Opinnäytetyö (AMK)

Insinööri, Tekniikka

2021

Juho Peltola

PROJEKTINA OHJELMISTOKEHITYKSEN OSTAMINEN

TURKU AMK
TURKU UNIVERSITY OF APPLIED SCIENCES

OPINNÄYTETYÖ (AMK) | TIIVISTELMÄ

TURUN AMMATTIKORKEAKOULU

Insinööri, Tekniikka

2021 | 43 sivua

Juho Peltola

PROJEKTINA OHJELMISTOKEHITYKSEN OSTAMINEN

Tässä opinnäytetyössä käsitellään ohjelmistokehityksen teettämistä alihankintana. Ohjelmistot ovat tärkeä osa kaiken kokoisten yritysten toimintaa. Kaikilla teollisuuden ja palveluiden aloilla tarvitaan erilaisia ohjelmistoja yrityksen sisäisessä toiminnassa, tuotteiden osana tai sellaisenaan myytävinä tuotteina. Aiemmin täysin mekaaniset ja fyysiset tuotteet saattavat nykyään vaatia lisäksi jonkin nettisovelluksen tai mobiiliaplikaation, jotta tuotteen on mahdollista pärjätä kilpailijoilleen. Useimpiin tarpesiin löytyy valmiit ratkaisut, mutta on myös paljon tilanteita, joissa aivan sopivaa ohjelmistoa ei löydy tai kaikkia tarvittavia ominaisuuksia ei ole valmiissa ohjelmistoissa. Näissä tilanteissa yrityksen on itse tuotettava ohjelmisto tai sen osia ja puuttuvia toimintoja.

Usein erityisesti pienissä yrityksissä ei ole omaa ohjelmistokehittäjää tai edes IT-osastoa, jolloin ohjelmistokehitys on luonnollisinta ostaa alihakintana alan yrityksiltä. Ohjelmiston tuotanto voidaan ostaa joko kokonaan alihankintana tai ostava yritys voi osallistua kehityksen eri vaiheisiin. On kuitenkin muutamia asoita, joihin teettävän yrityksen on tarpeellista osallistua hyvän lopputuloksen saavuttamiseksi.

Teettävällä yrityksellä saattaa olla jo melko tarkka vaatimusmäärittely tarvittavasta ohjelmistosta, mutta sen teettäminen valmiiksi ohjelmistoksi saattaa varsinkin pienissä yrityksissä olla haastavaa, erityisesti jos ohjelmistojen teettämisestä ei ole aiempaa kokemusta.

Tässä työssä selvitetään projektiluontoisen ohjelmistokehityksen ostamisen periaatteita ja vaiheita sekä perehdytään ohjemistotuotannon projektien hallintaan.

ASIASANAT:

alihankinta, ohjelmistokehitys, projektihallinta

BACHELOR´S THESIS | ABSTRACT

TURKU UNIVERSITY OF APPLIED SCIENCES

Engineer, technology

2021 | 43 pages

Juho Peltola

PROJECT: BUYING SOFTWARE DEVELOPMENT

This thesis examines outsourcing of software development and project working in software production. Software has an important role in companies of all sizes. All fields of industries, many types of software is required for company's internal operations, as part of products or as soldable products. Earlier only mechanical and physical products may now require an additional web application or mobile application to make the product stand out from its competitors. There are ready-made solutions for most needs, but there are also many cases where the right software cannot be found or not all the necessary features are in the ready-made software. In these cases, the company must produce the software to have missing functionalities.

Small companies do not usually have their own software developer. Especially in these companies is natural to buy software development as a subcontract from software companies. Software production can be outsourced either entirely or the buyer company can participate in various stages of development.

The company outsourcing development can already have a fairly precise specification of the software to produce. Even though, producing it into an actual software can be challenging, especially for small companies if there is no previous experience in buying software development.

This thesis focuses project working and principles and stages of outsourcing software development.

KEYWORDS:

outsourcing, project management, software development

SISÄLTÖ

1 OHJELMISTON HANKINNAN LÄHTÖKOHDAT 6

1.1 Liiketoiminnallinen tavoite 6

1.2 Taloudelliset lähtokohdat ja budjetti 7

2 VAATIMUSTEN MÄÄRITTELY 10

2.1 Vaatimusten ryhmittely ja määritysdokumentin teko 11

2.2 Mallinnustavat 14

2.3 Tietoturva 19

3 OHJELMISTOPROJEKTIN VAIHEET 22

3.1 Projektin suunnittelu 22

3.2 Projektityöskentely ja -organisaatio 23

3.3 Aikataulutus 24

3.4 Projektin seurannan työvälineet 25

4 OHJELMISTON SUUNNITTELUN JA TOTEUTUKSEN PERUSTEITA 27

4.1 Esitutkimus, vaatimusmäärittely ja toiminnallinen määrittely 27

4.2 Palvelinympäristön valinta 28

4.3 Toteutusprojekti 30

4.4 Testaus 31

4.5 Käyttöönotto ja elinkaaren hallinta 33

5 OHJELMISTOPROJEKTIEN TOTEUTUSMENETELMÄT 34

5.1 Vesiputousmalli 34

5.2 Ketterät menetelmät 35

6 DOKUMENTOINTI JA SOPIMUKSET 38

6.1 Dokumentointi 38

6.2 Sopimukset 38

7 YHTEENVETO 41

LÄHTEET 43

JOHDANTO

Tässä opinnäytetyössä selvitetään ohjelmistotuotannon tuottamisen vaiheita pienen yrityksen kannalta. Toimeksiantajana on Eventio Group Oy, joka on yhdistysten omistama, yhdistysten varainhankintaan keskittynyt yritys. Yrityksen päätuotteita ovat tavara-arpajaisiin ja bingopeliin liittyvät ohjelmistot, nettipeliportaali, pelitarvikkeet ja asiantuntijapalvelut.

