LAHDEN AMMATTIKORKEAKOULU
*Lahti University of Applied Sciences*

# SOVELLUKSEN MÄÄRITTELYTYÖ

Ohjelmistotuotantomallin toteutus käytännön projektissa

LAHDEN AMMATTIKORKEAKOULU
Tietojenkäsittelyn koulutusohjelma
Sovelluskehitys
Opinnäytetyö
Kevät 2009
Jouko Toivonoja

Lahden ammattikorkeakoulu
Tietojenkäsittelyn koulutusohjelma

TOIVONOJA, JOUKO:

Sovelluksen määrittelytyö
Ohjelmistotuotantomallin toteutus käytännön projektissa

Sovelluskehityksen opinnäytetyö, 45 sivua

Kevät 2009

TIIVISTELMÄ

---

Tämän työn aihe liittyy yritys X:n tiedonsiirtosovelluksen suunnittelu- ja määrittelytyöhön. Tarkoituksena on tutkia, kuinka käytännössä yhtä yrityksen teoreettista sovelluskehitysmallia toteutetaan kyseisen projektin osalta. Teoreettista mallia siis verrataan käytännön toteutukseen.

Työssä esitellään ensin teoreettista mallia yleisellä tasolla sekä käydään läpi yrityksen oma malli tarkemmalla tasolla. Opinnäytetyöhön liittyvää projektia, sen taustoja ja syitä kehittämiselle esitellään, jotta lukija saisi hieman käsitystä kyseisestä sovelluksesta, joka ei ole aivan jokapäiväinen työväline.

Tuloksia on pyritty saamaan esille sekä haastatteluin että havainnoimalla määrittelytyöhön osallistuessa. Näistä on koottu faktoja, sekä hieman omia mielipiteitä ja näkemyksiä. Haastateltavina ovat olleet projektin osalliset, jotka ovat olleet alusta pitäen sovellusta suunnittelemassa.

Opinnäytetyö vastaa tutkimuskysymyksen asettamiin tavoitteisiin ja tuo esille eroavaisuudet käytännön ja teorian välillä. Mallia noudatettiin melko hyvin kun sitä tarkastellaan pääkohdittain, mutta mitä syvemmälle malliin upotaan, sitä vähemmän sen yksityiskohtaisia ohjeita on noudatettu. Mallia on siis kyseisessä projektissa sovellettu melko rohkeasti. Haastateltavat pitivät tätä nimenomaisesti hyvänä asiana, jotta dokumentointia ei olisi tehty vain dokumentoinnin vuoksi.

Mallin noudattaminen vielä tarkemmin ei välttämättä olisi ollut lainkaan haitallista projektille, päinvastoin. Vaikeampaa on kuitenkin hahmottaa se piste, jonka jälkeen mallin noudattaminen ei enää tuota niin paljon hyötyä suhteessa työmäärään.

Tuloksista saadut päätelmät voidaan tiivistää muutamaan lauseeseen. Teoriamallit ovat varmasti hyödyllisiä sovelluskehitysprojektin onnistumisen kannalta, elleivät jopa välttämättömiä. Niiden hyödyllisyyden maksimointi kuitenkin vaatii mallin soveltamista projektin luonteen, koon ja tarpeiden mukaan. Ehkä vaikeinta onkin hahmottaa, mikä on oikea taso soveltamisen määrälle.

Avainsanat: ohjelmistotuotantomalli, sovelluskehitysmalli, suunnittelu, määrittely

Lahti University of Applied Sciences
Degree Programme in Computing

TOIVONOJA, JOUKO:

Specifying application requirements
Implementing application development model in practice

Bachelor's Thesis in Application Development, 45 pages

Spring 2009

ABSTRACT

---

This paper is related to company X's project to design and define a data transfer application. The purpose of the paper is to investigate how a theoretical application development model is implemented in practice in a project. The theoretical model is compared to the practical realization.

First a general theoretical model is presented and then the company's own model. The project, its background and the reasons to develop it are introduced so that the reader will understand what this application, which is not a daily tool, is all about.

The results were achieved by interviews and also by observing the project. Facts, opinions and point of views were gathered using these methods. The interviewed persons were those involved in the project, namely those who designed and defined the application from the beginning.

The thesis accomplished the goals set in the research question, the differences between theory and practice are highlighted. Generally, the theory model was followed quite well. But the deeper we go in the theory model the less its detailed instructions were followed. It could be said that the model was applied quite flexibly. This was considered a good thing since documentation was not made only for the purpose of documenting.

Compliance with the model in a little more detail might not have been bad for the project, maybe on the contrary. However, it is more difficult to grasp the point at which applying the model no longer gives as much benefit in the same proportion.

The conclusions can be summed up in a few sentences. For succeeding in an application development project, theory models are certainly very useful, if not necessary. However, maximizing their usefulness requires suiting the theory model to the size, needs and nature of the project. Perhaps the hardest aspect is to grasp what the correct level of applying a theory model is.

Key words: software production model, application development model, designing, specifying

SISÄLLYS

SANASTO

- Client, tiedonsiirtomoottori

  Yleisesti client tarkoittaa paikallisesti työasemalla tai palvelimella suoritettavaa sovellusta.

  Projektista B puhuttaessa ”Client” tarkoittaa pääasiassa palvelinalustalle asennettavaa pientä osaohjelmaa, joka suorittaa sille annettuja tiedonsiirtotehtäviä palvelimelta käsin.

  Vanhasta tuotteesta puhuttaessa tarkoitetaan koko sovellusta, joka on kokonaan client-tyyppinen.

- Evo-malli (vaihejakomalli)

  Mallin päätarkoitus on rakentaa niin sanottu ydin, jota jatkossa kehitetään edelleen. Evo-malli sisältää käytännössä useita pieniä vesiputousmalleja, eli sama prosessi käydään läpi useamman kerran.

- FTP (File Transfer Protocol)

  Salaamaton tiedostonsiirtomenetelmä kahden tietokoneen välillä.

- Inkrementaalinen prosessimalli (vaihejakomalli)

  Evo-mallin kaltainen kehitysmalli, jossa tuotetta kehitetään inkrementteinä, eli lisäyksinä.

- Installaatio = 1 sovelluksen instanssi = 1 client

  Yhdellä organisaatiolla voi olla useampia clienteja asennettuna eri palvelimille. Yksi installaatio vastaa siis yhtä toimivaa kokonaisuutta järjestelmässä. Yksi Client kuuluu yhteen installaatioon, eli instanssiin.

- Ohjelmiston elinkaari

  Aika, joka kuluu ohjelmiston kehittämisen aloituksesta sen käytöstä poistamiseen.

- Proof Of Concept – POC

  Proof of concept on teoreettisen idean testaamista, jolla pyritään selvittämään idean toimivuus ja tarkoitukseen soveltuvuus. Esimerkiksi

ennen uudentyyppisen arkkitehtuuri-idean toteuttamista suurella mittakaavalla on hyvä testata se kevyellä esiversiolla.

- Protoilu

  Periaatteessa mikä tahansa työskentelymalli, jossa tuotteen tiettyä piirrettä testataan ennen varsinaista toteuttamista.

- Relaatiotietokanta

  Tietokanta on kokoelma tietoja, eli tietovarasto. Relaatiotietokanta puolestaan tarkoittaa toisiinsa liittyvien tietojen kokoelmaa. Visuaalisesti esitettynä relaatiotietokanta on kokoelma tietotauluja, joissa on tauluja toisiinsa yhdistäviä kenttiä. Tunnetuimpia relaatiotietokantaohjelmistoja ovat muun muassa Microsoft Access, Microsoft SQL-server, MySQL, Oracle.

- SSH/SSH2-suojattu SFTP(SSH File Transfer Protocol)

  SSH-protokolla tiedonsuojausmenetelmä, joka tunneloi käytettävän yhteyden. SFTP on siis tunneloitu, eli suojattu FTP-yhteys.

- TCP/IP (Transmission Control Protocol / Internet Protocol)

  TCP/IP on verkkoteknologia, joka koostuu monista eri protokollista. Kaikki protokollakokoelman protokollista toimivat IP-protokollan päällä. Näistä tärkeimmät ja tunnetuimmat ovat TCP ja UDP (User Datagram Protocol).

- Tietue

  Yksi tietokokonaisuus. Esimerkiksi verkkopalkka-aineistossa voi olla sadan henkilön palkkakuitit. Yhden henkilön palkkakuitti on tällöin tietue, eli yksi tietokokonaisuus.

- Vaihejakomalli

  Tietojärjestelmän kehittämismalleja, jotka ovat selkeästi jaoteltu erilaisiin vaiheisiin toteutuksen edetessä. Yleensä niistä kaikista löytyy ainakin määrittely-, suunnittelu- ja toteutusvaiheet.

- Vesiputousmalli (vaihejakomalli)

  Vesiputousmallissa on kehityksen eri vaiheet jaoteltu hyvin selkeästi eri vaiheisiin ja toteutusjärjestykseen. Mallista on olemassa useita eri versioita, toisissa sallitaan edestakaisin liikkuminen ja toisissa sitä ei ehkä suositella.

- Web-sovellus

  Yleisesti web-sovellus tarkoittaa sovellusta, jota käytetään Internetin tai sisäverkon yli. Esimerkiksi Internetin karttasovellukset ovat web-sovelluksia.

  Projektista B puhuttaessa web-sovellus on Internet-selaimella käytettävä sovellus, jolla ohjataan asiakkaan omalla palvelimella sijaitsevaa client-osasovellusta.

- Windows, Linux, AIX ja VMS

  Tietokoneen käyttöjärjestelmiä, joista on erilaisia versioita. Versioilla on yleensä eri käyttötarkoitukset, toiset ovat suunniteltu palvelinkäyttöön ja toiset työasemakäyttöön.

# 1 JOHDANTO

## 1.1 Taustaa

Tämä opinnäytetyö käsittelee yritys X:n tuotekehitysprojektin suunnittelu- ja määrittelytyön toteuttamista. Tuotekehitysprojekti liittyy käytössä olevan tiedonsiirtojärjestelmän A korvaavan järjestelmän toteuttamiseen nykyaikaisella arkkitehtuurilla.

Tarkoituksena on verrata yhtä yrityksen sovelluskehitysmallia käytännön toteutukseen suunnittelu- ja määrittelytyön osalta. Tällä pyritään selvittämään kuinka mallia toteutetaan kyseisen projektin kohdalla.

Määrittelytyöksi lasketaan kaikki materiaali ja tieto, joihin tuotekehitysprojekti nojaa. Yleensä vaatimusmäärittelyt dokumentoidaan tekstimuotoon yksiselitteisiksi lauseiksi ominaisuusluetteloon. Käsiteltävässä tuotekehitysprojektissa käytetään paljon myös visuaalisia tapoja ilmaista toimintamalleja ja ominaisuuksia. Pääasiallisesti ne ovat itse tuotettuja kuvitteellisia kuvakaappauksia sovelluksen toiminnasta, mutta myös graafisia prosessikaavioita sekä tiedonkulkukaavioita. Projekti ja sen suunnittelu- ja määrittelytyö perustuu vesiputousmalliin, josta yritys X:llä on hyvin yksityiskohtaiset mallit ja pohjat olemassa.

