Vaatimusmäärittely projektinhallinnan malleissa

HÄMEEN AMMATTIKORKEAKOULU
HÄME UNIVERSITY OF APPLIED SCIENCES

Ammattikorkeakoulututkinnon opinnäytetyö

Tietojenkäsittelyn koulutusohjelma

Hämeenlinna, Kevät 2017

Teppo Tuominen

HÄMEEN AMMATTIKORKEAKOULU
HÄME UNIVERSITY OF APPLIED SCIENCES

Tietojenkäsittelyn koulutusohjelma
Hämeenlinna

Tekijä Teppo Tuominen **Vuosi** 2017

Työnnimi Vaatimusmäärittely projektinhallinnan malleissa

TIIVISTELMÄ

Työn tavoitteena oli etsiä vaatimusmäärittelyyn eroja eri projektinhallinnan mallien välillä, löytää hyvän vaatimusmäärittelyn piirteet ja selvittää mitä eroja vaatimusmäärittelyssä on projektinhallinnan menetelmissä perinteisien ja ketterien menetelmien välillä. Tutkimusmenetelmänä oli teoreettinen tutkimus.

Työn teoriaosassa käydään lävitse vaatimusmäärittelyä osa-alueittain ja ohjelmistoprojektinhallinta.
Työn keskeisenä osana oli listata eroja vaatimusmäärittelyssä eri projektinhallinnan malleissa.

Keskeinen yhteenveto, vaatimusmäärittelyn on perinteisissä projektimalleissa vaikea palata, jos vaatimuksia tulee lisää. Ketterissä menetelmissä tulee uusia vaatimuksia ohjelmistoprojektin vaiheissa toistuvasti ja tämän jälkeen pääsääntöisesti siihen palataan uudestaan silloin, kun määritetään lisää vaatimuksia. Ongelmana on se, että osa todellisista vaatimuksista voi tulla hyvin myöhäisessä vaiheessa ohjelmistokehitystä.

Avainsanat vaatimusmäärittely, ohjelmistoprojektinhallinta, projektimalli

Sivut 20 sivua

HÄMEEN AMMATTIKORKEAKOULU
HÄME UNIVERSITY OF APPLIED SCIENCES

Degree Programme in Business Information Technology
Hämeenlinna

Author Teppo Tuominen **Year** 2017

Subject Requirements specification on different project management models

ABSTRACT

The goal of the thesis was to look for requirement specification differences in project management models and to find the good quality of requirement specification and difference in requirement specification on the traditional development and agile development. The research method was theoretical study.

The theory part of the thesis looks for the closer requirement specification section and soft-ware project management. The main part of the thesis deals with requirement specification differences in project management models.

The conclusions of the thesis are that a change in requirement specification after it has been done on traditional development is hard to take steps back to add more requirements in requirement specification. In agile development it is part of the process to add requirements in requirement specification and there is new requirement specification where to add more requirements. The problem on agile development is that all requirements don't have specification the same time, so there is a danger that part of the real requirements are to be found out late in software development.

Keywords requirement specification, software project, project model

Pages 20 pages

SISÄLLYS

LYHENTEITÄ

IT	tietotekniikka
JHS	Julkisen hallinnon suositukset (JHS)
RUP	Rational Unified Process

1 JOHDANTO

Kiinnostuin aiheesta tehdessäni *service-desk* työtä. Kiinnostuin tästä, vaikka suoranaisesti en ollut ohjelmistoprosessissa mukana vain välillisesti. Esimerkiksi *Microsoft Office* version päivitys on oma projektinsa. Tämän kaltaisessa projektissa tulee väistämättä ongelmia, mutta onko mahdollista vähentää ongelmien määrää vaatimusmäärittelyllä asioita huomioiden? Onko tämänlaisissa projekteissa ongelman lähtökohtana huonosti toteutettu ohjelmistoprojekti ja sen vaatimusmäärittely.

Vaatimusmäärittelyä käytetään varsinkin ohjelmistonkehityksessä usein, jolloin vaikeita ja monimutkaisia järjestelmiä pystytään kuvaamaan ja esittämään kaikenlaiset mahdolliset toiminnot, joita järjestelmässä tulee olla. Toisaalta pystytään tunnistamaan ja löytämään järjestelmään tarvittavia kriittisiä kohtia. Vaatimusmäärittelyssä voi myös olla virheitä, joihin puuttuminen tulee huomioida mahdollisimman aikaisessa vaiheessa. Tutkimuksessa asiaa käsitellään toimittajan kannalta.
Työ rajataan projektinhallinnan malleihin, joita käytetään ohjelmistoprojekteissa.

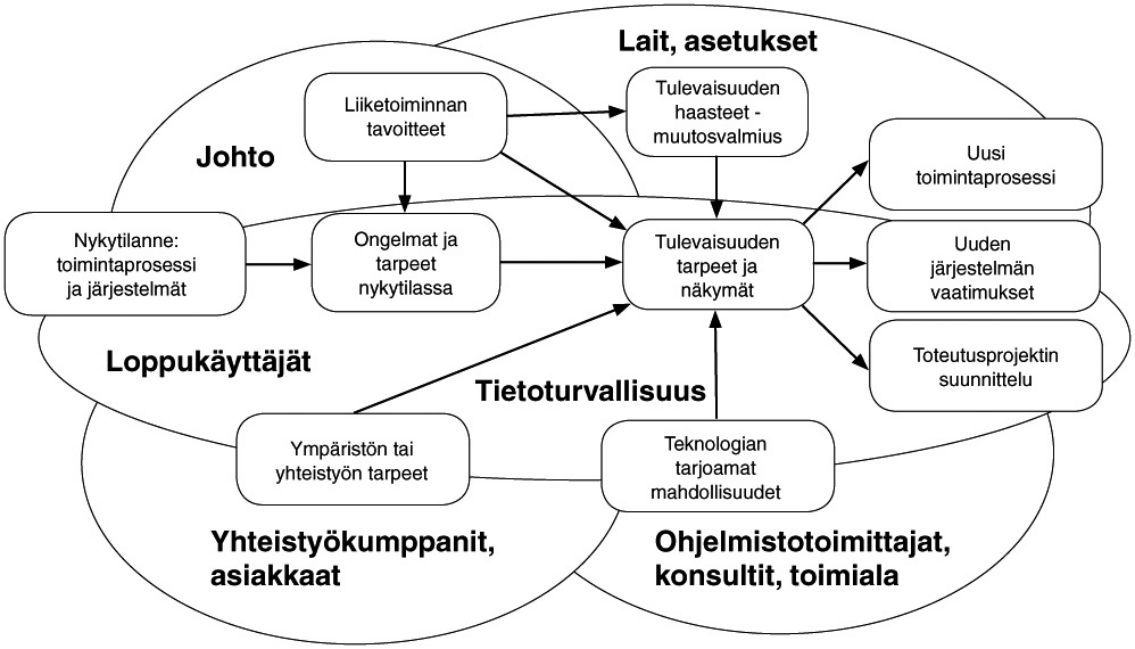
Työssäni hain vastaukset näihin tutkimuskysymyksiin:
Millainen on hyvä vaatimus?
Millainen on hyvä vaatimusmäärittely?
Millaisia eroja vaatimusmäärittelyllä on eri projektinhallinnan malleissa?
Millaisia eroja vaatimusmäärittelyssä ja sen tekemisessä on projektinhallinnan menetelmissä perinteisien ja ketterien menetelmien välillä?

2 VAATIMUSMÄÄRITTELY

“The hardest single part of building a software system is deciding precisely what to build. No other part of the conceptual work is as difficult as establishing the detailed technical requirements, including all the interfaces to people, to machines, and to other software systems. No other part of the work so cripples the resulting system if done wrong. No other part is more difficult to rectify later” (Brooks 1987, 13).

Vaatimukset tulevat tarpeista liiketoiminnassa, ja tärkeä lähde ovat asiakas ja kyseisen ohjelman tulevat käyttäjät (Haikala & Mikkonen 2011, 65).

Julkisen puolen tietojärjestelmän hankinnasta päättäessä suorittavalla organisaatiolla on oltava selkeä näkemys siitä, mihin järjestelmää tarvitaan. Näkemys asiaan on esimerkiksi voinut syntyä esiselvitystä tehdessä. Useasti alkuvaiheessa löydetyt tarpeet eivät ole tarpeeksi selkeitä, jotta niitä voisi kerätä vaatimuksiksi. Kehityskohteiden tunnistamisvaiheessa löydetyt tarpeet ja esiselvitysvaiheessa näistä tarkentuneet korkean tason vaatimukset eli käyttäjävaatimukset ovat hyvä lähtökohta varsinaiselle vaatimuksien määrittelylle. (JHS 2012b, 11.)



