojen määrää ei voi yksiselitteisesti laskea, eikä ohjelmistojen jakaminen vain kahteen selkeään kategoriaan ole todenmukaista. Onkin siis tyydyttävä johonkin epävarmempaan jaotteluun.

Jos ”eettinen ohjelmisto” olisi sellainen ohjelmisto, jonka kehitysprosessin aikana on tavoiteltu eettisesti oikeita ratkaisuja, vaikkei niitä olisi saavutettukaan, olisi pyrkimys

eettisyyteen riittävä kriteeri eettisyydelle. Todellisuudessa eettisiä ratkaisuja saatetaan yhden ohjelmistoprojektin sisällä etsiä joissain tapauksissa enemmän kuin toisissa tapauksissa. Tällöin voi kuitenkin kyseenalaistaa pyrkimyksen aitouden. Joissain tapauksissa eettisesti oikeaa ja toteutettavissa olevaa ratkaisua ei yrityksistä huolimatta kuitenkaan löydetä, joten on mielekkäämpää kiinnittää huomiota tavoitteisiin kuin varsinaisiin ratkaisuihin. Kriteerin kannalta on kuitenkin ongelmallista, että kehitysryhmän jäsenet eivät välttämättä tiedosta kaikkia osa-alueita, joissa eettisyyteen voi pyrkiä. Tällöin rehellisetkin pyrkimykset saattavat olla riittämättömiä huomioimaan jotakin yleisesti hyväksyttyä ohjelmiston eettisyyden aspektia ja eettisten ohjeistojen lukeminen olisi haitallista – sen jälkeen ohjelmistokehittäjällä olisi enemmän näkökulmia huomioitavaksi pyrkimyksissään, mikä uhkaisi tuotetun ohjelmiston nimikettä ”eettisenä ohjelmistona”. Tämä kuulostaa merkilliseltä ja eettisten ohjeistojen tarkoituksen vastaiselta. Kriteeri onkin vaikeasti sovellettava, sillä yksittäisten ohjelmistokehittäjien todellisia pyrkimyksiä ei ole yksinkertaista saada varmasti selville. Lisäksi nämä pyrkimykset tulisi objektiivisesti arvioida eettisesti oikeiksi ja vääriksi tavoitteiksi.

Kysymys siitä, mikä ”eettinen ohjelmisto” on, nivoutuukin siihen, mikä on eettistä. Tutkielman kohdassa 2.1 käsiteltiin erilaisia etiikkakäsityksiä, eikä ole syytä olettaa, että etiikkakäsitysten kehitys olisi lakannut. Niitä on useita ja ne ovat keskenään osin ristiriitaisia. Kenties olisi siis mielekkäintä puhua ohjelmistosta, jonka kehityksessä on tavoiteltu jonkin tietyn eettisen käsityksen mukaista hyvää. Tällöin monihenkisen kehitysryhmän henkilökohtaiset etiikkakäsitykset tulisi jotenkin koostaa yhteiseksi etiikkakäsitykseksi, joka voitaisiin esittää kerrottaessa, millä tavalla tietty ohjelmisto on ”eettinen ohjelmisto”. Itseasiassa juuri tällaisia eksplikoituja etiikkakäsityksiä ovat eettiset ohjeistot. Kehitysryhmän tulisi tietysti myös toimia ilmaisemiensa tavoitteiden mukaisesti eikä pelkästään väittää toimivansa.

Vaikkei olisi mielekästä puhua ”eettisestä ohjelmistosta”, saattaa olla mielekästä puhua eettisesti valveutuneesta kehitysmallista, eli kehitysmallista, joka huomioi ohjelmistokehityksen eettiset näkökulmat. Kun kehitysyhteisön pyrkimys tietyn etiikkakäsityksen mukaisiin ratkaisuihin on ainoa kriteerimme eettiselle ohjelmistolle, tulisi eettisesti valveutuneen kehitysmallin sisältää keinoja, joilla mahdollistetaan tämän pyrkimyksen esilletuonti ja toiminta sen mukaan. Käytännössä tämä tarkoittaa eksplisiittistä valitun eettisen näkemyksen huomiointia kehitysprosessin jokaisessa vaiheessa.

6.1 Kehitysmalli

Luvussa 2 eettiset ohjeistot esiteltiin ohjenuorina, jotka ohjelmistokehittäjien tulisi toteuttaa. Ohjeistoja voinee kuitenkin käyttää myös kuvailuina tai tuntomerkkeinä siitä, millainen eettisesti onnistunut ohjelmistohanke on. Kohdassa 2.3 on listattu eri ohjeistojen yhteiset piirteet: yhteiskunnan osana toimiminen, rehellisyys ja reiluus, vastuu päätöksistä, laadukas ohjelmisto sekä tietojenkäsittelytieteen edistäminen. Lista toimii siis ainakin osittaisena vastauksena kysymykseen, mitä ohjelmiston eettisyys on. Kolmannessa luvussa käsiteltiin käyttäjän luottamussuhdetta ohjelmistoon ja yritykseen merkkinä eettisestä onnistumisesta, sekä luottamuksen kognitiivista kuormaa vähentävää vaikutusta ja keinoja rakentaa sitä ohjelmistoon. Kohdassa 2.5 käsiteltiin ympäristöä ohjelmiston sidosryhmänä ja tuotiin esiin yrityksen ympäristötietoisuuden merkitys kehitysprosessissa.

Seuraavaksi esitettävä eettisesti valveutunut kehitysmalli käyttää käsityksenä eettisyydestä edellä kuvattuja ominaisuuksia. Ne muodostavat ohjelmiston kehitysprosessin osuuden ohjelmiston kokonaiseettisyydestä. Esiteltävän eettisesti valveutuneen ohjelmistokehitysmallin on käytännössä siis autettava ohjelmistokehittäjiä huomioimaan ja ratkomaan juuri edelläkuvatut ongelmat.

Rogerson ja Gotterbarn [1997] väittävät, ettei mikään nykyinen ohjelmistonkehitysmalli huomioi eettisiä seikkoja. Mistä johtuu, ettei tällaista mallia ole esitelty? Osin varmasti siitä, että eettisyyden on koettu voivan olla osa mitä tahansa kehitysmallia, onhan eettiset päätökset tehtävä mallista riippumatta. Tietyssä määrin näin varmasti onkin, mutta kuten tutkielman neljännessä luvussa kävi ilmi, jotkin kehitysmallit sopivat käytännön projekteihin toisia paremmin. Esimerkiksi inkrementaalinen malli on tällainen paljon käytetty ja usein myös lopputuloksen laatuun positiivisesti vaikuttava malli. Koska ohjelmiston korkea laatu on yksi osa ohjelmiston eettisyyttä, on oikean, eli korkeimman laadun takaavan, kehitysmallin valinta tärkeä kysymys. Oikea kehitysmalli on toki projektikohtainen valinta.

