LAPPEENRANNAN TEKNILLINEN YLIOPISTO
TIETOTEKNIIKAN OSASTO

VAATIMUSTENKÄSITTELY OHJELMISTOPALVELUYRITYKSESSÄ

Diplomityön aihe on hyväksytty Lappeenrannan teknillisen yliopiston Tietotekniikan osaston osastoneuvoston kokouksessa 12.3.2003.

Työn tarkastajina toimivat professori Heikki Kälviäinen ja DI Jyri Syväoja.
Työn ohjaajana toimi DI Jyri Syväoja.

Lappeenrannassa 19.8.2003

Pekka Riiali
Korpimetsänkatu 10 B 14
53850 Lappeenranta
+358 40 543 1128
pekka.riiali@iki.fi

TIIVISTELMÄ

Lappeenrannan teknillinen yliopisto
Tietotekniikan osasto

Riiali, Pekka Tapani

Vaatimustenkäsittely ohjelmistopalveluyrityksessä

Diplomityö
2003

54 sivua, 18 kuvaa ja 5 taulukkoa.

Tarkastajat: Professori Heikki Kälviäinen, DI Jyri Syväoja
Hakusanat: Vaatimustenkäsittely, laatu, prosessikehitys
Keywords: Requirements engineering, quality, process development

Vaatimustenkäsittely on erittäin tärkeä osa-alue tehtäessä uusia ohjelmistoja. Vaatimustenkäsittely ei ole vain vaatimusmääritettydokumentin kokoamista ohjelmistoprojektin alussa vaan siihen sisältyy vaatimusten määrittely, hallinta ja todentaminen koko ohjelmiston elinkaaren ajan.

Ohjelmistopalveluyrityksessä vaatimustenkäsittelyn merkitys korostuu entisestään ja tällaisella yrityksellä on oltava toimiva vaatimustenkäsittelyprosessi. Tässä työssä esitellään vaatimustenkäsittelyn teoriaa, prosesseihin liittyvää laadunvalvontaa sekä prosessien arviointi ja -kehittämismalleja.

Työssä tarkastellaan kahden erityyppisen ohjelmistopalveluyrityksen vaatimustenkäsittelyä ja esitetään havaintoja prosessimalleista. Työn tuloksena esitetään johtopäätöksiä vaatimustenkäsittelystä ja siihen liittyvistä prosesseista sekä laadunvalvonnasta.

ABSTRACT

Lappeenranta University of Technology
Information Technology

Riiali, Pekka Tapani

Requirements engineering in a service-programming company

Master's Thesis
2003

54 pages, 18 figures and 5 tables.

Supervisors: Professor Heikki Kälviäinen, M.Sc. Jyri Syväoja
Keywords: Requirements engineering, quality, process development

Requirements engineering is a very important part when making new software systems. Requirements engineering is not just collecting the requirements specification document in the beginning of the software project but it includes specification, managing, and verification of the requirements during the whole lifecycle of the software product.

In a service-programming company requirements engineering is even more important, and such a company has to have a working process for the requirements engineering. In the thesis it will be present the theory of requirements engineering, process related quality control, and process capability and maturity models.

In this study it will be scrutinized requirements engineering of two different service-programming companies and their process models will be observed. As a result of this study, it will be present conclusions from requirements engineering, and processes associating it, and quality control.

ALKUSANAT

Tämä työ on tehty Lappeenrannassa Intellitel Communications Oy:ssä ja Sonera Oyj:ssä Lappeenrannan teknillisen yliopiston Tietotekniikan osastolle.

Kiitän diplomityöni tarkastajia, professori Heikki Kälviäistä ja DI Jyri Syväojaa, kannustuksesta ja neuvoista sekä ohjauksesta, joita olen saanut työni aikana ja varsinkin sen alkuvaiheessa.

Eritykset kiitokset myös Intellitelin laatu- ja prosessikehitysosaston väelle, joka jaksoi kuunnella ja ymmärtää ideoitani sekä mielipiteitäni koskien vaatimustenkäsittelyä.

Lämpimät kiitokset myös läheisilleni, jotka ovat jaksaneet odottaa tämän työn valmistumista, ja ovat tukeneet minua sen teossa.

SISÄLLYSLUETTELO

LYHENTEET

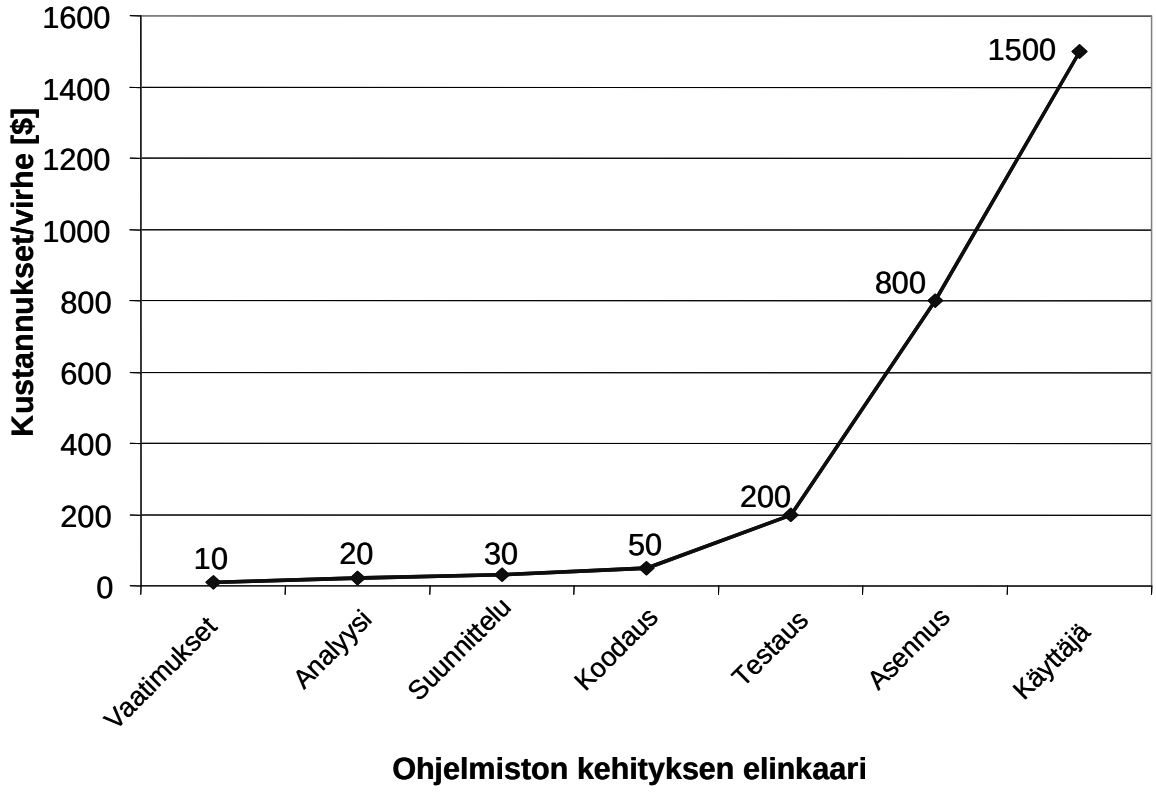
CMM	Capability Maturity Model
CM	Configuration Management
CR	Change request
DP	Decision point
FiSMA	Finnish Software Measurement Association
IEC	International Electrotechnical Commission
IEEE	Institute of Electrical and Electronics Engineers
ISO	International Organization for Standardization
MS	Milestone
NIM	Network intelligence middleware
RE	Requirements engineering
RM	Requirements management
SEI	Software Engineering Institute
SPICE	Software Process Improvement and Capability Determination
Std	Standard
UML	Unified Modelling Language

1 JOHDANTO

Tietotekniikan alalla on usein tarvetta toteuttaa tilauksesta laajoja ohjelmistoja ratkaisemaan muiden alojen ongelmia. Normaalitapauksessa toteuttaja ei kuitenkaan ole se, jolle tarve ohjelmistoon on syntynyt, vaan toteuttaja on jokin muu henkilö tai organisaatio. Ongelman ratkaisemiseen pyrkivää ohjelmiston määrittelyä, toteutusta, hallintaa ja testausta kutsutaan ohjelmistotekniikaksi, jonka avulla pyritään täyttämään ohjelmiston tilaajan asettamat tavoitteet sovitussa aikataulussa ja budjetissa.

Ohjelmiston tarvitsijan eli asiakkaan asettamat rajoitukset ja tarpeet muodostavat ohjelmistolle asetetut vaatimukset, joiden perusteella suunnittelijat ja ohjelmoijat toteuttavat ohjelmiston. Toteutettu ohjelmisto kuvastaa ennen ohjelmointia asetettuja vaatimuksia, joten tämä vaatimusmäärittelyksi kutsuttu osa ohjelmistotuotantoa on kriittistä onnistumisen kannalta. Vaatimustenkäsittely ei kuitenkaan koostu pelkästä ennen toteutusta tapahtuvasta vaatimusten määrittelystä, vaan erittäin tärkeässä roolissa on myös muutosten hallinta ja vaatimusten todentaminen.

Vaatimusten määrittelyssä on useita ongelmia, jotka voivat aiheuttaa lisäkustannuksia, lisätyötä, aikataulun viivästymistä ja jopa ohjelmiston hylkäämisen. Näiden vaatimusten todentamisen epäonnistumisen välttämistä varten on ohjelmistopalveluyrityksessä oltava formaali, dokumentoitu tapa vaatimustenkäsittelyyn ja testaukseen. Tältä osin vaatimustenkäsittely nivoutuu yhdeksi osaksi yrityksen laadunvalvontaa. Mitä aikaisemmassa vaiheessa löydetään aukkokohdat ohjelmiston määrityksessä, niin sitä edullisemmaksi niiden korjaaminen muodostunee, kuten kuvassa 1 on havainnollistettu.



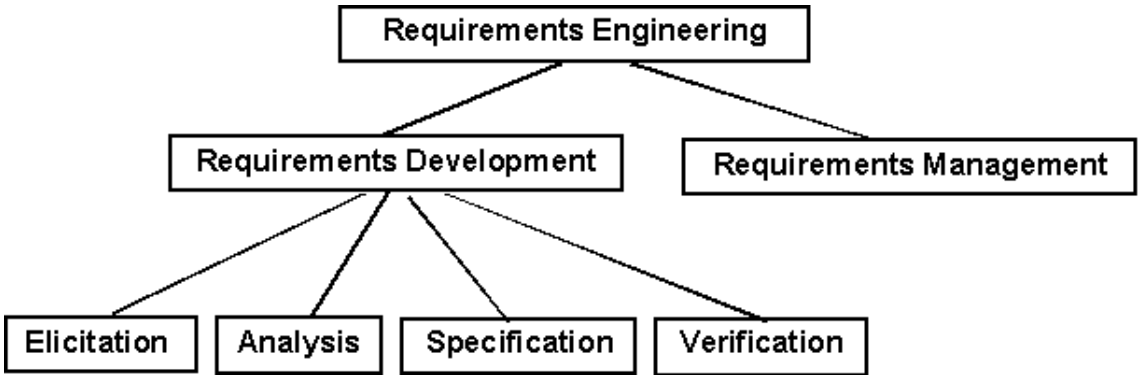
Kuva 1. Ohjelmiston kehityksen elinkaari/kustannukset per virhe. [STQE2001]

Tässä diplomityössä on tarkoituksena määrittää ohjelmistopalveluyrityksen vaatimustenkäsittelyn puitteet ja esittää ratkaisu, kuinka vaatimustenkäsittely yrityksessä hoidettaisiin. Työn alkuosassa esitellään vaatimustenkäsittelyä, prosessimalleja sekä näihin liittyviä laatujärjestelmiä teoreettisesta näkökulmasta ja tuodaan esiin vaihtoehtoisia tekniikoita. Yrityksen vaatimustenkäsittelyn lähtötilanne ja projekteissa käyttämä prosessimalli esitellään.

Työn tuloksena esitellään näkemys ja johtopäätökset siitä, kuinka yrityksen vaatimustenkäsittelyn pitäisi toimia ja millainen sen tulisi olla prosessina.

2 VAATIMUSTENKÄSITTELY

Vaatimustenkäsittely (*Requirements engineering*, RE) sisältää Wiegersin mukaan kaiken vaatimuksiin liittyvän [Wieg1999]. Terminologia kuitenkin vaihtelee eri lähteissä, joten on syytä määritellä eri osa-alueita yleispätevästi. Kuvassa 2 on esitetty yksi mahdollinen tapa jaotella vaatimustenkäsittelyä osa-alueisiin ja havainnollistettu englanninkielisen termistön suhdetta toisiinsa.



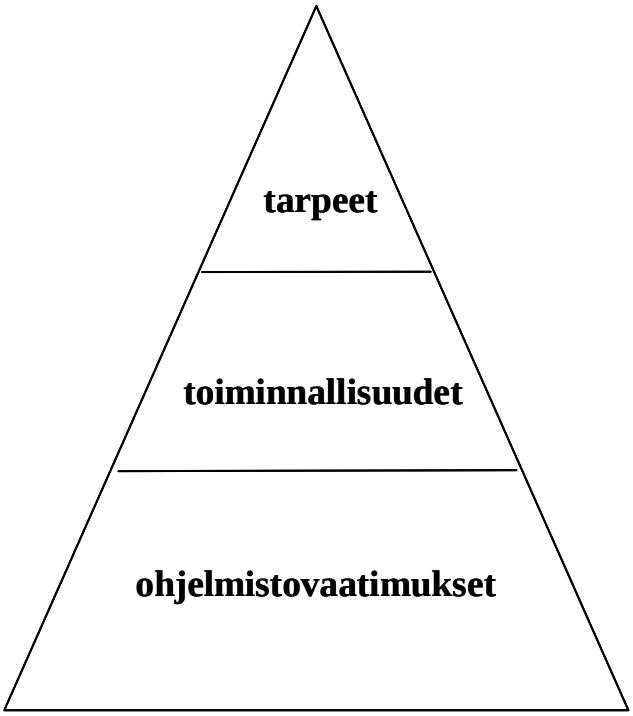
Kuva 2. Vaatimustenkäsittelyn hierarkiamalli. [Wieg1999]

2.1 Määritelmiä

Vaatimus (*Requirement*) on määritelty standardissa IEEE Std 610.12-1990 seuraavasti [IEEE1990]:

1. Edellytys tai kyvykkyys, joka tarvitaan, että käyttäjä voi ratkaista ongelman tai saavuttaa tavoitteen.
2. Edellytys tai kyvykkyys, jonka järjestelmän tai järjestelmäkomponentin pitää täyttää tai omata, jotta sopimus, standardi, määritelmä tai muu formaali dokumentti täytetään.
3. Dokumentoitu esitys edellytyksestä tai kyvykkyydestä kuten kohdissa 1 ja 2.

Kuten Wiegers osoittaa, niin pääkonsepti on, että vaatimus täytyy olla dokumentoitu. Vaatimuksia on useita eri tyyppejä, mutta tärkeää on, ettei niitä sekoiteta toiminnallisuuksiin (*feature*), jotka muodostuvat joukosta vaatimuksia [Wieg1999]. Kuvassa 3 esitellään tarpeiden, toiminnallisuuksien ja ohjelmistovaatimusten suhdetta toisiinsa pyramidimallin muodossa.



Kuva 3. Tarpeet ja toiminnallisuudet ovat läheisiä ja ovat lähde suurelle joukolle ohjelmistovaatimuksia. [Leff2000]

Vaatimustenhallinta (*Requirements management*, RM) määritellään Leffingwellin ja Widrigin mukaan systemaattiseksi tavaksi kerätä, luokitella ja dokumentoida järjestelmän vaatimuksia ja prosessiksi, joka luo ja ylläpitää yhtenevää näkemystä asiakkaan ja projektiryhmän välillä vaatimusten muuttamisesta [Leff2000]. Tämä määritelmä on itse asiassa hyvin lähellä tai lähes sama, kuin mitä vaatimustenkäsittelyllä ymmärretään [Wie1999], joten on järkevää jakaa se eri käsitteisiin.