## 1.2 Tutkimuskysymys

**Kuinka it-yrityksen suunnittelu- ja määrittelymalleja toteutetaan käytännön suunnittelu- ja toteutusprojektissa?**

Opinnäytetyössä tutkitaan työskentelytapoja, joita käytetään it-yrityksen sovelluskehityksen määrittelyssä sekä suunnittelu- ja määrittelymallin toteuttamista käytännössä. Tutkimuksen tarkoitus on tuoda esille, kuinka hyvin valmiita määrittely- ja työskentelymalleja noudatetaan pienehkössä ohjelmistoprojektissa. Samalla tuodaan esille, kuinka tärkeänä projektiin osallistuvat henkilöt pitävät mallien noudattamista projektin onnistumisen kannalta.

Tutkimus on rajattu vaatimusmäärittelyn ja suunnittelun tarkasteluun, jotta käsiteltävän aiheen laajuus pysyy kurissa. Teoreettisen mallin menetelmät on kuitenkin esitelty kokonaan, jotta kokonaiskuva sovelluksen elinkaaresta ei jäisi peittoon.

## 1.3 Tutkimusmenetelmä

Tässä tutkimuksessa käytetään haastatteluita ja osallistuvaa tutkimusmenetelmää. Oma osuuteni projektissa on ollut samankaltainen kuin haastateltavilla. Haastattelut toteutetaan teemahaastatteluina, joissa haastattelun kulku etenee melko vapaasti haastattelijan antamien aiheiden, eli teemojen mukaan.

Projektin määrittelyvaiheeseen osallistuvia henkilöitä haastatellaan ja pyritään saamaan heiltä yksityiskohtaista tietoa muun muassa mallin käytännöllisyydestä, sopivuudesta ja sen noudattamisesta kyseisessä projektissa.

Osallistuminen itse projektiin helpottaa määrittämään subjektiivisesti sen, kuinka hyvin mallia on todellisuudessa toteutettu sen eri osa-alueilla. Haastattelut ovat väistämättä objektiivisia näkemyksiä, joista saattaa heijastaa läpi esimerkiksi van-

hat tavat ja tottumukset. Oma osuuteni projektissa on ollut samankaltainen kuin haastateltavilla, eli suunnittelu- ja määrittelytyö yleisellä ja teknisellä tasolla.

## 1.4 Haastateltavat

Haastateltavat ovat projektin ydintiimiä. Haastatteluun otettiin mukaan 3 henkilöä, jotka työskentelevät X:llä samassa tiimissä, vaikkakin hieman eri tehtävissä. Haastateltavien työkokemukset eroavat toisistaan ja toimenkuvat työhistorian ajalta ovat monipuolisia. Kaksi heistä on ollut useamman kerran mukana sovelluskehitysprojektin suunnittelussa ja yksi on ensimmäistä kertaa. Yksi haastateltavista on ollut huomattavan pitkän ajan it-alan työtehtävissä ja muut sen sijaan suhteellisen vähän aikaa, puolestatoista vuodesta neljään vuoteen.

## 1.5 Haastatteluteemat

Haastattelussa edetään ennalta määriteltyjen teemojen ympärillä, kuitenkin melko vapaasti keskustellen aiheesta. Haastattelun aihe ja teemat ilmoitetaan haastateltaville hyvissä ajoin ennen haastattelua, jotta haastateltava voi hieman valmistautua aiheeseen. Teemat on asetettu seuraavalla tavalla, kaikki liittyen X:n vaihejakomallin vaatimusmäärittelyyn ja suunnitteluun.

- Työntekijän tausta
  - Tehtävät
  - Aikaisempaa sovelluskehityskokemusta?
- Taustatiedot yritys X:n malleista
  - Ovatko mallit tuttuja?
  - Aikaisempia kokemuksia?
- Mallin noudattaminen projektissa
- Mallin soveltuvuus/käytännöllisyys

## 1.6 Suunnittelumallista mitattavat kohteet

Yritys X:n vaihejakomalli on hyvin yksityiskohtainen ja syvälle ulottuva toimintamalli, minkä vuoksi tässä työssä ei mitata koko mallia ja sen noudattamista. Laajuutensa vuoksi osa kyseisestä mallista on jätetty tutkimuksen ulkopuolelle. Työssä keskitytään vaatimusmäärittelyn ja -analyysin pääkohtiin. Tutkimuksesta ulkopuolelle jäävät vaiheet, sovelluksen tekninen suunnittelu ja toteutus, ovat tässä projektissa selkeästi alihankkijan vastuulle jääviä asioita.

## 1.7 Analyysimenetelmä

Haastatteluissa saatuja tuloksia verrataan teoreettisen mallin ohjeisiin ja tuodaan esille eroavaisuudet teorian ja käytännön välillä kyseisen projektin osalta. Haastatteluiden tuloksista pyritään myös tuomaan esille mahdolliset syyt eroavaisuuksille, mikä niitä aiheuttaa ja miksi.

# 2 YRITYS X:N TYÖTAVAT JA MENETELMÄT

Kuten suurilla organisaatioilla yleensä, myös yrityksellä X on malleja kehittämishankkeiden läpiviennille. Tällaisten päätarkoitus on usein tukea liiketoiminnan kehittämishankkeita. Sovelluskehitysprojekteihin Yritys X:llä (2006) on käytössään SoftwareMethods-systeemityömenetelmä, jonka tarkoituksena on varmistaa tietojärjestelmien rakentamisen laatu. Laadun varmistaminen tarkoittaa suurempaa asiakastyytyväisyyttä, joka antaa paremmat edellytykset liiketoiminnan suorittamiseen. SoftwareMethods on vain osa yrityksen X liiketoimintakehitysmallin menetelmäkirjastosta. Muita malleja tästä kirjastosta löytyy esimerkiksi liiketoimintaprosessien kehittämiseen, projektityöhön ja testaukseen.

Systeemityömenetelmän tarkoitus on helpottaa tekemistä ja yhdenmukaistaa toimintatapoja. Se toimii projektin osituksen apuna ja antaa selkeät tehtävät ja niiden lopputulokset, työmenetelmät, työtavat, apuvälineet ja näiden ohjeistuksen. Näin ollen kaikki projektin osalliset tietävät mitä pitää tehdä, miten ja missä järjestyksessä. Ylimmälle johdolle SoftwareMethods on työkalu systeemityöhankkeiden läpiviennin ohjaukseen ja seurantaan. Asiakkaille SoftwareMethods on konkreettinen ja selkeä tapa olla mukana järjestelmän kehityksessä. Projektipäällikölle SoftwareMethods antaa pohjan suunnitella rakentamisen vaiheet ja tehtävät projektisuunnitelmaan, sekä projektin mukana oleville selkeät tehtävät roolista riippuen. Määrittelijät, suunnittelijat, toteuttajat ja testaajat saavat yksityiskohtaista tietoa siitä, mitä tehdään, miten tehdään, miten dokumentoidaan, sekä valmiita ohjeistuksia ja pohjia esimerkiksi dokumentointiin. (Yritys X. 2006.)

Käytössä oleva SoftwareMethods-systeemityömenetelmä ulottuu muun muassa seuraaviin prosessimalleihin: vaihejakomalli, inkrementaalinen prosessimalli ja protoilu. Näistä malleista projektissa B käytettiin hyväksi eräänlaista vaihejakomallia, joka tunnetaan myös nimellä vesiputousmalli.

## 2.1 Vaihejakomallit

Ohjelmistokehityksessä käytetään lähes poikkeuksetta erilaisia valmiita tuotantomalleja. Kehitys on yleensä mallista riippumatta jossain määrin vaiheittaista. Vaihejakomallit kuvaavat ohjelmiston elinkaarta vaiheittain, yleensä kehitystyöstä sen ylläpitovaiheeseen. Kehitysmenetelmiä on olemassa erilaisia, pääasiassa tässä opinnäytetyössä keskitytään vesiputousmalliin, eikä syvennytä muihin malleihin sen enempää. Seuraavaksi esitellään kuitenkin lyhyesti muutama vesiputousmallista poikkeava kehitysmalli, jotta saadaan pieni käsitys siitä, minkälaisia muita malleja ohjelmistokehityksessä voi käyttää. (Haikala & Märijärvi 2006, 45–47.)

Vaihejakomalleja on esimerkiksi *prototyyppimalli* (protoilumalli), jossa tehdään yksinkertainen ja karkea ohjelmistomalli, jolla on tarkoitus esitellä minkä tyyppinen lopullinen tuote olisi. Tämä karkea malli tuo esille jo alkuvaiheessa kysymyksiä ja ongelmia ohjelmistoon ja sen kehitykseen liittyen. Malli soveltuu tapauksiin, jossa tilaaja haluaa nopeasti saada jotain konkreettista nähtävää tai tilaajan on vaikea määritellä vaatimuksia järjestelmälle. (Haikala & Märijärvi 2006, 42–45; Pohjonen 2007, 41.)

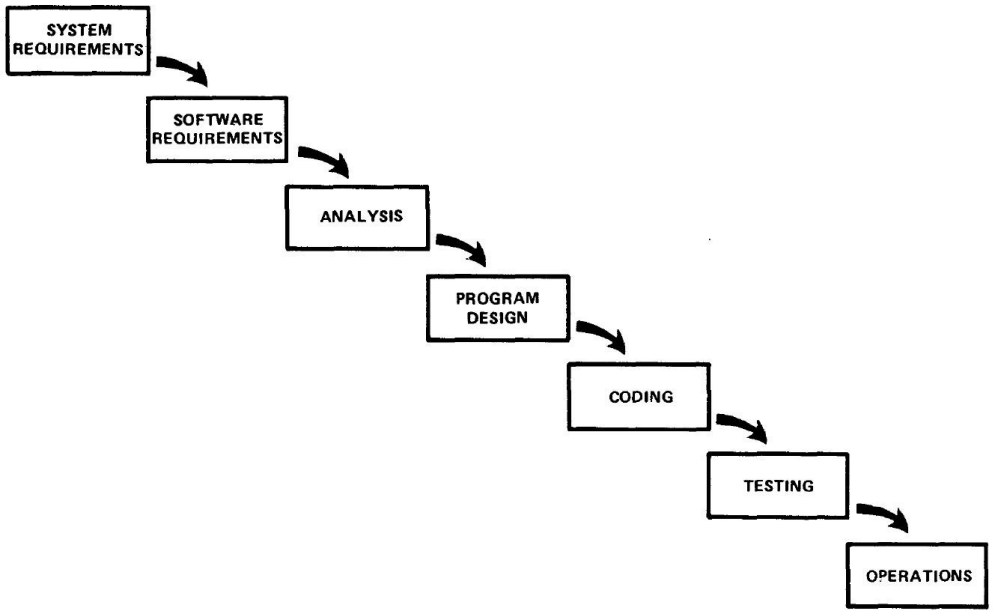
Toinen malli on *evo-malli* (evolutionary delivery), joka koostuu useista lyhyistä ohjelmistokehityksen jaksoista, joiden tuloksena syntyy ohjelmistosta aina uusi julkaisukelpoinen versio. Käytännössä siis tuotteen ominaisuuksia lisätään jatkuvasti, mutta annetaan jo olemassa olevat ominaisuudet käyttöön uudella julkaisulla. Monesti jokaiseen versioon tehdään omat määritykset ja suunnitelmat, sekä testaukset. Evo-mallin tapaisia kehitysmalleja kutsutaan joskus myös *inkrementaalisiksi malleiksi*. (Haikala & Märijärvi 2006, 45–47.)

