LAPIN AMK
Lapland University of Applied Sciences

VAATIMUSMÄÄRITTELYN TARKENNUSMENETELMÄT JA NIIDEN VAIKUTUKSET OHJELMISTOKEHITTÄJÄN JA ASIAKKAAN VÄLISEEN YHTEISYMMÄRRYKSEEN

Metsis Jarliin

Opinnäytetyö

Tietojenkäsittelyn koulutus
Tradenomi (AMK)

2022

LAPIN AMK
Lapland University of Applied Sciences

Opinnäytetyön tiivistelmä

Tietojenkäsittelyn koulutus
Tradenomi (AMK)

Tekijä	Jarliin Metsis	Vuosi	2022
Ohjaaja(t)	Ani Ruusila		
Työn nimi	Vaatimusmäärittelyn tarkennusmenetelmät ja niiden vaikutukset ohjelmistokehittäjän ja asiakkaan väliseen yhteisymmärrykseen		
Sivu- ja liitesivumäärä	30		

Opinnäytetyöni aiheena oli löytää ja tutkia erilaisia käytäntöjä sekä tarkennusmenetelmiä, joiden avulla pystytään muuttamaan vaatimusmäärittelyä selkeämmäksi. Tavoitteena oli löytää ratkaisu alalla yleisiin kommunikointiongelmiin asiakkaan ja ohjelmistokehittäjän välillä. Opinnäytetyössä perehdyttiin myös siihen, miten kommunikaation paranemisen myötä paranee myös tuotteen laatu.

Tutkimus toteutettiin laadullisena tutkimuksena ja aineistona käytettiin paljon monipuolista tutkimus- ja teoriatietoa. Opinnäytetyössä käytetty materiaali on suurilta osin peräisin ohjelmistoalalla työskentelevien asiantuntijoiden blogeista ja artikkeleista. Käytännön väärinkäsityksistä ohjelmistoalalla löytyi parhaiden tietoa blogeista. Tieto, mikä saatiin blogeista, korvasi mahdolliset haastattelut.

Opinnäytetyön tuloksena todettiin parhaimmiksi tarkennusmenetelmiksi prototypointi ja iterointi. Työssä kuitenkin todettiin, että menetelmiä kannattaa hyödyntää monipuolisesti ja valita käytettävissä olevien resurssien mukaisesti. Työssä nousi esille myös, mitä hyötyä on hyvästä vaatimusmäärittelystä ja miksi siihen kannattaa panostaa. Asiakaslähtöisellä vaatimusmäärittelykäytännöllä oli iso rooli projektin onnistumisen kannalta.

Avainsanat vaatimusmäärittely, asiakaslähtöisyys, laatuominaisuudet

LAPIN AMK
Lapland University of Applied Sciences

Abstract of Thesis

Degree Programme in Business
Information Technology
Bachelor of Business Administration

Author Jarliin Metsis Year 2022
Supervisor Ani Ruusila
Subject of thesis Methods of clarifying SRS document and their impacts on understanding between software developer and customer
Number of pages 30

The general aim of this thesis was to look at different practices and tools for clarifying the SRS document. The goal was to find a solution to common communication issues in the software development industry. The thesis also looked at how quality of the product improves together with communication.

The thesis was carried out as qualitative research. The theoretical information and research data were searched from different blogs and articles written by experts working in industry. Information about the issue of misunderstanding was gathered especially from blogs. The information that was gathered from blogs replaced possible interviews.

The results show that iterating and prototyping are the most useful tools for clarifying the SRS document. For the best results the tools should be selected according to available resources and use as diversely as possible. The thesis also highlighted the benefits of a good SRS document and the reasons to invest in SRS document improvement. Customer oriented requirement specification is the main keyword for successful software product.

Key words software requirements, customer orientation, quality characteristics

SISÄLLYS

KÄYTETYT TERMIT JA LYHENTEET

Bugi	virhe koodissa, joka aiheuttaa ohjelman toimimisen oikein (Hurja 2021).
Käytettävyys	tarkoittaa palvelun tai laitteen helppokäyttöisyyttä (Viljanen 2020).
Saavutettavuus	palvelu tai tuote on kaikkien käytettävissä yhdenvertaisesti, vammoista ja toimintarajoitteista huolimatta (Ojala 2022).
UML	graafinen mallinnuskieli, jonka tarkoituksena on visualisoida suunnitelma tuotteesta (Lynch 2022).

1 JOHDANTO

Asiakkaan ja ohjelmistokehittäjien välisistä väärinkäsityksistä johtuvat ongelmat ovat yleisiä ohjelmistokehityksessä. Väärinkäsitykset johtuvat usein siitä, että vaatimusmäärittelyn laatimiseen ei panosteta tarpeeksi. Lopputuotteena valmistuu silloin tuotteita, joissa on liikaa tai liian vähän ominaisuuksia, jolloin käytettävyys ja laatu kärsii. Pahimmillaan tuote ei vastaa asiakkaan tarpeita ja tarvitaan paljon korjauksia, mikä aiheuttaa määräajoista myöhästymistä, budjettien ylittymistä ja lisää työtunteja. (Ferens 2020.)

Ohjelmistokehittäjälle sanat eivät usein riitä kuvaamaan toivottua ominaisuutta tai tuotetta. Ominaisuuksien kartoittamista varten on hyvä käyttää vaatimusmäärittelyä ja muita tarkentavia menetelmiä. Tarkentavat elementit auttavat hahmottamaan ominaisuuksia visuaalisesti, mikä tekee asioista helpommin ymmärrettäviä ja selkeämpiä. Hyvin laadittu vaatimusmäärittely nopeuttaa myös itse kehitysprosessia. (Zomko 2021.)

Kun tuotteen suunnitteluun panostetaan ja se hoidetaan hyvin, saadaan tuloksena myös hyvä tuote. Siinä tapauksessa tuote vastaa asiakkaan toiveita ja tarpeita, työaika ja budjetti on käytetty kustannustehokkaasti sekä, tuote on laadukas ja tarjoaa hyvän käyttökokemuksen. Hyvästä suunnittelusta hyötyvät myös kaikki muut projektissa mukana olevat osapuolet, kuten esimerkiksi rahoittajat, testaajat ja kehittäjät. Yhteistyö ja yhteisymmärrys asiakkaan kanssa on tärkeässä roolissa tuotteen onnistumisen kannalta, ja siksi siihen on tärkeää panostaa. Tällä tavalla jää myös asiakkaalle hyvä asiakaskokemus, ja ohjelmistoyritys pystyy kasvattamaan sillä muun muassa mainetta. (Ferens 2020.)

Opinnäytetyön tutkimuskysymykset ovat:

- Miten voidaan parantaa asiakkaan ja ohjelmistokehittäjän yhteisymmärrystä tulevasta tuotteesta?
- Mitkä tekijät vaikeuttavat kehitysehdotuksien ymmärtämistä ja aiheuttavat väärinkäsityksiä?
- Millä menetelmillä saadaan tarkennettua vaatimusmäärittelyä?

- Mitkä ovat hyvän vaatimusmäärittelyn ja hyvän yhteisymmärryksen hyödyt?

2 VAATIMUSMÄÄRITTELY

2.1 Ohjelmiston vaatimusmäärittely

Ohjelmistokehityksessä vaatimusmäärittely on dokumentti joka, sisältää vaatimukset, odotukset ja standardit tulevalle tuotteelle tai ominaisuudelle. Kun kyseessä on ominaisuus, se on yleensä sen verran iso kokonaisuus, että se toteutetaan omana projektinaan. Dokumenttiin voi kuulua esimerkiksi tekstidokumentti, ajatuskartat, erilaiset kaaviot sekä muita elementtejä, joiden avulla tarkennetaan näkemystä tuotteesta. Vaatimusmäärittelyn avulla, projektissa työskentelevät tietävät, mitä tuotetta ollaan tekemässä, mitkä ovat tavoitteet ja prioriteetit sekä lopputulos ja miten siihen päästään. (Lane & Krüger 2021.)

Vaatimusmäärittely laaditaan yhdessä asiakkaan kanssa samalla kun he kuvailevat tarpeitansa ja ideoitansa tulevasta tuotteesta. Myöhemmin laaditaan vaatimusmäärittelyn perusteella myös tekninen vaatimusmäärittely. Yleensä laatijana toimii henkilö, joka on ollut yhteydessä asiakkaan kanssa, tutustunut heidän toimintaansa ja oppinut tutkimaan heidän tarpeitansa. Sellainen henkilö voi olla esimerkiksi projektipäällikkö tai tuotepäällikkö. (Zomko 2021.)

Vaatimusmäärittelyssä keskitytään enemmän ohjelman toiminnallisiin ominaisuuksiin ja asiakkaan toiveisiin kuin ohjelman tekniseen kuvaukseen. Siinä kerrotaan muun muassa projektin prioriteeteista, kuvaillaan ja avataan ohjelman toimintaa sekä kerrotaan, mitä prosesseja ohjelman tulee tukea. Näin pystytään ottamaan jo aikaisessa vaiheessa huomioon erilaisia näkökulmia ja havaitsemaan ongelmakohtia ohjelman toiminnassa. Hyvin laadittu vaatimusmäärittely onkin avain projektin onnistumiseen. (Zomko 2021.)

