
# Vaatimusmäärittelyn vaikutus ohjelmiston laatuun

Pirjo Ungert

Haaga-Helia ammattikorkeakoulu
Tietojenkäsittelyn koulutusohjelma
Opinnäytetyö
2025

## Tiivistelmä

**Tekijä(t)**
Pirjo Ungert

**Tutkinto**
Tradenomi, tietojenkäsittelyn koulutusohjelma

**Raportin/Opinnäytetyön nimi**
Vaatimusmäärittelyn vaikutus ohjelmiston laatuun

**Sivu- ja liitesivumäärä**
23 + 0

Ohjelmistokehityksen prosessi koostuu monista eri vaiheista. Tässä opinnäytetyössä tarkastellaan erityisesti vaatimusmäärittelyn vaikutusta ohjelmiston laatuun läpi projektin. Tutkimustyöhön lähdetään asettamalla kolme tutkimuskysymystä: miten vaatimusten laatu vaikuttaa ohjelmiston virheiden määrään, kuinka hyvin määritellyt vaatimukset ennustavat käyttäjätyytyväisyyttä ja miten vaatimusten muutokset vaikuttavat ohjelmiston laatuun. Perustaksi tutkimukselle luodaan aiheen osalta tärkeät tietoperustaosuudet. Tutkimus toteutetaan sisältöanalyysinä niin, että tutkitaan alan artikkeleita ja niissä esiintyviä yhtäläisyyksiä sekä esiin tulleita huomioita. Tämän opinnäytetyön sisältöanalyysin perusteella voidaan todeta, että vaatimusmäärittelyllä on laajoja vaikutuksia koko projektin ajan ohjelmiston laatuun.

**Asiasanat**
Vaatimusmäärittely, ohjelmistokehitys, ohjelmiston elinkaari, laadunvarmistus, toiminnalliset ja ei-toiminnalliset vaatimukset

## Sisällys

# 1 Johdanto

Aiheena opinnäytetyössä on ”Vaatimusmäärittelyn vaikutus ohjelmiston laatuun”. Aihe tuli opinnäytetyön laatijalle ajankohtaiseksi hänen osallistuttua työelämässä laajaan yrityksen ohjelmistokehitysprosessiin. Tästä seurauksena nousee kiinnostus yleisesti ilmiöön liittyen sekä siihen, miten vaatimusmäärittely ohjelmakehityksen prosessissa vaikuttaa yleisesti eri ohjelmistojen laatuun niin lopullisessa vaiheessa kuin myös aikaisemmissa vaiheissa, esimerkiksi eri testausvaiheissa.

Ohjelmistokehitysprosessi koostuu eri vaiheista, jotka kaikki tähtäävät toimivan ohjelmiston luontiin sekä ylläpitoon. Prosessi lähtee liikkeelle vaatimusten keräämisellä, jonka jälkeen luodaan toteutettavuusanalyysi. Näiden jälkeen aloitetaan järjestelmän suunnittelu sekä prototyyppien luominen. Edellä mainituista päästään siirtymään itse ohjelmistokehitykseen sekä testaukseen. Ohjelmiston kehityksen aikana sekä jälkeen luodaan tarvittavat järjestelmäintegraatiot. Lopuksi ohjelmisto julkaistaan, sitä ylläpidetään ja määrittelemättömän ajan jälkeen järjestelmä tulee elinkaarensa päähän. (BrowserStack 2024.)

Tutkimusta lähdetään luomaan lähtökohdasta, että hyvin määritellyt vaatimukset ovat perusedellytys ohjelmiston laadulle. Kun käyttäjävaatimukset ovat määritelty tarkasti, ohjelmisto pystyy paremmin täyttämään loppukäyttäjän tarpeet. Selkeä ja kattava määritystyö auttaa tiimejä pysymään aikarajoissa ja budjetissa. Hyvä dokumentaatio määrittelyvaiheessa voi vähentää muutostarvetta projektin edetessä. Organisaatiot, jotka pystyvät tuottamaan laadukkaita ohjelmistoja nopeammin ja vähemmillä virheillä, saavat kilpailuetua markkinoilla. Määritystyön vaikutuksen tutkiminen voi johtaa parempiin käytäntöihin ja menetelmiin, mikä voi parantaa koko ohjelmistokehitysprosessia. Se auttaa myös tunnistamaan mahdollisia tulevia riskejä aikaisessa vaiheessa. (Koski 2020, 1.)

## 1.1 Tutkimuksen tavoitteet, rajaukset sekä peittomatriisi

Tutkimuksen tavoitteena on selvittää, miten laajalti vaatimusmäärittelyn eri vaiheet ja lopputulos vaikuttavat ohjelmistojen laatuun eri vaiheissa, varsinkin kehitystyön aikana sekä sen jälkeen. Tutkimuksessa selvitetään mm. miten vaatimusmäärittelyn tarkkuus ja selkeys vaikuttavat läpi projektin. Lisäksi tarkastellaan, mikä vaikutus onnistuneella vaikutusmäärittelyllä on projektin aikarajoihin liittyvien tavoitteiden saavuttamiseen.

Tutkimuksen keskeisimmät kohdat määritellään seuraavien tutkimuskysymyksien avulla:

- Miten vaatimusten laatu (selkeys, tarkkuus, johdonmukaisuus) vaikuttaa ohjelmiston virheiden määrään?
- Kuinka hyvin määritellyt vaatimukset ennustavat käyttäjätyytyväisyyttä?
- Miten vaatimusten muutokset projektin aikana vaikuttavat ohjelmiston lopulliseen laatuun?

Tutkimusongelma suunnitellaan ratkaistavaksi seuraavien alaongelmien avulla:

1. Miten eri osapuolten (esim. asiakkaat, kehittäjät, testaajat) käsitykset vaatimuksista vaikuttavat ohjelmiston virheiden määrään?
2. Kuinka vaatimusten johdonmukaisuus eri vaiheissa vaikuttaa ohjelmiston kehityksen sujuvuuteen ja lopputulokseen?
3. Millä tavoin vaatimusten muutokset ja epäselvyydet projektin aikana vaikuttavat projektin aikarajoihin ja budjettiin?
4. Miten hyvin määritellyt vaatimukset ennustavat loppukäyttäjien tarpeita ja vaikuttavat käyttäjätyytyväisyyteen?
5. Miten vaatimusten muutokset ja virheet vaatimuksissa vaikuttavat ohjelmiston virheiden jäljittämiseen ja korjaamiseen julkaisun jälkeen?

Peittomatriisin avulla esitetään taulukkomuodossa tutkimus- ja alakysymykset sekä viittaukset edellä mainittujen kohtien tietoperustaan.

| Tutkimuskysymys | Alakysymykset | Tietoperusta |
|---|---|---|
| **Miten vaatimusten laatu (selkeys, tarkkuus, johdonmukaisuus) vaikuttaa ohjelmiston virheiden määrään?** | 1. Miten eri osapuolten (esim. asiakkaat, kehittäjät, testaajat) käsitykset vaatimuksista vaikuttavat ohjelmiston virheiden määrään?<br>2. Kuinka vaatimusten johdonmukaisuus eri vaiheissa vaikuttaa ohjelmiston kehityksen sujuvuuteen ja lopputulokseen? | **Luku 3.1:** Vaatimusmäärittelyn rooli ohjelmistokehityksessä<br>**Luku 3.2:** Vaatimusmäärittelyn prosessi<br>**Luku 4.2:** Laadunvarmistus ohjelmistokehityksessä |
| **Kuinka hyvin määritellyt vaatimukset ennustavat käyttäjätyytyväisyyttä?** | 1. Miten hyvin määritellyt vaatimukset ennustavat loppukäyttäjien tarpeita ja vaikuttavat käyttäjätyytyväisyyteen? | **Luku 3.1:** Vaatimusmäärittelyn rooli ohjelmistokehityksessä<br>**Luku 4.1:** Ohjelmiston laadun määritelmiä |

| | | |
|---|---|---|
| **Miten vaatimusten muutokset projektin aikana vaikuttavat ohjelmiston lopulliseen laatuun?** | 1. Miten vaatimusten muutokset ja virheet vaatimuksissa vaikuttavat ohjelmiston virheiden jäljittämiseen ja korjaamiseen julkaisun jälkeen?<br>2. Millä tavoin vaatimusten muutokset ja epäselvyydet projektin aikana vaikuttavat projektin aikarajoihin ja budjettiin? | **Luku 3.4:** Vaatimusmäärittelyn hallinta |

