HAAGA-HELIA
ammattikorkeakoulu

Vaatimusmääritys ALV-raportin automatisointisovelluk-selle

Kurosh Farsimadan

Opinnäytetyö
Tietojenkäsittelyn koulutusohjelma
2014

HAAGA-HELIA
ammattikorkeakoulu

Tiivistelmä

30.11.2014

Tekijä
Kurosh Farsimadan

Koulutusohjelma
HETI / Tietojenkäsittelyn koulutusohjelma

Opinnäytetyön otsikko
Vaatimusmääritys ALV-raportin automatisointisovellukselle

Sivu- ja liitesivumäärä
52 + 239

Opinnäytetyön otsikko englanniksi
SRS for automatisation of the periodic tax return declaration software

Tämän opinnäytetyön tavoitteena on selvittää ja dokumentoida kaikki vaatimusmääritykset kausiveroilmoituksen automatisoivalle sovellukselle, joka kehitetään ohjelmistokehityskurssin ulkoiselle asiakkaalle sekä tuottaa lisäarvoa vaatimusmäärityksele kartoittamalla kaikki taustatiedot, kehitettävän sovelluksen päämäärä ja tarkoitus. Opinnäytetyön teoriaosuudessa pyritään käymään läpi vaatimusmääritys työn prosessi ja eri prosessikohdan erilaiset menetelmät, standardit, mallit sekä pohjia joita voidaan käyttää vaatimusmääritys dokumentissa ja antaa kokonaiskuvan vaatimusmääritys työstä kehitettävän sovelluksen yhteydessä antamaan.

Erilaisia menetelmiä joita käytettiin opinnäytetyön kirjoittamiseen, oli lukemalla ja tutkimalla erilaisia kirjallisia ja elektronisia teoriakirjoja. Muita opinnäytetyön teoria-aineiston keruumenetelmät toteutuivat asiakastapaamisten kautta. Vaatimusmääritys dokumentin standardina ja pohjana käytetään Haaga-Helian omia materiaaleja, jotka mukailevat IEEE:n ja Voleren vaatimusmääritys dokumenttipohjan hyviä käytäntöjä ja muita kirjallisia lähteitä esim. Agile käyttötapauskuvaus ja kaavio menetelmiä, johon on sovellettu JHS-suosituksien mukaisia käyttötapauskuvaus – pohjaa.

Ohjelmiston vaatimusmäärityksdokumentaation tarkoituksena on vakiinnuttaa pohjan asiakkaalle ja sovelluksen tuottajalle, mitä lopullisen tuotteen pitäisi tehdä. Kehitettävän sovelluksen kaikkien toimintojen yksityiskohtaiset kuvaukset- ja kaaviot vaatimusmäärityksdokumentaatiossa tulevat avustamaan sovelluksen kehittäjiä, asiakkaan haluaman sovelluksen rakentamisessa.

Opinnäytetyön vaatimusmäärityksen tarkoituksena on listata kaikki tarpeelliset tiedot, jotka liittyvät kehitettävän sovelluksen tiedon kulkuun, ja tarjota kokonaiskuva kaikista sidosryhmistä ja toimijoista, taustatiedoista, toiminnollisista ja ei-toiminnollisista vaatimuksista, rajauksista, käyttötapauksista ja kehitettävän sovelluksen kehitysympäristöstä. Dokumentti on johdonmukainen, ristiriidaton ja ajantasainen asiakkaan vaatimusten kanssa.

Asiasanat
ohjelmistosuunnittelu, ohjelmoijat, vaatimusmääritys, ohjelmistotekniikka, Real-Time Economy – ohjelma, sovellusdokumentaatio

HAAGA-HELIA
University of Applied Sciences

Abstract

30.11.2014

Author
Kurosh Farsimadan

Degree programme
BITE / Business Information Technology programme

Report/thesis title	**Number of pages and appendix pages**
SRS for automatisation of the periodic tax return declaration software	52 + 239

The aim of this thesis is to research, find and document all the requirements for the automatization of the periodic tax return declaration that is being developed for the ordering client of the software development course. The aim is to bring more value to the developed software system by overviewing all the background information, the meaning and aim of the software and the requirements. The theory part of the study will also review the software requirements specification process and different process points by describing the various methods, standards and models, templates that can be used in the SRS and give a thorough picture of the SRS in the context of the software being developed.

The range of methods used within the framework of the study covered a survey of various literary sources and electronic books, meetings with the customer and HAAGA-HELIA -based resources. The standard and template used in writing the SRS documentation is the university template that follows the IEEE and Volere SRS template standards and good practices. Various supplementary sources for example, the JHS use case standards and template, agile methodologies and other types of SRS template content descriptions were also applied.

The meaning of the software requirements specification documentation is to establish the basis for agreement between the customers and the suppliers on what the software product is to do. The complete description of the functions to be performed by the software specified in the SRS will assist the potential users in determining if the software specified meets their needs or how the software must be modified to meet their needs.

The objective of the SRS in the study was to list all the necessary information relating to the data flow of the software being developed and offer a whole picture of all the factors, background, functional and non-functionalities, constraints, use cases and the development environment.

Keywords
Software engineering, programmers, software requirements specification, software architectures, Real-Time Economy –program, software documentation.

Sisällys

1 Johdanto

”If you don’t know where you’re going, any road will get you there.” (Carroll.)

Edellä mainittu sanaparsi ei ironisesti ole kovin kaukana totuudesta. Jos ei tiedetä, mitä halutaan tehdä tai mihin suuntaan halutaan viedä sovellus, niin suurella todennäköisyydellä ei myöskään päästä tavoitteisiin ja tullaan pääsemään lopputulokseen, joka on kaikkea paitsi sovelluksen tilaajan vaatimusten mukainen, mikä on myös pääsyy tämän opinnäytetyöprojektin alkuun saattamiseksi.

Opinnäytetyön idea sai alkuunsa HAAGA-HELIA ammattikorkeakoulussa suoritetun Softala I tai toisella nimellä Ohjelmistokehitys kurssin jälkeen, koska huomasin Softala I -Kurssin aikana, että hyvään onnistumiseemme vaikutti suurelta osin huomattavan laaja dokumentaatio, vaikka hyvillä kehittäjillä oli myös osansa siinä.

Dokumentaatiossamme oli kattavasti tietoa sovelluksen tarkoituksesta, sidosryhmistä, toiminnoista ja teknisen puolen yksityiskohdista ja tarpeista, ja se korostui, kun projektitiimi käytti ahkerasti vaatimusmäärityksessä tallennettuja tietoja sovelluksen suunnittelussa ja ohjelmointityössä.

1.1 Opinnäytetyön tavoitteet

Opinnäytetyön aiheena ja tavoitteena on tutkia, valmistella ja luoda vaatimusmääritys Tieto Finland Oy:n hankkeelle, jossa tavoitteena on luoda ISO 20022-standardeja huomioon ottava ja mukainen kausiveroilmoituksia esitäyttävä laskulogiikan automatisoitu demosovellus.

Demossa on casetapaus, jossa tietty yrittäjä syöttää omat tiedot ja kohdekauden, jonka kautta hän saa kaikki omat XML-muodossa olevat e-laskut, joita case tapauksessamme jokin tietty operaattori tai palveluntarjoaja on eritellyt tämän tietyn yrityksen kaikki osto- ja myyntilaskut erillisen tietovirran (esim. toisen yrittäjän laskuttaessa tai tämä yrittäjän myynti laskut) siitä lähtien, kun hän oli ottanut käyttöön e-laskutuspalvelun. Edellä mainitun jälkeen sovellus osaa muodostaa yrittäjän haluamalle kaudelle verkkolaskusanoman e-laskujen tietokenttien perusteella yrityksen esitäytetyn kausiveroilmoituksen alv-raportointiin tarvittavat tiedot. (Tieto Oyj. 3.4.2013.)

ISO 20022 Standardin käyttöönottaminen kuuluu suurempaan ohjelmaan, jonka Tieto ja Aalto-yliopisto (Kauppakorkeakoulu) käynnistivät vuonna 2006 nimellä ”Real-Time

Economy -Ohjelma” (tästä lähtien RTE) edistääkseen verkkolaskutusta Suomessa ja EU-tasolla. Itse RTE-yhteistyöverkoston suurimpia vaikuttajia ovat TEKES, Aalto-yliopisto, Tieto ja ADITRO, mutta verkoston laajuus on paljon kattavampi. Siihen kuuluu pankkeja, kuluttajia, pk-yrityksiä, tavallisia yrityksiä, akateemisia instituutioita, operaattoreita ja palveluntarjoajia, intressiryhmiä ja julkisen sektorin osastoja. (Harald 2013.)

Tekemässämme sovelluksen osa-alueessa Tieto Finland Oy (omistaja) on toimeksiantanut HAAGA-HELIA ammattikorkeakoulun Softala III -Ryhmälle kehitettäväksi ratkaisun, jolla voidaan luoda kausiveroilmoituksen esitäyttävä ohjelma, jotta ISO20022 Invoice Tax Report -Sanoma voidaan vähitellen muuttaa standardinomaiseksi. Tämän sovelluksen tarkoituksena on hakea ja tuoda yrittäjälle hänen tietyn kautensa laskut, muokata näiden laskujen sisältöä standardien mukaan ja laatia esitäytetty kausiveroilmoitus, joka voidaan lähettää oikeassa muodossa verottajalle.

Tämä sovellus on myös osa taloushallinnon runkoverkko -Projektia (TARU), joka on osittain Tekes-rahoitteinen kehityshanke ja jonka tavoitteena on kehittää taloushallinnon digitalisointia ja automaatiota edistäviä ratkaisuja. Projekti on myös kiteytynyt ICT2015 -Raportissa asetettuihin tavoitteisiin.

Vaatimusmäärityksessä tulen tietoa keräämällä ja jäsentämällä tietoa dokumentoimaan kattavasti sovelluksen kehittäjille (nykyisille ja tuleville), käyttäjille ja omistajille veroraporttisovelluksen yksityiskohtaiset tiedot siitä, mitkä ovat sen tavoite, toiminnollisuudet, funktiot, tietotarpeet, vaatimukset ja miten toiminnollisuuksien odotetaan toimivan eli dokumentaatio sisältää seuraavat asiat.

Kehitettävän toiminnan kuvaus, joka sisältää

- alv-raporttia automatisoivan sovelluksen toiminnan yleiskuvauksen
- toimijakuvauksen
- prosessin
- liittymät muihin toimintoihin
- reunaehdot toiminnan kehittämiselle
- toiminnan turvallisuuden ja laadun.

Vaatimukset ratkaisun toimivuudelle, joka sisältää seuraavat:

- toiminnalliset vaatimukset ratkaisulle
- ratkaisun yleiskuvaus
- käyttötapauskuvaukset ja -kaaviot
- luokkien kuvaukset ja kaaviot
- alustavat näyttö- ja tulostehahmotelmat
- käsittelysäännöt
- liittymät muihin järjestelmiin
- suoriutumisvaatimukset

- look & feel -vaatimukset
- käytettävyysvaatimukset
- turvallisuus ja laatu
- suunnittelun rajoitteet ja ympäristövaatimukset

Käytän vaatimusmääritys dokumentin laatimiseen eri metodeja, jotka ovat asiakastapaamisten, sähköpostien ja muistioiden kautta. Vaatimusmäärityksen laatimisessa on käytetty pohjana HAAGA-HELIA ammattikorkeakoulussa opittuja käytäntöjä, jotka mukailevat kovin läheltä ”Institute of Electrical and Electronics Engineers” -Organisaation (tästä lähtien IEEE) ja ”International Organization for Standardizatioia” (tästä lähtien ISO) laatimaa ”Software Requirements Specification” -Dokumenttia (tästä lähtien SRS).

Tämän opinnäytetyön teoriaosuudessa tulen selvittämään vaatimusmääritysdokumentaation ja -kulun rakennetta, sisältöä ja sen tuottamia hyötyjä Softala III -Demosovellukselle ja sovelluskehitysprojekteille yleensä ottaen käyttäen erilaisia menetelmiä kuten lukuisia kirjallisia ja elektronisia kirja- ja artikkelilähteitä, asiakastapaamisia ja HAAGA-HELIAN omia materiaaleja. Tulen myös käymään läpi vaatimusmääritystyön historiaa, menetelmiä ja rakennetta erittäin laajan aineiston ja aihealueen vuoksi

1.2 Sanasto

Taulukko 1. Sanasto

Käsite	**Selitys**
IEEE (engl. Institute of Electrical and Electronics Engineers)	Ammattiyhdistys, jonka tarkoituksena on elektronisten ja tietojenkäsittelytiede alojen innovaatioiden edistäminen ja olemassa olevien käytäntöjen ja standardien ylläpitäminen ja parantaminen.
ISO (engl. International Organization for Standardization)	Organisaatio, joka on vastuussa erilaisten kansainvälisten standardien hallinnoimisesta.
Vaatimusmääritys (engl. Software Requirements Specification / SRS)	Dokumentaatio, jonka tarkoitus on tarjota kehitettävän sovelluksen tarkoitusta, kehitysympäristöä ja mitä sovelluksen on tarkoitus toimia toimintokohtaisesti.
Real-Time Economy	Vuodesta 2006 alkanut ohjelma, jossa pyritään reaaliaikaisen talouden ympäristön automatisointiin, pystyttämiseen ja ylläpitoon.

ISO 20022	Standardi, jonka tarkoituksena on yhtenäistää, automatisoida, helpottaa ja nopeuttaa finanssialan instituutioiden sisäisiä ja välisiä prosesseja.
XML (engl. Extensible Markup Language)	Metadatatiedosto, jossa standardoidaan lähetettävän tiedon rakennetta helpottamaan suurien tekstimäärien lähettämistä.
Verkkolasku	Verkkolasku on sähköinen lasku. Sen tiedot voidaan käsitellä automaattisesti ja esitystapa voi olla tavallisen laskun näköinen tietokoneen näytöllä.
Kausiveroilmoitus	Kausiveroilmoitus on lomake, jonka yritys- ja yhteisöasiakkaat ja yleisesti arvonlisäverovelvolliset ja säännöllisesti palkkaa maksavat työnantajat joutuvat antamaan myös, kun on peritty ennakonpidätyksiä tai lähdeveroja.
Maria DB	MySQL Relaationaalisten tietokantojen hallintajärjestelmän tapainen, yhteisön kehittämä samantapainen järjestelmä.
Sovellus	Sovellus tarkoittaa tietokoneohjelmaa ja siihen liittyvää dokumentaatiota.
XSD	XML Schema Definition eli XML tiedostolle määrritelty skeema, jota XML tiedoston tietosisältö pitäisi noudattaa.
Käyttötapaus kaavio (engl. Use Case Diagram)	Kaavio, jossa on käytetty tikkuukkokaaviota. Siinä on palikka keskellä extend-selityksillä. Otetaan huomioon koko järjestelmän arkkitehtuurimallinnus yhteen kaavioon.
Käyttötapaus kuvaus (engl. Use Case Story)	Kirjataan järjestelmän eri käyttötapaukset esim. “Järjestelmänvalvoja haluaa poistaa käyttäjän”.
Tietokanta kaavio (engl. Database diagram)	Tietokannan yleisarkkitehtuurin kuvausta UML-kaavion kautta esim. Visio-Ohjelmalla, jossa on normalisoitu tietokanta 3 NF-muotoon.

Tietokanta kuvaus	Tietokannan kuvaus, esim. nimi, tarkoitus, attribuuttien nimet ja suhteet jne.
Järjestelmän käyttäjät / Sidosryhmät (engl. Actors)	Ei tarvitse olla henkilö vaan voi myös olla toinen käyttäjä tai jokin järjestelmän komponentti, esimerkiksi sähköpostipalvelin. Sidosryhmillä on nimensä mukaisesti jotain yhteyttä kehitettävässä sovelluksessa. Sidosryhmänä voi toimia joku integroitu palvelu/järjestelmä, pääkäyttäjistä riippumattomat hallinnon puolen käyttäjät, tuotteen omistaja, tuotteen ostaja tai muuten vain sellaiset henkilöt tai instituutiot tai järjestelmät, jotka jotenkin vaikuttavat siihen. Sidosryhmät vaikuttavat projektin päämäärään, joka vaikuttaa rajaukseen, joka taas määrittää sidosryhmät.
Softala III	HAAGA-HELIAN järjestämä kolmas ohjelmistokehityskurssi, jossa opiskelijat tutustetaan oikean elämän toimeksiantajaan, joka voi olla yritys, julkishallinto tai HAAGA-HELIA.
Demo	Demon tai toisin sanoen demonstraation tarkoituksena on vain havainnollistaa tietyn tuotteen, toiminnollisuuden tai palvelun ominaisuutta tilanteen ulkopuolella.
EDI-lasku	EDI on lyhennetty sen virallisesta EDIFAC – lasku nimestä. 1980-luvulla kehitetty kahden suuryrityksen väliseen hankintasanoman sähköistämiseen ja automatisointiin luotu tietojärjestelmä.
Scrum	Scrum on tietynlainen projektinhallinnan viitekehys miten, tiimi voi yhdessä kehittää jotain tiettyä tuotetta oli kyseessä suuri ja monimutkainen projekti, joka vaatii tehokasta tiimityöskentelyä tai pienempi projekti.
Liiketoiminnan käyttötapaus	Liiketoiminnan käyttötapaus tarkoittaa,

	jotain yleisesti kuvattua vaatimusta liike-toiminnan ongelmaa varten, josta saadaan yhdestä moneen käyttötapaus kuvausta.

2 Kehitettävä sovellus

2.1 Taustatietoa

Opinnäytetyön vaatimusmäärityksen sovelluksen tarkoituksena on lähettää asiakkaan laskuista saatu kausiveroilmoitus tietyltä kaudelta verottajalle. Sovelluksen pääpaino on automaattisessa laskentalogiikassa, ja se on ns. ”Proof of Concept” eli havainnollistamiseen tarkoitettu ohjelma.

Sovelluksen tarkoituksena on siis olla ALV-raporttia automatisoiva sovellus (lyhennyksenä ARA), joka kehitetään Tieto Finland Oy:n organisaatiolle. Sovelluksesta tulee hyötymään monet eri tahot, koska kyseessä on Real-Time Economy -ohjelman jälkeinen taloushallinnon runkoverkko -ohjelma (TARU), jossa pyritään yhtenäistämään Suomessa manuaaliset taloushallintoon liittyvät työt eli kehittämään taloushallinnon digitalisointia ja automaatiota edistäviä ratkaisuja sekä selvittää, miten taloustietoa voidaan entistä paremmin hyödyntää organisaatioiden johtamisessa.

TARU on Tekes-rahoitteinen kaksivuotinen kehityshanke, jonka piiriin myös tämä projekti kuuluu. Kehityshankkeen toteuttajina on neljä taloushallinnon ja IT-alan yritystä: Aditro Oy, Administer Oy, Tieto Finland Oy ja Kunnan Taitoa Oy. Tutkimuskumppaneina ovat Aalto-yliopisto ja HAAGA-HELIAN ammattikorkeakoulu.

Sovelluksen lopulliset asiakkaat ovat rahoituslaitokset, julkiset hallintoelimet, yliopistot ja ammattikorkeakoulut, yritykset ja yhteisöt jne. Kaikki tiedossa olevat sidosryhmät eivät siis ehkä vielä ole tiedossa.

Sovelluksen toiminta on demoversiona kykeneväinen hakemaan yrityksen laskuja (osto- ja myyntilaskut) tietyltä kaudelta, muokkaamaan laskuille laskurivejä erilliseen tiedostoon esim. käteisostoja varten, laatimaan esitäytetyn kausiveroilmoituksen ja tallentamaan käyttäjän koneelle kausiveroilmoitusraportin ja ilmoittimesta läpi menevän XML-metadatatiedoston.

2.2 Finvoice

Finvoice eli e-laskusanomaa voidaan käyttää laskutuksessa ja muissa erilaisissa liiketoimintasanomissa, esim. tarjousten, tilausten, tilausvahvistusten, hinnastojen ym. liiketoiminnan kannalta suurivolyymissä lomakkeissa ja papereissa. (Finanssialan keskusliitto 13.7.2012, 1-10.)