Vaatimusmäärittelyä voi olla haasteellista laatia, sillä ihmiset, jotka sitä käyttävät, ovat taustaltaan hyvin erilaisia ja sitä tutkitaan hyvinkin erilaisista näkökulmista. Dokumentin pitää olla kuitenkin kaikille ymmärrettävä, mahdollisimman selkeä, mutta kuitenkin tarkka. Vaatimusmäärittelyä käyttävät esimerkiksi ohjelmistokehittäjät, käyttöliittymäsuunnittelijat, testaajat, rahoittajat, asiakkaat ja niin edelleen. Kaikilla osapuolilla on iso rooli projektin onnistumisen kannalta. (Lane & Krüger 2021.)

Vaatimusmäärittelyä esitellään usein myös rahoittajille, jotta he saisivat mahdollisimman tarkan käsityksen tuotteesta ja sen ideasta. Siinäkin säästetään paljon resursseja, kun ei tarvitse tehdä heille erillisiä dokumentteja tuotteen toiminnasta. Myös testaajat käyttävät vaatimusmäärittelyä ja siellä esiteltyjä valmiin tuotteen kriteerejä apuna testien suunnittelussa. Asiakkailta varmistetaan vaatimusmäärittelyn avulla, että heidän ideaansa ja tarpeitansa on ymmärretty oikein ja suunnitelma vastaa niitä. (Ashraf, Aslam & Awan 2016.)

2.2 Vaatimusmäärittelyn tarkoitus

Vaatimusmäärittely laaditaan, jotta ohjelman kehittäjät ja käyttäjät ymmärtäisivät, miksi ohjelma on suunniteltu juuri tietyllä tavalla. Vaatimusmäärittelyn tarkoituksena on kartoittaa myös projektin etenemistä ja avata valmiin tuotteen toimintaa mahdollisimman tarkasti. Dokumentin laatimisen yhteydessä tulee varmasti vastaan myös ongelmatilanteita ohjelman toiminnallisuuksiin liittyen. Kun ongelmatilanteet huomataan aikaisessa vaiheessa, niihin voidaan reagoida niin, että se ei heikennä valmiin tuotteen laatua tai käytettävyyttä. (Santillan & Käkölä 2016.)

Vaatimusmäärittelyyn yritetään saada kirjattua mahdollisimman tarkka kuvaus ohjelman toiminnallisuuksista kaikissa mahdollisissa tilanteissa. Näin pysytään välttämään ongelmatilanteita sekä korjaamaan bugeja jo suunnitteluvaiheessa. Huomaamalla aikaisessa vaiheessa uusia näkökulmia pystytään uudet tilanteet ottamaan huomioon myös esimerkiksi käyttöliittymäsuunnittelussa. Silloin pystytään takaamaan muun muassa lopputuotteen parempi laatu, käytettävyys ja saavutettavuus. (Ashraf ym. 2016.)

Vaatimusmäärittelyn avulla pystytään myös jäsentelemään projekteja pienemmiksi osa-alueiksi tai jopa pieniksi työtehtäviksi. Jäsentelyn avulla avataan erilaisia tilanteita ja toiminnallisuuksia. Lisäksi määritellään projektin prioriteetit, tavoitteet sekä määritellään valmiin tuotteen kriteerit (*definition of done*). Sen avulla vaatimusmäärittelyn käyttäjät ymmärtävät paremmin miksi asiat on suunniteltu tietyllä tavalla ja mitä on tarkoitettu milläkin. Tällä tavalla projektissa työskentelevät tietävät myös, mitä vaaditaan projektin onnistumiseen ja miten projekti etenee. (Ashraf ym. 2016.)

Vaatimusmäärittelyn keskeisenä tehtävänä on myös parantaa keskustelua ja ymmärtämistä asiakkaan ja ohjelmistokehittäjän välillä. Väärinkäsitykset ovat yleinen ongelma ohjelmistoalalla. Asiakkaalla voi olla näkemys halutusta tuotteesta, mutta hän ei välttämättä osaa kuvailla sitä tarpeeksi hyvin tai pahimmassa tapauksessa näkemystä ei edes ole. Kehittäjä taas ymmärtää halutun tuotteen omalla tavallaan, mikä ei välttämättä ole sellainen, mitä asiakas tarvitsee tai on tarkoittanut. (Ashraf ym 2016.)

Lisäksi vaatimusmäärittelyä käytetään projektin jäsentelyssä pienemmiksi osa-alueiksi. Vaatimusmäärittelyn avulla varmistetaan myös, että asiakasta on ymmärretty oikein ja projektissa mukana olevat toimijat tietävät, mitä ovat tekemässä. Vaatimusmäärittelyn perusteella esimerkiksi sijoittajat voivat päättää, lähtevätkö he mukaan rahoittamaan projektia. Vaatimusmäärittelyn avulla nopeutetaan myös lopputuotteen julkaisua, sillä se nopeuttaa esimerkiksi testaamista ja kelpuutusta, kun on määritelty, minkälainen on valmis tuote. (Zomko 2021.)

Vaatimusmäärittely on tärkein tekijä projektin onnistumisen kannalta. Puutteellinen vaatimusmäärittely voi aiheuttaa väärinkäsityksien syntymistä, resurssien tuhlaamista, aikatauluista myöhästymistä sekä yrityksen maineen heikkenemistä. Välttääkseen tällaisia tilanteita on tärkeää panostaa vaatimusmäärittelyn huolelliseen laatimiseen. (Santillan & Käkölä 2016.)

2.3 Vaatimusmäärittelyn piirteet

Vaatimusmäärittely on muodoltaan pääasiallisesti tekstidokumentti. Lähtökohtaisesti sanat eivät riitä tarpeeksi tarkasti kuvailemaan ohjelman toimintaa tai graafista käyttöliittymää. Sen takia tarvitaan tekstidokumentin lisäksi tarkentavia elementtejä kuvailemaan, avaamaan ja perustelemaan tuotteen toiminnallisuuksia. Sellaisia tarkentavia elementtejä voivat olla esimerkiksi käyttäjätarinat, prototyypit, ajatuskartat ja niin edelleen. (Zomko 2021.)

Vaatimusmäärittelyyn kuuluu mahdollisimman tarkka kuvaus tuotteesta, kehitysprosessin vaiheet ja tavoitteet. Siinä kerrotaan myös, minkälaisia vaatimuksia ja odotuksia tuotteelle tai ominaisuudelle on. Vaatimusmäärittelyn tärkeimpänä tehtävänä on kuvailla mahdollisimman tarkasti, miten tuotteen tai ominaisuuden tulisi

toimia sekä miten kehittäjien tulisi toteuttaa se. Lisäksi priorisoidaan projektin osa-alueet ja työtehtävät. (Zomko 2021.)

2.4 Hyvän vaatimusmäärittelyn piirteet

Hyvän ja toimivan vaatimusmäärittelyn tuloksena syntyy laadukas, tehokas ja luotettavasti toimiva tuote. Siinä on huomioitu myös yrityksen tavoitteet, mittarit ja loppukäyttäjät. Lisäksi sen avulla pystytään välttämään määräajoista myöhästymistä ja lisäkustannuksia. Hyvän vaatimusmäärittelyn avulla pystytään saavuttamaan yrityksen tavoitteita ja ylläpitämään yrityksen mainetta sekä tuotteen laatua. (Zomko 2021.)

Hyvän vaatimusmäärittelyn ja projektin onnistumisen kannalta on tärkeää aloittaa tiivis yhteistyö asiakkaan kanssa heti projektin alussa. Kun asiakkaan kanssa ideoidaan yhdessä ja tehdään sen perusteella myös esimerkiksi luonnoksia, pystytään parhaiten varmistamaan, mitä asiakas haluaa ja tarvitsee. Myös vaatimusmäärittelyn laatimisen aikana on hyvä pitää asiakas ajan tasalla ja varmistaa, että kaikki on niin kuin pitää. Hyvän vaatimusmäärittelyn kulmakiviä onkin, että asiakkaan tarpeita on ymmärretty oikein heti alun pitäen. (Ferens 2020.)

Vaatimusmäärittelyssä tulee myös määrittää valmiin tuotteen kriteerit. Valmiin tuotteen kriteerit ovat tärkeitä, sillä niiden avulla pystytään hyvin pitkälti takaamaan tuotteen laatu. Valmiin tuotteen kriteerit eivät ole samoja kaikille projektin osapuolille, vaan jokaiselle osa-alueelle laaditaan omat kriteerit. Kriteerien avulla varmistetaan, että asiakkaalle toimitetaan tuote, mistä ei puutu ominaisuuksia, eikä niitä ole liikaa. Sellaiset tilanteet kuluttavat paljon resursseja, koska niissä tapauksissa tarvitaan tuotteeseen muokkauksia, mitkä vievät rahaa ja aikaa. (Huether 2017.)

Valmiin tuotteen kriteerien täyttyminen ei tarkoita välttämättä edes sitä, että kaikki käyttäjätarinat täytyisi olla toteutettuna. Tuote voidaan hyväksyä valmiiksi myös aikaisemmin, jos ominaisuus täyttää asiakkaan tarpeet. Valmiin tuotteen kriteereitä voivat olla esimerkiksi erilaiset testaukset tai asiakkaan määritellyt odotukset. (Huether 2017.)

2.5 Vaatimusmäärittelyn sisältö

Tässä luvussa esitellään vaatimusmäärittelydokumentin eri osa-alueita sekä, tehdään läpileikkaus vaatimusmäärittelyn osa-alueista ja siitä mitä ne pitää sisällään. Lisäksi kerrotaan millä keinoilla saadaan osa-alueesta selkeä ja helpommin ymmärrettävä kaikille. Puhutaan myös, mitä hyötyä on selkeästi ja huolellisesti laaditusta vaatimusmäärittelystä.

