TEKNILLINEN KORKEAKOULU
Sähkö- ja tietoliikennetekniikan osasto

Janne Huttunen

Ketterän ohjelmistokehitysmenetelmän määrittely, vertailu ja käyttäjäkysely

Diplomityö, joka on jätetty opinnäytteenä tarkastettavaksi diplomi-insinöörin tutkintoa varten
Espoossa 11.12.2006

Työn valvoja: Professori Eero Hyvönen

Työn ohjaaja: Agronomi Paavo Tuovinen

TEKNILLINEN KORKEAKOULU DIPLOMITYÖN TIIVISTELMÄ

Tekijä:
Janne Huttunen

Työn nimi:
Ketterän ohjelmistokehitysmenetelmän määrittely, vertailu ja käyttäjäkysely

Päivämäärä: 11.12.2006

Sivumäärä: 84 + 7

Osasto: Sähkö- ja tietoliikennetekniikan osasto

Professuuri: AS-75 Viestintätekniikka

Työn valvoja: Professori Eero Hyvönen

Työn ohjaaja: Agronomi Paavo Tuovinen

Tämän diplomityön tehtävä oli määritellä Finnish Net Solutions Oy:n (FNS) ohjelmistokehitysmenetelmä ja sen suhde muihin yleisesti käytettyihin menetelmiin, selvittää kyselytutkimuksella asiakkaiden mielipiteet käytetystä menetelmästä ja sillä toteutetuista ohjelmistoista sekä esittää näiden perusteella toimenpiteitä menetelmän kehittämiseksi.

Työn aluksi esiteltiin keskeiset yleisesti tunnetut suunnitelmaohjautuvat sekä ketterät ohjelmistokehitysmenetelmät sekä määriteltiin FNS:n käyttämä ohjelmistokehitysmenetelmä ja tutkittiin sen yhtäläisyyksiä ja eroavaisuuksia yleisesti tunnettuihin ohjelmistokehitysmenetelmiin. Menetelmän todettiin kuuluvan ketteriin ohjelmistokehitysmenetelmiin, mutta poikkeavan muista ketteristä menetelmistä erityisesti viestintätapana käytettävän kommentointijärjestelmänsä osalta.

Kyselytutkimuksessa selvitettiin ohjelmistokehitystyöhön kommentointijärjestelmän käyttäjinä osallistuneiden henkilöiden mielipiteitä menetelmällä tuotetuista järjestelmistä sekä varsinaisesta kommentointijärjestelmästä www-sivulla toteutetun kyselyn avulla. Tuloksien perusteella todettiin toteutettujen järjestelmien täyttävän asiakkaiden tarpeet sekä toimivan hyvin siinä tarkoituksessa, mihin ne on suunniteltu. Keskeisimmät asiakkaiden saavuttamat edut menetelmän käytöstä olivat kustannus- ja aikataulusäästöt verrattuna muihin ohjelmistokehitysmenetelmiin.

Kommentointijärjestelmän käyttäjät pitivät kommentointijärjestelmän käyttöä helppona sekä arvostivat Internetissä toimivan järjestelmän tarjoamaa kommentointiajan ja -paikan vapautta. Vastaajat kertoivat pystyvänsä tuomaan mielipiteensä esille järjestelmän kautta. Tuloksena saatiin myös tieto siitä, että kommentoijat haluaisivat käyttää vastaavaa menetelmää myös jatkossa osallistuessaan ohjelmistokehitykseen.

Menetelmän kehittämiseksi suositeltiin äänestysmahdollisuuden liittämistä kommentointijärjestelmään sekä kirjoitetutun viestinnän lisäksi myös muiden viestintätapojen käytön, kuten kuvien ja kaavioiden, mahdollistamista kommentointijärjestelmässä. Muiden ketterien menetelmien parhaiden käytäntöjen, kuten tarinakorttien ja Scrum-menetelmässä esiteltyjen työlistojen hyödyntämistä myös FNS-menetelmässä suositeltiin.

Avainsanat: Ohjelmistokehitys, ketterät menetelmät, suunnitelmaohjautuvat menetelmät, extreme programming, kyselytutkimus

Author:
Janne Huttunen

Title:
Definition, comparison and questionnaire study of an one agile software development method

Date:
11.12.2006

Number of pages:
84 + 7

Department:
Department of Electrical and Communications Engineering

Professorship:
AS-75 Media technology

Supervisor:
Professor Eero Hyvönen

Instructor:
M.sc. agr. Paavo Tuovinen

The goal of this thesis was to define the software development method of the Finnish Net Solutions (FNS) and it's relationship to other commonly used methods. Also a questionnaire study was performed to find out the opinions of the customers on the method and the software products developed by using it. A further aim was to give instructions for further development of the method.

In the first part of the work the most commonly known plan-driven and agile methods were introduced. The development method of FNS was defined and compared to the known methods. The method was found out to fall in to the category of agile methods, but vary from the known methods especially by the commenting system which is used as the main communication method. This introduced method was named as FNS-method.

The questionnaire study studied the opinions of the commenting system users about the products developed with FNS-methods and the commenting system itself. The questionnaire was performed as a www-page. From the results it was seen that the systems developed with the method fulfil the requirements of the users and work well in the purpose in which they were developed. The most important benefits for the customers were cost and time savings compared to other software development methods.

Users of the commenting system considered the usage of the commenting system easy, and appreciated the possibility to use the commenting system in any place and time using the Internet. Respondents were able to express themselves by using the system. Commentators were also interested to use the method again if they were taking part in software development.

For further development of the method it is recommended to add a possibility to vote on issues. It was recommended to make it possible to communicate in the commenting system in other ways than writing, for example by using pictures and diagrams. Also the use of the best practices of other agile development methods in FNS-method was recommended.

Keywords: Software development, agile methods, plan-driven methods, extreme programming, questionnaire study

Alkulause

Tämä diplomityö on tehty Finnish Net Solutions Oy:ssä vastaamaan tarpeisiin, joita yrityksen oman, käytännön työssä syntyneen ohjelmistokehitysmenetelmän käytön aikana on syntynyt. Meille arkipäiväinen ja tuttu menetelmä on nyt tässä työssä koottu kirjalliseen muotoon, jotta sen esittely sitä ennestään tuntemattomille olisi helpompaa. Saamme myös ohjeita jatkaa menetelmän kehittämistä eteenpäin vielä paljon paremmaksi.

Haluan kiittää työn ohjaamisesta agronomi Paavo Tuovista. Hänen antamallaan ohjauksella ja palautteella oli tärkeä rooli työn painotuksien etsimisessä ja sen loppuun saattamisessa. Professori Eero Hyvöstä kiitän työn tarkastamisen lisäksi työn aikana saamistani keskeisistä ohjeista ja neuvoista.

Lämpimät kiitokseni osoitan vanhemmilleni Hellille ja Eskolle sekä veljelleni Juhalle kaikesta siitä kannustuksesta ja tuesta mitä olen saanut tämän diplomityön tekemisen ja koko opiskelujeni aikana. Rakasta Elinaani haluan kiittää niin arkipäiväisen elämän jakamisesta kanssani kuin myös niistä kaikista keskusteluista, joita diplomitöitämme tehdessä olemme käyneet.

Kiitän myös opiskelijaystäviäni kaikista unohtumattomista yhdessä vietetystä hetkistä, jotka toivat paljon piristystä varsinaisen opiskelun oheen.

Espoossa 11.12.2006,

Janne Huttunen

Sisällysluettelo

1. Tutkimuskysymykset ja tavoitteet

Tämän työn perustana oli Finnish Net Solutions Oy:n (FNS) tarve selvittää yrityksen käyttämän ketterän ohjelmistokehitysmenetelmän vahvuudet ja heikkoudet. Yritys on kehittänyt käytännön työn myötä oman, yleisesti käytetyistä menetelmistä erityisesti viestintätapojen osalta merkittävästi poikkeavan tavan toteuttaa ohjelmistoja ja verkkopalveluita. Yritykselle itselleen menetelmä tarjoaa merkittävän nopeus- ja kustannusedun perinteisiin kehitysmenetelmiin verrattuna, mutta tarkkaa tutkimustietoa siitä, millaisia etuja tai haittoja asiakkaille menetelmän käytöstä ohjelmistokehityksessä syntyy, ei ole aikaisemmin ollut olemassa. Tämän työn tehtävänä on korjata tämä puute.

Työn tarkoitus määriteltiin työn alussa mahdollisimman tiiviisti:
työn tehtävä on määritellä FNS:n ohjelmistokehitysmenetelmä ja sen suhde muihin yleisesti käytettyihin menetelmiin, selvittää kyselytutkimuksella asiakkaiden mielipiteet käytetystä menetelmästä ja sillä toteutetuista ohjelmistoista sekä esittää selvitysten perusteella toimenpiteet menetelmän kehittämiseksi.

Tämän tehtävän toteuttamiseksi tässä työssä perehdytään ensin ohjelmistokehitysmenetelmiin yleisesti ja kuvataan sen jälkeen FNS:n omaa tapaan toteuttaa ohjelmistoja. Yleisesti tunnettuja ohjelmistokehitysmenetelmiä ja FNS:n menetelmää vertaillaan. Tämän jälkeen esitellään asiakaskysely, jonka perusteella arvioidaan FNS:n menetelmän hyviä ja huonoja puolia asiakkaiden näkökulmasta. Tämän perusteella esitetään yritykselle parannusehdotukset menetelmän kehittämiseksi.

Tutkimuskysymykset, joihin työssä etsitään vastauksia, ovat seuraavat:

- Mikä FNS:n ohjelmistokehitysmenetelmä on ja mitä uutta se tarjoaa muihin kehitysmenetelmiin verrattuna?
- Mitä etuja FNS:n ohjelmistokehitysmenetelmä tarjoaa asiakkaille ja asiakasohjelmistojen kehittämiseen osallistuville henkilöille?
- Miten kommentointiin osallistuvat asiakkaan edustajat kokevat ohjelmistokehitysmenetelmän?
- Mitä heikkouksia tässä ohjelmistokehitysmenetelmässä on?
- Miten ohjelmistokehitysmenetelmää voidaan parantaa?

Työn lopputuloksena halutaan saada yrityksen käyttöön selkeä kirjallinen kuvaus ohjelmistokehitysmenetelmästä sekä sen hyvistä ja huonoista puolista parannusehdotuksineen.

Työn alussa luvussa kaksi esitellään ohjelmistokehitysmenetelmiä yleisesti ja tarkemmin näistä keskeisimmät ohjelmistokehitysmenetelmät. Menetelmien esittelyssä painotutaan esittelemään perinteisten, suunnitelmaperusteisten menetelmien lisäksi uusia, niin kutsuttuja ketteriä sovelluskehitysmenetelmiä.

Luvussa kolme kuvataan FNS:n ohjelmistokehitysmenetelmä. FNS:n ohjelmistokehitysmenetelmää verrataan luvussa kaksi kuvattuihin sovelluskehitysmenetelmiin sekä arvioidaan, mitä menetelmiä FNS:n käyttämä menetelmä muistuttaa ja miten se eroaa niistä.

Luvussa neljä kuvataan kyselytutkimus, jonka tehtävänä on selvittää asiakkaiden kokemukset FNS:n ohjelmistokehitysmenetelmästä. Aluksi esitellään tutkimuksen toteutus sekä hypoteesit tuloksista, jonka jälkeen esitellään saadut tulokset ja arvioidaan hypoteesien toteutumista.

Luku viisi esittää työhön liittyvän pohdinnan ja johtopäätökset. Aluksi esitellään järjestelmän vahvuudet ja heikkoudet vertailussa muihin menetelmiin, jonka jälkeen esitetään ehdotukset ohjelmistokehitysmenetelmän kehittämiseksi. Luvussa kuusi tehdään tiivis yhteenveto koko työstä.

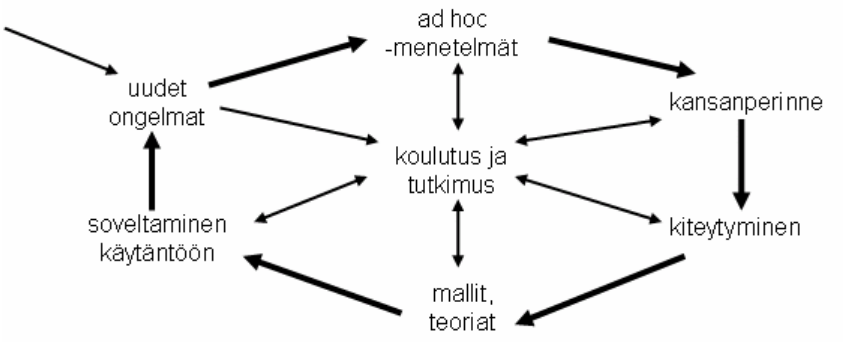
Ohjelmistokehitysmenetelmiin liittyvistä termistöstä monet ovat alun perin englanninkielisiä, eikä niille kaikille ole olemassa vakiintuneita suomennoksia. Tämän vuoksi työssä esitetään useimpien suomennettujen termien ensimmäisen esiintymiskerran yhteydessä suluissa alkuperäinen englanninkielinen termi.

2. Ohjelmistokehitysmenetelmät ohjelmistotuotannossa

Tässä luvussa kuvataan tausta ohjelmistokehitysmenetelmien kehittymiselle, jonka jälkeen käydään läpi keskeisimmät nykyään käytössä olevat ohjelmistokehitysmenetelmät. Menetelmät on jaoteltu kahteen pääryhmään: suunnitelmaohjautuviin ja ketteriin ohjelmistokehitysmenetelmiin.

2.1 Ohjelmoinnista ohjelmistokehitykseen

Ohjelmistokehitys (eng. software engineering, usein suomennettu myös *ohjelmistotuotanto*) esiteltiin terminä ensimmäisen kerran 1960-luvun lopun ohjelmistokriisin aikaan. Kriisin taustalla oli se, etteivät entiset ohjelmistokehitysmenetelmät enää sopineet uusille, tehokkaammille tietokoneille, vaan kehitysprojektit olivat jopa vuosia myöhässä, ylittivät kustannuksensa, ja tuloksena syntyneet ohjelmistot olivat hankalia ylläpitää. Tilanne on joiltain osin samanlainen vielä tänä päivänäkin, sillä tarve ohjelmistoille kasvaa jatkuvasti, mutta kehitysmenetelmien tehokkuus ei kasva samaa tahtia kuin ohjelmistojen kysyntä. Tämä johtaa edelleen vanhojen virheiden, kuten aikataulun venymisen, toistumiseen (Sommerville 1992). 1960-luvun ohjelmistokriisin jälkeen ohjelmien kehitystyöhön alettiin etsiä avuksi prosesseja ja malleja, jotka auttaisivat kohti tehokkaampaa ohjelmistojen tuotantoa. Ensimmäiset *ohjelmistokehitysmenetelmät* alkoivat syntyä muista insinööritieteistä haettujen menetelmämallien pohjalta. Ohjelmistokehityksen voidaan sanoa soveltavan tietojenkäsittelytiedettä ja informatiikkaa samoin kuin perinteinen insinööritaito sekoittaa ja soveltaa matematiikkaa, fysiikkaa ja kemiaa. Ohjelmistokehitystyön systemaattiset työskentelytavat ovat myös hyvin läheisesti yhteydessä muiden insinööritieteiden systemaattisiin toimintatapoihin (Wirgentius 2003). Kuvassa 1 esitetään kehämäinen malli ohjelmistotuotannon syntymiselle, jossa lähtökohtana ovat käytännön ongelmat, joihin etsitään ratkaisuja. Tämä malli sopii niin ohjelmistokehityksen malleihin kuin lähes mihin tahansa muunkin tekniikan alan kehittymiseen.



Kuva 1: Ohjelmistotuotannon kehittyminen (Haikala & Märijärvi 2002)

Haikala ja Märijärvi (2002) kuvaavat vaiheita ovat seuraavasti:

- Aluksi ongelman ratkaisuksi kelpaa mikä tahansa ongelman ratkaiseva tapa. Tällöin sovelletaan ns. ad hoc -menetelmiä.

- Hiljalleen löydetään ratkaisuja, jotka toimivat useammassakin kuin yhdessä tapauksessa. Näistä syntyy alan kansanperinnettä.

- Kun kansanperinteenä välittyvä osaaminen kehittyy järjestelmällisemmäksi, se kiteytetään heuristiikoiksi ja työmenetelmiksi.

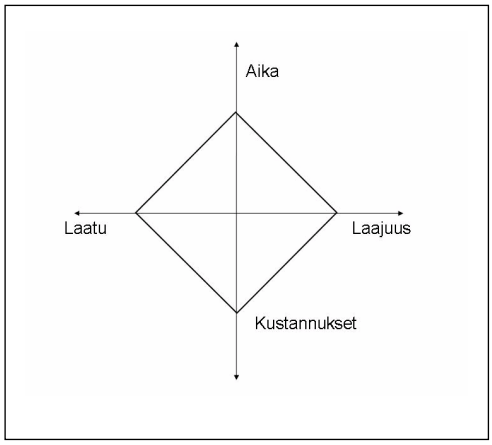
- Jatkossa nämä menetelmät kehittyvät riittävän järjestelmällisiksi muodostaakseen malleja ja teorioita

- Kun malleja sovelletaan käytäntöön, löydetään uusia ongelmia ja kokonaan uusia sovellusalueita, joille entiset ratkaisumallit eivät enää sovikaan – on aloitettava alusta ja palattava ad hoc -menetelmiin.

Ohjelmistokehityksessä tämä malli on edennyt hitaasti kansanperinteestä kohti malleja ja teorioita, palaten aina tarpeen mukaan takaisin aiempiin vaiheisiin.

Mallit ja teoriat sekä käytäntö yhdessä muodostavat ohjelmistokehityksessä käytettävän *ohjelmistokehitysmenetelmän*. Menetelmä terminä määritellään kokoelmaksi erilaisia teknisiä työvaiheita sekä ohjeita siitä, missä järjestyksessä ja millä tavalla nämä työvaiheet tulee suorittaa tietyn tavoitteen saavuttamiseksi. Ohjelmistokehitysmenetelmien tapauksessa on tarpeen tarkentaa, että menetelmä sisältää varsinaisten työvaiheiden lisäksi ohjeita myös ihmisten väliseen kommunikointiin sekä yhteistyöhön (Kalermo & Rissanen 2002).

Ohjelmistokehityksen tärkeisiin perusasioihin kuuluvat muut asiakkaan antamat puitteet, jotka ovat yleisesti tunnettuja kaikessa muussakin projektimuotoisessa toiminnassa. Ohjelmistokehityksen menetelmien valinnan taustalla olevia tekijöitä voidaan hahmottaa nelikenttämallin avulla (kuva 2), joka jakautuu seuraaviin tekijöihin: laajuus, laatu, aika ja kustannus (Lindberg 2003). Kuvion esittämien suureiden välillä pyritään saavuttamaan kussakin projektissa tarvittava painotus painottamalla eri tekijöitä.



Kuva 2: Ohjelmistojen kehitysprojekteja ohjaavat samat tekijät kuin muitakin projekteja (Lindberg 2003)

Erilaisten ohjelmistokehitysmenetelmien kehittymiseen ja erilaistumiseen ovat vaikuttaneet asiakkaiden erilaiset tarpeet. *Asiakkailla* tässä työssä tarkoitetaan ohjelmistoprojektien rahoittajia, ohjelmistojen loppukäyttäjiä sekä räätälöityjä ohjelmistoja toteuttavien ohjelmistoyritysten asiakasyrityksiä. Toisissa projekteissa nopea aikataulu on keskeistä asiakkaille, toisissa laadun on oltava täysin virheetöntä hinnan ja aikataulun ollessa toissijaisia asioita (Lindberg 2003). Esimerkki nopeutta ja kustannustehokkuutta vaativasta ohjelmistoprojektista voi olla kevyt mainoskampanjan osana toteutettava selainkäyttöinen peli, toisessa äärilaidassa hyvin luotettava mutta kalliisti toteutettu sairaalassa käytettävä lääkeannostelulaitteen ohjelmisto.

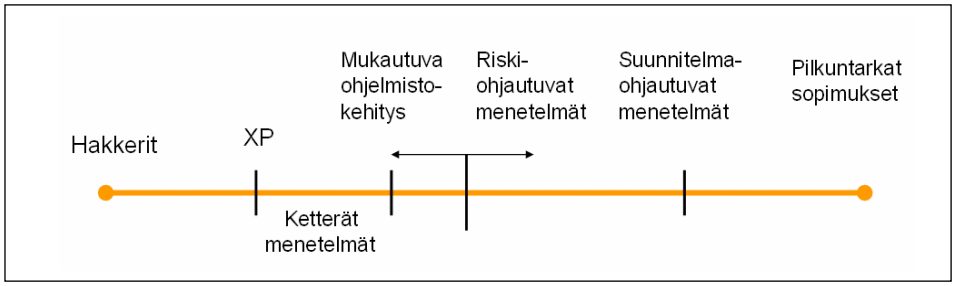
Ohjelmiston kehitysmenetelmän valinta tapahtuu sovelluksen käyttötarkoituksen, laajuuden, tavoitellun laadun sekä käytettävissä olevan ajan ja taloudellisten resurssien mukaan. Valintaan liittyy luonnollisesti myös toteuttajayrityksen sekä järjestelmän tilanneen asiakkaan tietoinen valinta ohjelmistokehitysprojektin painotuksesta - asiaanhan voidaan aktiivisesti vaikuttaa.

2.2 Ohjelmistokehitysmenetelmät yleisesti

Erilaisten ohjelmistokehitysmenetelmien kannattajat voidaan jakaa karkeasti kahteen koulukuntaan niiden tuottamien dokumentaation määrän perusteella. Ensimmäinen näistä kahdesta koulukunnasta kannattaa suunnitelmaohjautuvia (rakenteellisia) malleja (plan-driven methods, structural methods), joissa tehtävät, merkkipaalut, määrittelyt ja arkkitehtuurisuunnitelmat suunnitellaan sekä dokumentoidaan mahdollisimman tarkasti. Ohjelmiston kehitys nähdään elinkaarena, joka alkaa tarpeiden määrittelystä ja päättyy

ylläpitoon. Kaikkia tähän joukkoon kuuluvia kehitysmenetelmiä kutsutaan tässä työssä *suunnitelmaohjautuviksi ohjelmistokehitysmenetelmiksi.*

Toinen koulukunta kannattaa menetelmiä, joissa suunnittelu perustuu tarkan dokumentaation sijasta ihmisten välisellä keskustelulla tapahtuvaan tiedonhankintaan. Tähän menetelmään kuuluvia menetelmiä kutsutaan *ketteriksi ohjelmistokehitysmenetelmiksi* (eng. agile methods). Kuvassa 3 esitetään graafisesti eri ohjelmistokehitysmenetelmien suunnitelmallisuusasteita, jonka perusteella voidaan lajitella menetelmiä eri ryhmiin. Suunnitelmallisuusaste kuvaa sen kirjallisen esi- tai jälkityön määrää, jota ohjelmiston toteuttamiseksi tehdään (Boehm 2002).



Kuva 3: Eri ohjelmistokehitysmenetelmien suunnitelmallisuus (Boehm 2002)

Äärimmäisenä vasemmalla kuvassa 3 ovat hakkerit, jotka eivät käytä ohjelmistojen toteuttamisessa kirjallisia suunnitelmia, vaan ryhtyvät suoraan ohjelmoimaan. Keskiviivan oikealla puolella ovat suunnitelmiin pohjautuvat menetelmät, päätyen äärimmäisenä oikealla olevaan pilkuntarkkaan sopimuksessa ohjelmiston ominaisuudet määrittelevään suunnitelmaohjautuvaan ohjelmistokehitysmenetelmään. Näiden läheisyyteen kuviossa asettuvat myös muut uudemmat ohjelmistokehitysmenetelmät, joita ovat riskiohjautuvat menetelmät (Risk-driven software development) sekä mukautuva ohjelmistokehitys (Adaptive Software Development).

Seuraavissa kappaleissa kuvataan molempia ohjelmistokehityssuuntauksia ja niiden tyypillisimpiä menetelmiä tarkemmin. Kappaleessa 2.5 vertaillaan menetelmiä toisiinsa.

2.3 Suunnitelmaohjautuvat ohjelmistokehitysmenetelmät

Kaikkien suunnitelmaohjautuvien ohjelmistokehitysmenetelmien voidaan sanoa perustuvan ohjelmiston toteutuksen näkemiseen kokoelmana lineaarisesti eteneviä vaiheita, jotka seuraavat toinen toistaan. Yleensä nämä vaiheet ovat kaikissa suunnitelmaohjautuvissa malleissa perusajatukseltaan samoja, mutta niiden esiintymistiheys ja toteutustapa vaihtelevat. Haikala (2002) esittelee vaiheet seuraavasti:

- Määrittely
- Suunnittelu
- Ohjelmointi (toteutus)
- Testaus
- Käyttöönotto ja ylläpito

Nämä vaiheet, joita voidaan kutsua ohjelmiston elinkaaren vaihejaoksi, etenevät järjestyksessä ensimmäisestä viimeiseen, ja lopputuloksena saadaan asiakkaan tarpeita ja määrittelyjä vastaava ohjelmisto. Yleensä kaikkiin edellä kuvattuihin vaiheisiin liittyy myös dokumentaatio, joka valmistuu vaiheen päättyessä. Kunkin vaiheen dokumentaatio ohjaa osaltaan ohjelmistokehitystä eteenpäin seuraavaan vaiheeseen (Van Vliet 2000).

Suunnitelmaohjautuvissa menetelmissä on huomattava, että asiakas osallistuu ohjelmiston toteutustyöhön lähinnä määrittely- ja suunnitteluvaiheessa (Huhtamäki 2005). Muu osa prosessista voi tapahtua toteuttajayrityksen sisällä. Prosessin edetessä puhtaimmillaan asiakas tutustuu toteutettuun ohjelmistoon määrittelyjen jälkeen seuraavan kerran vasta käyttöönoton yhteydessä.

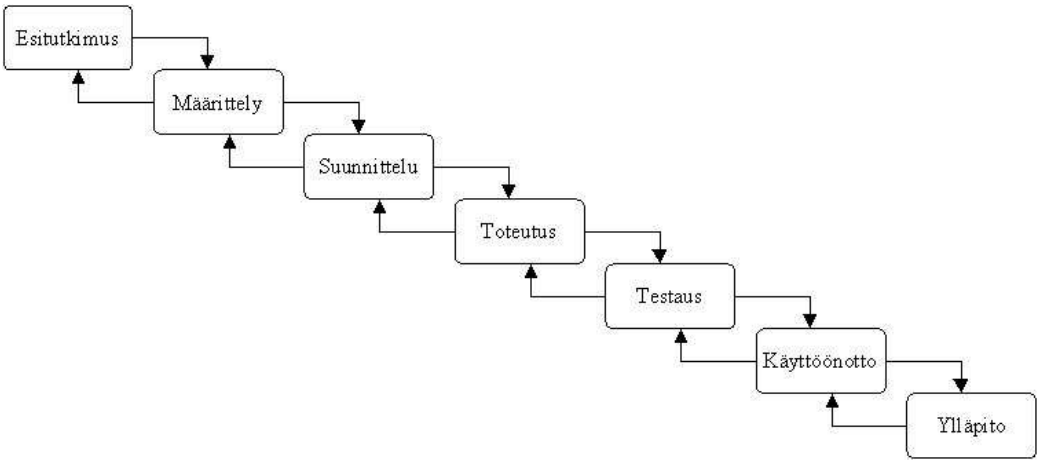
Yleisesti suunnitelmaohjautuvia menetelmiä ja niiden dokumentaatiota voidaan verrata esimerkiksi talon rakentamiseen: ensin rakennetaan perustukset, myöhemmin pystytetään seinät ja niin edelleen. Eri vaiheiden valmistumiset suunnitellaan yleensä jo projektin alussa, ja määritellään merkkipaaluiksi, joille asetetaan myös tavoiteltu valmistumisajankohta. Yleensä näiden erilaisten merkkipaalujen saavuttamiseen niin rakennustyössä kuin ohjelmistokehityksessäkin liittyy myös laskutus: kun määritelty merkkipaalu on saavutettu, voi toimittaja laskuttaa asiakasta. (Van Vliet 2000)

Seuraavassa esitellään kolme yleisesti tunnettua suunnitelmaohjautuvaa ohjelmistokehitysmallia, jotka noudattavat näitä viittä vaihetta kukin omalla tavallaan.

2.3.1 Vesiputousmalli

Vesiputousmalli (eng. waterfall model) on erittäin käytetty, perinteinen suunnitelmaohjautuva ohjelmistokehitysmenetelmä, jonka Winston W. Royce esitteli nykyisessä muodossaan vuonna 1970 (Haikala 2002; Wikipedia 2006). Vesiputousmalli toimii ideologialtaan pohjana useille muille malleille, jotka yleensä ovat saaneet alkunsa vesiputousmallin käytännön sovelluksista.

Vesiputousmalli on esitetty kuvassa 4. Sana ”vesiputous” mallin nimessä kuvaa sitä, että suunnittelu on jaettu osiin, joista jokainen tulee suorittaa loppuun ennen siirtymistä seuraavaan vaiheeseen. Vesi ei siis virtaa koskaan ylöspäin – ainakaan periaatteessa. Tästä periaatteesta aiheutuvat vesiputousmallin hyvät ja huonot puolet (Kangas 2002). Käytännössä reaalimaailmassa on kuitenkin joissakin tapauksissa tarpeen palata takaisin edelliseen vaiheeseen ja muuttaa sen toteutusta. Tämän vuoksi myös kuvaan 4 on piirretty nuolet kuvaamaan tätä edelliseen vaiheeseen palaamista.



Kuva 4: Vesiputousmalli (Hiltunen 2004)

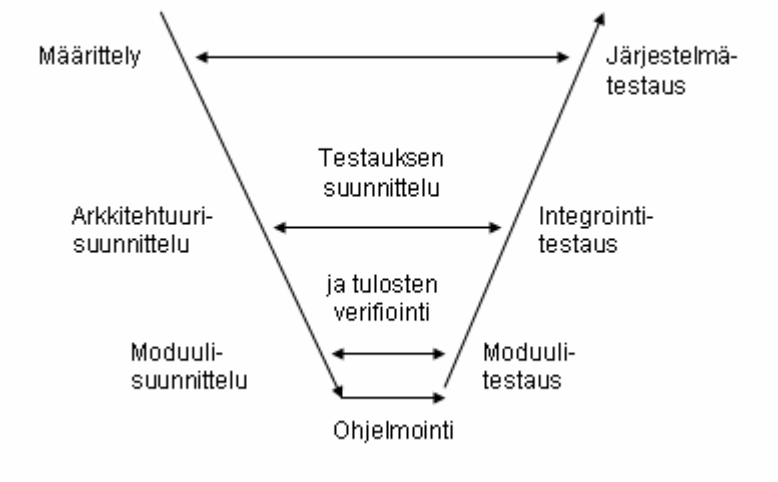
Vesiputousmallin *esitutkimusvaiheen* tehtävä on asettaa toteutettavalle järjestelmälle yleiset vaatimukset. Näitä vaatimuksia kutsutaan usein myös asiakasvaatimuksiksi, sillä ne määrittelevät asiakkaan tarpeen toteutettavalle järjestelmälle, mutta eivät ota vielä kantaa siihen, millainen ja miten toteutettu järjestelmä toteuttaa nämä tarpeet. Esitutkimus voidaan katsoa olevan myös osa määrittelyvaihetta, koska asiakastarpeita analysoidaan ja tarkennetaan yleensä käytännössä koko määrittelyvaiheen ajan (Haikala 2002). Esitutkimukseen liittyy usein myös alustava projektisuunnitelma.

Määrittelyvaiheessa kerätään ohjelmistolle asetetut toiminnalliset ja tekniset vaatimukset sekä ei-toiminnalliset vaatimukset ja rajoitukset. Ei-toiminnallisiksi vaatimuksiksi kutsutaan esimerkiksi suoritustehoon ja vasteaikaan sekä käytettävyyteen liittyviä vaatimuksia. Tästä vaiheesta tuotetaan yleensä vaatimusmäärittelyksi tai toiminnalliseksi määrittelyksi kutsuttu dokumentti. Määrittelyvaiheessa tarkennetaan myös esitutkimusvaiheen projektisuunnitelmaa saadun tarkemman tiedon perusteella. (Haikala 2002, Lindberg 2003)

Suunnitteluvaihe vastaa kysymykseen ”Miten ohjelmisto tulisi rakentaa?”. Ohjelmisto suunnitellaan teknisesti, mutta ei varsinaisesti ohjelmoida vielä mitään. Vaiheen lopputuloksena on yksi tai useampia teknisiä dokumentteja, joissa määritellään esimerkiksi ohjelmiston tekninen arkkitehtuuri, pysyvät tiedot, talletusratkaisut sekä käyttöliittymät. Usein suunnitteluvaihe voidaan jakaa kahteen pienempään kokonaisuuteen: arkkitehtuurisuunnitteluun ja moduulisuunnitteluun. Arkkitehtuurisuunnittelussa järjestelmä pilkotaan erilaisiin pienempiin osakokonaisuuksiin, moduuleihin (koko tyypillisesti alle 1000 koodiriviä). Moduulisuunnitteluvaiheessa suunnitellaan sitten eri moduuleiden varsinainen sisältö. (Lindberg 2003, Haikala 2002)

Toteutusvaiheessa (ohjelmointivaiheessa) kirjoitetaan varsinainen ohjelman lähdekoodi. Vaiheen tuloksena on varsinainen ohjelma sekä koodiin liittyvä dokumentaatio. (Lindberg 2003)

Testausvaiheessa on tarkoitus löytää ohjelmistosta virheitä. Ohjelmistojen testaus toteutetaan usein ns. V-mallin mukaisesti. V-malli on esitetty graafisesti kuvassa 5. V-mallin testaus on jaettu siinä kolmeen osaan: moduulitestaukseen (yksittäisten moduulien vianetsintä), integraatiotestaukseen (moduulien yhteistoiminnan vianetsintä) ja järjestelmätestaukseen (koko järjestelmän toiminta + suorituskyky). Viimeinen vaihe, järjestelmätestaus, suunnitellaan yleensä jo määrittelyvaiheessa ja toteutetaan vertaamalla valmista järjestelmää määrittelyyn. (Haikala 2002)



Kuva 5: Ohjelmistojen testauksen V-malli (OAMK 2006)

Käyttöönotossa valmis ohjelmisto toimitetaan asiakkaan käyttöön. *Ylläpidolla* tarkoitetaan valmiin järjestelmän käytössä havaittujen virheiden korjaamista, ohjelman

laajentamista ja muuttamista (Haikala 2002). Ylläpitoon liittyy usein myös erilaisia ylläpitosopimuksia ja asiakaspalvelun organisointitehtäviä.

Vesiputousmallin keskeinen ominaispiirre, sen heikkous ja vahvuus, keskittyy monilta osin prosessin alkupuoleen. Mallin ideologiaan kuuluu, että varsinaiseen toteutustyöhön, ohjelmointiin, halutaan käyttää niin vähän energiaa kuin mahdollista. Tähän pyritään mahdollisimman perusteellisella suunnittelulla: Asiakkaalta pyritään saamaan selville järjestelmän määrittelyt (vaatimusmäärittelydokumentti) täydellisinä ennen toteutustyön aloittamista. Todellisuudessa tähän tilanteeseen pääseminen on hyvin vaikeaa tai jopa mahdotonta. Siksi toteutuksen aikanakin on tarpeen varmistaa asiakkaalta aika-ajoin toteutuksen vastaavuutta todellisiin tarpeisiin (Van Vliet 2000).

Vesiputousmalli soveltuu sellaisiin tilanteisiin, joissa on hyvin tarkasti tiedossa, mitä toteutettavalta lopputuotteelta halutaan ja mihin sitä aiotaan käyttää. Todellisuudessa asetettujen määrittelyiden muuttumisriski on hyvin suuri, jonka vuoksi vesiputousmallin käyttäminen ideaalisena, ilman takaisinkytkentöjä edellisiin vaiheisiin, on käytännössä mahdollista erittäin harvoin. (Kangas 2003)

2.3.2 Prototyyppimalli

Prototyyppimalleilla tarkoitetaan yleisesti melkein mitä tahansa työskentelymallia, jossa jotakin ohjelmistotuotteen piirrettä tai ominaisuutta kokeillaan luonnoksella, prototyypillä, ennen varsinaisen tuotteen rakentamista (Haikala 2002). Useimmiten prototyyppien tarve liittyy asiakkaan vaatimusmäärittelyn vaikeuteen: nykyiseen tilanteeseen ei olla tyytyväisiä, vaan uuden ohjelmistotuotteen avulla halutaan päästä entistä parempaan tilanteeseen. Usein asiakkaan on kuitenkin hyvin vaikea tarkasti määritellä, millainen tämä parempi tilanne tarkalleen olisi. Tällöin esimerkkien antaminen prototyyppien avulla voi auttaa (Van Vliet 2000).

Kehitettävien ohjelmistojen prototyypit pyritään toteuttamaan mahdollisimman edullisesti ja nopeasti. Tämä voi tarkoittaa prototyypin eroamista varsinaisesta järjestelmästä kahdella tavalla esimerkiksi toteutusvälineiden (ohjelmointikieli) ja toteuttavien toimintojen osalta (Van Vliet 2000).