Yrityksessä on jo pitkään tuotettu ohjelmistoja omaan sisäiseen käyttöön sekä myytäviksi tuotteiksi. Pääsääntöisesti kaikki ohjelmistot on tuotettu alihankintana ohjelmistoalan yrityksiltä. Ohjelmistojen tuotantoprojektin lisäksi ohjelmistoihin tarvitaan myös jatkuvaa kehitystä ja ylläpitoa, joka ostetaan osin alihankintana.

Yrityksellä on pitkät perinteet ohjelmistokehityksen ostamisesta alihankintana. Ensimmäisiä myytäviä ohjelmia on teetetty jo 80-luvun alkupuolella. Tällöin ohjelmistojen ja IT-palveluiden osuus liikevaihdosta oli hyvn pieni. Nykyisin pääosa liikevaihdosta tulee erilaisista digitaalisista kanavista ja palveluista. Erityisesti viime vuosina digitalisaatio on ottanut isompia askelia, jolloin ohjelmistotuotantoon on pitänyt kiinnittää erityistä huomioita. Tämän työn tarkoituksena on kehittää hankintaprosessia ja projektien hallintaa. Käytännössä tämän työn on tarkoitus parantaa tulevien ohjelmistokehitysprojektien onnistumista niin teknisesti kuin taloudellisestikin.

Tässä työssä on käytetty aiemmin kertynyttä kokemusta ohjelmistojen tuottamisesta, sekä hankittu kirjallista eri lähteistä saatua tietoa. Tavoitteena myös luoda tietopaketti, jota myös jokin muu yritys voisi käyttää apuvälineenä ryhtyessään hankkimaan ohjelmistoa alihankintana. Erityisesti tilanteessa, jossa tilaaja osallistuu vahvasti määrittelyyn ja suunnitteluun. Ohjelmiston tuotantoa tarkastellaan käytännön läheisesti ja mahdollisimman ymmärrettävästi, jotta tästä työstä olisi hyötyä myös henkilöille, jotka eivät ole aiemmin ohjelmistoprojektien kanssa olleet tekemisissä. Lyhenteet ja IT-slangi ovat erityinen vitsaus ymmärrettävyyttä ajatellen. Tästä syystä mahdollimman suuri osa näistä käsitellään ja selvennetään samalla, kun tulevat työn aikana esille.

Iso osa kaikista ohjelmisto- ja IT-alaan liittyvistä kehitystyöstä tehdään eri asteisina projekteina. Tässä työssä käsitellään myös projektin hallintaa ja projektityöskentelyä, erityisesti ohjelmistokehitykselle ominaisilla vivahteilla.

1 OHJELMISTON HANKINNAN LÄHTÖKOHDAT

Ohjelmistoprojektiin lähdettäessä tulee selvitää projektin liiketoiminnallinen tavoite ja taloudelliset lähtökohdat, joiden puitteissa projekti voidaan toteuttaa.

1.1 Liiketoiminnallinen tavoite

Ohjelmistohankinnan projekti ja sen liiketoiminnalinen suunnittelu lähtee tarpeesta. Tarve voi olla saanut alkunsa tuotekehityksestä, sisäisten prosessien kehityksestä, vanhojen ohjelmistojen korvaamisesta tai muista lähtökohdista, jonka ratkaisemiseksi on tarve tuottaa uusi ohjelmisto tai sen osa. Heti alkuvaiheessa pitää löytää liikentoiminnallinen peruste hankinnan tekemiselle. Hankinta voi olla perusteltavissa myös liiketoimintaa tukevana, vaikka sille ei suoraa euromääräistä tuottoa voida tarkkaan laskea.

Ohjelmistotuotannon tai hankinnan aloittamiseen on oltava aina liiketoiminnallinen tavoite. Sen on toimittava liikentoiminnan kehityksen ja yrityksen toimintaa tukevana. Ohjelmistot on myös nähtävä kokonaishankkeena, jolla on elinkaari. Hanke lähtee alkuideasta ja päättyy lopulta tuotantokäytöstä poistoon. (Haikala & Mikkonen 2011, 19)

Ohjelmiston tuottamista on käsiteltävä kuten mitä tahansa investointia. Arvioitava on yhtälailla hankintahinta, kuin elinkaarikustannuksetkin. Usein keskityttään ainoastaan hankintahintaan, jolloin ylläpitokustannustannuksiin ei osata varautua, tai sitä ei oteta riittävästi huomioon ohjelmistokehityksen aikana. Kehitysvaiheessa tehdyt valinnat vaikuttavat siihen, mitä kokonaisuuden ylläpito tulevaisuudessa tulee maksamaan.

Liiketoiminnan kannalta on tärkeää tietää paljonko alkuinvestoinnin jälkeen järjestelmän ylläpitoon on varattava resursseja, niin rahallista kuin yrityksen sisäistä työpanostakin.