Vaatimusmäärittelyn alussa laaditaan esittely. Esittelyn keskeisin tehtävä on perustella, miksi tuotetta ollaan tekemässä. Siinä esitellään tuotteen idea ja perustellaan, mihin tuotetta tarvitaan. Esittelyssä kerrotaan, mitä lisäarvoa tuote tuo tuleville käyttäjille. Lisäksi määritellään projektin laajuus, tavoitteet ja prioriteetit. Esittely antaa hyvän yleiskuvan tuotteen ideasta ja tarkoituksesta sekä projektin. (Burak 2020.)

2.5.1 Kuvaus

Tuotteen kuvauksessa kuvataan mahdollisimman tarkkaan, miten tuotteen tulisi toimia eli avataan tuotteen ideaa. Kuvaillaan mahdollisimman laajasti, mitä tuotteella pitäisi pystyä tekemään. Siinä voi käyttää apuna esimerkiksi käyttötapauksia (*use case*). Käyttötapauksissa yritetään keksiä erilaisia tilanteita, mitä tuotteella pitäisi saada tehtyä. Tämä on hyvin selkeä tapa viestiä myös asiakkaalle, miten tuote on ymmärretty. (Bandakkanavar 2018.)

Seuraavaksi kuvataan vaatimusmäärittelyssä tuotteen omaisuuksia. Tuotteen ominaisuuksia voidaan kuvata esimerkiksi tietokantakaavion avulla. Siinä yhteydessä valmistuisi silloin myös suunnitelma tietokantarakenteen tekemiseen. Eli kerrotaan mitä tietoa ollaan tallentamassa ja mitä tietoa tarvitaan. Paras tapa kuvata tuotteen ominaisuuksia onkin jonkinlainen kaavio, esimerkiksi tietokantakaavio. Projektin edetessä tulevat ominaisuudet muuttumaan vielä paljon ja siksi tiedot on helpompi päivittää kaaviossa kuin esimerkiksi tekstidokumentissa. Kaaviosta tietoa on myös helpompi ja nopeampi lukea. (Bandakkanavar 2018.)

Tuotteen ominaisuuksien jälkeen esitellään tuotteen loppukäyttäjät ja kohderyhmä. Kerrotaan, ketkä ovat tuotteen käyttäjiä ja miten he käyttävät tuotetta. Määritellään tuotteen kohderyhmä ja kerrotaan jos on asioita mitä pitää ottaa

huomioon. Esimerkiksi jos tiedetään, että suurin osa tuotteen käyttäjistä on ikäihmisiä, niin se osattaisiin ottaa huomioon muun muassa käyttöliittymäsuunnittelussa. Tässä vaiheessa kerrotaan myös, jos tuotteella on eri käyttäjäryhmiä eli esimerkiksi eri tasoisia käyttäjiä, joilla on pääsy eri toiminnallisuuksiin. (Lane & Krüger 2021.)

Kuvaillakseen helposti ja selkeästi miten eri käyttäjäryhmät käyttävät tuotetta, on hyvä käyttää apuna esimerkiksi käyttäjätarinoita (*user stories*). Käyttäjätarinat ovat lyhyitä kuvauksia siitä, mitä käyttäjän pitäisi pystyä tuotteella tekemään ja miksi. Tapauksia käydään läpi mahdollisimman monipuolisesti ja eri käyttäjäryhmillä. Sillä tavalla saadaan kattava kuvaus myös siitä, miten ohjelman tulisi toimia. (Francino 2020.)

2.5.2 Tuotteen toiminnallisuudet ja rajapinnat

Tuotteen monipuolisen kuvauksen jälkeen esitellään muun muassa toimintavaiheet odotetun lopputuotteen saavuttamiseksi. Avataan vielä lisää ohjelman toiminnallisuuttaa esimerkiksi käyttäjätarinoiden avulla. Määritellään, minkälaisiin osa-alueisiin projekti jaetaan ja minkälaisia työtehtäviä ne sisältävät. Tässä vaiheessa on otettu jo huomioon erilaiset rajoitukset, toiveet ja ääritilanteet. (Burak 2020.)

Tuotteen toiminnallisuus, projektin osa-alueet ja työtehtävät pyritään kuvaamaan mahdollisimman tarkasti ja selkeästi, jotta ohjelmistokehittäjät osaisivat sen perusteella aloittaa työt. Tuotteen toiminnallisuuden kuvaamisessa on hyvä käyttää apuna erilaisia apuvälineitä. Sellaisia voivat olla esimerkiksi kaaviot tai luonnokset käyttöliittymästä. Tällaisten apuvälineiden avulla saadaan aikaiseksi mahdollisimman tarkka kuvaus tuotteen toiminnallisuuksista. (Burak 2020.)

Seuraavaksi vaatimusmäärittelyssä määritellään, millä tavalla tapahtuu vuorovaikutus käyttäjän ja ohjelman välillä ja mitä rajapintoja siihen tarvitaan. Siinä kerrotaan myös minkälaisille laitteille ja käyttöjärjestelmille tuote on suunniteltu. Esitellään myös minkälaista tietokanta ratkaisua ja mitä ohjelmointikieliä käytetään. Tässä kohtaa kerrotaan myös, miten projektissa mukana olevat työntekijät kommunikoivat keskenään, esimerkiksi käyttämällä Mattermost- tai Skype-palvelua (Lane & Krüger 2021).

2.5.3 Ei-toiminnalliset vaatimukset

Ei-toiminnallisiin vaatimuksiin määritellään, minkälaista suorituskykyä ohjelma vaatii. Siinä yritetään myös kartoittaa, kuinka paljon tuotteella tulisi olemaan käyttäjiä. Tässä kohtaa esitellään myös tietoturvaan liittyvät asiat ja kerrotaan, mitä aiotaan tehdä tietoturvan parantamiseksi ja ylläpitämiseksi. Päätetään myös projektin aikaiset kommunikointivälineet. (Burak 2020.)

Lisäksi kerrotaan tekijöistä, joilla voidaan vaikuttaa siihen, että julkaistava tuote olisi mahdollisimman laadukas. Siihen vaikuttavat tosi paljon esimerkiksi käytettävyys ja saavutettavuus. Kun käyttöliittymäsuunnittelussa otetaan huomioon tuotteen kohderyhmä ja panostetaan muutenkin siihen, että tuote olisi helppo ja johdonmukainen käyttää, ollaan lähempänä laadukasta lopputuotetta. (Burak 2020.)

2.6 Huonosta vaatimusmäärittelystä aiheutuvia ongelmia

Vaatimusmäärittely määrittelee aika pitkälti, että tuleeko projekti onnistumaan ja tuotteesta odotusten mukainen ja laadukas. Epäonnistunut tuote tarkoittaa yleensä tuotetta, mikä ei vastaa asiakkaan toiveita ja odotuksia. Siinä saattaa olla esimerkiksi liikaa tai liian vähän ominaisuuksia tai pahimmillaan asiakkaan toiveet on ymmärretty kokonaan väärin. (Lane & Krüger 2021.)

Vaatimusmäärittelyn epäonnistuminen aiheuttaa yleensä myös valtavaa taloudellista tappiota, resurssien tuhlausta, määräajoista myöhästymistä ja huonoa laatua. Kun asiakkaalle on toimitettu tuote mikä ei vastaa heidän odotuksiansa ja tarpeitansa, tarvitaan tuotteeseen korjauksia. Korjauksiin käytetään silloin paljon työtunteja ja rahaa. Resurssien tuhlausta voi aiheuttaa myös se, kun ohjelmistokehittäjät eivät tiedä mitä heidän pitää tehdä. Epäonnistunut tuote vaikuttaa myös yrityksen maineeseen. Yritys saattaa sitä myötä jopa menettää asiakkaita ja tuloja. Näitä ongelmia ei välttämättä olisi, jos vaatimusmäärittelyyn laatimiseen panostetaan. (Ashraf ym. 2016.)

Alan yleisenä ongelmana on, että asiakkaat ja ohjelmistokehittäjät tai vaatimusmäärittelyn laatijat eivät ymmärrä toisiaan oikealla tavalla. Monesti ongelmana on myös, että kun asiakkaat ovat tilaamassa tuotetta, mutta he eivät tarkalleen edes

tiedä mitä haluavat. Tällaiset tilanteet saattavat herkästi aiheuttaa väärinkäsityksiä ja jos vaatimusmäärittelyn laatimiseen ei panosteta ja sitä ei tarkistuteta asiakkaalla, tuotteella on suuri vaara epäonnistua. Yhteisymmärryksen parantamiseksi on hyvä käyttää kirjallisen vaatimusmäärittelyn lisäksi muita apuvälineitä viestimään mitä tarkoitetaan milläkin. Tällaisia apuvälineitä voivat olla muun muassa erilaiset kaaviot, havainnekuvat, luonnokset tai jo aikaisemmin mainitut käyttäjätarinat. (Ashraf ym. 2016.)

Ongelmia aiheuttaa myös tilanne, jos vaatimusmäärittely sisältää liikaa mielipiteitä tai keskustelua, eikä määritellä konkreettisesti mitä pitää tehdä. Mielipiteet ja keskustelut eivät kuulu vaatimusmäärittelyyn. Jos on vielä asioita, mitkä ovat auki tai päättämättä, ne laaditaan vaatimusmäärittelyyn vasta kun ne on selvitetty. (Golodov 2016.)