Toteutusvälineiden osalta on kyse yleensä varsinaisen ohjelmointikielen sijasta käytettävästä jostakin toisesta, prototyypin kannalta tehokkaammasta ohjelmointikielestä. Ohjelmistotuotteiden prototyypit liittyvät usein käyttöliittymiin, joissa niiden käyttö on osoittautunut erityisen hyödylliseksi (Haikala 2002). Prototyyppejä voidaan nykyään toteuttaa tehokkaasti erilaisilla RAD-työkaluilla (Rapid Application Development) (Lindberg 2003). Tällöin voidaan myös tehdä useita

iteratiivisia versioita prototyypistä ja parantaa prototyypin laatua tarpeellisilta osin jokaisessa versiossa kunnes asiakas on siihen tyytyväinen.

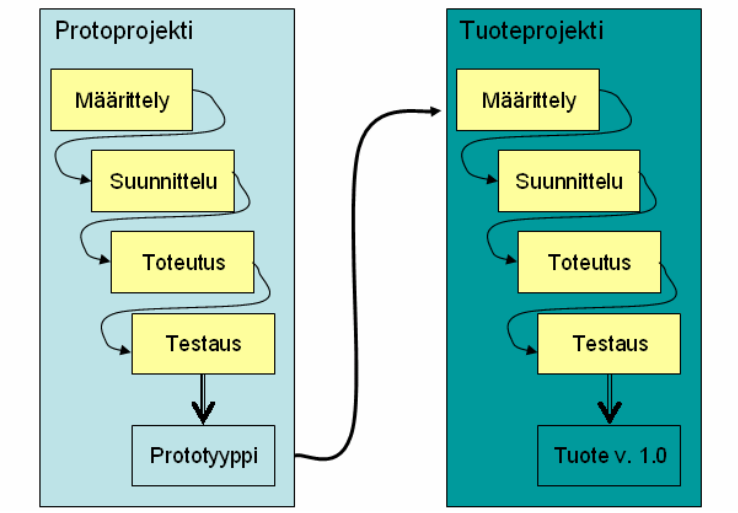
Toteutettavien toimintojen osalta eroavaisuudet todelliseen järjestelmään voivat koskea esimerkiksi tietojen hakua ja tallennusta: prototyypissä ei välttämättä tarvitse olla mitään sovelluslogiikkaa tietojen hakuun ja tallennukseen, jos mallinnuksen kohteena ovat ainoastaan käyttöliittymän erikoispiirteet (Lindberg 2003). Jos sovelluslogiikkaa toteutetaan jo prototyyppiin, voi prototyyppi erota varsinaisesta järjestelmästä siten, että jotkin järjestelmän tehokkuusmääreet kuten nopeus ja vikasietoisuus eivät ole samanlaiset kuin varsinaisessa järjestelmässä (Van Vliet 2000).

Prototyypin toteuttamisen jälkeen on tarjolla kaksi päävaihtoehtoa, joiden mukaan ohjelmistokehitysprojekti voi edetä (Haikala 2002):

a) Prototyypin valmistuttua prototyyppiä käytetään varsinaisen järjestelmän määrittelyyn. Varsinainen järjestelmä toteutetaan alusta alkaen uudelleen ja prototyyppi hylätään.

b) Prototyyppi kehitetään valmiiksi järjestelmäksi

Kuvassa 6 esitetään esimerkki vaihtoehdosta a. Tässä tapauksessa molempien, sekä prototyypin että varsinaisen järjestelmän suunnittelu ja toteutus etenee samaan tapaan; varsinaisen järjestelmän määrittelyssä on kuitenkin käytettävissä apuna prototyyppi, joten lähtökohta on hyvin toisenlainen.



Kuva 6: Prototyyppimallin eteneminen (Haikala 2002)

Usein kuitenkin vaihtoehto b, jossa prototyyppi toimii varsinaisen järjestelmän alustana, on houkuttelevampi. Tällöin valmista työtä ei tarvitse heittää hukkaan. Molemmilla on

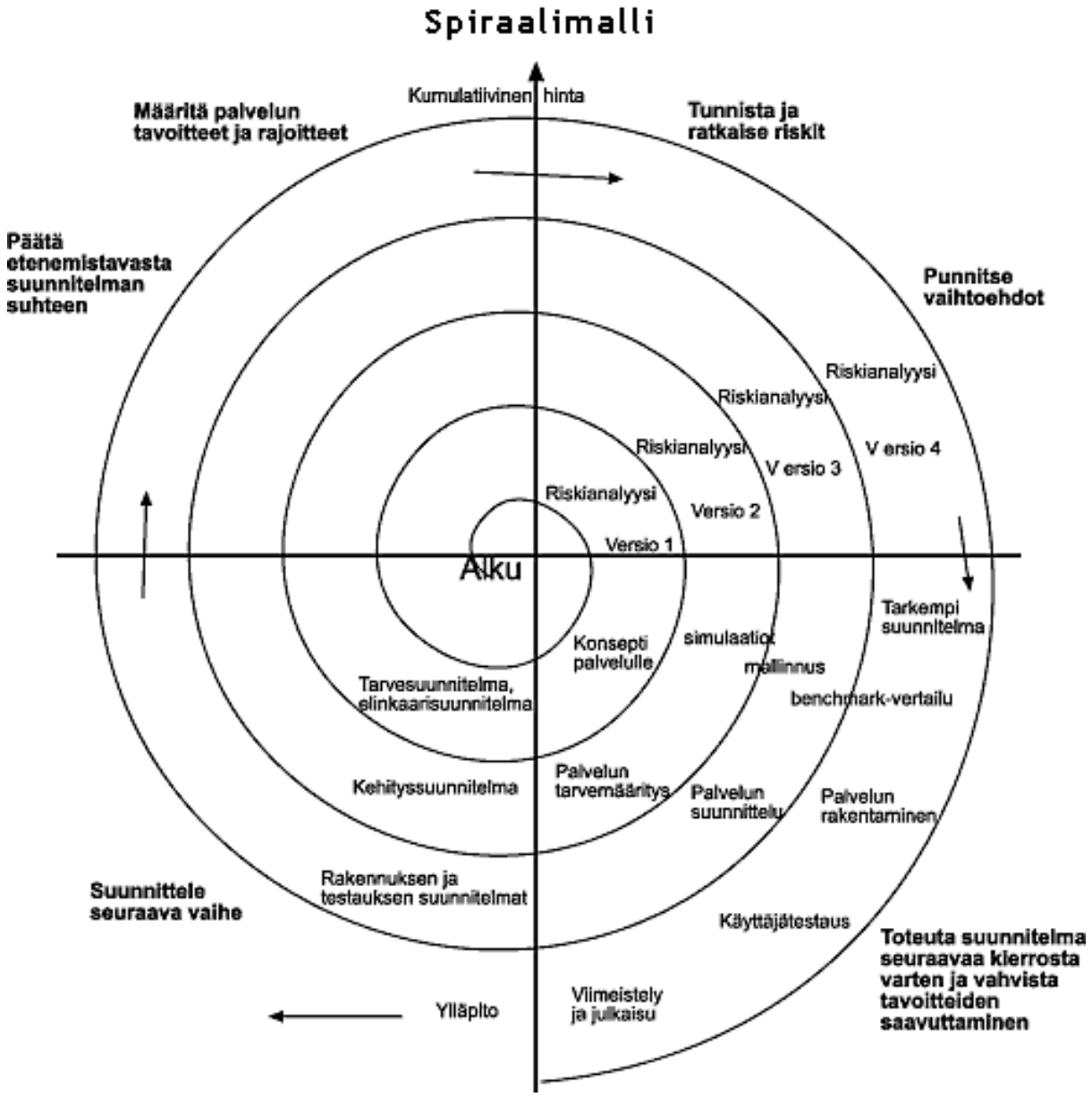
kuitenkin omat hyvät ja huonot puolensa. Esimerkiksi ohjelmiston ylläpitokustannukset voivat kohota merkittävästi, jos nopeasti koottua ja huonolla ohjelmointitavalla ohjelmoitua prototyyppiä käytetään varsinaisen järjestelmän runkona (Van Vliet 2000).

Van Vliet (2000) esittää prototyyppimallin hyviksi käyttötarkoituksiksi tilanteet, joissa asiakkaan antamat määrittelyt ovat epäselviä sekä sellaiset järjestelmät, joissa käyttöliittymillä on suuri rooli. Asiakkaan ja järjestelmän toteuttajien on syytä olla tietoisia prototyyppimallin käytön mahdollisuuksista ja riskeistä. Käyttäjille on syytä korostaa, että prototyyppi ei ole varsinainen järjestelmä, eikä prototyypin valmistuminen tarkoita koko järjestelmän valmistumista. Prototyyppien toteutuksen on myös oltava suunnitelmallista, joten jonkin ohjelmistokehitysmallin käyttäminen myös prototyypin kehityksessä on järkevää.

2.3.3 Spiraalimalli

Vuonna 1988 Boehmin esittämä spiraalimalli perustuu voimakkaasti vesiputousmalliin. Malli yhdistää kuitenkin myös muita ohjelmiston suunnittelumenetelmiä, kuten evoluutio- (evolutionary development) ja ohjelmoi-korjaa (eng. code-fix) -menetelmiä. Spiraalimallia voidaan kutsua myös riskiohjatuksi prosessiksi. Kuvassa 7 esitetään Hintikan ja Mielosen (1998) näkemys spiraalimallista. Mallista on olemassa myös monia muita muunnelmia ja yksinkertaistuksia (Kangas 2003).

Keskeisenä erona vesiputousmalliin verrattuna spiraalimallia eivät ohjaa dokumentit, vaan riskit. Riskien hallinta on spiraalimallin keskeinen lisäominaisuus aiempiin prosessimalleihin verrattuna (Halonen 2001).



Kuva 7: Ohjelmistonkehityksen spiraalimalli (Hintikka & Mielonen 1998)

Spiraalimallissa edetään iteraatioiden avulla lopulliseen ratkaisuun toistuvien *s*yklien avulla (Hintikka & Mielonen 1998). Jokainen sykli sisältää useita vaiheita, jotka toistuvat kaikissa sykleissä. Syklien määrää ei ole rajoitettu. Vaiheet ovat *suunnittelu, riskianalyysi, kehitys ja asiakasarviointi.* Kuvan 7 kuvaamassa mallissa spiraalimalli lähtee etenemään kuvan keskeltä päätyen kehän ulkoreunaan. Etäisyys keskipisteestä kuvaa projektin kokonaiskustannuksia ja sen kokonaisvaihetta – mitä kauempana keskipisteestä ollaan, sitä valmiimpi on lopputulos (Halonen 2001).

Spiraalimallissa on huomionarvoista muihin edellä esiteltyihin menetelmiin verrattuna se, että sen käyttöä voidaan jatkaa koko ohjelmiston elinkaaren ajan. Tällöin voidaan ajatella, että ensimmäinen kierros kuvan 7 spiraalilla tarkoittaa vain alkuperäisen

ohjelmiston suunniteltua toteutusta, ja seuraavat kierrokset ovat ohjelmiston kehitystyötä ja ylläpitoa. Spiraalimallin huonoja puolia ovat sen sopeutuminen talousnäkökohtiin: kehityksen eteneminen sykleissä voi aiheuttaa ongelmia sopimusneuvotteluissa ja sopimuksien määrittelyissä (Pressman 2005).

2.4 Ketterät ohjelmistokehitysmenetelmät

1990-luvun loppupuolella useat uudet, ohjelmistokehitysmenetelmät alkoivat saada ohjelmistotekniikan alalla huomiota. Nämä menetelmät olivat yhdistelmiä vanhoista, uusista sekä muunnelluista vanhoista ideoista. Yhteistä kaikille näille menetelmille oli muun muassa läheinen yhteistyö ohjelmoijien ja liiketoiminnan ammattilaisten kesken, kasvokkain tapahtuva kommunikointi, säännöllinen uuden vastineen tuottaminen asiakkaan investoinneille sekä ohjelmointikäytännöt ja tiimit, jotka kykenevät vastaamaan muuttuviin vaatimuksiin (Kosonen 2005).

Vuonna 2001 USA:ssa pidettiin työpaja, jossa monet näiden eri menetelmien kehittäjistä ja käyttäjistä kokoontuivat miettimään, mikä näissä menetelmissä oli yhteistä. Tällöin otettiin käyttöön termi 'ketterä' (englanniksi ’agile’), joka kuvaa tiiviisti näille menetelmille yhteisiä piirteitä. Tässä tapahtumassa muotoiltiin ja julkaistiin myös ketterän sovelluskehityksen manifesti (Agile Software Development Manifesto), joka määrittelee yleiset arvot ja periaatteet ketterälle sovelluskehitykselle. (Beck ym. 2001; Kosonen 2005; Abrahamsson 2002)

Ohjelmistokehitykselle asetetaan nykyään kovia paineita kustannusten ja aikataulujen osalta. Myös järjestelmien määrittelyt muuttuvat jatkuvasti. Nämä ilmiöt koskevat erityisesti selainpohjaisia ohjelmistoja. Tämän vuoksi kiinnostus nopeampia ja joustavia ohjelmistokehitysmenetelmiä kohtaan on jatkuvasti kasvanut, vaikka monien mielestä käyttöönotto voikin olla haastavaa. Ketterien sovelluskehitysmenetelmien avulla sovelluskehittäjät voivat kasvattaa tuottavuutta säilyttäen samalla ohjelmistojen laadun ja muokattavuuden korkeana (Maurer 2002).

Erilaisia ketteriä ohjelmistokehitysmenetelmiä on määritelty ja esitelty kymmenkunta erilaista. Keskeisimpiä yhteisiä piirteitä niissä ovat iteratiivinen ohjelmistokehitys sekä asiakkaan tiivis mukanaolo projektissa. Eroja ovat esimerkiksi keskittyminen ohjelmiston elinkaaren eri osiin tai mallin käytännön soveltamisen ohjeistuksen tarkkuus. Osa malleista on hyvin abstrakteja, kun taas osa koostuu konkreettisista ohjelmointityön käytännöistä. Ketterien menetelmien käyttöönottoa ja sopivuuden arviointia yrityksissä vaikeuttaa se, että menetelmistä on toistaiseksi melko vaikeaa löytää puolueetonta vertailutietoa. (Liedenpohja 2006)

Yleisesti tunnetuimpana pidetty ketterien menetelmien ohjelmistokehitysmenetelmä on Extreme Programming, XP. Menetelmän määritteli vuonna 1999 Kent Beck (Beck, 2000). XP-menetelmää on monissa yhteyksissä pidetty ketterien ohjelmistokehitysmenetelmien uranuurtajana ja ketterien menetelmien kehityksen alkuna. Muita ketteriä ohjelmistokehitysmenetelmiä ovat esimerkiksi Crystal Methods, Scrum, Feature Driven Development (FDD), Lean Development sekä Adaptive Software Development (ASD) (Abrahamsson 2002). Tässä työssä keskitytään erityisesti XP -menetelmän laajempaan esittelyyn sen keskeisen aseman vuoksi. XP-menetelmällä on myös selkeitä yhteyksiä FNS:n ohjelmistokehitysmenetelmään. Muut ketterät menetelmät esitellään hieman lyhyemmin kappaleessa 2.4.3.

2.4.1 Yleiset periaatteet

Ketterän ohjelmistokehityksen yleiset periaatteet ja perusarvot, joita kaikki yksittäiset menetelmät noudattavat, on esitetty ketterän ohjelmistokehityksen manifestissa (Beck ym. 2001), joka esitetään kokonaisuudessaan alkuperäismuodossa liitteessä 1. Manifesti alkaa ketterän ohjelmistokehityksen perusarvoilla, jotka suomennettuina voidaan esittää seuraavasti:

Yksilöt ja vuorovaikutus ovat tärkeämpiä kuin prosessit ja työkalut

Toimiva ohjelmisto on tärkeämpää kuin kattava dokumentaatio

Yhteistyö asiakkaan kanssa on tärkeämpää kuin sopimusneuvottelu

Muutokseen vastaaminen on tärkeämpää kuin suunnitelman noudattaminen

Nämä arvot kertovat, että ketterässä ohjelmistokehityksessä arvojen vasemman puolen kohdat ovat tärkeämpiä kuin oikean puolen. On syytä kuitenkin huomata, että myös oikeanpuoleisilla on arvoa – mutta ei niin paljon kuin vasemmanpuoleisilla kohdilla (Kosonen 2005). Oikeanpuoleisten kohtien voidaan myös ajatella olevan rinnastuksia alan yleiseen ajattelutapaan; niiden voidaan nähdä väittävän, että ohjelmistotuotannossa arvostetaan työkaluja, jäykkiä työtapoja, dokumenttien paljoutta, tiukkoja sopimuspykäliä ja tarkkaa suunnitelman mukaista etenemistä (Wirgentius 2003).

Perusarvojen lisäksi manifestissä (Beck ym. 2001) kuvataan 12 periaatetta, joita ketterä ohjelmistokehitys noudattaa. Suomennettuna ne kuuluvat seuraavasti:

1. Tärkeintä on täyttää asiakkaan vaatimukset jatkuvilla ja riittävän aikaisilla ohjelmistotoimituksilla.

2. Muuttuvat vaatimusmäärittelyt hyväksytään ja otetaan vastaan tervetulleina, myös kehityksen loppuvaiheessa. Ketterät menetelmät valjastavat muutoksen asiakkaan kilpailueduksi.

3. Toimivia ohjelmaversioita toimitetaan säännöllisesti, muutaman viikon tai kuukauden välein, mieluummin useammin kuin harvemmin.

4. Liiketoiminnan ammattilaisten ja ohjelmistokehittäjien on työskenneltävä yhdessä päivittäin koko projektin ajan.

5. Projektit rakennetaan motivoituneiden yksilöiden ympärille. Heille annetaan ympäristö ja tuki jota he tarvitsevat, sekä luotetaan siihen, että he saavat työn tehtyä.

6. Kaikkein tehokkain tapa tiedonvälityksessä kehitystiimille ja kehitystiimin sisällä on kasvokkain tapahtuva keskustelu.

7. Toimiva ohjelmisto on ensisijainen työn edistymisen mitta.

8. Ketterät menetelmät suosivat kestävää kehitystä. Rahoittajien, kehittäjien ja käyttäjien tulisi kyetä pitämään jatkuvasti yllä tasainen työtahti.

9. Jatkuva huomion kiinnittäminen tekniseen erinomaisuuteen, hyvään rakenteeseen sekä suunnitteluun lisää ketteryyttä.

10. Yksinkertaisuus – tekemättömän työn maksimoinnin taito – on olennaista.

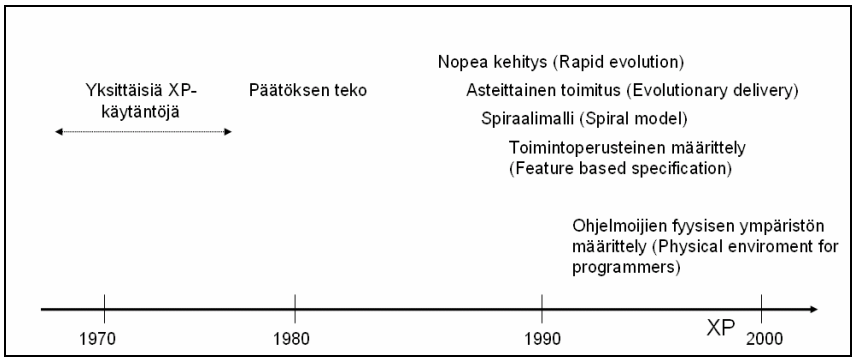
11. Parhaat rakenteet, vaatimukset ja suunnitelmat nousevat itseorganisoituvista tiimeistä.

12. Tasaisin väliajoin tiimi arvioi, miten voisi tulla entistä tehokkaammaksi, ja kehittää toimintaansa sen mukaisesti.

Ketterän kehityksen menetelmän erottamiseksi perinteisistä Abrahamsson (2002) koostaa nämä edellä kuvatut ketterän kehityksen arvot ja periaatteet seuraavanlaisiksi tunnusmerkeiksi: Ketterä ohjelmistokehitys on *inkrementaalista* (pienet versiojulkaisut tiheään tahtiin), *yhteistyössä tapahtuvaa* (asiakas ja kehittäjät toimivat jatkuvasti yhdessä ja pitävät tiiviisti yhteyttä), *suoraviivaista* (menetelmä itsessään on helppo oppia ja se on hyvin dokumentoitu) ja *sopeutuvaa* (on mahdollista tehdä viime hetken muutoksia).

2.4.2 Extreme programming (XP)

Extreme Programming -menetelmän (XP) kehittäjä Kent Beck aloittaa kirjansa "Extreme Programming Explained" (2004) esittelemällä tämän menetelmän sosiaaliseksi muutokseksi. Hän kertoo XP:n tarkoittavan vanhoista, aikoinaan hyvistä tavoista luopumista ja vastaanottavuutta uudelle, jossa on oltava avoin sille, mitä on kykenevä tekemään ja mitä muut ovat kykeneviä tekemään. Käytännössä Extreme Programming on kokoelma ideoita ja käytäntöjä aikaisemmista, hyviksi havaituista menetelmistä (Abrahamsson 2002). Ensimmäiset taustalla olevat ideat ovat lähtöisin jo 1970-luvun lopulta, mutta keskeisimmiltä osiltaan Beck ja muutamat muut henkilöt kehittivät XP-menetelmän 1990-luvun puolivälissä (Kalermo & Rissanen 2002). Kehityspolkua ja taustalla vaikuttavia menetelmiä esitellään kuvassa 8. Kent Beck teoretisoi ja nimesi tämän yhdistelmän "Extreme Programming" -menetelmäksi käyttämällä taustalla omassa työssään tehokkaiksi havaitsemiaan menetelmiä. Termi "Extreme", suomeksi "äärimmäisyys (äärimmäinen aste)" tai "ääripää" (Netmot-sanakirjasto 2006), menetelmän nimessä kuvaa näiden käytäntöjen viemistä äärimmäisille tasoille (Abrahamsson 2002).



Kuva 8: Extreme Programming -ideologian juuret (Abrahamsson 2002)

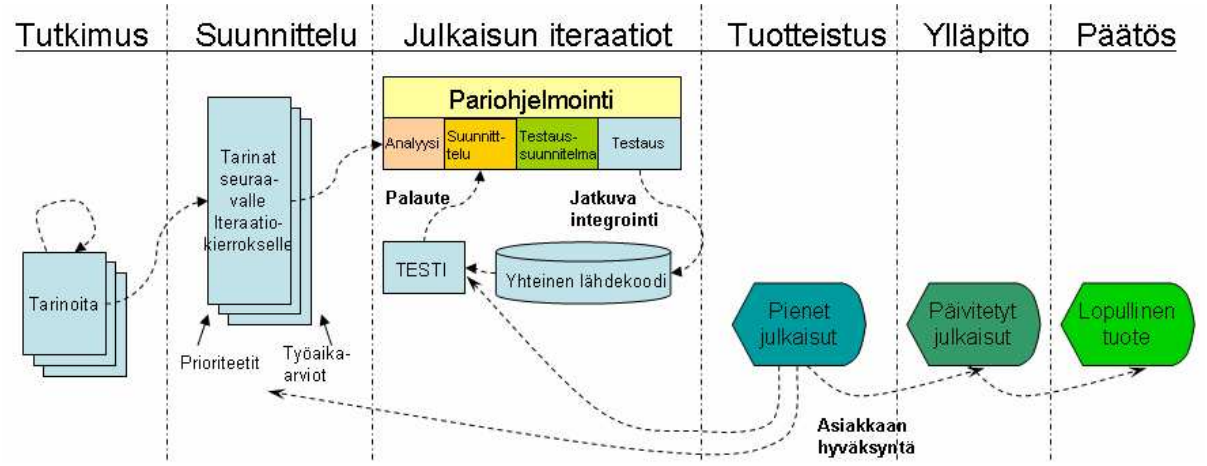
Extreme Programming -mallin vaiheet

Extreme Programming -malli etenee seuraavien kuuden päävaiheen mukaisesti (Abrahamsson 2002):

1. Tutkimus (Exploration phase)
2. Suunnittelu (Planning phase)
3. Julkaisun iteraatiot (Iterations to release phase)
4. Tuotteistus (Productionizing phase)
5. Ylläpito (Maintenance phase)

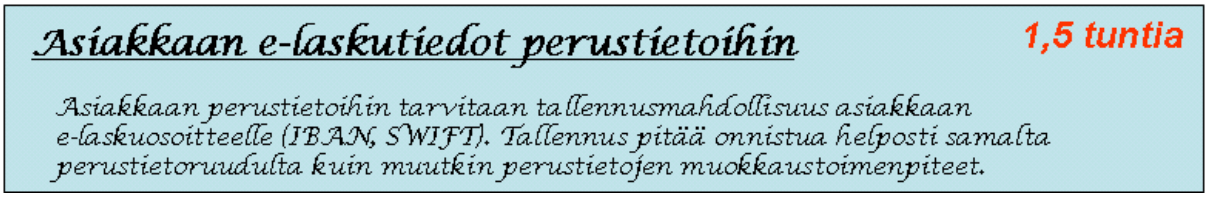
6. Päätös (Death phase)

Nämä vaiheet etenevät pääasiassa numerojärjestyksessä ensimmäisestä viimeiseen, mutta toisaalta voidaan palata myös taaksepäin edellisiin vaiheisiin (Kuva 9). Seuraavissa kappaleissa esitellään vaiheet Abrahamssonin (2002) kuvauksen mukaisesti.



Kuva 9: XP -prosessin elinkaari (Abrahamsson 2002)

Tutkimusvaiheessa asiakkaan edustajan tehtävänä on kirjoittaa lyhyitä tarinoita korteille, jotka kuvaavat ohjelmiston keskeisimpiä ominaisuuksia, jotka he haluavat ohjelmiston ensimmäiseen versioon. Jokainen tällainen *tarinakortti* (story card) kuvaa yhtä järjestelmän ominaisuutta (kuva 10). Tutkimusvaiheen aikana ohjelmoijat tutustuvat tekniikkaan, jota he tulevat toteutuksessa käyttämään. Menetelmään tutustutaan rakentamalla prototyyppiä toteutettavasta järjestelmästä. Vaiheen kesto on muutamista viikoista muutamaan kuukauteen.



Kuva 10: Esimerkki tarinakortista, johon on kirjattu myös arvio toteutusajasta

Suunnitteluvaiheessa määritellään asiakkaan antamien tarinoiden (ominaisuuksien) tärkeysjärjestys ja ensimmäisen version sisältö. Ohjelmoijat arvioivat ominaisuuden toteuttamiseen tarvittavan työmäärän ja merkitsevät sen muistiin tarinakorttiin. Tämän jälkeen sovitaan ominaisuuksien toteutusaikataulusta tarkemmin. Suunnitteluvaihe kestää muutamia päiviä.

Julkaisun iteraatiot sisältävät useita pienempiä iteraatiokierroksia, jotka ovat osia suunnitteluvaiheessa sovitusta aikataulusta. Ensimmäinen iteraatiokierros sisältää järjestelmän perusarkkitehtuurin rakentamisen, ja seuraavat rakentavat hiljalleen

ohjelmiston muita ominaisuuksia. Iteraatioiden edetessä asiakas voi milloin tahansa lisätä uusia tarinakortteja (järjestelmän ominaisuuksia), muuttaa niiden järjestystä, pilkkoa yhden tarinakortin useampaan tai poistaa niitä. Asiakas päättää aina seuraavaan iteraatioon otettavat toiminnot järjestelemällä tarinakortteja eri kategorioihin. Kategorioita voivat olla esimerkiksi *"toteutettu huomenna"*, *"toteutettu seuraavassa versiossa"* ja *"toteutetaan myöhemmin"*. Tarinakorttien ryhmittelyä voidaan tehdä myös iteraatiokierroksen sisäisen vaiheen mukaan. Vaihetta voidaan hahmottaa esimerkiksi ryhmittelemällä paperisia kortteja konkreettisesti esimerkiksi ilmoitustaululle tehtyihin kategorioihin (kuva 11). Viimeisen iteraatiokierroksen lopussa järjestelmä on valmis tuotantoon. Jokaisessa vaiheessa suoritetaan myös asiakkaan laatimat testit.

Kuva 11: Esimerkki tarinakorttien ryhmittelystä taululle

Tuotteistusvaihe sisältää lisää testausta esimerkiksi suorituskykyyn liittyvissä asioissa. Myös tuotteistusvaihe sisältää useita iteraatiokierroksia (Abrahamsson 2002).

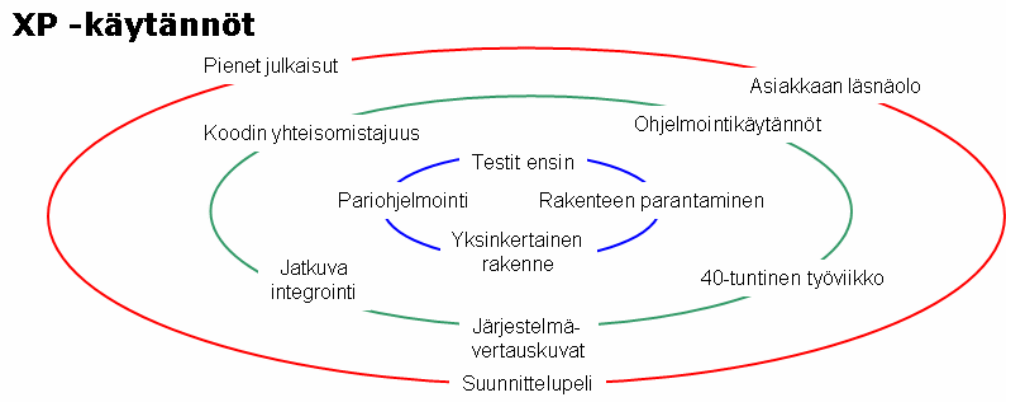
Ylläpitovaiheessa käyttäjät määrittelevät uusia ominaisuuksia, tarinakortteja, jotka ylläpitäjät toteuttavat. Myös käyttäjätukitoiminnot sisältyvät ylläpitovaiheeseen.

Päätösvaiheessa käyttäjillä ei enää ole uusia ominaisuuksia järjestelmän muuttamiseksi. Järjestelmästä kirjoitetaan dokumentaatio ja se todetaan valmiiksi. Jatkuvasti kehittyvän järjestelmän tapauksessa tämä voi tarkoittaa myös järjestelmän sulkemista ja lopettamista tilanteessa, jossa se ei enää täytä asiakkaan tarpeita tai sitä ei muuten haluta enää kehittää.

Perusarvot ja käytännöt

Extreme Programming -menetelmän kantavia perusasioita on ajatus kokonaisesta yhteisöstä, tiimistä, jossa jokainen osallistuja on keskeinen ja tärkeä osa tiimiä. Tämän määritelmän lisäksi Ronald E Jeffries (2001) kuvaa XP:n perusarvoiksi *yksinkertaisuuden*, *kommunikoinnin*, *palautteen* ja *rohkeuden*.

Samoin kuin ketterille menetelmille yleisesti, on XP-ohjelmoinnillekin määritelty sen keskeiset käytännöt. Ne voidaan muotoilla 12 käytännöksi, jotka on esitetty kuvassa 12 (Biddle et al. 2004).



Kuva 12: Extreme Programming -käytännöt (sovellettu lähteestä Jeffries 2001)

Pienet julkaisut (Small releases) tarkoittavat uusien ohjelmistoversioiden toimittamista asiakkaan käyttöön nopeaan tahtiin – jopa päivittäin tai ainakin kuukausittain. *Asiakkaan läsnäolo(On-Site customer)* kuvaa sitä, että asiakkaan edustajan on oltava järjestelmän toteuttajien käytettävissä kysymyksiä tai tarkennuksia varten (Liedenpohja 2006). *Suunnittelupeli (Planning game)* on tiivistä yhteistyötä ohjelmoijien ja asiakkaan välillä, jossa asiakas kuvaa pienillä tarinoilla ohjelmistoon tarvittavan ominaisuuden, ohjelmoijat arvioivat sen tarvitseman työmäärän, jonka jälkeen asiakas arvioi ominaisuuden kiireellisyyden ja mittakaavan (Abrahamsson 2002).

Koodin yhteisomistajuus (Collective code ownership) valtuuttaa kenet tahansa ohjelmoijista korjaamaan ja muuttamaan tarvittavia kohtia ohjelmistosta: näin laatu saadaan pidettyä kunnossa. *Ohjelmointikäytäntöjä (Code conventions)* tulee määritellä ja ohjelmoijien täytyy seurata niitä. *Jatkuvalla integroinnilla (Continuous integration)* varmistetaan ohjelmiston eri komponenttien yhteensopivuus ja aiemmin luotujen testien läpäisemisen varmistaminen missä tahansa ohjelmiston kehitysvaiheessa. (Abrahamsson 2002)

Järjestelmävertauskuva (System metaphor) tarkoittaa kehittäjien ja asiakkaiden yhteistä, työn aloituksessa sovittua vertauskuvallista tavoitetilaa johon kaikki pyrkivät. Sitä voidaan tarvittaessa muuttaa toimijoiden yhteisellä päätöksellä (Liedenpohja 2006). *40 -tuntinen työviikko (Forty hour week)* on listattu myös XP:n kantaviin perusajatuksiin, ja ylitöiden tekemistä kahtena peräkkäisenä viikkona ei sallita. Tämän tarkoituksena on varmistaa riittävä virkistäytyminen ja vapaa-aika ohjelmoijille (Liedenpohja 2006,

Abrahamsson 2002). Samaa, järkevää työtahtia tarkoittavaa ajatusta kutsutaan toisissa lähteissä myös nimellä säilytettävissä oleva tahti (Sustainable pace) (Jeffries 2001).

Testit ensin (Test first) -toimintatapa tarkoittaa sekä kehittäjien määrittelemien yksikkötestien että asiakkaan määrittelemien toiminnallisten testien tekemistä ennen toiminnon varsinaista toteutusta. Kun toiminto läpäisee testin, se täyttää tarkoituksensa ja on valmis. (Abrahamsson 2002). *Pariohjelmointi (Pair programming)* on työtapa, jossa kaksi ohjelmoijaa istuu yhdessä saman tietokoneen ääressä toisen kirjoittaessa koodia ja toisen seuratessa, miettiessä vaihtoehtoisia ratkaisutapoja ja valvoessa koodin laatua (Liedenpohja 2006).

Yksinkertainen rakenne (Simple design) vaatii ohjelmoijia toteuttamaan käsillä olevan ongelman ratkaisun yksinkertaisimmalla mahdollisella tavalla. Tarpeeton monimutkaisuus ja ylimääräinen koodi tulee poistaa välittömästi. Tulevaisuudessa mahdollisesti tarvittaviin ominaisuuksiin ei tule varautua nyt (Abrahamsson 2002, Liedenpohja 2006). *Rakenteen parantaminen (Refactoring)* tarkoittaa koodin jatkuvaa muokkaamista kaksinkertaisuuksien poistamiseksi ja yksinkertaisuuden lisäämiseksi.

2.4.3 Muut ketterät menetelmät

Keskeisimmän ketterän kehitysmenetelmän, XPn, lisäksi on olemassa muutamia muita yleisesti tunnettuja ketteriä menetelmiä, joista seuraavassa esitellään keskeisimpiä.

Crystal-metodologiaperhe

Crystal-metodologiat ovat Alistair Cockburnin ja Jim Highsmithin kehittämä menetelmä ketterien kehitysmenetelmien perhe. Se on kokoelma erilaisia menetelmiä, joista voidaan valita sopivin ja tarkoituksenmukaisin projektin tarpeiden mukaisesti. Menetelmän valintavaiheessa projekteja arvioidaan kahden tekijän perusteella: projektiryhmän koon ja projektin kriittisyyden mukaan. Ryhmän koko vaikuttaa kommunikointimenetelmiin, kriittisyys taas voidaan ryhmitellä erityyppisiin ryhmiin. Kriittisyysryhmät ovat:

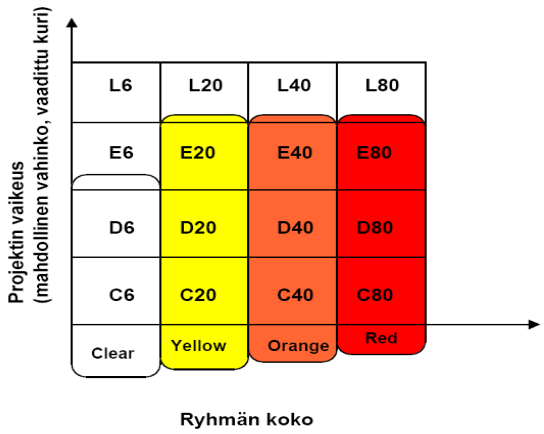
C = Mukavuus (Comfort): virhe aiheuttaa käsityötä, kun automaatio ei toimi

D = Kohtuullinen raha (Discretionary money): virhe aiheuttaa menetyksiä, jotka ovat myöhemmin korvattavissa

E = Kriittinen raha (Essential money): virheet ovat kohtuuttomia ja voivat aiheuttaa menetyksiä, joita ei voi korjata

L = Elämä (Life): virhe voi vaarantaa useiden ihmisten hengen

Crystal-perheen sisällä kehitysmenetelmän valinta tapahtuu yhdistämällä kriittisyyskoodi ja ohjelmiston kehitysryhmän koko. Esimerkiksi C10 tarkoittaa tilannetta, jossa kehitystyössä työskentelee kymmenen henkilöä, ja tuotettavan ohjelmiston virheellinen toiminta voi pahimmillaan aiheuttaa käsityötä. Eri Crystal-menetelmät on nimetty värien avulla. Kaikkien vähiten henkilöitä sisältävä menetelmä on Crystal Clear (kirkas), ja seuraavaksi tulevat keltainen (yellow), oranssi (orange) ja punainen (red) (Lindberg 2003). Kirkas (clear) on suunniteltu 1–6:n, keltainen alle 20:n, oranssi 20–40:n ja punainen 40–80:n henkilön projekteille (Wirgentius 2003). Crystal-perhe on esitelty kaaviona kuvassa 13.