Perinteisessä ostoprojektissa hankinnat jaetaan kolmeen pääryhmään, hankintaa edeltävät kustannukset, hankinnan aikaiset kustannukset ja hankinnan jälkeiset kustannukset. Perinteisen hankinnan tapauksissa pidetään yleisenä, että hankinnan aikaiset kustannukset ovat vain pieni osa kokonaishankintaa. (Huuhka 2017, 32)

Tätä perinteisen hankinnan määritelmää voidaan hyvin soveltaa myös ohjelmistohankinnoissa. Erityisesti, jos ohjelmistosta suuri osa ostetaan lisenssillä olemassa olevasta

ohjelmistosta on hankinnan aikaiset kustannukset pieniä, jolloin edeltävät ja jälkeiset kustannukset voivat olla määräääviä.

Valmiiden ohjelmistojen ostossa nykysuuntaus on suuren alkuinvestoinnin sijaan siirtyminen kuukausihinnoitteluun tai suoraan käytön laskutukseen. Esimerkkinä käytön laskutuksesta voidaan mainita vaikka useat taloushallinto-ohjelmistot ja kuukausihinnoittelusta vaikka kaikkien tuntema Microsoftin ohjelmistopaketti, Office 365. Jos taas ohjelmistosta isompi osa teetetään ohjelmistokehitystyönä, suurempi osa kustannuksista tulee jo hankintavaiheessa.

Tekniset valinnat vaikuttavat osaltaan syntyviin kustannuksiin, jolloin valintaan on hyvä käyttää tarvittaessa ulkoista apua, jotta varmasti kaikki eri vaiheista syntyvät kustannukset tulee huomioitua. Ennen teknisten valintojen tekoa ei välttämättä voida tietää ohjelmistojen elinkaarikustannuksia edes riittävällä tasolla.

Yhtenä hankinnan apukeinona voidaan käyttää julkisorganisaatiosta tuttua pisteytysperiaatetta. Pisteytysperiaatteen avulla saadaan helposti ja systemaattisesti vertailtua vaihtoehtoja keskenään, jolloin tunteisiin ja mielikuviin perustuva päätöksenteko ei aiheita vikavalintoja. (Alfame Oy 2015, 7)

Digitaalisen maailman erityispiirteenä taloudellisen kannattavuuden kannalta voidaan pitää sitä, että kerran tehtyä tuotetta voidaan helposti kopioida ja myydä aiheuttamatta parhaimmillaan yhtään lisäkustannuksia. Tilanteessa, jossa tuotettua ohjelmistoa myydään eteenpäin tuotteena, kaikki lisämyynti voidaan tavallaan laskea suoraan katteeksi. Toisaalta jos myydään alle suunnitellun ja oletetun määrän, niin kate jää kokonaan saavuttamatta.

1.2 Taloudelliset lähtokohdat ja budjetti

Ohjelmiston hankkiminen on aina oltava perusteltavissa taloudellisilla lähtökohdilla, joten hankinnan kannattavuuden kannalta laskenta pitäisi aina tehdä euromääräisenä. Vain euromääräsellä arvioinnilla voidaan perustella hankinnan taloudellinen kannattavuus. Esimerkkinä voidaan ajatella vaikka ohjelmiston käytön aiheuttama henkilötuntien säästö. Määriteltäessä ohjelmiston tuottama arvo euroissa, on myös ratkaisujen arviointi helpompaa. (Alfame Oy 2015, 4)

Projektin alkuvaiheessa on hyvä kartoittaa valmiita tai melkein valmiita ohjelmistoja, joista saadaan useimmin kustannustehokkain ratkaisu. Erityisen hyödyllistä on tutkia avoimeen lähdekoodiin perustuvat ratkaisut. Avoimen lähdekoodin ratkaisuhin on myös helpointa saada teetettyä lisätoimintoja, jotta ohjelmisto vastaa haluttua lopputulosta. (Alfame Oy 2015, 8)

Avoimen lähdekoodien ohjemistoilla tarkoitetaan yksinkertaisetettuna sellaisia ohjelmia, joita saa vaapasti levittää ja muuttaa, eikä niistä tarvitse erikseen maksaa. Tarkempi määritys tällaisille ohjelmille sisältää muutamia kohtia, joiden täyttyessä ohjelmaa voidaan kutsua avoimen lähdekoodin ohjelmaksi.

Avoimen lähdekoodiin perustuvan ohjelmiston määritelmän mukaiset ominaisuudet ja tekniset vaatimukset ovat:

- **Vapaa levitettävyys.** Ohjelmaa saa levittää ja kopioida ilman siitä perittävää maksua tai lisenssiä.
- **Lähdekoodi.** Ohjelmasta on oltava saatavilla ohjelman lähdekoodi. Lähdekoodi mahdollistaa muutosten tekemisen ohjemistoon. Myös ohjelman lähdekoodi pitää olla saatavilla ilman maksua.
- Ohjelmasta jatkokehitetyt versiot ja niiden levittämienen pitää sallia.
- Ohjelman käyttötarkoitusta ei saa rajoittaa.
- Avoimen lähdekoodin ohjelmaa voi käyttää myös osana kokonaisuuttta, joka sisältää ei avoimia ohjelmia.

(Open Source Initiative, 2021)

Ylläpitokustannuksia aiheuttaa esimerkiksi ohjelmistojen osana käytetyt ulkoiset palvelut, esimerkkinä asiakkaan vahva tunnistaminen ja maksunvälityspalvelut. Ohjelmistojen taustajärjestelmissä ja hallintasovelluksissa käytetään myös usein maksullisia osia tai työkaluja, esimerkkinä voidaan mainita muun muassa valmiit sisällönhallinta (CMS, Content Management System) työkalut tai valmiit rajapintaliitynnät vaikka postitusjärjestelmiin. Lisäksi säännöllisiä käytönaikaisia kustannuksia syntyy palvelimista, jossa palvelua tarjotaan, jos kyseessä on muu kuin paikallisesti asennettava ohjelma.