2.7 Tekninen vaatimusmäärittely

Kun yleinen vaatimusmäärittely on tehty kaikkien ymmärrettäviksi, luodaan sen perusteella tekninen vaatimusmäärittely, mikä on tarkoitettu ohjelmistokehittäjille. Yleinen vaatimusmäärittely kertoo mitä ollaan tekemässä, kun taas tekninen vaatimusmäärittely keskittyy siihen, miten tehdään. Teknisen vaatimusmäärittelyn ideana on antaa yleiskatsaus tuotteen arkkitehtuurista, määrittää työn laajuus sekä asettaa projektille välietapit. Laatimisen yhteydessä pyritään ratkomaan jo etukäteen isoimpia ja tärkeimpiä ongelmakohtia. Tekninen vaatimusmäärittely on yleensä esimerkiksi ohjelmistoarkkitehdin tai tiimin vetäjän laatima. (Altexsoft 2020.)

Teknisessä vaatimusmäärittelyssä voidaan kertoa esimerkiksi minkälaisia kirjastoja tullaan käyttämään, mitkä ovat yleiset linjaukset tyylien suhteen jne. Tarkentavina elementteinä saatetaan käyttää myös esimerkiksi kuvauksia käyttötapauksista ja pseudokoodia. Pseudokoodi on karkea rakenne siitä, miten koodin tulisi toimia ja miltä se voisi näyttää. (Altexsoft 2020.)

3 TARKENNUSMENETELMÄT

3.1 Tarkennusmenetelmät ja niiden merkitys

Vaatimusmäärittely on pääasiallisesti tekstidokumentti ja siinä kuvaillaan paljon sanoin, minkälainen tuotteesta olisi tarkoitus tulla ja mitä ominaisuuksia siihen tarvitaan. Siinä on huonona puolena muun muassa se, että tekstinä kuvatun asian voivat ihmiset ymmärtää todella monella eri tavalla. Lisäksi vaatimusmäärittelystä voi olla tosi työlästä löytää jotakin tiettyä asiaa. Tarkennusmenetelmät ovat vaatimusmäärittelyä tukevat menetelmät, jotka auttavat hahmottamaan tuotetta ja ominaisuuksia myös visuaalisesti. (Zomko 2021.)

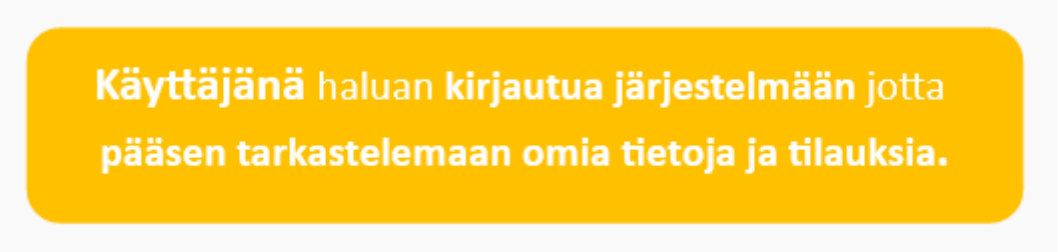
Ohjelmistokehityksessä on yleisenä ongelmana, että asiakkaita ei ymmärretä. Yleensä se johtuu ainakin osittain myös siitä, että asiakas ei tiedä itsekään mitä haluaa ja tarvitsee. Ongelmaa esiintyy myös toiseen suuntaan, eli asiakkaiden on hankala ymmärtää ohjelmistokehittäjiä. Tämä taas johtuu paljolti siitä, että ohjelmien toiminnallisuuksia on tosi hankala selittää ja usein kehittäjät eivät osaa asettaa itseään asiakkaan asemaan. Näiden ongelmien ratkaisemiseksi on hyvä käyttää tarkennusmenetelmiä, jotta päästään auttamaan asiakasta tunnistamaan tarpeitaan ja keskustelemaan ohjelman ominaisuuksista ja toiminnallisuuksista. (Ferens 2020.)

Tarkennusmenetelmät ovat usein muodoltaan esimerkiksi kaavioita tai kuvia. Niiden avulla vaatimusmäärittelystä tulee selkeämpi ja kaikille helpommin ymmärrettävä. Tarkennusmenetelmistä hyötyvät kaikki projektissa mukana olevat osapuolet. Kaavioita ja kuvia käytetään apuna muun muassa käyttöliittymän toteutuksessa ja testaamisessa. Tarkennusmenetelmien avulla on helppo varmistaa myös asiakkailta, että kaikki ovat ymmärtäneet asian samalla tavalla ja projekti etenee oikeaan suuntaan. (Ashraf ym. 2016.)

3.2 Käyttäjätarinat

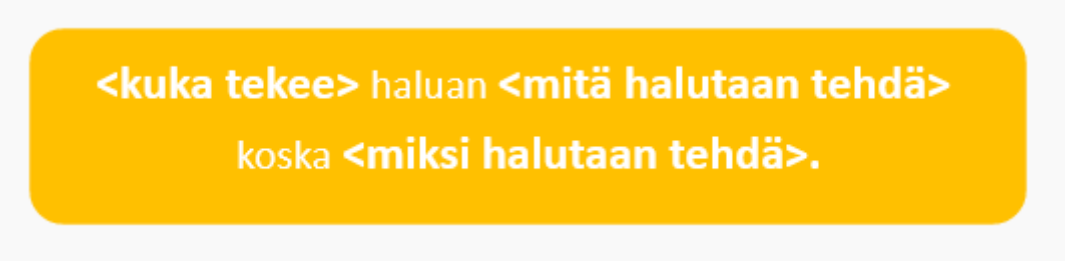
Käyttäjätarina on suosittu menetelmä vaatimusten esittämiseen ja sitä käytetään erityisen paljon ketterien menetelmien yhteydessä. Käyttäjätarinoiden tarkoituksena on kertoa mahdollisimman pienissä osissa, mitä käyttäjän tulee pystyä

ohjelmassa tekemään. Lisäksi tarkennetaan, mitä hyötyä siitä tekemisestä on tai mitä siitä seuraa (kuvio 1). Rakenteeltaan käyttäjätarinat ovat hyvin yksinkertaisia ja ne kertovat *kuka tekee, mitä halutaan tehdä* ja *mitä tekemisestä seuraa*. (Lucassen, Dalpiaz, van der Werf & Brinkkemper 2016.)



Kuvio 1. Mallirakenteen avulla täytetty esimerkki käyttäjätarinasta

Kuvion 2 mukaisesti *Kuka tekee* -osiossa määritellään rooli eli kuka on sillä hetkellä järjestelmän käyttäjänä. Rooli voi olla esimerkiksi käyttäjä, ylläpitäjä tai rekisteröitynyt käyttäjä. *Mitä halutaan tehdä*- osiossa määritellään mitä käyttäjän pitäisi pystyä ohjelmassa tekemään. Käyttäjätarinan tarkoituksena on kuvailla mahdollisimman pieni toiminto kerrallaan. Eli *mitä halutaan tehdä*- osioon voidaan merkitä esimerkiksi: haluaa täyttää lomakkeen ja lähettää sen tai painaa nappia x. *Mitä tekee*- osiossa kerrotaan miksi toimintoa pitää päästä tekemään, mahdollisesti voidaan myös tarkentaa mitä tulee tapahtumaan toiminnon jälkeen. Esimerkiksi jos tavoitteena on päästä ohjelmassa tiettyyn näkymään. Silloin voidaan esimerkiksi tarkentaa, että käyttäjänä haluan kirjautua järjestelmään, jotta pääsen kirjautuneen näkymään, missä pystyn tarkastelemaan omia tietoja ja tilauksia. (Lucassen ym. 2016.)



Kuvio 2. Käyttäjätarinan mallirakenne

Käyttäjätarina toimii tarkimpana kuvauksena vaatimuksesta mitä ohjelmistokehittäjät käyttävät rakentaessa ominaisuutta. Usein käyttäjätarinoille merkitään myös

otsikko eli osa-alue mihin käyttäjätarina liittyy, prioriteettiarvo sekä arvio kuinka kauan sen toteuttamiseen voisi kulua. Näiden avulla pystytään suunnittelemaan projektin toteutusta, eli missä järjestyksessä asioita tehdään ja kauanko siinä voisi arviolta kulua aikaa. (Zomko 2021.)

Käyttäjätarina on yksi selkeimmistä tavoista kommunikoida asiakkaan kanssa ohjelman toiminnallisuuksiin liittyen. Käyttäjätarina on tarpeeksi yksinkertainen ja helppo ymmärtää. Käyttäjätarinoissa käytetty kieli on kaikille ymmärrettävä ja se ei sisällä esimerkiksi ohjelmointiin liittyvää sanastoa tai muutenkaan hankalia termejä. Esimerkiksi, kun asiakas on kuvailemassa tuotteeseen toivottuja ominaisuuksia, kannattaa kirjoittaa kuvauksen perusteella käyttäjätarinat ja antaa sitten vielä asiakkaalle tarkistettaviksi. Tällä tavalla pystytään parantamaan kommunikaatiota asiakkaan kanssa. (Lucassen ym. 2016.)