Kuva 1. Tarpeet ja vaatimukset voivat tulla useilta eri alueilta JHS (2012b, 12).

Vaatimuksia on tavallisesti paljon ja eri sidosryhmien asettamat vaatimukset voivat olla keskenään ristiriidassa(kuva 1) (Pelin 2011, 198).

Perusominaisuuksia hyvälle vaatimukselle ovat virheettömyys ja selkeys. Tarkka ja ymmärrettävä riittävä tarkkuus kertoo sen, että vaatimuksen saavuttaminen voidaan mitata, mutta tarkka on usein ristiriidassa vaatimuksen ymmärrettävyyden kanssa. Testattava on pystyttävä tarkistamaan, onko vaatimus täytetty. Sekä taaksepäin jäljitettävä on pystyttävä selvittämään, mistä vaatimus on peräisin. Eteenpäin jäljitettävä

on voitava selvittää, mikä on vaatimuksen tekninen toteutus ja mitkä testitapaukset testaavat vaatimuksen toteutumisen. (Haikala & Mikkonen 2011, 64.)

Varmista että, jokainen vaatimus täyttää hyvän vaatimuksen ehdot. Samoin varmista se, että jokainen vaatimus täyttää hyvän vaatimuksen kriteerit. Tämä toimenpide vie paljon aikaa. Jokainen kriteeri on tärkeä, niiden voisi sanoa olevan vaikuttavia ja korvaamattomia. Mikäli on ongelmia ymmärtää syytä yhdelle tai useammalle kriteerille, tulisi käyttää aikaa kyseisten kriteerin tutkimiseen, jotta voi tulla siihen tulokseen, että jokainen kriteeri on välttämätön. (Young, 2003, 78-79.)

Vaatimukset ovat tärkeitä, koska ne luovat pohjaa kaikelle kehitystyölle, joka tulee tämän jälkeen. Kun vaatimukset on saatu määriteltyä, kehittäjät pääsevät aloittamaan teknisen työn. Kunnolla hoidettu vaatimustenkäsittely on osatekijä onnistuneessa ohjelmistoprojektissa. (Haikala & Mikkonen 2011, 61.)

Tietojärjestelmän vaatimuksien määrittely ja laadukas organisointi ovat onnistuneen tietojärjestelmähankkeen perusedellytys. Vaatimuksien määrittely on haastavaa, mutta se säästää projektin kuluissa, nopeuttaa hankkeen toteutusta ja varmistaa haluttujen ominaisuuksien toteuttamisen. Vaatimuksilla kerrotaan hankkeen toteuttajalle, millaisia ratkaisuja ollaan hankimassa. Vaatimuksien määrittely luo kuvaa siitä, mitä ollaan hankimassa ja määrittelee, millaisia tarpeita kyseiselle tietojärjestelmälle on. (JHS 2012b, 13.)

Monet asiakkaat ja projektipäälliköt uskovat varsinainen ohjelmointityön kertovan, että edistymistä tapahtuu. Alan kokemuksen *industry experience* mukaan, liian vähän aikaa ja pyrkimystä käytetään vaatimuksiin liittyvien asioiden tekemiseen systeemityössä. (Young, 2003, 2.)

Alan kokemuksen mukaan parempi lähestymistapa on käyttää enemmän aikaa vaatimuksien keräämiseen, analysoimiseen ja hallinnan tehtäviin. Siihen on tyypillisesti syynä se, että ohjelmointityö aloitetaan liian aikaisin ja tarvitaan lisäaikaa varsinaisten vaatimuksien selvittämiseksi sekä vaatimuksiin liittyvän toiminnan suunnitteluun. Yleensä eroa on ilmoitettujen ja todellisten vaatimusten välillä. Ilmoitetut vaatimukset ovat niitä, mitkä on saatu asiakkaalta aloittaessa järjestelmän tai ohjelmiston kehitystyö. Varsinaiset vaatimukset taas ovat niitä, jotka kuvaavat todennettuja vaatimuksia, joita käyttäjällä on kyseiselle järjestelmälle tai ominaisuudelle. (Young, 2003, 2.)

Tarvitaan tavoitteellista pyrkimystä, jotta löydetään ne todelliset vaatimukset, jotka on jätetty mainitsematta. Työskentele asiakkaan ja loppukäyttäjien kanssa löytääksesi todelliset vaatimukset. Tämä vaatii perinpohjaista ymmärrystä asiakkaan tarpeista, tarkkaavaisuutta järjestelmävaatimuksille ja perinpohjaista analyysiä. (Young, 2003, 79.)

Riittämätön vaatimuksien määrittely on yleisin suoranainen syy ohjelmistoprojektin epäonnistumiseen. Joidenkin tutkimusten mukaan vaatimuksien määrittelyssä on ollut puutteita yli 75 prosentissa epäonnistuneissa projekteissa. Syitä miksi vaatimuksen määrittely on haasteita. Vaatimuksien kerääjät ja käyttäjät eivät aina ymmärrä toisiaan täysin, tilaaja on yleensä eri kuin varsinainen loppukäyttäjän ja tilaajalla oleva käsitys

saattaa poiketa huomattavasti todellisen loppukäyttäjän vaatimuksista. Menetelmät vaatimuksien keräämiseksi ja dokumentointiin voivat olla puutteellisia ja jos määrittelyä ei toteuteta projektina, käytettävät resurssit saattavat olla alakanttiin. (JHS 2009, 7-8.)

2.1 **Vaatimusten tasot**

Vaatimukset on organisoitu neljälle eri tasolle. Tavoitteet, prosessi, tehtävä ja informaatio. Tavoitetason vaatimukset asettavat liiketoiminnallisesti tavoitteet systeemille, jota ollaan rakentamassa. Prosessitason vaatimuksista kertovat liiketoiminnan aktiviteetit ja työkalut, jotka ovat johdannaisia tavoite tason vaatimuksista. Tehtävätason vaatimukset kertovat vaiheista, jolla saavutetaan ne aktiviteetit, jotka on kerrottu prosessitason vaatimuksista. (Browne & Rogich 2001, 235.)

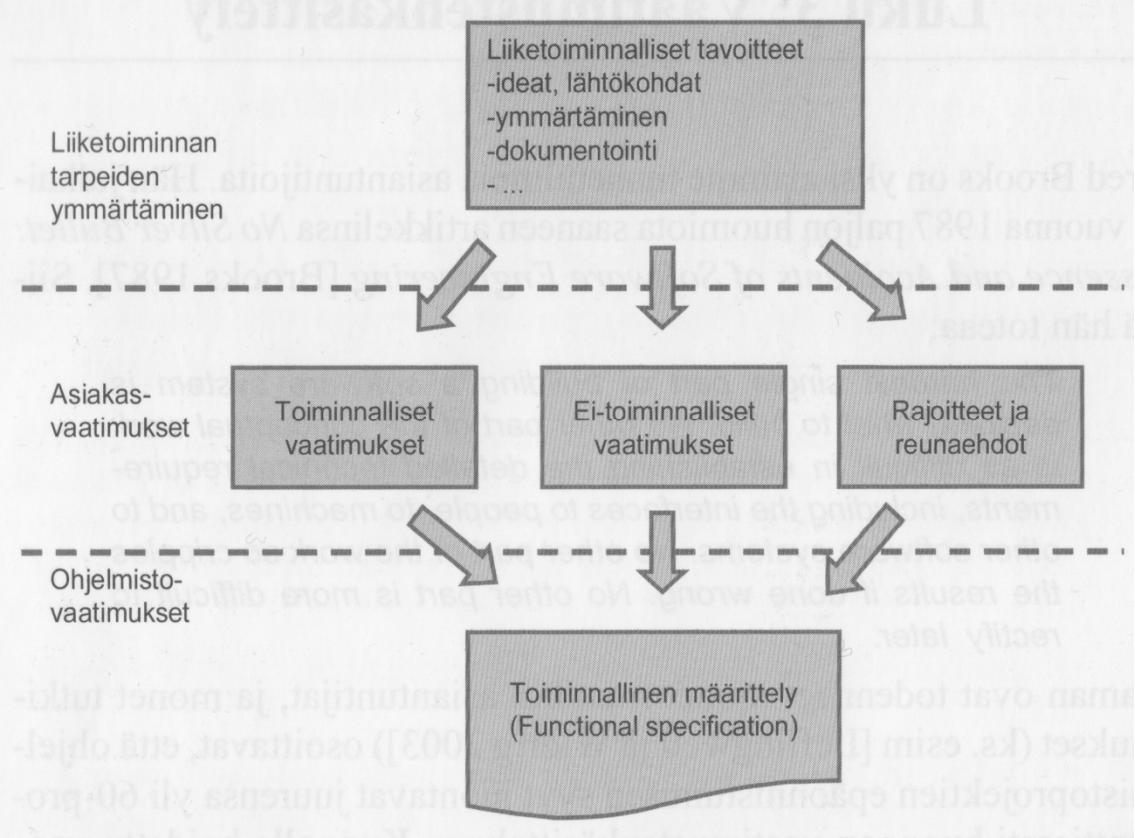
Ylimmän tason kehitysvaatimusten, joita kutsutaan yleensä suunnitteluvaatimuksiksi, pitää olla selvästi ymmärretty ennen suunnittelua. Alemmalla tasolla niistä kehitysvaatimuksista, jotka vastaavat tuloksen vaatimuksista, on yleensä käytetty nimitystä sisäänrakennetut vaatimukset. Vaatimukset, jotka ovat yli keskilinjan näiden kahden ääripään väliltä ja vastaavat prosessivaatimuksia, ovat toisin ilmaistuna työt ja suunnitelmat. Prosessivaatimukset ovat osa ohjelmoinnin suunnittelua ja proseduuri. Prosessivaatimukset on taltioitu ohjelmistovaatimuksiin, lähtökohtana on perimmäinen vaatimus eli asiakkaan tarve, jota ollaan vaatimuksilla täyttämässä.
Tarvitaan tehokkaita menetelmiä, joilla tarvevaatimuksia laajennetaan ja saadaan jalostettua pidemmälle. Näin ymmärretään asiakkaan tarpeet sekä saadaan tarkemmat suorituskyvyt vaatimukset. (Young, 2003, 48.)

