VAATIMUSTEN MÄÄRITTELY JA HALLINTA RAJAVARTIOLAITOKSEN TIETOJÄRJESTELMÄHANKKEISSA

Mika Ylinen

Tampereen yliopisto
Tietojenkäsittelytieteiden laitos
Ohjelmistokehityksen maisteriohjelma
Pro Gradu-tutkielma
Ohjaaja: Jyrki Nummenmaa
Syyskuu 2009

Tampereen yliopisto
Tietojenkäsittelytieteiden laitos
Ohjelmistokehityksen maisteriohjelma
Tekijän nimi: Mika Ylinen
Pro gradu-tutkielma, 51 sivua
Syyskuu 2009

TIIVISTELMÄ

Tämän tutkielman tavoitteena on kartoittaa vaatimusmäärittelyn kustannustehokkuuteen vaikuttavia tekijöitä Rajavartiolaitoksen tietojärjestelmähankkeita varten. Vaatimusmäärittely on keskeinen osa laajempaa vaatimusprosessia, joka pitää sisällään ongelman määrittelyn, vaatimusten keräämisen ja analysoinnin sekä vaatimusten määrittelyn ja hallinnan. Vaatimusprosessi on kokonaisuudessaan kriittinen osa järjestelmäkehitystä. Jotta järjestelmähanke voitaisiin viedä läpi suunnitellusti aikataulussaan ja lopputuloksena saataisiin mahdollisimman hyvin hankkeen sidosryhmien tarpeita vastaava järjestelmä, on kaikki vaatimusprosessin osa-alueet otettava toteutuksessa huomioon. Tämä on keskeistä myös vaatimusmäärittelyn kustannustehokkuuden kannalta. Jotta voitaisiin tuottaa oikeat vaatimukset aikataulussaan ja kohtuullisella työpanoksella, on vaatimusprosessin kokonaisuudessaan oltava toimiva. Ensin on tarpeen määritellä ongelmat, jotka järjestelmän odotetaan ratkaisevan. Kun tavoitteet ovat selvillä, kerätään vaatimusehdotukset ja analysoidaan ne huolella läpi. Ongelmien määrittely sekä vaatimusehdotusten kerääminen ja analysointi ovat vaatimusten määrittelyä edeltävää välttämätöntä pohjatyötä, joka edesauttaa oikeiden vaatimusten löytymistä ja määrittelyä. Lopuksi määritellyt vaatimukset siirretään vaatimusten hallintaan, jonka tavoitteena on valvoa vaatimuksiin kohdistuvia muutoksia ja arvioida niiden vaikutusta projektiin kokonaisuudessaan.

Tutkimukseni pääpaino on vaatimuskehityksessä, joka pitää sisällään vaatimusten keräämisen ja analysoinnin sekä vaatimusmäärittelyn, minkä lisäksi perehdyn vaatimusten hallintaan. Selvitän, mitä vaatimusten kerääminen, määrittely ja hallinta pitävät sisällään ja mitä näiden osa-alueiden osalta on otettava huomioon, jotta onnistunut ja kustannustehokas vaatimusmäärittely olisi mahdollista. Lopuksi analysoin vaatimusten keräämiseen suunniteltujen tekniikoiden soveltuvuutta Rajavartiolaitoksen tarpeisiin niiden teknisen vaativuuden, työmäärän ja kustannustekijöiden perusteella.

Avainsanat: Vaatimusmäärittely, vaatimuskehitys, vaatimusprosessi, tietojärjestelmä-hanke

University of Tampere
Department of Computer Sciences
Master of Science Degree Programme in Software Development
Author: Mika Ylinen
M.Sc. thesis, 51 pages
September 2009

ABSTRACT

The purpose of this research is to examine the process of requirements definition in a data system project based on the operational needs of an organization. Finnish Border Guard is presented as an example of an organization that is acquiring data systems and needs a cost effective way to run the requirements definition process.

The objective of this study is requirements engineering process including gathering, analyzing, defining and managing of requirements. The focus and main question of the research is how the requirements can be produced, described and managed reliable and cost effective way. To answer the needs of data system projects in the example organization, also technical methods for gathering requirements are examined. Those methods are analyzed based on their technical challenges, workload and cost factors. This research is based on theory and literal study.

Keywords: Requirements Definition, Requirements Engineering, Requirements Process, Data System Project

1 Johdanto

Käsittelen tässä tutkielmassa vaatimusmäärittelyä osana laajempaa vaatimusprosessia, joka koostuu tarpeiden kartoituksesta ja muuntamisesta vaatimuksiksi, vaatimuskehityksestä pitäen sisällään vaatimusten keräämisen, analysoinnin ja määrittelyn sekä vaatimusten hallinnasta. Keskityn vaatimuskehitykseen ja erityisesti vaatimusten keräämisessä ja määrittelyssä käytettyihin tekniikoihin. Työn tavoitteena on vertailla erilaisten vaatimusten keräämiseen ja määrittelyyn liittyvien tekniikoiden soveltuvuutta Rajavartiolaitoksen tietojärjestelmäkehitykseen ja löytää kustannustehokas tapa vaatimusmäärittelyprosessin läpiviemiseksi.

2 Aihealueen esittely

Vaatimusmäärittely on yksi tietojärjestelmähankkeissa hyödynnettävistä systeemityömenetelmistä, joiden lopputuloksena tietojärjestelmä syntyy. Rajavartiolaitoksen tietojärjestelmähankekäsikirjan uudistuksen tavoitteena on yhtenäistää laitoksessa käytettäviä systeemityömenetelmiä siten, että hankkeet ja projektit pysyvät hyvin hallinnassa ja niiden tulokset ovat yhdenmukaisia. Vaatimusmäärittely on keskeinen osa systeemityötä ja tämä tutkielma tukee osaltaan Rajavartiolaitoksessa meneillään olevaa selvitystä. Omat kokemukseni tässä mainitsemistani systeemityömenetelmistä ja tietojärjestelmähankkeista ajoittuvat vuosille 2005 ja 2006, jolloin työskentelin Rajavartiolaitoksessa atk-suunnittelijana. Osa tietojärjestelmähankkeista toteutettiin talon sisällä ja osassa toimittiin yhteistyössä ulkopuolisten sovellustoimittajien kanssa. Hankkeiden luonne vaihteli uusien järjestelmien perustamisesta jo olemassa olevien järjestelmien kehitystyöhön.

Vaatimusmäärittelyvaihe on kriittinen osa järjestelmäkehitystä. Vaatimusmäärittelyn avulla pyritään luomaan selkeä kuva tietojärjestelmähankkeen tavoitteista ja lopputuloksena syntyvästä järjestelmästä kaikille hankkeen sidosryhmille. Varsinkin hankkeen alkuvaiheessa eri osapuolilla saattaa olla hyvin erilaiset mielikuvat tavoitteista ja lopputuloksesta, mutta keskeisten toiminnallisten ja teknisten tarpeiden kartoituksella voidaan edesauttaa yhteisten vaatimusten ja päämäärän löytymistä. Kun tähän on päästy, voidaan hankkeessa edetä vaatimusmäärittelyn viitoittamaa tietä.

Tietojärjestelmähankkeen tavoitteena on tuottaa järjestelmä, joka toimii määriteltyjen vaatimusten mukaisesti. Jotta vaatimukset voidaan huomioida toteutusvaiheessa toivotulla tavalla, niiden taustalla olevat prosessit ja käyttötilanteet on määrittelyvaiheessa kuvattava riittävällä tarkkuudella. Määritellyt vaatimukset on myös dokumentoitava

asianmukaisesti, jotta järjestelmään toteutettavat toiminnot ja ominaisuudet ovat helposti jäljitettävissä niihin. Hankkeen lopputuloksen arvioinnin kannalta on mielekästä kysyä, kuinka vaatimukset toteutuvat valmiissa järjestelmässä.

2.1 Tutkimustilanne

Vaatimusmäärittelyä on tutkittu ja aiheesta on kirjoitettu paljon niin kansainvälisesti kuin Suomessakin. Tutkimusten näkökulmat vaihtelevat vaatimusmäärittelyn yleisistä teorioista, menetelmistä ja tekniikoista järjestelmä- ja sovelluskohtaisen vaatimusmäärittelyn case-tutkimuksiin, vaatimusmäärittelyn prosessien kehittämiseen sekä vaatimusten luokitteluun. Oma tutkielmani pohjautuu kirjallisuusselvitykseen vaatimusmäärittelyn teorioista ja käytännöistä. Työni pääpaino on toiminnallisten vaatimusten keräämisessä, määrittelyssä ja hallinnassa. Käsittelen Rajavartiolaitosta esimerkkinä tietojärjestelmiä ja ohjelmistoja hankkivasta organisaatiosta. Esimerkkiorganisaation lisäksi tutkimuskokonaisuus ja johtopäätökset ovat luultavimmin sovellettavissa myös joidenkin muiden organisaatioiden tarpeisiin. Pyrin organisaatiosidonnaisen näkökulman lisäksi käsittelemään tutkimusongelmaa myös laajemmin. En rajaa tutkimusta mihinkään yksittäiseen ohjelmistokehityshankkeeseen, vaan tarkastelen esimerkkiorganisaatiossa sovellettavia vaatimusmäärittelyn tekniikoita ja vertaan niitä alan muihin käytäntöihin, tutkimuksiin ja teorioihin. Tutkimusta voidaan vaatimusten määrittelyn osalta käyttää apuna laadittaessa ohjeistusta Rajavartiolaitoksen tietojärjestelmähankkeita varten. Uutta tutkimuksessa on, että vaatimusten määrittelyä ja hallintaa arvioivaa tutkimusta ei aikaisemmin ole tehty Rajavartiolaitoksen osalta.

Menetelmä- ja teorialähtöisesti aihetta on käsitelty mm. VTT:n tutkimuksessa ”Requirements Engineering, Inventory of technologies” (Parviainen ym., 2003). Tutkimus on laaja katsaus tekniikoihin, menetelmiin ja työkaluihin, joita vaatimuskehitysprosessi pitää sisällään. Näkökulma kattaa vaatimusten keräämisen, arvioinnin, vahvistamisen, dokumentoinnin, jäljitettävyyden ja hallinnan. Vaatimusten hallinta on kuvattu läpi projektin jatkuvana prosessina. Käsittelen vaatimuskehityksen teorioita, menetelmiä ja tekniikoita myös omassa työssäni, mutta näkökulma on organisaatiolähtöinen.

Toisessa VTT:n tutkimuksessa ”Riskitietoisen ohjelmiston vaatimusmäärittelyprosessin kehittäminen” (Pöyhönen ja Hukki, 2004) on käsitelty vaatimusmäärittelyn prosessien kehittämistä. Tutkimuksessa tarkastellaan luotettavuus- ja turvallisuusvaatimuksia sisältävien ohjelmistojen vaatimusmäärittelyn kehittämiseen liittyviä näkökohtia sekä teknisestä että asiantuntijoiden vuorovaikutuksen näkökulmasta. Keskeiset tutkimusongelmat

liittyvät riskitietoisen vaatimusmäärittelyprosessin kehittämiseen ja prosessiin osallistuvien asiantuntijoiden väliseen vuorovaikutukseen. Vaatimusmäärittelyn tason todetaan vaikuttavan merkittävästi projektin lopputulokseen. Riskien kartoitus on tärkeää ja siltä osin vaatimusmäärittelyssä on usein puutteita. Myös projektin sisäinen tiedonkulku ja asiantuntijoiden välinen vuorovaikutus vaikuttavat merkittävästi vaatimusmäärittelyyn. Aion huomioida nämä havainnot myös omassa työssäni, kun analysoin vaatimusmäärittelyn kustannustehokkuuteen vaikuttavia tekijöitä.

Seija Komi-Sirviö on käsitellyt väitöskirjassaan ”Development and Evaluation of Software Process Improvement Methods” ohjelmistokehityksen prosessien kehittämistä. Komi-Sirviön mukaan vaatimusmäärittelyn osalta on muodostunut uudeksi haasteeksi vaatimusten lyhyt elinkaari, joka on seurausta ohjelmistojen lisääntyneestä räätälöinnistä entistä eriytyneemmille ja alati muuttuville markkinoille. Keskeinen tutkimuskysymys on, miten tämä voidaan saavuttaa uhraamatta ohjelmiston laatua. Komi-Sirviön pääasiallinen tutkimusaihe on ohjelmistokehitysprosessien parantaminen, josta vaatimusmäärittely muodostaa yhden tärkeän osa-alueen. Ohjelmistovaatimusten hallinnan ja jäljitettävyyden merkitys korostuu erityisesti silloin, kun sovelluksen asiakaskohtainen räätälöinti lisääntyy. Lopputuloksen arvioinnin kannalta on tärkeää, että sovelluksen toiminnot ovat jäljitettävissä niiden toteutuksen taustalla olleisiin asiakaskohtaisiin vaatimuksiin. Vaatimusmäärittelyn ja vaatimusten hallinnan kehittäminen ovat tärkeä osa ohjelmistokehitysprosessin parantamista (SPI, Software Process Improvement). Hyödynnän näitä tuloksia myös omassa työssäni, kun arvioin vaatimusmäärittelyn asiakaslähtöisyyttä parantavia käytäntöjä.

Toronton yliopiston tutkijat Chung, Nixon ja Yu käsittelevät julkaisussaan “Using Quality Requirements to Systematically Develop Quality Software” vaatimusmäärittelyn laadullisia vaatimuksia ja niiden arviointia. Toiminnalliset vaatimukset ovat perinteisesti ohjanneet ohjelmistokehitystä, vaikka laadulliset tekijät, kuten esimerkiksi tarkkuus, turvallisuus, käyttäjäystävällisyys ja suorituskyky ovat ratkaisevan tärkeitä sovellusprojektin onnistumisen kannalta. Laadullisten vaatimusten saavuttamiseksi ei aina löydy johdonmukaisia menetelmiä. Tutkijat ovat luoneet laatuvaatimusten käsittelyä varten NFR-kehyksen (NFR, Non-Functional Requirements), johon ei-toiminnalliset laatuvaatimukset voidaan sijoittaa ohjelmistoprojektin tavoitteiksi. Saman ajatuksen pohjalta on myös syntynyt työkalusovellus nimeltään NFR-Assistant, jonka tavoitteena on helpottaa laadullisten vaatimusten luokittelua, hallintaa ja seurantaa. Laadullisten vaatimusten toteuttamisen kannalta on tärkeää määritellä, mitä toiminnallisia tavoitteita laatutavoit-

teet pitävät sisällään ja mihin toiminnallisuuksiin täytyy kiinnittää huomiota, jotta laadulliset vaatimukset voidaan täyttää. Oman työni painopiste ei ole vaatimuksissa itsessään, vaan käsittelen vaatimusprosessia kokonaisuudessaan. Pyrin kuitenkin huomioimaan toiminnallisten vaatimusten kytkökset laadullisiin vaatimuksiin.

2.2 Tutkimustehtävä

Tavoitteena on analysoida vaatimusten keräämiseen ja määrittelyyn kehitettyjä menetelmiä ja niiden soveltuvuutta Rajavartiolaitoksen tietojärjestelmäkehitykseen sekä selvittää, miten menetelmiä voitaisiin hyödyntää osana kustannustehokasta vaatimusmäärittelyä. Käsittelen vaatimusprosessia kokonaisuutena, johon sisältyvät ongelman määrittely, vaatimusten kerääminen ja analysointi sekä vaatimusten määrittely ja hallinta. Tutkielman tuloksia hyödynnetään päivitettäessä Rajavartiolaitoksen tietojärjestelmähankkeiden projektiohjeistoa.

2.3 Käsitteet, näkökulma ja rajaus

Tämän tutkielman keskeiset käsitteet ovat vaatimuskehitys, vaatimusten kerääminen ja analysointi, vaatimusmäärittely, vaatimusten hallinta ja tietojärjestelmähanke. Nimitän tätä kokonaisuutta vaatimusprosessiksi. Vaatimuskehitys pitää sisällään vaatimusten keräämisen, analysoinnin ja vaatimusmäärittelyn (McConnell, 1998). Vaatimuskehitysvaiheen tavoitteena on määritellä käyttäjien tarpeista ja järjestelmän ominaisuuksista saadun tiedon perusteella tarkat ohjelmistovaatimukset, joiden mukaan järjestelmän tulisi toimia. Kuvaan seuraavaksi lyhyesti, mitä nämä käsitteet pitävät sisällään.

Vaatimusten kerääminen voidaan toteuttaa esimerkiksi käyttäjiä haastattelemalla ja tarkkailemalla heidän nykyisiä työprosessejaan, tarkastelemalla kilpailevia tuotteita ja mahdollisesti myös rakentamalla vuorovaikutteisia prototyyppejä. Vaatimusten keräämiseen on myös muita tekniikkoja, joita käsittelen tarkemmin luvussa 4. Nimitän näitä tekniikkoja vaatimusmenetelmiksi. Vaatimusten keräämisen haasteena on saada käyttäjät ymmärtämään, mitä he oikein tahtovat. Vaatimusten analysointivaiheessa etsitään yhteneväisyyksiä ja eroavaisuuksia kerättyjen vaatimusten väliltä ja yhdistetään vaatimukset niin, että kaikki olennaiset piirteet jäävät jäljelle.

Vaatimusmäärittelyssä määritellään tietojärjestelmän tarkoitus tunnistamalla eri sidosryhmien tarpeet, jotka dokumentoidaan ymmärrettävään ja yksiselitteiseen muotoon. Vaatimusten dokumentoinnin kannalta on huomioitava, että vaatimukset jakautuvat toiminnallisiin, teknisiin ja laadullisiin vaatimuksiin. Tietojärjestelmän toiminnalliset vaatimukset koostuvat toimintaprosesseista ja käyttötilanteista ja toiminnallisen määrit-

telyn tuloksena voidaan luoda suunnitelma, jossa ohjelmisto kuvataan käyttöliittymää ja tietorakenteita myöten. Teknisillä vaatimuksilla tarkoitetaan ohjelmiston arkkitehtuuria, johon sisältyy mm. käyttöjärjestelmäympäristö, tietokantajärjestelmä ja valittu ohjelmointikieli. Laadullisiin vaatimuksiin taas luetaan mm. järjestelmän käytettävyys, luotettavuus, tehokkuus, siirrettävyys ja ylläpidettävyys.

Vaatimusten hallinnan tavoitteena on mahdollistaa järjestelmän suunnitelmallinen kehittäminen sen toimintaympäristössä tapahtuvien muutosten tai sidosryhmien tarpeiden muuttumisen vuoksi.

Tietojärjestelmähanketta käsitellään tässä projekteista koostuvana kokonaisuutena, joka pitää sisällään järjestelmäkehityksen esitutkimus- ja vaatimusmäärittelyvaiheet, prosessi- ja tietomallikuvaukset, suunnittelun, toteutuksen, testauksen ja käyttöönoton.

Tutkimukseni painopiste on vaatimuskehityksessä ja vaatimusten hallinnan menetelmissä itse vaatimusten ja niiden luokittelun jäädessä vähemmälle huomiolle. Käsittelen vaatimuksia niiden taustalla olevien tarpeiden pohjalta, jolloin vaatimusten toiminnallinen näkökulma korostuu. Pidän tarpeellisena korostaa toiminnallisia vaatimuksia, sillä ne heijastavat vahvasti tietojärjestelmän loppukäyttäjien ja muiden sidosryhmien tarpeita ja niiden määrittely perustuu järjestelmän toimintaprosessien ja käyttötilanteiden kuvaukseen. Toiminnalliset vaatimukset luovat pohjan tietojärjestelmän toteutukselle ja niiden selvittäminen on ratkaisevan tärkeä työvaihe määriteltäessä järjestelmävaatimuksia (Leffingwell ja Widrig, 2006). Onnistumisen edellytyksenä on, että vaatimusten määrittelijät tuntevat riittävän hyvin organisaation toimintaprosessit.

3 Tarpeista vaatimuksiin

3.1 Asiakkaan ongelmien ymmärtäminen

Menestyneimmät vaatimusselvitykset lähtevät liikkeelle asiakkaan ongelmien ja tarpeiden ymmärtämisestä (Leffingwell ja Widrig, 2006, 19). Leffingwellin ja Widrigin mukaan onnistuneen vaatimusselvityksen tärkein edellytys on, että järjestelmää rakentavalla projektiryhmällä on kyky ymmärtää tulevia loppukäyttäjiä ja heidän tarpeitaan. Projektiryhmän on ymmärrettävä ihmisiä, jotka tarvitsevat uutta järjestelmää ja ovat sen lopulliset käyttäjät. Tämän lisäksi projektiryhmän on tunnettava asiakkaan liiketoiminnan tarpeet ja kyseisen sovellusalueen tarpeet. Tämä tietämys on välttämätöntä, jotta ohjelmiston vaatimukset voidaan selvittää oikein.