Taulukko 1: Peittomatriisi

Tutkimus rajataan käsittelemään pääasiassa ketteriä menetelmiä (mm. Agile, Scrum), sillä ne ovat keskeisiä menetelmiä ohjelmistojen vaatimusten keräämiseen sekä hallintaan (Tolvanen 19.12.2023). Tutkimuksessa ei myöskään käsitellä laajemmin muita ohjelmistokehitysprojektin vaiheita, muuten kuin tietoperustassa kokonaisuuden selkeyttämiseen sekä sen kannalta, miten vaatimusmäärittely näihin vaiheisiin vaikuttaa. Tutkimuksessa keskitytään aineiston sisällönanalyysiin.

### 1.2 Tutkimusmenetelmät

Toteutettava tutkimus on laadullinen. Opinnäytetyön tutkimuksellinen osio toteutetaan sisältöanalyysina. Analyysissa kerätään sekä analysoidaan aiheeseen liittyvää tutkimuskirjallisuutta, joissa aihetta käsitellään ja näiden avulla luodaan pohja ja ymmärrys vastaukseksi tutkimuskysymyksiin. Analyysin kautta tutkitaan, miten vaatimusmäärittelyn laatu, tarkkuus sekä erilaiset toteutustavat vaikuttavat ohjelmiston laatuun sekä eri projektin vaiheisiin.

Aineisto kerätään akateemisista tietokannoista ja kirjastoista (kuten esim. Google Scholar, HH Finna, IEEE Xplore jne.) sekä ammatillisista julkaisuista, jotka käsittelevät aihepiiriä. Edellä mainittujen aineistojen avulla tutkitaan aihetta teoreettisesta näkökulmasta. Tämän lisäksi käytännön näkökulmasta opinnäytetyössä hyödynnetään mahdollisuuksien mukaan kirjoista löytyviä esimerkkejä aiheeseen liittyen. Työssä tarkastellaan myös alalla toimivien yrityksien verkkosivustoja sekä raportteja, joissa käsitellään vaatimusmäärittelyä, laatuvaatimuksia sekä muita aiheeseen liittyviä osia.

Käytännössä opinnäytetyössä aineistoista pyritään tunnistamaan toistuvia teemoja sekä käsitteitä. Analyysi alkaa teemojen tunnistamisella, jonka jälkeen ne luokitellaan. Tämän jälkeen aineistoista

löytyneistä teemoista etsitään mahdollisia yhteyksiä. Lisäksi mahdollisesti eri teorioita vertaillaan keskenään ja tästä laaditaan johtopäätökset.

### 1.3 Keskeiset käsitteet

Alla kuvataan muutamia käsitteitä, joilla on oleellinen merkitys kyseisen tutkimustyön kannalta. Käsitteet luetellaan suomeksi ja englanniksi (suluissa).

Agile-kehitys, ketterä kehitys (Agile Development) – Vuonna 2001 luotu projektinhallintamenetelmä, jonka keskiössä ovat yksilöt sekä vuorovaikutus prosessien ja työkalujen sijasta. Menetelmässä painotetaan perinteisen projektihallinnan menetelmien lisäksi toimivaa ohjelmistoa ja muutoksiin reagoimista (OpenText 2024).

Ei-toiminnalliset vaatimukset (Non-functional Requirements) – Vaatimukset koko ohjelmistolle, jotka eivät ole toiminnallisia, kuten esimerkiksi käytettävyyteen sekä tietoturvaan liittyvät vaatimukset tai toimintaympäristön rajoitteet (Luukkainen, M. 2022).

Käyttäjätyytyväisyys (User Satisfaction) – Mittari määrittelemään käyttäjien tyytyväisyyttä tuotteeseen, palveluun tai kokemukseen (Trymata 2024).

Ohjelmiston laatu (Software Quality) – Ohjelmiston kyky täyttää sille asettamat vaatimukset tarkkailemalla esim. ohjelmiston käytettävyyttä, siirrettävyyttä, luotettavuutta tai tehokkuutta (Geeks for Geeks 2025).

Ohjelmistokehitysprosessi (Software Development Process) – Prosessi, joka kattaa ohjelmiston kehittämisen alusta loppuun, sisältäen viisi päävaihetta: määrittely, suunnittelu, toteutus, testaus ja julkaisu (Rytkönen & Kakkonen 2023, 61).

Testaus (Testing) – Menetelmä, jolla varmistetaan, että ohjelmisto täyttää sille asetetut vaatimukset sekä on virheetön. Testauksessa kartoitetaan myös mahdollisia ohjelmiston sekä vaatimuksien puutteita (Hamilton T. 2024).

Toiminnalliset vaatimukset (Functional Requirements) – Vaatimukset ohjelmiston tarjoamille toiminnoille (Luukkainen, M. 2022).

Validointi (Validation) – Prosessi, jossa tarkastellaan, täyttääkö ohjelmisto vaatimukset (Geeks for Geeks 2025).

Vaatimusmäärittely (Requirements Definition, Requirements Analysis, Requirements Specification) – Ohjelmistokehityksen vaihe, jossa määritellään ohjelmistolle asetetut vaatimukset, jotka ohjelmiston tulee täyttää (Luukkainen, M. 2022).

Vaatimusmäärittelyprosessin hallinta (Requirements Management) – Käytäntö, jossa seurataan vaatimuksia koko kehitysprosessin ajan, hallitaan muutoksia ja puututaan epäjohdonmukaisuuksiin (Springer Nature 2023).