Ohjelmistonkehitys saattaa olla hyvinkin erilaista riippuen valitusta kehitysmallista, joten mallin valinta vaikuttaa myös kohdattaviin moraalisiin ongelmiin, kuten luvussa 4 kävi ilmi. Vesiputousmallissa saattaa nousta esimerkiksi työntekijöiden motivointiin liittyviä hallinnollisia moraaliongelmia enemmän kuin muissa malleissa. Prototyyppimallissa tyypillisiä ongelmia saattavat olla pikaisesti kirjoitetun, siis huonolaatuisen, lähdekoodin uusiokäyttöön liittyvät ongelmat. Inkrementaalisessa mallissa vaatimusten priorisointi on erityisen tärkeässä roolissa ja ongelmat korostuvat,

mikäli ohjelmiston julkistus aikaistuu ennen viimeisten inkrementtien valmistumista. Ketteriä menetelmiä käytettäessä ongelmia saattaa koitua vähäisestä esisuunnittelusta, joka ilmenee esimerkiksi ohjelmakoodin huonon skaalautuvuuden muodossa.

Kehitysmallia valittaessa on huomioitava, että esimerkiksi Gotterbarn [1992] pitää jatkuvaa keskustelua tehokkaimpana keinona opettaa etiikkaa, ja siksi olisikin suotavaa, että eettinen kehitysmalli tarjoaisi eksplisiittisiä mahdollisuuksia tällaiselle jatkuvalle keskustelulle. Inkrementaalisen mallin tasaisin väliajoin tapahtuvat inkrementtienvaihdokset, joihin on perinteisesti sijoitettu muun muassa asiakaskatselmoinnit, voisivatkin olla luonteva paikka tällaisille eettisille keskusteluille. Myös ketterien menetelmien periaatteisiin kuuluu kehitysryhmän keskinäinen kasvokkain puhuminen ja tasaisin väliajoin toistuva itsereflektio [Agile Manifesto, 2001]. Esimerkiksi Scrum-malliin kuuluu eksplisiittisiä päivittäisiä keskusteluja [Schweber, 2004].

Tässä tutkielmassa ehdotelma eettiseksi kehitysmalliksi rakennetaan inkrementaalisen mallin pohjalle, mutta myös erityisesti ketterät mallit huomioidaan, sillä ne puuttuvat todellisiin, havaittuihin ongelmiin ohjelmistokehityksessä. Osa esitetyn ehdotelman pohdinnasta on toki hyödynnettävissä kehitysmallista riippumatta.

6.2 Ohjelmistokonsepti

Ohjelmistoprojekti lähtee aina ohjelmistokonseptista, joten on tärkeää kysyä voiko jokin ohjelmisto olla jo lähtökohtaisesti toista eettisempi. Esimerkiksi ympäristönsuojelulaki esittää tavoitteekseen parantaa kansalaisten mahdollisuuksia vaikuttaa ympäristöä koskevaan päätöksentekoon. Voiko siis ajatella, että ohjelmistohankkeet, jotka ympäristönsuojelulain tavoitteiden mukaisesti edistävät tietoyhteiskuntaa tai parantavat kansalaisen vaikutusmahdollisuuksia, ovat saman tien muita parempia? Väite tuntuu perusteettomalta: niin kauan kuin ohjelmisto toimii ACM:n ohjeiston sanoin ”yhdenmukaisesti yhteiskunnan edun kanssa”, on se oikeuttanut olemassaolonsa. Ainahan näin ei ole: selkein esimerkki epäeettisistä ohjelmistoista ovat virukset, joiden kirjoittaminen on rikoslaissakin kielletty [Rikoslaki].

Yhdenmukaisuus yhteiskunnan edun kanssa on kuitenkin hatarasti määritelty käsite, ja virusten lisäksi löytyy varmasti muitakin käytännön esimerkkejä lähtökohtaisesti epäeettisistä ohjelmistoista, kuten esimerkiksi yksityisyyttä rikkovat vakoiluohjelmistot. Myös tiedonrikastusohjelmistot, joilla yhdistellään esimerkiksi asuinpaikka-, varallisuus- ja terveystietokantoja asuntolainapäätöksen muodostamiseen, voidaan

kokea eettisesti arveluttaviksi [Witten and Frank, 2000, p. 32]. Jotkin ohjelmistohankkeet voivat siis olla jo konseptina eettisesti onnistuneempia kuin toiset. Vartiainen [2009] esittää ohjelmistokonseptiin liittyvät eettiset ongelmat ”tarkoituksellisen tason” (intentional level) -moraaliristiriitoina: ohjelmistokehittäjä joutuu arvioimaan ohjelmiston aiottua käyttötarkoitusta, joka saattaa olla ristiriidassa kehittäjän omien arvojen kanssa. Joissain tapauksissa kehitettävää ohjelmistoa on mahdollista käyttää useaan tarkoitukseen, joista vain osan ohjelmistokehittäjä kokee olevan ristiriidassa omien arvojensa kanssa. Tällöin konseptin eettinen arviointi on erityisen vaikeaa.

Valittaessa tietty projekti jätetään jokin toinen projekti valitsematta. Corvellec ja Macheridis [2009] ovat erottaneet projektinvalinnan kolmivaiheiseksi prosessiksi. Prosessin ensimmäisessä vaiheessa kerätään ehdotukset toteutettavaksi projektiksi. Corvellec ja Macheridis esittävät projektinvalitsijoiden moraaliseksi velvollisuudeksi kommunikoida rehellisesti, ymmärrettävästi ja riittävän tarkasti, millaisia projekteja projektin toteuttava organisaatio etsii toteutettavaksi ja millä kriteereillä projektin valinta tehdään. Erityisenä hankaluutena esitetään avoimuuden ja rajoittuneisuuden välinen jännite: vaikka valitsijoiden tulisi hyväksyä kaikkien ehdotuksia listalleen, heidän tulisi hylätä vahingolliset ehdotukset, kuten sellaiset, joilla ehdottaja pyrkii ainoastaan saavuttamaan itselleen näkyvyyttä ja huomionarvoisuutta toisaalla.

Ensimmäisessä vaiheessa kerätään kokoelma mahdollisimman hyviä projektiehdotuksia, jotka prosessin toisessa vaiheessa arvioidaan. Jokainen ehdotus tulisi arvioida kokonaisuudessaan ja systemaattisesti. Vaiheen tuotoksena tulisi olla rehellinen arvio kunkin projektin ominaisuuksista, kuten hinnasta ja aika-arvioista. Arviota muodostaessa on oleellista tunnistaa eturistiriidat ja pyrkiä eliminoimaan niiden vaikutus.

Corvellec ja Macheridis väittävät, että valitsijoilla on velvollisuus arvioida myös projektin pitkän aikavälin seuraamukset kaikille projektin osapuolille. He kuitenkin huomauttavat, että vaikka valitsijoilla onkin vastuu projektin seuraamuksista, ei heitä voida pitää vastuullisina kaikista projektista johtuvista ilmiöistä, varsinkaan vaikeasti ennakoitavista tai pitkän ajan kuluttua toteutuvista.

Projektinvalinnan viimeisessä vaiheessa tehdään valinta toteutettavasta projektista ja kerrotaan päätöksestä eteenpäin. Päätös voi myös olla, ettei mitään ehdotuksista toteuteta. Valinnan tekeminen arviotujen ehdotusten kokoelmasta ei kuitenkaan ole suoraviivaista, eikä ”paras projekti toteutettavaksi” ei ole selkeästi määritelty käsite.