Asiakkaan tavoitteiden ymmärtäminen edellyttää projektiryhmän jäseniltä lusalueen teknistä ja liiketoiminnallista osaamista, kykyä kommunikoida erilaisia mistaustoja omaavien sidosryhmien kanssa, ryhmätyöskentelytaitoja ja ennen kaikkea kärsivällisyyttä ja aitoa halua työskennellä asiakkaan ongelmien ratkaisemiseksi (Wiegers, 2003, 68-70). Myös Gottesdiener painottaa kommunikointitaitojen merkitystä asiakkaan ongelmien kartoituksessa (Gottesdiener, 2005, 44). Gottesdienerin mukaan järjestelmän loppukäyttäjien tarpeet muodostavat sillan asiakkaan liiketoiminnallisten tarpeiden ja järjestelmän kehitystyön perustana olevien teknisten vaatimusten välille. Projektiryhmän ja loppukäyttäjien välisen kommunikoinnin toimivuus on erityisen tärkeää, jotta asiakkaan ongelmat ja tarpeet ymmärrettäisiin oikein jo projektin alkuvaiheessa. On kuitenkin mahdollista, että projektiryhmä ei heti löydä yhteistä kieltä asiakkaan loppukäyttäjien kanssa. Loppukäyttäjät saattavat puhua asioista eri nimillä ja eri näkökulmista kuin projektiryhmän sovelluskehittäjät. Yhtenä mahdollisena syynä tähän on se, että loppukäyttäjien tekninen tausta ja ymmärrys poikkeavat projektiryhmän vastaavista. (Leffingwell ja Widrig, 2006, 19.) Leffingwell ja Widrig muistuttavat, että tilanne ei välttämättä ole helpompi silloinkaan, jos loppukäyttäjät ovat projektiryhmän tavoin taustaltaan teknisiä henkilöitä, vaan asiakkaan ja projektiryhmän näkemykset ratkaistavasta ongelmasta voivat silti poiketa merkittävästi toisistaan. Vaatimusselvityksen haasteena onkin asiakkaan ja projektiryhmän erilaisista lähtökohdista ja näkökulmista huolimatta löytää loppukäyttäjien todelliset tarpeet ja kuvata ne selkeän yksiselitteisesti siten, että niistä vallitsee yhteisymmärrys asiakkaan ja projektiryhmän välillä. Vasta tämän jälkeen tarpeista voidaan jalostaa järjestelmävaatimuksia.

Vaatimusten selvitysvaiheen tavoitteena on saavuttaa ymmärrys ongelmasta ja niistä tarpeista, jotka täytyy täyttää, jotta ongelma voidaan ratkaista (Leffingwell ja Widrig, 2006, 19-20). Projektiryhmän on kyettävä vuoropuheluun asiakkaan loppukäyttäjien kanssa. Loppukäyttäjillä voi olla liiketoimintaan tai tekniikkaan liittyviä ongelmia, joihin he tarvitsevat projektiryhmän apua. Jotta projektiryhmä voisi ymmärtää loppukäyttäjien tarpeet oikein ja suunnitella eri sidosryhmät huomioivan kokonaisratkaisun, sen on tunnettava myös asiakkaan liiketoiminnan ja kyseisen sovellusalueen tarpeet (Pöyhönen ja Hukki, 2004, 11).

3.1.1 Ongelman analysointi

Ongelman analysoinnissa on kyse todellisten ongelmien ja käyttäjien tarpeiden ymmärtämisestä ja sellaisten ratkaisujen ehdottamisesta, jotka vastaavat noita tarpeita. Ongelman analysoinnin tavoitteena on saavuttaa parempi ymmärrys ratkaistavana olevasta

ongelmasta ennen kuin vaatimusmäärittely ja kehitystyö aloitetaan. (Leffingwell ja Widrig, 2006, 44.) Tämän lisäksi tavoitteena on, että järjestelmän sidosryhmät ja projektiryhmä voivat päästä ongelmasta yhteisymmärrykseen. Itse ongelman analysoinnin ja kuvaamisen lisäksi on tarpeen myös selventää, millainen on tavoitteena oleva ratkaisu. (Gottesdiener, 2005, 27 ja 31.) Jotta ongelman syyt voidaan tunnistaa ja taustat selvittää, on haastateltava sidosryhmiä ja esitettävä oikeat kysymykset oikeille henkilöille. Järjestelmän toimijoiden tunnistaminen on ongelman analysoinnin tärkein askel (Leffingwell ja Widrig, 2006, 50).

Ongelmien analysointi auttaa ymmärtämään, millaisia ratkaisuja rakennettavan järjestelmän tai sovelluksen on pidettävä sisällään, jotta se täyttää asiakkaan loppukäyttäjien tarpeet. Tässä tutkimuksessa ongelmien analysointia käsitellään prosessina, jonka tavoitteena on ongelmien ymmärtäminen ja ratkaisumahdollisuuksien kartoitus ennen vaatimusmäärittelyä. Leffingwell ja Widrig (2006) muistuttavat, että aina ongelman olemassaolo ei ole selvää. Tällöin syy järjestelmän tai sovelluksen rakentamiseen voi olla esimerkiksi se, että pyritään hyödyntämään markkinoilla olevia mahdollisuuksia. Tästä ovat esimerkkinä tietokonepelit. Vaikka sovelluksen tavoitteena ei olisikaan minkään nimetyn ongelman ratkaisu, voi se kuitenkin välillisesti toimia ratkaisuna johonkin ympäristössä olevaan ja taustalla vaikuttavaan ongelmaan ja tarjota todellista arvoa käyttäjälle. Esimerkiksi tietokonepelin tapauksessa ongelmana voi olla loppukäyttäjän liika vapaa-aika tai se, että käyttäjällä ei ole riittävästi hauskoja asioita, joita hän voisi tehdä tietokoneellaan (Leffingwell ja Widrig, 2006, 43).

Leffingwell ja Widrig muistuttavat, että vaatimuksia selvitettäessä ongelmat ja mahdollisuudet ovat vain saman kolikon eri puolia ja näkökulmasta riippuen jonkun ongelma on toisen mahdollisuus. Useimpien järjestelmien tarkoituksena on kuitenkin ratkaista jokin tietty ongelma, mistä johtuen projektiryhmän pääasiallinen tehtävä on toimia ongelmanratkaisijana (Leffingwell ja Widrig, 2006, 43). Jotta ratkaisuja voidaan ehdottaa, on saavutettava riittävä ymmärrys todellisista ongelmista ja käyttäjien tarpeista. Tehokkaan ongelmanratkaisun edellytyksenä on, että projektiryhmä pääsee asiakkaan kanssa yhteisymmärrykseen ongelman määrittelystä ja ymmärtää sen perimmäiset syyt, tunnistaa sidosryhmät ja käyttäjät, määrittelee järjestelmän rajat ja tunnistaa ratkaisun muuttujat. Myös eri ratkaisuvaihtoehdot on kartoitettava. Mahdollisia ratkaisuvaihtoehtoja voi olla useita ja projektiryhmän on valittava niistä sopivin kyseessä olevaan ongelmaan. Osana tätä prosessia on hyödyllistä ymmärtää ehdotetun ratkaisun edut ja kuvata ne asiakkaan ja käyttäjien termeillä (Leffingwell ja Widrig, 2006, 45). Näin projektiryhmä

voi saavuttaa paremman ymmärryksen siitä, miten sidosryhmät näkevät man. Käytännössä kyse on järjestelmän loppukäyttäjien ja Rajavartiolaitoksen tarpeiden ja ongelmien ymmärtämisestä ja mahdollisten ratkaisuvaihtoehtojen läpikäynnistä vaatimusmäärittelyn tueksi.

Leffingwell ja Widrig (2006) viittaavat Gausen ja Weinbergin (1989) esittämään määritelmään, jonka mukaan ongelma voidaan nähdä havaitun ja halutun asian välisenä erona. Määritelmä selkeyttää ongelman käsitettä. Sekä sovelluskehittäjä että loppukäyttäjä voivat ymmärtää ongelman, vaikka he kumpikin katsovat tilannetta omista näkökulmistaan. Havaittu ongelma on kuitenkin dokumentoitava selkeästi ja varmistettava, että kaikki sidosryhmät ja projektiryhmä ovat määritelmästä yksimielisiä. Jotta ongelmat voidaan dokumentoida selkeästi läpi projektin, niiden kuvaustapa on hyödyllistä pitää muodollisena ja yhdenmukaisena (Leffingwell ja Widrig, 2006, 45). Näin voidaan tukea sitä, että projektin sidosryhmät löytävät yhteisen kielen ongelmien kuvaamiseen ja pääsevät niistä yhteisymmärrykseen.

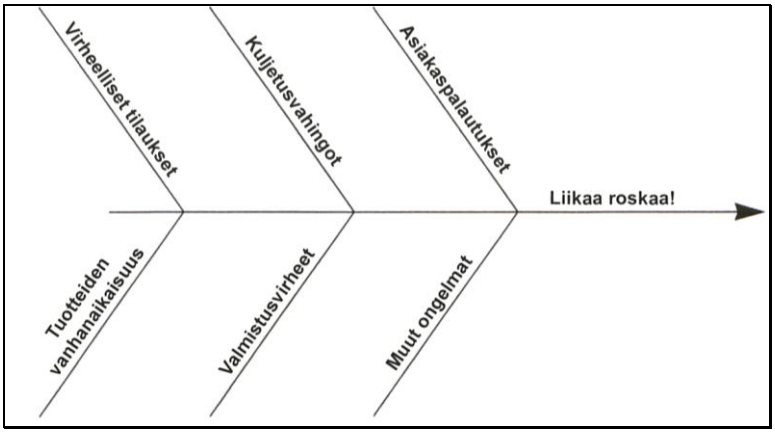
Ongelmien syvällinen ymmärtäminen ja analysointi selventävät vaatimusten keräämisen kannalta keskeisiä kysymyksiä ja tavoitteita. Tämä edellyttää projektiryhmältä kykyä ymmärtää sekä organisaation, kyseessä olevan sovellusalueen, että eri sidosryhmien tarpeita. Tämä on syytä huomioida myös Rajavartiolaitoksen tietojärjestelmäprojekteissa siten, että projektiryhmä koostuu sekä teknistä että operatiivista osaamista omaavista henkilöistä.

3.1.2 Ongelman taustasyiden ymmärtäminen

Yksi keino etsiä ongelman perussyitä on kysyä suoraan sidosryhmiltä heidän mielipidettään. Kysymykset on kuitenkin osattava esittää oikeille henkilöille ja aina edes organisaation johto ei osaa nimetä heitä suoralta kädeltä. Tällöin projektiryhmän tehtävänä on varmistaa, että loppukäyttäjiä kuunnellaan ja heillä mahdollisesti oleva tieto huomioidaan ongelman selvityksessä. (Leffingwell ja Widrig, 2006, 48.) Tässä vaiheessa on tarpeen kuulla sekä asiantuntijoita, että vähemmän kokemusta omaavia loppukäyttäjiä mahdollisimman kattavan ja objektiivisen käsityksen hankkimiseksi (Wiegers, 2003, 102).

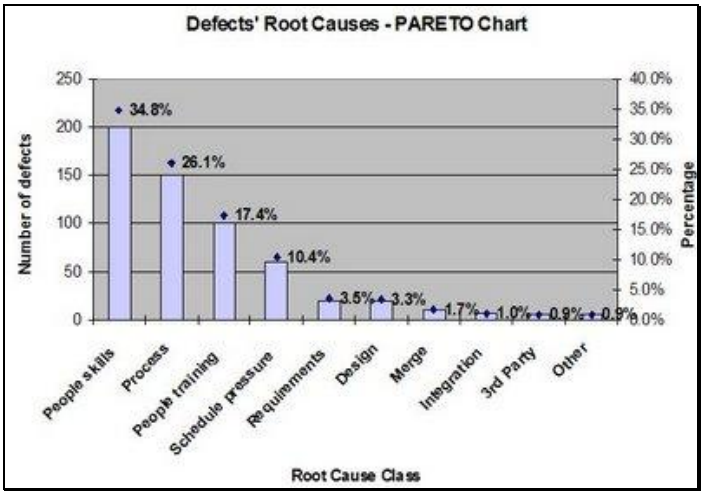
Ongelman juurien löytäminen edellyttää ongelman taustalla vaikuttavien tekijöiden järjestelmällistä kartoittamista. Näin on esimerkiksi silloin, kun etsitään syitä yrityksen valmistamien tuotteiden heikentyneeseen menekkiin tai ohjelmistoprojektin aikataulun pettämiseen. Ongelman koostumusta ja taustatekijöitä voidaan esittää kuvan 3-1 mukai-

sella kalanruoto-diagrammilla (fishbone diagram), jossa ongelman osatekijät ovat piikkeinä ruodossa.



Kuva 3-1: Kalanruoto-diagrammi
Lähde: Leffingwell ja Widrig, 2006

Joskus voi olla välttämätöntä viedä läpi yksityiskohtainen tutkimus kustakin ongelman osa-alueesta ja mitata niiden kunkin yksittäinen vaikutus. Perussyyn analysointi (root cause analysis) on ongelman taustalla vaikuttavien tekijöiden kartoittamiseen kehitetty tekniikka. Toteutus voi vaihdella yksinkertaisesta projektiryhmän ja sidosryhmien välisestä aivoriihestä pieneen tiedonkeruuprojektiin tai laajempaan tutkimukseen. (Leffingwell ja Widrig, 2006, 47.) Tällaisen tutkimuksen päämääränä on mitata ongelman kunkin perussyyn osuus. Silti kaikkia ongelman perussyitä ei välttämättä ole järkevää ratkaista. Syynä on se, että ratkaisun kustannukset saattavat ylittää ongelman rahallisen arvon. Jotta tiedettäisiin, mitkä perussyistä kannattaa korjata, on ensin määriteltävä kunkin perussyyn osuus ja rahallinen arvo. Tutkimuksen tuloksia voidaan esittää esimerkiksi kuvan 3-2 mukaisella Pareto-kaaviolla tai yksinkertaisella histogrammilla, joka paljastaa visuaalisesti todelliset syyt (Leffingwell ja Widrig, 2006, 48).



Kuva 3-2: Virheiden perussyiden analysointi
Lähde: http://www.softwareprojects.org/root-cause-analysis.htm (30.4.2009)

Yllä olevassa Pareto-kaaviossa on kuvattu virheiden aiheuttajat ja niiden suhteellinen osuus virheiden kokonaismäärästä. Samaa periaatetta voidaan hyödyntää ongelman taustalla vaikuttavien tekijöiden analysointiin. Kun tiedetään ongelmatekijät ja niiden kunkin prosentuaalinen osuus, voidaan päätellä mihin tekijöihin projektiryhmän on erityisesti syytä keskittyä, jotta ongelma voidaan ratkaista. Pareto-kaavio perustuu 80-20-sääntöön, joka syntyi italialaisen taloustieteilijä Vilfredo Pareton vuonna 1906 tekemästä havainnosta, jonka mukaan Italiassa 20 prosenttia väestöstä omisti 80 prosenttia kaikesta varallisuudesta (Reh, 2002). Pareton periaatetta voidaan soveltaa tietojärjestelmä- tai ohjelmistoprojektin ongelmien analysointiin esimerkiksi olettamalla, että pieni määrä ohjelmistomoduuleita todennäköisesti aiheuttaa suurimman osan ohjelmistovirheistä (Fenton ja Ohlsson, 2000, 798).

3.2 Sidosryhmien tunnistaminen

Minkä tahansa monimutkaisen ongelman tehokas ratkaiseminen pitää tyypillisesti sisällään eri sidosryhmien tarpeiden tyydyttämisen (Leffingwell ja Widrig, 2006, 50). Jotta projektiryhmä voi kehittää ongelmaan tehokkaan ratkaisun, sen on tiedettävä keitä sidosryhmät ovat ja mitkä ovat heidän tarpeensa. Sidosryhmiä kartoitettaessa on tarpeen tunnistaa ja listata sekä järjestelmän kanssa tekemisissä olevat toimijat että heidän roolinsa ja vastuunsa järjestelmän kannalta. (Gottesdiener, 2005, 144.) Projektiryhmän on ymmärrettävä käyttäjien ja muiden sidosryhmien vaihtelevia näkökulmia ja erilaisia tarpeita. Leffingwell ja Widrig (2006) määrittelevät sidosryhmän siten, että kyseessä ovat ”ketkä tahansa henkilöt, joiden työhön uusi järjestelmä tai sovellus tulee vaikuttamaan”. Tämä määritelmä kattaa sekä järjestelmän tulevat käyttäjät, että epäsuorasti sovelluksen vaikutuspiirissä olevat henkilöt. Epäsuorasti sovelluksen vaikutuspiirissä olevat henkilöt eivät itse käytä sovellusta, mutta järjestelmän avulla saavutettava tulos vaikuttaa myös heidän työhönsä. Kumpikin mainituista sidosryhmistä voi pyrkiä vaikuttamaan järjestelmän vaatimuksiin ja projektin lopputulokseen.

Käyttäjät voidaan sidosryhmänä jaotella omiin alaluokkiinsa. Tässä yhteydessä on kuitenkin syytä huomioida, että käyttäjäryhmiä ei välttämättä kannata muodostaa suoraan organisaation hierarkian pohjalta vaan pikemminkin sidosryhmien ja järjestelmän välisen vuorovaikutuksen perusteella (Wiegers, 2003, 98-99). Wiegersin mukaan kullakin käyttäjäluokalla on omat erityistarpeensa ja vaatimuksensa, jotka riippuvat pikemminkin suoritettavista tehtävistä kuin esimerkiksi siitä, missä organisaation yksiköissä henkilöt työskentelevät tai mikä on heidän maantieteellinen sijaintinsa. Näin ollen tiettyyn käyttäjäluokkaan kuuluvia henkilöitä yhdistää esimerkiksi se, mitä tehtäviä he suoritta-

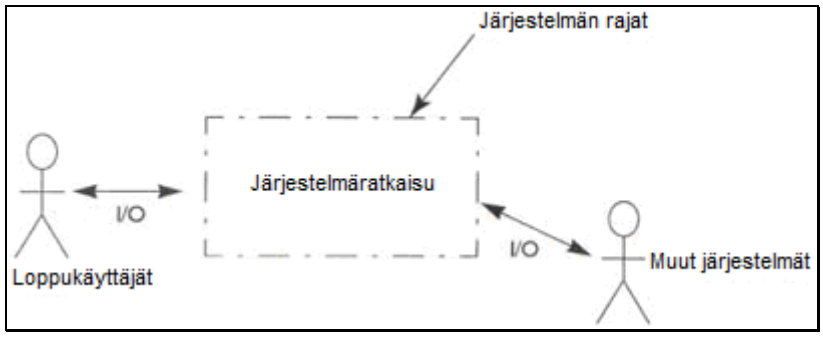
vat, mitä toimintoja he tarvitsevat ja mitä käyttöoikeuksia heillä tulee olla tehtäviensä suorittamiseksi.

Sidosryhmiä kartoitettaessa haastatellaan päätöksentekijöitä, loppukäyttäjiä ja muita sovelluksen vaikutuspiirissä olevia henkilöitä. Aluksi on tärkeää selvittää, ketkä ovat järjestelmän käyttäjät, kuka on asiakas ja järjestelmän ostaja ja keihin kaikkiin järjestelmän tuottamat tulokset vaikuttavat. Lisäksi on tiedettävä, kuka arvioi ja hyväksyy järjestelmän kun se on toimitettu ja kuka ylläpitää uutta järjestelmää (Leffingwell ja Widrig, 2006, 51).

3.3 Järjestelmän rajaaminen

Kun ongelmasta on päästy yhteisymmärrykseen ja käyttäjät sekä sidosryhmät on tunnistettu, voidaan siirtyä hahmottelemaan tulevan järjestelmäratkaisun rajoja. Järjestelmän rajaamisessa on kyse itse järjestelmän ja sen ympäristön välisen rajan määrittelystä. Tämä on tärkeää myös vaatimusten kokonaisuuden ymmärtämiseksi. Leffingwell ja Widrig (2006) näkevät järjestelmän toimijana syöttötietojen ja lopputuloksen välissä. Tietovirta syötön ja lopputuloksen muodossa (I/O, input/output) kulkee järjestelmän ja ulkomaailman välillä.

Ratkaisun rajoja etsittäessä on itse järjestelmän lisäksi kiinnitettävä huomiota myös järjestelmän kanssa vuorovaikutuksessa oleviin tahoihin, jotka voidaan määritellä toimijoiksi. Toimija voi olla joko yksittäinen käyttäjä, ryhmä henkilöitä tai toinen järjestelmä (Leffingwell ja Widrig, 2006, 52; Gottesdiener, 2005, 145). Toimija on siis joku tai jokin järjestelmän ulkopuolella, joka keskustelee järjestelmän kanssa. Kuva 3-3 esittää järjestelmän ja sen toimijoiden välistä vuorovaikutusta, jossa järjestelmä on rajattu kokonaisuus ja toimijat sen ulkopuolisia tahoja.



Kuva 3-3: Järjestelmän rajat ja toimijat
Lähde: Leffingwell ja Widrig, 2006

Toimijoita ovat sekä käyttäjät että muut järjestelmät. Leffingwell ja Widrig muistuttavat, että vaikka toimijoiden tunnistaminen vaikuttaa melko ilmeiseltä, se on kuitenkin

ongelman analysoinnin avainaskel. Toimijoiden löytämiseksi on selvitettävä, kuka jakelee, käyttää tai poistaa tietoa järjestelmästä, kuka operoi järjestelmää, kuka ylläpitää järjestelmää ja missä järjestelmää käytetään. Näiden lisäksi on selvitettävä, mistä järjestelmä saa tietonsa ja mitkä muut ulkoiset järjestelmät ovat tekemisissä järjestelmän kanssa (Leffingwell ja Widrig, 2006, 54). Kun tämä kokonaisuus tunnetaan, voidaan laatia järjestelmästä laaja yleiskuvaus, joka kattaa järjestelmän rajat, käyttäjät ja liittymät muihin sovelluksiin.