Sanoma soveltuu kaikenkokoisille yrityksille ja näiden välisiin laskutuksiin sekä kuluttaja-laskutuksiin, käyttöönoton helppouden ansiosta. (Finanssialan keskusliitto 13.7.2012, 1-10.)

Finvoice-sanoma käyttää XML-ohjelmointikieltä, jossa päätarkoituksena on laskun esittäminen sekä sovelluksen ymmärtämässä muodossa että selaimella paperilaskua vastaavassa muodossa. Finvoice on siis myös laskusanoma, joka kuvaa laskun tietosisällön rakenteellisessa muodossa. (Finanssialan keskusliitto 13.7.2012, 1-10.)

XML on puurakenteinen hierarkkinen tiedosto, jossa meillä on Matryoshka-lelun muotoinen tietosisältö elementeittäin. Siinä on sovellukselle helposti muokattavissa oleva rakenne, esim. haku-, syöttö-, päivitys- ja poistotoimintoja varten. (Finanssialan keskusliitto 13.7.2012, 1-10.)

XML:n hyötyihin myös kuuluu, että käyttäjälle saadaan helposti HTML-muotoon saatava tietosisältö esim. laskun tiedoista. Selaimella HTML tai XML muodossa oleva lasku voidaan tulostaa paperilaskuksi ja käsitellä perinteisellä tavalla esim. ostoreskontraan ja pankkimaksusovellukseen syötettäessä. (Finanssialan keskusliitto 13.7.2012, 1-10.)

Ensimmäinen Finvoice-sanomaversio julkaistiin vuoden 2003, toinen vuonna 2004 (versio 1.1) ja kolmas vuoden 2005 alussa (versio 1.2). Finvoice-versiot ovat alaspäin yhteensopivia, vaikka toisessa versiossa kentille ei ole määritelty maksimipituutta. Versio 1.3 julkaistiin vuonna 2008 ja 2.0 vuonna 2012 keväällä. Pankit ottivat sanoman käyttöönsä oman ajansa mukaan, ennen vuoden 2013 tammikuuta. (Finanssialan keskusliitto 13.7.2012, 1-10.)

Finvoice-standardin käyttöönottamisessa pitää muistaa noudattaa Finvoice-skeemaa eli ei mitään ylimääräistä tai vähemmän tietoa kuin on määritelty skeeman (engl. XML Schema Definition tai XSD) tiedostossa.

Käsiteltäessä Finvoice- ja e-laskuja pitää myös muistaa, että kummatkin ovat Finanssialan Keskusliiton rekisteröimiä tavaramerkkejä. Edellytyksenä käytölle on se, että välityspalvelu on kaikilta osin Finvoice-välityspalvelukuvausten ja -ehtojen mukainen. E-laskulla tarkoitetaan kuluttajalle osoitettua Finvoice–laskua.

2.3 Taloushallinnon runkoverkko – projekti (TARU)

Taloushallinnon runkoverkko -projekti eli TARU on osittain Tekes-rahoitteinen (kaksivuotinen) kehityshanke, jonka piiriin kuuluu tämän opinnäytetyön liitteenä oleva sovelluksen vaatimusmääritys. (Ainasvuori 2014.)

Toteuttajina on neljä yritystä eli Aditro Oy, Administer Oy, Tieto Finland Oy ja Kunnan Taito Oy. Tutkimuskumppaneina ovat Aalto-Yliopisto ja HAAGA-HELIA ammattikorkeakoulu, jossa tämän opinnäytetyön liitteenä oleva vaatimusmääritys laadittiin ja sovellus kehitettiin. (Ainasvuori 2014.)

Runkoverkko-projektin tavoitteena on kehittää taloushallinnon digitalisointia ja automaatiota edistäviä ratkaisuja sekä selvittää, miten taloustietoa voidaan entistä paremmin hyödyntää organisaatioiden johtamisessa. Lisätavoitteena on löytää automaatioita edistäviä ratkaisuja viranomaisraportointeihin, esimerkiksi opinnäytetyön aiheena olevan vaatimusmäritysdokumentin sovellus, joka liittyy kausiveroilmoituksen automatisointiin. Projekti kytkeytyy ICT2015-raportissa asetettuihin tavoitteisiin ja mm. kansallisen palveluväylän kehittämiseen. (Ainasvuori 2014.)

Opinnäytetyön aiheena oleva projekti on yksi osa ICT2015-kehityspolku toimenpiteistä, jossa TARU kuuluu ”Polku 2” osioon, joka keskittyy yritysten reaaliaikaisen talouden vaatiman infrastruktuurin kehittämiseen.

2.4 Verkkolasku

Verkkolasku eli sähköinen lasku (toisella nimellä e-lasku) on tiedosto, jonka tietoja voidaan käsitellä automaattisesti ja josta voidaan tulostaa tai tuottaa tietokoneen näytölle paperilaskua muistuttava näkymä. Verkkolaskun vastaanottajana voivat olla yritykset, julkiset hallintoelimet tai kuluttajat.

Verkkolaskujen ja sähköisten laskujen välillä ei ole mitään eroa, mutta sähköisestä laskusta puhuttaessa pitää ottaa huomioon, että siihen voi sisältyä kuluttajaverkkolasku, verkkopankkiliitin, EDI-laskut, sähköiset kirjeet ja sähköpostilaskut.

Verkkolaskusta kun taas puhutaan, pitää ottaa huomioon, että sen tunnusomainen piirre on automaattisuus tai toisin sanoen automaattisesti käsiteltävissä olevat tiedot.

Yritysten välisissä laskutuksissa kaikki verkkolaskut voidaan siirtää automaattisesti laskuttajan tai palveluntarjoajan järjestelmästä vastaanottajan taloushallinnon järjestelmään tai muuhun samaantyyppiseen järjestelmään.

Verkkolaskuille on nykyään tunnusomaista, että ne ovat XML-muodossa lähetettäessä järjestelmästä toiseen (automaattisuus), koska skannattuja paperilaskuja voidaan harvoin automaattisesti käsitellä sovelluksissa, sillä niiden tiedot pitää syöttää manuaalisesti tiettyyn taloushallintoa ylläpitävään järjestelmään.

Verkkolaskujen vastaanoton minimivaatimuksena on internetyhteys ja jokin päätelaite, jossa on selain, ja sopimus verkkopankin tai vastaavanlaisen laskuja kokoavan palvelun kanssa, johon laskut halutaan vastaanottaa. Ilman internetyhteyttä ei voida hakea laskuja automaattisesti ja reaaliaikaisesti, ja ilman palveluntarjoajaa ei voida käsitellä laskuja.

Verkkolaskuja voivat vastaanottaa sekä yksityiset ihmiset että yritykset, vaikka laskujen vastaanotto tapa voi suurella todennäköisyydellä olla erilaista näiden välillä esim. yrityksillä yleensä on taloushallinnon järjestelmä, joka voi vastaanottaa laskuja omaan järjestelmään tiettyjen palveluntarjoajien kautta kun taas kuluttajat saavat verkkolaskunsa joko sähköpostitse tai vastaavanlaiseen kevytrakenteiseen sovellukseen.

Verkkolaskutuksen hyödyt tulevat automaattisesti käsiteltävissä olevissa kentissä, joita järjestelmät osaavat syöttää ja käsitellä ilman käyttäjien manuaalisia syöttöjä ja niiden ympäristöystävällisyydestä unohtamatta kustannussäästöjä.

Laskujen lähetys tapahtuu yleensä laskutussovelluksesta tai selaimissa toimivissa palveluissa. Laskutussovellukset tai selaimissa toimivat palvelut voivat olla samoja taloushallinnon järjestelmiä tai erillisiä palveluita.

2.5 Kausiveroilmoitus ja sen automaatio

Kausiveroilmoitus on lomake, joka on pakollinen kaikille arvonlisäverovelvollisille ja säännöllisesti palkkaa maksaville työnantajille sekä satunnaisille työnantajille silloin, kun he ovat maksaneet palkkoja, perineet ennakonpidätyksiä tai lähdeveroja.

Kausiveroilmoituslomake on tällä hetkellä sähköisesti- ja paperiversiona lähetettävä versio, johon yrittäjän tai verovelvollisen pitää täyttää manuaalisesti käymällä läpi omasta kirjanpidostaan kauden laskuista lasketut tunnusluvut. Kirjanpidon kaikki arvot tulevat siis kaikista kauden laskuista ja niiden laskutoimituksista, eli mitään sellaista tunnettua auto-

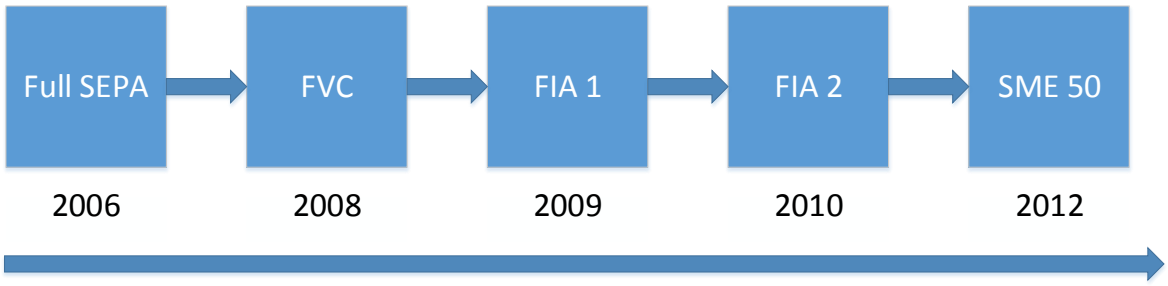
matisoitua järjestelmää ei ole vielä olemassa, joka osaisi hakea automaattisesti yrittäjän laskuja ja lähettää kausiveroilmoituksen verottajalle.

Kausiveroilmoituksen täydellinen automaatio olisi sovellus, joka osaisi hakea yrittäjän kaikki e-laskut tietystä paikasta, käsitellä ne ja laatia esitäytetyn kausiveroilmoituksen ja lähettää sen verottajalle oikeassa muodossa jos yrittäjä unohtaa lähettää lomakkeen ajoissa.

2.6 Real-Time Economy –ohjelma

Reaaliaikainen talous (engl. Real-Time Economy) tarkoittaa kansantalouden infrastruktuuria, jossa kaikkien organisaatioiden, julkisten toimielinten ym. instituutioiden väliset transaktiot ovat digitaalisessa muodossa. (Aalto-yliopisto 2014.)

Reaaliaikainen talous myös tarkoittaa, että kaikki tai melkein kaikki prosessit ovat automatisoituja ja toteutuvat reaaliaikaisesti. Kyseessä ei siis ole jaottelu liiketoiminnan ja IT-prosessien kesken vaan kaikki huomioon ottaen. Alla on lueteltu eri ohjelman vaiheet. (Aalto-yliopisto 2014; Toikka 212, 4.)



Kuva 1. Real-Time Economy -ohjelman vaiheet

Yllä olevasta kuvasta näkyy eri vaiheet, joiden selitykset ovat seuraavat:

- Full SEPA (engl. Single European Payment area): Keskittyi SEPA maksuihin ja sähköiseen laskutukseen.
- FVC (Full Value Chain): Kehitti talouden arvoketjujen sanomien harmonisointia.
- FIA 1 (engl. Fully Integrated Accounting 1): Keskiössä yritysten tiliöinnin, raportoinnin ja talouden arvoketjujen sähköistäminen.
- SME 50 (engl. Administrative costs in half): Automatisoidut hallintojärjestelmät.

Reaaliaikainen talous tarkoittaa selkokielellä sitä, että yritykset, kuluttajat ym. sidosryhmät saavat automaattisesti ja lyhyellä aikavälillä kaikki tilaukset, tilausvahvistukset, laskut ym. sähköiset dokumentit, taulukot ja tiedostot nopeasti järjestelmästä toiseen.

Suomessa Real-Time Economy ohjelma alkoi noin vuonna 2006. Ohjelma on Tekes-pohjainen ja erittäin laaja. Keskeisintä on Suomen kilpailukyvyn nostaminen uudelle asteelle. RTE vähentää yrityksien hallintokuluja ja harmaata taloutta automatisointia, sähköistämistä ja rajapintojen käyttämisen kautta, jolloin yritysten kilpailukyky parantuu ja Suomen ICT-osaaminen kasvaa. (Aalto-yliopisto 2014; Harald 2013, 8 – 11.)

RTE:n suurimpiin hyötyihin siis kuuluu lukematon määrä automatisoituja järjestelmiä, joissa aikaisemmin oli paljon manuaalista työtä, vaikka tietokoneita ja sovelluksia oli runsaasti käytössä eri julkisissa instituutioissa ja yrityksissä. Toinen hyöty on se, että voidaan siirtyä sähköiseen arkistointiin, sähköiseen kirjanpitoon ja automatisoituun taloushallintoon ja kirjanpitoon. Kolmas hyöty on kustannustehokkuus logistiikka- ja materiaalitasolla unohtamatta yhteiskunnalle kokonaisuudessaan saadut tuottavuushyödyt. (Harald 2013, 8 – 11.)

Sivuhyötynä ovat laadukkaat ja helppokäyttöiset palvelut. Vanheneva ja nuorempi väestön vapautus tuottavampaan työhön ja luodaan mahdollisuuksia pääkaupunki seudun yrityksille, esim. hallinnollisten kustannusten vähentyminen ja reaaliaikainen seuranta.

RTE-ohjelma on Tiedon, Aalto-yliopiston ja Aditron yhteinen kehityshanke. Osaamiskeskus toimii Aalto-yliopiston kauppakorkeakoulun puolella. Real-Time Economy – ohjelman ulkoiset taustatekijät olivat Euroopan komission käynnistämä ohjelma hallinnollisen taakan vähentämiseksi noin 25 % vuoteen 2012 mennessä.

2.7 Ilmoitin

Ilmoitin on verohallinnon tarjoama palvelu, jossa kehitteillä oleva tai valmis sovellus voi tarkistuttaa julkisten asiakirjojen oikeellisuuden, jotta järjestelmä voitaisiin integroida Verohallinnon ym. ulkoisten palveluntarjoajien järjestelmiin.

Järjestelmän perusperiaatteena on, että liikkeen- ja ammatinharjoittajat ja maa- ja metsätalousyrittäjät voivat lähettää omia ilmoituksiaan ilmoittimen kautta kirjautuneena.

Ilmoitin on siis, sekä yrittäjille että kehittäjille suunnattu palvelu, jossa kehittäjät voivat lähettää ja integroida ohjelmistojen tuottamia ilmoituksia, selata lähetettyjä ilmoituksia ja testata ilmoitustiedoston oikean muodon.

2.8 Yhtenäinen euromaksualue (SEPA)

Yhtenäisen euromaksualueen muodostamisen syihin kuuluu EU maiden välisten kauppojen lisääntyminen, jolloin vähitellen alkoi muodostua tarve yhtenäiseen ja koko euroalueen kattavaan yhteiseen talousalueeseen. Tällöin kaupankäynti ja tuotteiden siirtäminen maasta toiseen helpottuu, standardoituu ja nopeutuu. Tämän euromaksualueen nimitys on SEPA (Single Euro Payments Area). (Euroopan Keskupankki 2006, 5-6.)

EU-maiden säännösten standardoituminen mahdollistaa helpompaa ja nopeampaa maksujen maksamista ja siirtoa. Standardoitumisen hyötyihin kuuluu myös yksinkertaisempia erilaisia maiden välisiä maksuprosesseja. Standardoitumisen esimerkkejä ovat seuraavat:

- ISO20022, joka toimii tilisiirtojen maksustandardina. Siinä tilinumerot ovat kansainvälisessä IBAN-muodossa.
- Suomen sisäinen suoraveloitus päättyi vuoden 2014 alussa. Suoraveloituksen tilalla toimii nyt e-lasku, suoramaksu tai perinteinen paperilasku. Suoraveloituksen käyttö alkoi vuonna 2009 SEPA-ohjelman myötä.
- SEPA-maksukorteissa on siru, jossa käytön aikana allekirjoittaminen on korvattu tunnuksen syötöllä.

SEPA-Euroalueeseen kuuluu 31 maata, joihin kuuluu myös Suomi. SEPA-alueen kaikki pankit kuuluvat ja ovat mukana SEPA- ohjelmassa lukuun ottamatta investointipankkeja. Kaikki SEPA-palvelut ovat euromääräisiä (huom. Single Euro Payments Area). (European Payments Council 2014, 1-3.)

Tavalliselle kansalaiselle SEPA ei tule maksamaan mitään, mutta se tulee helpottamaan rahan nostamista, lähettämistä ja käyttämistä kaikkialla euroalueen sisällä olettaen, että kyseessä on EU:n kansalainen.

SEPAn käyttöönotto on säästänyt kaikille sen sidosryhmille noin 22 miljardia euroa per vuosi Cap Gemini tutkimuksen mukaan, joka alkoi vuonna 2008. Kokonaisuudessaan säästöjen arvioidaan olevan 123 miljardin suuruusluokkaa. (Euroopan komissio 2014.)

3 Vaatimusmääritys

3.1 Taustatietoa

Vaatimusmääritys (engl. Software Requirements Specification) eli määrittely vaatimuksille tarkoittaa sovelluskehitysorganisaation tulkintaa kaikista asiakkaan järjestelmän vaatimuksista, kuten järjestelmän tarkoitusperistä ja tavoitteista, integraatioista, toiminnoista, sidosryhmistä, säätelevistä laeista, rajoitteista, järjestelmän toimintaympäristöstä jne, eli mitä tuotteen pitäisi tehdä ja mitä sellaisia ominaisuuksia sillä yleisesti ottaen on, joita asiakas on halunnut tuotteen sisältävän. (Le Vie 2010; Wiegers & Beatty 2013, 16; Sommerville 2011, 9; Robertson's 2013, 9.)

Vaatimusmääritys dokumentaation tarkoitus kehitettävälle sovellukselle on helpottaa tuotteen ylläpitoa tekemällä projektin tekemisestä ja ylläpitämisestä kustannustehokkaan. Vaatimusmääritys-dokumentti myös pidentää kehitettävän sovelluksen elinkaarta tarjoamalla omistajille ja sen tuleville tai olemassa oleville kehittäjille kokonaiskuvan sovelluksen toiminnoista, luokista, säilytettävistä tiedoista, vaatimuksista ynnä muista tarpeellisista tiedoista, esim. eri toimintojen kytköksistä toisiinsa ja tietokantaluokkien suhteista ym. asioista kuten testauksista ja niiden tuloksista.

Vaatimusmääritykset ovat olennaisia osia jokaisessa sovelluskehitys projekteissa, koska ne tuottavat myös lisähyötyä kehittäjätiimille auttamalla projektipäällikköä ja tiimien vetäjiä hallitsemaan projektin kulkua oli kyseessä Scrum tai muita projektinhallinta viitekehyksiä käyttävä tiimi.

Vaatimuksen määrittelyssä kerätään tietoa tuotteen omistajilta ja käyttäjiltä halutun sovelluksen tarkoituksesta, jotta voidaan tietää, mitä järjestelmän pitäisi tehdä**.** On olemassa myös eri tieteenalojen määritysmetodeja ja tapoja, eri insinöörialoilla ja jokaisen eri insinööritieteen määrityksissä on erilainen rakenne, mutta perusidea on sama, vaikka ohjelmistokehitys voidaan sisällyttää yhdeksi insinööritiedealaksi.

Tapa miten voidaan kiteyttää mitä vaatimusmäärityksellä tehdään, on samaistaa dokumentaatio johonkin karttaan, jolla navigoidaan ja ohjataan työn kulkua ja tiedetään missä tilanteessa projektissa ollaan eli siis se on tapa saada vaatimukset oikein heti ensimmäisellä kirjauksella, jotta projektia ei muutettaisi ainakaan paljoa sen ollessa epätäydellinen vaatimusmäärityksen osalta. Karttaan myös merkitään uudet löydetyt alueet, eli kartoitetaan samalla, kun edetään. (Hull & Jackson & Dick. 2011, 1-2.)

