Kohti hajautettua ohjelmistokehitystä

Juha Rinne

Tampereen yliopisto
Tietojenkäsittelytieteiden laitos
Pro gradu -tutkielma
Marraskuu 2001

Tampereen yliopisto
Tietojenkäsittelytieteiden laitos
Juha Rinne: Kohti hajautettua ohjelmistokehitystä
Pro gradu –tutkielma, 54 sivua
Marraskuu 2001

Tiivistelmä

Tämän tutkielman tavoitteena on luoda metodi, jolla voidaan menestyksekkäästi siirtyä paikallisesta ohjelmistokehityksestä hajautettuun. Työssä pyritään tunnistamaan erot perinteisen, yhdellä paikkakunnalla tapahtuvan, ja hajautetun ohjelmistokehityksen välillä. Tästä edelleen tunnistetaan ongelmat, joita siirtyminen hajautettuun ohjelmistokehitykseen aiheuttaa, sekä hahmotellaan ratkaisu näiden ongelmien poistamiseksi tai minimoimiseksi. Lisäksi esitellään eräs projekti, jossa toiminnan hajautus aiheutti ongelmia. Tämä projekti toimi alkusysäyksenä koko tutkielmalle. Lopuksi hahmotellaan järjestelmää, jolla joitakin ongelmia voitaisiin ratkaista ja siten tehostaa hajautettua ohjelmistokehitystä.

Tutkielman keskeinen havainto on, että kun siirrytään hajautettuun työhön on tärkeintä, lähtötilasta riippuen, säilyttää tai kehittää toimiva viestintä. Muut hajautuksesta aiheutuvat ongelmat liittyvät pääasiassa ohjelmistokehityksen tukitoimintoihin. Viestintää ja tukitoimintoja vahvistamaan tarvitaan toimivat ja tehokkaat työkalut sekä hyvin määritellyt toimintatavat.

Avainsanat ja -sanonnat:

Ohjelmistotuotanto, hajautettu ohjelmistokehitys, viestintä, dokumenttien hallinta, DeltaV, konfiguraationhallinta, projektin hallinta, koordinointi, ongelmienhallinta

Sisällysluettelo

1. Johdanto

Tämän tutkielman tavoitteena on luoda metodi, jolla voidaan menestyksekkäästi siirtyä paikallisesta ohjelmistokehityksestä hajautettuun. Työssä pyritään tunnistamaan erot yhdessä paikassa tehtävän ja hajautetun ohjelmistokehityksen välillä. Tästä edelleen tunnistetaan ongelmat, joita siirtyminen hajautettuun ohjelmistokehitykseen aiheuttaa, sekä hahmotellaan ratkaisu näiden ongelmien poistamiseksi tai minimoimiseksi.

Tutkielman aihepiiri on ohjelmistokehitys, ohjelmistotuotanto tai ohjelmistotekniikka, kuitenkin tarkoittaen ohjelmistotyötä, jonka tuloksena syntyvät järjestelmät täyttävät käyttäjiensä kohtuulliset toiveet ja odotukset ja tämän lisäksi valmistuvat laadittujen aikataulujen ja kustannusarvioiden puitteissa [Haikala ja Märijärvi, 1998]. On esitetty arvioita, että nykyiset laajimmat ohjelmistokokonaisuudet ovat monimutkaisimpia ihmisten suunnittelemia ja toteuttamia hankkeita [Cugola ja Ghezzi, 1999]. Tämä monimutkaisuus tuo luonnollisesti mukanaan koko joukon ongelmia. Ohjelmistoista on tullut monilla aloilla kehitystä eteenpäin ajava voima [Pressman, 1997]. Ne vaikuttavat tärkeiden päätösten tekoon. Ne toimivat modernin tieteellisen tutkimuksen ja tekniikan ongelmanratkaisun perustyökaluina. Ne erottavat uudenaikaiset tuotteet ja palvelut entisistä. Niitä on kaikkialla, ne ovat osa ihmisten jokapäiväistä elämää. Suurimpia ohjelmistoihin liittyviä ongelmia ovat ohjelmistovirheet. Kun ohjelmistovirheestä aiheutuu järjestelmän virhetilanne, tapaus saa usein paljon julkisuutta. Monesti aiheutuu suuria taloudellisia tappioita.

Ohjelmistoprojekti jaetaan perinteisesti karkeimmillaan seuraaviin vaiheisiin: määrittely, suunnittelu, toteutus, testaus ja ylläpito. Näiden lisäksi on joitakin tärkeitä tukitoimintoja, kuten tuotteenhallinta (konfiguraationhallinta), laadunvarmistus ja dokumentointi. On olemassa monia erilaisia ohjelmistokehitysprosessin malleja, joiden yhteisiä nimittäjiä nämä osat ovat. Kuhunkin näistä osa-alueista liittyy omat käsitteistönsä, ongelmansa, menetelmänsä ja työkalunsa. En ole tässä tutkielmassa kiinnostunut prosessimallista sinänsä, vaan käytän ongelman määrittelyn ja tutkimisen ympäristönä erästä iteratiivista mallia.

Ohjelmistotuotteiden jatkuvasti suureneva koko ja monimutkaisuus aiheuttavat mm. sen, että sitä tekevien ihmisten lukumäärä kasvaa. Kokemuksesta tiedetään, että kun ohjelmistoprojektin henkilömäärä nousee, ryhmän tuottavuus todennäköisesti kärsii [Pressman, 1997]. Eräs tapa ratkaista tämä ongelma on luoda useita kehittäjäryhmiä.

Nykyisin nämä ryhmät ovat usein vielä sekä maantieteellisesti että ajallisesti erillisiä, siis hajautettuja. Ryhmissä tehdään rinnakkaista kehitystä.

Työvoima ja osaaminen on maantieteellisesti hajautunutta. Se tietää monille organisaatioille joko sisäistä hajautumista, ulkoistamista tai verkostoitumista. Ulkoistaminen on eräs tapa lisätä joustavuutta ja lyhentää kehitykseen käytettyä aikaa. Aika onkin yksi tärkeimmistä kilpailutekijöistä. On äärimmäisen tärkeätä, että tuotteet saadaan nopeasti ja aikataulun mukaisesti markkinoille. Ulkoistaminen, alihankinta ja verkostoituminen mahdollistavat osapuolten keskittymisen ydinosaamiseensa [VeTO, 2001]. Tavoitteena on kaikkien osapuolten hyöty, siis taloudellisen voiton saavuttaminen. Vähentynyt matkustaminen ja lisääntynyt rinnakkaisuus kehityksessä aikaansaavat kustannussäästöjä [Alho ja Sulonen, 1998].

Ohjelmistoprojektien asiakkaat toimivat nykyään usein globaalisti tai sijaitsevat ulkomailla. On yhteistyön molemmille osapuolille edullista, että osapuolet sijaitsevat maantieteellisesti lähellä toisiaan. Osaltaan hajautumiseen ovat vaikuttaneet myös sen mahdollistaneet teknologiat, joista tärkeimpänä voidaan pitää Internetin nousua tärkeäksi sekä tarpeeksi nopeaksi ja tehokkaaksi työvälineeksi.

Eräs ajankohtainen hajautetun ohjelmistokehityksen edustaja on avoimen lähdekoodin ohjelmistokehitys. Nämä projektit edustavat yhtä hajautuksen ääripäätä, käytännössä kaikki kehittäjät ovat maantieteellisesti ja ajallisesti erillään. Cubranic hahmottaa aihetta artikkelissaan [Cubranic, 1999]. Avoimen lähdekoodin ohjelmistokehitys on jo jonkin aikaa olemassaollut, mutta vasta nyt julkisuutta saavuttanut ohjelmistokehityksen muoto.

Useimmat hajautettuun ohjelmistokehitykseen liittyvät tutkimukset ovat keskittyneet mallintamaan hajautettua ohjelmistoprojektia ja tältä pohjalta asettaneet suuntaviivat hajautettua ohjelmistokehitystä tukevalle kehitysympäristölle tai työkalulle. Myös prosessin mallinnusta ja sen tukemista työkaluilla on tutkittu. Lisäksi näkökulmaan liitetään toisinaan jokin muu tutkimuskohde kuten esimerkiksi tietämyksenhallinta [Maurer ja Dellen, 1998] tai projektin johtaminen ja koordinointi [Bendeck *et al*., 1998]. Myös avoimen lähdekoodin ohjelmistokehitystä tutkitaan jonkin verran. Nämä tutkimukset ovat pääasiassa aihetta hahmottelevia. Joitakin mielenkiintoisia aiheita, joissa mielestäni lähestytään hajautetun ohjelmistotyön oleellisimpia ongelmia, ovat mm. hajautetun projektin organisoinnin näkökulma [Altmann ja Dopler, 1998], ongelmien ja ristiriitatilanteiden hallinta [Gao *et al*., 2000; Kudo *et al*., 1998], ihmisten osuus ohjelmistotyössä [Cockburn, 1999; Pape, 1996] sekä jotkin

konfiguraation- ja dokumenttien hallintaa [Hunt ja Reuter, 2001] käsittelevät tutkimukset.

Lähtökohta tälle tutkielmalle oli eräässä hajautetusti toteutettavassa ohjelmistoprojektissa havaitut ongelmat ja puutteet. Tätä projektia käytetään esimerkkinä sekä mallina, johon tulokset suhteutetaan.

Tutkielman keskeinen havainto on, että hajautuksesta aiheutuvat ongelmat liittyvät pääasiassa viestintään ja ohjelmistokehityksen tukitoimintoihin. Prosessin ei tarvitse muuttua, toimiva ohjelmistokehitys on pohjimmiltaan samanlaista kaikenlaisissa ryhmissä. Hieman kärjistäen voidaan tutkielman ongelmat tiivistää kolmeen ryhmään, viestintään, dokumenttien hallintaan ja projektin hallintaan. Näiden yläkäsitteiden alta voidaan tunnistaa tarkempia alueita, mm. projektin hallintaan sekä projektin toimintoja tukeviin työkaluihin liittyviä. Viestintä ja ristiriitojen hallinta vaativat enemmän huomiota hajautetussa ryhmässä, koska on vähemmän epämuodollista viestintää. Ihmislähtöisyys liiketoiminnassa ja erityisesti asiakassuhteissa mahdollistaa hyvän menestyksen.

Dokumenttien käsittelyn ja muokkauksen on oltava mahdollisimman helppoa ja läpinäkyvää. Näin vältetään turhaa työtä ja vääristä versioista johtuvia ongelmia. Kaikkien dokumenttien tulee olla versioituja. Kaikilla projektin jäsenillä tulee olla pääsy ja muokkausoikeudet kaikkiin projektin dokumentteihin. Dokumenttien käsittelyyn on hyvä määritellä joitakin toimintatapoja.

Tarvitaan toimiva konfiguraationhallintajärjestelmä. DeltaV –protokolla tarjoaa tuleville konfiguraationhallintajärjestelmille ja kehitysympäristöille monia mahdollisuuksia. Tällä hetkellä vaihtoehtoja on kaksi: olemassaolevan ratkaisun täydentäminen tukemaan hajautettua työtä tai sellaisen kokonaan uuden ratkaisun hankkiminen, joka tukee hajautettua työtä.

1.1 Käsitteitä ja pohdintaa

Käsitteistö aiheelle saadaan lähes suoraan ohjelmistotuotannon piiristä. On joitakin yksittäisiä käsitteitä, jotka ovat ominaisia hajautetulle työlle. Puhutaan *virtuaalisista ohjelmistoprojekteista* [Alho ja Sulonen, 1998] ja *virtuaalisista yrityksistä* [Alho *et al*., 1999]. Kuitenkaan nämä käsitteet eivät varsinaisesti ole oleellisia itse asian ymmärtämiseksi tai edes määrittelemiseksi. Kummassakin on kyse hajautetusta organisaatiosta, joka tekee yhteistyötä jonkin yhteisen tavoitteen saavuttamiseksi.

Fowler esittää, että myös lähdekooditiedostot ovat osa ohjelmiston dokumentaatiota [Fowler, 2000]. Näkemyksen mukaan ne ovat määrittely ja ohje kääntäjälle varsinaisen rakennustyön tekemiseen. Siten ne olisivat verrattavissa rakennuspiirustuksiin. Näin ajattelemalla päästäisiin päätelmään, että itse ohjelmistotuote olisi dokumenteista kääntämällä ja linkittämällä rakennettu kokonaisuus, ajettava ohjelmakoodi. Ihannetapauksessa työ sujuisi niin, että ihmiset suunnittelevat ja koneet rakentavat. Tehdessään näitä yksityiskohtaisia suunnitelmia ihmiset viestivät keskenään ja muodostavat ikään kuin tuon ohjelmiston; he toimivat ohjelmiston osina ja viestivät toistensa kanssa kuin rajapintojen kautta.

Cockburn [Cockburn, 1999] toteaa yli 20 vuoden kokemuksen pohjalta, että ihmiset eivät ole resursseja. Sen sijaan heitä kuvaavat yleisesti seuraavat luonnehdinnat:

- Ihmiset ovat viestiviä olentoja, jotka ovat parhaimmillaan kasvotusten kommunikoidessaan.
- Ihmiset eivät toimi kovinkaan säännönmukaisesti pitkällä aikavälillä.
- Ihmiset ovat vaihtelevia (ajoittain ja paikoittain).
- Ihmiset haluavat yleensä olla aloitteellisia ja tehdä kaiken tarvittavan.

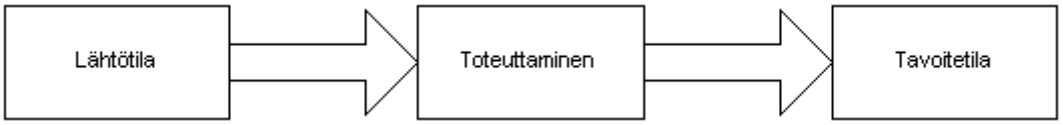
Tämä kaikki on kuitenkin vain käsitteellistä pohdintaa eikä muuta todellisuutta; Ihmiset rakentavat ohjelmistoja ja ryhmänä toimiessaan viestivät keskenään.

1.2 Ongelmanratkaisun lähestymistapa

Tutkielman lähestymistapa aiheeseen on konstruktiivinen [Järvinen ja Järvinen, 2000]. Tutkielmassa arvioidaan *innovaatiota* ja luodaan suunnitelma sen toteutukselle. Järviset määrittelevät innovaation seuraavasti: Innovaatio on ihmisen tai ihmiskollektiivin tavoittelema uudistus, jonka toivotaan tuottavan käyttäjälleen hyötyä. Kuten konstruktiiviselle tutkimukselle on luonteenomaista, lähtökohta on olemassa (kohdat 2.1 ja 6.1). Lähtökohtaa ja tavoitteita arvioidaan, jonka jälkeen edetään käytännöllisempään suuntaan ja tuotetaan suunnitelma tavoitteiden saavuttamiselle. On siis kyse soveltavasta tutkimuksesta. Tavoitetilan määrittely riippuu monista tekijöistä. Se muotoutuu suunnittelijoiden ja muiden asiaan vaikuttavien päättäjien arvoista ja tavoitteista. Konstruktiivisen tutkimuksen tulokset ovat yleensä neljäntyyppisiä: *käsitteistöjä*, *malleja*, *metodeja* ja *realisointeja*.

Tutkielman käsitteistö voidaan lähes sellaisenaan lainata ohjelmistotekniikan tutkimuksesta. Samoin malli, joka ilmaisee käsitteiden välisiä suhteita, saadaan

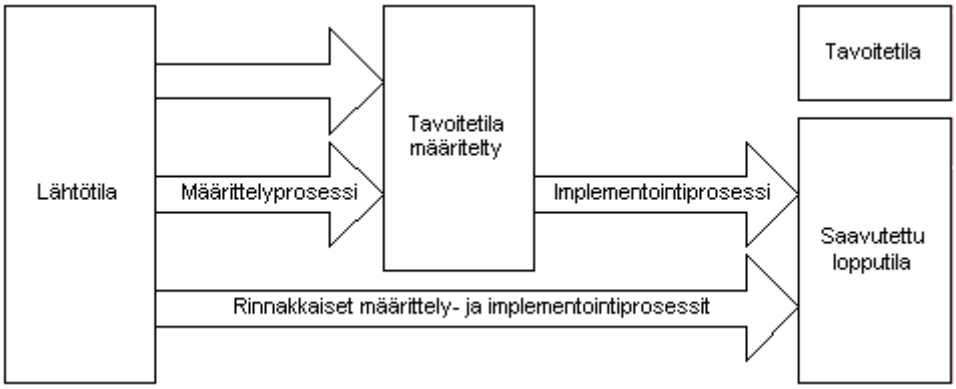
aihepiiristä. Käsitteistön ja mallin pohjalta kartoitetaan lähtötila, sen hyvät ja huonot puolet, sekä tavoitetila, johon pyritään. Metodi on joukko askelia, joita käytetään tehtävän tai innovaation suorittamisessa. Tämän tutkielman tuloksena pyritään luomaan metodi. Realisointiin asti ei mennä. Realisointi tarkoittaisi tässä yhteydessä luvussa 7 esiteltävien ratkaisumallien käyttöönottoa ja luvussa 8 kuvailtavan järjestelmän toteuttamista.