3.4 Ratkaisuun vaikuttavat rajoitukset

Mahdollisten ratkaisujen rajoitukset on syytä arvioida jo projektin alkuvaiheessa. Jos jo valitun ratkaisun rajoitukset huomataan vasta myöhemmässä vaiheessa projektia, tämä voi vakavasti heikentää mahdollisuuksia toimittaa järjestelmä asiakkaalle siten, kuin alun perin on suunniteltu. Seurauksena saattaa olla, että valittua teknologista lähestymistapaa joudutaan muuttamaan kesken projektin, mikä voi olla hyvin riskialtista ja kallista. Tämän vuoksi rajoitukset on huomioitava osana järjestelmän suunnitteluprosessia.

Potentiaalisia järjestelmärajoitusten lähteitä ovat esimerkiksi taloudelliset tekijät, yhteiskunnan ja organisaation toimintaan vaikuttavat lait, säännöt ja politiikka, teknologiset rajoitukset, ennestään käytössä olevat järjestelmät, turvallisuusvaatimukset, laatustandardit, projektin aikataulu ja resurssit (Leffingwell ja Widrig, 2006, 56). Gottesdiener (2005) huomauttaa, että rajoitukset vaikuttavat sekä järjestelmän suunnitteluun että tekniseen toteutukseen. Esimerkiksi, jos uuden järjestelmän on oltava yhteensopiva jo olemassa olevan sovellusympäristön kanssa tai jos vaatimuksena on käyttää jotain tiettyä ohjelmointikieltä, on tämä huomioitava jo suunnitteluvaiheessa teknistä toteutusta ja teknologiavalintoja koskevana rajoitteena. (Gottesdiener, 2005, 8.)

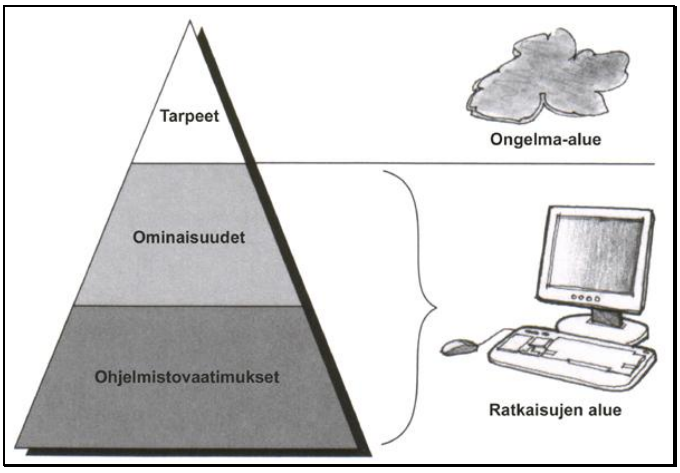
Vaikka osa järjestelmärajoituksista saattaa olla ilmeisiä, kaikki niistä eivät välttämättä ole projektiryhmän tiedossa, jolloin niitä täytyy aktiivisesti etsiä. Apuna rajoitusten etsimisessä on, jos tiedetään millaisia asioita tulisi etsiä. Kun rajoitukset on tunnistettu, jotkin niistä muuntuvat uuden järjestelmän vaatimuksiksi, kun toiset taas vaikuttavat resursseihin, toteutussuunnitelmiin ja projektisuunnitelmiin. Ongelman ratkaisijan vastuulla on ymmärtää rajoitusten potentiaaliset lähteet kunkin erillisen sovellusympäristön osalta ja päätellä, mikä on rajoitusten vaikutus eri ratkaisuvaihtoehtoihin.

3.5 Järjestelmän ominaisuudet

Kun projektiryhmä on saavuttanut ymmärryksen ratkaistavasta ongelmasta ja sen perimmäisistä syistä, tunnistanut sidosryhmät sekä ratkaisun rajat ja rajoitukset, se voi siirtyä yksityiskohtaisempaan järjestelmän ominaisuuksien määrittelyyn. Tässä vaiheessa järjestelmän ominaisuuksia on usein kuvattu vasta pelkistetysti käyttäjän tarpeiden pohjalta esimerkiksi seuraavasti: ”Ohjelma sallii tilausten www-pohjaisen syöttämisen”. (Leffingwell ja Widrig, 2006, 21.) Wiegers (2003) huomauttaa, että ominaisuudet, joita käyttäjät esittävät tarpeinaan, eivät välttämättä vastaa niitä toiminnallisuuksia, joita he tarvitsevat työnsä suorittamiseen (Wiegers, 2003, 95). Projektiryhmän tehtävänä onkin selvittää, mitä ominaisuuksia järjestelmällä on oltava, jotta kuvattu toiminto olisi mahdollista toteuttaa. Leffingwell ja Widrig (2006) muistuttavat, että toimintojen pelkistetyt kuvaukset ovat hyödyllisiä, sillä niiden avulla projektiryhmä voi kommunikoida käyttäjien kanssa heidän kielellään (Leffingwell ja Widrig, 2006, 21). Tällöin kuvattu ominaisuus voidaan nähdä palveluna, jonka järjestelmä tarjoaa ja joka täyttää yhden tai useamman sidosryhmän tarpeista.

4 Vaatimukset ja vaatimuskehitys

Kun tiedetään, millaisia ominaisuuksia järjestelmän on pidettävä sisällään, voidaan siirtyä määrittelemään vaatimuksia. Ominaisuudet vastaavat yhtä tai useampaa sidosryhmien tarpeista, joten tarpeet siirtyvät ominaisuuksien kautta vaatimuksiin. Leffingwell ja Widrig (2006) kuvaavat tarpeita, ominaisuuksia ja vaatimuksia kuvan 4-1 kaltaisella pyramidilla, jossa tarpeet muodostavat pyramidin ylimmän ongelmien tason. Ominaisuudet ja vaatimukset taas ovat alemmalla ratkaisujen tasolla. Kun pyramidin huipulta siirrytään alaspäin, ongelman määritelmä tarkentuu ja samalla lähestytään ratkaisua. Vaatimukset edustavat tarkinta ongelmanratkaisun tasoa.



Kuva 4-1: Ongelmien ja ratkaisujen kokonaisuus: tarpeet, ominaisuudet ja vaatimukset
Lähde: Leffingwell ja Widrig, 2006

Edistyminen voidaan kuvata siirtymisenä ongelmatasolta, jossa näemme vain käyttäjien tarpeet, järjestelmän määrittelyyn, joka muodostaa ratkaisutason, jota ilmentävät järjestelmän ominaisuudet ja ohjelmistovaatimukset. Ongelman ratkaisu tarpeiden kartoituksesta ominaisuuksien ja vaatimusten määrittelyyn etenee portaittaisella tavalla siten, että ensin on ymmärrettävä ongelma ja käyttäjän tarpeet, minkä jälkeen voidaan edetä ratkaisutasolle. (Leffingwell ja Widrig, 2006, 22.)

4.1 Vaatimusten käsite

IEEE-standardin mukainen vaatimuksen kuvaus määrittelee vaatimuksen ohjelmiston toiminnallisuutena, jota käyttäjä tarvitsee ratkaistakseen ongelman tai saavuttaakseen jonkin päämäärän. Standardin mukaan vaatimus voidaan lisäksi nähdä toiminnallisuutena, joka ohjelmistolla tai sen komponentilla täytyy olla, jotta voidaan täyttää sopimus, määritys tai jokin muun muodollisesti esitetty dokumentti (IEEE Standard Glossary of Software Engineering Terminology, 1990, 35). Edellä kuvaamani Leffingwellin ja Widrigin (2006) näkemys tarpeiden, ominaisuuksien ja vaatimusten hierarkkisesta olemuksesta tukee tätä ajatusta. Vaatimusten taustalla on ominaisuuksia, jotka järjestelmällä täytyy olla, jotta asiakkaan tarpeet voidaan täyttää. Myös Wiegers näkee vaatimukset järjestelmän ominaisuuksina, jotka ovat järjestelmän käyttäjille ja muille sidosryhmille tarjoaman lisäarvon edellytys (Wiegers, 2003, 8).

4.2 Vaatimuskehitys

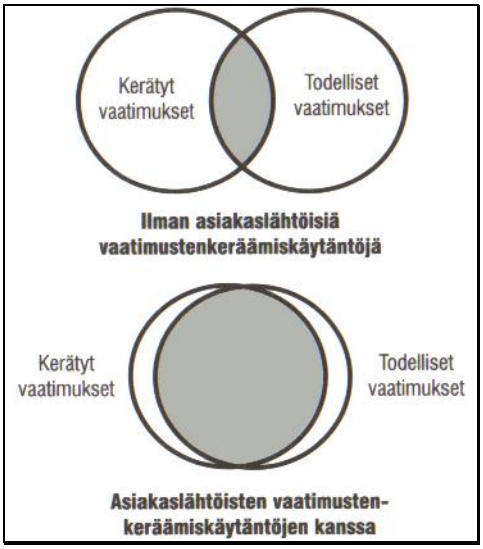
Vaatimuskehitysvaiheessa määritellään käyttäjien tarpeista ja järjestelmän ominaisuuksista saadun tiedon perusteella tarkat ohjelmistovaatimukset, joiden mukaan järjestelmän tulisi toimia. Tiivistetysti vaatimuskehitysvaihe sisältää kolme yhteenkuuluvaa aktiviteettia, jotka ovat vaatimusehdotusten kerääminen, vaatimusten analysointi ja vaatimusten määrittely (McConnel, 1998, 116).

Vaatimusehdotusten kerääminen pitää usein sisällään potentiaalisten käyttäjien haastattelun koskien heidän toivomaansa järjestelmää, kilpailevien tuotteiden tarkastelun ja mahdollisesti myös vuorovaikutteisten prototyyppien rakentamisen (McConnel, 1996, 116). Vaatimusten keräämisen haasteena on saada käyttäjät ymmärtämään, mitä he oikein tahtovat. Käyttäjien tarpeiden ylöskirjaaminen ei ole vaikeaa. Projektiryhmän haasteena on kuitenkin saada kerätyksi todelliset vaatimukset. McConnel muistuttaa, että asiakkaat ja sovelluskehittäjät ymmärtävät vaatimukset usein eri tavalla, mistä aiheutuu väärinymmärryksiä puolin ja toisin. Asiakkaat eivät aina kuvaile vaatimuksia täsmällisesti, vaan tulkitsevat niitä suurpiirteisesti. Kehittäjät taas tulkitsevat vaatimukset sana-

tarkasti asiakkaan kertoman perusteella. (McConnel, 1996, 239.) Asiakkaan ottaminen mukaan vaatimusten keräämiseen on ainoa tapa ehkäistä suuret eroavaisuudet loppukäyttäjien odotusten ja järjestelmän toteutuksen välillä (Wiegers, 2005, 95). Wiegersin mukaan ongelma ilmenee ristiriidassa olevina käsityksinä järjestelmästä, jonka loppukäyttäjät odottavat saavansa ja jota kehittäjät rakentavat. Asiakkaan määritelmä vaatimuksista saattaa sovelluskehittäjän mielestä kuulostaa korkean tason tuotekuvaukselta ja sovelluskehittäjän määritelmä vaatimuksista saattaa asiakkaan mielestä kuulostaa yksityiskohtaiselta käyttöliittymäkuvaukselta (Wiegers, 2005, 7).

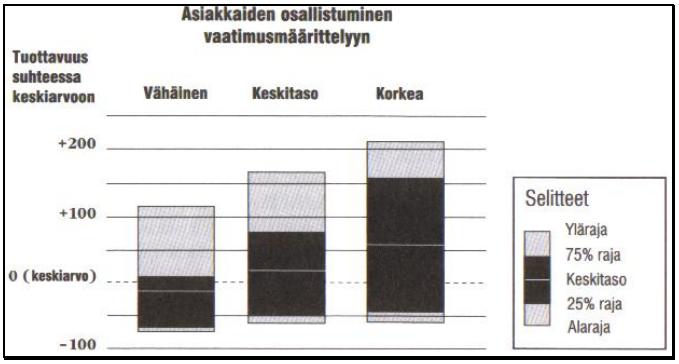
Ensimmäinen askel vuorovaikutteisessa vaatimusten keräämisessä on nimetä joukko asiakkaan avain- ja loppukäyttäjiä, jotka auttavat ohjelmiston vaatimusten määrittelemisessä (McConnel, 1996, 118). McConnellin mukaan asiakassuuntautuneet vaatimustenkeruukäytännöt auttavat selvittämään suuremman osan todellisista vaatimuksista ja parantavat sekä asiakkaan että projektiryhmän ymmärrystä vaatimuksista.

McConnell (1996) muistuttaa, että asiakkaan ottaminen mukaan vaatimuskehitykseen nopeuttaa projektia. Projektiryhmä voi keskittyä asiakkaan todellisiin vaatimuksiin eikä aikaa kulu epäoleellisten vaatimusten selvittelyyn. Käyttäjät on kuitenkin valittava oikein. McConnellin mukaan joukossa on hyvä olla sekä tehokäyttäjiä että tavallisia käyttäjiä. Käyttäjien on oltava riittävän hyvin perillä asiakkaan liiketoiminnan tarpeista, jotta projektiryhmä voi luottaa heidän sanaansa. ”Jos käyttäjät pitävät jotakin ominaisuutta tärkeänä, projektiryhmän on voitava luottaa siihen, että kyseessä on asiakkaalle oikeasti tärkeä ominaisuus. Samoin jos jonkin ominaisuuden voi käyttäjien mielestä jättää pois, on projektiryhmän voitava luottaa heidän asiantuntemukseensa”. (McConnel, 1996, 118.) Kuvassa 4-2 näkyy ero vaatimusten keräämisessä ilman asiakassuuntautuneita käytäntöjä ja niiden kanssa. Asiakaslähtöiset käytännöt kasvattavat todellisten vaatimusten osuutta kerätyistä vaatimuksista.



Kuva 4-2: Ero tavallisten ja asiakaslähtöisten vaatimustenkeräämiskäytäntöjen välillä.
Lähde: McConnel, 1996

Jos projektiryhmä kerää vaatimukset kokonaan ilman asiakkaan loppukäyttäjien apua tai siten, että heidän osallistumisensa on vähäistä, kerätyt vaatimukset edustavat vain osaa todellisista vaatimuksista. Loppukäyttäjien mukaan ottaminen parantaa tulosta merkittävästi. McConnell perustaa näkemyksensä tutkimukseen ”Client Participation in Requirements Specification” –tekijästä (Vosburgh et al. 1984). Tutkimuksen mukaan asiakkaiden aktiivinen osallistuminen voi parantaa kehittämisnopeutta huimasti, mutta ei yksinään riitä tuottavuuden parantamiseen. Kuva 4-3 esittää tutkimuksen tuloksia.



Kuva 4-3: “Client Participation in Requirements Specification”-tekijästä saadut tulokset (Vosburgh et al. 1984).
Lähde: McConnel, 1996

McConnell painottaa Vosburghin tutkimukseen viitaten, että vaikka asiakkaiden osallistuminen vaatimusten keräämiseen on tärkeää, heidän ei pidä antaa kirjoittaa vaatimusmäärittelyä itse. Tutkimuksen tulosten mukaan tuottavuus voi parantua korkean asiakkaiden osallistumisasteen myötä, mutta jos asiakkaat kirjoittavat määrittelyn itse, tuottavuus laskee. Vosburghin havaintojen mukaan yli puolet asiakkaiden kirjoittamista määrityksistä piti kirjoittaa uudelleen (McConnell, 1996, 239).

Myös Wiegers (2005) korostaa, että asiakaslähtöinen vaatimusten kerääminen on yksi projektin menestystekijöistä. Asiakkaan jatkuva osallistuminen on tärkeää, jotta vaatimuksiin liittyvä yhteisymmärrys säilyy ja riskit pysyvät hallussa. Wiegersin mukaan on kuitenkin mahdollista, että järjestelmä joudutaan määrittelemään uudelleen, jos se rakennetaan asiakkaan loppukäyttäjiltä projektin alussa kerättyjen vaatimusten pohjalta. Näin voi käydä varsinkin silloin, jos asiakkaan osallistuminen vaatimusprosessiin rajoittuu vain projektin alkuun eikä sen jälkeen ole jatkuvaa. Wiegers perustelee näkemystään sillä, että loppukäyttäjät eivät vielä projektin alussa välttämättä tiedä, mitä he todella tarvitsevat. Ominaisuudet joita käyttäjät esittävät tarpeinaan, eivät välttämättä vastaa toiminnallisuuksia, joita he tarvitsevat tehtäviensä suorittamiseen (Wiegers, 2005, 95).

Tässä kappaleessa esittämieni siteerausten perusteella oletan, että asiakaslähtöinen vaatimusten kerääminen on perusteltua myös Rajavartiolaitoksen tietojärjestelmäprojekteissa. Rajavartiolaitoksen tapauksessa asiakkaan loppukäyttäjät ovat tietojärjestelmän operatiivisia käyttäjiä ja projektiryhmä koostuu sekä operatiivista toimintaa että tietotekniikkaa tuntevasta henkilöstöstä ja ulkoisten sovellustoimittajien edustajista.

4.3 Vaatimusten kerääminen

Käsittelen seuraavaksi asiakaslähtöiseen vaatimusten keräämiseen soveltuvia menetelmiä, jotka perustuvat käyttäjien haastatteluun ja heidän toimintansa tarkkailuun, prototyypin tai käyttöskenaarioiden avulla tapahtuvaan mallinnukseen, järjestelmän käyttäjien ja sovelluskehittäjien yhteistyötä hyödyntäviin ryhmätekniikoihin sekä vaatimusten uudelleenkäyttöön.

4.3.1 Käyttäjien haastattelu

Perinteistä käyttäjien haastattelua voidaan soveltaa vaatimusten keräämiseen. Käytännössä haastattelut ovat projektiryhmän ja sidosryhmien välisiä keskusteluja, joiden tarkoituksena on saavuttaa ymmärrys sidosryhmien vaatimuksista. Haastattelu voi olla jäsennetty, jolloin se toteutetaan esimääritellyin kysymyksin, tai avoin, jolloin haastattelu on vapaamuotoista keskustelua siitä, mitä sidosryhmät haluavat järjestelmältä. Jäsennetyn haastattelun tavoitteena on saada sidosryhmiltä vastaukset esimääriteltyihin kysymyksiin, mutta avoimessa haastattelussa myös tavoiteltu lopputulos on avoin. (Parviainen ym. 2003, 50.) Wiegersin (2003) mukaan avoimella haastattelulla voidaan kartoittaa organisaation nykyisiä toimintaprosesseja, jotta voidaan nähdä, kuinka uusi järjestelmä voisi parantaa niitä (Wiegers, 2003, 115). Avoin ja jäsennetty haastattelu täyden-

tävät toisiaan. Avoimella haastattelulla voidaan hankkia alustava yleiskuva ja ymmärrys sidosryhmien ongelmista ja tarpeista kun taas jäsennetyssä haastattelussa kysymykset voidaan laatia koskemaan jotain tiettyä ongelma-aluetta tai ne voidaan esittää tarjottavan ratkaisun näkökulmasta (Leffingwell ja Widrig, 2006, 102).

Jäsennetyllä haastattelulla projektiryhmä voi kerätä tietoa organisaation toimintatavoista järjestelmän suunnittelun tueksi. Yksi tällainen menetelmä on Contextual Inquiry (CI), joka soveltuu käyttäjien haastatteluun kentällä. Jäsennetty haastattelu toteutetaan yleensä siten, että haastattelija keskustelee kunkin käyttäjän kanssa erikseen ja tavoitteena on kerätä mahdollisimman paljon tietoa myöhempää analysointia varten. (Parviainen ym. 2003, 48.) Menetelmän etuna on, että käyttäjiä voidaan haastatella samalla, kun he työskentelevät normaalissa toimintaympäristössään. Pyrkimyksenä on, että haastattelija esittää käyttäjille kysymyksiä heidän työstään mutta ei ota kantaa itse työhön. Tietoa ei analysoida vielä haastattelun aikana, vaan tavoitteena on ainoastaan kerätä raakadataa. Keskeistä CI-menetelmän soveltamisessa ovat käyttötilanteiden ymmärtäminen, käyttäjän osallistuminen suunnitteluprosessiin ja selkeän päämäärän asettaminen suunnitteluprosessille (Parviainen ym. 2003, 48). On hyvä myös varmistaa etukäteen, että haastattelu kohdistuu henkilöihin, jotka tarvitsevat tutkittavia työmenetelmiä ja että kyseiset henkilöt eivät suhtaudu haastatteluun negatiivisesti. Perinteisessä käyttäjien haastattelussa voi olla vaikeuksia löytää haastateltavia, jos nämä katsovat haastattelun vievän liikaa työaikaansa. CI-menetelmässä haastattelu kuitenkin koostuu pääosin käyttäjien tarkkailusta, kun he tekevät työtään ja ovat vuorovaikutuksessa kollegoidensa kanssa. Täten CI-menetelmän mukainen haastattelu ei välttämättä vie yhtä paljon käyttäjien työaikaa kuin perinteinen haastattelu**.** Parviainen ym. (2003) viittaavat Beyerin ja Holzblattin (1998) näkemykseen, jonka mukaan CI-prosessi on luonteeltaan enemmän tiedon löytämistä ja oppimista kuin arviointia ja testausta (Beyer ja Holtzblatt, 1998, 20).