Määrityksen laatimisessa ei ole olemassa tiettyä pakollista ja kokonaista standardia, mutta on ns. hyviä käytäntöjä ja malleja, esimerkiksi IEEE:n 830–1984 dokumentti ja sen jälkeen julkaistu uudempi versio nimeltään kuin 830–1998, joka ei ole pakollinen eikä ajantasainen mutta toimii yhtenä viitteenä monissa sovelluskehitys projekteissa, koska mallista on tehty monia erilaisia kansainvälisiä versioita ja sen pohjalta on luotu uudempia pohjia. Tässä opinnäytetyössä yritetään käyttää ICT-alan suomenkielisiä termejä englanninkielisestä käännöksestä. Englanninkielinen vastine on viitteenä suluissa, jotta lukija saa tietyn hahmotelman vaatimusmäärityksdokumentissa käytettyjen termien tarkoituksesta käytännössä. (huom. IEEE 830-dokumentit ovat maksullisia). (Karl 2007.)

Käyttämällä vaatimusdokumenttia voidaan välttää tai vähentää erilaisia vastakkaisia näkemyksiä, jolloin kaikilla on jonkinlainen yhtenevä kokonaiskuva sovelluksen eri funktioista ja siitä, miltä se tulee näyttämään asiakkaan, ylläpitäjän ja muiden järjestelmän käyttäjien osalta. Vaikka jos näkemykset vaihtuisivat myöhemmin, niin mahdollisimman selkeällä dokumentaatiolla nämä voidaan integroida kehitettävään ratkaisuun mahdollisimman kitkattomasti.

Huonolla projektin suunnittelulla ja vaatimusten määrittelemättömyydellä tai määrittelyllä esim. asiakkaan vaatimuksien vääränlainen määrittäminen voi yleensä johtaa katastrofaalisiin seurauksiin. Vaatimusten virheellisellä määrittelyllä esim. asiakastapaamisissa voi johtaa siihen, että ei saada kokonaiskuvaa ja täten ei voida jakaa kehitettävän sovelluksen toimintoja eri käyttötapauksiin ja ei voida tietää järjestelmän tietotarpeita. Tämä voi vaikeuttaa projektin eri toiminnollisuuksien jakamista projektia kehittäville ryhmille, ja työnjako voi muutenkin olla täynnä kitkoja, jotka voivat aiheuttaa projektin kaatumisen, koska tiimiä ja projektin töitä ei voida hallinnoida sen vaatimalla tavalla. (Gulla 2012.)

Kunnollisten vaatimusmääritysten puuttuminen on myös osoittanut monesti ohjelmistokehitysyritysten historiassa, että menestyksellinen tausta ei takaa menestyvää ohjelmistoa tulevaisuudessa ilman kunnollista toiminnollisuuksien kartoitusta. Epärealistiset tavoitteet, epäyhtenäinen näkemys asiakkaiden ja muiden sidosryhmien kanssa ja kehno vaatimusten määrittely ovat aina osoittaneet huonoja tuloksia asiakastapaamisissa ja melkein aina johtavat projektin kokonaisuuden hallinnan menetykseen ja lopulta tuotteen hylkäämiseen, oli kyseessä viimeisintä tyyppiä oleva hävittäjäkone tai kassajärjestelmä. (Charette 2005.)

Vaatimusmäärityksiin ei yleensä sisällytetä liian teknisiä yksityiskohtia, mutta niissä on juuri tarpeeksi tietoja sovelluksen kehittäjille, tiimivetäjille ja projektipäällikölle, jotta koko projektitiimillä olisi tarpeeksi oikea näkökulma kehitettävään sovellukseen. (Hull & Jack-

son & Dick. 2011.)

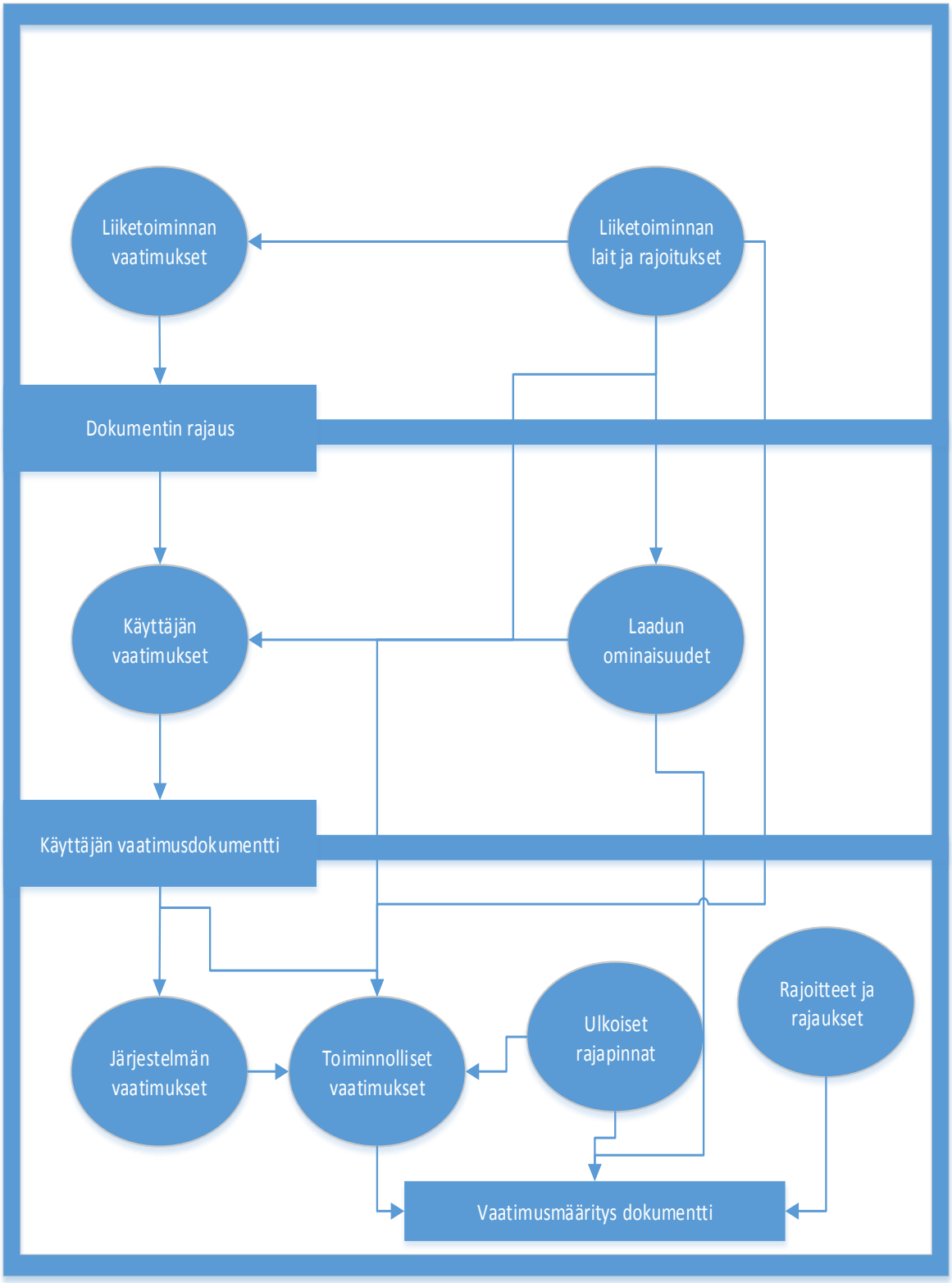
Vaatimusmäärityksen (tai sen tapaisten dokumenttien) tarkoituksena sovelluskehitysprojekteissa on ensin määritellä ja kirjata ylös asiakkaan ongelma ja tarve, joka pitää tyydyttää, esim. jokin liiketoiminnan prosessin alue, joka pitää automatisoida. Sitten pitää avata ongelma ja jaotella se pienempiin hallittavampiin osiin ja kokonaisuuksiin johdonmukaisella tavalla. Liiketoiminnan vaatimukset ovat syitä miksi tietynlaista järjestelmää ylipäätänsä halutaan rakennuttaa. (Wiegers & Beatty 2013, 8.)

Määrityksen teon ei aina tarvitse olla saman kehittäjäyrityksen laatima, vaan se voidaan teettää ulkoisella yrityksellä. Silloin voidaan kilpailuttaa eri sovelluskehitysyrityksiä tarjoamalla ohjelmiston valmiit taustatiedot, toiminnollisuuksien, ei-toiminnollisuuksien sekä rajausten tila ja sovelluksen tarkoitus ja päämäärä.

Vaatimuksilla halutaan rajata ohjelmisto mahdollisimman paljon kustannussyiden, riskienhallinnan ja toimivuuden takia, jotta projekti voidaan hallita mahdollisimman tehokkaasti ja se ei kaatuisi myöhemmin. Määritys on selkeäkielinen kirjallisesti ja kuvitettu dokumentaatio, jolla haluamme tuoda esille sovelluksen tietyn tarpeen ongelman ratkomiseen tarvittavat toiminnot.

Toki on aina mahdollisuus, että dokumentaatiolla ei tuoda esille kaikkia piileviä riskejä ja ristiriitaisuuksia, mutta määritysdokumentilla vähennetään riskejä. Tämä tosiasia on noussut esille varsinkin nykyään monien kaatuneiden suurprojektien huonon vaatimussuunnitteluiden, epärealististen tavoitteiden, resurssien puutteen ja hallinnoinnin takia. (Hull & Jackson & Dick 2011, 1-3; Kelley 19.11.2013.)

Vaatimusmäärityksen dokumentoinnissa pitää aina muistaa sen päämäärä ja tarkoitus. Dokumentaatiossa voidaan enemmän tai vähemmän lisätä vaatimusmäärityksen mitä ja miten tekstipohjaa ja kaavioita, jotta voidaan saada jopa sovelluksen suunnittelu asiakkaan niin vaatiessa sovelluksen rajauksia varten, mutta tässä tapauksessa suunnittelun pitäisi dokumentoida ”Suunnittelun rajaukset” tai vastaavanlaisen osion alle. Syy edellä mainittuun on vaatimusmääritys dokumentoinnin hyvien tapojen ylläpitäminen IEEE:n ja Voleren vaatimusmääritys dokumentti pohjiin viitaten.



Kuva 2. Pinnallinen näkemys eri vaatimusten suhteista

Dokumentoinnin tai toisin sanoen sen laatimisen menetelmätapoja on enemmän kuin pari, mutta yksi tunnetuimmista tavoista on suunnittelupohjainen (engl. plan-driven software) ja Agile-menetelmät. Eri menetelmiä voidaan myös yhdistää ja ulkopuolisten mallien tuominen ja integroiminen määritykseen ei ole kiellettyä, koska ajan myötä vanhat kaavat ja

kuvaustavat ovat kehittyneet paremmiksi. Silloin on mielekkäämpää ottaa mukaan uudenlainen tapa laatia kuvaus tai kaavio. Tavoitteena on, että tuotetaan laadukas ja ajantasainen dokumentaatio, jossa on järjestelmällisesti muutettu asiakkaan tarpeet ja vaatimukset eri osiin, joita voi sitten hyödyntää järjestelmän komponenttien rakentamisessa. (Bass & Bergey, 1-3.)

Tiettyjä tapoja ei ole olemassa siitä, miten järjestelmän kannalta riittävän olennaiset tiedot voidaan löytää, mutta on olemassa tiettyjä kriteerejä ja tutkimuksen alueita, miten nämä voidaan löytää jakamalla käyttäjien ja järjestelmän vaatimukset omiin segmentteihin ja siitä eteenpäin jakaa ja avata nämä omiksi.

Mitä monimutkaisemmaksi järjestelmä kasvaa sitä suurempi syy on ottaa käyttöön yksityiskohtainen määritys eri käyttötapauksille, koska käyttötapauksella voi olla monta eri mahdollista lopputulosta ja uusia käyttötapauksia ja näillä voi olla vielä enemmän kuin edellisellä, joten ohjelmoinnissa voi tulla loogisia virheitä eri tasolla. (Sommerville 2011, 83.)

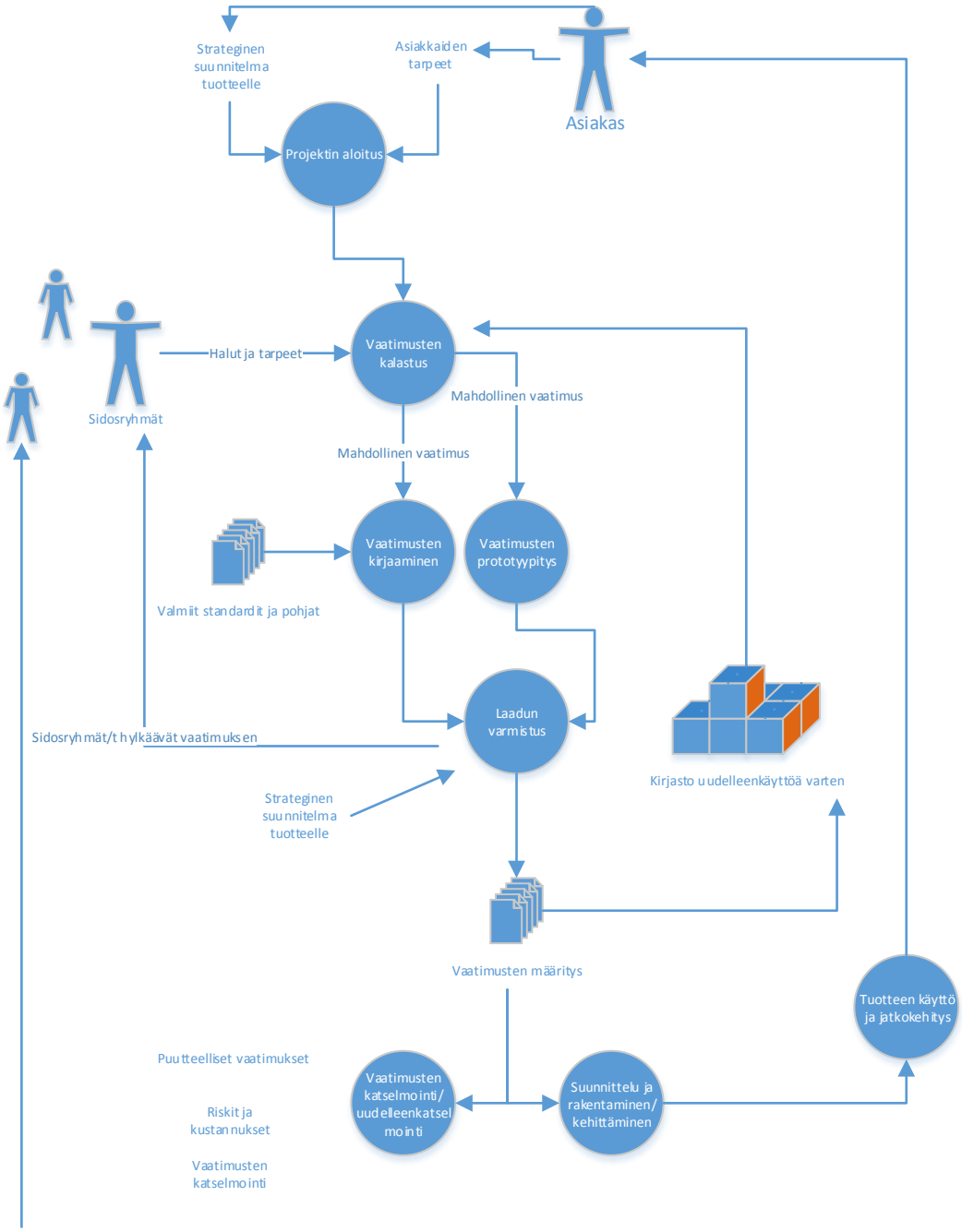
Määrityksen tavoitteena on siis toistamiseen kerrottuna kirjata kaikki asiakkaan vaatimat käyttötapaukset yhteen tai useampaan samantapaiseen dokumenttiin, jotta koko kehitystiimillä olisi kirjattuna kaikki projektin elintärkeät tiedot. Olisi erittäin hyödytöntä, jos laadittua dokumenttia ei olisi vaan kaikki tiedot olisi vain kerätty pieninä osina sieltä ja täältä. (Wiegers & Beatty 2013, 13.)

Kaikesta epävarmuudesta ja epätietoisuudesta huolimatta, vaikka millä mallilla, standardilla ja menetelmällä dokumentti on laadittu niin siinä pitää aina muistaa, että dokumentti ei ole staattinen vaan ns. elävä. Sitä päivitetään jatkuvasti ennen kuin aletaan kirjoittamaan ja rakentaa itse sovellusta samoin kuin sen aikanakin, oli syy mikä tahansa. Vaatimusten määrittäminen on kriittisen tärkeä vaihe sovelluksen kehitysprosessissa, koska dokumentoinnin virheellisyys lisää eri suunnitteluvirheiden riskiä myöhemmässä sovelluksen kehitysvaiheessa. (Sommerville 2011, 77–88; Wiegers & Beatty 2013, 17–19.)

3.2 Vaatimusmäärityksen tehtäväkokonaisuudet

Vaatimusmäärityksen dokumentoinnin prosessissa ei ole tiettyä pakollista kaavaa ja eikä siinä voi olla yksinkertaista tapaa aina tehdä asioita tietyllä tapaa, koska eri projekteissa tarvitaan projektiin mukautunut oikeanlainen lähestymistapa, mutta meillä on olemassa eri tapoja miten edetä asiakkaan vaatimusten dokumentoinnissa ja varmistamisella oikeaksi. Alla oleva kuva on Robertsonin (2012) Voleren prosessikaavio, joka on yleistetty ja mal-

linnettu kuvaus vaatimusprosessista.



Kuva 3. Volere prosessi kaavio

3.2.1 Projektin aloitus

Jokaisella projektilla ja työllä on alku joten vaatimusmäärityksen tai määritysten kokoaminen ei eroa mitenkään tässä asiassa. Hankkeen aloituksessa tai toisin sanoen aloituskokouksessa laaditaan perustuksia määritysten löytöä varten.

Projektin aloituskokouksessa kaikki projektin kriittiset tai pääasialliset sidosryhmät esim. tuotteen tilaaja, käyttäjät tai käyttäjien edustajat/asiantuntijat, analyytikot, kehittäjät ja tekniset asiantuntijat, liiketoiminnan johto tai asiantuntijat kerääntyvät yhteen saadakseen, jonkinlaisen yhteisymmärryksen hankkeen ratkaisevista taustatiedoista ja päämääristä.

Aloituskokouksissa yleensä tutustutaan projektin eri sidosryhmiin ja heidän taustoihin, kehitettävän sovelluksen ongelma-alue ja sen ratkaisemiseen haluttu toimintatapa esim. asiakkaan tietyt pakolliset vaatimukset sen käyttöliittymälle, ratkaisulle tai integraatioille. Kokouksessa määritellään liiketoiminnan ongelma ja rajataan kehitettävä työ ottamalla huomioon kaikki sidosryhmät ja heidän tarpeet/vaatimukset. (Robertson's, 35 – 37.)

Aloituskokouksen kulku voi olla minkälainen tahansa, mutta siinä luonnollisesti aloitetaan niin, että kokouksen vetäjä/vastaava tuo esille keskustelun aiheeksi ongelma, johon sovelluksella/järjestelmällä pitäisi saada ratkottua. Sidosryhmät esittelevät alkuesittelyn jälkeen itsensä, oman organisaationsa tai toimiston osaston ja heidän vaatimuksensa tai toisin sanoen tarpeet sovellukselle. Kokouksen kulussa voidaan myös käyttää eri visualisointi menetelmiä kuten konteksti- tai yhteyskaavio työn rajausta varten. (Robertson's, 35 – 37.)