## 2.2 Vesiputousmalli

Vesiputousmallista on olemassa useita eri variaatioita, jotka eroavat toisistaan melko paljonkin. Yhteinen piirre on kuitenkin se, että niistä löytyy ainakin määrittely-, suunnittelu- ja toteutusvaiheet. Vesiputousmalliin kuuluu esitutkimus, määrittely-, suunnittelu-, ohjelmointi-, testaus- ja ylläpitovaihe. ”Esitutkimuksen tehtävänä on asettaa yleiset järjestelmätason vaatimukset”. (Haikala & Märijärvi 2006, 36–37.)

**Esitutkimuksella** pyritään määrittelemään millainen järjestelmä täyttää asiakastarpeet ja miksi se kannattaa tehdä tai miksi sitä ei kannata tehdä. **Määrittelyvaiheessa** esitutkimuksen tuloksia analysoidaan ja luodaan niiden perusteella tekniset vaatimukset/ominaisuudet ohjelmistolle. **Suunnitteluvaihe** sisältää teknisten vaatimusten toteuttamisen suunnittelun. Suunnitteluvaiheessa on kaksi osaa, arkkitehtuuri- ja moduulisuunnitteluvaihe. Arkkitehtuurisuunnitteluvaiheessa suunnitellaan suuremmat toisistaan erilliset osat, moduulisuunnitteluvaiheessa suunnitellaan näiden erillisten osien sisäinen rakenne. **Ohjelmointivaihe** käsittää ”ohjelman kirjoitusvaihetta ensimmäiseen virheettömään käännökseen asti”. **Testauksen** tarkoituksena on löytää ohjelmistovirheet ja niiden aiheuttajat. **Ylläpito** on loppukäyttäjän palautteista tullutta järjestelmän kehittämistä, päivittämistä ja ongelmien ratkomista. (Haikala & Märijärvi 2006, 37–41; Harsu 2003, 16–18.)

Vesiputousmalli on yleisen käsityksen mukaan esitelty ensimmäisen kerran Dr. Winston W. Roycen toimesta (Royce 1970).



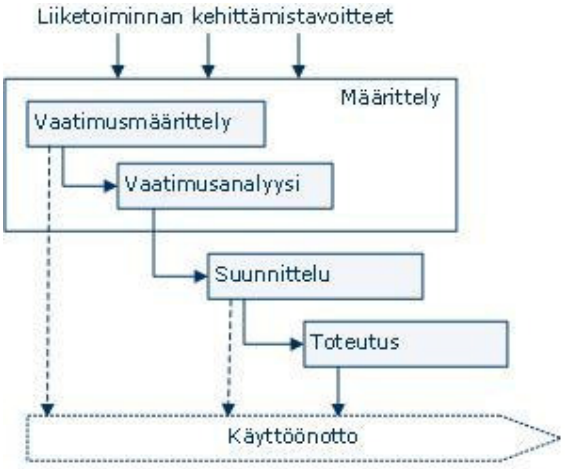

**Kuvio 1.**
**Vaiheet suuren ohjelmiston kehitykseen ja asiakkaalle toimittamiseen (suomennos kirjoittajan) (Royce, 1970).**

Vesiputousmallin idea kiteytyykin siihen, että ensin analysoidaan ratkaistava ongelma, suunnitellaan siihen ratkaisu, jonka jälkeen toteutetaan ratkaisu ja testataan se.

Ongelmana vesiputousmallissa koetaan sen käytännön toimivuus. Ohjelmistokehitys ei koskaan etene, eikä voi edetä, täysin vesiputousmallin mukaisesti, koska osa ohjelmistolle asetettavasti vaatimuksista selviää vasta kehityksen aikana. Toisena kriittisenä ongelmana nähdään tarkistuspisteiden ja dokumenttien kiinnittäminen tiettyyn vaiheeseen. Haittapuolena tästä aiheutuu suuriakin kustannuksia, mikäli projektissa joudutaan peruuttamaan. Se edellyttää edeltävien vaiheiden uusimista tai vähintään päivittämistä. (Haikala & Märijärvi 2006, 41; Pohjonen 2007, 40.)

## 2.3 Yritys X:n vaihejakomalli

Tässä kappaleessa käydään läpi yrityksen X käyttämää vaihejakomallia. Vaihejakomalli tarkoittaa tietojärjestelmän rakentamisen jakamista selkeästi eri vaiheisiin: määrittely, suunnittelu ja toteutus. Jokainen vaihe voi olla jopa oma projektinsa tai kaikki vaiheet voivat olla yhtä projektia. Myös ajallisesti vaiheet toteutetaan kokonaisuuksina, eli vasta kun määrittely on kokonaan tehty, aloitetaan suunnittelu ja niin edelleen. Neljä päävaihetta on jaoteltu pienempiin osiin ja kaikille näille on määritelty oma vastuuhenkilö.




**Kuvio 2.**
**Yritys X:n vaihejakomalli**

### 2.3.1 Roolit

Erilaisia rooleja vaihejakomalliin on asetettu kahdeksan. Yhdellä henkilöllä voi olla useampi rooli, jos on kyse pienemmästä projektista.

- Arkkitehti

  Arkkitehti vastaa teknisen sekä sovellusarkkitehtuurin määrittelystä ja huolehtii sovelluksen rakenteesta.

- Katselmoija

  Katselmoija on projektin laadunvarmistuksesta ja katselmusten läpiviennistä vastaava henkilö.

- Määrittelijä

  Määrittelijällä on merkittävä rooli vaatimusmäärittelyiden ja käyttöympäristön kuvaamisessa. Osallistuu aktiivisesti moniin projektin eri vaiheisiin.

- Omistaja

  Omistaja on työn tilaaja ja maksaja. Asettaa viime kädessä projektille tavoitteet. Omistaja voi olla ulkoinen asiakas tai sisäinen tilaaja.

- Projektipäällikkö

  Projektipäällikkö on toimittajan puolelta projektin läpiviennistä vastaava henkilö. Erikseen määriteltynä voi tarkoittaa myös tilaajan vastuuhenkilöä.

- Suunnittelija

  Suunnittelijan päätehtävä on suunnitella miten järjestelmän eri osat sovitetaan yhteen. Suunnittelija on vahvasti mukana kaikissa suunnitteluun liittyvissä vaiheissa.

- Testaaja

  Testaaja vastaa testitapauksista, testien tulosten analysoinnista ja raportoinnista sekä testiaineiston ylläpidosta.

- Toteuttaja

  Toteuttaja rakentaa järjestelmän määrittelyiden ja suunnitelmien perusteella. Toteuttaja voi olla myös kolmas osapuoli. (Yritys X 2006.)

### 2.3.2 Vaatimusmäärittely

Vaatimusmäärittelyssä haetaan vastausta kysymyksiin mitä ja kenelle. Eli mitä tietojärjestelmältä vaaditaan ja minkälainen sen tulisi olla, jotta se palvelisi asiakasorganisaation tavoitteita ja käyttäjiä. Vaatimusmäärittelyssä on tärkeää saada esille järjestelmälle asetettavat vaatimukset ja dokumentoida ne täsmällisiksi piirteiksi ja palveluiksi. Vaatimusmäärittelyn perusteella voidaan valita tietojärjestelmässä käytettävät tekniikat soveltuvuuden mukaisesti. Tämän vuoksi vaatimusmäärittely tehdään toteutusteknologiasta riippumatta.

- Toimintaympäristön kuvaaminen

  Toimintaympäristön kuvauksesta kaikki projektissa mukana olevat saavat käsityksen tietojärjestelmän toiminnasta ja tarkoituksesta. Kuvaus on kirjoitettava niin, että myös loppukäyttäjä ymmärtää sen. Kuvauksen kirjoittamisesta vastaa määrittelijä.

  Toimintaympäristön kuvaamiseen kuuluu viisi eri osatehtävää: sidosryhmien kuvaaminen, toiminta-/työprosessien kuvaaminen, käyttäjien kuvaaminen, tietojärjestelmän tarkoituksen kuvaaminen ja käytettävyyden arvioinnin suunnittelu.

- Sanaston kokoaminen

  Sanaston kokoaminen tehdään, jotta kaikki projektin osalliset voivat tarkistaa erilaisia projektissa esille tulleita termejä, jotka kaipaavat määrittelyä tai tarkennusta. Terminologiassa voi tulla vastaan samasta asiasta useita termejä. Toisaalta myös samalla termillä voi olla eri ihmisten mielessä täysin eri merkitys. Sanaston kokoaminen voidaan aloittaa heti projektin alkuvaiheesta lähtien. Sanaston kokoaminen koostuu kahdesta osatehtävästä: sanaston kerääminen, josta vastaa määrittelijä ja sanaston ryhmittely, jonka hoitaa omistaja.

- Vaatimusten kuvaaminen

  Vaatimusluettelo tehdään, jotta voidaan rakentaa asiakkaan tarpeita vastaava tietojärjestelmä. Sitä varten on tiedettävä vaatimukset ja tarpeet tietojärjestelmää kohtaan. Vaatimukset kerätään, luokitellaan ja priorisoidaan. Vaatimus tulee olla dokumentoituna yksiselitteisesti ja täsmällisesti. Vaatimusten kuvaaminen koostuu kahdesta osatehtävästä: vaatimusten kerääminen ja vaatimusten luokittelu. Omistajan vastuulla on kerätä ja listata vaatimukset, jotka määrittelijä luokittelee.

- Toiminnallisuuden mallintaminen

  Toiminnallisuuden mallintamisessa tärkein osa on käyttötapauksien kuvaaminen. Niiden avulla kuvataan toiminnalliset vaatimukset tietojärjestelmälle, sekä kerrotaan mitä tietojärjestelmä tekee. Mallintamisen tekee määrittelijä ja mallinnus tehdään käyttäjien näkökulmasta, mikä edellyttää myös, että se on tehty käyttäjien ymmärtämällä kielellä.

Toiminnallisuuden mallintamiseen kuuluu kolme osatehtävää: käyttäjäroolien kuvaaminen, käyttötapausten kerääminen, käyttötapausten kuvaaminen.