Isolle yleisölle tarjottavassa julkisesti näkyvässä palvelussa pitää erityisesti kiinnittää huomiota palvelinympäristön tietoturvallisuuteen ja kapasitteettiin. Palvelinten ylläpito ja palvelimet yleisesti on huomattava kustannus käytönaikaisista kustannuksista ja huomioitava erityisesti.

Taloudellisten vaikutusten arvioinnissa on myös tarpeellista arvioida tuotantoon ja käyttöönottoon liittyvät piilokustannukset, sekä myös piilohyödyt. Piilokustannuksina voidaan ajatella muun muassa koulutuskustannukset ja siirtymävaiheen lisätyöt. (Alfame Oy 2015, 9)

2 VAATIMUSTEN MÄÄRITTELY

Vaatimusmäärittelyssä kartoitetaan ja dokumentoidaan tuotettavan ohjelmiston ominaisuuksia ja käytön kannalta oleellisia asioita. Vaatimusmäärittelyn erityisessä keskiössä on tulevat ohjelmiston asiakkaat ja käyttäjät. Vaatimusmäärittely voi myös sisältää tuotantoon ja testaukseen liittyvää suunnittelua. Vaatimusmäärittely on tärkein yksittäinen dokumentti ohjelmistoprojektissa ja erityisesti onnistuneen lopputuloksen saavuttamisessa. Tilaaja voi aktiivisesti osallistua tähän vaiheeseen, jos tilaajalla ei ole tähän mahdollista osallistua on ohjelmistokehittäjien ja suunnittelijoiden tutustuttava toimialaan ja liikentoimintaan erityisen tarkasti. Tämä pitää tällöin huomioida myös kustannuksissa.

Vaatimusmäärittelyä tehdään yhdessä tilaajan ja ohjelmiston kehittäjien kanssa. Tekijöiden on oltava hyvin perillä tuottevan ohjelmiston käyttöympäristöstä, käyttäjistä ja muista tarpeista. Vaatimusmäärittelyn runko on sama sekä vesiputousmallissa, että ketterän kehityksen projektimallissa. Vesiputousmallissa määrittely ojn kuitenkin oltava pääosin valmis ennen projektin alkua, kun taas ketterän mallin mukaan määritelyä voi päivittää projektin edetessä. Jos määrittelyä muutetaan kesken projektin, on tärkeää huomioida sen vaikutukset sopimuksiin ja kustannuksiin. (Alfame Oy n.d., 7)

Vaatimusmäärittely voi olla valmisteluvaiheen suuritöisin työtehtävä. Siihen olisi hyvä osallistua ydinliiketoiminnan ja käyttöympäristön hyvin tuntevia henkilöitä eri aloilta. Pienessä yhtiössä tähän työhön olisi hyvä ottaa mukaan kaikki, jotka ohjelmiston kanssa ovat tulevaisuudessa tekemisissä. Ilman laajamittaista vaatimusten ja käyttötarpeiden listausta ja suunnittelua, saattaa jokin merkittävä toiminto jäädä liian pienelle huomiolle tai jopa kokonaan tekemättä. Vaatimusmäärittelyä ei myöskään saa tehdä liian yksityiskohtaisesti, koska se saattaa sulkea kokonaan pois parempia vaihtoehtoja, jolla haluttu toiminto olisi mahdollista tehdä paremmin ja joustavammin. (Forselius 2013, 29-30)

Määrittelydokumentissa on hyvä käyttää selkeää, ymmärrettävää tekstiä sekä selventäviä kuvia ja kaavioita. Dokumentin on oltava kaikkien ymmärrettävissä ainakin pääpiirteittäin ja on toisaalta liittää myös osaksi tilaajan ja tuottajan välistä sopimusta.

Määrittelyvaiheessa yksi tapa kerätä tarvittavia ominaisuuksia ja ymmärtää eri toimintojen tärkeys ovat käyttötarinat. Käyttötarinoita voi kerätä aivan normaaleilta ohjelmiston käyttäjiltä, jossa he kertovat miten he ohjelmaa käyttävät tai mihin he tulevaa ohjelmaa tulisivat käyttämään. Käyttötarinat myös osaltaan selventävät käyttäjien erilaisia

työtapoja. Työtapojen tunnistamisella voidaan myös helpottaa myöhemmin tehtävien käyttöohjeiden ja ohjeistusten tekoa. Näistä voidaan myös poimia käytettyä alakohtaista terminologiaa sekä hyödyntää testaussunnitelman tekemisessä. Käyttötarinoita on hyvä kerätä myös ohjelman valmistumisen ja käyttöönoton jälkeen. Näistä voidaan päätellä onko ominaisuus tehty oikein vai kaivataanko siihen vielä parannuksia tai muuta jatkokehtystä. (Forselius 2013, 32-33)

Vaatimusmäärittelystä vastaa loppukädessä tietojärjestelmän omistaja. Vaatimusmäärittelyprojektista vastaa luontevimmin projektipäällikkö, mutta määrittelyyn osallistuu myös vaatimuksien esittäjiä sekä muita asiantuntijoita. Mitä kokonaisvaltaisemmin määrittelyyn osallistuvat henkilöt tuntevat käyttöympäristön ja järjestelmään liittyvät osa-alueet sen parempi lopputulos. (Suomidigi 2018, 16)