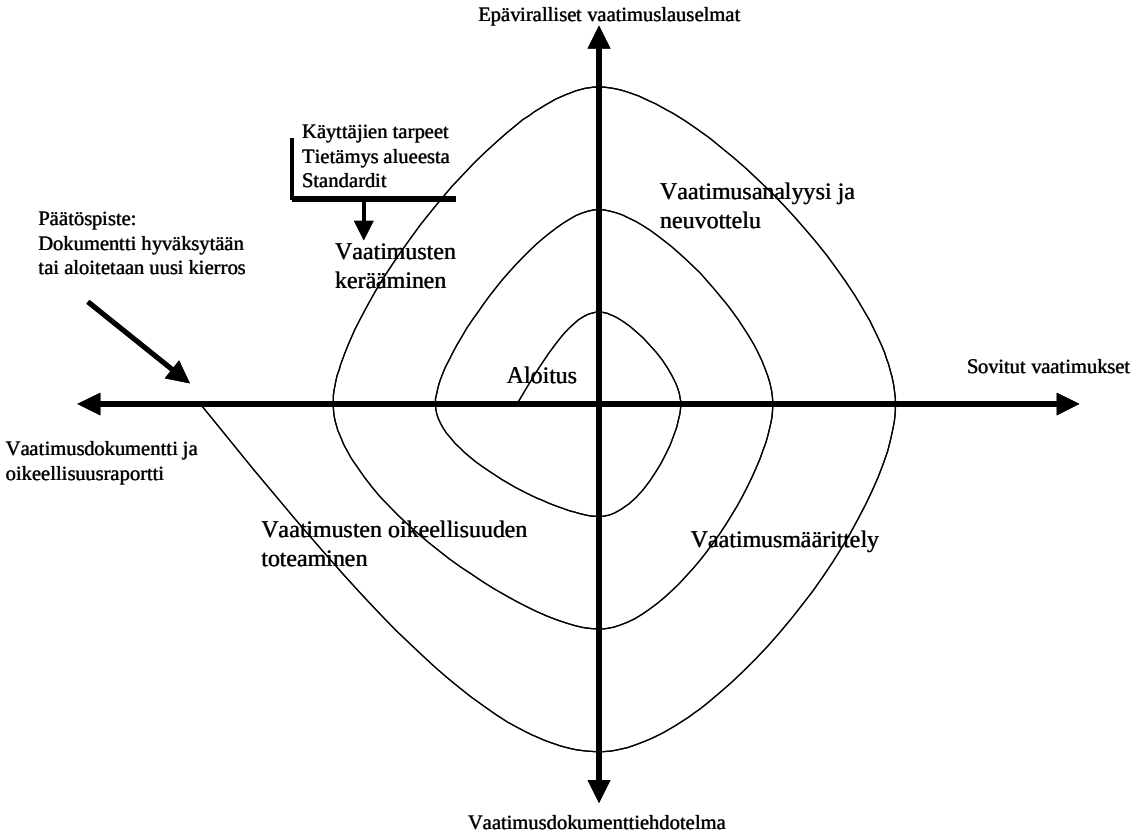
Rajoitetaan vaatimustenmäärittely (*Requirements development, Requirements definition*) sisältämään vaatimusten keräys, rajoittaminen, luokittelu sekä dokumentointi ja määritellään vaatimustenhallintaan kuuluvaksi vaatimusten [Hokk2001]:

1. muutoshallinta,
2. välisten riippuvuuksien hallinta,
3. vaatimusdokumenttien ja muiden ohjelmistoprosessissa tuotettujen dokumenttien välisten riippuvuuksien hallinta.

Lisäksi käsitellään vaatimustenkäsittelyprosessi prosessikehitykseen kuuluvana, mutta kuitenkin vaatimustenkäsittelyn kokonaisuuteen tiukasti nivoutuvana osana.

2.2 Vaatimustenkäsittelyprosessi

Vaatimustenkäsittelyprosessi sisältää esiselvitysvaiheen, vaatimusten keräämisen ja luokittelun, määrittelyn, todentamisen ja vaatimustenhallinnan. Kuva 4 esittää, kuinka eri vaatimustenmäärittelyn[1] tietämysalueet muodostavat iteratiivisen vaatimustenmäärittelyprosessin [Somm2001], [IEEE2001].



Kuva 4. Vaatimustenmäärittelyn spiraalimalli. [IEEE2001]

Vaatimustenmäärittelyn eri toimintoja toistetaan, kunnes saadaan aikaan hyväksyttävä vaatimusmäärittelydokumentti tai kunnes ulkoiset tekijät, kuten aikataulupaine tai resurssien puute, aiheuttavat prosessin terminoinnin. On tärkeää huomata, että

[1] Lähteessä [IEEE2001] vaatimustenkäsittelyllä (RE) tarkoitetaan välillä koko aluetta ja välillä pelkästään vaatimustenmäärittelyä.

vaatimustenmäärittelyprosessin ennenaikaisella terminoinnilla saattaa olla vahingollinen vaikutus järjestelmän suunnitteluun. Erityisesti jos vaatimuksia jää keräämättä tai niihin jää ristiriitaisuuksia, niin järjestelmän toimivuudesta kokonaisuutena ei ole takeita. Sen jälkeen, kun lopullinen vaatimusmäärittelydokumentti on tuotettu, jokainen muutos on osa vaatimustenhallintaprosessia, joka on toiminto, joka kattaa ohjelmiston koko elinkaaren [IEEE2001]. Taulukossa 1 on esitelty, kuinka vaatimustenkäsittelyprosessi liittyy muihin prosessien tietämysalueisiin.

Taulukko 1. Yhteydet muihin tietämysalueisiin [IEEE2001].

Laatu	Prosessilaatu ja –kehitys liittyvät laatuun. Tähän sisältyy linkkejä ohjelmistojen arviointi- ja kehitysmalleihin, kuten ISO/IEC 15504 ja ISO 9001-3 ohjeistuksiin. Vaatimustenhallinta on tason kaksi avainprosessialue CMM-mallissa ja tämä on lisännyt sen huomattavuutta laatutekijänä.
Standardit	Yllä olevien laatustandardien lisäksi ohjelmiston elinkaaristandardi ISO/IEC 12207-1995 kuvaa vaatimustenkäsittelytoimintoja ensisijaisten, tukevien ja organisaatiosidonnaisten elinkaariprosessien yhteydessä.
Mittaaminen	Prosessitasolla vaatimusten mitattavuus on yleensä suhteellisen karkeaa ja keskittyy vaatimusten ja muutosten määrän ja vaikutusten laskemiseen ja arviointiin.
Työkalut	Yleiset prosessinhallinnan työkalut.

2.2.1 Prosessimallit

Vaatimustenhallintaprosessi ei ole pelkkä diskreetti ohjelmiston elinkaaren etuosan toiminto. Se on prosessi, joka käynnistetään projektin alussa ja jatkuu säädettävänä läpi koko ohjelmistoprosessin elinkaaren. Prosessin pitää käsittää vaatimukset kokoonpanon osina ja hallita niitä osana samaa kokoonpanojärjestelmää kuin muitakin kehitysprosessin tuotteita. Prosessimalli on räätälöitävä organisaation ja projektin tarpeisiin [IEEE2001].

2.2.2 Prosessin toimijat

Vaatimustenkäsittely on periaatteeltaan kurinpitoa eri osapuolten välillä ja siinä on toimittava välittäjänä käyttäjä- ja ohjelmistosuunnittelupiirien välillä, sillä prosessissa on osallisena useita asianosaisia, jotka eivät yleensä ole homogeeninen ryhmä. Osallisena saattaa olla monia käyttäjiä ja monia asiakkaita, joilla kaikilla on omat käsityksensä, ulkoisia asianosaisia ja myös ohjelmiston tekijöitä, joiden tarpeet järjestelmän teossa poikkeavat toisistaan [IEEE2001].

Ei ole mahdollista täydellisesti toteuttaa jokaisen asianosaisen vaatimuksia ja siksi on neuvoteltava kompromissi, joka on pääasiallisten asianomaisten hyväksyttävissä sekä sopii budjetillisten, teknisten, säädöksellisten ja muiden rajoitteiden sisään. Edellytys tähän on, että kaikki asianosaiset ja heidän asiansa luonne on tunnistettu ja heidän vaatimuksensa on kerätty [IEEE2001].

2.2.3 Prosessin tuki ja hallinta

Vaatimustenkäsittelyprosessin tueksi tarvitaan ja se kuluttaa projektinhallinnan resursseja. Tämä on ymmärrettävä linkkinä prosessimallien aktiviteeteista kustannuksiin, henkilöstövoimavaroihin, koulutukseen ja työkaluihin [IEEE2001].

2.2.4 Prosessin laatu ja sen parantaminen

Vaatimustenkäsittely on avainasemassa, kun puhutaan ohjelmistotuotteiden kustannuksista, aikataulussa pysymisestä ja asiakastyytyväisyydestä. Prosessin laadun arviointi auttaa asettamaan vaatimustenkäsittelyprosessin suhteeseen ohjelmistojen ja järjestelmien laatustandardien ja prosessikehitysmallien kanssa. Prosessin laatu ja sen parantaminen on läheinen ohjelmistolaadun ja ohjelmistoprosessien tietämysalueita [Somm1997b], [IEEE2001].

2.3 Vaatimustenmäärittely

Vaatimustenmäärittelyn osa-alueita katsotaan olevan vaatimusten [Thay1997]:

1. kerääminen (*elicitation*),
2. luokittelu (*analysis*),
3. määrittely (*specification*),
4. todentaminen (*verification*),

5. hallinta (*management*).

Kuitenkin esimerkiksi Wiegers eriyttää hallinnan omaksi korkeamman tason alueeksi [Wieg1999] ja Leffingwell ja Widrig taasen merkitsemään koko aihetta [Leff2000].

2.3.1 Kerääminen

Vaatimukset kerätään eri sidosryhmiltä käyttäen erilaisia tekniikoita, jolloin saadaan kolmentasoisia vaatimuksia: liiketaloudellisia, käyttäjän ja toiminnallisia. On todettu, että määrittelemällä prosessi voidaan vaatimusten keräämistä nopeuttaa ja tehostaa. Myös vaatimusten uudelleenkäyttö on niiden keräämistä [Hokk2001], [Rein2000], [Wieg1999].

Keräämisen alkuvaiheessa on tärkeää löytää oikeat asianosaiset (*stakeholders*), mutta samalla on muistettava tarve löytää se toinen asianomaisryhmä, joka vaikuttaa vaatimusprosessiin ja ohjelmiston elinkaareen. Tämä jälkimmäinen ryhmä on usein ratkaiseva ohjelmiston tai palvelun julkaisun ja tuotannon kannalta [Mark2001].

2.3.2 Luokittelu

Etsitään järjestelmän rajoitteet, tutkitaan vaatimusten toteuttamiskelpoisuus sekä niihin liittyvät riskitekijät ja asetetaan prioriteetti vaatimuksille [Hokk2001], [Wieg1999]. Tilaajan ja toteuttajan välillä pitää olla selkeä käsitys siitä, mitä ollaan toteuttamassa [Mayr1990].

Jos ei tiedetä, millaisia tuotteita ollaan tekemässä, niin ei myöskään voida arvioida, kuinka paljon niiden kehittämisen kustannukset ovat. Tämä takia on tärkeää, että ohjelmistomääritykset ovat niin yksiselitteisiä kuin mahdollista [Boeh1981].

Luokittelu voidaan jakaa viiteen osa-alueeseen [Pres1997]:

1. Ongelman tunnistaminen.
2. Evaluointi ja synteesi.
3. Mallinnus.
4. Määritys.
5. Katselmointi.

Ohjelmistoprosessin luokitteluvaiheessa tuotetaan toteutettavuustutkimus, vaatimusmäärittelydokumentti, projektisuunnitelma ja hyväksymistestitapaukset [Mayr1990]. Suunnittelijan pitää arvioida ohjelmistovaatimuksia seuraavien kriteerien mukaisesti ja näistä syntyvät tulokset pitää dokumentoida [ISO1995]:

a) Jäljitettävyys järjestelmävaatimuksiin ja järjestelmäkuvaukseen.
b) Ulkoinen yhtenevyys järjestelmävaatimusten kanssa.
c) Sisäinen yhteneväisyys.
d) Testattavuus.
e) Ohjelmistokuvauksen toteutettavuus.
f) Toiminnan ja ylläpidon toimivuus.

2.3.3 Määrittely

Vaatimusten dokumentointia varten on hyvä omaksua asiakirjapohja määritykselle, jota käyttäen kirjataan kaikki vaatimukset. Vaatimuksille annetaan yksiselitteiset tunnisteet ja niiden alkuperä kirjataan jäljitettävyyttä varten. Myös liiketaloudelliset seikat kirjataan sekä luodaan jäljitettävyysmatriisi [Wieg1999], [IEEE1998].

Järjestelmän suunnittelijoiden tarvitsee tietää kaksi oleellista asiaa, jotka on hyvä erotella määrittelydokumentissa [Bray2002]:

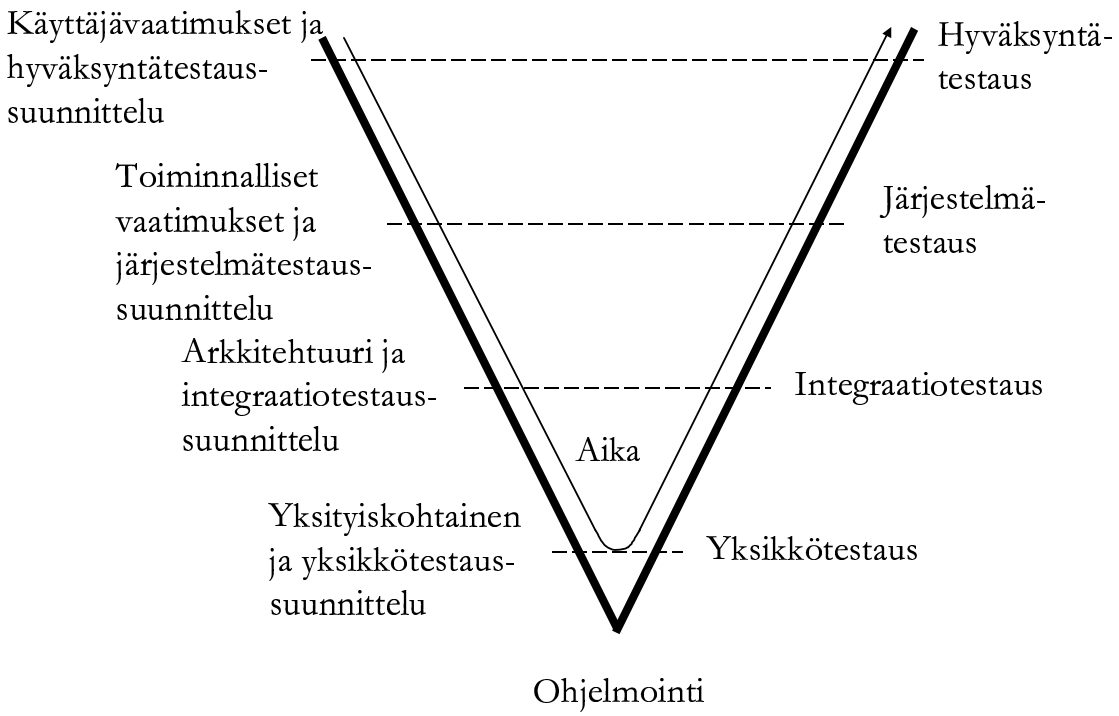
- Millainen ongelma-alue on – sen eri elementtien luonne ja keskinäiset vaikutukset.
- Mitkä ovat vaatimukset – mitä vaikutuksia asiakas haluaa uuden järjestelmän tuottavan tässä ongelma-alueessa.

2.3.4 Todentaminen

Vaatimukset on todennettava, jotta tiedetään niiden oikeellisuus. Todentaminen pitäisi rakentaa osaksi vaatimustenkäsittelyprosessia ja pitäisi suorittaa monella eri tasolla. Tämä suoritetaan katselmoimalla vaatimusdokumentit, tekemällä jäljitettävyysanalyysi sekä luomalla testitapauksia ohjelmistolle ja hyväksymiskriteerejä vaatimuksille [Bray2002], [Thay1997], [Wieg1999].

Yksi todentamismalli on esitelty kuvassa 5, jossa kuvataan kuinka suunnitteluvaiheet ja testausvaiheet ovat sidoksissa toisiinsa. Testauksen suunnittelu ja alustavien

testitapausten kehittäminen pitäisi aloittaa vastaavan suunnitteluvaiheen aikana. Suunnitteluvaiheessa voidaan luoda käsitteellisiä testitapauksia ja etsiä niiden avulla virheitä, ristiriitaisuuksia ja puutteita vaatimusmäärittelyssä ja analyysimalleissa paljon ennen itse ohjelmoinnin aloittamista [Wieg1999].