- Tietosisällön mallintaminen

Tietosisällön mallintamisen tulokset toimivat alustavana mallinna tietokannalle. Vaatimusmäärittelyssä määrittelijä kuvaa ne liiketoiminta-alueen käsitteet tai luokat, joita tietojärjestelmässä käsitellään. Vaatimismäärittelyssä tietosisällön mallintaminen pysyttelee karkeammalla tasolla, vaatimusanalyysissä niitä tarkennetaan ja käsitellään yksityiskohtaisemmin. Tietosisällön mallintaminen koostuu kahdesta vaiheesta, tietokokonaisuuksien kerääminen ja niiden kuvaaminen.

- Arkkitehtuurin määrittely

Arkkitehtuurin määrittelyn tavoite on näkemys sovelluksen arkkitehtuurista sekä teknisestä arkkitehtuurista. Tukirangan arkkitehtuurille antaa organisaation oma arkkitehtuuristrategia, joka määrittelee esimerkiksi käytettävät sovellusalustat, tietoturvan ja koodaukseen käytettävät kielet. Arkkitehtuurin määrittely on tässä vaiheessa suuntaa antava ja tarkentuu projektin aikana. Arkkitehtuurin suunnittelussa tulee ottaa huomioon järjestelmän ylläpidettävyys ja päivitettävyys. Sovellusarkkitehtuuri ja tekninen puoli ovat arkkitehdin vastuualueita.

Uusista ratkaisuista, joista ei ole vielä varmoja tuloksia toimivuudesta tai integroituvuudesta, toteutetaan Proof Of Consept -malli. Proof Of Concept pitäisi toteuttaa melko varhaisessa vaiheessa, jotta arkkitehtuurimalli saadaan testattua hyvissä ajoin.

Vaadittavat esitiedot arkkitehtuurimäärittelylle:

Organisaation arkkitehtuuristrategia, joka antaa linjat yksittäisten sovellusten arkkitehtuurille.

Muut järjestelmät, jotka vaikuttavat kyseisen järjestelmän arkkitehtuuriin välittömästi tai välillisesti, on otettava huomioon.

Tulevaisuudessa odotettavat muutos- tai kehityspaineet kuten käyttötapa, käyttäjät ja volyymit on otettava huomioon. Myös ympäristöstä, johon järjestelmä rakennetaan, pitää olla tarvittavat tiedot.

- Vaatimusmäärittelyn katselmointi

  Vaatimusmäärittelyiden katselmointi on projektipäällikön sekä katselmoijan toimesta koottu luettelo vaatimusmäärittelyn tuloksena syntyneistä dokumenteista. Katselmoinnissa varmistetaan, että dokumentit ovat suunnitelmien mukaiset, riittävän täsmällisiä ja kattavia. Projektipäällikkö vastaa tulosten kokoamisesta yhteen ja katselmoija varmistaa niiden laadun. (Yritys X 2006.)

### 2.3.3 Vaatimusanalyysi

Vaatimusanalyysissä nimensä mukaisesti analysoidaan määritettyjä vaatimuksia ja syvennetään sekä tarkennetaan niitä niin pitkälle, että suunnittelu voidaan aloittaa. Vaatimusanalyysiin kuuluu neljä pääkohtaa.

- Toiminnallisuuden analysointi

  Toiminnallisuuden analysointi kattaa tietojärjestelmälle asetettujen toiminnallisten vaatimusten tutkimista ja täsmentämistä. Täsmennetään ja todetaan, että vaatimukset ovat kuvattu oikein käyttäjän näkökulmasta. Analysointiin osallistuu useampia rooleja, mutta kokonaisuudessaan vastuu toiminnallisuuden analysoinnista on omistajalla.

  Toiminnallisuuden analysointi toteutetaan kolmessa osassa, jotka ovat käyttötapausten tarkennus ja palvelujen kuvaus, joista vastaa määrittelijä sekä uudelleenkäyttötarkoitus, joka on arkkitehdin vastuulla.

- Tietosisällön analysointi

  Tietosisällön analysoinnilla pyritään selvittämään, mitä tietoja tarvitaan, jotta voidaan täyttää tietojärjestelmälle asetetut tavoitteet. Analysointiin kuuluu luokkakaavioiden tarkennus ja koodistojen kuvaaminen, joista vastaa määrittelijä. Koodistolla tarkoitetaan tietoja, jotka tietojärjestelmässä esitetään eri tavalla kuin käyttäjälle. Numeerinen koodi voidaan esimerkiksi käyttäjälle esittää sanallisena. Luokkakaavioissa keskitytään liiketoimintaan liittyviin luokkiin, ei niinkään tekniseen ympäristöön. Koodistosta kerätään selkeä luettelo määrittelyineen.

- Käyttöliittymän mallintaminen

  Käyttöliittymän mallintamisen tarkoituksena on tuoda hahmotelmia tietojärjestelmän eri osien käyttöliittymäosista täyttäen annetut vaatimukset sekä sovelluksen toiminnallisuudesta että käytettävyydestä.

  Määrittelijän tehtävänä on käytettävyyden arviointisuunnitelman tarkennus ja sovelluksen käyttöliittymästandardin kuvaaminen, jonka tarkoituksena on olla tukena yhdenmukaisen käyttöliittymän suunnittelulle läpi linjan. Muita määrittelijän tehtäviä ovat käyttöliittymän rakenteen hahmottelu ja käyttöliittymän sisällön ja ulkoasun hahmottelu.

  Ensimmäinen käytettävyyden arviointi on sisällön asiantuntijan vastuualuetta. Siihen kuitenkin osallistuu yleensä 2-5 henkilöä, joista 1-2 asiantuntijoita, jotka ovat olleet tiiviisti mukana tietojärjestelmän rakentamisessa. Tavoitteena varmistaa, että tietojärjestelmä tukee loppukäyttäjän tarpeita mahdollisimman tehokkaasti. Käytettävyyden arvioinnissa on huomattava määrä seikkoja, joihin tulee kiinnittää huomiota, eikä niitä sen vuoksi tässä työssä sen tarkemmin tulla käsittelemään. Pääkohtia ensimmäisen käytettävyyden arvioinnissa ovat sisältö ja ulkoasu, ikkunoiden tai sivujen yhtenäisyys sekä selkeä käyttöliittymän rakenne.

- Vaatimusanalyysin katselmointi

  Katselmoinnilla varmistetaan, että vaatimusanalyysissä tuotetut dokumentit ovat määrityksen mukaiset ja vaaditulla tasolla sekä vastaavat tilaajan tai käyttäjän tarpeita. Katselmointi tulee ensin koota vaatimusanalyysin dokumenteista yhdeksi kokonaisuudeksi, minkä jälkeen se katselmoidaan, eli tarkistetaan vaatimusanalyysin kattavuus ja oikeellisuus. Projektipäällikkö kokoaa aineiston ja katselmoija tarkistaa sen. (Yritys X 2006.)

### 2.3.4 Suunnittelu

Määrittelyvaiheessa tuotetuista määrittelyistä työstetään asiat sille tasolle, että tiedetään miten tietojärjestelmä toteutetaan.

- Arkkitehtuurisuunnittelu

  Arkkitehtuurisuunnittelu jatkuu määrittelyiden pohjalta. Määrittelyn tulokset saatetaan tilaan, jossa ohjelmiston suunnittelu ja toteutus voidaan aloittaa luotettavasti. Arkkitehtuurisuunnittelusta vastaa arkkitehti. Suunnittelun tavoitteena ovat toimivat tekniset ratkaisut ja sovelluksen rakenne ja näiden tulee soveltua määritellyn järjestelmän toteutusratkaisuksi. Arkkitehtuurisuunnittelu koostuu teknisen ympäristön suunnittelusta ja sovellusarkkitehtuurin suunnittelusta.

- Käyttöliittymän suunnittelu

  Käyttöliittymän suunnittelu -vaiheeseen kuuluu käyttöliittymästandardin tarkennus ja rakenteen suunnittelu, joista vastaa omistaja, sekä sisällön ja ulkoasun suunnittelu, joka on suunnittelijan vastuulla. Mallintamisvaiheessa syntyneet määrittelyt työstetään sille tasolle, että niiden pohjalta voidaan toteuttaa käyttöliittymä tietojärjestelmälle sekä sen osille. Lähinnä siis tarkennetaan ja mallinnettua luonnosta ja tehdään yksityiskohtaisempi suunnitelma. Vaiheeseen kuuluu myös toinen käytettävyyden arviointi, joka toteutetaan samalla periaatteella kuin mallintamisvaiheessa, tosin yksityiskohtaisemmin. Päävastuu arvioinnista on omistajalla.

- Tietokannan suunnittelu

  Tietokannan suunnittelun tulee antaa kuvaus siitä, miten tietojärjestelmässä käsiteltävä tieto tallennetaan. Tietokanta sisältää tietojärjestelmän käsittelemät pysyvät tiedot.

  Useimmiten tietokanta on relaatiotietokanta, mutta se voi olla esimerkiksi tiedostopohjainen. Tietokannan suunnittelu jaetaan loogiseen ja fyysiseen. Looginen suunnittelu on tietokantariippumatonta, eli muunnetaan aiemmin määritelty luokkamalli relaatiomalliksi, joka on riippumaton käytettävästä tietokannasta. Fyysisen suunnittelun tavoite on sovittaa loogisen suunnitelman tietokantamalli kohdetietokannalle sopivaksi sekä varmistaa tietokannan tehokkuus ja tieto-

turva. Sekä loogisen että fyysisen tietokannan suunnittelusta vastaa tietokannat tunteva suunnittelija.

Mikäli tietojärjestelmästä on olemassa vanha versio ja tietoja halutaan siirtää vanhasta uuteen, tulee myös suunnitella konversiot kahden järjestelmän välille.

- Ohjelmistosuunnittelu

Ohjelmistosuunnittelun tarkoituksena on ratkaista miten tietojärjestelmän sisäiset luokat, kirjastot ja komponentit toimivat yhteistyössä eri käyttötapauksissa. Suunnittelija vastaa ohjelmistosuunnittelusta ja sen kahdesta osatehtävästä, ohjelmistorakenteen suunnittelusta ja toiminnallisuuden suunnittelusta.

Liittymät tai integrointi muihin järjestelmiin mietitään ohjelmistosuunnittelun yhteydessä, mikäli niihin tarvetta.

- Suunnittelun katselmointi

Katselmoinnilla varmistetaan suunnittelun tuloksien oikeellisuus ja kattavuus, jotta ne vastaisivat määrityksissä annettuja vaatimuksia ja asiakkaan tarpeita. Projektipäällikkö kokoaa dokumentit luetteloksi, josta katselmoija voi suunnittelun tulokset tarkistaa. (Yritys X 2006.)

