Anu Lampinen

TIETOJÄRJESTELMÄVAATIMUSDOKUMENTTIEN HYÖDYNTÄMINEN YLLÄPIDOSSA: TAPAUSTUTKIMUS

JYVÄSKYLÄN YLIOPISTO
TIETOJENKÄSITTELYTIETEIDEN LAITOS
2020

TIIVISTELMÄ

Lampinen, Anu
Tietojärjestelmävaatimusdokumenttien hyödyntäminen ylläpidossa: tapaustutkimus
Jyväskylä: Jyväskylän yliopisto, 2020, 77 s.
Tietojärjestelmätiede, pro gradu -tutkielma
Ohjaaja: Seppänen, Ville

Tässä tapaustutkimuksessa esitettiin vaatimusmäärittelyprosessi ja havainnollistettiin, miten se esiintyy kolmessa yleisessä tietojärjestelmän kehittämisprosessissa: vaihejakomalleissa, RUP-kehyksessä ja Scrum-prosessissa. Tutkielmassa avattiin vaatimusmäärittelyprosessin vaiheita ja pureuduttiin niistä yhteen, vaatimusten esittämiseen. Yleisimmät vaatimusten esittämistavat, kuten käyttötapaus, sekvenssikaavio sekä tilakaavio, esiteltiin, ja niitä vertailtiin toisiinsa yleisyyden ja formaalisuuden perusteella. Tämän jälkeen tutkimuksessa kuvattiin tietojärjestelmän ylläpidon konteksti ja ylläpitoon liittyvien tehtävien tyypit. Lisäksi tarkasteltiin kolmea eri ylläpitoprosessia: pikakorjausmallia, iteratiivista parannusmallia ja IEEE 1219-1998 -standardia ylläpidolle. Ylläpitoprosessin esittelyn jälkeen esitettiin yleisiä tietojärjestelmän ylläpitoon liittyvä haasteita sekä dokumentaation hyödyntämistä ylläpidossa. Tutkimuksessa tutkittiin viittä erilaista ylläpidossa olevaa järjestelmää, niiden ylläpidon organisointia sekä dokumentaation hyödyntämistä ylläpidon aikana. Ominaisuuksiltaan erilaiset tietojärjestelmän noudattelivat erilaista ylläpitoprosessia ja hyödynsivät sen aikana eri vaatimusdokumentteja. Yleisimmin hyödynnytetyt dokumentit olivat käyttötapaukset ja käyttöliittymän eritasoiset kuvaukset. Dokumenttien tärkein tehtävä oli toimia ylläpidon aikaisten muutosten lähtökohtana. Dokumentteja hyödynnettiin myös tietojärjestelmän opettelussa ja tiedon lähteenä. Dokumenttien hyödyntämistä vaikeutti erilaiset ongelmat. Ongelmia ylläpidossa tuottivat mm. puutteellinen tai puuttuva dokumentaatio ja vaikeaselkoiset tai heikkolaatuiset dokumentit. Ongelmien korjaaminen ei vaatisi ihmeitä, vaan ongelmat olisivat selvitettävissä systemaattisella otteella ja hyvällä harkinnalla.

Asiasanat: vaatimusmäärittelyt, vaatimukset, esittäminen, ylläpito, dokumentointi

ABSTRACT

Lampinen, Anu
Exploiting Information System Requirement Documents in System Maintenance: A Case Study
Jyväskylä: University of Jyväskylä, 2020, 77 p.
Information Systems, Master's Thesis
Supervisor: Seppänen, Ville

This case study presented the requirements engineering process in general and related to three well-known software engineering processes, life cycle model, Rational Unified Process and Scrum process. This thesis elaborated the phases of requirements engineering process and focused on one part of it, requirements presentation. A set of commonly used requirements presentation forms, such as use case diagram, use case, sequence diagram and state diagram, were presented and compared on the basis generality and formality. After that, software maintenance was introduced. The study then described the context of software maintenance and different types of maintenance tasks. In addition, three different maintenance processes were studied: quick-fix model, iterative-enhancement model and the IEEE 1219-1998 standard for software maintenance. After presenting the software maintenance process, general challenges related to software maintenance and the exploitation of documentation in maintenance were presented. This thesis studied five different software under maintenance. The characteristics of these software varied, they used different maintenance process and exploited different kind of documentation. The most commonly used documents were use cases and user interface descriptions at different levels. The most important function of documents was to serve as a starting point for maintenance tasks. They were also used in learning a new software and as a source of information. Some issues were found that complicated the exploitation of documents. Problems in maintenance had been caused by e.g. incomplete documentation, complete lack of documentation and documents that are hard to understand or of poor quality. Correcting these problems would not require miracles, but they could be overcome with a systematic approach and good judgement.
Keywords: requirements engineering, requirements, presentation, maintenance, documentation

KUVIOT

KUVIO 1 Vaatimusmäärittelyprosessin vaiheet 12
KUVIO 2Vaatimusmäärittelyprosessi kolmen ulottuvuuden näkökulmasta 13
KUVIO 3 Vesiputousmalli 16
KUVIO 4 RUP-kehys 17
KUVIO 5 Scrum-prosessin yleiskuva 19
KUVIO 6 Suhteet mallien, dokumenttien, dokumentaation ja lähdekoodin välillä 22
KUVIO 7 Dokumenttityypit 22
KUVIO 8 Vaatimusten nelijako 25
KUVIO 9 Esimerkki käyttötapauskaaviosta 26
KUVIO 10 Esimerkki käyttötapauksesta 27
KUVIO 11 Esimerkki rautalankamallista 29
KUVIO 12 Esimerkki luokkakaaviosta 30
KUVIO 13 Esimerkki kontekstitason tietovuokaaviosta 31
KUVIO 14 Esimerkki sekvenssikaaviosta 32
KUVIO 15 Esimerkki yksinkertaisesta tilakaaviosta 33
KUVIO 16 Vaatimusten esittämistapojen nelikenttä 34
KUVIO 17 Ohjelmiston ylläpidon konteksti. 37
KUVIO 18 Pikakorjausmalli. 40
KUVIO 19 Yksinkertainen pikakorjausmalli. 40
KUVIO 20 Iteratiivisen parantamisen malli 41
KUVIO 21 IEEE 1219-1998 Ohjelmiston ylläpidon prosessimalli 42
KUVIO 22 Ylläpitoprosessissa käytettävät mallit 45
KUVIO 23 Tutkimukseen valittujen tapausten vertailu 69

TAULUKOT

TAULUKKO 1 Ylläpitotyypit 38
TAULUKKO 2 Prosenttiosuudet ohjelmistokehittäjistä, jotka arvioivat dokumentaation hyödylliseksi tai erittäin hyödylliseksi tietyissä tehtävissä 46
TAULUKKO 3 Tutkimukseen valittujen tapausten vertailu 50
TAULUKKO 4 Olemassa olevat ja ylläpidossa hyödynnetyt dokumentit 62

SISÄLLYS

TIIVISTELMÄ 2

ABSTRACT 3

KUVIOT 4

TAULUKOT 4

SISÄLLYS 5

1 JOHDANTO 7

2 VAATIMUSMÄÄRITTELY 11
2.1 Vaatimusmäärittelyprosessi 11
2.2 Vaatimukset 14
2.3 Vaatimusmäärittely osana tietojärjestelmän kehittämistä 15
2.4 Yhteenveto 19

3 DOKUMENTOINTI 20
3.1 Dokumentoinnin tarkoitus, tarve ja luonti 20
3.2 Dokumenttityypit 22
3.2.1 Prosessidokumentit 23
3.2.2 Tuotedokumentit 23
3.3 Vaatimusten esittäminen 24
3.3.1 Skenaariopohjaisia esitystapoja 25
3.3.2 Luokkamallipohjaisia esitystapoja 30
3.3.3 Tietovuopohjaisia esitystapoja 30
3.3.4 Käyttäytymismallipohjaisia esitystapoja 31
3.3.5 Vertailu 33
3.4 Yhteenveto 35

4 OHJELMISTON YLLÄPITO 36
4.1 Ohjelmiston ylläpidon konteksti 36
4.2 Ylläpitotyypit 38
4.3 Ylläpitoprosessi 39
4.3.1 Pikakorjausmalli 39
4.3.2 Iteratiivinen parannusmalli 40
4.3.3 IEEE 1219-1998 -standardi järjestelmän ylläpidolle 41
4.4 Haasteet ylläpidossa 43
4.5 Dokumentaation hyödyntäminen ylläpidossa 45
4.6 Yhteenveto 47

5 TAPAUSTUTKIMUKSEN TOTEUTTAMINEN 48

5.1 Tutkimusmenetelmä 48
5.2 Tutkimuksen kohteet 50
5.2.1 Tapaus 1 51
5.2.2 Tapaus 2 51
5.2.3 Tapaus 3 51
5.2.4 Tapaus 4 51
5.2.5 Tapaus 5 52

6 TAPAUSTUTKIMUKSEN TULOKSET 53
6.1 Yleiskuvaus 53
6.2 Ylläpitoprosessi 54
6.3 Ylläpidon organisointi 54
6.4 Vaatimusdokumentaatio 55
6.5 Vaatimusdokumenttien hyödyntäminen 57
6.6 Dokumentaatioon liittyvät ongelmat 58
6.7 Dokumentaatioon liittyvien ongelmien ratkaisut 59
6.8 Dokumentaatioon liittyvät toiveet 60
6.9 Yhteenveto 61

7 POHDINTA 62
7.1 Ylläpidossa hyödynnettävät esittämistavat 62
7.2 Havaitut hyödyt ja puutteet 66
7.3 Tutkimuksen validiteetti ja reliabiliteetti 68
7.4 Jatkotutkimusaiheita 70

8 YHTEENVETO 72

LÄHTEET 73

1 JOHDANTO

Tässä luvussa taustoitetaan tutkimus. Ensin esitetään tutkimuksen tausta ja motivoidaan lukija. Toiseksi esitetään tutkimusongelma ja tutkimuskysymykset. Viimeiseksi esitetään tutkielman rakenne.

Vaatimusmäärittely on olennainen osa tietojärjestelmien kehittämistä. Lamsweerde (2011) rinnastaa vaatimusmäärittelyn ongelmanratkaisuun, jonka tavoitteena on määrittää, minkä ongelman kehitettävä ohjelmisto tulee ratkaisemaan, miksi ongelma tulee ratkaista ja keiden pitää osallistua ongelman ratkaisuun. Ennen järjestelmän kehittämistä tulee siis ymmärtää, mitä järjestelmän oletetaan tekevän ja kuinka sen käyttö tukee järjestelmän maksavien yksilöiden tai liiketoiminnan tavoitteita. Tähän sisältyy sovellusalueen tuntemus, järjestelmän operatiiviset rajoitteet, sidosryhmien vaatima toiminnallisuus sekä välttämättömät järjestelmän ominaisuudet kuten suorituskyky, turvallisuus ja luotettavuus. *Vaatimusmäärittelyllä* tarkoitetaan strukturoitua joukkoa aktiviteettejä, jotka auttavat saavuttaman tämän ymmärryksen ja dokumentoimaan järjestelmän määrittelyn sidosryhmiä ja järjestelmän kehittäjiä. (Sommerville, 2005)

Vaatimusmäärittelyä voidaan tehdä lukuisin eri tavoin riippuen projektin rakenteesta, hallinnasta ja laajuudesta. Yleisimmin vaatimusmäärittelyn todetaan sisältävän seuraavat vaiheet: vaatimusten kerääminen, analysointi, validointi, neuvottelu, dokumentointi ja hallinta (Sommerville, 2005). Vaatimusten keräämisvaiheessa etsitään ja löydetään vaatimukset tunnistetuista lähteistä. Analysointivaiheessa sisäistetään vaatimukset ja tunnistetaan vaatimusten keskinäiset ristiriitaisuudet ja päällekkäisyydet. Validointivaiheessa tarkistetaan, että vaatimukset vastaavat sidosryhmien tarpeita. Sidosryhmien ristiriitaiset vaatimukset sovitetaan yhteen neuvotteluvaiheessa, jonka tavoitteena on tuottaa johdonmukaiset vaatimukset yhteen sovittamalla erilaisia näkemyksiä. Yhteen sovitetut vaatimukset kirjataan dokumentointivaiheessa sidosryhmien ja tietojärjestelmäkehittäjien ymmärtämällä tavalla. Hallintavaiheessa hallitaan vaatimuksiin kohdistuvat muutokset. (Sommerville, 2005)

Vaatimusmäärittelyn tavoitteena on tuottaa laadukkaita vaatimuksia rakennettavalle tietojärjestelmälle. Vaatimukset jaetaan tyypillisesti toiminnallisiin

vaatimuksiin ja ei-toiminnallisiin vaatimuksiin. *Toiminnalliset* (functional) *vaatimukset* kuvaavat, mitä tietojärjestelmän tulee tehdä. Ne kuvaavat järjestelmälle annettavan syötteen, järjestelmän antaman vastauksen sekä käyttäytymisen niiden välillä (Young, 2003). *Ei-toiminnallisilla* (non-functional) *vaatimuksilla* tarkoitetaan tietojärjestelmän laatuvaatimusten lisäksi tietojärjestelmään ja sen kehittämiseen liittyviä rajoitteita (Kotonya & Sommerville, 1998). Näitä ovat esimerkiksi, että tietojärjestelmän on oltava käytettävissä arkipäivisin klo 8–16 tai tietojärjestelmä tulee olla käytettävissä Android-alustalla.

Vaatimuksia voidaan esittää hyvin monella tavalla. Tavat vaihtelevat hyvin strukturoidusta esitysmuodosta lähes vapaamuotoiseen. Yleisiä esitystapoja on muun muassa *käyttötapaukset* (use cases) (Gallardo-Valencia, Olivera & Sim, 2007) ja *käyttäjätarinat* (user stories) (Savolainen, Kuusela & Vilavaara, 2010). Näiden lisäksi käytetään esimerkiksi luonnollista kieltä (Denger, Berry & Kamsties, 2003) sekä erilaisia malleja ja kaavioita, kuten *tilakaavioita* (state diagram), *luokkakaavioita* (class diagram) ja *aktiviteettikaavioita* (activity diagram) (Pressman, 2010). Tietojärjestelmiä kehitettäessä vaatimusten esitystavat ovat usein menetelmän ohjeistamia. Esimerkiksi *RUP* (Rational Unified Process) (Rational Software, 1998) ohjaa käyttämään käyttötapauksia. Ennen dokumentoinnin aloittamista on kuitenkin tärkeää miettiä, millaisia esitystapoja on ja mitkä niistä palvelevat käyttötarkoituksia parhaiten.

Tietojärjestelmän elinkaaren vaiheista ylläpito vie eniten resursseja. *Ylläpidolla* tarkoitetaan kaikkia niitä muutoksia, jotka tehdään järjestelmään sen julkaisun jälkeen. Se on kallein ja pitkäkestoisin vaihe. Ylläpito voi olla *korjaavaa* (corrective), *mukauttavaa* (adaptive), *täydentävää* (perfective) tai *ennakoivaa* (preventive) (Swanson, 1976; ISO/IEC, 1999). Ylläpitoon liittyvät muutokset ovat moninaisia, sillä tietojärjestelmä tulee pitää ajan tasalla muuttuvassa maailmassa niin käytettävien toimintojen kuin käyttöjärjestelmien ja laitteiston osalta. (de Souza, Anquetil & de Oliveira, 2005).

Basili (1990) esittelee kolme erilaista ylläpidon prosessimallia: pikakorjausmalli (quick-fix model), iteratiivinen parannus -malli (iterative-enhancement model) ja uudelleenkäyttö mallin (full-resuse model). *Pikakorjausmallissa* muutetaan olemassa olevaa järjestelmää vain niiltä osin, kuin se on välttämätöntä. Ohjelmakoodimuutosten lisäksi saatetaan muuttaa myös esimerkiksi dokumentaatiota, mutta useimmiten dokumentaatio jää päivittämättä. Järjestelmää *iteratiivisesti parannettaessa* järjestelmän ylläpidon aikaisia muutoksia tehdään iteraatioissa samaan tapaan kuin järjestelmän kehitysvaiheessa. Kunkin iteraation aikana voidaan päivittää järjestelmän dokumentaatiota kaikilla abstraktiotasoilla ja muuttaa toteutusta pienestä virheenkorjauksesta aina koko järjestelmän uudelleensuunnitteluun saakka. Sen sijaan *uudelleenkäyttömalli* lähtee liikkeelle vaatimusten analysoinnista ja uuden järjestelmän suunnittelusta uudelleenkäyttäen ylläpidettävän tietojärjestelmän olemassa olevia osia: vaatimuksia, suunnittelua, ohjelmakoodia ja testejä. Olemassa olevia osia analysoidaan ennen niiden mahdollista uudelleenkäyttöä, ja uusia dokumentteja kirjoitetaan tarvittaessa. (Basili, 1990)

Ylläpidossa yksi suurimmista ongelmista on puutteellinen tai päivittämätön dokumentaatio. De Souzan, Anquetilin ja de Oliveiran (2005) mukaan suuri osa organisaatioista käyttää pika-korjausmallia järjestelmien ylläpidossa, eikä näistä suurin osa päivitä dokumentaatiota ohjelmakoodin päivitysten ohessa. Tämä johtaa siihen, että ylläpidossa tarvittavia tietoja joudutaan joskus etsimään jopa ohjelmakoodista asti (de Souza, Anquetil & de Oliveira, 2005; Dekleva, 1992.). Tämä voi johtaa siihen, että järjestelmään toteutetaan ylläpidon aikana käyttökelvotonta tai turhaa toiminnallisuutta. Se aiheuttaa resurssien haaskaamista ja menetettyjä mahdollisuuksia toisaalla. (Layzell & Macaulay, 1994). Deklevan (1992) mukaan puutteellisesta dokumentaatiosta aiheutuvat ongelmat liittyvät erityisesti vanhempiin järjestelmiin, joissa ylläpidon aikainen tuottavuus vähenee ja kustannukset nousevat. Ongelmia aiheutuu myös ylläpidon organisoinnille, kun uusien ylläpitäjien perehtyminen ylläpidettävään järjestelmään pitkittyy ja vanhat ylläpitäjät eivät voi siirtyä toisiin tehtäviin.

Tietojärjestelmävaatimusten dokumentoinnin hyödyntämisestä tai dokumenttien puutteista ja puuttumisesta on tehty useita tutkimuksia kymmenien vuosien ajan. Tässä tutkielmassa tutkitaan laajan organisaation joidenkin tietojärjestelmien vaatimusdokumentaatiota ja vertaillaan niissä havaittuja hyötyjä ja puutteita kirjallisuudessa esitettyihin. Tarkasteltavia tutkimuksia ovat mm. Dekleva (1992), Tryggeseth (1997a), Lethbridge, Singer & Forward (2003) ja de Souza, Anquetil & de Oliveira (2005).

Tämän tutkimuksen kiinnostuksen kohteena ovat tietojärjestelmiä koskevat vaatimusdokumentit ja niiden käyttö ylläpitovaiheessa. Tutkimusongelma voidaan muotoilla seuraavasti:

> *Millaisia seikkoja tulisi ottaa huomioon tietojärjestelmävaatimusten dokumentoinnissa tietojärjestelmän ylläpidon näkökulmasta?*

Tutkimusongelma voidaan jakaa seuraaviin tutkimuskysymyksiin:

- Millaisia tietojärjestelmävaatimusten esittämistapoja on olemassa?
- Miten tietojärjestelmiä ylläpidetään?
- Millä tavalla tietojärjestelmävaatimuksia koskevia dokumentteja hyödynnetään ylläpidossa?

Tutkielma rakentuu seitsemästä luvusta. Luvussa 2 kerrotaan, mitä vaatimusmäärittelyllä tarkoitetaan, millaisia vaiheita siihen kuuluu, miten vaatimuksia voidaan luokitella, millaisia laatukriteereitä vaatimuksiin voidaan liittää sekä miten vaatimusmäärittelyä tehdään kolmessa eri tietojärjestelmänkehittämismenetelmässä. Luvussa 3 esitetään ensin vaatimusten esitystapojen ryhmittelyjä, valitaan niistä tähän tutkimukseen soveltuvin ja sen mukaisesti esitellään erilaisia vaatimusten esitystapoja. Luvussa määritellään myös esittämistapojen arviointikriteerejä ja niiden mukaisesti arvioidaan edellä kuvattuja esittämistapoja. Luvussa keskitytään yleisimpiin esitystapoihin. Luvussa 4 kerrotaan, mitä ylläpidolla tarkoitetaan sekä esitellään ylläpitoprosessi ja prosessin vaiheet. Lisäksi esitetään, millaisissa haasteita ylläpitoon liittyy, miten ylläpidon aikana voidaan

hyödyntää vaatimusmäärittelydokumentaatiota ja millaista ohjeistusta hyödyntämiseen liittyy. Luvussa 5 esitetään, miten empiirinen tutkimus on toteutettu. Luvussa 6 esitetään tutkimuksen tulokset esittelemällä ensin valittujen tietojärjestelmien vaatimusten dokumentointi, niiden hyödyntäminen ylläpidossa sekä dokumentteja hyödyntävien ylläpitäjien kertomat hyödyt ja ongelmat. Lisäksi mainitaan joitain kehittämisehdotuksia. Luvussa 7 pohditaan, mistä tutkimuksessa löydetyt hyödyt johtuvat ja kuinka hyötyihin johtaneita seikkoja voidaan hyödyntää. Tämän lisäksi pohditaan syitä löytyneisiin puutteisiin ja kuinka niitä voitaisiin ehkäistä ja/tai korjata. Luvussa tarkastellaan myös tutkimuksen reliabiliteettia ja validiteettia. Luvussa 8 kerrataan tutkimuksen tulokset ja kuvataan, miten tuloksia voidaan hyödyntää. Lisäksi nostetaan esille mahdolliset puutteet tutkimuksessa sekä jatkotutkimusaiheet.

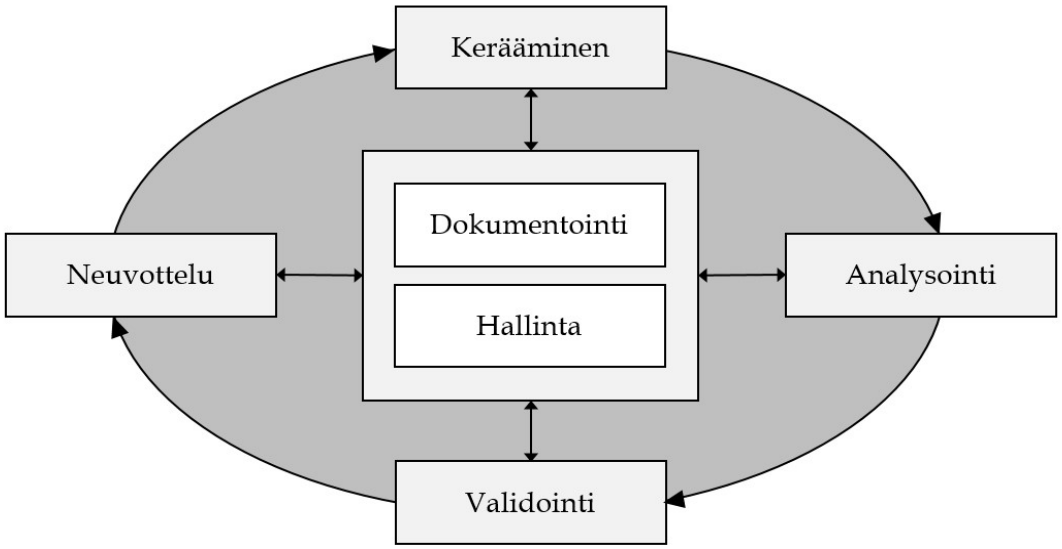
2 VAATIMUSMÄÄRITTELY

Vaatimusmäärittely on olennainen osa tietojärjestelmien kehittämistä. van Lamsweerde (2011) rinnastaa vaatimusmäärittelyn ongelmanratkaisuun, jonka tavoitteena on määrittää, minkä ongelman kehitettävä ohjelmisto tulee ratkaisemaan, miksi ongelma tulee ratkaista ja keiden pitää osallistua ongelman ratkaisuun. Ennen järjestelmän kehittämistä tulee siis ymmärtää, mitä järjestelmän oletetaan tekevän ja kuinka sen käyttö tukee järjestelmän maksavien yksilöiden tai liiketoiminnan tavoitteita. Tähän sisältyy sovellusalueen tuntemus, järjestelmän operatiiviset rajoitteet, sidosryhmien vaatima toiminnallisuus sekä välttämättömät järjestelmän ominaisuudet kuten suorituskyky, turvallisuus ja luotettavuus. *Vaatimusmäärittelyllä* tarkoitetaan strukturoitua joukkoa aktiviteettejä, jotka auttavat saavuttaman tämän ymmärryksen ja dokumentoimaan järjestelmän määrittelyn sidosryhmiä ja järjestelmän kehittäjiä varten. (Sommerville, 2005).

Tässä luvussa esitetään vaatimusmäärittelyprosessi ja siihen liittyvät vaiheet. Tämän jälkeen kerrotaan, mitä tarkoitetaan vaatimuksella, millaisia erilaisia vaatimuksia yleisesti tunnetaan ja miten vaatimusten laatu voidaan varmistaa. Lisäksi esitetään kolmen yleisen tietojärjestelmän kehitysmenetelmän näkökulmasta, miten vaatimusmäärittely näkyy osana tietojärjestelmän kehittämisprosessissa.

2.1 Vaatimusmäärittelyprosessi

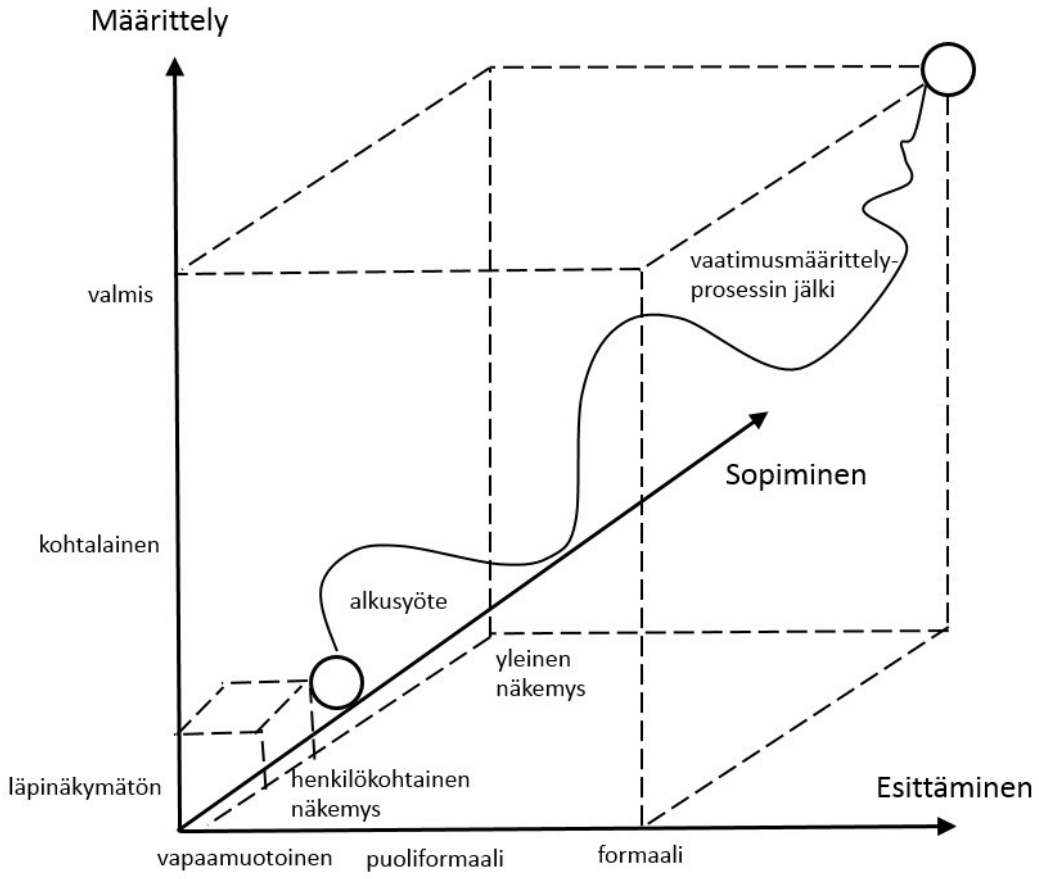
Vaatimusmäärittelyä voidaan tehdä lukuisin eri tavoin riippuen projektin rakenteesta, hallinnasta ja laajuudesta. Yleisimmin vaatimusmäärittelyprosessin todetaan sisältävän seuraavat vaiheet (Sommerville, 2005): vaatimusten kerääminen (elicitation), analysointi (analysis), validointi (validation), neuvottelu (negotiation), dokumentointi (documentation) ja hallinta (management). Vaatimusten *keräämisvaiheessa* etsitään ja löydetään vaatimukset tunnistetuista lähteistä. *Analysointivaiheessa* sisäistetään vaatimukset ja tunnistetaan vaatimusten keskinäiset ristiriitaisuudet ja päällekkäisyydet. *Validointivaiheessa* tarkistetaan, että vaatimukset vastaavat sidosryhmien tarpeita. Sidosryhmien ristiriitaiset vaatimukset sovitetaan yhteen *neuvotteluvaiheessa*, jonka tavoitteena on tuottaa johdonmukaiset vaatimukset yhteen sovittamalla erilaisia näkemyksiä. Yhteen sovitetut vaatimukset kirjataan *dokumentointivaiheessa* sidosryhmien ja tietojärjestelmäkehittäjien ymmärtämällä tavalla. *Hallintavaiheessa* hallitaan vaatimuksiin kohdistuvat muutokset. Vaatimusmäärittelyprosessin vaiheet kuvataan usein lineaarisena. Todellisuudessa vaiheita toistetaan syklisesti (Kuvio 1), kunnes vaatimukset on löydetty ja määritelty. Vaatimusten dokumentointi ja hallinta on sijoitettu kuviossa keskellä, sillä niitä tehdään koko vaatimusmäärittelyprosessin ajan. (Sommerville, 2005).



KUVIO 1 Vaatimusmäärittelyprosessin vaiheet (Sommerville, 2005, 17)

Pohlin (1994) mukaan vaatimusmäärittelyprosessi voidaan nähdä kolmen ulottuvuuden rajaamalla alueella. Ulottuvuudet ovat: määrittely (specification), sopiminen (agreement) ja esittäminen (representation) (Kuvio 2). *Määrittelyulottuvuudessa* parannetaan ymmärrystä vielä tuntemattomasta järjestelmästä tavoitteena järjestelmän valmis dokumentaatio. Vaatimusmäärittelyprosessin alussa tietojärjestelmän ja sen ympäristön määrittely on läpinäkymätöntä (opaque). Prosessin edetessä määrittely selkeytyy, jolloin järjestelmästä on kohtalainen (fair) määrittely. Prosessin lopussa määrittelyn tulisi olla valmis (complete). Valmiilla määrittelyllä tarkoitetaan standardien ja ohjeistuksien mukaista vaatimusmäärittelydokumentaatiota, joka sisältää tietojärjestelmän kaikki erityyppiset, validoidut vaatimukset. *Esittämisulottuvuudessa* muunnetaan vapaamuotoinen tieto formaalien esitystapojen mukaisiksi. Vaatimusmäärittelyprosessin alussa kuvataan tietojärjestelmän vaatimuksia *vapaamuotoisten* (informal) esittämistapojen avulla. Vapaamuotoisia esittämistapoja ovat esimerkiksi luonnollinen kieli, vapaasti piirretyt kuvat, esimerkinomaiset kuvaukset, äänet ja animaatiot. Vapaamuotoiset esitystavat ovat hyvin käyttäjälähtöisiä, ja niillä voidaan ilmaista kaikenlaisia vaatimuksia rajoitteitta. Prosessin edetessä vaatimuksia tarkennetaan puoliformaaleilla (semiformal) esitystavoilla. *Puoliformaalit* esitystavat sisältävät tyypillisesti graafisia malleja, joita käytetään antamaan hyvä yleiskäsitys laajemmasta alueesta tai tarkempi kuvaus jostakin sen osasta. Graafisen mallin abstrakti ja konkreetti syntaksi on määritelty tarkasti. Sen sijaan formaali semantiikka puuttuu. Prosessin päätteeksi vaatimusmäärittelyprosessin myötä vaatimukset voidaan kuvata formaalein esitystavoin. *Formaalit* (formal) esitystavat perustuvat johonkin formaaliin kieleen. Kieli on formaali, jos sen syntaksi ja semantiikka on määritelty formaalisti (Krogstie & Sölberg, 1996). Tämän vuoksi formaalilla ta-

valla esitetyistä vaatimuksista pystytään osittain jopa generoimaan ohjelmakoodia automaattisesti. Kaikki kolme tapaa (vapaamuotoiset, puoliformaalit ja formaalit) ovat tarpeellisia, ja jäljitettävyys niiden välillä tulisi säilyttää.