Kuva 1-1 Yleinen innovaation toteuttamisprosessi

Kuva 1-2 Innovaation toteuttamisprosessi tässä tutkielmassa

Kuvassa 1-1 on kuvattu abstraktilla tasolla innovaation toteuttamisprosessi sekä kuvassa 1-2 sama prosessi tässä tutkielmassa. Toteuttaminen (ongelmakenttä), polku lähtötilasta tavoitetilaan, voidaan tehdä ainakin kahdella tavalla, vaihejakomallin mukaisesti tai evolutionäärisesti (kuva 1-3). Pyrin noudattamaan vaihejakomallia, eli määrittelen ensin tavoitetilan ja sen jälkeen suoritan implementointiprosessin. Implementointi ei tässä tarkoita samaa kuin realisointi. Kuten kuvasta 1-3 nähdään, implementointiprosessin tuloksena saavutetaan lopputila. Tässä tapauksessa implementointiprosessi rajoittuu metodin jäsennettyyn suunnitteluun. Tavoitetila on tila, jossa lähtötilassa havaitut ongelmat on korjattu. Implementointiprosessissa toteutetaan tarvittavat korjaukset. Parhaassa tapauksessa saavutettu lopputila on sama kuin tavoitetila.



Kuva 1-3 Tapoja toteuttaa innovaatio

1.3 Tutkielman rakenne

Tutkielma jakautuu lukuihin seuraavasti. Toisessa luvussa todetaan ongelmat ja tarkemmat tutkimustavoitteet. Kolmannessa luvussa hahmotetaan tutkielman aihepiiriä, ohjelmistotuotantoa, ja sen keskeisimpiä piirteitä. Luvussa 4 kuvaillaan tutkielman aihetta, hajautuvaa ohjelmistokehitystä ja kootaan joitakin syitä tämän kehityksen taustalla. Aikaisemmin aiheesta tehtyjä tutkimuksia ja niiden tuloksia kartoitetaan luvussa 5. Esimerkkiprojekti, johon tutkielman lähtökohta ja tulokset liittyvät, kuvataan luvussa 6. Tärkein luku, tutkielman tulokset, esitellään luvussa 7. Edelleen luvussa 8 hahmotellaan tulosten pohjalta niiden implementointia. Luku 9 on varattu keskustelulle aiheen ja tulosten tiimoilta. Joitakin jatkotutkimusaiheita on tunnistettu.

2. Tutkimustavoitteet

Luvun tavoitteena on esitellä tutkimustavoitteet ja edelleen kartoittaa tutkimusaiheen ongelmia. Kohdassa 2.1 selvitetään tutkielman tavoitteita ja selvitetään taustoja, joita vasten ongelmia pyritään ratkaisemaan. Luvun kohdassa 2.2 tarkastellaan hajautetun ohjelmistokehityksen ongelmia.

2.1 Tavoite

Tutkielman tavoitteena on luoda metodi, jolla voidaan menestyksekkäästi siirtyä paikallisesta ohjelmistokehityksestä hajautettuun. Tämä tarkoittaa sitä, että työssä pyritään tunnistamaan erot yhdessä paikassa tehtävän ja hajautetun ohjelmistokehityksen välillä. Tästä edelleen tunnistetaan ongelmat, joita siirtyminen hajautettuun ohjelmistokehitykseen aiheuttaa, sekä hahmotellaan ratkaisu näiden ongelmien poistamiseksi tai minimoimiseksi.

Tutkielman lähtökohta on kasvava ohjelmistotyötä tekevä yritys, joka laajentaa toimintaansa ja hajautuu maantieteellisesti. Tästä johtuen yrityksessä toteutettavat projektit todennäköisesti tulevat hajautumaan yrityksessä olevan osaamisen ja työvoiman mukaan. On myös mahdollista, että yritys hankkii projekteissa tarvitsemaansa työvoimaa muualta ja siten hajautuu.

2.2 Ongelmat

Kun lähdin tutkimaan aihetta, selvisi nopeasti neljä ongelma-aluetta: työkalutuki hajautetulle ohjelmistokehitykselle, dokumenttien hallinta, konfiguraationhallinta ja viestintä. Nämä kaikki on havaittu tutkielman esimerkkiprojektissa, joka esitellään jäljempänä. Myös kirjallisuus tukee tätä käsitystä. Seuraavassa jokaista aluetta on hieman tarkennettu.

2.2.1 Työkalut

Hajautetut projektit useimmiten kärsivät huomattavista ylimääräisistä ajallisista rasituksista ja kustannuksista. Tämä johtuu siitä, että viestintään ja projektin johtamiseen käytetty työmäärä on suurempi, sattuu väärinkäsityksiä, tehdään päällekkäistä työtä ja puuttuu standardoituja toimintatapoja [Alho ja Sulonen, 1998]. Lisäksi usein päädytään laadultaan huonoon lopputulokseen. Yhtenä suurimmista syistä tähän Alho ja Sulonen pitävät sitä, että nykyiset ohjelmistokehitys- ja projektinhallintatyökalut on suunniteltu toimimaan yhden yksikön sisällä.

2.2.2 Dokumentit

On olemassa työkaluja, joita kutsutaan ryhmäohjelmiksi. Ne ovat ohjelmia, jotka on tarkoitettu mm. informaation jakamiseen sekä mahdollisesti dokumenttien yhteistoiminnalliseen muokkaamiseen. Ne ovat yleensä kuitenkin liian erikoistuneita yksittäisiin tehtäviin tai eivät sovellu hajautettuun toimintaan. Erityisesti dokumenttien hallinta hoidetaan hyvin usein manuaalisesti. Tämä tarkoittaa käytännössä, että dokumenteilla ei ole yhtä keskitettyä varastoa, vaan niitä on eri versioina useissa paikoissa, joiden välillä niitä kopioidaan, siirretään ja lähetetään sähköpostin liitteenä manuaalisesti. Joskus tärkeäkin dokumentti voi olla olemassa ainoastaan yhden projektin jäsenen paikallisella kovalevyllä. Tällaiset tapaukset tuottavat huomattavasti turhaa työtä ja sekaannuksia, jotka voitaisiin helposti välttää.

2.2.3 Konfiguraationhallinta

Tutkielman esimerkkiprojektissa on havaittu, että konfiguraationhallinta saattaa olla hyvinkin ongelmallista hajautetussa projektissa. Suurimmat ongelmat konfiguraationhallintaan liittyen ovat niin perustavaa laatua, että niiden ratkaisu on elintärkeää. Nämä ongelmat on tiivistetty kahteen kohtaan:

- Käytössä on konfiguraationhallinta, joka ei tue hajautusta tai on muuten sopimaton.
- Ei käytetä konfiguraationhallintaa oikein tai ollenkaan.

2.2.4 Viestintä

Viestintä on yksi suurimmista ongelmista hajautetuissa projekteissa. Useista yhteyksistä tiedetään, että kasvokkain viestintä on tehokkainta. Lisäksi nykyään jo tunnustettu tosiasia on, että epämuodollinen viestintä on tärkeä osa ihmisten välistä viestintää. Ilman näitä erityisesti ristiriitatilanteet saattavat jäädä ratkaisematta ja kasaantua suuremmaksi konfliktiksi. Jos viestintä projektin sisällä on puutteellista, suurimmaksi vaikeudeksi muodostuu projektin koordinoinnin vaikeus tai oikeastaan sen mahdottomuus. Myös tutkielman esimerkkiprojektin projektisuunnitelmassa on oikein ennustettu, että hajautus, ja erityisesti siihen liittyvä viestinnän vaikeus, koituvat ongelmaksi. On siis selvitettävä miten voidaan varmistaa riittävä ja toimiva viestintä hajautetun projektin sisällä.

3. Ohjelmistotuotanto

Tämä luku kokonaisuudessaan toimii orientaationa ja pohjana tutkielman varsinaisen aiheen käsittelylle. Luvun kohdassa 3.1 määritellään ohjelmisto ja joitakin siihen liittyviä käsitteitä, joita tässä tutkielmassa käytetään. Kohdassa 3.2 esitellään lyhyesti ohjelmistojen merkitystä yhteiskunnassa sekä joitakin yleisiä ongelmia, joita niihin liittyy. Viimeisessä kohdassa 3.3 kuvataan ohjelmiston elinkaari ja siihen liittyvät toiminnot.

Tutkielman aihepiiri on ohjelmistokehitys, ohjelmistotuotanto tai ohjelmistotekniikka, joilla kaikilla tarkoitetaan tässä yhteydessä suunnilleen samaa asiaa. Tarkempi määrittely annetaan jäljempänä kohdassa 3.1. Ohjelmistotuotanto teollisuudenalana on melko nuori, teollisiksi tuotteiksi luokiteltavia ohjelmistoja on tuotettu vasta joitakin vuosikymmeniä. Alan eliniän aikana ohjelmistotuotteet ovat kooltaan ja monimutkaisuudeltaan moninkertaistuneet. On esitetty arvioita, että nykyiset laajimmat ohjelmistokokonaisuudet ovat monimutkaisimpia ihmisten suunnittelemia ja toteuttamia hankkeita [Cugola ja Ghezzi, 1999]. Tämä monimutkaisuus aiheuttaa luonnollisesti koko joukon ongelmia.

3.1 Ohjelmisto

Pressman [Pressman, 1997] määrittelee ohjelmiston seuraavasti: *Ohjelmisto* on 1) käskyjä, jotka suoritettuna tuottavat toivotun toiminnallisuuden ja suorituskyvyn, 2) tietorakenteita, jotka mahdollistavat riittävän informaation käsittelyn sekä 3) dokumentteja, jotka kuvaavat ohjelmien toiminnan ja käytön. Tutkielman kannalta emme tarvitse tämän formaalimpaa tai täydellisempää määritelmää.

Ohjelmistoartefakti (artefakti) on ohjelmiston muodostava osa, esim. lähdekoodi tai määrittelydokumentti [Grundy *et al*., 1998].

Ohjelmistokehitysprosessi (prosessi) on kehys toimille, joita vaaditaan laadukkaan ohjelmiston tuottamiseksi. Prosessi määrittelee lähestymistavan, jota käytetään ohjelmiston tuottamisessa [Pressman, 1997].

Ohjelmistokehitys, *ohjelmistotuotanto* tai *ohjelmistotekniikka* on ohjelmistotyötä, jonka tuloksena syntyvät järjestelmät täyttävät käyttäjiensä kohtuulliset toiveet ja odotukset ja tämän lisäksi valmistuvat laadittujen aikataulujen ja kustannusarvioiden puitteissa [Haikala ja Märijärvi, 1998].

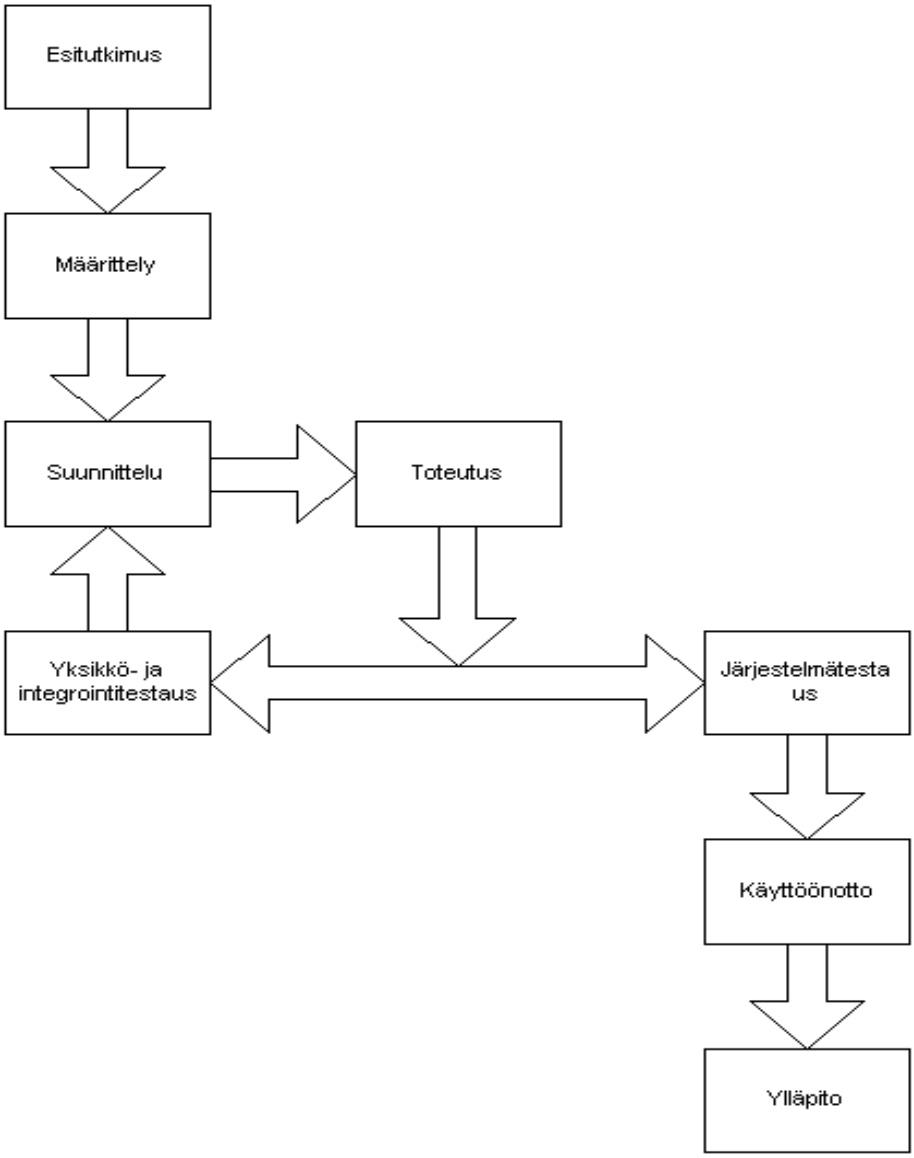
3.2 Ohjelmistojen merkitys

Ohjelmistoista on tullut monilla aloilla kehitystä eteenpäin ajava voima [Pressman, 1997]. Ne vaikuttavat tärkeiden päätösten tekoon. Ne toimivat modernin tieteellisen tutkimuksen ja tekniikan ongelmanratkaisun perustyökaluina. Ne erottavat uudenaikaiset tuotteet ja palvelut entisistä. Niitä on kaikkialla: liikenteessä, lääketieteessä, telekommunikaatiossa, asejärjestelmissä, tehtaissa, viihteessä, toimistoissa, ja tulevaisuudessa aina vain enemmän. Niistä tulee, tai ne oikeastaan jo ovat, osa jokapäiväistä elämäämme. Ihmiset käyttävät niitä sekä tietoisesti että tiedostamattaan.

Suurimpia ohjelmistoihin liittyviä ongelmia ovat ohjelmistovirheet. Usein suurimmat ja monimutkaisimmat ohjelmistot, joihin väistämättä jää ainakin loogisia virheitä, toimivat erittäin vaativissa ja tärkeissä tehtävissä. Kun ohjelmistovirheestä aiheutuu järjestelmän virhetilanne, tapaus saa usein paljon julkisuutta. Monesti aiheutuu suuria taloudellisia tappioita. Pahimmissa tapauksissa vaarannetaan ja jopa menetetään ihmishenkiä. Yrityksissä suurimmat ohjelmistoihin ja niiden kehitykseen liittyvät ongelmat ovat liian korkeat kustannukset, venyneet aikataulut ja joskus täysin epäonnistuneet projektit [Haikala ja Märijärvi, 1998].

3.3 Ohjelmistoprojekti

Ohjelmistoprojekti jaetaan perinteisesti karkeimmillaan seuraaviin vaiheisiin: määrittely, suunnittelu, toteutus, testaus ja ylläpito. Näiden lisäksi on joitakin tärkeitä tukitoimintoja, kuten tuotteenhallinta (konfiguraationhallinta), laadunvarmistus ja dokumentointi. On olemassa monia erilaisia ohjelmistokehitysprosessin malleja, joiden yhteisiä nimittäjiä nämä osat ovat. Kuhunkin näistä osa-alueista liittyy omat käsitteistönsä, ongelmansa, menetelmänsä ja työkalunsa. En ole tässä tutkielmassa kiinnostunut prosessimallista sinänsä, vaan käytän ongelman määrittelyn ja tutkimisen ympäristönä erästä iteratiivista mallia. Iteratiivisuus ilmenee suunnittelu–toteutus–testaus –syklinä, jossa inkrementaalisesti toteutetaan uusia versioita tuotettavasta ohjelmistosta. Prosessia on mallinnettu kuvassa 3-1.