2.1 Vaatimusten ryhmittely ja määritysdokumentin teko

Vaatimukset on hyvä tehdä riittävän tarkaksi, mutta ei liiasn tarkaksi. Tarkuudeltaan kuitenkin keskenään yhdenmukaisiksi. On parempi mielummin pilkkoa vaatimukset pienempiin kokonaisuuksiin kuin tehdä liian isoja kokonaisuuksia. Tämä autaa myös itse valmistusvaiheessa ja työn etenemisen seurannassa. Vaatimuksia voidaan selvyyden vuoksi kuitenkin ryhmitellä eri otsikoiden alle, jos ne kuuluvat johonkin yhteneväiseen kokonaisuuten. Vaatimusmäärittely tehdään aina kirjallisena dokumenttina ja sisältää esimerkisi seuraavia kohtia.

Lähtökohdat, käyttäjät ja tarpeet

Listataan ja analysoidaan ympäristö, johon ohjelmisto tuotetaan. Tunnistetaan asiakastyypit ja muut käyttäjät, jotka ohjelmistoa tulevat käyttämään. Yritetään tunnistaa myös tyypillisten käyttäjien profiili ja sen vaikutukset esimerkikisi käyttöliittymään. Selvitetään myös toivotut vaikutukset liiketoimintaan ja saada selville mitä toivotaan tapahtuvan helpommin ja nopeammin (Alfame Oy n.d., 10).

Listataan ja selvitetään muut sidosryhmät, jotka vaikuttavat ohjelmiston käyttöympäritöön. Arvioidaan käyttäjämääriä, käyttäjien sijainti ja laiteympäristö. Käyttäjistä saa usein parhaimman käsityksen tapaamalla ja haastattelemalla käyttäjiä. Haastattelut pitää käydä kansankielisesti, sekä ohjelmistosuunnittelijan, että käyttäjän pitäisi puhua termeistä ymmärrettävästi. Eri alojen ammattisanastot saattavat aiheuttaa kommunikaation väärinymmärryksiä. Kummankin osapuolen käyttämät termit tulee selventää, jos niissä

on epäselvyyksiä. Dokumentiin on hyvä myös kerätä alakohtainen sanasto, jotta ohjelmistoissa osataan käyttää oikeita termejä. (Alfame Oy n.d., 10, 12, 14)

Asiakaspolku

Asiakaspolkua suunniteltaessa, se piirretään graafisesti ja selostetaan tyypillinen asiakkaan tai käyttäjän kulkema polku ohjelmistossa. Asiakaspolut esitetään helpoiten vuokaaviona. On hyvä suunnitella ja piirtää useita asiakaspolkuja eri käyttötilanteista. Asiakaspolun ei tarvitse olla kovin yksityiskohtainen, se kuitenkin helpottaa tarkempia teknisisä suunnitelmia ja tuotantoa hahmottamaan tarvetta, johon ohjelmaa ollaan tekemässä.

Käyttötapaukset

Selostetaan ja analysoidaan tyypillisiä käyttötapauksia eli käyttötilanteita, jonka vuoksi ohjelmaa käytetään. Otetaan malliesimerkkejä tyypillisistä tapauksista, miksi ja miten käyttäjä ohjelmistoa käyttää ja mitä ohjelmiston käytöllä halutaan saavuttaa. Käyttötapauset esitetään tyypillisesti kaaviona, mutta voidaan esittää myös tekstimuotoisena. Esitettäessä käyttötapausta, myös tapaukseen liittyvä käyttäjärooli on hyvä tuoda esille.

Toiminnalliset- ja ei toiminnalliset vaatimukset

Vaatimukset voidaan jakaa kolmeen eri ryhmää.

- **Toiminnalliset vaatimukset**
- **Ei-toiminnalliset vaatimukset**
- **Reunaehdot**

(Haikala & Mikkonen 2011, 61)

Vaatimuksien ymmärtämiseksi, **toiminnallinen vaatimus** voi olla esimerkiksi sitä, että ohjelmistolla on voitava lähettää asiakkaalle SMS viestejä tai ohjelmassa on oltava mahdollisuus tuoda sisään asiakastietoja jostain toisesta ohjelmasta. Toiminnallisia vaatimuksia on kaikki timinnot, joita tulevalla ohjemistolla pystyy tekemään. **Ei-toiminnallisten** vaatimuksien esimerkkinä voidaan pitää esimerkiksi sitä, että ohjelmiston on oltava helppokäyttöinen tai ohjelmiston ulkoasu oltava yrityksen brändin mukainen. Ei-toiminnallisia vaatimuksia ovat myös tietoturva ja saatavuus. **Reunaehdot** tarkoittavat esimerkiksi, että ohjelmisto tehdään Android ja IPhone mobiililaitteisiin tai että ohjelma on

toteutettava käyttäen MySQL tietokantapalvelinta. Reunaehdoissa mainitaan usein myös käytetty ohjelmointikieli.

Toiminnallisia ja ei-toiminnallisia vaatimuksia esitetään usein listana, jaoteltuna ohjelman eri kokonaisuuksiin. Vaatimukset on myös hyvä numeroida selkeästi. Numeroinnista voi saada vielä lisähyötyä esimerkiksi aloittamalla eri ryhmän tai osa-alueen vaatimukset eri alkuisilla numeroilla. Esimerkiksi numerolla 1 alkavat ovat käyttöliittymävaatimuksia, numerolla 2 alkavat ovat arkkitehtuurivaatimuksia, numerolla 3 alkavat ovat tietoturvaan liittyviä ja niin edelleen. Vaatimusten numerointia käytetään yksilöivänä tunnisteena koko projektissa, jolloin on pidettävä huolta, että numerointi on uniikkina koko projektissa.