KUVIO 2Vaatimusmäärittelyprosessi kolmen ulottuvuuden näkökulmasta (Pohl, 1994, 8)

Kolmantena ulottuvuutena on vaatimusmäärittelystä *sopiminen*. Vaatimusmäärittelyprosessin alussa jokaisella vaatimusmäärittelyyn osallistuvalla on oma *henkilökohtainen näkemys* (personal view) siitä, mitä vaatimuksia tietojärjestelmän tulee täyttää. Näkemykset eroavat toisistaan mm. henkilön roolista (asiakas, järjestelmäkehittäjä) riippuen. Näiden henkilökohtaisten näkemysten kautta syntyy *yhteinen näkemys* (common view) siitä, mitä tietojärjestelmän vaatimukset ovat. Yhteisen näkemyksen saavuttaminen on tärkeä osa vaatimusmäärittelyprosessia, sillä samoista asioista on usein erilaisia näkemyksiä ja neuvottelemalla niistä saadaan paras mahdollinen lopputulos. (Pohl, 1994).

2.2 Vaatimukset

Vaatimusmäärittelyn tavoitteena on tuottaa laadukkaita vaatimuksia rakennettavalle tai muokattavalle tietojärjestelmälle. *Vaatimus* on tietojärjestelmän ominaisuus (property), joka ratkaisee tietyn ongelman. Ominaisuudet voivat liittyä toimintojen automatisointiin, organisaation liiketoimintaprosessien tukemiseen tai esimerkiksi muokattavan tietojärjestelmän puutteiden korjaamiseen. Ominaisuudet voivat liittyä niin tietojärjestelmän käyttäjään tai muihin henkilöihin, itse tietojärjestelmään tai laitteeseen, jolla tietojärjestelmää ajetaan. (IEEE, 2013)

Vaatimukset jaetaan tyypillisesti toiminnallisiin vaatimuksiin ja ei-toiminnallisiin vaatimuksiin. *Toiminnalliset* (functional) *vaatimukset* kuvaavat, mitä tietojärjestelmän tulee tehdä. Ne kuvaavat järjestelmälle annettavan syötteen, järjestelmän antaman vastauksen sekä käyttäytymisen niiden välillä (Young, 2003). *Ei-toiminnallisilla* (non-functional) *vaatimuksilla* tarkoitetaan tietojärjestelmän laatuvaatimusten lisäksi tietojärjestelmään ja sen kehittämiseen liittyviä rajoitteita (Kotonya & Sommerville, 1997). Näitä ovat esimerkiksi, että tietojärjestelmän on oltava käytettävissä arkipäivisin klo 8–16 tai tietojärjestelmän tulee olla käytettävissä Android-alustalla.

Laplante (2009) lisää toiminnallisten ja ei-toiminnallisten vaatimusten rinnalle on myös *liiketoiminta-alueen vaatimukset* (domain requirements). Liiketoiminta-alueen vaatimukset johdetaan *sovellusalueesta* (application domain). Liiketoiminta-alueen vaatimukset voivat olla uusia toiminnallisia vaatimuksia tai rajoitteita. Esimerkiksi ilmailualalla matkalaukkujen käsittelyjärjestelmään liittyvät standardit synnyttävät uusia vaatimuksia. (Laplante, 2009).

Kotonya ja Sommerville (1997) jaottelevat vaatimukset viiteen tyyppiin. Ensimmäisen tyyppiset vaatimukset ovat hyvin yleisiä vaatimuksia siitä, mitä järjestelmän tulee tehdä. Toiminnalliset vaatimukset sisältävät tarkemmalla tasolla järjestelmän jonkin toiminnon, esimerkiksi millä hakukriteereillä tulee pystyä hakemaan kirjaston kirjoja. Toteutusvaatimukset asettavat rajoituksia järjestelmän toteuttamiselle tai käyttämiselle. Suorituskykyvaatimuksilla asetetaan järjestelmälle vähimmäissuorituskyky. Tällä tarkoitetaan esimerkiksi yhtäaikaisten tapahtumien määrää. Viidentenä ovat käytettävyysvaatimukset, joilla ilmaistaan järjestelmän suurinta vasteaikaa. (Kotonya & Sommerville, 1997).

Vaatimusten laatu riippuu vaatimusmäärittelyn onnistumisesta. Vaatimusmäärittelyn epäonnistuminen voi johtaa siihen, ettei tietojärjestelmä valmistu suunnitellussa aikataulussa tai budjetissa. Asiakkaiden ja loppukäyttäjien ollessa tyytymättömiä järjestelmä saattaa jäädä osittain tai kokonaan käyttämättä. Lisäksi tietojärjestelmä saattaa olla epävakaa, virhealtis sekä kustannustehoton ylläpitää tai jatkokehittää (Kotonya & Sommerville, 1997). Kittlausin ja Cloughin (2009) mukaan useat tutkimukset osoittavat, että yleisin syy projektin epäonnistumiseen on riittämätön vaatimustenhallinta tai sen puuttuminen kokonaan. Yksi osa vaatimustenhallintaa on vaatimusten dokumentointi.

Vaatimusten laadun tarkistamiseen ja parantamiseen voidaan käyttää erilaisia laatukriteerejä. Attarha ja Modiri (2011) esittävät vaatimuksille neljä laatukriteeriä: vaatimuksen tulee olla kattava (complete), yhteen sopiva (compatible), yksiselitteinen (unambiguous) ja testattava (testable). Hull, Jackson ja Dick (2005) esittävät tarkemman laatukriteeristön, jossa laatuvaatimuksia esitetään sekä yksittäiselle vaatimukselle että koko vaatimusjoukolle. Jokaisen yksittäisen vaatimuksen tulee olla *atominen* (atomic). Atominen vaatimus sisältää yhden jäljitettävän elementin. Vaatimuksen tulee olla *laillinen* (legal), jolloin se on toteutettavissa lakien rajoissa, sekä *tunnistettava* (unique), jolloin se voidaan yksilöidä. Vaatimuksen tulee olla myös *toteuttamiskelpoinen* (feasible), jotta se pystytään toteuttamaan teknisesti kustannusten ja aikataulun puitteissa. Vaatimuksen tulee olla *selkeä* (clear) eli selkeästi ymmärrettävissä sekä ytimekäs ja *tarkka* (precise), mutta samalla sen tulee olla *abstrakti* (abstract), jottei vaatimus määrää suunnittelutason ratkaisuja. Lisäksi vaatimuksen tulee olla *todennettavissa* (verifiable) tunnetulla tavalla. (Hull ym., 2005.)

Yksittäisen vaatimuksen laadun lisäksi koko vaatimusjoukon laatu on tärkeää. *Kattava* (complete) vaatimusjoukko sisältää kaikki tarpeelliset vaatimukset, jotka on ilmaistu kerran. Tällöin vaatimusjoukko *ei sisällä toisteisuutta* (non-redundant). *Johdonmukaisen* (consistent) vaatimusjoukon vaatimukset eivät ole ristiriidassa keskenään. *Lajitellussa* (modular) vaatimusjoukossa toisiinsa liittyvät vaatimukset on kirjattu lähekkäin. Vaatimusjoukon tulee olla *rakenteinen* (structured), jolloin vaatimukset on kirjattu dokumenttiin, jossa on selkeä rakenne. Lisäksi vaatimusjoukon tulee olla jäljitettävissä (satisfied, qualified). (Hull ym., 2005.)

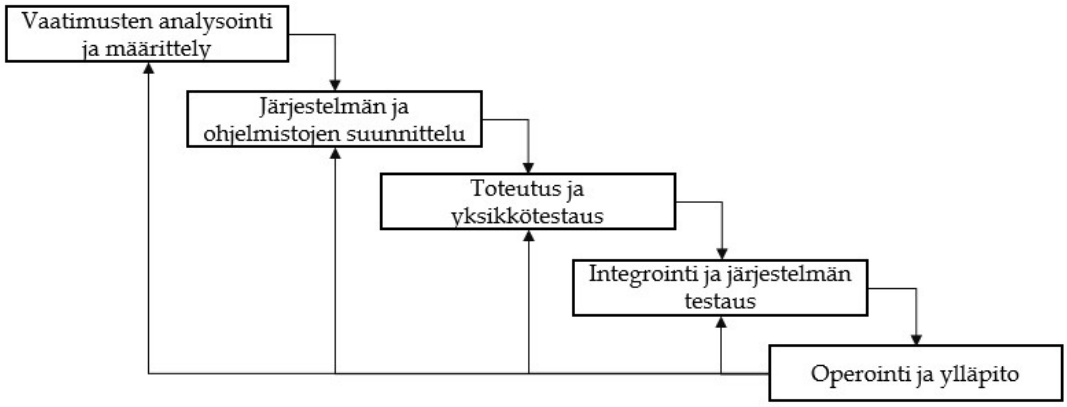
2.3 Vaatimusmäärittely osana tietojärjestelmän kehittämistä

Tietojärjestelmän kehitysprosessissa tunnistetaan tyypillisesti neljä yleistä aktiviteettia: määrittely (specification), suunnittelu (design) ja toteutus (implementation), validointi (validation) ja edelleen kehittäminen (evolution). *Määrittelyssä* etsitään toteutettavan tietojärjestelmän toiminnallisuudet ja sen operoinnin rajoitteet. *Suunnittelun* ja *toteutuksen* tavoitteena on tuottaa määritelty tietojärjestelmä. *Validoinnilla* varmistetaan, että tietojärjestelmä täyttää asiakkaan toiveet. *Edelleen kehittämisessä* tietojärjestelmä pidetään asiakkaan muuttuneiden toiveiden tasalla. (Sommerville, 2007.)

Vaatimusmäärittelyyn liittyviä tehtäviä suoritetaan tietojärjestelmän kehittämisen eri vaiheissa. Vaihe riippuu käytettävästä tietojärjestelmän kehittämisprosessista. Seuraavaksi vaatimusmäärittelyn asemaa tietojärjestelmän kehittämisessä tarkastellaan ensin vesiputousmallin (Sommerville, 2007), sitten RUP-kehyksen (Rational Software, 1998) ja lopuksi ketterän kehittämisen yhteydessä.

Tunnetuin tietojärjestelmien kehittämisprosessi on vesiputousmalli (waterfall model) eli vaihejakomalli (life cycle model) (Royce, 1970). Vaihejakomallissa tietojärjestelmän kehittämisen perusaktiviteetit esitetään lineaarisina prosessin

vaiheina. Roycen (1970) malliin pohjautuvia vaihejakomalleja on esitetty kirjallisuudessa paljon. Yksi näistä on Sommervillen (2007) malli. Kuviossa 3 esitetyn prosessin vaiheet ovat vaatimusten analysointi ja määrittely (requirements analysis and definition), järjestelmän ja ohjelmiston suunnittelu (system and software design), toteutus ja yksikkötestaus (implementation and unit testing), integrointi ja järjestelmän testaus (integration and system testing) sekä operointi ja ylläpito (operation and maintenance) (Sommerville, 2007).

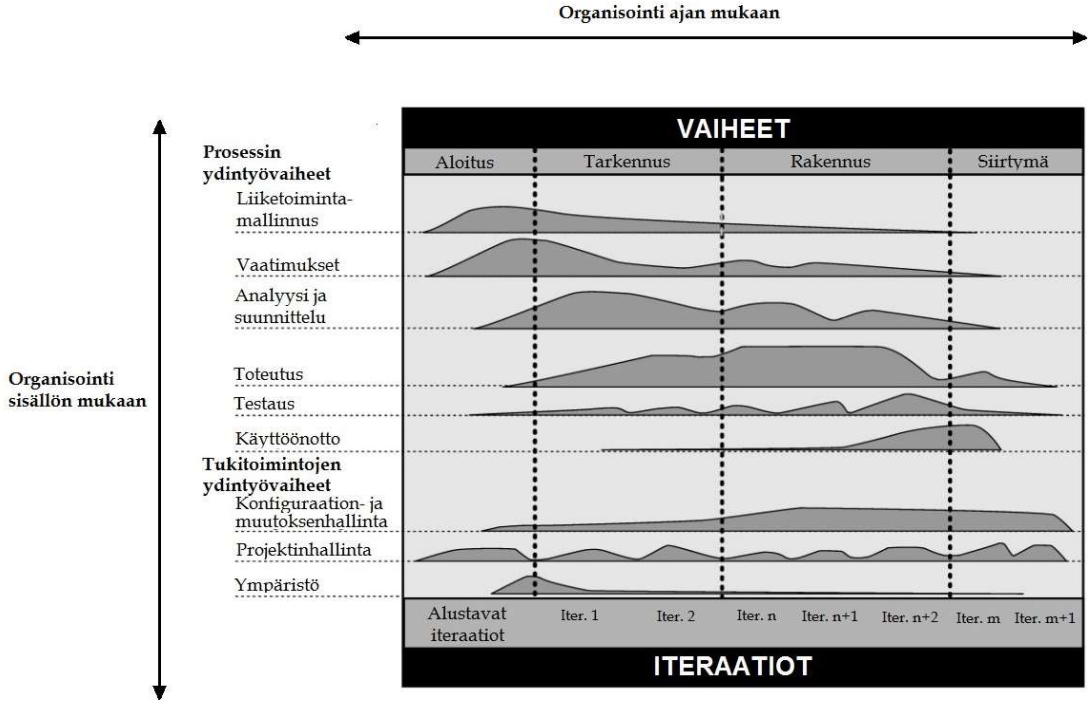


KUVIO 3 Vesiputousmalli (Sommerville, 2007, 66)

Vaatimusten analysoinnissa ja määrittelyssä etsitään tietojärjestelmän tarjoamat palvelut, tavoitteet ja rajoitteet yhteistyössä asiakkaan kanssa. Nämä tiedot muunnetaan vaatimuksiksi ja dokumentoidaan usein tarkalla tasolla jonkin standardin mukaisesti. Samalla varmistetaan, että tietojärjestelmä hyödyttää liiketoimintaa. *Järjestelmää ja ohjelmistoa suunniteltaessa* muunnetaan vaatimukset suunnitelmiksi laitteistoista ja ohjelmistoista. Nämä kuvataan järjestelmän arkkitehtuuriin. Ohjelmistosuunnittelussa tunnistetaan ohjelmiston tarpeelliset abstraktiot ja niiden väliset suhteet. *Toteutuksen ja yksikkötestauksen* tarkoitus on toteuttaa tietojärjestelmä vaatimusmäärittelyn ja suunnittelun mukaisesti. Yksikkötestauksen avulla varmistetaan ohjelmiston eri osien toimiminen suunnitellusti. *Integroinnissa ja järjestelmän testauksessa* kootaan ohjelmiston eri osat yhteen ja varmistetaan niiden yhteensopivuus. Tässä vaiheessa varmistetaan koko tietojärjestelmän toiminen asiakkaan kanssa sovittujen vaatimusten mukaisesti. Tietojärjestelmä myös toimitetaan asiakkaalle. *Operointi ja ylläpito* -vaihe on usein tietojärjestelmän elinkaaren pisin elinkaarivaihe. Siinä tietojärjestelmä asennetaan ja otetaan käyttöön, korjataan ilmaantuvat virheet ohjelmistossa sekä kehitetään ohjelmiston palveluja uusien vaatimusten ilmaantuessa. (Sommerville, 2007.)

Rational Unified Process (RUP) -kehys (Rational Software, 1998) lähtee ajatuksesta, jonka mukaan tietojärjestelmän kehittämistehtäviä suoritetaan tilannekohtaisesti eri tavalla. Se ei pyri kuvaamaan prosessia lineaarisena vaan kaksiulotteisena tehtävien joukkona (Kuvio 4). Vaaka-akseli esittää tietojärjestelmän kehittämisprosessin dynaamisen puolen: prosessin toteuttamisen ajan mukaan

(organization along time). Prosessin säädettävät osat ilmaistaan sykleinä (cycle), vaiheina (phase), virstanpylväinä (milestone) ja iteraatioina (iteration). Tietojärjestelmän elinkaari koostuu useasta kehitysvaiheesta, syklistä. Jokainen *sykli* jakaantuu neljään eri vaiheeseen: aloitus-, tarkennus-, rakennus- ja siirtymävaiheeseen. Jokaisella *vaiheella* on omat, tarkasti määrittelyt tavoitteensa, joita kutsutaan *virstanpylväiksi*. Vaiheet jaetaan yhteen tai useampaan iteraatioon. *Iteraatiossa* tuotetaan julkaisukelpoinen osa rakennettavasta tietojärjestelmästä eli inkrementti. Iteraation aikana suoritetaan tietojärjestelmän kehittämisprosessin kaikkia tehtäviä määrittelystä testaamiseen. (Rational Software, 1998.)



KUVIO 4 RUP-kehys (Rational Software, 1998, 3)

Pystyakseli esittää prosessin staattisen puolen: prosessin organisoinnin sisällön mukaan (organization along content). Prosessin sisältö kuvataan työntekijöittäin (worker), aktiviteetein (activity), artefaktein (artifact) ja työvaiheittain (workflow). *Työntekijät* ovat tietyssä roolissa työskenteleviä yksittäisiä tai ryhmä henkilöitä. Työntekijällä voi olla useita rooleja, joissa he suorittavat aktiviteettejä. *Aktiviteetit* ovat hallittavan kokoisia, konkreetteja tehtäviä, joiden lopputuotteena syntyy uusi tai muokattu artefakti. *Artefaktit* voivat olla erilaisia malleja, dokumentteja tai ohjelmakoodia. Työntekijöiden suorittamat aktiviteetit liittyvät erilaisiin *työnkulkuihin* (work flow), joiden tavoitteena on saavuttaa näkyvä lopputulos aktiviteettien kautta. Tietojärjestelmän kehittämisprosessin työnkulut jakaantuvat kuuteen eri ydintyövaiheeseen (disciplines) ja kolmeen tukityövaiheeseen (Kuvio 4). Ydintyövaiheita ovat liiketoiminnan mallintaminen, vaatimukset, analyysi ja suunnittu, toteutus, testaus ja käyttöönotto. (Rational Software, 1998.)

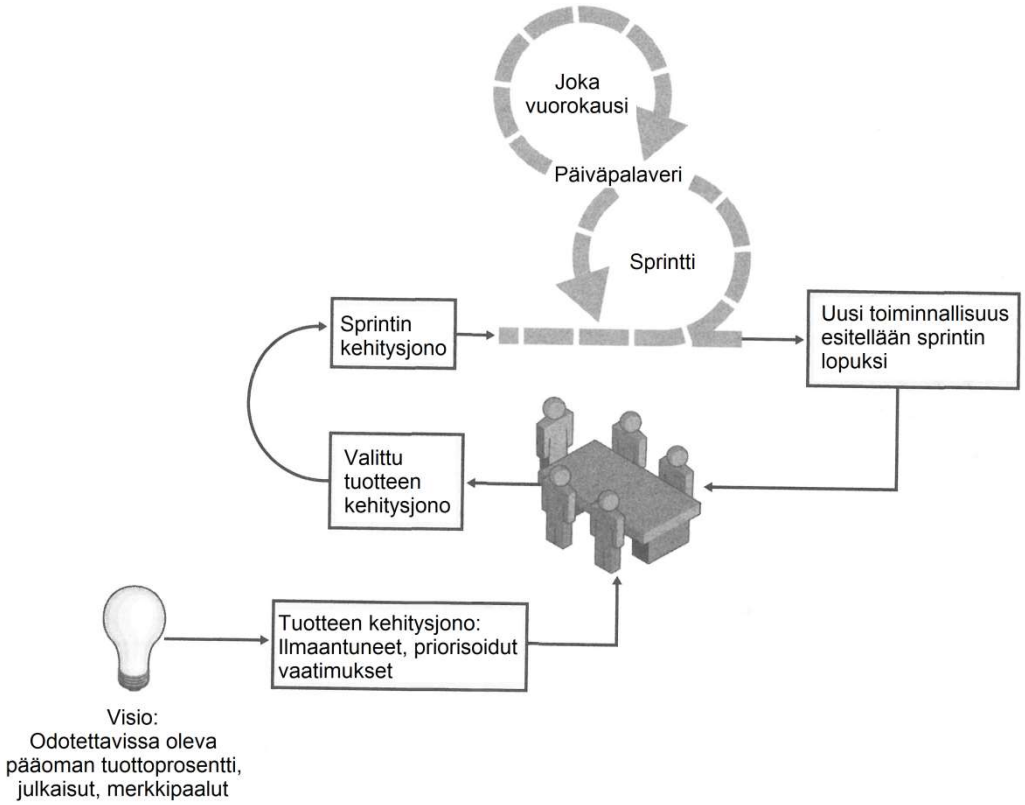
Vaatimustyövaiheen tavoitteena on sopia toteutettavista vaatimuksista asiakkaan ja kehittäjien kesken. Tämä sisältää vaatimusten löytämisen, organisoinnin ja dokumentoinnin. Sidosryhmien tarpeet kuvataan visiodokumenttiin. Visioon kuvataan myös järjestelmän toimijat, jotka kuvaavat käyttäjiä, sekä toimijoihin liittyvät järjestelmän toiminnot. Järjestelmän toiminnot kuvataan tarkemmalla tasolla käyttötapauskuvauksiin (use-case description). Käyttötapauksissa kerrotaan askel askeleelta, kuinka toimija on vuorovaikutuksessa järjestelmän kanssa. Järjestelmään liittyvät ei-toiminnalliset vaatimukset kuvataan täydentävät (supplementary) vaatimukset -dokumentissa. Työvaihe aloitetaan tietojärjestelmän kehitysprosessin alussa, ja se painottuu voimakkaasti kehittämisprosessin alkuvaiheisiin. Tarkennusvaiheen päätteeksi vaatimuksista tulisi olla 80 % määriteltynä. (Rational Software, 1998.)

Vaatimusmäärittely ketterissä menetelmissä eroaa vesiputousmalliin ja RUP-kehykseen nähden Laplanten (2009) mukaan erityisesti vaatimusmäärittelyn tehtävien ajoituksessa. Vaatimuksia ei kirjata tarkalla tasolla kehitysprosessin alussa, vaan alustavia vaatimuksia tarkennetaan koko prosessin ajan erityisesti ennen vaatimukset toteuttavan, uuden rakennettavan osan aloittamista. Uusia vaatimuksia löydetään tuotteelle koko järjestelmäkehitysprosessin ajan. (Laplante, 2009.)

Ketteristä menetelmistä tunnetuimpia ovat XP (Beck, 1999) ja Scrum (Schwaber, 2004; Schwaber & Sutherland, 2012). Scrum-menetelmässä kuvataan tarkemmin prosessi (Kuvio 5). Kehittäminen alkaa tuotevision tuottamisella. Tuotevisio (product vision) sisältää ajatuksen siitä, miten uusi tietojärjestelmä tuottaa arvoa. Tarkemmat ideat ja ajatukset kerätään vaatimuksiksi *tuotteen kehitysjonoon* (product backlog). Tuotteen kehitysjonossa olevista vaatimuksista valitaan vaatimukset rakennettavaan tuotteeseen. Kehitysjono järjestetään vaatimusten prioriteetin mukaan, ja siinä olevia vaatimuksia voidaan lisätä muokata ja poistaa koska tahansa tarpeen mukaan. (Schwaber & Sutherland, 2012.)

Tuotteen kehitysjonossa olevat vaatimukset toteutetaan *sprinteissä* (sprint). Ennen sprintin alkamista tuotteen kehitysjonossa olevia, korkeimmalle priorisoituja vaatimuksia tarkennetaan. Vaatimusten tulee olla niin tarkalla tasolla, että niistä voidaan valita olennaisimmat sprintissä toteutettavat asiat ja vaatimusten toteutus voidaan jakaa korkeintaan päivän mittaisiin tehtäviin. Nämä vaatimukset ja tehtävät kirjataan *sprintin kehitysjonoon* (sprint backlog). (Laplante, 2009.)

Vaatimusten tarkentaminen tehdään sprintin suunnittelukokouksessa. Suunnittelukokouksessa tuoteomistaja (product owner) esittelee tuotteen kehitysjonossa korkeimmalle priorisoituja vaatimuksia. Kehitystiimi kyselee tuoteomistajalta tarkentavia kysymyksiä sisällöstä, tarkoituksesta jne., kunnes he pystyvät valitsemaan sprintissä toteutettavat vaatimukset ja jakamaan vaatimuksen toteutuksen tehtäviksi. (Schwaber, 2004.)



KUVIO 5 Scrum-prosessin yleiskuva (Schwaber, 2004, 9)

2.4 Yhteenveto

Tietojärjestelmän kehityksessä käytetään erilaisia kehittämismenetelmiä kuten vaihejakomallia, RUP-kehystä ja Scrumia. Kaikissa tietojärjestelmän kehittämismenetelmissä esiintyy samoja vaatimusmäärittelyn aktiviteetteja, joita suoritetaan tietojärjestelmäkehityksen eri vaiheissa kehittämismenetelmästä riippuen. Niissä voidaan tunnistaa myös samat vaatimusmäärittelyvaiheet: kerääminen, analysointi, validointi, neuvottelu, dokumentointi ja hallinta. Menetelmät eroavat toisistaan siinä, missä kohdassa kehittämisprosessia vaatimusmäärittelytehtäviä suoritetaan ja miten.

3 DOKUMENTOINTI

Tässä luvussa esitetään, mitä dokumentoinnilla tarkoitetaan ja miten dokumentteja hyödynnetään. Lisäksi esitetään kaksi dokumenttityyppiä, prosessidokumentit ja tuotedokumentit (Sommerville, 2010b). Lopuksi tarkastellaan Pohlin (1994) vaatimusmäärittelyprosessin kolmesta ulottuvuudesta tarkemmin yhtä eli vaatimusten esittämistä. Vaatimusten esittäminen mallien ja dokumenttien avulla on tietojärjestelmän dokumentointia. Esittäminen kattaa kaiken, mitä käytetään rakennettavaa tietojärjestelmää koskevan tietämyksen esittämiseen. Luvussa keskitytään toiminnallisiin vaatimuksiin. Ensin esitetään luokittelu toiminnallisten vaatimusten esittämiseen ja esitellään esitystapoja sen mukaisesti. Lopuksi vertaillaan esitystapoja eri kriteerein määritellyistä näkökulmista.

3.1 Dokumentoinnin tarkoitus, tarve ja luonti

Dokumentointi on olennainen osa tietojärjestelmän kehittämistä ja ylläpitoa. Dokumentoinnin tarkoituksena on tallettaa välttämätöntä tietoa järjestelmästä pysyväisluonteisesti jollain yleisesti käytetyllä tavalla. Tietoja tarvitaan järjestelmäkehitys- ja ylläpitotyön tekemiseen ja tehdyn validointiin, tietojen jakamiseen ja asioista sopimiseen sekä muistin tueksi. Dokumentoinnista voi löytyä ratkaisu ristiriidoille, hiljaisen tiedon esiintymispaikka tai perustelu poikkeavalle ratkaisulle. (Vuori, 2010.)

Dokumentoinnilla tarkoitetaan tässä kaikkien varsinaista tietojärjestelmää kuvaavien dokumenttien lisäksi järjestelmän luomisen, muokkaamisen ja poistamisen prosesseihin liittyvien vaiheiden kuvauksia (Kajko-Mattsson, 2001). Dokumenttien luominen aloitetaan usein jo ennen varsinaisen tietojärjestelmäkehityksen aloittamista. Muun muassa ehdotus järjestelmän rakentamisesta käsitellään usein järjestelmää kuvaavien tai jopa kattavan vaatimusdokumentin avulla. Dokumenttien luomista jatketaan järjestelmän julkaisemiseen asti järjestelmän suunnittelu- ja toteutusratkaisujen tarkentuessa. Ohjelmistokehittäjät luovat suurimman osan dokumenteista. (Sommerville, 2010b.)

Tietojärjestelmään liittyvien dokumenttien luomiselle on useita syitä. Ambler (2002) esittelee dokumentoinnille neljä ylätason syytä:

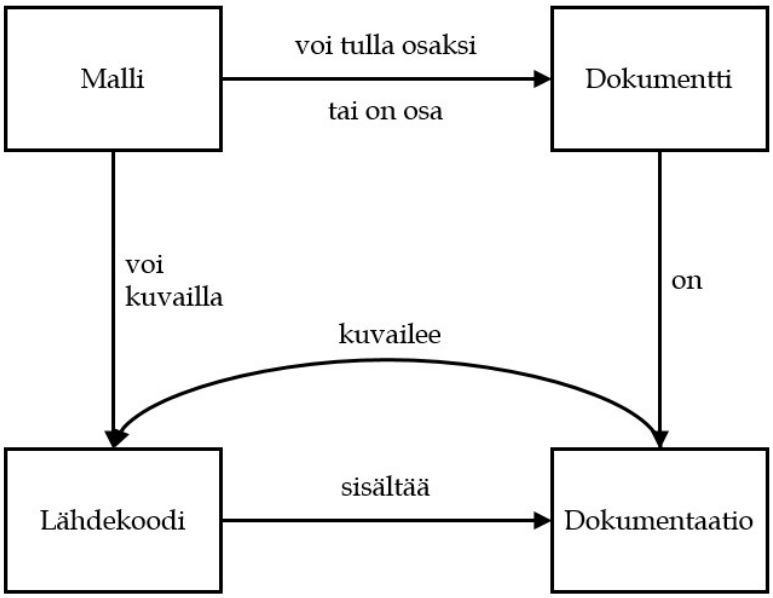
1. *Projektin sidosryhmät vaativat sitä*. Projekteilla on usein lukuisia sidosryhmiä aina ohjelmistoa käyttävistä asiakkaista ja järjestelmän ylläpitäjistä aina sitä operoiviin henkilöihin asti. Kaikkien sidosryhmien dokumentointitarpeet on otettava huomioon, ja "jonkun", joka usein toimii myös maksajan roolissa, on tehtävä päätös luotavista dokumenteista.
2. *Sopimusmallin määrittelemiseksi*. Sopimusmalli määrittää vuorovaikutuksen järjestelmän ja ulkoisen järjestelmän välillä. Vuorovaikutus voi olla yhden- tai kahdensuuntainen. Ulkoinen järjestelmä voi olla esimer-

kiksi tietokanta, tietopalvelu tai perinnejärjestelmä. Sopimusten osapuolien tulee päästä yhteisymmärrykseen sopimusmallista, dokumentoida se ja muuttaa ajan myötä, mikäli tarpeellista. Myös sopimusmallin tekemisestä päättää projektin sidosryhmä.

3. *Kommunikoinnin tukemiseksi ulkoisen ryhmän kanssa***.** Kommunikointi pelkästään dokumentoinnin avulla aiheuttaa väärinkäsityksiä, mutta sen sijaan kommunikoinnin tukena dokumentointi voi lisätä ymmärrystä. Tätä on hyvä hyödyntää silloin, kun kehitystiimin jäsenet eivät työskentele samassa paikassa tai tarvittavat sidosryhmät eivät ole koko ajan saatavilla. Dokumentointia käytetään usein ajoittaisten tapaamisten, videoneuvotteluiden, sähköpostin ja kommunikointivälineiden ohella.
4. *Jonkin asian läpikäymiseen*. Dokumentoimalla voi varmistaa ryhmätyönä tehtyä asiaa itsellensä tai lisätä omaa ymmärrystä jostain asiasta. Usein asioiden kirjoittaminen ”paperille” auttaa jäsentämään ajatuksia tai havaitsemaan ongelmia suunnitellussa ratkaisussa.

Tietojärjestelmäkehittäjät ja -ylläpitäjät käyttävät dokumentaatiota ymmärtääkseen monimutkaisia järjestelmiä, niiden toiminnallisuutta, korkean tason arkkitehtuuria sekä toteutusta. Ei ole kuitenkaan yksiselitteistä, minkä tyyppiset dokumentit ovat hyödyllisimpiä ja tarpeellisimpia tai kenen dokumentit pitäisi tehdä ja ylläpitää. Myös dokumenttien luomisen ajoitus voi vaihdella. (Thomas & Tilley, 2001.)