Kuva 5. Ohjelmistokehityksen V-malli. [Wieg1999]

2.4 Vaatimustenhallinta

Ongelmana vaatimustenkäsittelyssä on, että hyvin harvoin vaatimukset ovat täydellisiä ohjelmistoprojektin alussa tai että ne pysyvät muuttumattomina läpi projektin. Vaatimustenmäärittely vaatii taitoa sekä kokemusta ja tällaisia henkilöitä on harvassa. Ohjelmistoja suunniteltaessa toiminnallisuuksia mietitään perinpohjin, joten on selvää, että uusia ideoita ja parempia ratkaisutapoja syntyy [Hump1995].

Vaatimustenhallinnan osa-alueita ovat [Wieg1999]:

1. muutoshallinta (*change control*),
2. versionhallinta (*version control*),
3. jäljitys (*tracing*),
4. tilaselvitys (*status tracking*).

Kaksi viimeistä eli jäljitys ja tilaselvitys on syytä käsitellä yhdessä vaatimuksen jäljitettävyytenä (*traceability*).

2.4.1 Muutoshallinta

Muutoshallinta on keskeinen kohta vaatimustenkäsittelyä ja toimiva prosessi on tässä tärkeä työkalu. Organisaatioissa on usein toteutettu virheilmoitusten käsittelyjärjestelmä, mutta sillä voidaan hallita myös parannusehdotuksia, toiminnallisuusmuutoksia ja uusien järjestelmien vaatimusmuutoksia [Wieg2001].

Vaatimusten muutoshallinta on myös erittäin monimutkaista, vaikka kirjallisuudessa se esitetään yksikertaisena eteenpäin menevänä prosessina, joka koostuu useista eri vaiheista [Kauk2000]. Ohjelmistokehityksessä vaatimuksia hallittaessa kohdataan seuraavia ongelmia, joista muutokset aiheutuvat [Crnk1999]:

- Nopeammin muuttuvat teknologiat, työmenetelmät ja -välineet suunnittelussa, toteutuksessa ja muutosprosessissa, kuten myös teknologiamuutokset kesken toteutuksen (laitteisto ja ohjelmisto).
- Tuote on saatava nopeammin markkinoille.
- Vaatimusten sisältöä ei ymmärretä puuttuvan teknologisen kompetenssin takia tai vaatimusten taustat jäävät epäselviksi.

Muutokset ovat prosessissa väistämättömiä ja niihin on varauduttava, jotta olisi järkevä todennäköisyys onnistumiselle. Tämän vuoksi vaatimusten jatkuva muuttuminen pitää pysäyttää tai vähintään vähentää hallittavalle tasolle [Leff2000] ja seuraavat ongelmat nousevat enenevässä määrin esiin [Crnk1999]:

- Riittämätön vaatimusten käyttö ja puutteellinen suunnittelun ja toteutuksen ohjaus vaatimusten suhteen.
- Riittämätön valveutuneisuus vaatimusten muuttumisesta kesken järjestelmän kehityksen.
- Riittämätön hallinta siitä, mitkä vaatimukset ovat toteutuksessa ja mitkä vaatimukset on jo kokonaan toteutettu.
- Riittämätön toteutuksen verifiointi ja todentaminen.

Jotta prosessissa pystyttäisiin hallitsemaan muutoksia tehokkaammin, täytyy siihen sisällyttää seuraavat vaiheet [Leff2000]:

1. Tunnista, että muutos on väistämätön ja valmistaudu sen varalta.
2. Kiinnitä vaatimukset perustasoon.
3. Luo yksi kanava muutoshallintaan.
4. Käytä muutoshallintajärjestelmää muutosten taltiointiin.
5. Hallinnoi muutosta hierarkkisesti.

Muutoshallintaprosessi alkaa muutospyynnöllä, joka johtaa päätökseen joko toteuttaa muutos tai hylätä se ja kulminoituu ohjelmistokokoonpanoyksikön hallittuun päivitykseen [Pres1997].

2.4.2 Versionhallinta

Vaatimusdokumenttien oikeellisuutta varten pitää niillä olla toimiva versionhallinta. Yksinkertaisimmillaan tämä hoidetaan dokumenttien revisioilla ja muutoshistorialla, mutta tehokkainta on käyttää kaupallisia vaatimustenhallintatyökaluja, joissa vaatimukset säilytetään tietokannassa ja vaatimusten tiedot ovat aina ajan tasalla [Wieg1999].

Versionhallintaa saatetaan kutsua myös nimellä kokoonpanonhallinta (*configuration management*, CM), jossa versionhallinta yhdistetään kiinteästi muutoshallintaan. CM-pohjaisessa vaatimustenhallintaprosessissa on seuraavia etuja [Leff2000], [Crnk1999]:

- Estää luvattomat ja potentiaalisesti tuhoisat muutokset vaatimuksiin.
- Säilyttää revisiot vaatimusdokumenteista.
- Mahdollistaa aikaisempien vaatimusdokumenttien palauttamisen ja/tai uudelleenluomisen.
- Tukee hallittua, organisoitua perustasoitettua ”julkaisustrategiaa” järjestelmän portaittaisiin parannuksiin ja päivityksiin.
- Estää dokumenttien samanaikaisen päivittämisen tai ristiriitaisen ja epäkoordinoidun päivittämisen eri dokumentteihin samanaikaisesti.

- Takaa suunnittelijoille ja ohjelmoijille suoran ja helpon pääsyn tarvittavaan vaatimustietoon.
- Yksinkertaistaa aikaisemmin käytettyjen ja toteutettujen vaatimusten uudelleenkäyttöä.

2.4.3 Jäljitettävyys

Jäljitettävyydelle on määritetty ehdot standardissa [IEEE1998], jossa perusajatuksena on, että jokaiselle vaatimukselle löytyy lähde ja jokainen vaatimus täsmää ohjelmistokomponenttiin ja jokaisen ohjelmistokomponentin olemassaolon oikeutus löytyy vaatimuksista. Jäljitettävyyden tarkoituksena on tukea todentamista, jolloin jäljitettävyystyökaluilla voidaan varmistaa, ettei todentamisketjuja jää käsittelemättä ja ettei kuitenkaan liiallista todentamista suoriteta [Leff2000].

On huomattavaa, että parantamalla jäljitettävyyttä parannetaan koko ohjelmistoprosessia. Seuraavat kohdat ovat keskeisiä [Kuja2001]:

1. Vaatimusprosessi ja sen ympäristö.
2. Vaatimusluettelon rakenne.
3. Jäljitettävyysohjekirja.

Vaatimustenhallintatyökalun käyttöönotto jäljitettävyyden hallinnointiin on suositeltavaa, sillä se helpottaa vaatimustenhallintaa ja jäljitettävyystiedon päivittämistä. Muistettava kuitenkin on, ettei pelkkä työkaluohjelmisto itsessään ratkaise ongelmia [Hokk2001].

3 LAADUNVALVONTA

Ohjelmiston laadunvalvonta on monimutkaista ohjelmistojen kompleksisen luonteen takia. Tällöin, jos halutaan suorittaa kunnollista ohjelmistolaadunvalvontaa, on ohjelmistokehitysprosessista kerättävä tietoa, joka arvioidaan ja käsitellään. Tilastollinen ohjelmistolaadunvalvonta auttaa parantamaan tuotteen laatua ja itse ohjelmistoprosessia [Pres1997].

3.1 Laatutavoitteet

Laatua on vaikeaa mitata, mutta kuitenkin on välttämätöntä asettaa keinoja mitata ja parantaa määritelmien laatua. Ei ole mahdollista yrittää päätellä laadun tasoa vain siitä, millainen tuote saadaan projektin päätteeksi, sillä monet muutkin seikat vaikuttavat lopputulokseen [Leff2000].

Mittaustietoa saadaan ja tieto siitä, onko mittaustieto kyseiseen tapaukseen sopivaa, yleensä aikaisemmista tuloksista ja kokemuksista. Mittaustieto ja sen myöhempi evaluointi muodostavat pohjan seuraaville toiminnoille ja päätöksille. Tietoa saadaan moniin eri tarkoituksiin: helpottamaan tilanteen ymmärtämistä, analysoivaksi, prosessinhallintaan, säätelyyn sekä hyväksymistä tai hylkäämistä varten [Ishi1986].

3.1.1 Vaatimuksen laatutavoite

Jotta laatu saataisiin ohjelmistoon alusta lähtien, täytyy vaatimusten olla dokumentoidut ja ymmärretyt [Horc1996]. Se, että on olemassa vaatimusmääritelmädokumentti, on ensisijainen laatutavoite, mutta lisäksi itse vaatimusten on täytettävä tietyt laadulliset tavoitteet [Leff2000]. Näitä ovat Horchin mukaan [Horc1996]:

- Tarpeellisuus.
- Toteutettavuus.
- Oikeellisuus.
- Täydellisyys.
- Selkeys.
- Mitattavuus.
- Testattavuus.

Suosituksia siitä, miten näihin tavoitteisiin päästään, antavat Leffingwell ja Widrig [Leff2000]:

- Käyttämällä luonnollista kieltä aina kuin se on mahdollista.
- Käyttämällä kuvia ja kaavioita selkeyttämään tarkoitusta.

Joskus vaatimus on otettu mukaan yksinkertaisesti siksi, että se tuntuu hyvältä idealta, mutta kokonaisuuden kannalta se ei tuo mitään lisäarvoa järjestelmään. Vaatimusten katselmoinnissa arvioidaan jokaisen vaatimuksen tarpeellisuus. Tarpeellisuus on yhteydessä vaatimuksen toteutettavuuteen, sillä jos vaatimus ajatellaan tarpeelliseksi, mutta se ei ole saavutettavissa, niin täytyy miettiä uusi lähestymistapa tai metodi vaatimuksen tuomiseksi julki [Horc1996].

Täydellisyys, oikeellisuus ja selkeys ovat kaikki kriteerejä, jotka kohdistuvat siihen, miten vaatimus on ilmaistu. Vaatimuksesta ei saa puuttua osia, sen pitää olla oikea ja siinä ei saa olla tulkinnanvaraisuuksia. Jos suunnittelija ei ymmärrä vaatimuksen sisältöä, niin vaatimus on käyttökelvoton. Vaatimusten kielen pitäisi olla mahdollisimman yksinkertaista, eikä erikoiskieltä tulisi käyttää. Tämä myös tarkoittaa, että vaatimusmäärittelydokumentissa on selkeästi määriteltävä käytetyt termit ja lyhenteet [Horc1996].

Mitattavuus ja testattavuus ovat yhteydessä toisiinsa. Jokainen vaatimus täytyy lopulta pystyä näyttämään, toteen ennen kuin ohjelmistoa voidaan pitää valmiina. Vaatimusta, jolla ei ole mitattavaa arvoa tai määrettä, jonka voidaan osoittaa joko olevan tai puuttuvan, ei voida testata. Testattavuus on paras tapa arvioida missä laajuudessa järjestelmän toteutuksen kustannukset ovat ennustettavissa [Horc1996], [Boeh1981].

3.1.2 Vaatimusmäärittelyn laatutavoite

Standardissa IEEE Std 830-1998 määritellään kahdeksan laatutavoitetta vaatimusmäärittelydokumentille ja Leffingwell ja Widrig lisäävät niitä vielä yhden. Moderni ohjelmistovaatimusmäärittely on korkeaa laatua, jos se on [IEEE1998]:

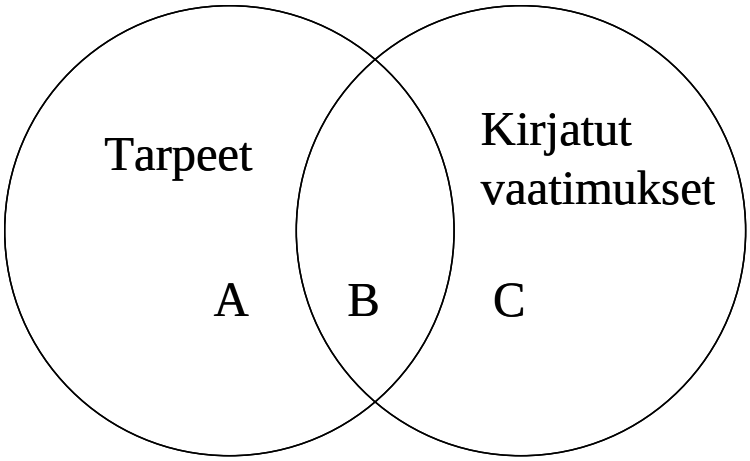
- Oikea.
- Yksiselitteinen.

- Täydellinen.
- Yhtenevä.
- Luokitettu suhteessa tärkeyteen ja pysyvyyteen.
- Todennettava.
- Muokattava.
- Jäljitettävä.
- Ymmärrettävä [Leff2000].

Näiden tavoitteiden lisäksi jo Davis [Davi1993] määritteli useampia lisätavoitteita vaatimusmäärittelylle, mutta toteaa, että kaikkien tavoitteiden saavuttaminen on mahdotonta ja yritys saavuttaa tiettyjä tavoitteita yhtä aikaa (luettava, täydellinen ja yksiselitteinen) aiheuttaa vaatimusmäärittelyn kustannusten karkaamisen käsistä. Yhteenvetona Davis [Davi1993] toteaa, että ei ole olemassa sellaista asiaa kuin täydellinen ohjelmistovaatimusmäärittely.

3.1.2.1 Oikea

Vaatimusmäärittely on oikea, jos ja vain jos jokainen siihen sisältyvä vaatimus edustaa jotain, mitä rakennettavassa järjestelmässä vaaditaan olevan. Ei ole olemassa työkalua tai toimintatapaa, joka takaisi määrittelyn oikeellisuuden, mutta jäljitettävyydellä voidaan helpottaa oikeellisuuteen pääsyä ja ehkäistä virheitä. Kuvassa 6 on havainnollistettu, kuinka oikeat vaatimukset löytyvät leikkaavasta osiosta B. Ohjelmistoprojekteissa usein käy, että osiosta A jätetään toteuttamatta tilaajan tarpeita ja osioon C taasen kirjataan ominaisuuksia, joita tilaaja ei ole ilmaissut haluavansa [Davi1993], [IEEE1998], [Leff2000].



Kuva 6. Käyttäjän tarpeet / ohjelmistovaatimukset. [Davi1993]

3.1.2.2 Yksiselitteinen

Vaatimusmäärittely on yksiselitteinen, jos ja vain jos jokaisella kirjatulla vaatimuksella on vain yksi tulkinta. Jos kirjattu vaatimus voidaan ymmärtää eri tavalla kehittäjän, käyttäjän tai muun asianomaisen toimesta, niin on aivan mahdollista, että rakennetaan järjestelmä, joka on jotakin aivan muuta, kuin mitä käyttäjällä oli mielessä. Koska vaatimukset on usein kirjoitettu luonnollisella kielellä, niin on varottava tilanteita, jossa sama sana tarkoittaa eri tilanteissa tai eri kulttuureissa eri asioita. Tätä voidaan välttää käyttämällä sellaista kieltä, joka on nimenomaan tarkoitettu vaatimusmäärittelyjen kirjoittamiseen [IEEE1998], [Leff2000].

3.1.2.3 Täydellinen

Vaatimusmäärittely on täydellinen, jos ja vain jos se sisältää seuraavat elementit [IEEE1998]:

a) Kaikki merkitsevät vaatimukset, liittyvät ne sitten toiminnallisuuteen, suorituskykyyn, suunnittelurajoitteisiin, ominaisuuksiin tai ulkoisiin rajapintoihin.
b) Määritelmät ohjelmiston vasteille kaikkiin realistisiin syöteluokkiin kaikissa realistisissa tapahtumaluokissa. Huomattavaa on, että on tärkeää määrittää sekä kelvot että epäkelvot syötearvot.
c) Täydelliset tunnisteet ja viitteet kaikkiin kuviin, taulukoihin ja diagrammeihin vaatimusmäärittelyssä ja kaikkien termien ja mittayksiköiden määritelmät.

3.1.2.4 Yhtenevä

Yhteneväisyydellä viitataan sisäiseen yhteneväisyyteen, sillä jos vaatimusmäärittely ei ole yhtenevä jonkin korkeamman tason dokumentin kanssa, niin silloin se ei ole oikea. Vaatimusmäärittely on sisäisesti yhtenevä, jos ja vain jos mikään yksittäisten vaatimusten osajoukko ei ole ristiriidassa keskenään [IEEE1998].

Kolme tyypillisintä ristiriitaisuustyyppiä ovat IEEE Std 830-1998 mukaan [IEEE1998]:

a) Kohteena olevien reaalimaailman asioiden ominaisuudet ovat ristiriidassa.
b) Toiminnallisuuksien välillä on looginen tai ajallinen ristiriitaisuus.
c) Sama reaalimaailman kohde kuvataan eri vaatimuksissa eri termein.