Kuva 3-1 Iteratiivinen prosessi

3.3.1 Vaihejako

Yllä esitellyn vaihejaon mukaisesti ohjelmiston elinkaari jaetaan vaiheisiin [Haikala ja Märijärvi, 1998]. Määrittelyä saattaa edeltää esitutkimus, jonka tehtävänä on asettaa yleiset vaatimukset. Sitä saatetaan kutsua myös asiakasvaatimusten määrittelyksi. Usein esitutkimus mielletään osaksi määrittelyvaihetta, koska se määrittelee perusvaatimukset ja kuvailee sovellusaluetta. Edelleen määrittelyvaiheessa tuotetaan ohjelmistovaatimukset, jotka määrittelevät toteutettavan järjestelmän. Suunnitteluvaiheessa määrittelyn kuvaamien toimintojen toteutus suunnitellaan. Tyypillisesti suunnittelu jaetaan kahteen osaan: arkkitehtuurisuunnittelu ja moduulisuunnittelu. Arkkitehtuurisuunnittelu jakaa järjestelmän osiin, moduuleihin. Moduulisuunnitteluvaiheessa jokaisen moduulin sisäinen rakenne suunnitellaan. Toteutusvaiheessa ohjelma kirjoitetaan eli tuotetaan lähdekoodi, josta toimiva ohjelma

voidaan kääntää. Testauksen tarkoitus on varmistaa ohjelmiston toimivuus. Testausta suoritetaan tyypillisesti usealla tasolla. Iteratiivisessa mallissa iterointi tapahtuu suunnittelu-, toteutus- ja osin testausvaiheen välillä. Ylläpitovaiheessa ratkotaan asiakkaan ongelmia, korjataan virheitä sekä mahdollisesti muutetaan ohjelmaa ja lisätään piirteitä.

3.3.2 Tukitoiminnot

Ohjelmiston laadulla tarkoitetaan yleensä ohjelmistotuotteen kykyä täyttää käyttäjänsä kohtuulliset toiveet ja odotukset [Haikala ja Märijärvi, 1998]. Tämä määritelmä sisältyy myös aiemmin esitettyyn ohjelmistotekniikan määritelmään. Se tuo selvästi esille ajatuksen, että laatu on oleellinen osa ohjelmistotuotetta. Samaan aikaan laatu on kuitenkin hyvin subjektiivinen käsite. Laadunvarmistus on tukitoiminto, jota suoritetaan koko ohjelmiston elinkaaren ajan. Se koostuu seuraavista toimista [Pressman, 1997]:

- Laatujohtaminen
- Tehokkaat menetelmät ja työkalut
- Tekniset katselmoinnit, joita suoritetaan koko prosessin ajan
- Monitasoinen testaus
- Dokumentoinnin ja muutosten hallinta
- Standardienmukaisuuden varmistaminen (milloin tarpeellista)
- Mittaus- ja raportointimekanismit

Konfiguraationhallinta on myös tukitoiminto, jota suoritetaan koko ohjelmiston elinkaaren ajan. Se koostuu muutosten hallintaan tähtäävistä pienemmistä toiminnoista [Pressman, 1997]:

- Muutosten tunnistaminen
- Muutosten hallinta
- Muutoksen vaatimien toteutusten oikeellisuuden varmistaminen
- Muutoksen raportointi kaikille kiinnostuneille osapuolille

On syytä pitää mielessä, että konfiguraationhallinta on eri asia kuin ohjelmiston ylläpito. Se ei myöskään ole sama asia kuin versionhallinta, vaikkakin versionhallinta on osa konfiguraationhallintaa [Dart, 1992]. Konfiguraationhallinnan tulisi olla läsnä kaikkialla ohjelmistokehityksen piirissä koko ohjelmiston elinkaaren ajan.

Dokumentointi on tärkeä tukitoiminto, jonka minimissään tulisi tuottaa projektisuunnitelma, vaatimusmäärittely, tekninen määrittely, testaussuunnitelma sekä

käyttöopas. Ohjelmiston toteuttamisen näkökulmasta dokumentoinnin voidaan katsoa olevan tukitoiminto mutta kuten aikaisemmin ohjelmiston määritelmässä todettiin, dokumentaatio on itse asiassa osa ohjelmistoa. Näin voidaan käsittää, että dokumentaation tekeminen on osa ohjelmiston toteutustyötä. Tämä pohdiskelu ei kuitenkaan varsinaisesti johda mihinkään. Dokumentointi pitää tehdä, dokumentaation tulee olla ajan tasalla ja sen tulee olla hyödynnettävissä myöhemminkin ohjelmiston elinkaaren aikana. Dokumentointi voidaan nykyisin osin automatisoida.

4. Hajautuva ohjelmistokehitys

Tämän luvun tavoitteena on luoda pohja tutkielman aiheen tarkemmalle käsittelylle. Luvun ensimmäisessä kohdassa kootaan yhteen syitä, joiden vuoksi ohjelmistokehityksessä mennään kohti hajautettuja projekteja. Kohdassa 4.2 tarkastellaan lyhyesti erästä hajautetun ohjelmistokehityksen ääriesimerkkiä, avoimen lähdekoodin ohjelmistoprojekteja.

4.1 Miksi ohjelmistokehitysprojektit hajautuvat?

Ohjelmistokehitys on suuntautumassa hajautettuihin projekteihin ja organisaatioihin. Ohjelmistotuotteiden jatkuvasti suureneva koko aiheuttaa mm. sen, että niitä tekevien ihmisten lukumäärä kasvaa. Kokemuksesta tiedetään, että kun ohjelmistoprojektin henkilömäärä nousee, ryhmän tuottavuus saattaa kärsiä [Pressman, 1997]. Eräs tapa ratkaista tämä ongelma on luoda useita kehittäjäryhmiä. Nykyisin nämä ryhmät ovat usein vielä sekä maantieteellisesti että ajallisesti erillisiä ts. hajautettuja. Ryhmissä tehdään rinnakkaista kehitystä.

4.1.1 Työvoima ja osaaminen

Työvoima ja osaaminen on maantieteellisesti hajautunutta. Se tietää monille organisaatioille joko sisäistä hajautumista, ulkoistamista tai verkostoitumista. Useimpien ohjelmistoyritysten päätoimipaikat sijaitsevat kehittyneissä maissa, joissa työvoima on suhteellisen kallista ja sitä voi olla joskus hankala löytää. On taloudellisesti kannattavampaa teettää osa työstä muualla kuin päätoimipaikassa, esim. halvemman työvoiman maissa. Paitsi, että monissa kehittyvissä maissa on halvempaa työvoimaa, myös työvoiman saatavuus saattaa olla huomattavasti parempi. Tämä voidaan käyttää hyödyksi palkkaamalla työntekijöitä näistä maista, perustamalla tytäryhtiöitä sinne tai ulkoistamalla osa kehitystyöstä. Ulkoistaminen on eräs tapa lisätä joustavuutta ja lyhentää kehitykseen käytettyä aikaa. Aika onkin yksi tärkeimmistä kilpailutekijöistä. On äärimmäisen tärkeätä, että tuotteet saadaan nopeasti ja aikataulun mukaisesti markkinoille. Ajankohtaiset ja paljon julkisuudessa esiintyvät trendit ulkoistaminen, alihankinta ja verkostoituminen mahdollistavat osapuolten keskittymisen ydinosaamiseensa [VeTO, 2001]. Trendikkäiden käsitteiden takaa löytyy kuitenkin tärkeää asiaa. Tavoitteena on tehokkuus ja kaikkien osapuolten hyöty, siis taloudellisen voiton saavuttaminen

4.1.2 Kustannussäästöt?

On esitetty, että vähentynyt matkustaminen ja lisääntynyt rinnakkaisuus kehityksessä aikaansaisivat kustannussäästöjä [Alho ja Sulonen, 1998]. Tämä pitää varmasti jossakin määrin paikkansa, erityisesti rinnakkaisuuden osalta. Toimivaa rinnakkaista kehitystyötä ei kuitenkaan saavuteta kovinkaan helposti. Se, vähentääkö hajautuminen matkustamista ja siten aikaansaa kustannussäästöjä, on kyseenalaista. En pidä sitä myöskään kovin oleellisena tekijänä. Usein ohjelmiston markkinoille tai tuotantoon saaminen on huomattavasti tärkeämpää kuin satunnaiset kustannussäästöt. Hyvin toteutettu hajautus lisää rinnakkaisuutta ja ennen kaikkea siten nopeuttaa tuotteen valmistumista. Toisaalta nopea kehitys tuo mukanaan myös kustannussäästöjä, jolloin saavutettu etu on kaksinkertainen.

4.1.3 Asiakasrajapinta

Ohjelmistoprojektien asiakkaat toimivat nykyään usein globaalisti tai sijaitsevat ulkomailla. On yhteistyön molemmille osapuolille edullista, että osapuolet sijaitsevat maantieteellisesti lähellä toisiaan. Yhteistyö, jota ohjelmistokehitys väistämättä vaatii, helpottuu. Täten ohjelmistoja tuottavan yrityksen on myös helpompi pitää asiakkaat tyytyväisinä. Yritys voi saavuttaa tärkeän kilpailuedun toimimalla tehokkaasti hajautettuna organisaationa. Kosketuspinta jo olemassa oleviin sekä potentiaalisiin asiakkaisiin on suurempi. Pape [Pape, 1996] näkee, että yhtiön resurssien hajauttamisesta lähelle asiakasta tai osaamiskeskuksiin saatava hyöty on suurempi kuin sen aiheuttama tehokkuuden aleneminen yhtiön sisällä. Edelleen, hajautus ja läheiset asiakaskontaktit mahdollistavat nopeamman kasvun kuin perinteinen keskitetysti organisoitu malli.

4.1.4 Hajautumisen mahdollistaneet teknologiat

Osaltaan hajautumiseen ovat vaikuttaneet myös sen mahdollistaneet teknologiat, joista tärkeimpänä voidaan pitää Internetin nousua tärkeäksi sekä tarpeeksi nopeaksi ja tehokkaaksi työvälineeksi. Ohjelmistoja, jotka mahdollistavat yhteistoiminnallisen työn tietoverkoissa, kehitetään lisääntyvissä määrin [Alho *et al.*, 1999].

4.2 Avoimen lähdekoodin projektit

Eräs ajankohtainen hajautetun ohjelmistokehityksen alue on avoimen lähdekoodin ohjelmistokehitys. Nämä projektit edustavat yhtä hajautuksen ääripäätä, käytännössä kaikki kehittäjät ovat maantieteellisesti ja ajallisesti erillään. Cubranic hahmottaa aihetta artikkelissaan [Cubranic, 1999]. Avoimen lähdekoodin ohjelmistokehitys on jo

jonkin aikaa olemassaollut mutta vasta nyt julkisuutta saavuttanut ohjelmistokehityksen muoto. Sitä on usein pidetty liian kontrolloimattomana vaativille ohjelmistoprojekteille. Kuitenkin ohjelmistotuotteet kuten Apache, Perl ja muutamat muut ovat saavuttaneet vankan jalansijan ja siten tuoneet niiden kehityksessä käytetyn lähestymistavan vaihtoehdoksi perinteiselle ohjelmistokehitykselle.

Jos avoimen lähdekoodin projektissa käytetään versionhallintaa tai oikeammin konfiguraationhallintaa, työkalu useimmiten on CVS, Concurrent Versions System [Hoek, 2000]. Se onkin erittäin sopiva juuri tähän tarkoitukseen, se tarjoaa hyvin toimivat perusversionhallintaominaisuudet, tukee hajautusta, on ilmainen ja ylläpidetty. Kuitenkaan sen toiminnallisuutta ei ole juuri laajennettu viime aikoina. Siinä on joitakin puutteita soveltuakseen laajempien projektien työkaluksi.

Avoimen lähdekoodin projektit nojautuvat lähes poikkeuksetta viestinnässään postituslistoihin ja/tai uutisryhmiin. Myös projektin koordinointi yleensä hoidetaan samoin välinein. Tälle käytännölle on olemassa hyvät perusteet. Sähköposti on käytännössä pienin yhteinen nimittäjä internetissä tapahtuvalle viestinnälle. Näin kenen tahansa on mahdollisimman helppo ottaa osaa kehitykseen tai seurata sitä. Korkea hajautuksen aste puolestaan käytännössä estää synkronisten viestinten käytön. Lisäksi kehittäjien työmäärät vaihtelevat suuresti riippuen heidän muista sitoumuksistaan. Nämä ovat samalla osittain niitä piirteitä, jotka erottavat avoimen lähdekoodin projektit ohjelmistoyritysten ohjelmistoprojekteista. Yhdenmukaiset työkalut ja käytännöt sekä sitoutuminen säännölliseen työhön poistavat joitakin esteitä hajautetun ohjelmistokehityksen tieltä.

5. Aikaisemmat tutkimustulokset

Luvun tavoite on kartoittaa aiemmissa tutkimuksissa saavutettuja tuloksia ja siten hahmottaa tutkielman aiheen ja aihepiirin ympäristöä. Aiempia tuloksia tarkastellaan saman jäsennyksen mukaisesti kuin havaittuja ongelmia. Lisäksi tällä pyritään hahmottamaan mielestäni tärkeimpiä aihealueita selkeiksi kokonaisuuksiksi. Alussa hahmotetaan aiemmin luotuja malleja ja hieman niistä johdettuja toteutussuunnitelmia. Mielestäni mielenkiintoisimpia ja ehkä tämän tutkielman kannalta tärkeimpiä tuloksia esitellään sen jälkeen. Kohdissa 5.1 – 5.3 tarkastellaan lähemmin joitakin alakohtia, jotka olen katsonut tutkielman kannalta hedelmällisimmiksi. Lopuksi annan hieman kritiikkiä joitakin tutkimustuloksia kohtaan.

5.1 Yleiskuva aiheeseen

Useat aiheeseen liittyvät tutkimukset ovat keskittyneet mallintamaan hajautettua ohjelmistoprojektia ja tältä pohjalta asettaneet suuntaviivat hajautettua ohjelmistokehitystä tukevalle kehitysympäristölle tai työkalulle [Alho *et al*., 1999; Alho ja Sulonen, 1998; Cugola ja Ghezzi, 1999; Gao *et al*., 2000; Grundy, 2000; Grundy *et al*., 1998]. Myös prosessin mallinnusta ja sen tukemista työkaluilla on tutkittu. Usein nämä työkalut lisäksi tukevat samanaikaista muokkausta tai muuten muistuttavat ryhmäohjelmia. Myös projektin hallintaa tukevia työkaluja on kehitetty. Yleensä näihin on tuotettu jonkinlaista ”älykkyyttä” esimerkiksi agenttien muodossa [O’Connor ja Jenkins, 1999]. Useimmissa tapauksissa perusajatus on tuottaa käsitteellinen malli hajautetusta työstä, jonka pohjalta pyritään ymmärtämään hajautettua prosessia ja sen vaatimaa infrastruktuuria. Tyypillisesti tätä seuraa hahmotelma toteutuksesta jollakin tasolla. Lisäksi näkökulmaan liitetään toisinaan jokin muu tutkimuskohde kuten esimerkiksi tietämyksenhallinta [Maurer ja Dellen, 1998] tai projektin johtaminen ja koordinointi [Bendeck *et al*., 1998].

Myös avoimen lähdekoodin ohjelmistokehitystä tutkitaan jonkin verran. Nämä tutkimukset ovat pääasiassa aihetta hahmottelevia. Tämä on ymmärrettävää tutkimusaiheen epämääräisyys huomioonottaen. Alueelta kuitenkin löytyy hyviä huomioita esimerkiksi projektin koordinointiin liittyen [Cubranic, 1999]. Myös konfiguraationhallintaa, joka on käytännössä CVS, ja sen käyttöä on tutkittu jonkun verran [Hoek, 2000].

Joitakin mielenkiintoisia aiheita, joissa mielestäni lähestytään hajautetun ohjelmistotyön oleellisimpia ongelmia, ovat mm. hajautetun projektin organisoinnin

näkökulma [Altmann ja Dopler, 1998], ongelmien ja ristiriitatilanteiden hallinta [Gao *et al*., 2000; Kudo *et al*., 1998], ihmisten osuus ohjelmistotyössä [Cockburn, 1999; Pape, 1996] sekä jotkin konfiguraation- ja dokumenttien hallintaa [Hunt ja Reuter, 2001] käsittelevät tutkimukset.