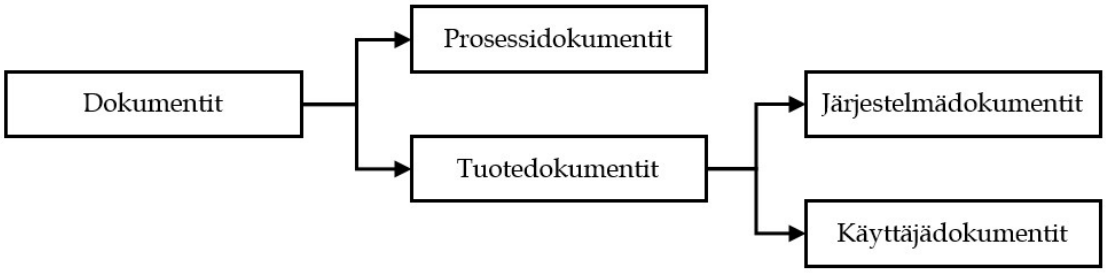
Ambler (2002) esittelee dokumenttien (document), mallien (model), dokumentaatio (documentation) ja lähdekoodin (source code) suhteita toisiinsa (Kuvio 6). *Dokumentilla* tarkoitetaan Amblerin esityksessä, mitä tahansa artefaktia lähdekoodia lukuun ottamatta, jonka tarkoituksena on ilmaista tietoa tietojärjestelmästä pysyvällä tavalla. *Malli* esittää abstraktion jollekin ongelmalle tai sen ratkaisulle yhdestä tai useammasta näkökulmasta. Usein malleja hyödynnetään ongelman tai sen ratkaisun ymmärtämiseen, jolloin ne hävitetään käytön jälkeen. Malleista voi kuitenkin muodostua itsenäisiä dokumentteja, tai ne voidaan sisällyttää osaksi dokumenttia. Malli voi kuvata myös lähdekoodia. *Lähdekoodilla* tarkoitetaan tietokonejärjestelmää ohjaavia käskysarjoja kommentteineen. *Dokumentaatio* koostuu dokumenteista sekä lähdekoodin kommenteista ja kuvaa lähdekoodista koostuvaa järjestelmää. (Ambler, 2002.)



KUVIO 6 Suhteet mallien, dokumenttien, dokumentaation ja lähdekoodin välillä (Ambler, 2002, 144).

3.2 Dokumenttityypit

Sommerville (2010b) jakaa dokumentit kahteen päätyyppiin, prosessidokumentteihin ja tuotedokumentteihin (kuvio 7). *Prosessidokumentit* kuvaavat kehitys- ja ylläpitoprosessin, ja ne sisältävät kaikki kehitykseen ja ylläpitoon liittyvät suunnitelmat, aikataulut, prosessinlaatudokumentit sekä organisaatioon ja projektin standardeihin liittyvät dokumentit. *Tuotedokumentit* sisältävät kehitteillä tai ylläpidossa olevan tuotteen järjestelmädokumentit ja käyttäjädokumentit. Järjestelmädokumentit kuvaavat tuotteen sovelluskehittäjien ja ylläpitäjien näkökulmasta ja käyttäjädokumentit käyttäjän näkökulmasta. (Sommerville, 2010b.)



KUVIO 7 Dokumenttityypit

Seuraavaksi tarkastellaan lähemmin ensin prosessidokumentteja ja sen jälkeen tuotedokumentteja.

3.2.1 Prosessidokumentit

Tehokas prosessijohtaminen vaatii Sommervillen (2010b) mukaan prosessin tekemistä näkyväksi. Koska ohjelmistoihin liittyvät prosessit ovat aineettomia, prosessien näkyvyys edellyttää prosessidokumentaation käyttöä. Prosessidokumentit voidaan luokitella viiteen alaluokkaan:

1. *Suunnitelmat, arviot ja aikataulut*. Johdon luomien dokumenttien avulla voidaan ennustaa ja kontrolloida ohjelmistoprosesseja.
2. *Raportit*. Raporttien avulla mitataan resurssien käyttöä kehitysprosessin aikana.
3. *Standardit*. Standardit määrittävät, miten prosessia noudatetaan (suora käännös täytäntöön pannaan). Ne voivat noudatella organisaationaalisia, kansallisia tai kansainvälisiä standardeja.
4. *Työdokumentit.* Projektissa työskentelevien sovelluskehittäjien ideoita ja ajatuksia sisältäviä dokumentteja, joita käytetään projektin kehitysvaiheessa apuna toteutusvaihtoehtojen tai esiintyneiden ongelmien ilmaisemiseen. Työdokumentit edeltävät varsinaista tuotedokumentaatiota, ja niistä löytyvät usein epäsuorasti perustelut suunnitteluratkaisuille.
5. *Sähköpostit, wikit jne*. Tallenteet jokapäiväisestä viestinnästä johtajien ja sovelluskehittäjien välillä.

Prosessidokumentaatiolle on ominaista, että se vanhenee nopeasti. Prosessidokumentaatiota tarvitaan harvemmin julkaisun jälkeen. Prosessidokumenteista voi olla kuitenkin hyötyä historiatietona, joten ne kannattaa säilyttää. Prosessidokumentaatiosta tulee myös siirtää tuotedokumentaatioon tarpeelliset asiat, kuten esimerkiksi suunnitteluratkaisuiden perustelut. Ketterien menetelmien periaatteet kehottavat minimoimaan prosessidokumentaation määrän, mutta tehtäessä projektia ulkoiselle asiakkaalle prosessidokumentaation määrässä tulee ottaa huomioon sopimukselliset asiat asiakkaan ja toimittajan välillä, sovelluskehittäjien työhön liittyvät riskit sekä mahdolliseen ulkoiseen sääntelyyn ja standardointiin liittyvät vaatimukset. (Sommerville, 2010b.)

3.2.2 Tuotedokumentit

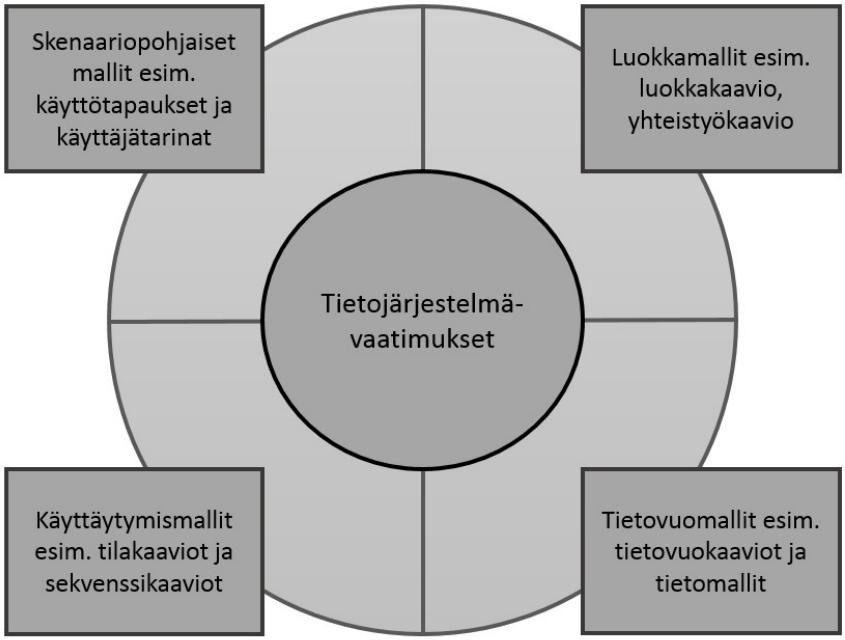
Tuotedokumentaation tarkoitus on kuvata tuotetta koko sen elinkaaren ajan ensimmäisestä julkaisusta lähtien. Se sisältää käyttäjädokumentaation, joka kertoo miten ohjelmistotuotetta käytetään, ja järjestelmädokumentaation, jonka pääasiallinen käyttäjäkunta on ylläpidosta vastaavat sovelluskehittäjät. *Käyttäjädokumentaatio* sisältää sekä loppukäyttäjille että järjestelmänvalvojille (system administrator) kohdistuvan dokumentaation. Se sisältää käyttäjien eri tehtävien ohjeistukset eritasoisille ja kokemuksen omaaville käyttäjille strukturoidusti. Loppukäyttäjille tarkoitettu dokumentaatio ohjeistaa käyttäjiä suorittamaan haluamansa tehtävät ohjelmiston avulla. Dokumentaatiosta löytyy usein hyödyllistä

tietoa niin järjestelmän uusille käyttäjille kuin sitä paljon käyttäneille. Dokumentaatio sisältää järjestelmän yleiskuvauksen lisäksi järjestelmän vaatimukset ja sen tarjoamien palveluiden kuvauksen. Järjestelmänvalvojat ovat vastuussa loppukäyttäjien käyttämän ohjelmiston hallinnoinnista. Järjestelmänvalvojille suunnattu dokumentaatio voi kattaa dokumentteja järjestelmän asennuksesta, tunnettuihin virhetilanteista, keskuskoneiden operoinnista, tietoverkoista, ohjelmiston ongelmia ratkovista ohjelmistokehittäjistä kuin yhteistyöstä käyttäjien tai ohjelmiston toimittajien kanssa. (Sommerville, 2010b.)

Järjestelmädokumentaatio sisältää kaikki järjestelmää kuvaavat dokumentit vaatimusmäärittelystä suunnittelun, toteutuksen ja testauksen kautta aina hyväksymistestaussuunnitelmaan asti. Järjestelmädokumentaatio on järjestelmän ylläpidon kannalta välttämätön. Hyvin strukturoitu järjestelmädokumentaatio lisää ymmärrettävyyttä ja ylläpidettävyyttä. Laajempien järjestelmien dokumentaation olisi hyvä sisältää dokumentit, jotka kuvaavat järjestelmän vaatimukset perusteluineen, järjestelmäarkkitehtuurin, järjestelmän jokaisen sovelluksen oman sovellusarkkitehtuurin järjestelmän kaikki komponentit toimintojen ja rajapintojen kuvauksineen sekä listauksen ohjelman lähdekoodeista, jonka tulee olla itsensä dokumentoivaa eli strukturoitua, hyvin nimettyä, kommentoitua ja monimutkaisemmat ratkaisut selitettyä. Järjestelmädokumentaation tulee kattaa myös jokaisen ohjelman validointikriteerit, jotka tulee olla linkitettynä niihin liittyviin vaatimukset. Lisäksi ohjelmiston ylläpidon opas sisältää tunnetut virheet, riippuvuudet laitteistoihin ja sovelluksiin sekä kuvauksen siitä, miten tehdyissä suunnitteluratkaisuissa on otettu huomioon ohjelmiston jatkokehitys. (Sommerville, 2010b.)

3.3 Vaatimusten esittäminen

Vaatimusten esittämiselle on esitetty erilaisia luokituksia. Pressman (2010) esittelee tietojärjestelmävaatimuksille jaon neljään eri mallityyppiin (Kuvio 8), skenaariopohjaisiin malleihin, käyttäytymismalleihin, luokkamalleihin ja tietovuomalleihin. *Skenaariopohjaisissa malleissa* (scenario-based models) vaatimukset esitetään eri toimijoiden näkökulmasta. Toimijat kuvataan usein käyttäjärooleina, jotka voivat olla henkilöiden lisäksi myös järjestelmiä. Skenaariopohjaisten mallien avulla kuvataan, kuinka käyttäjä haluaa toimia järjestelmän kannalta. Nämä toimivat tärkeänä lähtökohtana muiden mallien rakentamiselle. *Luokkamallien* (class models) avulla esitetään sovellusalueeseen liittyvät tiedot eri abstraktiotasoilla. Näitä tietoja käsitellään järjestelmässä. *Tietovuomallit* (flow-oriented models) esittelevät järjestelmän toiminnalliset yksiköt ja tietojen liikkumisen ja muutokset yksiköiden välillä. *Käyttäytymismallit* (behavioral models) kuvaavat tietojärjestelmän käyttäytymisen ikään kuin ulkopuolisten tapahtumien seurauksena. (Pressman, 2010.)



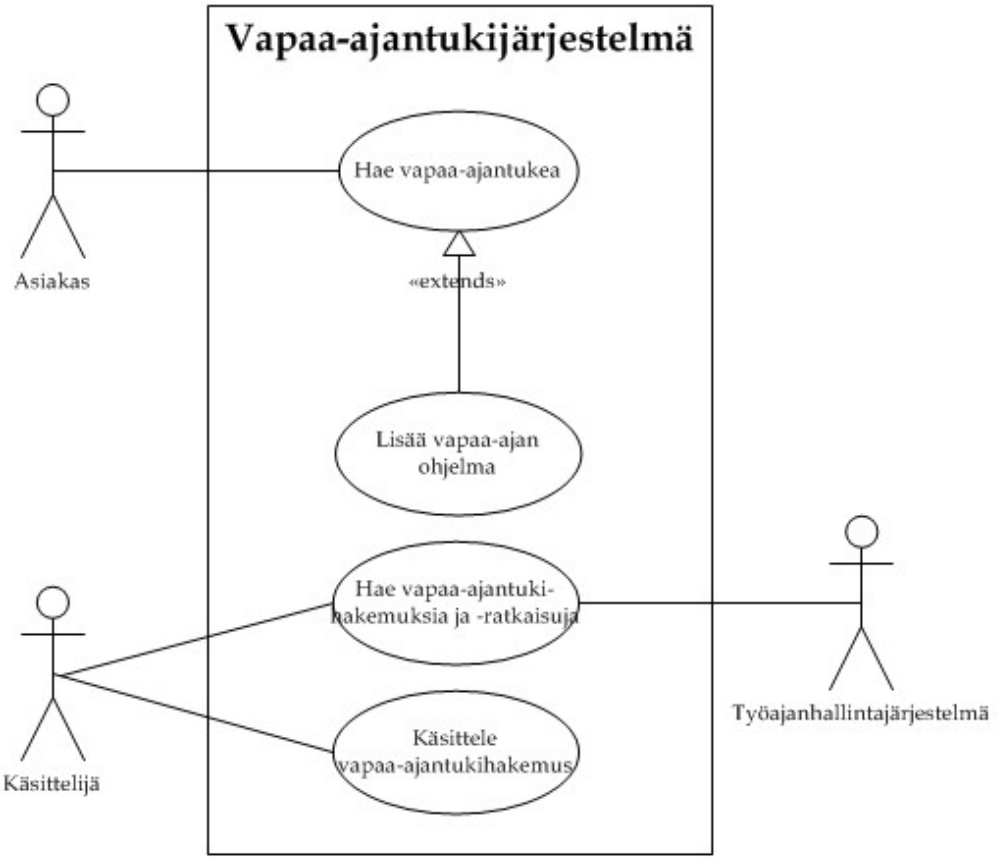
KUVIO 8 Vaatimusten nelijako (Pressman, 2010, 154)

Seuraavaksi esitellään erilaisia vaatimusten esittämistapoja tämän luokituksen mukaisessa järjestyksessä. Lopuksi vertaillaan esittämistapoja toisiinsa.

3.3.1 Skenaariopohjaisia esitystapoja

Skenaariopohjaisissa esitystavoissa kuvataan järjestelmän käyttöskenaarioita. Käyttöskenaario voidaan esittää tavoitehierarkiana, jolloin se kuvaa järjestelmän sidosryhmille tarjoamat kyvykkyydet kuvaamatta, miten järjestelmä tarjoaa ne (Hull, Jackson & Dick, 2011). Käyttöskenaarioita voidaan kuvata mm. käyttötapauksina tai käyttäjätarinoina.

Käyttötapauskaavio (use case diagram) kuvaa tietojärjestelmään liittyvät käyttöskenaariot otsikkotasolla (Kuvio 9). Kaaviossa kuvataan järjestelmän ulkopuoliset toimijat eli käyttäjät (actors). Käyttäjä on henkilö tai toinen tietojärjestelmä toimien tietyssä roolissa. Jokaiseen käyttäjään liitetään ne toiminnot eli käyttötapaukset (use case), joita käyttäjän on mahdollista suorittaa. Käyttötapaukseen voi liittyä käyttötapausta laajentava alikäyttötapaus, joka voi sisältyä aina toiminnon suorittamiseen tai olla valinnainen. (van Lamsweerde, 2009.)



KUVIO 9 Esimerkki käyttötapauskaaviosta

Käyttötapauskaaviossa käyttäjät kuvataan ”tikku-ukkoina”, jotka nimetään sen mukaan, missä roolissa kukin käyttäjä toimii. Jokaiseen käyttäjään liittyy aina vähintään yksi käyttötapaus, mutta käyttäjiä voi olla myös useita. Käyttötapaus kuvataan ellipsinä sisältäen sen toiminnon eli käyttötapauksen nimen. Käyttäjien ja käyttötapausten välinen suhde kuvataan viivoin. Itse tietojärjestelmää kuvaa suorakulmio, jonka sisällä on tietojärjestelmän avulla suoritettavat toiminnot. Käyttäjät kuvataan tietojärjestelmän ulkopuolelle. (Laplante, 2009.)

Kuviossa 9 esitetään vapaa-ajantukijärjestelmään liittyvät käyttäjät ja käyttötapaukset. Vapaa-ajantukijärjestelmän avulla asiakas voi hakea vapaa-ajantukea, johon voi liittyä vapaa-ajan ohjelman lisääminen. Käsittelijä voi järjestelmän avulla hakea vapaa-ajantukihakemuksia ja -ratkaisuja sekä käsitellä hakemuksia. Myös työajanhallintajärjestelmä voi hakea tehtyjä hakemuksia ja ratkaisuja.

Haikala ja Mikkonen (2011) esittelevät käyttötapauskaavioon kaksi käyttötapausten välistä suhdetta. Käyttötapaus voi *sisältyä* (include) toiseen käyttötapaukseen, jolloin suoritettaessa ensimmäinen käyttötapaus suoritetaan myös toinen. Käyttötapaus voi myös *laajentaa* (extend) toista käyttötapausta. Laajentavan käyttötapauksen suorittaminen toisen käyttötapauksen yhteydessä ei ole pakollista. Sisällytettävä ja laajentava käyttötapaus kuvataan katkoviivana, jossa nuoli

osoittaa toiseen käyttötapaukseen. Käyttötapausten välinen suhteen tyyppi, laajentava tai sisällytettävä, kuvataan tekstinä. (Haikala & Mikkonen, 2011.)

Käyttötapauskaavio ei kuvaa käyttäjän ja järjestelmän välistä vuorovaikutusta tarkalla tasolla, vaan tarkempi taso kuvataan käyttötapauksissa. *Käyttötapaus* (use case) kuvaa tietyn käyttöskenaarion käyttäjän näkökulmasta. Käyttöskenaariossa kuvataan askel askeleelta, kuinka käyttäjä on vuorovaikutuksessa tietojärjestelmän kanssa. Käyttäjä on useimmiten henkilö, mutta se voi olla myös toinen järjestelmä tai laitteiston osa. Käyttötapaus kuvaa usein käyttöskenaarion askeleiden lisäksi myös mahdolliset poikkeukset askeleisiin. Lisätietoina voidaan antaa myös käyttötapauksen skenaarion aloittavat tekijät, esiehdot, riskit, kriittisyys, tärkeys ja esiintymistiheys. Käyttötapaukseen voi liittyä käyttöskenaariota laajentavia käyttötapauksia, jotka toteutuvat aina tai vaihtoehtoisesti tietyin ehdoin. (Goldsmith, 2004.) Kuviossa 10 on esitetty käyttötapaus, joka kuvaa skenaarion, jossa asiakas hakee vapaa-ajantukea käyttäen järjestelmää. Tunnistauduttuaan asiakas suorittaa yksittäisiä askeleita, joiden lopputulemana asiakkaan tukihakemus on tallennettu. Käyttötapauksessa kuvataan myös, mitä yleisimmästä tapahtumien kulusta poikkeavia askeleita asiakas voi suorittaa päästäkseen haluttuun lopputulokseen.

Esiehdot: Asiakas on kirjautunut sisään verkkopankkitunnusten avulla, ja pankkijärjestelmä on välittänyt asiakkaan henkilötunnuksen ja koko nimen.
Normaali tapahtumien kulku:

1. Järjestelmä näyttää asiakkaan nimen ja syntymäajan.
2. Asiakas syöttää osoitteensa, puhelinnumeronsa sekä sähköpostiosoitteen.
3. Asiakas syöttää ajan, jolle hän hakee tukea.
4. Asiakas valitsee hakevansa rahallista tukea vapaa-aikaan, kirjoittaa perustelut rahallisen tuen tarpeelle sekä syöttää hakemansa rahamäärän.
5. Asiakas valitsee tallennustoiminnon, ja järjestelmä tallentaa hakemuksen.
6. Käyttötapaus päättyy

Vaihtoehtoiset tapahtumien kulut:
Normaalin tapahtuman kulun kohdissa 2, 3 ja 4 järjestelmän havaitessa virheitä asiakkaan syöttämissä tiedossa järjestelmä näyttää asiakkaalle virheilmoitukset, jotka tämä korjaa.
Normaalin tapahtuman kulun kohdassa 4 asiakas voi valita hakevansa vapaa-ajalleen ajallista tukea. Asiakas kirjoittaa perustelut ajallisen tuen tarpeelle sekä syöttää hakemansa vapaa-ajan määrän. Lisäksi asiakas suorittaa käyttötapauksen Lisää vapaa-ajan ohjelma.
Esiintymistiheys: Käyttötapaus toteutuu noin 10 kertaa tunnissa klo 8–16 välillä.

KUVIO 10 Esimerkki käyttötapauksesta

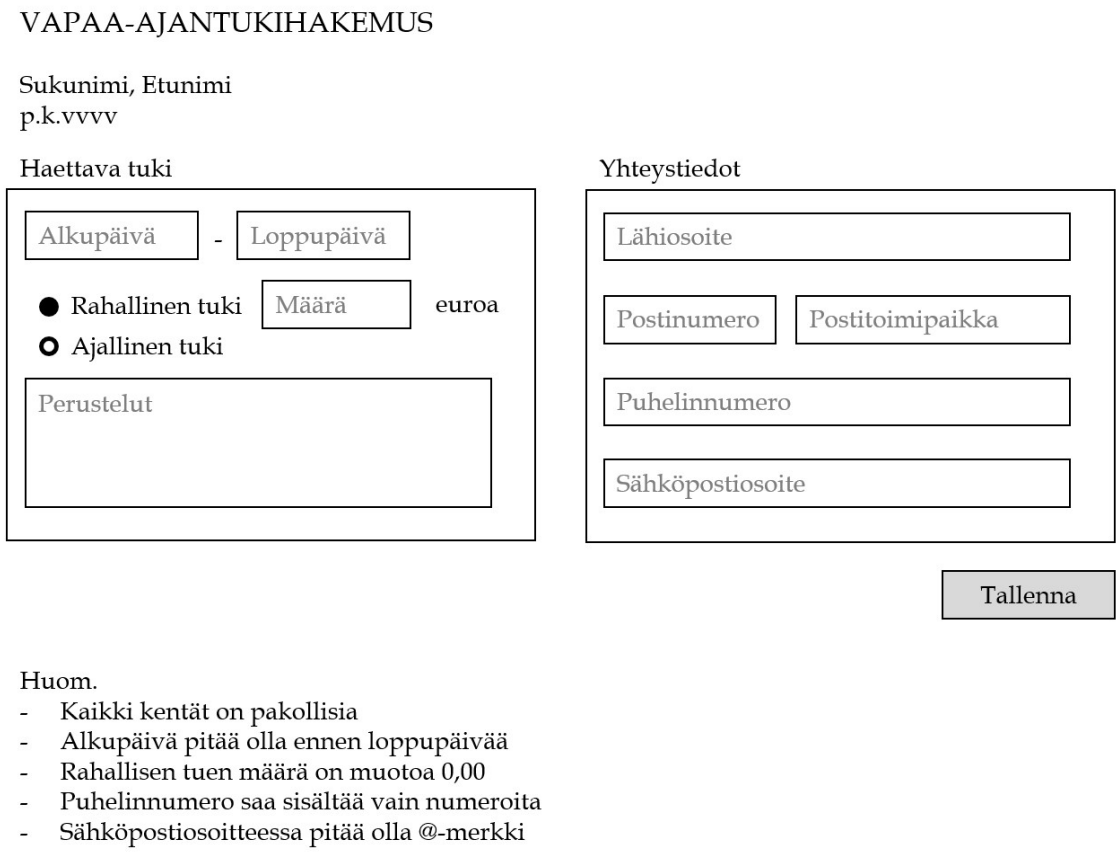
Alexander (2002) esittää käyttötapauksiin uuden näkökulman: järjestelmän negatiivisen käytön. Järjestelmän negatiivinen käyttö kuvataan *väärinkäyttötapauksina* (misuse case). Väärinkäyttötapauksen toimija on vihamielinen käyttäjä. Väärinkäyttötapaus kuvaa käyttäjän toimia tämän yrittäessä vahingoittaa järjestel-

mää. Väärinkäyttötapauksia käytetään erityisesti tietoturvavaatimuksia kerätessä ja analysoitaessa. Väärinkäyttötapaus kuvataan käyttötapauskaaviossa mustataustaisella ellipsillä.

Ketterissä menetelmissä, kuten XP:ssä ja Scrumissa, kirjoitetaan tarinoita (story) tai käyttäjätarinoita (user story). Vaatimussanaa ei käytetä, sillä se pakottaa sisällyttämään kaiken rakennettavaan tietojärjestelmään sellaisenaan. Tarinoiden tavoitteena onkin toimia keskustelun pohjana. Hyvä *tarina* on asiakkaan ja kehitystiimin luonnollisella kielellä kirjoittama ja molempien osapuolten ymmärtämä lyhyt ja ytimekäs lausahdus järjestelmän ominaisuudesta, joka tuottaa käyttäjilleen arvoa. Tarinat kirjoitetaan pienille muistilapuille, jotka sijoitetaan seinälle. Muistilaput heitetään pois tarinan valmistumisen jälkeen. (Leffingwell, 2007.)

Scrumissa käytetään käyttäjätarinoita (user story). *Käyttäjätarina* kuvaa tietyn toiminnon, jonka tietty käyttäjä suorittaa saavuttaakseni tietyn lisäarvon liiketoiminnalle. Toisin kuin XP:n tarinoilla Scrumin käyttäjätarinoilla on tietty muoto. Käyttäjätarinoiden muoto on seuraava: <Käyttäjänä> haluan <toiminto> niin, että saan <liiketoiminta-arvo>. Käyttäjätarinan ominaisuuksina kerrotaan lisäksi käyttäjätarinan hyväksymiskriteerit, prioriteetti ja työmäärä. Hyväksymiskriteerit sisältävät toimintoon liittyvä tarkempia määrityksiä, esimerkiksi käyttöliittymän ominaisuuksia. Käyttäjätarina ja hyväksymiskriteerit kirjoitetaan vapaalla luonnollisella kielellä. (Saddington, 2013.)

Skenaariopohjaisia esitystapoja voidaan täydentää eri tasoisilla käyttöliittymää koskevilla kuvauksilla. Uutta ohjelmistoa suunniteltaessa yksi ensimmäisistä käyttöliittymää koskevista dokumenteista on käyttöliittymän rautalankamalli (wireframe). Puertan, Michelettin ja Makin (2005) mukaan *rautalankamalli* on yksinkertainen, selitystekstein varustettu kuvaus elementeistä, jotka tulee toteuttaa internet-sivulle tai työpöytäohjelman näytölle. Rautalankamallit ei ole tarkoitettu käyttöliittymän tarkaksi kuvaukseksi, vaan ne on tarkoitettu karkean tason kuviksi. Evans, Sherin ja Lee (2013) tarkentavat rautalankamallin sisältävän tietoja käyttöliittymän rakenteesta, sisällöstä ja toiminnallisuudesta ottamatta liian tarkkaan kantaa ulkoasuseikkoihin kuten väreihin tai fontteihin. Kuviossa 11 esitetään rautalankamalli, joka perustuu kuviossa 10 esitettyyn käyttötapaukseen.



KUVIO 11 Esimerkki rautalankamallista

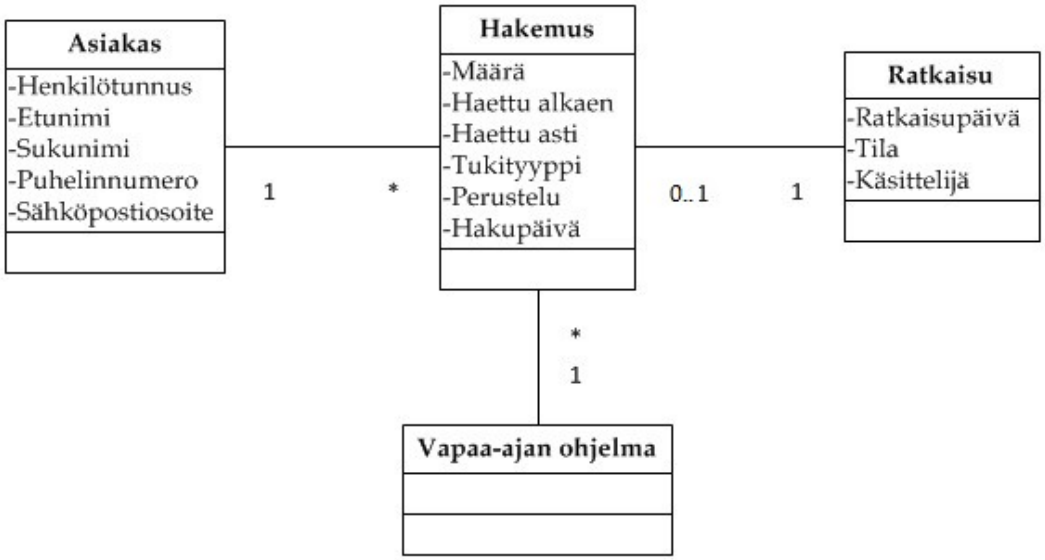
Rautalankamallia tarkempi käyttöliittymän kuvaus on prototyyppi. *Prototyyppi* on Arnowitzin, Arentin ja Bergerin (2007) mukaan hahmotelma, jonka tarkoituksena realisoida ohjelmisto tai ohjelmiston osa. Prototyypillä voidaan kuvata vuorovaikutteisuutta, navigointia, tietosisältöä, ulkoasua, suorituskykyä tai vaatimuksia. Vaatimukset voivat olla niin liiketoimintavaatimuksia, teknisiä vaatimuksia, toiminnallisia vaatimuksia, käyttäjävaatimuksia tai edellä mainittujen yhdistelmiä. Prototyyppiä voidaan hyödyntää vaatimusmäärittelyssä sekä ohjelmiston tai tuotteen dokumentoinnissa toiminnallisten kuvausten, kuten käyttötapausten, ohella. (Arnowitz, Arent & Berger, 2007.)

Rautalankamalli tai prototyyppi voivat olla osa tarkempaa kuvausta, käyttöliittymäkuvausta, tai niitä voidaan hyödyntää käyttöliittymäkuvauksen tekemisessä. *Käyttöliittymäkuvaus* (user interface specification) kattaa kolme osa-aluetta. Ensimmäinen osa-alue on käyttöliittymän objektien ulkoasu, kuten fontit ja värit. Toiseksi kuvataan käyttäjän ja käyttöliittymän väliseen dialogiin liittyvät säännöt, kuten mitä tapahtuu esimerkiksi painettaessa Tallenna-painiketta. Kolmanneksi kuvataan jokainen yksittäinen toiminto, esimerkiksi valintalistan sisältö. (Yip & Robson, 1991.)

3.3.2 Luokkamallipohjaisia esitystapoja

Luokkakaavio (class diagram) pyrkii kuvaamaan todellisen maailman järjestelmään liittyviä tietotyyppejä eli entiteettejä sekä niiden välisiä suhteita. Entiteetit ovat yksilöitävissä olevia järjestelmän kohteita, joilla on niitä kuvaavia *ominaisuuksia* (attribute). Entiteettien välillä on *suhteita* (relationship), jotka voivat olla esimerkiksi yhdestä yhteen tai yhdestä moneen. Entiteetit voivat olla myös toistensa alientiteettejä. Lisäksi luokkakaaviossa kuvataan entiteettien *operaatiot* (operation). Luokkakaaviossa entiteetit kuvataan kolmiosaisina suorakulmioina, joista ylimmässä on entiteetin nimi, keskimmäisessä sen ominaisuudet ja alimmassa sen operaatiot. Suhteet kuvataan entiteettien välisinä viivoina. Suhteen roolia voidaan täydentää suhteeseen liittyvällä tekstillä. (Hull, 2011.)

Kuviossa 12 on esitetty vapaa-ajantukijärjestelmään liittyvät entiteetit, asiakas, hakemus, ratkaisu ja vapaa-ajan ohjelma, niiden attribuutit ja entiteettien väliset suhteet. Asiakas on voinut tehdä yhden tai useamman hakemuksen, tai olla tekemättä lainkaan. Yksi hakemus liittyy aina yhteen asiakkaaseen. Hakemusta voi täydentää vapaa-ajan ohjelma. Hakemukseen voi liittyä enintään yksi ratkaisu, mutta ei välttämättä yhtään.