Kokouksessa myös tullaan käymään läpi mahdollinen ratkaisu ongelma alueelle tai päämäärät, jotka sovelluksen pitää tavoitella, alustavat kustannusarviot ym. riskitekijät sovellukselle. Tässä vaiheessa sovelluskehitys firmalla tai tiimillä on teoriassa vielä mahdollista perääntyä, jos työ vaikuttaa teknologisesti liian vaikealta, kustannuksiltaan liian suuret tai muiden vastaavanlaisten asioiden takia, koska on aina parempi perua kuin mennä eteenpäin mahdottomalla projektilla, joka voi viedä resursseja, voimia ja onnistumismahdollisuudeltaan olematon. (Robertson's, 35 – 37.)

3.2.2 Vaatimusten kalastelu

Aloituskokouksen jälkeen aloitetaan yleensä vaatimusten kalastelua tai toisin sanoen määritysten löytämistä ja tutkimista, jossa vaatimusmäärittelijä aloittaa työnsä löytääkseen eri liiketoiminnan ongelmakohtia ja miten eri sidosryhmät haluavat, että se halutaan ratkaista. Ratkaisu yleensä on liiketoiminnan käyttötapauksen muodossa, esim. ”Haluan automatisoida laskujen laskentalogiikan, manuaalisen laskutoimituksen ja syötön sijasta”. Jokainen liiketoiminnan käyttötapaus, luo eri toiminnollisuuksia vaaditulle työlle, saadakseen aikaan halutun lopputuloksen. (Robertson's, 16 – 20.)

Kalastelulla, löytämisellä ja tutkimisella tarkoitetaan vaatimusten kartoittamista. Vaatimusmäärittelijä yleensä on yhteydessä työalueen vastaavien kanssa tai on jopa paikan

päällä työn tapahtumapisteessä ja ottaa muistiinpanoja ja haastattelee eri sidosryhmiä esim. työn vaadituista käytettävyyksistä, turvallisuuksista ja eri toiminnoista, jotta saataisiin kokokonaisvaltainen ja tarkka käsitys siitä, mitä työ pitää sisällään, mitä sillä pitäisi ja ei saisi tehdä. (Robertson's, 16 – 20.)

Tämän prosessin ongelma alueisiin kuuluu, yleensä vaadittavien vaatimusten löytämisessä, koska toimijat/sidosryhmät yleensä voivat kertoa omia näkemyksiään siitä miten työ tai ongelma-alue pitäisi ratkaista eli he ilmaisevat omia toivomuksia miten haluaisivat sovelluksen/järjestelmän toimivan. Prosessiin liittyvät lisätiedot tulevat esille opinnäytetyön luvussa 3.6, joka käsittelee määritysten löytämistä ja tutkimista tarkemmin. (Robertson's, 16 – 20.)

3.2.3 Vaatimusten dokumentointi

Vaatimusmääritys prosessin yksi tärkeimmistä vaiheista on dokumentointi. Dokumentoinnissa kirjataan kaikki tiedot ylös, jotka tulivat ilmi tietojen kalastelun ja tutkimisen vaiheessa. Dokumentoinnissa on eri pohja malleja ja erinäisiä syitä kenttien ylös kirjaamiselle, jotka tässä luvussa tullaan käymään läpi. (Robertson's, 20 – 22.)

Dokumentoinnin yksi suurimmista ongelmista on aina ollut väärällä tavalla kirjatut vaatimukset esim. tiedot puuttuvat tai määritys on epälooginen ja sitä ei voida mitata. Vaatimuksia voidaan aina myös dokumentoida väärin, jolloin määritetään, jotain mitä ei saisi olla tai unohdetaan kirjata eri tietoja. (Robertson's, 20 – 22.)

Jotta vaatimusmäärittelijä voi välttää vääränlaisia dokumentteja ja väärinymmärryksiä, hänellä täytyy aina olla kaikki kirjaukset yksiselitteisiä ja määrityksiä pitää voida, jotenkin mitata ja testata. Määrityksien pitää siis tuottaa saman tiedon sekä asiakkaalle, että sovelluskehittäjille. (Robertson's, 20 – 22.)

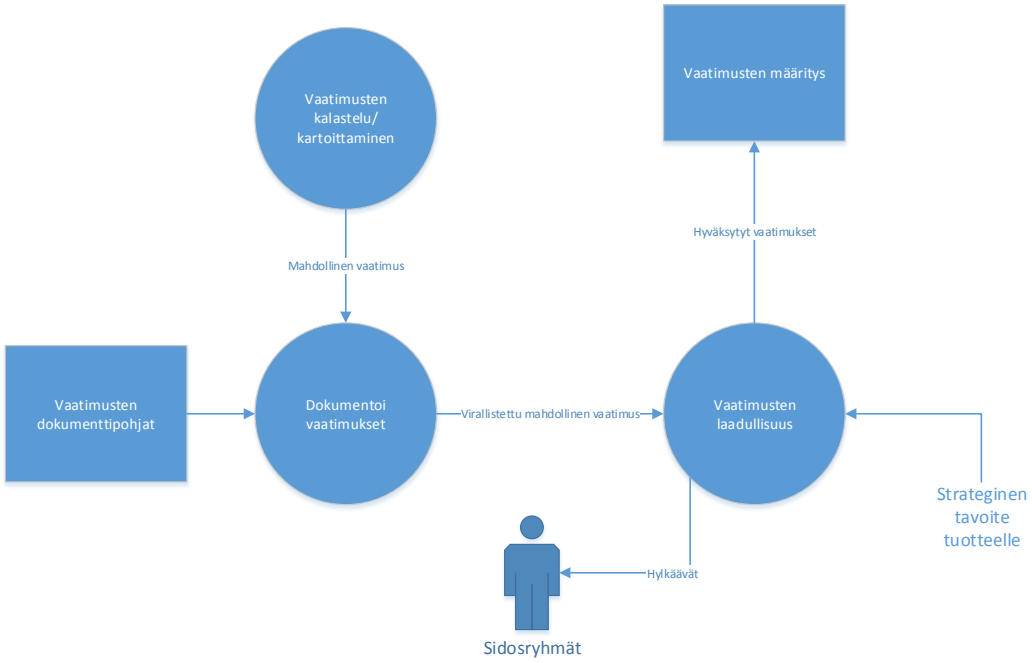
Vaatimusten kirjauksessa voidaan kirjata seuraavanlaisia asioita, jossa otetaan huomioon vaatimuksen perustelu, eli miksi tämä tietty vaatimus on tarpeellinen ja läpipääsyn kriteerit, eli erinäisiä mitattavissa olevia tietoja, joilla voidaan nähdä, että onko tämä tietty määritys läpäissyt sille asetetut tavoitteet. (Robertson's, 20 – 22.)

Syy laadullisuuden tutkimiseen on se, että halutaan sovelluskehittäjille mahdollisimman tarkat, täydelliset ja yksiselitteiset dokumentaatiot, joissa annetaan selkeä kuva kehitettävästä kohteesta eli pyritään mahdollisimman täydelliseen kirjattuun dokumentaation. Vaa-

timusten dokumentointia käydään läpi enemmän opinnäytetyön 3.3 ja 3.4 luvuissa. (Robertson's, 23 – 23.)

3.2.4 Vaatimusten laadullisuus

Määritettyjen laadullisuuden tutkimisessa, pyritään estämään sopimattomien vaatimuksien pääsyä osaksi itse vaatimusmääritys dokumentaatioon, joka myöhemmin luovutetaan sovelluskehitys organisaatiolle. Vaatimusten laadullisuuden tutkimisessa on kyseessä jo olemassa olevien ja kirjattujen vaatimusten läpikäyntiä yksitellen ja katsoen niitä perusteellisesti läpi, että onko vaatimuksia antanut henkilö ollut täysin varma oman haluamansa toiminnollisuuden tarpeellisuudesta ja onko hän pystynyt selittämään mitä hän on halunnut, jotta vaatimusmäärittelijä olisi voinut kirjata yksiselitteisen määrityksen vaatimukselle. Tässä vaiheessa oletetaan, että vaatimusmäärittelijöillä on ollut hyviä kalastelu, tiedon keräys ja dokumenttipohjia- ja hyviä käytäntöjä kuten alla olevassa Robertsonien (2011, 23 – 23.) laatimassa kaaviossa näkyy.



Kuva 4. Vaatimusmääritys prosessin laadullisuuden tarkastusvaihe

Kuvasta näkyy, että hyväksi nähtyjä pohjia käyttäen dokumentoidut vaatimukset, joutuvat sidosryhmien hyväksyttäväksi ennen lopullista lisäystä vaatimusmääritys dokumentaatioon. Jos vaatimusta ei hyväksytä niin se palaa takaisin sen kirjoittajalle ja henkilölle, joka oli vaatinut kyseistä asiaa lisäselvennystä varten. Lisää vaatimusten laadullisuudesta käydään läpi opinnäytetyön 3.7 luvussa.

3.2.5 Vaatimusten katselmointi

Kun kaikki vaatimusten laadullisuus ja oikeellisuus on tarkistettu ja ollaan varmoja, että määritykset ovat mahdollisimman täydellisiä, niin siirrytään seuraavaan vaiheeseen, jossa katselmoidaan aikaan saatu dokumentaatio. Katselmoinnissa varmistetaan, että mitään vaatimusta ei jäänyt pois määrittelystä ja, että kaikki vaatimukset ovat toistensa kanssa johdonmukaisia. (Robertson's, 23 – 23.)

3.2.6 Vaatimusmääritys prosessin menetelmät

Vaatimusmäärityksessä voidaan noudattaa kahta prosessi mallia. On olemassa toistuva (engl. iterative) ja askel askeleelta etenevä tyyli eli vesiputousmalli (engl. waterfall model). Kumpikin on yhtä hyvä vaikka yleinen oletus on, että vaatimusmäärittelyssä pitäisi edetä prosessissa lineaarisesti ylöspäin, alaspäin, sivusuuntaan tai muuhun totuttuun mallinäkemykseen. (Robertson's, 323 – 336.)

Toistuvassa mallinnus tavassa, pyritään aloittamaan sovelluksen rakentaminen, tuotteen omistajalta saaduilta ja tutkituilta esitiedoilla ja tarinoilla. Tuotteen omistajalta saatu palaute muuttaa tarinoita, joka taas muuttaa sovellusta kunnes tuote on valmis. Tämä tyyli on yleisesti käytössä Agile sovelluskehitys menetelmissä, jossa heti aloituskokouksen jälkeen tuotteen sidosryhmät priorisoivat tuotteelle tärkeimmät liiketoiminnan käyttötapaukset, jolloin sovellusta aletaan rakentamaan niiden määrityksen jälkeen. (Robertson's, 323 – 336.)

Vesiputousmallissa tutkitaan alustava liikeidea. Kalastelun ja tutkimisen jälkeen kirjataan kaikki vaatimukset ylös hyväksi nähtyihin mallipohjiin kunnes ollaan valmiita. Vaatimusten kirjaamisen jälkeen täydennetään vaatimusmääritys dokumentti kunnes se on täydellinen, jolloin voidaan aloittaa sovelluksen rakentaminen. Täydellinen vaatimusmääritysdokumentti voi olla itse sovelluskehitys firmassa tuotettu tai sen on annettu toimeksiantona ulkopuoliselle yritykselle, joka on erikoistunut liiketoimintojen tutkimiseen ja vaatimusmääritysten laatimiseen. (Robertson's, 323 – 336.)

3.2.7 Vaatimusmääritys prosessin retrospektiivi

Retrospektiivi tarkoittaa jo tapahtuneen miettimistä nykyajassa, jossa pyritään jälkiviisaina käymään läpi asioita. Vaatimusmääritys prosessin retrospektiivissä pyritään parantamaan omia opittuja prosessimenetelmiä, jossa painotus on opitun asian lisäys jo olemassa olevaan osaamiseen. (Robertson's, 25 – 25.)

Retrospektiivi työkaluna on yksi tehokkaimmista tavoista hyvien ja huonojen prosessien kartoittamista, jolloin voidaan parantaa omaa osaamista. Retrospektiivissä tiimi käy erinäisiä keskustelun aiheita, jossa jokainen tiimin jäsen käy läpi:

- Mikä meni hyvin?
- Mikä meni huonosti?
- Jos olisi mahdollisuus niin mitä tehtäisiin toisin ja miksi?

Retrospektiivin tulos muokkaa olemassa olevia tapoja ja näin muuttaa tulevia projekteja, joita otetaan työn alle. (Robertson's, 323 – 336.)

3.2.8 Eri projektityyppejä

Vaatimusmäärittelyä aloitettaessa pitää muistaa, että on olemassa erityyppisiä projekteja ja erityyppisiin projekteihin tarvitaan erilaisia lähestymistapoja. Meillä on olemassa Robertsonien (2012, 32.) mukaan:

- Pieniä projekteja, joissa sovelluskehitys prosessi käyttää Agile menetelmiä, jolloin vaatimusmäärityksenkin pitäisi olla toistuvaa mallia huomioiva. Tällaisissa projekteissa, liiketoiminnan käyttötapaukset löydetään pieninä osina kun edetään ja rakennetaan eri toimivia toiminnollisuuksia, jossa saadaan palautetta tuotteen omistajalta. Palautetta käyttäen voidaan korjata olemassa olevaa tai lisätä uusia toiminnollisuuksia. Tämän tyyppisissä projekteissa ei käytetä liian paljon aikaa dokumentointiin heti alussa, mutta lopputulos vaatimusmääritys dokumentin suhteen voi olla sama loppupuolella monien erityyppisten projektien kanssa.
- Keskisuuria projekteja, jossa virallinen dokumentointi ja käytäntö voi olla vajaata, mutta silti tarpeeksi suuri, jossa kirjatut vaatimukset ja dokumentaatiot, joutuvat liikkumaan toimistosta toiseen. Tämän tyyppisissä projekteissa voi olla monia eri sidosryhmiä, jotka sijaitsevat eri toimipaikoilla. Ennen sovelluksen rakentamista ei tarvitse olla täydellinen vaatimusmääritys olemassa.
- Suuria projekteja, jossa koko sovelluskehitys firma tarvitsee täydellisen vaatimusmääritys dokumentaation edetäkseen. Tämän tyyppisissä projekteissa on yleensä kyse valtiotason projekteista esim. terveydenhuolto järjestelmästä, uudesta lentokoneesta jne.

3.3 Kriteerit

3.3.1 Dokumentin laatimisen kriteerit

Vaaditut tietotarpeet dokumentaatiossa määräytyvät sen mukaan, että mistä vaatimusdokumentin osa-alueesta on kyse. On olemassa olevan rakenteen määritteleminen, liiketoiminnan tai toisin sanoen järjestelmän vaatimusmäärittelytyö (ympäristö, järjestelmän prosessit, määritelmät ja liiketoiminnalliset prosessit) sekä sovelluksen määritystyö (käyttötapaukset, arkkitehtuurikuvaukset, sekvenssikaaviot, rautalankamallit jne.).

Vaatimusmäärityksen kokonaisrakenteen ja pohjan pitäisi tukea järjestelmän rakentamisen kannalta olevien tietojen löytämisestä ja segmentoinnista vaivatonta. Dokumentin pitäisi myös pystyä edustamaan liiketoiminnan tavoitteita, olemaan johdonmukainen ja tukea testaamista. (Bass & Bergey, 5.)

Ottamalla huomioon kriteerit voidaan segmentoida ja luokitella järjestelmän määritys niin tehokkaasti, että jäljitettävyys on mahdollista ilman suurta vaivaa. Siinä jokaisella käyttötapauksella on järjestelmän toimintojen kannalta omat määritykset ja kuvaukset tarkasti lueteltuna niin teknisille kuin manageritasolla oleville henkilöille, joilla ei ole ICT-alan osaamista. (Sommerville 2011, 84.)

Määritys pitäisi myös jakaa selkeään toiminnollisuus- sekä ei-toiminnollisuus-segmenttiin ilman, että jakaa nämä kaksi omaksi käsiteltäväksi aihealueeksi. Yksi tapa toteuttaa edellä mainittu on laatia tarkkoja, mutta selkeitä kaavioita.

3.3.2 Määrityksen hyväksyntäkriteerit

Kriteerit määrityksille ja niiden hyväksynnälle (engl. acceptance criteria) tarkoittaa sitä, että hyvin kirjoitetun ja toimivan sovelluksen lisäksi, meillä olisi myös sovellus, joka noudattaisi sille asetettuja vaatimuksia eli se tekee mitä sen alun perin toivottiin suorittavan ilman turhia lisäominaisuuksia, kunhan siitä ei puutu mitään vaatimusta.

Vaatimusmäärityksen määritysten hyväksynnälle ja sen laatimisessa ei ole tiettyjä kriteerejä, mutta mallipohjia ja standardeja tarkastelemalla sekä kirjallisia lähteitä lukemalla, voidaan saada tietynlainen johtopäätös, että mitkä olisi hyvä ottaa huomioon dokumenttia laatiessa. Heti ensimmäinen mainitsemisen arvoinen asia on sovelluksen tai määrityksen mitattavuus.

Hyvät määrityksen/vaatimuksen hyväksyntä kriteerit ovat seuraavanlaiset, jossa sovelluksen täytyy olla käyttäjien, asiakkaan ja muiden sidosryhmien hyväksymiä. Kriteerit ovat ennestään asetettuja standardeja tai vaatimuksia tuotteelle tai projektilla, joita sen pitäisi tavoitella. (Jackson 2013.)

Kriteerien hyväksymisessä pitää muistaa, kuten aikaisemmin mainittiin, niiden mitattavuus. Syy on, että kriteereillä on aina kaksi tilannetta eli ne ovat joko hyväksyttyjä tai ei-hyväksyttyjä ja vaihtoehtoisesti ei-katselmoituja. Hyväksynnässä pitää muistaa, että siinä pitää sisällyttää sekä toiminnalliset, ei-toiminnalliset ja rajaavat toiminnot.

Hyväksyntä kriteerien pitää olla erittäin selkeästi kuvattuja, yksinkertaisella kieliasulla, jota asiakas käyttäisi tai ymmärtäisi kuten käyttötapauksissa ilman mitään auki jääneitä tai epäselviä kohtia. Niissä pitää näkyä, että mitkä seikat ovat hyväksyttäviä ja mitkä eivät ole ja niiden pitäisi olla nopeasti muutettavissa toiminnalliseen muotoon esim. manuaalisia tai automaattisia testitapauksia varten. (Jackson 2013.)

Suurin syy hyväksyntä kriteerien olemassaoloon on se, että vaatimuksien pitää aina tyydyttää, joku tarve. Asiakkaan tarve on optimaalisen hyödyn maksimointi eli kustannuksien suhde tehtyyn työhön. Edellä mainitussa kohdassa saattaa huomata, että asiakkaan halukkuus maksamaan työstä alenee jos tehty työ ei tuo tarvittavaa hyötyä. Vain koska kirjoitetaan toimiva sovellus, joka tekee jotain toimintoa oikein, ei tarkoita, että sillä voisi ratkoa liiketoiminta prosessin tiettyä ongelmaa. Kirjoitetun koodin rivimäärä ei tarkoita laadullista työtä, joten määritysten hyväksynnälle asetetut vaatimukset ja dokumentaatiot ovat olennainen prosessin osa-alue vaatimusmääritystyössä.

3.4 Vaatimusmääritys dokumentin rakenne

Määrityksen rakenne ja sisältö on periaatteessa ns. ohjelmistokehittäjien allekirjoitettu sanoma siitä, mitä kehitettävän sovelluksen pitäisi toimittaa asiakkaalle ja määritysdokumentin sisältö ei saisi sisältää ns. auki jääneitä epäkohtia (engl. TBD tai to be determined.), koska muuten dokumentaatio voidaan mieltää keskeneräiseksi.