Maurer ja Dellen [Maurer ja Dellen, 1998] kuvaavat ohjelmistokehitystä tietämysintensiiviseksi prosessiksi, jossa korkeasti koulutetut ihmiset toimivat yhdessä saavuttaakseen liiketoiminnalliset tavoitteet. Edelleen he pitävät pohjimmaisina ongelmina seuraavia:

- Ohjelmistokehitys on kallista ja aikaa vievää.
- Ohjelmistoprojektit usein ylittävät aikarajat ja budjetit.
- Aika- ja kustannusarvioiden tekeminen on virhealtista.

Heidän mukaansa siirtyminen suurempiin kehitysryhmiin, jotka ovat maantieteellisesti hajautettuja, johtaa lisäksi seuraaviin ongelmiin:

- Kuinka koordinoida ja hallita kehittäjien hajautettua työtä?
- Kuinka tuoda projektin tietämys kaikkien ulottuville?
- Kuinka hallita edellisissä projekteissa kerääntynyt tietämys ja saada se nykyisten käytettäväksi?

He tarkastelevat asiaa tietämyksenhallinnan näkökulmasta, mutta siitä huolimatta heidän hajautetusta ohjelmistokehityksestä esille tuomansa ongelmat ovat selkeästi toteutuneet mm. tutkielman esimerkkiprojektissa.

5.2 Viestintä

Viestintä on kaiken ihmisten välisen yhteistyön perusta. Se on prosessi, jossa informaatiota siirretään ja vaihdetaan [Altmann ja Dopler, 1998].

Kun projektien koko kasvaa, luodaan useita erillisiä kehittäjäryhmiä, jotka lisäksi ovat usein sekä maantieteellisesti että ajallisesti erillisiä, siis hajautettuja. Tästä seuraa ongelma ryhmien välisessä viestinnässä: siitä tulee tehotonta tai sitä ei ole tarpeeksi [Pressman, 1997]. Tämä on eräs keskeisimpiä lähtökohtia tälle tutkielmalle. Useissa lähteissä viestintä on todettu joko suoranaisesti tai välillisesti ongelmalliseksi [Altmann ja Dopler, 1998; Pape, 1996]. Viestintä on projektissa sekä voimavara että työväline. Se on muihin toimintoihin nähden sikäli erikoisasemassa, että se on välttämätöntä [Ruuska, 2001].

5.3 Konfiguraationhallinta

On todettu, että konfiguraationhallinta on 10%:sesti tekninen ongelma, loppu muodostuu johtamisesta, prosessista sekä käyttäjien koulutuksesta [Dart, 1992]. Jos konfiguraationhallinta on liian hankala käytettäväksi tai jos sen koetaan häiritsevän työtä, sitä ei käytetä.

Kuten kohdassa 4.2 todettiin, CVS tukee hajautettua ohjelmistotyötä ja on käyttökelpoinen konfiguraationhallintajärjestelmä avoimen lähdekoodin projekteille. CVS toimii tässä tutkielmassa eräänlaisena referenssinä, johon muita konfiguraationhallintajärjestelmiä verrataan. Se sopii hyvin myös pienempiin kaupallisiin ohjelmistoprojekteihin. Siinä on kuitenkin selviä puutteita ollakseen vakava haastaja kaupallisille kilpailijoilleen [Hoek, 2000]. Niistä akuuteimpia ovat

- Ei tukea useille tietolähteille
- Ei tukea hakemistojen versioinnille
- Ei tukea yksityiselle versioinnille
- Ei tietolähteiden replikointia

Useimmista kaupallisista vaihtoehdoista nämä ominaisuudet löytyvät.

Hoek [Hoek, 2000] esittää joitakin ajatuksia ja ominaisuuksia järjestelmästä, joka on täysin hajautettu ja joka hallitsee siinä olevat ohjelmistoartefaktit niiden koko elinkaaren ajan. Elinkaari ulottuisi ensimmäisestä versiosta niiden lopulliseen sijoitukseen asennettuna ohjelmiston osana asiakkaan luona. Siinä olisi CVS:iin nähden kaikki siihen aikaisemmin esitetyt parannukset sekä

- Julkistuksen hallinta
- Virheiden seuranta
- Tuki ohjelmiston käyttöönotolle

Grundy [Grundy, 2000] on tutkinut komponenttipohjaisten ohjelmistojen tuottamista hajautetussa ohjelmistoprojektissa. Hänen tunnistamansa ongelmat ovat hyvin samankaltaisia kuin muissakin aihetta käsittelevissä tutkimuksissa löydetyt. Hän näkee tärkeänä ja komponenttipohjaiselle ohjelmistokehitykselle ominaisena osa-alueena ohjelmistokomponenttien jakelun. Tämä näkökulma on usein esillä myös muussa kuin komponenttipohjaiseen ohjelmistotuotantoon liittyvissä tutkimuksissa. Toimiva ratkaisu ohjelmistojen hallintaan aina kehityksestä asiakkaalle toimitukseen asti poistaisi monia hajautettuun ohjelmistokehitykseen liittyviä ongelmia. Tämä ei kuitenkaan ole triviaali ongelma ja joudutaan jättämään tämän tutkielman ulkopuolelle.

5.4 Projektin hallinta

Bendeck ja kumppanit, samoin kuin Maurer ja Dellen edellä, näkevät hajautettujen projektien suurimpina ongelmina koordinoinnin ja projektin tilasta tiedottamisen [Bendeck *et al*., 1998]. Molemmat ovat projektin johdon aktiviteetteja, joihin liittyy viestintää. On tärkeätä, että kehittäjille hajautuneisuus on mahdollisimman läpinäkyvää, siis heillä on aina tarvitsemansa informaatio saatavilla.

Tämän lisäksi Altmann ja Dopler [Altmann ja Dopler, 1998] määrittelevät hajautetulle ohjelmistokehitykselle suuntaa antavia vaatimuksia, joista mielestäni keskeisimmät ajatukset ovat:

- Ryhmän jäsenet tarvitsevat sekä oman yksityisen näkymänsä että yleiskuvan tarjoavan näkymän projektin tilaan.
- Kaikilla projektin jäsenillä pitää olla selvä kuva siitä, mitä projektissa on tehty, mitä tehdään ja mitä tullaan tekemään.
- Ristiriitatilanteet tulee voida ratkaista neuvottelemalla.
- On oltava olemassa ennalta määritelty prosessi.
- On oltava olemassa ennalta määritellyt dokumenttipohjat.
- Viestinnän tulee olla mahdollista ajasta ja paikasta riippumatta.
- Dokumenttien hallinnan tulee toimia.
- On oltava olemassa toimiva konfiguraationhallinta.

Grundy et al. [Grundy *et al*., 1998] tarkastelevat aihetta projektin koordinoinnin näkökulmasta ja tunnistavat joitakin ongelmia hajautetun ohjelmistoprojektin johtamisessa:

- Projektin jäsenillä pitää olla tarkasti määritellyt tehtävät, joita pitää koordinoida hyvien lopputulosten saavuttamiseksi.
- Projektin jäsenten pitää ajoittain viestiä ja tehdä yhteistyötä ja toisinaan työskennellä itsenäisesti oman osa-alueensa parissa.
- Ohjelmistoartefaktien tulee olla jaettuja ja ajan tasalla.
- Tarvitaan useita työkaluja artefaktien editointiin. Näistä joidenkin pitää tukea samanaikaista editointia, toisten löyhempää synkronointia.
- Kehitystä kohti määritettyjä tavoitteita pitää voida seurata, projektin jäsenten tulee olla tietoisia toistensa työstä ja monimutkaisten ohjelmistojen tulee olla hajautetusti kehitetyistä osista konfiguroituja.
- Projektin jäsenten täytyy joustavasti konfiguroida ympäristönsä tuki artefaktien hallinnalle, viestinnälle sekä työn koordinoinnille.

En kuitenkaan pidä neljättä kohtaa kovinkaan suurena ongelmana siinä mielessä, että ohjelmistokehityksessä tarvittaisiin tukea samanaikaiselle muokkaukselle. Työkaluja kaikkien artefaktien muokkaukseen ja käsittelyyn tietenkin tarvitaan, mutta niiden samanaikainen muokkaus yhdessä toisen projektin jäsenen kanssa tuskin on oleellisimpia hajautetun ohjelmistokehityksen ongelmia.

5.4.1 Ongelmien hallinta

Ohjelmistoprojektien ongelmienhallinnassa on tärkeätä informaation jakamisen ohella myös tehtävien ja vastuun selvä koordinointi [Kudo *et al*., 1998]. Kudo ja kumppanit pitävät ongelmanratkaisun tärkeimpänä seikkana ongelman olemassaolosta sekä sisällöstä tiedottamisen oikealle ryhmälle mahdollisimman aikaisin. Tähän päästäkseen he ehdottavat ongelmienhallintajärjestelmää, joka

- on www-pohjainen, siis käytetään www-selaimesta
- tarjoaa reaaliaikaisen näkymän ongelmiin
- tarjoaa mahdollisuuden osoittaa ongelman ratkaisu ryhmälle keskustelupalstan avulla. Ongelma voidaan osoittaa ryhmälle myös suoraan
- käyttää taulukkolaskentaa apuohjelmana.

Heidän ehdottamansa järjestelmä toimii pääpiirteissään seuraavasti: Ryhmä huomaa ohjelmistovirheen tai tunnistaa jonkin ongelman ja antaa siitä ilmoituksen järjestelmän avulla projektipäällikölle tai jollekin muulle, joka vastaa ongelmienhallinnasta. Tämä joko osoittaa ongelman suoraan jonkun ryhmän ratkaistavaksi tai julkaisee sen keskustelupalstalla, josta joku ryhmä tunnistaa sen omakseen ja ottaa hoitaakseen. Jos kuitenkin käy niin, että ongelma ei kuulunutkaan tämän ryhmän alueeseen tai se vaatii lisäselvittelyä, joka ei kuulu tämän ryhmän alueeseen, ryhmä voi puolestaan joko suoraan osoittaa sen toiselle ryhmälle tai julkaista keskustelupalstalla, josta toinen ryhmä ottaa sen hoitaakseen. Ratkaistusta ongelmasta ilmoitetaan järjestelmällä. Tiedot talletetaan tietokantaan, josta projektipäällikkö voi tuottaa raportteja työn seuraamiseksi.

Samansuuntaisia tuloksia ongelmienhallinnan parantamiseksi ovat saaneet aikaan Gao et al. [Gao *et al.*, 2000]. He havaitsivat hajautetussa ohjelmistotyössä Fujitsulla ja sen tytäryhtiöissä seuraavia ongelmia:

- Ongelmien seuranta on tehotonta.
- Ongelmia koskevan tiedon jakamien on rajoittunutta koska ei ole keskitettyä tietovarastoa.

- Ongelmienhallinta on ad-hoc –tyyppistä.
- Kehitystyökalut ja –ympäristöt ovat heterogeenisiä.

Näiden ongelmien ratkaisemiseksi he ehdottavat seuraavia toimia, jotka Kudo edellä tämän kohdan alussa oli pukenut konkreettisempaan asuun:

- Määritellään käsitteellinen malli ongelmien analysoinnin ja hallinnan tukemiseksi. Mallin avulla voidaan muodostaa yhdenmukaiset käytännöt.
- Kehitetään verkkokeskeinen ongelmatietovarasto, jolloin informaation jakaminen helpottuu.
- Tarjotaan systemaattinen tapa, jolla ongelmanratkaisun tilaa voidaan seurata.
- Tarjotaan tehokas tapa, jolla ongelmat ja niiden tila synkronoidaan.

5.4.2 Työnjako ja roolit

Bendeck ja kumppanit [Bendeck *et al*., 1998] listaavat erityisesti projektin hallintaan liittyviä rooleja.

- *Projektinsuunnittelija* luo projektisuunnitelman. Apuna tässä olisi hyvä olla tietoja ja kokemuksia entisistä projekteista. Projektin suunnittelijan pitää myös voida muuttaa suunnitelmaa projektin edetessä. Hänelle ilmoitetaan kun muutokset ovat tarpeellisia. Hän informoi tarpeellisia tahoja muutoksista.
- *Mittauksen suunnittelija* suunnittelee prosessin, ohjelmistotuotteen ja resurssien mittaamisen sekä määrittelee tavoitearvot. Hän tekee tämän yhteistyössä Laadunvarmistajan (alla) kanssa, joka suorittaa ja valvoo suunnitelman toteutusta projektin aikana. Mittauksen suunnittelijan tulee voida tehdä mittauksiin liittyviä lisäyksiä projektisuunnitelmaan. Projektin edetessä sen tavoitteet voivat muuttua ja voi ilmetä tarvetta muuttaa mittaussuunnitelmaa, joka puolestaan saattaa aiheuttaa muutoksia projektisuunnitelmaan.
- *Projektipäällikkö* varmistaa projektin oikean ja oikea-aikaisen suorituksen. Hän jakaa resurssit ja sopii tehtävät projektin henkilöstölle, määrittelee projektin aikarajat projektisuunnitelman mukaan sekä valvoo toteutusta. Mahdolliset muutokset projektisuunnitelmaan projektipäällikkö suorittaa yhteistyössä Projektinsuunnittelijan kanssa.

- *Laadunvarmistaja* on vastuussa tuotteen ja prosessin laadusta. Laatuvaatimukset määritellään tyypillisesti yrityksen laatujärjestelmässä sekä sopimuksessa ohjelmistotuotteen tuottamiseksi. Hän osallistuu mittauksen suunnitteluun sekä suorittaa ja valvoo sen toteutusta. Jos tarvetta muutoksille esiintyy, Laadunvarmistaja ilmoittaa siitä Projektipäällikölle.

Näiden lisäksi projektissa on prosessiin liittyviä teknisiä rooleja.

5.5 Kritiikkiä

Hajautuksen tukemisessa ollaan eräissä tutkimuksissa menty varsin pitkälle, joissakin tapauksissa mielestäni liiankin pitkälle. Useimmiten tutkimuksen kohteena ovat olleet samanaikaisen työskentelyn saman artefaktin parissa mahdollistavat ryhmäohjelmat. Esimerkiksi ohjelmistojen katselmointi Internetin välityksellä [Harjumaa ja Tervonen, 2000] vaikuttaa ainakin tämän tutkielman lähtökohdista tarkasteltuna hedelmättömältä. Katselmoinnin järjestämiseen tapaamisena ja kaikkiin siihen liittyviin valmisteluihin kuluu eittämättä aikaa ja rahaa mutta hyvin suoritettuna katselmointi on hyvin tehokas väline ohjelmistojen laadun ylläpitämiseksi ja parantamiseksi.

6. Havaintoja todellisesta projektista

Luvun tavoite on esitellä eräs projekti, jota käytetään esimerkkinä sekä mallina, johon saavutetut tulokset suhteutetaan. Kohdassa 6.1 annetaan yleiskuva projektista ja kohdassa 6.2 kuvaillaan havaittuja ongelmia. Lopuksi kohdassa 6.3 tehdään pieni yhteenveto projektista.

6.1 Yleiskuva projektista

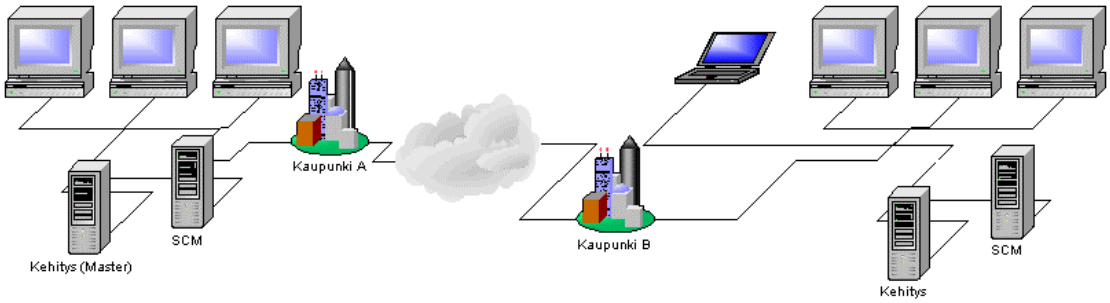
Lähtökohta tälle tutkielmalle olivat ongelmat ja puutteet, joita havaittiin eräässä ohjelmistoprojektissa, jossa työskentelin. Tämä projekti oli hajautetusti toteutettu asiakasprojekti. Sen tavoitteena oli kehittää keskitetty komponenttipohjainen palvelinohjelmisto asiakkaan liiketoiminnan tueksi.