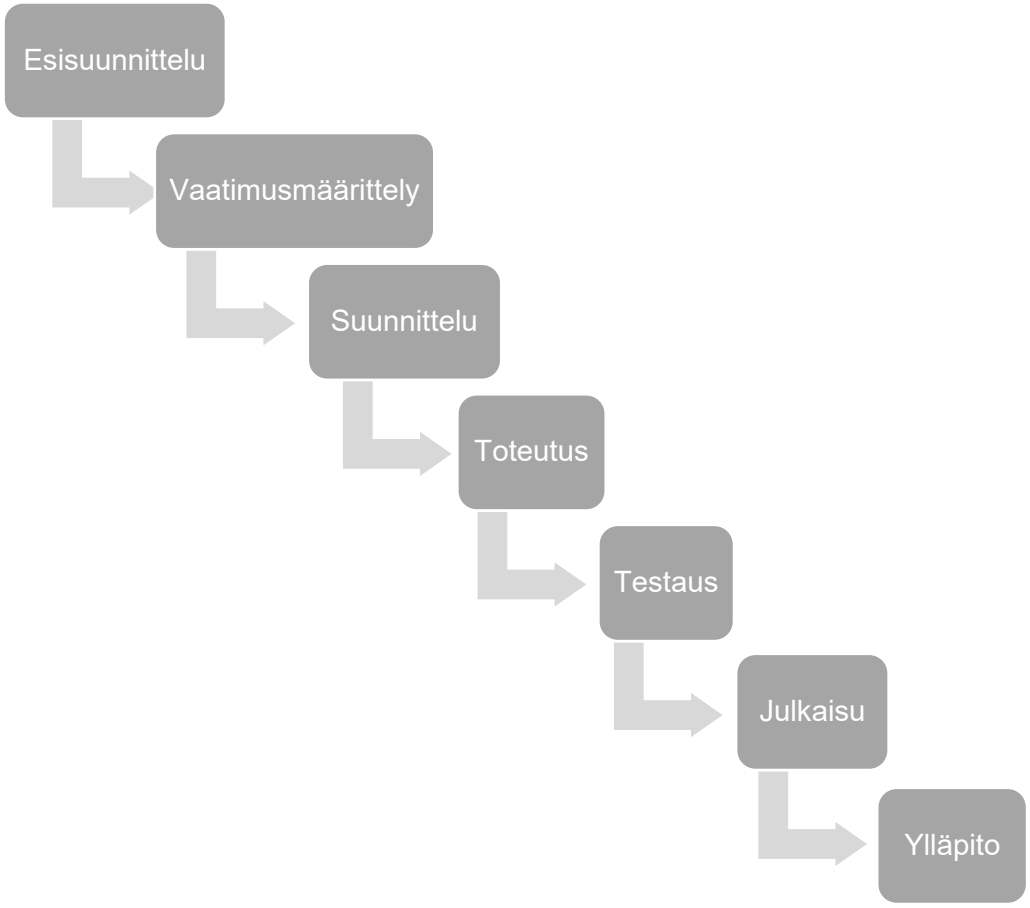
# 2 Ohjelmistokehitysprosessi

Luvun tarkoituksena on tarkastella tarkemmin ohjelmistokehitysprosessia kokonaisuutena sekä siihen kuuluvia vaiheita, menetelmiä sekä koko prosessin elinkaarta, jotta vaatimusmäärittelyn tarkoitus, osallisuus ohjelmistokehitysprosessissa sekä tutkimuksen lähtökohdat olisivat selkeät.

Ohjelmistokehityksen prosessi koostuu eri vaiheista, joiden tuloksena on saada aikaan toimiva, käyttäjien vaatimukset ja tarpeet täyttävä ohjelmisto. Ohjelmisto suunnitellaan toteutettavaksi tarpeesta, joka voi olla esim. tarve ratkaista jokin ongelma, henkilökohtainen käyttö tai liiketoiminnallinen tarve. Prosessia varten on kehitetty monia eri menetelmiä, joille on ominaista erilaiset toiminta- sekä lähestymistavat ohjelmistokehitykseen. (BrowserStack 2024.)

## 2.1 Ohjelmistokehityksen vaiheet

Jotta ohjelmistokehitysprosessi olisi toimiva, loogisesti ja tehokkaasti etenevä kokonaisuus, on sille määritelty eri vaiheet, jotka kukin sisältävät omat tärkeimmät tehtävänsä onnistuneen kokonaisuuden luomiseksi.




Kuva 1: Ohjelmistokehityksen vaiheet (muokattu, Harness 2023.)

Esisuunnitteluvaihe käynnistää koko prosessin. Tässä vaiheessa määritellään sekä rajataan tarve ohjelmistolle, sen tarkoitus ja tavoitteet sekä koko projektin laajuus. Osana esisuunnittelua tarkastellaan myös toteutettavuutta. Toteutettavuuteen vaikuttavat niin tekniset kuin myös taloudelliset seikat. Projektin toteutukseen luodaan projektisuunnitelma sekä määritellään vastuut ja roolit projektissa työskenteleville ihmisille. (Harness 2023.)

Esisuunnittelun jälkeen siirrytään keräämään tarkat vaatimukset ohjelmistolle niin sidosryhmiltä, käyttäjiltä kuin asiakkailtakin (Harness 2023). Tätä prosessin vaihetta kutsutaan vaatimusmäärittelyksi ja vaiheen tarkempi läpikäynti tapahtuu luvussa 3.

Kun tulevan ohjelmiston tärkeimmät vaatimukset on koottu, päästään suunnittelemaan ohjelmiston rakennetta, ohjelmistotekniikkaa, laitteistovaatimuksia ym. ohjelmistoarkkitehtuuria. Tässä vaiheessa tehdään myös käyttöliittymäsuunnitelmat, josta hahmottuu mahdolliset ensimmäiset versiot ohjelmiston näkyvimmästä osasta. Vaihe on ikään kuin linkki vaatimusten ja toteutuksen välillä ja sitoo nämä kaksi vaihetta oleellisesti yhteen. Tuloksena syntyy toteutustyötä ohjaava ohjelmistosuunnitteludokumentti. (Harness 2023.)

Toteutusvaihe on näkyvimmistä vaiheista koko ohjelmistokehitysprosessissa, jonka lopputuloksena on luotu ohjelmisto vaatimusten ja suunnittelun mukaisesti. Kehittäjät, insinöörit ym. lähtevät rakentamaan itse ohjelmistoa ja kehitystyötä seurataan tarkasti, jopa päivittäisellä tasolla. Toteutusvaihe elää limittäin testausvaiheen kanssa. Tämä tarkoittaa sitä, että koodia ja itse ohjelmistoa testataan jo, vaikka koko ohjelmiston kehitystyö ei ole vielä valmis. (Harness 2023.)

Ohjelmiston testaus toteuttaa laajalti ohjelmiston kehityksen laadullista näkökulmaa. Testauksen laajuus sekä yksityiskohdat määritellään ennalta määritettyjen vaatimusten pohjalta. Näiden pohjalta laaditaan testitapaukset, joiden avulla suoritetaan ohjelmiston testaus. Testausta on erilaista, kuten esim. yksikkötestaus, integraatiotestaus, turvallisuustestaus, järjestelmätestaus ja hyväksyntätestaus. Eri testauksilla varmistetaan, että ohjelmiston eri osat ja näiden yhdistelmät testataan laajalti. Testauksissa esille nousseet huomiot sekä virheet toimitetaan kehittäjille, jonka jälkeen testaus suoritetaan uudestaan. (Harness 2023.)

Julkaisuvaiheessa testattu ohjelmisto julkaistaan tuotantoon. Tämä vaihe tuo ohjelmiston loppukäyttäjän käytettäväksi. Tapa, miten ohjelmisto julkaistaan tuotantoon, vaihtelee sen mukaan miten ohjelmistoa käytetään ja mikä vaikutus tuotantoon viennillä on jo olemassa oleviin järjestelmiin sekä tuotantoprosessiin. Julkaisuvaiheeseen kuuluu oleellisesti myös käyttäjien sopeuttaminen järjestelmän käyttöön esim. koulutuksien pitämisellä ja ohjeiden laatimisella. Tämä vaihe ei kuitenkaan ole vielä ohjelmistokehitysprosessin loppu. (Harness 2023.)

Ohjelmistokehitysprosessin viimeinen vaihe on ylläpitovaihe. Tämä voi ohjelmasta riippuen kestää vuosikausia. Ylläpitovaiheen tärkeimpiä tehtäviä on tuki ohjelmiston käyttöön sekä käytöstä heränneiden tarpeiden havainnoimisella ja käytäntöönpanolla. Vaiheessa kiinnitetään huomiota käyttäjien palautteeseen ohjelmistosta sekä muista huomioon tulleista virhe- tai kehityskohdista. Säännölliset ohjelmistopäivitykset edellä mainittujen johdosta kuuluvat oleellisesti ylläpitovaiheeseen. (Harness 2023.)

### 2.2 Ohjelmistokehityksen menetelmät

Ohjelmistokehityksen prosessien hallintaan ja läpivientiin on vuosikymmenien saatossa kehitetty monia eri menetelmiä. Prosessin hallinta sekä itse kehitystyö tukevat toisiaan ohjelmiston kehitysprosessissa. Edellä mainittujen yhteistyön vaikutus on monien tahojen toimesta todettu parantavan koko ohjelmiston kehitysprosessia mm. laadukkaan lopputuotteen sekä tehokkaan resurssienhallinnan näkökulmasta. (Al-Saqqa S. 2020.)