3.1.2.5 Luokiteltu

Vaatimusmäärittely on luokiteltu suhteessa tärkeyteen ja pysyvyyteen, jos jokaisessa vaatimuksessa on indikaattori, joka ilmoittaa joko sen tärkeyden tai pysyvyyden. Tyypillisesti kaikki vaatimukset, jotka liittyvät ohjelmistotuotteeseen, eivät ole yhtä tärkeitä. Jotkut vaatimukset voivat olla välttämättömiä (*mandatory*, *required*), kun taas toiset voivat olla toivottavia (*optional*). Luokittelemalla voidaan näin kohdentaa rajallisia resursseja oikein [IEEE1998], [Leff2000].

Toinen tapa luokitella vaatimuksia on jakaa ne luokkiin tarpeellisuuden mukaan [IEEE1998]:

a) Välttämätön: Osoittaa, että ohjelmisto ei ole hyväksyttävissä ellei näitä vaatimuksia ole toteutettu sovitulla tavalla.
b) Ehdollinen: Osoittaa, että nämä ovat vaatimuksia, jotka ovat parannuksia ohjelmistotuotteeseen, mutta niiden puute ei aiheuta ohjelmiston hylkäämistä.
c) Valinnainen: Osoittaa, että vaatimusten kohteena olevat toiminnallisuudet saattavat olla hyödyllisiä. Antaa toimittajalle mahdollisuuden harkintaan ja ehdottamiseen.

3.1.2.6 Todennettava

Vaatimusmäärittely on todennettava, jos ja vain jos jokainen kirjattu vaatimus voidaan todentaa. Vaatimus on todennettava, jos ja vain jos on olemassa äärellinen kustannustehokas prosessi, jolla henkilö tai laitteisto voi tarkastaa, että ohjelmistotuote vastaa vaatimusta. Yleisesti ottaen yksikään moniselitteinen vaatimus ei ole

todennettavissa. Jos ei voida löytää keinoa, miten ohjelmisto täyttää jonkin vaatimuksen, on vaatimus silloin poistettava tai sitä on korjattava [IEEE1998].

3.1.2.7 Muokattava

Vaatimusmäärittely on muokattava, jos ja vain jos sen rakenne ja tyyli on sellaista, että mitkä tahansa muutokset vaatimuksiin voidaan tehdä helposti, täydellisesti ja ristiriidattomasti säilyttäen edelleen rakenteen ja tyylin. Muokattavuuden ehtoina on yleensä [IEEE1998]:

a) Johdonmukainen ja helppokäyttöinen järjestys, joka sisältää sisällysluettelon, indeksin ja ristiviittaukset.
b) Ei ole päällekkäisyyksiä vaatimuksissa.
c) Jokainen vaatimus on kirjattu erikseen.

Päällekkäisyys ei sinänsä ole virhe ja saattaa tietyissä tapauksissa helpottaa luettavuutta, mutta se johtaa helposti virheisiin, kun vaatimusmäärittelyä päivitetään. Vaatimuksia päivitetään aina vaikka sitä ei haluttaisikaan tehdä, sillä sellainen vaatimusmäärittely, jota ei ole mahdollista muokata, muuttuu hyvin nopeasti olemattomaksi vaatimusmäärittelyksi [IEEE1998], [Leff2000].

3.1.2.8 Jäljitettävä

Vaatimusmäärittely on jäljitettävä, jos sen jokaisen vaatimuksen alkuperä on selvillä ja jos siinä on mahdollista jokaisen vaatimuksen kohdalla tehdä viittaus tulevaan kehitykseen tai muutosdokumenttiin. Seuraavia kahta jäljitettävyystyyppiä suositellaan [IEEE1998]:

a) Taaksepäin jäljitettävyyttä eli aikaisempaan kehitysvaiheeseen. Jokaisella vaatimuksella on viittaus sen lähteeseen aikaisemmissa dokumenteissa.
b) Eteenpäin jäljitettävyys eli kaikkiin vaatimusmäärittelystä kehitettyihin dokumentteihin. Jokaisella vaatimuksella pitää olla uniikki nimi tai viitenumero.

Eteenpäin jäljitettävyys on erityisen tärkeää, sillä kun ohjelmistotuote on tuotannossa ja ylläpidettävänä ja kun koodi- ja suunnitteludokumentteja muutetaan, niin on välttämätöntä pystyä varmistamaan se vaatimusjoukko, joihin nämä muutokset saattavat

vaikuttaa. Automaattisista työkaluista ja jäljitettävyysmatriiseista on tässä muutosseurannassa erittäin suurta apua [IEEE1998], [Leff2000].

3.1.2.9 Ymmärrettävä

Vaatimusmäärittely on ymmärrettävä, jos sekä käyttäjä- että kehittäjäryhmät pystyvät täysin ymmärtämään yksittäiset vaatimukset ja kokoamaan määrittelyn sisältämän toiminnallisuuden. Täten niiden, jotka kirjoittavat vaatimuksia, pitää ymmärtää molempien ryhmien sanastot, erityistermit ja kulttuurit [Leff2000].

3.2 Laatujärjestelmä

Ohjelmistokehityksen ja laatujärjestelmän suhde johtaa ohjelmistoprojektille tärkeän komponentin määrittelemiseen, joka tunnetaan laatusuunnitelman nimellä. Laadunvalvonnan ytimenä on laatujärjestelmä, jota vasten ohjelmistotuotteiden laatua pitäisi arvioida, ja johon ohjelmistotuotteille asetetut laatutavoitteet perustuvat. Laatujärjestelmä sisältää hallinnointirakenteen, vastuut, toiminnat, kyvykkyydet ja resurssit. Nämä varmistavat, että projekteissa tuotetut ohjelmistotuotteet sisältävät ne halutut laatuseikat, jotka sekä asiakas että kehittäjä ovat niihin halunneet sisällyttää [Pres1997], [ISO2001].

Täten laatujärjestelmä sisältää seuraavia toimintoja [Pres1997]:

- Projektien auditointia, jotta varmistetaan, että laadunvalvontaa on käytetty. Laatujärjestelmän katselmointia sen kehittämiseksi.
- Laadunvalvontahenkilöstön kehittymistä.
- Resurssineuvotteluja, jotta laadunvalvontahenkilöstöllä on mahdollisuus toimia kunnolla.
- Syötteen antaminen kehityspuolen parannuksiin.
- Standardien, toimintamallien ja ohjeistuksen kehittäminen.
- Raportointi ylimmälle johdolle laatujärjestelmän toimivuudesta.
- Raportointi laatujohdolle, jotta se voi käynnistää aktiviteetteja, jotka avustavat laadun parantamista.

Nämä ovat aktiviteetteja, jotka yleensä yhdistetään laadunhallintajärjestelmään. Laatujärjestelmän konkreettisia yksityiskohtia ylläpidetään laatuohjekirjassa, joka normaalisti sisältää laatustandardit ja kehitystoiminnot, joita voidaan käyttää projekteissa [Pres1997].

3.2.1 ISO 9000 -laatujärjestelmä

ISO 9000 on monissa maissa käyttöönotettu laadunhallintastandardisarja, jonka variantteja on kehitetty myös ohjelmistokehitykseen. Standardin ongelmana on kuitenkin sen laajuus sekä se, ettei sitä ole sidottu mihinkään teollisuudenalaan [Somm1997a], [Pres1997].

ISO 9000 koostuu viidestä laatustandardista [Leff2000]:

1. ISO 9000: Ohjeistus valinnalle ja käytölle.
2. ISO 9001: Laadunvalvonnan ohjeistus suunnittelulle, kehitykselle, tuotannolle, asennukselle ja huollolle.
3. ISO 9002: Laadunvalvonnan ohjeistus pääasiasiassa valmistuksessa toimiville yrityksille.
4. ISO 9003: Laadunvalvonnan ohjeistus pääasiassa jakelussa toimiville yrityksille.
5. ISO 9004: Ohje siitä, miten laadunhallintajärjestelmän eri osia sovelletaan.

Standardissa painotetaan tarvetta molemminpuoliselle yhteistyölle asiakkaan ja ohjelmistotoimittajan välille, erityisesti [Leff2000]:

- Asetettava molemmista ryhmistä vastuulliset henkilöt vaatimusten löytämiseen.
- Luotava metodit ja toimintatavat vaatimuksista sopimiseen ja muutoshallintaan.
- Pyrittävä estämään väärinkäsitykset vaatimuksissa.
- Luotava toimintatavat vaatimuskeskustelujen kirjaamiseen ja tulosten katselmointiin.

Kuitenkaan ISO 9000 ei kerro, kuinka itse vaatimustenkäsittelyä tulisi toteuttaa käytännössä. Tämän takia on ollut taipumusta, että prosessikehityksessä on keskitytty sellaisiin prosesseihin, jotka on standardissa, mainittu vaikka kehittämisvaraa olisi nimenomaan standardissa mainitsemattomissa prosesseissa [Somm1997a], [Leff2000].

4 PROSESSIMALLI

Koska yksi tärkeimpiä osa-alueita vaatimustenkäsittelyssä on vaatimusten dokumentointi, niin on myös oltava dokumentoitu tapa hoitaa vaatimustenkäsittelyä eli prosessikuvaus vaatimustenkäsittelystä. Jos asiaa ei ole yksiselitteisesti kuvattu ja eikä sitä ole ymmärretty, niin sen kehittäminen tai toteuttaminen on lähes mahdotonta. Prosessien kuvaamista ja kehittämistä varten on erilaisia ohjelmistoprosessin kehittämismalleja, joiden avulla pyritään kohti parempilaatuisia ohjelmistotuotteita sekä parantamalla prosessien laatua että vähentämään kustannuksia ja kehitystyöhön käytettyä työmäärää [Mann2001], [Rein2000].

Prosessikuvaus on kuitenkin enemmän kuin pelkkä dokumentti. Johdon sitoutuminen prosesseihin on oltava näkyvää ja pakollista, jotta saavutetaan prosessikulttuuri, joka on kurinalainen ja jonka on mahdollista saavuttaa kehitystasoja [Zahr1998]. Ei voida myöskään käsitellä yksistään prosessimallia vaatimustenkäsittelyn näkökulmasta, sillä kuten Smolander [Smol2002] toteaa, niin vaatimusanalyysimalli on abstraktio konkreettisista liiketoimintatarpeista ja ohjelmistoarkkitehtuurisuunnitelma on abstraktio konkretisoivasta ohjelmistoarkkitehtuuritoteutuksesta. Kaikkia näitä on arvioitava yhtenä kokonaisuutena, jotta saadaan kattava ratkaisu ja alustava arkkitehtuurimalli tukee myös vaatimusmäärittelyä [Smol2003].

4.1 CMM-malli

Yksi yleisimmin käytetty ja ohjelmistoalan vakavasti otettavan prosessikehityksen käynnistäjänä toiminut prosessimalli on SEI:n (*Software Engineering Institute*) kehittämä CMM (*Capability Maturity Model*). Tämä prosessimalli sisältää viitemallin prosessikehitykselle, jossa etsitään ja tunnistetaan organisaation avainprosessialueet ja niiden perusteella luokitellaan organisaatio asteikolla 1-5. Tasot toimivat eräänlaisina portaina, joissa organisaatio pyrkii nousemaan ylöspäin prosessien kehittyessä [Leff2000], [Rein2000], [Somm2001]. Tasot ja niihin liittyvät avainprosessialueet on esitelty taulukossa 2.

Taulukko 2. CMM-mallin tasot ja avainprosessialueet [Leff2000].

Taso	**Avainprosessialueet**
1. Lähtötaso Ad hoc, usein kaoottinen, menestys riippuu yksilösuorituksista.	--
2. Toistettava Projektihallinnan perusteet toiminnallisuuksien, kustannusten ja aikataulun seurantaan	Vaatimustenhallinta Projektisuunnittelu Projektiseuranta ja -valvonta Alihankintojen hallinta Laadunvalvonta Kokoonpanonhallinta
3. Määritelty Hallinta- ja tuotantoprosessit on dokumentoitu, standardoitu ja integroitu. Kaikki projektit käyttävät hyväksyttyä, räätälöityä versiota prosessista.	Organisaatioprosessifokus Organisaatioprosessimäärittely Koulutusohjelma Integroitu ohjelmistohallinta Ohjelmistotuotesuunnittelu Sisäinen koordinointi Katselmoinnit
4. Hallittu Yksityiskohtaista metriikkaa ohjelmistoprosesseista ja ohjelmiston laadusta kerätään. Sekä prosessi että ohjelmistotuotteet on ymmärretty ja valvottu.	Kvantitatiivinen prosessihallinta Ohjelmistolaatuhallinta
5. Optimoitu Jatkuva prosessinkehitys käyttämällä metriikkaa ja pilotoimalla innovatiivisia ideoita ja teknologioita.	Virheiden estäminen Teknologiamuutoshallinta Prosessimuutoshallinta

Jotta organisaatio pääsee CMM:n tasolle kaksi, on sen siis täytettävä tavoitteet kuudella avainprosessialueella. Näistä yksi on vaatimustenhallinta, jolla on seuraavat tavoitteet [Wieg1999]:

1. Ohjelmistovaatimuksia valvotaan luomalla perusta ohjelmistokehityksen ja hallinnan käyttöön.
2. Ohjelmistosuunnitelmat, -tuotteet ja aktiviteetit pidetään ristiriidattomina ohjelmistovaatimusten kanssa.

Vaatimustenhallinta-avainprosessialueessa ei kuitenkaan puututa vaatimusten keräämiseen ja analysointiin vaan oletetaan, että nämä kohdat vaatimustenkäsittelystä on jo suoritettu aikaisemmin. Alueessa suositellaan käytettävän version- ja muutoksenhallintaa [Wieg1999].

CMM-malli antaa monipuolisen näkymän aktiviteeteista, jotka täytyy toteuttaa ohjelmiston laadun parantamiseksi ja tuottavuuden lisäämiseksi. Vaatimustenkäsittely on kiinteä osa tätä prosessia, jossa vaatimukset ovat kokonaisuuksia ja jotka ovat kehitystoiminnan keskellä. Kerran kerättyinä vaatimuksia dokumentoidaan ja hallitaan samalla huolellisuudella kuin ohjelmistokoodituotteita. Tämä prosessi auttaa projektiryhmää hallitsemaan sekä projektia että sen vaikutusaluetta. Lopulta aktiivinen muuttuvien vaatimuksien hallinnointi auttaa pitämään projektin hallinnassa ja auttaa mahdollistamaan korkealaatuisten ohjelmistotuotteiden luotettavan ja toistettavan tuotannon [Leff2000].

4.2 Arviointi- ja kehittämismalli SPICE

Edellä esitetystä CMM-mallista ja muista vastaavista malleista on kansainvälisen standardisoimisjärjestöjen työryhmätyön tueksi aloitettu SPICE-projekti (*Software Process Improvement and Capability Determination*), jonka tarkoituksena on tuottaa standardoitu ohjelmistoprosessien kehitys- ja arviointimalli [Haik1998], [Mann2001]. Vaikka standardia (ISO/IEC 15504) ei ole vielä lopullisesti hyväksytty, niin työryhmäraporttia [ISO1998] on käytetty arvioinneissa julkaisusta lähtien. Taustalla SPICE-arviointimallin kehittämisessä on ollut se, että ISO 9000 -laatujärjestelmän sertifiointi ei ollut riittävä ja olemassa olevat mallit eivät kattaneet tarpeita [Grön1998].

Organisaatioilla on omat organisaatiokohtaiset prosessinsa. SPICE-mallissa on sovellustuotantoon liittyviä yleisiä prosesseja, joiden katsotaan liittyvän laadukkaaseen ja tuottavaan sovellustuotantoon. SPICE-prosessit ovat organisaatioriippumattomia, perustuvat käytännön kokemukseen ja ne on kerätty yhteen kansainvälisellä yhteistyöllä [Grön1998].

SPICE-mallissa arvioidaan prosessien kyvykkyyttä ja prosessit on jaettu prosessikategorioihin (taulukko 3) sen mukaan millaisia aktiviteetteja ne edustavat.

Taulukko 3. SPICE-mallin prosessikategoriat [EDM1998].