Sekä jäsennetty että avoin käyttäjien haastattelu soveltuvat RVL:n tietojärjestelmäprojektien vaatimusten keräämiseen. Molempien haastattelumenetelmien etuna on, että ne ovat helposti omaksuttavissa ja sovellettavissa kaikenkokoisiin projekteihin. Avoin haastattelu ei vaadi merkittävästi taustatyötä, mutta jäsennetty haastattelu edellyttää ennalta kuhunkin tilanteeseen sopivien kysymysten laatimista.

4.3.2 Käyttäjien tarkkailu

Käyttäjiä tarkkailemalla voidaan selvittää, kuinka he toimivat vuorovaikutuksessa järjestelmän kanssa (Parviainen ym. 2003, 50). Kyseessä voi olla todellinen tai simuloitu

käyttötilanne. Tavoitteena on analysoida käyttäjien reaktioita ja päätellä, kuinka hyvin todellisen tai simuloidun järjestelmän suunnittelu on onnistunut ja missä on ongelmia. Tarkkailijat voivat pyytää käyttäjiä kuvailemaan tehtäviään, niiden tavoitteita ja taustalla vaikuttavia prosesseja (Gottesdiener, 2005, 84). Samalla voidaan kerätä vaatimuksia uutta järjestelmää varten. Tulosten oikeellisuuden kannalta on kuitenkin tärkeää, että ulkopuoliset tarkkailijat eivät puutu liikaa testihenkilöiden toimintaan.

Tarkkailemalla voidaan saavuttaa parempi ymmärrys käyttäjän ja järjestelmän välisestä vuorovaikutuksesta, kuin käyttäjälle esitettävillä kirjallisilla kysymyksillä, jotka eivät välttämättä kata kaikkea toimintaa (Wiegers, 2003, 97). Käyttäjien tarkkailu soveltuu esimerkiksi tehtävien analysointiin, kun ollaan suunnittelemassa kehitteillä olevan järjestelmän toimintalogiikkaa, tai olemassa olevan järjestelmän vahvuuksien ja heikkouksien arviointiin, kun tavoitteena on kerätä kehitteillä olevan järjestelmän vaatimukset. Lisäksi käyttäjien tarkkailua on mahdollista hyödyntää järjestelmäprototyypin kehittämisessä. Käyttäjien toimintaa seuraamalla voidaan analysoida, kuinka prototyypin suunnittelu ja toteutus ovat onnistuneet. (Parviainen ym. 2003, 51.) Kerätyn tiedon perusteella voidaan kehittää prototyypistä seuraava versio. Tämän jälkeen prototyyppi annetaan käyttäjien testattavaksi ja järjestetään uusi tarkkailukierros, jotta voidaan nähdä, paransivatko edelliseen prototyyppiin tehdyt muutokset järjestelmän toimintaa. Tarkkailukierroksia ja prototyypin kehittämistä voidaan toistaa niin kauan, että on saatu kehitettyä mahdollisimman hyvin asiakkaan tarpeita vastaava lopputulos, josta voidaan johtaa myös vaatimukset (Wiegers, 2003, 235).

Joskus voi olla tarpeen selvittää tarkemmin, miten käyttäjä kokee vuorovaikutuksen järjestelmän kanssa. Protokolla-analyysi (Protocol Analysis) on tekniikka, jota voidaan käyttää muodostamaan ymmärrys käyttäjän ajatusmalleista, kun hän suorittaa valmiiksi jäsenneltyä tehtävää (Parviainen ym. 2003, 55). Parviainen ym. (2003) viittaavat Ericssonin ja Simonin (1993) esittämään kuvaukseen Protocol Analysis-istunnosta. Istunnossa käyttäjä suorittaa annettua tehtävää ja selittää ajatteluprosessejaan suullisesti tarkkailijalle, joka ottaa tiedot ylös. Istunnon jälkeen käyttäjältä saatu tieto analysoidaan ja hänen todellisia ajatteluprosessejaan verrataan järjestelmän prosesseihin. Näin voidaan tunnistaa järjestelmän vahvuudet ja heikkoudet. (Parviainen, 2003, 51.)

Myös käyttäjien tarkkailu soveltuu RVL:n tietojärjestelmäprojektien vaatimusten keräämiseen. Menetelmää voidaan soveltaa esimerkiksi siten, että käyttäjien toimintaa tarkkaillaan jo valmiissa järjestelmässä, jotta voidaan kerätä tietoa nykyisen tai uuden

järjestelmän kehitystyötä varten. Myös kuvitteellinen käyttötilanne on lista toteuttaa samalla tavalla. Jos tarkkailutilanteen järjestäminen edellyttää järjestelmäprototyypin rakentamista, tällöin menetelmän kustannukset saattavat nousta merkittävästi, mikä rajoittaa menetelmän hyödyntämistä varsinkin pienissä projekteissa. Käyttäjien tarkkailu edellyttää projektiryhmältä taustatyönä tarkkailtavan käyttötilanteen suunnittelua ja järjestämistä.

4.3.3 Prototypointi

Prototypoinnin (Prototyping) tavoitteena on luoda nopeasti alustava ja edullinen järjestelmän prototyyppi, joka on saatavana jo aikaisessa vaiheessa kehitysprosessia (Parviainen ym. 2003, 50). Prototypoinnin avulla voidaan kartoittaa käyttäjien tarpeita ja helpottaa vaatimusten löytämistä ja tunnistamista, sillä toisinaan käyttäjillä on vaikeuksia hahmottaa tarpeitaan ilman konkreettista mallia järjestelmästä (Wiegers, 2003, 233). Aluksi käyttäjiä haastatellaan ja selvitetään alustavat vaatimukset, joiden pohjalta rakennetaan yksinkertainen ja vuorovaikutteinen käyttöliittymäprototyyppi (McConnell, 1996, 117). Myös Beyer ja Holtzblatt (1998) korostavat, että ensimmäisten prototyyppien on syytä olla mahdollisimman yksinkertaisia hahmotelmia, jotta epäoleelliset yksityiskohdat eivät häiritse järjestelmän perusrakenteen ymmärtämistä ja kokonaisuuden hahmottamista (Beyer ja Holtzblatt, 1998, 372). Prototyyppiä kehitetään jatkuvasti käyttäjiltä saadun palautteen perusteella, jolloin käyttäjät saadaan mukaan järjestelmän kehitystyöhön jo varhaisessa vaiheessa projektia. Prototyyppiä laajennetaan, kunnes sen avulla voidaan havainnollistaa kattavasti kaikki ohjelmiston toiminnalliset osa-alueet. Kun prototyypistä tehdään koko järjestelmän kattava, suunnittelijat hahmottavat ohjelmiston paremmin, mikä taas mahdollistaa paremman arkkitehtuurin ja moduulisuunnittelun. Käyttäjiltä saatu palaute auttaa lopulta luomaan loppukäyttäjille erittäin tyydyttävän ohjelmiston juuri ja juuri ”vaatimukset täyttävän” sijasta (McConnell, 1996, 117). Prototyyppiin perustuva vaatimusten kerääminen auttaa sovelluksen tulevia käyttäjiä ja muita sidosryhmiä saavuttamaan yhteisen ymmärryksen järjestelmän vaatimuksista (Wiegers, 2003, 233). Sen lisäksi että prototyypit helpottavat yhteisen päämäärän hahmottamista rakennettavasta järjestelmästä, ne toimivat käyttäjien ja projektiryhmän suunnittelijoiden välisenä kommunikointikielenä, jota kumpikin osapuoli ymmärtää (Beyer ja Holtzblatt, 1998, 372). Beyer ja Holtzblatt mainitsevat tästä esimerkkinä tilanteen, jossa käyttäjä ehdottaa muutosta prototyypin toimintaan tietämättä sitä, miten laajasti muutos vaikuttaa hahmoteltuun järjestelmäkokonaisuuteen ja mitä muutoksen toteuttaminen vaatii. Tällöin suunnittelijat voivat prototyypin avulla selvittää, kuinka eh-

dotettu muutos sijoittuu suhteessa järjestelmän kokonaisrakenteeseen ja edellyttääkö muutos suunnittelun painopisteen uudelleenarviointia (Beyer ja Holtzblatt, 1998, 372). Sekä käyttäjät että suunnittelijat katsovat kehitystyötä omista näkökulmistaan, mutta prototyyppi tarjoaa kaikille projektin osapuolille yhteisen näkymän rakennettavaan järjestelmään.

Prototypoinnin avulla voidaan minimoida riskejä, jotka liittyvät myöhäisessä projektin vaiheessa muuttuviin vaatimuksiin. Kesken projektin muuttuvat vaatimukset ovat ohjelmistokehitysprojektin kohtaamista riskeistä kolmen vakavimman joukossa (Leffingwell ja Widrig, 2006, 7). Myöhäiset muutokset johtuvat usein siitä, että käyttäjät eivät vielä projektin alussa tiedä, mitä ohjelmalta haluavat, ennen kuin he näkevät sen ja muuttavat mieltään kesken projektin (McConnell, 1998, 117). Prototypointi parantaa käyttäjien tietoisuutta vaatimuksista, sillä prototyypin jalostus etenee heidän palautteensa pohjalta.

Lopullinen prototyyppi toimii ohjelmiston pohjamäärittelynä, joka alistetaan muutostenhallintaan. Kehitettävä ohjelmisto noudattaa prototyyppiä ja vain muutostenhallintaprosessin läpäisseet muutokset huomioidaan toteutuksessa. (McConnell, 1998, 117.) McConnellin mukaan prototyypistä on paljon arvokasta etua sen jälkeen, kun se on hyväksytty pohjamäärittelyksi, sillä vakaampien vaatimusten lisäksi kattava käyttöliittymäprototyyppi mahdollistaa loppukäyttäjädokumentaation ja testaussuunnitelmien kehittämisen samanaikaisesti arkkitehtuurin, moduulisuunnittelun ja toteutuksen rinnalla.

Prototypointiin on olemassa erilaisia tekniikkoja ja lähestymistapoja, mutta yleensä ne kuuluvat jompaankumpaan kahdesta pääryhmästä: kertakäyttöinen prototypointi ja evolutiivinen prototypointi (McConnell, 1996, Wiegers, 2003). Kertakäyttöisessä prototypoinnissa luodaan poisheitettävä prototyyppi helpottamaan järjestelmävaatimusten hankkimista ja analysointia, jotka ovat usein asiakkaan näkökulmasta vaikeita ymmärtää. Projektin alussa tapahtuvan vaatimusten selvityksen ja suunnitteluvaihtoehtojen kartoituksen lisäksi prototypoinnin tavoitteena voi olla myös täysin valmiin järjestelmän luominen, jolloin kyseessä on evolutiivinen prototypointi. (Wiegers, 2003, 238.) Evolutiivisessa prototypoinnissa tehdään rajoitetun toiminnallisuuden sisältävä järjestelmäprototyyppi käyttäjien saataville kehitysprosessin aikaisessa vaiheessa. Tätä prototyyppiä muokataan ja laajennetaan myöhemmin, jotta voidaan tuottaa lopullinen järjestelmä. (Wiegers, 2003, 238.)

Wiegers erittelee prototyypit horisontaalisiin ja vertikaalisiin prototyyppeihin (Wiegers, 2003, 235-236). Horisontaalinen prototyyppi pitää sisällään toiminnallisesti vähäisen tai rajoitetun version järjestelmän käyttöliittymästä. Horisontaalisen prototyypin tavoitteena on kuvata pinnallisesti, miltä valmis järjestelmä tulee näyttämään ja mitä toimintoja siihen aiotaan toteuttaa. Vertikaalinen prototyyppi taas sisältää käyttöliittymän lisäksi valmiita toimintoja ja toimii ainakin joiltain osin samoin, kuin valmiin järjestelmän on suunniteltu toimivan. Vertikaalisten prototyyppien avulla voidaan simuloida valmiin järjestelmän toimintoja jo varhaisessa vaiheessa projektia. (Wiegers, 2003, 236). Kehitystiimi voi vertikaalisten prototyyppien avulla hankkia kokemusta järjestelmän suunnittelusta ja kehityksestä jo projektin alkuvaiheessa (Gottesdiener, 2005, 78). Tällöin on myös mahdollista löytää vaatimuksiin, suunnitteluun ja toteutukseen liittyviä ongelmia, jotka muutoin havaittaisiin vasta myöhemmässä vaiheessa projektia. Näin prototypoinnin avulla voidaan minimoida kyseisiin työvaiheisiin myöhemmin kohdistuvia riskejä.

Myös prototypointi soveltuu tietyin varauksin RVL:n tietojärjestelmäprojektien vaatimusten keräämiseen. Prototyyppiin perustuvan vaatimusten keräämisen etuna on, että suunnittelijat hahmottavat ohjelmiston paremmin ja käyttäjät saadaan mukaan järjestelmän kehitystyöhön jo projektin alkuvaiheessa. Tämän ansiosta projektiryhmä, sovelluksen tulevat käyttäjät ja muut sidosryhmät voivat saavuttaa yhteisen ymmärryksen järjestelmän vaatimuksista, mikä vähentää tarvetta muuttaa vaatimuksia myöhemmässä vaiheessa projektia. Sen sijaan järjestelmäprototyypin rakentamisesta aiheutuvat kustannukset ja siihen kuluva aika sekä prosessin edellyttämä tekninen osaaminen ovat tekijöitä, joiden vuoksi menetelmää ei ole mielekästä hyödyntää kaikissa projekteissa, kun tavoitteena on viedä vaatimusten kerääminen läpi osana kustannustehokasta vaatimusprosessia. Tämä pätee erityisesti pieniin ja resursseiltaan rajallisiin projekteihin sekä projekteihin, joilla on aikataulupaineita. Tällaisissa projekteissa tulisi käyttää tekniseltä toteutukseltaan ja vaativuudeltaan prototypointia kevyempiä menetelmiä. Testausvalmiin prototyypin rakentaminen edellyttää huomattavasti työaikaa, teknistä osaamista ja taloudellisia resursseja.

4.3.4 Käyttöskenaariot

Skenaariot ovat esimerkkejä käyttäjän ja järjestelmän välisistä vuorovaikutustilanteista ja koostuvat jaksottaisten toimintojen kuvauksista. Skenaariot ovat hyödyllisiä, koska loppukäyttäjät ja muut järjestelmän sidosryhmät ymmärtävät helpommin tosielämän esimerkkejä kuin abstrakteja toimintojen kuvauksia (Parviainen ym., 2003, 51). Tämän

vuoksi skenaariot ovat sovellettavissa järjestelmävaatimusten hankkimiseen ja selventämiseen. Kun loppukäyttäjä simuloi järjestelmän käyttöä, hänen kommenttinsa, ongelmansa ja ehdotuksensa otetaan ylös ja hänen toimintaansa seurataan kunkin skenaarion osalta. Skenaario pitää sisällään kuvauksen tilasta, jossa ollaan skenaarioon siirryttäessä ja tilasta kun skenaario on toteutunut. Tämän lisäksi skenaariot sisältävät yleensä kuvaukset normaalista tapahtumaketjusta, poikkeuksista normaaliin tapahtumaketjuun ja samanaikaisesti meneillään olevista asioista. (Kotonya ja Sommerville, 1998, 65.) Wiegers (2003) täsmentää käyttötapausten ja käyttöskenaarioiden välistä suhdetta. Käyttötapaus kuvaa abstraktilla tasolla tietyn tehtävän asiayhteydessään, mitä muuta järjestelmässä tapahtuu samaan aikaan ja mitä pitäisi saada aikaan. Käyttöskenaario taas vastaa kysymykseen, miten tavoitteeseen voidaan päästä ja kuvaa tarvittavat askeleet tehtävän suorittamiseksi. (Wiegers, 2003, 134).

Skenaarioiden avulla voidaan tunnistaa tehtävät, joihin käyttäjä tarvitsee järjestelmää. Tehtävien perusteella voidaan määritellä järjestelmän käyttötapaukset, joista voidaan johtaa toiminnalliset vaatimukset (Wiegers, 2003, 134-135). Skenaariopohjainen vaatimuskehitys (Scenario-based Requirements Engineering) on menetelmä, jossa yhdistyy käyttötapaus-lähestymistapa oliolähtöiseen sovelluskehitykseen. Skenaariot luodaan esittämään mahdollisen toiminnan polkuja, joita analysoimalla voidaan kehittää järjestelmävaatimuksia. (Parviainen ym. 2003, 56.)

Skenaariot ovat käyttökelpoisia vaatimusten keräämiseen myös RVL:n tietojärjestelmäprojekteissa. Käytännössä menetelmää voitaisiin soveltaa siten, että loppukäyttäjän ja järjestelmän vuorovaikutusta seurataan kuvitteellisessa käyttötilanteessa, jonka projektiryhmä on luonut organisaation todellisten prosessien pohjalta. Koehenkilön toimintaa, kommentteja, ongelmia ja ehdotuksia analysoimalla voidaan tunnistaa sellaisiakin vaatimuksia, jotka eivät kävisi ilmi suorassa haastattelussa. Skenaarioihin perustuvan vaatimusten keräämisen etuna on, että loppukäyttäjän tarpeet voidaan kartoittaa kokonaisvaltaisesti ottaen huomioon sekä hänen mielipiteensä että reaktiot ja vuorovaikutus kuvitteellisen järjestelmän kanssa kussakin skenaariossa. Menetelmän hyödyntäminen edellyttää projektiryhmältä taustatyönä testattavien skenaarioiden määrittelyä ja luontia sekä organisaation prosessien tuntemusta.

4.3.5 Ryhmätekniikat

Ryhmätekniikoilla pyritään edistämään vaatimusten löytämistä ja vaatimusten taustalla vaikuttavien tekijöiden ymmärrystä. Ryhmätekniikat perustuvat ajatukseen, jonka mu-

kaan ryhmä ihmisiä voi saavuttaa paremman ymmärryksen vaatimuksistaan, kuin jos he työskentelisivät yksilöinä (Parviainen ym. 2003). Ryhmätekniikoilla voidaan luoda uusia käytännöllisiä ideoita, tukea luovaa ajattelua ja edesauttaa projektiryhmän sekä käyttäjien välistä yhteisymmärrystä vaatimuksista. Lisäksi tavoitteena on tukea järjestelmän sidosryhmien kommunikointia ja saada heidät kaikki mukaan järjestelmäkehitykseen. Pyrkimyksenä on, että sidosryhmät voivat tunnistaa nykyisen tilanteen ja halutun tavoitetilan sekä laatia etenemissuunnitelman tavoitteen saavuttamiseksi (Parviainen ym. 2003; Hollander ja Mirlocca, 1995). Ryhmätekniikoita ovat muun muassa aivoriihi (Brainstorming) ja JAD (Joint Application Development).

Brainstorming on ryhmätekniikka, jota voidaan soveltaa ongelman määrittelyyn ja mahdollisten ratkaisujen kartoittamiseen (Parviainen ym., 2003, 48). Brainstorming-istunnossa on kolmenlaisia rooleja: johtaja, ryhmän jäsen ja kirjuri. Johtaja valvoo, että istunto etenee sääntöjen mukaan ja tarvittaessa muistuttaa osallistujia yhteisestä päämäärästä. Istunnon alussa osallistujille esitetään selvä kysymys, johon haetaan vastausta. Tämän jälkeen ryhmän jäsenet voivat vapaasti esittää ehdotuksia ongelman ratkaisuksi. Kirjuri kirjaa ideat ylös siten, että kaikki ryhmässä voivat nähdä ne. Tavoitteena on kerätä ryhmältä tulevia käyttäjiä lyhyessä ajassa suuri määrä ideoita, jotka analysoidaan vasta istunnon jälkeen. (Leffingwell ja Widrig, 2003, 120-121.) Brainstorming-prosessi pitää sisällään ideoiden tuottamisen ja vähentämisen. Ideoiden tuottamisen pääasiallinen tavoite on saada aikaan niin paljon ideoita kuin mahdollista. Vähentämisessä taas on kyse ideoiden analysoimisesta, karsimisesta, ryhmittelystä, priorisoinnista ja uudelleenmäärittelystä. (Leffingwell ja Widrig, 2006, 122.) Brainstorming- tekniikkaa voidaan hyödyntää tunnistamattomien vaatimusten löytämiseen ja vaatimusten hankkimiseen hankkeen eri sidosryhmiltä (Parviainen ym., 2003, 48). Brainstorming-tekniikkaa on sovellettu ohjelmistokehitysprosessien parantamiseen tähtäävän SPI (Software Process Improvement) -menetelmän kehittämiseen ja lisäksi Brainstorming on myös yksi SPI-menetelmään sisällytetyistä tekniikoista (Komi-Sirviö, 2004, 24).

Brainstorming-istunnot soveltuvat vaatimusehdotusten keräämiseen myös RVL:n tietojärjestelmäprojekteissa ja istunnoista on jo käytännön kokemusta. Menetelmän ansiosta on mahdollista saada laaja joukko sidosryhmiä mukaan ongelman määrittelyyn ja mahdollisten ratkaisujen ja vaatimusehdotusten ideoimiseen, mikä edesauttaa yhteisten ongelmien, ratkaisuiden ja päämäärän löytämistä. Menetelmän hyödyntäminen edellyttää projektiryhmältä taustatyönä istuntoa ohjaavien kysymysten laatimista, kykyä ohjata istuntoa ja analysoida kerättyjä ratkaisu- ja vaatimusehdotuksia.