Perinteisistä ohjelmistokehityksen menetelmistä tunnetuin on vesiputousmalli (Dima, A. & Maassen, M. A. 2018). Perinteiset menetelmät soveltuvat yksinkertaisten, ennalta ennustettavien projektien hallintaan hyvin. Jos kuitenkin tiedossa on tulevan projektin monimutkaisuus sekä monivaiheisuus, on riskinä, että perinteinen vesiputousmalli ei taivu kehitystyöhön tai mahdollisiin muutoksiin tarpeeksi ketterästi. Tällä voi olla taas projektin etenemisen kannalta ei-toivottuja seurauksia. Perinteisten menetelmien lähtökohtana on, että projektisuunnitelma, vaatimukset sekä muu dokumentointi laaditaan sekä määritellään ennen kehitystyön alkua. Tämän on todettu johtavan ongelmiin mm. ylläpidon sekä muuttuvien käyttäjien vaatimusten osalta. Muutosten ilmettyä reagointi niihin on hidasta. (Al-Saqqa S. 2020.)

Ketterien menetelmien (Agile) lähestymistavat toivat mukanaan vuosisadan vaihteessa ratkaisuja yllä oleviin perinteisten menetelmien yleisimpiin ongelmiin. Vuonna 2001 julkaistussa Agile Manifestossa esitettiin neljä keskeistä arvoa uudenlaiselle ohjelmistokehityksen prosessimenetelmälle. Nämä neljä arvoajatusta ovat seuraavia:

1. Yksilöt ja vuorovaikutukset vs. prosessit ja työkalut
2. Toimiva ohjelmisto vs. kattava dokumentaatio
3. Yhteistyö asiakkaan kanssa vs. sopimusneuvottelut
4. Muutoksiin vastaaminen vs. suunnitelman noudattaminen

Yhteen vedettynä ketterien menetelmien pääperiaatteena onkin olla nimensä mukaisesti joustava sekä asiakkaan vaatimuksiin taipuva. (OpenText 2024.)

Yksi viitekehyksistä, jonka avulla toteuttaa ketteriä menetelmiä on Scrum. Scrum pohjautuu sekin kolmeen perusperiaatteeseen: läpinäkyvyys, tarkastelu ja sopeutuminen. Scrum tarjoaa toteuttajilleen (ml. yksilöt, tiimit, organisaatiot) menetelmät mukauttaa ratkaisuja esille tulleisiin ongelmiin. Scrumissa määrätään periaatetta eteenpäin vievä Scrum Master, joka ohjaa projektin parissa työskenteleviä tuoteomistajaa tai heidän ryhmäänsä, muuta Scrum-tiimiä sekä sidosryhmiä. Scrum-tiimi on yleiskäsite koko projektin parissa työskenteleville, sisällyttäen tuoteomistajat, kehittäjät ja Scrum Masterin. (Schwaber K. & Sutherland J. 2020.)

Scrum-viitekehyksen mukaisesti työskentely tapahtuu muutaman viikon sprinteissä. Tuoteomistajat määrittelevät ongelmat tuoteportfolioon (Product Backlog). Jokaiselle sprintille valitaan portfoliosta työt sprintin suunnitellussa (Sprint Planning), jotka toteutetaan sprintin aikana. Sprintin lopuksi saadut tulokset käydään läpi sprintin tarkastelussa (Sprint Review) sekä tehdään valinnat ja muutokset seuraavaa sprinttiä varten. Sprintin retrospektiivissä tiimi yhdessä arvioi työskentelyään ja päättää konkreettiset toimenpiteet seuraavaa sprinttiä varten. Sprintin aikana työskentelyä seurataan päivittäisissä, lyhyissä Daily-palavereissa, jossa puretaan esille nousseet esteet ja sovitaan seuraavat askeleet. (Schwaber K. & Sutherland J. 2020.)

Perinteiset ohjelmistokehityksen menetelmät sekä ketterät menetelmät tarjoavat molemmat erilaisia ratkaisuja ohjelmistokehityksen prosessiin. Mutta mitkä ovat isoimpia eroavaisuuksia näiden kahden välillä ja miten työskentely näiden osalta eroaa? Tarkastellaan näitä seuraavan taulukon avulla.

| | Perinteiset menetelmät | Ketterät menetelmät |
|---|---|---|
| **Muokattavuus** | Hankala | Helppo |
| **Kehityksen lähestymistapa** | Ennustava | Mukautuva |
| **Kehityksen suuntautuminen** | Prosessikeskeinen | Asiakaslähtöinen |
| **Projektin koko** | Suuri | Keskikokoinen tai pieni |
| **Suunnittelun aikaväli** | Pitkäaikainen | Lyhyt |
| **Johtaminen** | Komento- ja kontrollityyli | Johtajuus ja yhteistyö |
| **Oppiminen** | Toissijaista kehitykselle | Jatkuvaa |
| **Dokumentointi** | Kattava | Ei-kattava |

| | | |
|---|---|---|
| **Organisaation tyyppi** | Korkea liikevaihto | Kohtalainen tai matala liike-vaihto |
| **Työntekijöiden määrä** | Suuri | Pieni |
| **Budjetti** | Korkea | Matala |
| **Tiimien määrä** | Monia | Yksi |
| **Tiimin koko** | Keskikokoinen | Pieni |

Taulukko 2: Perinteiset menetelmät vs. ketterät menetelmät (Al-Saqqa S. 2020.)

Kuten taulukosta voidaan päätellä, ketterät menetelmät ovat hieman enemmän matalan kynnyksen projektimenetelmiä ja sopivat muokattavuudeltaan erityisesti pienempiin sekä matalakustanteisempiin projekteihin, jossa keskiössä on asiakas ja yhteistyö. Kaikkein tärkeintä projektia aloittaessa ja toteutusmenetelmää päättäessä on miettiä, mitkä ovat projektin tavoitteet, rajaukset sekä millä tavoin sitä halutaan tai pystytään toteuttamaan ottaen huomioon loppukäyttäjät ja asiakkaat.

# 3 Vaatimusmäärittely

Vaatimusmäärittely on yksi koko ohjelmistokehityksen prosessin ensimmäisistä vaiheista sekä koko projektin peruspilareista. Vaatimusmäärittelyä tehdään lähestulkoon jokaisen ohjelmiston kehityksessä, jossa mukana on suunnittelua sekä vaativimpia teknisiä tai toiminnallisia piirteitä. Ohjelmistokehityksen alkaessa oli tavallista, että ohjelmistot olivat hyvin pieniä eikä näin ollen vaatineet paljoa määrittelyä. Ohjelmistojen monimutkaistuessa huomattiin kuitenkin pian vaatimusmäärittelyn tärkeys koko projektin kannalta. (Paakki J. 2011.)

Seuraavissa luvuissa syvennytään vaatimusmäärittelyn keskeisimpiin kohtiin.

## 3.1 Vaatimusmäärittelyn rooli ohjelmistokehityksessä

Jotta ohjelmistoa on mahdollista lähteä rakentamaan tavoitteellisesti sekä määrätietoisesti laajassakin projektissa, on sille määriteltävä projektin alussa vaatimuksia. Vaatimuksia voivat olla esim. toiminnallisia sekä ei-toiminnallisia. Lähtökohtana vaatimuksille on se, että projektin lopussa ja kehitystyön valmistuttua on ohjelmiston täytettävä sille asetetut vaatimukset. Vaatimuksia voivat olla myös rajoitukset, joiden sisään ohjelmiston on asetuttava. Sidosryhmien tärkeimpänä tehtävänä projektin alussa on juuri koota vaatimukset, jotka asettuvat projektin edetessä ohjelmiston ominaisuuksiksi. (Paakki J. 2011.)