6.1.1 Tuotettava ohjelmisto

Ohjelmisto käyttää suurta tietokantaa ja ohjelmalogiikka toteutettiin Microsoftin komponenttiteknologiaa käyttäen. Ohjelmistoa käytetään selaimen avulla. Käyttöliittymä on toteutettu ASP-sivuilla. Tietokanta ja palvelimet sijaitsevat keskitetysti yhdessä toimipisteessä. Ohjelmisto on asiakkaan liiketoiminnan kannalta keskeinen ja käyttäjiä on lukuisissa toimipisteissä ympäri Suomen.

6.1.2 Projektin infrastruktuuri

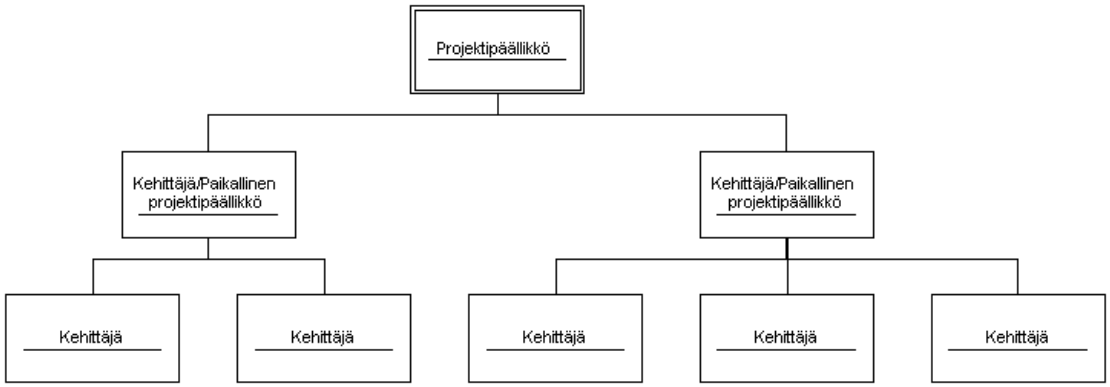
Projektin varsinainen kehitystyö tapahtui rinnakkaisesti kahdella eri paikkakunnalla. Kaupungissa A oli kolme henkilöä, oma konfiguraationhallinta sekä projektin pääkehityspalvelin. Kaupungissa B työskenteli neljä henkilöä. Heillä oli samoin kehityspalvelin sekä oma konfiguraationhallintansa (kuva 6-1). Palvelinjärjestely on toteutettu näin puhtaasti teknisistä syistä. Ohjelmistokehitys on ennen tapahtunut ainoastaan yhdellä paikkakunnalla ja entiset käytännöt siirrettiin sellaisenaan hajautettuun projektiin. Toimipisteet on yhdistetty VPN –ratkaisulla mutta yhteys on hyvin hidas. Aikaeroa toimipisteiden välillä ei ole kuin tunti.



Kuva 6-1 Projektin infrastruktuuri

6.1.3 Projektiorganisaatio

Projektiorganisaatio on kuvattu kuvassa 6-2. Työntekijöitä projektissa oli pääasiallisesti seitsemän. Projektipäällikkö työskenteli kaupungissa B ja oli samalla paikallinen projektipäällikkö siellä. Hänen lisäkseen kaupungissa B työskenteli kolme henkilöä. Kaupungissa A työskenteli kolme kehittäjää, joista yksi oli paikallinen projektipäällikkö. Asiakas toimi kaupungissa A ja asioi pääsääntöisesti kaupungin A paikallisen projektipäällikön kanssa.



Kuva 6-1 Projektin organisaatio

6.2 Havaitut ongelmat

Projektin hajautus on huomioitu projektisuunnitelmassa, jossa se on nimetty yhdeksi mahdollisista projektin kohtaamista riskeistä. Samassa yhteydessä tuotiin erityisesti esiin viestinnän tärkeys kahden ryhmän välillä. Projektin kuluessa on ilmennyt lisäksi seuraavia ongelmia, jotka eivät kuitenkaan ole välttämättä sidoksissa projektin hajautettuun olemukseen: erilaiset kehitysympäristöt, mittaus, työnjako sekä konfiguraation- ja dokumenttien hallinta.

6.2.1 Ryhmien välinen viestintä

Hankalaksi havaittiin erityisesti ryhmien välinen viestintä. Sitä pyrittiin parantamaan mm. ottamalla käyttöön uutisryhmä-tyylinen keskustelupalsta, jossa voitiin viestiä projektin asioita. Molemmat ryhmät olivat melko tiiviitä ja hyvin toimivia. Niissä viestintä toimi hienosti sisäisesti mutta ryhmien välinen viestintä hieman kangerteli. Yksi syy oli kaksikielisyys, suurin osa viestinnästä voitiin hoitaa suomeksi mutta välillä jouduttiin puhumaan englantia. Keskustelufoorumi helpotti tilannetta hieman, mutta ei kuitenkaan poistanut ongelmaa.

6.2.2 Erilaiset kehitysympäristöt

Kaupungissa A käytettiin kehitysympäristön alustana eri käyttöjärjestelmää kuin kaupungissa B. Tästä seurasi muun muassa komponenttien yhteensopivuusongelmia ja siten ylimääräistä työtä. Tällaisessa tilanteessa olisi mielestäni peruslähtökohta oltava se, että kaikilla on samanlainen kehitysympäristö. Mitään teknistä estettä sille ei ollut. Myös tietokantaohjelmisto vaihdettiin uudempaan versioon kesken projektin. Yllättäen tästä ei kuitenkaan koitunut juuri minkäänlaisia ongelmia.

6.2.3 Mittaus ja työnjako

Ohjelmistotuotteen mittaukseen liittyvät ongelmat tarkoittavat, ettei olla pystytty kovinkaan tarkasti seuraamaan projektin valmistumisastetta. Samoin projektin laajuuden tai sen koon arviointi on ollut hankalaa ja siten myös täsmällisen työnjaon suorittaminen oli alkuvaiheessa ongelma. Lähinnä asiakasta sijaitseva ryhmä (kaupunki A) joutui toimimaan kehitystyön ohella asiakasrajapinnassa, jolloin kehitystyön tehokkuus luonnollisesti kärsi. Lisäksi projektin edetessä muodostui ikään kuin kaksi asiakasrajapintaa, yksi molempiin kaupunkeihin. Tästä syystä projektin koordinointi vaati ylimääräisiä ponnistuksia erityisesti projektin johdolta.

6.2.4 Dokumentit ja konfiguraationhallinta

Kaksi erillistä versionhallintakantaa teki toimivan konfiguraationhallinnan käytössä olleella työkalulla käytännössä mahdottomaksi. Niinpä jouduttiin tyytymään lähinnä versionhallintaan. Tämä ei tietenkään ole toivottavaa. Lisäksi dokumenttien hallinta tuotti ylimääräistä työtä ja epäselvyyttä oikeista versioista koska niiden välityksessä jouduttiin usein turvautumaan sähköpostiin.

6.3 Yhteenveto projektista

Tästä projektista saadaan kattava kuva ongelmista, jotka aiheutuvat hajautuksesta. Kuva saattaa vaikuttaa liiankin ongelmaiselta ja projekti tuhoon tuomitulta, mutta se johtuu vain siitä, että tässä keskitytään kartoittamaan ongelmia eikä tuoda esiin projektin monia hyviä puolia. Näiden havaintojen pohjalta tätä tutkielmaa on lähdetty tekemään. On huomioitava, että kaikki projektin ongelmat eivät suinkaan johdu sen hajautetusta luonteesta vaan yleisesti ohjelmistotyön haastavuudesta. Tarve näiden ja muiden hajautetusta kehityksestä johtuvien ongelmien ratkaisemiseksi on kuitenkin ilmeinen.

7. Ratkaisumalleja

Luvun tavoite on hahmotella ratkaisut tutkimusaiheen ongelmiin. Alussa, kohdassa 7.1, tiivistetään tutkielman keskeisimmät havainnot. Kohdassa 7.2 ja sen alakohdissa selvitetään miten hajautetun ohjelmistoprojektin viestintää voidaan parantaa. Seuraavassa kohdassa 7.3 käsitellään dokumenttien hallintaa, joka sisältää sekä dokumentit (7.3.1) että konfiguraationhallinnan (7.3.2). Selvitetään miten ne tulisi hoitaa niin, että hajautuksesta siltä kannalta saataisiin mahdollisimman läpinäkyvää. Viimeisessä kohdassa 7.4 pohditaan miten projektia tulisi koordinoida ja johtaa.

7.1 Keskeisimmät havainnot

Tutkielman keskeinen havainto on, että siirryttäessä hajautettuun työhön on tärkeintä, lähtötilasta riippuen, säilyttää tai kehittää toimiva viestintä. Jos viestintä ei toimi, mikään muukaan osa-alue ei voi toimia. Muut hajautuksesta aiheutuvat ongelmat liittyvät pääasiassa ohjelmistokehityksen tukitoimintoihin. Kun ne toimivat oikein, vältytään turhalta työltä ja voidaan keskittyä olennaiseen. Prosessin ei tarvitse muuttua siirryttäessä hajautettuun ohjelmistokehitykseen. Toimiva ohjelmistokehitys on pohjimmiltaan samanlaista kaikenlaisissa ryhmissä. Hajautettu ohjelmistokehitys siis lisää vaatimuksia viestinnän ja tukitoimintojen osalle. Näitä tukemaan tarvitaan toimivat ja tehokkaat työkalut sekä hyvät toimintatavat.

Tutkielman tulokset voidaan tiivistää kolmeen ryhmään: viestintään, dokumenttien hallintaan ja projektin hallintaan. Näiden yläkäsitteiden alta voidaan tunnistaa tarkempia alueita, mm. projektin hallintaan sekä projektin toimintoja tukeviin työkaluihin liittyviä. Tässä yhteydessä dokumentin käsitettä on siis lavennettu kattamaan myös lähdekoodit kuten kohdassa 1.1 esitettiin. Tämän jaottelun tarkoitus on tehdä tutkielman tulosten jäsentäminen selkeämmäksi. Se, ovatko lähdekoodit dokumentteja vai itse ohjelma, ei ole todellisuuden kannalta merkityksellistä.

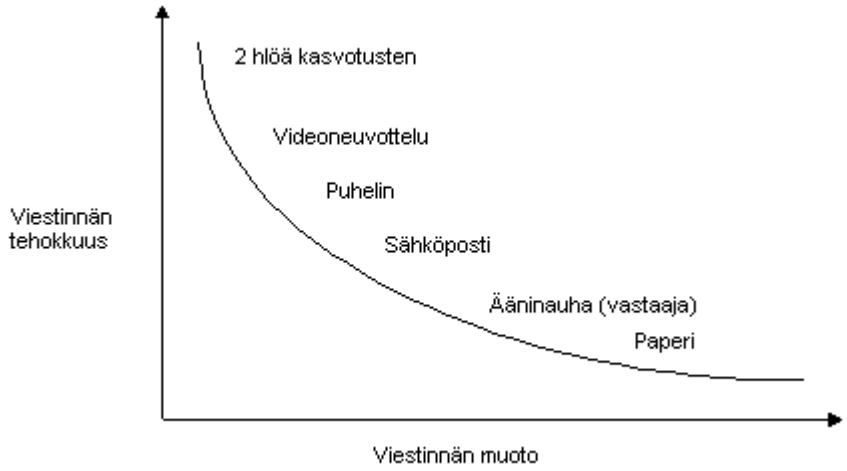
7.2 Viestintä

Viestintävälineet jaetaan kahteen ryhmään sen mukaan, minkälaista viestintää ne tukevat. Kuten kuvasta 7-1 nähdään, ryhmät ovat asynkroninen ja synkroninen. Asynkronisia ovat muun muassa sähköposti, uutisryhmät ja tekstiviestit. Synkronisia puolestaan ovat muun muassa puhelin, videoneuvottelulaitteet sekä erilaiset samanaikaisen yhteistoiminnan mahdollistavat ryhmäohjelmistot. Monet uudet synkroniset viestintävälineet lupaavat parantaa informaation vaihdon tehokkuutta ja laatua. Useimmiten nykyään jo yleisesti käytetyt synkroniset välineet, kuten

matkapuhelin ja videoneuvottelu- tai videopuhelulaitteistot, tarjoavat oikein käytettynä riittävät työkalut suoraan viestintään. Monet perinteisinä pidetyt asynkroniset välineet, kuten sähköposti ja uutisryhmät, tarjoavat toimivat ja tehokkaat työkalut erityisesti ajallisesti hajautuneelle projektiryhmälle.

Kuva 7-1 Viestintävälineet

Hajautettuun ohjelmistokehitykseen pätevät samat vuorovaikutus- ja johtamistekijät kuin muuhunkin ohjelmistokehitykseen ja projektityöhön. Kuitenkin asiat kuten viestintä ja ristiriitojen hallinta vaativat enemmän huomiota hajautetussa ryhmässä, koska on vähemmän epämuodollista viestintää. Lisäksi monikulttuurisissa ryhmissä kulttuurien väliset tekijät tulee ottaa huomioon [Alho *et al.*, 1999]. Kuvassa 7-2 kuvataan suuntaa antavasti viestinnän tiloja ja niiden tehokkuutta. Kuten kohdassa 1.1 todettiin, ihmiset ovat parhaimmillaan kasvotusten kommunikoidessaan. Kun viestintä köyhtyy, viestinnän tehokkuus laskee. Näin ollen ryhmän viestintä tulisi sijaita mahdollisimman ylhäällä kuvatulla käyrällä, kuitenkin viestintätilanteen vaatimukset huomioiden ja siihen sopeutuen. Tärkeimmät päätökset on syytä tehdä kasvotusten koska ne vaativat tiivistä viestintää ja yhteistoimintaa. Muutenkin hajautetuissa projekteissa on syytä käyttää hiukan aikaa ja rahaa säännöllisiin tapaamisiin. Tämä lisää viestinnän tehokkuuden ohella myös projektin jäsenten luottamusta ja tunnetta siitä, että heidät pidetään ajan tasalla. Lisäksi he tuntevat olevansa mukana tekemässä päätöksiä. Toivottavasti todellisuus myös vastaa tätä tuntemusta eli kaikki projektin jäsenet ovat jollakin tasolla mukana päätöksenteossa.



Kuva 7-2 Viestinnän tilat

7.2.1 Toimintatavat

Papen [Pape, 1996] kokemukset hajautetusta toiminnasta vahvistavat käsitystä viestinnän tärkeydestä. Hänen näkökulmastaan ensisijassa johtavassa asemassa olevien projektin jäsenten tulee huolehtia viestinnän ja työntekijöiden välisen kanssakäymisen toimivuudesta. Edelleen Papen mielestä ihmislähtöisyys liiketoiminnassa ja erityisesti asiakassuhteissa mahdollistaa hyvän menestyksen, mitä mikään sähköinen työkalu ei voi tehdä. Hän listaa joitakin periaatteita, joita noudattamalla hajautetun projektin viestintää voidaan parantaa tai ylläpitää. Nämä periaatteet on todettu toimiviksi tutkielman esimerkkiprojektissa. Ne eivät ratkaise kaikkia viestintäongelmia mutta ovat hyvä ohjenuora toiminnalle.

- Vaikka toimitaan hajautetusti, Papen mielestä on tärkeätä, että sekä projektipäälliköt että muu johtava henkilöstö vierailee säännöllisesti sivukonttoreissa tapaamassa työntekijöitä henkilökohtaisesti.
- Epämuodollisen viestinnän lisäämistä tulisi rohkaista. Tämä on mahdollista mm. puhelimen, videoneuvottelujen ja epämuodollisten keskustelukanavien avulla. Näiden käyttöönottamisessa ja käyttämisessä auttaa toimintatapojen määrittely. On hyvä sopia esimerkiksi koska ja minkälaisissa tilanteissa pidetään videoneuvottelu tai milloin käytetään puhelinta. Usein epämuodollisesti käyty keskustelu saattaa johtaa tuloksiin nopeammin kuin jos ryhdyttäisiin organisoimaan virallista videoneuvottelua tai tapaamista.
- Eräs erittäin tärkeä havainto on, että ristiriitojen ratkaisu sähköpostilla on hankalaa. Usein pienet ärsyttävät asiat kerääntyvät ja saavat aikaan suuremman konfliktin. On tärkeätä oppia ”olemaan hajautetusti eri mieltä”.

- Työntekijät täytyy pitää mahdollisimman hyvin ajan tasalla. Eräs keino on lähettää koko projektin henkilöstölle säännöllinen tiedotusposti.