JAD on vaatimusten keräämiseen ja hallintaan soveltuva ryhmätekniikka, joka on alun perin kehitetty tietokone-pohjaisten järjestelmien suunnitteluun. IBM kehitti tekniikan 1970-luvun lopulla ja se on yhä nykyäänkin käyttökelpoinen menetelmä vaatimusten keräämiseen käyttäjiltä ja muilta sidosryhmiltä (Parviainen ym., 2003, 24). JAD-istunnon tavoitteena on saada aikaan prosessi, jossa järjestelmän tulevat käyttäjät ja sovelluskehittäjät mallintavat rakennettavaa järjestelmää ja sen vaatimuksia yhdessä. Vaatimusten tunnistamisen ohella tavoitteena on hakea ratkaisua havaittuihin ongelmiin mahdollisimman aikaisessa vaiheessa projektia, jolloin virheiden korjaaminen on edullisinta. Vaatimusten kartoituksessa voidaan käyttää apuna prototypointia ja käyttötapausten määrittelyä. (Parviainen ym., 2003, 24.)

JAD-menetelmän ansiosta projektiin käytettävää aikaa voidaan lyhentää merkittävästi, eliminoida viivästyksiä ja väärinkäsityksiä sekä parantaa järjestelmän laatua (Hollander ja Mirlocca, 1995, 26-27). Vaatimusten laatu paranee, kun järjestelmän käyttäjät ja sovelluskehittäjät kartoittavat niitä yhdessä. Näin voidaan välttää tilanne, jossa vaatimukset ovat joko liian yksityiskohtaiset tai vastaavasti epämääräiset, mikä saattaa aiheuttaa ongelmia projektin myöhemmissä toteutus- ja hyväksymisvaiheissa. (Hollander ja Mirlocca, 1995, 27). Hollanderin ja Mirloccan mukaan JAD-menetelmällä voidaan parantaa järjestelmäkehityksen laatua ja lyhentää kehitykseen kuluvaa aikaa, mikä parantaa ratkaisevasti organisaation kykyä reagoida nopeasti muuttuvan ympäristön haasteisiin.

Myös JAD soveltuu vaatimusten keräämiseen RVL:n tietojärjestelmäprojekteissa. Menetelmä mahdollistaa sidosryhmien osallistumisen sovelluskehittäjien ohella rakennettavan järjestelmän ja sen vaatimusten mallintamiseen. Tämä ohjaa projektin toteutusta loppukäyttäjien todellisten tarpeiden suuntaan ja vähentää loppukäyttäjien vähäiseen osallistumiseen liittyviä riskejä. Brainstorming-istuntojen tavoin JAD edesauttaa yhteisten ongelmien, ratkaisuiden ja päämäärän löytämistä, kun järjestelmän tulevat käyttäjät ja sovelluskehittäjät toimivat yhdessä. Vaatimusten tunnistamisen ohella tavoitteena on hakea ratkaisua havaittuihin ongelmiin mahdollisimman aikaisessa vaiheessa projektia, jolloin virheiden korjaaminen on edullisinta. Menetelmän hyödyntäminen edellyttää projektiryhmältä sovellusalueen tuntemusta, taitoa järjestelmäanalyysiin ja kykyä keskustella eri sidosryhmien kanssa heidän kielellään (Cline, 2000). Lisäksi menetelmä vaatii projektiryhmältä taustatyönä perehtymistä käyttäjävaatimuksiin ja niihin liittyviin prosesseihin sekä ongelmiin (Spina ja Rolando, 2002).

4.3.6 Vaatimusten uudelleenkäyttö

Vaatimusten kerääminen ei aina ole uuden tiedon tuottamista ja luokittelua. Joissain tapauksissa vaatimusten keräämiseen kuluvaa aikaa ja resursseja sekä prosessiin liittyviä riskejä voidaan rajoittaa huomattavasti käyttämällä vaatimuksia uudelleen (Parviainen ym., 2003, 76). Vaatimusten uudelleenkäytöllä voidaan parantaa sekä kehitystyön tuottavuutta että järjestelmän laatua (Toval ym., 2002). Uudelleenkäyttö tarkoittaa, että samalla sovellusalueella olevan toisen järjestelmän vaatimuksia sovelletaan uuteen rakennettavaan sovellukseen. Tavoitteena on tunnistaa sellaiset vaatimuskuvaukset, joita voidaan käyttää uudelleen joko kokonaan tai osittain minimaalisin muutoksin ja samalla rajoittaa kehitystyöhön kuluvia resursseja (Toval ym., 2002). Kotonya ja Sommerville (1998) muistuttavat, että uusia järjestelmiä kehitettäessä on yleisesti hyvä käyttää jo olemassa olevaa tietämystä uudelleen niin paljon kuin mahdollista. Vaatimusten uudelleenkäyttö järjestelmien välillä on mahdollista esimerkiksi tilanteissa, joissa vaatimus liittyy selkeästi samaan sovellusalueeseen tai tiedon esittämisen tapa on sama. (Kotonya ja Sommerville, 1998, 72.) Vaatimusten uudelleenkäytöllä voidaan saavuttaa enemmän hyötyä kehitystyölle kuin vain suunnittelun tai koodin uudelleenkäytöllä, koska samalla on mahdollista hyödyntää myös järjestelmän vaatimusten ja niistä kumpuavien arkkitehtuurin ja toteutuksen välisiä jäljitysyhteyksiä, minkä ansiosta myöhempää kehitysprosessia voidaan nopeuttaa (Toval ym., 2002).

Myös RVL:n tietojärjestelmäprojekteissa on tilanteesta riippuen mahdollista hyödyntää jo olemassa olevien järjestelmien vaatimuksia. Vaatimusten uudelleenkäyttö voi olla perusteltua varsinkin saman sovellusalueen järjestelmien osalta. Uutta järjestelmää suunniteltaessa voidaan itse vaatimusten lisäksi hyödyntää aikaisemmissa tietojärjestelmäprojekteissa saatuja kokemuksia laajemminkin.

4.3.7 Analysointi ja yhteenveto

Vaatimusten kerääminen ja analysointi ovat toisistaan läheisesti riippuvaisia prosesseja. Kerätyt vaatimukset ovat vasta alustavia ehdotuksia siitä, kuinka järjestelmän odotetaan toimivan, mutta ne eivät välttämättä ole vielä valmiita järjestelmävaatimuksia. Ennen kuin vaatimusehdotuksista voidaan jalostaa järjestelmävaatimuksia, niiden mahdolliset ristiriidat, päällekkäisyydet ja epäjohdonmukaisuudet on syytä tutkia. (Parviainen, ym. 2003, 29.) Vaatimuksia voidaan luokitella ja analysoida jo keräämisvaiheessa, mutta vaatimukset kokonaisuudessaan huomioiva perusteellinen analysointi on mahdollista vasta siinä vaiheessa, kun pääosa järjestelmän vaatimuksista on kerätty. Vaatimusten analysointivaiheessa etsitään yhteneväisyyksiä ja eroavaisuuksia kerättyjen vaatimusten

väliltä ja yhdistetään vaatimukset niin, että kaikki olennaiset piirteet jäävät
le (McConnel, 1998, 116).

Järjestelmään kohdistuvien vaatimusten analysoinnilla pyritään saamaan aikaan suunnitteluehdot, joiden pohjalta voidaan määritellä lopulliset järjestelmän vaatimukset. Vaatimusehdotusten analysointi jalostaa järjestelmän vaatimuksia, edesauttaa toimintaprosessien löytymistä ja tukee järjestelmän toiminnallista suunnittelua. Kun kerätyt vaatimusehdotukset on analysoitu, niiden pohjalta voidaan määritellä lopulliset järjestelmävaatimukset. Lopulta vaatimusmäärittely tarjoaa kehittäjälle ja asiakkaalle keinot arvioida laatua, kun järjestelmä on rakennettu. (Pressman, 1992, 55-57.)

Olen käsitellyt tässä vaatimusten keräämistä prosessina, jonka tavoitteena on hankkia ja analysoida tietoa järjestelmän vaatimusmäärittelyn tueksi. Tavoitteena ei niinkään ole tuottaa valmiita järjestelmävaatimuksia, vaan kerätä tietoa, jonka perusteella voidaan hakea vastausta kysymykseen, mitä vaatimuksia järjestelmän on täytettävä, jotta se parhaiten vastaisi sidosryhmien odotuksia. Kerätyn tiedon pohjalta voidaan vaatimusmäärittelyvaiheessa muodostaa varsinaiset järjestelmävaatimukset. Käsittelen seuraavaksi vaatimusten määrittelyä.

4.4 Vaatimusten määrittely

Vaatimusten määrittelyvaiheen voidaan katsoa jatkuvan niin kauan, kunnes projektiryhmä on saanut aikaan selkeän ja vakaan käsityksen siitä, mitä käyttäjät haluavat ohjelmiston tekevän. Määrittelyvaiheen tulokset saatetaan kirjalliseksi dokumentiksi, tapahtumakaavioiksi tai vuorovaikutteiseksi käyttöliittymäprototyypiksi. (McConnell, 1998, 116.) Vaatimusten määrittely pitää sisällään vaatimusten laatimisen, jalostamisen, järjestämisen ja määrittelyprosessin lopputulosten dokumentoinnin. Itse määrittely ja määrittelyprosessin ohjaus ovat usein projektiryhmän tai ulkopuolisen asiantuntijan vastuulla. Sen sijaan määrittelyn tulosten arvioinnissa ja vaatimusten vahvistamisessa myös loppukäyttäjien osallistuminen on tärkeää. (Gottesdiener, 2005, 12.) Määrittelyn lopputuloksia arvioitaessa on syytä varmistaa, että kuvatut vaatimukset todella edustavat sitä, mitä loppukäyttäjät ja organisaation muut sidosryhmät järjestelmältä haluavat (Kotonya ja Sommerville, 1998, 37).

4.4.1 Vaatimustyypit

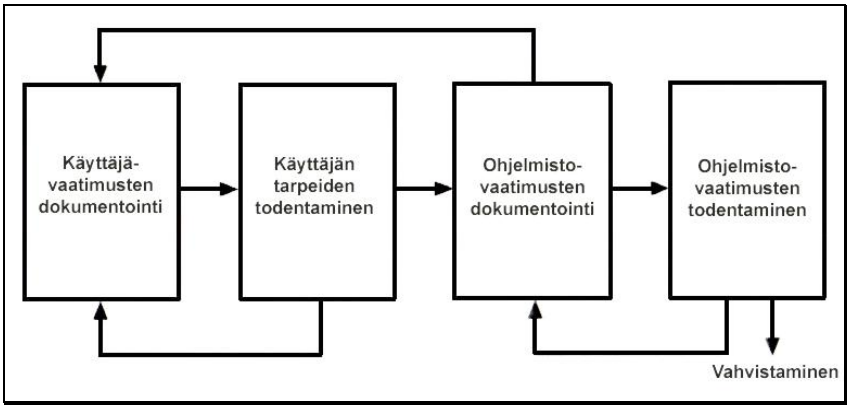
Vaatimuksia voidaan tarkastella sekä käyttäjien että ohjelmiston näkökulmasta. Käyttäjävaatimukset ovat kuvaus käyttäjien tarpeista, mitä he tarvitsevat ja miksi. Käyttäjävaatimuksia voidaan kuvata luonnollisella kielellä pelkistetysti ja abstraktilla tasolla siten,

että käyttäjät voivat ne ymmärtää. Sen sijaan ohjelmistovaatimusten sa tarvitaan enemmän tarkkuutta ja yksityiskohtaisempia määritelmiä. (Leffingwell ja Widrig, 2006, 232.) Ohjelmistovaatimukset ovat yksityiskohtainen kuvaus sekä sovelluksen toiminnoista, laatutekijöistä ja järjestelmärajapinnoista että suunnitteluun ja toteutukseen liittyvistä rajoituksista (Gottesdiener, 2005, 11). Ohjelmistovaatimukset kertovat kehittäjille, mitä järjestelmän täytyy tehdä. Tavoitteena on kuvata ohjelmistovaatimukset niin tarkalla tasolla, että kuvausten perusteella voidaan ohjelmoida ja myöhemmin testata järjestelmän toimintoja. Riittävän yksityiskohtaisten vaatimuskuvausten avulla voidaan testausvaiheessa varmistaa, että järjestelmän toiminnot vastaavat määriteltyjä vaatimuksia. (Leffingwell ja Widrig, 2006, 234.)

Ohjelmistovaatimukset luokitellaan joko toiminnallisiin tai ei-toiminnallisiin. (mm. McConnell, 1996, Leffingwell ja Widrig, 2006, Gottesdiener, 2005.) Toiminnalliset ohjelmistovaatimukset ilmaisevat kuinka järjestelmä käyttäytyy ja kuvaavat järjestelmän syötteet, tulosteet ja käyttäjille tarjottavat toiminnot. Nämä ovat järjestelmän toimintaan suuntautuneita vaatimuksia, joita voidaan havainnollistaa esimerkiksi käyttötapauksien avulla. Kaikkia vaatimuksia ei kuitenkaan ole mahdollista yksilöidä ja kuvata toiminnallisesta näkökulmasta, jolloin kyseessä ovat ei-toiminnalliset vaatimukset. Näitä ovat esimerkiksi järjestelmän käytettävyys, luotettavuus, suorituskyky ja tuettavuus. (Leffingwell ja Widrig, 2006, 259.) Toiminnallisten ja ei-toiminnallisten vaatimusten lisäksi Leffingwell ja Widrig mainitsevat suunnittelurajoitukset yhtenä vaatimustyyppinä. Myös vaatimuksen määritelmä tukee tätä tulkintaa. Vaatimus kuvaa toiminnallisuutta, joka ohjelmistolla tai sen komponentilla täytyy olla, jotta voidaan täyttää sopimus, määritys tai jokin muun muodollisesti esitetty dokumentti (Dorfman ja Thayer, 1990, 29). Suunnittelurajoitukset vaikuttavat järjestelmän suunnitteluun ja kehitysprosessiin mutta eivät välttämättä järjestelmän ulkoiseen toimintaan. Suunnittelurajoitukset voivat olla luonteeltaan teknisiä, liiketoiminnallisia tai sopimuksellisia velvoitteita, sääntöjä tai pakollisia standardeja, jotka rajoittavat mahdollista toteutusta ja jotka siten muuntuvat sovelluksen kehitystyöhön kohdistuviksi vaatimuksiksi. (Leffingwell ja Widrig, 2006, 263.)

4.4.2 Määrittelyprosessin vaiheet

Gottesdiener (2005) jakaa kuvan 4-4 mukaisesti vaatimusten määrittelyprosessin neljään vaiheeseen, jotka ovat käyttäjävaatimusten dokumentointi, käyttäjän tarpeiden vahvistaminen, ohjelmiston vaatimusten dokumentointi ja ohjelmistovaatimusten vahvistaminen (Gottesdiener, 2005, 233).



Kuva 4-4: Vaatimusmäärittelyprosessin vaiheet
Lähde: Gottesdiener, 2005

Gottesdienerin mukaan käyttäjävaatimusten dokumentoinnin tavoitteena on kuvata käyttäjien näkökulmasta, mitä he tarvitsevat ja miksi, ja keskittää eri käyttäjäryhmien tarpeet saman muodollisen määrittelyn alle. Käyttäjävaatimukset kuvataan korkealla tasolla siten, että projektiryhmä voi päästä käyttäjien kanssa yksimielisyyteen siitä, mitä ominaisuuksia järjestelmällä on oltava, jotta se täyttää käyttäjien tarpeet. (Gottesdiener, 2005, 231.) Näin käyttäjävaatimusdokumentti toimii siltana asiakkaan toiminnallisten vaatimusten ja ohjelmistovaatimusten välillä. Wiegers (2003) painottaa käyttötapausten tarpeellisuutta erityisesti korkealla tasolla kuvattavien käyttäjävaatimusten dokumentoinnissa. Käyttötapauksia kuvaamalla voidaan havainnollistaa järjestelmän toimintaa loppukäyttäjien näkökulmasta ja täten helpottaa käyttäjävaatimusten dokumentointia. Sen sijaan ohjelmistovaatimusten dokumentointiin käyttötapaukset eivät yksistään riitä. (Wiegers, 2003, 165.)

Käyttäjien tarpeiden todentaminen on tärkeää, jotta voidaan varmistua siitä, että kuvatut käyttäjävaatimukset vastaavat käyttäjien tarpeita ja asiakkaan liiketoiminnan tarpeita. Tässä vaiheessa sidosryhmien on tarkistettava, että vaatimukset ovat kokonaisia, johdonmukaisia ja korkealaatuisia ja tarvittaessa vaatimusdokumentaatiota voidaan korjata. (Gottesdiener, 2005, 234.) Loppukäyttäjien tarpeiden todentamista helpottaa, jos he ovat mukana määrittelyprosessissa. Leffingwellin ja Widrigin (2006) mukaan asiakkaan osallistuminen määrittelyprosessiin helpottaa sekä käyttäjien tarpeiden että ohjelmistovaatimusten todentamista (Leffingwell ja Widrig, 2006, 223).

Ohjelmiston vaatimusten dokumentoinnin kohderyhmänä ovat sovelluksen toteuttajat. Ohjelmistovaatimusten määrittelydokumentissa (Software requirements specification, SRS) kuvataan toiminnalliset vaatimukset, laatutekijät, järjestelmärajapinnat sekä suunnitteluun- ja toteutukseen liittyvät rajoitukset. Määrittelydokumentti on tarkka asiakirja

vaatimuksista ja sen avulla voidaan suunnitella, kehittää ja testata ohjelmistosovellusta. (Gottesdiener, 2005, 237.) Myös Wiegers (2003) korostaa ohjelmistovaatimusten dokumentoinnin merkitystä määrittelyprosessissa. Hänen mukaansa dokumentoinnissa tiivistyy määrittelyprosessin lopputulos ja määrittelyprosessin aikana tuotetut dokumentit ilmentävät asiakkaiden ja sovelluskehittäjien välistä yhteisymmärrystä rakennettavasta järjestelmästä. (Wiegers, 2003, 165.)

Ohjelmistovaatimusten todentamisella pyritään varmistamaan, että ohjelmiston vaatimusdokumentaatio kuvaa oikein järjestelmän suunniteltuja toimintoja ja ominaisuuksia ja että ohjelmistovaatimukset on johdettu oikein käyttäjävaatimuksista, järjestelmävaatimuksista ja muista lähteistä. Tässä vaiheessa voidaan myös tunnistaa ja korjata tarpeettomia ja vääriä vaatimuksia. Lopuksi on syytä varmistaa, että vaatimusmäärittely tarjoaa riittävän perustan edetä suunnittelussa, rakentamisessa ja testauksessa ja samalla vaatimusmäärittely myös vahvistetaan. (Gottesdiener, 2005, 261.) Vaatimusten todentamisella voidaan ehkäistä virheellisten tai tarpeettomien vaatimusten siirtymistä toteutukseen ja valmiiseen järjestelmään. Asiakkaan osallistuminen on tärkeää sekä käyttäjätarpeiden todentamisen että lopullisten ohjelmistovaatimusten vahvistamisen kannalta ja parantaa määrittelyprosessin luotettavuutta. Leffinwell ja Widrig (2006) kuitenkin huomauttavat, että tästä huolimatta osa vaatimusvirheistä voi siirtyä valmiiseen järjestelmään, jolloin virheistä johtuvat ongelmat havaitaan vasta ylläpitovaiheessa (Leffingwell ja Widrig, 2006, 12).

Muodollinen vaatimusten määrittelyprosessi on projektin onnistumisen kannalta ratkaisevan tärkeä tekijä. Määrittelyn selkeä dokumentointi ja vahvistaminen sekä käyttäjä- että ohjelmistovaatimusten osalta on edellytys sille, että vaatimukset voidaan ymmärtää ja kuvata oikein sekä järjestelmän loppukäyttäjien että sovelluskehittäjien näkökulmasta. Näihin työvaiheisiin on syytä panostaa riittävästi aikaa ja asiantuntemusta, sillä määrittelyprosessissa säästäminen voi osoittautua kalliiksi virheeksi myöhemmässä vaiheessa projektia. Vaikka vaatimusmäärittelyn kriittinen merkitys ohjelmistoprojektin onnistumiselle tunnustetaan laajasti, silti huomattava osa valmiissa ohjelmistossa esiintyvistä virheistä johtuu juuri puutteellisesta vaatimusmäärittelystä (Leffingwell ja Widrig, 2006, 9).

McConnell (1998) muistuttaa, että kustakin huonosti määritellystä vaatimuksesta aiheutuneen virheen korjaaminen esimerkiksi vasta sovelluksen ylläpitovaiheessa maksaa moninkertaisesti enemmän, kuin jos virhe havaitaan ja korjataan jo määrittelyvaiheessa

(McConnell, 1998, 116). Myös Leffingwellin ja Widrigin (2006) havainnot tukevat tätä näkemystä, minkä lisäksi he huomauttavat, että määrittelyvaiheen jälkeen havaittujen vaatimusvirheiden korjaaminen voi joissain tapauksissa olla niin kallista, että virheen korjaamisen hinta ylittää virheestä aiheutuvan haitan arvon (Leffingwell ja Widrig, 2006, 12). Myös McConnell (1998) viittaa tutkimuksiin, joiden mukaan virheen korjaaminen ylläpitovaiheessa voi olla jopa 200 kertaa kalliimpaa kuin jos se havaitaan ja korjataan jo määrittelyvaiheessa (McConnell, 1998, 116). Vaatimusvirheisiin liittyvien merkittävien kustannusriskien vuoksi myös RVL:n tietojärjestelmäprojektien vaatimusmäärittelyssä on perusteltua huomioida vaatimusten selkeän dokumentoinnin ja vahvistamisen tärkeys sekä vaatimusmäärittelyn riittävä resursointi kokonaisuudessaan.