Ohjelmistokehityksen voi ajatella olevan yksikertaisesti kuvaus hyvin korkealta abstraktilta tasolta täsmälliseksi ja yksityiskohtaisen tarkaksi kuvaukseksi. Esimerkiksi ohjelmistoprojektitapauksessa koko hankkeen liiketoiminnallinen tavoite voi olla asiakirjojen laadun parantaminen. Kyseistä vaatimusta analysoidessa voidaan tulla siihen tulokseen, että asetettu tavoite saavutetaan parhaimmillaan siten, että vaatimuksena on tuki oikeinkirjoituksen tarkistamiselle. Tällaista vaatimusta sanotaan asiakasvaatimukseksi, koska vaatimus tulee suoraan asiakkaan tarpeesta. Tämän kaltaisen tavoitteen kuvaamiseen ei tarvita erillistä sanastoa. (Haikala & Mikkonen 2011, 62.)

Sidosryhmiä ovat projektista hyötyjät, sekä kaikki ne, jotka mitenkään ovat tekemisissä systeemin kanssa. Mahdollisia sidosryhmiä ovat asiakas, joka maksaa työstä, käyttäjät jotka käyttävät kyseistä systeemiä, neuvonantaja ja projektiryhmät, jotka ovat osallisena ohjelmiston kehityksessä. Aina on olemassa enemmän sidosryhmiä kuin mitä alun perin on odotettu. (Young, 2003, 65.)

2.2 **Asiakasvaatimukset**

Vaatimukset luokitellaan yleensä kolmeen luokkaan (kuva 2). Ohjelmistovaatimukset on luokiteltu: toiminnalliset vaatimukset, ei-toiminnalliset vaatimukset ja rajoitteet ja reunaehdot. (Haikala & Mikkonen 2011, 61.)



Kuva 2. Asiakas- ja ohjelmistovaatimukset (Haikala & Mikkonen 2011, 62).

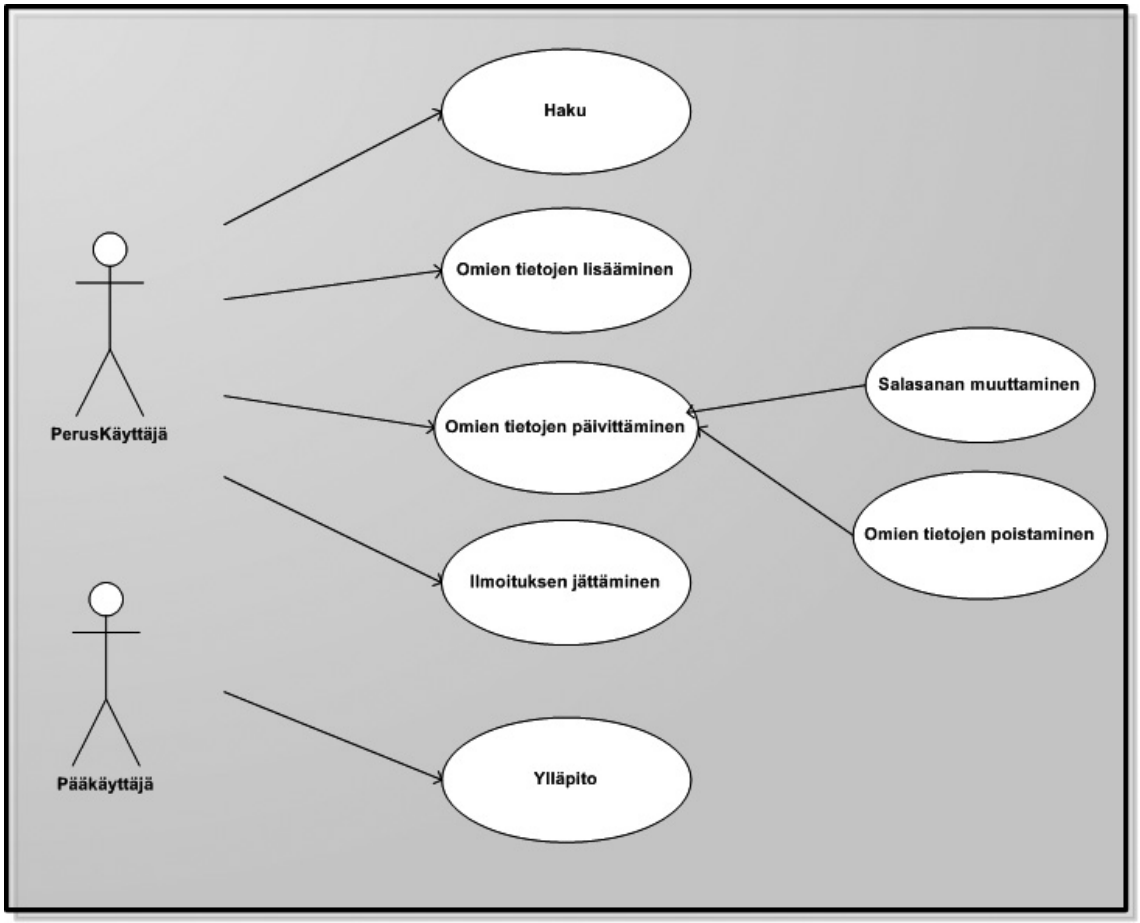
2.2.1 Toiminnalliset vaatimukset

Toiminnalliset vaatimukset kertovat ne toiminnot, jotka systeemin pitää voida toteuttaa (Young, 2003, 46). Toiminnalliset vaatimukset on tärkeä kategoria todellisille vaatimuksille. Toiminnallisia vaatimuksia kutsutaan joskus myös käytös- tai toimintavaatimuksiksi, koska ne täsmentävät syötön systeemille ja systeemin vastauksen kyseiseen syöttöön sekä näiden toiminnallisen käyttäytymisen suhteen näiden kahden välillä. (Young, 2003, 51.) Toiminnalliset vaatimukset kuvataan ensisijaisesti toimintaprosesseina kokonaisuuksien hahmottamiseksi (JHS 2012b, 12).

Prosessikuvauksissa on hyvä esittää järjestelmään käyttäjärooleja. Kuvataan lyhyesti käyttäjärooli käyttötilanteessa, jolla kuvataan käyttäjää esimerkiksi käynnistämässä tapahtumaa tai vaikuttamassa käytössä olevaan tilanteeseen esimerkiksi poistumalla kesken tapahtuman. Käyttäjärooli voi olla käyttäjä tai toinen tietojärjestelmä. (JHS (2012b, 25.)