Esimerkiksi toteutusorganisaation taloudellinen tilanne saattaa määrätä, kuinka suuri riski kannattaa ottaa ja yrityksen maine saattaa vaikeuttaa tietynlaisten, jopa eettisesti hyvien, valintojen selittämistä sidosryhmille. Ympäröivä organisaatio asettaa siis rajoituksia valitsijoiden vapaudelle tehdä päätöksiä. Corvellec ja Macheridic eivät esitä tätä rajoitusta ongelmallisena, mutta kuten kohdassa 2.4 kävi ilmi, tästä saattaa seurata vastuun katoaminen yksittäisiltä ihmisiltä organisaatiolle ja ei-kenellekään. Ongelmia valintaan aiheuttaa myös organisaation kirjoittamattomat säännöt, työntekijöiden valtapeli, piilotavoitteet ja muut senkaltaiset seikat.

6.3 Vaatimusmäärittely

Konseptin keksimisen tai hyväksymisen jälkeen ohjelmistonkehityksen seuraava vaihe on vaatimusmäärittely, joka inkrementaalisessa mallissa on hieman karkealinjaisempi kuin jäykemmissä malleissa. Rogerson ja Gotterbarn [1997] korostavat tässä vaiheessa tapahtuvaa sidosryhmien tunnistamista. Heidän mukaansa sidosryhmiä ovat paitsi ne, jotka rahoittavat luotavan järjestelmän tai ovat poliittisesti merkityksellisiä, myös kaikki ne yksilöt tai yhteisöt, joihin projekti suoraan tai epäsuoraan vaikuttaa. Erityisen huomion kohteeksi Rogerson ja Gotterbarn haluavat asettaa ne, joihin projekti vaikuttaa negatiivisesti, sillä nuo tahot helposti sivuutetaan. He myös listaavat suuren määrän sidosryhmiä, jotka ovat läsnä jokaisessa ohjelmistoprojektissa: ohjelmiston käyttäjät, käyttäjien perheet, ne yhteiskunnalliset instituutiot, joihin ohjelmiston ilmestyminen vaikuttaa, luonnollinen ympäristö, sosiaaliset yhteisöt, ohjelmistoalan ammattilaiset, ohjelmistokehitysorganisaation työntekijät ja tuo organisaatio itse. Lisää sidosryhmiä voi tunnistaa alakohdassa 2.2.4 esitellyillä Gertin kymmenellä moraalisäännöllä. Näitä sääntöjä käyttämällä voi myös priorisoida sidosryhmiä. [Rogerson and Gotterbarn, 1997].

Sidosryhmien priorisointi on välttämätöntä, sillä suuri joukko sidosryhmiä johtaa vääjäämättä intressikonflikteihin, jotka tulee jossain vaiheessa ratkaista. Silti priorisoinnissa on riskinsä, sillä sen tekevä taho, kenties jokin ohjelmistokehitysorganisaation nimeämä työryhmä, on itsekin yksi ohjelmiston sidosryhmistä. Täysin objektiiviseen priorisointiin ei siis päästäne, mutta nimenomaan tässä eksplisiittiset listat, kuten Gertin moraalisäännöt, edistävät priorisoinnin läpinäkyvyyttä.

Intressikonfliktien ratkaiseminen saattaa olla helpompaa, kun ymmärrys etiikasta ja eettisistä velvollisuuksista kypsyy. Stoodley [2009] esittää viisiportaisen asteikon

eettisen valveutuneisuuden tasoista, jossa kukin porras edustaa laadullisesti erilaista tapaa kokea etiikka. Yksi portaikon ulottuvuus on hyötyjänäkökulma, jonka tasot ovat:

1. lähipiiri
2. organisaatio
3. asiakas ja itse
4. asiakas
5. ihmiskunta.

Portaikko kertoo, kenen etua ohjelmistoalan ammattilainen päätöksissään hakee. Edetetessään tasolta seuraavalle hän saattaa esimerkiksi todeta, ettei halua työskennellä yrityksessä, joka tuottaa palveluita, jotka hän kokee yhteiskunnalle haitalliseksi, vaikkapa vedonlyönnin ja sotatoimien parissa. Eettisesti kypsä ammattilainen on sisäistänyt kaikki portaikon tasot. Vaikka korkea valveutuneisuuden taso tarkoittaakin kykyä tehdä eettisesti parempia päätöksiä, portaikko itsessään on vain eettistä päätöksentekoa selittävä malli, eikä siis selkeä keino eettisesti parempien päätösten löytämiseen. Se korostaa ammattilaisten henkilökohtaista eettisen valveutuneisuuden merkitystä päätöksenteossa ja siten vastaa hyvin esimerkiksi Gotterbarnin [1992] varoituksiin tarkistuslista-tyyppisestä tavasta ajatella eettisiä kysymyksiä.

Tässä vaiheessa projektia on myös hyvä tehdä selväksi yrityksen ja asiakkaan välinen vastuunjako. Sanasto-esimerkkiprojektin haastattelu- ja kyselylomakevastauksissa korostettiin tätä aspektia jopa enemmän kuin muita eettisiä kysymyksiä. Kohdassa 5.1 esiteltiin joitain asiakkaan ja yrityksen välisiä vastuunjakomalleja, mutta siinä esitelty Johnsonin [2001] ohjelmistoalalle suosittelema 'uskottu'-suhde ei ota kantaa siihen, miten vastuut tulisi käytännössä jakaa. Vastuut ja niiden jakaminen vaihtelevatkin projekteittain ja oleellista on nimenomaan se, että ne eksplikoidaan kummallekin osapuolelle niin kehityksen kuin julkaisunkin jälkeiseksi ajaksi.

Vaatimusmäärittelyvaiheessa kerätään ohjelmiston vaatimukset ja laaja sidosryhmien tunnistaminen mahdollistaa useiden vaatimusten tunnistamisen. Kerätyt vaatimukset priorisoidaan sidosryhmien priorisoinnin pohjalta. Tässäkin tarvitaan keskustelua, sillä esimerkiksi useilla vähemmistösidosryhmillä saattaa olla yhteisiä vaatimuksia, jotka ovat kenties ristiriidassa tietyn tärkeämmän sidosryhmän vaatimusten kanssa.

Tämän vaiheen lopputuloksena tulisi olla lista vaatimuksista. Vaatimusten tulee olla siinä määrin karkeita ja tarkentamattomia, että projektin edetessä niitä voidaan

inkrementti inkrementiltä tarkentaa. Projektin edetessä ymmärrys ohjelmiston toiminnasta ja vaatimuksista tarkentuu, joten tarkat vaatimukset tulisi muodostaa vasta tällä suuremmalla ymmärryksellä. Toisaalta tässä esivaatimusmäärittelyvaiheessa kerätyn listan tulisi myös olla siinä määrin kattava, että ohjelmiston tekninen perusarkkitehtuuri voidaan luoda riittävän kattavaksi, eikä tarkentuvista vaatimuksista muodostuisi kohtuutonta koodin uudelleenkirjoitusrasitetta.