Kuva 13: Crystal-metodologiaperhe (Tervonen 2002)

Crystal-metodologioiden ydinelementit ovat ihmis- ja kommunikaatiokeskeisyys, muokattavuus, inkrementaalinen kehitys ja lyhyt kehityssykli, menetelmän sovitus tarpeen mukaan sekä arviointipalaverit ennen ja jälkeen sykliä. (Wirgentius 2003)

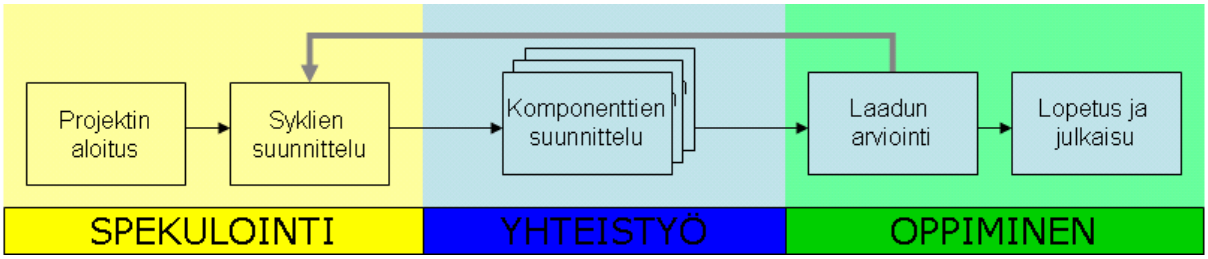
On tärkeää huomata, että Crystal-metodologiat eivät tarjoa tarkempia ohjeita siitä, miten varsinainen sovelluskehitystyö konkreettisesti tulisi tehdä, joten ne eroavat tässä kohdin esimerkiksi Extreme Programming -menetelmästä (Lindberg 2003). Sovelluskehitystyötä voidaan hyvin siis tehdä esimerkiksi XP-menetelmällä tai vaikka suunnitelmaohjautuvilla sovelluskehitysmenetelmillä.

Adaptive Software Development (ASD)

Adaptive Software Development -menetelmän (myöhemmin ASD), joka voidaan suomentaa *sopeutuvaksi ohjelmistokehitykseksi,* esitteli Jim Highsmith vuonna 2000 (Abrahamsson 2002). Menetelmän taustalla on Brian Arthurin vuonna 1996 esittämä näkemys, jonka mukaan ohjelmistoprojektien toimintaympäristö on ennalta-arvaamaton ja jatkuvasti muuttuva. ASD esiteltiin menetelmänä, joka soveltuu erityisen hyvin

monimutkaisten ohjelmistojen ja järjestelmien toteutukseen. Se hylkää ennustavuuteen perustuvan vesiputousmallin ja pohjautuu lyhyisiin, toistuviin toimitussykleihin. (Pressman 2005; Lindberg 2003)

ASD:n keskeisiä tausta-ajatuksia ovat *ihmisten keskinäinen yhteistyö* ja *itseorganisoituvat tiimit*. ASD-mallille on määritelty oma, kolmeen vaiheeseen jakautuva elinkaarensa. Vaiheet ovat **spekulointi** (speculation), **yhteistyö** (collaboration) ja **oppiminen** (learning) (Pressman 2005). Vaiheet esitetään graafisesti kuvassa 14. Spekulointi korvaa suunnitelmaohjautuvien menetelmien määrittelyvaihetta. Sitä kutsutaan spekuloinniksi, koska muutosalttiissa projekteissa ei voida etukäteen tietää lopullisen tuotteen tarkkoja yksityiskohtia, vaan spekuloimalla pyritään pääsemään yksimielisyyteen suurista suuntaviivoista. Ohjelmiston rakentamisvaihetta korvaa yhteistyövaihe, sillä ihmisten välinen yhteistyö nähdään ASD-mallissa kriittisenä menestystekijänä koko projektille. Projektin laaduntarkkailu, loppupalaverit ja katselmoinnit on korvattu sanalla oppiminen. (Lindberg 2003)



Kuva 14: ASD-mallin vaiheet (Highsmith 2000)

Spekulointivaihe aloittaa ASD-projektin. Aloituksessa hyödynnetään koko ohjelmistoprojektin aloitusinformaatiota: asiakkaan asettamia tavoitteita, aikataulua, ja perusvaatimuksia. Niiden perusteella määritellään tarvittava joukko ohjelmiston kehitysversioiden julkaisuja, joille sovitaan julkaisuajankohdat. Myös koko projektin valmistumisajankohta päätetään. Iteraatiokierrosten ja koko projektin valmistumispäivät ovat kiinteitä, eikä niistä jousteta. Jos jokin tavoitepäivämääristä vaikuttaa vaarantuvan, ei asian korjaamiseksi tehdä ylitöitä tai vaaranneta ohjelmiston laatua, vaan vähennetään kyseiseen iteraatioon kuuluvia ominaisuuksia. (Pressman 2005; Lindberg 2003)

Yhteistyövaihe tarkoittaa ohjelmiston varsinaista toteutustyötä. ASD antaa toteuttajille paljon valtaa. Pienet tiimit voivat käyttää toteutuksessaan esimerkiksi Extreme Programming -menetelmiä, kun taas suuremmille tiimeille on mahdollista käyttää jotakin muuta menetelmää. Kulloisenkin iteraatiokierroksen määrittely annetaan toteuttajille tavoitteiden muodossa, jonka jälkeen tiimi toimii itseohjaavasti näiden tavoitteiden saavuttamiseksi (Lindberg 2003). Yhteistyövaiheessa voi olla useita rinnakkaisia tiimejä,

jotka toteuttavat samanaikaisesti järjestelmän eri osakokonaisuuksia. Yhteistyövaiheessa korostetaan viestintää ja hiljaisen tiedon jakamista tiimin jäsenten ja eri tiimien kesken. (Highsmith 2000)

*Oppimisvaihee*ssa tarkastellaan yhden iteraatiokierroksen tai koko projektin työn tuloksia. Laatuarvioinnin keskeisenä tarkoituksena on huolehtia siitä, että keskittymisen kohteena ovat jatkuvasti alussa määritellyt päämäärät ja tavoitteet. ASD-tiimit arvoivat työn laatua ainakin seuraavasta neljästä näkökulmasta (Highsmith 2000):

- Asiakasnäkökulma (täyttyvätkö asiakkaan tarpeet?)
- Tekninen näkökulma (koodikatselmukset, testaukset)
- Toteutustiimin toiminta ja käytössä olevat menetelmät
  (tiimin suorituskyvyn arviointikysymyksiä: Mitkä osiot toimivat? Mitkä eivät toimi? Mitä pitää tehdä enemmän? Mitä pitää tehdä vähemmän?)
- Projektin tilanne (Missä vaiheessa projekti on? Missä vaiheessa projekti on suhteessa suunnitelmaan? Missä vaiheessa projektin pitäisi olla?)

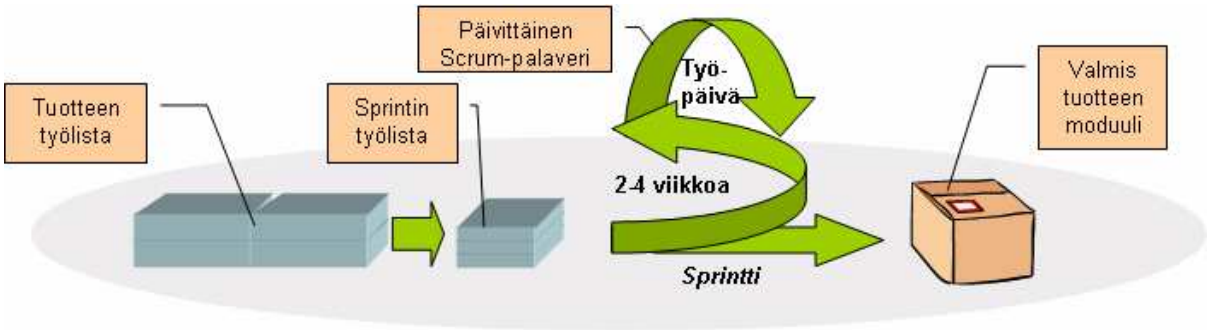
Nämä kolme vaihetta, spekulointi, yhteistyö ja oppiminen, toistuvat iteraatiokierroksissa niin pitkään, kunnes saavutetaan projektille määritetty valmistumispäivä. ASD-mallin mukaan on keskeistä toimittaa lopputuote asiakkaalle ennalta määritettynä valmistumispäivänä – vaikka muutamilta ominaisuuksiltaan puutteellisenakin. Aikataulu on tärkeämpi kuin ohjelmiston täydellinen laajuus, ellei asiakas päätä toisin. (Lindberg 2003)

Scrum

Rugby-pelistä nimensä lainannut *Scrum-menetelmä* on Jeff Sutherlandin 1990-luvun alkupuolella kehittämä menetelmä, jota ovat myöhemmin kehittäneet Schwaber ja Beedle (Pressman 2005). Scrumin näkemys ohjelmistokehityksen toimintaympäristöistä on vastaava kuin ASD-menetelmällä; toimintaympäristö sisältää useita hallitsemattomia muuttujia, joihin kehitysprojektin on sopeuduttava. Scrum on ajateltavissa kehyksenä tai runkona ohjelmistojen toteutukseen liittyvään epävarmuuden kontrollointiin (Lindberg 2003). Scrum-menetelmän periaatteet ovat linjassa ketterän kehityksen periaatteiden kanssa. Kommunikaation maksimointi, muutoksiin sopeutuminen, toteutustyön inkrementaalisuus ja runsas testaaminen ovat myös Scrumin kantavia perusperiaatteita.

Scrum-menetelmän eteneminen kuvataan graafisesti kuvassa 15. Scrum-menetelmä listaa toteutettavan työn ominaisuuslistaan, jota kutsutaan *tuotteen työlistaksi* (product

backlog). Siihen on koottu kaikki tarvittavat työt tuotteen tavoitellun lopputilan saavuttamiseksi. Tätä työlistaa lähdetään purkamaan niin kutsuttujen sprinttien avulla. Jokaisen sprintin alussa pidetään *sprintin suunnittelupalaveri (sprint planning meeting)*, jossa asiakas tai muu tuotteen omistaja (product owner) määrittelee tärkeysjärjestyksen työlistassa luetelluille töille. Tämän jälkeen Scrum-tiimi valitsee tulevan sprintin aikana tehtävissä olevat työt. Työt siirretään tuotteen työlistasta *sprintin työlistaan*. Sprintit kestävät yleensä kahdesta neljään viikkoa. (Mountain Goat Software 2006)



Kuva 15: Scrum -prosessi (Mountain Goat Software 2006)

Sprintin aikana Scrum-tiimi pitää päivittäisiä palavereja. Palaverien kesto on hyvin lyhyt, noin 15 minuuttia. Palaverissa jokainen tiimin jäsen vastaa seuraaviin kysymyksiin:

> Mitä olet tehnyt viime palaverin jälkeen? Mitä esteitä sinulla on työsi edistämisessä? Mitä töitä aiot tehdä tekemättömien töiden listasta ennen seuraavaa palaveria?

Sprint-palaveria johtaa projektipäällikkö (Scrum master), joka huolehtii palaverin etenemisestä ja siitä, että kaikki vastaavat näihin kolmeen kysymykseen. Scrum-palaverien ideana on tunnistaa mahdolliset ongelmat niin pian kuin mahdollista. Palaverit myös sosialisoivat tietoa tiimin jäsenten kesken ja mahdollistavat näin itseorganisoituvaa tiimirakennetta. (Pressmann 2005)

Jokaisen sprintin päätteeksi Scrum-tiimi esittelee *sprintin loppupalaverissa* (sprint review meeting) päättyneen sprintin aikana toteutetut toiminnot asiakkaalle. Näissä esittelyissä, demoissa, asiakkaat arvioivat toteutettuja toimintoja ja antavat toteuttajille palautetta. Tilaisuudessa voidaan päättää myös uuden sprintin aloittamisesta käymällä läpi koko tuotteen työlistaa ja palaamalla näin uuden sprintin suunnitteluvaiheeseen. (Pressmann 2005; Mountain Goat Software 2006).

2.5 Suunnitelmaohjautuvien ja ketterien menetelmien vertailu

Tämän luvun alussa ohjelmistokehitysmenetelmät jaettiin kahteen ryhmään dokumentaation määrän perusteella. Dokumentaation määrän lisäksi menetelmille voidaan löytää myös muita kuvaavia piirteitä. Barry Boehm määritteli artikkelissa *"Get ready for agile methods, with care"* (2002) taulukossa 1 kuvatut keskeiset osa-alueet molemmille menetelmille.

Taulukko 1: Ketterien ja suunnitelmaohjautuvien menetelmien tyypillisiä piirteitä (Boehm 2002)

Osa-alue	Ketterät menetelmät	Suunnitelmaohjautuvat menetelmät
Kehittäjät	Ketteriä, asiantuntevia, rinnakkaiseen työhön kykeneviä, yhteistyökykyisiä	Suunnitelmaorientoituneita, keskitasoiset taidot, pääsy ulkoisiin tietolähteisiin
Asiakkaat	Omistautuneita, asiantuntevia, rinnakkaiseen työhön kykeneviä yhteistyöhaluisia, riittävät valtuudet	Saavat tietonsa asiantuntevilta henkilöiltä, yhteistyöhaluisia, valtuutettuja
Määrittelyt	Pitkälti työn ohessa esiin tulevia, nopeasti muuttuvia	Aikaisin tiedossa olevia, pitkälti muuttumattomia
Järjestelmän arkkitehtuuri	Suunniteltu tämänhetkisiin tarpeisiin	Suunniteltu tämänhetkisiin ja tulevaisuuden tarpeisiin
Rakenteen parantaminen	Edullista	Kallista
Koko	Pienemmät tiimit ja tuotteet	Suuremmat tiimit ja tuotteet
Ensisijainen tavoite	Nopea hyödyn tuottaminen	Korkea luotettavuus

Kehittäjille (ohjelmoijille, suunnittelijoille ja laajemmin koko toteuttajayritykselle) asetettavat vaatimukset poikkeavat ketterissä ja suunnitelmaohjautuvissa menetelmissä merkittävästi. Useimmat ketterät menetelmät edellyttävät kehittäjiltä laajaa asiantuntemusta, hyvää kommunikaatiotaitoa, lahjakkuutta ja omistautumista työlleen (Boehm 2002). Erinomaisen henkilöstön hankkiminen on keskeinen osa ketteriä menetelmiä. Suunnitelmaohjautuvat menetelmät eivät nojaa yksittäisten henkilöiden kykyihin niin merkittävästi: yksilöiden tietojen ja taitojen vajavaisuutta täydennetään laajemmilla suunnitelmilla, dokumentaatiolla ja tiedon hakemisella ulkoisista tietolähteistä, esimerkiksi toisilta henkilöiltä, kirjoista, Internetistä tai konsulteilta (Boehm 2002). Ohjelmoijien kommunikaatiotaitoihin liittyy myös kirjoittamattoman hiljaisen tiedon (tacit knowledge) siirtyminen henkilöltä toiselle: ketterissä menetelmissä periaatteena tiedon siirrossa on ihmisten välinen keskustelu, suunnitelmaohjautuvissa välineenä ovat paperit ja tiedostot. Kommunikaatiotapojen eroihin liittyvät myös viestinnän suunnat; keskustelu on yleensä kaksisuuntaista, kun dokumentit siirtävät viestiä vain yhteen suuntaan (Boehm & Turner 2004). Boehm (2002) esittää yhdeksi ketterien menetelmien ongelmaksi hyvien kehittäjien saamisen. Hän väittää, että 49.999% maailman ohjelmistokehittäjistä ovat keskitasoa huonompia työssään. Tämä

tilanne rajaa merkittävästi ketterien sovelluskehitysmenetelmien käyttömahdollisuuksia, sillä henkilöstön taitojen puute voi puoltaa suunnitelmaohjautuvien menetelmien käyttöä ketterien menetelmien sijasta.

Asiakkailta edellytetään ketterissä menetelmissä paljon osallistumista työhön. Keskeisin ero suunnitelmaohjautuviin menetelmiin on, että ketterät menetelmät edellyttävät pitkälti ohjelman kehitystyöhön omistautunutta, järjestelmän tarpeet perusteellisesti tuntevaa asiakasta, kun suunnitelmaohjautuvat menetelmät edellyttävät asiakasta, jonka kanssa pystytään etukäteen tekemään tarkat sopimukset, suunnitelmat ja määrittelyt. (Boehm & Turner 2004) Ketterille menetelmille aiheutuu tästä vaatimuksesta se riski, että asiakkaalta ei saada käyttöön riittävän pätevää ja tarvittaessa käytettävissä olevaa asiantuntemusta. Usein yritykset tarjoavat kehittäjien avuksi sen henkilön, jolla on eniten vapaata aikaa – ei eniten asiantuntemusta. On tutkittu (Boehm & Turner 2004), että menestyksekäs ohjelmistokehitysprojekti edellyttää asiakkaalta seuraavia piirteitä: yhteistyökykyä, asiantuntemusta, valtaa ja sitoutumista. Asiakkaan yhteistyökyvyttömyys murentaa kehittäjien kiinnostusta työhön ja asiantuntemattomuus johtaa kelvottomaan lopputulokseen, riittämätön valta taas aiheuttaa viiveitä asiakkaan edustajan etsiessä valtuutusta omasta organisaatiostaan. Sitoutumattomuus tarkoittaa asiakkaalta pyydettyjen vastausten viipymistä ja tavoittamattomuutta silloin kun heitä eniten tarvittaisiin.

Edellä kuvatut hyvän asiakkaan ominaisuudet ovat vastaavat myös suunnitelmaohjautuville menetelmille, mutta ne eivät ole niin välttämättömiä kuin ketterissä menetelmissä. Suunnitelmaohjautuvien menetelmien suurin asiakasriski on byrokratia; liian tarkka sopimuksien laatiminen ja seuraaminen voi johtaa keskittymisen vinoutumiseen, jolloin varsinaisen työn tekemisen sijasta aletaan keskittyä byrokratiaan. (Boehm & Turner 2004).

Määrittelyt kuvataan useimmissa ketterissä menetelmissä joustaviksi, epäformaaleiksi tarinoiksi. Nopea uusien versioiden tuottaminen ja luottamus siihen, että tarvittavat muutokset pystytään tekemään, mahdollistavat tämän. Edellisen version valmistuttua seuraavaa versiota aloitettaessa voidaan asiakkaan kanssa neuvotella seuraavaan versioon kipeimmin tarvittavista uusista ominaisuuksista (Boehm & Turner 2004). Ketterät menetelmät olettavat myös, että tulevaisuuden suunnittelu pitkälle ei kannata – suunnitelmat eivät kuitenkaan pidä, sillä muutoksia tulee aina (Liedenpohja 2006). Suunnitelmaohjautuvat menetelmät suosivat yksityiskohtaisia, formaaleja määrittelyjä, jotka esitetään kirjallisessa dokumentissa. Niiden heikkoutena on kuitenkin ominaisuuksien tärkeysjärjestykseen asettelun vaikeus sekä muutokseen sopeutumattomuus. Suunnitelmaohjautuvien menetelmien määrittelytyön

menettelytavoissa on kuitenkin tiettyjä vahvuuksia; epäfunktionaaliset määritteet, kuten luotettavuus, suorituskyky ja skaalautuvuus tulevat paljon paremmin käsitellyiksi suunnitelmaohjautuvissa menetelmissä. Näillä on keskeinen merkitys erityisesti suurissa, kriittisiä tehtäviä hoitavissa järjestelmissä (Boehm & Turner 2004).

Järjestelmän arkkitehtuurilla tarkoitetaan toteutettavan järjestelmän teknisiä ratkaisuja. Ketterät menetelmät kannustavat käyttämään tekniikkaa, joka täyttää minimissään tämän hetkisen tarpeen. Tämä pitää paikkansa erityisesti silloin, kun tulevaisuus on ennustamaton. Tilanteessa, jossa tulevaisuuden tarpeet tiedetään, tämä voi kuitenkin tarkoittaa tehottomuutta. Suunnitelmaohjautuvissa menetelmissä varaudutaan alusta alkaen etukäteismäärittelyillä ja arkkitehtuurisuunnitelmilla ennustettavissa olevaan muutokseen. Tämä antaa toteuttajille mahdollisuuden tehdä järjestelmään liityntöjä ja ominaisuuksia, jotka tulevaisuudessa auttavat toteuttamaan uudet ominaisuudet nopeasti. (Boehm & Turner 2004)

Rakenteen parantamisen (refaktoroinnin) edullisuus ketterissä menetelmissä perustuu siihen, että rakenteen parantamista tehdään jatkuvasti. Idea on, että jatkuvat parannukset ehkäisevät isojen ja kalliiden muutostöiden tarvetta. Boehm & Turner (2004) esittävät kuitenkin, että tämä ei välttämättä pidä paikkaansa. Menetelmä toimii silloin, jos kyseessä ovat suhteellisen pienet järjestelmät ja ohjelmoijat ovat todellisia asiantuntijoita. Heikompitaitoisilla ohjelmoijilla ja laajemmilla järjestelmillä tilanne voi olla toinen, jolloin rakenteen parantamiseen joudutaan panostamaan paljon. Muutenkaan jatkuvaan, edulliseen rakenteen kehitystyöhön ei voida luottaa, jos järjestelmien koot kasvavat. Esimerkiksi jotkin aluksi hyvänä pidetyt yksinkertaiset arkkitehtuuriratkaisut eivät välttämättä sopeudu suurempiin järjestelmiin. (Boehm & Turner 2004)

Ketterillä menetelmillä toteutettavien projektien koko on yleensä pienempi kuin suunnitelmaohjautuvilla menetelmillä. Yleisesti ollaan sitä mieltä, että erityisesti ketterien menetelmien viestintätavat estävät yli 40 henkilön tiimien toiminnan. Optimikokona pidetään noin kymmentä henkilöä (Boehm & Turner). Ketterien menetelmien skaalaaminen suurempiin projekteihin perustuukin pilkkomiseen pienempiin osiin; projekti pilkotaan sopiviin paloihin ja pienet ketterät kehitystiimit toimivat itsenäisesti omaa osaansa kehittäen. Yhteensopivuudesta varmistutaan liittämällä paloja yhteen tiheään tahtiin ja toimittamalla uusia versioita usein (Beck 2004). Vaikka on olemassa menestystarinoita 250 hengen ketteristä projekteista (Highsmith 2002), ollaan yleisesti sitä mieltä, että suunnitelmaohjautuvat menetelmät ovat tehokkaampia suurissa projekteissa.

Ensisijainen tavoite ketterissä menetelmissä on Agile manifeston mukaan *”täyttää asiakkaan vaatimukset jatkuvilla ja riittävän aikaisilla ohjelmistotoimituksilla.”* Useimmissa tapauksissa tämä periaate toimii hyvin, kunhan järjestelmien koko ei ole kovin suuri. Suurten järjestelmien tapauksessa keskittyminen konkreettisten lopputulosten saamiseen asiakkaan nähtäville voi olla väärä ratkaisu, joka saattaa johtaa merkittävän suuriin ongelmiin, jos mittakaavaa ei voidakaan suurentaa olemassa olevalla rakenteella. Suunnitelmaohjautuvien menetelmien tavoitteita ovat perinteisesti olleet *ennustettavuus, toistettavuus ja optimointi.* Nämä tavoitteet ovat toimivia tilanteissa, joissa tulevaisuus on tiedossa olevaa, mutta tilanteessa, jossa muutostarve on jatkuvaa ja muutoksen suunta ennalta-arvaamatonta, on tavoitteen oltava toisenlainen. Suunnitelmaohjautuvat menetelmät sopivat erityisen hyvin tavoitteidensa puolesta myös tilanteisiin, joissa vaaditaan erityisen korkeaa luotettavuutta, kuten elintärkeitä järjestelmiä ylläpitävissä ohjelmistoissa. (Boehm 2002)

Arvioitaessa ketterien menetelmien soveltuvuutta käsillä olevaan työhön, on olemassa muutamia keskeisiä asioita, joihin on syytä kiinnittää huomiota. Turk esitti vuonna 2002 artikkelissaan ”Limitations of Agile Software Processes” seuraavat kuusi tilannetta, jotka asettavat rajoitteita ketterien menetelmien hyödyntämiselle:

1. Hajautetut kehitysympäristöt
2. Alihankinta
3. Uudelleenkäytettävien ohjelmistokomponenttien kehitys
4. Suuret kehitystiimit
5. Turvallisuuskriittiset ohjelmistot
6. Laajat, monimutkaiset ohjelmistot

Hajautetuissa kehitysympäristöissä työskenneltäessä ketterien menetelmien korostama kasvokkain tapahtuma kommunikointi on hankalaa tai mahdotonta, vaikkakin videoneuvotteluilla ja muilla nykyaikaisilla viestintävälineillä tilannetta voidaankin parantaa. Alihankinnan haasteet liittyvät sopimusteknisiin asioihin: toteutettavaksi sovittavan osakokonaisuuden rajaaminen on ketterien menetelmien työtavoilla haastavaa. Menetelmäksi ehdotetaan alihankintasopimusten toteuttamista kaksiosaisena. Sopimuksen ensimmäinen osa kuvaa keskeisen, ennalta tiedossa olevan perusrakenteen (ja on mahdollisesti kiinteähintainen). Toinen osa taas on muuttuva, joka voi sisältää (ensimmäisessä osassa määritellyllä välillä) vaihtelevan määrän ominaisuuksia ja työtä. (Turk 2002)

Ketterillä menetelmillä kehitettyjen ohjelmistojen komponenttien uudelleenkäytettävyys kyseenalaistetaan usein ketterien menetelmien perustavoitteen, akuutin ongelman ratkaisevan ohjelman tuottamisen, takia. Tuotetut ohjelmistot keskittyvät ratkaisemaan juuri nyt käsillä olevaa ongelmaa, eikä tulevaisuuteen varauduta. Jos ohjelmistoprojektin tuotteena ovat ohjelmistokomponentit, joita tullaan tarvitsemaan myöhemmin, voi tilanteeseen sopia paremmin jokin suunnitelmaohjautuva ohjelmistokehitysmenetelmä. Suurien kehitystiimien kohdalla organisaation hallinnointi edellyttää useampia kommunikaatioväyliä kokonaisuuden hallintaan, ja tähän ketterien menetelmien keskusteluun pohjautuva viestintä ei välttämättä skaalaudu. Turvallisuuskriittisiin ohjelmistoihin liittyy haaste laadunvarmistuksessa. Turk esittää artikkelissaan epäilyksen, etteivät ketterien kehitysmenetelmien tavat ole riittäviä turvallisuuskriittisille ohjelmistoille. Suunnitelmaohjautuvien kehitysmenetelmien yksityiskohtaisesti määritellyt testitapaukset ja testaussuunnitelmat tarjoavat tähän paremman, mutta myös kalliimman tavan. Laajojen järjestelmien kohdalla palataan aiemmin mainittuun havaintoon siitä, ettei koodin jatkuva parantaminen (refaktorointi) poista suunnittelun tarvetta suurissa järjestelmissä. Tietyn koon ylitettyään ohjelmiston komponenttien uudelleenohjelmointi tarvittaessa ei ole enää järkevää, jonka vuoksi tarpeisiin kannattaa varautua etukäteen suunnittelemalla komponentit perusteellisesti (Turk 2002).

Tällä hetkellä monet ohjelmistoalan yritykset harkitsevat, kannattaisiko heidän käyttää suunnitelmaohjautuvia vai ketteriä ohjelmistokehitysmenetelmiä. Useimmissa yrityksissä käytetään tällä hetkellä suunnitelmaohjautuvia menetelmiä, joihin verrattuna ketterät menetelmät voivat vaikuttaa epäjärjestelmällisiltä ja huonosti hallittavilta. Asiakkaiden tarpeet ja liiketoimintaprosessit vaativat kuitenkin nopeampaa ja asiakkaalle pikaisemmin arvoa tuottavaa toimintaa, joka puoltaa ketteriä menetelmiä. Kiinnostuksesta kertoo, että esimerkiksi T-lehti esitteli artikkelissaan ”Suomalainen softateollisuus venyttelee ketterämmäksi” (Niittymies 2006) ketterän ohjelmistokehityksen menetelmät aseeksi, jolla koko suomalainen ohjelmistokehityssektori voi pärjätä kilpaillessaan laajenevaa ohjelmointityön ulkomaille vientiä vastaan. Vaikka ulkomailla ohjelmistokehitystyön kustannukset ovat merkittävästi edullisempia kuin Suomessa, voivat tehokkaat menetelmät olla silti peruste toteuttaa työ Suomessa.

Valinnan tekeminen menetelmäsuuntausten välillä nähdään usein polarisoituneena, ja valinnan tekeminen siksi vaikeana (Kangas 2003). On kuitenkin esitetty, että tällaisesta kaksijakoisuudesta olisi syytä luopua, koska ohjelmistokehityksessä voidaan hyvin yhdistää piirteitä suunnitelmaohjautuvista ja ketteristä menetelmistä (Boehm 2002). Yhteistyössä voi siis olla mahdollisuus.

Vertailun lopuksi voidaan lainata Kent Beckin, XP-menetelmän kehittäjän tiivistystä siitä, miten XP (ja muut ketterät menetelmät) eroaa suunnitelmaohjautuvista menetelmistä, joka kuuluu vapaasti suomennettuna seuraavasti:

”Uskon, että ’suunnitelmaohjautuvuus’ on avaintermi. Kuvaisin XP:tä ’suunnitteluohjatuksi’ menetelmäksi. XP:ssä keskeistä on suunnittelutyössä tapahtuva yhteistyö ja tasapainottelu eri tavoitteiden välillä. Sen tuloksina saadut suunnitelmat ovat lyhytikäisiä – ei siksi että ne olisivat huonoja, vaan siksi, että niiden taustalla olevat oletukset ovat lyhytikäisiä. ”

3. Ohjelmistokehitys FNS:ssä

Tässä luvussa esitellään Finnish Net Solutions Oy:n oma ohjelmistokehitysmenetelmä, jonka yritys on kehittänyt itselleen toimintahistoriansa aikana. Menetelmä esitellään teoreettisesti ja yhden käytännön esimerkin avulla. Sen jälkeen menetelmää verrataan edellisessä luvussa esiteltyihin yleisesti tunnettuihin ohjelmistokehitysmenetelmiin.

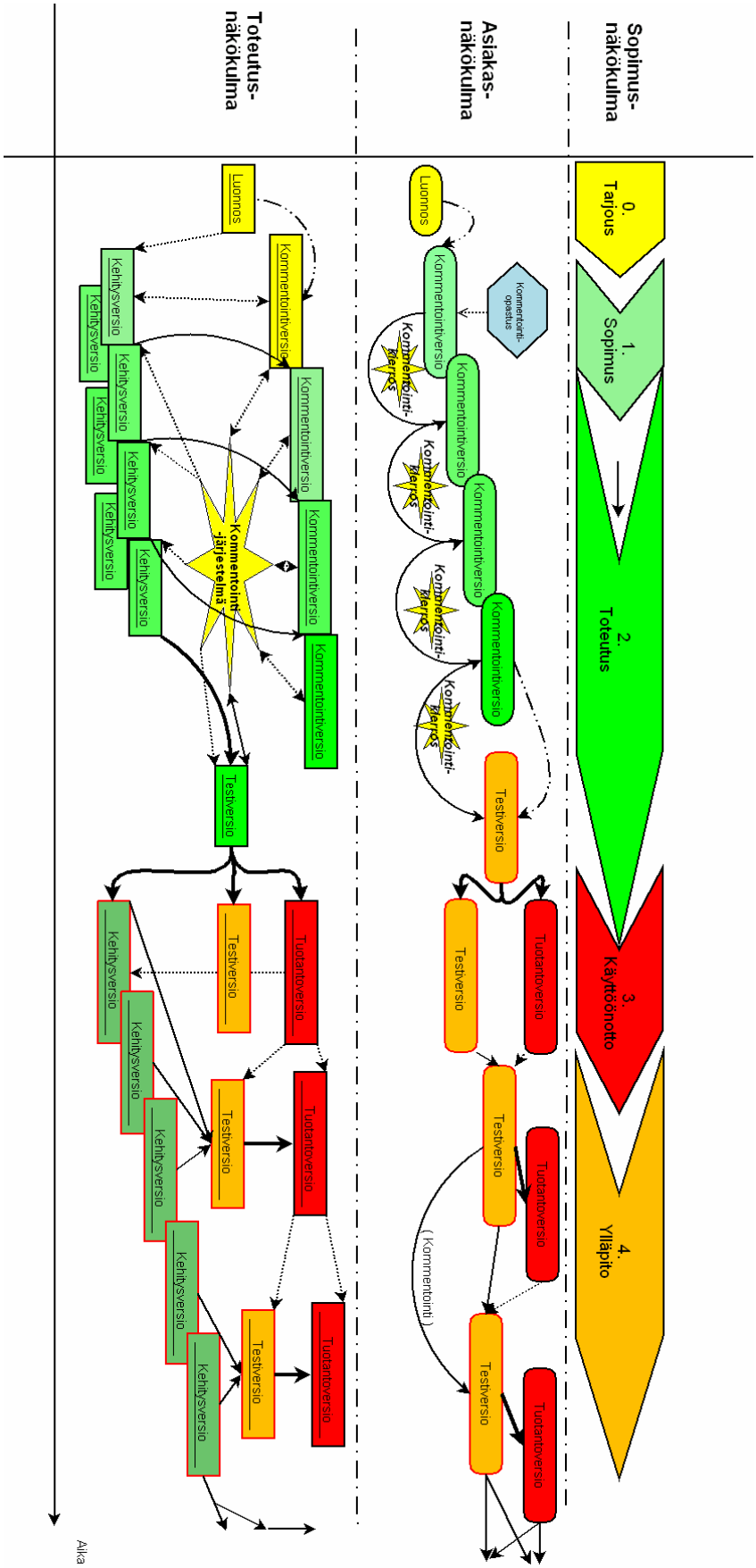
Esitettyä menetelmää käytetään ensisijaisesti niissä asiakasprojekteissa, joissa FNS toteuttaa alusta alkaen asiakkaan tarpeisiin räätälöidyn tietojärjestelmän ilman valmista ohjelmistotuotetta. Myös valmiita ohjelmistotuotteita käytettäessä menetelmää hyödynnetään soveltuvin osin ohjelmistotuotteiden asiakaskohtaisessa räätälöinnissä.

Taustana FNS:n oman ohjelmistokehitysmenetelmän kehittymiselle on ollut tarve viestiä tehokkaasti laajojen, maantieteellisesti hajallaan olevien asiakasryhmien kesken. Kiinnostus oman, mahdollisimman tehokkaan menetelmän rakentamiseksi on ollut jatkuvaa laadullisesti paremman lopputuloksen, tehokkaamman oman toiminnan sekä sitä kautta paremman taloudellisen tuloksen saavuttamiseksi. Menetelmä perustuu hyvin pitkälti sähköiseen viestintään, jota toteutetaan FNS:n menetelmässä erityisen kommentointijärjestelmän avulla.

FNS:n ohjelmistokehitysmenetelmän tausta-ajatuksiin kuuluu myös muutamia tärkeitä asioita, joita selvitetään ennen varsinaiseen ohjelmistokehitystyöhön ryhtymistä. Näitä ovat asiakkaan toimialaan ja liiketoimintaan tutustuminen sekä asiakkaan keskeisimpien tarpeiden selvittäminen. Kehitystyötä aloitettaessa kootaan ensin yhteen tieto siitä, mitä ongelmia halutaan ratkaista. Keskeisiä menetelmään kuuluvia perusasioita on myös aktiivinen yhteydenpito asiakkaaseen: asiakkaan edustajille esitetään ehdotuksia ja kysymyksiä, joilla toteuttajat pyrkivät aktiivisesti selvittämään lisää tietoa asiakkaan toiminnasta mahdollisimman hyvien ratkaisujen löytämiseksi. Ideointia ja ajatustyötä määrittelyjen kokoamiseksi pyritään siis tekemään myös toimittajan toimesta.

3.1 Kuvaus FNS:n ohjelmistokehitysmenetelmästä

FNS:n käyttämää ohjelmistokehitysmallia voidaan tarkastella kolmesta eri näkökulmasta: sopimus-, asiakas- ja toteutusnäkökulmasta. Nämä näkökulmat sisältöineen on esitetty kuvassa 16.



Kuva 16: FNS:n ohjelmistokehitysmenetelmä

Eri näkökulmien tapahtumat on esitetty kuvan 16 kaaviossa Y-akselilla. X-akseli kuvaa ohjelmistoprojektin etenemistä ajan suhteen.

Sopimusnäkökulma kuvaa ohjelmistoprosessin keskeisimmät vaiheet sopimusteknisestä näkökulmasta. ***Asiakasnäkökulma*** tarkoittaa asiakkaan kussakin vaiheessa näkemää ohjelmistoprojektin tuotetta, käytettävissä ja nähtävissä olevaa konkreettista ohjelmistoa tai palvelua. Lisäksi asiakastasolla on kuvattu asiakkaan suorittamia konkreettisia viestintätoimenpiteitä, joita hän tekee välittääkseen ohjelmalta tarvittavia tarpeita FNS:lle. ***Toteutusnäkökulma*** tarkoittaa FNS:n teknisen toteutuksen, pääasiassa ohjelmoijien ja suunnitelijoiden, näkökulmaa ohjelmistokehitysprojektiin.