Liitynnät muihin järjestelmiin

Liittymät muihin olemassa oleviin ja tulevaiduudessa kehitettäviin järjestelmiin voidaan kuvata kokonaan omana kohtanaan määritysdokumentissa. Tästä on hyvä ilmetä tiedon siirron suunta sekä tiedon sisältö. Tähän kappaleeseen kuuluvat myös eri rajapintojen kuvaukset sekä käyttötarkoitus ja käyttötilanteet, jolloin tietoa siirretään. Myös tiedot siirtojen tiheys ja datamäärä on hyvä tulla esille, se saataa vaikuttaa suunnitteluvaiheessa teknisien ratkaisujen valinnassa. Kaikki liitynnät sekä sisäiset ja ulkoiset järjestelmät on hyvä piirtää samaan kuvaan, jotta kokonaisuus on helpompi hahmottaa. (Suomidigi 2018, 25)

Käyttäjäroolit

Ohjelmiston käytön kannalta keskeinen toiminnallisuus on käyttäjäroolit. Rooleilla tarkoitetaan yksinkertaisimmillaan eri käyttöoikeustasoja ohjelmistossa. Usein käyttäjärooleja on huomattavasti enemmän eri tarkoituksiin. Myös kytkettävillä ulkosilla tietojärjestelmillä on oma käyttäjäroolinsa. Yleisimmät roolit ovat pääkäyttäjä sekä peruskäyttäjä. Määritysdokumentissa on hyvä listata kaikki tulevat roolit, sekä niille annettavat käyttöoikeustasot. Käyttöoikeustasolle voidaan määritellä jo suunnitteluvaiheesa myös tarkemmin mihin osiin ohjelmistoa kukin käyttötaso pääsee katsomaan, muuttamaan, poistamaan ja niin edelleen. (Suomidigi 2018, 25)

Ohjelmistoympäristön arkkitehtuurit

Ohjelmistoarkkitehtuurin määrittää pääosin ohjelmistokehittäjät ja -suunnittelijat aiemmin listattujen vaatimuksien perusteella. Ohjelmistoarkkitehtuuri tarkoittaa ohjelmiston rakennetta ja se voidaan esittää eri tavoin. Yksinkertaistettuna ohjelmistoarkkitehtuurissa esitetään ohjelmiston toiminnaliset osat ja niiden väliset suhteet.

Lisäksi on hyvä kuvata myös tietojärjestelmän arkkitehtuuri, joka sisältää ohjelmapalvelimet, tietokantapalvelimet, ulkoiset palvelut joiden raameissa toimitaan. Tietojärjestelmä voi myös olla olemassa olevan yrityksen järjestelmä, jolloin myös se on hyvä mainita ja rakenne varmentaa.

Arkkitehtuuri voidaan myös ajatella teknologia-arkkitehtuurina, jolloin otetaan kantaa käytettyihin teknogioihin ja standardeihin ja muihin teknologisiin ratkaisuihin.

Käyttöliittymän MockUp

Määrittelyyn on hyvä jo alustavasti suunnitella ja luonnostella käyttöliittymän perusteita. Tämä helpottaa ymmärtämään miten ohjelmistoa käytetään ja mitä kaikkea toiminnalista pitää huomioida. Usein käyttöliittymän suunnittelu jo varhaisessa vaiheessa auttaa hahmottamaan kokonaisuutta myös loppukäyttäjälle.

Testaus ja julkaisu

Vaatimusten määrittelyssä on hyvä ottaa myös kantaa testaukseen. Tarkempi testisuunnitelma tehdään erikseen, mutta tässä voidaan huomioida esimerkiksi erityistarpeet jonkin ohjelmiston osan testauksessa tai muuta huomioin arvoista jatkoa ajatellen.

Julkaisu voidaan tehdä osissa tai aluksi vain pienelle kohderyhmälle, tämä tai muita vastaavia julkaisuun liittyviä asioita voidaan myös mainita jo vaatimusmäärittelyssä.