Asiakas-toimittaja	Sisältää prosessit, jotka kohdistuvat suoraan asiakkaaseen tukemalla ohjelmiston kehittämistä ja siirtoa asiakkaalle sekä takaavat ohjelmiston oikeellisuuden ja käytön.
Valmistus	Sisältää prosessit, jotka suoraan määrittelevät, toteuttavat tai ylläpitävät järjestelmää tai ohjelmistotuotetta sekä sen käyttäjädokumentaatiota.
Projekti	Sisältää prosessit, jotka muodostavat projektin sekä koordinoivat ja hallitsevat sen resursseja, jotta syntyisi tuote tai palvelu, joka tyydyttää asiakasta.
Tuki	Sisältää prosessit, jotka mahdollistavat ja tukevat muiden projektin prosessien toimintaa.
Organisaatio	Sisältää prosessit, jotka muodostavat organisaation liiketoimintatavoitteet ja kehittävät prosessia, tuotetta ja resursseja, jotka auttavat organisaatiota saavuttamaan liiketoiminnalliset tavoitteet.

Kyvykkyystasoja ovat taso 0 Ei suoritettu – taso 5 Jatkuvasti kehittyvä. Kyvykkyystasot osoittavat, miten toisten käytäntöjen tehokkuus on riippuvainen toisista käytännöistä ja siten osoittaa, mitä käytäntöjä tulee ensin parantaa [Grön1998]. Eri tasot on esitelty taulukossa 4. Tasoja mitataan luokitteluasteikolla, jossa luokat ovat [Grön1998], [EDM1998]:

- N – Ei saavutettu
- P – Osittain saavutettu
- L – Suurimmaksi osaksi saavutettu
- F – Täysin saavutettu.

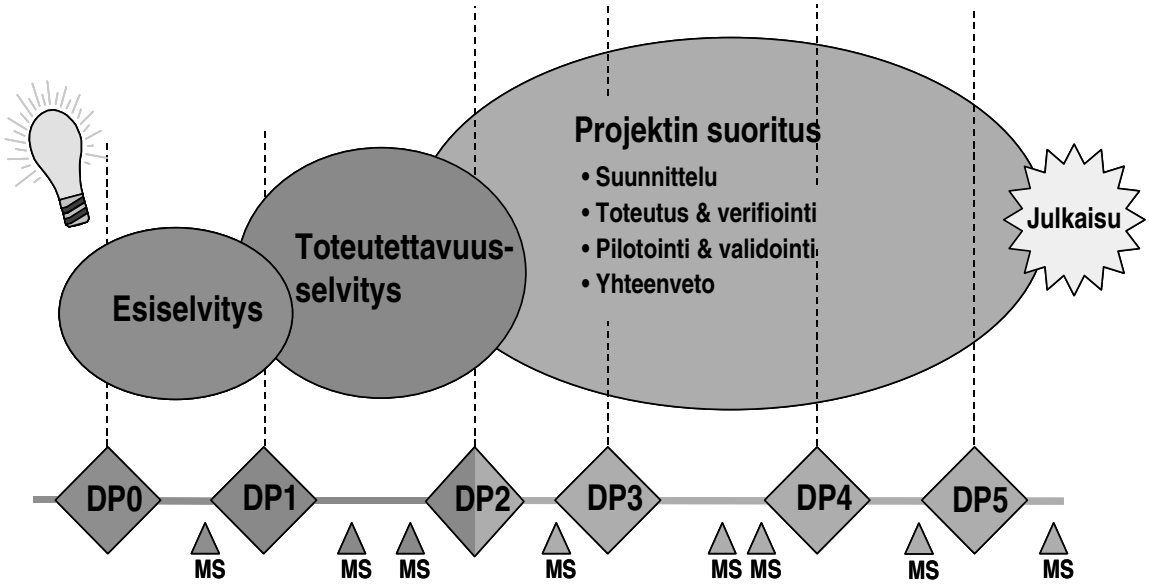
Taulukko 4. SPICE-kyvykkyystasot [EDM1998], [Grön1998].

Taso	**Tason nimi**	**Kyvykkyystason määritelmä**
Taso 0	Puutteellinen	Prosessia ei pystytä havaitsemaan, sitä ei ole tai sen toteuttamisessa on epäonnistuttu.
Taso 1	Suoritettu	Prosessin tarkoitus on pääasiallisesti saavutettu.
Taso 2	Hallittu	Prosessin tuotokset ovat standardien ja vaatimusten mukaisia.
Taso 3	Perustettu	Prosessi suoritetaan ja hallitaan yhteensopivasti hyvin määritellyn prosessin kanssa.
Taso 4	Ennustettava	Prosessi on hyvin määritelty, mitattu ja säädelty.
Taso 5	Optimoiva	Prosessin suoritus on optimoitu kohtaamaan nykyiset ja tulevat liiketoiminnan tarpeet ja prosessi saavuttaa toistuvasti sille määritellyt liiketoimintatavoitteet.

5 SONERA OYJ:N PROSESSIMALLIT

5.1 Konsernin prosessimalli

Soneran prosessimalli sisältää kahdentasoisia pisteitä: päätöspisteitä (*decision point*, DP) ja tarkastuspisteitä (*milestone*, MS). Päätöspisteet ovat ulkoisia kontrollikohtia, joiden sisältö on standardoitu. Tarkastuspisteet ovat projektikohtaisia ja ovat projektin sisäisiä kontrollikohteita [Suut2000], [Sone1999]. Kuvassa 7 on havainnollistettu päätöspisteiden suhdetta toisiinsa ja tarkistuspisteisiin.



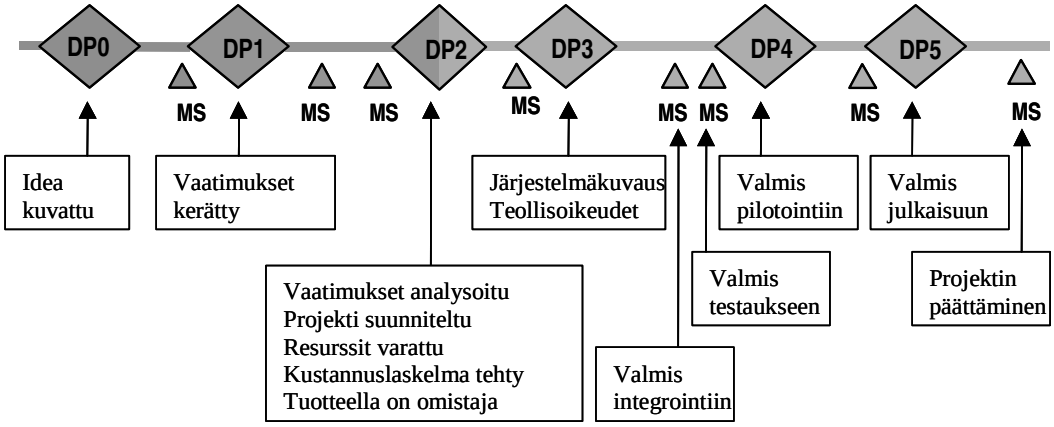
Kuva 7. Soneran yleinen prosessimalli. [Sone1999]

Jokaisessa päätöspisteessä tekee ohjausryhmä päätöksen siitä, että hyväksytäänkö, hylätäänkö vai määritelläänkö uudelleen projektin tai selvityksen seuraava vaihe. Päätöspistetoimintaan kuuluu materiaalin valmistelu ja arviointi päätöksenteon tueksi [Sone99]. Päätöspisteiden suhde projektin päätapahtumiin on esitelty kuvassa 8.

Päätöspisteet ovat seuraavanlaiset [Sone1999]:

- DP0 – päätös aloittaa esiselvitys.
- DP1 – päätös aloittaa toteutettavuusselvitys.
- DP2 – päätös aloittaa projekti.
- DP3 – päätös aloittaa toteutus suunnittelun pohjalta.

- DP4 – päätös aloittaa projektin tuotoksen pilotointi.
- DP5 – päätös projektin lopettamisesta ja tuotoksen julkaisusta.



Kuva 8. Projektin päätapahtumat. [Sone1999]

Vaatimustenkäsittely keskittyy pääasiassa DP0 ja DP2 välille niin, että DP2-pisteeseen mennessä vaatimusmäärittelydokumentti tulisi olla valmis ja hyväksyttävissä. Tämän pisteen jälkeen suoritettavat toimenpiteet vaatimusten suhteen ovat vaatimustenhallinnan piiriin kuuluvia. Erillistä kuvausta aliprosesseista ei ole, vaan prosessit kuvataan yksikkökohtaisesti.

Esiselvitysvaiheessa ennen DP1-pistettä kerätään pääasialliset vaatimukset. Tällöin mietitään asiakkaan tarpeita ja toiveita kuitenkin ottaen samalla huomioon tekniset seikat, kilpailijoiden toiminta sekä markkinatilanne. Toiminta tässä vaiheessa keskittyy pääasiassa vaatimustenmäärittelyn osa-alueeseen kerääminen. Pisteiden DP1 ja DP2 välillä analysoidaan vaatimukset ja muodostetaan niistä vaatimusmäärittelydokumentti. Tällöin suoritetaan vaatimustenmäärittelyn osa-alueet luokittelu ja määrittely sekä luodaan perusta todentamiselle ja vaatimustenhallinnalle [Sone1999].

Vaatimusten todentaminen suoritetaan ennen DP4-pistettä ja DP5-pisteeseen mennessä on suoritettu hyväksyntätestaus asiakasvaatimuksia vasten ja saatu ohjelmistolle hyväksyntä julkaisuun [Sone1999].

Uusimmassa konsernin projektimallissa, joka ei vielä tämän työn aikana ollut käytössä, on esiselvitykset siirretty ennen DP0-pistettä ja loppuun on lisätty DP6-piste, jossa projekti todetaan päättyneeksi.

5.2 Ohjelmistopalveluyksikön prosessimalli

Soneran sisäiseen ohjelmistokehitykseen keskittyvässä ohjelmistopalveluyksikössä on käytössä hieman Soneran yleisestä prosessimallista poikkeava erityisesti ohjelmistokehitykseen painotettu prosessimalli, jossa on etsitty sopivia työkaluja monimutkaisten tietojärjestelmien kehittämiseen. Pyrkimyksenä on ollut päästä järjestelmiin, jotka vastaavat asiakkaan toiveita sekä toiminnallisuuden ja aikavaatimusten suhteen [Suut2000].

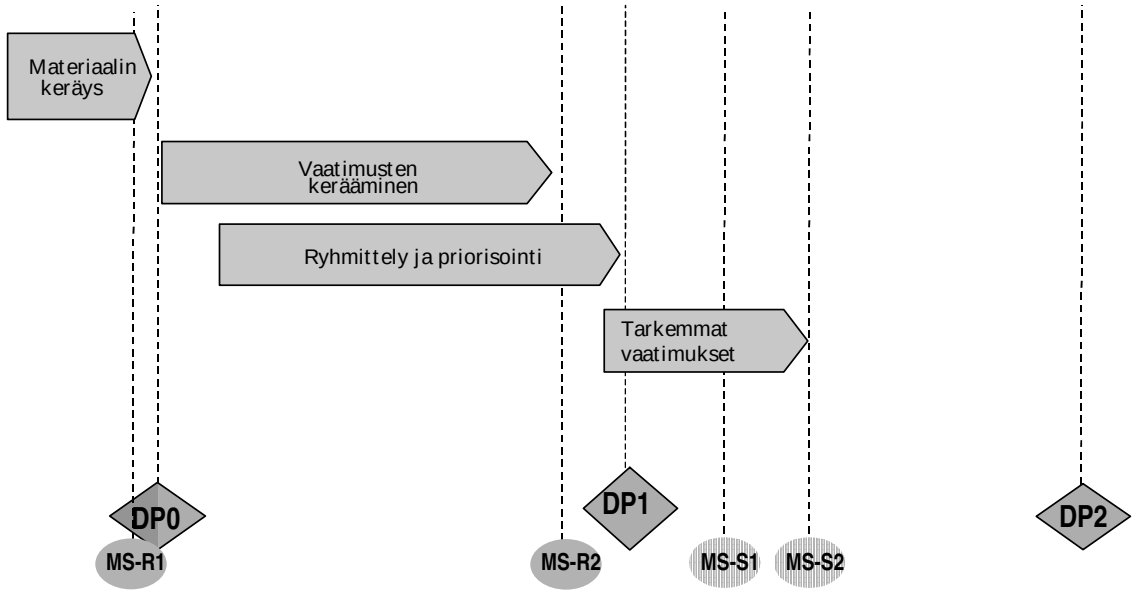
Vaatimusten vuoksi ohjelmistokehitysprosessimalli on oltava sellainen, että sillä voidaan varmistaa laadun säilyminen koko kehitysprosessin lävitse. Tätä tuetaan ohjelmistoprojektin vaihejaolla (määrittely, analyysi, suunnittelu, toteutus, testaus ja ylläpito). Ohjelmistonkehitysprosessi on komponenttipohjainen, eli kehitettävä ohjelmisto koostuu komponenteista, jotka kytkeytyvät toisiinsa rajapinnoin. Suunnittelussa käytetään UML-kieltä (*Unified Modelling Language*), joka on olennainen osa prosessia [Suut2000].

5.2.1 Vaatimustenkäsittelyprosessi

Yhtenäistä prosessikuvausta ei ole tehty vaan vaatimustenkäsittelyprosessi on jaettu yleiseen vaatimusten keräämistä ja analysointia kuvaavaan prosessiin sekä projektihallinnassa kuvattuun muutoshallintaprosessiin.

Vaatimusten keräämisessä ja analysoinnissa tarkoituksena on muodostaa keskinäinen yhteisymmärrys asiakkaan kanssa kaikista niistä järjestelmän vaatimuksista, jotka aiotaan toteuttaa. Prosessin syötteinä ovat asianosaisten kaikki tarpeet ja toiveet, jotka liittyvät uuteen tuotteeseen tai palveluun. Toisin sanoen tuotteen tai palvelun toiminnalliset, ei-toiminnalliset, laadulliset ja kaupalliset vaatimukset. Prosessin tuloksena syntyy asiakasvaatimukset DP1-pisteessä, mahdollinen demo ja käyttöliittymähahmotelma, täydellinen ja hyväksytty vaatimusluettelo sekä lyhyt tuotekuvaus tai -esitys [Lein2001].

Ennen DP1-pistettä vaatimusprosessi keskittyy vaatimusten keräämiseen ja priorisointiin ja tuotteen korkean tason toiminnallisuuden kuvaamiseen. DP1-pisteen jälkeen keskitytään ei-toiminnallisten vaatimusten keräämiseen ja valitun toiminnallisuuden tarkempaan määrittämiseen [Lein2001]. Prosessin tapahtumat päätöspisteiden suhteen on esitelty kuvassa 9 ja tarkistuspisteiden sisältö on tarkennettu taulukossa 5.



Kuva 9. Aliprosessi: vaatimusten kerääminen ja analysointi. [Lein2001]

Huomattavaa on, että vaikka yksikössä on tehty useita opinnäytetöitä vaatimustenkäsittelyn alalta, niin töiden tulosten hyödyntäminen ja jalkautus vaatimustenkäsittelyprosessin kehittämisessä on ontunut pahasti. Tuloksia on otettu käyttöön yksittäisillä osa-alueilla, kuten vaatimusten keräämisessä, mutta johdonmukaisesta läpi koko prosessin olevaa hyödyntämistä ei ole havaittavissa.

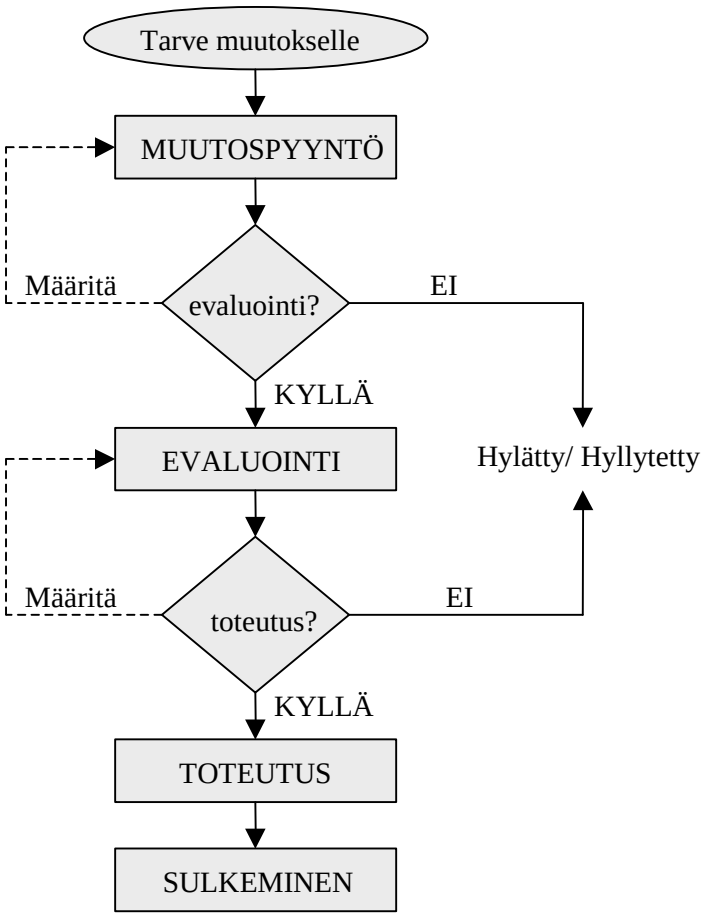
Taulukko 5. Tarkistuspisteiden alueet [Lein2001].