Holtinin, Perryn ja Brownswordin (2012) mukaan vaatimusmäärittelyn tarkoitus on selkeää ohjelmistokehityksen kannalta ja tähän on viisi syytä. Määritellyt vaatimukset toimivat ikään kuin projektin ajurina: tehdyt toiminnot liitetään aina johonkin vaatimukseen. Ilman vaatimuksia tekeminen ei olisi niin suoraviivaista tai selkeää. Koko vaatimusmäärittelyn prosessin tarkoitus on auttaa selventämään ja esittämään vaatimukset projektin sekä ohjelmiston kannalta oikein sekä selkeästi. Lisäksi vaatimukset toimivat kriteereinä projektin loppuvaiheessa hyväksymistestauksessa sekä tuotteen hyväksynnässä yleisesti. Yhtenä syynä tuodaan esille myös itsevarmuus projektin kulussa. Kun vaatimukset ovat selkeitä, tuo se myös varmuutta toteutusvaiheessa. Viimeisenä ja merkityksellisempänä syynä tämän opinnäytetyön kannalta mainitaan vaatimusten tärkeys tuotteen laadun kannalta.

## 3.2 Vaatimusmäärittelyn prosessi

Vaatimusmäärittelyn prosessi mukautuu ohjelmiston sekä yrityksen tarpeisiin projektin osalta, mutta Sommervillen (2016) mukaan prosessi voidaan karkeasti jakaa viiteen eri osaan. Vaatimusmäärittelyn osalta prosessin kulku ei kuitenkaan ole yleensä lineaarista ketterässä projektinhallinnassa. Työn eteneminen on enemmän spiraalimaista, joka mahdollistaa uusien vaatimuksien ilmenemisen ja työstön läpi prosessin samalla kuin muita vaiheita on käynnissä.


Kuva 2: Vaatimusmäärittelyn vaiheet (Sommerville I. 2016.)

Prosessi lähtee liikkeelle vaatimusten selvittämisellä sekä löytämisellä. Tässä vaiheessa vaatimusmäärittelyn prosessia sekä koko kehitystyötä on tärkeää, että sidosryhmät on löydetty sekä määritelty ja sitoutettu osaksi projektia. Näin pystytään kartoitusvaiheessa hyödyntämään tulevan ohjelmiston kokonaisuutta ja tarpeita ymmärtäviä henkilöitä esimerkiksi haastattelemalla tai ideointipalaverien muodossa. Kartoitusvaiheessa myös kehittäjätiimi osallistuu vaatimuksien laatimiseen sekä lähdetään yhdessä luomaan eri käyttäjärooleja. (Luukkainen, M. 2022.)

Analysointivaiheessa löydetyt vaatimukset tarkastellaan sekä niitä verrataan keskenään. On tärkeää kiinnittää huomiota siihen, etteivät vaatimukset ole ristiriidassa keskenään. Lisäksi on hyvä huomioida se, ovatko vaatimukset toteutettavissa, myös budjetin kannalta. Vaatimukset olisi hyvä myös tässä vaiheessa saada tarpeeksi yksiselitteisiksi. Tämä helpottaa järjestelmän eri osioiden toteuttamista, mutta myös testaamista. Analyysivaiheessa vaatimukset myös priorisoidaan ja luokitellaan, tarvittaessa tarkennetaan järkeviksi osasiksi. Jotta vaatimukset pystyy toteuttamaan ja myöhemmin testaamaan, on ne jossakin muodossa dokumentoitava kehittäjä- sekä testaustiimin käyttöä varten. Dokumentoiduista vaatimuksista puhutaan usein ohjelmiston spekseistä. (Luukkainen, M. 2022.)

Ideointi- sekä dokumentointivaiheessa saattaa uudet ideat sekä toteutumistavat lennellä liiankin lennokkaasti sidosryhmien sekä kehitystiimin kesken. Validointivaiheessa on tärkeää varmistaa, että vaatimukset vastaavat ensinnäkin asiakkaan tarpeita sekä vaatimuksia. Tässä vaiheessa on myös hyvä varmistaa, että asiakkaiden tarpeiden lisäksi ohjelmisto täyttää myös yleisesti liiketoiminnan tarpeet. Ketterien menetelmien mukaisesti uusia vaatimuksia voi tulla läpi projektin sekä vanhoja vaatimuksia saatetaan tarkentaa tai muuttaa. Tätä vaihetta kutsutaan vaatimusten hallinnaksi. Vaatimusten hallintaan palataan tarkemmin luvussa 3.4. (Luukkainen, M. 2022.)

### 3.3 Vaatimuslajit

Ohjelmistolle määritetyt vaatimukset jaottuu karkeasti kahteen eri kategoriaan. Näitä ovat toiminnalliset vaatimukset sekä ei-toiminnalliset vaatimukset.

Toiminnallisilla vaatimuksilla tarkoitetaan vaatimuksia, jotka ensisijaisesti kuvaavat ohjelmistoa siltä kantilta, mitä se tekee. Näin ollen voisi sanoa, että ne ovat vaatimuksia liittyen ohjelman tarjoamiin

palveluihin ja tarkemmin määriteltynä toimintoihin. Tämä sisällyttää monesti myös sen, mitkä ovat ohjelmiston rajoitteet toiminnallisuuksien näkökulmasta eli mitä ohjelmisto ei tee. Toiminnalliset vaatimukset kuvataan usein käyttöskenaarioina, jolloin niistä käytetään ketterissä menetelmissä termiä käyttäjätarinat. Kuvaamalla vaatimukset skenaarioina, saadaan tarkkojakin vaatimuslistoja eri toiminnoista, jotka ohjelmiston on toteutettava. (Luukkainen, M. 2022.)

Sommervillen (2016) mukaan ei-toiminnallisilla vaatimuksilla tarkoitetaan vaatimusjoukkoa, joka kuvaa enemmän ohjelmiston ominaisuuksia ja tapaa, millä ja miten ohjelmisto toimii. Tästä syystä ei-toiminnalliset vaatimukset voivat monesti olla kriittisempiä, kuin toiminnalliset vaatimukset. Luukkaisen (2022) mukaan ei-toiminnalliset vaatimukset voidaan jakaa vielä erikseen laatuvaatimuksiin sekä toimintaympäristön vaatimuksiin. Laatuvaatimukset voivat sisällyttää määrityksiä mm. siitä, miten ohjelmisto toimii käytettävyyden osalta, siitä miten tietoturva-asiat hoidetaan sekä suorituskyvyn liittyvistä seikoista. Toimintaympäristön vaatimuksissa taas kartoitetaan mm. toteutusteknologiat, käyttöympäristöt ja integraatiot muihin ohjelmistoihin. Näin ollen ei-toiminnalliset vaatimukset sisältävät myös tulevan ohjelmiston rajoitteet.

### 3.4 Vaatimusmäärittelyn hallinta

Ketterien menetelmien periaatteiden mukaisesti voi myös vaatimuksia tulla lisää projektin aikana tai niitä voidaan tarvittaessa muokata. Tämän takia on tärkeää, että nämä muutokset ovat ylipäänsä mahdollisia sekä muutoksia voidaan hallita, dokumentoida sekä jäljittää.

Vaatimusmäärittelyn hallinta voidaankin karkeasti jakaa neljään tunnistettuun osa-alueeseen. Näitä ovat versionhallinta, muutoksenhallinta, vaatimusten tilan seuranta sekä vaatimusten jäljentäminen. Versionhallinnan avulla päästään käsiksi vaatimusten aikaisempiin versioihin ja mahdollisiin muutoksiin. Muutoshallinta mahdollistaa vaatimusten muutokset kesken projektin. Tilan seurannan tarkoituksena on pysyä kartalla siitä, missä tilassa eri vaiheissa kukin vaatimus on projektin edistymisen kannalta. Nämä eri tilat määritetään projektikohtaisesti ja tarpeiden mukaan. Vaatimusten jäljentäminen taas on tärkeää, jotta projektityöskentelyssä pystytään seuraamaan vaatimusten muutoshistoriaa sekä tarvittaessa palaamaan täysin alkuperäisvaatimuksen juurille. (Wiegers, K. E. & Beatty, J. 2013)

Sommervillen (2016) mukaan vaatimusmäärittelyn muutoshallinta koostuu kolmesta osasta. Muutostarve syntyy tunnistetusta ongelmasta kesken projektin. Tunnistettu ongelma analysoidaan, jonka jälkeen aloitetaan muutoksen määrittely. Määrittelyn avulla päästään tarkemmin analysoimaan itse muutosta sekä mahdollisia kustannuksia osana projektia. Jos muutos päätetään toteuttaa, seuraavana vaiheena on muutoksen toteutus. Lopputuotteena tästä on muutettu vaatimus. Mahdollisesti tärkeimpänä vaiheena on muutoksen analysointi, sillä se antaa tärkeää tietoa

muutoksen laajuudesta sekä sen avulla pystytään tunnistamaan muutoksesta riippuvat dokumentit sekä muu materiaali.