Seuraavassa kappaleessa esitellään FNS:n ohjelmistokehitysmenetelmän keskeinen työkalu, **kommentointijärjestelmä**, joka liittyy oleellisesti sopimusvaiheesta alkaen kaikkiin prosessin vaiheisiin. Tämän jälkeen seuraavissa kappaleissa kuvataan tarkemmin sopimus-, asiakas- ja toteutustasojen sisällöt jaoteltuna sopimusvaiheiden mukaisesti.

3.1.1 Kommentointijärjestelmän esittely

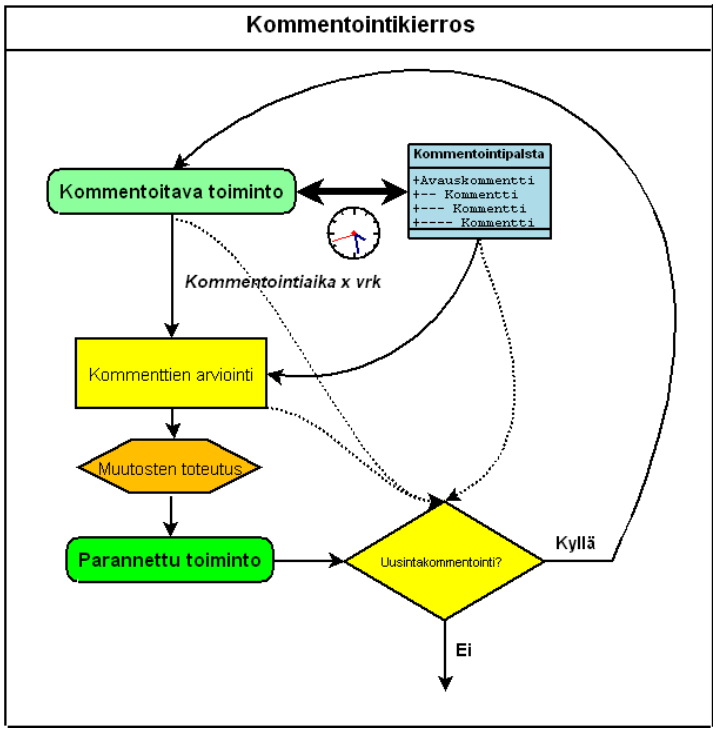
FNS:n ohjelmistokehitysmenetelmän ydin on kommentointijärjestelmä, jonka varaan koko ohjelmistoprosessi nojaa. Kommentointijärjestelmän idea perustuu toimittajan ja asiakkaan välisen viestinnän tehostamiseen, asiakkaan edustajien keskinäisen viestinnän tehostumiseen sekä asiakkaan mahdollisuuteen hahmottaa järjestelmän kokonaisuus, osat ja tavoitteet jatkuvasti kehittyvän kommentointiversion avulla.

Kommentointijärjestelmä koostuu *kommentointipalstoista* (kuva 17), jotka ovat pieniä keskustelupalstoja, joita on liitetty kulloinkin kehitettävänä oleviin toimintoihin rakennettavana olevan järjestelmän *kommentointiversion* käyttöliittymässä. Kommentointiversio on usein aluksi pelkkä käyttöliittymäluonnos järjestelmän käyttöliittymästä, mutta projektin edetessä se kehittyy jatkuvasti eteenpäin kohti lopullista järjestelmää. Kommentointia suorittaa projektin alussa valittu *kommentointiryhmä*, joka koostuu kaikista rakennettavan järjestelmän tulevien käyttäjien ja käyttäjäryhmien edustajista sekä heidän varahenkilöistään. Kommentointipalstoihin liittyy käsite *kommentointiaika*. Kommentointiaika on rajattu aika, esimerkiksi 5 työpäivää, jonka ajan kehitettävänä oleva toiminto on kommentoitavana. Tämän jälkeen kommentointi sulkeutuu. Kommentointiajan päätyttyä *kommentointikierros* on päättynyt, ja *asiakkaan yhteyshenkilö* (kommentointiryhmän johtaja) käy FNS:n edustajan kanssa kommentit läpi ja päättää muutosehdotusten toteuttamisesta.

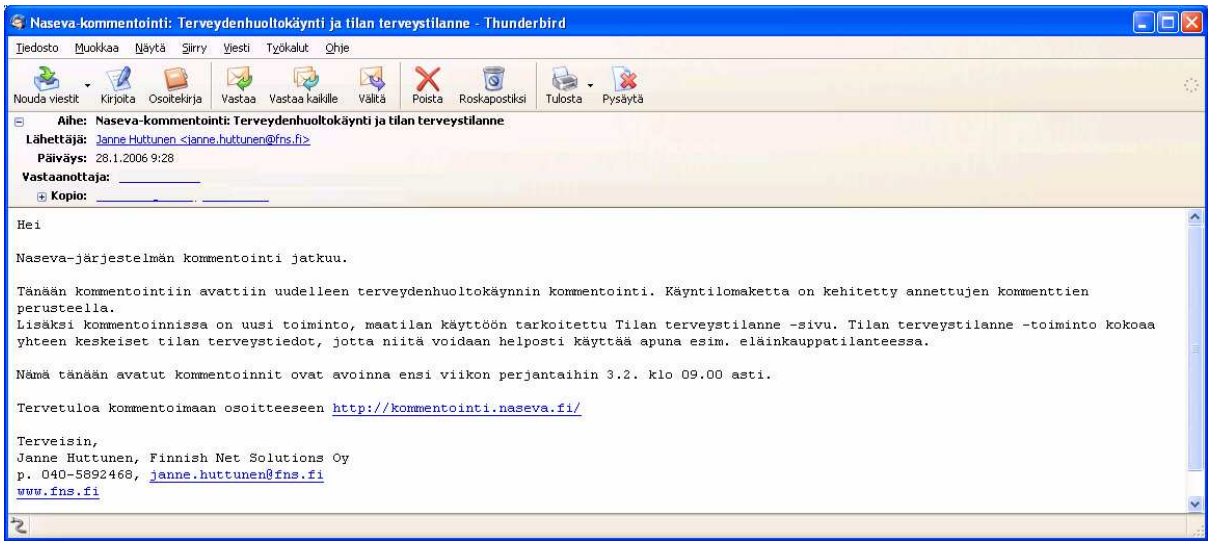
Kuva 17: Esimerkki kommentointipalstan näkymästä

Kommentointikierrokset etenevät iteratiivisesti. Ensimmäinen kommentointiversio kommentointiin tulevasta toiminnosta on yleensä toimittajan ja asiakkaan yhteyshenkilön yhdessä tekemä ehdotus, joka on tehty joillakin oletuksilla ja esitiedoilla toiminnallisuudesta ja tietojen tallennustarpeesta. Ehdotus annetaan kyseisen toiminnon asiantuntijoiden, eli järjestelmän tulevien käyttäjien arvioitavaksi kommentointikierroksen ajaksi. Kommentointikierroksen päätyttyä yhteyshenkilö arvioi yhdessä toteuttajan kanssa saatujen kommenttien perusteella toiminnon muutostarpeita, jolloin osa ehdotuksista valitaan toteutettavaksi, osa hylättäväksi. Ristiriitaisissa tapauksissa yhteyshenkilö tekee ratkaisun muutoksista. Muutostarpeiden vaikutus kokonaisuuteen arvioidaan näiden päätösten jälkeen. Jos muutokset ovat yhteyshenkilön mielestä merkittäviä, toiminto voidaan ottaa uudelleen kommentointiin.

Kommentointikierros (kuva 18) on kommentointijärjestelmän perussykli, joita voi olla menossa useita rinnakkaisia. Uusista kommentointiin tulevista toiminnoista (1. kommentointikierros) ja uudelleen kommentoitavista toiminnoista ilmoitetaan kommentointiryhmälle aina kommentoinnin avautuessa sähköpostitse suunnittelijoiden toimesta. Viimeinen kommenttien jättöaika tuodaan kommentoijille selkeästi esille (kuva 19). Perusajatuksena on, että kommentointi on mahdollista ainoastaan kommentointiaikana, ei muulloin. Tällä tavoin työ pystytään pitämään jatkuvasti etenemässä ja projekti aikataulussa.



Kuva 18: Kommentointikierroksen eteneminen



Kuva 19: Ilmoitus uudesta kommentoitavasta toiminnosta ja uusintakommentoinnista

Kommentointihistoria sekä tällä hetkellä kommentoitavana olevat toiminnot ovat koko kehitystyön ajan nähtävillä kommentointiversiossa, ja kenellä tahansa kommentointiryhmän jäsenellä on mahdollisuus selata vanhoja kommentoitavana olleita toimintoja ja niihin liittyviä kommentteja. Kun kaikki toiminnoista jätetyt kommentit näkyvät kaikille kommentoijille, saadaan tieto välittymään tehokkaasti. Tällä tavoin kehittäjät eivät esimerkiksi saa samaa ideaa tai muutosehdotusta sähköpostilla kymmentä kertaa. Yksittäisten kommentointipalstojen lisäksi kokonaiskuvaa kaikkien

kommentointien vaiheesta antaa kommentointiversioon liitetty kommentointien yhteenvetosivu, joka kokoaa aktiiviset ja päättyneet kommentoinnit yhteen luetteloon.

Kommentointijärjestelmää käytetään kaikissa ohjelmistokehitysprojektin vaiheissa lukuun ottamatta tarjousvaihetta. Tarjousvaiheessa tarjouksen mukaan liitettävän käyttöliittymäluonnoksen tarkoituksena on nostaa esille ihmisten luontainen tarve kommentoida ja kehittää saatua luonnosta eteenpäin. Tähän viestintätarpeeseen kommentointijärjestelmä tarjoaa välineen.

3.1.2 Tarjousvaihe

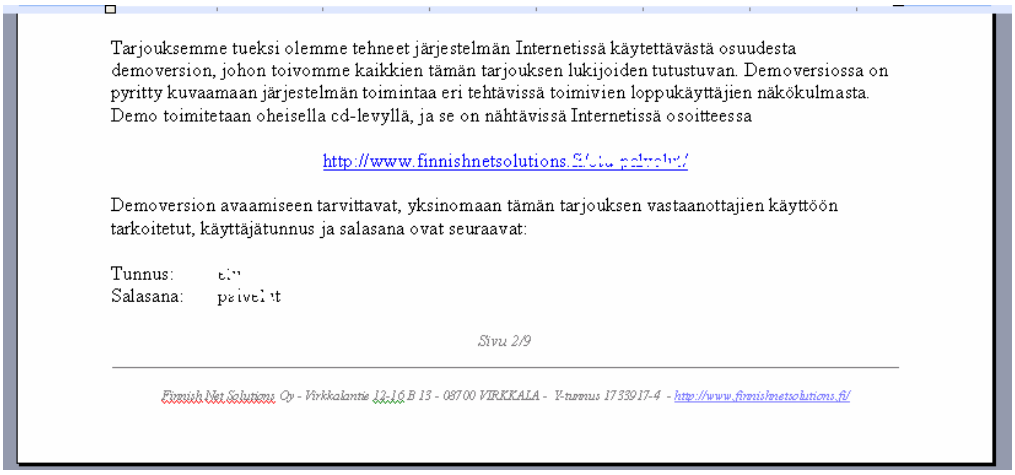
FNS aloittaa lähes kaikki ohjelmistoprojektinsa tekemällä ensimmäiseksi asiakkaalle tarjouksen toteutettavasta järjestelmästä. Asiakkaalta pyritään ennen tarjouksen tekemistä saamaan mahdollisimman tarkka kuvaus toteutettavasta järjestelmästä. Keskeisintä on saada selville ne tavoitteet, joihin asiakas järjestelmän hankinnalla pyrkii. Asiakkaan tavoitteena saattaa olla esimerkiksi tiedonkulun nopeutuminen yrityksen sisällä, yrityksen tuottamien tuotteiden nopeampi toimitus loppukäyttäjälle tai kilpailuedun saavuttaminen paremmalla palvelulla.

Useimpien tarjousten tietolähteenä ovat asiakkaan edustajien kanssa käydyt keskustelut sekä kirjalliset, useimmiten muutamalla sivulla tiiviisti keskeisimmät tarvittavat ominaisuudet esittelevät vapaamuotoiset määrittelydokumentit. Nämä määrittelydokumentit voivat olla myös entisestä, korvattavaksi halutusta järjestelmästä otettuja malleja, sähköpostiviestissä esitettyjä listoja tai muita, hyvin vaihtelevalla tarkkuudella esitettyjä kuvauksia asiakkaan tarpeista, toiveista ja odotuksista.

Vaikka on olemassa poikkeuksiakin, useimmat FNS:n saamat tarjouspyynnöt voisivat olla tarjouksen tekijän näkökulmasta tarkempia. Yleisin tilanne tarjouspyyntövaiheessa on, ettei asiakas aivan tarkasti tiedä, mitä haluaa. Asiakas haluaa useimmiten tarjoajan, FNS:n, tarjoavan valmista ratkaisua ongelmaansa. Asiaa voidaan ajatella niin, että tarjouspyyntö ja tarjous ovat ajateltavissa ensimmäiseksi vaiheeksi ohjelmistokehitysmenetelmissä tunnettua *vaatimusmäärittelyä.* Vaatimusmäärittelyn tekeminen on yleisesti vaikeana pidetty asia, ja siihen tiedetään liittyvän hyvin monia haasteita (Riekkinen 2006).

Tehdessään tarjouksen FNS toimittaa asiakkaalle usein tyypillisen tarjousasiakirjan liitteenä alustavan, tarjotun järjestelmän mallin. Tämä *luonnos*, joskus demoversioksikin kutsuttu malli toteutettavasta järjestelmästä on otettu yrityksen käsityksen mukaan hyvin vastaan asiakkaiden keskuudessa, ja on nykyään keskeinen osa FNS:n tekemiä

tarjouksia erityisesti isoissa, useita kuukausia kestävissä projekteissa. Esimerkki luonnoksen liittymisestä tarjoukseen esitetään kuvassa 17.


Tarjouksemme tueksi olemme tehneet järjestelmän Internetissä käytettävästä osuudesta demoversion, johon toivomme kaikkien tämän tarjouksen lukijoiden tutustuvan. Demoversiossa on pyritty kuvaamaan järjestelmän toimintaa eri tehtävissä toimivien loppukäyttäjien näkökulmasta. Demo toimitetaan oheisella cd-levyllä, ja se on nähtävissä Internetissä osoitteessa

http://www.finnishnetsolutions.

Demoversion avaamiseen tarvittavat, yksinomaan tämän tarjouksen vastaanottajien käyttöön tarkoitetut, käyttäjätunnus ja salasana ovat seuraavat:

Tunnus:
Salasana: päivelät

Sivu 2/9

Finnish Net Solutions Oy - Virkkalantie 12-16 B 13 - 08700 VIRKKALA - Y-tunnus 1733917-4 - http://www.finnishnetsolutions.fi/

Kuva 17: Luonnoksen esittely FNS:n tarjouksessa

Asiakasnäkökulmasta tarjoukseen liittyvä luonnos tarkoittaa mahdollisuutta päästä konkreettisesti kokeilemaan ja käsittelemään jonkinlaista ennakkoaavistusta tulossa olevasta järjestelmästä. Luonnokset ovat asiakkaalle mahdollisuus tarkistaa, onko tarjoaja, FNS, ymmärtänyt tavoitteen oikein. Tarjouksissa kerrotaan myös tiiviisti kommentointijärjestelmän käytöstä ja sen toimintaperiaatteesta.

Toteutusnäkökulmasta luonnos tarkoittaa yleensä (web-järjestelmien osalla) kevyen HTML-mallin toteuttamista tai jonkin olemassa olevan ohjelmiston tai ohjelmistokomponentin hyödyntämistä luonnoksen toteutuksessa. Luonnoksen tekeminen antaa toteuttajille myös alustavan kuvan järjestelmän toteutukseen tarvittavasta työmäärästä, joka auttaa taas tarjouksen kokonaisuuksien ymmärtämisessä.

FNS tekee tarjoukset yleensä kiinteähintaisina tarjouksina. Kiinteähintaisuus voidaan tarvittaessa rajata joihinkin osiin tarjousta. Tällöin kiinteähintaisesta tarjouksesta pois rajattu osa tarjotaan yleensä tuntityönä. Tuntityömäärästä annetaan yleensä arvio. Kiinteähintaisuus perustuu lähes poikkeuksetta siihen, että asiakkaalle tarjottava kokonaisuus määritellään keskeisiltä osiltaan luonnoksen ja tarjoustekstin mukaiseksi.

Tiivistetysti tarjousprosessi etenee seuraavin vaihein:

1. Asiakkaalta saadaan tarjouspyyntö
2. FNS kerää asiakkaalta keskeisen saatavissa olevan tiedon ohjelman tarpeista ja vaatimuksista

3. Saatujen tietojen perusteella laaditaan luonnos

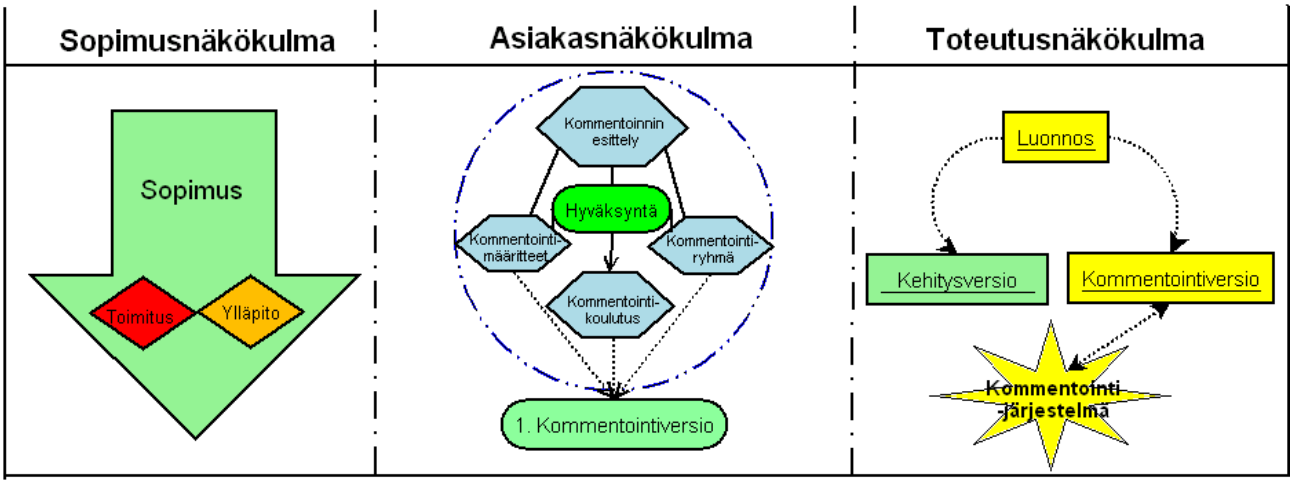
4. Luonnoksen perustella laaditaan tarjous. Tarvittaessa rajataan osa tarjouksesta kiinteähintaisen osan ulkopuolelle.

5. Tarjous ja luonnos toimitetaan asiakkaalle arvioitavaksi

Tarjouksen ja luonnoksen perusteella asiakas arvioi FNS:n soveltuvuutta tarjotun palvelun toimittajaksi. Projekteissa, joissa FNS on toimittanut tarjouksen ohessa luonnoksen ja tullut valituksi, on asiakas usein myöhemmin kertonut demoversion vaikuttaneen merkittävästi valintaan.

3.1.3 Sopimusvaihe

Sopimusvaiheessa toiminta etenee sopimusnäkökulmasta katsottuna sopimusasiakirjojen työstämisellä, yksityiskohdista keskustelemisella ja sopimuksien allekirjoittamisella. Asiakasnäkökulmasta katsottuna valmistellaan asiakkaan kanssa FNS:n kommentointijärjestelmän käyttöönottoa muutamilla valmistelutehtävillä. Toteutuksen osalla valmistaudutaan siirtymään luonnoksesta järjestelmän varsinaiseen toteutusvaiheeseen. Nämä sopimusvaiheen osakokonaisuudet on esitetty graafisesti kuvassa 18, ja kuvataan tarkemmin seuraavissa kappaleissa.



Kuva 18: Sopimusvaihe eri näkökulmista

Sopimusnäkökulmasta sopimusvaihe tarkoittaa asiakkaan tarjoukselle antamaa hyväksyntää ja siitä seuraavaa sopimusta. Hyväksyntä voi usein olla puhelu, sähköposti, keskustelu tai joskus myös virallinen tarjouksen hyväksyntäkirje. Hyväksynnän jälkeen tehdään kirjallinen toimitussopimus asiakkaan kanssa. Toimitussopimuksen yhteydessä solmitaan yleensä myös järjestelmän ylläpitosopimus. FNS käyttää yleensä sopimuksien pohjana tietotekniikka-alan IT2000–sopimusehtoja. Valmiilla sopimuspohjilla sopimuksen

tekoon käytetty aika pyritään minimoimaan. Erityispiirteenä yleisistä tietotekniikka-alan sopimuksista poiketen FNS:n tekemiin sopimuksiin kirjataan näkyviin myös FNS:n kommentointijärjestelmän käyttö siihen liittyvien keskeisten tietojen kanssa.

Asiakasnäkökulmasta yrityksen ohjelmistoprosessissa sopimusvaihe on käytännössä kommentointijärjestelmän käyttämisen hyväksymistä ja varsinaisen määrittelytyön aloittamista. Kommentointijärjestelmän käytön aloittaminen asiakkaan kanssa tarkoittaa yleensä seuraavia vaiheita:

1. Kommentointijärjestelmän esittely
2. Kommentointijärjestelmän käytöstä sopiminen
3. Kommentointimääritteiden asettaminen
4. Kommentointiryhmän kokoonpano
5. Kommentointikoulutuksen/-ohjeistuksen antaminen

Kommentointijärjestelmän esittelyssä esitellään asiakkaan edustajalle kommentointijärjestelmän perusidea (eteneminen toiminto kerrallaan, kommentointiryhmä, keskustelupalstat ja aikarajat). Tähän mennessä kaikki FNS:n asiakkaat ovat pitäneet kommentointijärjestelmän käyttöä hyvänä ideana, ja suostuneet heti sen käyttöön. *Käytöstä sovitaan* merkitsemällä se toimitussopimukseen.

Kommentointimääritteillä tarkoitetaan keskeisiä kommentoinnin toiminnan rajauksia. Keskeiset kommentointimääritteet ovat seuraavat:

1. Kommentointiajan pituus
2. Kommentointiryhmän koko ja edustajat
3. Kommentoitavien toimintojen maksimimäärä
4. Kommentoinnin kokonaiskesto (= projektin aikataulu)

Kommentointimääritteiden jälkeen sovitaan käyttäjäryhmän kokoonpanosta. Asiakas ohjeistetaan kokoamaan mahdollisimman heterogeeninen ryhmä henkilöistä, jotka ovat sitoutuneita ja aidosti kiinnostuneita järjestelmän toteutuksesta. Henkilöille etsitään varahenkilöt sen vuoksi, että pääasiallisen kommentoijan ollessa estynyt projektin eteneminen ei saa pysähtyä; varahenkilön on pystyttävä tarvittaessa osallistumaan kommentointiin. Kommentointiryhmän johtaja ja lopullisten päätösten tekijä on

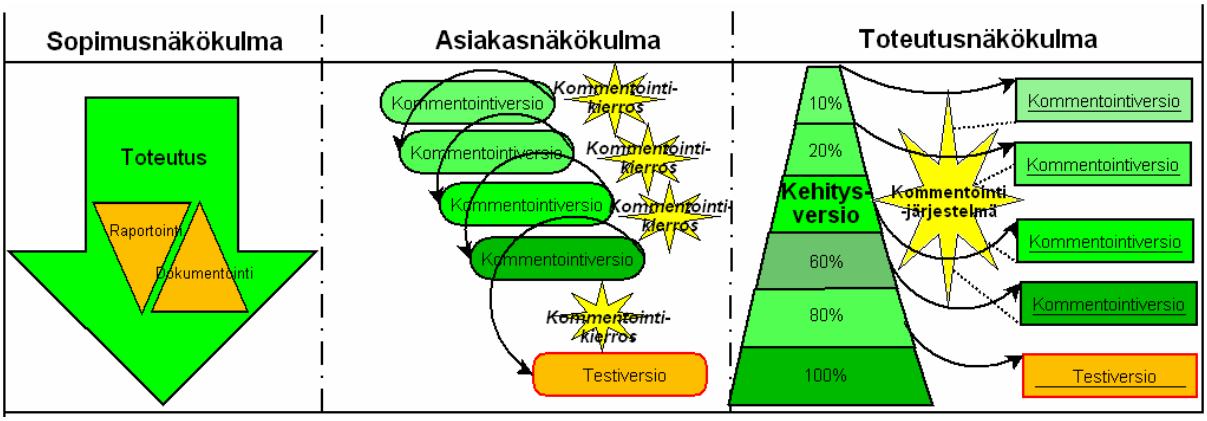
asiakkaan yhteyshenkilö. Kommentointiryhmän kokoamiseen asiakkaan yhteyshenkilö saa tyypillisessä projektissa aikaa noin viikon.

Kommentointikoulutus tai –ohjeistus annetaan asiakkaan koottua kommentointiryhmän. Usein ohjeistus toteutetaan Internetissä, mutta jos on mahdollista, niin koulutus voidaan toteuttaa myös aloitustilaisuudessa, jossa kaikki kommentointiryhmän jäsenet ovat paikalla. Tyypillisesti ohjeistuksessa näytetään videoprojektorilla tai esitetään kirjallisesti kommentoinnin kulku ja kommenttien kirjoittaminen sekä kommentointimahdollisuuden katoaminen kommentointiajan päättyessä. Ohjeistuksen keskeinen osa on vastuun ja kommentointiaikataulun painottaminen kommentoijille, sillä se on tärkeimpiä välineitä sovitun aikataulun pitämiseksi. Esimerkki www-sivulla annetusta ohjeistuksesta on esitetty liitteessä 2.

Toteutusnäkökulmasta sopimusvaihe tarkoittaa, että tarjousvaiheessa tehdystä luonnoksesta aloitetaan työn jatkaminen eteenpäin kohti varsinaista järjestelmää. Jos kommentointi halutaan aloittaa hyvin nopeasti, voidaan ensimmäisenä tai ensimmäisinä kommentointiversioina käyttää luonnoksen ympärille rakennettua kommentointiversiota. Toteuttajien on kuitenkin heti aloitettava varsinaisen järjestelmän suunnittelu ja toteutus. Toteuttajilla työn alla olevaa versiota järjestelmästä kutsutaan *kehitysversioksi.* Kehitysversioon rakennetaan ensimmäiseksi järjestelmän perusrunko, jota kehitetään koko ohjelmiston elinkaaren ajan inkrementaalisesti pieninä palasina kerrallaan. Jos projektin toteutusaikataulu antaa siihen mahdollisuuden, voidaan ensiksi toteuttaa ensimmäinen kehitysversio, josta tehdään ensimmäinen kommentointiversio liittämällä siihen kommentointijärjestelmän toiminnallisuus. Toteutuksellisista näkökohdista ajatellen useimmat FNS:n toteuttamista järjestelmistä ovat käyttäneet perusidealtaan samanlaista kommentointijärjestelmää, useimmiten edellisestä versiosta hieman kehitettynä.

3.1.4 Toteutusvaihe

Toteutusvaiheessa tapahtuu keskeisin osa järjestelmän kehitystyöstä. Vaihe sisältää prosessin merkittävimmän ohjelmointityömäärän lisäksi sekä runsaasti tietojen vaihtoa asiakkaan vastuuhenkilön ja erityisesti kommentointiryhmän kanssa. Toteutusvaihe asiakkaan ja toteuttajien näkökulmasta on hyvin inkrementaalinen. Uusia versioita järjestelmästä julkistetaan tiheään tahtiin. Toteutusvaiheen rakenne eri näkökulmista esitetään graafisesti kuvassa 19.



Kuva 19: Toteutusvaiheen eri näkökulmista

Sopimusnäkökulmasta toteutusvaihe tarkoittaa sopimusvaiheessa tehdyn sopimuksen käytännön toteuttamista. Sopimusasioihin voidaan katsoa kuuluvan myös toteutusvaiheen raportointi asiakkaan sopimuksen allekirjoittajille. Tämä voi tarkoittaa esimerkiksi sähköpostitse lähetettäviä tiettyä muotoa noudattavia tai vapaamuotoisia yhteenvetoja, joita lähetetään sovitulle vastaanottajajoukolle. Usein se tarkoittaa vain sähköpostiviestiä tai tekstiviestiä, että allekirjoittajien kannattaa käydä katsomassa Internetissä työn edistymistä. Kommunikointia pyritään kuitenkin siirtämään muodollisesta sopimusviestinnästä mahdollisimman paljon asiakastasolle kommentointijärjestelmää hyödyntämällä. Kaikki viestintä, joka liittyy konkreettisiin järjestelmän toimintoihin, pyritään keskittämään kommentointijärjestelmään. Sopimusasioihin voidaan laskea myös kuuluvan sovitun dokumentaation tuottaminen. Vähimmillään tämä tarkoittaa muutaman sivun pikaohjetta järjestelmän käyttöön, joskus tarkempia teknisempiä dokumentteja kuten esimerkiksi asennusohjeita, toteutusdokumentteja sekä ylläpito-ohjeita.

Asiakasnäkökulmasta toteutusvaihe tarkoittaa järjestelmän kehitystä versio versiolta kehittyvien kommentointiversioiden avulla. Toteutusvaiheen alettua asiakkaalla on jo yleensä sopimusvaiheen aikana ollut käytössään ensimmäinen (usein tarjouksen liitteenä olleeseen demoon perustuva) kommentointiversio, joka on esitellyt ensimmäisiä näkemyksiä järjestelmästä. Toteutusvaiheessa jatketaan työtä demon ja ensimmäisen kommentointiversion avulla saatujen kommenttien perusteella. Asiakkaalle julkaistaan sovituin määräajoin, tyypillisesti 1-4 viikon välein uusi kommentointiversio. Jokainen uusi kommentointiversio pyrkii olemaan kehittyneempi kuin edeltäjänsä. Uuden *kommentointiversion julkaisuehdot* voidaan tiivistää seuraaviin tavoitteisiin:

1. Uusi versio muuttaa edellisellä kommentointikierroksella muutettavaksi sovitut tiedot tai toiminnot

2. Uusi versio täydentää edellisessä kommentoinnissa puuttuvaksi ilmoitetut tiedot tai toiminnot

3. Uudessa versiossa on toteutettu toimiviksi edellisellä kommentointikierroksella hyväksyttyjä käyttöliittymän osia

4. Uusi versio esittelee lisää käyttöliittymiä kommentointiryhmälle

Kohdat yksi ja kaksi tarkoittavat, että kunkin kierroksen kommentointiajan päätyttyä ja asiakkaan yhteyshenkilön annettua lopullisen vahvistuksen muutos- ja laajennusehdotuksista FNS toteuttaa muutokset ja täydennystarpeet. Tyypillinen esimerkki muutostarpeesta on jonkin käyttöliittymässä näkyvän kentän uudelleen nimeäminen (esim. ”henkilön nimi” → ”henkilön tai yrityksen nimi”), kun taas täydennystarve voi tarkoittaa esimerkiksi lisätietokenttiä tietokantaan (esim. yhden puhelinnumeron sijasta tallennetaan matkapuhelinnumero, työpuhelinnumero ja kotipuhelinnumero).

Kolmas kohta tarkoittaa käyttöliittymämalleista todelliseen toiminnon toteutukseen siirtymistä. Käytännössä tämä tarkoittaa tietokannan ja sovelluslogiikan rakentamista käyttäen määrittelynä käyttöliittymämallia. Toteutuksen tekeminen käyttöliittymämallia määrittelynä käyttäen edellyttää, että käyttöliittymämalli on riittävän tarkka ja kuvaa kaikki tarvittavat käyttötapaukset. Päätös siitä, onko kommentoitu käyttöliittymä jo toteuttamiskunnossa, voidaan muotoilla seuraavaksi säännöksi:

> ”Jos toiminnon tallentamaa tietoa ja tarvittavaa toiminnallisuutta ei voida hahmottaa toteutukseen tarvittavalla tarkkuudella käyttöliittymämallin perusteella, on mallia tarkennettava ja se on asetettava uudelleen kommentoitavaksi.”

Edellistä sääntöä voidaan kommentointimenettelyä käytettäessä rikkoa harvoin. Säännöstä voidaan kuitenkin poiketa esimerkiksi silloin, jos toteuttajat arvioivat pystyvänsä toteuttamaan tulevat muutokset ajassa, joka on paljon lyhyempi kuin uuden käyttöliittymämallin tekeminen tai nykyisen muuttaminen. Toteutusajan on oltava merkittävästi lyhyempi sen vuoksi, että tällöin otetaan riski erehtymisestä, sillä asiakas voi kuitenkin haluta muuttaa toimintoa oletuksesta poikkeavaksi.

Hyväksyttyjen käyttöliittymämallien pohjalta toteutetaan toiminto ja tarvittavat tietorakenteet. Käyttöliittymämallien tulee kuvata toiminto esimerkkinäyttöjen avulla niin hyvin kuin mahdollista, jotta varsinaisessa toteutuksessa toiminto voidaan toteuttaa samannäköiseksi kuin käyttöliittymämalli on ollut. Tällöin valmista toimintoa ei enää seuraavalla kommentointikierroksella aseteta kommentoitavaksi, vaan sen ilmoitetaan

olevan valmis. Jos kuitenkin toteutuksessa on jouduttu poikkeamaan esitetystä mallista merkittävästi, voidaan valmiskin toiminto asettaa vielä kommentoitavaksi ja asiakkaan hyväksyttäväksi.

Neljäs kohta kuvaa inkrementaalisuuden perusperiaatetta. Edellisissä versioissa esiteltyjen toimintojen käyttöliittymämallien täydentämisen sekä malleista toteutukseen siirtymisen lisäksi jokaisessa uudessa kommentointiversiossa tulee esitellä joitakin kokonaan uusia toimintoja. Esiteltävien uusien toimintojen määrä kommentointiversiota kohden riippuu järjestelmän laajuudesta ja aikataulusta. Kireä aikataulu tai suuri järjestelmä vaatii useampien uusien toimintojen esittelyä jokaisella kommentointikierroksella. Tyypillisesti toteutetuissa projekteissa on esitelty 1-5 uutta toimintoa jokaisella kommentointikierroksella.

Kommentointiversioiden määrä vaihtelee ohjelmistoprojektin koosta ja aikataulusta riippuen merkittävästi. Kuvassa 19 on esitetty asiakastasolla neljä kommentointiversiota, mutta todellisuudessa kommentointikerrosten määrä on yleensä suurempi. Tähänastisissa kommentointijärjestelmää hyödyntäneissä projekteissa kommentointikierrosten määrät ovat vaihdelleet n. 15 kierroksesta useisiin kymmeniin kierroksiin. Kommentointiversioiden määrään vaikuttaa myös kommentointikierrosten kesto: jos kommentointikierroksia on paljon, on kommentointiajan kierrosta kohden oltava lyhyt.

Asiakkaan näkökulmasta kommentointivaihe voidaan katsoa päättyneeksi silloin, kun kommentointiryhmällä ei ole enää muutosehdotuksia toimintoihin, eikä kommentointiin ole enää tulossa uusia toimintoja. Kommentoinnin jälkeen järjestelmästä julkaistaan toteutusvaiheen päätteeksi vielä niin kutsuttu *testiversio*, joka esittelee varsinaisen käyttöönottoa odottavan järjestelmän. Yleensä tässä vaiheessa järjestelmästä poistetaan kommentointijärjestelmän toiminnot, jolloin tämä vastaa todellista käyttöön otettavaa järjestelmää. Testiversiota käytetään myös usein myös käyttökoulutustarkoituksiin. Testiversioon liittyy usein myös testitiedon luominen tai lataaminen vanhoista järjestelmistä. Jos testitietoa luodaan, niin sen kirjaaminen järjestelmään voi usein tapahtua myös asiakkaan toimesta, jolloin järjestelmään kirjataan sisään todellisuutta tietomuodoiltaan vastaavia, mutta kuvitteellisia tietoja.

Toteutusnäkökulmasta toteutusvaihe tarkoittaa käytännössä toteutustyön tekijöille, suunnittelijoille ja ohjelmoijille, jatkuvaa kehitystyötä *kehitysversion* parissa. FNS:ssä kehitysversio mielletään yhdeksi jatkuvasti kehittyväksi versioksi ohjelmistosta. Siihen rakennetaan uusia toimintoja toteuttamalla ensin käyttöliittymämalli, jonka jälkeen

tietorakenne ja toiminnallisuus. Kehityksessä noudatetaan aina kommentointiversiossa näkyvää ”ensin aina käyttöliittymä” –periaatetta.

Toteutuksen näkökulmasta järjestelmän kommentointiversio tarkoittaa tietyllä, sopivalla hetkellä otettua kopiota kehitysversiosta, johon on liitetty kommentointijärjestelmä. Kommentointijärjestelmän tekniikka on käytännössä mukana myös kehitysversiossa, jolloin kommentointiversio saadaan tehtyä kehitysversiosta ottamalla kehitysversiosta kopio ja liittämällä tähän kopioon tuorein kommenttitietokanta. Lisäksi kommentointiversion avauksen yhteydessä avataan edellisen version jälkeen tehtyihin uusiin käyttöliittymämalleihin liittyvät kommentointipalstat sekä kirjoitetaan näille palstoille toimintoihin liittyvät avauskommentit (esim. ”Asiakasrekisterin luonnos on nyt kommentoinnissa, kertokaa mielipiteenne.”). Vanhoihin, vielä kommentoinnissa oleviin kommentointeihin liittyen kirjataan kommentointipalstoille näkyviin tehdyt muutokset (esim. ”Pyydetyt kentät puhelinnumeroille on lisätty. Automaattinen sähköpostiosoitteen muodon tarkistus on lisätty. Onko toiminto nyt ok toteutusta varten?”). Tarvittaessa kommentointiaikoja jatketaan uusien kommentointikierrosten mukaan.