Tarkistuspiste	Alue
MS-R1	Vaatimusten ja alkutietojen evaluointi. Jos vaatimukset eivät ole riittävät, täytyy niitä kerätä. Jos ne ovat riittävät, niin voidaan siirtyä suoraan ryhmittelyyn ja priorisointiin. Projektia edeltävät toimenpiteet sovitaan alkutilanteen perusteella.
MS-R2	Asiakasvaatimusten katselmointi ja hyväksyntä (jäädytys). Tämän tarkistuspisteen jälkeen lisäykset tuotteen toiminnallisuuteen on tehtävä muutoshallinnan kautta.
Järjestelmäkonsepti-prosessin tarkistuspisteet	MS-S1: Kaikki vaatimukset kerätty. Asiakasvaatimukset analysoitu ja analyysin perusteella on identifioitu tekniset vaatimukset. MS-S2: Toteutettavuus arvioitu. Vaatimusluettelo jäädytetty. Yhteisymmärrys saavutettu asiakkaan kanssa siitä, millainen tuote tai palvelu toteutetaan.

Muutoshallinnassa käsittelyvaiheita ovat dokumentoitu muutospyyntö, evaluointi, päätöksenteko hyväksynnästä tai hylkäämisestä, toteutus ja sulkeminen. Muutoshallinta on tärkeää, koska se auttaa sekä projektinhallinnassa että tuotteen tai palvelun kehittämisessä. Toimiva muutoshallinta tarjoaa seuraavat edut [Ylim2001]:

- Se, mitä on suunniteltu ja sovittu, myös tehdään.
- Asiakasta suojellaan odottamattomilta kuluilta.
- Estää muutospyyntöjen leviämisen.
- Tarpeellisten muutosten parempi näkyvyys.
- Laadukkaampia päätöksiä, kun kaikki osapuolet on mukana tekemässä niitä.
- Kaikki muutokset on dokumentoitu.
- Hyvä muutoshallinta on edellytys tuotetiedonhallinnalle.

Muutoshallinnan prosessikaavio on esitelty kuvassa 10, jossa syötteenä on tarve muutokselle ja lopputuloksena on toteutettu muutos tai hylätty/hyllytetty muutospyyntö.



Kuva 10. Muutospyynnön käsittelyvaiheet. [Yli2001]

6 INTELLITEL COMMUNICATIONS OY:N PROSESSIMALLI

6.1 Yleinen prosessimalli

Intellitel Communications Oy (Intellitel) kehitti, markkinoi ja tuki tietoliikenneverkkoihin älykkäitä väliohjelmaratkaisuja (*Network Intelligence Middleware, NIM*), joilla pystyttiin luomaan ja ottamaan käyttöön uusia tele- ja tietoliikennepalveluja. Palveluohjelmistot räätälöitiin asiakkaiden tarpeiden mukaan itse kehitetyn palvelualustan päälle, joka alkujaan oli ollut tarkoitettu älyverkkopalveluiden kehitys- ja tuotantoalustaksi [Inte1999].

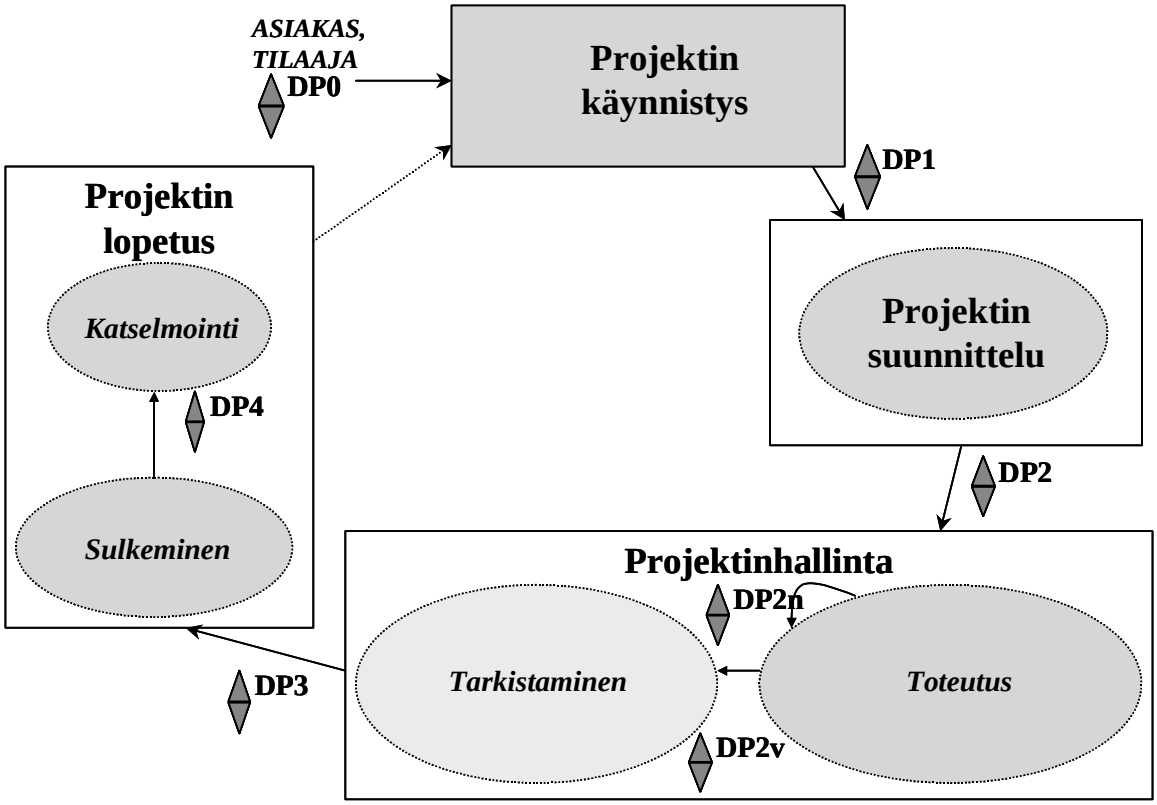
Prosessikehitys alkoi Intellitelissä yrityksen kasvaessa samalla, kun syntyi tarve tuotteistamiseen siirryttäessä yhdestä tuotteesta ja asiakkaasta useiden tuotteiden ja sidosryhmien ympäristöön. Ensimmäiseksi keskityttiin yleiseen tuotekehitysprosessiin ja erityisesti testauksen kehittämiseen, jotka ovat asiakkaalle suoraan näkyviä osioita lopputuloksen ja dokumenttien muodossa. Tuotekehitysprosessissa tunnistettiin kyllä eri vaiheita, joiden mukaan toimintaa ja dokumentteja oli ryhmitelty, mutta suoranaisia aliprosesseja ei vielä tässä vaiheessa oltu tunnistettu.

Vaatimustenkäsittely nähtiin vain kahtena dokumenttina, vaatimusluettelona ja analysoituna vaatimusluettelona. Myöskään jäljitettävyydestä läpi ohjelmistotuotteen kehitysprojektin tai tuotteen elinkaaren ei ollut selkeää ohjeistusta.

Intellitelin prosessimallia (kuva 11) lähdettiin muokkaamaan Soneran (silloin Telecom Finland) ohjelmistopalveluyksikön edeltäjän prosessimalliin pohjautuen. Prosessimallin päätarkoituksena oli tukea syklistä ohjelmistokehitystä, missä asiakasvaatimukset ohjelmiston räätälöinnistä toimivat syötteenä uuden ohjelmistoversion suunnittelulle. Selkeimpänä erona aikaisemmin esiteltyihin prosessimalleihin on, että tässä mallissa päätöspisteitä oli vain neljä, joista DP2-pistettä voidaan toistaa projektinhallintavaiheessa niin useasti kuin tarve vaatii.

Prosessikehityksen avuksi otettiin FiSMA (Finnish Software Measurement Association) SPICE –menetelmä, joka on sovitettu suomalaisten yritysten tarpeisiin ja tarjoaa lisäksi

valmiit arviointilomakkeet. Prosessien kyvykkyyttä projekteissa arvioitiin tällä menetelmällä myös ulkopuolisten arvioitsijoiden toimesta.



Kuva 11. Intellitelin prosessimalli. [Inte2002]

6.2 Vaatimustenkäsittelyprosessi

Alkutilassa vaatimuksia koottiin, ei siis järjestelmällisesti kerätty kappaleen 2.1.1 merkityksessä, vaatimusdokumenttiin ja niitä analysoitiin jonkin verran. Vaatimusten todentamista ja havaittavaa muutoshallintaa ei myöskään ollut, vaan saatettiin projektin loppuraportissa mainita vaatimuksiin tulleita poikkeamia. Nämä muutosten kohteet olivat kuitenkin useissa tapauksissa enemmän toiminnallisia kokonaisuuksia kuin varsinaisia yksiselitteisiä vaatimuksia.

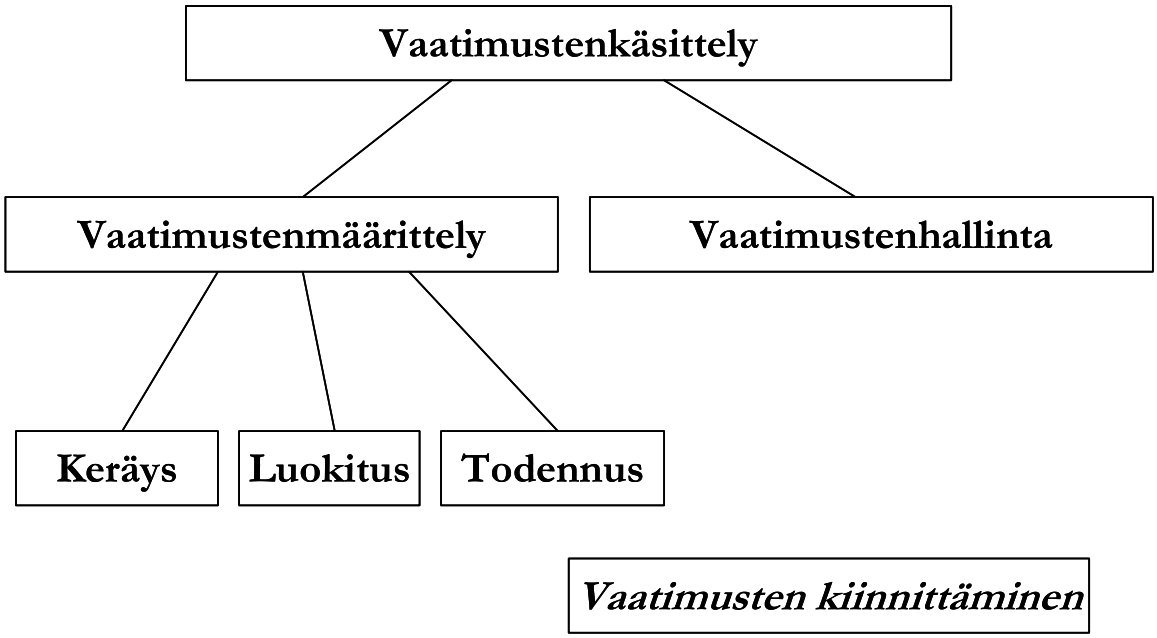
Tällainen tilanne ei kuitenkaan tuolloin ollut mitenkään harvinainen suomalaisissa ohjelmistoyrityksissä. Kuten tutkimusraportissa [Niku2000] on havainnoitu, että vaatimustenkäsittely oli uutta ja tuntematonta useissa pienissä ja keskisuurissa yrityksissä ja myös prosessikehityksen aloittaminen tällä alueella oli vaikeaa. Samaan

aikaan tietoisuuden kasvaessa erilaisista arviointimenetelmistä, alkoi tarve vaatimustenkäsittelyprosessin kehittämiselle ja käyttöönotolle kasvaa.
Erityisinä tarpeina nähtiin juuri raportissakin [Niku2000] mainitut:

1. omien vaatimustenkäsittelyprosessitapojen kehitys,
2. vaatimustenkäsittelyprosessin parantaminen ja
3. vaatimustenkäsittelyn automatisoiminen.

Aluksi tutkittiin eri vaatimustenkäsittelyn tapoja ja havaittiin, että on tarve tarkentaa vaatimukselle standardissa IEEE Std 610.12-1990 asetettua määritelmää siten, että vaatimuksen on oltava atomaarinen eli sitä ei voida enää jakaa osiin. Tietenkin myös edellytetään, että vaatimus on todennettavissa ja mitattavissa. Jälkimmäisen ehdon täyttäminen nähtiin erittäin haastavaksi ja sen tarpeellisuudesta oltiin erimielisiä. Lopulta ymmärrettiin, että kyse on vain uusien mittareiden ja mittaustapojen kehittämisestä ja että mitattavuus on yksi laatutavoite.

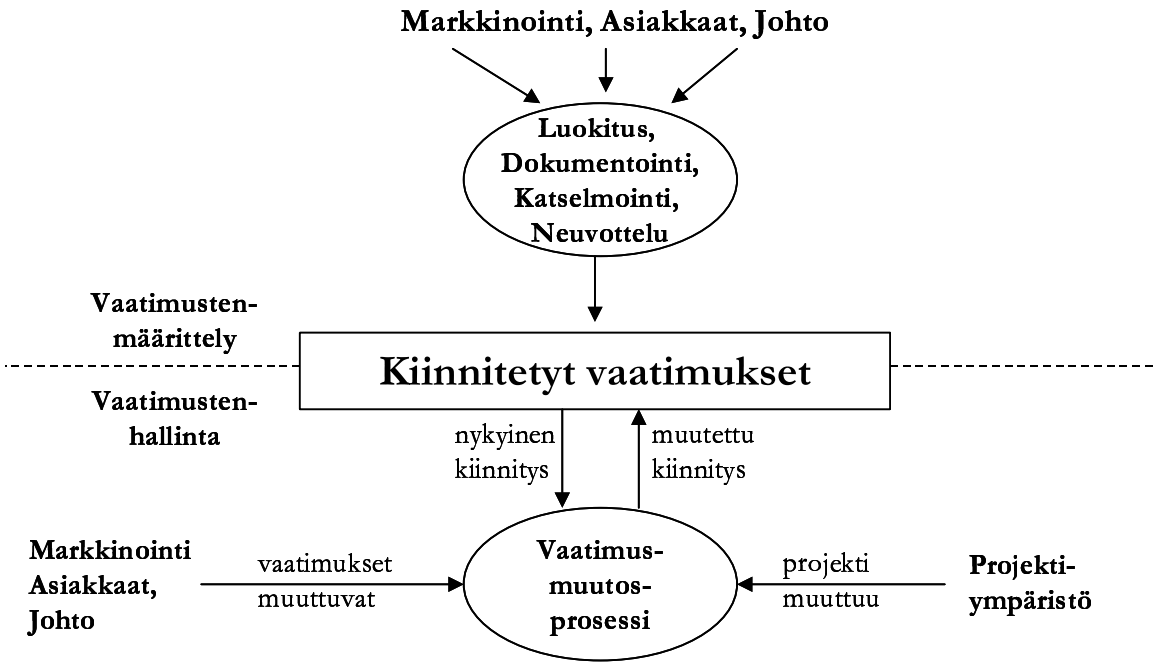
Todettiin, että vaatimustenkäsittelyssä on hyvä noudattaa kuvassa 3 esitettyä kolmijakoa asiakkaan tarpeiden, toiminnallisuuksien ja ohjelmistovaatimusten välillä. Kuvassa 12 on esitelty Intellitelissä käyttöön otettu jako vaatimustenkäsittelyn tietämysalueella.