KUVIO 12 Esimerkki luokkakaaviosta

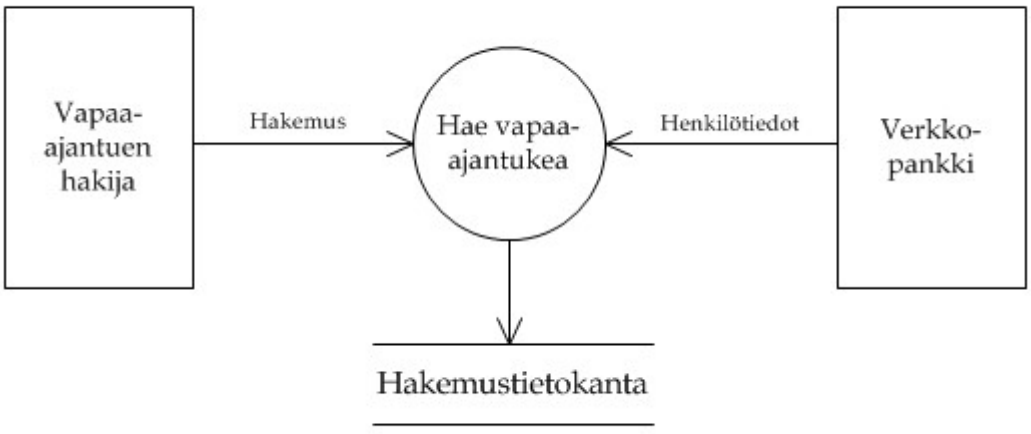
3.3.3 Tietovuopohjaisia esitystapoja

Tietovuokaavio (data flow diagram) mallintaa järjestelmässä liikkuvat tiedot ja niiden keskinäisen vuorovaikutuksen. Kaaviossa kuvataan ulkopuoliset entiteetit, prosessit, tietovuot ja tietovarastot. Kaaviossa keskitytään usein kontekstitasoisiin kaavioihin, joissa järjestelmää käsitellään ikään kuin mustana laatikkona

(black box). Tietovuomallintamisen seuraava vaihe on tietovuokaaviossa esitetyn järjestelmän jakaminen yksityiskohtaisempiin komponentteihin, joista jokainen on mahdollinen osajärjestelmä. (Kotonya & Sommerville, 1997.)

Tietovuokaaviossa liikkuva tieto kuvataan nimellä varustettuna nuolena. Tietojen transformaatio kuvataan ympyränä, joka on myös nimetty. Tietojen transformaatioita käytetään jaettaessa järjestelmä komponentteihin. *Terminaattoreiksi* (terminator) kutsuttavia tietojen lähteitä ja kohteita kuvataan suorakulmioina. Tietojen staattisiin varastoihin, kuten tietovarastoihin, tarkoitettuja kohteita kuvataan kahdella horisontaalisella viivalla, joiden sisällä on tietovarastoa kuvaava nimi. (Kotonya & Sommerville, 1997.)

Kuvion 13 tietovuokaaviossa vapaa-ajantuen hakija -terminaattori on tietojen lähde hakemukselle. Verkkopankki-terminaattori lähettää asiakkaan henkilötiedot. Nämä tiedot transformoidaan ja tallennetaan tietovarastoon eli hakemustietokantaan.



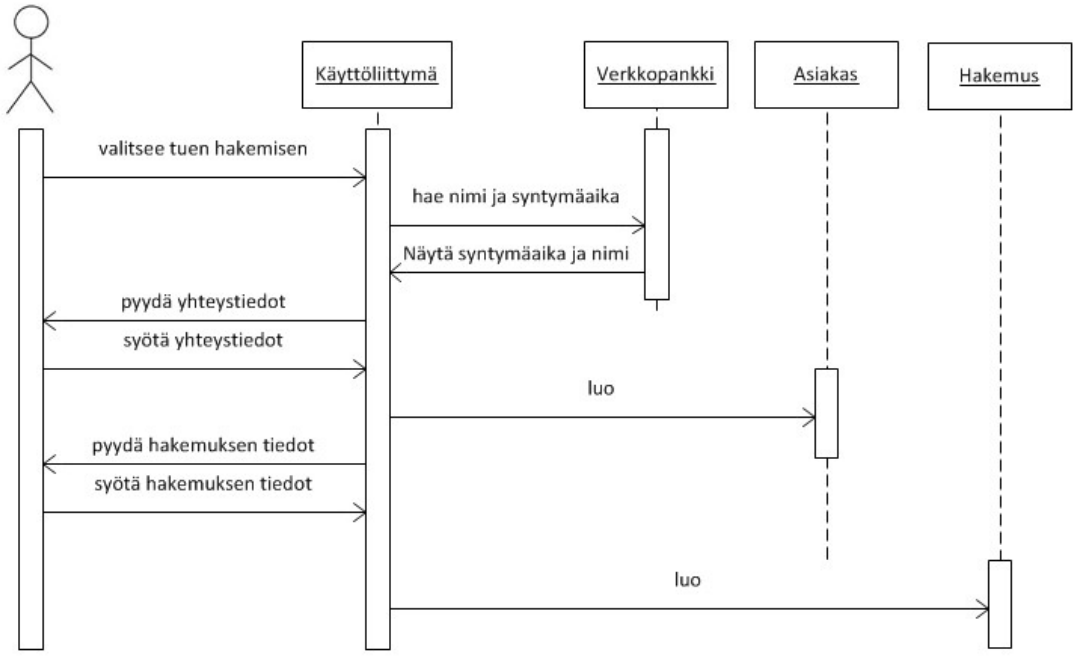
KUVIO 13 Esimerkki kontekstitason tietovuokaaviosta

3.3.4 Käyttäytymismallipohjaisia esitystapoja

Sekvenssikaavio (sequence diagram) kuvaa jonkin tapahtuman aiheuttaman vuorovaikutuksen objektien välillä. Sekvenssikaavion avulla voidaan mallintaa esimerkiksi käyttötapauksen toteutuminen olennaisimpien objektien avulla aikajärjestyksessä. Sekvenssikaaviossa aikaa kuvataan vertikaalisuunnassa, kun taas tapahtumaan liittyvät, objektien väliset vuorovaikutukset tapahtuvat vaakasuunnassa. Objektit kuvataan kaavion yläosassa nimettyinä suorakulmioina. Ensimmäinen objekti on usein tapahtuman käynnistäjä ja sama kuin käyttötapauksen käyttäjä. Objektien alapuolelle piirrettävän katkoviivan päällä sijaitsevilla kapeilla, vertikaalisilla suorakulmioilla kuvataan aikaa, jona objekti suorittaa tietyn aktiviteetin. Leveämmillä suorakulmioilla voidaan kuvata objektin tilaa. Aktiviteetistä tai tilasta alkava viesti kuvataan yhdensuuntaisilla nuolilla päättyen toisen objektin aktiviteettiin tai tilaan. Viestejä aktiviteettien välillä täydennetään

kuvaavin otsikoin ja/tai ehdoin. Sekvenssikaaviossa käytettyjä objektien tilojen kuvauksia voidaan täydentää myös erillisellä tilakaaviolla (state machine diagram). Tilakaaviota käytetään myös irrallisena kaaviona. (Pressman, 2010.)

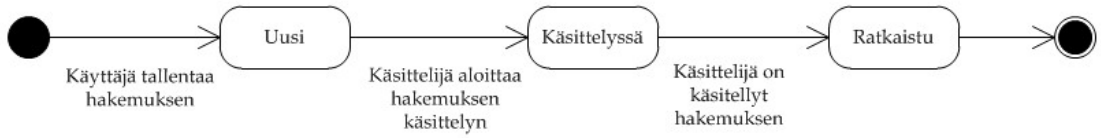
Kuviossa 14 kuvataan sekvenssikaaviona vapaa-ajantuen hakemisen toteutuminen sen olennaisimpien objektien avulla. Näitä ovat käyttöliittymä, verkkopankki, asiakas ja hakemus. Järjestelmän tärkeimpien objektien lisäksi sekvenssikaaviossa kuvataan vapaa-ajantuen hakemiseen liittyvän käyttötapauksen toimija eli käyttäjä. Käyttäjä valitsee tuen hakemisen. Käyttöliittymä hakee verkkopankista asiakkaan nimen ja syntymäajan. Käyttöliittymä pyytää käyttäjää syöttämään yhteystietonsa. Asiakas syöttää tiedot ja luodaan Asiakas-objekti. Tämän jälkeen käyttöliittymä pyytää käyttäjää syöttämään vapaa-ajantukihakemuksen tiedot, jotka käyttäjä syöttää. Tämän jälkeen luodaan Hakemus-objekti käyttäjän antamin tiedoin.



KUVIO 14 Esimerkki sekvenssikaaviosta

Tilakaavion (state diagram) avulla kuvataan järjestelmän jonkin komponentin hyväksyttävä käyttäytyminen. Käyttäytyminen esitetään komponentin hallinnoimien kohteiden tilasiirtymien sarjana. Ensimmäinen tilasiirtymä lähtee alkupisteestä, joka kuvataan ympyränä. Tilasiirtymä kuvataan yhdensuuntaisella nuolella, ja sen yhteyteen lisätään usein tilasiirtymää kuvaava otsikko. Tilasiirtymä voi olla myös ehdollinen. Ehdollista tilasiirtymää kuvataan yhdensuuntaisella nuolella, jonka yhteydessä ehto kuvataan hakasulkein. Tilasiirtymät johtavat johonkin komponentin hyväksytyistä tiloista, jotka kuvataan suorakulmioina pyöristetyin kulmin. Komponentin hyväksyttävä käyttäytyminen päättyy tilakaaviossa päätepisteeseen, jota kuvataan ympyrällä ääriviivan sisällä. (van Lamsweerde, 2011.)

Kuviossa 15 esitetään hakemuksen tilat yksinkertaisena tilakaaviona. Käyttäjän tallennettua hakemuksen on hakemuksen tila uusi. Tilaksi muuttuu käsittelyssä, kun käsittelijä aloittaa hakemuksen käsittelyn. Käsittelyn jälkeen hakemuksen tila on ratkaistu.



KUVIO 15 Esimerkki yksinkertaisesta tilakaaviosta

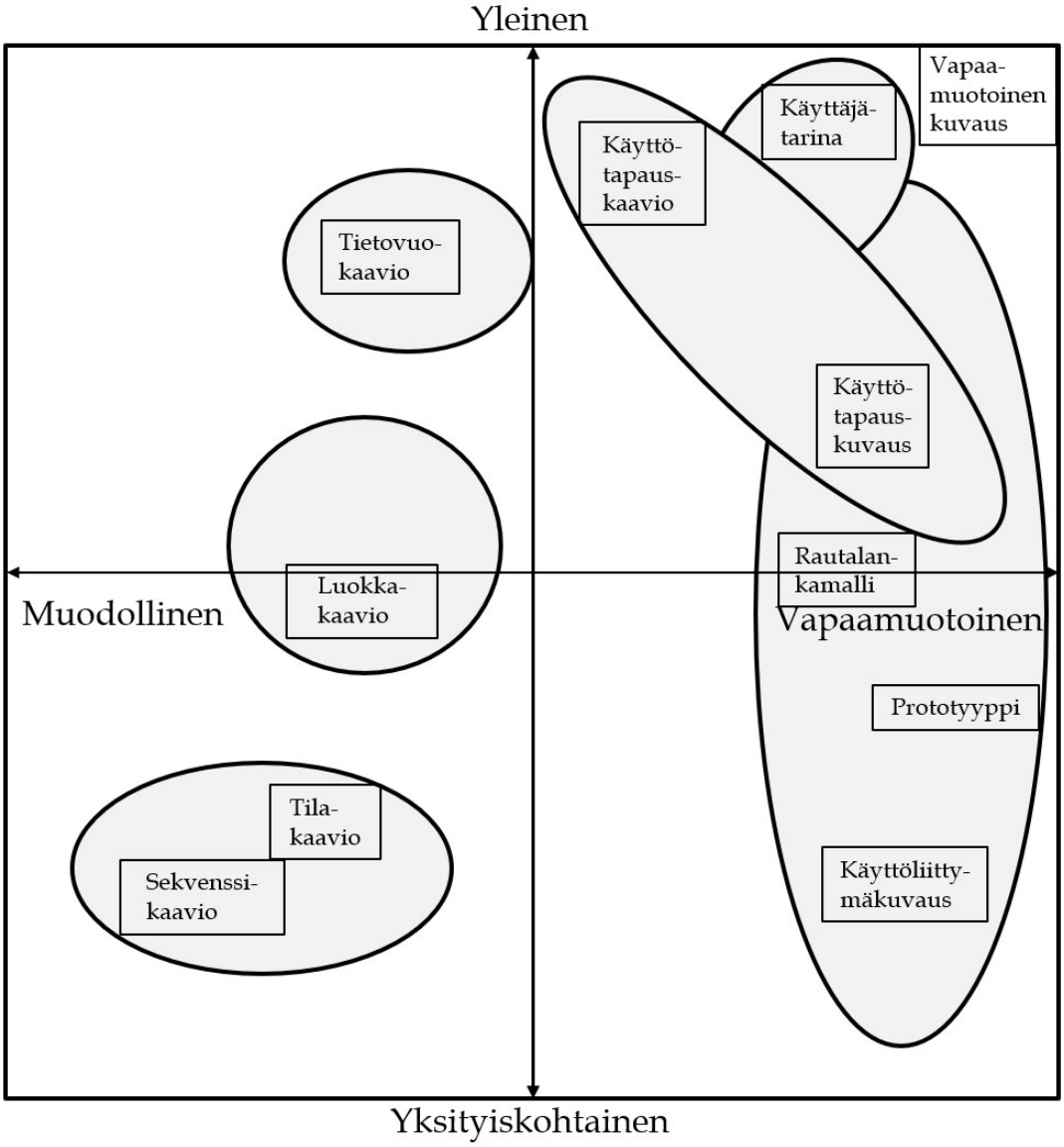
3.3.5 Vertailu

Erilaisia vaatimusten esittämistapoja on suuri määrä. Yksi syy tähän on niiden erilaiset käyttötarkoitukset. Haikala ja Mikkonen (2011) esittävät kolme eri tasoa vaatimuksille. Ensimmäisellä tasolla ovat *asiakasvaatimukset* (customer requirements), jotka ovat lähtöisin asiakkaalta. Asiakasvaatimukset kattavat kaikki vaatimukset, joita asiakas toivoo järjestelmältä. Asiakasvaatimuksista johdetaan tietojärjestelmää koskevat vaatimukset, *ohjelmistovaatimukset* (software requirements). Ohjelmistovaatimukset ovat konkreetteja vaatimuksia, joiden perusteella voidaan saavuttaa ymmärrys kyseessä olevaan järjestelmään toteutettavasta toiminnallisuudesta. Ohjelmistovaatimuksista johdetaan *tekniset vaatimukset* (technical requirements). Tekniset vaatimukset kuvaavat teknisestä näkökulmasta, mitä tietojärjestelmän toiminnallisuuksia on toteutettava, jotta ohjelmistovaatimukset täyttyvät. Jotkut asiakasvaatimukset voivat olla myös suoraan kelpoja teknisiä vaatimuksia. (Haikala & Mikkonen, 2011.)

Näiden kolmen eri tason vaatimuksia voidaan esittää erilaisin tavoin. Riippuen vaatimustasosta ja esittämistavasta vaatimuksen esitys vaihtelee yleisestä yksityiskohtaiseen. Eri vaatimusten esitykset vaihtelevat myös formaalisuudeltaan. Formaalit vaatimusten esittämistavat ovat tarkasti määriteltyjä, kun taas vapaamuotoiset esittämistavat voivat olla hyvin erilaisia ja monimuotoisia. Kuviossa 16 on sijoitettu tässä luvussa esitetyt esittämistavat formaalisuuden ja yleisyyden mukaan vaatimusten esittämistapojen nelikentäksi. Esittämistavat on esitetty kuviossa nimetyin suorakulmioin. Esittämistapojen yleisyyden ja formaalisuuden variaatiot on esitetty ellipsein. Formaalisuus on sijoitettu vaaka-akselille ja yleisyys pystyakselille.

Kuviossa 16 esitetyn nelikentän oikeaan yläneljänneksen ylänurkkaan sijoittuu vapaamuotoinen kuvaus. Se on tyypiltään yleisluonteinen ja vapaamuotoinen, koska esittämistavalla ei ole rajoittavia tekijöitä. Tällä esittämistavalla kuvataan usein tietojärjestelmään liittyviä karkean tason vaatimuksia. Näitä vaatimuksia voidaan kuvata myös käyttäjätarinoilla. Käyttäjätarinat ovat vapaamuotoista kuvausta formaalimpia, sillä käyttäjätarinat noudattavat tiettyä syntaksia. Myös käyttötapauskaaviolla voidaan kuvata karkean tason vaatimuksia, mutta

sen syntaksi on käyttäjätarinan syntaksiakin tarkempi. Molemmat, sekä käyttäjätarina että käyttötapauskaavio, ovat kuitenkin yleisluontoisia, ja niillä usein ilmaistaan järjestelmän karkean tason vaatimuksia. Käyttötapauskaaviossa esitettyjä vaatimuksia voidaan kuvata tarkemmin käyttötapauksissa, jotka ovat yksityiskohtaisempia. Käyttötapaus on kuitenkin käyttötapauskaaviota vapaamuotoisempi. Käyttötapauksen tarkka formaatti voidaan esimerkiksi määritellä organisaatio- tai järjestelmäkohtaisesti. Tietovuokaavio sijaitsee vasemmassa yläneljänneksessä. Tietovuokaavio on esitystapana puoliformaali, mutta sillä kuvataan yleensä järjestelmää koskevia ylätason vaatimuksia. Luokkakaaviolla esitetään useimmiten formaaleja ja yksityiskohtaisia vaatimuksia, jotka sijaitsevat kuvion vasemmassa alaneljänneksessä. Luokkakaavion muotoa voidaan käyttää myös yleisempien vaatimusten esittämiseen. Tilakaavion ja sekvenssikaavion muoto on tarkkaan määritetty, ja niillä esitetään tietojärjestelmän yksityiskohtaisia vaatimuksia.



KUVIO 16 Vaatimusten esittämistapojen nelikenttä

Esittämistavat sopivat eri tavoin Haikala ja Mikkosen (2011) esittämille kolmelle eri vaatimusten tasolle. Asiakasvaatimusten esittämiseen sopivat parhaiten yleiset ja vapaamuotoiset vaatimukset kuten käyttäjätarinat ja käyttötapauskaaviot. Ohjelmistovaatimusten tulee olla tarkemmalla tasolla, mutta silti asiakkaan ymmärrettävissä, jolloin esittämistavoiksi sopivat niin käyttötapaus kuin tietovuokaaviokin. Yleisimmillään myös luokkakaaviota voidaan käyttää tässä tarkoituksessa, mutta useammin luokkakaaviolla kuvataan teknisiä vaatimuksia. Teknisten vaatimusten kuvaamiseen käytetään myös sekvenssikaaviota ja tilakaaviota. Joskus asiakkaan vaatimuksen esittämiseen voidaan käyttää myös tarkempia ja formaalimpia esittämistapoja kuten esimerkiksi tilakaaviota vapaaajan hakemuksen käsittelyn tilan esittämiseen.

3.4 Yhteenveto

Tietojärjestelmän dokumentoinnissa hyödynnetään usein kahta dokumenttityyppiä, prosessidokumentteja ja tuotedokumentteja. Tuotedokumenteissa pitävät sisällään käyttäjädokumentteja ja järjestelmädokumentteja. Järjestelmädokumenteissa esitetään tietojärjestelmään kohdistuvia vaatimuksia eri esittämistavoilla. Toiminnallisten vaatimusten esittämistapoja ovat muun muassa käyttäjätarinat, käyttötapauskaavio ja -kuvaukset, tila- ja sekvenssikaavio sekä luokkakaavio. Näitä voidaan täydentää eri tasoisilla käyttöliittymäkuvauksilla. Esittämistapojen muodollisuus ja yksityiskohtaisuus vaihtelee, ja niitä käytetään tarpeen mukaan.

4 OHJELMISTON YLLÄPITO

Ohjelmiston ylläpito on IEEE:n (2013) mukaan ohjelmiston muokkaamista sen julkaisemisen jälkeen tarkoituksena korjata löytyneitä virheitä, parantaa ohjelmiston suorituskykyä tai muita ominaisuuksia tai muuttaa ohjelmisto toimimaan muuttuneessa ympäristössä. Ohjelmistoon kohdistuvissa muutoksissa tulee säilyttää ohjelmiston eheys.

Tässä luvussa tarkastellaan tietojärjestelmän pisintä elinkaaren vaihetta, ylläpitoa. Ensin esitetään, millaisessa kontekstiin ylläpito asettuu. Toiseksi esitetään, minkä tyyppisiä ylläpitotehtäviä on. Lisäksi esitetään ylläpitoprosesseista pikakorjausmalli, iteratiivinen parannusmalli ja IEEE 1219-1998 -standardin mukainen prosessimalli sekä siihen liittyvät vaiheet. Lisäksi esitellään ylläpitoon liittyviä haasteita, dokumentaation ylläpidon aikaisia hyödyntämiskohteita sekä dokumentaatioon liittyviä ongelmia.

4.1 Ohjelmiston ylläpidon konteksti

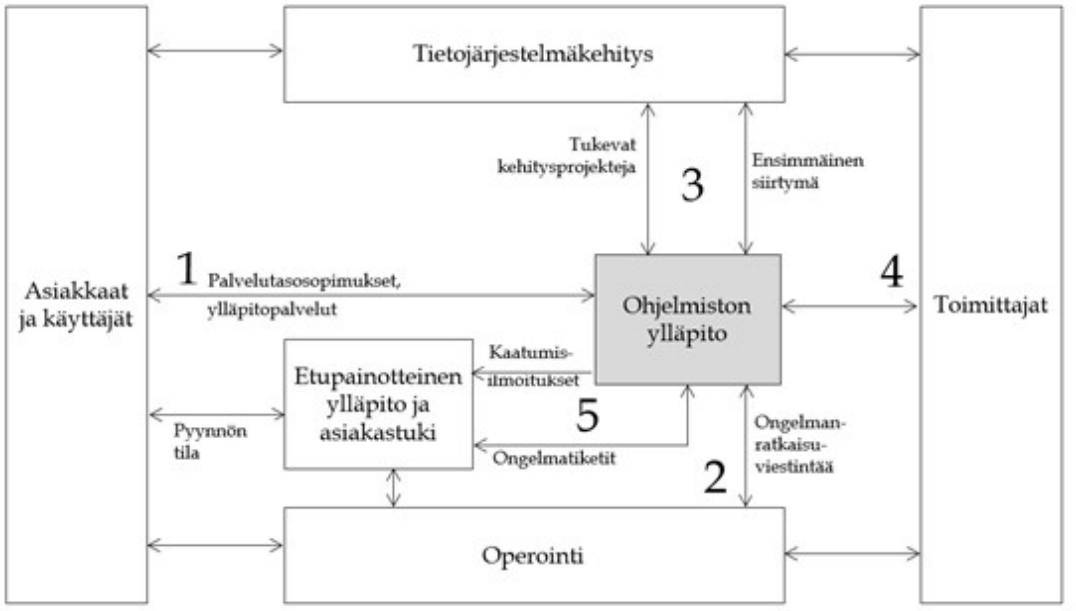
Ylläpidon tehokkuuden ja laadun varmistamiseksi on tärkeä ymmärtää, millaiseen kontekstiin ohjelmiston ylläpito asettuu. Ohjelmiston ylläpidon konteksti on moniulotteinen, ja sillä on monia rajapintoja ympäröivään maailmaan. Nämä tekijät tulee ottaa huomioon, jotta ylläpitäjät pystyvät päivittäisellä toiminnallaan varmistamaan ohjelmiston sujuvan toiminnan, ongelmien sattuessa nopean reagoinnin ja toiminnan palauttamisen sekä palvelutason vähimmäisvaatimusten täyttymisen. Muun muassa näillä tekijöillä varmistetaan osapuolten tyytyväisyys ja luottamus ylläpitäjien toimintaan ja heidän osaamiseensa.

Kuviossa 17 on esitetty ohjelmistojen ylläpitoon liittyvät tahot ja niiden väliset rajapinnat (April ym., 2005). Ylläpitoon osallistuvat tahot on esitetty suorakulmioina ja tahojen väliset suhteet nuoliviivoin. Ohjelmistojen ylläpito on keskeisenä kohteena tummennettu. Tyypillisessä ylläpidon organisaation kontekstissa on ohjelmiston ylläpidolla seuraavat rajapinnat (numeroinnilla viitataan kuvioon 17):

1. Asiakkaat ja ylläpidettävän ohjelmiston käyttäjät
2. Operointiosasto
3. Kehittäjät
4. Toimittajat
5. Etupainotteinen ylläpito ja asiakastuki

Etupainotteinen ylläpito ja asiakastuki on organisaatioissa se taho, joka vastaa kommunikoinnista asiakkaisiin ja käyttäjiin. Tätä kautta tulevat myös ilmoitukset ohjelmiston virheellisestä toiminnasta ja erilaiset palvelupyynnöt. Asiakastuki kommunikoi tarvittaessa ohjelmiston ylläpitäjien kanssa ja esimerkiksi viestii käyttäjälle aiemmin ilmoitetun virheen korjauksesta. (April ym., 2005).

Toinen rajapinta, *Operointi*, vastaa infrastruktuurista, jossa sovelluksia ajetaan. Siihen kuuluvat kaikki toiminnot, jotka liittyvät sovellusympäristön päivittäiseen hoitamiseen kuten esimerkiksi alustat, tietoliikenneyhteydet ja työasemat sekä varmuuskopiointi, palautukset ja järjestelmän hallinta. Lisäksi tärkeänä mutta vähemmän käytettynä osana on palveluiden häiriönhallinta ja häiriöistä toipuminen SLA-sopimuksen mukaisesti. (April ym., 2005).



KUVIO 17 Ohjelmiston ylläpidon konteksti (April ym., 2005, 199).

Kuvion 17 kolmas rajapinta on tietojärjestelmäkehityksen ja ohjelmiston ylläpidon välillä. Rajapinta muodostuu kehitettäessä uutta sovellusta. Ylläpitäjät osallistuvat usein ennen sovelluksen julkaisua suoritettaviin tehtäviin ja siirtymävaiheeseen sekä voivat tukea tai osallistua useisiin laajoihin kehitysprojekteihin yhtäaikaisesti. Kehitettäessä perinnejärjestelmää korvaavaa järjestelmää tai liittymää siihen ylläpitäjät ovat ensiarvoisen tärkeitä esimerkiksi liiketoimintasääntöjen ymmärtämisessä, olemassa olevien tietojen siirtämisessä tai perinnejärjestelmästä uuteen järjestelmään siirtymisen suunnittelussa. (April ym., 2005).

Neljäs rajapinta liittyy Aprilin ym. (2005) mukaan alati kasvavaan joukkoon toimittajia, jotka voivat olla ulkoistuksen hoitavia toimittajia ja toiminnanohjausjärjestelmien toimittajia. Sovelluksen ylläpitäjien tulee tuntea erilaisia sopimustyyppejä ja hallita niitä tehokkaasti varmistaakseen toimittajien suorituskyvyn. Tämä vaikuttaa usein palvelutasosopimuksen (service level agreement, SLA) tuloksiin. Ylläpitäjille on useita erilaisia suhteita toimittajiin, kuten esimerkiksi seuraavat (April ym., 2005):

- uutta järjestelmää kehittävät tai ERP-järjestelmää konfiguroivat toimittajat
- alihankkijat, jotka osallistuvat ylläpitoon ja omaavat erityisosaamista sekä tuovat lisäresursseja tarvittaessa

- toimittajat, jotka tarjoavat spesifejä tukipalveluita tietojärjestelmälle
- toimittajat, joille on ulkoistettu jokin IT-organisaation toiminto osittain tai kokonaan.

Asiakastuki on ylläpidon viides rajapinta. Asiakastuki huolehtii ongelmatilanteiden tehokkaasta ratkaisusta ylläpitoon liittyvien tahojen, varsinaisen asiakkaan, ylläpitäjien, infrastruktuurin ja operoinnin kanssa. Asiakastuki voi olla osa organisaation yksikköä, tai se voi olla oma itsenäinen yksikkönsä. (April ym., 2005).

4.2 Ylläpitotyypit

Ylläpidolla tarkoitetaan kaikkia niitä muutoksia, jotka tehdään järjestelmään sen julkaisun jälkeen. Muutokset ovat moninaisia, sillä tietojärjestelmä tulee pitää ajan tasalla muuttuvassa maailmassa niin käytettävien toimintojen kuin käyttöjärjestelmien ja laitteiston osalta. Ylläpidon aikana tehtävät muutokset voivat olla korjaavia (corrective), mukauttavia (adaptive), täydentäviä (perfective) tai ennakoivia (preventive) (Swanson, 1976; ISO/IEC, 1999). Nämä ylläpitotyypit on esitelty taulukossa 1. (de Souza, Anquetil & de Oliveira, 2005).

TAULUKKO 1 Ylläpitotyypit

YLLÄPITOTYYPPI	KUVAUS
Korjaava	• Virheiden ja häiriöiden korjaus, jotka liittyvät ohjelmakoodiin, ajoympäristön tai suorituskyvyn häiriöihin. (Swanson, 1976) • Vaatimuksissa, suunnittelussa tai ohjelmakoodissa havaittujen virheiden korjaus. (Sommerville, 2010a) • Suunnittelun, logiikan tai ohjelmakoodin virheet. (Grupp, 2003) • Virheiden korjaus järjestelmän julkaisun jälkeen (Yang & Ward, 2002; IEEE Computer Society, 2013; IEEE 1219-1998, 1998)
Täydentävä	• Ohjelmiston parantaminen esimerkiksi ajoympäristön, suorituskyvyn tai ylläpidettävyyden osalta. (Swanson, 1976) • Järjestelmän vaatimusjoukon laajentaminen, esimerkiksi toiminnallisuuden lisäys tehokkuuden parantaminen. (Grupp, 2003) • Suorituskyvyn tai muiden ominaisuuksien parantaminen järjestelmän julkaisun jälkeen (IEEE Computer Society, 2013; IEEE 1219-1998, 1998)
Ennakoiva	• Heikkenevän rakenteen aiheuttamien ongelmien korjaus ja ylläpidettävyyden parantaminen ohjelmakoodin sekä dokumentoinnin avulla. (Grupp, 2003) • Tulevaisuuden ongelmien ennakointi ja tulevan ylläpidon helpottaminen. (Grupp, 2003)

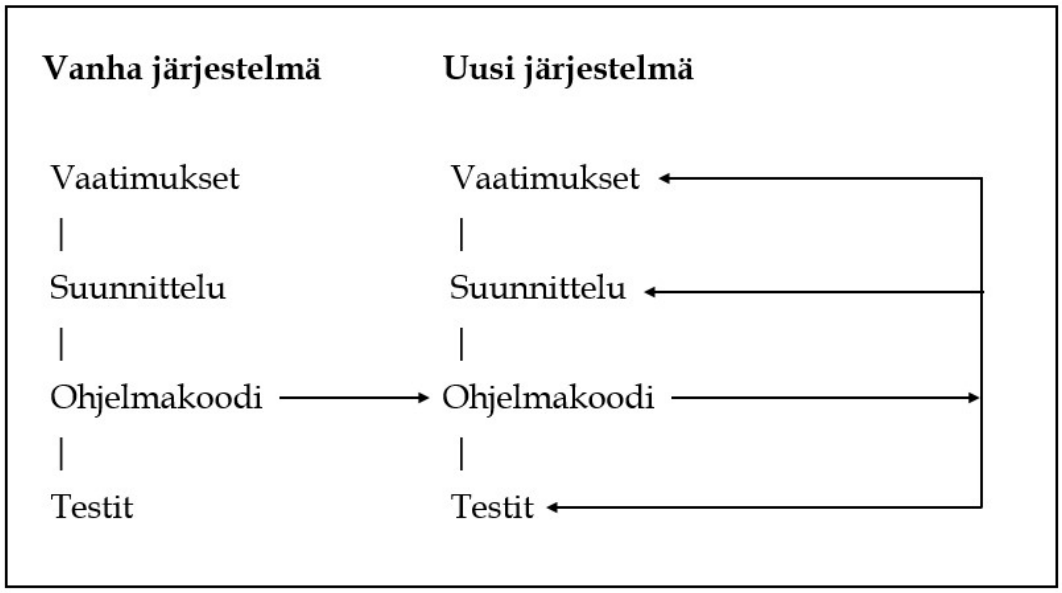
Ylläpidon aikana tehtävät *korjaavat* muutokset sisältävät korjauksia ohjelmakoodin virheisiin, kuten esimerkiksi virheelliseen tiedon käsittelyyn, sekä korjauksia erilaisiin häiriötilanteisiin, jotka voivat johtua esimerkiksi ajoympäristöstä, kuten esimerkiksi korjauksen sovelluspalvelimen konfiguraatioon. *Täydentävät* ylläpitotehtävät sisältävät niin vaatimuksiin liittyviä kuin teknisiäkin parannuksia. Näitä ovat esimerkiksi uusi vaatimus tai suorituskyvyn parantaminen tehokkaamman tietokantahaun avulla. *Ennakoivat* ylläpitotyöt korjaavat tai parantavat ohjelmistoa etukäteisesti. Tyypillinen ennakoiva ylläpitotyö on ohjelmakoodin refaktorointi esimerkiksi, kun ohjelmistoon on tehty paljon korjauksia kiinnittämättä huomiota ohjelmiston rakenteeseen.