Toteuttajille kommentointiversiot ovat tiedonlähteitä: suunnittelijat ja ohjelmoijat seuraavat kommentointipalstoja tiiviisti, ja osallistuvat myös keskusteluun kommentointiaikana aina tarvittaessa. Tyypillinen teknisen toteuttajan tarve osallistua keskusteluun on jonkin kommentoijan kirjoittama kysymys johonkin tekniseen yksityiskohtaan liittyen (Esim. ”Voisiko tähän kohtaan tallentaa liitteenä myös .jpg –kuvatiedostoja?”).

Konkreettinen toteuttajien työnteko joka jakautuu yleensä neljään päätyyppiin:

- Uusien käyttöliittymämallien suunnittelu ja toteutus
  (uudet, ennen kommentoimattomat toiminnot)
- Entisten käyttöliittymämallien parannus kommentoinnin perusteella
- Toimintojen tekninen suunnittelu ja toteutus (ohjelmointityö) käyttöliittymämallien pohjalta
- Kommentointiversioiden julkaisu

Uusien käyttöliittymämallien suunnittelu tapahtuu yhteistyössä asiakkaan yhteyshenkilön kanssa. Ensimmäisen käyttöliittymämallin suunnittelu tehdään joillakin suunnittelijoille ja ohjelmoijille annetuilla perustiedoilla. Nämä tiedot on voitu saada esimerkiksi sähköpostiviestissä, mallikuvana, entisen järjestelmän osana tai vaikkapa paperille piirrettynä käyttöliittymämallina. Tämä ensimmäisen hahmotelma siirretään heti

käyttöliittymäluonnokseksi, joka tyypillisesti toteutetaan HTML-koodia ja Javascript–ohjelmointia hyödyntäen. Entisten käyttöliittymämallien parantaminen tapahtuu aina edellisen kommentointikierrokseen palautteen mukaisesti. Esitetyt toiveet muokataan käyttöliittymämalliin ja esitellään seuraavalla kommentointikierroksella.

Toimintojen ja tietorakenteiden varsinainen toteutustyö on tavanomaista ohjelmointityötä. Tietokannat tai muut tietorakenteet suunnitellaan yleensä ensin käyttöliittymämallien antamien tietojen perusteella. Apuna käytetään myös tietokantakaavioita, jotka piirretään tietokoneohjelmilla (esim. Dia-ohjelma). Tietorakenteiden toteuttamisen jälkeen käyttöliittymämallia aletaan muokata varsinaiseksi, toimivaksi järjestelmän osaksi. Ohjelmointityössä käytetään yleensä PHP-ohjelmointikieltä. Ohjelmoinnissa käytetään hyödyksi edellisten ohjelmistoprojektien soveltuvia moduuleita sekä yrityksen omia komponenttikirjastoja. Erityisen paljon hyödynnetään Internetissä saatavilla olevaa aineistoa. Internetin Open Source–ohjelmistot, hakukoneet, erilaiset ohjelmistokehittäjien keskustelupalstat sekä työkalujen valmistajien tarjoamat dokumentaatiot sekä valmistajien kotisivut ovat jatkuvasti käytettäviä lähteitä. Tietolähteenä ei ole syytä unohtaa myöskään keskeisessä roolissa olevaa tekijöiden välistä keskustelua. FNS:n työyhteisö on tiivis. Työn teko tapahtuu pääosin avokonttorissa, missä kaikki henkilöt ovat samassa tilassa ja jossa yhteydenpito keskustellen on helppoa ja välitöntä.

Uusien toimintojen julkaisussa kommentointiversioon voidaan noudattaa kahta, pienellä sävyerolla toisistaan poikkeavaa tapaa: versioitu kommentointi ja inkrementaalinen kommentointi

Versioidussa kommentoinnissa kommentointikierroksen päätyttyä pidetään pieni tauko kommentoinnissa. Tauon aikana toteuttajat tekevät muutokset käyttöliittymiin, jonka jälkeen julkaistaan muutettu sekä mahdollisia uusia toimintoja sisältävä kommentointiversio. Tauko kestää muutosten vaativuudesta riippuen yleensä 1-5 päivää. Edellisessä kommentoinnissa valmiiksi todettuja käyttöliittymämalleja toteutetaan todellisiksi toiminnoiksi kommentointivaiheen aikana.

Inkrementaalinen kommentointi tarkoittaa menettelyä, jossa kehittäjät päivittävät kommentointimalleja jatkuvasti; aina kun uusi käyttöliittymämalli on valmis, se asennetaan välittömästi kommentointiversioon. Tätä kommentointitapaa käytettäessä ei eri kommentointiversioita eroteta selkeästi toisistaan, vaan kommentointi muodostaa yhtenäisen, jatkuvasti kehittyvän järjestelmän. Tätä kommentointitapaa käytetään silloin, kun projektin aikataulu on kireä, ja kommentointiryhmä voi osallistua kommentointiin erittäin nopealla aikataululla. Uusia käyttöliittymämalleja voidaan tällöin

päivittää kommentointiversioon jopa useita kertoja päivässä. Tällöin myös kommentointiryhmän tulee olla riittävän pieni. Usein Internetissä toimivaa kommentointijärjestelmää täydennetään tällöin myös puhelinkeskusteluilla kommentointiryhmän jäsenten kanssa riittävän nopeuden saavuttamiseksi.

Kommentointivaiheen viimeinen vaihe, kommentointiversiosta testiversioon siirtyminen tarkoittaa toteutustasolla keskeisimmillään yhtä asiaa: testiversio on valmiiksi todettu versio, jossa ei enää ole kommentointimahdollisuutta. Testiversio vastaa todellista järjestelmää, joka ei enää ole (kehitysvaiheen) kommentoinnin kohteena. Testiversioon liittyy usein myös testitiedon hankkiminen järjestelmään. Tämä voi tarkoittaa tietojen lataamista vanhasta järjestelmästä tai muusta lähteestä tai toisaalta kuvitteellisen tiedon. Testiversiot voidaan testitiedon sisällön perusteella luokitella kahteen kategoriaan: testitiedoilla toimiva testiversio ja tuotantotiedoilla toimiva testiversio

Testitiedoilla toimiva testiversio tarkoittaa, että järjestelmään syötetään kuvitteellisia, mutta rakenteeltaan todellisuutta vastaavia tietoja (esim. ”Matti ja Maija Meikäläinen”, ”Testitie 6”, ”99999 KORVATUNTURI”). Usein tällaisten testitietojen syöttäminen sovitaan asiakkaan tehtäväksi. Testitietojen syötön tekeminen asiakkaan toimesta tarjoaa mahdollisuuden testata todellista järjestelmää ja varmistaa järjestelmän valmiutta käyttöönottoon. Testitietojen syötöllä on usein myös koulutuksellinen merkitys, sillä jos testitiedon syöttää järjestelmään joku järjestelmän tulevista käyttäjistä, hän saa runsaasti harjoitusta tulevaa käyttöönottoa ajatellen. Testitiedoilla toimivien testiversioiden hyödyntäminen on usein mahdollista myös järjestelmän käyttöönoton jälkeen. Tällaista järjestelmää voidaan käyttää uusien, paranneltujen ominaisuuksien testauksessa.

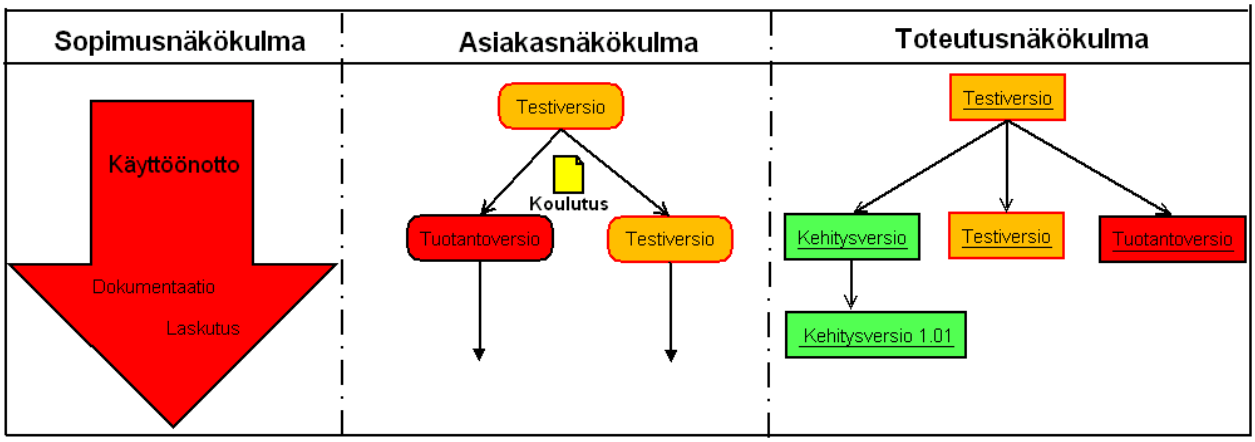
Tuotantotiedoilla toimivaan testiversioon ladataan tiedot yleensä vanhasta tietojärjestelmästä tai muusta olemassa olevasta tietovarastosta, kuten Excel- tai tekstitiedostosta. Tuotantotiedoilla toimivan testiversion toteutus on mahdollista yleensä silloin, kun järjestelmää toteutetaan korvaamaan vanha, olemassa oleva järjestelmä. Tietojen lataamiseen uuteen järjestelmään liittyy yleensä keskeisenä osana tietojen konvertointi uuteen järjestelmään sopivaksi. Jos testiversio tehdään tuotantotiedoilla, valmistaudutaan samalla käyttöönottoon. Samalla tulee testatuksi myös järjestelmän kyky käsitellä tarvittavan suuruisia tietomääriä.

Testitiedoilla ja tuotantotiedoilla toimivilla testiversioilla on molemmilla hyvät ja huonot puolensa. Tuotantotiedoilla toimivaa järjestelmää ei esimerkiksi voi esitellä asiakkaan ulkopuolisille toimijoille. Testitiedoilla toimiva järjestelmä ei taas välttämättä anna käsitystä toteutetun järjestelmän suorituskyvystä, jolloin suorituskykymittaukset pitää

tehdä toisella tavalla. Siksi toisinaan päädytäänkin tekemään molemmat testiversiot. Näin tapahtuu erityisesti projekteissa, joissa kehitettävä järjestelmä korvaa vanhan, aiemmin käytössä olleen järjestelmän.

3.1.5 Käyttöönottovaihe

Käyttöönottovaiheessa valmis järjestelmä otetaan käyttöön asiakkaalla. Jos kyse on julkisesta järjestelmästä, liittyy tähän vaiheeseen usein myös julkistaminen, johon kuuluu palvelusta tiedottaminen asiakkaan yhteistyökumppaneille tai laajemmin tiedotusvälineille. Käyttöönotto on ohjelmiston varsinaisen kehitysprosessin viimeinen vaihe, jonka jälkeen siirrytään ylläpitovaiheeseen. Vaihe on kuvattu eri näkökulmista kuvassa 20.



Kuva 20: Käyttöönottovaihe eri näkökulmista

Sopimusnäkökulmasta käyttöönottovaihe tarkoittaa alkuperäisen työn, ohjelmiston toimituksen, valmistumista. Käyttöönotossa ohjelmiston toimitus todetaan toteutetuksi yhteisesti tilaajan ja toimittajan välillä (tarvittaessa kirjallisesti), jonka jälkeen FNS laskuttaa ohjelmiston lopputoimituksen asiakkaalta. Käyttöönotto tarkoittaa myös ylläpitosopimuksen voimaanastumista (ja tarvittaessa tekemistä, mikäli sopimusta ei ole tehty toimitussopimuksen yhteydessä). Ohjelman kehityksestä ylläpitoon siirtyminen tarkoittaa, että jatkossa ohjelmistoon tehtävät muutokset ja ominaisuudet eivät ole enää osa alkuperäistä ohjelmiston kehitystyötä vaan jatkokehitystä. Jatkokehitystyö hinnoitellaan yleensä ylläpitosopimuksessa erikseen. Tyypillisesti laskutusperusteena on tuntihinta. Laajemmissa ohjelmistoprojekteissa siirrytään käytännössä välittömästi käyttöönoton jälkeen – tai jopa sen aikana – ylläpitotyöhön. Rajaa alkuperäisen ohjelmiston ja muutostöiden välille on hankala vetää, sillä määritteet ja tarpeet ohjelmistolta muuttuvat usein jatkuvasti. Tärkeintä on, että asiakas ja toimittaja ovat yhtä mieltä alkuperäisen toimitussopimuksen määrittelyiden riittävästä täyttymisestä, jolloin ylläpitovaiheeseen voidaan siirtyä.

Asiakasnäkökulmasta käyttöönotto tarkoittaa kehitetyn järjestelmän saamista konkreettisesti hyötykäyttöön. Riippuen siitä, korvaako kehitetty järjestelmä vanhan ohjelman, vai tuleeko se täyttämään jotakin kokonaan uutta tiedonkäsittelytarvetta, on käyttöönottotilanne yleensä merkittävästi erilainen. Kokonaan uuden järjestelmän tapauksessa käyttöönotto voidaan usein aloittaa rauhallisesti aloittamalla pienten tietomäärien syötöllä kerrallaan. Korvattaessa vanha tietojärjestelmä uudella käyttöönoton tulee yleensä tapahtua hyvin lyhyen ajanjakson aikana. Tilanne on tällainen etenkin silloin, jos tietoja ei voida saada käsiteltäväksi sekä uuden että vanhan järjestelmän kautta. Käyttöönottoon liittyvät myös usein käyttökoulutukset. Järjestelmän laajuuden ja käyttäjien tarpeen mukaan järjestetään ennen järjestelmän käyttöönottoa käyttökoulutus sekä tehdään käyttökoulutusmateriaalit.

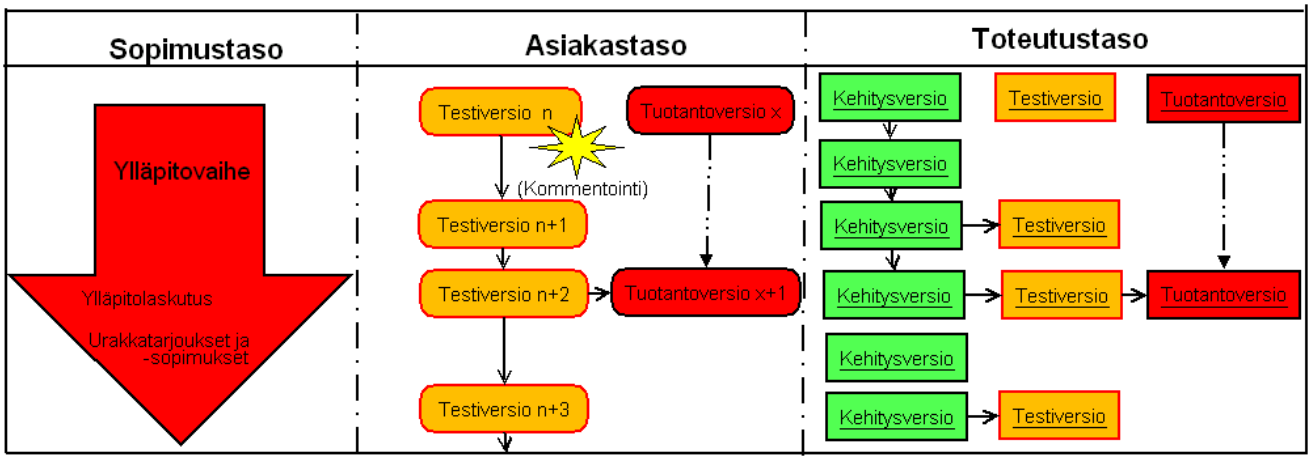
Käyttöönotto tarkoittaa asiakastasolla myös kommentointivaiheen päätteeksi valmistuneen testiversion jakaantumista kahdeksi versioksi: todelliseksi järjestelmäksi sekä testiversioksi. Todellisen, tuotantotiedot sisältävän järjestelmän lisäksi FNS jättää asiakkaan käyttöön yleensä myös testiversion, jota usein kutsutaan myös *harjoitteluversioksi*. Testiversio palvelee asiakkaita harjoittelu- ja koulutustarkoituksissa, ja ylläpitotyössä tätä testiversiota hyödynnetään myös uusien ominaisuuksien esittelypaikkana. Testiversiolla tarjotaan asiakkaalle mahdollisuus tutustua vapaasti järjestelmän toimintaan ilman pelkoa tuotantotietojen vahingoittumisesta.

Toteutusnäkökulmasta käyttöönottovaihe tarkoittaa kahdesta versiosta kolmeen siirtymistä, jolloin kehitys- ja testiversion lisäksi mukaan tulee myös järjestelmän tuotantoversio. Toimintojen osalta tuotantoversio vastaa käyttöönottovaiheessa täydellisesti testiversiota, jolloin ainoastaan tietosisältö on erilainen. Mahdollisen vanhan tiedon siirto tuotantoversioon liittyy toteutustasolla keskeisenä osana käyttöönottovaihetta. Tiedon siirtoon liittyy usein erilaisia formaattimuunnoksia, jotka voivat muodostaa merkittävän osan toteuttajien työstä. Toteutusvaiheessa siirrytään yleensä samalla aikaa myös ylläpitovaiheeseen, ja kehitysversio jää edelleen toteuttajien käyttöön jatkokehitystä silmällä pitäen. Usein uusia toimintoja aletaan valmistella kehitysversioon jo heti käyttöönoton yhteydessä. Tätä kuvaa kuvassa 20 esitetty kehitysversio 1.01.

3.1.6 Ylläpito

Ylläpitovaihe liittyy keskeisenä osana käytännöllisesti katsoen kaikkiin yrityksen toteuttamiin ohjelmistoihin. Ajan myötä merkittävin tietojärjestelmän kehittyminen tapahtuu nimenomaan ylläpitovaiheen aikana, jolloin alkuperäinen järjestelmä toimii vain alkusysäyksenä kehitystyölle. Ylläpitovaiheessa kehitystyö jatkuu ylläpitosopimuksessa määritellyllä tavalla. Ylläpitotyössä käytetään apuna testiversiota

sekä tarvittaessa myös kommentointijärjestelmää vastaavaan tapaan kuin varsinaisessa järjestelmän kehitystyössäkin. Ylläpitovaiheen eteneminen on esitetty kaaviona kuvassa 21.



Kuva 21: Ylläpitovaihe eri näkökulmista

Sopimusnäkökulmasta ylläpitovaihe on yleensä hyvin selkeää. Tavanomaisesti kehitystyötä tehdään asiakkaan kanssa jatkuvasti, mutta määrät vaihtelevat aika-ajoin. Tyypillisesti FNS laskuttaa asiakkaita muutostöistä aikaperusteisesti (tuntihinta). Työt ja niihin käytetty aika kootaan luetteloon (esimerkiksi Excel-tiedostoon) ja ne laskutetaan kahden tai kolmen kuukauden välein asiakkaalta. Jos järjestelmään toteutetaan yhdellä kertaa normaalia ylläpitotyötä suurempia muutoksia (tyypillisesti yli viikon työpanoksen vaativa muutos), voidaan asiakkaan kanssa tehdä muutoksista erillinen urakkatarjous. Tyypillisesti erilliset tarjoukset ovat kiinteähintaisia tai niiden kustannusten sovitaan mahtuvan johonkin ennalta määritettyyn vaihteluväliin.

Muuttuvien kustannusten lisäksi sopimustasolle liittyy FNS:n kuukausittainen ylläpitomaksu palvelinpalveluista sekä sovitusta palvelutasosta. Palvelutasolla tarkoitetaan, että FNS sitoutuu ylläpitosopimuksessa toteuttamaan tiettyjä asioita sovituilla vasteajoilla. Vasteaikoja määritetään esimerkiksi siten, että tietyn kokoluokan muutokset toteutetaan asiakkaalle jollakin sovitulla vasteajalla, esimerkiksi kolmen vuorokauden kuluessa tilauksesta. Vasteajat määritetään myös vikatilanteiden korjaukseen.

Asiakasnäkökulmasta ylläpitovaihe tarkoittaa käytössä olevan järjestelmän toiminnassa pitämistä sekä aina tarvittaessa myös kehittämistä ja muuttamista. Järjestelmän kehitysvaiheen lopusta ja käyttöönotosta tuttu testiversio toimii apuvälineenä asiakkaalle myös ylläpitovaiheessa. Asiakkaan toivoessa jotakin muutosta järjestelmään, esitetään muutokset tyypillisesti aina ensin järjestelmän testiversiossa, josta ne siirretään sitten

tuotantoversioon. Tarvittaessa käytetään useita testiversioita, jos asiakas ehdottaa testiversion nähtyään vielä merkittäviä muutoksia toimintoihin. Kun muutokset on esitetty asiakkaalle testiversiossa ja asiakas on hyväksynyt ne, siirretään ne tuotantoversioon. Kuvassa 21 esitetään tämä vaiheittainen eteneminen nuolilla eri versioiden välillä. Katkoviivanuoli tuotantoversioiden välillä kuvaa tietosisällön siirtoa. Tietosisältö pysyy muuttumattomana edellisestä tuotantoversiosta siirryttäessä, ainoastaan toiminnot siirretään testiversiosta.

Myös ylläpitovaiheessa käytetään vähänkin suurempien muutostöiden (yli yhden-kahden työpäivän työpanos) tapauksessa käyttöliittymämalleja, jotka esitellään ensin testiversiossa, ennen kuin varsinaista tekniikkaa aletaan toteuttamaan. Jos muutoksien hyväksyntään osallistuu useita henkilöitä, tai muutokset ovat muutoin laajoja, on myös ylläpitovaiheessa mahdollista käyttää kommentointijärjestelmää eri testiversioiden välillä vastaavasti kuten kehitysvaiheessakin. Tähän ei kuitenkaan ole yleensä ollut tarvetta, sillä muutoksien kommentointiin on tähän mennessä osallistunut yleensä vain muutamia henkilöitä.

Toteutusnäkökulmasta katsottuna ylläpitovaiheen suunnittelu- ja ohjelmointityö etenee hyvin vastaavasti kehitysversiota kehittämällä kuin varsinaisen projektin toteutusvaiheessakin. Kun muutokset ovat edenneet riittävän pitkälti tai kun käyttöliittymämallit ovat valmiit (tai edellisestä testiversiosta päivitetyt), julkaistaan kehitysversiosta uusi testiversio. Asiakas antaa kommenttinsa testiversion perusteella, ja kommenttien sisällöstä riippuen muutokset joko julkaistaan tuotantoversioon tai kehitystyötä jatketaan edelleen. Periaatteena on, että kaikki vähänkin merkittävämmät muutokset, jotka eivät ole täysin itsestään selviä, kierrätetään testiversion kautta asiakkaan hyväksynnän saamiseksi.

Toteuttajien näkökulmasta haasteena ylläpitovaiheen työssä on keskeneräisen, kehitettävänä olevan järjestelmän toteuttamiseen verrattuna operatiivisen järjestelmän päivittäminen. Se vaatii tarkkuutta ja sopivia työvälineitä. Muutoksia paikalleen asennettaessa on varauduttava aina mahdollisiin ongelmatilanteisiin; varmuuskopioinnin on oltava kunnossa ja vanhan version palautus mahdollisessa ongelmatilanteessa (roll-back) on onnistuttava nopeasti. FNS pyrkii rakentamaan järjestelmät siten, että useimmat muutokset voitaisiin asentaa operatiivisten järjestelmien käytössä ollessa. Aina tämä ei ole mahdollista, mutta usein pyritään sulkemaan ainoastaan päivityksen kohteena oleva järjestelmän osa, ei koko järjestelmää.

Yleisesti ylläpitovaiheesta voidaan sanoa, että FNS korostaa ylläpitotyössä reilua ja avointa yhteistyötä asiakkaan kanssa. Asiakkaalle tehdään ehdotuksia uusista, matkan

varrella mahdollisesti tarpeelliseksi huomattujen toimintojen toteuttamisesta sekä puhutaan avoimesti ongelmallisista päivitys- tai kehityskohteista. Tiheän testiversioiden päivitystahdin myötä muutostöiden vaihe pyritään pitämään myös asiakkaan edustajien mielessä.

3.2 FNS:n esimerkkiprojekti: Sikaloiden terveysluokitusrekisteri

FNS toteutti vuonna 2003 (29.4.2003 – 5.11.2003) Eläintautien torjuntayhdistys ETT ry:lle Suomen sikaloiden terveysluokitusrekisterin. Rekisterin tilaajana toimi ETT ry taustaorganisaatioineen. Taustaorganisaatioista keskeisimpiä terveysluokitusrekisterin kannalta ovat teurastamot. ETT:n toiminnassa mukana ovat lähes kaikki Suomen teurastamot.

Terveysluokitusrekisteri on tietojärjestelmä, jonka keskeistä sisältöä ovat sikatilojen ja eläinlääkäreiden väliset terveydenhuoltosopimukset, eläinlääkäreiden tiloilla tekemien terveydenhuoltokäyntien käyntitiedot sekä tilojen eläimiin liittyvät laboratoriotulokset (Etu-palvelut 2006). Järjestelmää hyödynnetään tilojen terveydenhuoltotietojen keräämiseen sekä tilojen terveysluokittamiseen näiden tietojen pohjalta. Terveydenhuoltotilanteensa mukaan tilat luokitetaan joko kansalliselle tasolle tai perusluokkaan (lakisääteinen taso). Teurastamot maksavat kansallisella tasolla oleville tuottajille tyypillisesti hieman korkeampaa hintaa lihasta kuin perusluokassa oleville tiloille. Ennen Etu-palveluiden tietojärjestelmää jokainen teurastamo hoiti tilojen terveysluokitusta itsenäisesti omien lihantuottajiensa keskuudessa. Yhtenäinen järjestelmä perustettiin tehostamaan ja yhtenäistämään tilojen terveydenhuoltotoimintaa sekä säästämään toiminnan kustannuksia.

Terveysluokitusrekisterillä on yli 10 erilaista käyttäjäryhmää: järjestelmää käyttävät sikatilojen, eläinlääkäreiden ja teurastamoiden käyttäjien lisäksi erilaiset muut sidosryhmät, kuten läänineläinlääkärit viranomaisten edustajina. Järjestelmässä oli yli 5300 erilaista käyttäjätunnusta syyskuussa 2006. Terveydenhuoltokäyntejä samana ajankohtana oli tallennettu 29317 kappaletta (Etu-palveluiden tietojärjestelmä 2006). Järjestelmä on käyttöönotostaan alkaen toiminut hyvin ja sopinut täysin tehtäväänsä. Järjestelmää jatkokehitetään FNS:n toimesta tiiviissä yhteistyössä ETT ry:n alaisuudessa toimivan Etu-palvelut-yksikön kanssa.

Terveysluokitusrekisteri oli FNS:n ensimmäinen ohjelmistokehitysprojekti, jossa käytettiin edellä kuvatun kaltaista kommentointijärjestelmää. Kommentointijärjestelmän perusajatus kehitettiin mahdollistamaan projektille määritetty erittäin kireä aikataulu. FNS:n esitettyä ajatuksen kommentointijärjestelmän käytöstä, haluttiin se asiakkaan puolella ottaa hyvin nopeasti käyttöön. Edellytyksenä tähän oli motivoituneiden ja

ennakkoluulottomien asiakkaan edustajien sekä projektia vetäneen ELT Veikko Tuovisen kiinnostus uutta, tehokkaalta kuulostavaa järjestelmää kohtaan.

Sikaloiden terveysluokitusrekisteri toimii Internetissä osoitteessa www.etu-palvelut.net. Järjestelmän harjoitteluversio on julkisesti kaikkien kiinnostuneiden käytettävissä.

3.3 FNS:n menetelmä vertailussa muihin ohjelmistokehitysmenetelmiin

FNS:n menetelmästä voidaan havaita monia yhteneväisiä piirteitä muihin, tämän työn alussa esitettyihin kehitysmenetelmiin. Samoin on löydettävissä muutamia hyvin keskeisiä eroavaisuuksia.

Aluksi vertaillaan FNS:n menetelmän ominaisuuksia ketterien kehitysmenetelmien tyypillisiin piirteisiin, joita Abrahamsson (2002) esittelee:

1. Kehitysprosessi on modulaarinen
2. Iteratiivisuus lyhyillä sykleillä mahdollistaa varmistukset ja korjaukset
3. Aikasidonnaisuus sykleissä: syklin kesto yhdestä kuuteen viikkoa
4. Nuukuus kehitysprosessissa poistaa kaikki turhat aktiviteetit
5. Sopeutuvuus myöhemmin esiin tuleviin riskeihin
6. Inkrementaalinen lähestyminen kehitysprosessiin mahdollistaa toimivan sovelluksen rakentamisen pienissä askelissa
7. Yhteenliittävä (ja inkrementaalinen) lähestymistapa minimoi riskit
8. Henkilöorientoituneisuus: ketterät prosessit arvostavat ihmistä enemmän kuin prosesseja ja teknologiaa
9. Yhteistyökykyinen ja yhteisöllinen työntekotyyli

FNS:n menetelmää voidaan hyvin pitää modulaarisena, sillä jokainen vaihe voidaan ajatella omana moduulinaan. Myös jokainen kommentointikierros tai testiversion toteutus on oma moduulinsa, joita voidaan toistaa tarvittava kierrosmäärä. Toinen peruspiirre, kehityksen iteratiivinen eteneminen lyhytkestoisten syklien avulla, on selkeästi sama: nämä syklit on FNS:n menetelmässä nimetty kommentointikierroksiksi, ja niille on asetettu hyvin tiukat aikarajat. Abrahamssonin esittämässä määrittelyssä kohdassa kolme mainitaan myös aikasidonnaisuus. Tyypillisen syklin keston kerrotaan

ketterillä menetelmillä olevan yleensä yhdestä kuuteen viikkoa. Tyypillisessä FNS:n kehitysprojektissa syklin kesto on juuri tämän aikavälin alarajalla: voidaan siis sanoa, että FNS:n mallissa syklit ovat hieman nopeampia kuin keskimääräisessä ketterien menetelmien mallissa. Kohta neljä viittaa kehitysprosessin yksinkertaisuuteen: turhia aktiviteettejä vältetään myös FNS:llä. Fyysisesti samassa tilassa pidettävien palaverien välttäminen kommentointijärjestelmää käyttämällä on tästä keskeisimpiä esimerkkejä. Kohdan viisi kuvaamaa riskienhallintaa toteutetaan FNS:n järjestelmässä käyttöliittymien suunnittelulla etukäteen ja nopealla kommentointi-/testiversioiden julkaisulla: näin mahdolliset muutostarpeet saadaan nopeasti esille.

Luettelon kohdissa kuusi ja seitsemän mainittu inkrementaalisuus toteutuu FNS:n menetelmässä edistyvillä kommentointiversioilla: ensin toteutetaan pelkät käyttöliittymämallit, sitten tekniikka niiden taustalle. Näin toiminnallisuutta kasvatetaan hiljalleen. Ensin keskitytään myös ainoastaan keskeisimpiin perustoimintoihin, minkä jälkeen laajennetaan harvemmin tarvittuihin lisätoimintoihin. Yhteen liitettävyyskin toteutuu FNS:n menetelmässä osana jatkuvaa kehitysversioiden toteuttamista: eri osien yhteenliittämistestausta tapahtuu jatkuvasti, kun kehitysversiota pidetään jatkuvasti toiminnallisena kehitystyön ohessa.

Henkilöorientoituneisuus FNS:n prosessissa näkyy kommentointiryhmän ja sen vastuuhenkilön keskeisessä roolissa: keskeisin tieto eri toimintojen ja tietojen määrittelyistä tulee kommentointiryhmän antamien kommenttien ja esimerkkien kautta, ei erillisten, etukäteen kirjattuina määrittelydokumentteina. Henkilöiden viestintätapaan FNS:n menetelmä ottaa vahvasti kantaa: ensisijainen viestintätapa on aina kommentointijärjestelmä. Tässä voidaan nähdä eroavaisuus muihin ketteriin menetelmiin: Esimerkiksi XP-menetelmässä parhaaksi viestintätavaksi esitettiin asiakkaan kanssa samassa tilassa käyty suullinen keskustelu. FNS:n työntekotapa on yhteisöllinen ja yhteistyökykyinen. Kommentointijärjestelmä pyrkii tarjoamaan hajautetulle kommentointiryhmälle virtuaalisen paikan viestiä yhteisönä yhteisellä keskustelupalstalla. FNS:n oma tiivis työyhteisö huolehtii jatkuvasti yrityksen sisäisen viestinnän yhteisöllisestä toiminnasta.

Eroavaisuuksia FNS:n menetelmän ja yleisesti tunnettujen menetelmien välillä on löydettävissä tarkastelemalla taulukossa 1 (s. 26) esitettyjä tyypillisiä suunnitelmaohjautuvien menetelmien piirteitä. Kehittäjien, käytännössä siis suunnittelijoiden ja ohjelmoijien, keskeisenä ominaisuutena kerrotaan olevan suunnitelmaorientoituneisuuden. Suunnitelmaorientaatio ei ole mahdollista FNS:n menetelmässä, sillä tarkkaa ja yksiselitteistä työsuunnitelmaa ei ole työn alkaessa olemassa: se muodostetaan käyttöliittymämallien kehitystyön aikana.

Asiakkaan ominaisuuksiin listataan suunnitelmaohjautuvissa menetelmissä kyky koota tietoa henkilöiltä, jotka tietävät asioista. FNS edellyttää, että asiakkaan kommentointiryhmään kuuluu kaikkia järjestelmän tulevien käyttäjien edustajia: tiedon tulee tulla kehittäjien käyttöön suoraan työn alla olevasta asiasta parhaiten tietävältä henkilöltä ilman välikäsiä. Määrittelyt ovat suunnitelmaohjautuvissa menetelmissä tiedossa jo hyvissä ajoin ennen työn aloittamista: FNS:n menetelmässä tätä ei edellytetä – riittää, että keskeiset suuntaviivat ovat selvillä. Arkkitehtuurin osalta FNS:n menetelmä pitäytyy yleensä muiden ketterien menetelmien tapaan tämänhetkisessä tilanteessa huomioiden vain lähitulevaisuudessa (muutamien kuukausien sisällä) tulossa olevat muutostarpeet. Suunnitelmaohjautuvat menetelmät pyrkivät varautumaan tulevaisuudessa tarvittaviin muutoksiin paljon perusteellisemmin. FNS:n toteuttajatiimi on pieni, joka mahdollistaa oman ketterän kehitysmenetelmän käytön. FNS:n toteuttaessa järjestelmiä nopealla aikataululla kohtuullisen pieniin erityistarpeisiin, on nopea hyödyn tuottaminen keskeisempää kuin hyvin korkea luotettavuus. Nämä seikat suuntaavat FNS:n menetelmän selkeästi enemmän ketteriin menetelmiin kuin suunnitelmaohjautuviin menetelmiin.

3.3.1 Eroavaisuuksia esiteltyihin ketteriin menetelmiin

Edellä esitetyn vertailun perusteella voidaan sanoa, että keskeiset ketterien menetelmien tunnusmerkit voidaan tunnistaa FNS:n ohjelmistokehitysmenetelmästä. Selkeitä eroavaisuuksia esitettyihin ketteriin menetelmiinkin kuitenkin on löydettävissä:

- FNS painottaa viestinnässä ensisijaisena tapana verkkoviestintää kasvokkain tapahtuvan viestinnän sijasta. Tehdyn kirjallisuustutkimuksen perusteella ketterissä menetelmissä kasvokkain tapahtuva viestintä on ensisijaisen tärkeää.

- Aikataulut ovat vielä nopeampia kuin muissa ketterissä menetelmissä yleensä: tyypillinen kehityssyklin kesto on yksi viikko.

- Kommentointijärjestelmä on vapaata sanallista keskustelua koordinoidumpaa:
  Työhön voi osallistua huomattavasti suurempi joukko asiakkaan edustajia kuin esitellyissä ketterissä menetelmissä. Kommentoijien kaikki kommentit tallentuvat järjestelmään.

- Fyysisesti läsnä olevaa asiakasta ei edellytetä:
  Esimerkiksi XP-menetelmä korostaa samassa tilassa olevan asiakkaan merkitystä: FNS:n menetelmä ei vaadi asiakkaan fyysistä läsnäoloa, sillä seuranta ja ohjeistus voidaan tehdä verkon kautta kommentointijärjestelmää käyttäen.

- Määrittelyt esitetään suusanallisen tai kirjoitetun kuvauksen sijasta käyttöliittymillä: Käyttöliittymämallien käyttö keskeisenä määrittelyjen esitysvälineenä ei esiinny missään esitellyssä ketterässä menetelmässä.

- Hajautetut kehitysympäristöt eivät ole FNS:n menetelmässä ongelmallisia. Ketterien menetelmien haasteeksi esitetty (Turk 2002) hajautetun kehitysympäristön ongelma helpottuu merkittävästi verkossa toimivan kommentointijärjestelmän avulla: asiakkaan kommentit ovat kommentointijärjestelmän kautta käytettävissä missä vain. FNS on toiminut aiemmin useita vuosia hajautetussa ympäristössä, jossa käyttöliittymämallien teko on tapahtunut fyysisesti eri sijainnissa kuin ohjelmamoduulien varsinainen toteutus. Tämä on onnistunut ongelmitta, kun kommentointijärjestelmän tiedot ovat olleet yhtäläisesti kaikkien tekijöiden saatavilla, ja tiedonsiirto muuten on voitu toteuttaa sähköisesti.