Kuva 12. Vaatimustenkäsittelyn alueet. [Inte2002]

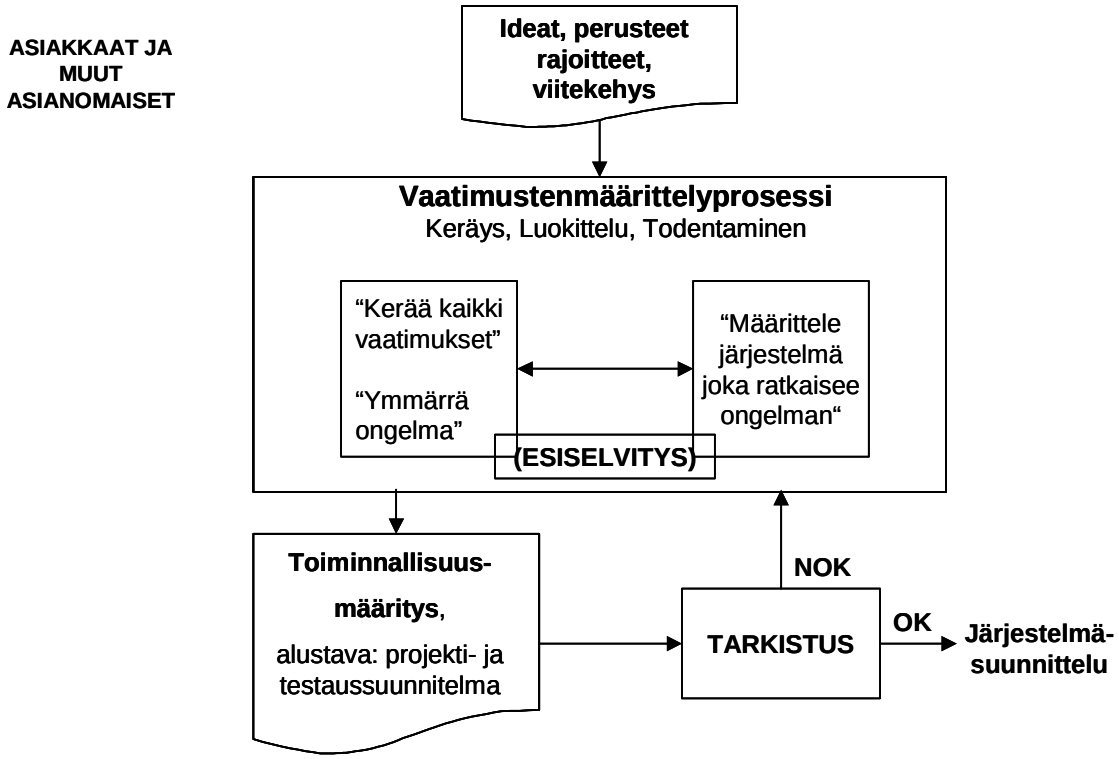
Voidaan huomata, että käyttöön otettu malli on lähes sama kuin kuvassa 2 esitetty malli. Poikkeamina vain, että vaatimusmäärittely on yhdistetty osaksi luokittelua ja uutena osana on nostettu esiin vaatimusten kiinnittäminen. Kiinnittäminen on versionhallintaa, mutta erittäin tärkeä osa sitä ja keino millä siirtyminen vaatimustenmäärityksestä vaatimustenhallintaan tapahtuu sujuvasti ja informaatiota kadottamatta.

Kuvassa 13 on hahmoteltu, mistä peruslähtökohdista vaatimustenkäsittely muodostuu ja miten sen eri osa-alueet ovat suhteessa vaatimusten kiinnittämiseen.



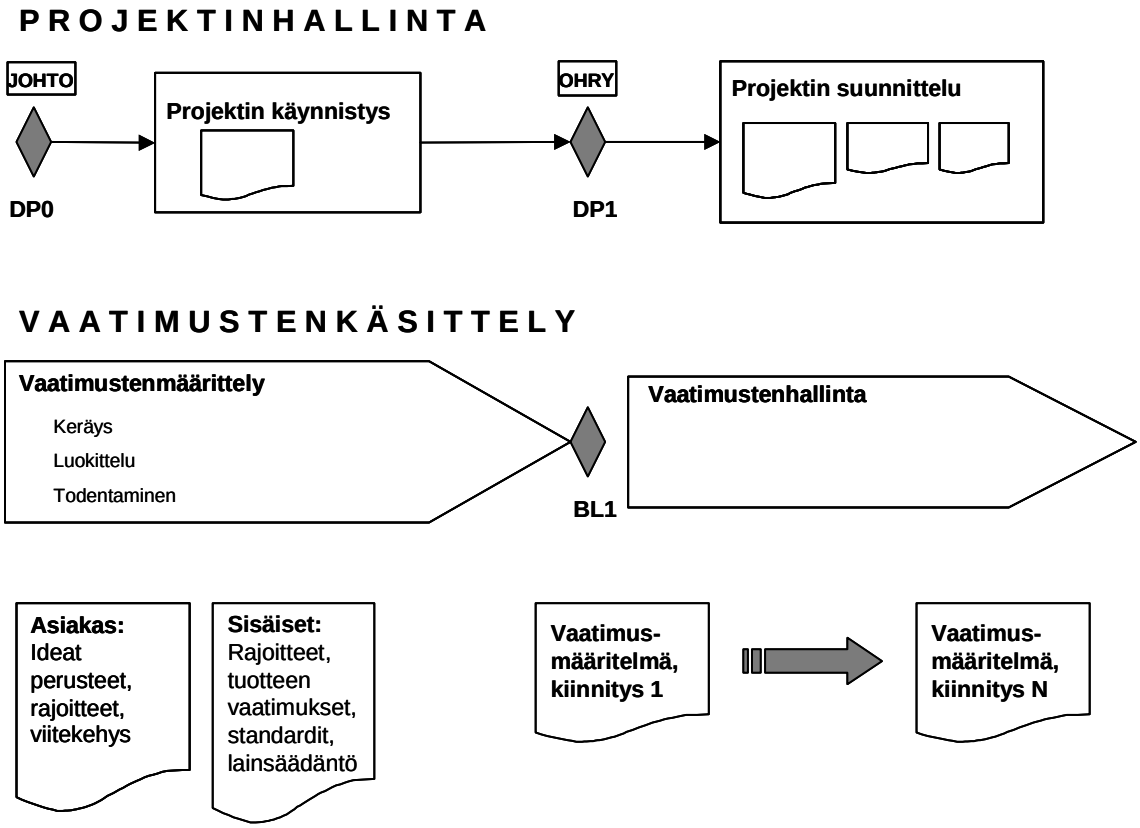
Kuva 13. Vaatimustenkäsittelyn peruslähtökohdat. [Inte2002]

Kuvassa 14 on esitelty vaatimustenmäärittelyn lähtökohdat, jossa on asiakkaan ja muiden asianomaisten toimesta tuotu esille lähtökohtia, joista vaatimustenmäärittelyprosessin kautta lähdetään muokkaamaan vaatimusmäärittelyä, johon järjestelmäsuunnittelu perustuu. Yhtenä useimmin käytettynä aloitustapana on käynnistää esiselvitys, jossa etsitään korkean tason vaatimukset ja tehdään alustava järjestelmäkuvaus, etsitään rajapinnat sekä toiminnallisuudet. Nämä osa-alueet avustavat tarvittavan työmäärän, aikataulun ja kustannusten määrittelyssä ja suunnittelussa.



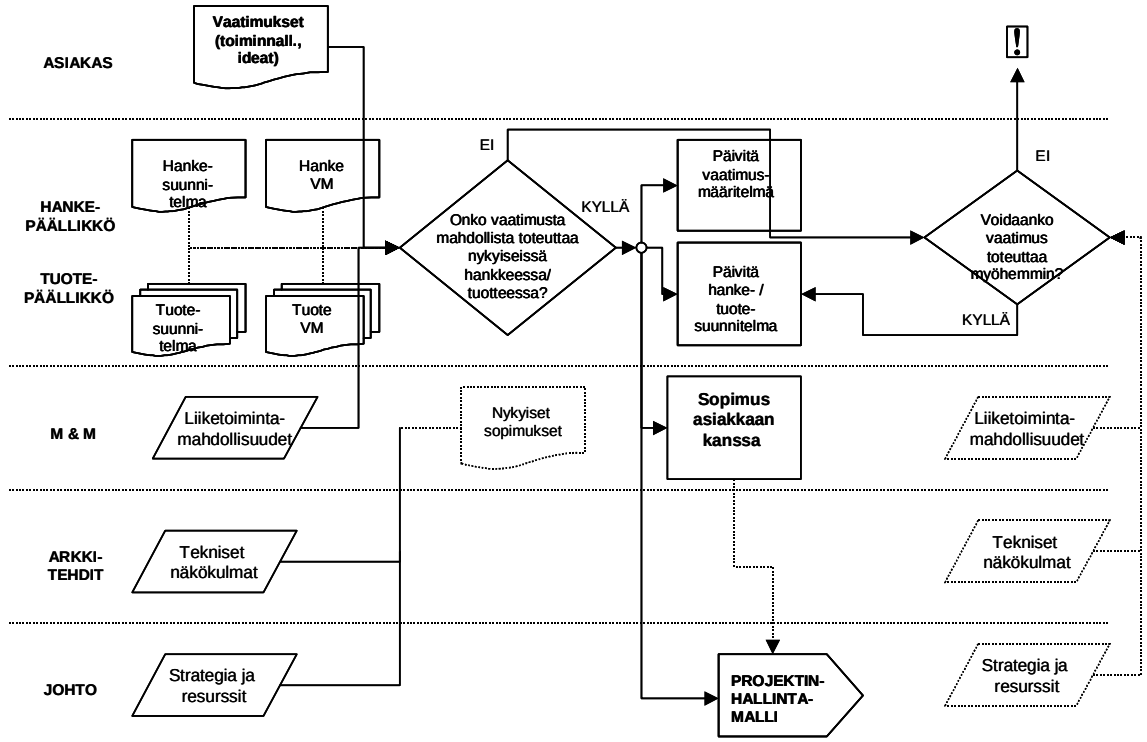
Kuva 14. Vaatimustenmäärittelyn lähtökohdat. [Inte2002]

Projektihallintamallin päätöspisteiden ja vaatimustenkäsittelyprosessin suhde toisiinsa ja vaatimusten kiinnittämiseen on esitelty kuvassa 15. Huomattavaa on, että vaatimustenmäärittely suoritetaan ennen DP1-pistettä ja sen jälkeen tehtävät toimenpiteet katsotaan olevan vaatimustenhallintaa. Varsinaisessa projektin suunnittelussa pitäisi olla jo olemassa ensimmäinen hyväksytty versio vaatimusmäärittelydokumentista.

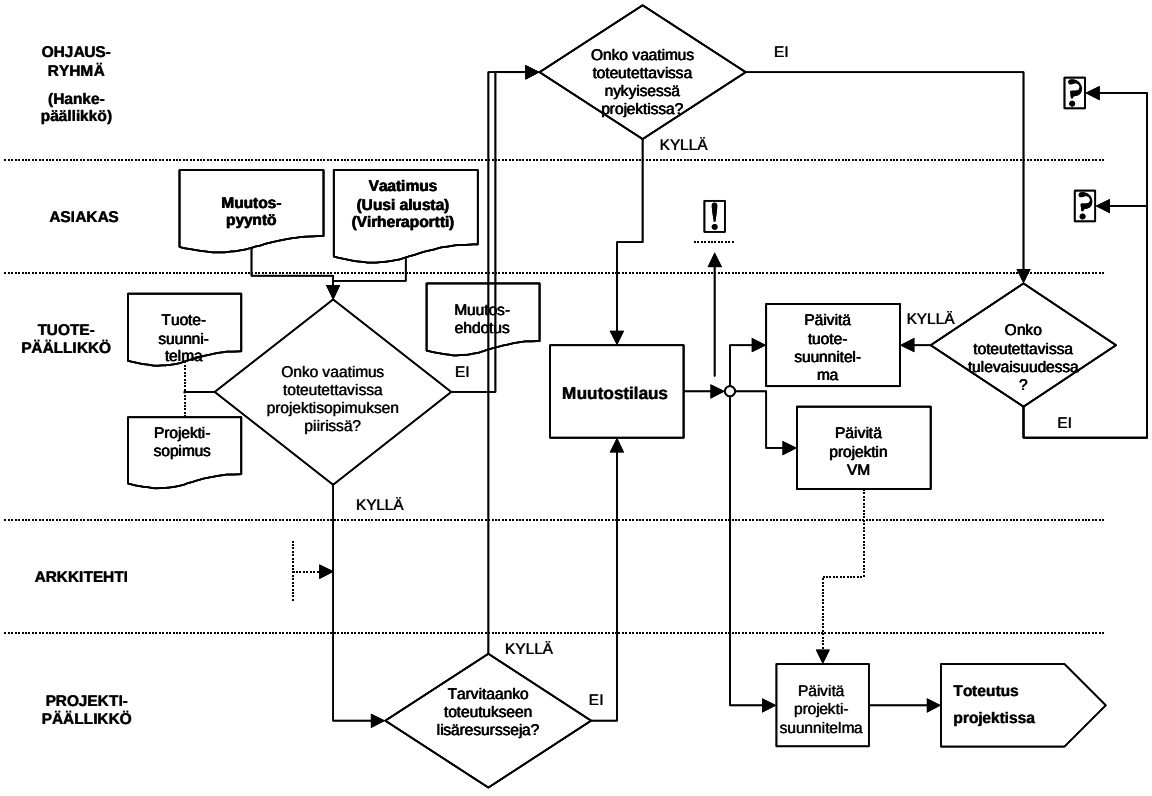


Kuva 15. Vaatimusten kiinnittäminen. [Inte2002]

Intellitelin vaatimustenkäsittelyprosessin suunnittelussa tärkeänä lähtökohtana oli, että kehitetty prosessimalli on käytettävissä sekä ohjelmistotuotteiden teossa että ohjelmistoprojektien suorittamisessa. Kuvassa 16 esitellyssä vaatimustenmäärittelyprosessissa ja kuvassa 17 esitellyssä vaatimustenhallintaprosessissa mainittu tuote voidaan ilman muutoksia prosessiin korvata ohjelmistoprojektilla ja vastaavasti tuotepäällikkö projektipäälliköllä tai hankepäälliköllä. Tällöin vaatimustenhallintaprosessissa projektipäällikkö on kahdessa roolissa, jos projekti ei kuulu mihinkään tuoteryhmään tai hankeohjelmaan.

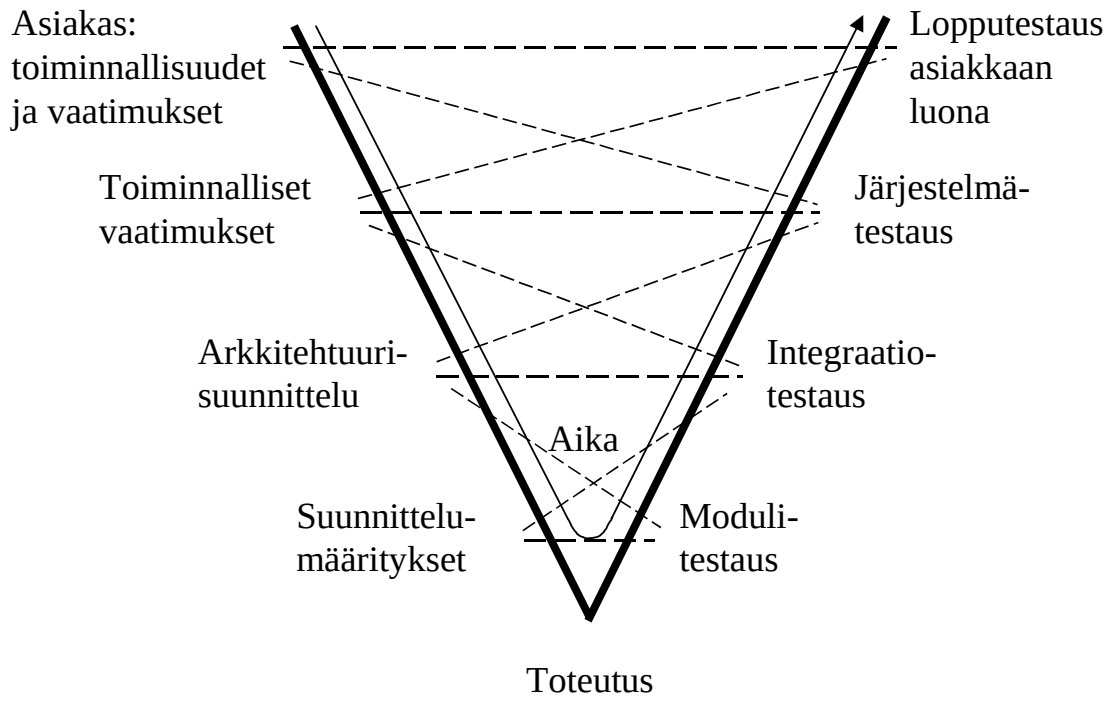


Kuva 16. Vaatimustenmäärittelyprosessi. [Inte2002]



Kuva 17. Vaatimustenhallintaprosessi. [Inte2002]

Vaatimusten todentamista varten ja jäljitettävyyden havainnollistamiseksi Intellitelissä otettiin käyttöön muunneltu versio ohjelmistokehityksen V-mallista. Tässä mallissa eri suunnitteluvaiheet eivät vastaa ainoastaan yhtä vastakkaista testausvaihetta vaan myös rinnakkaisia vaiheita. Samoin toisinpäin eli testausvaiheen vastakappaleena on vastakkaisen suunnitteluvaiheen lisäksi sen viereiset suunnitteluvaiheet. Vastaavuudet on esitelty kuvassa 18.