6.4 Inkrementteihin jako

Seuraavassa vaiheessa lista priorisoiduista vaatimuksista on jaettava inkrementteihin. Tämä vaatii projektiin käytettävien resurssien ja aikataulun tunnistamista. Agile manifesto [2001] kehottaa yhteistyöhön asiakkaan kanssa mieluummin kuin tarkan sopimuksen neuvotteluun. Ketterien menetelmien kannattajien mielestä resurssien, aikataulun ja vaatimusten vaikeasti yhteensovitettava liitto on siis ratkaistavissa sopimusteknisesti. Ohjelmistontuotanto on sikäli erityislaatuinen tuotannonala, että jokainen tuote, siis ohjelmisto, on omalla tavallaan uniikki, eikä kohdattavien ongelmien määrää ja laatua voida etukäteen tarkasti tuntea. Braithwaite [2007] korostaa ohjelmistonkehityksen satunnaista luonnetta ja väittää, ettei ole minkäänlaisia muuttujia, joista voisi päätellä ohjelmiston valmistumisajankohdan ja vertaa aika-arvioita ohjelmiston valmistumisesta valheisiin.

Sanasto-esimerkkiprojektin kyselytutkimuksessa eräs vastaajista puki tämän ongelman sanoiksi eettisistä valinnoista kysyettäessä seuraavasti: ”*Yleisin valinta on tasapainoilu laskutettavien ominaisuuksien tekemisen ja poikkeamien korjaamiseen kulutetun ajan välillä.*”

Kyseiseen ongelmaan on vaikea puuttua, mutta esimerkiksi inkrementtien määrääminen ajallisesti tasamittaisiksi olisi suositeltavaa. Tästä seuraa tasaisin aikavälein asiakkalle toimitettava versio ohjelmasta, jolloin kehityksen tarkkailu helpottuu. Kun ominaisuudet jaetaan inkrementteihin jo projektin alkuvaiheessa ja jotain jää mahdollisesti valmistumatta aiotussa inkrementissä, siirretään vaatimuksia seuraaviin inkrementteihin. Tämä antaa mahdollisuuden tunnistaa yhtäläisyyksiä hitaasti toteutuvien vaatimusten ja tulevien inkrementtien arvioitujen kuormitusten välillä. Tällöin karkealinjaisten vaatimusten inkremetteihin jakoa voidaan päivittää ja projektin lopullisen valmistumisen ajankohtaa pystytään ennakoimaan kenties hieman paremmin.

Puuttumatta tarkemmin juridisiin yksityiskohtiin, eettisesti valveutunut ohjelmistokehitysmalli sisältäisi asiakkaan kanssa tehdyn sopimuksen, joka sitoisi

yrityksen saaman rahallisen kompensaation kyllin tarkasti ohjelmistossa tapahtuvaan kehitykseen. Toisaalta sen pitäisi pystyä estämään tilanne, jossa asiakas pidetään jatkuvassa maksukierteessä ilman lopullisesti valmistuvaa tuotetta. Esimerkiksi inkrementteihin sidottu maksuaikataulu yhdistettynä asiakkaalle kuuluvaan lähdekoodin jatkokehitysoikeuteen voisi olla tällainen järjestely. Mahdollisen valmistumismääräajan lähestyessä ja projektin paljastuessa arvioitua haastavammaksi, onnistuneesti inkrementteihin priorisoidut vaatimukset johtaisivat vain ydinominaisuuksilla varustetun ohjelmiston julkaisuun. Tämä olisi kuitenkin usein parempi vaihtoehto kuin suurempi joukko rikkinäisiä toimintoja tai ei ohjelmistoa lainkaan.

Kun on olemassa mahdollisuus julkaista ohjelmisto, jossa on suunniteltua pienempi toimintokokoelma, vähentyvät myös tietyt jännitteet kehitysryhmän sisällä: kun tavoitellaan tiettyä ennaltamäärättyä ominaisuusjoukkoa määrätyssä ajassa ja ohjelmisto osoittautuukin ennakoitua haastavammaksi, on tehtävä päätöksiä julkaisuajankohdan siirtämisestä. Kärjistäen voi sanoa, että myöhästynyt projekti on projektipäällikön ongelma, mutta virheellisesti toimiva ohjelmisto on taaseen ohjelmoijien ongelma. Mikäli projektipäällikkö on vastuussa julkaisupäätöksestä, eturistiriita on ilmeinen.

On myös huomioitava, että projektiin käytetyn ajan ja rahan määrä on helppo laskea. Tuotetun laadun arviointi on haastavampaa, joten toimitukseen liittyvät sopimukset ja projektin hallinnointi saattavat helposti jättää laadun aika- ja rahakriteereitä epätarkemmin määritellyksi.

Hyvin toteutettuna tällainen vaatimusmäärittelyvaihe vastaa hyvinkin selkeästi luvussa 6.1 esiteltyihin eettisen ohjelmistokehitysmallin vaatimuksiin sekä yhteiskunnan osana toimimisesta että rehellisyydestä ja reiluudesta.

6.5 Kehityssykli

Kuten kohdassa 4.3 esiteltiin, inkrementaalisessa mallissa ohjelmistonkehitys etenee inkrementeissä, eli tiettyjä toimenpiteitä syklisesti toistaen. Jokainen näistä inkrementeistä on kuin pienimuotoinen vesiputousmallilla toteutettu projekti sisältäen vaatimusmäärittelyn, suunnittelun, toteutuksen ja testauksen. Näistä vaiheista ei ole syytä poiketa tämän eettisesti valveutuneen kehitysmallin ehdotelman sisällä.

Yksittäisen inkrementin sisäisessä vaatimusmäärittelyssä karkeat kyseiselle inkrementille määrätyt vaatimukset kehitetään joukoksi tarkempia vaatimuksia. Tämä on luonteva vaihe huomioida kohdassa 6.1 vaadittu luottamussuhteen rakentaminen.

Luvun 3.2 keinot luottamuksen rakentamiseen ovat siten yksityiskohtaisia, että ne on mielekkäintä hyödyntää inkrementtikohtaisia tarkkoja vaatimuksia määritellessä.

Suunnitteluvaiheen pituus riippuu inkrementin vaatimuksista: ensimmäisten inkrementtien sisältö saattaa olla niin perustavanlaatuista arkkitehtuurin tekoa, että valtaosa inkrementin ajasta kuluu suunnitteluun. Myöhemmissä inkrementeissä ominaisuuksien lisääminen taas saattaa syntyä hyvän pohjatyön päälle vähäiselläkin suunnittelulla ja ajallinen pääpaino on toteutuksella. Alun arkkitehtuurivaiheessa suunnittelun ja toteutuksen välinen raja saa myös olla häilyvä: suunniteltua arkkitehtuuria voi testata kirjoittamalla sen ohjelmaksi, jota käytettäessä havaitaan ongelmia, jotka korjataan toteutukseen. Tällöin myös alkuperäisen arkkitehtuurisuunnitelman dokumentaatio on korjattava, mutta Agile Manifesto [2001] kehottaakin suosimaan toimivaa koodia kattavan dokumentaation sijaan. Tällöin aikaa ei kulu dokumentaation korjaamiseen ja laadukas koodi toimii omana dokumentaationaan.

ACM:n ohjeiston [ACM/IEEE-CS, 1999] kohta 8.03 kehottaa ohjelmoijaa kehittämään taitojaan tarkan, avuliaan ja hyvin kirjoitetun dokumentaation kirjoittamisessa. Vaikka tämä ohje tuntuisi korostavan dokumentaation merkitystä vastoin manifestin neuvoa, on kyseessä itseasiassa osin sama ohje: dokumentaatiota ei tule kirjoittaa vain sen vuoksi, että projekti tuottaisi dokumentaation, vaan sen on oltava mieluummin avuliasta kuin kattavaa. Manifestin ohjehan ei kiellä dokumentaation kirjoittamista.