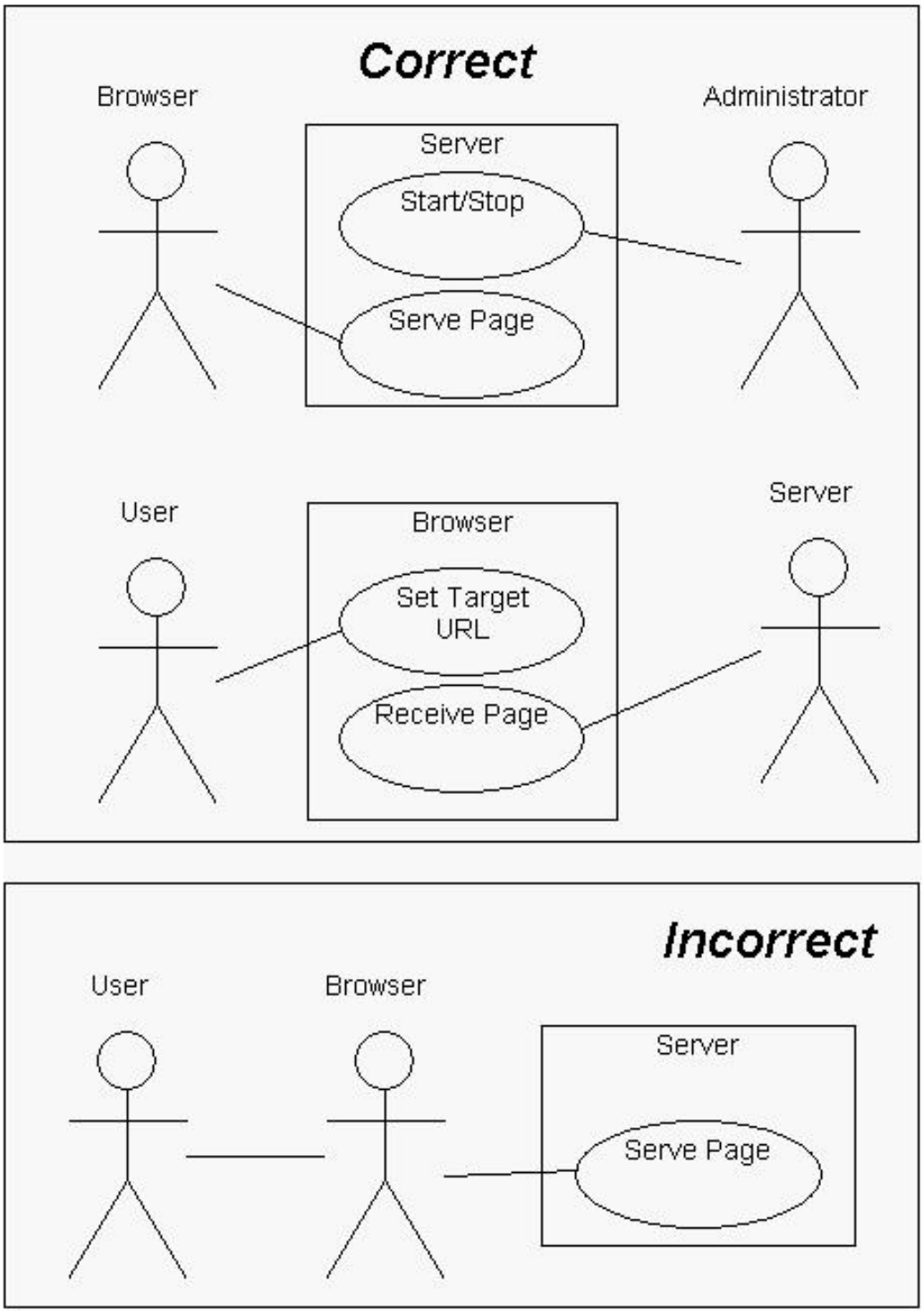
3.3 UML- käyttötapauskaavio

Käyttötapauskaaviossa yhdistetään käyttäjätarinat ja eri osapuolet toisiinsa sillä tavalla, että siitä pystytään hahmottamaan ohjelman tietyn osa-alueen toimintaa. Kaavion avulla nähdään käyttäjien ja toiminnallisuuksien välinen vuorovaikutus, sen avulla nähdään myös, miten osa-alueen tulisi odotetusti toimia. Yhteen käyttötapauskaavioon on mahdoton saada mukaan koko ohjelman toimintaa ja käyttäjätarinoita. Sen takia ohjelma pilkotaan pienemmiksi osa-alueiksi ja jokaiselle niistä luodaan omat käyttötapauskaaviot. Näin kaaviot pysyvät myös selkeinä ja helposti ymmärrettävinä. Käyttötapauskaavioissa on keskitytty enemmän toiminnallisuuksiin kuin esimerkiksi yhteyskaaviossa. (Jama Software 2019.)

Eri osapuolet esitetään käyttötapauskaavioissa yleensä tikku-ukkoina. Käyttäjätarinat on taas kirjattu lyhyinä toiminnallisuuksina ympyröihin. Käyttötapauskaavio rajataan niin, että siinä esitetään vain yhden osapuolen toiminnallisuudet kerrallaan. Yhteydet osapuolten ja käyttäjätarinoiden välillä esitetään vetämällä viivoja niiden väliin. Lopputuloksena saadaan johdonmukainen kaavio, esittämään tietyn osapuolen toimintaa ja sen yhteyksiä muihin osapuoliin. Käyttötapauskaaviot luodaan käyttämällä UML:ää. (Jama Software 2019.)

Käyttötapauskaavion avulla pystytään selittämään helpommin myös ohjelman logiikkaa ja toiminnallisuutta esimerkiksi asiakkaalle. Asiakkaille on usein hankala

kuvailla selkeästi ja ymmärrettävästi ohjelman toiminnallisuuksia ja yhteyksiä osapuolten välillä. Asiakkaat käsittävät asian merkittävästi selkeämmin, kun suunnitelma visualisoidaan heille, kuten kuviossa 3. Käyttötapauskaaviot auttavat parantamaan yhteisymmärrystä ja sitä myötä myös säästämään resursseja. (Jama Software 2019.)

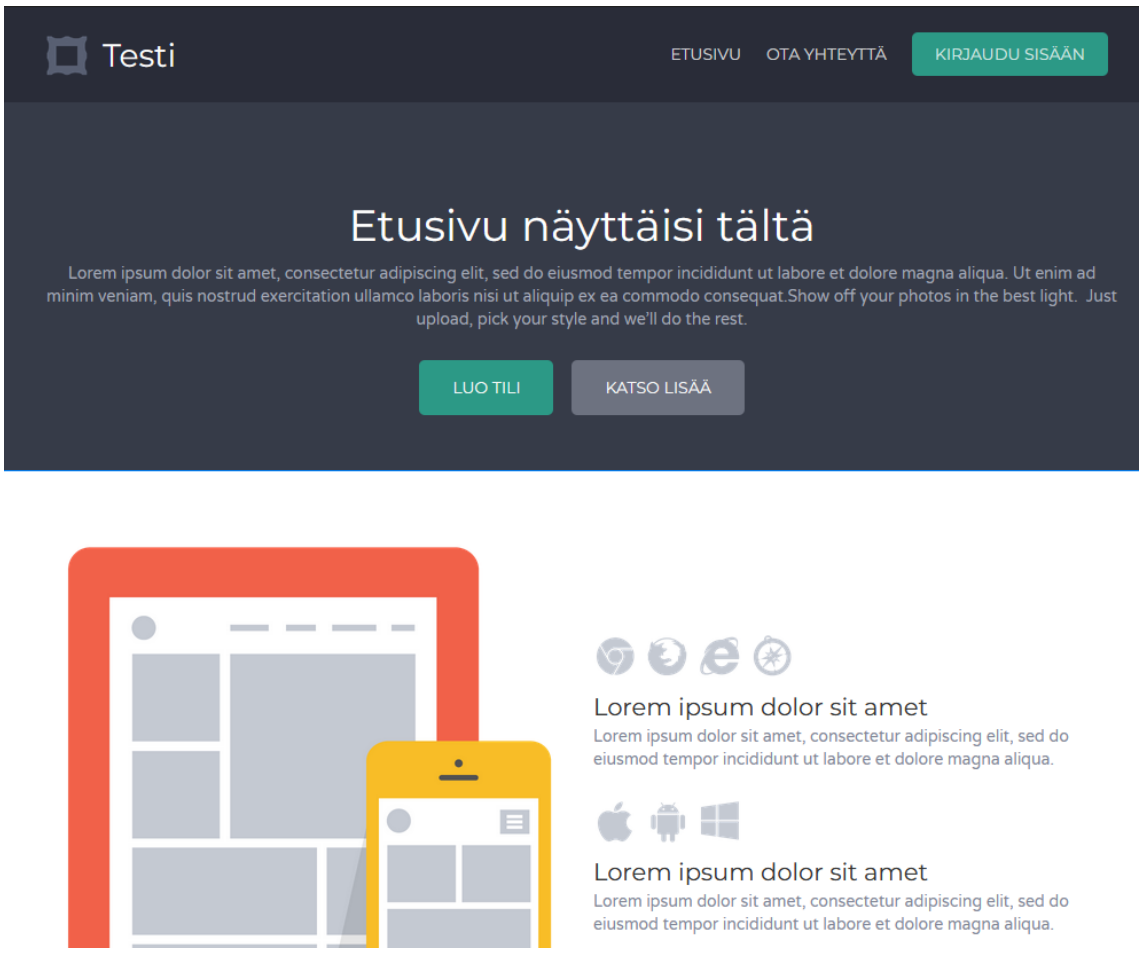


Kuvio 3. Käyttötapauskaavioesimerkki oikein ja väärin tehdystä kaaviosta, esimerkkinä käyttäjän, selaimen, palvelimen ja ylläpitäjän välinen vuorovaikutus (Heywood 1999)