# 4 Ohjelmiston laatu

Ohjelmiston laatu määrittyy useasta eri tekijästä. Pelkästään ohjelmiston toimivuus oikein määritelmien mukaan ei ole yksinään kovinkaan pätevä laatumittari. Ohjelmiston laatu ei myöskään määrity vain sen mukaan, täyttääkö se käyttäjien vaatimukset, vaan sen laatua pitää tarkastella myös esim. käytettävyyden, ylläpidettävyyden, koodin ymmärrettävyyden sekä ylläpidettävyyden ja yleisesti tarkoituksenmukaisuuden avulla. (Geeks for Geeks. 2025.)

Seuraavissa kappaleissa syvennytään ohjelmiston laadun yleisempiin kohtiin, jotta voidaan ylipäänsä opinnäytetyön aiheen mukaisesti tarkastella eri laatuun vaikuttavia tekijöitä vaatimusmäärittelyn näkökulmasta.

## 4.1 Ohjelmiston laadun määritelmiä

Geeks for Geeks (2025) nimeää useita eri tekijöitä, joiden täyttymisen avulla voidaan mitata yleisesti ohjelmiston laatua kokonaisvaltaisesti:

1. **Siirrettävyys**
   Hyvän siirrettävyyden avulla ohjelmisto saadaan toimimaan eri käyttöympäristöissä ja sujuvasti myös yhdessä muiden ohjelmistojen kanssa.
2. **Käytettävyys**
   Käytettävyyttä voidaan mitata eri käyttäjäryhmien vaivattomalla ohjelmiston käyttämisellä riippumatta siitä, onko ohjelmisto sinulle jo entuudestaan tuttu.
3. **Uudelleenkäytettävyys**
   Mikäli uusia ohjelmistoja halutaan kehittää, hyvällä uudelleenkäytettävyydellä saadaan käytettyä ohjelmiston osia hyödyksi uudessa projektissa.
4. **Oikeellisuus**
   Ohjelmiston oikeutta mitataan sen mukaan, ovatko kaikki sille määritetyt vaatimukset toteutuneet.
5. **Ylläpidettävyys**
   Ylläpidettävyydellä tarkoitetaan ohjelmiston kykyä tulla muutetuksi ilman suurempaa vaivaa tai kustannuksia esim. virheiden ilmentyessä tai uusia osia lisätessä.
6. **Luotettavuus**
   Ohjelmiston luotettavuus määritellään siinä esiintyvien virheiden määrällä. Mitä vähemmän virheitä ohjelmiston käytössä esiintyy, sitä luotettavampi ohjelmisto on.

7. **Tehokkuus**

   Esim. suoritinaikaa, muistia, levytilaa, verkkokaistaa sekä muiden resurssien käyttö optimoidusti tekee ohjelmistosta tehokkaan.

### 4.2 Laadunvarmistus ohjelmistokehityksessä

Ohjelmistokehitysprosessin laadunvarmistusta toteutetaan osana ohjelmiston korkealaatuista kehittämistä. Laadunvarmistuksella halutaan taata, että vaatimukset täyttyvät sekä käyttäjien odotukset tulevat yhtä lailla täytetyksi. Laadunvarmistus on siis ikään kuin tapa toteuttaa ohjelmiston laadullisia vaatimuksia. Laadunvarmistuksen tarkoituksena on olla ohjelmiston kehitystä tukeva jatkuva prosessi eikä niinkään yksittäinen vaihe osana ohjelmistokehitystä. Laadunvarmistuksen avulla ohjelmistoa ikään kuin testataan jatkuvasti ja päätarkoituksena on myös etukäteen ehkäistä virheiden syntymistä sen sijaan, että niitä yritettäisiin vain löytää ohjelmistosta. (Jaramillo, F. 2024.)

Laadunvarmistusprosessi alkaa suunnittelulla sekä strategialla tavoitteiden määrittämiseen yhteistyössä sidosryhmien kanssa. Seuraavassa vaiheessa näitä tavoitteita toteutetaan sekä seurataan. Konkreettisia tapoja toteuttaa laadunvarmistusta on monia. Manuaalinen ohjelmiston testaus on yksi yleisimpiä laadunvarmistusmenetelmiä automaatiotestauksen ohella. Lisäksi kehittäjät voivat itse tarkastaa luomiaan koodeja eri menetelmillä. Laadunvarmistusprosessiin kuuluu myös arviointi ja parantaminen, jossa tarkastellaan prosessia ja kehitetään sitä tarvittaessa. (Jaramillo, F. 2024.)

# 5 Vaatimusmäärittelyn vaikutus ohjelmiston laatuun

## 5.1 Analysoitava aineisto

Ainestoa lähdetään tutkimaan tutkimuskysymysten kautta, jotka esiteltiin luvussa 1.1. Aineistoksi valikoitui 2 alan artikkelia, joiden keskiössä oli tähänkin opinnäytetyöhön oleelliset aihealueet eli vaatimusmäärittely ja ohjelmiston laatu sekä erityisesti näiden keskinäinen sidos ohjelmistoprojekteissa.

Pargaonkarin (2023) artikkeli on kirjoitettu asiantuntijalähtöisestä ja käytännönläheisestä näkökulmasta aihepiiriin liittyen ja keskittyy ammatilliseen pohdintaan saatavilla olevaan tietoon pohjautuen. Kirjoittaja yhdistää teoriaa ja käytännön näkökulmia. Hofmannin sekä Lehnerin (2001) artikkeli perustuu kenttätutkimukseen, johon osallistui 15 eri vaatimusmäärittelytiimiä ohjelmistoprojekteista ja yhteensä haastatteluihin osallistui 76 sidosryhmän jäsentä. Tutkimuksessa yhdistyy niin määrällinen kuin myös laadullinen tutkimusote.

Seuraavassa kappaleessa esitetään artikkeleista löydetyt havainnot sekä näiden teemat liittyen vaatimusmäärittely- sekä laatutekijöihin. Nämä löydökset esitetään lukijalle taulukkomuodossa. Näistä löydöksistä tehdään johtopäätökset ja pohdintaosuus luvussa 5.3.