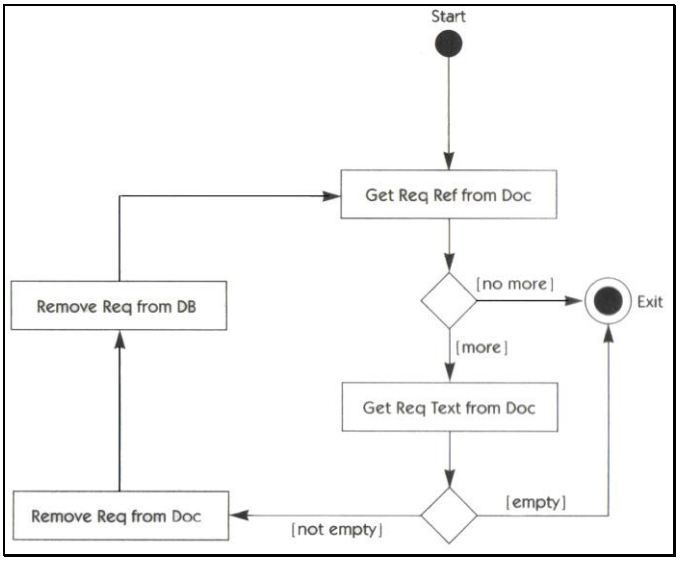
4.4.3 Vaatimusten kuvaaminen

Vaatimusten kuvaaminen on keskeinen osa vaatimusmäärittelyprosessia. Vaatimusten oikeellisuudella ja selkeydellä on merkittävä vaikutus suunnittelun, kehitystyön ja koko projektin onnistumiseen, minkä vuoksi vaatimusten kuvaamiseen ja täsmentämiseen on syytä käyttää riittävästi aikaa, vaivaa ja resursseja. Kuvausten täytyy olla mahdollisimman selviä ja yksiselitteisiä, jotta kaikki järjestelmän sidosryhmät voivat ymmärtää vaatimukset oikein. (Leffingwell ja Widrig, 2006; McConnell, 1998.) Jotta vaatimukset voitaisiin ymmärtää helposti, ne täytyy mallintaa käyttäen luonnollista kieltä ja kaavioita (Kotonya ja Sommerville, 1998, 33). Vaatimusten kuvaamiseen on kehitetty luonnollista kieltä täydentäviä visuaalisia menetelmiä, jotka ovat tarkoituksenmukaisia varsinkin silloin, kun kuvaus on liian monimutkainen luonnolliselle kielelle tai jos ei ole varaa siihen, että määrittely ymmärretään väärin (Leffingwell ja Widrig, 2006, 279). Leffingwell ja Widrig muistuttavat, että luonnollisen kielen epätarkkuus ei ole siedettävissä varsinkaan silloin, kun kuvataan erityisen tärkeitä tai monimutkaisia vaatimuksia. Toisaalta mikään yksittäinen kuvaus ei tarjoa täydellistä ymmärrystä vaatimuksista, vaan sen sijaan tarvitaan yhdistelmä kirjallisia ja visuaalisia kuvauksia, jotka yhdessä helpottavat kokonaiskuvan muodostamista suunnitellusta järjestelmästä (Wiegers, 2003, 193). Myös Wiegers perustelee visuaalisten kuvausten tärkeyttä sillä, että pelkästään luonnollisella kielellä kirjoitettuihin vaatimuskuvauksiin liittyy usein väärinymmärryksen riski. Wiegersin mukaan visuaalisten kuvausten etuna on, että ne kuvaavat vaatimuksia yksiselitteisemmin ja tehokkaammin kuin luonnollisella kielellä tehdyt kuvaukset. Kun verrataan eri henkilöiden luonnollisella kielellä kirjoittamia kuvauksia vaatimuksesta, paljastuu epäyhtenäisyyksiä ja ristiriitaisuuksia. Sen sijaan visuaalisilla menetelmillä voidaan yhdenmukaistaa vaatimusten kuvaamista ja edesauttaa niiden ymmärrettävyyttä.

Wiegers kuitenkin muistuttaa, että visuaaliset kuvaukset eivät korvaa sella kielellä laadittuja vaatimusmäärittelyjä, vaan täydentävät niitä. (Wiegers, 2003, 194.)

Vaatimusten kuvaamiseen ja havainnollistamiseen on kehitetty lukuisia edistyneitä tekniikkoja, joiden perusteellinen käsittely ei mahdu tähän tutkimuskokonaisuuteen. Sen vuoksi käsittelen vaatimusten kuvaamista ensisijaisesti luonnollisella kielellä laadittavan kirjallisen dokumentoinnin prosessina, jossa visuaalisilla kaavioilla ja pseudokoodilla on kuvausta täydentävä rooli. Luonnollisella kielellä laadittavan kuvauksen lisäksi vaatimuksia voidaan tarvittaessa havainnollistaa visuaalisesti esimerkiksi UML-kieleen perustuvilla toiminto- ja käyttötapauskaavioilla tai pseudokoodilla, jotka esittelen seuraavaksi lyhyesti.

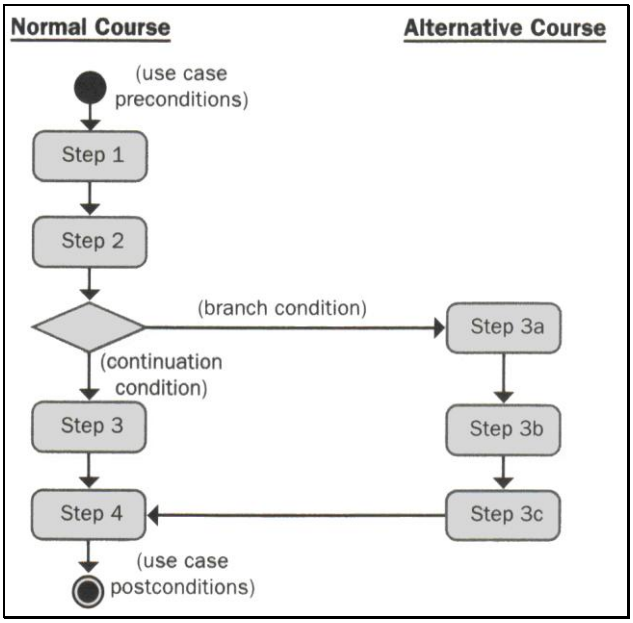
Unified Modeling Lanquage (UML) on graafinen kuvauskieli, joka soveltuu ohjelmistovaatimusten ja käyttötapausten visuaaliseen mallintamiseen ja dokumentointiin. UML sisältää valmiit kaaviot järjestelmän rakenteelliseen ja toiminnalliseen mallinnukseen ja varsinkin toiminnalliset kaaviot soveltuvat hyvin vaatimusmäärittelyn tarpeisiin. (Parviainen ym., 2003, 69.) Kuvassa 4-5 on esimerkki UML-kuvaukseen perustuvasta toimintokaavio.



Kuva 4-5: UML-toimintokaavio
Lähde: Leffingwell ja Widrig, 2006

UML-toimintokaavioiden etuna on, että ne ovat yleisesti sovellettuina jo laajasti tunnettuja ja tarjoavat vakiintuneen käytännön vaatimuskokonaisuuksien havainnollistamiseen. Toisaalta toimintokaavioiden ongelmana on jatkuva päivittämisen tarve, kun vaatimuskuvaus muuttuu projektin aikana. (Leffingwell ja Widrig, 2006, 284-285.) Vaati-

muksia voidaan havainnollistaa myös käyttötapausten kautta kuvan 4-6 la UML-kaaviolla.



Kuva 4-6: UML-käyttötapauskaavio mallintaa käyttötapauksen tavallista ja vaihtoehtoista tapahtumaketjua
Lähde: Wiegers, 2003

Pseudokoodi on kvasiohjelmointia, joka yhdistää luonnollisen kielen epämuodollisuuden ohjelmointikielen tarkkaan syntaksiin ja kontrollirakenteisiin (Leffingwell ja Widrig, 2006, 280). Pseudokoodi koostuu yhdistelmästä käskylauseita, joissa on toimintaa kuvaava verbi ja tekemisen kohde. Päätöksiä voidaan esittää ohjelmoinnista tutuilla muodollisilla if-else-endif-rakenteilla ja toistuvia toimintoja do-while- ja for-next-rakenteilla. Pseudokoodilla voidaan selkeyttää monimutkaisia ja virhealttiita vaatimuksia, joiden kuvaaminen on luonnollisella kielellä hankalaa. (Leffingwell ja Widrig, 2006, 280.) Kuvan 4-7 mukaisesti luonnollista kieltä ja ohjelmoinnin syntaksia hyödyntävän pseudokoodin etuna on, että sillä kuvatut vaatimukset ovat myös sellaisten henkilöiden ymmärrettävissä, joilla ei ole ohjelmointikokemusta.

The algorithm for calculating deferred-service revenue earned for any month is:

```
Set SUM(x)=0
FOR each customer X
   IF customer purchased paid support
      AND ((Current month) >= (2 months after ship date))
      AND ((Current month) <= (14 months after ship date))
   THEN Sum(X)=Sum(X) + (amount customer paid)/12
END
```

Kuva 4-7: Pseudokoodilla voidaan kuvata monimutkaisia ja virhealttiita vaatimuksia
Lähde: Leffingwell ja Widrig, 2006

5 Vaatimusten hallinta

5.1 Muutostekijät

Ohjelmistovaatimukset eivät aina ole vakaita ja kiinteitä, vaan saattavat muuttua erilaisten tekijöiden seurauksena. Gottesdiener (2005) mainitsee muutostarpeiden syiksi muun muassa vaatimusten hankintavaiheessa tapahtuneet virheet tai väärinkäsitykset, parantuneen ongelmien ymmärryksen, teknologisen kehityksen ja muuttuneet liiketoiminnalliset tai lainsäädännölliset vaatimukset (Gottesdiener, 2005, 281). Gottesdienerin mukaan muuttuvien vaatimusten vajavainen hallinta on usein syy projektien viivästymiselle ja resurssien tuhlaukselle. Myös McConnell (1998) sekä Leffingwell ja Widrig (2006) painottavat, että kyvyttömyys hallita vaatimusten lisäilyä luo paineita projektin aikataululle ja heikentää tuottavuutta.

Muutostarpeiden syyt voidaan jakaa sisäisiin ja ulkoisiin muutostekijöihin (Leffingwell ja Widrig, 2006, 337). Sisäisiin muutostekijöihin kuuluvat epäonnistunut vaatimusten kerääminen, toimivan vaatimustenhallintaprosessin puute ja se, että osa vaatimuksista saatetaan löytää vasta järjestelmän suunnitteluvaiheessa, jolloin järjestelmän vaatimukset voivat muuttua kokonaisuudessaan. Ulkoisilla muutostekijöillä Leffingwell ja Widrig viittaavat sellaisiin muutoslähteisiin, joihin projektiryhmällä on vain vähän tai ei ollenkaan mahdollisuuksia vaikuttaa. Ulkoiset muutoslähteet voivat ilmetä järjestelmän kannalta keskeisten perusongelmien muuttumisena, jolloin järjestelmän tavoitteena olevalta ongelmanratkaisulta katoaa pohja. Leffingwell ja Widrig mainitsevat ulkoisina muutoslähteinä teknologisen kehityksen, uusien järjestelmien käyttöönoton, taloudelliset muutokset ja lakien muuttumisen (Leffingwell ja Widrig, 2006, 338). Jotta vaatimusten muutostekijät voitaisiin huomioida ja muutoksiin reagoida hallitusti, ajoissa ja tehokkaasti, on projektille luotava keskitetty vaatimusten hallintaprosessi muutosten käsittelyyn.

5.2 Muutosten käsittely

Vaatimusten hallinta on prosessi, jonka tavoitteena on valvoa vaatimusten tilaa ja kontrolloida vaatimuksiin kohdistuvia muutoksia (Gottesdiener, 2005, 281). Koska muutosten pysäyttäminen täysin on harvoin projektin etujen mukaista, useimmat projektit joutuvat pohtimaan, kuinka muutoksia voisi hallita tehokkaimmin (McConnell, 1996, 338). McConnell painottaa, että vaatimusten muutoksiin liittyy projektin onnistumisen kannalta vakavia riskejä, minkä vuoksi muutostenhallinta on tärkeä osa projektisuunnittelua ja yksi projektin kriittisistä menestystekijöistä. Vaatimusten hallinta kattaa projektin koko elinkaaren alkaen vaatimuskehityksestä ja jatkuen läpi ohjelmistokehityksen.

Muuttuvien vaatimusten hallinnan edellytyksenä on sellaisten menettelytapojen ja sääntöjen soveltaminen, jotka tukevat sekä muutosten vaikutusten ymmärtämistä että muutosten käsittelyä. Muuttuvien vaatimusten vuoksi projektin sidosryhmien on myös neuvoteltava uudelleen vaatimuksiin liittyvistä sitoumuksista (Gottesdiener, 2005, 281). Gottesdienerin mukaan menettelytavoista ja säännöistä sopiminen on tärkeää siksi, että voitaisiin luoda tehokkaita toimintamalleja, jotka hallitusti sallivat muutokset vaatimusten pohjamäärittelyyn ja samalla minimoivat projektisuunnitelman keskeytykset ja mahdollistavat tehokkaimman mahdollisen tavan käyttää sidosryhmien aikaa muutosten arviointiin ja ratkaisuun (Gottesdiener, 2005, 282).

Wiegers (2003) kuvaa vaatimusten hallintaa prosessina, jonka tavoitteena on vaatimusten pohjamäärittelyyn kohdistuvien muutosten kontrollointi, vaatimusten pitäminen ajan tasalla projektisuunnitelmissa, yksittäisten vaatimusten ja vaatimusdokumenttien versioiden kontrollointi, vaatimusten tilan seuranta ja yksittäisten vaatimusten ja projektin osakokonaisuuksien välisten loogisten linkkien hallinta (Wiegers, 2003, 313). Wiegers jakaa vaatimustenhallinnan muutostenhallintaan, versionhallintaan, vaatimusten tilan seurantaan ja vaatimusten jäljitykseen. Muutostenhallinta pitää sisällään muutosehdotusten käsittelyn, muutosten vaikutusten analysoinnin, muutoksista päättämisen, vaatimusdokumenttien päivityksen ja epävakaiden vaatimusten kartoituksen. Versionhallinta taas on järjestelmäversioiden, vaatimusdokumenttien ja yksittäisten vaatimusten versioiden yksilöimistä. Vaatimusten tilan seurantaan kuuluu vaatimusten mahdollisten tilojen määrittely, tallennus ja raportointi. Vaatimusten jäljittäminen taas on linkkien määrittelyä muihin vaatimuksiin ja järjestelmän elementteihin. (Wiegers, 2003, 314.)

Jotta projektin vaatimusmuutosten hallinta olisi tehokasta, on tärkeää luoda toimintatavat muutosten ehdottamiseen, muutosten vaikutusten arviointiin, muutoksista päättämiseen, vaatimusdokumenttien päivittämiseen ja muutosten seurantaan (Gottesdiener, 2005, 282-283). Wiegersin (2003) tavoin Gottesdiener (2005) mainitsee vaatimusten versionhallinnan ja tilaraportoinnin tärkeinä muutostenvalvontamenetelminä. Gottesdiener muistuttaa myös sujuvan päätöksenteon tärkeydestä vaatimusten hallinnassa. Muutosten valvonnan roolit ja vastuut on tunnistettava, jotta vaatimusmuutoksia koskeva päätöksenteko toimisi ja olisi tehokasta. Gottesdienerin mukaan tämä voidaan parhaiten saavuttaa siten, että projektille perustetaan muutostoimikunta, joka käsittelee muutosehdotukset, arvioi niiden vaikutukset projektin kokonaisuuden kannalta, hyväksyy tai hylkää muutosehdotukset ja dokumentoi päätökset (Gottesdiener, 2005, 285). McConnellin (1998) mukaan muutostoimikunnan tehtävä on analysoida jokainen ehdotettu

muutos siltä kannalta, kuinka muutos vaikuttaa projektin aikatauluun, kustannuksiin ja ominaisuuksiin. Jos muutosehdotus tai ominaisuuspyyntö ei ole sen analysointiin kuluvan ajan arvoinen, se ei myöskään ole sen toteuttamiseen kuluvan ajan arvoinen ja muutostoimikunnan tulisi hylätä muutosehdotus saman tien (McConnell, 1996, 339). Gottesdiener kuitenkin muistuttaa, että myös hylätyt muutosvaatimukset tulisi ottaa ylös perusteluineen, sillä samoja muutoksia saatetaan vaatia myöhemmin (Gottesdiener, 2005, 284).

Gottesdiener painottaa, että muutostoimikunnassa olisi hyvä olla sekä toiminnallisia että teknisiä asiantuntijoita. Muutostoimikunnan jäsenten on pystyttävä punnitsemaan esimerkiksi vaatimusten muutosvaikutusten aiheuttamia kustannus- ja aikatauluvaikutuksia, vaatimusten priorisointia ja useiden muutosten koordinointia (Gottesdiener, 2005, 285). Gottesdiener muistuttaa, että tehokkaan työskentelyn turvaamiseksi muutostoimikunnalla on oltava vahva johtajuus ja hyvin suunnitellut ja toteutetut tapaamiset. Tämän lisäksi olisi varmistettava, että kaikki projektin sidosryhmät ymmärtävät, kuinka päätösprosessi toimii ja mikä on päätösten logiikka. McConnellin (1996) mukaan viralliset muutostoimikunnat torjuvat tehokkaasti hallitsematonta vaatimusten lisäilyä sekä suurissa että pienissä projekteissa. Samassa yhteydessä McConnell muistuttaa, että tarpeettomat tai harkitsemattomat vaatimukset lisäävät suhteellisesti järjestelmän kompleksisuutta ja ovat sekä kustannus- että aikatauluriski. Jotta muutosehdotus voitaisiin hyväksyä toimikunnassa, on muutoksen vaikutukset selvitettävä ja ymmärrettävä ennen päätösten tekoa ja muutosten toteuttamista (McConnell, 1996, 337). Kun kaikki muutosehdotukset kulkevat muutostoimikunnan käsittelyn kautta, valikoituvat toteutukseen vain sellaiset muutokset, joiden vaikutukset on arvioitu projektin kokonaisuuden kannalta. Näin voidaan vähentää muutoksista projektille aiheutuvia riskejä. McConnellin (1998) mukaan yksi muutostenhallinnan tavoitteista onkin varmistaa, että muutosten vaikutukset ymmärretään projektiin kokonaisvaltaisesti vaikuttavina tekijöinä (McConnell, 1998, 337).

Kun vaatimustenhallinnan toimintatavoista ja päätöksenteosta on sovittu, voidaan siirtyä vaatimusten pohjamäärittelyyn. Pohjamäärittelyn laatiminen on yksi onnistuneen vaatimusten hallinnan edellytyksistä. Pohjamäärittely kuvaa projektin vaatimukset kokonaisuudessaan sisältävää vaatimusjoukkoa, jossa kukin vaatimus on yksilöity. Pohjamäärittelyn perustana on alustava vaatimusmäärittelydokumentti, joka tarkastuksen jälkeen hyväksytään pohjamäärittelyksi ja alistetaan muutostenhallintaan (McConnell, 1998,

118). Kun vaatimusjoukko on hyväksytty pohjamäärittelyksi, voidaan ten muutoksia ja niiden vaikutusta arvioida suhteessa pohjamäärittelyyn.

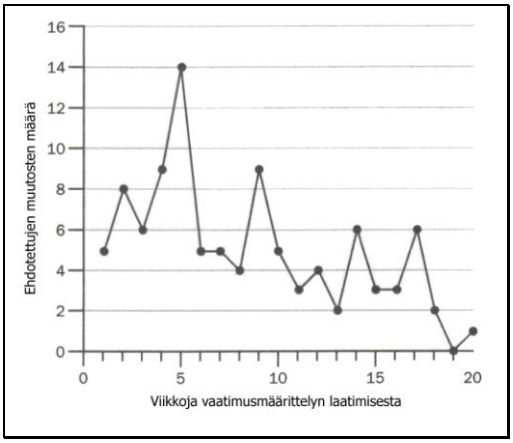
Keskitetty muutosten käsittely ja päätöksenteko yhdessä vaatimusten pohjamäärittelyn kanssa luovat pohjan vaatimusten muutostenhallintaprosessille. Nämä ovat tärkeitä tekijöitä varsinkin suurissa projekteissa, joissa haasteena ovat suuren vaatimusjoukon järjestäminen, vaatimusten priorisointi ja projektin resursointi (McConnell, 1996, 61). Projektin koko ja monimutkaisuus ovat tässä avaintekijöitä. McConnellin (1996) mukaan pienten ja vähemmän riskialttiiden projektien muutostenhallinta voidaan hoitaa ilman muodollista prosessia. Tällöinkin on kuitenkin tarpeellista määritellä, kellä on valta päättää vaatimusmuutoksista ja miten versionhallinta ja muutosten seuranta hoidetaan.