3.4 Prototypointi

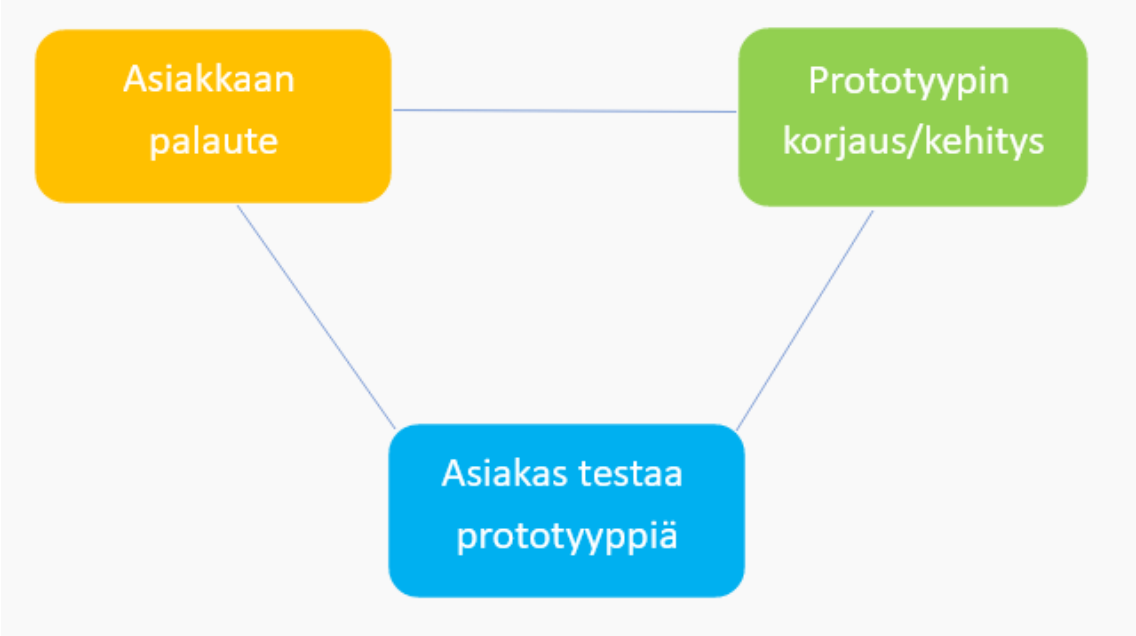
Prototypoinnin avulla pystytään luomaan simulaatio tulevasta tuotteesta. Prototyypin avulla pääsee tutustumaan, miltä valmiin tuotteen käyttäminen alkaisi näyttämään, toimimaan ja tuntumaan. Prototyyppi voi olla esimerkiksi valmis tuote ilman toiminnallisuuksia. Prototyyppejä ei käytetä ainoastaan tuotteen vaatimusmäärittely- vaiheessa. Ohjelmasta voidaan tehdä ihan toimivakin prototyyppiversio ja antaa se loppukäyttäjille testattavaksi. Sen avulla pystytään kartoittamaan ohjelman menestymistä sekä saamaan suoraan loppukäyttäjiltä palautetta kehityskohteista. (Mistry 2021.)

Prototypoinnin malleja on monia erilaisia. Yksinkertaisimmillaan prototyyppi voi olla paperille piirretty suunnitelma ohjelman eri näkymistä. Nykyään on kuitenkin helppo toteuttaa myös prototyyppejä, missä onnistuu esimerkiksi nappien painaminen ja näkymien vaihto. Esimerkiksi kuvio 4 kuvastaa käyttöliittymäprototyyppiä, missä onnistuu myös näkymien vaihto. Halutessaan voi tehdä myös eri osa-alueista omia prototyyppejä ja yhdistää ne sitten lopuksi yhdeksi kokonaisuudeksi. (Sharma 2022.)



Kuvio 4. Esimerkki prototyypistä käyttäen Webflow- palvelua.

Prototypoinnissa tehdään ensimmäinen prototyyppi asiakkaalta saatujen vaatimusten perusteella. Sen jälkeen annetaan prototyyppi asiakkaalle testattavaksi ja arvioitavaksi. Tällainen kierto on kuvattu kuviossa 5. Yleensä erityisesti ensimmäisiin prototyyppeihin saadaan asiakkaalta paljon kehitys- ja parannusehdotuksia. Palautteen jälkeen tehdään prototyyppiin tarvittavat korjaukset ja parannukset, jonka jälkeen prototyyppi annetaan uudestaan arvioitavaksi. Tätä toistetaan niin kauan, kunnes kaikki osapuolet ovat hyväksyneet prototyypin ja kaikki tilanteet on otettu prototyypissä huomioon. Sen jälkeen voidaan alkaa rakentamaan oikeaa ohjelmaa prototyypin perusteella. (Sharma 2022.)



Kuvio 5. Prototypoinnin vaiheet

Prototypoinnin keskeinen idea on säästää resursseja, varmistamalla asiakkaalta, että tuotteen prototyyppi vastaa tarpeita, toiveita ja odotuksia. Huomattavasti kalliimpaa ja haastavampaa on alkaa muuttamaan asioita oikean tuotekehityksen aikana, missä rakennetaan samalla myös toiminnallisuuksia. Siksi on tärkeää, että ennen oikeaa tuotekehitystä on saatu aikaiseksi hyväksytty prototyyppi minkä perusteella aletaan kehittämään oikeaa tuotetta. (Mistry 2021.)

Käyttämällä prototypointia, pystytään välttämään paljon turhaa työtä ja parantamaan yhteisymmärrystä asiakkaan kanssa. Varmin tapa viestiä, miten on ajatellut tulevan ohjelman toimivan ja näyttävän on antaa asiakkaalle prototyyppi testattavaksi. Sanoin kuvaillessa jokainen ihminen käsittää asian omalla tavallaan ja siitä voi aiheutua väärinkäsityksiä. Kun taas asiakas pääsee kokeilemaan ja testailemaan prototyyppiä, asiakas näkee ja kokee ohjelman paljon paremmin. Jos siinä kohtaa on jokin asia ymmärretty tai käsitetty väärin, sen huomaa helposti ja korjaaminen on siinä vaiheessa vielä helppoa. (Sharma 2022.)

3.5 Iterointi

Iterointi tarkoittaa toimintamallia, missä toistetaan tiettyjä työvaiheita säännöllisesti. Jotta iterointi onnistuisi toivotulla tavalla, työvaiheiden kierron tulee olla

suhteellisen lyhyt. Riippuen tilanteesta, kierron pituus voi olla esimerkiksi päivä, viikko tai kuukausi. Siihen vaikuttaa muun muassa projektin laajuus ja minkälaisia työvaiheita kierto pitää sisällään. Projektissa voi olla menossa myös samanaikaisesti monen eri laajuista iteraatiota. Esimerkiksi pienempi iteraatio tietyn ominaisuuden tai osa-alueen rakentamiselle ja isompi iteraatio koko projektin edistymiselle. (Allan 2019.)

Iterointi on erittäin hyvä tapa parantaa asiakkaan ja kehitystiimin välistä yhteisymmärrystä, kun asiakas otetaan mukaan iteraatioon. Esimerkiksi sillä tavalla, että suunnitelmat tai luonnokset esitellään säännöllisesti asiakkaalle tai pidetään säännöllisesti tilannekatsauksia asiakkaan kanssa. Silloin kaikilla on hyvä käsitys missä vaiheessa projekti on menossa. Jos huomataankin, että jokin asia on menossa väärään suuntaan tai asiaa ei ole ymmärretty oikein. Iteroinnin avulla siihen pystytään reagoimaan nopeasti ja se ei aiheuta vielä mitään merkittävää vahinkoa. (Allan 2019.)

Iterointia käytetään myös jo aikaisemmin esitellyssä prototypoinnissa. Iterointia pystyy hyödyntämään kuitenkin lähes kaikkeen ja kaikissa ohjelmistokehityksen vaiheissa. Kuviossa 4 on esitetty työvaiheet prototyypin rakentamiselle, joita toistetaan iteraatiossa. Iteraatiota toistetaan niin pitkään, kunnes on saavutettu toivottu lopputulos. Lopputulos voi olla esimerkiksi prototyyppi, mikä on saanut kaikkien osapuolten hyväksynnän. (Allan 2019.)