Käyttötapauksina toimintoja hahmotellaan siten, että keskeiset järjestelmän käyttötilanteet kuvataan. Esimerkkinä olevassa (kuva 3.) järjestelmän pääkäyttäjän tehtävänä olisi järjestelmän ylläpito. (JHS 2012b, 13.)

Prosessikuvauksista on esiselvitys- tai vaatimusmäärittelyvaiheessa etsittävä kaikki mahdolliset järjestelmän ominaisuudet, joiden pohjalta tehdään järjestelmän toiminnalliset vaatimukset. Nämä vaatimukset kuvataan, dokumentoidaan, kommunikoidaan ja hallitaan käyttötapausten avulla. Käyttötapaukset koostuvat tekstinä olevista käyttötapauskuvauksista ja käyttötapauskaavioista. Prosessimalli kuvaa toimintaa, mutta käyttötapausmalli kuvaa käyttäjän ja järjestelmän vuorovaikutusta (kuva 3). (JHS 2012b, 26.)



Kuva 3. Käyttötapausmalli

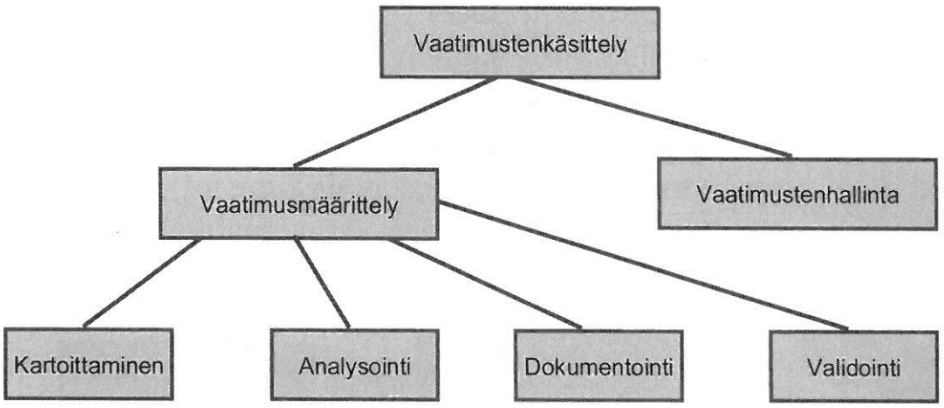
2.2.2 Tietojärjestelmän ei-toiminnallisia vaatimuksia

Tietojärjestelmän ei-toiminnalliset vaatimukset ovat tarpeellisia suunnittelun- ja toteutustyön haastavuuden arvioimiseksi. Laatuvaatimuksia on laadittu julkisen puolen useille eri alueille ja niihin on hyvä perehtyä määrittelyä tehdessä. (JHS 2012b, 24.) Ei-toiminnalliset vaatimukset täsmentävät systeemin ominaisuuksia, kuten luotettavuutta tai turvallisuutta (Young, 2003, 46).

2.2.3 Rajoitteet ja reunaehdot

Rajoitteet ja reunaehdot liittyvät käyttäjien tarpeisiin, esimerkiksi käytettävät työasemat, kämmentietokoneet ja päätteet voivat asettaa rajoituksia. Tietohallinnon näkökulmassa tuotettu palvelinympäristö, tietokantajärjestelmän jne. voi tuoda rajoituksia ohjelmistonkehitysprojektissa.(JHS 2012b, 24.) Reunaehto voi olla voisi esimerkiksi se, että ohjelmisto on toteutettava Windows-ympäristöön (Haikala & Mikkonen 2011, 62).

2.3 **Vaatimusmäärittely osa-alueet**



Kuva 4. Vaatimuskäsittelyn osa-alueet (Haikala & Mikkonen 2011, 65).

Vaikeudet, jota tavataan tietojärjestelmän kehityksessä ovat hyvin tunnettuja. Huolimatta hyvästä tahdosta organisaation, analyytikon ja käyttäjien taholta, suurin osa järjestelmistä on hylätty ennen valmistumistaan tai ne eivät savuta järjestelmälle asetettuja käyttäjävaatimuksia. (Browne & Rogich 2001, 224.) Viime aikoina on yleistynyt tehtävänhallinta ohjelmiston käyttö vaatimuksien kirjaamisessa. (Haikala & Mikkonen 2011, 67)

2.3.1 Kartoittaminen

Yleisesti vaatimuksien kartoittamiseen käytettyjä tapoja ovat tulevan ohjelmiston käyttäjien haastattelu, aivoriihet ja työpajat. (Haikala & Mikkonen 2011, 66)

Työpajat ovat ohjattua kokouksia, jossa tarkasti valittu ryhmä sidosryhmiä ja aiheen asiantuntijoita työskentelevät yhdessä, määrittelevät, luovat, jalostavat ja aikaansaavat päätökset toivotuista tuloksista, esimerkiksi mallit ja dokumentit, jotka edustavat asiakasvaatimuksia. Hyötyjä työpajaprosessista ovat sen edistävä vaikutus tiimin kommunikaatioon, päätöksentekoon ja yhteiseen ymmärrykseen. Työpajat ovat tehokas keino tuoda yhteen asiakas, käyttäjät ja ohjelmiston toimittaja. (Young, 2003, 112.)

Vaatimukset, jotka on saatu käyttäjiltä ovat yleensä epäselviä ja eivät tarjoa selvää tietoa järjestelmästä, jota ollaan kehittämässä (Würfel, Lutz & Diehl 2016).

2.3.2 Analysointi

Vaatimusanalyysissä voidaan tarkentaa vaatimuksia sekä selvitellä niiden keskinäisiä suhteita ja prioriteettia. (Haikala & Mikkonen 2011, 66).

2.3.3 Dokumentointi

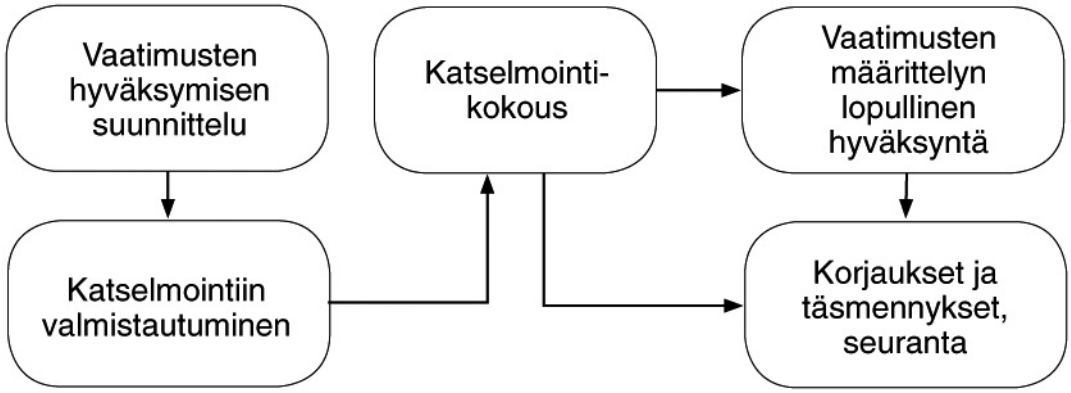
Vaatimukset dokumentoidaan sovitulla tavalla dokumenttiin tai yleisesti Excel-taulukkoon. (Haikala & Mikkonen 2011, 66).

2.3.4 Validointi

Vaatimuksien validointi tehdään yleensä katselmoimalla vaatimusmäärittelydokumentti yhdessä asiakkaan kanssa (Haikala & Mikkonen 2011, 66).

Vaatimuksien hyväksymisessä käytetään apuna katselmointia (kuva 5). Katselmointitilaisuuden vetävän puheenjohtajan olisi hyvä olla ulkopuolinen henkilö. Tämän henkilön ei tarvitse olla sisällöntuntija. Puheenjohtajan tehtävänä on huolehtia siitä, että tilaisuus etenee jouhevasti. Sekä siitä, ettei tilaisuudessa aleta ratkoa ohjelmistoprojektin ongelmia ja virheitä. Katselmuksesta on monenlaisia hyötyjä.

Ohjelmistoprojektin kannalta katselmointi auttaa projektin etenemisessä, sekä siinä, että mahdollisia vaatimuksia käydään lävitse asiakkaan kanssa ja ulkoisessa laadunvarmistuksessa. Merkittävä asia on, että siihen mennessä tehty työ vastaa asiakkaan näkemystä projektin kulusta ja tarpeista. Hyvin järjestetyssä vaatimuksien katselmuksessa osataan hyödyntää asiakkaan osaamista, jotta löydetään virheellisiä vaatimuksia ja keinoja näiden korjaamiseksi. Tällöin saadaan asiakkaan hyväksyntä tehdylle työlle ja mahdollisille muutoksille. Tämän tulisi myös antaa asiakkaalle selkeäkuva projektin kulusta. Asiakkaan, loppukäyttäjän ja sidosryhmien näkökulmat tuovat selviä hyötyjä ja sen vuoksi katselmointitilaisuuteen olisi hyvä saada mahdollisimman laaja osanotto asiakas- ja sidosryhmiltä. (JHS 2012b, 15.)