3.3.2 Uusi ketterä menetelmä: FNS-menetelmä

Tehtyjen vertailujen perusteella FNS:n ohjelmistokehitysmenetelmän ei voida sanoa suoraan olevan mikään edellä kuvatuista ketteristä menetelmistä. Menetelmä yhdistää ketteriä ja suunnitelmaohjautuvia menetelmiä ja luo kokonaan uuden käytännön kommentointijärjestelmän vuoksi. FNS:n menetelmän sisältämät toimintatavat mahdollistavat joitakin tyypillisesti vain suunnitelmaohjattuja menetelmiä käyttämällä saavutettuja etuja. Tämän vuoksi voidaan määritellä uusi ketterän ohjelmistokehityksen menetelmä: ***FNS-menetelmä***.

FNS-menetelmä voidaan määritellä seuraavien keskeisten piirteiden avulla:

- Käyttöliittymämallit toimivat keskeisimpänä tallennettavien tietojen ja toiminnallisten ominaisuuksien määrittelytapana

- Viestintä asiakkaan ja FNS:n välillä tapahtuu ensisijaisesti (verkossa) järjestelmän demoon liitettyä kommentointijärjestelmää käyttäen

- Asiakkaan palauteajat ovat hyvin nopeita ja tarkkaan määriteltyjä (kommentointiaika 1-7 vrk)

- Ohjelmistokehittäjät tekevät asiakkaille runsaasti ehdotuksia, joiden pohjalta asiakas kertoo mielipiteensä

- Kehitystyö on erittäin modulaarista: kommentoinnissa on vain muutamia järjestelmän osia kerrallaan

- Järjestelmän kehityssyklit ovat nopeita: asiakkaalta ja yritykseltä odotetaan nopeaa toimintaa kommentoinnissa ja uusien mallien sekä valmiiden toimintojen toteutuksessa.

Edellä mainitut piirteet esittävät keskeiset FNS-menetelmän toimintaperiaatteet. Osa niistä poikkeaa tunnetuista ketterien kehitysmenetelmien määritelmistä, mutta minkään ei voida sanoa olevan ristiriidassa esimerkiksi Agile Manifeston (Beck ym. 2001) keskeisten perusajatusten kanssa. Tämän vuoksi FNS-menetelmää voidaan pitää omana, uudenlaisena menetelmänään.

4. Kyselytutkimus

4.1 Kyselytutkimuksen tavoitteet

Työn osana toteutettiin kyselytutkimus, jolle asetettiin kaksi päätavoitetta, joihin sen haluttiin antavan vastaukset. Tavoitteena oli selvittää

- Millaisia aika- ja kustannusvaikutuksia menetelmästä oli asiakkaille?
- Miten tyytyväisiä asiakkaat olivat menetelmään ja sillä aikaan saatuun lopputulokseen?

Lopputuotteen asiakastyytyväisyyden selvittämisellä halutaan määrittää, kuinka hyvin laadullisesti ohjelmistoprojektit ovat onnistuneet. Onnistumista arvioidaan sekä toiminnallisesta että taloudellisesta näkökulmasta. Tyytyväisyyden mittaaminen toteutusmenetelmän osalta kertoo asiakkaiden mielipiteen FNS-menetelmän onnistumisesta. Tutkimuksesta saatujen tulosten perusteella pyritään erottamaan FNS-menetelmän vahvuudet ja kehityskohteet.

4.2 Hypoteesit tuloksista

Kyselystä saataville tuloksille asetettiin ennen kyselyä seuraavat hypoteesit:

1. FNS-menetelmä nopeuttaa ohjelmistokehitystyötä
2. Menetelmä säästää ohjelmistoprojektin kustannuksia
3. Menetelmällä toteutetut ohjelmistotuotteet vastaavat hyvin asiakkaiden tarpeita
4. Kommentointiryhmän jäsenet käyttävät kommentointijärjestelmää mielellään

Hypoteesien asettelu perustuu FNS:n omaan käytännön kokemukseen ja aiempaan, suullisesti saatuun palautteeseen eri projektien kommentointiryhmien jäseniltä.

4.3 Kohderyhmä ja vastaajat

Kyselyn kohderyhmää päätettäessä arvioitiin mahdollisia kohderyhmiä FNS:n toteuttamien projektien valikoimaa tutkimalla. Kohderyhmäksi valittiin FNS:n tähänastisen historian kolmen suurimman ohjelmistokehitysprojektin kommentointiryhmien kaikki jäsenet. Perusteena vain kolmen suurimman projektin valinnassa on tiedon saaminen kohtuullisessa ajassa ja kohtuullisilla kustannuksilla. Kolmea suurinta projektia käyttämällä saadaan tehokkaimmin koottua suurin ja heterogeenisin vastaajajoukko. Näiden kolmen projektien kommentoijien joukossa kysely on kokonaistutkimus kaikista perusjoukon jäsenistä (kaikki kommentoijat kutsutaan), kun taas FNS:n toteuttamien projektien joukkoa ajatellen kyse on otantatutkimuksesta (Heikkilä 2005). Valitut projektit ovat seuraavat:

- Suomen sikaloiden terveysluokitusrekisteri (2003)
- Naseva: nautatilojen terveysluokitusrekisteri (2005-2006)
- Provet YES: Yliopistollisen eläinsairaalan potilastietojärjestelmä (2005-2006)

Valitut projektit edustavat eri aikakausia FNS:n historiassa. Yksi projekteista on hieman vanhempi, kolme vuotta käytössä ollut järjestelmä, jonka käytöstä ja ylläpidosta on ehditty saamaan jo jonkin verran kokemusta. Kaksi muuta järjestelmää ovat uudempia, kuluvana vuonna käyttöönotettuja järjestelmiä. Sekä vanhempien että uudempien ohjelmistoprojektien mukaan ottamisella pyritään saamaan laajempi käsitys eri vaiheissa olevien projektien kannalta.

Kutsu osallistua tutkimukseen toteutettiin lähettämällä kommentointiryhmiin kuuluneille henkilöille sähköpostitse kutsu, jossa kuvataan tutkimus ja pyydetään osallistumaan tutkimukseen. Kutsuviestin sisällön suunnittelussa käytettiin apuna Heikkilän (2005) antamia ohjeita saatekirjeen laatimisesta. Kutsukirjeen malli on esitetty liitteessä 3. Kysely lähetettiin henkilöille riippumatta siitä, oliko henkilö enää nykyisin tekemisissä kehitettyjen järjestelmien kanssa. Kohderyhmään, joille kutsu lähetettiin, kuului 238 henkilöä. 11 vastaanottajan sähköpostiosoite oli vanhentunut, joten kysely saatiin 227 vastaanottajan tietoon. Vastaanottajat olivat kunkin projektin asiakkaan (tilaajan) edustajia sekä järjestelmien muita käyttäjiä, asiakkaan sidosryhmiä. Henkilöt jakautuivat melko tasaisesti naisiin ja miehiin.

Kyselyyn vastasi kaiken kaikkiaan 43 henkilöä. Vastausprosentiksi saadaan näin 43/227 * 100% = 18,95 % ~ 19%. Heikkilän (2005) mukaan saaduista vastauksista kannattaa tehdä kokonaistutkimus, jos vastausjoukko on pieni (alle 100 vastausta), joten tutkimuksen kohteeksi valittiin kaikki vastaajat (perusjoukko).

4.4 Toteutustapa ja rakenne

Kyselytutkimus toteutettiin Internetissä www-sivulla vastattavissa olevana kyselynä. Heikkilän (2005) mukaan Internet-kyselyt ovat nykyään yleisiä, ja niitä voidaan käyttää silloin kun vastaajilla on mahdollisuus käyttää Internetiä – tässä tilanteessa tämä oli selvää, sillä käyttäjät ovat osallistuneet järjestelmien kehitykseen nimenomaan Internetissä. Kyselyn osoite lähetettiin vastaajille sähköpostitse. Kyselyyn annettiin vastausaikaa yksi viikko kutsun lähettämisestä. Kyselyn vastaukset tallennettiin suoraan Php –ohjelmointikielellä toteutetun apuohjelman avulla MySQL-tietokantaan helposti jatkokäsiteltävään muotoon tulosten analysoinnin helpottamiseksi.

Pääosa kyselyn kysymyksistä esitettiin asenneasteikkoina, joihin annetaan vastausvaihtoehdot Likertin asteikolla (Heikkilä 2005) 1-5 olevista valintavaihtoehdoista. Tällä asteikolla arvo 1 tarkoittaa ääripäätä ”Täysin eri mieltä” ja arvo 5 ”Täysin samaa mieltä”. Lisäksi kyselyssä oli muutama vaihtoehdot antava taustatietokysymys sekä kyselyn lopuksi muutamia avoimia kysymyksiä. Osallistuneiden henkilöiden kesken arvottiin pieni palkinto, jonka vuoksi kyselyssä kysytään myös vastaajan yhteystiedot. Vastaajien yhteystiedot tallennettiin erilliseen tietokantatauluun, jota käytettiin ainoastaan arvontaan – yhteystietoja ei yhdistetty kyselyn vastauksiin.

Kyselyn kysymykset jakautuivat neljään pääosaan: taustatietokysymykset, järjestelmän kokonaisuuden arviointi, kehitysmenetelmän arviointi sekä vapaat kehitysajatukset. Ensimmäisessä osiossa kartoitettiin vastaajan perustiedot. Toinen osio mittasi vastaajien mielipiteitä toteutetun järjestelmän kokonaisuudesta. Kolmas osio koski vastaajan mielipiteitä kommentointijärjestelmästä. Neljäs osio oli vapaita, tekstimuotoisia ajatuksia varten.

Mittauksen sisäisen luotettavuuden tarkistamiseksi kyselyssä kysyttiin Heikkilän (2005) antaman ohjeen mukaisesti muutamia samaan asiaa mittaavia kysymyksiä useampaan kertaan. Näiden toistomittauksien avulla laskettiin korrelaatiokertoimia, jotka esitetään tulosten yhteydessä. Edellä mainittujen varsinaisten kysymyksien jälkeen pyydettiin vastaajan yhteystiedot arvontaa varten. Kyselyn yhteydessä annettiin myös mahdollisuus vapaan palautteen jättämiseen koskien koko FNS:n toimintaa. Tätä yleistä, yritystä koskevaa palautetta ei käsitellä tässä diplomityössä.

Kyselyn kysymyslomake on esitetty kokonaisuudessaan liitteessä 4.

4.5 Tulosten arvioinnissa käytetyt menetelmät ja esitystapa

Kyselyn tulosten analysoinnissa on ensisijaisena tunnuslukuna käytetty aritmeettista keskiarvoa. Keskiarvoa on käytetty siksi, että keskiarvolla saadaan tiivistettyä aineiston

tieto tiiviiseen muotoon, vaikka osa informaatiosta samalla häviääkin (Heikkilä 2005). Heikkilän mukaan keskiarvo on vakaa suure erityisesti suurissa havaintomäärissä, kun taas pienissä havaintomäärissä ääriarvojen vaikutus keskiarvoon voi olla huomattava. Keskiarvoa on käytetty tässä tutkimuksessa myös pienien havaintomäärien analysoinnissa, mutta koska näissä tapauksissa havaintoarvot ovat olleet ennalta määrättyjä (pääasiassa lukuarvot väliltä 1-5), eivät ääriarvot pääse vaikuttamaan keskiarvoon yhtä paljon, kuin tilanteessa, jossa vastaukset olisi voinut antaa avoimella asteikolla. Keskiarvon lisäksi asteikolla 1-5 vastatuille tuloksille lasketaan keskihajonnat. Korrelaatiokertoimet lasketaan niille kysymyksille, jotka ovat samaan asiaan vastauksia antavia kontrollikysymyksiä.

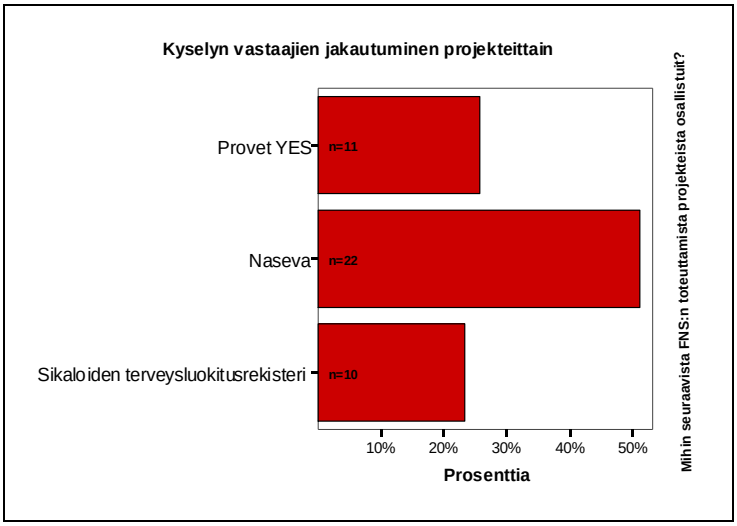
Vastaukset jaotellaan useimpien projektikokonaisuuteen liittyvien kysymysten kohdalla kolmeen osajoukkoon vastaajien projektin mukaan: Sikaloiden terveysluokitusrekisterin, Nasevan ja Provet YES:in vastaajien kesken. Kyselyn kysymyksiä laadittaessa yhtenä mahdollisena jaottelukriteerinä arvioitiin voivan olla myös vastaajien käyttäjäryhmä toteutetuissa järjestelmissä. Saatujen vastausten lukumäärä huomioiden tätä jakoa ei ole kuitenkaan järkevää käyttää tulosten esittämisessä. Kommentointijärjestelmää koskevassa osiossa kysymyksiä on käsitelty yhtenä vastausjoukkona.

Tulokset esitetään joko taulukkona tai pylväsdiagrammeina. Tulokset on laskettu siitä vastaajajoukosta, jotka ovat vastanneet kyseiseen kysymykseen. Kyselyssä oli mahdollista jättää vastaamatta joihinkin kohtiin, eivätkä kaikki kysymykset koskeneet kaikkia vastaajia. Tulosten esityksessä käytetyt keskiarvot ja keskihajonnat on pyöristetty kahden desimaalin tarkkuuteen, mutta prosenttiluvut on pyöristetty kokonaisluvuiksi. Tuloksissa esitetyt laskelmat, taulukot ja kaaviot tehtiin SPSS 14.0 for Windows –ohjelmistolla. Joidenkin taulukoiden ja kaavioiden muotoilussa käytettiin apuna myös Microsoft Excel 2003 –ohjelmaa.

4.6 Kyselyn tulokset

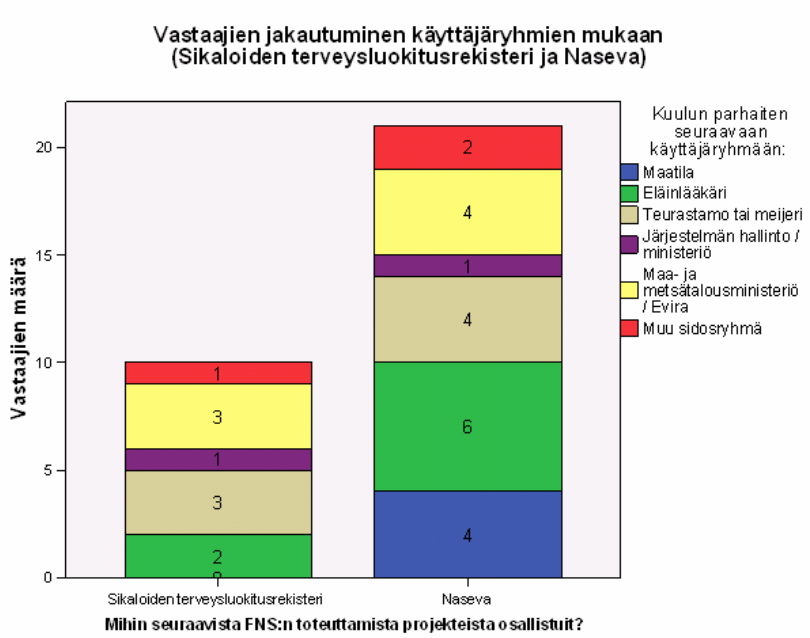
4.6.1 Vastaajajoukon taustatiedot

Kyselyssä kysyttiin ensimmäisenä projektia, johon vastaaja oli osallistunut kommentoijana. Kaikista vastaajista (n=43) suurin osa, yli puolet, oli osallistunut Nasevan kommentointiin. Loppu vastaajajoukko jakautui lähes tasan Sikaloiden terveysluokitusrekisterin ja Provet YES:in välillä. Vastaajajoukon jakauma on esitetty graafisesti kuvassa 22.

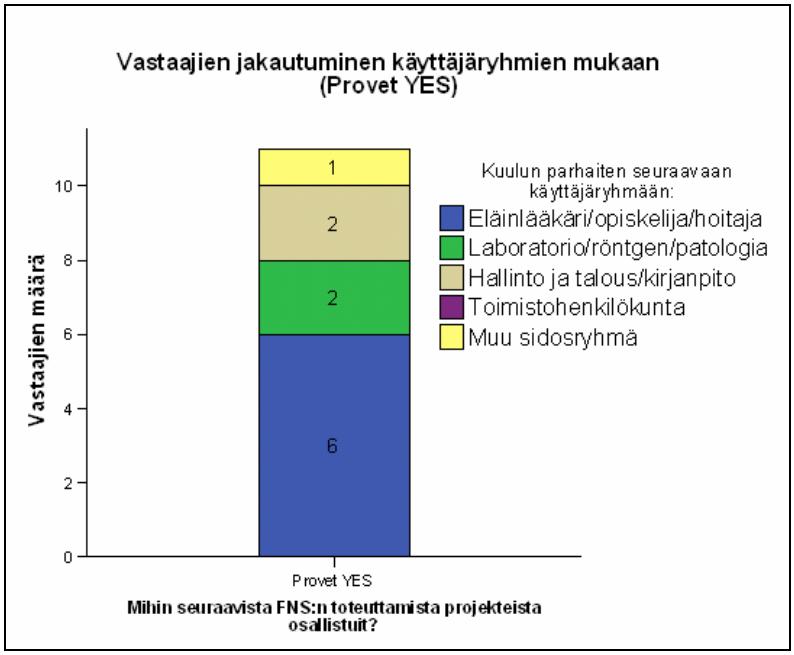


Kuva 22: Kyselyn vastaajien jakautuminen projekteittain

Taustatiedoissa kysyttiin myös vastaajan käyttäjäryhmää. Käyttäjäryhmät jakautuvat hyvin tasaisesti, mutta eläinlääkärit erottuvat kaikissa kolmessa projektissa suurimmaksi käyttäjäryhmäksi. Nasevan ja Sikaloiden terveysluokitusrekisterin käyttäjäryhmien jakautuminen (ryhmät olivat samat molemmissa projekteissa) on esitetty kuvassa 23. Provet YES:in vastaajien käyttäjäryhmien jakauma on esitetty kuvassa 24.



Kuva 23: Sikaloiden terveysluokitusrekisteri ja Naseva -projektien vastaajien käyttäjäryhmät



Kuva 24: Provet YES –projektin vastaajien käyttäjäryhmät

Esikysymyksissä kysyttiin myös, oliko vastaaja osallistunut aiemmin tietojärjestelmien kehityshankkeisiin sekä oliko hän toiminut tämän tietojärjestelmän kehityksessä kustannuksista vastaavassa roolissa. Kustannuksista vastaavassa roolissa vastaajista ilmoitti toimineensa 12% vastaajista (5 vastaajaa), kun taas aiempaa tietojärjestelmien kehityskokemusta oli 40%:lla kaikista vastaajista (17 vastaajaa).

4.6.2 Järjestelmän kokonaisuuden arviointi

Kyselyn ensimmäisessä varsinaisessa kysymysosassa pyydettiin vastaajia arvioimaan järjestelmän kokonaisuutta yhdeksällä eri kysymyksellä, jotka voidaan tulosten arvioinnissa jakaa kolmeen eri kategoriaan: järjestelmän laadun arviointi, ohjelmistoprojektin kustannustehokkuus sekä aikataulun onnistuminen. Seuraavassa esitetään järjestelmän kokonaisuutta koskevien kysymysten tulokset näiden kolmen ryhmän mukaan.

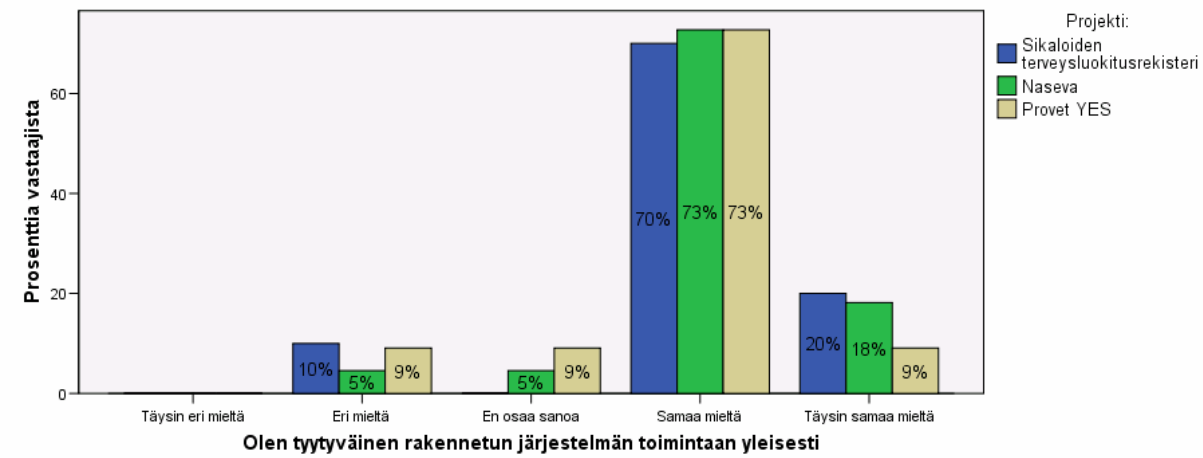
<u>Laadun arviointi</u>

Laatua arvioitiin neljällä kaikille vastaajille yhteisellä kysymyksellä, johon vastattiin asteikolla 1= Täysin eri mieltä … 5=Täysin samaa mieltä. Näihin kysymyksiin saatujen vastausten keskiarvot ja keskihajonnat on esitetty taulukossa 2.

Taulukko 2: Laatua mittaavien kaikille yhteisten kysymysten tuloksien tunnusluvut

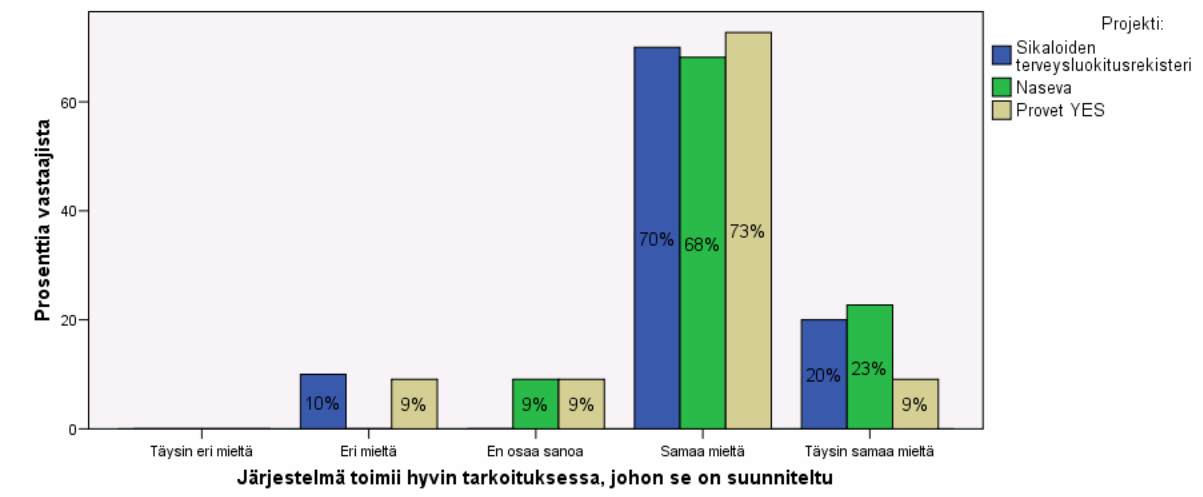
Kysymys	n	Keskiarvo	Keskihajonta
Olen tyytyväinen rakennetun järjestelmän toimintaan yleisesti	43	3,98	0,70
Järjestelmä toimii hyvin tarkoituksessa, johon se on suunniteltu	43	4,02	0,67
Järjestelmä toteuttaa minulle tärkeät toiminnallisuudet	43	3,65	0,79
Järjestelmä ei valmistuessaan täyttänyt sille asetettuja toimintatavoitteita	43	2,49	1,10

Rakennetun järjestelmän laatuun oltiin yleisesti tyytyväisiä. Kysymyksen ”Olen tyytyväinen rakennetun järjestelmän toimintaan yleisesti” vastausten keskiarvoksi saatiin 3,98, joka tarkoittaa vastaajien olevan samaa mieltä väittämän kanssa. Kuvassa 25 on esitetty vastauksien järjestelmän jakautuminen ohjelmistoprojekteittain. Kuvasta nähdään, että projektien kesken ei ole merkittävää eroa. Pääosa vastaajista on samaa mieltä tai täysin samaa mieltä väittämän kanssa.



Kuva 25: Tyytyväisyys järjestelmän toimintaan yleisesti projekteittain

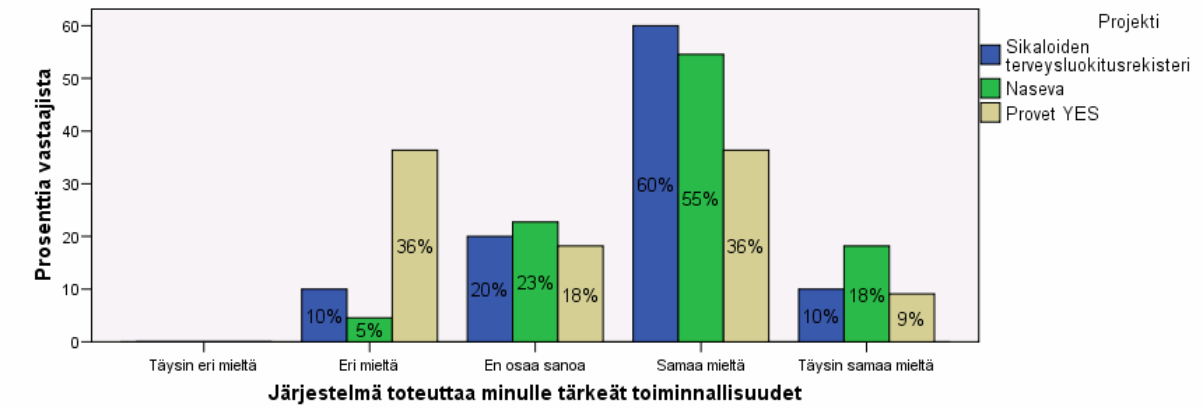
Kysymykseen ” Järjestelmä toimii hyvin tarkoituksessa, johon se on suunniteltu” saatiin vastausten keskiarvoksi 4,02. Tämä tarkoittaa, että vastaajat ovat keskimäärin samaa mieltä väittämän kanssa, joten järjestelmä toimii hyvin tarkoituksessa, johon se on suunniteltu. Kuvassa 26 esitettyjen kaavioiden perusteella eri projektien välillä ei ole merkittäviä eroja; ainoastaan Provet YES –projektin vastaajat ovat hieman enemmän samaa mieltä kuin täysin samaa mieltä väittämän kanssa.



Kuva 26: Järjestelmän toimivuus tarkoituksensa mukaisessa tehtävässä projekteittain

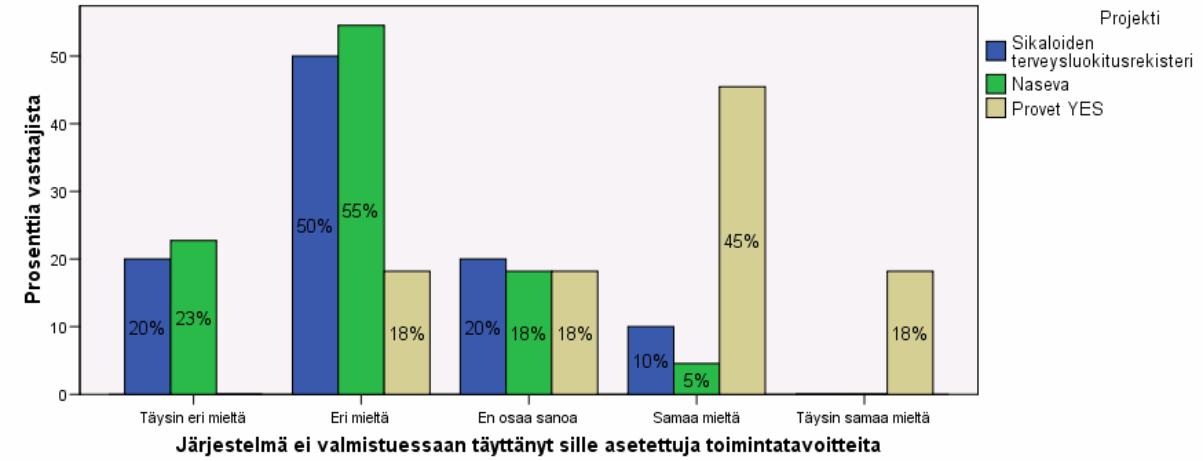
Edellä käsitellyt kaksi kysymystä käsittelevät samaa asiaa, ja ne ajateltiin kyselyn suunnittelussa kontrollikysymyksiksi toisilleen. Heikkilä (2005) esittää korrelaatiokertoimien laskemisen tavaksi arvioida kyselyn sisäistä luotettavuutta. Tämän vuoksi laskettiin kysymysten välinen Pearsonin tulomomenttikerroin r, jonka arvoksi saatiin SPSS-ohjelman toiminnoilla r=0,803 merkitsevyystasolla p=0,000. Näin voidaan sanoa, että korrelaatio vastausten välillä on merkittävää, ja kysymysten välillä on lineaarista riippuvuutta.

Projektin tuotteena olleen järjestelmän sopivuutta vastaajan omiin tarpeisiin kartoitettiin kysymyksellä ” Järjestelmä toteuttaa minulle tärkeät toiminnallisuudet”. Kaikkien tuloksien keskiarvoksi saatiin 3,65, joka on lähimpänä vastausvaihtoehtoa ”Samaa mieltä”. Keskihajonta on kuitenkin 0,90, joka on jonkin verran suurempi kuin edellä käsitellyissä kysymyksissä. Kuvassa 26 esitetään vastaukset kysymykseen projekteittain. Tässä kysymyksessä voidaan havaita merkittävämpiä eroja eri projektien välillä; Provet YES:n kohdalla yli 20 prosenttiyksikköä vastaajista on ollut eri mieltä väittämän kanssa kuin Sikaloiden terveysluokitusrekisterin ja Nasevan tapauksessa.



Kuva 27: Vastaajan kannalta tärkeitten toiminnallisuuksien toteutuneisuus projekteittain

Kysymyksen ”Järjestelmä ei valmistuessaan täyttänyt sille asetettuja toimintatavoitteita” asettelu oli edellisiin kysymyksiin verrattuna vastakkainen. Positiivisin vastaus saavutettiin ”Täysin eri mieltä” –mielipiteellä. Keskiarvoksi tämän kysymyksen vastauksille saatiin 2,49, joka on lähinnä vaihtoehtoa ”Eri mieltä”. Käänteisen kysymysasettelun vuoksi voidaan saadun keskiarvon perusteella sanoa vastaajien olevan samaa mieltä siitä, että järjestelmä täyttää sille asetetut toimintatavoitteet. Kuvasta 28 nähdään vastausten jakauma projekteittain. Projektien väliset erot ovat merkittävät: Provet YES:in kohdalla käyttäjät ovat olleet muita projekteja merkittävästi enemmän sitä mieltä, ettei järjestelmä valmistuessaan täyttänyt sille asetettuja toimintatavoitteita.



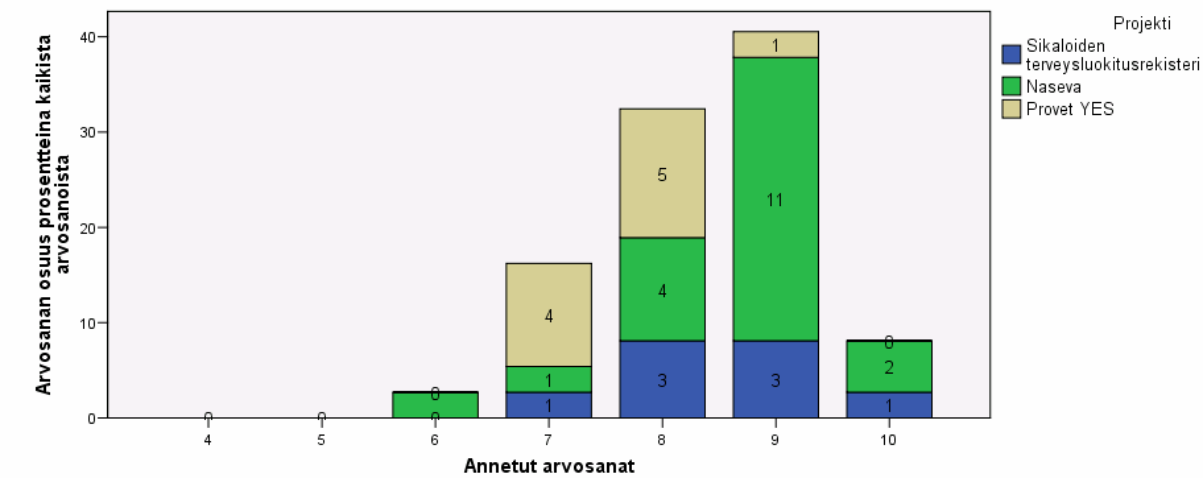
Kuva 28: Asetettujen toimintatavoitteiden täyttyminen projekteittain

Laatuun liittyvissä kysymyksissä oli myös yksi kysymys henkilöille, joilla oli aiempaa kokemusta ohjelmistokehitysprojekteihin osallistumisesta. Tällaisia vastaajia oli esitietokysymysten perusteella 17 henkilöä, mutta kysymykseen ” Järjestelmän toteutus onnistui huonommin kuin ne ohjelmistoprojektit, joihin olen aiemmin osallistunut” saatiin vain 15 vastausta. Pienen vastaajajoukon takia vastauksia ei tarkasteltu projektikohtaisesti edellisten kysymysten tavoin, vaan taulukossa 3 esitetään vastausten frekvenssit ja prosentuaaliset osuudet. Vastausten keskiarvoksi saatiin 2,40 (= Eri mieltä). Kysymyksen käänteisen asettelun vuoksi tämä tarkoittaa vastaajien olleen keskimäärin sitä mieltä, että järjestelmän toteutus onnistui paremmin kuin ne ohjelmistoprojektit, johon vastaaja on aiemmin osallistunut.

Taulukko 3: Vastaukset ja tunnusluvut järjestelmän toteutuksen vertailusta muihin ohjelmistoprojekteihin

Vastaus	Frekvenssi	Prosenttia vastauksista	Tunnusluvut	
Täysin eri mieltä	2	13,3	*Vastausten määrä*	15
Eri mieltä	8	53,3	*Keskiarvo*	2,40
En osaa sanoa	2	13,3	*Keskihajonta*	0,986
Samaa mieltä	3	20		
Täysin samaa mieltä	0	0		
Yhteensä	*15*	*100*		

Viimeisessä järjestelmien laatua arvioivista kysymyksistä pyydettiin antamaan kouluarvosana koko järjestelmän kehitysprojektista esittämällä väite ”Anna kouluarvosana asteikolla 4-10 koko projektille”. Vastauksia tähän väittämään saatiin 37, joten kuusi vastaajaa ei vastannut kysymykseen. Saatujen vastausten keskiarvoksi saatiin 8,35 ja keskihajonnaksi 0,95. Kuvasta 29 voidaan nähdä, että Naseva ja Sikaloiden terveysluokitusrekisteri ovat saaneet hieman parempia arvosanoja kuin Provet YES. Pylväisiin merkityt numerot ovat vastausmääriä.



Kuva 29: Projektikokonaisuudelle annettujen arvosanojen jakaantuminen.