### 2.3.5 Toteutus

- Arkkitehtuurin ja kehitysympäristön toteutus

Kehitysympäristön perustaminen ja ohjeistus ovat edellytys toteuttamisen aloittamiselle. Arkkitehtuurisuunnittelussa määritellyt laitteistot ja ohjelmistot hankitaan kehitys- ja testausympäristöihin sekä huolehditaan niiden ohjeistus. Kehitysympäristön käytön ja pystytyksen ohjeistus on tärkeä osa toteuttamisen aloitusta. Vastuullinen rooli on arkkitehdillä.

Uudelleenkäytettävien komponenttien, luokkien tai ohjelmien käyttö ja hankita perustuu ohjelmistosuunnittelussa tuotettuun kuvaukseen. Sen perusteella voidaan harkita käytetäänkö valmiita osaohjelmia tai luokkia vai koodataanko sellainen kyseiseen tietojärjestelmään. Suunnittelun perusteella tulisi tietää mitä uudelleenkäytettävältä komponentilta vaaditaan, jotta sitä voitaisiin hyödyntää.

Mikäli arkkitehtuurissa käytetään uutta teknologiaa, viimeistään tässä vaiheessa on suoritettava Proof Of Consept.

- Käyttöohjeiden toteutus

Käyttöohjeiden tavoite on tuottaa käyttöönottovaiheeseen sekä käyttöä varten ohjeistus ja käsikirja. Käyttöohjeiden tai käsikirjan käytettävyyteen on kiinnitettävä huomiota, koska käytännössä niihin turvaudutaan vasta kun ongelma on jo muodostunut. Ohjeiden selkeä ja hyvä toteutus takaa onnistuneen käyttöönoton. Hyvä ohjeistus mahdollisesti myös helpottaa käyttäjiltä tulevaa vastarintaa uutta järjestelmää kohtaan. Päävastuu ohjeistuksista on projektipäälliköllä.

Määrittelijä laatii yhteistyössä toteuttajien kanssa käyttäjien ohjeet, jotka voivat olla sähköisessä muodossa järjestelmän yhteydessä. Suunnittelijan vastuulla on toteuttaa teknisen toteutuksen käsikirja, joka sisältää tietojärjestelmän hallinnoinnin ja arkkitehtuurin.

- Tietokannan toteutus

Suunnitteluvaiheessa tehty suunnitelma tietokannasta toteutetaan. Tietokannan toteutukseen liittyy tärkeänä osana myös huollon suunnittelu. Tarkoituksena on miettiä, miten tuotannossa hoidetaan varmistukset, tietoturva ja tietokannan palautus sekä valmistautua mahdollisiin vikatilanteisiin, jolloin niistä selvittäisiin mahdollisimman nopeasti ja kivuttomasti. Tietokannan huoltoon liittyvät vastuuroolit/-henkilöt nimetään huoltosuunnitelmassa. Tietokannan luomisesta ja huoltosuunnitelmasta vastaa suunnittelija.

- Ohjelmiston toteutus

Ohjelmisto voidaan tuottaa ohjelmoimalla ja/tai kokoamalla se uudelleenkäytettävistä komponenteista tai moduuleista. Toteutus jaetaan varsinaisen ohjelmiston toteutukseen, yleisosien toteutukseen ja valmisosien liittämiseen. Toinen osa ohjelmiston toteutusta on yksikkö- ja integrointitestaus, jossa varmistetaan ohjelmien toimivuus. Ohjelmisto-osat testataan testaussuunnitelman mukaisesti. Tarpeen mukaan testaustapauksia suunnitellaan ja toteutetaan lisää.

Ohjelmoinnin suorittavat ammattitaitoiset henkilöt, jotka pystyvät toteuttamaan sekä toiminnalliset vaatimukset että ei-toiminnalliset, kuten suorituskykyyn tai skaalautuvuuteen liittyvät tekijät. Toteutuksesta vastaa toteuttaja. Ohjelmistosuunnittelun tulokset, luokka-, komponentti ja tapahtumakulkukaaviot antavat lähtökohdan ohjelmakoodille.

Ohjelmiston toteutukseen liittyy vahvasti järjestelmän testaus. Kun järjestelmä testataan perusteellisesti, syntyy siitä suunnittelun tuloksiin perustuva, helposti ylläpidettävä ja toimiva ohjelmisto.

Ohjelmointi ja testaus kestävät yleensä joitakin kuukausia, mutta ohjelmiston jatkuva kehittäminen voi kestää vuosia.

- Toteutuksen katselmointi

Toteutuksen katselmoinnissa varmistetaan, syntyneiden dokumenttien kattavuus ja oikeellisuus määritysten valossa. Toteutuksen katselmointi on jaettu testauskatselmukseen, toteutuksen tulosten kokoamiseen ja ohjelmistokatselmukseen. Päävastuussa toteutuksen katselmoinnista on katselmoija.

Pääpaino katselmuksessa on selvittää, vastaako ohjelmisto määrityksiä ja asiakkaan tarpeita sekä onko testaus toteutettu testaussuunnitelman mukaisesti ja täyttävätkö testaustulokset testaussuunnitelmassa asetetut kriteerit. Näiden tehtävien vastuu on omistajalla sekä katselmoijalla, projektipäällikkö kokoaa toteutuksen tulokset. (Yritys X 2006.)

# 3 SOVELLUSKEHITYSPROJEKTI

## 3.1 Yleistä

Tarve suunnitella ja rakentaa kokonaan uusi tuote vanhan sovelluksen johtui pääosin edeltäjänsä vanhanaikaisuudesta ja ylläpidon tuottamasta työkuormasta. Suuri työkuorma johtuu pääosin järjestelmän A arkkitehtuurista. A on paikallinen sovellus, joka edellyttää etätyöpöytäyhteyttä aina ylläpitotehtäviä suoritettaessa. Ylläpito etätyöpöytäyhteyksien kautta on työlästä sekä hidasta. Näistä syistä aloitettiin uuden tuotteen suunnittelu.

Yksi hallinnollinen syy tuotteen uudelleenrakennukselle on siirtomäärien ja -tapahtumien seurannan puuttuminen A:sta. Tulevaisuudessa on tarve pystyä asiakaskohtaisesti laskemaan siirrot kappalemääräisesti, tavuissa ja tietueissa. Näiden järkevään hallinnointiin ei ole muuta vaihtoehtoa kuin kootussa paikassa oleva automaattinen laskuri, johon uuden sovelluksen arkkitehtuuri soveltuu erittäin hyvin.

## 3.2 Järjestelmä A

Tiedonsiirtojärjestelmä A (Yritys X 2007a) on yritysten ja organisaatioiden tiedonsiirtotarpeisiin tehty integraatiosovellus. Sillä voidaan toteuttaa monimutkaisia ja vaativia tiedonsiirtoketjuja salattuna tai salaamattomana organisaation sisäisesti sekä ulkoisesti. A on luotettava ja turvallinen järjestelmä, jossa on todella monipuoliset määrittelymahdollisuudet ja on siksi myös hyvin joustava.

Tiedonsiirtojärjestelmällä A onnistuu tiedonsiirto niin sisäverkossa kuin Internetin välityksellä. Molemmissa tapauksissa tiedonsiirto voidaan salata, jotta asiattomat eivät pääse vakoilemaan siirrettäviä aineistoja. Järjestelmällä voidaan siirtää tiedostoja myös eri käyttöjärjestelmäpohjaisten palvelimien välillä. Se tukee mm.

Windows-, Linux-, AIX- ja VMS-järjestelmiä. A:n perusominaisuuksiin kuuluu myös lokitoimenteet, sähköpostiraportointi, merkkikonversiot sekä tiedostojen pakkaus tai purkaminen lähde- tai kohdepäässä.

Käyttäjille on olemassa erilaisia käyttäjätasoja eri oikeuksilla. Järjestelmää A ohjataan suoraan palvelimelta tai etänä erikseen asennettavalta Client-sovellukselta. Käytäntö on kuitenkin osoittanut, että yleensä on helpompi tehdä asetukset etäyhteydellä palvelimelle, koska etäyhteys on joka tapauksessa oltava ylläpitoa varten.

A on käyttäjälle käytännössä näkymätön, koska se toimii automaattisesti tarvittavien määritysten jälkeen. Se toimii siis käyttäjän tai ylläpitäjien tekemien määritysten, siirtoasetusten ja ajastusten perusteella niin kauan kunnes toisin määritetään.

A perustuu TCP/IP yhdistelmäprotokollaan. Mahdollisia tiedonsiirtoprotokollia A:ssa ovat esimerkiksi suojaamaton FTP(File Transfer Protocol) ja SSH/SSH2-suojattu SFTP(SSH File Transfer Protocol).

Asiakkaiden suunnalta tullut palaute on ollut A:n osalta lähinnä käytettävyyteen ja saatavuuteen liittyvää. Saatavuudella tarkoitan tässä yhteydessä, sitä kuinka helposti sovellukseen pääsee käsiksi On toivottu, että tapahtuneista siirroista kirjoitettuja lokeja olisi helpompaa päästä lukemaan, koska on tilanteita, jolloin asiakas haluaa varmistua tärkeiden siirtojen toteutumisesta.

## 3.3 Järjestelmä B

Järjestelmä B on järjestelmä A:n korvaava tuote, jolla voidaan hoitaa esimerkiksi organisaation henkilöstön- ja taloudenohjausjärjestelmien sekä sisäinen että ulkoinen liikenne. Järjestelmän lähtökohdat ovat helppokäyttöisyys, turvallisuus, tehokkuus ja luotettavuus.

Järjestelmä koostuu kahdesta erillisestä osasovelluksesta. Yritys X:n tuotantoverkossa sijaitsevasta Web-hallintaliittymästä, sekä asiakkaan sisäverkossa sijaitsevalle palvelimelle asennettavasta Client-osasta, tiedonsiirtomoottorista. Web-sovellus tuo todella merkittäviä uusia ominaisuuksia vanhaan A:han verrattuna. Suurimpia etuja on keskitetty hallinta ja ylläpito. Tämä tuo suuren hyödyn, koska jokaisen asiakkaan käyttäjätietoja, siirtoasetuksia, lokeja yms. voidaan tarkastella saman hallintaliittymän kautta. Näin päästään eroon etäyhteyksien epäkäytännöllisyydestä. (Yritys X 2009.)

### 3.3.1 Web-sovellus

Web-hallintaliittymä sijaitsee palvelimella tai palvelimilla, jotka on sijoitettu X:n tuotantoverkkoon ja hallintaliittymää kutsutaan julkisella http-osoitteella. Mikäli palvelimia tulee useampia, ne klusteroidaan eli hajautetaan ja näin saadaan palvelimelle kohdistettua kuormaa tasattua.

Pääsyä järjestelmään rajoitetaan käyttäjäkohtaisilla käyttäjätunnuksilla, jotka ovat liitetty erikseen määritetyin oikeuksin asiakkaan yhteen tai useampaan installaatioon. Käyttäjätunnuksia voi asiakkaalla olla kahdentyyppisiä, pääkäyttäjä tai käyttäjä. Pääkäyttäjällä on oikeudet kaikkiin toimintoihin, jotka kohdistuvat kyseiseen installaatioon. Hän voi myös myöntää oikeuksia eteenpäin. Käyttäjällä on kyseiseen installaatioon ne oikeudet, jotka pääkäyttäjä hänelle myöntää.