Ensimmäinen kohta on tärkeä siinä mielessä, ettei synny eräänlaista Pääkonttori vs. Sivukonttori –asetelmaa. Se saattaisi työntekijöiden mielessä asettaa toimipisteet eriarvoiseen asemaan. Heidän on tunnettava olevansa tilanteen tasalla ja tärkeä osa yrityksen toimintaa. Kolme muuta kohtaa ovat myös erittäin tärkeitä huomioita. Viestintää helpottaa tietenkin myös yhtenäinen kieli. Sama kieli tulisi olla käytössä koko projektissa asiakasta myöten.

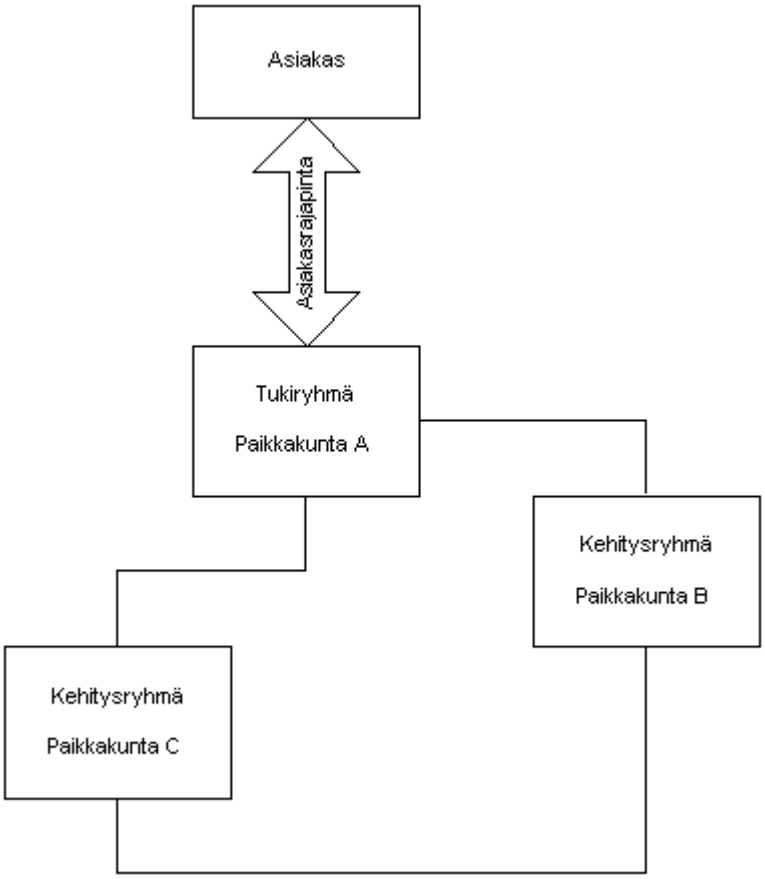
Toimivan viestinnän varmistamiseksi on tärkeätä, että kaikki projektin jäsenet tuntevat toisensa mahdollisimman hyvin. Tietenkin suurissa hajautetuissa projekteissa tämä saattaa olla käytännössä mahdotonta. Eräs keino voisi olla järjestää projektin alkuvaiheessa tapaaminen, johon kaikki osallistuvat ja jossa epämuodollisen tapaamisen ja tutustumisen ohella sovitaan yhteisistä tavoitteista, työnjaosta ja määritellään toimintatavat. Sovittuja toimintatapoja voidaan myöhemmin muuttaa tarpeen mukaan, mutta alussa on oleellista varmistaa riittävä viestintä. Näin helpotetaan epämuodollisen viestinnän kehittymistä projektin sisällä. Tapaamisia on syytä pitää säännöllisesti myöhemminkin. Ne voisivat olla ryhmän sisäisiä tapaamisia, joissa tavataan projektin muita jäseniä ja viestitään projektin tila sekä mahdollisesti tehdään joitakin päätöksiä.

7.2.2 Asiakasrajapinta

Asiakasrajapinta on joukko henkilöitä, yksi tai useampia, jotka hoitavat viestinnän asiakkaalle ja jonka kautta asiakas viestii projektille. Ajatus on esitetty kuvassa 7-3. Vaikka ajatus on, että kaikki viestintä asiakkaan ja projektiryhmän välillä tapahtuu tämän rajapinnan kautta, ei ole hyvä viedä periaatetta liian pitkälle. Joustavuus takaa, että viestinnästä ei tule hankalaa ja tehotonta. Hajautetussa projektissa on ihanteellista jos asiakasrajapinta sijaitsee mahdollisimman lähellä asiakasta. Tällöin viestintä asiakkaan kanssa on mahdollisimman sujuvaa. Siten vältytään väärinkäsityksiltä ja epäselvyyksiltä sekä mahdolliset ristiriitatilanteet voidaan tehokkaimmin ratkaista.

Hyvä käytäntö olisi jos ainakin projektipäällikkö toimisi tässä rajapinnassa. Tällöin hän voisi koordinoida viestintää ja kohdistaa sitä oikeille henkilöille sekä suodattaa turhat asiat pois antaen muiden keskittyä oleellisiin asioihin. Tämä käytäntö on esimerkkiprojektissamme havaittu toimivaksi. On myös otollista, jos asiakasrajapinnassa tai sen lähellä (maantieteellisesti) toimii projektin jäseniä, jotka

vastaavat tukitoimista ja projektin suunnittelusta. Tätä on kuvattu kuvassa 7-3. Tukitoimia voisivat olla esimerkiksi laadunvarmistus, mittaus ja testaus. Toteutustyöstä vastaavien projektin jäsenten ei tarvitse eikä ehkä toisinaan kannatakaan olla näin lähellä asiakasta. Esimerkkiprojektissamme havaitsimme, että lähinnä asiakasta oleva ryhmä joutuu usein käyttämään huomattavan osan ajastaan sellaisiin toteutustyötä tukeviin toimiin, jotka eivät suoranaisesti vie kehitystyötä eteenpäin. Edellä mainitulla tavalla toimimalla kehitystyötä voitaisiin tehdä ilman häiriöitä.



Kuva 7-1 Asiakasrajapinta

7.3 Dokumenttien hallinta

Dokumenttien hallinta pitää sisällään varsinaiset ohjelmiston määrittelydokumentit, lähdekoodit ja teknisen dokumentaation. Seuraavassa käsitellään erikseen dokumentit ja lähdekoodien hallinta konfiguraationhallinnan muodossa. Hyvä periaate on, että projektissa ei tuoteta mitään sellaisia dokumentteja tai välituloksia, joita ei myöhemmissä vaiheissa hyödynnetä [Ruuska, 2001]. Ensin on siis tiedettävä mitä

tietoa missäkin vaiheessa tarvitaan. Sen jälkeen tieto pitää dokumentoida, ja jatkossa kaikkia aikaansaatuja dokumentteja tulee ylläpitää.

7.3.1 Dokumentit

Tavoite dokumenttien hallinnan toteuttamisessa on, että dokumenttien käsittely ja muokkaus on mahdollisimman helppoa ja läpinäkyvää. Näin vältetään turhaa työtä ja vääristä versioista johtuvia ongelmia. Noudatetaan seuraavia periaatteita:

- Kaikki dokumentit pidetään versionhallinnassa. Tämä täytyy tehdä jo siksikin, että tutkielman alussa esitetyn ohjelmiston määritelmän mukaan ne ovat osa ohjelmistoa ja niiden tulisi siten olla mukana konfiguraationhallinnassa. Lisäksi näin kaikilla osapuolilla on saatavilla aina oikea versio helposti ilman sähköpostien lähettelyä.
- Kaikilla projektin jäsenillä tulee olla pääsy ja muokkausoikeudet kaikkiin dokumentteihin.

On usein hyödyllistä seurata dokumentin muutoshistoriaa, jotta voidaan selvittää kenen toimesta ja miksi muutos on tehty, tai palata takaisin aiempaan versioon [Whitehead, 2001]. Lähdekoodien osalta näin on toimittu jo pitkään mutta ohjelmiston dokumentointi ja dokumenttien hallinta muilta osin tehdään tyypillisesti manuaalisesti kopioimalla ja tiedostoja siirtämällä. Muutoksia seuraamalla saavutetaan parempi muutosten hallinta ja lisäksi automaattinen arkisto dokumenteille.

Edellisessä luvussa lueteltujen periaatteiden lisäksi on syytä noudattaa seuraavia ohjeita:

- Ei liitetä tiedostoja sähköposteihin eikä suoriteta manuaalista kopiointia yleensäkään. Posteissa ja muussa viestinnässä viitataan haluttuun dokumenttiin.
- Dokumentit on pidettävä ajan tasalla.
- Dokumentteja tulee voida myöhemmin hyödyntää.
- Myös asiakkaalla tulisi olla rajoitettu pääsy asianmukaisiin dokumentteihin.

7.3.1.1 DeltaV

DeltaV on protokolla, joka mahdollistaa versioinnin ja konfiguraationhallinnan www-palvelimella sijaitseville dokumenteille [Whitehead, 2001]. Se tukee hyvin hajautettuja projekteja. DeltaV –protokollaa tukevalla palvelimella voidaan

esimerkiksi versioida hajautetusti päivitettävää dokumenttia tai www-sivuja. Sitä voidaan myös käyttää perinteisen versionhallinnan tai oikeammin konfiguraationhallinnan tapaan ohjelmistokehityksessä. DeltaV on avoin ja standardeihin perustuva protokolla. Sitä kehittää IETF:n (Internet Engineering Task Force) WebDAV –projektin DeltaV –työryhmä. Se on WebDAV –protokollan (Web Distributed Authoring and Versioning) laajennus, josta kehitetään standardi HTTP:n päällä tapahtuvalle versioinnille ja konfiguraationhallinnalle. WebDAV puolestaan on rakennettu HTTP:n päälle. Nimestään huolimatta WebDAV:ssa ei ole versiointiominaisuuksia, vaan vain tuki dokumenttien yhteistoiminnalliseen muokkaukseen. Vasta DeltaV toteuttaa jo WebDAV:iin aiotut versiointi- ja konfiguraationhallintaominaisuudet. Taulukossa 7-1 on lueteltu HTTP –protokollan metodit sekä sitä laajentavien WebDAV:n ja DeltaV:n metodit.

WebDAV:ia tukevia ohjelmistoja on jo useita, esimerkkeinä MS Office 2000, Adobe Acrobat 5, Photoshop 6 ja Macromedia Dreamweaver 4. DeltaV –protokolla on vielä Draft –asteella mutta kokeiluluonteinen toteutus on jo tehty [Hunt ja Reuter, 2001]. Joitakin ongelmia löydettiin mutta pääsääntöisesti kuitenkin protokollaa pidettiin onnistuneena. Näyttää hyvinkin todennäköiseltä, että versiointi ja konfiguraationhallinta ovat tuettuja monissa dokumentointi- ja kehitystyökaluissa tulevaisuudessa. Silloin voitaisiin esimerkiksi DeltaV:tä tukevalla tekstinkäsittelyohjelmalla muokata dokumentteja, jotka olisivat keskitetysti DeltaV:tä tukevalla palvelimella hallittuja, versioituja ja konfiguraationhallinnan piirissä. Sama palvelin hallitsisi myös lähdekoodeja, joita muokattaisiin jollakin DeltaV:tä tukevalla kehitystyökalulla. Tämä on vain yksi esimerkki monista mahdollisuuksista.

DeltaV, Web Versioning and Configuration Management Protocol CHECKIN, CHECKOUT, UNCHECKOUT, VERSION-CONTROL, REPORT, UPDATE, LABEL, MERGE, MKWORKSPACE, BASELINE-CONTROL, MKACTIVITY
WebDAV, Distributed Authoring Protocol LOCK, UNLOCK, PROPFIND, PROPPATCH, COPY, MOVE, MKCOL
HTTP, Hypertext Transfer Protocol GET, HEAD, POST, PUT, DELETE, OPTIONS, TRACE, CONNECT

Taulukko 7-1 HTTP, WebDAV ja DeltaV –protokollien määrittelemät metodit

Tärkeintä on, että kaikki projektin dokumentaatio on kaikkien projektin jäsenten saatavilla ja muokattavissa mahdollisimman tehokkaasti. Lisäksi versiohistoria tulee olla tallessa. DeltaV:n kehityksen ollessa vielä kesken paras ratkaisu lienee tallettaa kaikki dokumentit käytössä olevaan konfiguraationhallintajärjestelmään.

7.3.2 Konfiguraationhallinta

Konfiguraationhallinta on kaikkien kehitysympäristöjen perusta. Hyvä ratkaisu luo hyvän pohjan lopulle ympäristölle. Huono konfiguraationhallintatuki saattaa tehdä kehitysympäristöstä lähes käyttökelvottoman. Myöskään pelkkä versionhallinta ei riitä. Onneksi käytännössä kaikki nykyisin käytössä olevat konfiguraationhallintatyökalut tarjoavatkin enemmän ominaisuuksia [Dart, 1992]. Hyvä kehitysympäristö luo edelleen hyvän pohjan koko ohjelmistokehitystyölle. Tarkastelen ympäristöjä ja muita työkaluja tarkemmin jäljempänä. Kuten aiemmin kohdassa 5.3 todettiin, hyvä konfiguraationhallinta on vain noin 10%:sesti tekninen ratkaisu. Loppu on johtamista, toimivaa prosessia sekä käyttäjien koulutusta ja osaamista. Siten siis voidaan päätellä, että yrityksen tai projektin on syytä valita konfiguraationhallintajärjestelmä, joka vastaa omia tarpeita, tukee kaikkia tarvittavia ominaisuuksia sekä skaalautuu tarvittaessa. Tästä eteenpäin on kyse lähinnä osaamisesta ja kuvaan tulevat mukaan asiat, joita käsitellään kohdassa 7.4. Seuraavassa esitellään joitakin mahdollisia kehityssuuntia konfiguraationhallinnassa tutkielman lähtökohdista tarkasteltuna.

7.3.2.1 Tulevaisuuden mahdollisuuksia

Mahdollinen CVS:n korvaaja tulevaisuudessa voi olla DeltaV –protokollaan pohjautuva Subversion [Subversion, 2001]. Avoimeen lähdekoodiin ja Apache/BSD –tyyliseen lisenssiin pohjautuvana sitä pidetään todennäköisenä seuraajana CVS:lle. Subversion lupaa tukea kaikkia CVS:n ominaisuuksia sekä paikata siinä vuosien saatossa havaitut puutteet. Kuten edellä dokumenttien hallinnan yhteydessä todettiin, muitakin toteutuskokeiluja on tehty ja protokolla vaikuttaa lupaavalta. DeltaV pohjautuu siis HTTP-protokollaan ja siten sulautuu www-maailmaan saumattomasti. Se tarjoaakin monia mahdollisuuksia tuleville konfiguraationhallintajärjestelmille ja kehitysympäristöille.

Jää nähtäväksi, minkälaisia ratkaisuja konfiguraationhallintaan tulevaisuudessa kehitetään. Tällä hetkellä kuitenkaan ei löydy valmiita DeltaV –protokollaan perustuvia konfiguraationhallintajärjestelmiä. Todennäköisimmät vaihtoehdot ovatkin seuraavassa käsiteltävät ratkaisut.

7.3.2.2 Olemassa olevan ratkaisun täydentäminen

Jos projektin käytössä on jo olemassa oleva konfiguraationhallinta, joka ei tue hajautettua ohjelmistokehitystä sellaisenaan, voidaan käyttää kolmansien osapuolten ohjelmistoja, joilla tuki voidaan lisätä jälkikäteen. Käytetään esimerkkinä tutkielmassa esiteltyä projektia, jossa käytössä on MS SourceSafe. Eräs tällainen ohjelma on SourceOffSite [SourceOffSite, 2001]. Sillä voidaan käyttää SourceSafe-konfiguraationhallintaa hajautetusti. Se on asiakasohjelma, jolla tyypillisesti korvataan SourceSafe–asiakasohjelma hajautetuissa toimipisteissä. Myös UNIX-asiakasohjelma on saatavissa. Se toimii minkä tahansa TCP/IP-yhteyden päällä, tarjoaa salauksen sekä integroituu joihinkin kehitysympäristöihin. Esimerkkiprojektin tapauksessa tämänkaltainen ratkaisu olisi nopeimmin käyttöönotettava ja mahdollisesti kustannuksiltaan edullisin, joskaan ei ehkä pitkällä tähtäyksellä paras. SourceSafen oman asiakasohjelman käyttö on myös mahdollista jos toimipisteet on yhdistetty jollakin VPN-ratkaisulla. Verkkoyhteyden tulee kuitenkin olla kohtuullisen nopea, muuten päivittäinen käyttö on hankalaa.