Kuva 18. Intellitellissä käyttöön otettu todentamisketju ja eri vaiheiden väliset yhteydet. [Inte2002]

7 JOHTOPÄÄTÖKSET

Ohjelmistotuotteen tai -palvelun laatu on hyvin abstrakti käsite ja usein toimittajan on vaikeaa todentaa asiakkaille, kuinka laadukkuus on saavutettu. Toimivan vaatimustenkäsittelyprosessin lopputuloksena saavutetaan tällainen todentamisketju (kuva 18) laadulle asiakkaan tarpeista ja niistä muokatuista vaatimuksista valmiiseen tuotteeseen tai palveluun.

Tätä tuotekehityksen tai palvelun tilauksen yhteydessä tehtyä tarvekartoitusta ja vaatimusten keräämistä ei riittävästi osata tuoda esiin puhuttaessa lopputuloksen laadusta. Usein vaatimustenkäsittelyä tärkeyttä perustellaan sillä, että mitä myöhemmin ohjelmisto elinkaaren aikana virhe havaitaan, niin sitä kalliimmaksi sen korjaaminen muodostuu. Tämä on totta, mutta se ei ole ohjelmistopalvelutyritykselle oikea tapa perustella asiakkaalle vaatimustenkäsittelyn tärkeyttä.

On vähättelevää pitää vaatimustenkäsittelyä vain tapana säästää kustannuksissa ja sellainen vähättely myös vähentää asiakkaan halua panostaa resursseja siihen ja hyväksyä sen aiheuttamia kustannuksia. Olisi ymmärrettävä, että panostukset vaatimustenkäsittelyyn alkuvaiheessa näkyvät lopputuotteessa kilpailuetuna laadussa ja toivottuina ominaisuuksina. Tämä ei kuitenkaan nykyään toteudu ilmeisesti se vuoksi, että toimitussopimuksia tekeviltä ja hyväksyviltä puuttuu riittävä tieto ja ymmärrys ohjelmistotuotteiden ja -palveluiden laadunvalvonnasta.

Vaatimustenkäsittelyprosessin aikaansaaminen yritykseen ei ole helppoa, vaikkakin nykyään on olemassa menetelmiä valmiina käyttöönotettavista malleista. Kuitenkin on todennäköisempää, että prosessin käyttöönottaminen vie aikaa ja vaatii panostusta sekä räätälöintiä yrityksen tarpeisiin kuin että löydettäisiin yritykselle sopiva sellaisenaan valmis prosessimalli. Prosessimallipohjat, jotka ovat adaptoitavissa yrityksen tarpeisiin, ovat kuitenkin tarpeellisia ja niitä tulisikin kehittää enemmän räätälöitäviksi, koska yrityksillä on usein samoista lähtökohdista olevia ongelmia, joihin voidaan soveltaa räätälöitäviä ratkaisuja.

Laatujärjestelmiä yrityksiin luotaessa usein korostetaan, että johto on sitoutettava laatuun, jotta asiassa voidaan onnistua. Sama pätee vaatimustenkäsittelyssä eli jalkauttamisen on lähdettävä johdosta, eikä johdolle tai myynnille tule sallia myöskään poikkeuksia, vaikka kuinka asiakasprojekti olisi tärkeä ja että asiakasta ei pidä vaivata liialla byrokratialla. Todennäköisempää on, että projekti lopulta epäonnistuu puutteellisen muutoshallinnan takia, jolloin olisi ehkä ollut parempi, ettei sitä olisi aloitettukaan.

Jalkauttamisessa ei myöskään riitä, että kirjataan noudatettavat prosessit ja sen jälkeen voidaan todeta kaiken olevan kunnossa. Johdon on valvottava, että prosesseja noudatetaan. Vaatimustenkäsittelyprosessia on verrattava muihin prosesseihin sekä arviointimenetelmiä käyttäen että ulkoisella ja sisäisellä auditoinnilla, jolloin voidaan tunnistaa puutteita ja pullonkauloja.

Kuitenkaan prosessikehitysmalleihin perustuvat arviointimenetelmät eivät ole tässä yksistään riittäviä, sillä niissä käytännössä prosessia arvioidaan dokumenttien ja haastateltavan subjektiivisen näkemyksen perusteella. Tällöin ulkopuolisen arvioijan on vaikeaa hahmottaa – varsinkin jos prosessit ja projektit ovat epätyypillisiä – millainen todellinen käytäntö yrityksessä on ollut. Tähän olisi ratkaisuna, että ohjelmistopalveluyrityksissä on sisäisiä arvioitsijoita, mutta valitettavasti nykyisessä talouden tilanteessa yritykset näkevät tällaiset erikoisosaajat vain tuottamattomina henkilöinä.

Jotta vaatimusten todentaminen on mahdollista, täytyy ohjelmistopalveluyrityksellä olla toimiva tapa suorittaa todentamista, mikä käytännössä tapahtuu vaatimusten jäljitettävyyden avulla. Jäljitettävyyden ylläpitäminen manuaalisesti on hyvin työlästä ja suorastaan mahdotonta suurissa ohjelmistoprojekteissa. Vaatimustenkäsittelyyn tarkoitetuilla työkaluohjelmistoilla jäljitettävyys saadaan pysymään helposti yllä koko vaatimuksen ja ohjelmiston elinkaaren.

Työkaluohjelmien käyttö on erittäin suositeltavaa, sillä tällöin on helpompaa keskittyä itse vaatimusten sisältöön ja jättää jäljitettävyys, versionhallinta sekä muut

rutiiniluontoiset toiminnot ohjelmiston suoritettavaksi. Kuitenkin tässä piilee myös vastakkainen ongelma, sillä herkästi unohdetaan ettei ohjelmisto itsessään tee mitään vaan edelleen tarvitaan henkilöstön panosta vaatimusten keräämisessä ja luokittelussa. Henkilöstöä on koulutettava nimenomaan vaatimustenmäärittelyssä ja käyttämään työkaluohjelmistoa vaatimustenkäsittelyn apuvälineenä.

8 YHTEENVETO

Vaatimustenkäsittely ja sen osa-alueista varsinkin muutoshallinta on erittäin kriittinen osa ohjelmistoprojektien onnistumisen kannalta. Voidaan perustellusti sanoa, että ohjelmistopalveluyrityksen elinehtona on toimiva vaatimustenkäsittelyprosessi. Tässä työssä on esitelty määritelmiä vaatimustenkäsittelyn eri osille, kuvattu prosesseihin liittyvää laadunvalvontaa ja kuinka laatua voidaan parantaa sekä esitelty prosessien arviointi- ja kehittämisjärjestelmiä.

On myös tarkasteltu vaatimustenkäsittelyä kahdessa yrityksessä: räätälöityjä ohjelmistotuotteita ja ohjelmistoprojekteja tekevässä sekä pelkkiä ohjelmistoprojekteja tekevässä yrityksessä ja vertailtu näiden yritysten vaatimustenkäsittelyprosesseja. Tuloksena on esitelty havaintoja ohjelmistopalveluyrityksen vaatimustenkäsittelystä ja prosessimallista sekä mahdollisia parannusehdotuksia näihin.

Vaatimustenkäsittelyyn on olemassa paljon erilaisia työkaluja, mutta on muistettava, että työkalut eivät kuitenkaan yksinään tee sitä työtä, mitä vaatimustenkäsittely ohjelmistopalveluyrityksessä vaatii. Kuten kaikessa laatutoiminnassa ja prosessikehityksessä, niin vaatimustenkäsittelyssäkin ratkaisevaa on kuinka toiminta pystytään jalkauttamaan ja että koko organisaatio sitoutuu myös siihen johdosta yksittäisiin työntekijöihin asti. Jalkautuksen kompastuskivinä esiintyy usein muutosvastarintaa, joka ilmenee haluttomuutena muuttaa totuttuja toimintatapoja sekä opetella uusien työkalujen käyttöä.

Valitettavan usein ohjelmistoalalla ylenkatsotaan vaatimusmäärittelyn tekemistä projektin alkuvaiheessa. Asiakkaalla on kiire saada tuote markkinoille eikä myöskään olla valmiita maksamaan muusta kuin toteuttamiseen kohdistuvasta työstä. Myöskään vaatimustenhallintaan ja sen formaaliin toteuttamiseen ei nähdä tarpeelliseksi sitoutua, eikä myöskään ole otettu vaatimustenkäsittelyn puutteita huomioon riskienhallinnassa.

Tämä lyhytnäköisyys on kostautunut projektien viivästymisillä, budjetin ylityksinä ja jopa koko projektin epäonnistumisina, jolloin nopeasti markkinoille haluttua tuotetta ei

ole koskaan saatu myyntiin ja myyntitulot ovat jääneet saamatta ja sitoutuneet kustannukset sekä panostukset on menetetty.

9 LÄHTEET

[Boeh1981] Boehm, Barry W., Software engineering economics, 1981, ISBN 0-13-822122-7.

[Bray2002] Bray, Ian K., An Introduction to Requirements Engineering, 2002, ISBN 0-201-76792-9.

[Crnk1999] Crnkovic, Ivica, Funk, Peter, Larsson Magnus, Processing Requirements by Software Configuration Management, Proceedings of the 25th EUROMICRO Conference, Milan, 8-10 Sept. 1999, Volume 2, ISSN 1089-6503, pp. 260-265.

[Davi1993] Davis, Alan M., Software Requirements: Objects, Functions, and States, 1993, ISBN 0-13-562174-7.

[EDM1998] El Emam, Khaled, Drouin, Jean-Normand, Melo Walcélio, SPICE: the theory and practice of software process improvement and capability determination, 1998, ISBN 0-8186-7798-8.

[IEEE1990] Institute of Electrical and Electronics Engineers, IEEE Std 610.12-1990, IEEE Standard Glossary of Software Engineering Terminology, 1990, ISBN 1-55937-067-X.

[IEEE1998] Institute of Electrical and Electronics Engineers, IEEE Std 830-1998, IEEE Recommended Practice for Software Requirements Specification, 1998, ISBN 0-7381-0332-2.

[IEEE2001] IEEE SWEBOK, Guide to the software engineering body of knowledge : trial version (version 1.00), toim.: Abran, Alan, Moore, James W., 2001, ISBN 0-7695-1000-0.

[Inte1999] Intellitel Communications Ltd, Enabling Convergence: Network Intelligence Middleware, White Paper, 3.3.1999.

[Inte2002] Intellitel Communications Ltd, Sisäisiä kalvoesityksiä vuosilta 2001-2002.

[Ishi1986] Ishikawa, Kaoru (ed.), Guide to Quality Control, 2nd Revised Edition (clarified), 1986, ISBN 92-833-1036-5.

[ISO1995] International Organization for Standardization, ISO/IEC 12207:1995(E), Information technology – Software life cycle processes, 1998.

[ISO1998] International Organization for Standardization, ISO/IEC TR 15504, Information technology – Software process assessment, 1998.

[ISO2001] International Organization for Standardization, ISO/IEC 9126-1:2001(E), Software engineering – Product quality – Part 1: Quality model, 2001.

[Grön1998] Gröndahl, Lars, Telen tietoliikenneohjelmistojen alihankintaprosessien arviointi ja kehittäminen, 1998, Helsingin yliopisto, Fysiikan laitos, Pro Gradu -tutkielma.

[Haik1998] Haikala, Ilkka, Märijärvi, Jukka, Ohjelmistotuotanto, 6. painos, 1998, ISBN 951-762-696-7.

[Hokk2001] Hokkanen, Mia, Requirement Traceability, 2001, Lappeenrannan teknillinen korkeakoulu, Diplomityö.

[Horc1996] Horch, John W., Practical guide to software quality management, 1996, ISBN 0-89006-865-8.

[Hump1995] Humphrey, Watts S., A Discipline for Software Engineering, 1995, ISBN 0-201-54610-8.

[Kauk2000] Kaukavuori, Sami, Requirements Management in the Software Development Process, 2000, Lappeenrannan teknillinen korkeakoulu, Diplomityö.

[Kuja2001] Kujala, Ari, Achieving Traceability, 2001, Lappeenrannan teknillinen korkeakoulu, Diplomityö.

[Leff2000] Leffingwell, Dean, Widrig, Don, Managing Software Requirements: A Unified Approach, 2000, ISBN 0-201-61593-2.

[Lein2001] Leino, Kari, Sub-process: Requirement capture process, Version 1.1, 8.8.2001, Sisäinen dokumentti, Sonera Oyj.

[Mann2001] Mannio, Markus, Software process improvement using an electronic process guide, 2001, Lappeenrannan teknillinen korkeakoulu, Diplomityö.

[Mark2001] Markkula, Pekka, Requirements Engineering and Guidelines for a Product Development Process, 2001, Tampereen teknillinen korkeakoulu, Diplomityö.

[Mayr1990] Von Mayrhauser, Anneliese, Software engineering: methods and management, 1990, ISBN 0-12-727320-4.

[Niku2000] Nikula, Uolevi, Sajaniemi, Jorma, Kälviäinen, Heikki, A State-of –the-Practice Survey on Requirements Engineering in Small- and Medium-Sized Enterprises, 2000, Telecom Business Research Center Lappeenranta, Research Report 1, ISBN 951-764-431-0.

[Pres1997] Pressman, Roger S., Software Engineering: A Practitioner's Approach, 4th Edition, International Edition, 1997, ISBN 0-07-114603-2.

[Rein2000] Reinikainen, Lea, Elicitation of Customer Requirements with Group Methods in Software Engineering, 2000, Lappeenrannan teknillinen korkeakoulu, Diplomityö.

[Smol2002] Smolander, Kari, K. Hoikka, J. Isokallio, M. Kataikko ja T. Mäkelä, What is Included in Software Architecture? A Case Study in Three Software Organizations, Proceedings on the 9th annual IEEE International Conference and Workshop on the Engineering of Computer-Based Systems (ECBS), 8-11 April 2002, Lund, Sweden, IEEE Computer Society, pp. 131-138.

[Smol2003] Smolander, Kari, On the Role of Architecture in Systems Development, Doctoral Discussion, Lappeenranta University of Technology, 2003, ISBN 951-764-735-2.

[Somm1997a] Sommerville, Ian, Sawyer Pete, Requirements Engineering: A Good Practice Guide, 1997, ISBN 0-471-97444-7.

[Somm1997b] Sommerville, Ian, Sawyer Pete, Viewpoints: principles, problems and a practical approach to requirements engineering, Annals of Software Engineering, Volume 3, 1997, ISSN 1022-7091, pp. 101-130.

[Somm2001] Sommerville, Ian, Software engineering, 6th edition, 2001, ISBN 0-201-39815-X.

[Sone1999] Sonera Ltd, Sonera R&D Process Model, Revision 1.1, 15.3.1999.

[STQE2001] STQE Magazine, Nov/Dec 2001, ISSN 1532-3579.

[Suut2000] Suutari, Hanna-Kaisa, Toimintopisteanalyysin soveltaminen ohjelmistoprojektin koon ja laadun arviointiin, 2000, Lappeenrannan teknillinen korkeakoulu, Diplomityö.

[Thay1997] Thayer, Richard H., Dorfman, Merlin, ed., Software Requirements Engineering, 2nd edition, 1997, ISBN 0-8186-7738-4.

[Wieg1999] Wiegers, Karl E., Software Requirements, 1999, ISBN 0-7356-0631-5.

[Wieg2001] Wiegers, Karl E., Requirements: When the Field Isn't Green, STQE Magazine, May/June 2001, ISSN 1532-3579, pp. 30-37.

[Ylim2001] Ylimäki, Yrjö, Change management procedure, Version 1.2, 5.11.2001, Sisäinen dokumentti, Sonera Oyj.

[Zahr1998] Zahran, Sami, Software Process Improvement: Practical Guidelines for business success, 1998, ISBN 0-201-17782-X.