Ohjelmistoprojektin kustannustehokkuus

Ohjelmistoprojektin kustannustehokkuuden arviointia varten kyselyssä esitettiin kaksi kysymystä koko projektia koskien. Kysymykset esitettiin ainoastaan henkilöille, jotka olivat toimineet projekteissa kustannuksista vastaavassa roolissa (n=5). Väittämän ”Tarjouksessa ja sopimuksessa sovitut kustannukset eivät pitäneet paikkansa” vastausten keskiarvoksi saatiin 1,40 (= Täysin eri mieltä), joka tarkoittaa, että

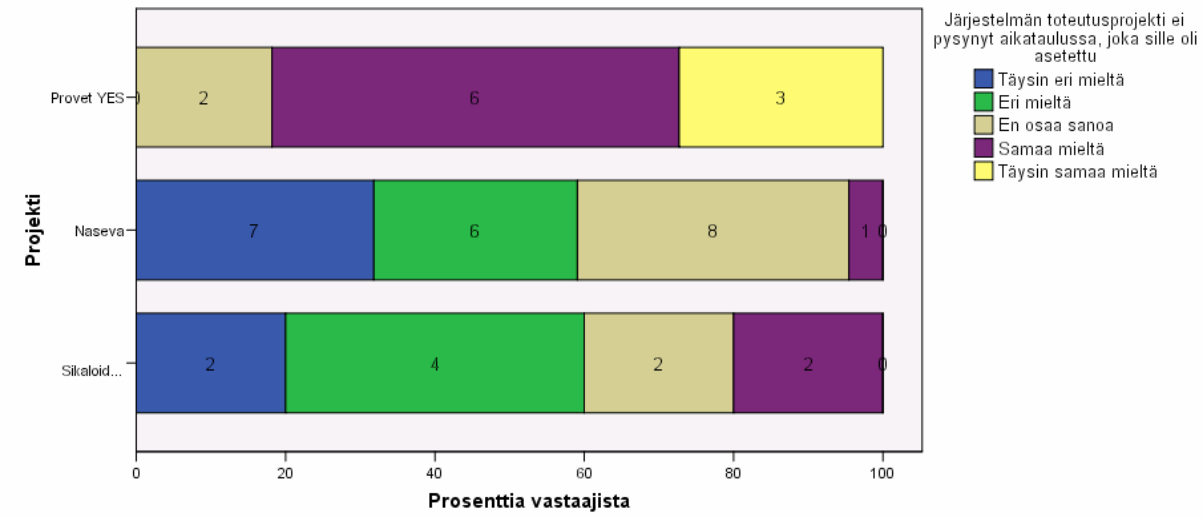
projektien kustannukset ovat pitäneet paikkansa hyvin. Kysymyksessä ”Järjestelmän laatu/hinta-suhde on hyvä” vastaajat olivat yhtä lukuun ottamatta olleet täysin samaa mieltä, jolloin keskiarvoksi saadaan 4,80. Kuvaajaa ei ole piirretty, koska vastaajia oli vain viisi.

Taulukko 4: Projektin kustannustehokkuutta mittaavien kysymysten vastausten jakaumat ja keskiarvot

	Täysin eri mieltä	Eri mieltä	En osaa sanoa	Samaa mieltä	Täysin samaa mieltä	Keskiarvo
Tarjouksessa ja sopimuksessa sovitut kustannukset eivät pitäneet paikkansa	3	2	0	0	0	**1,4**
Järjestelmän laatu/hinta –suhde on hyvä	0	0	0	1	4	**4,8**

Aikataulun onnistuminen

Aikataulun onnistumisen mittaamista varten esitettiin kaikille vastaajille väittämä ”Järjestelmän toteutusprojekti ei pysynyt aikataulussa, joka sille oli asetettu”. Kysymykseen vastasivat kaikki vastaajat (n= 43). Tulokseksi saatiin eri projekteissa hyvin erilaisia tuloksia. Provet YES:n kohdalla aikataulussa pysyminen oli heikompaa kuin Nasevan ja Sikaloiden terveysluokitusrekisterin kohdalla. Eri vastausten jakaumat projekteittain on esitetty kuvassa 30. Pylväisiin merkityt numerot ovat vastausmääriä. Vastausten keskiarvoksi saatiin 2,70, joka tarkoittaa vaihtoehtoa ”En osaa sanoa”. Keskihajonnan arvoksi saatu 1,23 kertoo myös merkittävästä vaihtelusta vastaajien mielipiteiden välillä.



Kuva 30: Aikataulun onnistumista kuvaavan kysymyksen vastausjakaumat projekteittain

4.6.3 Kehitysmenetelmän arviointi

Kehitysmenetelmää ja erityisesti kommentointijärjestelmää käsittelevässä kyselyn osiossa pyydettiin vastaajia arvioimaan kommentointijärjestelmän ominaisuuksia 16 eri kysymyksellä. Näistä viiteen kysymykseen pyydettiin vastaamaan vain osaa vastaajista. Tulosten käsittelyssä kysymykset jaetaan seuraaviin ryhmiin: kommentointijärjestelmän sopivuus vastaajan toimintatapoihin, käyttömukavuus, aikatauluvaikutukset sekä kustannusvaikutukset. Seuraavassa esitellään tulokset näiden neljän ryhmän mukaan.

Sopivuus vastaajan toimintatapoihin

Kommentointijärjestelmän sopivuutta vastaajan toimintatapoihin arvioitiin esittämällä kahdeksan väitemuodossa ollutta kysymystä. Kaikki vastaajat vastasivat kysymyksiin (n=43). Vastausten keskiarvot ja keskihajonnat esitetään taulukossa 5. Vastausten jakauma eri vastausvaihtoehtoihin esitetään sekä vastausmäärinä että prosentteina taulukossa 6.

Taulukko 5: Viestintämenetelmän sopivuutta vastaajalle arvioineiden kysymysten keskiarvot ja keskihajonnat

Kysymys	**n**	**Keskiarvo**	**Keskihajonta**
Sain mielipiteeni hyvin esille kommentointijärjestelmän kautta	43	4,00	0,90
Kommentointijärjestelmä sopii tapaani viestiä	43	3,60	1,14
Kommentointijärjestelmä sopii korvaamaan palavereita	43	3,51	1,14
Sovittu kommentointiaika (7vrk) riitti minulle hyvin	43	3,37	1,11
Kuvailisin tarvitsemani toiminnot mielellään paperilla tekstinä ja piirroksina	43	2,40	0,98
Esittäisin omat ehdotukseni mieluummin suullisesti kuin kirjallisesti	43	2,65	0,97
Toisten kirjoittamien kommenttien näkeminen helpotti omaa kommentointiani	43	3,98	0,80
Pystyn kuvaamaan kirjallisesti mielipiteeni ja ehdotukseni kommentointijärjestelmän kautta	43	3,84	0,72

Taulukko 6: Viestintämenetelmän sopivuutta vastaajalle arvioineiden kysymysten jakaumat vaihtoehtojen kesken vastausmäärinä ja prosentteina.

Kysymys	Täysin eri mieltä	Eri mieltä	En osaa sanoa	Samaa mieltä	Täysin samaa mieltä
Sain mielipiteeni hyvin esille kommentointijärjestelmän kautta	1	2	5	23	12
Osuus prosentteina %	2	5	12	53	28
Kommentointijärjestelmä sopii tapaani viestiä	1	10	3	20	9
Osuus prosentteina %	2	23	7	47	21
Kommentointijärjestelmä sopii korvaamaan palavereita	1	11	4	19	8
Osuus prosentteina %	2	26	9	44	19
Sovittu kommentointiaika (7vrk) riitti minulle hyvin	0	15	3	19	6
Osuus prosentteina %	0	35	7	44	14
Kuvailisin tarvitsemani toiminnot mielellään paperilla tekstinä ja piirroksina	7	20	8	8	0
Osuus prosentteina %	16	47	19	19	0
Esittäisin omat ehdotukseni mieluummin suullisesti kuin kirjallisesti	3	21	7	12	0
Osuus prosentteina %	7	49	16	28	0
Toisten kirjoittamien kommenttien näkeminen helpotti omaa kommentointiani	0	3	5	25	10
Osuus prosentteina %	0	7	12	58	23
Pystyn kuvaamaan kirjallisesti mielipiteeni ja ehdotukseni kommentointijärjestelmän kautta	0	3	6	29	5
Osuus prosentteina %	0	7	14	67	12

Vastaajat ovat saaneet mielipiteensä hyvin esille kommentointijärjestelmän kautta. Väitteen ”Sain mielipiteeni hyvin esille kommentointijärjestelmän kautta” vastauksien keskiarvoksi saatiin tasan 4,00, joka vastaa vaihtoehtoa ”Samaa mieltä”. Vastaajista 81% on joko täysin samaa mieltä tai samaa mieltä esitetyn väitteen kanssa.

Kommentointijärjestelmä sopii vastaajien viestintätapoihin. Vastauksien keskiarvo 3,60 vastaa parhaiten vaihtoehtoa ”Samaa mieltä”. Keskihajonta tässä vastausvaihtoehdossa on kuitenkin melko suuri (1,14). Myös vaihtoehto ”Eri mieltä” on saanut 23% vastauksista. Samaa mieltä tai täysin samaa mieltä vastaajista oli kuitenkin selkeä enemmistö, 68% vastaajista.

Väitteeseen mahdollisuudesta korvata palavereita kommentointijärjestelmän avulla vastatattiin siten, että vastausten keskiarvoksi saatiin arvo 3,51. Tämän perusteella keskimääräiseksi tulokseksi saadaan vaihtoehto ”samaa mieltä”. Keskihajonta on myös tässä kysymyksessä melko suuri, 1,14. Vastaajissa on myös palavereiden kannattajia, sillä täysin eri mieltä tai eri mieltä oli väittämästä yli neljännes vastaajista (28%). Näissä kehitysprojekteissa käytettyä viikon kommentointiaikaa pidettiin juuri ja juuri riittävänä. Vastauksina väittämään ” Sovittu kommentointiaika (7vrk) riitti minulle hyvin” saatiin 3,37. Tämä vastaa keskimmäistä vaihtoehtoa ”En osaa sanoa”. Samaa mieltä tai täysin

samaa mieltä vastaajista oli kuitenkin vähän yli puolet (58%), mutta toisaalta ”Eri mieltä” –vaihtoehdon valitsi 35% vastaajista.

Paperin ja tekstin käyttäminen toimintojen kuvailuun ei vastaajien mielestä ole hyvä toimintatapa. Väitteeseen ” Kuvailisin tarvitsemani toiminnot mielellään paperilla tekstinä ja piirroksina” vastattiin keskiarvon 2,40 perusteella keskimääräisesti ”Eri mieltä”. Täysin samaa mieltä väitteen kanssa ei ollut kukaan vastaajista. Suullisen viestinnän paremmuuttakaan ei vastaajajoukossa tunnustettu todeksi. Keskiarvo 2,65 tarkoittaa ”En osaa sanoa” –vaihtoehtoa väitteeseen ” Esittäisin omat ehdotukseni mieluummin suullisesti kuin kirjallisesti”. Vastaukset painottuvat paremminkin kirjallisen viestinnän puolelle, sillä yli puolet vastaajista (56%) vastasi tähän kysymykseen ”Eri mieltä” tai ”Täysin eri mieltä”.

Toisten kommentoijien kirjoittamien kommenttien näkyminen kommentointipalstalla nähtiin selkeästi hyvänä asiana. Keskiarvolla 3,98 päädyttiin vaihtoehtoon ”Samaa mieltä”, mutta kolme vastaajaa (7%) oli myös eri mieltä väittämästä. Myös kirjallinen mielipiteiden ja ehdotusten esittäminen onnistuu vastaajilta, sillä 67% vastaajista on samaa mieltä ja 12% täysin samaa mieltä siitä, että pystyy kuvaamaan mielipiteensä kirjallisesti. ”Samaa mieltä” –vaihtoehto valitaan myös keskiarvon 3,98 perusteella. Keskihajonta on näiden kysymysten pienin, 0,72.

Kysymyksiä aseteltaessa arvioitiin, että muutamien kysymysten välillä pitäisi olla havaittavissa selkeää korrelaatiota. Väitteiden ”Sain mielipiteeni hyvin esille kommentointijärjestelmän kautta” ja ” Kommentointijärjestelmä sopii tapaani viestiä” välinen korrelaatio selvitettiin laskemalla Pearsonin tulomomenttikerroin SPSS-ohjelmalla. Arvoksi saatiin r=0,582 merkitsevyysarvolla p=0,000. Voidaan todeta, että korrelaatio on merkittävää.

Käyttömukavuus

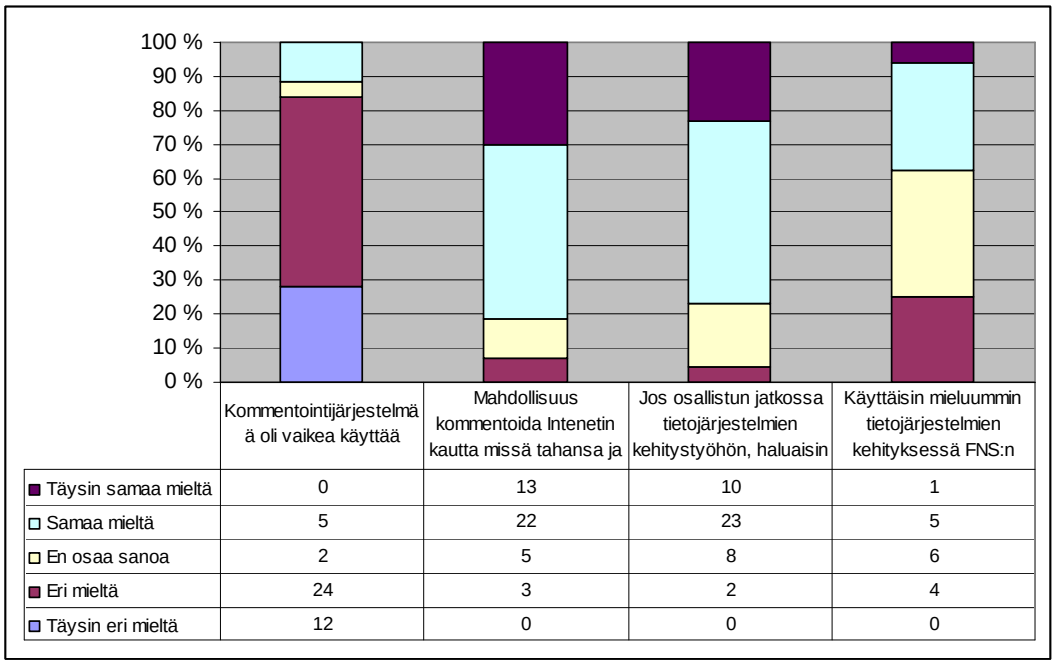
Kommentointijärjestelmän käyttömukavuuden tutkimiseksi kyselyssä kysyttiin kolme kaikille vastaajille yhteistä kysymystä (n = 43) sekä neljäs, aikaisemminkin ohjelmistokehitykseen osallistuneille vastaajille tarkoitettu kysymys (n=16). Näiden kysymysten vastausten tunnusluvut, keskiarvo ja keskihajonta, esitetään taulukossa 7. Vastausten jakaumat eri vastausvaihtoehtojen välillä esitetään kuvassa 31 lukuina ja graafisesti prosentteina.

Taulukko 7: Käyttömukavuuteen liittyvien vastausten tunnusluvut ja eri vastausvaihtoehtojen prosentuaaliset osuudet

Kysymys	n	Keskiarvo	Keskihajonta
Kommentointijärjestelmää oli vaikea käyttää	43	2,00	0,90
Mahdollisuus kommentoida Intenetin kautta missä tahansa ja milloin tahansa helpotti järjestelmän kehitystyötä	43	4,05	0,84
Jos osallistun jatkossa tietojärjestelmien kehitystyöhön, haluaisin käyttää vastaavaa kommentointijärjestelmää	43	3,95	0,79
Käyttäisin mieluummin tietojärjestelmien kehityksessä FNS:n kommentointimenetelmää kuin muita menetelmiä, joita on käytetty niissä projekteissa, joihin olen aiemmin osallistunut	16	3,19	0,91

Vastausten prosentuaaliset osuudet

Kysymys	Täysin eri mieltä	Eri mieltä	En osaa sanoa	Samaa mieltä	Täysin samaa mieltä
Kommentointijärjestelmää oli vaikea…	28 %	56 %	5 %	12 %	0 %
Mahdollisuus kommentoida Intenetin…	0 %	7 %	12 %	51 %	30 %
Jos osallistun jatkossa tietojärjestelmien kehitystyöhön…	0 %	5 %	19 %	53 %	23 %
Käyttäisin mieluummin tietojärjestelmien kehityksessä…	0 %	25 %	38 %	31 %	6 %



	Kommentointijärjestelmä ä oli vaikea käyttää	Mahdollisuus kommentoida Intenetin kautta missä tahansa ja	Jos osallistun jatkossa tietojärjestelmien kehitystyöhön, haluaisin	Käyttäisin mieluummin tietojärjestelmien kehityksessä FNS:n
Täysin samaa mieltä	0	13	10	1
Samaa mieltä	5	22	23	5
En osaa sanoa	2	5	8	6
Eri mieltä	24	3	2	4
Täysin eri mieltä	12	0	0	0

Kuva 31: Kommentointijärjestelmän käyttömukavuuteen liittyvien vastausten jakaumat eri vastausvaihtoehtojen välillä graafisesti ja kunkin vaihtoehdon vastausten määrinä

Väite ” Kommentointijärjestelmää oli vaikea käyttää” saa vastausten keskiarvoksi arvon 2,00, joka vastaa vaihtoehtoa ”Eri mieltä”. Vastausten keskihajonta on melko suuri, 0,9. Vastausten kappalemääristä nähdään, että seitsemää vastaajaa lukuun ottamatta kaikki muut (84%) ovat olleet eri mieltä tai täysin eri mieltä väitteen kanssa. Kommentointijärjestelmä on siis melko helppokäyttöinen. 81% vastaajista kertoo olevansa samaa mieltä tai täysin samaa mieltä Internetin kautta tapahtuvan

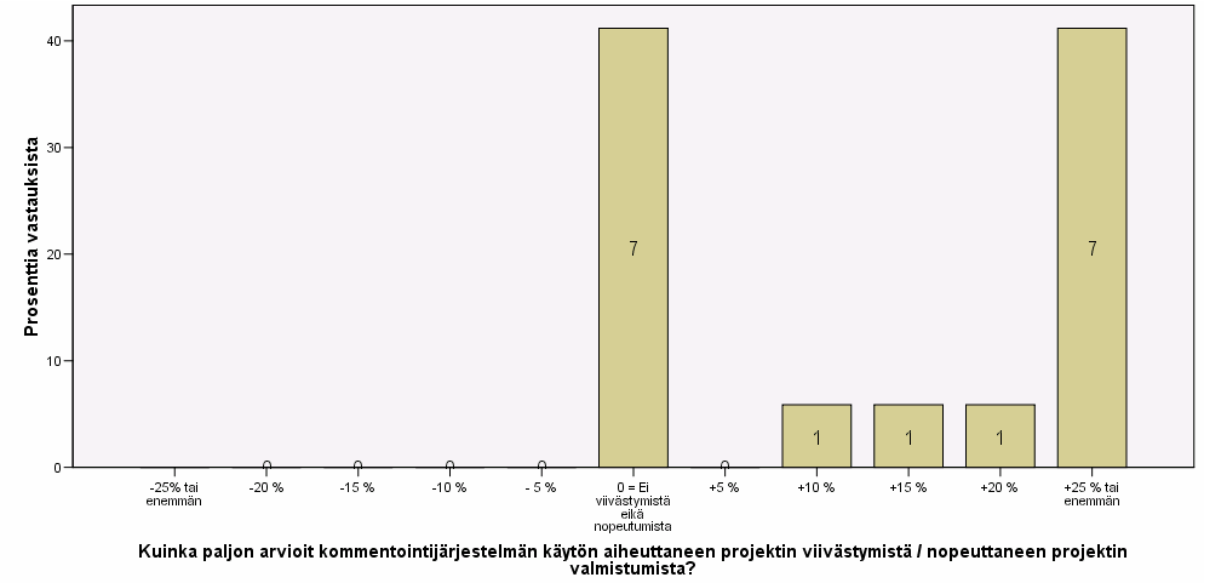
kommentoinnin kätevyydestä. Riippumattomuus kellonajasta ja paikasta on siis selkeästi hyvä ominaisuus.

Väitteeseen ”Jos osallistun jatkossa tietojärjestelmien kehitystyöhön, haluaisin käyttää vastaavaa kommentointijärjestelmää” antaa 76% vastaajista positiivisen vastauksen. Keskiarvoksi saatu 3,95 tarkoittaa vaihtoehtoa ”Samaa mieltä”. Ainoastaan kaksi vastaajaa vastasi ”Eri mieltä”. Mielipidettään ei osannut määrittää 19% vastaajista.

Vastaajilta, joilla oli aiempaa kokemusta ohjelmistokehityksestä, kysyttiin kiinnostusta FNS-menetelmän käyttöön muiden aiemmin käytettyjen menetelmien sijasta. 16 vastaajasta suurin osa, 38%, ei osannut sanoa kantaansa. Seuraavaksi suurin ryhmä (31%) oli samaa mieltä väitteen kanssa. Keskiarvoksi saatiin 3,19, joka vastaa parhaiten vaihtoehtoa ”En osaa sanoa”, joskin on hieman enemmän positiivisten vaihtoehtojen puolella.

Aikatauluvaikutukset

Kommentointijärjestelmän käytön vaikutusta projektin valmistumiseen kysyttiin vastaajilta väittämän ”Kuinka paljon arvioit kommentointijärjestelmän käytön aiheuttaneen projektin viivästymistä / nopeuttaneen projektin valmistumista?” avulla niiltä vastaajilta, jotka olivat osallistunut aiemmin ohjelmistokehitystyöhön (n = 17). Saadut tulokset esitetään kuvassa 32. Kysymykseen vastattiin valitsemalla positiivinen tai negatiivinen prosenttilukuarvo, joka vastaajan mielestä oli lähinnä vaikutusta nopeuteen. Vastaajia pyydettiin valitsemaan sopivin prosenttiarvioista, jotka olivat porrastettu välille [-25% .. +25%] viiden prosenttiyksikön välein. Negatiivinen arvo tarkoitti viivästymistä, positiivinen nopeutumista Myös vaihtoehto ”0 = ei viivästystä eikä nopeutumista” oli mahdollinen. Tuloksista nähdään, että vaihtoehto ”ei viivästymistä eikä nopeuttamista” sekä ”+25% tai enemmän ovat saaneet yhtä monta vastausta (7). Yhtään viivästystä tarkoittavaa vastausta ei ole annettu. Tuloksista lasketuksi keskimääräiseksi aikataulusäästöksi saadaan 13% verrattuna perinteisiin menetelmiin. Keskihajonnaksi saatiin 12%. Keskihajonta on suuri, mutta joka tapauksessa kommentointijärjestelmä nähdään vaikuttavan aikatauluihin nopeuttavasti. Kuvassa 32 numeroarvot pylväissä kertovat vastausten lukumäärän.



Kuva 32: Kommentointijärjestelmän aikatauluvaikutus -kysymyksen vastausten jakauma.

Kommentointijärjestelmän kustannusvaikutukset

Kommentointijärjestelmän vaikutuksia projektin kokonaiskustannuksia arvioitiin kolmella kysymyksellä, jotka kohdistettiin niille vastaajille, jotka ilmoittivat toimineensa projektissa kustannuksista vastaavassa roolissa (n = 6). Näistä henkilöistä viisi vastasi kysymyksiin. Kaksi kysymyksistä oli mielipideasteikolla arvioituja väittämiä, kolmas kustannussäästön/lisäkustannusten prosentuaalisen määrän arviointikysymys. Taulukossa 8 esitetään mielipideväittämien vastausten jakautumat sekä keskiarvot ja keskihajonnat.

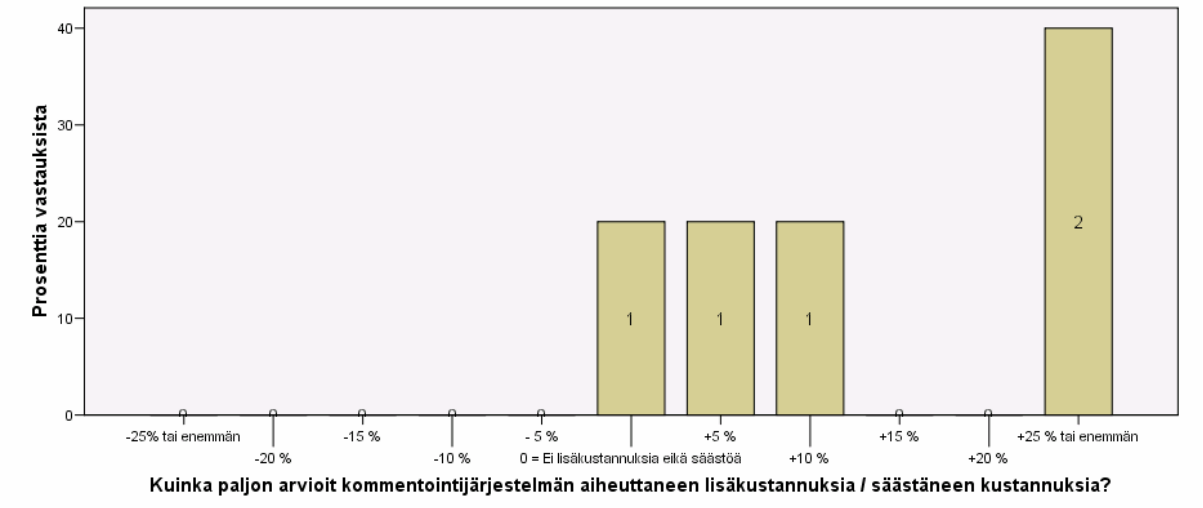
Taulukko 8: Jakaumat, keskiarvot ja keskihajonnat kustannusvaikutuksiin liittyvistä mielipideasteikkokysymyksistä

Kysymys	n	Täysin eri mieltä	Eri mieltä	En osaa sanoa	Samaa mieltä	Täysin samaa mieltä	*Keski-arvo*	*Keski-hajonta*
Kommentointijärjestelmä säästi projektin kokonaiskustannuksia	5	0	0	0	1	4	*4,80*	*0,45*
Kommentointijärjestelmän käyttö ei säästänyt matkakustannuksia eikä aikaa	5	3	1	0	1	0	*1,80*	*1,30*

Tuloksista nähdään, että vastaajat ovat selkeästi olleet sitä mieltä, että kommentointijärjestelmä säästi projektin kokonaiskustannuksia. Kaikki vastaajat ovat olleet samaa tai täysin samaa mieltä väitteen kanssa. Kommentointijärjestelmän vaikutuksesta ajan ja matkakustannusten säästöön mielipide on lähes yhtä selkeä: vastaajat ovat eri mieltä kuin väittämä, joka tarkoittaa, että vastaajien mielestä kommentointijärjestelmä voi säästää matka- ja aikakustannuksia. Yksi vastaajista tosin

oli myös samaa mieltä väittämän kanssa, joka aiheuttaa näin pienessä otosjoukossa keskihajonnan kasvamisen arvoon 1,30.

Mielipidettä kommentointijärjestelmän kustannusvaikutuksista kysyttiin esittämällä kysymys ”Kuinka paljon arvioit kommentointijärjestelmän aiheuttaneen lisäkustannuksia / säästäneen kustannuksia?”. Samoin kuin aikatauluvaikutukset, arvioitiin tämäkin kohta valitsemalla sopivin vaihtoehto välillä [-25% .. +25%] olevien viiden prosenttiyksikön välillä porrastetuista vaihtoehdoista. Positiiviset lukuarvot tarkoittavat kustannussäästöä, negatiiviset lisäkustannuksia. Saatujen vastausten jakauma esitetään kuvassa 33. Tuloksista nähdään, että suurin osa vastaajista (kaksi vastaajaa) on ilmoittanut kustannussäästöksi 25% tai enemmän. Kaksi muuta vastaajaa ovat ilmoittaneet hieman pienemmän kustannussäästöprosentin, ja yksi vastaaja arvioi, ettei menetelmä vaikuta kustannuksiin. Keskimääräiseksi kustannussäästöksi laskettiin 13%, keskihajonnaksi 12%. Menetelmän voidaan siis todeta tuottavan kustannussäästöjä.



Kuva 33: Kommentointijärjestelmän kustannusvaikutuksien arviointikysymyksen vastausjakauma

4.6.4 Vapaat kehitysajatukset

Vapaat kehitysajatukset jaettiin kahteen osaan: koko projektia koskeviin kehitysajatuksiin ja kommentointijärjestelmää koskeviin kehitysajatuksiin. Suurin osa vastaajista ei kirjoittanut vapaita kehitysajatuksia, mutta koko järjestelmän kehitystä koskevia kommentteja annettiin kuitenkin yhdeksän kappaletta ja kommentointijärjestelmään liittyviä kommentteja saatiin 12.

Useimmat vastaajien kirjoittamat koko projektia koskevat kehitysajatukset liittyivät johonkin järjestelmän tekniseen yksityiskohtaan, eivätkä varsinaisesti koskeneet toimintatapoja, joihin liittyviä kommentteja erityisesti haluttiin saada. Muutamia esimerkkejä ajatuksista voidaan kuitenkin esittää:

" Olisin halunnut tietoa myös projektin loppumisesta, nyt sitä ei tullut lainkaan. "

"Kyselyjä liian pienistä asiakokonaisuuksista. Ei ollut tarkkaa tietoa miten projekti etenee, tuli kommentoitua vääristä asioista väärään aikaan."

"Kommentointiryhmälle olisi pitänyt saada varata työaikaa protoon tutustumiseen ja kommentointiin. Olisin mielelläni varannut toimittajalle useamman käynnin sekä henkilöstölle aikaa perustarvekartoitukseen alkuvaiheessa."

Kommentointijärjestelmän kehitykseen liittyvät kehitysajatukset olivat muutamaa poikkeusta lukuun ottamatta selkeitä lisätoiveita kommentointijärjestelmän ominaisuuksiin. Monessa kommentissa toivottiin äänestysmahdollisuuden lisäämistä kommentointijärjestelmään. Muutamat toivoivat myös kuvien ja muiden visuaalisten viestintäkeinojen hyödyntämistä kommentointijärjestelmässä. Yhdessä kommentissa toivottiin myös Microsoft Messenger –pikaviestiohjelman kaltaista reaaliaikaista keskustelumahdollisuutta. Palautteissa viitattiin myös kommentoijien ajankäyttöresurssien varaamiseen; aikaa kommentointityöhön kuulostaa olevan hankalaa löytää. Seuraavassa esitetty muutamia esimerkkejä saaduista kehitysajatuksista:

"Kommentointiin sitoutuneille voisi olla painike, jolla he ilmaisisivat, että he ovat katsoneet sivut ja heidän puolestaan sivu on ok. Siitä tietäisi, että he ovat olleet aktiivisia."

Muiden töiden takia ei aina ollut aikaa katsoa kommentoitavaa toimintoa ennen kuin kommentointiaika oli jo ohi."

"Pidin järjestelmästä. Se oli selkeä ja helppokäyttöinen. Värikoodit selkeyttivät toimintaa (siis esim. se että niissä missä kommentointiaika oli päättynyt näkyivät punasena)"

4.7 Tulosten luotettavuus ja käyttöarvo

Arvioitaessa tutkimuksen tulosten luotettavuutta on ensimmäiseksi kiinnitettävä huomiota vastaajaryhmän kokoon. 43 vastaajan ryhmä, joka osassa kysymyksiä rajautui vielä pienemmäksi, on liian pieni luotettavien tulosten esittämiseen. Myös 19% vastausaktiivisuus ei ollut kovin suuri. Vastausaktiivisuudessa on kuitenkin huomioitava,

että läheskään kaikki kommentointiryhmien jäsenet eivät koskaan osallistuneet kommentointiin ainakaan aktiivisesti, eli kirjoittamalla uusia kommentteja. Niistä henkilöistä, jotka osallistuivat kommentointiin aktiivisesti näissä projekteissa, on 43 vastaajaa merkittävästi suurempi kuin 19%. Saadut tulokset antavat kuitenkin hyvää tietoa verrattuna entiseen tilanteeseen, jossa minkäänlaista tutkimustietoa ei ollut ollenkaan olemassa. Sen vuoksi tuloksilla on merkittävää käyttöarvoa.

Muutamien samankaltaisia asioita mittaavien kysymysten korrelaation perusteella voidaan arvioida, että tutkimuksen sisäinen luotettavuus on hyvä, sillä saadut korrelaatiokertoimet olivat odotetunlaisia. Mittauksen ulkoisen luotettavuuden arviointi vaatisi kyselyn toistamista.

Tuloksia arvioitaessa ovat riskinä luotettavuudelle myös tulostietojen käsittelyssä ja tulkinnassa mahdollisesti tapahtuneet virheet. Ne on kuitenkin pyritty eliminoimaan täysin sähköisellä tulosten käsittelyllä ja manuaalisen tietojen kirjauksen minimoimisella.

4.8 Tulosten ja hypoteesien vastaavuus

Ensimmäisessä hypoteesissä oletettiin, että FNS-menetelmä nopeuttaa ohjelmistokehitystyötä. Tämä pitää saatujen tuloksien perusteella paikkansa: keskimääräiseksi aikataulusäästöksi saatiin 13%. Myös toisen hypoteesin oletus kustannussäästöistä saatiin todistetuksi säästön ollessa perinteisiin menetelmiin verrattuna keskimäärin 13%.

Kolmas hypoteesi esitti oletuksen, että menetelmällä toteutetut ohjelmistot vastaavat hyvin asiakkaiden tarpeita. Tätä tutkittiin kyselytutkimuksen alkupuolella ohjelmistotuotteiden laatua mitanneissa kysymyksissä. Tuloksina saatiin tieto siitä, että vastaajat ovat keskimäärin tyytyväisiä rakennetun järjestelmään toimintaan. Vastaajat olivat myös samaa mieltä järjestelmän toimimisesta hyvin siinä tarkoituksessa, johon se on suunniteltu. Samoin samaa mieltä oltiin myös väittämästä ”Järjestelmä toteuttaa minulle tärkeät toiminnallisuudet”. Tämän perusteella voidaan sanoa myös kolmannen hypoteesin toteutuvan.

Viimeisessä hypoteesissä oletettiin, että kommentoijat käyttäjät mielellään kommentointijärjestelmää. Tähänkin liittyviin kysymyksiin saatiin positiivisia vastauksia: Järjestelmää pidettiin helppokäyttöisenä, järjestelmän Internetissä toimivuuden tarjoamaa kommentointiajan ja paikan vapautta arvostettiin sekä mielipiteet saatiin esille järjestelmän kautta. Tärkeimpänä tämän hypoteesin vahvistavana tuloksena saatiin tieto siitä, että vastaajat haluavat käyttää järjestelmää myös uudelleen.

5. Johtopäätökset ja pohdinta

5.1 Järjestelmän vahvuudet ja heikkoudet

FNS-menetelmän selkeitä vahvuuksia voidaan löytää niin tehtäessä vertailua yleisesti tunnettuihin ketteriin menetelmiin kuin kyselytutkimuksen tuloksia analysoitaessakin. Keskeisin vahvuustekijä on kommentointijärjestelmän myötä asiakkaalle tarjoutuva mahdollisuus osallistua ohjelmistokehitykseen ajasta ja paikasta riippumattomasti. Tämä seikka oli kyselytutkimuksen tulosten perusteella selkeästi hyvänä pidetty ominaisuus. Muitten ketterien ohjelmistokehitysmenetelmien painottaessa asiakkaan fyysistä paikallaoloa ja kasvokkain tapahtuvaa keskustelua on tehokkaan hajautetun toiminnan mahdollistava sähköiseen viestintään nojautuva vaihtoehto hyvä kilpailuvaltti.

Yrityksen omalle toiminnalle FNS-menetelmä antaa mahdollisuuden toimia tehokkaasti. Asiakkaan luona käyntiä, palavereita sekä näistä aiheutuvaa matkustamista voidaan vähentää merkittävästi kommentointijärjestelmää käyttämällä. Tämä tarjoaa kustannussäästömahdollisuuksia, jotka näkyvät myös asiakkaalle palveluiden hinnoissa. Kyselytutkimuksessa saatiin selkeästi tieto siitä, että menetelmä säästää rahaa (13% kustannussäästö). FNS-menetelmän tapa edetä käyttöliittymämalleja tekemällä antaa toteuttajille myös mahdollisuuden tarjota asiakkaalle aina teknisesti yksinkertaisinta ratkaisua, joka ratkaisee esitetyn ongelman. Näin pystytään vähentämään turhien, teknistä toteutusta merkittävästi monimutkaistavien mutta ei ollenkaan tai vain vähän lisähyötyä käytännössä tuottavien ominaisuuksien suunnittelu ja toteutus.

Kyselyssä kysyttiin myös vastaajien kiinnostusta järjestelmän käyttämiseen tulevaisuudessa, ja saadut tulokset olivat selkeästi positiivisia. Yrityksen käytössä on menetelmä, jota käyttäjät ovat halukkaita käyttämään jatkossakin. Tämä kannustaa käyttämään ja kehittämään menetelmää eteenpäin.

FNS-menetelmän heikkoutena ovat sitouttamiseen ja ajankäyttöön liittyvät haasteet, joita menetelmä asettaa asiakkaalle ja erityisesti kommentointiryhmälle. Kyseltytutkimuksen tuloksista selvisi, että monet vastaajat kaipasivat lisää aikaa järjestelmien kehitykseen. Lisäksi käytännössä havaittu merkittävä vaihtelu kommentointiryhmän jäsenten aktiivisuuden välillä kertoo siitä, että kaikilla ei ole aikaa tai kiinnostusta kommentoida, vaikka heidät on valittu kommentointiryhmään.

Yksi menetelmän heikkoudeksi laskettava ominaisuus on kommentointijärjestelmän rajoittuneisuus vain tekstimuotoiseen viestintään. Vaikka kyselytutkimuksen tulosten perusteella vastaajat näyttävät osaavan ilmaista mielipiteensä tekstinäkin, saatiin

avoimista vastauksista selkeästi pyyntöjä muiden viestintämenetelmien hyödyntämisestä.