Määrityksen rakenne voi vaihdella määrityksestä ja ohjelmistokehitys yrityksestä toiseen riippuen kehitettävän ohjelman tyypin mukaan. Rakenne voi myös vaihdella yrityksestä ja ohjelmoijista toiseen ja niin poispäin vaikka kyseessä on toistuvasta työstä, jossa on pohjat ja standardit, kunhan siinä on huomioitu seuraavat seikat kuten:

- Oikeellisuus
- Yksiselitteisyys
- Kokonainen
- Johdonmukainen
- Varmistettavissa oleva
- Muokattavissa
- Jäljennettävissä

Vaikka vaatimusmääritys dokumentaatiot kirjoitetaan samassa organisaatiossa missä kehitetään dokumentoinnissa kyseessä olevaa sovellusta ja dokumentaatiolla hallinnoidaan käynnissä olevaa projektia, niin toimitettava ratkaisu voidaan myöhemmin antaa toiselle organisaatiolle kehitettäväksi, jolloin dokumentin muodon pitäisi olla suhteellisen helposti ymmärrettävissä ja selkeä. Edellä mainittu tarkoittaa, että dokumentaation pitäisi:

- määritellä täsmällisesti kaikki sovelluksen vaatimukset
- olla kuvaamatta mitään suunnittelun tai toteutuksen yksityiskohtia
- olla rajoittamatta sovellusta lisää erinäisillä vaatimuksilla

Vaatimusmääritys dokumentaatio voidaan jakaa erillisiin osiin, jos sen laajuus paisuu liian suureksi, jotta sen ylläpito olisi mahdollista, mutta kaikki riippuu kehittäjistä. IEEE/ISO-standardit eivät ole pakollisia, mutta silti niitä kannattaa aina käyttää viitteenä, kun laaditaan pohjaa, mutta taas sen ymmärtää täysin hyvin jos pohja eroaa projektista toiseen, kuten edellä mainittiin, koska meillä on monia erityyppisiä ohjelmia, joiden piirteet ja tarkoitusperät voivat olla hyvin erilaisia.

Sulautettu järjestelmä toimii täysin eri tavalla kuin pieni pelisovellus tai massiivisia tietokantamuutoksia ja hakuja tekevä sovellus, mutta silti niissä on jotain yhdistäviä piirteitä. Jokaisen tieteenalalla, jossa käytetään automatisoituja tai elektronisia ratkaisuja on erilainen, mutta se ei tarkoita, ettei sovelluskehitysprojektien välillä olisi suuria muutoksia, joita tulemme tässä osassa käymään läpi. (Sommerville 2011, 10.)

Sommerville (2011, 93) on luetellut seuraavanlaisen sisältörakenteen vaatimusmäärityksselle. Lukija huomatkoon, että järjestys ei aina ole alla olevan kaltainen ja nimeäminen tulee olemaan erilainen:

- Kansi: Tulee ilmi vaatimusmäärityksen tekijä, versio ym. tarpeelliset dokumentin tunnistus tiedot.
- Johdanto: Kuvaa järjestelmän taustaa, tarpeita ja tehtäviä. Tässä osiossa pitäisi selittää järjestelmän toiminnot lyhyesti ja miten sovellus toimii muiden järjestelmien kanssa. Tässä osassa pitäisi myös pystyä kirjoittamaan, miten sovellus palvelee sen tilaajaorganisaation toimintoja ja liiketoimintaa.
- Käsitteet: Tässä kohdassa pitäisi kirjoittaa kaikista käytetyistä teknisistä termeistä ja lyhenteistä, jotta lukija ymmärtää dokumentaatiossa selitetyn asian kokonaisuutta.
- Käyttötapauskuvaus ja -Kaaviot: Tässä osiossa pitäisi pystyä luettelemaan kaikki käyttäjille tarjotut toiminnot ja palvelut. Ei-toiminnolliset vaatimukset pitäisi sisällyttää tähän osioon myös. Tässä kappaleessa voidaan käyttää diagrammeja, kaavioita, tauluja ja kuvauksia tai muita tapoja, kunhan asiakas ymmärtää asiakokonaisuutta. Tuotteen prosessi on myös suotavaa pystyä määrittelemään tässä vaiheessa. Tämä osio yleensä kuuluu järjestelmän yleiskuvauksen otsikoinnin alle,

jossa on sitten alaotsikoita käyttäjistä, rajauksesta, sen toiminnoista ja yllä olevista asioista.

- Järjestelmän arkkitehtuuri: Tämä kappale esittää korkean tason karkean kokonaiskuvauksen odotetusta ohjelmasta ja toiminnollisuuksien kytköksistä ja suhteista koko järjestelmässä.
- Järjestelmän vaatimusten määrittäminen: Kuvataan yksityiskohtaisemmin toiminnalliset sekä ei-toiminnalliset vaatimukset, sekä pyritään kuvaamaan rajapinnat muihin järjestelmiin ja toimintoihin.
- Järjestelmän mallit: Kuvataan graafisesti järjestelmän toiminnot. Siinä on kuvattu suhteet toimintojen kesken ja itse järjestelmään ja ympäristöön. Esimerkiksi luokkakaaviot ja mallit, oliomallit tai kuvaukset, tiedonkulkumallit tai semanttiset tietomallit.

Rakenteen eri osa-alueissa voidaan Wiegersin (17.1.2013) mukaan käyttää seuraavanlaisia mallinnustapoja, joissa meillä voi olla.

- tekstipohjainen kuvaus
- graafinen analyysimalli
- taulu ja – päätöspuu
- testi tapauksia
- toimiva prototyyppi
- rautalankamalli
- kuvankaappauksia
- tauluja
- matemaattisia lausekkeita
- tietosanakirja
- kuvia, videoita, äänitallenteita

3.5 Kohteen rajaus

Tuotteen rajaukset ovat yleisiä tai ns. globaaleja vaatimuksia, jotka voivat asettaa rajoituksia projektille tai tuotteelle kokonaisuudessaan, jopa tulevissa jatkokehityshankkeissa. Esimerkiksi yksi rajaus olisi seuraavantapainen, jossa asiakas haluaa, että tuote tai toiminto on vain käytössä tietyn kauden, kuukauden tai jakson aikana.

Rajauksella yritetään tuoda esiin, että jos tuote ei toimi tietyllä tavalla, siitä ei ole mitään hyötyä tiettyyn haettuun ongelmaan. Esimerkiksi jos tuote ei ole käytettävissä vain tiettynä aikana tai tietyn ajan, se menettää koko tarkoituksena. Yksi hyvä esimerkki on yliopisto- ja

ammattikorkeakouluhakusivustojen aikajaksot, jolloin hakemuksia voidaan ottaa vastaan sähköisesti. Rajauksen asettaminen tai rajauksen kartoittamien voi auttaa lopullisen tuotteen toimintojen rakentamisessa, jotta tuote hyödyntäisi olemassa olevaa manuaalista työtä.

Vaatimusten rajaukset vaikuttavat kehitettävän sovelluksen laajuuteen siinä mielessä, että mitä toiminnollisuuksia ja ei-toiminnollisuuksia sisällytetään lopulliseen sovellukseen. Rajauksien hyötyihin tai niiden kirjaamisen hyötyihin kuuluu ajan ja rahankäytön tehokkuus projektin kannalta, että turhaa aikaa ei käytetä tarpeettomiin asioihin.

Rajoitukset myös voivat toimia sovelluksen toiminnollisuuksien ohjaajana siinä mielessä, miten niiden halutaan toimivan tai jättämään pois toiminnoista. Määrityksen rajauksessa esimerkiksi voidaan rajata tietyn sovelluksen toimivuus vain mobiililaitteisiin toimivaksi tai siinä voidaan myös ottaa huomioon kulttuuripoliittiset seikat. Syitä voi olla monia sovelluksen rajoitteisiin, kuten integroitavan palvelun, järjestelmän tai sovelluksen tietyt vaadittavat tiedot, tietovirrat tai toiminnollisuudet, kuten tietokantajärjestelmissä yleensä ovat tietyt tietotyypit, jotka rajoittavat sovellusta jne. (Robertson’s 2013, 60–61.)

Rajauksien haittapuolina toimii se, että jos ei tiedetä kaikkia tarpeellisia toiminnollisuuksia ja käyttötapauksia sovelluksen tavoitteisiin pääsemiseksi, rajaukset voivat toimia projektin päämäärää vastaan. (Robertson’s 2013, 60–61.)

3.6 Vaatimuksen löytäminen ja tutkiminen

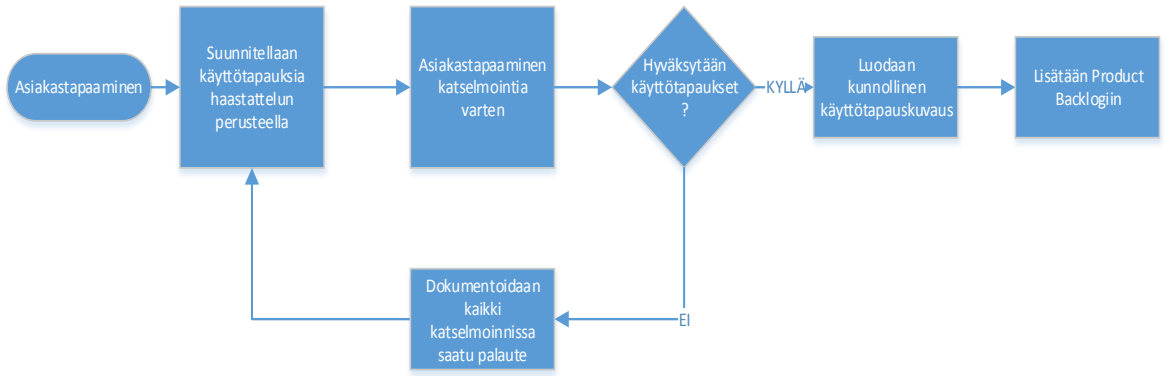
Ohjelmiston vaatimusmäärittely (engl. requirements elicitation) eli sananmukaisesti vaatimusten määritteleminen on pakollinen osa kaikissa kehitettävissä sovellusjärjestelmissä. Sovelluksen vaatimusmääritystä on tehty siitä lähtien, kun ohjelmistoja alettiin kirjoittaa, mutta tieteenalana se on ollut vuodesta 1985 lähtien. (Paakki 2011, 2.)

Määritysten löytämiseksi prosessissa kerätään tietoa halutusta järjestelmästä ja olemassa olevista järjestelmistä, jotta saadaan määriteltyä käyttäjä ja – Järjestelmävaatimukset. Määritysten keräilyssä haetaan ja tallennetaan tietoa kirjallisista ja sähköisistä dokumenteista, omistajilta ja asiakkailta ja samantapaisista sovelluksista. (Hull 2011, 164.)

Tuotteen omistajien tai sidosryhmien kanssa voidaan kerätä tietoa haastatteluilla ja kysymyslistoilla. Esimerkiksi tämän opinnäytetyön aiheena olevan kehitettävän komponentin vaatimusmääritykset kerättiin asiakastapaamisen kautta saaduilla kysymyslistoilla, kaavi-

oilla ja rautalankamalleilla. Vaatimusten määrittelyssä voidaan myös ns. kalastella oikeita ja tarvittavia toiminnollisuuksia ja suodattaa tarpeettomat siten, että ollaan itse työtapahtuman keskipisteessä tai työprosessin kulun lähellä esim. dokumentoimalla työntekijän vieressä kaikista esiin tulevista asioista. Paras tapa saada selville vaatimus on nähdä ja dokumentoida itse työtä, kun yrittää saada itse kuvan esim. työntekijän tai sidosryhmän kuvailujen kautta. (Sommerville 2011, 103; Jansson 2014; Hull 2011, 164; Robertson's 2013, 100 – 103.)

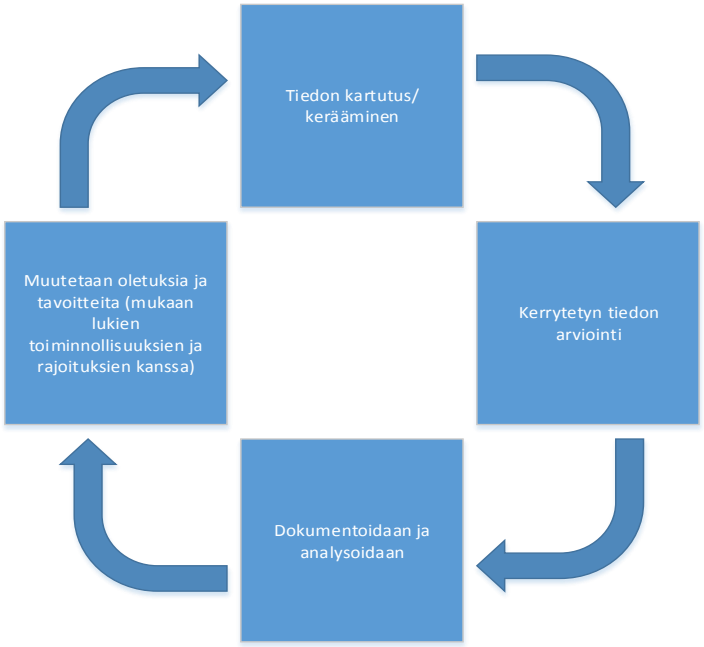
Toiminnollisuuksien, käyttötapausten tai eri sidosryhmien vaatimusten dokumentoimisen jälkeen haastatteluiden, kokousten, kyselyiden tai muita vastaavanlaisia tekniikoita käyttämällä tai niiden yhdistelmillä voidaan vähitellen jakaa kaikki vaatimukset eri käyttötapauksiin. Käyttötapauksissa on tietty formaatti, jossa määritetään eri ns. toiminnon prosessi. Tämän jälkeen kaikki käyttötapaukset käydään läpi toistuvien ja epäloogisten kohtien korjaamista varten ja valmistellaan asiakastapaamisessa käytävää hyväksyntää varten. (Hull 2011, 164.)



Kuva 5. Käyttötapausten hyväksyminen

Määrityksissä yritetään saada tuotteen alkuperäidea, käyttötarkoitus, käyttäjät, mahdolliset eri toiminnot ja integraatiot selville, ja näiden ym. muiden tietojen analysoiminen ja mallintaminen määrää sovelluksen luonteen ja rajaukset. (Hullin & Jacksonin & Dickin mukaan 2011, 36.)

Tietenkin on eri näkemyksiä, siitä mitä määritysvaiheessa oikeastaan tehdään, mutta se on avonainen kysymys, josta kullakin sovelluskehitysorganisaatiolla on omat kantansa. Paakki (2011, 4.) on vetänyt alla olevan kuvan tapaisen johtopäätöksen vaatimusmäärityksen prosessista:



Kuva 6. Vaatimusmäärityksen prosessi

Kuten näemme yllä olevasta kuvasta, se muistuttaa Sommervillen (2011, 103) omia kantoja siitä, mitä prosessi tulee sisältämään. Siinä asiakkaan kanssa käydään läpi oletetun järjestelmän määrityksiä ja korjataan väärinkäsitykset ja luodaan uusi malli. Uudella mallilla ei siis tarkoiteta uutta versiota vaatimusmäärityksestä vaan korjattua uutta versiota, jossa on muutettu käyttötapauksia ym. väärin pääteltyjä asioita. Määritysprosessissa ennen työn aloittamista kysytään aina Hullin, Jacksonin ja Dickin mukaan (2011, 34–35.) seuraavia asioita:

- Onko määrityksen käyttötapauksen teko loppunut?
- Onko määrityksen käyttötapaus selkeä?
- Onko määrityksen käyttötapaus toteutettava?
- Onko määrityksen käyttötapausmalli ja suunnitelma selkeä ja hyväksytty?

Vastaukset edellä mainittuihin kysymyksiin ovat seuraavat:

- Puuttuvat tiedot: Sidosryhmiltä, eri käyttötapausten yksityistiedoista jne.
- Selkeyden puuttuminen: Heti ensimmäisistä ongelmista voi olla huono kansiorakenne dokumenteilla. Itse dokumenteissa voi olla toistuvaa tietoa, toistuvaa tietoa erilaisissa dokumenteissa, toistuvaa ja eri version tietoja eri paikoissa, ristiriitaisuuksia ja huonosti kirjoitettuja tietoja ja määrityksiä.
- Eri määritykset/käyttötapaukset/toiminnot on mahdotonta toteuttaa jostain syystä.
- Suunnitelmaa ei ole hyväksytty.

Koko määritysprosessi on jatkuvaa. Siinä palataan aina takaisin suunnittelutaululle, kunnes saadaan haluttu lopputulos tiimin sekä asiakkaan kannalta. Määrittelyssä on tosin myös otettava huomioon, että kaikki asiakastapaamisessa saatu tiedot järjestelmän toimintojen tutkimisessa ei ehkä ole vaatimuksen kannalta olennaisia asioita vaan enemmänkin suunnitteluun liittyviä tarpeita, mutta vaatimukset voidaan jakaa käyttötapauksiin ja priorisoida. (Wiegers 17.1.2013; Hullin & Jacksonin & Dickin mukaan 2011, 34–35.)

Vaatimusten määrityksessä jaetaan siis kaikki ongelmat ja kysymykset pienempiin osiin. Esimerkiksi jos meillä on ns. asiakassegmentti, meidän pitää jakaa asiakkaat vielä alempiin osiin, jossa käydään läpi, onko siellä olemassa erilaisia asiakkaita jne. Kaikki kysymykset yleensä kuitenkin saadaan vastatuksi, kun kehitettävää sovellusta rajataan haastatteluilla tuotteen omistajien kanssa. Kehityksen alussa kannattaa tosin olla erittäin varovainen projektin rajauksen kanssa, jotta kehittäjät eivät lupaa ja haukkaa liian isoa palaa kehitettäväksi, koska määrittelyllä pääosin tarkoitetaan sovelluksen tulevia toimintoja. Senhän takia toiminnollisuuksien mallinnuksessa käyttötapaukset toimivat ns. eri sprinttien rajauksina, jolloin kehitystiimi voi itse jakaa ja rajata kehitetyn demon laajuutta ja toimintoja. Selkeämmin selitettynä vaatimusmäärityksen eri aihe-alueet eivät käsittele samaa aluetta täysin samalla tavalla, mutta ne voivat olla osa eri aihe-alueissa läpikäytyä käyttötapausta. (Hullin & Jacksonin & Dickin mukaan 2011, 36.)

3.7 Määrityksen varmentaminen todeksi ja rajaaminen

Projektin alkuun saattamisesta lähtien määrityksen tarkoituksena on ollut todentaa ja selventää asiakkaan tarpeet. Olisi erittäin epäloogista ja harmillista, jos määrityksen eri vaiheissa ja versioissa ei pidettäisi kokouksia tai asiakastapaamisia, joissa käytäisiin läpi kaikki tehdyt olettamukset aina käyttötapauskuvauksista rautalankamalleihin.