5.3 Muutosvaikutusten analysointi

Muutosvaikutusten analysointi (Impact analysis) on yksi vastuullisen vaatimustenhallinnan avaintekijöistä (Wiegers, 2003, 345). Tavoitteena on saavuttaa tarkka ymmärrys muutoksen vaikutuksista päätöksenteon tueksi ja arvioida muutoksen toteuttamiseen tarvittavia resursseja. Vaikutuksia analysoitaessa on syytä ottaa huomioon ainakin kustannusvaikutukset, järjestelmän toimivuus ja mahdolliset seuraukset projektin osapuolille, asiakkaille ja muille järjestelmän sidosryhmille. (Leffingwell ja Widrig, 2006, 346.) Muutosehdotuksen hyväksymisen edellytyksenä on muutoksen taustalla olevan logiikan perusteellinen selvitys ja ymmärtäminen. Pienilläkin muutoksilla voi olla odottamattomia seurauksia, minkä vuoksi muutosvaikutusten analysointi on ilmeisen tarpeellista kaikkien muutosehdotusten osalta. Wiegers (2003) painottaa, että muutosvaikutuksia analysoitaessa on keskeistä selvittää muutoksen aikataulu- ja kustannusvaikutukset sekä kartoittaa muutoksen toteutukseen sisältyvät työvaiheet ja työmäärä. Lisäksi tässä vaiheessa on myös paikallaan perustella muutoksen hyödyt ja haitat (Wiegers, 2003, 344).

5.4 Muutosaktiviteetin mittaaminen

Vaatimusten muutosaktiviteettia mittaamalla voidaan hankkia tietoa vaatimusprosessista ja projektin tilasta. Wiegersin (2003) mukaan muutosaktiviteetti kertoo pohjamäärittelyyn sisällytettyjen vaatimusten vakaudesta (Wiegers, 2003, 342). Kaavio 5-1 kuvaa projektin edetessä ilmenneiden muutosehdotusten määrää suhteessa vaatimusten pohjamäärittelyyn, joka muodostaa mittauksen lähtöpisteen (0,0).



Kuva 5-1: Kaavio vaatimusten muutosaktiviteetista pohjamäärittelyn jälkeen
Lähde: Wiegers, 2003

Vaatimusten pohjamäärittelyyn kohdistuvien muutosehdotusten määrä voi kasvaa etenkin projektin alussa. Syynä tähän ovat pohjamäärittelyyn sisällytettyjen vaatimusten mahdollinen epätäydellisyys ja kehittyminen. (Wiegers, 2003, 343.) Wiegers kuitenkin muistuttaa, että muutosehdotusten määrän tulisi kääntyä laskuun, kun projekti lähestyy valmistumispäiväänsä. Wiegersin mukaan muutosten pysyvästi korkea taajuus osoittaa riskin mahdollisesta aikataulullisten sitoumusten pitämättömyydestä ja kertoo tarpeesta parantaa vaatimusprosessin laatua vaatimusten keräämisen osalta (Wiegers, 2003, 343). Wiegersin mukaan muutosaktiviteettia ja vaatimusten vakautta analysoitaessa on tärkeää huomioida muutosehdotusten ja toteutettujen muutosten määrä vaatimuksittain, muutoslähteet ja muutosehdotusten käsittelyyn sekä toteutukseen kulunut aika. Muutoslähteiden jäljittäminen on erityisen tärkeää, sillä jatkuvat muutokset pohjamääritellyissä vaatimuksissa vaarantavat projektin aikataulun ja voivat lisätä kustannuksia. Keskeisten muutoslähteiden ymmärtäminen mahdollistaa pohjamäärittelyyn kohdistuvien muutospyyntöjen vähentämisen tulevaisuudessa.

5.5 Yhteenveto

Olen tässä kappaleessa käsitellyt vaatimustenhallintaa kokonaisuutena, jossa muutosten ehdottaminen, muutosten vaikutusten arviointi, muutoksista päättäminen, vaatimusdokumenttien päivittäminen ja muutosten seuranta kuuluvat saman keskitetysti hallitun prosessin alle. Olen keskittynyt menetelmiin, joista vaatimustenhallinnan prosessi koostuu, mutta rajannut vaatimustenhallinnan ohjelmistotyökalut tarkastelun ulkopuolelle.

6 Vaatimusprosessin arviointitarpeet RVL:ssä

Käsittelen seuraavaksi vaatimuskehitystä Rajavartiolaitoksen tietojärjestelmäprojektien osalta. Pyrkimyksenä on kartoittaa vaatimusprosessin ja sitä tukevien menetelmien kus-

tannustehokkuuteen vaikuttavia tekijöitä erityisesti vaatimusten keräämisen osalta ja analysoida, mitä vaatimusmenetelmien tehokas ja taloudellinen soveltaminen edellyttää.

6.1 Haasteet ja tavoitteet

Vaatimusprosessin ja siihen sisältyvien menetelmien soveltamiseen liittyy erilaisia haasteita, joihin projektiryhmän on löydettävä ratkaisu. Perusedellytyksenä on ymmärtää syvällisesti ongelmat, joihin rakennettavan järjestelmän odotetaan tarjoavan ratkaisuja, minkä jälkeen voidaan siirtyä keräämään vaatimusehdotuksia. Vaatimusehdotukset on pyrittävä keräämään riittävän kattavalta joukolta sidosryhmiä, jotta saadaan luotettava kuva Rajavartiolaitoksen tarpeista kokonaisuudessaan. Tämän jälkeen vaatimusehdotukset analysoidaan ja pyritään tunnistamaan niistä oleellisimmat ja keskeisesti järjestelmän toteutukseen vaikuttavat vaatimukset. Ongelmien ymmärryksen ja kerättyjen vaatimusehdotusten pohjalta on kyettävä määrittelemään vaatimukset, jotka rakennettavan järjestelmän on täytettävä. Määrittelyvaiheessa onnistuminen on ratkaisevan tärkeää, jotta vaatimukset vastaisivat Rajavartiolaitoksen tarpeita ja ohjaisivat järjestelmän kehitystyötä oikeaan suuntaan. Koska vaatimuksilla on ratkaiseva rooli tietojärjestelmäprojektin toteutuksessa, niiden tilaa ja muutoksia on tärkeää seurata. Toisin sanoen, sen jälkeen kun vaatimukset on määritelty, hyväksytty ja vahvistettu, tarvitaan muutostenhallintaa. Jotta vaatimusten tilaa voitaisiin hallita ja tarvittaessa muuttaa, on vaatimuksille luotava peruslinja. Peruslinja pitää sisällään määrittelyn jälkeisen vaatimusten lähtötilanteen, johon kohdistuvien muutosehdotusten vaikutukset arvioidaan kokonaisvaltaisesti projektiin vaikuttavina tekijöinä. Arvioinnin suorittaa projektin muutostoimikunta, johon kuuluu projektissa työskenteleviä operatiivisia ja teknisiä asiantuntijoita. Muutosehdotukset käyvät läpi muutostoimikunnan hyväksymismenettelyn, jolloin peruslinjaan kohdistuvat muutokset tapahtuvat aina hallitusti. Ohjattu vaatimusten muutosprosessi on tärkeä projektin onnistumiseen vaikuttava tekijä, sillä hallitsemattomiin vaatimusmuutoksiin liittyy merkittäviä taloudellisia ja aikataulullisia riskejä.

Nämä yllä kuvatut haasteet liittyvät vaatimusprosessiin kokonaisuudessaan. Olen tarkastellut tutkimuksessani kaikkia vaatimusprosessin osa-alueita, sillä tavoitteena on ollut selvittää, mitä vaatimusprosessin kokonaisvaltaisesti tehokas ja taloudellinen soveltaminen edellyttää Rajavartiolaitoksen tietojärjestelmäprojekteilta. Vaikka kustannustehokkuuden tavoite koskee vaatimusprosessia kokonaisuudessaan, keskityn seuraavaksi analysoimaan vaatimusten keräämisen kustannustehokkuuteen vaikuttavia tekijöitä, joiden kartoittamiseen erityisesti on tarvetta Rajavartiolaitoksessa. Analyysin tavoit-

teena on selvittää, mistä vaatimusten keräämisen kustannukset muodostuvat konaisuudessaan ja mitä tulisi ottaa huomioon, jotta vaatimukset saataisiin kerättyä luotettavasti ja kustannustehokkaasti.

6.2 Arviointikriteerit

Analysoin luvussa 7 käyttäjien haastattelun ja tarkkailun, prototypoinnin, käyttöskenaarioiden, ryhmätekniikoiden ja vaatimusten uudelleenkäytön soveltuvuutta RVL:n tietojärjestelmäprojektien vaatimusten keräämiseen. Arviointikriteereinä toimivat asiakaslähtöisyys, tekninen vaativuus ja työmäärä sekä kustannustekijät, jotka esittelen seuraavaksi.

6.2.1 Asiakaslähtöisyys

Vaatimusten keräämisen haasteena on saada käyttäjät ymmärtämään, mitä he oikein tahtovat, ja kerätä todelliset vaatimukset. Totesin kappaleessa 4.2, että asiakaslähtöinen vaatimusten kerääminen on perusteltua myös Rajavartiolaitoksen tietojärjestelmäprojekteissa. Tämän olettamuksen tärkeimmät perustelut ovat, että asiakaslähtöisyyden ansiosta vaatimukset vastaavat käyttäjien todellisia tarpeita ja lisäksi asiakkaan aktiivinen osallistuminen voi parantaa järjestelmäkehityksen nopeutta. Ilman asiakkaan osallistumista kerätyt vaatimukset edustavat vain osaa todellisista vaatimuksista. Asiakaslähtöisyys mahdollistaa vaatimuksiin liittyvän yhteisymmärryksen loppukäyttäjien ja projektiryhmän välillä ja samalla kasvattaa todellisten vaatimusten osuutta kerätyistä vaatimuksista. Järjestelmäkehityksen nopeutuminen taas perustuu siihen, että asiakkaan osallistumisen myötä projektiryhmä voi keskittyä todellisiin vaatimuksiin eikä aikaa kulu epäoleellisten vaatimusten selvittelyyn. Asiakaslähtöinen lähestymistapa tarkoittaa tässä yhteydessä asiakkaan jatkuvaa osallistumista vaatimusten kartoitukseen ja vaatimusprosessiin kokonaisuudessaan. Useat lähteet korostavat asiakaslähtöisen vaatimusten keräämisen merkitystä projektin kriittisenä menestystekijänä (mm. Hollander ja Mirlocca, 2005; Wiegers, 2005; McConnell, 1998; Komi-Sirviö, 2004). Komi-Sirviön mukaan erityisesti ohjelmistojen räätälöinnin lisääntyminen asettaa uusia haasteita vaatimusprosessille ja edellyttää aiempaa asiakaslähtöisempää lähestymistapaa, jotta asetetut tavoitteet voitaisiin saavuttaa uhraamatta ohjelmiston laatua (Komi-Sirviö, 2004). Asiakaslähtöisyys tehostaa vaatimusten keräämistä ja parantaa prosessin luotettavuutta. Nämä ovat merkittäviä tekijöitä haettaessa optimaalista vaatimusten keräysmenetelmää, joka on samalla tehokas, luotettava ja kustannuksiltaan kohtuullinen.

6.2.2 Tekninen vaativuus ja työmäärä

Menetelmät poikkeavat toisistaan niin teknisen vaativuutensa kuin työmääränsä suhteen. Työmäärä on yksi tekijä arvioitaessa vaatimuskartoitukseen kehitettyjen menetelmien soveltuvuutta RVL:n tietojärjestelmäprojektien vaatimusten keräämiseen. Menetelmän työmäärää arvioitaessa on otettava huomioon menetelmän tekninen vaativuus, eli onko menetelmä sovellettavissa ilman valmistelevaa taustatyötä vai edellyttääkö menetelmän soveltaminen projektiryhmältä etukäteissuunnittelua ja perehtymistä organisaation prosesseihin tai jopa prototyypin luomista rakennettavasta järjestelmästä. Menetelmää valittaessa on syytä huomioida myös projektin laajuus ja käytettävissä olevat aika ja resurssit. Riippumatta siitä, mitä menetelmää vaatimusten keräämiseen käytetään, laadukas vaatimuskartoitus edellyttää projektiryhmältä merkittävää työpanosta, osaamista ja resursseja.

6.2.3 Kustannukset

Menetelmän kustannuksia arvioitaessa on otettava huomioon sekä menetelmän soveltamisesta aiheutuvat välittömät kustannukset, että kustannukset koko järjestelmäprojektin kannalta. Välittömät kustannukset koostuvat esimerkiksi menetelmän vaatimista henkilö- ja työaikaresursseista. Ei kuitenkaan riitä, että keskitytään pelkästään välittömiin kustannuksiin, sillä menetelmän kustannukset koko projektin osalta riippuvat vaatimuskartoituksen lopputuloksesta. Hyvä menetelmä on kokonaistaloudellinen, luotettava ja tukee oikeiden vaatimusten löytymistä. Jos menetelmän välittömät kustannukset ovat vähäiset ja lopputulos hyvä, ollaan ihannetilanteessa. Kuitenkin on varauduttava myös siihen, että menetelmällä voi olla vähäiset välittömät kustannukset, mutta lopputulos on huono, jolloin määrittelyn muuttaminen ja korjaaminen myöhemmin projektin aikana nostavat menetelmän kustannuksia huomattavasti. Tällöin kokonaistaloudellisesti edullisempi vaihtoehto voisi olla jopa sellainen menetelmä, jossa välittömät kustannukset ovat korkeat mutta lopputulos hyvä. Tämän analyysin kannalta kiinnostavia menetelmiä ovat sellaiset, jotka mahdollistavat kustannuksiinsa nähden laadukkaan lopputuloksen. Laadukas lopputulos edellyttää menetelmältä luotettavuutta ja tehokkuutta. Luotettavuus on sitä, että menetelmä tukee oikeiden vaatimusten löytymistä riittävän perusteellisella ja asiakkaan tarpeet huomioivalla tarpeiden kartoituksella ja mahdollistaa asiakkaan jatkuvan osallistumisen vaatimuskehitykseen. Tehokkuuden mittarina taas on se, kuinka laadukas vaatimuskartoitus menetelmällä voidaan tuottaa suhteessa menetelmän kustannuksiin.

6.3 Analyysin perusteet

Esittelin luvussa 4 vaatimusten keräämiseen soveltuvia menetelmiä, joita nimitän tässä vaatimusmenetelmiksi. Pyrin nyt analysoimaan näitä menetelmiä sen perusteella, kuinka ne tukevat vaatimuskartoituksen asiakaslähtöistä lähestymistapaa, paljonko menetelmän soveltaminen vaatii projektiryhmältä työtä ja mikä on menetelmän tekninen vaativuus. Tämän analyysin pohjalta pyrin kartoittamaan vaatimusmenetelmien välittömiin kustannuksiin vaikuttavia kustannustekijöitä. Asiakaslähtöisyys on tärkeä vaatimuskartoituksen luotettavuutta ja tehokkuutta parantava tekijä, minkä vuoksi olen ottanut sen yhdeksi arviointikriteeriksi. Asiakaslähtöisyydellä saavutettava luotettavuus ja tehokkuus vaikuttavat myös myönteisesti menetelmän kustannustehokkuuteen. Vaadittu työmäärä taas on arviointikriteerinä siksi, että se on tärkeä tekijä arvioitaessa menetelmän soveltuvuutta kyseessä olevaan projektiin. Käytännössä vaatimuskartoituksen työmäärä koostuu sekä itse valitun menetelmän työmäärästä, että projektissa käsiteltävän vaatimustiedon määrästä, joka on riippuvainen projektin laajuudesta. Keskityn tässä kuitenkin itse menetelmän soveltamisen vaatimaan työmäärään, joka muodostuu tiedon keräämisestä ja analysoinnista sekä tarvittavasta esivalmistelusta ja taustatyöstä. Vaatimusmenetelmän soveltamisen kannalta on paikallaan kysyä, paljonko vaaditaan taustatyötä ja valmistelua, jotta päästään keräämään varsinaista vaatimustietoa. Tarvittavan taustatyön määrä ja laatu vaihtelevat menetelmittäin ja riippuvat valitun menetelmän teknisestä vaativuudesta, minkä olen myös ottanut yhdeksi arviointikriteeriksi. Teknistä vaativuutta arvioitaessa on huomioitava, mitä työvaiheita menetelmään sisältyy ja millaista osaamista ne edellyttävät. Työvaiheita voivat olla esimerkiksi organisaation nykyisiin järjestelmiin ja prosesseihin perehtyminen, käyttäjien haastattelu tai tarkkailu, käyttötilanteen simulointi, järjestelmäprototyypin suunnittelu, luonti ja kehittäminen sekä kerätyn vaatimustiedon analysointi.

Vaatimusmenetelmien arviointi perustuu oletukseen, että kutakin menetelmää sovelletaan kerran, ellei kyseessä ole lähtökohtaisesti iteratiivinen menetelmä, jota tyypillisesti sovelletaan useita kertoja peräkkäin tai jatkuvana prosessina. Lisäksi arviointi perustuu kunkin tekniikan soveltamiseen erikseen, vaikka tekniikkoja voidaan soveltaa myös rinnakkain, jolloin ne täydentävät toisiaan ja parantavat vaatimuskartoituksen lopputulosta.

7 Vaatimusmenetelmien analysointi

7.1 Asiakaslähtöisyys

7.1.1 Käyttäjien haastattelu ja tarkkailu

Haastattelu ei tue asiakaslähtöistä vaatimusten löytymistä parhaalla mahdollisella tavalla. Näin on varsinkin silloin, jos asiakkaan osallistuminen tapahtuu projektiryhmän ehdoilla ja aloitteesta. Tällöin vuorovaikutteisuus haastateltavan käyttäjän ja projektiryhmän välillä voi olla vähäistä. Tämä heikentää menetelmän luotettavuutta ja lopputuloksen laatua. Menetelmän heikkoon asiakaslähtöisyyteen myötävaikuttaa osaltaan se, että haastattelun tavoitteet eivät välttämättä ole haastateltavan loppukäyttäjän tiedossa, jolloin hänen tietämyksestään saadaan kerättyä vain se osa, joka ymmärretään kysyä ja jonka kysymykset kattavat. Tällöin projektiryhmän haasteena on osata kysyä ongelman ratkaisun kannalta oikeat kysymykset. Näistä haastattelun heikkouksista voi aiheutua lisäkustannuksia, jos osa vaatimuksista jää löytymättä tai ymmärretään väärin.

Käyttäjien tarkkailu mahdollistaa vaatimusten tarkastelun käyttäjän toiminnan näkökulmasta, jolloin käyttäjän ja projektiryhmän välinen kommunikointi on kahdensuuntaista ja vuorovaikutteista, mikä tukee asiakaslähtöisyyttä. Näin vaatimuskartoituksen asiakaslähtöisyys toteutuu tarkkailussa hiukan paremmin kuin esimerkiksi haastattelussa. Tarkkailu voidaan toteuttaa joko valmiissa järjestelmässä tai kuvitteellisessa käyttötilanteessa.

7.1.2 Prototypointi ja käyttöskenaariot

Prototypointi tukee asiakaslähtöistä vaatimusten kartoitusta, mikä parantaa menetelmän luotettavuutta ja lopputuloksen laatua. Loppukäyttäjän ja projektiryhmän välinen kommunikointi on kahdensuuntaista ja vuorovaikutteista. Prototypointi parantaa käyttäjien tietoisuutta vaatimuksista, sillä prototyypin jalostus etenee heidän palautteensa pohjalta. Tämän ansiosta käyttäjät ja projektiryhmä voivat päästä vaatimuksista yhteisymmärrykseen. Näin voidaan myös minimoida riskejä, jotka liittyvät myöhäisessä projektin vaiheessa muuttuviin käyttäjävaatimuksiin.

Käyttöskenaariot-menetelmä on asiakaslähtöinen ja mahdollistaa loppukäyttäjän osallistumisen todellisten käyttäjän ja järjestelmän välisten vuorovaikutustilanteiden kartoittamiseen. Asiakaslähtöisyyttä vahvistaa myös se, että loppukäyttäjän simuloidessa järjestelmän käyttöä, hänen kommenttinsa, ongelmansa ja ehdotuksensa otetaan ylös ja hänen toimintaansa seurataan kunkin skenaarion osalta. Menetelmä mahdollistaa vaatimusten tarkastelun käyttäjän toiminnan näkökulmasta, jolloin käyttäjän ja projektiryh-

män välinen kommunikointi on kahdensuuntaista ja vuorovaikutteista. Käyttöskenaariot kuvaavat käyttäjän ja järjestelmän välisiä vuorovaikutustilanteita, joita tarkkailemalla voidaan tunnistaa tehtävät, joihin käyttäjä tarvitsee järjestelmää. Menetelmä parantaa vaatimusten löytymistä, sillä koehenkilön toimintaa, kommentteja ja ongelmia analysoimalla voidaan tunnistaa sellaisiakin vaatimuksia, jotka eivät kävisi ilmi suorassa haastattelussa.