FNS-menetelmän käyttäminen edellyttää yritykseltä paljon osaamista. Verrattuna kaikkiin muihin menetelmiin, niin ketteriin kuin suunnitelmaohjautuviinkin, vaatii FNS-menetelmä selkeästi enemmän perustietoa asiakkaan toiminnasta. Valmiiden ehdotusten tekeminen asiakkaalle käyttöliittymämallien avulla vaatii paljon esitietoa mallien suunnittelijoilta. Tämä edellyttää ennen työn aloitusta tutustumista asiakkaan toimialaan ja liiketoimintaprosesseihin. Tämän tyyppistä osaamista ei ainakaan tässä laajuudessa edellytetä muissa esitellyissä menetelmissä. Lisäksi FNS-menetelmän käyttö edellyttää ohjelmoijilta paljon taitoa; muutoksiin on sopeuduttava nopeasti ja sovitut muutokset tulee toteuttaa virheettömästi, pitäen yksittäisen muutoksen rinnalla mielessä asiakkaan tavoitteet koko järjestelmältä. Myös suorituskyky ja koko järjestelmän luotettavuus tulee huomioida muutoksia toteutettaessa. Suunnitelmaohjautuvia menetelmiä käytettäessä ohjelmoijille asetettavat taitovaatimukset eivät ole läheskään yhtä kovat.

5.2 Ehdotukset menetelmän kehittämiseksi

Kyselytutkimuksen avoimet kysymykset antoivat muutamia selkeitä toiveita FNS-menetelmän keskeisimmän osuuden, kommentointijärjestelmän kehittämiseksi. Eniten toivottu asia oli äänestysmahdollisuuden lisääminen kommentointipalstalle, jolloin kommentointipalstat voisivat toimia myös päätöksenteon työkaluna: tällä hetkellä päätökset muutoksista pitää tehdä kommentointijärjestelmän ulkopuolella, ja enintään kirjoittaa näkyviin kommentointipalstalle.

Myös kuvien ja kaavioiden lisäysmahdollisuutta toivottiin kommentointijärjestelmään liittyvissä avoimissa kommenteissa. Tämä on selkeästi järkevä laajennus. Tällä hetkellä käytössä olleet kommentointipalstat ovat tukeneet vain leipätekstimuotoista kommentointia, mutta samalla kun äänestys- ja kuvatoimintoja kehitetään, on nykyaikaisten tekstin editointitoimintojen (lihavointi, kursivointi ym.) lisääminen myös ehdottomasti järkevää. Myös ääniviestien tallentaminen voisi tulla kyseeseen lähitulevaisuudessa. Ei liene myöskään mahdotonta ajatella mahdollisuutta reaaliaikaisen viestintämahdollisuuden tarjoamiseen – joko tekstikeskusteluna tai miksi ei äänikeskustelunakin nykyaikaisten pikaviestiohjelmien tapaan.

Nämä edellä kuvatut kehitysmahdollisuudet liittyvät ihmisten erilaisten viestintätapojen huomioon ottamiseen: kaikille mielipiteen ilmaiseminen tekstiä kirjoittamalla ei ole tehokkain tapa viestiä, joskin kyselyn mukaan se näyttää vastaajilta kuitenkin

onnistuvan hyvin. Tekniikka antaa kuitenkin nykyään mahdollisuuksia muidenkin viestintätapojen huomiointiin, joten mahdollisuus kannattaa hyödyntää.

Kommentointiryhmän sitouttaminen on asiakkaiden ja FNS:n yhteinen haaste. Menetelmään tulisi siksi lisätä kommentointiryhmän jäsenten ohjeistusta erityisesti ennen kommentoinnin aloitusta. Myös ryhmän valinnassa tulee ottaa huomioon, että kommentoijilla on oltava realistiset mahdollisuudet osallistua kommentointiin. Tämä tärkeä asia tulee välittää erityisesti kommentointiryhmää kokoavan asiakkaan yhteyshenkilön tietoon.

Muutoin kuin kommentointijärjestelmän osalta FNS-menetelmän kehitystyössä kannattaa huomioida muiden ketterien menetelmien parhaiden osien hyödyntäminen; esimerkiksi XP-menetelmän käyttämät tarinakortit ja niiden ryhmittely sopisivat hyvin FNS-menetelmän laajennukseksi. Tarinakorttien ja niiden lajitteluun käytetyn ilmoitustaulun toteutus sähköisenä versiona sopisi FNS-menetelmään hyvin; tällöin työ pysyisi edelleen riippumattomana fyysisestä paikasta. Scrum-menetelmän yhteydessä esitellyt tuotteen ja sprintin työlistat ovat toinen selkeä lisä, joiden liittäminen FNS-menetelmän osaksi kannattaa selvittää.

FNS-menetelmä kaipaa myös tiedotuksen tehostamista: kyselyn koko projektia koskevissa avoimissa vastauksissa kävi ilmi, että riittävää tietoa projektin vaiheesta eikä päättymisestä ollut tullut kaikille kommentoijille. Selkeän, projektin kokonaisvaiheen ja aikataulun raportointi ajankohtaiset kommentoinnit kertovien viestien lisäksi olisivat hyvä lisä.

5.3 Jatkotutkimusaiheita

FNS-menetelmän jatkotutkimusaiheet liittyvät pääasiassa kommentointijärjestelmään ja sen käytön vaikutusten tarkempaan mittaamiseen. Vastaavan kyselyn toistaminen laajemmalle vastaajajoukolle ja erityisesti suurempi vastausmäärä antaisi tuloksille paremman luotettavuuden. Kyselyn uusimista sellaisenaankin voidaan suositella, kunhan kommentointijärjestelmän aktiivikäyttäjiä on saatu lisää, ja kyselyyn vastaaminen tehdään riittävän houkuttelevaksi.

Kiinnostava jatkotutkimusmahdollisuus voi olla myös kehitystyöhön osallistuvien henkilöiden taustaominaisuuksien, kuten koulutuksen ja yleisen tietotekniikan käyttötaidon selvittäminen taustatieto-osiossa. On mahdollista, että henkilöiden erilaiset taustatekijät saattavat vaikuttaa kommentoijien antamiin mielipiteisiin järjestelmästä merkittävästikin. Netissä toteutettavan lomakemuotoisen kyselytutkimuksen lisäksi myös kommentoijien henkilökohtaiset haastattelut tai laajempien avointen vastausten

pyytäminen kyselyn yhteydessä antaisivat todennäköisesti myös erittäin hyvää tietoa järjestelmän jatkokehittämiseksi.

Läheskään kaikki kommentointiryhmiin valitut henkilöt eivät koskaan osallistuneet kommentoimalla mihinkään kommentoinnissa olleeseen asiaan. Tutkimusta voitaisiin tehdä myös siitä, kuinka paljon kommentoijista käy passiivisesti selaamassa kommentointia ottamatta kuitenkaan kantaa keskusteluun. Tätä voitaisiin teknisin keinoin tilastoida. Myös kommentoimattomuuden syitä olisi mielenkiintoista selvittää; syynä ei aina voi olla ehdotuksen osuminen täysin kohdalleen. Kiinnostavaa olisi myös selvittää, olisivatko nämä henkilöt kommentoineet toimintoja kasvotusten keskusteltaessa. eli vaikuttaako viestintapa mielipiteen ilmaisuhalukkuuteen. Oletettavammin myös kynnys palaveriosallistumisen peruuttamisen on suurempi kuin kommentointipalstan kysymykseen vastaamatta jättämiseen.

Myös kirjallisuuteen perustuvaa jatkotutkimusta olisi mahdollista suorittaa erityisesti ketteriin kehitysmenetelmiin liittyen. FNS-menetelmää voidaan kehittää edelleen paljon tehokkaammaksi tutkimalla lisää ketterästä sovelluskehityksestä olemassa olevaa aineistoa ja liittämällä niissä hyväksi todettuja käytäntöjä osaksi FNS-menetelmää.

6. Yhteenveto

Työn alussa asetettiin ensimmäiseksi tutkimuskysymykseksi etsiä vastaus siihen mikä FNS:n ohjelmistokehitysmenetelmä on, ja eritellä sen uudet, muista menetelmistä poikkeavat piirteet. Tämä toteutettiin esittelemällä ohjelmistokehitysmenetelmän toiminta vaihe vaiheelta sopimus- asiakas ja toteutusnäkökulmista tarjousvaiheesta ylläpitovaiheeseen asti. FNS-menetelmäksi nimetyn ohjelmistokehitysmenetelmän ominaispiirteiksi eriteltiin tärkeimpinä ominaisuuksina kommentointijärjestelmän käyttö ensisijaisena viestintätapana ja käyttöliittymämallien keskeinen rooli määrittelyssä.

Toiseen tutkimuskysymykseen, jossa kysyttiin asiakkaiden ja ohjelmiston kehittämiseen osallistuvien henkilöiden saavuttamia etuja, saatiin tunnettuihin menetelmiin tehdyn vertailun lisäksi vastauksia kyselytutkimuksen tuloksista. Yhdeksi keskeiseksi asiakasyritysten saavuttamaksi eduksi voidaan kustannus- ja aikataulusäästöjen lisäksi lukea laajan kommentoijaryhmän osallistumisen mahdollistuminen; ilman Internet-kommentointia läheskään saman laajuisten ryhmien osallistuminen ohjelmistokehitykseen ei olisi ollut mitenkään mahdollista. FNS-menetelmää yleisesti tunnettuihin ketteriin kehitysmenetelmiin vertailtaessa todettiin kehitystyöhön osallistuville kommentoijille koituvaksi selkeäksi eduksi riippumattomuus fyysisestä ajasta ja paikasta. Tästä pidettiin myös kyselytutkimuksen perusteella odotetusti hyvänä ominaisuutena.

Kyselytutkimuksella selvitetyt kommentoijien mielipiteet vastasivat tutkimuskysymykseen, joka määritteli työn tehtäväksi selvittää käyttökokemuksia kommentointijärjestelmästä. Tutkimus vastasi kokemusten olevan hyviä. Järjestelmän käyttöä pidettiin helppona, vastaajat kertoivat saaneensa sen kautta mielipiteensä esille ja toisten kirjoittamien kommenttien näkemistä arvostettiin. Tärkeimpänä tuloksena voidaan nostaa esille se, että selkeästi suurin osa vastaajista haluaisi käyttää järjestelmää myös uudelleen ohjelmistokehityksen apuvälineenä.

FNS-menetelmän heikkouksiksi todettiin haasteet, jotka liittyvät kommentoijien aktivoimiseen ja kommentointiajan riittävyyteen. Monet vastaajat kertoivat erityisesti avoimissa kommenteissa, ettei heillä ollut riittävästi aikaa kommentoida toimintoja. Samoin myös rajoittuneisuus keskustelupalstamuotoiseen viestintään on yksi kommentointijärjestelmän heikkous, vaikka kyselytuloksien perusteella kommenttien esittäminen kirjallisessa muodossa keskimäärin onnistuu, toivottiin kommentointijärjestelmään myös visuaalisten viestintämahdollisuuksien hyödyntämismahdollisuutta.

Parantamisen mahdollisuuksia voidaan löytää myös tutkimalla ketteriä menetelmiä tarkemmin ja liittämällä niiden parhaita ominaisuuksia FNS-menetelmän osaksi.

Lopuksi voidaan todeta, että työssä FNS:n ohjelmistokehitysmenetelmä todettiin omanlaiseksensa ketteräksi menetelmäksi, joka lainaa paljon toiminnastaan muilta menetelmiltä, mutta tekee erityisesti viestintään liittyvät asiat muista poikkeavalla tavalla. Kyselytutkimuksesta saaduista hyvistä tuloksista huolimatta menetelmää on tarpeen kehittää edelleen pitäen mielessä tavoiteltava päämäärä: laadukkaiden, asiakkaiden tarpeisiin täydellisesti sopivien ohjelmistojen kehittämiseen tehokkaasti ja tuottavasti.

Lähteet

Biddle, R., Marshall S., Noble, N.; (2004): Less Extreme Programming, Conferences in Research And Practice in Information Technology Vol 30, s. 217-226. Australian Computer Society Inc.

Beck, K., Beedle, M., Bennekum, A., Cockburn, A.: (2001): Manifesto for Agile Software Development. Saatavilla WWW-muodossa <URL: http://www.agilemanifesto.org/>, viitattu 28.7.2006

Beck, K. (2004): Extreme Programming Explained – Embrace change, Second Edition. Addison-Wesley, Boston.

Boehm, B. (2002): Get ready For the Agile Methods, With Care. Computer 35(1): s. 64-69.

Boehm, B. & Turner, R. (2004): Balancing Agility and Discipline: A Guide for the Perplexed. Pearson Education, Inc.

Etu-palvelut (2006): Www-sivusto, https://www.etu-palvelut.net/

Haikala, I. ja Märijärvi J. (2002): Ohjelmistotuotanto. Suomen ATK-kustannus Oy

Halonen K., Kemppainen, O., Laitinen S., Niittylampi, K., Rautiainen, J. ja Utriainen O. (2001): TIE330 Ohjelmistoprojektien hallinta vesiputous vs. spiraalimallilla. Ohjelmistotuotanto -kurssin harjoitustyö, Jyväskylän yliopisto. Saatavilla www-muodossa: <URL:http://193.167.110.132/marko.forsell/Harjoitustyot/R_14b.pdf>, viitattu 13.8.2006

Heikkilä, T. (2005): Tilastollinen Tutkimus. Edita, Helsinki.

Highsmith, J. (2002): Does Agility Work? Dr. Dobb's Portal: The World of Software development. Saatavilla www-muodossa: <URL: http://www.ddj.com/dept/architect/184414858>, viitattu 23.8.2006.

Highsmith, J. (2000): Retiring lifecycle dinosaurs. Software Testing and Quality Engineering, July/August. s. 22-28.

Hiltunen, M. (2004): Johdatus tietojärjestelmiin, verkkokurssimateriaali. Oulun kauppaoppilaitos. Saatavissa www-muodossa: <http://www.okol.org/verkkokurssit/datanomi/tietojarjestelmien_kaytto_ja_kehittaminen/johdatus_tietojarjestelmiin/johdatus_tietojarjestelmiin.html>, viitattu 15.8.2006.

Hintikka, K. & Mielonen, S. (1998): Web-palveluiden käytettävyys ja tuotanto. Taideteollinen korkeakoulu, Koulutuskeskus, Mediastudio. Saatavissa www-muodossa <URL:http://www2.uiah.fi/mediastudio/survey4/>, Viitattu 13.8.2006

Huhtamäki, J. (2005): Kohti ketterää ohjelmistokehitystä, Hypermedian ohjelmointi kevät 2005, Luentokalvot luento 14, Tampereen teknillinen yliopisto. Saatavissa www-muodossa <URL:http://matwww.ee.tut.fi/hmopetus/hm-ohj/2005/pruju/hmohj05-132-143.pdf>, Viitattu 28.7.2006

Jeffries, R.E. (2001): What is Extreme Programming?
Saatavissa www-muodossa: <URL:http://www.xprogramming.com/xpmag/whatisxp.htm >, viitattu 7.8.2006

Kalermo J. & Rissanen J. (2002): Agile software development in theory and practice. Software Business Program, Master's thesis 6.8.2002, University of Jyväskylä, Department of Computer Science and Information Systems, Jyväskylä. Saatavissa www-muodossa
<URL:www.cs.jyu.fi/sb/Publications/KalermoRissanen_MastersThesis_060802.pdf>, Viitattu 28.7.2006

Kangas, J. (2003): Sulautetun ohjelmiston suunnitteluprosessi kehityshanke. Diplomityö, Teknillinen korkeakoulu, Sähkö- ja tietoliikenetekniikan osasto.

Kosonen, S. (2005): Ohjelmoinnin opetus Extreme Programming -hengessä. Tietotekniikan pro gradu-tutkielma 8. elokuuta 2005. Jyväskylän yliopisto, Tietotekniikan laitos, Jyväskylä.

Liedenpohja, H. (2006): Ketterät ohjelmistoprosessit ja laadunhallinta. Seminaarityö (luonnos), Helsingin yliopisto, Tietojenkäsittelytieteen laitos. Helsinki 13.3.2006.

Lindberg, H. 2003: Extreme Programming. Tampereen yliopisto, Tietojenkäsittelytieteiden laitos, Tietojenkäsittelyoppi, Pro gradu –tutkielma

Maurer F. ja Martel S. (2002): Extreme programming: Rapid development for Web-based applications. IEEE Internet Computing 6(1): 86-90

Mountain Goat Software (2006): Kotisivut. Saatavissa www-muodossa: <URL:http://www.mountaingoatsoftware.com/>, viitattu 15.8.2006

Niittymies, T. (2006): ”Suomalainen softateollisuus venyttelee ketterämmäksi”. T-lehti: Nro 3/2006. s. 13

OAMK (Oulun seudun ammattikorkeakoulu) 2006: Software Business Competence – www-sivusto. Saatavissa www-muodossa: <URL: http://www.oamk.fi/sbc/index.htm>. Viitattu 10.12.2006

Pressman, R. (2005): Software Engineering: A Practitioner's approach. Sixth Edition. McGraw-Hill International Edition. Singapore.

Riekkinen, J. (2006): Vaatimusmäärittelyn vaikeus ja laatuvaikutukset. Seminaarityö, Helsingin yliopisto, Tietojenkäsittelytieteen laitos. Helsinki 20.4.2006. Saatavissa www-muodossa: <URL: http://www.cs.helsinki.fi/u/rkkauppi/laadunhallinta/lopulliset/riekkinen.pdf> viitattu 29.8.2006

Sommerville, I. (1992): Software Engineering, Fourth Edition. Addison-Wesley.

Tervonen, I. (2002): Nykyaikaiset tarkastus- ja testausmenetelmät ohjelmistotyön laadun tukena. Oulun yliopisto. Improved Inspection Initiative Group in Oulu (i3GO). Esitys 25.4.2002. Saatavilla www-muodossa: <URL:http://www.tol.oulu.fi/i3/presentations/Tervonen_Potky_2002.pdf>, Viitattu 13.8.2006

Turk, D., France, R. ja Bernhard, R. (2002): Limitations of Agile Software Processes. Third International Conference on Extreme Programming and Flexible Processes in Software Engineering, XP2002, May 26-30, Alghero, Italy, pg. 43-46, 2002. Saatavissa www-muodossa: <URL: http://wwwbroy.informatik.tu-muenchen.de/~rumpe/ps/XP02.Limitations.pdf >, viitattu 23.8.2006

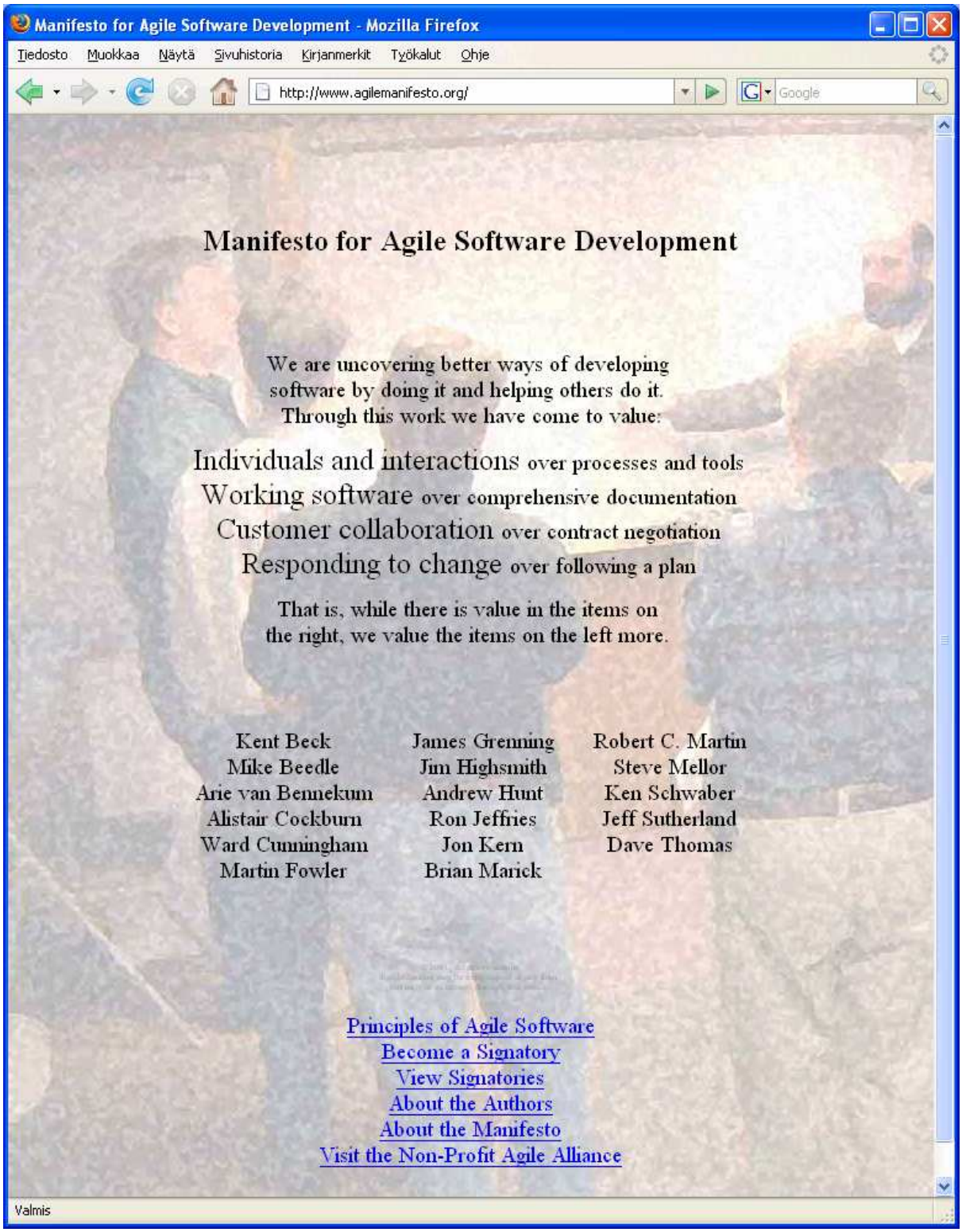
Van Vliet, H. (2000): Software Engineering, Principles and Practice, Second edition. John Wiley & Sons, Ltd. England

Wirgentius, M. (2003): Agile-menetelmät ja Crystal Clear ohjelmistotuotantoprojektissa. Insinöörityö 15.3.2003. Espoo-Vantaan teknillinen ammattikorkeakoulu.

Wikipedia (2006): W.W. Royce. Saatavissa www-muodossa: <http://en.wikipedia.org/wiki/W._W._Royce>, viitattu 3.1.2007

Liitteet

Liite 1: Agile manifesto



Liite 2: Esimerkki kommentointiohjeista

Kommentointiohjeet

Käyttäjien tarpeiden, kehitysehdotusten ja mielipiteiden kerääminen Naseva-järjestelmästä tapahtuu verkossa kommentointipalstatoiminnon avulla. Kommentointipalsta on nähtävissä nyt selaamassasi Nasevan kommentointiversiossa.

Nasevan kommentointiversio on luonnos (demoversio) todellisesta Naseva-järjestelmästä. Luonnos esittelee lähinnä Naseva-järjestelmän käyttöliittymää. Se mallintaa järjestelmään syötettäviä, siellä näytettäviä ja sieltä haettavia tietoja. Todellinen, varsinaisen tietokannan sisältävä järjestelmä toteutetaan tämän luonnoksen pohjalta. Kommentointiversio Nasevasta on nähtävissä koko ajan aina siihen asti, kun järjestelmä on täysin valmis ja otettu käyttöön.

Kommentointi tapahtuu sivuilla olevista **"Kirjoita / näytä kommentit"** -linkeistä. Kulloinkin kommentoitavana olevat toiminnot näet etusivulla olevasta listasta. Listassa on myös kerrottu, millä käyttäjäryhmällä ja mistä valikosta kyseisen kommentoitavan sivun löytää.

Muutamia ohjeita kommentoijille:

- Käy säännöllisesti, vähintään kerran viikossa, Naseva-kommentointisivuilla osoitteessa http://kommentointi.naseva.fi/
- Seuraa sähköpostiasi! Uusista avatuista kommentoinneista ilmoitetaan aina sähköpostitse.
- Kerro myös, jos olet yhtä mieltä jostakin ehdotuksesta muiden kanssa; näin tiedämme, että annetulle ehdotuksella on kannatusta
- Myös "ei muutostarvetta" -tyyppiset kommentit ovat tervetulleita. Ne kertovat, että ehdotustamme ei tarvitse muuttaa.
- Kommentointiaikaa on yksi viikko. Kommentointi tulee suorittaa kommentointiaikana.
- Annettujen kommenttien perusteella kommentoinissa oleviin sivuihin voidaan tehdä pieniä muutoksia myös kommentointiajan kuluessa
- Äkä kommentoi vielä sivuja/toimintoja, jotka eivät vielä ole kommentoitavana. Niiden aika tulee pian.
- Käytä mielipiteiden kertomiseen ensisijaisesti kommentointipalstaa, älä esimerkiksi sähköpostia. Näin tiedot tulevat heti kaikkien nähtäville.
- Kommettejasi tarvitaan - osallistu aktiivisesti!

Jos kommentoinnissa on ongelmia, niin ota heti yhteyttä FNS:ään:

Janne Huttunen
040-589 2468
janne.huttunen@fns.fi

Olli Ojala
040-720 8267
olli.ojala@fns.fi

Liite 3: Kyselyn kutsuviesti

Hyvä vastaanottaja

Teen Teknillisessä korkeakoulussa diplomityötäni Finnish Net Solutions Oy:n tietojärjestelmien kehityksessä käyttämästä Internet-pohjaisesta kommentointijärjestelmästä. Järjestelmä on FNS:lle erittäin käyttökelpoinen, ja tarjoaa yritykselle merkittävän aikataulu- ja kustannusedun perinteisiin kehitysmenetelmiin verrattuna. Työhön liittyen teen kyselytutkimusta, johon pyydän myös sinun vastaavan. Tutkimuksen tarkoituksena on selvittää käyttäjien kokemukset kommentointijärjestelmästä. Saatujen tutkimustulosten perusteella jatkamme kommentointijärjestelmän kehitystyötä.

Tämä kysely lähetetään kaikille, jotka osallistuivat Provet YES -potilasohjelmiston kehitystyöhön. Tämän tietojärjestelmän kehitystyö toteutettiin netissä FNS:n kommentointijärjestelmän avulla. Lisäksi kyselyyn osallistuvat kahden muun tietojärjestelmän kommentointiryhmät. Kaikkiaan kysely lähetetään 238 eri henkilölle.

Pyydän sinua auttamaan tutkimustyössä ja vastaamaan tutkimuslomakkeen kysymyksiin. Kyselyyn vastataan www-sivulla osoitteessa http://www.fns.fi/kysely/ .
Vastausaikaa on sunnuntaihin 26.11.2006 klo 12.00 asti.

Annetut vastaukset käsitellään nimettöminä. Varsinaisen kyselyn jälkeen pyydetään eri lomakkeella vapaaehtoiset yhteystiedot arvontaa varten, jossa palkintona on Apple iPod Nano -musiikkisoitin 2Gt muistilla.

Annan tarvittaessa mielelläni lisätietoja tutkimuksesta.

Kiitos vastauksistasi jo etukäteen!

Ystävällisin terveisin,
Janne Huttunen
Finnish Net Solutions Oy
p. 040-589 2468, janne.huttunen@fns.fi

Liite 4: Kyselylomake (www-sivu)

Tervetuloa vastaamaan kyselyyn Finnish Net Solutions Oy:n kommentointijärjestelmästä

Olet saanut kutsun tulla vastaamaan FNS Oy:n kyselytutkimukseen.

Sinut on kutsuttu mukaan, koska olet osallistunut jonkin toteuttamamme tietojärjestelmän kehitykseen toimimalla kommentointiryhmän jäsenenä. Pyydämme sinua vastaamaan, sillä olemme kiinnostuneita mielipiteestäsi koskien kommentointijärjestelmää ja sen avulla toteutettuja tietojärjestelmiä.

Tämän kyselyn tavoitteena on selvittää, millaisia aika- ja kustannusvaikutuksia kommentointimenetelmästä on, ja miten sen käyttö järjestelmien kehitysvaiheessa vaikuttaa työn tuloksena valmistuvan tietojärjestelmän laatuun. Saatujen tulosten perusteella voimme jatkaa kommentointijärjestelmän kehitystyötä.
Kysely kuuluu Janne Huttusen Teknillisessä korkeakoulussa valmistumassa olevaan diplomityöhön.

Kyselyyn vastaan nimettömänä, mutta kyselyn jälkeen voit antaa yhteystietosi arvontaa varten. Yhteystietonsa antaneiden vastaajien kesken arvotaan **Apple iPod Nano MP3-soitin 2Gt muistilla** (arvo n. 200 €).

Kyselyyn vastaaminen kestää noin viisi minuuttia. Kiitos vastauksistasi!

Osio 1/4: Taustatietokysymykset

1. Mihin seuraavista FNS:n toteuttamista projekteista osallistuit?

(Jos olet osallistunut useampaan, valitse sopivin. Vastaa jatkossa valitsemaasi järjestelmää ajatellen.)

- ○ Sikaloiden terveysluokitusrekisteri (Etu-palvelut)
- ○ Naseva
- ○ Provet YES (Yliopistollisen eläinsairaalan potilastietojärjestelmä)

2. Kuulun parhaiten seuraavaan käyttäjäryhmään:

Etu-palvelut ja Naseva

- ○ Maatila
- ○ Eläinlääkäri
- ○ Teurastamo tai meijeri
- ○ Järjestelmän hallinto / ministeriö
- ○ Maa- ja metsätalousministeriö / Evira
- ○ Muu sidosryhmä

Provet YES

- ○ Eläinlääkäri/opiskelija/hoitaja
- ○ Laboratorio/röntgen/patologia
- ○ Hallinto ja talous/kirjanpito
- ○ Toimistohenkilökunta
- ○ Muu sidosryhmä

3. Olitko tässä kehitysprojektissa järjestelmän kehityskustannuksista vastaavassa roolissa tai muuten taloudellisessa vastuussa projektista?

○ En ○ Kyllä

4. Oletko osallistunut koskaan aikaisemmin ennen tätä projektia jonkin muun tietojärjestelmän tai tietokoneohjelmiston kehityshankkeeseen?

○ En ○ Kyllä

Osio 2/4: Toteutetun järjestelmän kokonaisuuden arviointi

Seuraavassa esitetään väitteitä toteutettuun järjestelmään liittyen yleisesti. Valitse vaihtoehdoista sopivin.

5. Olen tyytyväinen rakennetun järjestelmän toimintaan yleisesti

○ Täysin samaa mieltä ○ Samaa mieltä ○ En osaa sanoa ○ Eri mieltä ○ Täysin eri mieltä

6. Järjestelmä toimii hyvin tarkoituksessa, johon se on suunniteltu

○ Täysin samaa mieltä ○ Samaa mieltä ○ En osaa sanoa ○ Eri mieltä ○ Täysin eri mieltä

7. Järjestelmän toteutusprojekti ei pysynyt aikataulussa, joka sille oli asetettu

○ Täysin samaa mieltä ○ Samaa mieltä ○ En osaa sanoa ○ Eri mieltä ○ Täysin eri mieltä

8. Järjestelmä toteuttaa minulle tärkeät toiminnallisuudet

○ Täysin samaa mieltä ○ Samaa mieltä ○ En osaa sanoa ○ Eri mieltä ○ Täysin eri mieltä

9. Järjestelmä ei valmistuessaan täyttänyt sille asetettuja toimintatavoitteita

○ Täysin samaa mieltä ○ Samaa mieltä ○ En osaa sanoa ○ Eri mieltä ○ Täysin eri mieltä

Vastaa seuraaviin kahteen kysymykseen (10 ja 11) vain, jos olit kehitystyötyössä kustannuksista vastaavana henkilönä

10. Tarjouksessa ja sopimuksessa sovitut kustannukset eivät pitäneet paikkansa

○ Täysin samaa mieltä ○ Samaa mieltä ○ En osaa sanoa ○ Eri mieltä ○ Täysin eri mieltä

11. Järjestelmän laatu/hinta –suhde on hyvä

○ Täysin samaa mieltä ○ Samaa mieltä ○ En osaa sanoa ○ Eri mieltä ○ Täysin eri mieltä

Vastaa seuraavaan kysymykseen (12) vain, jos olet aiemmin osallistunut ohjelmistokehitystyöhön

12. Järjestelmän toteutus onnistui huonommin kuin ne ohjelmistoprojektit, joihin olen aiemmin osallistunut

○ Täysin samaa mieltä ○ Samaa mieltä ○ En osaa sanoa ○ Eri mieltä ○ Täysin eri mieltä

13. Kysymys kaikille: Anna kouluarvosana asteikolla 4-10 koko projektille

○ 4 ○ 5 ○ 6 ○ 7 ○ 8 ○ 9 ○ 10

Osio 3/4: Kysymykset kommentointijärjestelmästä

Seuraavassa esitetään väitteitä FNS:n kommentointijärjestelmästä, jota käytettiin järjestelmän kehittämisessä. Valitse vaihtoehdoista sopivin.

14. Sain mielipiteeni hyvin esille kommentointijärjestelmän kautta

○ Täysin samaa mieltä ○ Samaa mieltä ○ En osaa sanoa ○ Eri mieltä ○ Täysin eri mieltä

15. Kommentointijärjestelmä sopii tapaani viestiä

○ Täysin samaa mieltä ○ Samaa mieltä ○ En osaa sanoa ○ Eri mieltä ○ Täysin eri mieltä

16. Kommentointijärjestelmää oli vaikea käyttää

○ Täysin samaa mieltä ○ Samaa mieltä ○ En osaa sanoa ○ Eri mieltä ○ Täysin eri mieltä

17. Kommentointijärjestelmä sopii korvaamaan palavereita

○ Täysin samaa mieltä ○ Samaa mieltä ○ En osaa sanoa ○ Eri mieltä ○ Täysin eri mieltä

18. Mahdollisuus kommentoida Intenetin kautta missä tahansa ja milloin tahansa helpotti järjestelmän kehitystyötä

○ Täysin samaa mieltä ○ Samaa mieltä ○ En osaa sanoa ○ Eri mieltä ○ Täysin eri mieltä

19. Jos osallistun jatkossa tietojärjestelmien kehitystyöhön, haluaisin käyttää vastaavaa kommentointijärjestelmää

○ Täysin samaa mieltä ○ Samaa mieltä ○ En osaa sanoa ○ Eri mieltä ○ Täysin eri mieltä

20. Sovittu kommentointiaika (7vrk) riitti minulle hyvin

○ Täysin samaa mieltä ○ Samaa mieltä ○ En osaa sanoa ○ Eri mieltä ○ Täysin eri mieltä

21. Kuvailisin tarvitsemani toiminnot mielellään paperilla tekstinä ja piirroksina

○ Täysin samaa mieltä ○ Samaa mieltä ○ En osaa sanoa ○ Eri mieltä ○ Täysin eri mieltä

22. Esittäisin omat ehdotukseni mieluummin suullisesti kuin kirjallisesti

○ Täysin samaa mieltä ○ Samaa mieltä ○ En osaa sanoa ○ Eri mieltä ○ Täysin eri mieltä

23. Toisten kirjoittamien kommenttien näkeminen helpotti omaa kommentointiani

○ Täysin samaa mieltä ○ Samaa mieltä ○ En osaa sanoa ○ Eri mieltä ○ Täysin eri mieltä

24. Pystyn kuvaamaan kirjallisesti mielipiteeni ja ehdotukseni kommentointijärjestelmän kautta

○ Täysin samaa mieltä ○ Samaa mieltä ○ En osaa sanoa ○ Eri mieltä ○ Täysin eri mieltä

Vastaa seuraaviin kahteen kysymykseen (25 ja 26) vain, jos olet aiemmin osallistunut ohjelmistokehitystyöhön

25. Käyttäisin mieluummin tietojärjestelmien kehityksessä FNS:n kommentointimenetelmää kuin muita menetelmiä, joita on käytetty niissä projekteissa, joihin olen aiemmin osallistunut

○ Täysin samaa mieltä ○ Samaa mieltä ○ En osaa sanoa ○ Eri mieltä ○ Täysin eri mieltä

26. Kuinka paljon arvioit kommentointijärjestelmän käytön aiheuttaneen projektin viivästymistä / nopeuttaneen projektin valmistumista?

○ 0% = ei viivästymistä eikä nopeutumista

Viivästyminen	**Nopeutuminen**
○ 5%	○ 5%
○ 10%	○ 10%
○ 15%	○ 15%
○ 20%	○ 20%
○ 25% tai enemmän	○ 25% tai enemmän

Vastaa seuraaviin kolmeen kysymykseen (27,28,29) vain, jos olit taloudellisessa vastuussa projektista.

27. Kommentointijärjestelmä säästi projektin kokonaiskustannuksia

○ Täysin samaa mieltä ○ Samaa mieltä ○ En osaa sanoa ○ Eri mieltä ○ Täysin eri mieltä

28. Kommentointijärjestelmän käyttö ei säästänyt matkakustannuksia eikä aikaa

○ Täysin samaa mieltä ○ Samaa mieltä ○ En osaa sanoa ○ Eri mieltä ○ Täysin eri mieltä

29. Kuinka paljon arvioit kommentointijärjestelmän aiheuttaneen lisäkustannuksia / säästäneen kustannuksia?

○ 0% = ei lisäkustannuksia eikä säästöä

Lisäkustannukset	**Kustannussäästö**
○ 5%	○ 5%
○ 10%	○ 10%
○ 15%	○ 15%
○ 20%	○ 20%
○ 25% tai enemmän	○ 25% tai enemmän

30. Vapaat kehitysajatukset

Kerro, mitä olisit halunnut tehdä toisin...

a) ...koko projektissa?

b) ...kommentointijärjestelmässä?

31. Muu palaute

Tähän voit halutessasi kirjoittaa muun palautteen FNS:n väelle

Tallenna vastaukset