4.3 Ylläpitoprosessi

Ylläpidon prosessimalleja on lukuisia. Osa prosessimalleista on hyvin suoraviivaisia, ja ne soveltuvat hyvin pieniin virhekorjauksiin. Tällaisia on esimerkiksi pikakorjausmalli. Osa prosessimalleista lähtee liikkeellä olemassa olevan järjestelmän analysoinnista, jonka jälkeen voidaan edetä ylläpitotyypin mukaisiin tehtäviin.

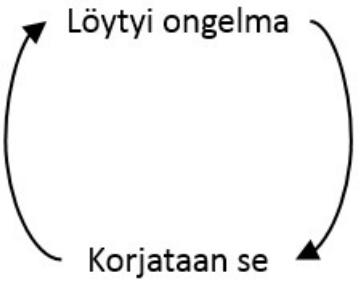
4.3.1 Pikakorjausmalli

Basili (1990) esittelee usein käytetyn pikakorjausmallin (quick-fix model). *Pikakorjausmallissa* muutetaan olemassa olevaa järjestelmää vain niiltä osin kuin se on välttämätöntä. Kuvion 18 mukaisesti ohjelmaan tehdään ohjelmakoodiin muutos, joka voi aiheuttaa muutoksia vaatimuksiin, ohjelmiston suunnitteluun ja testeihin. Useimmiten nämä kuitenkin jäävät päivittämättä, joten muutoksia tehdään ainoastaan ohjelmakoodiin. Tämä on tavanomaista erityisesti silloin, kun ohjelmakoodiin tehtävä muutos on virheen korjaus. Virhe ei ole jäänyt käytetyissä testeissä kiinni, mutta testejä ei virheenkorjauksen yhteydessä päivitetä tarkistamaan virheen kiinni jäämistä.



KUVIO 18 Pikakorjausmalli (Basili, 1990, 20).

Pikakorjausmallista on esitetty myös yksinkertaisempi versio. Gruppin (2003) mallissa (kuvio 19) odotetaan ongelman esiintymistä, ja sen ilmaannuttua se korjataan. Korjauksien tekemiseen ei liity pitkäaikaisvaikutusten analysointia kuten koodin hyvän rakenteen säilymistä. Virheen korjauksen yhteydessä dokumentaatiota korjataan olla vähän, jos ollenkaan. Pikakorjausmalli on parhaimmillaan yhden henkilön ylläpitämissä järjestelmissä, jolloin korjausten tekeminen on nopeaa ja edullista.



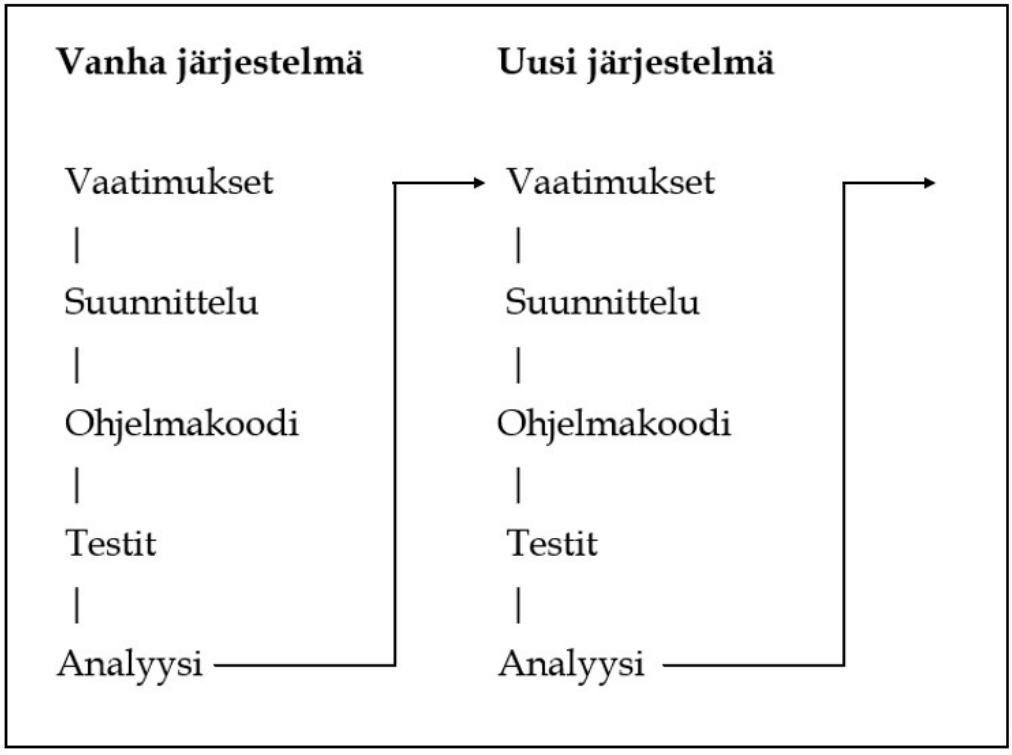
KUVIO 19 Yksinkertainen pikakorjausmalli (Grupp, 2003, 76).

4.3.2 Iteratiivinen parannusmalli

Iteratiivisen parantamisen mallissa (iterative-enhancement model) järjestelmän ylläpidon aikaisia muutoksia tehdään iteraatioissa samaan tapaan kuin järjestelmän kehitysvaiheessa. Kunkin iteraation aikana voidaan päivittää järjestelmän

dokumentaatiota kaikilla abstraktiotasoilla ja muuttaa toteutusta pienestä virheenkorjauksesta aina koko järjestelmän uudelleensuunnitteluun saakka. (Basili, 1990).

Järjestelmän ylläpidon aikana vanhaan järjestelmään toteutetaan muutos kuvion 20 mukaisesti. Muutoksen lähtökohtana ovat olemassa olevan järjestelmän vaatimukset, suunnittelu, ohjelmakoodi, testit ja järjestelmän muutostarpeiden analyysi. Järjestelmän muutos lähtee muuttamalla korkeimman abstraktiotason dokumentteja, joihin tehtävä muutos kohdistuu. Muutoksen ollessa esimerkiksi uusi vaatimus ensin päivitetään järjestelmän vaatimusdokumentit. Vaatimusten päivittämisen jälkeen tutkitaan, vaikuttaako uusi vaatimus vanhan järjestelmän suunnitteluun ja muutetaan tarvittaessa. Tämän tehdään sama ohjelmakoodille ja järjestelmän testeille. Viimeisenä vaiheena analysoidaan, onko tarvetta toteuttaa uusia muutoksia tähän uuteen järjestelmään. (Basili, 1990).

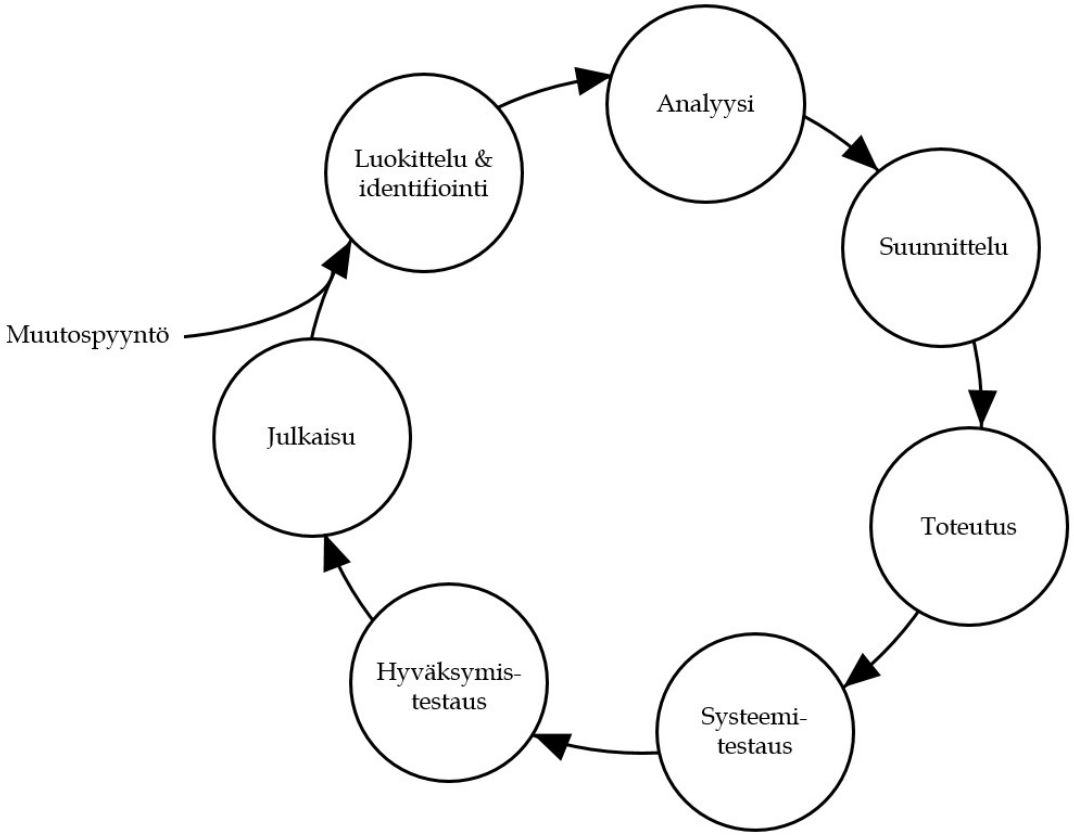


KUVIO 20 Iteratiivisen parantamisen malli (Basili, 1990, 20).

4.3.3 IEEE 1219-1998 -standardi järjestelmän ylläpidolle

IEEE 1219-1998 -standardi määrittelee ylläpitoprosessin koostuvaksi seitsemästä vaiheesta (Kuvio 21). Prosessin ensimmäisessä vaiheessa, *Luokittelu & identifiointi*, tunnistetaan muutostoive, ongelma tai virhe ja luokitellaan se eli määritellään minkä tyyppisestä ylläpitotehtävästä on kyse. Lisäksi se priorisoidaan, yksilöidään ja päätetään jatkotoimenpiteistä aikatauluineen. Kaikkia muutostoiveita ei

toteuteta tai ne voidaan toteuttaa myöhemmin. Tehtävän hyväksymiskriteerit ja arvio tarvittavista resursseista kirjataan myös ylös.



KUVIO 21 IEEE 1219-1998 Ohjelmiston ylläpidon prosessimalli (Abran & Moore, 2004, 6-7).

Prosessin toinen vaihe on analyysi. *Analyysivaiheessa* arvioidaan aiemmin kirjatun ylläpitotehtävän ja järjestelmän dokumentaation pohjalta tehtävän toteuttamiskelpoisuutta ja suunnitellaan alustavasti tehtävään liittyvän muutoksen suunnittelu, toteutus, testaus ja julkaisu. Toteuttamiskelpoisuuden arviointi sisältää mm. eri toteutusvaihtoehdot, vaatimusten muutokset, lyhyen ja pitkän aikajänteen kustannukset sekä muutoksen tuottama hyöty. Muutoksen kohteet tulee tunnistaa niin vaatimuksista ja muista dokumenteista, sovelluksesta kuin tietokannastakin. Testausstrategia testitapauksineen ja hyväksymistestauksen kriteereineen tulee määrittää. Näiden lisäksi on tärkeää suunnitella, miten muutoksen toteutus viedään loppuun aina julkaisemiseen saakka niin, että tämä vaikuttaa järjestelmän käyttäjiin mahdollisimman vähän. (IEEE 1219-1998, 1998.)

Seuraava vaihe on *suunnitteluvaihe*, joka pitää sisällään tarkan suunnitelman tekemisen muutoksen läpiviemisestä. Analyysivaiheen tuloksia hyödyntäen identifioidaan muutosten vaikutukset ohjelmiston moduuleihin, niitä kuvaavien dokumenttien päivityksen, testitapausten luonnin ja regressiotestit. Lisäksi päivitetään listaa muokattavista kohteista. Neljäs eli *toteutusvaihe* sisältää

neljä tärkeää tehtävää: koodaamisen ja yksikkötestauksen, muutettujen moduulien integroinnin olemassa olevaan järjestelmään, vaiheen aikana ja lopussa tehtävän riskianalyysin sekä ja sen arvioinnin, onko järjestelmä muutoksineen valmis systeemitestaukseen. (IEEE 1219-1998, 1998).

Systeemitestausvaiheessa varmistetaan, että järjestelmään tehdyt muutokset toimivat määritellysti ja olemassa olevat järjestelmän toiminnot toimivat, eivätkä muutokset ole aiheuttaneet uusia virheitä. Vaihe sisältää järjestelmän toiminnallisen testauksen, rajapintojen testauksen, regressiotestauksen ja hyväksymistestaukseen siirtymisen valmiuksien arvioinnin. *Hyväksymistestausvaiheessa* valmis järjestelmä testataan asiakkaan, muutoksia hyödyntävän käyttäjän tai asiakkaan osoittaman kolmannen osapuolen toimesta. Vaiheen aikana varmistetaan, että järjestelmä on valmis julkaistavaksi. *Julkaisuvaiheessa* järjestelmän fyysinen kokoonpano tarkistetaan, tiedotetaan käyttäjiä, muodostetaan ja varmuuskopioidaan arkistoitava versio järjestelmästä sekä suoritetaan asennukset ja koulutetaan käyttäjät. (IEEE 1219-1998, 1998)

4.4 Haasteet ylläpidossa

Ylläpito on tietojärjestelmän elinkaaren vaiheista kallein (Sommerville, 2010a; de Souza, Anquetil ja de Oliveira, 2005; Yip, 1995). Koska ylläpito maksaa paljon, siihen liittyviä haasteita on tutkittu paljon. Sharonin (1996) mukaan ylläpidon tuottavuutta heikentävät ja kustannuksia lisäävät seuraavat haasteet:

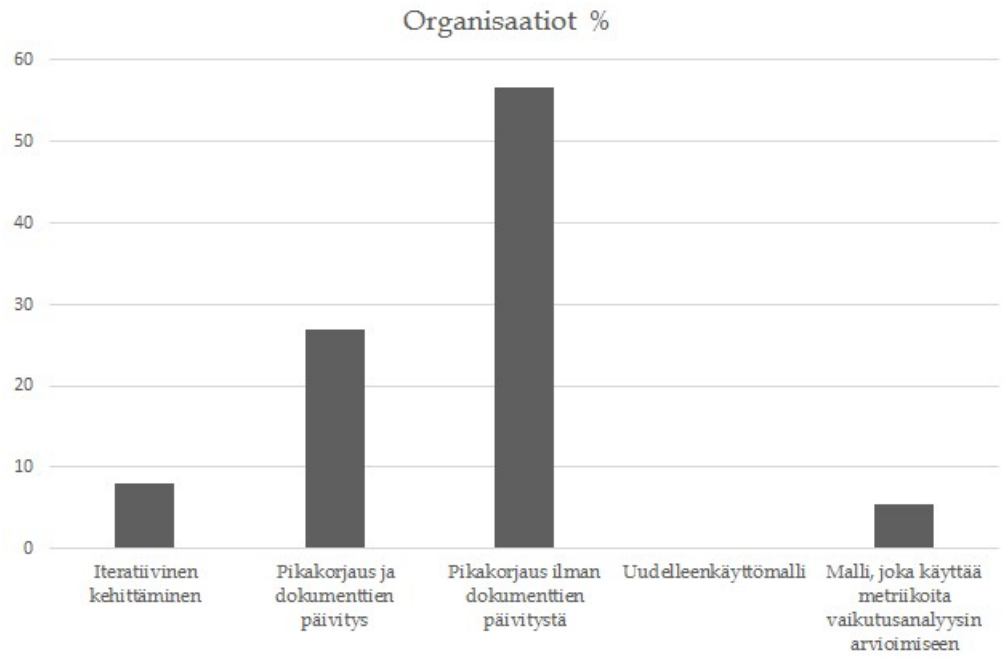
- Huono järjestelmän suunnittelu ja rakenne
- Ylenmääräinen järjestelmän monimutkaisuus
- Rajoitettu järjestelmän joustavuus
- Rajoitettu tai olematon dokumentaatio
- Riittämätön projektin- ja prosessinhallinta
- Riittämätön muutoksen- ja versionhallinta
- Riittämätön julkaisunhallinta
- Riittämättömät ylläpidon työkalut

Prosessinhallinnan avulla varmistetaan, että noudatetaan oikeita menettelytapoja ja käytetään oikeita työkaluja. *Projektinhallinnan* avulla pidetään yllä johdonmukaista työn jakamista hallittaviin osiin ja projektiryhmän jäsenten välistä kommunikointia. *Konfiguraationhallinnalla* hallitaan järjestelmän useat versiot, eroavaisuudet versioiden välillä, järjestelmän uudelleenkäytettävien komponenttien synkronoinnin ja hallinnan eri projektien välillä sekä komponentit, versiot ja muutokset sisältävän säilytyspaikan. Hyvä *ylläpidon työkalu* auttaa ymmärtämään nykyistä järjestelmää ja tekemään siihen muutoksia ja korjauksia nopeasti ja tarkasti. Työkalu edesauttaa myös järjestelmän uudelleen kehittämistä. (Sharon, 1996.).

Sousa ja Moreira (1998) tutkivat käytettyjä tietojärjestelmän ylläpitoprosesseja Portugalissa. Tutkimuksen tuloksena selvisi kolme merkittävintä ylläpitoon haitallisesti vaikuttavaa tekijää. Tekijät ovat ylläpidon prosessimallien puute, dokumentaation puute ja olemassa olevan dokumentaation päivittämiseen käytettävän ajan puute. Krogstie ja Solvberg (1994) tutkivat ylläpitoa Norjassa. Kyselyyn vastanneiden 52 norjalaisen organisaation vastauksista kävi ilmi, että kolme suurinta ongelmaa ylläpidossa ovat alkuperäisen sovellusten laatu, dokumentaation laatu ja ylläpitäjien vaihtelu.

Yip (1995) tutki tietojärjestelmin ylläpitoa Hong Kongissa ja havaitsi neljä merkittävää ylläpitoon vaikuttavaa ongelmaa. Ensimmäinen ongelmakategoria liittyy ympäristöön, jossa ylläpito suoritetaan. Nämä voivat olla resurssien puute, ohjelmointityylien erot, riittämätön dokumentointi ja välineiden tai ylläpitotyöhön tarkoitettujen graafisten käyttöliittymien puute. Toinen kategoria sisältää ylläpidon johtamiseen liittyviä ongelmia, jotka kulminoituvat puuttuviin johtamiskäytänteisiin dokumentaation hankkimisessa, vaatimisessa ja päivittämisessä. Kolmas ongelmien aiheuttaja on työntekijöiden ailahtelevuus ja nopea vaihtuvuus, mikä johtaa osaavien ylläpitäjien puuttumiseen. Neljäs ongelmakategoria on käyttäjien muuttuvat vaatimukset, joka yksinään on useimmin mainittu ongelma. Kolme ensimmäistä ongelmakategoriaa sisältävät useampia eri ongelmia.

Useassa tutkimuksessa mainittu *puutteellinen tai päivittämätön dokumentaatio* on yksi suurimmista ylläpidon ongelmista. De Souza, Anquetil ja de Oliveira (2005) tutkivat portugalilaisia finanssialan organisaatioita ja esittävät kuviossa 22 organisaatioiden käyttämät prosessimallit ylläpidossa. Käytettyjä prosessimalleja on viisi: iteratiivinen kehittäminen, pikakorjaus ja dokumenttien päivitys, pikakorjaus ilman dokumenttien päivitystä, uudelleenkäyttömalli sekä malli, joka käyttää metriikoita vaikutusanalyysin arvioimiseen. Prosessimallien käyttöastetta organisaatioiden ylläpidossa kuvaa mallikohtainen prosenttilukupylväs. Kuvion mukaan suuri osa organisaatioista käyttää pika-korjausmallia järjestelmien ylläpidossa, eikä näistä suurin osa päivitä dokumentaatiota ohjelmakoodin korjausten yhteydessä. Tämä johtaa siihen, että ylläpidossa tarvittavia tietoja joudutaan joskus etsimään jopa ohjelmakoodista asti (de Souza, Anquetil & de Oliveira, 2005; Dekleva, 1992; Thomas & Tilley, 2001.). Tämä voi johtaa siihen, että järjestelmään toteutetaan ylläpidon aikana käyttökelvotonta tai turhaa toiminnallisuutta. Se aiheuttaa resurssien haaskaamista ja menetettyjä mahdollisuuksia toisaalla. (Layzell & Macaulay, 1994). Deklevan (1992) mukaan puutteellisesta dokumentaatiosta aiheutuvat ongelmat liittyvät erityisesti vanhempiin järjestelmiin, jolloin ylläpidon tuottavuus vähenee ja kustannukset nousevat. Tästä aiheutuu ongelmia myös ylläpidon organisoinnille, kun uusien ylläpitäjien perehtyminen ylläpidettävään järjestelmään pitkittyy ja vanhat ylläpitäjät eivät voi siirtyä toisiin tehtäviin.



KUVIO 22 Ylläpitoprosessissa käytettävät mallit (de Souza, Anquetil & Oliveira, 2005, 274).

4.5 Dokumentaation hyödyntäminen ylläpidossa

Dokumentaatiota hyödynnetään monessa eri tarkoituksessa. Lethbridge, Singer ja Forward (2003) tutkivat telekommunikaatioyrityksen ylläpitotehtäviä suorittavia ohjelmistokehittäjiä. He havaitsivat, että dokumentaatio hyödyttää tietyssä roolissa toimivia henkilöitä tietyissä tehtävissä. Taulukossa 2 on esitetty prosenttiosuudet ohjelmistokehittäjistä, jotka arvioivat dokumentaation hyödylliseksi tai erittäin hyödylliseksi tietyissä tehtävissä. Dokumentaation katsottiin olevan hyödyllisin tietojärjestelmää opeteltaessa. Vähintään puolet vastaajista katsoi myös dokumentaation hyödylliseksi tietojärjestelmän testauksessa, työskennellessä uuden tietojärjestelmän kanssa tai ongelmien ratkaisussa toisten kehittäjien ollessa tavoittamattomissa vastaamaan kysymyksiin. Tietojärjestelmän kokonaiskuvaan liittyvien tietojen etsiminen pääse sekin lähelle 50 prosenttia. Noin kolmasosa vastaajista katsoi dokumentaation hyödylliseksi tai erittäin hyödylliseksi tietojärjestelmän ylläpidossa, vastattaessa johdon tai asiakkaiden kysymyksiin, etsittäessä tietojärjestelmään liittyvää syvällistä tietoa tai työskenneltäessä julkaisun tietojärjestelmän parissa. (Lethbridge, Singer & Forward, 2003).

TAULUKKO 2 Prosenttiosuudet ohjelmistokehittäjistä, jotka arvioivat dokumentaation hyödylliseksi tai erittäin hyödylliseksi tietyissä tehtävissä (Lethbridge, Singer ja Forward, 2003, 37).

TEHTÄVÄ	**%**
Tietojärjestelmän opettelu	61
Tietojärjestelmän testaus	58
Työskentely uuden tietojärjestelmän kanssa	54
Ongelmien ratkaisu toisten kehittäjien ollessa tavoittamattomissa vastaamaan kysymyksiin	50
Tietojärjestelmän kokonaiskuvaan liittyvien tietojen etsiminen	46
Tietojärjestelmän ylläpito	35
Johdon tai asiakkaiden tietojärjestelmään liittyviin kysymyksiin vastaaminen	33
Tietojärjestelmään liittyvän syvällisen tiedon etsiminen	32
Työskentely julkaistun tietojärjestelmän parissa	32

Vaikka uudet tietojärjestelmän kehittämisprosessit, kuten luvussa 2.3 esitetyt RUP-kehys ja ketterät menetelmät XP ja Scrum, ovat tulleet jäädäkseen, ne eivät muuta tietojärjestelmien ylläpitoa olennaisesti dokumentaation osalta. Dokumentaatio on edelleen olennainen osa tietojärjestelmien ylläpitoa. (de Souza, Anquetil & de Oliveira, 2005).

Cioch, Palazzolo ja Lohrer (1996) esittelevät, minkä tyyppistä informaatiota ylläpitoon osallistuvat henkilöt, erityisesti ohjelmoijat, tarvitsevat. Uudet ylläpitoon osallistuvat henkilöt tarvitsevat käsitteellisiä yleiskuvauksia järjestelmästä. Järjestelmään tutustuvat henkilöt perehtyvät jo tarkemmin järjestelmän toimintaan: mitä järjestelmä tekee, miten se sen tekee ja millaisessa ympäristössä järjestelmä toimii. Varsinaista ylläpitotyötä aloittavat tarvitsevat konkreettisempaa tietoa itse koodista, sen kääntämisestä ja ohjelman ajamisesta. Tässä vaiheessa hyödyllisiä dokumentteja ovat esimerkiksi käyttötapaukset ja sekvenssikaaviot. Järjestelmän asiantuntijoille on järjestelmän ylläpitovastuun edellyttämä asiantuntemus ja korkea tieto- ja taitotaso sekä kokemusta järjestelmän ylläpidosta valvotuissa olosuhteissa. Heille riittää ns. viitemateriaali (reference material) kuten vaatimukset ja suunnitteludokumentit. (Cioch, Palazzolo & Lohrer, 1996).

Ylläpitäjien dokumentaatio koskevat tarpeet ovat moninaisia. Moninaisia ovat myös ylläpitäjät sekä ylläpidossa suoritettavat tehtävät. Lisäksi myös itse dokumentaatio ja siihen liittyvät ongelmat voivat olla monenlaisia. De Souzan, Anquetilin ja de Oliveiran (2005) mukaan dokumentaatioon liittyy usein seuraavat ongelmat:

- Olematon tai huono laatu
- Päivittämätön
- Liian laaja tai vailla selvää päämäärää
- Vaikea päästä käsiksi (esimerkiksi dokumenttien sijoittelu eri tietokoneille tai eri formaatteihin)
- Ohjelmoijien kiinnostuksen puute

- Standardoinnin vaikeus

4.6 Yhteenveto

Tietojärjestelmän pisin elinkaaren vaihe on ylläpito. Ylläpidossa suoritetaan erityyppisiä tehtäviä, kuten korjaavia ja ennakoivia tehtäviä. Ylläpito sijoittuu omaan kontekstiinsa ja liittyy eri sidosryhmiin, kuten asiakkaisiin ja käyttäjiin, ja toimintoihin, kuten asiakastukeen ja operointiin. Ylläpitoa toteutetaan ylläpitoprosessin mukaisesti. Ylläpitoprosessi voi olla pikakorjausmalli, iteratiivinen parannusmalli tai IEEE 1219-1998 -standardi. Ylläpitovaihe sisältää monenlaisia tehtäviä, joissa kokemukseltaan eritasoiset ylläpitäjät hyödyntävät dokumentaatiota erilaisiin tehtäviin kuten tietojärjestelmän opetteluun, asiakkaan kysymyksiin vastaamiseen tai varsinaisiin tietojärjestelmän muutos- ja korjaustöihin. Ylläpitoon liittyy lukuisia haasteita, joista osa liittyy tietojärjestelmän dokumentaatioon. Dokumentaatiota hyödynnetään ylläpidossa paljon, joten myös dokumentaatioon liittyy ongelmia.

5 TAPAUSTUTKIMUKSEN TOTEUTTAMINEN

Luvussa kerrotaan, kuinka empiirinen tutkimus on toteutettu. Ensiksi kuvataan käytetty tutkimusmenetelmä, kuvataan tutkimusprosessi ja tiedonkeruutavat: millä tavalla vaatimusmäärittelydokumentteja on tarkoitus analysoida, miten haasteltavat valitaan ja mitä heiltä kysytään. Tämän jälkeen kuvataan lyhyesti tapaustutkimukseen valitut tietojärjestelmät. Lopuksi kerrotaan, miten kerättyjä tietoja käsitellään.

5.1 Tutkimusmenetelmä

Tutkimus toteutetaan empiirisenä, viiden tapauksen tapaustutkimuksena. Tapaustutkimuksella tarkoitetaan Runesonin ja Höstin (2009) mukaan empiiristä metodia, jonka tarkoituksena on tutkia nykyajan ilmiöitä niiden omassa kontekstissaan. Yinin (2014) mukaan tapaustutkimus voi olla tutkiva (exploratory), selittävä (explanative) tai kuvaileva (descriptive). *Tutkivan* tapaustutkimuksen tavoitteena on nimensä mukaisesti tutkia ja selvittää, mihin lopputulemaan tutkimus edetessään vie sen sijaan, että tutkimus todistaisi jo alkuvaiheessa esitetyn väittämän lopputuloksista. *Selittävä* tapaustutkimus pyrkii selittämään havaittuun ilmiöön liittyviä syy-seuraus-suhteita samaan tapaan kuin voidaan tutkia esimerkiksi jotain historian merkittävää tapahtumaa. *Kuvaileva* tapaustutkimus keskittyy kuvaamaan jotain ilmiötä tai toteutunutta tapahtumaketjua. (Yin, 2014).

Tämä tapaustutkimus on selittävä tapaustutkimus. Tutkimuksessa pyritään kuvaamaan ylläpidossa dokumentaatiosta saatavia hyötyjä ja dokumentaatioon liittyviä ongelmia sekä tarkastelemaan niihin johtaneita seikkoja. Näiden selvittämiseksi tämä tapaustutkimus noudattelee Runesonin ja Höstin (2009) esittämää viittä tapaustutkimuksen prosessin etenemisen vaihetta:

1. Tapaustutkimuksen suunnittelu: tavoitteet on asetettu ja tapaustutkimus on suunniteltu
2. Tietojen keräyksen valmistelu: tietojen keräämisen tapa ja protokolla on määritetty
3. Todisteiden kerääminen: tiedon keruu tutkimuksen kohteena olevista tapauksista
4. Kerätyn tiedon analysointi
5. Raportointi

Tutkimuksen tavoitteena on löytää vastauksia seuraaviin kysymyksiin:

- Millaiset tietojärjestelmävaatimusdokumentit hyödyttävät ylläpitoa?
- Minkä asioiden kehittämiseen kannattaa kiinnittää huomiota tietojärjestelmävaatimusdokumenttien hyödyntämisessä?

Näihin kysymyksiin pyritään vastaamaan tapaustutkimuksen avulla. Tietoja kerätään tutustumalla tietojärjestelmien dokumentaatioon sekä sähköpostitse tapahtuvalla haastattelututkimuksella. Haastatteluun valitaan ylläpidosta vastaavia henkilöitä, jotka antavat merkittävän panoksen kyseisten tietojärjestelmien ylläpitoon ja jotka käyttävät ylläpitotyössään vaatimusmäärittelydokumentteja. Haastattelussa käytetään avuksi kysymyspatteristoa, joiden avulla pyritään saamaan haastateltavat kuvaamaan ja tarkastelemaan ylläpitoon liittyviä seikkoja johdattelematta liikaa. Tarvittaessa voidaan pyytää tarkennuksia ja täydennyksiä vastauksiin. Sähköpostitse lähetetään seuraavat kysymykset:

1. Mikä ja millainen on ylläpitämäsi järjestelmä? Kerro lyhyesti järjestelmästä esimerkiksi sen laajuudesta, kriittisyydestä, toiminnallisuuksien ja käyttäjien määrästä.
2. Millainen on järjestelmän ylläpitoprosessi? Kerro erityisesti määrittelyyn ja toteutukseen liittyvistä vaiheista.
3. Miten järjestelmän ylläpito on järjestetty? Kerro ylläpitotehtävien laajuudesta ja määrästä sekä ylläpitäjistä määrä ja vaihtuvuus, tieto- ja taitotaso ja kuinka tuttu järjestelmä heille on.
4. Millaisia vaatimusdokumentteja järjestelmästä on tehty?
5. Mitä dokumentteja ylläpidossa käytetään, ja miten itse hyödynnät niitä? Miten dokumentit ovat hyödyttäneet järjestelmän ylläpitoa?
6. Mitä ongelmia ylläpidon aikana on ilmennyt dokumentaatioon liittyen?
7. Millaisia ratkaisuja ongelmiin löydät?
8. Miten vaatimusdokumentaatio hyödyttäisi järjestelmän ylläpitoa? Millainen dokumentaatio?