7.3.2.3 Uusi ratkaisu

Monet kaupalliset tuotteet nykyään tarjoavat kaikki hajautetussa ohjelmistokehityksessä tarvittavat ominaisuudet. Onkin projektin harkittavissa ja päätettävissä mikä ratkaisu on oikea. Tässä esimerkinomaisesti joitakin tuotteita, jotka

täyttävät muun muassa kohdassa 5.3 esitetyt vaatimukset ja tukevat hajautettua kehitystä:

- AccuRev/CM
- Rational ClearCase
- Merant PVCS
- Perforce

7.4 Projektin hallinta

Tärkeintä projektin hallinnassa on koordinointi ja projektin tilan viestintä kaikille osapuolille. Hajautetussa projektissa viestinnän puutetta voidaan jossain määrin korvata ja laatua parantaa sillä, että projektin jäsenille on aina oikeaa informaatiota tarjolla mahdollisimman helposti. Tämä yhdistettynä siihen tosiasiaan, että ihmiset haluavat olla aloitteellisia ja tehdä kaiken tarvittavan [Cockburn, 1999], helpottaa hajautettua työskentelyä.

Kaikessa ohjelmistotyössä tärkeätä on työn kehityksen mittaaminen. Tällä helpotetaan projektin seurantaa joidenkin määriteltyjen arvojen perusteella. Mittaaminen antaa arvoja, joiden perusteella projektissa voidaan tehdä päätöksiä, ymmärtää kokonaisuuksia ja kysyä oikeita kysymyksiä [Royce, 1998]. Erityisen tärkeätä hajautetussa projektissa on, että tarpeelliset arvot ovat yhtä helposti saatavilla kaikkialta projektin piiristä. Tämän toteuttamiseen tarvitaan mittaamisen automatisointia ja jonkinasteista raportointia. Aihe on kuitenkin niin laaja, että vaatisi oman tutkielmansa, niinpä tyydymme tässä toteamaan asian tärkeyden.

Eräs tämän tutkielman lähtökohdista nähden tärkeä ja käytännössä havaittu tekijä on yhdenmukaiset kehitysympäristöt ja työkalut kaikilla projektin jäsenillä. Jos kaikki käyttävät täydellisesti yhteensopivia työkaluja, vältytään monilta harmeilta. Laajemmalta kannalta hajautetussa ohjelmistokehityksessä tämä ei useinkaan ole mahdollista mutta tässä yhteydessä vaatimus on kohtuullinen ja teknisesti helposti toteutettavissa. Aina on kuitenkin jossain määrin välttämätöntä, että projektin jäsenet käyttävät yhteensopivia työkaluja. On vain määriteltävä pienin yhteinen tekijä. Seuraavassa pohditaan tarkemmin joitakin suurempia kokonaisuuksia, jotka on havaittu ongelmallisiksi.

7.4.1 Ongelmien hallinta

Gaon ja Kudon havaintoja ongelmien hallinnasta hahmoteltiin kohdassa 5.4.1. Näiden havaintojen perusteella hahmotellaan luvussa 8 ongelmienhallintajärjestelmää. Yksi

lisäys Kudon malliin on kuitenkin paikallaan. Virheen tai ongelman tunnistaneen ryhmän tai henkilön on voitava osoittaa se suoraan asianomaiselle ryhmälle tai henkilölle. Usein projektin piirissä tiedetään toisten vastuualueet, jolloin on todennäköisesti turha kierrättää tietoa projektipäällikön kautta. Olisi lisäksi toivottavaa, että ongelmista ja niiden ratkaisuista jäisi tieto talteen johonkin tietämystietovarastoon.

7.4.2 Työnjako

Hajautetussa projektissa selvä ja toimiva työnjako on todella tärkeä. Koska välitön viestintä on hankalampaa kuin yhtenä ryhmänä työskenneltäessä, on työnjaon ja toimintamallien oltava kaikille selvät. Jotta työnjako projektiryhmässä olisi selkeä, on tärkeätä määritellä projektissa vaadittavat roolit. Kun projektissa on kaikki tarpeelliset roolit ja roolien haltijat sekä tuntevat että osaavat suorittaa rooliinsa liittyvät tehtävät, saavutetaan selkeän työnjaon lisäksi paremmat mahdollisuudet toimivaan viestintään. Tarpeellisia rooleja ovat projektipäällikkö, projektinsuunnittelija, mittauksen suunnittelija ja laadunvarmistaja kuten kohdassa 5.4.2 todettiin. Edellä mainitut liittyvät erityisesti projektin hallintaan. Lisäksi tarvitaan teknisiä rooleja kuten vaatimusmäärittelijä, suunnittelija, ohjelmoija, testauksen suunnittelija ja testaaja. Nämä liittyvät projektin varsinaiseen toteutustyöhön. Projektipäällikön roolia lukuun ottamatta kaikki roolit voi toteuttaa yksi tai useampia henkilöitä. Edelleen yksi henkilö voi toteuttaa useamman kuin yhden roolin.

7.4.3 Yhteistyö ja sen koordinointi

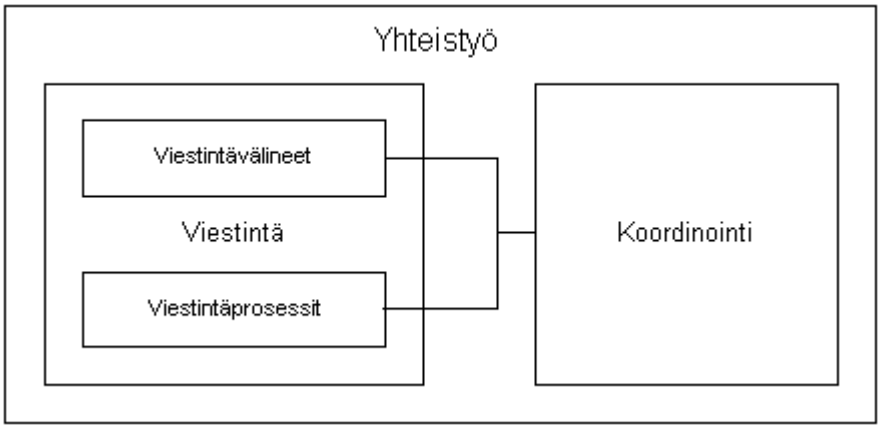
Koordinointi perustuu riittäviin viestintäprosesseihin ja –välineisiin [Altmann ja Dopler, 1998]. Se sisältää kaikki toimet, joita tarvitaan yksittäisten tehtävien synkronisoimiseksi laajemman kokonaisuuden sisällä. Jotta koordinointi olisi tehokasta, viestintävälineiden tulee tukea käytettäviä sähköisessä muodossa olevia dokumentteja. Niitä siis pitää voida lukea, muokata ja siirtää. Yksinkertaisesti projektin koordinointi koostuu tehtävien jakamisesta alitehtäviin, niiden jakamisesta ryhmän jäsenille, alitehtävien järjestämisestä ajallisesti sekä alitehtävien tulosten yhdistämisestä. Jos tässä on onnistuttu, ryhmän jäsenet työskentelevät saavuttaakseen yksityiset tavoitteensa alitehtävän parissa. Alitehtävien yhdistäminen tuottaa tuloksia, jotka saavuttavat projektille asetetut tavoitteet.

Yhteistyö on eräänlaista koordinointia ryhmän jäsenten välillä [Altmann ja Dopler, 1998]. Samassa lähteessä on listattu yhteistyön asettamia vaatimuksia, jotka toimivat

mielestäni hyvänä runkona toimivalle hajautetun työn ja erityisesti hajautetun ohjelmistokehityksen koordinoinnille:

- tavoitteiden tunnistaminen; yhteistyön osapuolten on oltava yhtä mieltä yhteisistä tavoitteista
- suunnitelmien yhteensopivuus; yhteistyön osapuolten suunnitelmien on oltava yhteensopivia
- resurssien vaihto
- mukauttaminen; suunnitelmien yhteensopivuutta ja resurssien vaihtoa ei voida ehkä täysin määritellä etukäteen, tarvitaan joustavaa neuvottelua ja sopeuttamista
- kontrolli; Yhteistyön osapuolten toimien kontrolloinnin on oltava mahdollista, jotta voidaan arvioida kehityksen astetta.

Yhteistyötä voidaan lisäksi luokitella useilla tavoilla. Keskityin tässä kuitenkin vain ohjelmistoprojekteille ominaiseen ryhmäperustaiseen ja projektimuotoiseen yhteistyöhön. Kuvassa 7-4 on Altmannia ja Dopleria [Altmann ja Dopler, 1998] mukaillen muodostettu yksinkertainen malli ryhmän yhteistyöstä. Kuten mallista nähdään ja aiemmin todettiin, tarvitaan yhteistyön aikaansaamiseksi viestintää. Koordinointi perustuu viestintään ja yhteistyö on koordinointia projektin jäsenten välillä. Jälleen siis korostuu viestinnän merkitys. Aiemmin on jo todettu miten ratkaisevan tärkeää viestintä on hajautetuissa projekteissa. Näin siis lähes joka kannalta aihetta tarkasteltaessa päädytään samaan päätelmään, että pohjimmaisena tekijänä on viestintä.



Kuva 7-1 Malli ryhmän yhteistyöstä

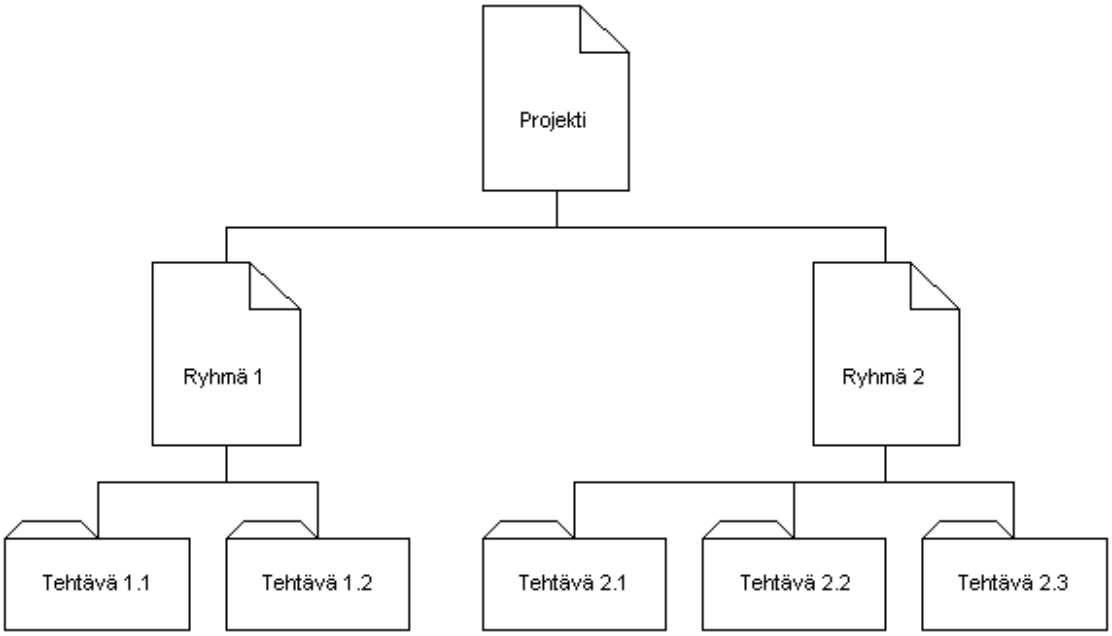
8. Hahmotelma hajautettua ohjelmistokehitystä tukevaksi järjestelmäksi

Luvun tavoite on kuvailla hahmotelmani järjestelmästä, jossa on toteutettu edellisessä luvussa esitetyt ratkaisut. Tämä hahmotelma koostuu periaatteista, toimintaehdotuksista sekä joistakin ehdotuksista teknisiksi ratkaisuiksi. Kohdassa 8.1 kootaan yhteen hahmotelma tällaisen järjestelmän mallista sekä kohdassa 8.2 ehdotus siitä miten se voitaisiin toteuttaa.

8.1 Järjestelmän ydin

8.1.1 Ytimen malli

Altmannia ja Dopleria mukaillen muodostettu malli (kuva 8-1) ohjelmistokehitysprojektin organisoinnista on periaatteiltaan hyvin yksinkertainen ja perinteinen. Kuten aiemmin koordinoinnin yhteydessä mainittiin, projekti jaetaan hierarkkisesti alitehtäviin, tehtävät jaetaan projektiryhmän jäsenille, tehtävät järjestetään ja tulokset yhdistetään. Tämä malli kuvaa projektin jakamisen alitehtäviin. Ylimmällä tasolla (kuva 8-1) on *projekti*, joka kuvaa koko projektia. Siihen liittyy projektikohtaista tietoa jäsenistä, ryhmistä, dokumenttipohjista, ohjeista, standardeista, aikatauluista ja työmääristä. Projekti koostuu yhdestä tai useammasta *ryhmästä*. Ryhmä koostuu joukosta yhteen liittyviä *tehtäviä*, joista jokaista suorittaa joku projektin jäsen. Tehtävä on joku työkokonaisuus, joka on osoitettu projektin jäsenen suoritettavaksi. Siihen liittyy vastuuhenkilö, toteuttaja, aikaraja, määrittely, muut osallistujat, tarvittavat artefaktit sekä mahdolliset alitehtävät. Malli kuvaa tehtävien lisäksi myös projektin organisointia.



Kuva 8-1 Malli projektista

8.1.2 Ryhmien muodostaminen

Tyypillisesti ryhmät muodostetaan maantieteellisin perustein. Ihannetapauksessa järjestelmä tekisi ohjelmistoprojektin hajautuksesta tehtävien koordinoinnin kannalta täysin läpinäkyvää. Projektin jäsenet voisivat olla missä tahansa ja työnjako voitaisiin tehdä projektin jäsenten osaamisen perusteella optimaalisesti. Kuitenkin esimerkiksi epämuodollisen ja välittömän viestinnän, ns. kahvipöytäkeskustelujen, merkitys on niin tärkeä, että jako on käytännössä aina kompromissi. Vaikka samantyyppistä osaamista olisi eri paikkakunnilla ja olisi houkuttelevaa muodostaa hajautetun projektin sisälle hajautettu ryhmä, ei se todennäköisesti toimisi yhtä hyvin kuin yhdessä työskentelevä ryhmä. On todennäköisintä, että jako ryhmiin tapahtuu maantieteellisin perustein. Ryhmän sisäisen viestinnän kannalta tämä on paras ratkaisu. Käytettävään ohjelmistokehitysprosessiin tai työkaluihin ei oteta kantaa.

8.2 Järjestelmän toteutushahmotelma

8.2.1 Käyttötavat

Tavoite on, että järjestelmä olisi käytettävissä organisaation sisäisessä verkossa sekä internetissä. Tällöin on yleensä välttämätöntä käyttää salattua yhteyttä. Järjestelmää käytetään ainakin selaimella. On hyvä ennakoida myös muita käyttötapoja, esimerkiksi erilaisia mobiileja päätelaitteita. Tällöin tavoitettavien palveluiden määrää voidaan rajoittaa laitteen asettamien rajoitusten mukaiseksi. Myös yrityksen ulkopuolisille, siis alihankkijoille, partnereille ja asiakkaille voitaisiin ajatella

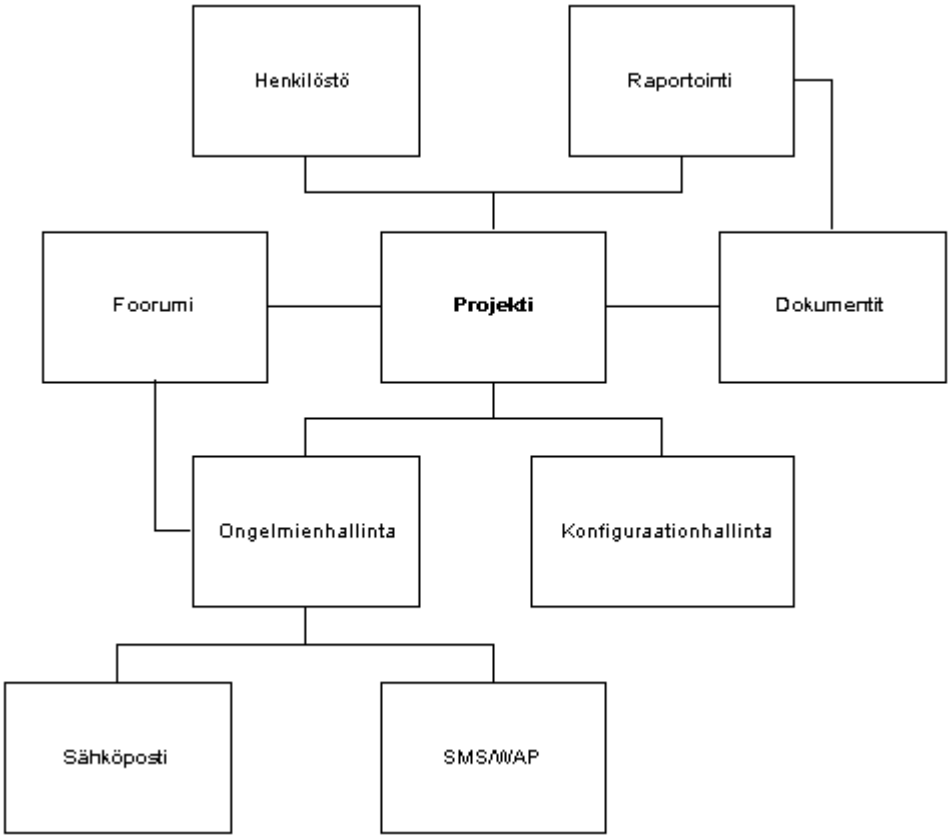
rajoitettua pääsyä järjestelmään. Näin he voisivat saada luotettavasti ja varmasti ajantasaisia dokumentteja ja raportteja projektista sekä viestiä projektiryhmän kanssa. Käyttäjä voi räätälöidä omaa liittymäänsä järjestelmään. Hän voi esimerkiksi määritellä haluaako ilmoituksen jostakin järjestelmän tapahtumasta sähköpostiinsa tai tekstiviestinä vai seuraako tapahtumia itse aktiivisesti.