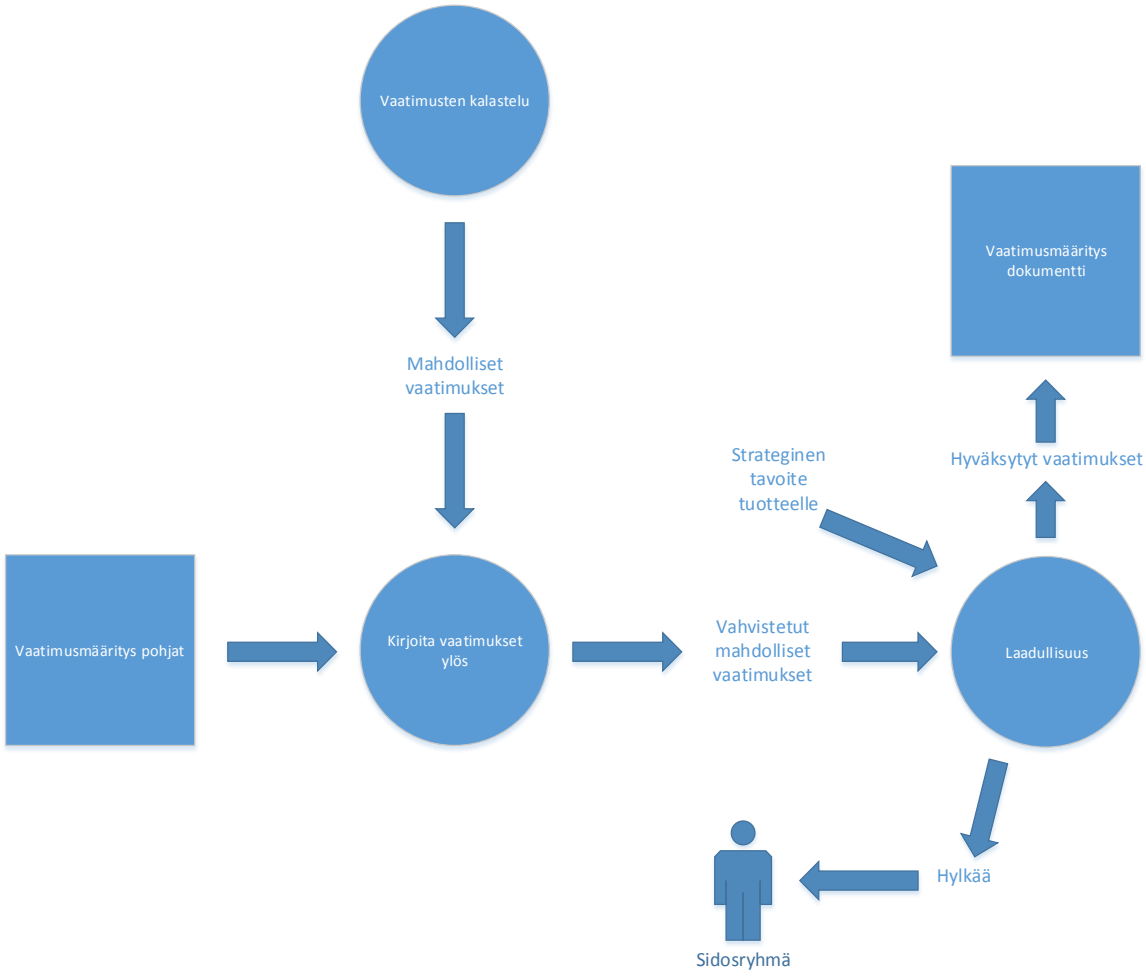
Määrityksen ei tarvitse olla dokumentaatiomuodossa. Se voi olla missä tahansa formaatissa, kunhan siitä saa tietyn käsityksen määritetyistä vaatimuksista. Asiakkaan kanssa käydyissä keskusteluissa on aina mielekkäämpää antaa ärsyttävä mielikuva kuin olla se yksi monista epäonnistuneista ohjelmistokehityshankkeista, jotka kaatuvat puutteellisen hallinnan, rajauksen ja tiedon vuoksi.

Paakkin (2010, 41.) vaatimusmäärityksen tutkimuksessa ja teoriaosuudesta tuli ilmi, että vaatimusmääritykset eivät ilmesty tyhjästä. Toisaalta liiallinen dokumentaatio, jossa ei rajattu kunnolla sovelluksen toimintoja, voi nostaa sovelluskehityksen kustannukset tähtitieteellisiin lukemiin ja kaataa kokonaisia ohjelmia. (Bell & Thayer, 1979; Ditmore 2013;

Paakki 2011, 5.)

Voimme siis vetää johtopäätöksen, että vaikka rajaus olisi hyvä pitää mielessä määrittelyssä, niin on myös hyvä muistaa, että se määrittely, joka meillä tulee olemaan pitää mennä asiakkaan vaatimuksista läpi ja olla mahdollisimman laadullinen ja jäljitettävissä. Jos tiimissä ei olla varmoja kehitettävästä toiminnosta ja siinä on tietoaukko, ei ikinä voida olla varmoja oletetuista määrityksistä ja niiden oikeellisuudesta. (Robertson's 2013, 321.)

Alla on kuvattuna Robertsonien laatima (2013, 303–304.) Voleren vaatimusprosessin osa-alue. Prosessin osa-alueessa keskitytään vaatimusten hyväksymiseen ja hylkäämiseen:



Kuva 7. Voleren prosessikaavion liittyen vaatimusten hyväksyntään/hylkäämiseen

3.8 Vaatimustyypit

3.8.1 Toiminnalliset vaatimukset

Toiminnallisuuksien määritykse pitäisi sisältää kuvaus järjestelmän palveluista, joita sen pitäisi tarjota, miten sen pitäisi toimia eri syötöissä ja toiminnoissa ja yleisesti ottaen miten sen pitäisi toimia eri tapauksissa. Wiegersin ja Beattyn (2013, 7.) ovat tarkentaneet käsitystä siten, että toiminnolliset vaatimukset tarkoittavat kuvausta siitä miten järjestelmä käyttäytyy tietyissä yksityiskohtaisesti määritellyissä oloissa. Esimerkkitapauksessa.

- ”Asiakaspalvelija voi kirjautua sisään tilaustenhallintaohjelmaan.”
- ”Asiakas voi selata ravintolan ruokalistaa ravintolan kotisivulla.”
- ”Internetin kautta ostetut lentoliput voidaan tulostaa kotona.”
- ”Käyttäjä voi hakea vapaita tapaamisaikoja optikolle.”

Toiminnolliset vaatimukset ovat siis ns. ”voi”-muodossa (engl. shall statement). Ne ovat testattavissa ja kuvaavat, mitä järjestelmän pitäisi pystyä tekemään. Toiminnollisuuden pitäisi myös kuvata yhtä tiettyä toimintoa, joka testauksessa joko läpäisee tai saa hylättymerkin kokonaisuudessaan. Toiminnollisuuksien kuvauksen pitäisi myös kuvata itse haluttua toimintoa samoin kuin sen toteutusta. Järjestelmän pitäisi esimerkiksi pystyä tallentamaan oppilaiden kaikki tiedot, kun taas huono toteutus olisi kirjoittaa, että järjestelmän pitäisi pystyä tallentamaan oppilaiden kaikki nimet MariaDB-tietokantaan käyttäen Java-ohjelmointikieltä. (Fleck 2009.)

Toiminnollisuuden vaatimuksilla siis tarkoitetaan toimintaa, jota tuotteen pitäisi suorittaa ollakseen hyödyllinen sen käyttäjille oli toiminnollisuus asiakkaan vaatimuslistalla tai ei. Toiminnollisuuden alue voi olla jonkin lomakkeen automaattisesta laskemisesta tutkimiseen, julkaisemiseen tai muihin samantapaisen asia. Toiminnollisuus on vaatimus, joka nousee tuotteen sidosryhmien tarpeesta saada tietty työ tai prosessi tehtyä, automatisoitua tai/ja sähköistettyä, jos kyseessä on sovellus. Toiminnolliset vaatimukset ovat edellä mainittujen syiden takia ohjelmointikielestä riippumattomia, koska ne pyrkivät tuomaan esille sovelluksen eri sidosryhmien tarpeita ja toimintoja, jotta tietty työn osa saadaan tehtyä (Robertson’s 2013, 9.)

Vaikka toimintojen määrittely näyttää melkein identtiseltä käyttötapausten kuvauksen kanssa, nämä kaksi pitää erottaa toisistaan. Käyttötapauksissa puhutaan ns. minä-muodossa askelkuvauksina, joissa on parempi kuvaus, kun taas toiminnon kuvaus on enemmän ulkoapäin selitettynä vapaampi kuvaus siitä, mitä käyttäjän pitää saada tehtyä, jotta tietty liiketoiminnan kannalta oleva tarve saataisiin tyydytettyä. (Wiegers & Beatty

2013, 7.)

Toiminnolliset vaatimukset siis tulevat tietystä organisaatio- tai liiketoiminnallisesta tarpeesta. Toimintokohtaiset vaatimukset voivat tulla käyttötapauksien jälkeen, mutta loogisesti ne tulevat yleensä ennen käyttötapauksia, koska toiminnollisuudet ovat yleiskuvaus tietystä halutusta toiminnollisuudesta, joka voi sisältää monta käyttötapausta, joissa on avattu sovelluksen toimintalogiikka. Meillä voi olla tarve esim. pizzeriasovelluksen kielen muuttamiseen. Käyttötapaus tulisi edellä mainittuun tarpeeseen seuraavasti, jossa ”Asiakkaana haluan pystyä vaihtamaan kielen englanniksi”. Toimintokohtainen selitys olisi seuraavasti, jossa ”Asiakkaan pitäisi pystyä vaihtamaan pizzerian kotisivulla kieliasetuksia”. Toisaalta mikään ei estä käyttötapauksien tulevan ennen toiminnallisia vaatimuksia, koska käyttötapaukset voidaan purkaa toiminnalliseksi vaatimukseksi ja toiminnallinen vaatimus voidaan muuttaa käyttötapaukseksi. (Wiegers & Beatty 2013, 11–12; Robertson’s 2011, 159–161.)

Toiminnallisia vaatimuksia laatiessa pitää olla erittäin tarkka, että kaikki toiminnollisuuksia kuvaavat tapaukset ovat erittäin selkeitä, täydellisiä tai ns. koko järjestelmän kattavia kokonaisuudessaan ja johdonmukaisia. Johdonmukaisuudella tarkoitetaan sitä, että mitään duplikaatteja (kaksi samantapaista, mutta eri tavalla kuvattuja toimintokuvauksia) tai ristiriitaisia toimintokuvauksia ei saisi olla. (Sommerville 2011, 85–86.)

Sovelluksen toiminnallisten vaatimusten analysoimisessa, hahmotuksen kirjoittamisessa ja kaavioiden laatimisessa pitää tosin aina muistaa, että toiminnalliset vaatimukset, käyttötapaukset ja järjestelmän eri komponenttien prosessit ovat aina kytköksissä ja muutos esimerkiksi toiminnollisuuksien dokumentissa muuttaa myös käyttötapauksien sisältöä, nimeämistä jne. Ovathan käyttötapaukset mallinnuksia asiakkaan vaatimista toiminnoista. Toinen varteenotettava asia, joka tulee ottaa huomioon, varsinkin erittäin monimutkaisia järjestelmiä rakennettaessa, on se, että virheitä tulee vääjäämättä ja toiminnollisuuksien kuvaus ei välttämättä tule kattamaan koko järjestelmään, koska vaatimusmäärittelijöillä ei ole loputtomasti resursseja kaikkien tapausten huomioon ottamiseksi. (Hull & Jackson & Dick 2011, 36; Sommerville 2011, 87.)

3.8.2 Ei-toiminnalliset vaatimukset

Wiegers ja Beatty (2013, 7.) määrittelivät ei-toiminnolliset vaatimukset tietyiksi yleisiksi ominaisuuksiksi, joita järjestelmän pitää noudattaa.

Ei-toiminnollinen vaatimus ei ole selkeästi nähtävissä, mutta se voi rajata tai vaatia tiettyä järjestelmän toimintoa, esim. rajapintaa, lakiin liittyvää määritystä, taustalla toimivien muuttujien tietotyyppejä, tietoturvallisuutta, suorituskykyä, laatua (käyttöliittymä ja toimintalogiikka tasolla) ym. asioita.

Tämä segmentti siis sisältää järjestelmän toiminnolle asetetut vaatimukset, eri toimintojen vaatiman ajan ja miten sen, toimintojen taustalla oleva järjestelmä pitäisi esim. olla integroitunut toisiin järjestelmiin. Kyseessä ei siis ole toimintokohtainen selitys, vaan tässä otetaan huomioon kokonaisarkkitehtuuri yleisellä tasolla, eli kuinka hyvin tai miten järjestelmän pitäisi tehdä asioita. Toiminnollinen vaatimus ja määritys kuvaavat taas enemmän sitä, mitä järjestelmän pitäisi tehdä yksityiskohtaisesti eri tapauksissa.

Ei-toiminnalliset vaatimukset voidaan, kuten edellä mainittiin, jakaa ja avata vielä pienempiin osiin, joissa esimerkiksi tietylle toiminnolle tai järjestelmän osa-alueelle on asetettu tiettyjä yleiskielellä selitettyjä vaatimuksia, kuten eri komponenttien toiminta-aika, tehokkuus, suorituskyky, muistinkäyttö, luotettavuus, turvallisuus ja käytettävyys.
Järjestelmälle on voitu myös asettaa yleisellä tasolla joitakin organisaatiokohtaisia vaatimuksia, esim. mitä eri organisaatioissa pitäisi tehdä sen käyttämiseksi.

Muita ei-toiminnallisia vaatimuksia voivat olla ulkoasuun liittyvät seikat, integraatiovaatimukset, tietyn verkkokaupan kirjautuneiden asiakkaiden lukumääräkapasiteetti ja esimerkiksi tietyn lain lakipykälän x kohdan tai tietyn lain kokonaisuus tulee otettua huomioon esimerkiksi standardiin liittyvissä asioissa (lomakkeet, lähetystapa, formaatti, jne.). (Sommerville 2011, 89.)

Ongelmat, jotka nousevat esiin ei-toiminnallisten vaatimusten tulkitsemisessa ovat, asiakaan tarpeiden looginen jäsentäminen mitattaviin vaatimuksiin. Joitakin vaatimuksia ei edes voida jäsentää mitattaviin osiin, vaan ne voivat olla koko järjestelmään toimintaa koskevia asioita esim. asiakas voi vaatia järjestelmän helppoa ylläpidettävyyttä. Edellä mainittu ei ole mitenkään jäsennettävissä ja voi aiheuttaa ristiriitoja tuotteen loppusuoralla, koska tämä tietty vaatimus aiheuttaa yllätyksellisiä lisäkustannuksia, mutta vaatimusmäärittelijät yleensä huomaavat asiakkaiden kanssa käydyissä tapaamisissa eri vaatimusten mitattavuuden olemattomuuden, jolloin ne yleensä tullaan hylkäämään tai niistä pyydetään lisäselvitys. Jotkut asiakkaan vaatimukset voivat olla myös ristiriidassa muitten toiminnollisuuksien kanssa.

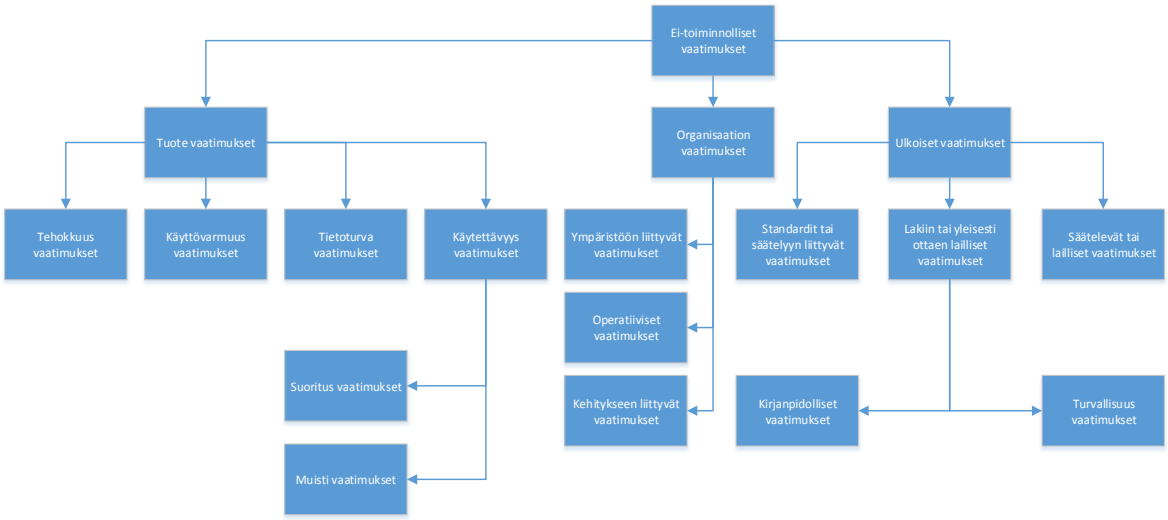
Wiegers (17.1.2013) on jakanut ei-toiminnalliset vaatimukset seuraaviin osa-alueisiin:

- Rajaukset: suunnittelu, käyttöönotto.
- Suorituskyky: viive, suoritusaika.
- Ulkoiset vaatimukset: laitteisto, ohjelmisto, viestitys, käyttäjä/t.
- Laadullisuus: käytettävyys, vankkuus, epävakaus, saatavuus.

Robertsonin (2011, 10) casetapauksia käyttäen yllä olevat ei-toiminnalliset vaatimukset voivat näyttää seuraavilta:

- Tuote voi erottaa ystävällisen vihollisesta murtosekunnin osassa.
- Tuotetta voivat käyttää kansainväliset matkaajat, jotka eivät osaa paikallista kieltä lentokentillä.

Sommerville (2011, 88–89.) jakoi ei-toiminnolliset vaatimukset vielä tarkemmin seuraavan kuvan mukaisesti:



Kuva 8. Ei-toiminnolliset vaatimukset

Yllä olevasta kuvasta näkyy, että ei-toiminnolliset vaatimukset on jaettu tuotteen, organisaation sekä ulkoisia seikkoja huomioon ottaviin segmentteihin. Sommerville (2011, 88–89.) vielä täsmensi yleisellä tasolla, että mitä nämä kolme segmenttiä sisältävät seuraavanlaisella tavalla:

- Tuotevaatimukset: Tähän segmenttiin kuuluvat vaatimukset tarkentavat tai rajaavat sovellusta. Esimerkkeihin kuuluva tehokkuuteen liittyvä vaatimus sisältää sisällään esim. sen, kuinka nopeasti sovelluksen pitäisi suorittaa/suoriutua eri toiminnoista, kuinka paljon muistia sovellus voi enimmillään käyttää, kuinka luotettava sen pitää olla jne.

- Organisaation vaatimukset: Tähän segmenttiin kuuluvat vaatimukset pitää sisällään esim. tietyn tavan, miten sovellusta tullaan käyttämään organisaatiossa, kehitysprosessi kuten käytettävä ohjelmointikieli, kehitysympäristö tai prosessin standardit tai ympäristöön liittyvät vaatimukset, jotka kuvaavat järjestelmän toimintaympäristöä ja rajaavat näin sitä.
- Ulkoiset vaatimukset: Tähän segmenttiin kuuluvat kaikki ulkoiset vaatimukset, jotka vaikuttavat sovellukseen ja sen kehittämiseen. Nämä vaatimukset voivat liittyä eri säädöksiin, esim. mitä tässä järjestelmässä pitäisi ottaa käyttöön, jotta se voidaan ottaa käyttöön pankissa, jotta se toimisi laillisten ja eettisten rajojen sisäpuolella.

Yllä olevasta listauksesta on monia eri versioita, ja monien kirjoittajien sisältö on vertailussa aika samanlaisia. Meillä esimerkiksi voi olla järjestelmän toiminnollisuuksiin liittyviä vaatimuksia tuotteen omistajalta. Silloin itse huolenaihe voi olla esimerkiksi se, että kehitettävissä olevan sovelluksen pitäisi olla erittäin helppokäyttöinen, koska sillä on tietty käyttäjäryhmä, joka tarvitsee enemmän huomiota, jolloin ei-toiminnollinen vaatimus voi liittyä käytettävyyteen. (Cremers & Alda.)

Vaikka näin päällisin puolin lukija ei ehkä ymmärrä heti, mikä on ei-toiminnollisuuksien sekä toiminnollisuuksien ero, siitä on tullut tietynlainen tutkimusaihe, koska itse aihealueena se sisältää monia eri asioita. Goldsmithin (1.6.2014) mukaan:

> “Nonfunctional requirements refer to a whole slew (I've identified more than 30) of attributes including performance levels, security, and the various ‘ilities,’ such as usability, reliability, and availability. Invariably, requirements definers get wrapped up in how the product/system is expected to function.” (Goldsmith 1.6.2014.)