Lisäksi haastateltavia pyydetään lähettämään tietojärjestelmän vaatimusdokumentteja sähköpostitse. Tarkoituksena on tutustua joihinkin vaatimusdokumentteihin, jotka kuvastavat hyvin millaisia dokumentteja tietojärjestelmän ylläpidossa hyödynnetään.

Tiedonkeruun jälkeen tutustutaan huolellisesti sekä haastateltavien vastauksiin että lähetettyihin tietojärjestelmän vaatimusdokumentteihin. Jokaisen tietojärjestelmän jokaisen haastattelun kysymyksen vastaus luetaan huolellisesti. Samalla pohditaan, onko vastaus tarpeeksi selkeä ja kattava vai tarvitseeko se täydentämistä. Lisäksi pohditaan, voidaanko lisäkysymyksin herättää haastateltavassa lisäajatuksia tai esimerkiksi toisenlaista näkökulmaa asiaan.

Vastausten läpikäynnin jälkeen vastauksista raportoidaan haastattelukysymyksittäin. Jokaisesta tapauksesta kerrotaan yleiskuvauksen lisäksi tutkimuksen kannalta olennaisimmat seikat. Tämän jälkeen luvussa 7 analysoidaan tuloksia vertailemalla tapauksia ja niissä esiintyneitä löydöksiä toisiinsa ja pohditaan, mistä tulokset johtuivat.

5.2 Tutkimuksen kohteet

Tutkimuksessa tarkastellaan erään julkishallinnon organisaation tietojärjestelmiä. Organisaatiolla on mittava historia tietojärjestelmäkehityksessä, ja tietojärjestelmiä on lukuisia. Suurin osa organisaation tietojärjestelmistä kehitetään ja ylläpidetään organisaatiossa. Esimerkiksi alihankkijoita käytetään hyvin vähän. Organisaatiossa useiden tietojärjestelmien elinkaari on poikkeuksellisen pitkä, mutta henkilöstön vaihtuvuus ei ole suurta. Tietojärjestelmissä löytyy eroavaisuuksia liiketoimintakriittisyyden ja palveluajan, käyttäjämäärien, tekniikoiden, laajuuden, muutosten määrän ja tiheyden, ylläpidettävyyden, organisoinnin ja rahoituksen suhteen.

Tutkimuksen kohteena on julkishallinnon organisaation viisi eri tietojärjestelmää. Tietojärjestelmät edustavat organisaation tyypillistä, nykyaikaista tietojärjestelmää. Järjestelmien vaatimusdokumentaation esittämistavat moderneja käytäntöjä, ja valittujen ja niiden kaltaisten järjestelmien elinkaari jatkuu pitkälle tulevaisuuteen. Valitut tietojärjestelmät on rakennettu 2000-luvulla, ja ne ovat olleet ylläpidossa useita vuosia. Kokemukset kyseisten järjestelmien ylläpitotyöstä usein vuosien ajalta antaa edellytykset tutkia vaatimusmäärittelydokumenttien hyödyntämistä ylläpidon aikana.

Valitut tietojärjestelmät täyttävät edellä mainitut ehdot, mutta ne eivät ole täysin samanlaisia. Taulukossa 3 vertaillaan tutkimukseen valittuja tietojärjestelmiä eri kriteereillä. Tietojärjestelmän koko on arvioitu karkeasti tietojärjestelmään liittyvien toimintojen ja koodimäärän perusteella. Tietojärjestelmän kriittisyyden ollessa suuri jo lyhyet katkot palvelun käytössä aiheuttavat merkittävää haittaa organisaatiolla, kun kriittisyyden ollessa pieni viikonkin käyttökatko ei vaikuta organisaation toimintaan merkittävästi. Muutosten määrän ollessa suuri uusia muutostarpeita ilmaantuu vähintään viikoittain, kun taas sen ollessa pieni uusia muutostarpeita ei ilmaannu edes kuukausittain. Ylläpitäjien lukumäärä voi olla pieni, 1–2 henkilöä, tai suuri, yli 8 henkilöä. Ylläpitäjät vaihtuvuus on suurta, kun uusia ylläpitäjiä tulee puolivuosittain, ja pieni kun ylläpitäjät ovat ylläpitäneet tietojärjestelmää yli kolmen vuoden ajan. Ylläpidon kesto vaihtelee suuresta (yli 10 vuotta) pieneen (alle 2 vuotta).

TAULUKKO 3 Tutkimukseen valittujen tapausten vertailu

OMINAISUUS	TAPAUS 1	TAPAUS 2	TAPAUS 3	TAPAUS 4	TAPAUS 5
Tietojärjestelmän koko	Suuri	Suuri	Pieni	Keski	Pieni
Tietojärjestelmän kriittisyys	Suuri	Suuri	Keski	Suuri	Pieni
Ylläpitotehtävien määrä	Suuri	Suuri	Pieni	Keski	Pieni
Ylläpitäjien määrä	Keski	Suuri	Keski	Keski	Pieni
Ylläpitäjien vaihtuvuus	Pieni	Keski	Pieni	Pieni	Keski
Ylläpitovaiheen kesto	Suuri	Pieni	Suuri	Pieni	Keski

5.2.1 Tapaus 1

Tapauksen 1 tietojärjestelmä on organisaatiossa laajalti käytetty ns. erillisjärjestelmä. Järjestelmän tarkoitus on tukea organisaation ydinliiketoimintaa, ja sillä on liittymiä moniin järjestelmiin. Toiminnallisuuksiltaan ja tietosisällöltään järjestelmä on ainutlaatuinen, mutta perustoiminnallisuudet ovat tuttuja useimmille organisaation työntekijöille. Kyseessä on ensiarvoisen tärkeä järjestelmä, johon kohdistuu suuret muutostarpeet. Ylläpito on hyvin organisoitu, ja vastuu on useilla tahoilla ja henkilöillä.

5.2.2 Tapaus 2

Tapauksen 2 tietojärjestelmä on merkittävä osa organisaation ydinliiketoimintaa. Järjestelmän toiminnallisuus on laaja. Järjestelmä ei poikkea tekniikoiltaan tai tietosisällöltään merkittävästi organisaation muista ydintoiminnallisuuksiin liittyvistä järjestelmistä, mutta kriittisyydeltään se on muita selkeästi korkeampi. Itse järjestelmään liittyy useita eri sidosryhmiä, kun taas järjestelmän ylläpitoon osallistuu vain joitain ylläpitäjiä. Organisaation mittapuulla nuorehkon järjestelmän ylläpito on kuitenkin kohtuullisesti organisoitu, ja osa ylläpitäjistä on toiminut järjestelmän kanssa sen elinkaaren alusta asti.

5.2.3 Tapaus 3

Tapauksen 3 tietojärjestelmä on pitkän elinkaaren omaava järjestelmä, joka edustaa organisaatiolle tyypillistä liiketoimintaa sekä tietyn sukupolven sovellukselle tyypillistä tekniikkaa. Toiminnoiltaan järjestelmä ei ole kovin laaja, eikä monimutkainen. Ylläpitoon on riittävät resurssit, ja se on hyvin organisoitu. Suurimmat haasteet ylläpidossa liittyvät elinkaaren pituuteen. Käytetyt tekniikat eivät ole kovin uusia, eikä myöskään vaatimusmäärittelyssä käytetyt dokumentit ja esittämistavat ole moderneimpia. Nekin edustavat tiettyä sukupolvea organisaation tietojärjestelmien moninaisessa joukossa.

5.2.4 Tapaus 4

Tapauksen 4 tietojärjestelmä toimii organisaation ydinliiketoimintaan liittyvien tietojärjestelmien tukijärjestelmänä. Tämän tietojärjestelmän ylläpitovaihe on alkanut vasta hiljattain, minkä vuoksi ylläpidon organisointiin ei ole kohdistunut vielä isoja muutoksia. Ylläpitotyö on kuitenkin hyvin organisoitua, ja järjestelmään liittyvien muutosten määrä on kohtuullinen. Järjestelmän toiminnallisuudet ovat hyvin fokusoituja, mutta hyvin merkittäviä toiminnan sujuvuuden kannalta. Järjestelmä ei ole laaja, mutta toiminnallisuudet ovat liiketoiminnaltaan monimutkaisia.

5.2.5 Tapaus 5

Tapauksen 5 tietojärjestelmä toimii organisaation ydinliiketoimintaan liittyvien tietojärjestelmien tukijärjestelmänä. Tietojärjestelmä ei ole kriittinen, ja sen käyttäjiä on rajallinen määrä. Tietojärjestelmän ylläpitovaihe on kestänyt vuosia, joten sen ylläpito on vakiintunutta. Järjestelmä sisältää rajallisen määrän toiminnallisuuksia. Järjestelmän monimutkaisin osio liittyy järjestelmän integraatioihin. Järjestelmä sisältää joitain integraatioita, joista osa on organisaation ulkopuolisia.

6 TAPAUSTUTKIMUKSEN TULOKSET

Luvussa esitetään empiirisen tutkimuksen tulokset. Aluksi kerrotaan, millaisessa muodossa ja missä määrin vaatimuksia valituille tietojärjestelmille on esitetty dokumentteina. Toiseksi kerrotaan, millä tavalla näitä dokumentteja on pyritty hyödyntämään ylläpidossa. Kolmanneksi selvitetään, miten ylläpitäjät kokevat vaatimusmäärittelydokumenttien hyödyt ja ongelmat. Neljänneksi kerrotaan esille tulleista kehittämisideoista.

6.1 Yleiskuvaus

Tapauksen 1 tietojärjestelmä on hyvin keskeinen tietojärjestelmän organisaatiossa. Yhtäaikaisten käyttäjien määrä kuvastaa hyvin tietojärjestelmän kriittisyyttä; heitä on yli puolet organisaation henkilöstöstä. Tietojärjestelmä sisältää suuren määrän toimintoja sekä integraatioita toisiin järjestelmiin.

Tapauksen 2 tietojärjestelmä on myös keskeinen tietojärjestelmä organisaatiossa. Myös yhtäaikaisten käyttäjien määrä on suuri, mutta kuitenkin alle neljännes tapauksen 1 tietojärjestelmän käyttäjämäärästä. Tietojärjestelmän toimivuus on organisaation asiakkaiden kannalta merkittävä, joten järjestelmän toimimattomuudella olisi sekä organisaatiolla ja sen asiakkaille suuri merkitys. Tietojärjestelmä itsessään ei ole kovin laaja, mutta sen sisältämät toiminnallisuudet ovat kompleksisia. Järjestelmä on vahvasti integroitu muihin tietojärjestelmiin.

Tapauksen 3 tietojärjestelmä on tyyppiesimerkki organisaatiolle tyypillisestä tietyn sukupolven järjestelmästä. Tietojärjestelmän toimivuus näkyy organisaation asiakkaille, joten se on merkittävä, mutta ei kriittinen. Yhtäaikaisten käyttäjien määrä on maltillinen. Tietojärjestelmä ei ole laaja tai kompleksinen, eikä se sisällä suurta määrää integraatioita toisiin järjestelmiin.

Tapauksen 4 tietojärjestelmä edustaa organisaatiossa uudemman sukupolven tietojärjestelmää. Tietojärjestelmällä on merkittäviä integraatioita, joiden vuoksi sen oikeellinen toiminta on ensiarvoisen tärkeää. Yhtäaikaisia käyttäjiä on rajallinen määrä, mutta järjestelmän toiminnot ovat tärkeitä organisaation toiminnan kannalta.

Tapauksen 5 tietojärjestelmä on pienehkö, moderni tietojärjestelmä. Järjestelmä ei ole kriittinen, eikä sen käyttäjämäärä ole suuri. Tietojärjestelmä ei ole toiminnoiltaan monimutkaisimmasta päästä, mutta sillä on joitain integraatioita. Järjestelmä on integroitu organisaation ulkopuolisiin tietojärjestelmiin, mikä vaikuttaa myös tietojärjestelmän ylläpitoon.

6.2 Ylläpitoprosessi

Tapauksen 1 ylläpidossa noudatetaan luvussa 4.3.3 esitettyä IEEE 1219-1998 -standardia järjestelmän ylläpidolle. Erilaiset muutostarpeet ja -ehdotukset analysoidaan säännöllisesti ylläpitäjien kesken. Muutoksesta käydään läpi karkealla tasolla liiketoimintatoiveet sekä muutoksen toteuttamiskelpoisuus. Mikäli muutos katsotaan tarpeelliseksi ja toteuttamiskelpoiseksi, varmistetaan tarvittavat resurssit sekä sovitaan aikataulusta. Tämän jälkeen tehdään muutokseen liittyvä määrittelytyö, minkä jälkeen muutos etenee toteutettavaksi ja systeemitestattavaksi. Hyväksymistestauksen jälkeen muutos julkaistaan. Ylläpitoprosessissa hyödynnetään muutoksenhallintavälineitä.

Tapauksen 2 ylläpidossa muutostarpeiden ja -ehdotusten käsittely painottuu liiketoiminnasta vastaaville. Muutosten toteuttamiskelpoisuutta arvioidaan tarvittaessa myös toteuttamisen näkökulmasta. Määrittelyjen valmistuttua arvioidaan priorisoidun työn työmäärä, josta työ etenee toteutukseen, testaukseen ja julkaisemiseen.

Tapauksen 3 ylläpidossa noudatetaan pitkälti samankaltaista prosessia kuin tapauksen 2 ylläpidossa. Liiketoiminnasta vastaavat ovat vastuussa muutostarpeiden ja -ehdotusten käsittelystä mukaan lukien niiden laillisuus ja validiteetti. Tehtävistä muutoksista sovitaan toteuttajien kanssa. Liiketoiminnasta vastaava vastaa myös muutosten testauksesta yleensä etsimällä testaajan roolissa toimivan henkilön. Myös tapauksen 4 ylläpidossa noudatetaan tapausten 2 ja 3 ylläpidon kaltaista prosessia.

Tapauksen 5 ylläpidossa on ollut tehtäviä harvakseltaan. Muutosvaatimuksen tullessa ilmi muutos aikataulutetaan tavoiteaikataulun ja kiireellisyyden mukaan sekä sovitaan muutoksen toteutuksesta ja testauksesta. Järjestelmän ylläpidossa käytetään pääsääntöisesti pikakorjausmallia, sillä useimmiten vaatimusdokumentteja ei päivitetä.

6.3 Ylläpidon organisointi

Tapauksen 1 tietojärjestelmän ylläpito on hyvin organisoitu. Ylläpitoon osallistuu vain keskimääräinen määrä henkilöitä. Ylläpitäjien tieto- ja taitotaso ovat kuitenkin erinomaisella tasolla niin tietojärjestelmän ylläpidon kuin tietojärjestelmän vaatimusten, määrittelyjen ja toteutuksen osalta. Tietojärjestelmään tehtävien muutosten laajuus vaihtelee pienistä muutostöistä kuukausia kestäviin muutoskokonaisuuksiin.

Tapauksen 2 tietojärjestelmän ylläpidon organisointi on hyvällä tasolla. Organisointia kehitetään edelleen, sillä järjestelmä on organisaation mittakaavassa kohtuullisen tuore. Ylläpitoon osallistuu sekä järjestelmän elinkaaren alusta asti mukana olleita syväosaajia että uusia ylläpitäjiä. Tämän vuoksi myös ylläpitäjien

tieto- ja taitotaso vaihtelevat paljon. Tietojärjestelmän ylläpitoon kohdistuu paljon tehtäviä ja järjestelmään tehtävien muutosten määrä on kohtuullisen suuri. Tehtävät vaihtelevat hyvin pienistä projektin kokoisiin tehtäviin.

Tapauksen 3 tietojärjestelmän ylläpito on kohtuullisesti organisoitu. Järjestelmä on osalle ylläpitäjistä hyvin tuttu, mutta osalle vielä tuntematon. Ylläpidon organisoinnissa on otettu hyvin huomioon tiedon, myös hiljaisen tiedon, jakaminen. Ylläpitäjien tieto- ja taitotason katsotaan olevan kuitenkin erittäin riittävä suhteessa ylläpidettävään järjestelmään.

Tapauksen 4 tietojärjestelmän ylläpito on organisoitu organisaation ohjeistusten mukaisesti ja lyhyestä elinkaarestaan huolimatta hyvin. Liiketoiminnasta vastaavissa henkilöissä on ollut suuri vaihtuvuus, mutta muutoin toiminta on ollut stabiilihkoa. Tietojärjestelmään liittyvillä eri sovelluksilla ja komponenteilla on määritelty omat, erilliset vastuuhenkilöt, jotka vastaavat oman vastuualueensa ylläpitotöistä.

Tapauksen 5 tietojärjestelmä ei ole kriittinen järjestelmä, mikä näkyy tietojärjestelmän ylläpidon organisoinnissa. Tietojärjestelmän ylläpidossa on tällä hetkellä mukana vain kaksi henkilöä, joista toinen ei tunne järjestelmää. Järjestelmän tuntevalla ylläpitäjällä on erinomainen tieto- ja taitotaso järjestelmän ylläpitämiseen. Ylläpitoa ollaan organisoimassa parhaillaan uudelleen, jolloin ylläpitäjien määrä kasvaa.

6.4 Vaatimusdokumentaatio

Tapauksen 1 tietojärjestelmän vaatimusdokumentit ovat pääasiassa vain käyttötapauksia. Tapauksen käyttötapaukset ovat kuitenkin huomattavasti laajempia kuin mitä on esitetty luvussa 3.3.1. Käyttötapaukset sisältävät varsinaisen liiketoimintalogiikan lisäksi tarkan kuvauksen käyttöliittymän tiedoista ja toiminnoista. Voidaan sanoa, että tietojärjestelmän käyttötapaukset sisältävät varsinaisen käyttötapauksen lisäksi käyttötapaukseen liittyvien internet-sivun käyttöliittymäkuvauksen. Kuvauksissa ei kuitenkaan oteta kantaa käyttöliittymän ulkoasuun tai esimerkiksi elementtien sijoitteluun internet-sivuilla. Nämä kuvaukset sisältävät esimerkiksi erilaisia tietoluetteloita, käsittelysääntöjä, tarkistuksia ja järjestelmän integraatioihin liittyviä toimintoja ja tietoja. Lisäksi kuvauksissa määritellään eri kohteiden tiloja. Erillisiä tilakaavioita ei kuitenkaan ole hyödynnetty. Käyttötapausten lisäksi ylläpidossa käytetään teknisiä palvelukuvausdokumentteja, joissa kuvataan tietojärjestelmän tarjoamat integraatiorajapinnat.

Tapauksen 2 tietojärjestelmän vaatimusdokumentit koostuvat toiminnallisista ja ei-toiminnallisista vaatimuksista, käyttötapauksista ja käyttöliittymäkuvauksista. Tapauksen 1 tietojärjestelmän tapaan myös tässä tietojärjestelmässä on hyvin laajoja käyttötapauksia. Yksi käyttötapaus saattaa olla kymmenien sivujen mittainen. Osa käyttötapauksista on kuitenkin luvussa 3.3.1 esitetyn kaltaisia suppeampia käyttötapauksia. Myös käyttötapausten kirjoitustyyli vaihtelee paljon. Osa käyttötapauksista on hyvin selkeitä ja yksinkertaisia. Toiset käyttötapaukset ovat sitä vastoin pitkiä ja rönsyileviä. Käyttötapausten yhteismäärä on

yli 70. Ei-toiminnallisia vaatimuksia on jonkin verran, ja toiminnallisia vaatimuksia on satoja. Käyttöliittymäkuvauksia on lukuisia, ja ne ovat tarkalla tasolla. Kuvauksissa otetaan kantaa käyttöliittymän toiminnallisuuteen, tietosisältöön sekä ulkoasuun. Osa ulkoasua koskevista ohjeistuksista on ohjeellisia, ja tarkka ulkoasu määritellään organisaation yleisissä ohjeissa. Ne sisältävät tietoja komponenttien tarkoista tyypeistä, kuten myös joitain tekniseen toteutukseen liittyvä asioita mm. komponenttikirjastoista. Lisäksi käyttöliittymäkuvaus sisältää tietoja käyttöliittymän vaatimista integraatioista sekä integraatioiden toteutukseen liittyviä yksityiskohtia. Lisäksi on laadittu erilaisia teknisiä kuvauksia ja ohjeita. Tällaisia ovat esimerkiksi arkkitehtuurikuvaus ja ohjeet virheiden selvittelyn tueksi.

Tapauksen 3 tietojärjestelmää varten on tehty kattava vaatimusmäärittelydokumentti, joka sisältää tietoja muun muassa järjestelmän tarkoituksesta ja vaatimuksista. Lisäksi käytössä on käyttöliittymän rautalankamalleja. Rautalankamallit eivät sisällä visuaalista ilmettä, vaan ne sisältävät karkean käyttöliittymäkomponenttien asettelun ja tekstien lisäksi myös joitain vaatimuksia tietojen ja toiminnallisuudenkin suhteen. Lisäksi järjestelmän dokumentaatiossa on erillisiä, erilaisiin muutoksiin liittyviä dokumentteja sekä tämän organisaation internetkäyttöliittymiä koskevia yleisiä ei-toiminnallisia vaatimuksia.

Tapauksen 4 tietojärjestelmä sisältää kattavasti erilaisia vaatimusdokumentteja, joita on tehty eri esittämistavoilla. Näitä ovat muun muassa seuraavat:

- toiminnalliset vaatimukset
- käyttötapaukset
- käyttöliittymäkuvaukset
- luokkakaaviot
- sekvenssikaaviot
- tilakaaviot

Tapauksen 4 tietojärjestelmän käyttötapaukset ovat laajoja tapausten 1 ja 2 tietojärjestelmien tapaan. Ne sisältävät liitteinä erilaisia selventäviä kuvia, prosessikuvauksia jne. Käyttöliittymäkuvauksia on runsaasti. Kuvaukset ovat tapauksen 2 tietojärjestelmän kuvausten kaltaisia. Myös tämän tapauksen käyttöliittymäkuvaukset kuvaavat joitain järjestelmän käyttöliittymän integraatioita ja ovat erityisesti toiminnallisuuksiltaan ja tietosisällöiltään yksityiskohtaisia. Sekvenssikaaviot ovat järjestelmätason sekvenssikaavioita, joissa kuvataan tapauksen tietojärjestelmän ja siihen integroituvien tietojärjestelmien välisiä palvelukutsuja sekä järjestelmän sisäiset tietokantakutsut. Tapauksen 4 tietojärjestelmän vaatimusdokumentaatio pitää sisällään myös tilakaavioita. Tilakaavioissa kuvataan kattavasti jonkin järjestelmän objektin tiloja, tilojen välisiä tilasiirtymiä sekä niihin liittyviä järjestelmän toimintaa ja toimintoja. Lisäksi tietojärjestelmästä on piirretty luvussa 3.3.2 esitetyn kaltaisia luokkakaavioita.

Tapauksen 5 tietojärjestelmä sisältää tietojärjestelmän kokoon nähden vähintään riittävästi erilaisia vaatimusdokumentteja. Järjestelmän dokumentaatio

kattaa muun muassa toiminnalliset ja ei-toiminnalliset vaatimukset, käyttötapauskaavion ja -kuvauksia sekä sekvenssikaavioita. Tietojärjestelmään liittyvät käyttötapaukset ovat kattavia. Järjestelmä ei ole yksittäisiltä toiminnallisuuksiltaan kovin monimutkainen, joten käyttötapaukset ovat lyhyitä ja selkeitä. Järjestelmän käyttöliittymää kuvataan käyttöliittymäkuvauksessa, jossa on rautalankamallikuvien lisäksi runsaasti selitystekstiä. Kuvauksessa kerrotaan laajasti käyttöliittymän toiminnallisuudesta, mutta niukasti ulkoasuun liittyviä vaatimuksia ja vain vähän sen teknisiä yksityiskohtia. Järjestelmän integraatioita kuvataan eritasoisissa prosessikuvauksissa sekä sekvenssikaavioissa. Sekvenssikaavioissa kuvataan järjestelmien välistä toimintaa, joihin ei henkilökäyttäjä osallistu. Sekvenssikaavioissa kuvataan objekteina tietojärjestelmän komponentit sekä tietojärjestelmään integroituvat toiset järjestelmät. Kaavioissa kuvataan sekvenssikaavioille tyypilliseen tapaan objektien välinen vuorovaikutus. Tämän lisäksi kuvataan karkealla tasolla eri objektien välillä liikkuvia tietoja ja käyttöoikeuksiin liittyviä yksityiskohtia. Järjestelmään on tehty myös muita dokumentteja kuten erilaisia arkkitehtuurikuvauksia, tietojen kuvausdokumentteja sekä järjestelmän ylläpitoa varten erillinen ohjeistus.

6.5 Vaatimusdokumenttien hyödyntäminen

Tapauksen 1 ylläpidossa käytetään eniten internet-sivu-kohtaisia kuvausdokumentteja, joiden sisältö koostuu käyttötapauksesta ja käyttöliittymäkuvauksesta. Dokumentit ovat ajantasaisia, ja niitä on laajennettu osittain myös tekniselle tasolle. Kun muutostarve tai -ehdotus on sovittu toteutettavaksi, dokumenttia päivitetään muutoksen mukaisesti. Muutokset kuvataan dokumenttiin tarkasti, joten dokumentista saa lähes aina kaiken tarvittavan tiedon. Ylläpitäjille kyseiset dokumentit ovat ylläpidon tärkein dokumentti, ja niistä löytyvät liki kaikki tarvittava tieto. Palvelukuvausdokumentit palvelevat enemmän muiden tietojärjestelmän ylläpitäjiä kuin tapauksen 1 tietojärjestelmän ylläpitäjiä.

Tapauksen 2 tietojärjestelmän vaatimusmäärittelydokumentaatio on ylläpitäjien mukaan kattava. Tietojärjestelmän ylläpidossa käytetään toiminnallisia ja ei-toiminnallisia vaatimuksia sekä käyttötapauskuvauksia. Näistä eniten hyödyttävä dokumentaatio on käyttötapaukset. Lisäksi hyödynnetään käyttöliittymäkuvauksia ja teknistä ohjeistusta sekä ylläpitoa varten erikseen laadittuja ohjeita. Vaatimusdokumentteja hyödynnetään tehtäessä uutta toiminnallisuutta sekä perehdyttäessä ylläpitäjälle uuteen toiminnallisuuteen. Käyttötapauksia käytetään myös selvitettäessä, kuinka järjestelmän kuuluu toimia. Erityisesti käyttötapaukset auttavat tilanteissa, joissa selvitetään, toimiiko järjestelmä vaatimusten mukaisesti vai onko kyseessä virhetilanne.

Tapauksen 3 ylläpidossa hyödynnetään rautalankamalleja sekä erillisiä muutoksiin liittyviä dokumentteja. Rautalankamallit sisältävät joitain käyttöliittymään liittyviä visuaalisia vaatimuksia sekä tietoihin ja toiminnallisuuteen liittyviä vaatimuksia. Yksittäisiä muutoksia kuvaavat dokumentit kertovat, kuinka jonkun järjestelmän osion tai toiminnallisuuden kuuluisi jatkossa toimia. Lisäksi

tietojärjestelmän ylläpidossa hyödynnetään organisaation internet-käyttöliittymiä koskevia yleisiä ei-toiminnallisia vaatimuksia. Näihin vaatimuksiin liittyvät tehtävät tulevat liiketoiminnasta vastaavien kautta tehtäväksi ylläpitäjille.

Tapauksen 4 ylläpidossa käyttötapauskuvausten katsotaan olevan täysin välttämättömiä ylläpidon kannalta, sillä tietojärjestelmään liittyy monimutkaista liiketoimintaa. Vuosien mittaan jokainen yksityiskohta ja liiketoiminnan kiemura ei säily aktiivimuistissa. Henkilöstön vaihtuvuus tuo painetta dokumentaatiolle. Käyttötapausten ohella ajoittain hyödynnetään sekvenssikaavioita erityisesti teknisemmässä suunnittelussa. Tilakaaviot on havaittu hyödyllisiksi, sillä tilakaavioista käy nopeasti ilmi tietyn asian tilat ja tilasiirtymät. Käyttöliittymäkuvauksia ei ole vielä hyödynnetty järjestelmän lyhyen ylläpitovaiheen vuoksi. Ne kuitenkin katsotaan hyvin tarpeellisiksi dokumenteiksi, sillä ne helpottavat asioiden ymmärtämistä. Kuvauksilla on myös tehokasta ja helppoa ilmaista, mitä ylläpidossa halutaan tehtävän. Muutosten lähtökohtana on aina jokin vaatimusdokumentti; esimerkiksi käyttötapauskuvaus, käyttöliittymäkuvaus, palvelun tai eräajon kuvaus, käsitemalli tai tietokantamalli.

Tapauksen 5 ylläpidossa on havaittu erityisen hyödylliseksi integraatioita kuvaavat sekvenssikaaviot, joista käy ilmi eri järjestelmien välillä liikkuvat tiedot. Näissä dokumenteissa kuvataan myös integraatioiden teknisiä vaatimuksia. Lisäksi on hyödynnetty käyttötapauskuvauksia, käyttöliittymäkuvauksia, arkkitehtuurikuvauksia sekä ylläpidon ohjetta ja tietokannan kuvausta.

6.6 Dokumentaatioon liittyvät ongelmat

Tapauksen 1 tietojärjestelmän vaatimusdokumentaatiossa on havaittu ongelmaksi dokumentaation tason suppeus. Tason suppeuden lisäksi yhdessä dokumentissa saattaa olla useaan eri muutokseen liittyvää määrittelyä, mikä voi aiheuttaa sekaannuksia ja vaikeuttaa lukemista. Ongelmalliseksi on havaittu myös tietyn toiminnallisuuden määrittelyn löytäminen. Lisäksi nykyisiä ylläpitäjiä mietityttää, miten hyvin nykyinen vaatimusdokumentaatio palvelee tietojärjestelmän tulevia, uusia ylläpitäjiä.

Tapauksen 2 ylläpidossa dokumentaation löytäminen on koettu ongelmalliseksi, sillä dokumentit ovat olleet hajautuneena useaan eri paikkaan. Lisäksi osa dokumentaatiosta ei ole ollut ajan tasalla. Paljon hyödynnettyjen käyttötapausten sisältö on vaihtelevaa, eivätkä ne ole määrämuotoisia. Myös käyttötapausten kirjoitustyyli vaihtelee yksinkertaisen ja rönsyilevän välillä. Käyttöliittymään liittyvä dokumentaatio on puutteellinen, eikä kaikista käyttöliittymän internet-sivuista tai tilanteista ole tehty kuvauksia. Teknisiä dokumentteja ei ylläpidetä, joten ne eivät ole enää ajantasaisia.

Tapauksen 3 tietojärjestelmän vaatimusdokumentaatiossa on parannettavaa. Iso kattava vaatimusdokumentti on tehty tämän pitkän elinkaaren omaavan tietojärjestelmän alkutaipaleella, eikä sitä ole päivitetty vuosien varrella. Paljon käytetyillä rautalankamalleilla ei pystytä ilmaisemaan riittävästi itse järjestelmän

toimintaa. Rautalankamallien staattisilla kuvilla on myös vaikea ilmaista käyttöliittymän rakenteeseen tai lukuisiin komponentteihin ja elementteihin liittyvää monimutkaisuutta ja interaktiivisuutta. Kuvat luodaan välineillä, joilla on hankala saada haluttua tai toteuttamiskelpoista asettelua.