Ohjelmiston lähdekoodi kertoo toteutuksen ajantasaisen tilan, joten kun manifestin mukaisesti panostetaan koodin laatuun ja selkeyteen dokumentaationa, on dokumentaatio aina varmasti ajantasainen, toisin kuin erillistä dokumenttia ylläpidettäessä. Tämä koskee toki vain ohjelmiston teknisen toteutuksen dokumentaatiota. Projektissa on usein muutakin ylöskirjattavaa.

Suunnittelu- ja toteutusvaiheet ovat avainasemassa vastatessa kehitysmallin vaatimuksiin tietojenkäsittelytieteen edistämisestä ja laadukkaasta ohjelmistosta. Tässä vaiheessa ohjelmistokehittäjät oppivat ja opettavat toisilleen parempia ohjelmointikäytäntöjä ja sisällyttävät niitä tuottamaansa ohjelmistoon.

Testausvaiheessa sovellusta testataan tämän ja aiempien inkrementtien vaatimuksia vastaan. Tämä vaihe ei siis tarkoita käytettävyystestausta, joka on keino vastata eettisen kehitysmallin vaatimuksiin laadukkaasta ja luotettavasta ohjelmistosta. Käytettävyys nousee vaatimukseksi eettisen kehitysmallin vaatimuksista käyttäjän luottamussuhdetta

ja laadukasta ohjelmistoa rakentavana tekijänä. Koska ohjelmiston käytettävyys on vaatimus ja käytettävyystestaus keino saavuttaa se, tulisi käytettävyystestauksen suorittaminen sisällyttää vaatimukseksi joihinkin valittuihin inkrementteihin, ohjelmistosta riippuen kerran tai useamminkin.

Näiden vaiheiden jälkeen inkrementti on saatettu päätökseen ja voidaan siirtyä seuraavaan inkrementtiin. Inkrementtien välinen siirtymävaihe on kuitenkin huomionarvoisa vaihe. Inkrementin tuottama sovellus arvioidaan sisäisesti ja se olisi hyvä esitellä myös asiakkaalle. Yhdessä asiakkaan kanssa tulisi käydä läpi sovelluksen tila suhteessa lopulliseen tavoitteeseen. Agile Manifesto [2001] painottaa yhteistyötä asiakkaan kanssa, ja inkrementtien välisissä toiminnallisuusarvioissa voikin käydä niin, että asiakas huomaa aiempaa tarkemmin, millaisia ominaisuuksia tarvitsee ja millaisia ei. Näin saadaan tarkennettua seuraavan inkrementin tavoitteet suhteessa koko sovellukseen ja lopullinen sovellus vastaa paremmin sitä, mitä asiakas todella haluaa.

Agile Manifesto esittää myös vaatimuksen, jonka mukaan kehitysryhmän on tasaisin aikavälein arvioitava kuinka se voisi toimia tehokkaammin ja korjata toimintaansa vastaavasti. Manifestissa “tehokas toiminta” on kuitenkin jätetty tarkemmin määrittelemättä, ja onkin luontevaa ymmärtää se kattamaan koko ohjelmistontuotannon tapa, johon eettinen valveutuneisuuskin kuuluu. Luvussa 6.1 todettiin jatkuvan keskustelun olevan hyvä keino opettaa etiikkaa ja niinpä inkrementtien väliset eettiset keskustelut vaikuttavat hyvältä tavalta itsearviontiin ja -kehittymiseen. Myös Yang ja muut [2007] kehoittavat säännöllisesti pitämään sessioita, joissa työntekijöille kerrotaan ja muistutetaan organisaation arvoista. Heidän mukaansa sessiot rakentavat eettistä toimintaa tukevaa organisaatiokulttuuria. Tällaiset sessiot voisi liittää inkrementtien välisiin keskustelutilanteisiin.

Keskustelunvirikkeitä tapaamisiin voi löytää eettisistä ohjeistoista tai viime inkrementin aikaisista tapahtumista, mutta lopulta keskustelun ja ideoinnin tulisi olla avointa, eikä ennaltarajattu aihevalikoima ole mielekäs. Kohdassa 3.3 esiteltiin Aaltosen ja Junkkarin [2001] lista eettistä toimintaa tukevista organisaatiopiirteistä. Tämäkin lista vaatii avoimen keskustelun mahdollisuutta sekä työntekijälle oikeutta esittää poikkeaviakin kannanottoja ja ideoita, erityisesti omaa työtä koskevia.

On syytä huomata, että Agile Manifesto kehottaa välittömään käytäntöjen muuttamiseen sen sijaan, että ideoita kerättäisiin seuraavaa projektia varten. Organisaatio- tai toimintatapojen muutos vie kuitenkin resursseja, mikä tulisi huomioida seuraavan inkrementin rasittavuutta arvioidessa.

6.6 Julkaisu ja jatkotoimenpiteet

Ohjelmistoprojektin julkaisu voi projektista riippuen olla vaiheittaista, aluksi pienelle ydinryhmälle ja myöhemmin suuremmalle joukolle sallittua, tai kertaluontoista, esimerkiksi samantien ympäri maailmaa levitettävänä mediatallenteena. Julkaisutavasta ja aikatauluista myös riippuu, kuinka valmiina ja missä vaiheessa ohjelmisto julkaistaan ja millainen on sen jatkokehitys- tai käyttäjätukivaihe. Joka tapauksessa ohjelmisto on julkaisun jälkeen ei-tietojenkäsittelijöiden käytössä ja se vaikuttaa suuren yleisön mielikuvaan ohjelmistoalan maineesta. Tuki-, asiakaspalvelu- ja kehitystoimenpiteillään kehitysryhmä voi vaikuttaa alan mainetta parantavasti, mikä on eettisten ohjeistojen vaatimus, kuten kohdassa 2.3 kävi ilmi.

Jatkokehitys on tapauskohtaisesti joko oma projektinsa tai vain lisäinkrementtejä ohjelmiston tuottaneeseen projektiin. Kuten luvun 6 johdannossa kävi ilmi, käyttäjätukivaihe, siinä missä esimerkiksi mainontakin, on tämän tutkielman ulkopuolinen asia, eikä sitä tarkastella lähemmin.

Esitelty kehitysmalli puuttuu yksityiskohtaisesti vain osaan tutkielmassa esitellyistä eettisistä vaatimuksista ja kehotuksista. Moni niistä, kuten Aaltosen ja Junkkarin [2001] kehotus organisaation, vastuiden ja roolien selkeään määrittelyyn, ACM:n [ACM/IEEE-CS, 1999] kehotus reiluun ja sopivaan palkkiojärjestelmään tai ympäristönsuojelulain [Ympäristönsuojelulaki, 1§] tavoite ehkäistä jätteiden syntyä, ovat sisällöltään sellaisia, että niiden huomionti on luontevinta organisaatiotasolla, ei niinkään yksittäisen ohjelmistoprojektin kehityksen yhteydessä. Pelkkä kehitysmallin vaihtaminen ei siis ole riittävä keino kaikkien ohjelmistoalan eettisten ongelmien huomioinniksi.