## 5.2 Aineiston läpikäynti

Pargaonkar, S. 2023. Synergizing Requirements Engineering and Quality Assurance: A Comprehensive Exploration in Software Quality Engineering.

| Rivi | Viittaus artikkeliin | Koodi ja/tai teema |
|---|---|---|
| **1.** | “Projektin onnistumisten edellytykset lähtevät selkeistä ja loppuun asti hiotuista vaatimuksista…” | Vaatimusmäärittelyn perusta |
| **2.** | “Työn uudelleen tekeminen ja kustannukset ovat seuraamuksia puutteellisista vaatimuksista…” | Puutteellisen vaatimusmäärittelyn seuraukset |
| **3.** | “Vaatimusmäärittelyn avulla luodaan pohja testaukselle, hyväksyntätestaukselle sekä ylläpitovaiheeseen.” | Vaatimusmäärittelyn perusta, laatuvaatimukset |
| **4.** | “Jäljitettävyyden menetelmät... tarjoavat keinoja analyysiin sekä muutoshallintaan.” | Jäljitettävyyden tärkeys |
| **5.** | “Puutteelliset vaatimukset, niiden dokumentointi tai analysointi vaikuttaa käyttäjien tarpeiden täyttämiseen.” | Vaatimusten selkeyden ja ymmärrettävyyden tärkeys, käyttäjätyytyväisyys |

| | | |
|---|---|---|
| **6.** | “Projektia ohjaa vaatimusmäärittelyn tavoitteet, toiminnallisuudet ja rajoitukset.” | Vaatimusmäärittelyn roolin tärkeys projektissa |
| **7.** | “Vaatimusmäärittely ja sen selkeys ohjaavat suunnittelusekä kehitysprosessia.” | Vaatimusmäärittelyn roolin tärkeys projektissa |
| **8.** | “Projektin suunnittelu, resurssien määrittely ja aikataulutukset pohjautuvat hyvään vaatimusmäärittelytyöhön.” | Vaatimusmäärittely ja projektin selkeys |
| **9.** | “Muutokset projekteissa ovat vääjäämättömiä. Muutoshallinnan lähtökohtana ovat hyvin dokumentoidut vaatimukset…” | Vaatimusmäärittelyn dokumentoinnin vaikutus muutoksiin |
| **10.** | “Laadunvarmistuksen avulla todennetaan, että ohjelmisto vastaa laadittuja vaatimuksia…” | Laadunvarmistus ja vaatimusmäärittely |
| **11.** | “Jäljitettävyyden avulla… varmistetaan, että kaikki vaatimukset testataan sekä validoidaan.” | Jäljitettävyyden tärkeys |
| **12.** | “Kun asioita tulkitaan väärin, sidosryhmien tarpeet kehittyvät tai vaatimukset ovat epäselviä, syntyy erimielisyyksiä.” | Riskit laadunvarmistuksessa ja kommunikaatiossa |
| **13.** | “Hyvien hyväksymiskriteerien avulla voidaan varmistua siitä, että testauksen tavoitteet täyttävät suunnitellut toiminnallisuudet.” | Vaatimukset, laadunvarmistus sekä testaus |

Hofmann, H. & Lehner F. 2001. Requirements Engineering as a Success Factor in Software Projects.

| Rivi | Viittaus artikkeliin | Koodi ja/tai teema |
|---|---|---|
| **1.** | "Menestyvät tiimit sitouttavat asiakkaan sekä käyttäjät osaksi vaatimusmäärittelyprosessia ja ylläpitävät hyviä suhteita muihinkin sidosryhmiin.” | Vaatimusmäärittelyn ja sidosryhmien välisen yhteistyön tärkeys |
| **2.** | "Vaatimusmäärittelytiimejä ohjaavat sidosryhmien priorisoimat vaatimukset." | Vaatimusmäärittelyn ja sidosryhmien välisen yhteistyön tärkeys |

| | | |
|---|---|---|
| **3.** | “Alkuperäisiä määrittelyjä sekä toteutuksia seurataan yl-läpidettävien vaatimusmäärittelymatriisin avulla.” | Jäljitettävyyden tärkeys |
| **4.** | ”Vaatimuksia parannellaan läpi projektin vertaisarvioin-tien, skenaarioiden ja läpikäyntien avulla.” | Arvioinnin tarve vaati-musmäärittelyssä |
| **5.** | “Onnistuneet tiimit iteroivat vaatimusmäärittelyprosesse-jaan keskimäärin kolme kertaa.” | Vaatimusmäärittelyn prosessi |
| **6.** | "Vaatimusmäärittelyä toteutettiin läpi koko projektin.” | Vaatimusmäärittelyn prosessi |
| **7.** | "Prototyyppien avulla kehitettiin eri malleja." | Vaatimusten ymmärtäminen |
| **8.** | "Vaatimusten muutoshallinta oli avointa sen sijaan, että määrittelyt olisi lyöty lukkoon.” | Vaatimusmäärittelyn vaikutus muutoksiin |
| **9.** | "Ei-toiminnallisten vaatimuksien liian vähäinen huomi-ointi johti suorituskyky-, kapasiteetti- ja rajapinnan vaati-muksiin." | Laadunvarmistus |
| **10.** | "Vaatimusmäärittelyprosessi jäi etäiseksi tiimeille, joissa havaittiin heikkoa projektinhallintaa ja puutteellista kou-lutusta.” | Puutteellisen vaatimusmäärit-telyn seuraukset |
| **11.** | "Eniten haasteita tiimeille tuotti vaatimusten priorisointi.” | Puutteellisen vaatimusmäärit-telyn seuraukset |
| **12.** | “Puutteelliset suorituskyky-, kapasiteetti- ja rajapintavaa-timukset saivat osakseen puutteita, kun johto keskittyi tietovaatimuksiin, kun taas muu tiimi keskittyivät toimin-nallisuuksiin.” | Laadunvarmistus |
| **13.** | "Heillä oli vaikeuksia löytää tasapaino muutostarpeiden sekä vakauden välillä." | Vaatimusmäärittelyn vaikutus muutoksiin |
| **14.** | “Monet sidosryhmät näkivät vaatimusmäärittelyn satun-naisena prosessina.” | Vaatimusmäärittelyn vaikutus muutoksiin, laadunvarmistus |

| | | |
|---|---|---|
| **15.** | “Projektipäälliköt päätyvät tiimeihin saatavuuden mukaan, eikä niinkään osaamistason mukaan.” | Sidosryhmät |
| **16.** | "Yhtä tapausta lukuun ottamatta nämä työkalut haittasivat vaatimusmäärittelytoimia." | Vaatimusmäärittelyn työkalut |
| **17.** | “Hyvät suhteet sidosryhmissä vaikuttavat asiakkaiden tarpeiden täyttämiseen” | Sidosryhmät, käyttäjätyytyväisyys |
| **18.** | "He kokivat tyytyväisyyttä ratkaisuun ja sen sopivuuteen enemmän kuin vaatimusmäärittelyyn." | Vaatimusmäärittelyn prosessin kehittäminen |

### 5.3 Johtopäätökset ja pohdinta

Aineistot eroavat toisistaan sisällöllisesti jonkin verran. Toinen on teoreettisempi, kun taas toinen tarjoaa suoriltaan käytännön vinkkejä tutkimustuloksiin pohjautuen. Opinnäytetyön tutkimusosuuden kannalta tämä oli mielenkiintoinen lähtökohta, koska artikkeleita analysoidessa esille tuli näistä eroavaisuuksista huolimatta monia yhtäläisyyksiä sekä keskeisiä teemoja, joissa yhdistyvät vaatimusmäärittely sekä laadullinen ote osana ohjelmistokehitysprosessia. Jos näitä teemoja kasaa yhteen, johtopäätökset ovat jokseenkin yksiselitteisiä.

Selkeä, konkreettinen, yhteistyössä sidosryhmien kanssa luotu vaatimusmäärittely luo merkittävän perustan kehitettävän ohjelmiston laadulle. Panostaminen vaatimuksiin läpi projektin vaikuttavat pitkään myös ylläpitovaiheeseen. Ilman selkeää ja systemaattista vaatimusprosessia kärsivät myös suunnittelu-, toteutus ja testausprosessit. Puutokset sekä epäselvyydet vaatimusmäärittelyssä johtavat virheisiin sekä huonompaan käyttäjätyytyväisyyteen. Jäljitettävyyden tärkeys korostuu vaatimusmäärittelyprosessissa läpi ohjelmistokehitysprosessin ja se tukee olemassaolollaan vaatimusten muutoshallintaa, testausta sekä validointia. Tehokkaan sekä laadullisen testauksen pohjana korostuvat myös hyvin laaditut vaatimukset, joiden avulla määrittyvät myös oleelliset hyväksymiskriteerit. Vaatimusmäärittely sekä laadunvarmistus eivät ole toisistaan irrallaan olevia projektin osia, vaan niiden välillä on selkeää vuorovaikutusta. Laadunvarmistuksessa peilataan ohjelmiston tilaa vaatimuksiin sekä vaatimusten avulla toteutetaan korkealaatuista laadunvarmistusta.