7.1.3 Ryhmätekniikat ja vaatimusten uudelleenkäyttö

Ryhmätyöskentely mahdollistaa kahdensuuntaisen ja vuorovaikutteisen viestinnän loppukäyttäjien ja projektiryhmän välillä, mikä tukee asiakaslähtöistä vaatimusten keräämistä. Ryhmätekniikat perustuvat ajatukselle, että ryhmä ihmisiä voi saavuttaa paremman ymmärryksen vaatimuksistaan kuin jos he työskentelisivät yksilöinä. Ryhmätyöskentelyn tavoitteena on edesauttaa projektiryhmän ja loppukäyttäjien välistä yhteisymmärrystä vaatimuksista, tukea järjestelmän sidosryhmien kommunikointia ja saada sidosryhmät mukaan järjestelmäkehitykseen. Asiakaslähtöisyyttä vahvistaa, että projektiryhmä ja loppukäyttäjät voivat yhdessä tunnistaa nykyisen tilanteen ja halutun tavoitetilan sekä laatia etenemissuunnitelman tavoitteen saavuttamiseksi.

Vaatimusten uudelleenkäytössä asiakaslähtöisyys voi toteutua siten, että ehdotus uudelleenkäytettävistä vaatimuksista tulee loppukäyttäjiltä tai joltain muulta järjestelmähankkeen sidosryhmältä. Vaatimusten uudelleenkäyttö mahdollistaa kehitystyön tuottavuuden parantamisen rajoittamalla vaatimusten keräämiseen kuluvaa aikaa ja resursseja sekä vähentämällä prosessiin liittyviä riskejä. Myös projektiryhmä voi tehdä aloitteen vaatimusten uudelleenkäytöstä ja esitellä ehdotuksensa loppukäyttäjille. Näin asiakkaan ja projektiryhmän välinen kommunikointi voi olla kahdensuuntaista ja vuorovaikutteista. Tavoitteena on tunnistaa sellaiset vaatimuskuvaukset, joita voidaan käyttää uudelleen joko kokonaan tai osittain minimaalisin muutoksin. Vaatimusten uudelleenkäyttö järjestelmien välillä on mahdollista esimerkiksi tilanteissa, joissa vaatimus liittyy selkeästi samaan sovellusalueeseen tai tiedon esittämisen tapa on sama. Parhaassa tapauksessa uudelleenkäytettäväksi ehdotetut vaatimukset ovat sidosryhmille ennestään tuttuja ja niistä voidaan päästä helpommin yhteisymmärrykseen, kuin jos olisi kyse täysin uusien vaatimusten hyväksymisestä.

Taulukossa 7-1 vaatimusmenetelmät on pisteytetty sen mukaan, kuinka hyvin ne tukevat asiakaslähtöistä lähestymistapaa. Asiakaslähtöisyys on arvioitu asiakkaan ja projektiryhmän välisen vuorovaikutteisuuden ja asiakkaan osallistumisen perusteella.

TEKNIIKKA	Asiakaslähtöisyys
Käyttäjien haastattelu (H)	x
Käyttäjien tarkkailu (T)	xx
Prototypointi (P)	xxx
Käyttöskenaariot (S)	xx
Ryhmätekniikat (R)	xx
Vaatimusten uudelleenkäyttö (U)	xx

x = Vuorovaikutteisuus on vähäistä. Asiakkaan osallistuminen tapahtuu projektiryhmän ehdoilla ja aloitteesta.
xx= Asiakkaan ja projektiryhmän välinen kommunikointi on kahdensuuntaista ja vuorovaikutteista.
xxx= Asiakkaan ja projektiryhmän välinen kommunikointi on kahdensuuntaista ja vuorovaikutteista. Menetelmän soveltaminen edellyttää asiakkaan jatkuvaa osallistumista.

Taulukko 7-1: Asiakaslähtöisyyden tuki

7.2 Tekninen vaativuus ja työmäärä

7.2.1 Käyttäjien haastattelu ja tarkkailu

Haastattelu edellyttää ongelman ratkaisun kannalta oleellisten kysymysten laatimista, haastattelutekniikan hallintaa ja kerätyn vaatimustiedon analysointia. Menetelmän työmäärä koostuu kysymysten laatimisesta, haastattelutilanteen järjestämisestä ja kerättyjen vaatimusehdotusten analysoinnista. Haastattelu on teknisen vaativuutensa ja työmääränsä perusteella toteutettavissa vähäisillä henkilöresursseilla varsinkin pienissä projekteissa.

Käyttäjien tarkkailu edellyttää projektiryhmältä taustatyönä tarkkailtavan käyttötilanteen etukäteissuunnittelua ja toteuttamista sekä perehtymistä käyttötilanteen taustalla vaikuttaviin organisaation prosesseihin. Menetelmän työmäärä koostuu tarkkailutilanteen suunnittelusta, toteutuksesta ja seurannasta sekä käyttäjien havaintojen tallentamisesta ja kerätyn tiedon analysoinnista. Tarkkailu on teknisen vaativuutensa ja työmääränsä perusteella toteutettavissa kohtuullisen vähäisillä henkilöresursseilla varsinkin pienissä projekteissa.

7.2.2 Prototypointi ja käyttöskenaariot

Prototypointi edellyttää projektiryhmältä menetelmän tuntemusta, sovelluskehitysosaamista ja perehtymistä organisaation prosesseihin. Menetelmän työmäärä koostuu prototyypin suunnittelusta, rakentamisesta, testauksesta ja kehittämisestä. Järjestelmäprototyypin rakentaminen on teknisesti haastavaa ja työlästä. Teknisen vaativuutensa ja työmääränsä perusteella prototypointi edellyttää huomattavasti osaamista ja henkilöresursseja myös pienissä projekteissa.

Käyttöskenaarioiden avulla toteutettava vaatimuskartoitus edellyttää projektiryhmältä taustatyönä testattavien skenaarioiden määrittelyä ja luontia sekä perehtymistä skenaarioiden taustalla vaikuttaviin organisaation prosesseihin. Menetelmän työmäärä koostuu skenaarioiden määrittelyn ja luonnin lisäksi testitilanteen seurannasta ja käyttäjien havaintojen tallentamisesta sekä analysoinnista. Skenaariot ovat teknisen vaativuutensa ja

työmääränsä perusteella toteutettavissa kohtuullisen vähäisillä
la varsinkin pienissä projekteissa.

7.2.3 Ryhmätekniikat ja vaatimusten uudelleenkäyttö

Ryhmätekniikoiden avulla toteutettava vaatimuskartoitus edellyttää projektiryhmältä perehtymistä ratkaistavaan ongelmaan ja sen taustalla vaikuttaviin organisaation prosesseihin, ryhmäistunnon suunnittelua, toteutusta ja ohjausta sekä istunnossa kerätyn tiedon tallennusta ja analysointia. Ryhmätekniikoilla voidaan kerätä paljon tietoa vaatimusten määrittelyn tueksi, mutta tekniikoiden soveltaminen edellyttää projektiryhmältä kykyä organisoida ryhmätyöskentely tehokkaasti yhteisen päämäärän löytämiseksi. Haasteena on löytää ja tunnistaa suuresta tietomäärästä oleellisimmat vaatimuksiin vaikuttavat tekijät. Kerättyä tietoa voi olla tarpeen myös syventää ja tarkentaa joko järjestämällä uusi ryhmätapaaminen tai käyttämällä muita menetelmiä. Pienissä projekteissa ryhmätekniikat ovat teknisen vaativuutensa ja työmääränsä perusteella sovellettavissa kohtuullisen vähäisillä henkilöresursseilla. Kuitenkin tarvittavaa työmäärää, resursseja ja kustannuksia arvioitaessa on syytä huomioida, että ryhmäistunto vaatii projektiryhmän lisäksi työaikaa myös muilta istuntoon osallistuvilta.

Vaatimusten uudelleenkäyttäminen edellyttää projektiryhmältä jo olemassa olevan ja kopioinnin kohteena olevan järjestelmän vaatimuksiin perehtymistä ja niiden ymmärtämistä sekä kykyä sijoittaa uudelleenkäytettävät vaatimukset osaksi uuden järjestelmän vaatimuskokonaisuutta. Kopioitavan vaatimuksen syvällinen ymmärtäminen voi edellyttää myös järjestelmä- ja sovelluskehitysosaamista. Teknisen vaativuutensa ja työmääränsä perusteella menetelmää on mahdollista soveltaa kohtuullisen vähäisillä henkilöresursseilla varsinkin pienissä projekteissa.

Taulukossa 7-2 vaatimusmenetelmät on pisteytetty niiden teknisen vaativuuden mukaan. Tekninen vaativuus on arvioitu sen perusteella, vaatiiko menetelmä käyttötilanteen simulointia ja/tai sovelluskehitysosaamista, vai onko se sovellettavissa ilman niitä.

TEKNIIKKA	Tekninen vaativuus
Käyttäjien haastattelu (H)	x
Käyttäjien tarkkailu (T)	xx
Prototypointi (P)	xxx
Käyttöskenaariot (S)	xx
Ryhmätekniikat (R)	x
Vaatimusten uudelleenkäyttö (U)	x

x = Menetelmän soveltaminen ei välttämättä edellytä käyttötilanteen simulointia tai sovelluskehitysosaamista
xx= Menetelmän soveltaminen edellyttää todellisen tai simuloidun käyttötilanteen järjestämistä
xxx= Menetelmä edellyttää sekä käyttötilanteen järjestämistä että sovelluskehitysosaamista

Taulukko 7-2: Tekninen vaativuus

Taulukossa 7-3 vaatimusmenetelmät on pisteytetty niiden työmäärän mukaan. Työmäärä on arvioitu sen perusteella, onko kyse välittömästi tapahtuvasta tiedon keräämisestä ja analysoinnista, vai vaaditaanko taustatyönä käyttötilanteen, non tai järjestelmäprototyypin toteutusta.

TEKNIIKKA	Työmäärä
Käyttäjien haastattelu (H)	x
Käyttäjien tarkkailu (T)	xx
Prototypointi (P)	xxx
Käyttöskenaariot (S)	xx
Ryhmätekniikat (R)	xx
Vaatimusten uudelleenkäyttö (U)	x

x = Menetelmän soveltaminen edellyttää vähimmillään tiedon keräämistä ja analysointia.
xx= Menetelmän soveltaminen edellyttää käyttötilanteen tai ryhmäistunnon suunnittelua ja toteutusta sekä tiedon keräämistä ja analysointia.
xxx= Menetelmän soveltaminen edellyttää järjestelmäprototyypin suunnittelua, rakentamista ja kehittämistä sekä tiedon keräämistä ja analysointia.

Taulukko 7-3: Työmäärä

7.3 Kustannustekijöiden arviointi

Mahdollisia kustannuksia arvioitaessa on syytä huomioida menetelmän tekninen vaativuus ja työmäärä. Tekninen vaativuus vaihtelee menetelmittäin sen mukaan, edellyttääkö vaatimustiedon keruu todellisen tai simuloidun käyttötilanteen järjestämistä, sovelluskehitysosaamista vaativaa järjestelmäprototyypin kehittämistä, vai riittääkö itse menetelmän teoreettinen osaaminen. Työmäärä taas koostuu vähimmillään vaatimustiedon keräämisestä ja analysoinnista, minkä lisäksi edellytyksenä voi olla käyttötilanteen tai ryhmäistunnon toteutus ja joskus myös järjestelmäprototyypin suunnittelu ja kehitystyö. Menetelmien kokonaisvaltainen testaamiseen perustuva hintavertailu ei ole tämän tutkimuksen laajuudessa mahdollista, mutta asiakaslähtöisyyden, teknisen vaativuuden ja työmäärän perusteella on mahdollista saada suuntaa antava yleiskuva siitä, mitä tekijöitä on syytä ottaa huomioon etsittäessä kustannustehokasta tapaa kerätä vaatimustietoa.

Siteerasin luvussa 4 tutkimusta, jonka mukaan asiakaslähtöinen toimintatapa parantaa vaatimuskartoituksen luotettavuutta ja vähentää myöhäisessä projektin vaiheessa ilmenevien kalliiden vaatimusmuutosten riskiä. Tämän perusteella asiakaslähtöisyys on arvo, joka kannattaa huomioida menetelmää valittaessa. Kuitenkin asiakaslähtöisyyden lisääntyessä myös menetelmän tekninen vaativuus lisääntyy. Jotta loppukäyttäjän toimintaan ja havaintoihin perustuva tieto voitaisiin tehokkaasti hyödyntää vaatimuskehityksessä, on menetelmän mahdollistettava joko yksittäisen käyttötilanteen mallinnus tai rakennettavan järjestelmän laajamittainen prototypointi. Tämä on oletettavasti kalliimpaa ja vaatii projektiryhmältä enemmän aikaa, työtä ja teknistä osaamista kuin käyttäjien haastattelu tai jo valmiin järjestelmän vaatimusten uudelleenkäyttö. Menetelmää valittaessa on ennen kaikkea syytä huomioida projektin laajuus ja käytettävissä olevat ai-

ka, resurssit ja osaaminen. Kaikki menetelmät eivät sovellu pieniin projekteihin teknisen vaativuutensa vuoksi, sillä tarvittavan erityisosaamisen hankkiminen voi nostaa projektin kustannuksia kohtuuttomasti. Myös projektin tiukka aikataulu voi rajoittaa sellaisten menetelmien soveltamista, jotka perustuvat järjestelmän mallinnukseen tai prototypointiin.

8 Yhteenveto

8.1 Johtopäätökset

Onnistunut vaatimusmenetelmien soveltaminen edesauttaa luotettavan ja kustannustehokkaan vaatimusprosessin toteutumista. Kokonaisuudessaan kustannustehokas vaatimusprosessi edellyttää onnistumista ongelman määrittelyssä, vaatimusten keräämisessä ja analysoinnissa sekä vaatimusten määrittelyssä ja hallinnassa. Näiden edellytysten täyttämiseksi on tärkeää valita oikeat ja juuri kyseiseen projektiin soveltuvat vaatimusmenetelmät. Itse olen tutkielmassani keskittynyt analysoimaan vaatimusprosessin kustannustehokkuutta erityisesti vaatimusten keräämisen osalta. Se, millaista menetelmää vaatimusten keräämiseen on järkevää käyttää, riippuu pitkälti projektin laajuudesta ja resursseista. Onnistuneen vaatimuskartoituksen perusedellytyksenä on, että valittu menetelmä soveltuu hyvin kyseessä olevaan projektiin. Jotta projektiryhmä voisi valita projektin tarpeet ja rajoitukset optimaalisesti huomioivan vaatimusmenetelmän, on sillä oltava asiantuntemusta eri menetelmistä ja kyky nähdä niiden mahdollisuudet ja rajoitukset projektin näkökulmasta. Vaatimusmenetelmän mahdollisia hyötyjä on punnittava suhteessa kustannustekijöihin. Tässä yhteydessä on syytä arvioida projektikohtaisesti, miten tarkasteltava menetelmä tukee asetettuja tavoitteita ja onko menetelmän hyödyntäminen mahdollista ja järkevää juuri kyseisessä projektissa, vai saavutettaisiinko jollain muulla menetelmällä mahdollisesti parempi lopputulos.

8.2 Jatkotutkimusaiheita

Aiheen jatkotutkimuksen kannalta olisi mielenkiintoista tehdä pieneen koeluontoiseen ohjelmistoprojektiin sidottu case-tutkimus siitä, millaisia eroja tässä tutkimuksessa käsitellyillä vaatimusmenetelmillä kerätyssä tiedossa on ja miten kerätty tieto kuvaa asiakkaan todellisia tarpeita. Samalla voisi hakea vastausta siihen, millainen yhteys kullakin menetelmällä on vaatimusten hallintavaiheessa ilmenevään vaatimusten lisäily- ja muutostarpeeseen. Toinen mahdollinen jatkotutkimuksen aihe olisi verrata tässä tutkimuksessa käsiteltyjä perinteisiä vaatimusmenetelmiä nopean sovelluskehittämisen Rapid Application Development (RAD)-menetelmiin vaatimusten toteutumisen osalta. Eri menetelmillä saatuja lopputuloksia vertailemalla voitaisiin arvioida, onko perinteisen

vaatimusprosessin noudattamisesta järkevää tinkiä RAD-menetelmien lupaaman nopeamman projektiaikataulun toivossa.

LÄHDELUETTELO

Beyer, Hugh, **Holtzblatt**, Karen (1998), "Contextual Design: Defining Customer-Centered Systems", Morgan Kaufmann Publishers, Inc, 1998

Chung, Lawrence, **Nixon**, Brian A, **Yu**, Eric (1994), "Using Quality Requirements to Systematically Develop Quality Software", Department of Computer Science, University of Toronto, Canada, Fourth International Conference on Software Quality 1994, ftp://ftp.cs.utoronto.ca/pub/eric/ICSQ4Paper.pdf (11.6.2009)

Cline, Alan (2000), "Joint Application Development (JAD) for Requirements Collection and Management", Carolla Development, 2000, http://www.carolla.com/wp-jad.htm (11.6.2009)

Dorfman, Merlin, **Thayer**, Richard (1990), "Standards, Guidelines, and Examples of System and Software Requirements Engineering", IEEE Computer Society, 1990

Ericsson, K, **Simon**, H (1993), "Protocol Analysis - Revised Edition", MIT Press, 1993

Fenton, Norman, **Ohlsson**, Niclas (2000), "Quantitative Analysis of Faults and Failures in a Complex Software System", IEEE Computer Society, 2000, http://opera.cs.uiuc.edu/probe/reference/debug/faultanalysis/QuantitativeAnalysis_Fenton00.pdf (11.6.2009)

Gause, Donald, **Weinberg**, Gerald (1989), "Exploring Requirements: Quality Before Design", Dorset House Publishing, 1989

Gottesdiener, Ellen (2005), "The Software Requirements – Memory Jogger", GOAL/QPC, 2005

Hollander, Nathan, **Mirlocca**, Naomi (1995), "Conducting Facilitated Workshops: Empowering the User to Develop Quality Systems Faster", PM Network, 1995

IEEE (1990), IEEE Std 610.12-1990: "IEEE Standard Glossary of Software Engineering Terminology", IEEE Computer Society Press, 1990, http://www.dsc.upe.br/~mlc/artigos/ieeeStandardGlossarySwEngTerminology.pdf

Komi-Sirviö, Seija (2004), "Development and Evaluation of Software Process Improvement Methods", VTT & Oulun yliopisto, väitöskirja, 2004, http://www.vtt.fi/inf/pdf/publications/2004/P535.pdf (11.6.2009)

Kotonya, Gerald, **Sommerville**, Ian (1998), "Requirements Engineering: Process and Techniques", John Wiley & Sons, 1998

Leffingwell, Dean, **Widrig**, Don (2003), "Managing Software Requirements – A Use Case Approach", Addison Wesley, 2003

McConnell, Steve (1996), "Rapid Development: Taming Wild Software Schedules", [käännös: Tarmo Toikkanen & Jussi Arola], "Ohjelmistotuotannon hallinta", Edita, IT Press, 2002

McConnell, Steve (1998), "Software Project Survival Guide", [käännös: Marko Juoperi], "Ohjelmistoprojektit: Selviytymisopas", Suomen atk-kustannus, 1998

Parviainen, Päivi, **Hulkko**, Hanna, **Kääriäinen**, Jukka (2003), "Requirements Engineering, Inventory of technologies", VTT, 2003, http://www.vtt.fi/inf/pdf/publications/2003/P508.pdf (11.6.2009)

Pressman, Roger S. (1992), "Software Engineering, A Practitioners Approach, 6th Edition", McGraw-Hill, 2005, http://ce.sharif.ir/courses/84-85/1/ce474/resources/root/Pressman_Sofware%20Engineering.pdf (11.6.2009)

Pöyhönen, Ilpo, **Hukki**, Kristiina (2004), "Riskitietoisen ohjelmiston vaatimusmäärittelyprosessin kehittäminen", VTT, 2004, http://www.vtt.fi/inf/pdf/tiedotteet/2004/T2263.pdf (11.6.2009)

Reh, John (2002), “Pareto's Principle - The 80-20 Rule”, About.com, Management, http://management.about.com/cs/generalmanagement/a/Pareto081202.htm (11.6.2009)

Spina, Mario, **Rolando**, John (2002), “JAD on a Shoestring Budget”, Software Technology Support Center, 2002, http://www.stsc.hill.af.mil/crosstalk/2002/07/spina.html (11.6.2009)

Toval, Ambrosio, **Olmos**, Alfonso, **Piattini**, Mario (2002), “Legal Requirements Reuse: A Critical Success Factor for Requirements Quality and Personal Data Protection,” 10th Anniversary Joint IEEE International Requirements Engineering Conference (RE'02), http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=1048511&isnumber=22462 (11.6.2009)

Vosburg, J, **Curtis**, Bill, **Wolverton**, R, **Albert**, B, **Malec**, H, **Hoben**, S, **Liu**, Y (1984), ”Productivity Factors and Programming Environments”, Proceedings of the 7th International Conference on Software Engineering, IEEE Computer Society, (1984), http://portal.acm.org/citation.cfm?id=801963&dl=GUIDE&coll=GUIDE&CFID=40177734&CFTOKEN=18350584 (11.6.2009)

Wiegers, Karl (2003), “Software Requirements”, Microsoft Press, 2003