7 Yhteenveto

Tutkielmassa käsiteltiin ohjelmistokehityksen eettisiä kysymyksiä ja pyrittiin luomaan keinoja niiden huomiointiin. Ohjelmistoalalle ominaiset ongelmat sijoitettiin etiikan historian kontekstiin ja esiteltiin joitain alalla työskenteleville suunnattuja eettisiä ohjeistoja. Ohjeistojen havaittiin nostavan esille useita tärkeitä näkökulmia, mutta olevan itsessään riittämättömiä työkaluna ohjelmistokehitykseen liittyvien ongelmien välttämiseen ja käsittelyyn.

Ohjelmiston käyttöä lähestyttiin luottamuksen näkökulmasta ja sijoitettiin yksittäinen ohjelmisto sen tuottaneen yrityksen organisaatiokulttuurin tuotteeksi. Yrityksen eettistä toimintaa tukevan organisaatiokulttuurin, asiakas-yritys -luottamussuhteen ja

kaupallisen menestyksen välillä havaittiin synergiaa ja esiteltiin joitain yritystasolla toteutettavia keinoja parempaan eettisten kysymysten huomiointiin.

Tutkielmassa esiteltiin ohjelmistojen rakentamisprosessia erilaisilla kehitysmalleilla ja arvioitiin valitun kehitysmallin vaikutusta ohjelmistoprojektin aikana kohdattaviin eettisiin ongelmiin. Teoreettisten viitekehysten lisäksi ohjelmistokehityksen eettisiä ongelmia tutkittiin myös käytännön projektissa. Tutkimuksen tulokset korostivat yritys-asiakas -suhteen merkitystä eettisten ongelmien ilmenemisessä ja ratkaisussa.

Yksilö- ja yritystasolle suunnatut ohjeet ja kehoituksen koottiin yhteen muodostamalla ohjelmistokehitysmalli toteutusesimerkiksi siitä, kuinka etiikan huomiointi yksittäisen ohjelmistoprojektin aikana onnistuisi. Ohjelmistokehitysmalli rakennettiin inkrementaalisen mallin pohjalle, mutta siinä huomioitiin myös monia ketterien menetelmien ominaisuuksia. Esitelty malli sisältää keinoja ennaltaehkäistä ja ratkaista eettisiä ongelmia, mutta tutkielmassa myös havaittiin, että eettisten näkökulmien huomiointi vaatii yksittäistä ohjelmistoprojektia suurempaa toimikenttää, eli joidenkin eettisten näkökulmien huomiointia organisaatiotasolla.

Tutkielmassa havaittiin myös, että eettisesti valveutuneesta kehitysmallista on mielekästä puhua vain, kun on tarjota yksityiskohtainen esitys siitä, mikä ohjelmistokehittäjien mielestä on eettistä. Tässä tutkielmassa esitellyn kehitysmallin etiikkakäsityksenä on käytetty useiden eettisten ohjeistojen ja muiden kehotusten synteesia, ja olisikin mielenkiintoista nähdä, kuinka erilainen kehitysmalli voisi olla, mikäli sen etiikkakäsityksenä käytettäisiin jotakin tyystin erilaista.

Vaikka tutkielmassa esiteltiin vain yksi kehitysmalliesimerkki, käytetyn etiikkakäsityksen pohjalta nousseet toimintamallit voisi koettaa integroida myös joihinkin toisiin kehitysmalleihin. Organisaatiossa jo käytössä olevan kehitysmallin vaiheittainen kohentaminen esitellyillä ideoilla saattaa olla helpompaa ja mielekkäämpää kuin täysin uuden kehitysmallin käyttöönotto.

Tutkimusta olisi kuitenkin hyvä jatkaa kokeilemalla esitetyn kehitysmallin käyttöönottoa. Käytännön testaus toisi luultavasti ilmi joitain ongelmia, joiden pohjalta mallia voisi parantaa. Lisäksi olisi hyvä kehittää jonkinlainen menetelmä, jolla voisi vertailla eri kehitysmallien vaikutusta ohjelmistoprojektin yleiseen eettiseen onnistumiseen. Tämän tutkielman empiriaosuudessa käytetyt haastattelu- ja lomakekysymykset voisivat toimia tällaisen menetelmän pohjana, mutta tutkielmassa on myös useasti käynyt ilmi, että tarkistuslistamainen lähestymistapa etiikkaan on

ongelmallinen. Käytännön ohjelmistoprojekteissa käytettyjen kehitysmallien ja projektien tuotosten laaja-alainen vertailu olisi myös keino havaita uusia välineitä etiikan huomiointiin.

Lähteet

[Aaltonen ja Junkkari, 2003] Tapio Aaltonen ja Lari Junkkari, *Yrityksen arvot & etiikka*, WSOY, 2003.

[Abrahamsson et al., 2002] Pekka Abrahamsson, Outi Salo, Jussi Ronkainen and Juhani Warsta, Agile software development methods – review and analysis, VTT Publications 478, 2002.

[Agile Manifesto, 2001] Manifesto for Agile Software Development, 2001, http://agilemanifesto.org/ ja http://agilemanifesto.org/principles.html, checked 09.09.2009.

[Airaksinen, 1995] Timo Airaksinen, *Lukion filosofia: filosofisen etiikan perusteet,* Otava, 1995.

[ACM, 1992] ACM Council*,* ACM Code of Ethics and Professional Conduct, 1992, http://www.acm.org/about/code-of-ethics, checked 17.09.2008.

[ACM/IEEE-CS, 1999] ACM/IEEE-CS joint task force on Software Engineering Ethics and Professional Practices, Software Engineering Code of Ethics and Professional Practice, 1999, http://www.acm.org/about/se-code, checked 17.09.2008.

[Bate, 2007] Steve Bate, Do You Write Green Software?, 2007, http://blog.technoetic.com/2007/03/25/green-software/, checked 23.09.2008.

[Beck, 1999] Kent Beck, *Extreme Programming Explained: Embrace Change*, Addison-Wesley, 1999.

[Bentham, 1789/2005] Jeremy Bentham, *An Introduction to the Principles of Morals and Legislation*, 1789. Kessinger Publishing, 2005.

[Berleur and Brunnstein, 1996] Jacques Berleur and Klaus Brunnstein, *Ethics of Computing*, International Federation for Information Processing, 1996, http://books.google.com/books?id=56I2wXbEv_YC, checked 23.02.2009.

[Braithwaite, 2007] Reginald Braithwaite, Which theory fits the evidence, 2007. http://weblog.raganwald.com/2007/06/which-theory-first-evidence.html, checked 14.03.2010.

[Corritore et al., 2001] Cynthia L. Corritore, Susan Wiedenbeck and Beverly Kracher, The Elements of Online Trust, *Proceedings of CHI 01*, March 2001, pp. 504-505, NY: ACM, 2001.

[Elshamy and Elssamadisy, 2007] Ahmed Elshamy and Amr Elssamadisy, Applying Agile to Large Projects: New Agile Software Development Practices for Large Projects, *Lecture Notes in Computer Science 4536,* Springer, 2007.

[Giddens, 1991] Anthony Giddens, *The Consequences of Modernity*, Stanford University Press, 1991.

[Gilb, 1981] Tom Gilb, *Evolutionary Development, Software Engineering Notes*, 6, 2, (April 1981), 17.

[Gotterbarn, 1992] Don Gotterbarn, Do the Right Thing, *Chief Information Offices Journal* 5, 12, (May 1992), 26-27.