Web-liittymä on pelkästään hallintasovellus. Sillä ei ole minkäänlaista toiminnallisuutta liittyen itse siirtoihin. Web-hallintaliittymä on käytännössä loppukäyttäjälle ainoa näkyvä osa järjestelmästä. Sen avulla määritetään kaikki sovelluksen asetukset ja siirtomääritykset. (Yritys X 2009.)

### 3.3.2 Client

Client, eli tiedonsiirtomoottori on järjestelmän toiminnallinen osa, joka suorittaa kaikki määritellyt siirtotapahtumat. Se on paikallisesti organisaation sisäverkossa sijaitsevalle palvelimelle asennettava osasovellus. Client suorittaa myös mahdollisen väliaikaisarkistoinnin. Kaikki nämä ohjaustiedot client noutaa web-sovelluksen palvelimelta clientin ID:tä vastaavasta instanssista. Asetusten säätäminen tapahtuu pääasiassa vain web-hallintaliittymän avulla. Clientin puolella asetusten muokkaus on kuitenkin jätetty mahdolliseksi.

Client on itsenäisesti toimiva osa sovelluskokonaisuudesta. Se ei ole riippuvainen hallintasovelluksesta ohjaustietojen syötön jälkeen. Client on suunniteltu siten, että sen toiminta ei lakkaa vaikka web-hallintaliittymän ja clientin välillä olisikin esimerkiksi tietoliikennekatkoksia. Client suorittaa sille määritettyjä tiedostonsiirtotapahtumia uusimmilla määrityksillä, jotka se on noutanut web-sovelluksen palvelimelta. Osasovellus välittää myös lähes reaaliaikaisesti tietoa omasta tilasta, suoritetuista siirroista, ongelmista yms., eli lokitietoa web-palvelimelle. Tällöin lokitiedot ovat nopeasti saatavilla web-hallintaliittymän kautta.

Clientin tärkeimpiä vaatimuksia on luotettavuus. Koska se on sovelluskokonaisuudessa varsinaisen työn tekijä, on sen oltava erittäin luotettava. Luotettavuutta nostetaan muun muassa ratkaisulla, joka tarkkailee clientin tilaa palvelimella. Mikäli se huomaa clientin toiminnassa virheitä, esimerkiksi palvelu on kaatunut, niin se pyrkii uudelleenkäynnistämään tämän. (Yritys X 2009.)

# 4 KÄYTÄNNÖN SUUNNITTELU JA MÄÄRITTELYT

Tässä kappaleessa käydään läpi projektin toteutusta. Ensin otetaan esiin hieman taustoja ja syitä projektin toteutukselle, jonka jälkeen esitellään työskentelytapoja ja työkaluja. Tärkeimpänä tulee haastatteluiden tuloksien esittely, eli teoriamallin noudattaminen pääkohdittain. Tämän jälkeen käsitellään vielä hieman dokumentoinnin tasoa ja tärkeyttä sekä tuodaan hieman esille yhteistyötä alihankkijan kanssa.

## 4.1 Projektin aloitusvaihe

Ensimmäiset ajatukset järjestelmän A kehittämisestä syntyivät keväällä 2007, lähinnä uuden järjestelmän C pohjalta. C on palvelujärjestelmä, jolla voi lähettää aineistoa ennalta määritetyille vastaanottajille. C:stä oli olemassa vanha järjestelmä, mutta se rakennettiin käytännössä kokonaan uudelle arkkitehtuurille. C muutettiin täysin, ohjelmointikieli vaihtui, arkkitehtuuri suunniteltiin täysin alusta ja tärkein muutos oli C:n muuttaminen paikallisesta sovelluksesta web-sovellukseksi. Kaikki tämä helpotti ylläpitoa ja päivittämistä vanhaan sovellukseen nähden todella huomattavasti. (Yritys X, 2007b.)

Koska C:n ratkaisumalli oli todettu erittäin toimivaksi, ajatuksena oli, että samaa toimintamallia voitaisiin hyödyntää myös A:n mahdollisessa uudistamisessa. Järjestelmät C ja A ovat käytännössä kuitenkin hyvin lähellä toisiaan toimintamalliltaan. Molemmilla voi suorittaa osittain samat toimenpiteet. C:llä siirto tapahtuu esimääritellyille vastaanottajille manuaalisesti, kun taas A:lla kohdepäätä ei ole rajoitettu, vaan se voidaan itse määritellä jokaista siirtoa varten erikseen. Manuaalilähetys A:lla taas on hyvinkin paljon hankalampaa kuin C:llä, koska järjestelmä on asennettu paikallisesti palvelimelle, joka yleensä sijaitsee organisaation ATK-konesalissa tai palveluntarjoajan tiloissa. C:llä lähetys voidaan suorittaa miltä tahansa työasemalta, jolta on pääsy Internetiin. (Yritys X, 2007b.)

Ensimmäisten alustavien suunnittelupiirrosten jälkeen pääpaino ajatustyössä ja palavereissa siirtyi tuotteen markkinahyödyn tutkimiseen. Miksi toimiva tuote pitäisi uusia? Mitä taloudellista hyötyä siitä saadaan, kuinka paljon ja miten? Uutta tuotetta ei kuitenkaan suunnitella ja toteuteta vain sen vuoksi, että se olisi parempi kuin edeltäjänsä. Kaiken toiminnan on kuitenkin aina tähdättävä taloudelliseen voittoon.

Pääosin järjestelmä B tulisi vaikuttamaan ylläpidettävyyden helppouteen ja asiakasmassojen hallittavuuteen. Melko suuri työkuorma kertyy A:ssa jo pelkästään asiakkaiden pyynnöistä tarkistaa siirtojen lokit. Järjestelmä B:ssä kuka tahansa tiimistä voi tehdä tämän tai jopa asiakas, koska siirtojen tarkistamiseen ja lokien lukemiseen ei enää tarvittaisi etäyhteyttä asiakkaan palvelimelle. B tarjoaisi myös kattavan listan käyttäjistä ja käyttäjäkohtaisista tapahtumista, jolloin asiakasmassojen hallittavuus olisi huomattavasti helpompaa. Käytännössä siis ylläpidon työmäärä olisi huomattavasti pienempi kuin A:ssa.

Yksikön liiketoiminta painottuu pääosin kuntatoimialalle ja julkishallintoon. Lähitulevaisuudessa suurin työllistävä tekijä on kuntaliitokset monien kuntien välillä. Monilla näistä kunnista on käytössään järjestelmä A, mikä tarkoittaa suuret määrät muutostöitä. Kuntaliitosten yhteydessä monissa paikoissa palvelimet vaihtuvat, osa palvelimista poistuu käytöstä ja koska kahden tai useamman kunnan tiedonsiirtotarpeet muuttuvat käytännössä täysin, on monessa tapauksessa siirtomääritykset suunniteltava ja määriteltävä täysin alusta. B toisi mukanaan huomattavia helpotuksia näiden määritysten luomiseen ja hallittavuuteen, mikä taas helpottaisi käyttöönottojen suurten muutosten toteuttamista.

Jo nämä parannukset helpottaisivat ylläpidon ja uusien asennusten työkuormaa siinä määrin, että järjestelmä olisi järkevää toteuttaa. Varmasti myös tuotteen markkinointi ja myynti helpottuisi sen käytettävyyden ja uusien ominaisuuksien myötä. Vaikka nykypäivän tekniikka ja tietoturva ovat tärkeitä myyntiargumentteja, niin silti usein tärkeimpänä yksittäisenä tekijänä pidetään luotettavuutta ja helppokäyttöisyyttä.

## 4.2 Työskentelytavat ja työkalut

Ennen kuin projekti virallisesti alkoi, oli määrittelyjä työstetty ominaisuuksien osalta melko runsaastikin. Sovelluksen edeltäjän ylläpitäjät olivat koonneet reilun vuoden ajan toiveita ja ominaisuuksia, joita tuotteeseen kaipaisivat. Käytännössä he keräsivät kehityslistaa vanhalle tuotteelle, koska uudesta tuotteesta ei ollut vielä puhetta tässä vaiheessa. Ominaisuuksien keräämisvaiheessa ei kuitenkaan ollut vielä uusi sovellus tähtäimessä, vaan enemmänkin päivityksiä vanhaan.

Ensimmäiset visiot uudesta korvaavasta tuotteesta tulivat pääosin yhdeltä henkilöltä. Myöhemmin koko tiimi osallistui ideoimiseen ja uusien ideoiden räätälöintiin. Kun ideoita uuden tuotteen toteuttamiselle oli niin paljon, että tiimin keskuudessa oltiin vakuuttuneita, että tuote on arkkitehtuurillisesti toteutettavissa, ja mahdollisesti myös taloudellisesti tuottava, alettiin pitää entistä vakavampia suunnittelupalavereita. Alkuun palavereissa oli koko tiimi mukana, mutta mitä syvemmälle ja teknisemmäksi suunnittelu eteni, sitä pienemmäksi ja intensiivisemmäksi suunnitteluryhmä kävi.

Teknisen suunnittelun osalta ryhmässä oli kolme henkilöä. Näistä kaksi henkilöä työskenteli samalla konttorilla ja yksi eri paikkakunnalla. Tämä tietysti asetti eräänlaisia haasteita kommunikointiin ja tiedon jakamiseen kaikkien välillä. Työskentely tapahtui muun muassa sähköpostin, puhelimen ja pikaviestimien välityksellä. Tehokkaimpina näistä pidettiin reaaliaikaisia pikaviestimiä, kuten Microsoft Office Communicator ja Skype, sekä tietysti sähköposti ja puhelin.

Työt olivat monesti myös yksilötyöskentelyä, kuten ominaisuusluettelon päivittämistä. Käytössä oli CaliberRM-järjestelmää, joka on suunniteltu nimenomaan sovelluskehitystä varten ja sitä käytetään vaatimusluettelon dokumentointiin. Jokaiselle vaatimukselle voitiin määrittää esimerkiksi vaatimustaso, mihin versioon se toteutetaan, vastuuhenkilö, kommentit ja selventävät selitykset, sekä paljon muuta. Usein Caliber-järjestelmän käyttö tuli eteen uusien ideoiden myötä, jolloin ne kaikki kirjattiin ylös. Suurimmaksi osaksi vaatimukset tulivat vanhan A:n yllä-

pitäjiltä, koska heillä oli käytännön kokemusta siitä, mitä puutteita järjestelmässä A on ja vastaavasti mikä siinä on ollut hyvää.

Jo aiemmin esitellyllä SoftwareMethods-systeemityömenetelmällä oli tietysti merkittävä rooli työkalujen joukossa, vaikka se onkin enemmän menetelmämalli. Se kuitenkin tarjoaa hyvät pohjat dokumentoinnille ja määrittelytyön seurantaan.