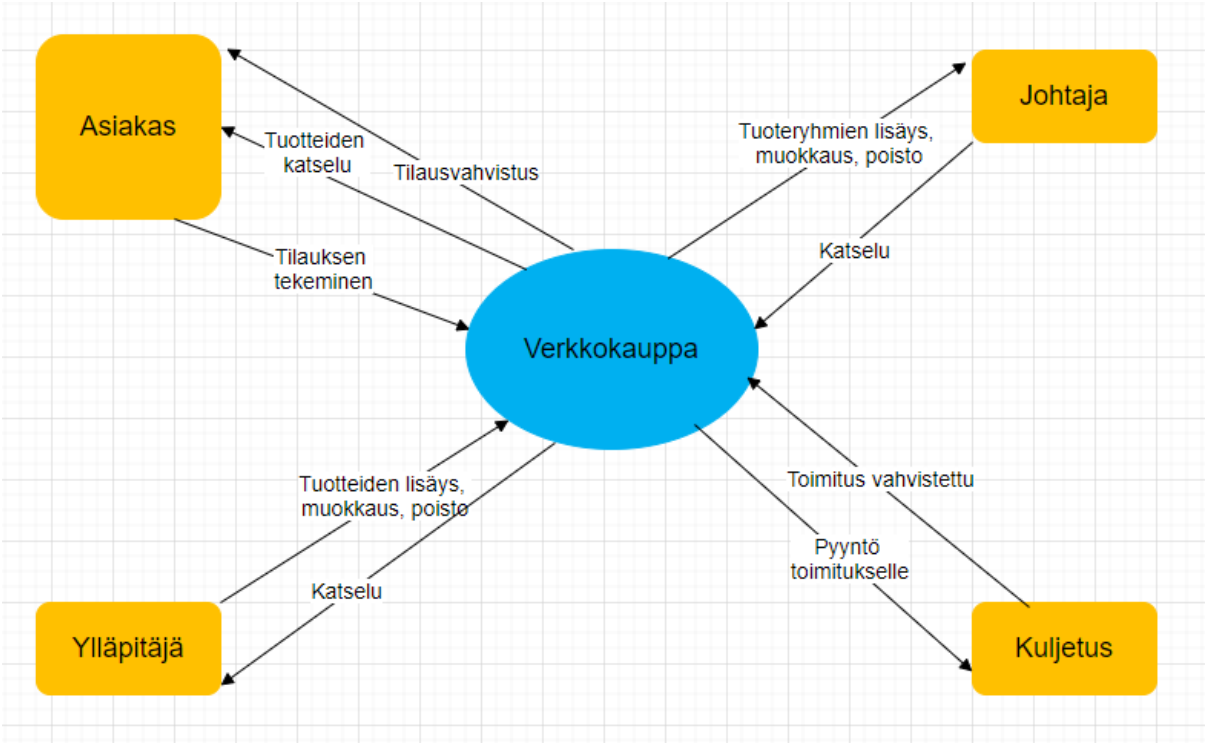
3.6 Yhteyskaavio

Yhteyskaavio (*Context Diagram*) esittää tiedonkulkua järjestelmän ja ulkoisten komponenttien välillä. Kaaviota käytetään apuna vaatimusmäärittelyssä projektin yleiskuvan hahmottamiseen. Kaaviota hyödynnetään suunnittelun alkuvaiheessa, sillä sen avulla huomaa helposti esimerkiksi puuttuvat ominaisuudet. Näin pystytään muun muassa säästämään resursseja ja parantamaan lopputuotteen laatua, kun asioita huomioidaan hyvissä ajoin. Lisäksi kaaviota hyödynnetään tuotteen kulujen arvioimiseen. (Pedriquez 2022.)

Kaavio on selkeä ja helposti ymmärrettävissä kaikille. Yhteyskaavion keskellä on kontekstikupla ja siitä aletaan purkamaan yhteyksiä eri komponentteihin. Erityyppiset komponentit erotetaan toisistaan käyttämällä eri muotoisia laatikoita.

Yhteydet merkataan viivalla ja nuolella kontekstikuplan ja komponenttien väliin. Komponentteja ovat esimerkiksi käyttäjät ja tietokannat. (Pedriquez 2022.)

Kuviossa 6 on esitetty esimerkki verkkokaupan toiminnasta. Kontekstikuplasta ilmenee, mitä kaavio koskee. Keltaisiin laatikoihin on merkattu komponenteiksi erilaiset käyttäjät eli ylläpitäjä, johtaja ja asiakas sekä ulkopuolinen palveluntarjoaja kuljetukselle. Nuolien avulla on merkitty, minkälaisia yhteyksiä verkkokaupan toiminta vaatii ja mihin suuntaan ne liikkuvat. Esimerkiksi kun asiakas on selaillut ja katsonut verkkokaupassa tuotteita, hänen pitää päästä tekemään myös tilaus. Kun verkkokauppa on vastaanottanut tilauksen, verkkokauppa lähettää asiakkaalle tilausvahvistuksen.



Kuvio 6. Yhteyskaavio esimerkki verkkokaupan toiminnasta

Yhteyskaavio on helposti ymmärrettävä kaikille vaatimusmäärittelyä käyttäville osapuolille. Kaavio on rakenteeltaan hyvin yksinkertainen eikä se sisällä ammattisanastoa tai muutenkaan hankalia termejä, sillä kaavio on suunnattu pääasiallisesti rahoittajille ja asiakkaalle kuin vaikkapa ohjelmistokehittäjille. Se antaa kattavan yleiskuvan koko projektista ja sen avulla pystytään varmistamaan, että kaikki osapuolet ovat ymmärtäneet asiat samalla tavalla. (Pedriquez 2022.)

4 HYVÄN VAATIMUSMÄÄRITTELYN VAIKUTUKSET

Hyvin laadittu vaatimusmäärittely auttaa ratkaisemaan monia kommunikaatio ongelmia asiakkaan ja ohjelmistokehittäjän välillä. Tällä tavalla voi vaikuttaa myös siihen, että pystyy vastaamaan parhaalla mahdollisella tavalla asiakkaan tarpeisiin ja toiveisiin. Vaatimusmäärittely toimii osapuolten välillä ikään kuin sopimuksena tulevasta tuotteesta. Kun vaatimusmäärittelyssä on selkeästi ja tarkasti kuvailtu tuote, pystyy varmistamaan asiakkaalta, että asia on ymmärretty oikein ja tuotteesta tulee toivotunlainen. Näin voi välttää yleisiä väärinkäsityksiä ja parantaa yhteisymmärrystä ja asiakaskokemusta. (Ashraf ym. 2016.)

Hyvässä vaatimusmäärittelyssä arvioidaan tiukasti kaikki vaatimukset ja silloin vasta aloitetaan työvaiheiden ja toiminnallisuuksien tarkempi suunnittelu. Sen avulla pystytään välttymään korjauksilta ja uudelleen suunnittelulta. Tämä auttaa säästämään myös resursseja, tekee projektin etenemisestä tehokkaampaa ja testauksesta jouhevamman. (Golodov 2016.)

Tuotteen kehitys etenee jouhevasti ja tehokkaasti kun kehittäjät tietävät hyvin projektin prioriteetit, kontekstin ja etenemisvaiheet. Siinä kohtaa pitäisi olla selkeää, mistä aloitetaan tuotteen kehitys ja mitä tehdään seuraavaksi. Näin pystytään käyttämään resurssit tehokkaasti. Hyvän vaatimusmäärittelyn avulla osataan ottaa käyttöliittymäsuunnittelussa jo aikaisessa vaiheessa huomioon mahdollisimman monia asioita ja erilaisia tilanteita. Tällä tavalla pystytään takaamaan tuotteen hyvä käytettävyys ja saavutettavuus, mikä on tosi tärkeä osa tuotteen laatua. (Golodov 2016.)

Testauksessa käytetään paljon apuna muun muassa vaatimusmäärittelyssä esiteltyjä valmiin tuotteen kriteerejä ja käyttäjätarinoita. Niiden avulla on erittäin helppoa luoda testitapauksia ja testata. Lisäksi pystytään mittaamaan helposti, että onko tuote valmis ja vastaako se vaatimuksia ja odotuksia. Tällä tavalla pystytään varmistamaan tuotteen laatu ja toimivuus. Hyvä vaatimusmäärittely on perusta laadukkaalle tuotteelle. (Golodov 2016.)

Vaatimusmäärittelyn laatimisen yhteydessä saadaan hyvä arvio myös siitä, minkä verran resursseja tuotteen kehitys vaatii. Vaatimusmäärittelyn avulla pystyy antamaan hyvin realistisen arvion esimerkiksi tuotekehityksen kustannuksista

tai projektin aikataulusta. Nämä ovat varmasti asiat, mitkä kiinnostavat asiakkaita, kun he ovat tilaamassa tuotetta. Hyvän asiakaskokemuksen tarjoamiseksi on tärkeää antaa julkaisupäivästä ja budjetista mahdollisimman realistinen arvio. (Golodov 2016.)

Hyvin laaditusta vaatimusmäärittelystä on paljon hyötyä myös, kun tuotteelle aletaan suunnittelemaan jatkokehitystä. Silloin on hyvä perehtyä tuotteen vaatimusmäärittelyyn ja alkaa sen perusteella suunnittelemaan lisäominaisuuksia tai muita muutoksia tuotteelle. Lisäksi vaatimusmäärittely tarjoaa hyvän tietoperustan uusille käyttäjille tai toimintaympäristöille. Esimerkiksi tuotteen ominaisuuksia ja käyttöä voi esitellä uusille asiakkaille vaatimusmäärittelyn avulla. (Golodov 2016.)

5 POHDINTA

Opinnäytetyön tarkoituksena oli löytää ratkaisuja ohjelmistokehittäjien ja asiakkaiden välisiin kommunikaatio ongelmiin. Kommunikaatio ongelmat ovat yleisiä ongelmia ohjelmistokehityksessä, ja ne saattavat aiheuttaa paljon kuluja, joita aiheuttaa vaatimusmäärittelyn huoleton laatiminen. Muun muassa taloudellisia kuluja, paljon ylimääräistä työtä, heikkolaatuisen lopputuotteen ja maineen heikkenemistä. Tämä kävi ilmi useista eri lähteistä, missä kuvailtiin käytännön tilannetta työelämässä ohjelmistokehityksen parissa. Opinnäytetyössä tutkittiin erilaisia vaatimusmäärittelyä tarkentavia menetelmiä, joilla pystytään parantamaan kommunikaatiota ja tekemään vaatimusmäärittelystä selkeämmän.