[HE 126/2006 vp] Hallituksen esitys Eduskunnalle laiksi tekijänoikeuslain 19 §:n muuttamisesta. http://www.finlex.fi/fi/esitykset/he/2006/20060126.pdf, haettu 13.07.2009.

[Highsmith, 2000] James A. Highsmith, *Adaptive Software Development: A Collaborative Approach to Managing Complex Systems*, Dorset House Pub., 2000.

[Hirvonen, 2000] Päivi Hirvonen, *Eettiset ohjeistot tietotekniikka-alalla*, Pro gradu -tutkielma, Joensuun yliopisto, 2000.

[Hughes and Cotterell, 2006] Bob Hughes and Mike Cotterell, *Software Project Management*, 2006, McGraw-Hill Education.

[Jargonfile] The Jargon File, http://www.catb.org/jargon/html/H/hacker-ethic.html, checked 19.09.2008.

[Jargonfile archive] JARGON FILE TEXT ARCHIVE, http://jargon-file.org/archive/, checked 20.09.2008.

[Journalistiliitto] Journalistin ohjeet, https://www.journalistiliitto.fi/Resource.phx/sivut/sivut-journalistiliitto/pelisaannot/journalistinohjeet/uudet.htx, haettu 19.09.2008.

[Kaipainen, 2009] Kirjeenvaihto Tietotekniikan Liitto ry:n etiikan työryhmän kanssa.

[Koehn, 2003] Daryl Koehn, The Nature of and Conditions for Online Trust, *Journal of Business Ethics* 43, 3 (2003) 3-19.

[Marovic et al., 2009] Branko Marovic, Gina Kramer, Marcin Wrzos, Marek Lewandowski, Andrzej Bobak, Antoine Delvaux, Waldemar Zurowski, Tihana Žuljevic, Candido Rodriguez, Spass Kostov, Ognjen Blagojevic, Ian Thomson, Rade Martinovic and Szymon Kupinski, *GN3 Software Architecture Strategy Best Practice Guide 1.0 ,* Géant, 2009. http://www.geant.net/Media_Centre/Media_Library/Media%20Library/GN3-09-185v2_Software_Architecture_Strategy_Best_Practice_Guide_1.0.pdf, checked 02.05.2010.

[Milgram, 1974] Stanley Milgram, Obedience to Authority, 1974. http://www.panarchy.org/milgram/obedience.html, checked 20.09.2008.

[Mill, 1864] John Stuart Mill, *Utilitarianism*, Parker, Son and Bourn, 1864. http://www.archive.org/download/a592840000milluoft/a592840000milluoft.pdf, checked 20.09.2008.

[Moeller, 2002] Steven T. Moeller, *Energy Efficiency: Issues and Trends*, Nova Publishers, 2002.

[Myskja, 2008] Bjørn K. Myskja, The categorical imperative and the ethics of trust, 2008. http://www.springerlink.com/content/a741k63527434324/?p=c1f571e5880f461f80bf43c347bfcd7e&pi=1, checked 14.01.2009.

[Mäkinen, 2006] Olli Mäkinen, *Internet ja etiikka*, Gummerus, 2006.

[Raymond, 2000] Eric S. Raymond, *The Cathedral and the Bazaar*, version 3.0, 2000. http://www.catb.org/~esr/writings/cathedral-bazaar/cathedral-bazaar/, checked 03.09.2009.

[Rational, 1998] Rational Software, *Rational Unified Process – Best Practices for Software Development Teams*, 1998.
http://www.augustana.ab.ca/~mohrj/courses/2000.winter/csc220/papers/rup_best_practices/rup_bestpractices.pdf, checked 02.05.2010.

[Rikoslaki] Rikoslaki, 34 luku, 9 a §, muutossäädös 11.5.2007/540.
http://www.finlex.fi/fi/laki/ajantasa/1889/18890039001, haettu 01.05.2010.

[Rogerson and Gotterbarn, 1997] Simon Rogerson and Don Gotterbarn, The Ethics of Software Project Management, 1997.
http://www.ccsr.cse.dmu.ac.uk/staff/Srog/teaching/sweden.htm, checked 17.09.2008.

[Rogerson et al., 2001] Simon Rogerson, Steve McRobb and Ben Fairweather, New Code of Ethics, 2001.
http://www.ccsr.cse.dmu.ac.uk/resources/general/ethicol/Ecv11no1.html, checked 30.03.2009.

[Saarinen, 1994] Esa Saarinen, *Filosofia*, WSOY, 1994.

[Schwaber, 2004] Ken Schwaber, *Agile Project Management with Scrum*, Microsoft Press, 2004.

[Schwaber and Sutherland, 2010] Ken Schwaber and Jeff Sutherland, SCRUM, 2010.
http://www.scrum.org/storage/scrumguides/Scrum%20Guide.pdf, checked 02.05.2010.

[Stoodley, 2009] Ian Stoodley, *IT professionals' experience of ethics and its implications for IT education*, PhD Thesis, Faculty of Science and Technology, Queensland University of Technology, Brisbane, Australia, 2009.

[Talouselämä, 2008] Taru Taipale, Talouselämä, Wahlroos sysäsi moraalikysymykset omistajille, http://www.talouselama.fi/uutiset/article167692.ece, haettu 09.02.2009.

[Tekijänoikeuslaki, 2006] Tekijänoikeuslaki, 19§, muutossäädös 22.12.2006/1228. http://www.finlex.fi/fi/laki/ajantasa/1961/19610404, haettu 01.05.2010.

[Thomson and Schmoldt, 2001] Alan J. Thomson and Daniel L. Schmoldt, Ethics in computer software design and development, 2001. *Computers and Electronics in Agriculture*, 30, 1-3, (February 2001), 85-102.

[Tieto] Tieto lyhyesti, http://www.tieto.fi/default.asp?path=408,409, haettu 18.10.2009.

[Tieto, Q2] Tieto Oyj, osavuosikatsaus 2/2009, 17.7.2009, 8.00. http://www.tieto.com/binary.asp?GUID=D8383CC6-D93A-4DBB-AD83-0E38E9FE78DA&field=Document, haettu 18.10.2009.

[Tukiainen, 1999] Arto Tukiainen, *Etiikka ja filosofia – Eräiden Wittgensteinin näkemysten kritiikkiä*, Väitöskirja, Helsingin yliopisto, Valtiotieteellinen tiedekunta, Käytännöllisen filosofian laitos. Helsingin yliopiston filosofian laitoksen julkaisuja 3/1999.

[Vartiainen, 2009] Tero Vartiainen, Four Levels of Moral Conflicts in ISD. In Papadopoulos, G. A., Wojtkowski, W., Wojtkowski, W. G., Wrycza, S., & Zupancic, J. (eds) (2009), Information Systems Development: Towards a Service Provision Society, Springer, New York, 2009, 811-819.

[Witten and Frank, 2000] Ian H. Witten and Eibe Frank, *Data Mining*, Academic Press, 2000.

[WWF] WWF, Helka Julkunen, Green Office, http://www.wwf.fi/yritykset/green_office/, haettu 4.6.2009.

[Ympäristönsuojelulaki, 1§] Ympäristönsuojelulaki, 4.2.2000/86. http://www.finlex.fi/fi/laki/ajantasa/2000/20000086, haettu 01.05.2010.