Kuten aikaisemmin yllä olevan kirjoituksen mukaisesti todettiin, niin ei-toiminnalliset vaatimukset määrittävät järjestelmän toimintoja yleisellä tasolla. Goldsmith (1.6.2014) mukaan ei-toiminnolliset vaatimukset on yleensä unohdettu tai jätetty huomioimatta, koska sovelluskehittäjät ja määrittelijät ovat keskittyneitä vain itse toimintoihin, vaikka kaikki toiminnollisuuksien ei-toiminnolliset vaatimukset ovat erittäin tärkeitä. Olisi todella valitettavaa, jos tietokantahaku on epäluotettava, tietoturvallisesti epävakaa ja vie 2 tuntia, kuten hän asetti seuraavasti:

> “Invariably, requirements definers get wrapped up in how the product/system is expected to function and lose sight of these added elements. When such factors are not addressed adequately, seemingly proper product/system functioning in fact fails to function. For example, a system may identify customers in such a slow, insecure, and difficult to use manner that it can cause mistakes which make data unreliable, provoke frustration-based attempted work-arounds that can create further problems,

and ultimately lead to abandonment.” (Goldsmith 1.6.2014.)

Kun ei-toiminnollisia vaatimuksia ei oteta huomioon sovellusta kirjoittaessa, niin vaikka toiminto on teoriassa toimintakunnossa, se ei ehkä ole oikeissa tilanteissa, jolloin järjestelmän hitaus ja epäkäytännöllinen käytettävyys voivat aiheuttaa tilanteita, joissa käyttäjä, jostain syystä haluaa mennä takaisin selaimessa tai tekee jotain muita toimintoja (manuaalisesti yrittää tehdä jotain prosessia koneella), jolloin itse järjestelmän tietokanta voi muuttua epävakaaksi.

Järjestelmän epävakauden tai muiden virheiden takia se voidaan hylätä ja ottaa kokonaan pois käytöstä, jolloin sekä kehittäjäfirma ja asiakas ovat häviäjiä. Siellä on monia muitakin casetapauksia, esim. järjestelmä, jonka pitäisi toimia tietyllä aikavälillä joka päivä maailmanlaajuisesti noudattaa vain tietyn maan aikaa, jolloin muissa maissa käyttö voi olla mahdollista vain 2 - 4 aamuyön aikoihin. (Goldsmith 1.6.2014; Sommerville 2011, 89.)

Ei-toiminnollisia vaatimuksia tosin voidaan aina rakentaa testattavaan muotoon vaikka vaatimukset yleensä kuulostavat erittäin abstrakteilta kehittäjien mielessä, koska aina voidaan jakaa yleisesti kuvatut vaatimukset pienempiin mitattaviin osiin. Muuten niitä ei voida mitenkään ottaa käyttöön sovelluksessa.

Kvantitatiivisesti testattavat ei-toiminnolliset vaatimukset eivät ole mahdottomuus, ja ovat jopa suositeltavia. Esimerkiksi suorituksen nopeus voidaan mitata, muistinkäyttö voidaan mitata megabyte/gigabyte/terrabyte-tasolla ja jopa luotettavuus voidaan mitata eritapaisilla testitapauksilla.

Syy, miksi kvantitatiiviseen lähestymistapaan yritetään panostaa ei-toiminnallisissa vaatimuksissa, on myös se, että jokin sovellus tai järjestelmä voi sisältää kriittisesti tärkeitä toimintoja esim. ihmishengen säilymisessä. (Sommerville 2011, 89; Cremers & Alda.)

Voleren vaatimusmääritysdokumenttipohjan ei-toiminnallisiin vaatimuksiin on Robertsoneiden (2007, 2.) mukaan seuraavat vaatimustyypit:

- Look & feel -Vaatimukset
- Käytettävyysvaatimukset
- Suoritusvaatimukset
- Operationaaliset ja ympäristön vaatimukset
- Ylläpidettävyys ja tukivaatimukset
- Tietoturvavaatimukset
- Kulttuuriset ja poliittiset vaatimukset
- Lailliset tai standardeihin liittyvät vaatimukset.

3.9 Vaatimusmäärityksen laatimisen hyödyt

Vaikka vaatimusmäärityksen hyödyt ovat laajalti tunnettuja, valitettavasti (siellä) on joitakin projekteja, joissa työt (ohjelmointi, tietokannan pystyttäminen ja käyttöliittymän implementoiminen) aloitetaan liian ajoissa ilman tarpeellista vaatimusmäärittelyä. Suurin syy edellä mainittuun on, että tuotteiden omistajat mittaavat rahalleen saatua hyötyä koodirivien määrinä ja itse ohjelmointiin käytetyt ajat.

Puutteellisen vaatimusmäärittelyn ensimmäisiä myöhemmin esille tulevia asioita ovat asiakkaan lisäominaisuuksien ja itse organisaatioprosessiin tarvittujen toimintojen ristiriitaisuus. Esimerkiksi asiakkaan haluama toiminto ei välttämättä ole olennainen sovelluksen päämääränä olevan ratkaisun automatisoiminen.

Toinen heti esille tuleva asia on se, että dokumentaation puutteellisuuden takia voidaan huomata vasta kehitys- tai loppuvaiheilla, että projekti ei olekaan täysin toteutumiskelpoinen esim. teknologiavaatimuksien jäädessä vähän tyngäksi tai vastaavan asian takia, koska teknologian taso ei ehkä ole ajan tasalla vaatimuksien kanssa tai vaadittuja teknologioita ei ole kartoitettu määrittelyvaiheessa. (Young 2004, 2; Robertson’s 2013, 23 – 24.)

Erittäin harvoin tulee vastaan projekteja, joissa kustannukset ja deadline eivät ole rajattuja, ja usein projekteissa on erittäin tiukka budjettikirjanpito, tuntikirjanpito ja työsuunnitelma. Kustannusten määrää ja projektin valmistumisen päivämäärää voi olla mahdotonta ennustaa ilman kunnollista projektinhallintatyökalua (esim. dokumentaatiota tai siihen liittyvää järjestelmää), koska projektit ovat aina erittäin dynaamisia ja kaikki ennalta sovittu tulee ennemmin tai myöhemmin muuttumaan, mikä voi aiheuttaa eheän ymmärryksen menetystä koko projektin infrastruktuurista. Vaatimusmääritykset ovat siis myös aina dynaamisia, olettaen, että se laaditaan ja sitä päivitetään jatkuvasti. (Young 2004, 2; Hull 2011, 159.)

Sovellus, jota ei ole dokumentoitu ja asiakkaiden tarpeet/vaatimukset ovat vain yksilöllisesti tiettyjen henkilöiden kesken jaettu sovelluskehitysfirmassa tai ei ole jaettu lainkaan (hiljainen tieto), voi aiheuttaa sekasortoa, päällekkäisiä töitä ja epäyhtenäisen sovelluslogiikan. Edellä mainittu tapaus voi tapahtua siitä, että asiakkaan kanssa käydyt neuvottelut ja päätökset on tehty etänä esimerkiksi netin tai muun kommunikointivälineen kautta, silloin voi olla erittäin rajattu aikaraja ym. viestimistä rajoittavat asiat. Aina pitää myös muistaa, että sähköisessä viestinnässä kaikki projektin sidosryhmät, kehittäjät ja päättäjät eivät ole paikalla, jolloin virheellisten tietojen tullessa esille niitä ei ehkä voida oikaista oli se asiakkaan ryhmän tai sovelluskehittäjien puolelta. (Young 2004, 2; Hull 2011, 159.)

Määritystä tehtäessä pitää siis aina muistaa, että asiakkaan läsnäololla tapaamisissa on suuri osuus projektin onnistumisessa. Joskus voi tulla esille, että asiakkaan suullisesti viestittämä tieto esim. järjestelmän toiminnosta ja sen integraatiomahdollisuuksista ei pidä paikkansa. Kehitystiimi tekee edellä mainitun tilanteen jälkeen turhaa työtä eikä tehdylle työlle löydy hyötyä myöhemmässäkään vaiheessa vaikka mainittu ongelma voidaan nopeasti todistaa vääräksi, jos asiakkaalla on myöhemmin vaatimusmäärityksen hyväksyntää koskevassa kokouksessa oman organisaationsa teknologiapuolen esimies paikalla, jolloin kaikki kirjatut asiat voidaan nähdä ja korjata oikeiksi. (Young 2004, 2; Hull 2011, 159.)

Vaatimuksen määrityksessä, joka käydään asiakkaan kanssa läpi ja jota jatkuvasti korjataan, voi myös tuoda esille organisaation sisäisiä ennalta tuntemattomia tai hämärän peitossa olevia prosesseja pinnalle ja mallinnettavaksi, jolloin analyytikot, asiakkaat, kehittäjät ym. sidosryhmät saavat paremmin ennalta tiedostamatonta informaatiota.

Vaatimusten määrittäminen ja mallintaminen voidaan rinnastaa tietokantojen arkkitehtuurien rakentamisessa olevaan tuttuun käsitteeseen nimeltä tiedon mallinnus (engl. data modeling), ja tiedon mallinnuksen teoriaa käytetään vaatimusmäärityksen tietokantojen kuvauksissa ja kaavioissa.

Dokumentilla voidaan myös tuoda esille erittäin yksinkertaisia asioita, jotka joskus olivat päivänselviä, mutta ajan myötä niiden merkitys olisi voinut haihtua, esimerkiksi sidosryhmien luetteleminen. Sidosryhmillä tarkoitetaan kaikkia eri segmentoituja ryhmiä, jotka hyötyvät kehitettävästä järjestelmästä. Tärkeimpiin asioihinhan kuuluu sidosryhmien analysointi ja tutkiminen. Ryhmien tuntemisen hyötyihin kuuluu paremmin suunniteltu eri ryhmien tarpeiden määrittely, koska eri ryhmiin voi kuulua sovelluksen kehittäjäorganisaatio, tilaaja, hyötyjä ym. kuten käyttäjät. Tietämällä tarkasti eri ryhmät voidaan suunnitella haastattelu ja tapaamisia eri ryhmän osapuolien kanssa, mikä itsessään on laaja käsite ja osa-alue. (Young 2004, 4; Hull 2011, 163.)

Vaatimusmäärityksen laatimisen hyödyt ovat projektin kokonaiskuvaa otettaessa huomioon vähintään seuraavanlaiset, jossa dokumentaatio

- vähentää sovelluksen kehittämiseen käytettyä aikaa
- vähentää sovelluksen arkkitehtuuri ja logiikkavirheitä
- vähentää kehittämiseen käytettäviä kustannuksia
- lisää asiakastyytyväisyyttä (esim. oikeanlaiset toiminnollisuudet jne.)

- lisää ja selventää kokonaiskuvaa kehitettävän sovelluksen toiminnollisuuksista ja integraatioista, mitä sovellus tekee ja ei tee.

Vaatimusmääritys dokumentin hyödyt tiivistettynä voivat olla seuraavanlaisia:

- dokumentilla voidaan selvästi viestittää asiakkaalle sovelluksen toiminnot
- asiakas voi viestittää selkeämmin kehitystiimille, mitä hän haluaa sovellukselta
- kehitystiimi voi suunnitella kunnollisen sovelluksen
- määritysten hyväksyntään erikoistunut tiimi voi käydä läpi kaikki vaatimuksen sovellukselle yksitellen ja merkata paremmin projektin tilanteen
- sovelluskehitys tiimi voi laskea kokonaiskustannukset paljon yksityiskohtaisemmin.

4 Pohdinta

Opinnäytetyön aihe-alueena oli pääasiallisesti käydä vaatimusmääritystyö prossin läpi ja tutkia eri prosessikohdan määritystyön kulkua ja sisältöä ottamalla huomioon opinnäytetyön aiheena olevan kehitettävä sovellus. Kehitettävän sovelluksen taustatiedoissa käytiin läpi opinnäytetyön aloittamisen syy, eri sidosryhmät, toiminnallisen työn tausta ja toimeksiantajan organisaatio. Taustatietojen lähteinä käytettiin toimeksiantajalta saatuja materiaaleja ja eri sähköisiä artikkeleita.

Opinnäytetyön teoriaosuudessa käytiin läpi määritystyön prosessi, jossa yritetään tuoda esille eri määritystyön hyödyt ja työn sisältö, käyttämällä eri kirjallisia ja elektronisia lähteitä. Teoriaosuuden määritystyön prosessin eri vaiheille käytettiin myös eri casetapauksia täydennyksesi eri haitoista ja niiden vaikutuksista, jos määritystyön prosessia ei noudateta kunnolla tai määritystyötä ei noudateta ollenkaan. Teoriaosuutta pyrittiin ohjaamaan käyttämällä HAAGA-HELIASSA toteutettujen edellisten ohjelmistokehitys projektien eri ominaisuuksien onnistumisten ja epäonnistumisten syitä eri määritystyön prosessin vaiheissa ja tuomalla eri casetapauksia peilaamalla kokemuksen kautta saatuja tietoja, jotta määritystyön tarpeellisuus saataisiin mahdollisimman selkeästi esille ottamalla huomioon opinnäytetyön aiheena oleva kehitettävän sovelluksen määritystyön prosessi, jotta siitä saataisiin mahdollisimman oikea-oppisesti rakennettu dokumentaatio luovutettavaksi asiakkaalle.

Opinnäytetyön laatimisessa sain paljon perustietoa määritystyön kulusta niin teoriassa kuin käytännössä ja sain osaamista määrittelytyöhön kehitettävälle sovellukselle. Sain myös paljon yksityiskohtaista tietoa vaatimusmääritystyön dokumentoinnin oikea-oppisesta laatimisesta ja vaatimusmääritys prosessin hallinnoimisesta. Opin myös eri syitä siihen, että miksi tiety vaatimusmääritys työt ovat virheellisiä ja miten virheitä voitaisiin välttää tai vähentää, jotta sovelluskehitys projektit eivät kaatuisi tai valmistuisi väärillä ominaisuuksilla.

Eri teoria-aineistoja käymällä läpi sain myös suuren määrän tietoa eriävistä mielipiteistä vaatimusmäärityksen termeistä ja määritystyön eri prosessikohdan työnkulusta, koska yhtä standardoitua pakollista tapaa ei ole prosessin läpikäymiseksi ja mitään tiettyä pohjamalleja ja standardeja ei ole eri määritystyön kohdissa. Vaatimusmäärittelijöiden osaamisen taso ja kokemus näkyi erilaisissa julkaistuissa casetapauksissa ja tieteellisissä artikkeleissa. Vaatimusmäärittelijöillä tosin on tietty yhteneväinen linja siitä mitä vaatimusmääritys dokumentin päämäärä on ja tietyt yleiset termit kuten toiminnallisuudet ja ei-toiminnallisuudet ominaisuudet tarkoittavat. (Paakki 2011.)

Vaatimusmääritystyön ja sen prosessin ja tarkoituksen ymmärtäminen voi ehkäistä tai vähentää sidosryhmien erinäisten kiistojen takia aiheutuvia virheitä (yleensä ristiriitaiset tavoitteet), huonosti hallittuja projekteja, kehnoja sovelluskehitys käytäntöjä, kustannusten vääriä arviointeja, väärin tai puutteellisesti määriteltyjä vaatimuksia ynnä muita hallintoon ja epäselvyyteen liittyviä asioita. (Carlos 2014.)

Vaatimusmäärityksen laadullisuus riippuu noudatetun prosessin tyypin mukaan eli onko iteroiva vai vesiputous malli kyseessä ja miten tarkasti on noudatettu IEEE 830-vuosiluku tai Voleren vaatimusmääritys työn pohjamalleja, jotka noudattavat IEEE hyvän vaatimusmääritystyön käytäntöjä ja pohjia. Vaatimusten määritysten laadullisuus riippuu myös siitä, että miten jäljitettäviä, tarkkoja, selkeitä ja yhteneväisiä ne ovat eli mitään ristiriitaisuuksia ei saa olla ja määritys pitää olla tarpeeksi selkeä järjestelmää rakentaville ohjelmoijille.

Sovelluskehitys projekteissa, määritystyön tärkeys riippuu hallinnoitavan tiimin ja projektin laajuuden mukaan. Agile menetelmien käyttäminen esimerkiksi suurissa projekteissa voi suurella todennäköisyydellä johtaa epäonnistumiseen, koska iteroiva prosessi malli (jota Agile menetelmä noudatta eri asteissa) ei sovi vankkaa perustaa tarvitseville projekteille, jossa suunnitteleminen ja testaaminen on tärkeydeltään ensisijainen. (Ditmore 2013)

Hyvänä esimerkkinä määritystyön ja vaatimusdokumentin oikean standardinmukaisen rakenteen tärkeydestä nousi jokaisen HAAGA-HELIAN ohjelmistokehitys kurssin aikana, joissa pääasiallisina ongelmina olivat aina dokumentaation ajantasaisuudesta johtuvat ongelmat. Ongelmiin kuuluivat muun muassa päämäärän löytäminen, koska kirjaamatonta tietoa ei kukaan tule muistamaan myöhemmin ja/tai väärin kirjattu tieto, joka rakennettaan juuri miten se on dokumentoitu eli toisin sanoen saadaan väärällä ominaisuuksilla toimiva järjestelmä, jota joudutaan jatkokehityksessä muuttamaan. Oikean yrityksen tilanteessa tällaiset virheet voivat johtaa satojen tuhansien ellei miljoonien eurojen vahinkoihin siinä mielessä, että lisäkustannukset olisi voitu välttää tarkalla suunnittelulla. Edellä mainitulla tarkoitettiin myös suunnitteludokumentin ja siihen sisältyvän työ huomioiminen vaatimusmääritysten pohjalta.

Oikean maailman casetapauksia tutkiessa tuli monta kertaa esille, että kuinka monen projektin kohdalla olisi voitu välttää monia eri ongelmia, jotka kuulostavat erittäin päivänselviltä ja yksinkertaisilta asioilta ulkopuolisille, mutta monimutkaistuvat yleensä projekteissa ja niitä rakentavissa sovelluskehitysyrityksissä jos hyvän dokumentoinnin käytäntöjä ja vaatimusmäärityksprosessia ei noudatettu.

Vaatimusmääritys dokumentaation laadullinen sisältö olisi voinut myös monen eri epäonnistuneen suurprojektin kohdalla estää projektin alulle saattamisen, koska näissä kyseisissä projekteissa ainoastaan järjestelmän päämäärä ja eri pääkäyttötapaukset olivat selkeästi tiedossa, mutta kunnollista dokumentaatiota ja suunnittelua ei oltu yhtään tehty tai ne olivat vähäisiä ja niissä oli paljon epäkohtia, jolloin myös kustannukset olivat jatkuvasti kasvavia, koska mitään tiedossa olevaa täydellistä tietoa ei ollut kaikista vaadittavista töistä. Kattavalla kelpoisuusselvityksellä olisi voitu monesti välttää tuleva ohjelmistokatastrofi. Syitä edellä mainittuun voi olla ylityöllistetty IT osasto, vähäiset resurssit tai tiukka aikataulu. (Florentine 2013; Kogekar 2013; Paakki 2011, 15.)

Vaatimusmääritys prosessin kokonaisuuden hahmottaminen ei loppujen lopuksi ollut kovin hankalaa, mutta toisaalta yksi asia, joka jatkuvasti tuli esille oli se, että mitään tiettyä yhtä tapaa ei käytetä. On aina olemassa monta eri vaatimusmääritys prosessin toteutus mallia. Jos sovellukseen tarvitaan vaatimusmäärittelyä, niin paras tapa on aina valita yksi ristiriidaton ja kokonainen prosessimalli tai sellainen pitää muokata olemassa olevista prosessimalleista omaan projektiin sopivaksi. Paras malli löytyy aina niiltä henkilöiltä, joilla on paljon kokemusta vaatimusmäärityksestä ja jotka osaavat räätälöidä jokaiselle projektille kokoon ja vaatimuksiin nähden sopivimman prosessikulun ja dokumentaatiopohjan (IEEE-En 830 vaatimusmäärityksen dokumenttipohjia).