Tapauksen 4 tietojärjestelmän vaatimusdokumentteihin liittyvät ongelmat ovat liittyneet pääosin dokumentaatiossa käytettyihin välineisiin. Välineet muuttuvat aivan liian usein, käytetyt välineet toimivat virheellisesti tai välineiden lisenssien määrää on rajoitettu. Osa käytetyistä dokumenteista on laajoja, ja olennaisten asioiden löytäminen niistä on haastavaa. Useimmat dokumentit ovat kuitenkin hyvin rajattu ja hyvin löydettävissä. Dokumenteissa on esiintynyt aika ajoin joitain pieniä puutteita tai epäselvyyksiä sisällössä. Käyttöliittymäkuvauksiin liittyy välineongelmien lisäksi resursointiin liittyvä ongelmia. Organisaatiossa käyttöliittymäsuunnittelu on keskitetty omalle taholleen. Tämä on aiheuttanut ongelmia kehityksen aikana, ja se todennäköisesti aiheuttaa ongelmia myös ylläpidon aikana. Ongelmia ovat muun muassa dokumenttien ajantasaisuus ja päivittäminen sekä käyttöliittymäkuvausten yhteensopivuus muun dokumentaation kanssa. Mikäli tietojärjestelmän käyttöliittymän toimintaa ja ulkoasua mietitään omilla tahoillaan, nämä ovat todennäköisesti ristiriidassa keskenään ja siten kelvottomia muutoksen lähtökohdaksi.

Tapauksen 5 tietojärjestelmän ylläpidossa on havaittu, että vaatimusdokumentaation ylläpito unohtuu helposti, eivätkä dokumentoidut tiedot ole silloin ajan tasalla.

6.7 Dokumentaatioon liittyvien ongelmien ratkaisut

Tapauksen 1 tietojärjestelmän vaatimusdokumentaatioon liittyvät ongelmat ratkaistaan useimmiten kommunikoimalla vaatimusdokumentaatiosta ja määrittelystä vastaavien tahojen kanssa. Usean yhtäaikaisen muutoksen tuomat haasteet on pyritty ratkomaan erottamalla muutokset toisistaan manuaalisesti. Tietyn toiminnallisuuden määrittelyn löytämiseen voidaan käyttää hakutoimintoja. Tämä ongelma on ratkennut toistaiseksi sillä, että ylläpitäjät tuntevat järjestelmän niin hyvin, ettei varsinaiselle etsimiselle ole ollut juuri tarvetta.

Tapauksen 2 ylläpidossa vaikeudet löytää ajantasainen dokumentaatio on korjattu päättämällä sijoittaa kaikki järjestelmän dokumentaatio yhteen paikkaan, ja ylläpitää niitä vain siellä. Vaatimusdokumenttien päivittäminen katsotaan osaksi ylläpitotyötä. Päivittäminen ja selkeyttäminen tulisi kuitenkin ottaa paremmin huomioon töiden suunnittelussa ja priorisoinnissa, jotta ne tulisivat ylläpitäjien tehtäväksi. Lisäksi dokumentteja muokatessa tulisi niitä samalla parannella, korjata ja päivittää.

Tapauksen 3 ylläpidossa kommunikointi on ensiarvoisen tärkeää hyvän lopputuloksen kannalta. Ylitsepääsemättömiä ongelmia ei ole ollut puutteellisesta ns. pysyväisluonteisesta vaatimusdokumentaatiosta huolimatta. Yksittäisiä muutoksia koskevien erillisdokumenttien hyödyntämisen voidaan katsoa toimi-

van väliaikaisena ratkaisuna puutteelliseen dokumentaatioon. Rautalankamallien ja muutosdokumenttien avulla saadaan sovittua varsinaisesta muutoksesta, jaettua tietoa ja toteutettua muutos julkaisuun saakka. Vaikka kommunikoinnilla on pystytty paikkaamaan puuttuvaa dokumentaatio, ylläpidossa on koettu, että järjestelmän toiminnan tarkempi kuvaaminen vaatisi esimerkiksi käyttötapauskuvauksia, joita tästä järjestelmästä ei ole tehty lainkaan.

Tapauksen 4 ylläpidossa dokumenteissa muutokset merkitään manuaalisesti, jolloin ylläpitotehtäviin liittyvät dokumentaatiomuutokset löytyvät laajoista dokumenteista helpommin. Lisäksi esimerkiksi liian laajaksi paisuneet käyttötapaukset on pilkottu pienemmiksi. Sisällölliset pienet haasteet on selätetty dokumenttien täydentämisellä, yhteistyöllä ja kommunikoinnin avulla. Välineisiin liittyviä ongelmia on mahdollista ratkaista kohtuullisen rajoitetusti, ja ratkaisunakin on useimmiten manuaalityö. Käyttöliittymäkuvauksien odotetut ongelmat voidaan ratkaista kahdella tavalla. Ensimmäinen tapa on siirtää käyttöliittymäsuunnitteluun liittyvä työ muun ylläpitotyön rinnalle, jolloin käyttöliittymään liittyvät seikat huomioidaan siinä missä toiminnallisetkin seikat. Toinen tapa on parantaa yhteistyötä käyttöliittymäsuunnittelun kanssa, kehittää resursointia sekä ottaa paremmin käyttöliittymää koskevat seikat huomioon muuta toiminnallisuutta tarkasteltaessa.

Tapauksen 5 ylläpidossa muutoksia ja ylläpitäjiä on tullut verrattain vähän. Lisäksi ylläpitoa ollaan siirtämässä organisaation sisällä toisaalle. Näiden seikkojen vuoksi ongelmia ei ole ratkaistu. Ongelmille löytyy kuitenkin ratkaisuvaihtoehtoja nykyisessä ylläpito-organisaatiossa. Järjestelmän ylläpitoa hyödyttäisi ylläpidettävien dokumenttien määrän rajaaminen ja vain olennaisimpien (1–2 kpl) dokumenttien päivittäminen. Lisäksi dokumenttien sisältöä voisi kehittää: liian tarkalla tasolla esitetty dokumentaatio on turhaa, sillä asiat käyvät ilmi myös suoraan ohjelmakoodista. Dokumenttien sisällön tulisi myös määrämittaista; hyvän rakenteen mukaisesta dokumentista tiedot löytyvät helpommin, jolloin niitä on helpompi myös ylläpitää.

6.8 Dokumentaatioon liittyvät toiveet

Tapauksen 1 tietojärjestelmän ylläpidossa ei ole havaittu merkittäviä ongelmia vaatimusdokumenttien suhteen, eikä ylläpitäjillä ole erityisiä toiveita dokumentaation suhteen.

Tapauksessa 2 ylläpitäjät ovat hyötyneet nykyisistä toiminnallisiin vaatimuksiin liittyvästä dokumentaatiosta. Dokumentaatio hyödyttäisi vieläkin enemmän, mikäli ylläpitäjien käytettävissä olisi ajantasainen ja määrämuotoinen dokumentaatio, joka on selkeä ja silti riittävän kattava. Lisäksi tietojärjestelmän nykyisin puutteellisia ei-toiminnallisia vaatimuksia tulisi täydentää, jotta ne tukisivat paremmin muun muassa järjestelmän kehitystä sekä ylläpidon organisointia.

Tapauksen 3 ylläpidossa hyödyttäisiin vaatimusmäärittelydokumentista ja käyttötapauskuvauksista. Vaatimusmäärittelydokumenttia ei ole päivitetty tietojärjestelmän elinkaaren alun jälkeen. Käyttötapauksia ei tällä hetkellä ole lainkaan. Näitä dokumentteja voitaisiin hyödyntää järjestelmän ylläpidossa käyttää esimerkiksi tutustuttaessa järjestelmän toimintaan. Lisäksi dokumenteista kävisi ilmi erikoisemmat ja poikkeuksellisemmatkin tapahtumien kulut, jolloin myös järjestelmän toimintaa pystyttäisiin validoimaan dokumentaatiota vasten. Tämän lisäksi paremmat ja oikeanlaiset työkalut nykyisten rautalankamallien tekemiseen helpottaisi ja parantaisi ylläpitotyötä.

Tapauksen 4 ylläpidossa esiintyvät ongelmat liittyvät pääsääntöisesti dokumentoinnissa käytettyihin välineisiin sekä ylläpitotyön tekemisen tapaan ja organisointiin. Tällä hetkellä suurimmat toiveet kohdistuvat stabiilimpaan ja toimivampaan välineistöön.

Tapauksen 5 ylläpidossa toiveet kohdistuvat itse dokumentaatioon. Nämä toiveet liittyvät havaittujen ongelmien ratkaisuehdotuksiin. Dokumenttien määrän kaventaminen sekä dokumenttien sisällön tarkkuuden vähentäminen toisivat lisähyötyä ylläpidolle kuten myös dokumenttien hyvä rakenne ja määrämittaisuus. Lisäksi ylläpidossa toivotaan ohjelmakoodista generoitavaa dokumentaatiota, erityisesti JavaDoc-kommentteja.

6.9 Yhteenveto

Tutkituissa tapauksissa hyödynnetään erilaisia vaatimusdokumentteja. Hyödynnetyimpiä dokumentteja ovat käyttötapaukset ja käyttöliittymää kuvaavat dokumentit; rautalankamallit ja käyttöliittymäkuvaukset. Dokumentteja hyödynnetään moneen tarkoitukseen. Yleisin käyttötarkoitus on dokumenttien käyttö ylläpidon aikaisten muutosten ja virheenkorjausten lähtökohtana. Dokumentteja hyödynnetään myös järjestelmään tutustuttaessa ja selvitettäessä, kuinka järjestelmän kuuluu toimia. Järjestelmien ylläpitäjät katsovat vaatimusdokumentaation välttämättömäksi osaksi järjestelmän ylläpitoa. Ongelmia ylläpidossa aiheuttavat olematon tai päivittämätön dokumentaatio. Ajoittain ylläpitäjillä on vaikeuksia löytää oikeita asioita, sillä osassa tapauksia dokumentaatio on laaja ja yksittäiset dokumentit ovat laajoja. Ylläpitäjien toiveena on, että dokumentaatio olisi ajantasainen, tarpeeksi kattava, muttei turhan laaja, määrämuotoinen ja rakenteeltaan hyvä sekä tehty järkevillä työkaluilla.

7 POHDINTA

Luvussa suoritetaan tulosten pohdintaa. Ensin pohditaan, mistä dokumentaation tuottamat hyödyt johtuvat ja voiko hyötyihin johtaneita seikkoja hyödyntää myös muissa organisaation tietojärjestelmien vaatimusten esittämisessä. Myös löytyneiden puutteiden syitä ja seurauksia analysoidaan ja pohditaan keinoja niiden ehkäisemiseen ja/tai korjaamiseen. Luvussa verrataan tutkimuksen tuloksia aiempiin tutkimuksiin. Lisäksi luvussa tarkastellaan tutkimuksen validiteettia ja reliabiliteettia. Lopuksi esitetään tutkimuksen aikana heränneitä jatkotutkimusaiheita.

7.1 Ylläpidossa hyödynnettävät esittämistavat

Tutkituissa tapauksissa tietojärjestelmät sisältävät erilaisia vaatimusdokumentteja. Taulukossa 4 on esitetty, miten eri tapauksissa on käytetty eri vaatimusdokumentteja. Dokumenttien käyttöön on eritelty, onko dokumentti vain olemassa vai onko sitä myös hyödynnetty ylläpidossa.
Virhe. Linkki ei kelpaa.
TAULUKKO 4 Olemassa olevat ja ylläpidossa hyödynnetyt dokumentit

DOKUMENTTI	TAPAUS 1	TAPAUS 2	TAPAUS 3	TAPAUS 4	TAPAUS 5
Toiminnalliset vaatimukset		Hyöd.	Olemassa	Olemassa	Olemassa
Ei-toiminnalliset vaatimukset		Hyöd.	Hyöd.		Olemassa
Käyttötapauskaavio					Olemassa
Käyttötapaus	Hyöd.	Hyöd.		Hyöd.	Hyöd.
Käyttöliittymäkuvaus	Hyöd.	Hyöd.		Hyöd.	
Rautalankamalli			Hyöd.		Hyöd.
Tilakaavio				Hyöd.	
Sekvenssikaavio				Hyöd.	Hyöd.
Luokkakaavio				Olemassa	

Käyttötapauksia hyödynnetään yhtä tietojärjestelmää lukuun ottamatta kaikissa tapauksissa. Sen sijaan sekä käyttötapauskaavio että käyttötapauskuvaus löytyy vain yhden (tapaus 5) tietojärjestelmän vaatimusdokumentaatiosta. Tapauksessa ei kuitenkaan ole hyödynnetty käyttötapauskaaviota lainkaan, eikä käyttötapauskuvauksiakaan kovinkaan paljon. Järjestelmän toiminnallisuus painottuu enemmän järjestelmien väliseen tekniseen vuorovaikutukseen ja tietojen välitykseen kuin sellaiseen käyttäjän ja järjestelmän väliseen vuorovaikutukseen, jota luontaisesti kuvataan käyttötapauskuvauksilla. Tähän tietojärjestelmään ei ole myöskään ylläpidon aikana kohdistunut sellaisia muutoksia, jotka olisivat vaikuttaneet järjestelmän vähälukuisissa käyttötapauskuvauksissa kuvattuun toiminnallisuuteen.

Sen sijaan tapauksen 3 tietojärjestelmän ylläpidossa käyttötapaukset puuttuvat kokonaan. Deklevan (1992) sekä Yipin ja Robsonin (1991) mukaan puutteellinen dokumentaatio onkin yleinen ongelma tietojärjestelmien ylläpitovaiheessa. Tapauksen 3 tietojärjestelmästä on tehty dokumentteja, mutta dokumentaatio ei kata kaikkea järjestelmän toiminnallisuutta. Tätä puutetta korjaamaan ylläpidossa toivotaan käyttötapausten luomista.

Kahdessa tapauksessa (tapaus 2 ja tapaus 4) käyttötapaukset ovat selkeästi laajempia kuin luvussa 3.3.1. Käyttötapauksia on laajennettu hyvinkin tarkoilla liiketoimintasäännöillä kuin teknisillä yksityiskohdillakin. Näissä tapauksissa käyttötapauksia pidetään pääasiallisena lähtökohtana kaikille ylläpidon aikaisille ylläpitotehtäville, kuten muutoksille, ja siten erittäin olennaisina ja tärkeinä vaatimusdokumentteina. Käyttötapauksia hyödynnetään ylläpidossa eri tarkoituksissa. Niitä käytetään usein selvittämään, kuinka tietojärjestelmän kuuluisi toimia. Käyttötapauksista tutkittiin, toimiko järjestelmä oikein vai oliko kyseessä virhetilanne. Lisäksi käyttötapauksia käytettiin opettelumateriaalina. Molemmat tietojärjestelmät sisältävät paljon toiminnallisuutta, johon liittyy hyvin runsaasti erilaisia yksityiskohtia kuten liiketoiminnan sääntöjä.

Käyttötapausten ensiarvoisuuteen ja monikäyttöisyyteen johtavana seikkana on dokumenttien laajuus ja tarkkuus erityisesti monimutkaisissa järjestelmissä. Käyttötapausten tapahtumienkulkua voidaan hyödyntää järjestelmän opettelussa ja toimintojen hahmottamisessa. Tapahtumankulkua voidaan hyödyntää tehtäessä suuria muutoksia tai laaja-alaista kehitystyötä, sillä tapahtumankulut on kirjoitettu karkealla tasolla. Isoja liiketoiminnallisia muutoksia sovelluksiin tehdään harvemmin, joten tietojärjestelmän hyvin ennestään tunteville tapahtumankulusta ei ole suuresti hyötyä. Pienempiä muutoksia ja korjauksia tehdään sen sijaan usein. Näissä pienemmissä muutoksissa hyödynnetään usein käyttötapauskuvausten tapahtumankulkuun verrattuna yksityiskohtaisempia tietoja. Liki kaikki käyttötapaukset sisältävät toiminnallisen tason lisäksi erilaisia detaljeja järjestelmän toiminnasta. Niitä käytetään usein muutoksien lähtökohtana. Tällaisia detaljeja voivat esimerkiksi olla teknisten yksityiskohtien ohella erilaiset valintavaihtoehdot, tietojen validointi, tallennus tai tietojen muunto integroitavaan tietojärjestelmää varten. Yksityiskohtaisten käyttötapauskuvausten hyödyt tulevat laajuudesta ja tarkkuustasosta.

Käyttötapausten hyöty jää merkittävästi vähäisemmäksi, mikäli dokumentit eivät ole ajantasaisia. Päivittämätön dokumentaatio onkin de Souzan, Anquetilin ja de Oliveiran (2005) sekä Singerin (1998) mukaan yleinen ongelma. De Souzan ym. (2005) mukaan toinen ongelma on dokumenttien laajuus. Käyttötapausten hyödyntämistä tehokkaasti vaikeuttaa se, että liiketoiminnaltaan monimutkaisissa tietojärjestelmissä käyttötapaukset ovat paikoin erittäin pitkiä dokumentteja. Tällöin kärsivät niin luettavuus kuin ylläpidettävyyskin. 30 sivun dokumentista voi olla vaikea hahmottaa tärkeimpiä seikkoja, kun se tärkein hukkuu detaljeihin. Ilman detaljeja taas käyttötapaukset ovat järjestelmän jo tunteville ylläpitäjille liki täysin turhia. Jotta näin laajat käyttötapaukset hyödyttäisivät järjestelmän ylläpitoa tehokkaasti, kuvausten rakenteen ja kielen tulee olla selkeitä. Erityisesti tapauksen 2 tietojärjestelmän ylläpidossa on havaittu tähän

liittyviä ongelmia. Tietojärjestelmän vaatimusdokumentaatiossa on suuri määrä käyttötapauksia, joiden rakenne, taso, laajuus sekä kirjoitustyylin tarkkuus vaihtelevat runsaasti. Toki liian suppea käyttötapaus ei hyödytä kunnolla ketään pidemmällä ajalla, mutta hyvin runsaasta ja rönsyilevästä käyttötapauksesta voi olla hyvin vaikea löytää kaikkia kokonaisuuteen liittyviä asioita tai olennaisimpia asioita. Pitkät vapaamuotoiset tekstit tekevät dokumenteista myös vaikea ylläpitää. Dokumenttien rakennetta on hyvä pohtia ja koestaa samalla, kun selkeyttää tekstiä. Pitkää tekstiä on helpompi lyhentää kuin toisinpäin.

Viimeisessä tapauksessa (tapaus 1) tietojärjestelmän jokainen käyttötapaus on yhdistetty käyttöliittymäkuvaukseen, ja suuri osa ylläpidon aikaisista muutoksista on kohdistunut dokumentin käyttöliittymäkuvaus-osion sisältöön. Jokainen näistä käyttötapauksen ja käyttöliittymäkuvauksen yhdistävistä dokumenteista sisältää paljon asiaa. Ylläpitoa hyödyttää näissä dokumenteissa varmasti moni osa-alue. Dokumenttien laajuus ja yksityiskohtaisuus toimivat lähtökohtana kaikille ylläpidon aikaisille muutoksille. Isommatkin muutokset pystytään tekemään näihin dokumentteihin. Isommissa muutoskokonaisuuksissa on todennäköistä, että useampi dokumentti muuttuu. Ylläpitotyötä pystytään kuitenkin jakamaan dokumenttien perusteella helposti sekä RUP-kehyksen että Scrum-mallin mukaisiksi tehtäviksi. Samalla tehtäviä voidaan helposti jakaa eri ylläpitäjien kesken. Myös dokumenttien laajuus palvelee järjestelmän entuudestaan tuntevia ylläpitäjiä. Hakusanoilla voi etsiä dokumentin sisältä oikean tiedon, ja ajan kanssa myös dokumenttien hieman poikkeava rakenne on tullut tutuksi. Tutkimuksen aikana järjestelmän ylläpitoon osallistui järjestelmän hyvin tuntevia henkilöitä, mutta tilanne tulee muuttumaan. Nykyisetkin ylläpitäjät pohtivat, miten dokumentit palvelee uusia ylläpitäjiä. Näin laajoista dokumenteista voi olla vaikea hahmottaa kokonaiskuvaa kaikkien yksityiskohtien keskeltä. Lisäksi dokumenteissa saattaa esiintyä yleisten tietojen tai sääntöjen kohdalla vanhentunutta tietoa, koska järjestelmän ylläpitoon osallistuvat liiketoiminnasta sekä ohjelmista vastaavat henkilöt tuntevat järjestelmän hyvin. Dokumentit sisältävät paljon myös liiketoimintalähtöistä, vapaamuotoisesti kirjoitettua pitkää tekstiä, joka osaltaan vaikeuttaa dokumenttien ylläpidettävyyttä sekä järjestelmän toiminnan hahmottamista. Nykyisellään dokumentit vaikuttavat toimivan erittäin hyvin, mutta vähemmän järjestelmästä tietävänä dokumentteja piti lukea useampaan kertaan, jotta kokonaisuus avautui. Voisi olla, että ainakin uudet ylläpitäjät hyötyisivät rakenteisemmista kuvauksista, mahdollisesti jopa käyttötapausten ja käyttöliittymäkuvausten erottamisesta. Vaarana on kuitenkin todennäköisesti etenkin uusia ylläpitäjiä hyödyttävien yleisten ja informatiivisten liiketoimintaa kuvaavien sääntöjen, tekstien ja pohja- ja historiatietojen väheneminen, joiden avulla voidaan lisätä yleistä ymmärrystä järjestelmän liittyvästä liiketoiminnasta. Hyötynä saattaisi sitä vastoin olla tietojen moninkertaisen dokumentoinnin väheneminen.

Tapauksen 1 lisäksi myös tapauksissa 2 ja 4 hyödynnetään käyttöliittymäkuvauksia. Kahdessa muussa tapauksessa (tapaukset 3 ja 5) hyödynnetään rautalankamalleja. Käyttöliittymään liittyvät kuvaukset onkin tämän tutkimuksen

ainoa vaatimusdokumenttityyppi, jota hyödynnetään jokaisen tapauksen ylläpidossa jollain tapaa. Tapauksessa 5 muutokset eivät ole kohdistaneet käyttöliittymään liittyviin toiminnallisuuksiin, ja tietojärjestelmä on ylläpitäjille entuudestaan tuttu. Näin ollen rautalankamallista ei ole ollut juuri hyötyä. Sen sijaan tapauksessa 3 rautalankamallit ovat olleet tietojärjestelmän ylläpidon keskiössä. Rautalankamallit ovat olleet ainoa ylläpidossa käytetty toiminnallisiin vaatimuksiin liittyvä dokumentti. Tämän suppeahkon tietojärjestelmän toiminnallisuus on käyttöliittymäpainotteinen, joten rautalankamallien hyödyntäminen on luonnollista. Rautalankamallit hyödyntävät ylläpitoa oppimisnäkökulmasta sekä muutoksien lähtökohtana. Malleissa kuvataan vain käyttöliittymän esittämiskerrosta, eikä siinä kuvata esimerkiksi käyttöliittymän syöttökenttiin liittyvä tarkistuksia tai muita toimintoja. Rautalankamallien piirtämiseen ei myöskään ole sopivaa välinettä. Välineongelmat ovat usein organisaatioiden ratkaistavissa. Rautalankamallien sisällön osalta ylläpitäjien tulisi pohtia, täydennetäänkö nykyisiä rautalankamalleja vai pystytäänkö puuttuvat tiedot kuvaamaan esimerkiksi järjestelmän toiminnallisissa kuvauksissa kuten käyttötapauksissa.

Varsinaisista käyttöliittymäkuvauksista tapauksen 2 tietojärjestelmän ylläpidossa hyödynnetään käyttöliittymäkuvauksia ylläpidossa usein ja erityyppisissä tehtävissä. Tietojärjestelmässä on paljon käyttöliittymätoiminnallisuutta, joten tarkat ja laajat käyttöliittymäkuvaukset toimivat hyvin ylläpitotehtävissä. Näin tarkat kuvaukset vaatinevat ylläpidon aikana myös jonkin verran ylläpitotyötä. Kuvauksia on kuitenkin helppo hyödyntää, sillä ne vastaavat todellisuutta hyvin toisin kuin esimerkiksi rautalankamallit. Kuvaukset tukevat hyvin organisaation tiettyihin tehtäviin erikoistuneita rooleja ja vastuita. Erikoistuminen aiheuttaa kuitenkin ongelmia dokumenttien päivityksessä, sillä käyttöliittymääkuvauksiin liittyvien välineiden käyttöä on rajoitettu. Tällöin käyttöliittymäkuvaukset voivat jäädä esimerkiksi resursoinnin vuoksi päivittämättä. On selvää, että vain olemassa olevat ja ajantasaiset käyttöliittymäkuvaukset voivat hyödyttävät ylläpitoa. Tapauksen 4 käyttöliittymäkuvaukset ovat hyvin samankaltaisia tapauksen 4 tietojärjestelmän käyttöliittymäkuvausten kanssa: ne ovat tarkkoja ja laajoja, ja niitä hyödynnetään erilaisissa ylläpitotehtävissä. Tämä tietojärjestelmä ei sisällä niin paljon käyttöliittymään liittyvää toiminnallisuutta, joten vaatimusdokumentaationkin painopiste on muissa dokumenteissa.

Tilakaavioita on hyödynnetty tapauksen 4 tietojärjestelmän ylläpidossa, jossa ne on havaittu erityisen hyödyllisiksi. Tilakaavioita voidaan luoda, mikäli tietojärjestelmässä on jokin tai joitain objekteja, joilla on olemassa tila. Tapauksen 4 tietojärjestelmässä sellaisia on. Järjestelmän ylläpidossa hyödynnetyt tilakaaviot kuvaavat havainnollisella tavalla joidenkin järjestelmän toiminnallisuuksiin liittyvää liiketoimintalogiikkaa. Tilojen, tilasiirtymien ja tarkentavien kuvausten ja tietojen avulla yhden A4-paperiarkin kokoiseen kuvaan saadaan kattavasti tietoa, josta on nopea silmäillä ja tarkistaa tietoja. Kuten Briandkin (2003) on todennut, sidosryhmät tai uudet ylläpitäjät voivat saada tilakaavioista myös nopeasti ehjän kuvan järjestelmän toiminnasta. Silloin kun tilakaavioita voidaan luoda, tilakaavioiden tulisi kuvata pelkkien objektin tilojen lisäksi tarpeeksi kuvaavasti ja

kattavasti myös tilasiirtymiä. Lisäksi kaavioita on kannattavaa täydentää erilaisilla tarkentaville selityksillä tarpeen mukaan. Näillä tavoin piirrettyjä tilakaavioita voitaisiin selkeästi hyödyntää laajemminkin.

Tilakaavioitakin enemmän sidosryhmiä saattaa hyödyttää sekvenssikaaviot. Sekvenssikaavioista on hyötyä erityisesti, kun järjestelmään integroituu toisia tietojärjestelmiä. Tapausten 4 ja 5 tietojärjestelmien ylläpidossa hyödynnetään sekvenssikaavioita, jotka ovat molemmissa järjestelmissä karkeamman tason sekvenssikaavioita kuin luvussa 3.3.4 esitetty sekvenssikaavio. Näiden kahden järjestelmän sekvenssikaavioissa objekteina on kuvattu tietojärjestelmän omia komponentteja, kuten esimerkiksi eri sovellukset ja tietokanta, sekä tietojärjestelmään integroituvat järjestelmät. Objektien välinen vuorovaikutus on kuvattu tarpeen vaatimalla tarkkuudella.

Sekvenssikaavioissa olennaista hyötyä tuo oikea tarkkuustaso. Palvelukeskeisessä arkkitehtuurissa (service-oriented architecture, SOA) tietojärjestelmillä voi olla todella suuri määrä integraatioita, mikä voi tehdä järjestelmän toiminnan tai virheenselvittelyn tai liiketoimintaketjun seuraamisen kovin hankalaksi, jolloin on kannattavaa hyödyntää sekvenssikaavioita. Sekvenssi kaavio on yksi UML:n (Unified Modeling Language) yksi kaaviotyyppi, ja se auttaa Arisholmin, Briandin, Howen ja Labichen (2006) mukaan kokonaiskuvan ja rakenteen hahmottamisessa. Sekvenssikaavioilla voidaan selkeyttää eri tietojärjestelmien ja/tai yhden tietojärjestelmän komponenttien välistä vuorovaikutusta ja vastuita. Kaaviosta voidaan nähdä nopeasti järjestelmän tiettyyn toiminnallisuuteen liittyvät, tähän tietojärjestelmään integroituvat järjestelmät, tietojärjestelmän rakenne ja komponentit sekä järjestelmän sisäinen toiminta ylätasolla. Mikäli sekvenssikaavioita luotaisiin tarkemmalla tasolla, todennäköisesti niiden ylläpidettävyys kärsisi. Myöskään edellä mainittuja hyötyjä ei saavuteta, jos sekvenssikaavio kuvaa vain rajattua osaa sovelluksen toimintaa. Vaihtoehtoisesti jos kaavio kuvataan tarkemmin koko järjestelmän tasolta, se usein laajenisi niin luku- ja silmäilykelvottomaksi (esimerkiksi lukuisten A4-arkin mittaiseksi), että sen hyödyllisyys kärsisi.

Luokkakaavioita on olemassa vain tapauksen 4 tietojärjestelmän vaatimusdokumentaatiossa. Luokkakaavioita ei kuitenkaan ole hyödynnetty. Suoraan ohjelmakooditoteutusta vastaavaa kaaviota ei kannattane luoda erillisenä kaaviona, sillä kaaviot voidaan generoida helposti tarpeiden mukaan ohjelmakoodista. Näin luokkakaaviot pysyvät myös aina ajantasaisina. Sen sijaan käsitteellisen tason kaavioita kannattanee luoda, mutta usein niitä käytettäneen enemmän tietojärjestelmän kehitysvaiheessa. Jos niitä hyödynnetään ylläpidon aikana, hyödyntäjät ovat todennäköisesti organisaatiossa esim. liiketoiminnasta vastaavat henkilöt.

7.2 Havaitut hyödyt ja puutteet

Ylläpidossa hyödynnettyjen dokumenttien tärkein hyöty on toimia ylläpidon aikaisten muutosten lähtökohtana. Liiketoiminnasta vastaavat henkilöt vastaavat

usein vaatimusdokumenttien päivittämisestä, ja muutosten havainnollistamiseen ja niitä käytetään Vuoren (2010) ehdotuksen tapaan myös muutoksen tarkastamiseen. Varsinaisten muutosten lisäksi dokumentaatiota käytetään myös tutkittaessa tietojärjestelmää esimerkiksi selvitettäessä, kuinka tietojärjestelmän kuuluu toimia eri tilanteissa. Liiketoiminnaltaan monimutkaisissa järjestelmissä tässä on hyötyä muun muassa virhetilanteita selvittäessä ja virheen sijaintia paikallistettaessa. Joskus dokumenteista voi selvitä harvoin toteutuva skenaario, jolloin järjestelmä toimii oikein, mutta tilanne on ylläpitäjille entuudestaan tuntematon. Dokumentaatiosta hyödynnetään niin toiminnallisuuksien karkean ja tarkan tason kuvauksia sekä tietoja järjestelmän integraatioista. Muutoinkin dokumentaatiota käytetään ylläpidossa muistin tukena ja jatkeena (Vuori, 2010). Dokumentit ovat erityisen hyödyllisiä myös silloin, kun järjestelmän ylläpitoon siirtyy uusia henkilöitä. Uudet ylläpitäjät voivat hyödyntää dokumentaatiota ymmärtääkseen järjestelmää ja sen toimintaa paremmin (Tryggeseth, 1997a; Kajko-Mattsson, 2001). Hyvä dokumentaatio vähentää henkilöstöön kohdistuvaa painetta monin tavoin.

Yleisimmäksi ongelmaksi havaittiin päivittämätön dokumentaatio. Sama ongelma on havaittu hyvin yleisesti muuallakin, ja sen ovat raportoineet myös muun muassa de Souza, Anquetil ja de Oliveira (2005), Singer (1998), Smith, Thomas & Tilley (2001) sekä Lethbridge, Singer & Forward (2003). Päivittämätöntäkin dokumentaatiota suurempi ongelma lienee se, jos dokumentaatiota löydetä ollenkaan, kuten Singer (1998) sekä Das, Lutters ja Seaman (2007) ovat havainneet. Joskus dokumentaatio on hyvin rajallinen, tai sitä ei ole edes olemassa (Sharon, 1996). Tutkituissa tapauksissa ei esiintynyt dokumentaation täydellistä puuttumista ja ongelmia löytää dokumentaatio oli vain yhdessä tapauksessa. Tässä tapauksessa kyseisiä vaatimusdokumentteja ei kaivattu. Sen sijaan toisessa tapauksessa vaatimusdokumentaatio oli selkeästi vajavainen, ja siitä puuttui dokumentteja järjestelmän toiminnallisuuksista.