[Yang et al., 2007] Yinan Yang, Ed Lewis and Lawrie Brown, Cultural and Social Aspects of Security and Privacy – The Critical Elements of Trusted Online Service. Usability and Internationalization, Part II, HCII 2007, LNCS 4560. Springer, 2007, 533-546.

Liitteet:

Liite 1: Haastattelukysymykset

Kysymyksiä Sanastosta

- Mikä on Sanasto-järjestelmä?
- Kuka on tilannut Sanasto-järjestelmän?
- Mitä korvattavan järjestelmän puutteita uuden järjestelmän on tarkoitus paikata?
- Keiden hyvinvointia uusi järjestelmä parantaa?
- Keiden hyvinvointia uusi järjestelmä heikentää?
- Mitkä tahot ovat olleet edustettuina vaatimusmäärittelyssä?
- Minkä tahojen etuja vaatimusmäärittelyssä on pohdittu ilman läsnäolevaa edustajaa?
- Millaisia ehdotettuja ominaisuuksia on päätetty jättää toteuttamatta?
  - Miksi nämä jätetään tekemättä?
    - Onko Eduix kieltäytynyt toteuttamasta joitain ominaisuuksia?
  - Toteutetaanko joitain näistä ominaisuuksista myöhemmin?
- Millaisia ympäristövaikutuksia järjestelmällä arvellaan olevan?
  - Entä verrattuna korvattavaan järjestelmään?
- Mitä tietoa kirjailijoista tallennetaan järjestelmään?
- Kenellä on oikeus tarkastella järjestelmään tallennettavaa tietoa?
- Kuinka kirjailijoiden yksityisyydensuojaa varjellaan?
- Kenellä on oikeus katselmoida ohjelmakoodia?
  - Jos ulkopuolisilla, onko katselmointeja sovittu/odotettavissa?
  - Miten arvioitte asiakkaan suunnittelemien katselmointien vaikuttavan ohjelmakoodin laatuun?
- Kuinka Sanasto-järjestelmästä tiedotetaan sen tavoitelluille käyttäjille?
- Kuinka Sanasto-järjestelmästä tiedotetaan julkiselle yleisölle?
- Mistä olemassa olevista eettisistä ohjeistoista olette tietoisia?
  - Aiotaanko näitä huomioida projektin aikana? Jos, niin kuinka?
- Millaisia eettisiä ongelmia projektissa on jo havaittu?
  - Miten niitä on ratkaistu?
- Kuinka eettiset asiat pitäisi teidän mielestänne ylipäänsä huomioida ohjelmistokehityksessä?
- Kuinka korkeaan laatuun ohjelmistoa tuottaessa tulee pyrkiä?
  - Kuinka toimia tilanteessa, jossa asiakas ei vaadi ohjelmistolta yhtä korkeaa laatua kuin sen toimittaja? (Esimerkiksi tietoturvan tai standardien suhteen)
- Millaista etua tai haittaa eettinen ohjelmistonkehitys antaa kaupallisilla markkinoilla?
- Millainen on yrityksen yhteiskuntavastuu?
  - Kuinka se näkyy Eduixissa Sanasto-järjestelmän osalta?

Liite 2: Kyselytutkimuksen kysymyslomake

Lainauskorvausjärjestelmä

Tämä on kysely Sanaston tilaaman lainauskorvausjärjestelmän kehityksestä. Kysely on osa Iikku Mattilan pro gradu -tutkielmaa ja suunnattu kaikille kehitysprosessissa mukana olleille. Vastaa kysymyksiin omasta näkökulmastasi.

Kyselyssä ei kysytä henkilötietoja, eikä tutkimustuloksia julkaistaessa vastauksia voida yhdistää vastaajiin. Vastaukset julkistetaan vain pro gradu -työssä ja pidetään muutoin luottamuksellisina.

Kyselyn avulla tutkitaan ohjelmistokehitysprosessin eettisiä ulottuvuuksia, eli ohjelmistoa kehittävän yrityksen, sen työntekijöiden, ohjelmiston tilanneen asiakkaan ja ympäröivän yhteiskunnan välisiä suhteita, sopimuksia, toiminnan periaatteita ja tavoitteita.

Perustiedot

Montako kuukautta olet ollut mukana projektissa?
Montako vuotta olet työskennellyt ohjelmistoalalla?

Järjestelmä

Järjestelmä ja sen tarkoitus

	Ei lainkaan	Hieman	Keskiverrosti	Paljon	Täysin
Kuinka hyvin järjestelmä täyttää tarkoituksensa?					
Kuinka tyytyväinen luulet asiakkaan olevan toteutuneeseen järjestelmään?					
Kuinka lähellä toteutunut järjestelmä on alun perin suunniteltua järjestelmää?					

Yleistä

	Ei lainkaan	Hieman	Keskiverrosti	Paljon	Täysin
Kuinka helppokäyttöinen järjestelmä on?					
Kuinka energiatehokas järjestelmä on?					
Paljonko koet järjestelmän lisäävän oikeudenmukaisuutta ?					
Paljonko koet järjestelmän lisäävän ihmiskunnan hyvinvointia?					

Ammatillinen kehittyminen

Ammatillinen kehittyminen

	En lainkaan	Jonkin verran	Paljon
Koetko kehittyneesi ammatissasi tämän projektin aikana?			

Kehittymisen riittävyys

	Riittämättömäksi	Riittäväksi	Odotettua runsaammaksi
Kuinka riittäväksi olet kokenut ammatillisen kehityksesi projektin aikana?			

Työn laatu

	Todella huonoa	Huonoa	Keskivertoa	Hyvää	Todella hyvää
Kuinka laadukasta työtä olet tehnyt?					
Kuinka laadukasta työtä muu kehitysryhmä on mielestäsi tehnyt?					

Kehitysyhteisö

Kommunikaatio

	Tällaista ei ollut	Lähinnä ongelmallista	Kelvollista	Sujuvaa
Millaista kommunikaatio kollegoidesi kanssa on ollut?				
Millaista kommunikaatio esimiehesi kanssa on ollut?				
Millaista kommunikaatio alaistesi kanssa on ollut				
Millaista kommunikaatio asiakkaan kanssa on ollut?				

Sisäinen palaute

	Ei lainkaan	Niukasti	Jonkin verran	Runsaasti
Paljonko olet arvioinut tai kommentoinut muun ohjelmistokehitystyöryhmän työtä?				
Paljonko ohjelmistokehitystyöryhmän muut jäsenet ovat arvioineet tai kommentoineet työtäsi?				

Aikataulutus

	Aivan liian vähän	Hieman liian vähän	Riittävästi	Mukavasti	Aivan liian paljon
Kuinka paljon sinulla oli aikaa tehdä työsi?					

Henkilöstöresurssit

	Ei koske	Työtä enemmän	Työmäärä ja resurssit	Resursseja enemmän

	minua	kuin resursseja	tasapainossa	kuin työtä
Onko työmäärien ja henkilöstöresurssien yhteensovittaminen onnistunut?				

Lopuksi

Millaisia eettisiä valintoja ohjelmistokehityksen aikana joutuu mielestäsi yleensä tekemään?

Millaisia eettisiä valintoja olet tehnyt lainauskorvausjärjestelmää kehittäessäsi?