Kuva 5. Vaatimusten katselmointi (JHS 2012b, 15.)

Vaatimusmäärittelyn tuloksena on yleensä dokumentti toiminnallinen määrittely *functional specification*. Se sisältää tavallisesti sekä asiakas- että ohjelmistovaatimukset. (Haikala & Mikkonen 2011, 68.)

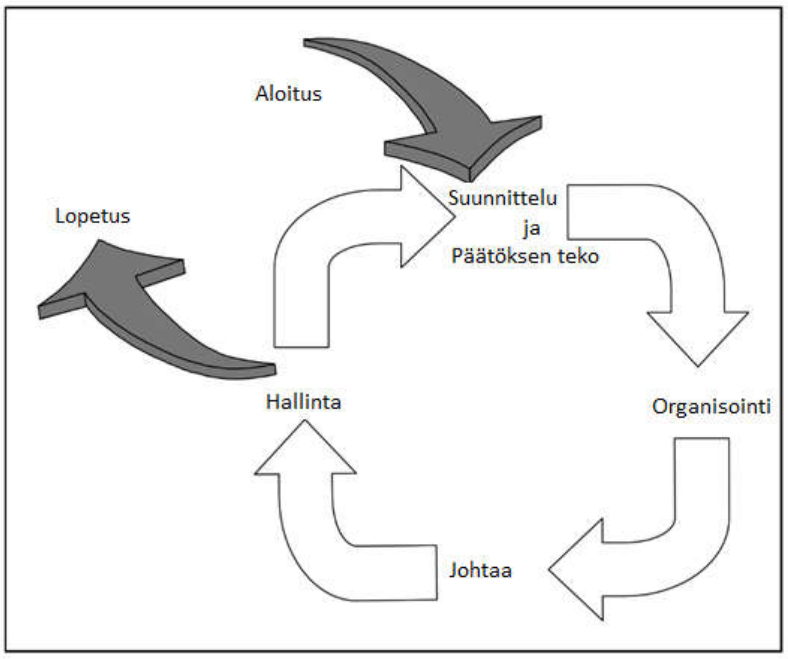
3 OHJELMISTOPROJEKTINHALLINTA

Projekti määritellään yleensä kertaluonteiseksi tehtäväksi, jolla on suunniteltu alku ja loppu. Projekti voi olla itsenäinen tai yksi osatekijä hankkeessa. Projekti toteutetaan vaiheittain. Näihin vaiheisiin kuuluu aloitusvaihe ja yksi tai useampi välivaihe sekä lopetusvaihe (kuva 7). (Cagley & Chemuturi 2009, 4.)

Projektinhallinnan periaatteet ovat toimialariippumattomia. (Haikala & Mikkonen 2011, 153). Onnistunut suunnittelu, toteuttaminen, osittaminen, aikataulu ja riskianalyysi ovat yleisesti käytetyt työkalut. Työkalut ovat samoja, mutta projektit erilaisia. Erilaiset projektit vaativat erilaisia teknisiä ja hallintalähestymistapoja. Perinteiset projektinhallinnat työkalut ja tekniikat toimivat useimmiten vähemmän tehokkaasti it:n puolella. Perinteistä projektinhallinnan mallia voisi käyttää, mutta niissä ei ole otettu huomioon it-projektien ainutlaatuisia ominaisuuksia. (Taylor 2003, 38.)

Projektinhallinta on erikoistunut lähestyminen liiketoiminnan hallintaan. Perinteinen näkökulma liiketoiminnan hallintaan on suunnittelu, organisointi, johtaminen ja hallinta. Projektinhallinta sisältää nämä asiat sekä projektin aloittamisen ja lopetuksen. Projektinhallinta voidaan määritellä osaamiseksi ja tieteeksi, joka määrittelyssä aikataulussa, mahdollisesti määritetyllä budjetilla vastaa asiakkaan suoritusvaatimuksiin käytössä olevilla resursseilla. Projektinhallinta on osaamista, joka käsittää kanssakäymistä monimuotoisten ryhmien välillä. Tämä osa projektinhallintaa tekee siitä haastavaa. (Taylor 2003, 13.)

Yleisesti ottaen isompien projektien ongelmat liittyvät ihmisten ongelmien ratkaisemiseen, tiimityöskentelyyn ja neuvottelemiseen. Teknilliset ongelmat ovat paljon helpompia ratkaista, kuin ihmisiin liittyvät ongelmat. Tekniset työkalut tekevät projektinhallinasta osittain tiedettä. (Taylor 2003, 13.)



Kuva 6. Ohjelmistoprojektinhallinta (Taylor 2003, 15).

It-projektin suunnittelussa ja hallinnassa on tärkeä ymmärtää projektinhallinnan elinkaari (kuva 6). Elinkaaren vaiheet sovittavat projektin tavoitteet yhteen, jotta ohjelmiston vaatimukset toteutuvat. (Taylor 2003, 39.)

Ohjelmistojen tuotantoa varten on olemassa rajaton määrä monenlaisia lähestymistapoja (Haikala & Mikkonen 2011, 29.) Vuosien varrella on kehittynyt lukuisia julkisesti saatavilla olevia projektimalleja, joista osa täysin yleisiä ja osa nimenomaan it-projekteihin tarkoitettuja. (Haikala & Mikkonen 2011, 34.)

Ohjelmistoprojektin toteutuksessa on kaksi osatekijää, nimellisesti systeemityö ja ohjelmistoprojektinhallinta. Systeemityö pitää sisällään kaikki tietotekniikkaan kuuluvat aktiviteetit, joita tehdään projektin toteuttamiseksi. Ohjelmistotekniikka käsittelee komponenttien rakentamista, niiden integrointia, verifiointia, validointia sekä lopuksi kaikkien komponenttien yhdistämistä lopputulokseen valmiiksi tuotteeksi. Lisäksi se vakuuttaa siihen, että asiakas hyväksyy lopputuloksena olevan ohjelman toimituksen. Ohjelmistoprojektinhallinta varmistaa sen, että projektin toimitettavat osat valmistuvat ajallaan, tehokkaasti ja ilman virheitä. (Cagley & Chemuturi 2009, 19.)

On olemassa kahdenlaista koulukuntaa liittyen kytkökseen systeemityön ja johtamismenetelmien välillä: vahvasti kytköksissä ja löyhästi kytköksissä. (Cagley & Chemuturi 2009, 19.)

Vahvasti kytköksissä olevassa ajatuksena on se, että molemmat menetelmäopilliset tavat ovat vahvasti kytköksissä ja johtaminen on riippuvainen systeemityön metodologiasta, jota käytetään projektin toimitettavien osien rakentamiseen. Tämän takia johtamisen pitää olla tiivisti yhteydessä systeemityön kanssa. (Cagley & Chemuturi 2009, 19.)

Löyhästi kytköksissä olevassa mallissa nämä kaksi systeemityön näkökulmaa ovat löyhästi kytköksissä, mutta vaikuttavat kuitenkin toisiinsa.
Tämän ajatustavan projektinhallinnalla on monia tavoitteita. Päätavoitteena on rakentaa toimitettava tuote. Muita tavoitteita ovat projektiaikataulun hallinta, tuottavuus, laatu, moraali, asiakas ja tuotto. Löyhästi kytköksissä oleva koulukunta ajattelee, että ohjelmistoprojektin hallinnassa johtaja on ensisijaisena ja tietoinen systeemityömenetelmä toissijaisena. (Cagley&Chemuturi 2009, 19.)

Kaiken a ja o projektinhallinnan metodologiassa systeemityön metodologia on monen eri tekijöiden summa, kuten mikä on organisaation koko ja mitä projektihallinnan mallia käytetään tietyssä projektissa. (Cagley&Chemuturi 2009, 19.)

It-projektit ovat luoneet lisää projektinhallinnan haasteita. Vaikeuksia lisää it-projektien monimutkaisuus, melkein jokaisessa it-projektissa rajoitteena on tiukka aikatauluja, sekä tarve saada it-projektin tuotos markkinoille. It:tä käytetään lähes kaikkialla. Tämä on osaltaan ratkaissut liiketoiminnan ongelmia ja tuonut uusia liiketoiminallisia mahdollisuuksia, mutta samalla luonut uusia vakavia hallinnan ongelmia. (Taylor 2003, 10.)