Merkittävässä asemassa oli myös Microsoftin Office-tuoteperheen Microsoft PowerPoint -sovellus, sekä Microsoft Paint, joita käytettiin hyvin paljon järjestelmän toiminnallisuuden hahmotteluun ja yhteyskaavioihin.

## 4.3 Vesiputousmallin noudattaminen

Määrittelymallin noudattamista on tutkittu tarkkailemalla projektin etenemistä ja haastattelemalla projektin osallisia. Huomioitava seikka on se, että mallin eri vaiheissa saattavat samat aiheet toistua. Myöhemmin esiintyvässä vaiheessa vaatimukset ovat kuitenkin huomattavasti syvällisempiä. Usein pienissä projekteissa näitä vaiheita on hyvin vaikea erottaa toisistaan. Ne on monesti toteutettu yhdellä kertaa tarvittavalle tasolle ja näin on toimittu myös tässä projektissa. Siitä syystä tässä työssä samankaltaisia vaiheita on vertailua suoritettaessa yhdistelty.

Toinen asia, joka on otettava huomioon, on kokonaan tai osittain alihankkijan vastuulla olevat osat toimintamallista. Näitä ovat sovelluskehitysmallin suunnittelu- ja toteutusvaihe.

### 4.3.1 Vaatimusmäärittely

Tässä kappaleessa käsitellään työn pääasiaa, eli haastatteluiden ja tarkkailun tuloksia. Tarkoituksena on tuoda esille mitkä kohdat sovelluskehitysmallista on toteutettu ohjeita noudattaen ja missä mallia on sovellettu. Samassa on tuotu esille haastateltavien näkemyksiä kyseisestä sovelluskehitysmallista yleisellä tasolla sekä projektiin liittyen. Haastatteluiden teemat on kerrottu luvussa 1.

Mallin eri vaiheet on eroteltu pienemmiksi kokonaisuuksiksi ja niitä käydään läpi jo edellisessä kappaleessa mainitulla tavalla. Ei siis täysin teoriamallin mukaisesti vaan vaiheiden yhdistäminen huomioiden.

- Toimintaympäristön kuvaaminen

  Toimintaympäristön kuvaaminen on osa-alue, jota projektin osalliset pitivät suhteellisen tärkeänä, mutta ei välttämättä kyseisen projektin kohdalla. Syy tälle on projektin koko ja se, että kaikki projektin osalliset periaatteessa tuntevat toimintaympäristön sovellukselle, koska vanhassa tuotteessa on lähes sama toimintaympäristö.

  Siihen nähden tämä kohta oli toteutettu riittävällä tasolla, että alihankkija sai hyvän käsityksen sovelluksen toimintaperiaatteesta, mutta ei täysin samalla tasolla kuin määrittelymalli sitä edellyttäisi. Käytännössä siis tämä osa-alue oli huomioitu, mutta sovelletulla tasolla.

- Sanaston kokoaminen

  Sanaston kokoaminen oli haastateltavien mielestä yleisellä tasolla hyvin tärkeää. Usein teknisissä projekteissa on hankalaa sanastoa sekä sellaista sanastoa, jolla on päällekkäisiä tarkoituksia. Nämä ovat

hyvin tärkeitä erotella ja määritellä, jotta projektin alusta lähtien puhutaan oikeista asioista oikeilla nimillä.

Mielipiteet kyseisen projektin osalta hieman vaihtelivat tämän osalta. Oltiin sitä mieltä, että kyseisessä projektissa useimmat termit ovat itsestäänselvyyksiä kaikille osallisille ja ne ovat yleisesti tunnettuja termejä. Toisaalta oltiin myös mieltä, että projekti sisältää niin paljon spesiaalisanastoa, että sanaston kokoaminen on hyvin tärkeää projektin sujuvan toteutuksen kannalta.

Projektissa ei kuitenkaan varsinaisesti ole kerätty sanastoluetteloa, vaan mikäli dokumentissa esiintyy sana, joka ehkä vaatii selventämistä, on se selitetty kyseisessä dokumentissa. Sanaston kokoamisen osalta voisi siis todeta, että sitä ei ole noudatettu täysin ohjeen mukaan, mutta joitain selvennystä vaativia sanoja on kuitenkin dokumentoinnissa avattu.

- Vaatimusten kuvaaminen

Kuten projektin aloitusvaiheen kuvauksessa jo käy ilmi, vaatimusluettelon kerääminen oli aloitettu jo ennen varsinaista projektin aloitusta. Haastatteluissa tuli ilmi, että työvälineeseen ei juuri oltu tyytyväisiä, mikä ehkä osaltaan vähän jarrutti vaatimusluettelon keräämistä. Vaihetta pidettiin kyllä yhtenä tärkeimmistä asioista määrittelyvaiheessa, koska pienetkin muutokset jälkikäteen voivat tulla todella kalliiksi. Näin ollen huomattavasti helpommalla pääsee, kun vaatimusluettelo on niin kattava kuin mahdollista.

Vaatimuksia kirjoitettiin paljon. Vaatimusluettelo on vienyt yksittäisenä osa-alueena ehkä eniten aikaa määrittelyiden työmäärästä. Toisaalta niitä on kerätty muiden töiden ohessa. Kaikki olivat samaa mieltä siitä, että vaatimuksia on kerätty tarpeeksi. Ilmi kävi kuiten-

kin, että ne olisivat voineet olla ehkä vielä nykyistä tarkempia kuvauksia.

Kun projekti oli virallisesti aloitettu, vaatimukset myös luokiteltiin ja jäsenneltiin eri kategorioihin. Voisi sanoa, että tätä osa-aluetta mallista on noudatettu ehkä niin tarkkaan kuin projektin kannalta on järkevää, poislukien se, että vaatimusluettelon kerääminen alkoi jo ennen projektin perustamista.

- Toiminnallisuuden mallintaminen / analysointi

Määrittelyn ja suunnittelun aikana piirretyistä kuvitteellisista kuvakaappauksista ja tekniikkaa mallintavista kuvioista selviää jonkin verran toiminnallisia vaatimuksia järjestelmälle. Käyttötapauksia tai -rooleja ei ole erikseen kuvattu tai kerätty, mutta projektin osallisilla ne ovat kyllä tiedossa. Toisaalta, kuten aiemmin on esitelty, järjestelmän ideat pohjautuvat pitkälle järjestelmään C. Näin ollen myös toiminnallisuus on pitkälti järjestelmän C kaltainen. Näistä syistä tähän osa-alueeseen ei ollut panostettu sen enempää.

Yleisesti ottaen osa-alue on haastateltavien mukaan hyödyllinen, mutta tärkeydessään kuitenkin enemmän jälkipäässä, tai ei ainakaan kärkipäässä. Toiminnallisuuden mallintamista ei siis ollut sellaisenaan juurikaan huomioitu, mutta sen vaatimat asiat tulivat selviksi muun dokumentoinnin yhteydessä vähintäänkin projektin osallisille.

- Tietosisällön mallintaminen / analysointi

Tietosisällön mallintamista pidettiin enemmänkin alihankkijan vastuulla. Kuitenkin, jotta alihankkija voisi tästä suoriutua, heillä pitää olla hyvät tiedot suunniteltavasta järjestelmästä, jotta tietosisällön voi mallintaa tehokkaasti. Varsinaisesti tähän osa-alueeseen ei keski-

tytty yksinään, mutta jollain tasolla näitä tietoja löytyy muusta dokumentaatiosta ja toteuttajalle jää vapaammat kädet.

Yleisellä tasolla kyseinen osa-alue nähtiin tärkeänä. Jos tässä mentäisiin pieleen, niin on vaara, että koko projekti menisi väärään suuntaan.

- Arkkitehtuurin määrittely

Sovelluksen arkkitehtuuria on määritelty piirtämällä ja on haastateltavien mielestä sillä tasolla kuin pitääkin. Arkkitehtuuri myös testattiin POC-versiolla, joka oli juuri sopivassa mittakaavassa toteutettu; ei liian raskas tai työläs. Se ei myöskään olisi tuottanut liiallisia taloudellisia tappioita, jos arkkitehtuuri olisi todettu toimimattomaksi.

Määrittelyissä syntynyt arkkitehtuuri mukailee toimialan arkkitehtuuristrategiaa niin pitkälle kuin uudentyyppisen arkkitehtuurin puitteissa on mahdollista.

Kaikki haastateltavat näkevät tämän osa-alueen yhtenä tärkeimmistä asioista sovelluskehityksen määrittelyvaiheessa. Varsinkin kyseisessä projektissa, jossa sovelluksen arkkitehtuuri on sellaista, jota ei aiemmin yrityksessä ole käytetty.

- Katselmoinnit

Katselmointi nähtiin siinä mielessä tärkeänä asiana, että tuotetut dokumentit olisivat riittävällä tasolla, jotta niiden perusteella voidaan toteuttaa järjestelmä. Haastateltavien mukaan katselmointi ei välttämättä ole tärkeää toteuttaa jokaisen vaiheen jälkeen, varsinkaan kyseisessä projektissa.

Vaatimusmäärittelyn katselmointia ei projektissa B ollut toteutettu, ainakaan vielä haastatteluhetkellä. Joitain dokumentteja on jo määrittelyn aikana yhdistelty ja kerätty nippuun, mutta varsinaista katselmointia dokumentoinnille ei ollut toteutettu.

### 4.3.2 Vaatimusanalyysi

Kuten aiemmin jo mainittu, kyseisen projektin osalta vaatimusanalyysin ja määrittelyn vaiheita on hyvin vaikea erotella toisistaan, koska niissä käsitellään samoja asioita, ainoastaan eri tasolla. Varsinkin vaatimusanalyysissä tämä korostuu ja tästä syystä vaatimusanalyysistä on nostettu esille vain yksi kohta, jota ei vielä määrittelyvaiheessa käsitelty.

- Käyttöliittymän mallintaminen

  Käyttöliittymän mallintaminen oli aloitettu hyvissä ajoin. Mallintamiseen käytettiin kuvitteellisia kuvakaappauksia ja ne ovat haastateltavien mukaan pääosin hyvällä tasolla, vaikka joiltain osin olisi voinut olla jopa tarkempia ja pidemmälle suunniteltuja.

  Kuvitteelliset kuvaruutukaappaukset toimivat suunnittelussa pohjana ja visuaalisena ilmentymänä tulevasta sovelluksesta. Piirtäminen koettiin tässä tapauksessa helpommaksi ja paremmin havainnollistavaksi kuin pelkät sanalliset selitykset.

  Jo POC-vaiheessa hyödynnettiin käyttöliittymän mallintamisen tuloksia, eli POC näytti karkealla tasolla hieman samansuuntaiselta kuin tuleva sovellus tulee näyttämään.