Tutkimuksesta selvisi, että väärinkäsityksiä asiakkaan ja ohjelmistokehittäjän välillä pystytään korjaamaan käyttämällä selkeämpiä viestintämenetelmiä. Useiden lähteiden mukaan vaatimusmäärittelyssä kannattaa hyödyntää tuotteen toimintaa ja suunnitelmaa kuvaavia visuaalisia tarkennusmenetelmiä. Niiden avulla saadaan vaatimusmäärittelystä selkeämpi ja helpommin ymmärrettävä kaikille osapuolille.

Opinnäytetyössä esiteltiin erilaisia ratkaisuja vaatimusmäärittelyn laatimisen eri vaiheissa, ratkaisemaan ongelmaa. Asiakaslähtöisen vaatimusmäärittelykäytännön käyttäminen ja vaatimusmäärittelyssä tarkentavien elementtien käyttö edesauttaa tuotteen tavoitteiden saavuttamisessa. Tarkennusmenetelmistä todettiin tutkimuksessa toimivimmiksi prototypointi ja iterointi. Kummatkin menetelmät ovat helppoja ottaa käyttöön, ja niitä pystyy soveltamaan hyvin omien tarpeiden ja resurssien mukaisesti.

Opinnäytetyössä esitetyt tiedot ovat luotettavia, sillä niitä tukee useat lähteet, joista voi tehdä yleistyksiä aiheesta. Lähdeaineistojen julkaisijat ja kirjoittajat ovat ohjelmistoalalla työskenteleviä. Käytännön ongelmia on kuvailtu nojaten ohjelmistoalalla työskentelevien asiantuntijoiden blogi- kirjoituksiin aiheesta. Tämän avulla ei ollut välttämätön kerätä aineistoa esimerkiksi haastatteluilla.

Kirjoittaja sai opinnäytetyön parissa työskennellessä hyvän teoreettisen käsityksen vaatimusmäärittelyn laatimisen prosessista ja dokumentin hyödyntämisestä ohjelmistokehityksen eri vaiheissa. Opinnäytetyö antoi kirjoittajalle paljon uutta

tietoa vaatimusmäärittelyn osa-alueista sekä vaatimusmäärittelyn, iteroinnin ja prototypoinnin tärkeydestä. Kirjoittaja tulee hyödyntämään opittuja asioita jatkossa työelämässä.

Opinnäytetyöstä hyötyvät ohjelmistoyritykset, jotka kokevat, että heillä on kehitettävää vaatimusmäärittelyn laatimisen kanssa. Tutkimus auttaa kartoittamaan kehityskohteita vaatimusmäärittelyn laatimisessa ja tarjoaa ratkaisuja parempaan kommunikointiin asiakkaan kanssa. Vaikka uskoo, että käytössä on hyvä malli vaatimusmäärittelyn laatimiselle, luulen, että jokainen löytää tästä tutkimuksesta itselleen jotakin.

Tarkennusmenetelmiä mitä käyttää vaatimusmäärittelyn tarkentamiseen on paljon. Vielä ei ole kuitenkaan työkalua mikä korvaisi perinteisen vaatimusmäärittelyn kokonaan. Jatkotutkimuksena voisi pohtia esimerkiksi iteroinnin hyödyntämistä vielä monipuolisemmin vaatimusmäärittelyn kanssa. Myös prototypointi on laaja ja mielenkiintoinen aihe missä on vielä paljon tutkittavaa.

LÄHTEET

Allan, M. 2019. What is Interactive Development – An Easy Guide for Beginners. GoodCore 25.10.2019. Viitattu 3.6.2022 https://www.goodcore.co.uk/blog/iterative-development/.

Altexsoft 2020. Technical Documentation in Software Development: Types, Best Practices, and Tools. Alexsoft 1.12.2020. Viitattu 10.4.2022 https://www.altexsoft.com/blog/business/technical-documentation-in-software-development-types-best-practices-and-tools/.

Ashraf, M. A., Aslam, M. & Awan, S. M. 2016. Causes and Impact of Slipped/Misunderstood Requirements on Large Scale Software Projects. University of Management and Technology. Viitattu 12.4.2022 http://admin.umt.edu.pk/Media/Site/icic/FileManager/Proceedings/Papers/49%20ICIC_2016_paper_49.pdf.

Bandakkanavar, R. 2018. Software Requirements Specification document with example. Krazytech 4.7.2018. Viitattu 20.4.2022 https://krazytech.com/projects/sample-software-requirements-specificationsrs-report-airline-database.

Burak, A. 2020. Your 2022 Guide to Writing a Software Requirements Specification (SRS) Document. Relevant 10.9.2020. Viitattu 12.4.2022 https://relevant.software/blog/software-requirements-specification-srs-document/#Define_the_Purpose.

Ferens, K. 2020. The ugly truth about software development process. The Codest 7.9.2020. Viitattu 3.6.2022 https://thecodest.co/blog/the-ugly-truth-about-software-development-process.

Francino, Y. 2020. User story. TechTarget. Viitattu 12.4.2022 https://www.techtarget.com/searchsoftwarequality/definition/user-story.

Golodov, S. 2016. SRS Document Helps to Protect IT Projects From Failure. Belitsoft 08.08.2016. Viitattu 20.4.2022 https://belitsoft.com/php-development-services/software-requirements-specification-helps-protect-it-projects-failure.

Heywood, R. 1999. UML Use Case Diagrams: Tips and FAQ. Www.andrew.cmu.edu. Viitattu 3.6.2022 https://www.andrew.cmu.edu/course/90-754/umlucdfaq.html.

Huether, D. 2017. Definition of Done. Leadingagile 2.2017. Viitattu 11.4.2022 https://www.leadingagile.com/2017/02/definition-of-done/.

Hurja 2021. Mitä bugeista ajatellaan ohjelmistotalossa? Hurja 30.4.2021. Viitattu 6.6.2022 https://www.hurja.fi/blogi/mita-bugeista-ajatellaan-ohjelmistotalossa/.

Jama Software 2019. Defining Project Scope: Context and Use Case Diagrams. Jama Software 24.9.2019. Viitattu 3.6.2022 https://www.jamasoftware.com/blog/defining-project-scope-context-use-case-diagrams/.

Lane, C. & Krüger, N. 2021. How to Write a Software Requirements Specification (SRS Document). Perforce 16.12.2021. Viitattu 20.4.2022 https://www.perforce.com/blog/alm/how-write-software-requirements-specification-srs-document#describe.

Lucassen, G., Dalpiaz, F., Van der Werf, J. M. & Brinkkemper, S. 2016. The Use and Effectiveness of User Stories in Practice. ResearchGate. 3.6.2022 https://www.researchgate.net/profile/Fabiano-Dalpiaz/publication/312702795_The_Use_and_Effectiveness_of_User_Stories_in_Practice/links/5b61642c0f7e9bc79a72dc21/The-Use-and-Effectiveness-of-User-Stories-in-Practice.pdf?origin=publication_detail.

Lynch, A. 2022. What is UML | Unified Modeling Language. Edraw 15.2.2022. Viitattu 7.6.2022 https://www.edrawsoft.com/what-is-uml-diagram.html.

Mistry, P. 2021. Why Software Prototyping Should Be a Crucial Part of Your Development Process. Radik 31.8.2021. Viitattu 3.6.2022 https://radixweb.com/blog/what-is-software-prototyping.

Ojala, J. 2022. Miltä saavutettavuus näyttää 2022–2025? Vincit 11.1.2022. Viitattu 7.6.2022 https://www.vincit.fi/fi/milta-saavutettavuus-nayttaa-2022-2025/.

Pedriquez, D. 2022. What is a Context Diagram (and How Can You Create One)? Venngage 6.4.2022. Viitattu 03.06.2022 https://venngage.com/blog/context-diagram/.

Santillan, M. F. & Käkölä, T. 2016. Evaluation framework for analysing the applicability of criteria lists for the selection of requirements management tools supporting distributed collaboration and software product line requirements management. 2016 49th Hawaii International Conference on System Science. Viitattu 3.6.2022 https://jyx.jyu.fi/bitstream/handle/123456789/80181/Evaluation_Framework_for_Analyzing_the_Applicability_of_Criteria_Lists_for_the_Selection_of_Requirements_Management_Tools_Supporting_Distributed_Collaboration_and_Software_Product_Line_Requ.pdf?sequence=1&isAllowed=y.

Sharma, I. 2022. What is Software Prototyping and Its Types? TatvaSoft 5.2022. Viitattu 03.06.2022 https://www.tatvasoft.com/outsourcing/2022/05/what-is-software-prototyping.html.

Viljanen, V. 2020. Käytettävyys ja käyttökokemus. Valkohattu 22.2.2020. Viitattu 6.6.2022 https://valkohattu.fi/artikkeli/kayttokokemus.

Zomko, R. 2021. Guide to Software Requirements Specification in Development. Impressit 28.10.2021. Viitattu 10.04.2022 https://impressit.io/blog/software-requirements-specification-guide.