Ohjelmistoprojekteille on tyypillistä, että projektipäällikölle jää tarkennettavaa ja valinnanvaraa verrattain paljon (Haikala & Mikkonen 2011, 153).

Ohjelmistoprojektit voidaan jaotella niiden ominaisuuksien mukaan erilaisiin kategorioihin. Näillä pyritään luonnehtimaan ohjelmistoa periaatteisella tasolla, mikä usein auttaa ymmärtämään toteutustyön kannalta tärkeimpiä haasteita. (Haikala & Mikkonen 2011, 12.) Ohjelmistoprojektiin katsotaan tavallisemmin liittyvän ainakin määrittelyä, suunnittelua, ohjelmointia ja testausta (Haikala & Mikkonen 2011, 153).

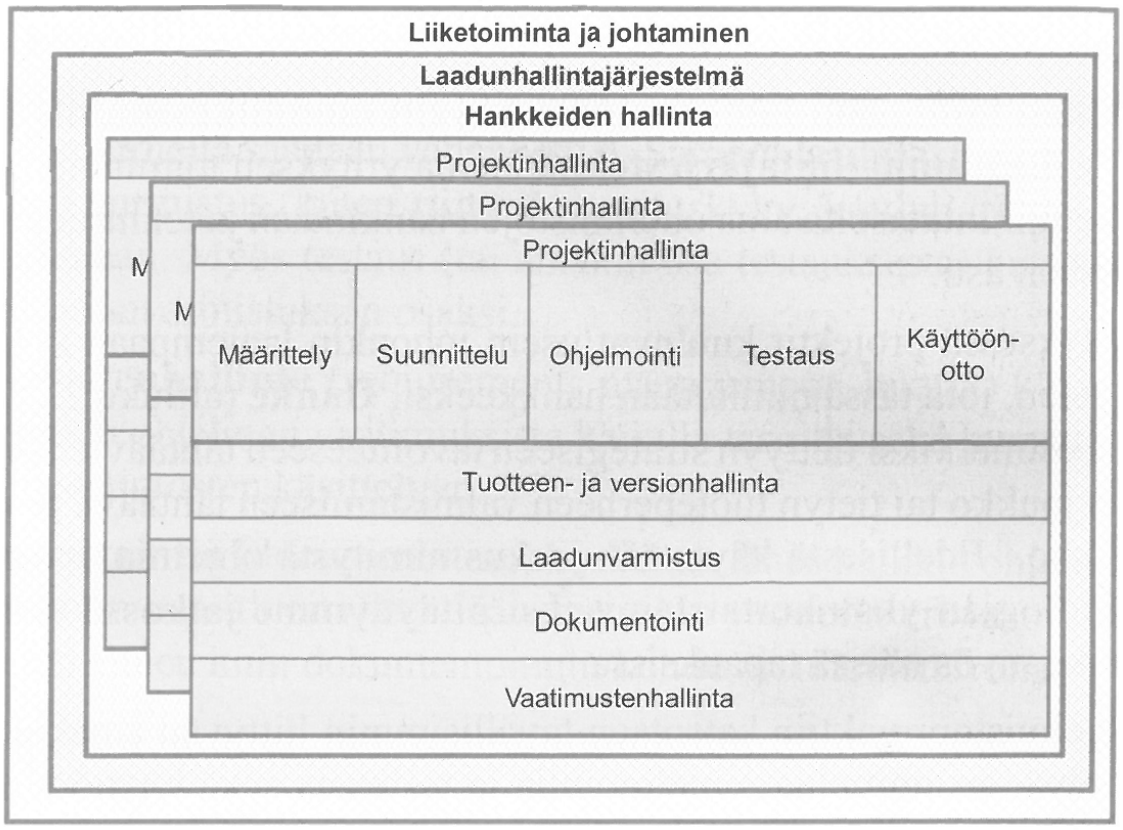
Vaatimusmäärittely voi olla oma projekti suuremmissa hankkeissa, jossa tavoitteena on saada selville järjestelmän asiakasvaatimuksia mahdollisimman perusteellisesti. Monimutkainen ympäristö, jossa ohjelmisto-organisaatiot toimivat aiheuttaa sen, että suurin osa ohjelmistoprojekteista ei valmistukkaan. Sen mukaan mitä toivotaan määrittelyn, suunnitellun budjetin ja sovitun aikataulun suhteen. Minkä lisäksi tulee pitää tulevan järjestelmän käyttäjät ja sidosryhmät tyytyväisinä. (Luna-Reyes, Zhang, Gil-Garcia & Cresswell 2005.)

Ohjelmistoprojektin erikoisuuksiin kuuluu, että lopputulos ei ole aineellista. Tuloksena on toimiva ohjelma ja todellista komponentteja ei toimiteta. Melkein kaikki lopputuloksesta on tietokoneen sisällä. (Cagley & Chemuturi 2009, 4.)

Ohjelmistoprojektissa henkilön edistymistä voidaan mitata toimivan ohjelmiston tai kirjoitetun ohjelmistokoodin mittapuulla. Ohjelmistontuotannossa visuaalinen arviointi ei riitä varmistamaan, että henkilö tekee oman osansa. Huolimatta edistyksestä ohjelmistotekniikan- ja kaavio-työkaluissa. Ne eivät ole nostaneet tarkkuutta, jolla päästäisiin samaan tarkkuuteen, johon päästään muilla tekniikan tieteenaloilla. (Cagley & Chemuturi 2009, 4.)

4 VAATIMUSMÄÄRITTELY ERI PROJEKTINHALLINNAN MALLEISSA

Vuosien varrella on kehittynyt monia julkisesti saatavilla olevia projektimalleja, joista osa on täysin yleisiä ja osa nimenomaan it-projekteihin tarkoitettuja. (Haikala & Mikkonen 2011, 34.) Eri projektimallit (kuva 8) poikkeavat yleensä, miten alla olevan kuvassa 7 on osa-alueita sovelletaan projektivaiheissa. (Haikala & Mikkonen 2011, 29)



Kuva 7. Projektinhallinan osa-alueet (Haikala & Mikkonen 2011, 29.)

4.1 Perinteiset projektimallit

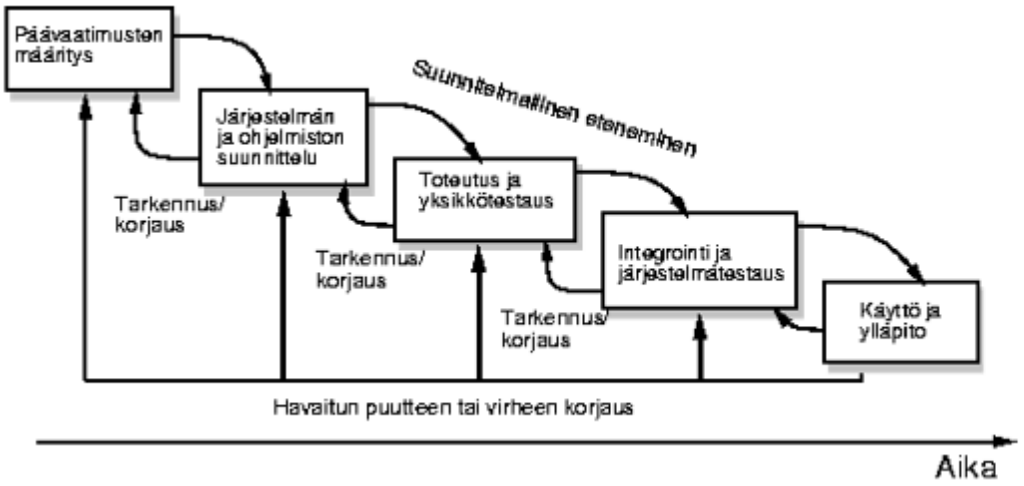
Suomessa käytettään varsin yleisesti IPMA-, PRINCE2-, PMI-pohjaisia projektimalleja (Haikala & Mikkonen 2011, 34). Vähänkin monimutkaisempaa ohjelmaa kehittäessä ohjelmistokehitykseen kuuluu muuta, kuin itse ohjelmointi (kuva 8) (Haikala & Mikkonen 2011, 29).