  Projektin osalliset kokivat tämän osa-alueen yhtenä tärkeimpänä koko sovelluskehityksen määrittelyvaihetta.

## 4.4 Dokumentointi

Dokumentaation merkitys on tunnetusti hyvin tärkeässä roolissa riippumatta toimialasta. Sovelluskehityksessä dokumentaatiolla pyritään selvittämään mahdollisimman yksiselitteisesti esimerkiksi järjestelmän vaatimukset, mitä järjestelmän pitää pystyä tekemään ja miten se toteutetaan. Sovelluskehityksessä dokumentaatiota tehdään monista eri asioista, kuten arkkitehtuurista, tietoturvasta, käyttäjistä, toiminnallisuudesta ja markkinoinnista.

Kyseisessä projektissa dokumentoinnin suhteen oltiin siinä mielessä hyvin valveutuneita, että dokumentointia ei haluttu tehdä ainoastaan dokumentoinnin vuoksi. Dokumentteja tehtiin niistä asioista, joita osalliset kokivat olevan tarpeellisia kokonaisuuksien ymmärtämiselle ja muistamiselle. Dokumentointia syntyi muun muassa arkkitehtuurista, tekniikasta, toiminnallisuudesta ja käyttöliittymästä sekä vaatimusluettelo. Dokumentaatiota ei ole lähdetty suorittamaan laajan mittakaavan mukaan, koska itse projekti on suhteellisen pieni. Dokumentaatio on syntynyt käytännössä suunnittelun yhteydessä, palavereissa sekä alihankkijan toimesta.

Dokumentaatio koostuu siis pääasiassa siis CaliberRM:stä luodusta vaatimusluettelosta, toiminnallisuutta ja arkkitehtuuria mallintavista tekniikkapiirroksista sekä kuvitteellisista kuvakaappauksista, joista selviää hyvinkin tarkasti, mitä ominaisuuksia ja toiminnallisuuksia sovellukselta halutaan.

## 4.5 Yhteistyö alihankkijan kanssa

Määrittelyvaiheessa alihankkijan kanssa pidettiin muutamia palavereita, joissa käytiin läpi tehtyjä suunnitelmia ja pyrittiin selvittämään mahdollisimman pitkälle järjestelmän toteutuskelpoisuus. Kun alihankkijan kanssa oltiin yhtä mieltä toteutuskelpoisuudesta, jatkettiin vielä suunnittelua yhteistyössä heidän kanssaan. Alihankkijalta saatiin lisää näkemystä ja toteutustapoja tekniseltä kantilta, sekä hyviä ideoita käytettävyyden kehittämiseksi.

Mitä pidemmälle määrittelytyöstä projekti eteni, sitä enemmän yhteistyö alihankkijan kanssa astui kuvaan. Jatkuva suunnittelu ja testaus yhdessä alihankkijan kanssa vain lisääntyivät mitä pidemmälle projektissa ja POC:ssa edettiin.

# 5 YHTEENVETO TULOKSISTA

## 5.1 Yleistä

Yleisesti ottaen projektin määrittelyvaihe oli onnistunut suunnitelman mukaisesti. POC-vaiheessa todettiin suunnitellut tekniikat mahdollisiksi ja oltiin valmiita jatkamaan eteenpäin. Projektin läpivienti siihen asti oli ollut riittävällä tasolla, joten mitään tärkeää tai projektin kannalta kriittistä ei ollut unohdettu tai jätetty huomiotta. Joistain seikoista myös POC:n jälkeen oli jäänyt asioita vielä auki, mutta ei mitään, joka estäisi jatkamisen.

Haastattelut tutkimuksessa onnistuivat, niistä saatiin juuri ne tiedot, joita haluttiin. Mallin ja käytännön toteutuksen erot tulivat selkeästi esiin haastateltavien mielipiteiden kera. Näin ollen tuloksista tuli selkeitä ja erot teorian ja käytännön toteutuksen välillä ovat hyvinkin selkeästi havaittavissa.

## 5.2 Tulokset

Yrityksen vesiputousmalli on hyvin laaja, monitasoinen ja mahdollisimman yleispäteväksi tehty. Ohjeena tällaisen mallin käyttöön on yleensä sen soveltaminen projektin luonteen mukaan. Projektipäällikön tehtävänä on miettiä, mitkä asiat mallista ovat projektille tarpeellisia.

Kyseinen sovelluskehitysprojekti itsessään on yrityksen mittakaavassa suhteellisen pieni, jos sitä vertaa esimerkiksi taloudenohjausjärjestelmiin. Näin ollen alusta asti oli odotettavissa, ettei mallia toteuteta orjallisesti noudattaen. Oletus piti hyvin paikkansa.

Käytännössä teoriamalli oli mukana tiiviisti ja antoi hyvän viitekehyksen projektin läpiviennille. Aivan alusta lähtien kuitenkaan mallia ei käytetty. Kuten tutkimuk-

sessa tuli ilmi, suunnittelu oli alkanut jo ennen kuin virallinen projekti oli perustettu. Myöskään mallin monitasoisuutta ei noudatettu tarkasti. Tällä tarkoitan, ettei samaa aihetta välttämättä jaoteltu erikseen määrittely- ja analyysivaiheeksi, vaan niitä käsiteltiin yhtenä kokonaisuutena. Projektin koosta johtuen myöskään yksittäisiä vaiheita ei yleensä käyty läpi teoriamallin edellyttämällä tasolla. Esimerkiksi dokumenttipohjia ei projektissa juurikaan käytetty, vaan asiat dokumentoitiin omilla tavoillaan. Ne olivat yleensä piirroksia, PowerPoint-esityksiä tai vapaasanaisia määrityksiä. Joitain teoriamallin kohtia jäi selkeästi jopa kokonaan toteuttamatta.

Se miltä osin mallia noudatettiin hyvin, oli sen antama pohja kaikelle huomioitaville seikoille sovelluskehityksessä. Sen vaiheita noudatettiin hyvässä järjestyksessä, vaikka usein palattiinkin ”taaksepäin”. Vaiheiden välillä siirtyminen on monesti hyvin suositeltavaa. Sillä mahdollistetaan aikaisempien dokumentointien tarkentaminen uusien ideoiden ja muutosten jälkeen, mikä parantaa kokonaisuutta.

Yleisesti ottaen kaikki haastateltavat pitivät mallia hyvin tehokkaana ja hyödyllisenä. He kaikki olivat sitä mieltä, että kaikki hiemankin suuremmat sovelluskehitysprojektit vaativat tämänkaltaisen ”ohjenuoran” sillä painotuksella, että sitä pitää ehdottomasti soveltaa ottaen huomioon projektin koko, luonne ja sovelluksen käyttötarkoitus. Sellaisinaan mallit ovat yleensä aivan liian raskaita ja teettävät tarpeetonta työtä.

Kyseisen projektin kohdalla haastateltavat olivat pääosin tyytyväisiä mallin toimivuuteen ja siihen, miten sitä noudatettiin. Dokumentoinnissa oli saavutettu yleisesti riittävä taso ja mallin käyttäminen esti asioiden unohtumisen tai puuttumisen toteutetusta määrittelystä. Kuitenkaan ei suljettu pois sitä vaihtoehtoa, että mallia olisi käytetty vieläkin enemmän, ainakin joiltain osin. Siitä ei varmasti olisi ollut haittaa, mutta siitä ei myöskään oltu varmoja olisiko hyöty ollut enää yhtä lineaarista kuin siihen asti. Vai olisiko siihen käytetyn ajan hyöty ollut jo huomattavasti pienempi.

## 5.3 Päätelmiä

Ainakin näiden tulosten ja käyttäjien mielipiteiden mukaan voisi päätellä, että sovelluskehityksen tueksi tuotetut mallit ovat hyviä, elleivät jopa välttämättömiä kehityksen onnistumisen kannalta. Sellaisenaan malli saattaisi toisaalta aiheuttaa projektille jopa suurta hidastumista ja esteitä, mutta sovellettuna se on hyödyllinen. Tutkimuksessa ei tullut ilmi mitään, mikä olisi tukenut ajatusta olla käyttämättä mitään mallia apuna. Mallit ovat kuitenkin, ainakin pääosin hyväksi havaittuja ja auttavat pitämään projektin urillaan. En näe mitään syytä olla käyttämättä ainakin jonkinlaista mallia kaikessa sovelluskehityksessä, poislukien todella pienet muutamien päivien projektit. Kysymykseksi jääkin se, että kuinka laajalti tällaisia malleja tulisi oman projektin kohdalla käyttää.

Yritykselle tästä opinnäytetyöstä on ollut varmasti useita eritasoisia hyötyjä. Tämänkaltaisten projektien läpivientiä harvoin seurataan tällä tasolla. Työn onnistumista ei juuri analysoida muuten kuin lopputuotteen onnistumisen mukaan, vaikka jälkianalysointi on erittäin hyödyllistä, ellei jopa välttämätöntä. Tämän työn tuloksista kuitenkin selviää missä onnistuttiin, mitä kannattaisi jatkossa tehdä toisin. Tämä taas auttaa toteuttamaan seuraavia projekteja entistä paremmin. Työllä saavutettiin jälkianalysointia jo ainakin projektin suunnittelu- ja määrittelyvaiheesta.

Dokumentointi opinnäytetyön kautta on myös ollut todennäköisesti tarkempaa ja parempaa kuin mitä se olisi normaalisti ollut. Haastattelut, mallin peilaaminen käytäntöön ovat osaltaan myös varmasti vaikuttaneet haastateltavien omiin mielipiteisiin mallin tärkeydestä. Varsinkin haastattelut saivat henkilöt ajattelemaan asioita tarkemmin ja uudesta näkökulmasta. Seuraavissa projekteissa malli varmasti otetaan myös hyötykäyttöön.

LÄHTEET

*Haikala, I., Märijärvi J.2006. Ohjelmistotuotanto. 11. painos. Helsinki: Talentum.*

*Pohjonen Risto. J.2007. Tietojärjestelmän kehittäminen. 3. painos. Jyväskylä: Docendo.*

*Harsu Maarit. J.2003. Ohjelmien ylläpito ja uudistaminen. Helsinki: Talentum.*

*Yritys X. 2006. SoftwareMethods, Vaihejakomalli. Yritys X. [Viitattu 10.12.2008]. Saatavissa Yritys X:n intraverkossa: http://ruori.yritysX.fi/sites/Ruorimalli/sivut/SoftwareMethods.aspx*

*Yritys X. 2007. A-tuotekuvaus. Esite. Yritys X..*

*Yritys X. 2007. C-palvelukuvaus. Palvelukuvaus. Yritys X.*

*Yritys X. 2009. B-järjestelmäsuunnitelma. Järjestelmäsuunnitelma. Yritys X.*

*Royce, W. 1970. Managing the development of large software systems. University of Maryland, Department of computer science. [Viitattu 19.2.2009]. Saatavissa: http://www.cs.umd.edu/class/spring20 03/cmsc838p/Process/waterfall.pdf*