Vaatimusmääritys prosessin osaaminen voidaan rinnakkaistaa arkkitehtien työhön. Määritystyö ja sen tuloksena saatu dokumentaatio (unohtamatta suunnitteludokumenttia) on samanlainen tuotos kuin arkkitehdin laatima rakennekaava. Ohjelmistokehittäjien huippuosaaminen ei takaa menestyvää lopputulosta, ilman suunnittelua ja hyvää kokonaiskuvaa tehtävästä työstä. Oikean maailman rakennusprojekteissa on myös huippuinsinöörejä, mutta ilman minkäänlaista kaavaa ja pohjamallia tehtävästä työstä, niin työtä ei voida tehdä. Ei voida edetä askel askeleelta ja rakentaa talon tai projektin osa kerrallaan. Tuotos voi olla erittäin kehno ja jopa vaarallinen ja sama pätee sovelluskehitys projekteihin.

Vaikka jos dokumentaatiota ei aiota kirjoittaa kattavasti ennen sovelluskehitysprojektin alkua, niin pari hyvää varteenotettavaa ohjetta ja hyvää käytäntöä on varsinkin Agile menetelmiä käyttäville. Yksi hyvä käytäntö on, että tiivis asiakasyhteistyö ja kommunikointi ei katoa projektin eri vaiheissa ja tuotteen omistajan mielipiteitä otetaan huomioon unohtamatta muita sidosryhmiä. Näin voidaan vähentää laajaa dokumentaatiota karttavien sovelluskehitys tiimien virhemarginaalia ja vähitellen tutustuttaa heidät dokumentaation ja vaatimusmäärityksen hyötyihin. Määritystyön ei tarvitse myöskään olla vesiputousmallin tietty osa, koska vaatimusmääritys voidaan toteuttaa samaan aikaan kuin ohjelmistokehitys on meneillään, kunhan kaikki vähimmäistiedot ovat tiedossa projektin aloittamista varten.

Huono kommunikaatio asiakkaan ja sidosryhmien kanssa on osoittautunut monen projektin kohtaloksi. Suurin osa kiistelyistä ja käräjöinneistä epäonnistuneissa sovelluskehityksissä, johtuu asiakaan kanssa käydyistä vähäisistä vaatimusmäärityksistä. Näissä projekteissa dokumentaatio on aina ollut erittäin pinnallista ja alkeellista. (Robertson 2013, 5-7; Kolehmainen 2014.)

Epäonnistuneet ratkaisut ja asiat opinnäytetyössä oli vaatimusmääritys dokumenttiin sisällytetyt suunnittelu dokumentin aihe-alueiden tiedot. Opinnäytetyössä ja vaatimusmäärityksessä epätäydellinen rakenne, koska ennen aloittamista minulla ei ollut paljon kokemusta määritysdokumentin rakenteesta ja prosessista ja eri sekaannuksia oli sattunut ajan myötä termistöjen ja käsitteiden ymmärtämisen kanssa. Valintojen osalta opinnäytetyön aihe olisi voitu rajata vain suunnitteludokumentin laatimiseen kehitettävälle sovellukselle käyttämällä hyväksi jo määriteltyjä dokumentteja, jotta toimeksiantajalle olisi saatu mahdollisimman suuri hyöty sovellukselle laaditusta dokumentoinnista. Olisin voinut myös tutkia opinnäytetyön aihe-alue eli vaatimusmääritys prosessi paljon tarkemmin, jotta olisin saanut kokonaiskuvan ennen opinnäytetyön rakenteen laatimista.

Lähteet

Ainasvuori, O. 2014. Taito. Taloushallinnon automaatio uudelle tasolle. Luettavissa: http://www.taitoa.fi/ajankohtaista-taloushallinnon_automaatio_uudelle_tasolle. Luettu: 30.9.2014.

Aalto-yliopisto 2014. Real-Time Economy – osaamiskeskus. Luettavissa: http://information.aalto.fi/fi/research/rte/. Luettu:3.9.2014.

Bell, T. E. & Thayer, T. A. 1979. Software Requirements: Are they really a problem? TRW Defense and Space Systems Group. Luettavissa: http://pdf.aminer.org/000/361/405/software_requirements_are_they_really_a_problem.pdf. Luettu: 1.9.2014.

Bass, L & Bergey, J. & Clements, P. & Merson, P. & Ozkaya, I. & Sangwan, R. 2006. Carnegie Mellon Software Engineering Institute. A Comparison of Requirements Specification Methods from a Software Architecture Perspective. Luettavissa: http://www.sei.cmu.edu/reports/06tr013.pdf. Luettu: 5.9.2014.

Bank of America Merrill Lynch 2014. Opportunity Knocks on the Future of ISO 20022. Luettavissa: http://corp.bankofamerica.com/documents/10157/67594/Opportunity_knocks_on_the_Future_of_ISO_20022.pdf. Luettu: 3.9.2014.

Bo, H. 2013. Suomen Pankki. Maksu-liikepalvelut uusiin arvodimensioihin Luettavissa: http://www.suomenpankki.fi/fi/rahoitusjarjestelman_vakaus/maksuneuvosto/documents/06_bo_harald.pdf. Luettu: 30.8.2014.

Carlos, T. 2014. Reasons Why Projects Fail. Luettavissa: http://www.projectsmart.co.uk/reasons-why-projects-fail.php. Luettu: 2.10.2014.

Charette, R. N. 2005. Why Software fails. IEEE SPECTRUM. Luettavissa: http://spectrum.ieee.org/computing/software/why-software-fails. Luettu: 26.11.2014.

Cockburn, A. 2001. Writing Effective Use Cases. Luettavissa: http://alistair.cockburn.us/get/2465. Luettu: 17.10.2014.

Dr. Cremers, A.B. & Alda, S. Non-functional Requirements. Luettavissa: http://www.iai.uni-bonn.de/III/lehre/vorlesungen/SWT/RE05/slides/09_Non-functional%20Requirements.pdf. Luettu: 18.9.2014.

Ditmore, J. 2013. Why do Big IT Projects Fail So Often. Luettavissa: http://www.informationweek.com/strategic-cio/executive-insights-and-innovation/why-do-big-it-projects-fail-so-often/d/d-id/1112087? Luettu: 5.9.2014.

E. Hull, K. Jackson, J. Dick. 2011. Requirements Engineering. 3. uudistettu painos. Springer.

E. Karl. 2007. Software requirements specification and the IEEE standard. Techtarget. Luettavissa: http://searchsoftwarequality.techtarget.com/answer/Software-requirements-specification-and-the-IEEE-standard. Luettu: 31.8.2014.

Euroopan komissio 2014. Single Euro Payments Area (SEPA).

Luettavissa: http://ec.europa.eu/internal_market/payments/sepa/index_en.htm. Luettu: 31.10.2014.

Euroopan Keskuspankki 2006. YHTENÄINEN EUROMAKSUALUE (SEPA). Luettavissa: https://www.ecb.europa.eu/pub/pdf/other/sepa_brochure_2006fi.pdf. Luettu: 17.10.2014.

European Payments Council 2014. EPC List of SEPA Scheme Countries. Luettavissa: http://www.europeanpaymentscouncil.eu/index.cfm/knowledge-bank/epc-documents/epc-list-of-sepa-scheme-countries/epc409-09-epc-list-of-sepa-scheme-countries-v2-0-january-201/. Luettu: 18.10.2014.

Finanssialan keskusliitto 2012. Yhtenäisen Euromaksualueen Toteutuminen Suomessa. Luettavissa: http://www.fkl.fi/teemasivut/sepa/tekninen_dokumentaatio/Dokumentit/SEPA_siirtymasuunnitelma_v5.pdf. Luettu: 4.9.2014.

Finanssialan keskusliitto 13.7.2012. Finvoice-välityspalvelun kuvaus ja ehdot. Luettavissa: http://www.fkl.fi/teemasivut/finvoice/finvoice-tuotekuvaus/Dokumentit/verkkolaskuvalitys.pdf. Luettu: 1.9.2014.

Fleck, D. 2009. Requirements Spec Revisited. Luettavissa: http://cs.gmu.edu/~dfleck/classes/cs421/fall07/slides/FunctionalRequirements.ppt. Luettu: 17.9.2014.

Finanssialan keskusliitto 2012. ISO 20022 MAKSAMISOPAS. Luettavissa: http://www.fkl.fi/teemasivut/sepa/tekninen_dokumentaatio/dokumentit/iso20022_maksut.pdf. Luettu: 3.10.2014.

Finanssialan Keskusliitto 2010. Ympäristöystävällinen verkkolasku. Luettavissa: http://www.fkl.fi/materiaalipankki/tutkimukset/Dokumentit/Ymparistoystavallinen_verkkolasku.pdf. Luettu:3.9.2014.

Forbes 2011. Going Somewhere?
Robertson S. & J. 13.8.2007. Volere Requirements Specification Template. Luettavissa: http://www11.informatik.uni-erlangen.de/Lehre/SS2014/PR-SWE/Material/volere-template.pdf. Luettu: 12.11.2014.

Gulla, J. IBM Systems Magazine 2012. Seven Reasons IT Projects Fail. Luettavissa: http://www.ibmsystemsmag.com/mainframe/tipstechniques/applicationdevelopment/project_pitfalls/. Luettu: 10.11.2014.

Goldsmith, R. F. 1.6.2009. Techtarget. Differentiating between Functional and Nonfunctional Requirements. Luettavissa: http://searchsoftwarequality.techtarget.com/answer/Differentiating-between-Functional-and-Nonfunctional-Requirements. Luettu: 17.9.2014.

Goldsmith, R. F. 1.11.2009. Techtarget. Functional vs. non-functional requirements, what is the difference? Luettavissa: http://searchsoftwarequality.techtarget.com/answer/Functional-vs-non-functional-requirements-what-is-the-difference. Luettu: 17.9.2014.
Luettavissa: http://www.forbes.com/sites/greggfairbrothers/2011/11/29/going-somewhere-2/. Luettu: 23.10.2014.

Hämeen kauppakamari 2010. Real-Time Economy – ohjelma.
Luettavissa:
http://www.hamechamber.fi/easydata/customers/hame/files/liiketoimintaa_verkossa_-/04_vuokko_makinen_2411_rte_ja_fia2_ja_raportointikoodisto_2010_intro.pdf. Luettu: 8.10.2014.

ICT 2015. 2014. Polku 2: Rakennetaan Yritysten Reaaliaikaisen Talouden Vaatima Infrastruktuuri. Luettavissa: http://ict2015.fi/toimenpiteet/polku-2-rakennetaan-yritysten-reaaliaikaisen-talouden-vaatima-infrastruktuuri/. Luettu 30.10.2014.

IEEE Xplore® Digital Library 2012. 830-1984 – IEEE Guide To Software Requirements Specifications. Luttavissa:
http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=278253. Luettu: 30.10.2014.

IEEE 2014. About IEEE. Luettavissa: https://www.ieee.org/index.html. Luettu: 3.9.2014.

Jackson, W 2013. What Characteristics Make Good Agile Acceptance Criteria?. Luettavissa: http://www.seguetech.com/blog/2013/03/25/characteristics-good-agile-acceptance-criteria. Luettu 17.11.2014.

Janssen, C. Business Object (BO). techopedia**.**
Luettavissa: http://www.techopedia.com/definition/25982/business-object-bo. Luettu: 2.11.2014.

Jansson, F. 4.9.2014. Tuoteomistaja. Tieto kehitysprojektin alustavat tiedot. HAAGA-HELIA ammattikorkeakoulu. Asiakastapaaminen. Helsinki.
Luettu: 31.10.2014.

JUHTA-julkisen hallinnon tietohallinnon neuvottelukunta 2012. Liite 1: Käyttötapauslomake – mallipohja.
Luettavissa: http://docs.jhs-suositukset.fi/jhs-suositukset/JHS173_liite1/JHS173_liite1.pdf. Luettu: 1.9.2014.

JUHTA-julkisen hallinnon tietohallinnon neuvottelukunta 2012. JHS 173 ICT – palvelujen kehittäminen: Vaatimusmäärittely.
Luettavissa: http://docs.jhs-suositukset.fi/jhs-suositukset/JHS173/JHS173.pdf. Luettu 1.9.2014.

Kolehmainen, A. 2014. Tulli heitti CGI:n miljoonien eurojen it-työn roskiin. Luettavissa: http://summa.talentum.fi/article/tv/uutiset/113927. Luettu 30.11.2014.

Kogekar, H. 2013. Why IT projectsreally fail. Luettavissa:
http://www.cio.com.au/article/533532/why_it_projects_really_fail/. Luettu: 8.10.2014.

Kelley, M. B. 19.11.2013. Business Insider. Why The Obamacare Website Failed In One Slide. Luettavissa: http://www.businessinsider.com/why-the-healthcaregov-website-failed-at-launch-in-one-slide-2013-11. Luettu: 7.9.2014.

Le Vie, D. Jr. 2010. TECHWHIRL. Writing Software Requirements Spcecifications.
Luettavissa: http://techwhirl.com/writing-software-requirements-specifications/. Luettu:3.9.2014.

Microsoft 2014. UML Class Diagrams: Guidelines.
Luettavissa: http://msdn.microsoft.com/en-us/library/dd409416.aspx. Luettu: 2.11.2014.

Nishadha, S. 2.2.2014. The Complete Guide to UML Diagram Types with Examples. creately. Luettavissa: http://creately.com/blog/diagrams/uml-diagram-types-examples/. Luettu: 3.9.2014.

Oracle 2007. Getting Started With UML Class Modeling. Luettavissa: http://www.oracle.com/technetwork/developer-tools/jdev/gettingstartedwithumlclassmodeling-130316.pdf. Luettu 2.11.2014.

Osuuspankki 2014. SEPA. Luettavissa: https://www.op.fi/media/liitteet?cid=151539783&srcpl=4

OMG (Object Management Group) 2010. OMG Unified Modeling Language, Infrastructure. Luettavissa: http://www.omg.org/spec/UML/2.4.1/Infrastructure/PDF. Luettu: 9.9.2014.

Paakki, J. 2010, Helsingin Yliopisto. Ohjelmistojen vaatimusmäärrittely. Luettavissa: http://www.cs.helsinki.fi/u/paakki/Vaatimus-11-Luentokalvot-1.pdf. Luettu: 5.9.2014.

Penttinen, E. 2008. Electronic Invoicing Initiatives in Finland and in the European Union. Luettavissa: http://epub.lib.aalto.fi/pdf/hseother/b95.pdf. Luettu: 13.11.2014.

Dr. Paul, D. 1995. Top 10 Reasons Why Systems Projects Fail. Dulcian, Inc. Luettavissa: http://www.hks.harvard.edu/m-rcbg/ethiopia/Publications/Top%2010%20Reasons%20Why%20Systems%20Projects%20Fail.pdf

Real-Time Economy –ohjelma 2010. Fully Integrated Accounting 2 -hanke Luettavissa: http://www.hamechamber.fi/easydata/customers/hame/files/liiketoimintaa_verkossa_-/04_vuokko_makinen_2411_rte_ja_fia2_ja_raportointikoodisto_2010_intro.pdf. Luettu: 1.10.2014.

Robertson, S. & J. 2013. Mastering the Requirements Process. 3 painos. Pearson.

Rouse, M. Techtarget 2007. Software Requirements Specification (SRS). Luettavissa: http://searchsoftwarequality.techtarget.com/definition/software-requirements-specification. Luettu: 27.10.2014.

Toikka, S. 2012. Rea-Time Economy Update. Luettavissa: https://www.vero.fi/download/RealTime_Economy_Update_pdf/%7BB363385E-C82D-42B0-8CDA-5BEE20A85D12%7D/7612. Luettu: 14.11.2014.

Software Engineering Standards Committee of the IEEE Computer Society 1998. IEEE Recommended Practice for Software Requirements Specifications. Luettavissa: http://www.math.uaa.alaska.edu/~afkjm/cs401/IEEE830.pdf. Luettu 13.11.2014.

Sandahl, K. 2009. Model-Based Requirements Engineering. Luettavissa: http://www.ida.liu.se/~petfr/MODPROD2010talks/Tutorials/modprod2010-Tutorial3-Kristian-Sandahl-Requirements-analysis.pdf. Luettu: 28.10.2014.

Sommerville, I. 2011. Software Engineering. Pearson.

Sommerville, I. 2008. The IEEE standard for requirements documents. http://ifs.host.cs.st-andrews.ac.uk/Books/SE9/Web/Requirements/IEEE-standard.html

Stellman, A. & Greene, J. 2006. Applied Software Project Management. O'REILLY. California. Luettavissa: http://computercollege.ac.ae/itbooks/6.pdf. Luettu: 25.10.2014.

Stephen Armitage 1996. Software Requirements Specification.
Luettavissa: http://www4.informatik.tu-muenchen.de/proj/va/SRS.pdf. Luettu: 30.10.2014.

Florentine, S. 2013. Why Are So Many IT Projects Failing? Luettavissa:
http://www.cio.com/article/2380469/careers-staffing/why-are-so-many-it-projects-failing-.html. Luettu 1.10.2014.

The Unified Modeling Language, 2014. Class Diagrams.
Luettavissa: http://www.uml-diagrams.org/class-diagrams.html. Luettu: 2.11.2014.

Tietoyhteiskunnan kehittämiskeskus (TIEKE) 2014. Verkkolaskusta.
Luettavissa: http://www.tieke.fi/display/verkkolasku/Verkkolasku. Luettu:2.9.2014.

Toikka, S. 2012. Vero. Real-Time Economy Update. Luettavissa:
https://www.vero.fi/download/RealTime_Economy_Update_pdf/%7BB363385E-C82D-42B0-8CDA-5BEE20A85D12%7D/7612

Taitoa 2014. Taloushallinnon automaatio uudelle tasolle.
Luettavissa: http://www.taitoa.fi/ajankohtaista-taloushallinnon_automaatio_uudelle_tasolle. Luettu: 29.10.2014.

Tietoyhteiskunnan kehittämiskeskus 2014. Verkkolaskuista.
Luettavissa: http://www.tieke.fi/display/verkkolasku/Verkkolasku. Luettu: 10.10.2014.

Tieto Oyj. 2014. Tieto promotes adoption of ISO 20022 e-invoice standard.
Luettavissa: http://www.tieto.com/services/business-process-services/business-information-exchange/e-invoicing-solution-makes-it-easy/tieto-promotes-adoption-iso-20022-e-invoice-standard. Luettu: 20.9.2014.

Ultrecht University: Department of Information and Computing Sciences 2010. Software Requirements Specification Document.
Luettavissa: http://www.cs.uu.nl/wiki/Spm/SpecificationDocument. Luettu: 25.10.2014.

Vero 2014. Kausiveroilmoituksen antaminen. Yritys ja yhteisöasiakkaat.
Luettavissa: http://www.vero.fi/fi-FI/Yritys_ja_yhteisoasiakkaat/Kausiveroilmoitus. Luettu:28.8.2014.

Wiegers, K. & Beatty, J. 2013. Software Requirements. 3 painos. Microsoft Press.

Wiegers, K.E. 2007. TechTarget. Software requirements specification and the IEEE standard.
Luettavissa: http://searchsoftwarequality.techtarget.com/answer/Software-requirements-specification-and-the-IEEE-standard. Luettu:1.9.2014.

Wiegers, K. E. 17.1.2013. Konsultti. Thorny Issues in Software Requirements. Modern analyst: Webinar series. Seminaari. Katsottavissa:
https://www.youtube.com/watch?v=kqk5tW5FY8c. Katsottu: 6.9.2014.

W3Schools 2014. Extensible Markup Language (XML).
Luettavissa: http://www.w3.org/XML/. Luettu: 3.9.2014.

Westfall, L. 2006. The Why, What, Who, When and How of Software Requirements. Luettavissa:

http://users.csc.calpoly.edu/~csturner/courses/300f06/readings/%5B3%5D_%20The_Why_What_Who_When_and_How_of_Software_Requirements.pdf. Luettu: 26.11.2014.

Young, R. R. 2004. The Requirements Engineering Handbook. Artech House.

Liitteet

Liite 1. Vaatimusmääritys dokumentti

(Salainen)

Liite 2. Vaatimusmäärityksen luokkakuvaukset ja – kaaviot

(Salainen)

Liite 3. Vaatimusmäärityksen käyttötapaukset-, kaaviot- ja sekvenssikaaviot

(Salainen)