2.2 Mallinnustavat

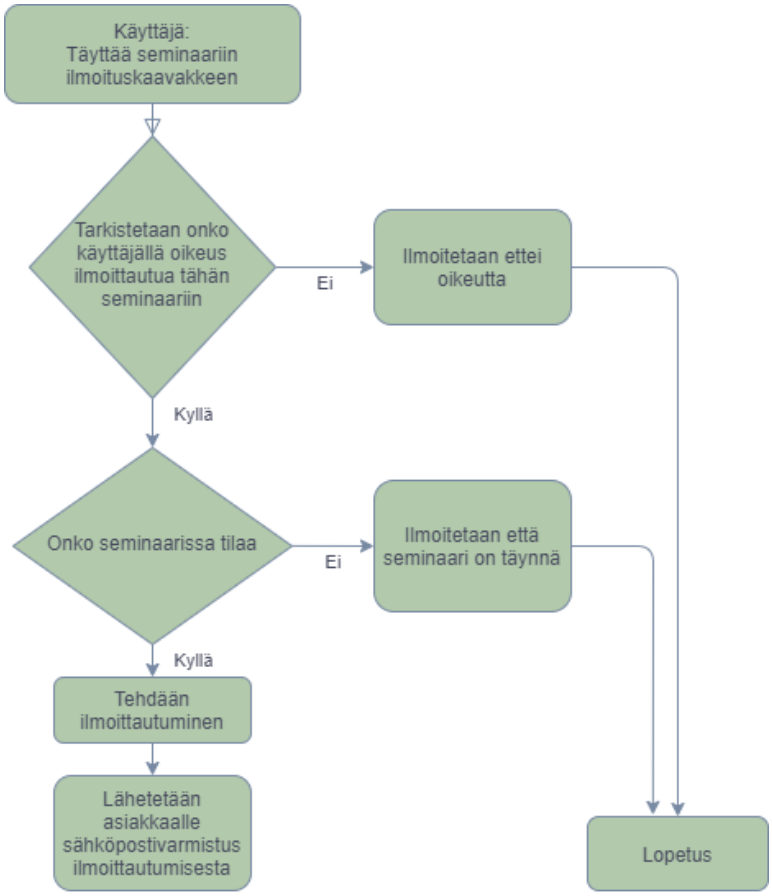
Vaatimusmäärittelyn on hyvä liittää konkreettisia esimerkkejä tuotettavan ohjelmiston toiminnasta. Näitä voi olla muun muassa käyttötapauksien kuvaukset, vuokaaviot, sekvenssikaaviot, näkymäkartat ja käyttötapauskaaviot. Tämänkin osuuden voi tehdä joko tilaaja tai toimittaja. Tilaajan on kuitenkin hyvä vähintään tutustua suunnitelmiin väärinymmärrysten välttämiseksi, vaikka tilaaja olisikin ulkoistanut suurimman osan suunnittelusta. Ohjelman konkreettista toimintaa voidaan esittää tekstinä, sekä kuvina ja kaavioina ja näiden yhdistelminä. Usein kaaviokuvat ovat havainnollisimpia ja tilaaja saa helposti oikean kuvan ohjelman toiminnasta. Mallinnustapoja on paljon, joista voi valita projektiin sopivimmat esitysmuodot. Mallinnustapoja valittaessa on huomioida kenelle se on suunnattu. Eri kohderyhmät odottavat ja tarvitsevat hyvin erilaisen esitystavan, riippuen

muun muassa tietotaidosta ja siitä mitä informaatiota kohderyhmä odottaa saavansa mallista.

Ohjelmien mallinnusta varten Object Management Group on kehittänyt ja stadardoinut Unified Modelling Language (UML) graafisen mallinnuskielen ohjelmistokehityksen tarpeisiin. Suuri osa on kohdistettu suoraan ohjelmistokehittäjille, mutta mukana on muutama yksinkertaisempi graafinen esitystapa, joista voi olla hyötyä ohjelmiston suunnittelussa ja mallinnuksessa myös kokemattomammalle.

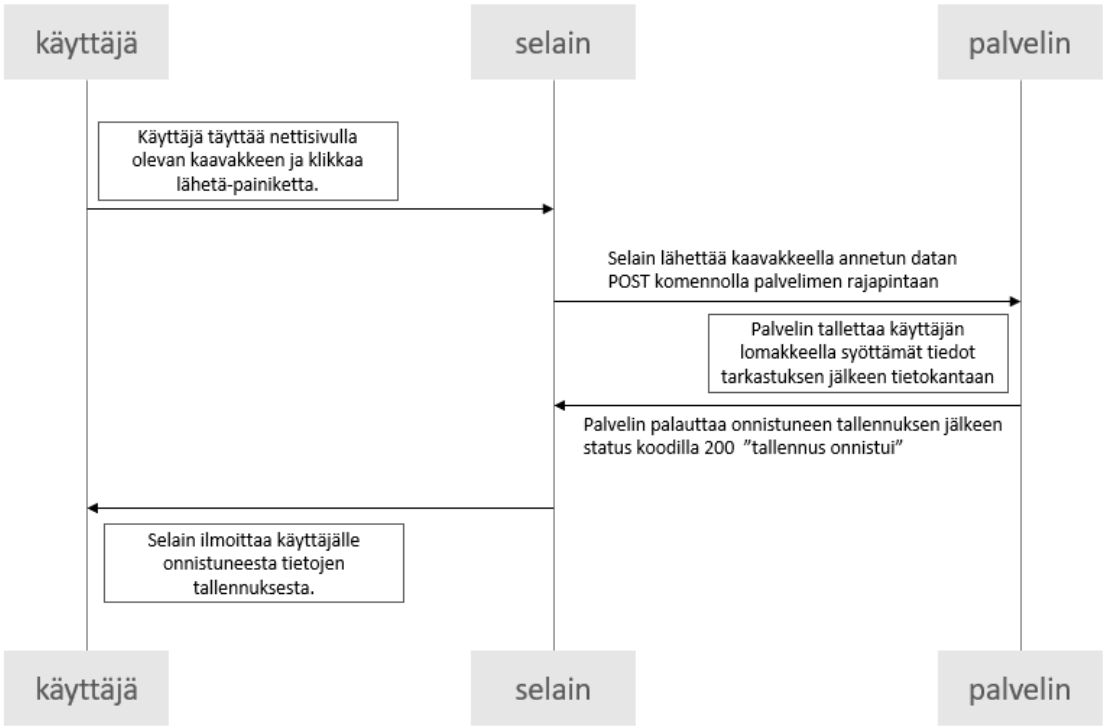
(Object Management Group 2017)

Vuokaaviolla voidaan havainnollisesti kuvata ohjelmiston toimintaprosesseja eri tinateissa. Vuokaavio etenee tapahtumajärjestyksessä vaiheittain. Vuokaaviossa on määritetyt muodot eri tapahtumille, kuten salmiakkikuvio tarkoittaa valintaa ja suorakulmio tarkoittaa prosessointia.