4.1.1 Vesiputousmalli

Ensimmäisiä suuria tietokoneohjelmia tehdessä, havaittiin tarvetta järjestelmälliselle työprosessille. Se noudattaa tavallisesti tuttua järjestystä, jolla määritellään asiakkaan tarpeet. (Haikala & Mikkonen 2011, 36-37.)

Keskeisenä ideana vesiputousmallissa on se, että jokaisessa vaiheessa keskitytään pelkästään tämän prosessivaiheen saamiseksi loppuun. Vaiheen valmistuttua siirrytään

seuraavaan tai palataan edelliseen, mikäli aikaisempaa vaihetta on tarvetta muuttaa. Vaatimusmäärittely toteutuu (kuva 8) päävaatimuksin määritys-, järjestelmän- ja ohjelmistonsuunnittelu kohdissa. (Jyväskylän Yliopisto n.d.)

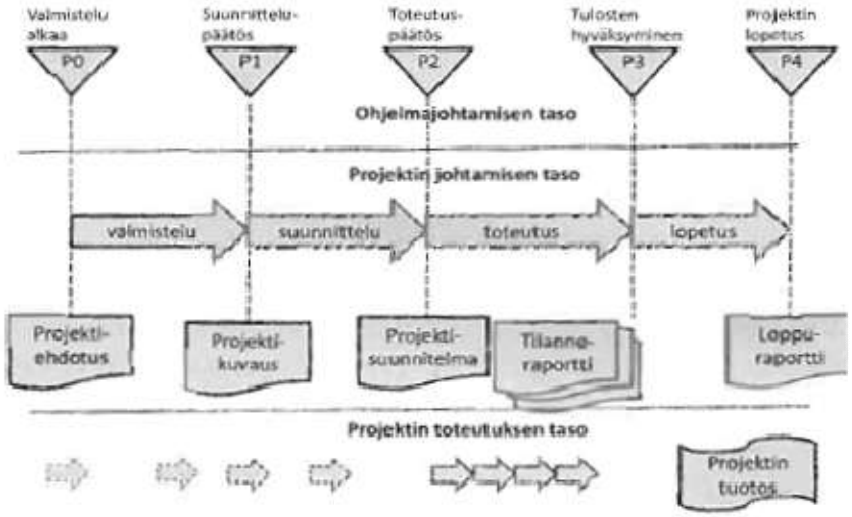


Kuva 8. Vesiputousmalli (Jyväskylän Yliopisto n.d.)

Tarkistellessa uudempia projektinhallinnan malleja, löytyy niiden sisältä jossain muodossa vesiputousmallin alkuperäinen idea. Teollisuudessa käytetään yleensä pelkistettyä mallia, josta on poistettu iterointi. (Haikala & Mikkonen 2011, 36-37.)

4.1.2 PMI

PMI-mallissa projektin suunnitteluvaiheessa tehdään vaatimusmäärittely, mutta projektin toteutusvaiheessa projektin vaatimusmäärittely päivittyy ja tarkentuu. Alla kuvassa 9. kohdassa P2 (Haikala & Mikkonen 2011, 36)



Kuva 9. PMI projektin kulku (Haikala & Mikkonen 2011, 35)

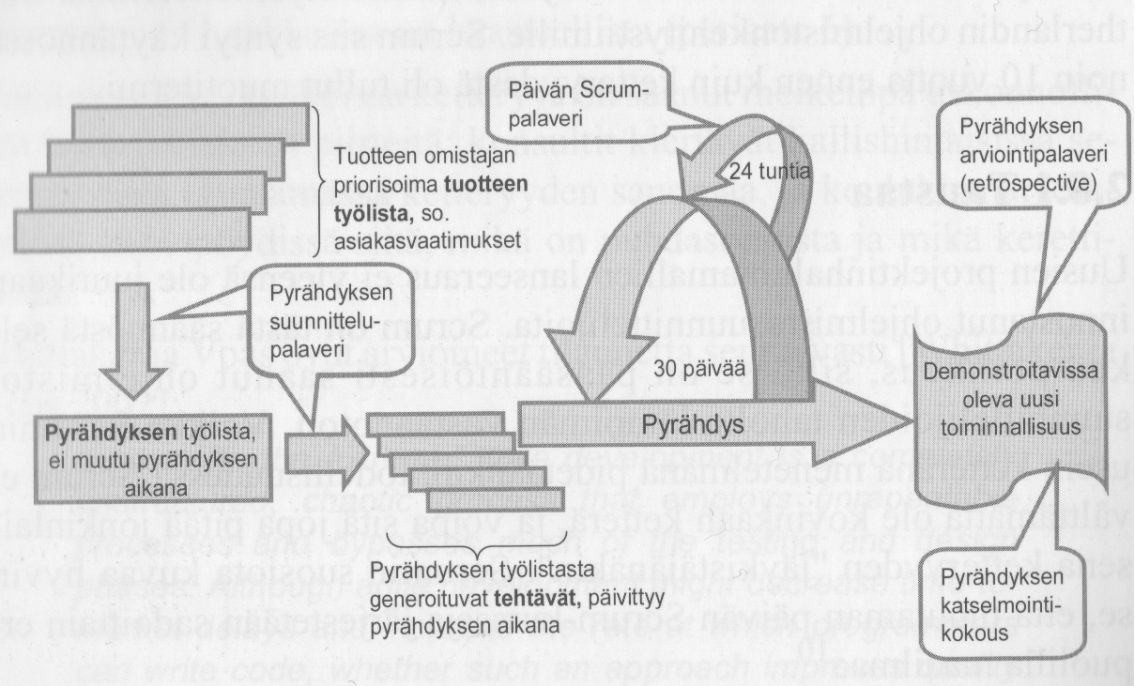
4.2 **Ketterät menetelmät**

Ohjelmistokehitystä tekeminen paremmaksi, että osallisena sitä ja kehittää ohjelmistokehitystä eteenpäin. Kokemuksen mukaan kannatta arvostaa henkilöitä ja keskustelua enemmän, kuin malleja ja menetelmiä. Valmista toimivaa ohjelmistoa enemmän kuin ohjelmiston laajaa dokumentaatiota. Yhteistyötä asiakkaan kanssa kuin sopimusneuvotteluja. Tehdä muutoksia tarpeen mukaan kuin suunnitelman tarkkaa noudattamista. Kaikissa mainituilla asioilla on arvoa, mutta ensiksi kerrottuja pidetään tärkeämpänä. (Beck &Beedle& van Bennekum& Cockburn & Cunningham & Fowler &Grenning& Highsmith & Hunt & Jeffries & Kern &Marick& Martin & Mellor 6 Schwaber& Sutherland & Thomas. 2001.)

Ketterät menetelmät poissulkien RUP kannustavat minimaaliseen vaatimuksien dokumentointiin systeemityössä. (Cagley&Chemuturi 2009, 19).

4.2.1 Scrum

Yleisin käytössä oleva ketterä menetelmä on Scrum, jopa siihen asti sanasta on tullut ketterän ohjelmistokehityksen synonyymi. Scrumin hyvinä puolina pidetään sen yksinkertaisuutta. Scrumissa on vain kolme roolia, tuotteen omistaja jonka tehtävät muistuttavat tuotepäällikköä, Scrum-mestari vastaa projektipäällikköä ja tiimi vastaa projektiryhmää. Huomionarvoista on, että projektin epäonnistumiseen johtaneet syyt liittyvät lopulta vaatimusten hallintaan. Scrum ei ota vaatimuksienhallintaan mitenkään kantaa vaan siirtää vastuun tuotteen omistajalle. Srum ottaa kantaa vain osaan ohjelmiston elinkaaren tehtävistä ja tarvitsee yhdistämisen projektinhallinnan välineistöön. (Haikala & Mikkonen 2011, 46-49.) Jopa ketterässä Scrum-mallissa (kuva 10) voidaan nähdä yhdistyminen toisiaan seuraavaksi lyhyiksi vesiputouksiksi (kuva 8) (Haikala & Mikkonen 2011, 37.)