Sidosryhmien osallistamisella projektiin sekä vaatimusten luomiseen ja ylläpitoon on suoria vaikutuksia toimivaan kommunikaatioon, vaatimusten hyvään laatuun sekä käyttäjäkokemukseen ohjelmiston osalta. Toimivan sekä selkeästi määritetyn vaatimusmäärittelyprosessin koetaan vaikuttavan myös merkittävästi ohjelmiston laatuun sekä aikarajoissa pysymiseen. Ohjelmiston

vaatimusten ollessa kattavia varmistutaan siitä, että kokonaisvaltainen laatu saavuttaa toivotun tason sekä sisältää monenlaiset vaatimukset, ei esim. vain suorituskykyyn liittyvät.

Viimeisimpinä nostoina artikkeleista haluaisin esittää tiimin osaamisen sekä tarvittavat resurssit vaatimusmäärittelyyn. Vaatimuksia laativien tiimin jäsenten osaaminen on laatukysymys siinä missä muutkin. Mikäli osaaminen on vajavaista tai vasta kehittyvää, tämä voi näkyä viivästyksinä ja epäonnistumisina. Lisäksi tarvittavien resurssien määrittäminen vaatimuksien laatimiseen, jäljittämiseen, dokumentointiin sekä tarkasteluun läpi projektin on avainasemassa onnistuneessa ohjelmistokehitysprosessissa.

Vaatimusmäärittely sekä sen laatu (mm. selkeys, tarkkuus, johdonmukaisuus) vaikuttaa ohjelmiston virheiden määrään suoriltaan niin projektin aikana kuin myös julkaisun jälkeen. Jos vaatimusmäärittelyyn on laitettu resursseja ja huomiota, ohjelmiston laatu kasvaa ohessa. Koska vaatimukset vaikuttavat ohjelmiston laatuun sekä luovat pohjan testaukselle, ml. käytettävyystestaukset, sillä on suora vaikutus myös käyttäjätyytyväisyyteen. Mikäli vaatimusmäärittelyn prosessiin kiinnitetään huomiota ja sille varataan tarpeeksi resursseja sekä näin ollen dokumentointiin ja hallintaan panostetaan, vaatimusten muutokset parhaimmillaan ylläpitävät ja kasvattavat ohjelmiston laatua. Kun on tilaa ja mahdollisuuksia muuttua, niin on tilaa kehitykselle.

Vaatimusmäärittelyn tärkeys ohjelmistokehitysprosessissa on kiistatonta. Laadukas vaatimusmäärittely ohjaa, tukee sekä vetää raameja ja luo perustan laadulliselle kehitystyölle läpi projektin. Vaatimusmäärittelyn laadulla on yhteys koko ohjelmiston laatuun. Onnistunut vaatimusmäärittely edellyttää teknistä ja organisatorista osaamista, tehokasta viestintää, iteratiivista kehitystapaa ja vahvaa prosessinhallintaa. Näiden elementtien yhdistäminen mahdollistaa sen, että ohjelmisto todella vastaa sille asetettuja odotuksia – ja jopa ylittää ne.

## Lähteet

Al-Saqqa S. 2020. Agile Software Development: Methodologies and Trends. The University of Jordan. Luettavissa: https://core.ac.uk/download/pdf/482969304.pdf. Luettu: 27.3.2025

BrowserStack 2024. Understanding the Software Development Process. Luettavissa: https://www.browserstack.com/guide/learn-software-development-process. Luettu: 21.2.2025

Dima, A. & Maassen, M. A. 2018. From Waterfall to Agile software: Development models in the IT sector, 2006 to 2018. Impacts on company management. Luettavissa: https://www.jois.eu/files/21_557_Dima.pdf. Luettu 27.3.2025

Geeks for Geeks. 26.3.2025. Software Quality – Software Engineering. Luettavissa: https://www.geeksforgeeks.org/software-engineering-software-quality/. Luettu: 5.4.2025

Haikala, I. & Mikkonen T. Ohjelmistotuotannon käytännöt. 12. painos. Talentum. Helsinki.

Hamilton, T. 18.12.2024. What is Software Testing? Luettavissa: https://www.guru99.com/software-testing-introduction-importance.html. Luettu 15.3.2025

Harness. 2023. The Seven Phases of the Software Development Life Cycle. Luettavissa: https://www.harness.io/blog/software-development-life-cycle-phases. Luettu: 17.3.2025

Hofmann, H. & Lehner F. 2001. IEEE Software. Requirements Engineering as a Success Factor in Software Projects. Luettavissa: https://ics.uci.edu/~wscacchi/SA/Readings/Req-Engr-SuccessFactors-Software-July01.pdf. Luettu: 13.4.2025

Holt, J. & Perry, S. A. & Brownsword, M. 2012. Model-based requirements engineering. The Institute of engineering and technology. Lontoo.

Jaramillo, F. 2024. Leanware. What is Quality Assurance (QA) in Software Development? Luettavissa: https://www.leanware.co/insights/what-is-quality-assurance-qa-in-software-development. Luettu: 7.4.2025

Koski, J. 2020. Määrittelymenetelmät ketterässä ohjelmistokehityksessä. Diplomityö. Tampereen yliopisto, informaatioteknologian ja viestinnän tiedekunta. Luettavissa: https://trepo.tuni.fi/bitstream/handle/10024/121614/KoskiJoonas.pdf?sequence=2. Luettu: 13.3.2025

Luukkainen, M. 2022. Ohjelmistojen vaatimusmäärittely, tuotteen ja sprintin hallinta. Helsingin avoin yliopisto. Luettavissa: https://ohjelmistotuotanto-hy-avoin.github.io/osa2/?utm_source=chat-gpt.com. Luettu: 13.3.2025

OpenText 2024. What is Agile Development? Luettavissa: https://www.opentext.com/what-is/agile-development#agile-development. Luettu: 15.3.2025

Paakki J. 2011. Ohjelmistojen vaatimusmäärittely. Helsingin yliopisto. Luettavissa: https://www.cs.helsinki.fi/u/paakki/Vaatimus-11-Luentokalvot-1.pdf. Luettu: 30.3.2025

Pargaonkar, S. 2023. Synergizing Requirements Engineering and Quality Assurance: A Comprehensive Exploration in Software Quality Engineering. Texasin yliopisto. Luettavissa: https://www.researchgate.net/publication/375450681_Synergizing_Requirements_Engineering_and_Quality_Assurance_A_Comprehensive_Exploration_in_Software_Quality_Engineering. Luettu: 13.4.2025

Rytkönen, M. & Kakkonen, K. 2023. ACT 2 LEAD. Ohjelmistotestauksen johtamisen käsikirja. 1. painos. Karkkila.

Schwaber K. & Sutherland J. 2020. The 2020 Scrum Guide. Luettavissa: https://scrum-guides.org/scrum-guide.html. Luettu: 28.3.2025

Sommerville I. 2016. Software Engineering. Pearson. 10. painos.

Springer Nature 11.10.2023. What do we know about requirements management in software ecosystems? Luettavissa: https://link.springer.com/article/10.1007/s00766-023-00407-w. Luettu: 14.3.2025

Tolvanen, P. 19.12.2023. Ketterät menetelmät olivat pieni askel oikeaan suuntaan. Vierityspalkki. Luettavissa: https://vierityspalkki.fi/2023/12/19/ketterat-menetelmat-olivat-pieni-askel-oikeaan-suuntaan/. Luettu: 10.3.2025

Trymata 2024. What is User Satisfaction? Luettavissa: https://trymata.com/blog/what-is-user-satisfaction/. Luettu 15.3.2025

Wiegers, K. E. & Beatty, J. 2013. Software requirements 3rd edition. Microsoft Press.