Vaillinaisen dokumentaation ohella puutteita esiintyi yksittäisissä dokumenteissa. Tätä suuremmaksi ongelmaksi paljastui dokumenttien sisältö. Das, Lutters ja Seaman (2007) tiivistävät dokumenttien sisältöongelman kahteen sanaan: kirjoitustyyli ja rakenne. Tutkimuksessa tarkastelluissa dokumenteissa oli selkeitä puutteita dokumenttien kirjoitustyyleissä. Kirjoitustyyli oli vaihtelevaa, ja paikoin teksti oli liian pitkää ja rönsyilevää, hankalasti luettavaa ja vaikeaa ymmärtää edes huolellisesti lukien. Ajoittain teksti myös jätti asioita tulkinnan varaan. Myös dokumenttien rakenne oli vaihtelevaa. Samalla esittämistavalla kuvattujen, kuten käyttötapausten, dokumenttien rakenne erosi eri järjestelmien välillä paljon. Joissain tapauksissa eroja oli myös yhden tietojärjestelmän dokumenttien kesken. Dokumentin hyvä rakenne auttaa etsimään tietoa. Lethbridgen, Singerin ja Forwardin (2003) mukaan hyödyllisen sisällön löytäminen dokumenteista voi olla niin haastavaa, ettei sitä edes yritetä etsiä. Tässä tutkimuksessa ilmeni vaikeuksia oikean sisällön ja tietyn toiminnallisuuden määrittelyn löytämisessä. Ylläpitäjät kuitenkin hyödynsivät välineiden hakutoimintoja etsimisessä.

Näiden toimintojen hyödyntäminen edellyttää kuitenkin, että ylläpidossa ei ole ns. välineongelmia. Välineet voivat aiheuttaa ongelmia ylläpidossa (Dekleva, 1992; Sousa & Moreira, 1998; Yip & Robson, 1991). Tryggesethinkin (1997a) mukaan on välineet ovat tärkeitä dokumentaation tehokkaassa hyödyntämisessä. Tässä tutkimuksessa välineisiin liittyviä ongelmia oli muun muassa välineiden vaihtuvuus, välineiden käyttörajoitukset sekä liian monet välineet, jolloin ei tiedetä mitä on ja mitä ylläpidetään missäkin välineessä. Myös puuttuva tai epäsopiva väline koettiin ylläpidon ongelmaksi.

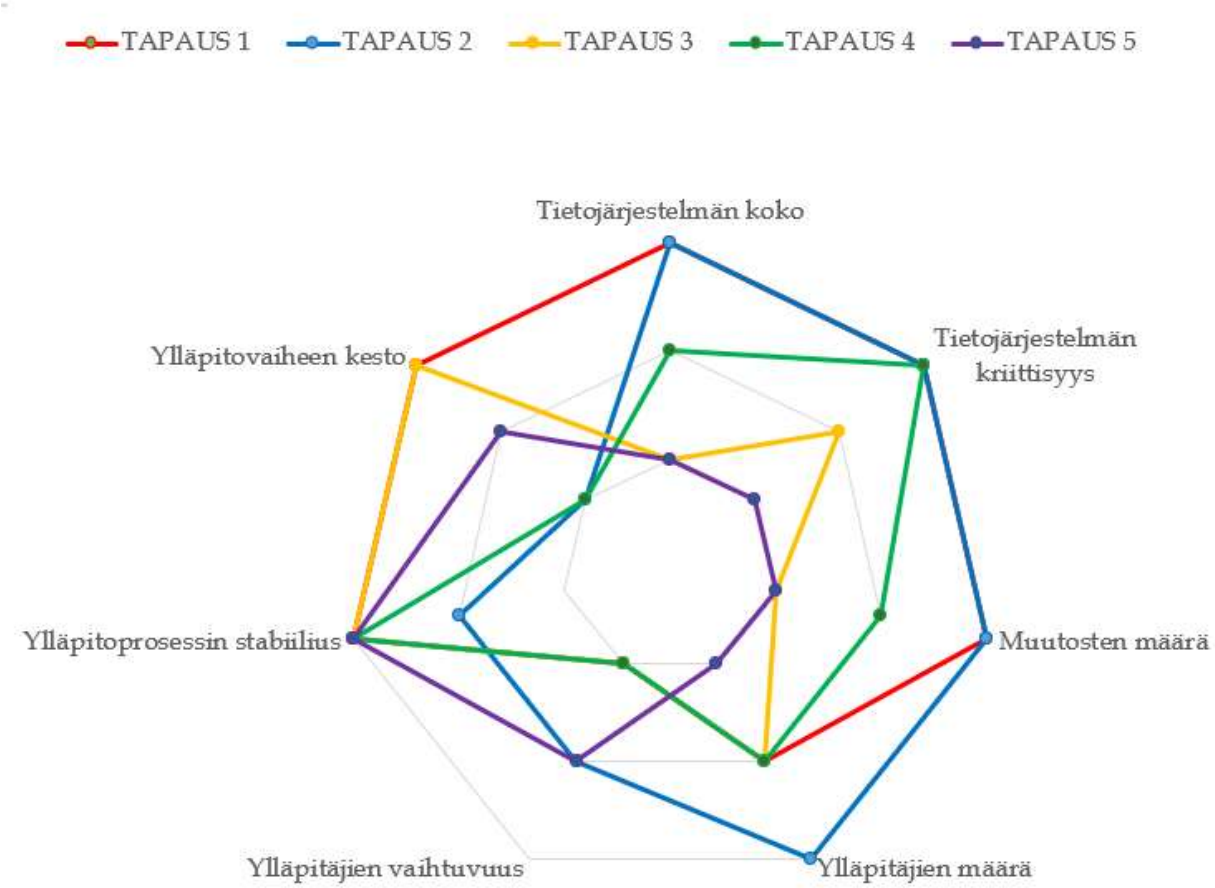
Ylläpidossa on monia erilaisia dokumentaatioon liittyviä ongelmia. Lethbridgen, Singerin & Forwardin (2003) havaitsema ongelma löytyi myös yhdestä tutkimuksen tapauksesta: liikaa dokumentteja. Liian suuri dokumenttien määrä vaikeuttaa oikean asian löytämistä. Se lisää myös dokumenttien ylläpitoon käytettyä työmäärää, ja siten vähentää ylläpidon tehokkuutta. Liian suuren dokumenttimäärän voidaan katsoa kuitenkin olevan positiivinen ongelma, sillä dokumenttien määrää on huomattavasti helpompi vähentää kuin kasvattaa.

Tässä tutkimuksessa löydetyt ongelmat näyttävät olevan muillekin organisaatioille tutuksi tulleita ongelmia. Nämä ongelmat ovat myös usein ratkaistavissa kohtuullisella työpanoksella. Tärkeintä on ottaa dokumentaation ylläpito ja kehittäminen osaksi ylläpitoprosessia. Dokumentaatio on tärkeä osa tietojärjestelmän ylläpitoa. Dokumentaatio, kuten ohjelmakoodikin, on olennainen osa ylläpitoa, ja sen laatuun ja hyödytettävyyteen tulee kiinnittää ylläpidossa aktiivisesti huomiota. Aktiivisen otteen lisäksi vaaditaan pohdintaa siitä, mikä on riittävää, tarpeellista ja hyödyllistä sekä pohdintojen tulosten realisoimista kyseisen tietojärjestelmän ylläpitoa hyvin palvelevaksi, ylläpidettäväksi dokumentaatioksi.

7.3 Tutkimuksen validiteetti ja reliabiliteetti

Tässä tapaustutkimuksessa tutkittiin viittä eri tietojärjestelmää, joista osassa haastateltiin useampaa henkilöä. Tapauksissa oli edustettuna hyvin organisaation erilaiset tietojärjestelmät, jolloin tutkimuksen kohderyhmän voidaan katsoa

olleen onnistunut. Kuviossa 24 on esitetty tutkimukseen valittujen tapausten vertailu seitsemän eri ominaisuuden perusteella. Kuviosta on havaittavissa, että tietojärjestelmät edustavat hyvin kattavaa, heterogeenistä perusjoukkoa.



KUVIO 23 Tutkimukseen valittujen tapausten vertailu

Tutkimuksessa haastatelluille esitetyt kysymykset tuottivat pääsääntöisesti hyviä tai erittäin hyviä vastauksia. Osassa kysymyksiin pyydettiin tarkennuksia tai lisätietoja erityisesti silloin, kun käsitteen selityksestä ja esimerkeistä huolimatta vaatimusdokumentaation käsitettiin kattavan virheellisesti ainoastaan toiminnalliset ja ei-toiminnalliset vaatimukset. Tutkimuksessa valittiin haastateltaviksi kokeneita ylläpitäjiä. Kokeneiden ylläpitäjien haastattelu oli tutkimuksen alussa päätetty tietoinen valinta, jottei tuloksissa korostuisi, mitä vaatimusdokumenttien esittämistapoja on hyödynnetty erityisesti uuden tietojärjestelmän opettelussa ja sen ylläpidossa.

Tutkimukseen valittiin viisi tietojärjestelmää, ja haastateltaviksi valittiin kokeneita ylläpitäjiä. Jos tutkimus toteutettaisiin uudelleen käyttäen samoja järjestelmiä ja ylläpitäjiä, tulokset olisivat hyvin samankaltaiset. Jos tutkimukseen valittaisiin eri tietojärjestelmät ja/tai ylläpitäjät, todennäköisesti tutkimuksessa havaittaisiin varianssia yksittäisten järjestelmien osalta esimerkiksi käytetyissä vaa-

timusdokumenteissa ja havaituissa hyödyissä ja ongelmissa. Tästäkin syystä tutkimuksessa tutkittiin useita tietojärjestelmiä ja haastateltiin useita ylläpitäjiä. Näin tutkimuksen tulokset saatiin tarpeeksi kattavaksi ja siten hyödyllisiksi. Tutkimuksen tutkimusmenetelmän valinta oli onnistunut.

Tutkimusmenetelmä ja sen soveltaminen pyrittiin kuvaamaan luvussa 5 mahdollisimman läpinäkyvästi ja riittävällä tarkkuudella. Luvussa kuvattiin selkeästi tässä tutkimuksessa käytetyn tutkimusprosessin eri vaiheet, ja miten ne tullaan toteuttamaan. Erityistä huomiota kiinnitettiin tutkimusaineistoon keruuvaiheeseen, joka kuvattiin kattavasti.

7.4 Jatkotutkimusaiheita

Tämä tutkimus antoi vastauksia joihinkin kysymyksiin, mutta niiden ohella se herätti myös lisäkysymyksiä. Tutkimukseen liittyvien tapausten ylläpidossa hyödynnettiin pääasiassa käyttötapauskuvauksia. Olisi mielenkiintoista tutkia laajemmin erilaisten esittämistapojen hyödyntämistä ylläpidossa. Sitäkin mielenkiintoisempaa olisi jatkaa käyttötapausten hyödyntämisen tutkimista. Tutkittujen tapausten käyttötapaukset poikkeavat tutkielmassa esitetystä mallista, ja osassa tapauksia erot ovat todella suuret. Eroja oli myös eri järjestelmien käyttötapausten välillä. Näitä eroja, käyttötapausten laajuutta, rakenteita, tarkkuustasoa ja sisällön erilaisia ilmaisukeinoja voisi tutkia ylläpidonaikaisen hyödyntämisen näkökulmasta. Tällainen tutkimus saattaisi tuoda yksityiskohtaisemmin ja selkeämmin esille ne seikat, jotka tekevät nimenomaan käyttötapauksista hyödyllisiä. Näitä seikkoja voisi hyödyntää kehittämään organisaation ohjeistusta. Tällaisessa tutkimuksessa voisi hyödyntää käyttötapausten soveltamiseen liittyviä tutkimuksia, kuten Tiwarin & Guptan (2013) tutkimusta kahdeksasta eri käyttötapauksen mallipohjasta.

Käyttötapaukset eivät ole ainoa esittämistapa, jonka tutkimiselle löytyisi motivaatio. Sovelluskehityksessä ketterä kehittäminen on yhä suositumpi tietojärjestelmän kehittämismenetelmä, ja siinä hyödynnetään usein käyttäjätarinoita. Tässä tutkimuksessa ei yksikään tapaus hyödyntänyt käyttäjätarinoita, mutta myös tässä organisaatiossa niiden käyttö lisääntyy koko ajan. Käyttäjätarinat eroavat käyttötapauksista paljon, ja laajoissa, liiketoiminnaltaan hyvin monimutkaisissa tietojärjestelmissä käyttäjätarinat eivät istune niin helposti ylläpitovaiheeseen kuin käyttötapaukset. Kysymys onkin, miten suuren tietojärjestelmän ylläpito saadaan toimimaan käyttäjätarinoiden avulla?

Järjestelmän toiminnallisuuksien tarkka dokumentointi herättää myös kolmannen mielenkiintoisen jatkotutkimusaiheen, jonka avulla voitaisiin kehittää ylläpidon tehokkuutta ja kasvattaa ylläpitotyön mielekkyyttä: JavaDoc-kommenttien käyttö vaatimusdokumentaationa. De Souzan, Anquetilin ja de Oliveiran (2005) mukaan tutkimukset osoittavat, että lähdekoodi ja sen kommentit ovat tärkeimmät artefaktit ylläpidettävän järjestelmän ymmärtämisessä. Samaan pyritään myös dokumentaatiolla. Kaikkea dokumentaatiota ei voi, eikä kannata korvata kommenteilla. Sen sijaan herää kysymys, voisiko JavaDoc-kommentteja

käyttää korvaamaan jotain osaa dokumentaatiosta esimerkiksi tarkistusten tai sääntöjen dokumentoimista. Kenties näin dokumentaatio säilyisi ajantasaisempana lähdekoodiin verrattuna. Erityisen mielenkiinnon kohteenani olisi tutkia, miten vaatimusmäärittelyä, suunnittelua ja lähdekoodia voisi tarkastella saman asian eri näkökulmina nykyistä enemmän toisiinsa sidottuina. Tällainen lähestymistapa voisi soveltua esimerkiksi tutkitun kaltaisen suuren organisaation vahvasti integroituvien, monimutkaisien tietojärjestelmäkokonaisuuksien osien dokumentoimiseen erityisesti palvelukeskeisessä arkkitehtuurissa tai mikropalveluarkkitehtuurissa. Soveltumista voisi tutkia dokumentaation kannalta, mutta mielenkiintoisempaa olisi tutkia asiaa myös sovelluksen kannalta. Tutkimuksessa voisi esimerkiksi selvittää, auttaako dokumentaation kiinteämpi sitominen lähdekoodiin edellä mainittujen ylläpidettävyydessä tai kehittämisessä muun muassa selkeän ja kestävän rakenteen näkökulmasta.

8 YHTEENVETO

Tämän tutkielman tavoitteena on ollut selvittää, miten viittä tietojärjestelmää ylläpidetään ja millä tavoin niissä hyödynnetään tietojärjestelmävaatimuksia koskevia dokumentteja. Tässä tarkoituksessa tutkielmassa on ensin kuvattu vaatimusmäärittelyprosessi, siihen liittyvät vaiheet sekä kuvattu, miten vaatimusmäärittelyn aktiviteetit sijoittuvat kolmen eri tietojärjestelmän kehittämismenetelmän, vaihejakomallin, RUP-kehyksen ja Scrum-menetelmän, vaiheisiin. Tämän jälkeen tutkielmassa kuvattiin vaatimusten yleisimpiä esittämistapoja kuten käyttötapauksia ja eritasoisia käyttöliittymän kuvauksia. Näitä esittämistapoja vertailtiin lyhyesti toisiinsa yleisluonteisuuden ja formaalisuuden mukaan nelikentän avulla.

Vaatimusmäärittelyn ja vaatimusten esittämistapojen jälkeen kuvattiin järjestelmän ylläpidon konteksti ja ylläpidon keskeiset toimijat rajapintoineen. Järjestelmän julkaisun jälkeisiä eli ylläpidon aikaisia järjestelmään kohdistuvia mutuoksia on erityyppisiä: korjaavia, mukauttavia, täydentäviä tai ennakoivia. Näitä tehtäviä suoritetaan ylläpitoprosessissa. Ylläpitoprosesseja on lukuisia, joista esimerkkeinä on kuvattu pikakorjausmalli, iteratiivinen parannusmalli sekä IEEE 1219-1998 -standardi järjestelmän ylläpidolle.

Tässä tutkielmassa tutkittiin erään julkishallinnon organisaation viitta ylläpidossa olevaa tietojärjestelmää. Nämä viisi tietojärjestelmää erosivat toisistaan eri ominaisuuksiltaan kuten kooltaan, ylläpitovaiheen kestoltaan ja muutoksien määrältään. Tietojärjestelmien ylläpidossa olennaisimmiksi dokumenteiksi katsottiin käyttötapaukset sekä käyttöliittymän eritasoiset kuvaukset. Lisäksi hyödynnettiin mm. sekvenssi- ja tilakaavioita. Ylläpidossa vaatimusdokumenttien tärkein tehtävä oli toimia ylläpidon aikaisten muutosten lähtökohtana. Dokumentteja hyödynnettiin myös järjestelmän toimintojen opetteluun ja toiminnan oikeellisuuden tarkistamiseen sekä tietojen etsimisessä ja erilaisessa tiedonhankinnassa. Dokumentaation hyödyntämiseen liittyi myös ongelmia. Näitä olivat mm. päivittämätön tai kokonaan puuttuva dokumentaatio, dokumenttien heikko ja vaihteleva rakenne sekä tekstin rönsyilevyys ja selkeyden puute. Myös välineisiin liittyi ongelmia. Dokumentaatioon liittyviä ongelmia voidaan pyrkiä ratkaisemaan eri tavoin. Dokumentaation ylläpito ja kehittäminen tulee ottaa osaksi järjestelmien ylläpitoa, ja sen tarkoitusta, rakennetta ja laatua pitää tarkastella ja korjata aktiivisella otteella. Tärkeintä on tietää, mitä ylläpidossa tarvitaan.

Tähän tutkimukseen liittyy joitakin rajoituksia. Tähän tutkimukseen liittyvien tapausten ylläpidossa ei hyödynnetty käyttäjätarinoita, vaikka ne ovat yleistyneet paljon viime vuosina. Tutkittujen tietojärjestelmien ylläpidossa hyödynnettiin suppeaa määrää eri esittämistapoja. Eri esittämistapojen lisäksi voisi tutkia nyt hyödynnettyjä vaatimusten esittämistapoja tarkemmin. Tässä tutkimuksessa keskityttiin tutkimaan kokeneita ylläpitäjiä. Järjestelmien uusilla ylläpitäjillä olisi todennäköisesti erilaiset tarpeet sekä tuore näkökulma ylläpitoon.

LÄHTEET

Abran, A. & Moore, J. W. (2004). *Guide to the Software Engineering Body of Knowledge (SWEBOK)*.

Alexander, I. (2002). Initial Industrial Experience of Misuse Cases in Trade-off Analysis. Teoksessa D. C. Martin (toim.), *Proceedings of the IEEE Joint International Conference on Requirements Engineering (RE'02*) (s. 61-68). Los Alamitos: IEEE Computer Society.

Ambler S. (2002). *Agile Modeling: Effective Practices for eXtreme Programming and the Unified Process*. John Wiley & Sons, New York.

April, A., Huffman Hayes, J., Abran, A. & Dumke, R. (2005). Software Maintenance Maturity Model (SMmm): the software maintenance process model. *Journal of Software Maintenance and Evolution: Research and Practice, 17(3)*, 197-223.

Arisholm, E., Briand, L., Howe, S. & Labiche, Y. (2006). The impact of UML documentation on software maintenance: an experimental evaluation. *IEEE Transactions on Software Engineering, 32(6),* 365-381.

Arnowitz, J., Arent, M. & Berger, N. (2007). *Effective Prototyping for Software Makers.* Haettu 7.5.2020 osoitteesta: http://web.b.ebscohost.com.ezproxy.jyu.fi/ehost/ebookviewer/ebook/bmxlYmtfXzE4NjA0N19fQU41?sid=32945ae2-3bd7-4a47-bfc3-6d9f6edfcf4f@pdc-v-sessmgr01&vid=0&format=EB&rid=1

Attarha, M. & Modiri, N. (2011). Focusing on the Importance and the Role of Requirement Engineering. Teoksessa H. Rim Choi, S. Goo Hong & F. Ko (toim.), *2011 Proceedings of the 4th Internation Conference on Interaction Sciences: IT, Human and Digital Content (ICIS 2011)* (s. 181-184). Busan: IEEE Computer Society.

Basili, V. R. (1990). Viewing maintenance as reuse-oriented software development. *IEEE Software, 7(1),* 19-25.

Briand, L. V. (2003). Software Documentation: How Much Is Enough? Teoksessa Danielle C. Martin (toim.), *Proceedings of the seventh European Conference on Maintenance and Reengineering, 2003* (s. 13–15). Los Alamitos, CA: IEEE Computer Society.

Cioch, F. A., Palazzolo, M. & Lohrer, S. (1996). A documentation suite for maintenance programmers. Teoksessa Bob Werner (toim.), *Proceedings of 1996 International Conference on Software Maintenance* (s. 286–295). Los Alamitos: IEEE Computer Society.

Das, S., Lutters, W. & Seaman, C. (2007). Understanding documentation value in software maintenance. Teoksessa Eser Kandogan & Patricia M. Jones (toim.), *Proceedings of the 2007 symposium on Computer human interaction for the management of information technology, CHIMIT '07* (s. 2–es). New York, NY: ACM.

Dekleva, S. (1992). Delphi study of software maintenance problems. Teoksessa *Proceedings of Conference on Software Maintenance* (s. 10-17). Los Alamitos: IEEE Computer Society.

Denger, C., Berry, D. M. & Kamsties, E. (2003). Higher Quality Requirements Specifications through Natural Language Patterns. Teoksessa R. S. Bilof (toim.), *Proceedings of IEEE International Conference on Software: Science, Technology and Engineering 2003 (SwSTE'03)* (s. 80-90). Herzlia: IEEE Computery Society.

Evans, P., Sherin, A. & Lee, I. (2013). *The graphic design reference & specification book. Rockport Publishers.* Haettu 7.5.2020 osoitteesta: http://web.a.ebscohost.com.ezproxy.jyu.fi/ehost/ebookviewer/ebook/bmxlYmtfXzY5OTAzOF9fQU41?sid=2214f6f5-2f0d-4e80-a74f-c70da1d9674c@sessionmgr4006&vid=0&format=EB&rid=1

Gallardo-Valencia, R. E., Olivera, V. & Sim, S. E. (2007). Are Use Cases Beneficial for Developers Using Agile Requirements? Teoksessa *Fifth International Workshops on Comparative Evaluation in Requirements Engineering (CERE'07)* (s. 11-22). New Delhi: IEEE Computer Society.

Goldsmith, R. F. (2004). *Discovering Real Business Requirements for Software Project Success.* Norwood, MA: Artech House.

Grupp, P. (2003). *Software Maintenance*. River Edge, NJ: World Scientific. Haettu 13.7.2015 osoitteesta http://site.ebrary.com/lib/jyvaskyla/detail.action?docID=10083771.

Haikala, I. & Mikkonen, T. (2011). *Ohjelmistotuotannon käytännöt* (12. uudistettu painos). Helsinki: Tammi.

Hull, E., Jackson, K. & Dick, J. (2011). *Requirements Engineering* (3. painos). London: Springer.

IEEE 1219-1998 (1998). *IEEE Standard for Software Maintenance*. The Institute of Electrical Standards Organization.

ISO/IEC (1999). ISO/IEC FDIS 14764:1999(E) *Software Engineering – Software Maintenance*. Geneva: International Standards Organization.

Kajko-Mattsson, M. (2001). The state of documentation practice within corrective maintenance. Teoksessa Bob Werner (toim.), *Proceedings of IEEE International Conference on Software Maintenance 2001* (s. 354-363). Los Alamitos, CA: IEEE Computer Society.

Kittlaus, H.-B. & Clough, P. N. (2009). *Software Product Management and Pricing: Key Success Factors for Software Software Organizations*. Heidelberg: Springer-Verlag.

Kotonya, G. & Sommerville, I. (1997). *Requirements Engineering*. West Sussex: John Wiley & Sons.

Krogstie, J. & Solvberg, A. (1994). Software Maintenance in Norway: A Survey Investigation. Teoksessa Hausi A. Müller & Mari Georges (toim.) *Proceedings of the International Conference on Software Maintenance (ICSM 1994)* (s. 304-313). Victoria, BC: IEEE Computer Society.

Krogstie J. & Sövberg A. (1996). A classification of methodological frameworks for computerized information systems support in organizations. Teoksessa S. Brinkkemper, K. Lyytinen & R. Welke (toim.), *Proceedings of the IFIP TC8 WG 8.1/8.2 Working Conf. on Method Engineering: Principles of Method Construction and Tool Support* (ss. 278-295), London: Chapman & Hall.

Lamsweerde van, A. (2009). *Requirements Engineering: From System Goals To UML Models To Software Specifications*. West Syssex: John Wiley & Sons.

Laplante, P. A. (2009). *Requirements Engineering for Software and Systems*. Boca Raton, FL: Taylor & Francis Group.

Layzell, P. J. & Macaulay, L. A. (1994). An investigation into software maintenance—perception and practices. *Journal of Software Maintenance: research and practice, 6(3),* 105-120.

Leffingwell, D. (2007). *Scaling Software Agility: Best Practices for Large Enterprises*. Boston, MA: Addison-Wesley.

Lethbridge, T., Singer, J. & Forward, A. (2003). How Software Engineers Use Documentation: The State of Practice. *IEEE Software, 20(6),* s. 35-39.

Lutters, W. & Seaman, C. (2007). Revealing actual documentation usage in software maintenance through war stories. *Information and Software Technology, 49(6)*, 576-587.

Phoha, V. (1997). A standard for software documentation. *Computer, 30(10)*, s. 97-98.

Pohl, K. (1994). The Three Dimensions of Requirements Engineering: A Framework and its Applications. *Information Systems* 19(3), 243-258.

Pohl, K. (1996). Requirements engineering: An overview. Teoksessa *Encyclopedia of Computer Sience and Technology* 36. New York, NY: Marcel Dekker, Inc.

Poole, C. J., Murphy, T., Huisman, J. W. & Higgins, A. (2001). *Extreme maintenance*. Teoksessa Bob Werner (toim.) *Proceedings of IEEE International Conference on Software Maintenance 2001* (ICSM 01) (s. 301-209). Los Alamitos, CA: IEEE Computer Society.

Pressman, R. S. (2010). *Software Engineering : A Practitioner's Approach* (7. painos). New York, NY: McGraw-Hill.

Puerta, A., Micheletti, M. & Mak, A. (2005). The UI Pilot: A Model-Based Tool to Guide Early Interface Design. Teoksessa *Proceedings of the 2005 International Conference on Intelligent User Interfaces* (s. 215-222). New York: Association for Computing Machinery.

Rational Software (1998). *Rational Unified Process : Best Practices for Software Development Teams.* Haettu 13.11.2013 osoitteesta http://www.ibm.com/developerworks/rational/library/content/03July/1000/1251/1251_bestpractices_TP026B.pdf

Royce, W. (1970). Managing the Development of Large Software Systems. Teoksessa *Proceedings of 1970 IEEE Western Electronic Show and Conventum (WESCON)* (s. 1-9). TWR.

Runeson, P. & Höst, M. (2009). Guidelines for conducting and reporting case study research in software engineering. *Empirical Software Engineering, 14*, 131-164.

Saddington, Peter (2013). *Agile pocket guide : a quick start to making your business agile using Scrum and beyond*. Hoboken, N.J.: John Wiley & Sons.

Savolainen, J., Kuusela, J. & Vilavaara, A. (2010). Transition to Agile Development - Rediscovery of Important Requirements Engineering Practices. Teoksessa Bob Werner (toim.), *2010 18th IEEE International*

Requirements Engineering Conference (RE'1 19930) (s. 289-294). Sydney: IEEE Computer Society.

Schwaber, K. (2004). *Agile Project Management with Scrum*. Redmond, WA: Microsoft Press.

Schwaber, K. & Sutherland, J. (2012*). Software in 30 Days: How Agile Managers Beat the Odds, Delight Their Customers, and Leave Competitors in the Dust*. Hoboken, NJ: John Wiley & Sons.

Sharon, D. (1996). Meeting the Challenge of Software Maintenance. *IEEE Software* 13(1), 122-125.

Singer, J. (1998). Practices of Software Maintenance. Teoksessa Regina Spencer Sipple (toim.), *Proceedings of 1998 International Conference on Software Maintenance* (s. 139-145). Los Alamitos: IEEE Computer Society.

Smith, D., Thomas, B. & Tilley, S. (2001). Documentation for Software Engineers: What is Needed to Aid System Understanding? Teoksessa *Proceedings of the 19th annual international conference on Computer documentation SIGDOC'01* (s. 235-236). New York: Association for Computing Machinery.

Sommerville, I. (2005). Integrated Requirements Engineering: A Tutorial. *IEEE Software* 22(1), 16-23.

Sommerville, I. (2007). *Software engineering*. (8. painos). New York: Addison-Wesley.

Sommerville, I. (2010a). *Software Engineering*. (9. painos). Addison-Wesley.

Sommerville, I. (2010b). *Chapter 30: Documentation*. Haettu 23.7.2015 osoitteesta http://ifs.host.cs.st-andrews.ac.uk/Books/SE9/Web/ExtraChaps/Documentation.pdf.

Sousa, M. J. C. & Moreira, H. M. (1998). A Survey on the Software Maintenance Process. Teoksessa Taghi M. Khoshgoftaar & Keith Bennett (toim.) *Proceedings of the International Conference on Software Maintenance (ICSM 1998)* (s.265-274). Los Alamitos, CA: IEEE Computer Society

Souza de, S., Anquetil, N. & de Oliveira, K. (2005). The study of the documentation essential to software maintenance. Teoksessa *Proceedings of the 27th annual international conference on Computer documentation (SIGDOC '05)* (s. 68–75). New York, NY: ACM Press.

Souza de, S., Anquetil, N. & de Oliveria, K. (2006). Which documentation for software maintenance? *Journal of Brazilian Computer Science, 12(3)*, s. 31-44.

Swanson, E. (1976). The dimensions of maintenance. Teoksessa *Proceedings of the 2nd International Conference on Software engineering, ICSE '76* (pp. 492-497). Los Alamitos: IEEE Computer Society.

Thomas, B. & Tilley, S. (2001). Documentation for software engineers: what is needed to aid system understanding? Teoksessa Association for Computing Machinery (toim.), *Proceedings of the 19th annual international conference on Computer documentation, SIGDOC '01* (s. 235–236). New York, NY: ACM.

Tiwari, S. & Gupta, A. (2013). A Controlled Experiment to Assess the Effectiveness of Eight Use Case Templates. Teoksessa Muhammad Firdaus Harun & Ali Selamat (toim.) *Proceedings 2013 20th Asia-Pacific Software Engineering Conference (APSEC)* (s. 207-214). IEEE Computer Society.

Tryggeseth, E. (1997a). Report from an experiment: Impact of documentation on maintenance. *Empirical Software Engineering, 2(2),* 201-207.

Tryggeseth, E. (1997b). *Support for understanding in software maintenance*. Väitöskirja. Norwegian University of Science and Technology. Haettu 1.11.2014 osoitteesta http://citeseerx.ist.psu.edu/viewdoc/download;jsessionid=A69F8222A2793DE558E50F9E2A1EF275?doi=10.1.1.2.4853&rep=rep1&type=pdf

Uikey N., Suman U. & Ramani A. (2011). A documented approach in agile software development. *International Journal of Software Engineering* 2(2), 13-22.

Visaggio, G. (1997). Relationships between documentation and maintenance activities. Teoksessa Penny Storms (toim.), *Proceedings of 1997 Fifth International Workshop on Program Comprehension (IWPC '97)* (s. 4-16). Los Alamitos: IEEE Computer Society.

Vuori, M. (2010). *125 pointtia dokumentoinnista*. Haettu 17.7.2015 osoitteesta http://www.mattivuori.net/julkaisuluettelo/liitteet/satavartti_point-tia_dokumentoinnista.pdf.

Yang, H. & Ward, M. (2002). *Successful Evolution of Software Systems*. Noston: Artech House. Haettu 13.7.2015 osoitteesta http://site.ebrary.com/lib/jyvaskyla/detail.action?docID=10081931.

Yin, R. (2014*). Case study research : design and methods* (5. painos). Los Angeles, CA : Sage.

Yip, S. W. (1995). Software Maintenance in Hong Kong. Teoksessa *Proceedings of the International Conference on Software Maintenance (ICSM 1995)* (s. 88–95). Opio: IEEE Computer Society.

Yip, S. W. L. & Robson, D. J. (1991). Applying formal specification and functional testing to graphical user interfaces. Teoksessa *Proceedings, Advanced Computer Technology, Reliable Systems and Applications* (s. 557-561). IEEE Computer Society.

Young, R. (2003). *Requirements Engineering Handbook*. Norwood, MA: Artech House.