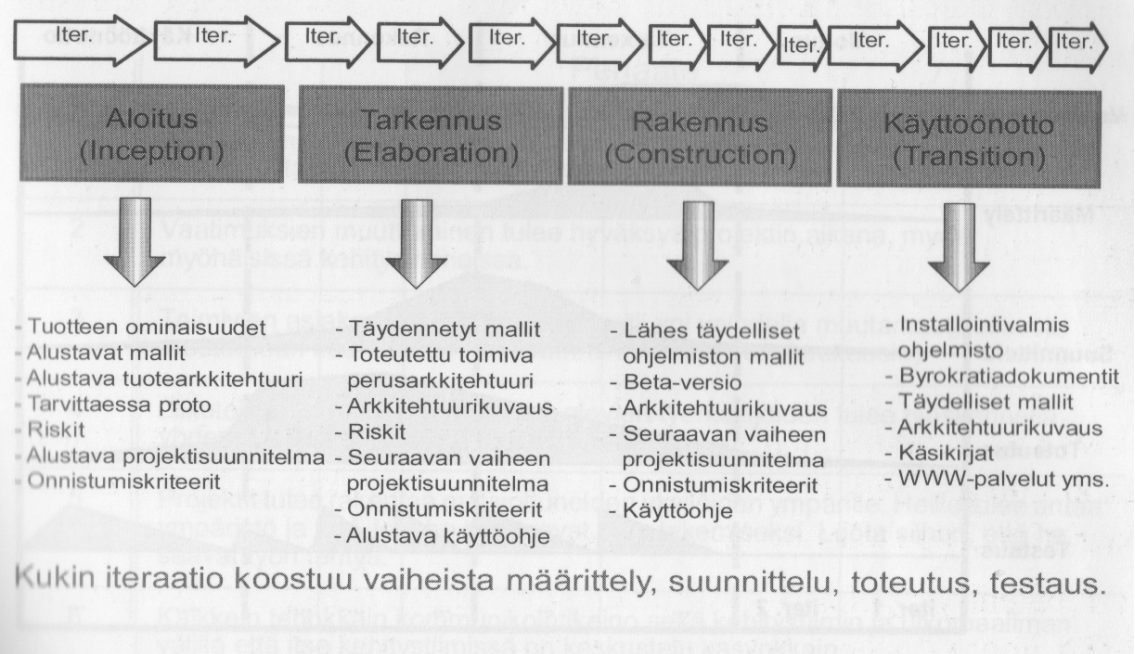


Kuva 10. Scrum-prosessi (Haikala & Mikkonen 2011, 48.)

4.2.2 RUP (*Rational Unified Process*)

On kehitetty laaja ohjelmistokehityksen prosessikehikko, joka on räätälöitävissä moneen eri tarkoitukseen. (Haikala & Mikkonen 2011, 42.) *RUP* on tarkka systeemityön menetelmä, jossa on hyvin muiden menetelmien tapaan dokumentaatio perinteisien menetelmien mukaisesti. (Cagley & Chemuturi 2009, 19.)

Tässä mallissa toteutettavat vaatimukset voidaan jakaa iteraatioihin. Hieman vesiputousmallia muistuttava vaihteet (kuva 9) tehdään tosin erilaisilla painotuksilla, mutta jokaisessa RUP:n vaiheessa (kuva 12). (Haikala & Mikkonen 2011, 42.)



Kuva 11. RUP-vaiheistus (Haikala & Mikkonen 2011, 43.)

5 YHTEENVETO

Hyvän ohjelmistoprojektin vaatimus on, että se vastaa asiakkaan tarpeisiin. Riittämätön vaatimuksien määrittely on yleisin syy ohjelmistoprojektin epäonnistumiseen. Perusominaisuutena hyvälle vaatimukselle on luonnollinen selkeys ja virheettömyys. Vaatimukset lähtevät liiketoiminnan tarpeista, ja niiden tärkeät lähteet ovat asiakas ja tulevan ohjelman käyttäjät.

Kunnollisen vaatimusmäärittelyn pohjana on selkeä näkemys siitä, mihin järjestelmää tarvitaan. Älä odota vaatimusmäärittelyn olevan koskaan aivan valmis, muuten vaatimusmäärittely ei koskaan valmistu, se ei ole koskaan ajan tasalla tai projektin lopputulos on myöhässä.

Vaatimusmäärittely eri projektinhallinnan malleissa toteutuu kaikissa käytetyissä malleissa, mutta se miten paljon aikaa vaatimusmäärittelyyn käytetään vaihtelee huomattavasti.

Perinteisellä menetelmällä vaatimukset määritellään hyvin aikaisessa vaiheessa ohjelmistonkehitystä ja tämän jälkeen pääsääntöisesti vaatimuksien määrittelyyn ei enää palata millään tavalla. Tämä luo ongelmia, jos ohjelmistoprojektin loppuvaiheessa olisi tarvetta muuttaa tai lisätä uusia vaatimuksia. Tämä johtaa helposti ohjelmiston vaatimusmäärittelyn epäonnistumiseen ja tätä kautta siihen, että projekti ei täytä sille asetettuja tavoitteita.

Ohjelmistoprojekteissa ketterissä menetelmissä vaatimusmäärittely tehdään ohjelmistoprojektin eri vaiheissa toistuvasti ja tämän jälkeen pääsääntöisesti siihen voidaan palata uudestaan. Näin uudet ja löydetyt vaatimukset voidaan myöhäisessäkin vaiheessa tuoda ilman ongelmia käyttöön. Ongelmana on se, että osa todellisista vaatimuksista voi tulla hyvin myöhäisessä vaiheessa ohjelmistokehitystä.

LÄHTEET

Beck, K. Beedle, M. van Bennekum, A. Cockburn, A. Cunningham, W. Fowler, M. Grenning, J. Highsmith, J. Hunt, A. Jeffries, R. Kern, J. Marick, B. Martin, R. Mellor, S. Schwaber, K. Sutherland, J. & Thomas, D. 2001.Ketterän ohjelmistokehityksen julistus. Haettu 1.3.2017 osoitteesta http://agilemanifesto.org/iso/fi/manifesto.html

Brooks, F. (1987). No Silver Bullet — Essence and Accidents of Software Engineering Haettu 23.1.2017 osoitteesta http://faculty.salisbury.edu/~xswang/Research/Papers/SERelated/no-silver-bullet.pdf

Browne, G. & Rogich, M. (2001). *An empirical investigation of user requirements elicitation: Comparing the effectiveness of prompting techniques*. Journal of Management Information Systems, 17(4), 223-249. Haettu 31.1.2017 osoitteesta https://search-proquest-com.ezproxy.hamk.fi/docview/218918623?accountid=27301

Cagley, T. & Chemuturi, M. (2009). Mastering Software Project Management. Haettu 13.2.2017 osoitteesta https://ebookcentral.proquest.com/lib/hamk-ebooks/detail.action?docID=3319451

Haikala. I. & Mikkonen T. (2011). *Ohjelmistotuotannon käytännöt*. Helsinki: Talentum Media Oy

JHS (2012a). JHS 172 ICT-palvelujen kehittäminen: Esiselvitys. Haettu 27.02.2017 osoitteesta http://www.jhs-suositukset.fi/suomi/jhs172

JHS (2012b). JHS 173 ICT-palvelujen kehittäminen: Vaatimusmäärittely. Haettu 27.02.2017 osoitteesta http://www.jhs-suositukset.fi/suomi/jhs173

JHS (2009) . JHS 165 ICT-palvelujen kehittäminen: Vaatimusmäärittely. Haettu 27.02.2017 osoitteesta http://www.jhs-suositukset.fi/c/document_library/get_file?uuid=b8118ad7-8ee4-459a-a12b-f56655e4ab9d&groupId=14

Jyväskylän Yliopisto n.d. TIE358 – Verkkokurssin tuotantoprosessi. Haettu 28.2.2017 osoitteesta tahttp://www.mit.jyu.fi/ope/kurssit/TIE358/sivusto/johdanto/ohjelmistoprosessi.html

Luna-Reyes, L., Zhang, J., Gil-Garcia, J.R. &Cresswell, A. (2005), Information systems development as emergent socio-technical change: a practice approach. *European Journal of Software developments*, Vol. 14 No. 1, pp. 93-105. Haettu 22.2.2017 osoitteesta https://www.researchgate.net/publication/220393211_Information_systems_development_as_emergent_socio-technical_change_A_practice_approach

Pelin, R. (2011). *Projektihallinnan käsikirja.* 7. painos. Helsinki: Projektijohtaminen oy Risto Pelin

Taylor, J. (2003). Managing Information Technology Projects. New York: AMACOM. Haettu 15.2.2017 osoitteesta https://ebookcentral.proquest.com/lib/hamk-ebooks/detail.action?docID=243057

Würfel D, Lutz R., Diehl S. (2016). Grounded requirements engineering: An approach to use case driven requirements engineering. *Journal of Systems and Software,* Vol 117, pp. 645-657. Haettu 10.2.2017 osoitteesta http://www.sciencedirect.com/science/article/pii/S0164121215002277

Young, Ralph. (2003). *Requirements Engineering Handbook*. Norwood, US: Artech House Books