8.2.2 Järjestelmän osat

Kuvan 8-2 mallissa on kuvattu osia, jotka muodostavat järjestelmän. Mallin on tarkoitus kuvata osien suhteita, ei toimia minkäänlaisena arkkitehtuurisuunnitelmana. Joitakin näistä osista on saatavilla valmiina tuotteina. Seuraavassa lyhyt kuvaus jokaisesta.

- *Projekti* kuvaa yksittäisen projektin. Se noudattaa karkeasti ottaen edellä kohdassa 8.1 esitettyä mallia. Tämä on koko järjestelmän ydin, joka sitoo muut osat yhteen.
- *Henkilöstö* sisältää tiedot työntekijöistä ja resursseista. Näitä tietoja ovat mm. työntekijöiden yhteystiedot, osaaminen ja työtehtävät. Tämä osa voi olla yhteydessä johonkin laajempaan henkilöstön- tai yrityksenhallintajärjestelmään.
- *Raportointi* muodostaa raportteja. Projektipäällikkö raportoi asiakkaalle tai tekee raportteja sisäiseen seurantaan. Raportit tallentuvat automaattisesti projektin dokumenttikantaan ja ne linkitetään projektin tietoihin.
- *Dokumentit* on dokumenttikanta DeltaV –protokollaa tukevalla www-palvelimella ja sen mukaisessa tietovarastossa (tiedostojärjestelmä tai tietokanta). Dokumenttien muokkaus tapahtuu protokollaa tukevilla työkaluilla.
- *Ongelmienhallinta* on kohtien 5.4.1 ja 7.4.1 mukainen järjestelmä, jolla hallitaan ongelmia ja ohjelmistovirheitä sekä seurataan niiden ratkaisua. On saatavissa myös valmiita tuotteita.
- *Foorumi* on keskustelupalsta, jossa jokaisella projektilla on oma alueensa, jonka alle luodaan automaattisesti ja manuaalisesti tarpeelliset alaryhmät. Foorumissa julkaistaan projektin jäsenten keskustelujen lisäksi automaattisesti monia järjestelmän tapahtumia.
- *Konfiguraationhallinta* on organisaation valitsema konfiguraationhallintajärjestelmä. Mahdollisuuksia kartoitettiin kohdassa 7.3.2.

- Ilmoitukset järjestelmän tapahtumista voidaan ohjata *sähköpostiin* jos käyttäjä niin haluaa.
- Ilmoitukset järjestelmän tapahtumista voidaan ohjata *tekstiviestiksi* tai muokata jollakin *WAP* –laitteella tarkasteltaviksi jos käyttäjä niin haluaa.



Kuva 8-1 Järjestelmän malli

8.2.3 Ehdotuksia teknisiksi ratkaisuiksi ja jatkokehitysmahdollisuuksia

Kuten edellä todettiin, dokumenttien versiointi voitaisiin hoitaa DeltaV –protokollan avulla. Valmiita tuotteita ei vielä ole, joten riippuu paljolti protokollan kehityksestä ja ohjelmistovalmistajista tuleeko tämä ehdotus toteutumaan. Tarvitaan kuitenkin siis www-palvelin ja tietovarasto. Tietovaraston tulisi mieluummin olla tietokanta koska muutakin tietoa pitää voida tallettaa ja näistä tiedoista tullaan tekemään raportteja. *Sähköposti* ja *SMS/WAP* ovat vain komponentteja, jotka välittävät viestejä kyseisiin järjestelmiin.

Tietosisältö siis talletetaan tietokantaan, josta sitä voidaan hakea ja josta siitä voidaan tehdä raportteja. Kun näin saadaan talteen runsaasti tietoa projektin tapahtumista ja ratkaisuista, voidaan ehkä miettiä miten tallennettu tieto voitaisiin käyttää hyödyksi tulevaisuuden projekteissa. Tällainen tietämyksen kerääminen kuitenkin on niin vaativa alue, että jätämme sen jatkotutkimusaiheeksi. Vaativinta tässä järjestelmässä on ehkä osajärjestelmien välisten yhteyksien toteuttaminen, erityisesti jos käytetään valmiita tuotteita.

9. Yhteenveto

Luvun tavoite on tehdä loppuyhteenveto tutkielman tuloksista. Ensimmäisessä kohdassa kerrataan tulokset, seuraavassa asetetaan joitakin rajoituksia tuloksille. Tämän jälkeen esitellään muutamia mahdollisia jatkotutkimusaiheita.

9.1 Havainnot

Tutkielman keskeinen havainto on, että siirryttäessä hajautettuun työhön, tärkeintä on säilyttää tai kehittää toimiva viestintä riippuen lähtötilasta. Jos viestintä ei toimi, mikään muukaan osa-alue ei voi toimia. Muut hajautuksesta aiheutuvat ongelmat liittyvät pääasiassa ohjelmistokehityksen tukitoimintoihin. Tutkielman ongelmat voidaan tiivistää kolmeen ryhmään: viestintään, dokumenttien hallintaan ja projektin hallintaan. Dokumenttien käsittelyn ja muokkauksen on oltava mahdollisimman helppoa ja läpinäkyvää. Kaikki dokumentit pidetään versionhallinnassa ja kaikilla projektin jäsenillä tulee olla pääsy niihin. Konfiguraationhallintajärjestelmän tulee vastata omia tarpeita, tukea hajautusta, tukea kaikkia muita tarvittavia ominaisuuksia sekä skaalautua tarvittaessa. Projektin hallinnassa tärkeintä on koordinointi ja projektin tilan viestintä kaikille osapuolille. Hajautetussa projektissa viestinnän puutetta voidaan jossain määrin korvata ja laatua parantaa sillä, että projektin jäsenille on aina oikeaa informaatiota tarjolla mahdollisimman helposti.

9.2 Tulosten rajaus

Tulokset rajoitetaan esimerkkiprojektin kaltaiseen tilanteeseen, jossa hajautus on lähinnä ohjelmistokehitystä tekevän yrityksen sisäistä. Asiat eivät kuitenkaan muutu kovin ratkaisevasti, vaikka huomioon otettaisiin myös yhteistyökumppaneita, muita yrityksiä ja asiakkaita. Vaikka tutkielma on tehty hajautettujen ohjelmistokehitysprojektien toiminnan parantamiseksi, tuloksia voidaan hyödyntää jollakin tasolla hyvin monenlaisissa hajautetuissa projekteissa koska kyse on äärimmäisen perustavanlaatuisista ratkaisuista.

9.3 Jatkotutkimusaiheita

Tutkielman rajauksista johtuen monia aiheita on jouduttu jättämään käsittelemättä. Eräitä näistä pitäisin hedelmällisinä jatkotutkimusaiheina. Seuraavassa luettelen nämä lyhyiden kuvausten kera. Joidenkin tiimoilta on jo tutkimusta suoritettukin ja siinä tapauksessa mainitsen jonkin lähteen, josta aiheeseen voi hyvin päästä käsiksi.

- Miten luvussa 8 kuvattu järjestelmä voidaan toteuttaa? Loogisin jatko tälle tutkielmalle olisi tietenkin toteuttaa järjestelmä, joka hahmoteltiin edellisessä luvussa. Se vaatii kuitenkin melkoisesti työtä jo siitäkin syystä, että sen sisältä voidaan tunnistaa useita itsenäisiä kokonaisuuksia, jotka vaativat tarkempaa paneutumista aiheeseen.
- Minkälaisella järjestelmällä tai ohjelmistolla voitaisiin hallita ohjelmistoja niiden kehityskaaren alusta asiakkaalle toimitukseen ja tuotantoasennukseen asti? Tästä aiheesta on tehty tutkimusta ja se vaikuttaakin äärimmäisen mielenkiintoiselta [Murer ja Van De Vanter, 1999].
- Kuinka neuvottelut toteutetaan hajautetussa ohjelmistokehityksessä? Miten neuvottelut voidaan hoitaa tehokkaasti hajautetussa ohjelmistokehityksessä? Mitä välineitä käyttäen; videoneuvottelu, ryhmäohjelmat. Tähän aiheeseen liittyy läheisesti myös muita tieteenaloja.
- Miten mittaus ja raportointi järjestetään hajautetussa projektissa? Erityisen tärkeätä hajautetussa projektissa on, että tarpeelliset arvot ovat yhtä helposti saatavilla kaikkialta projektin piiristä. Tämän toteuttamiseen tarvitaan mittaamisen automatisointia ja jonkinasteista raportointia.
- Miten hajautetun projektin tietämys saadaan talteen ja hyödynnettyä? Kuvailin tätä aihetta kohdassa 8.2.3. Hyvä lähtökohta voisi olla esimerkiksi kohdassa 5.1 esitelty lähde [Maurer ja Dellen, 1998].

Lähteet

[Alho *et al.*, 1999] Kari Alho, Timo Helander, Pekka Isto, Arno Karatmaa ja Reijo Sulonen, *A Framework for Engineering in Virtual Enterprises*. Helsinki University of Technology, TAI Research Centre, Helsinki, 1999.

[Alho ja Sulonen, 1998] Kari Alho ja Reijo Sulonen, *Supporting Virtual Software Projects on the Web*. Submission for Workshop on Coordinating Distributed Software Development Projects, IEEE 7th International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises, Stanford University, 1998.

[Altmann ja Dopler, 1998] Josef Altmann ja Heinz Dopler, *Organizational Aspects of Distributed Software Development*. Research documentation, Johannes Kepler University, Linz, 1998.

[Bendeck *et al.*, 1998] Fawsy Bendeck, Sigrid Goldmann, Harald Holz ja Boris Kötting, *Coordinating Management Activities in Distributed Software Development Projects*. Submission for Workshop on Coordinating Distributed Software Development Projects, IEEE 7th International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises, Stanford University, 1998.

[Cockburn, 1999] Alistair Cockburn, *Characterizing People as Non-Linear, First-Order Components in Software Development*. Humans and Technology Technical Report, TR 99.05, Salt Lake City, 1999.

[Cubranic, 1999] Davor Cubranic, *Coordinating Open-Source Software Development*. Submission for 2nd Workshop on Coordinating Distributed Software Development Projects, IEEE 8th International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises, Stanford University, 1999.

[Cugola ja Ghezzi, 1999] Gianpaolo Cugola ja Carlo Ghezzi, *Design and Implementation of PROSYT: a Distributed Process Support System*. Submission for 2nd Workshop on Coordinating Distributed Software Development Projects, IEEE 8th International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises, Stanford University, 1999.

[Dart, 1992] Susan A. Dart, *The Past, Present, and Future of Configuration Management*. Technical Report, Software Engineering Institute of Carnegie Mellon University, Pittsburgh, 1992.

[Fowler, 2000] Martin Fowler, *The New Methodology*. http://www.martinfowler.com/articles/newMethodology.html, USA, 2000, päivitetty maaliskuussa 2001, tarkastettu 20.11.2001. Julkaistu myös lyhennetyssä muodossa lehden Software Development joulukuun 2000 numerossa.

[Gao *et al.*, 2000] Jerry Z. Gao, Fukao Itaru ja Y. Toyoshima, *Global Problem Management System – Development Experience and Lessons*. Submission for 3rd Workshop on Software Engineering Over the Internet, The 22nd International Conference on Software Engineering, Limerick, 2000.

[Grundy, 2000] John Grundy, *Distributed Component Engineering using a Decentralised, Internet-based Environment*. Department of Computer Science, University of Auckland, 2000.

[Grundy *et al*., 1998] John Grundy, John Hosking ja Rick Mugridge, *Coordinating Distributed Software Projects with Integrated Process Modelling and Enactment Environments*. Submission for Workshop on Coordinating Distributed Software Development Projects, IEEE 7th International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises, Stanford University, 1998.

[Haikala ja Märijärvi, 1998] Ilkka Haikala ja Jukka Märijärvi, *Ohjelmistotuotanto*. Suomen Atk-kustannus Oy, Espoo, 1998.

[Harjumaa ja Tervonen, 2000] Lasse Harjumaa ja Ilkka Tervonen, *Virtual Software Inspections over the Internet*. Submission for 3rd Workshop on Software Engineering Over the Internet, The 22nd International Conference on Software Engineering, Limerick, 2000.

[Hoek, 2000] André van der Hoek, *Configuration Management and Open Source Projects*. Submission for 3rd Workshop on Software Engineering Over the Internet, The 22nd International Conference on Software Engineering, Limerick, 2000.

[Hunt ja Reuter, 2001] James J. Hunt ja Jürgen Reuter, *Using Web for Document Versioning: An Implementation Report for DeltaV*. Report on prototype

implementation of DeltaV based on the 04.5 draft specification, Submission for 23rd International Conference on Software Engineering, Toronto, 2001.

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

[Kudo *et al.*, 1998] Yutaka Kudo, Shinobu Koizumi, Osamu Ohno, Hiroshi Kawabe ja Yukari Furuhata, *Problem Management System for Distributed Software Development*. Submission for Workshop on Software Engineering over the Internet, 20th International Conference on Software Engineering, Kioto, 1998.

[Maurer ja Dellen, 1998] Frank Maurer ja Barbara Dellen, *An Internet Based Software Process Management Environment*. Submission for Workshop on Software Engineering over the Internet, 20th International Conference on Software Engineering, Kioto, 1998.

[Murer ja Van De Vanter, 1999] Tobias Murer ja Michael L. Van De Vanter, *Replacing Copies with Connections: Managing Software across the Virtual Organization*. Submission for 2nd Workshop on Coordinating Distributed Software Development Projects, IEEE 8th International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises, Stanford University, 1999.

[O'Connor ja Jenkins, 1999] Rory O'Connor ja John Jenkins, *Using Agents for Distributed Software Project Management*. Submission for 2nd Workshop on Coordinating Distributed Software Development Projects, IEEE 8th International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises, Stanford University, 1999.

[Pape, 1996] William R. Pape, *Remote Control*. Article in the September 17, 1996 issue of INC. – The Magazine for Growing Companies, USA, 1996.

[Pressman, 1997] Roger S. Pressman, *Software Engineering – A Practitioner's Approach (European Adaptation)*. McGraw-Hill, 1997.

[Royce, 1998] Walker Royce, *Software Project Management: A Unified Framework*. Addison-Wesley, Massachusetts, 1998.

[Ruuska, 2001] Kai Ruuska, *Projekti hallintaan*. Talentum Media, Jyväskylä, 2001.

[SourceOffSite, 2001] SourceOffSite, *A Remote Access Solution for Visual SourceSafe*. http://www.sourcegear.com/sos/, tarkastettu 20.11.2001. SourceGear Corporation.

[Subversion, 2001] Subversion, *A Replacement for CVS*. http://subversion.tigris.org, tarkastettu 20.11.2001.

[VeTO, 2001] Marko Hakonen, Sari Kela, Casper Lassenius ja Kerttu Visuri, *VeTO – Improving the Controllability of Globally Networked R&D Projects*. Helsinki University of Technology, Software Business and Engineering Institute. Esitys Tekesin UTT-seminaarissa 2.2.2001. Tekesin teknologiaohjelma UTT (Uusi teollinen toimintatapa) ja siellä tutkimusprojekti TKK-TAI; Verkostoituneiden tuotekehitysprojektiien ohjattavuuden parantaminen (VeTO).

[Whitehead, 2001] Jim Whitehead, *DeltaV: Adding Versioning to the Web*. WWW10 (10th International World Wide Web Conference) Tutorial Notes, Hong Kong, 2001.