Kuva 1. Esimerkki vuokaaviosta

Sekvenssikaaviolla voidaan kuvata ohjelmistojen eri osien (olioiden) välistä vuorovaikutusta. Sekvenssikaaviosta käytetään myös nimitystä viestiyhteyskaavio. Sekvenssikaaviossa oliot ovat pystysuoria ja tapahtumat kulkevat poikittaisina nuolviivoina aikajärjestyksessä ylhäältä alas. Sekvenssikaavio on UML-standardin mukainen esitystapa.

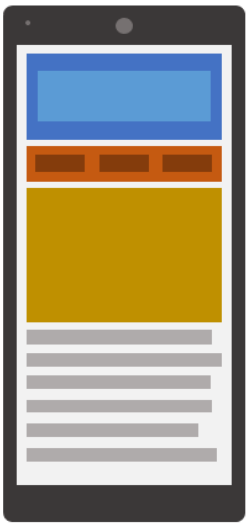


Kuva 2. Yksinkertaistettu esimerkki sekvenssikaaviosta

Näkymäkartalla havannollistetaan ohjelman käyttöliittymän ulkoasua ja eri toimintojen asettelua. Erityisen tärkeää käyttöliittymän ulkoasun suunnittelu on mobiililaitteilla, joissa tilaa on vähän ja laiteympäristöt ovat haastavia. Usein mobiilikäyttöliittymien asettelut suunnitellaan kokonaan ominaan, johtuen laitteiden näytön pienuudesta ja kosketusnäytöstä. Lisäksi mobiililaitteita käytetään yleensä pystyasennossa, jolloin käyttöliittymän sama asettelu ei toimi, kuten muissa vaaka-asennossa käytettävissä laitteissa. Näkymäkarttaan voidaan myös liittää suunnittelua graafisesta ulkoasusta.

Näkymäkartasta voidaan tehdä useampi malli eri pääteleaitteille, jolla ohjelmistoa käytetään. Kuvissa 3. ja 4. havainnolistetaan sitä eroa, joka pitää huomioida eri kokoisissa ja eri kuvasuhteella olevissa näytöissä.

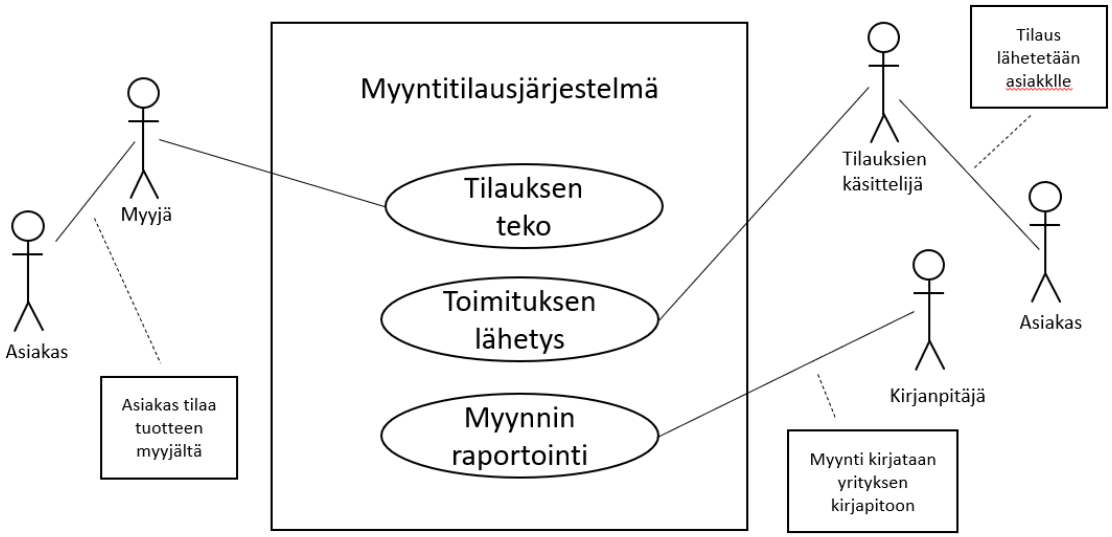
Kuva 3. Näkymäkartan haivainnekuva, vakaanäyttö esim. laptop tietokone



Kuva 4, Näkymäkartan havainnekuva, pystynäyttö esim. puhelin

Käyttötapauskaaviolla voidaan esittää havannollisesti miten ja miksi eri käyttäjäryhmät ohjelmistoa käyttävät. Yksinkertaisimmillaan käyttötapauskaaviossa on esitetty käyttäjäryhmät ja käyttötapaukset, joita kukin käyttäjäryhmä käyttää. Erikoisesta ulkoasustaan huolimatta käyttötapauskaavio on UML-standardin mukainen esitystapa.

(Object Management Group 2017, 639-646)



Kuva 5. Yksinkertainen esimerkki käyttötapauskaaviosta

Vaatimusluettelolla yksinkertaisimmillaan listataan ja numeroidaan vaatimukset. Vaatimusluettelosta on hyvä löytyä ainakin seuraavat sarakkeet:

- **Vaatimuksen tunnistetieto (ID).** Yksilöllinen numero kullekin vaatimukselle. Yksilöllisellä numeroinnilla voidaan muutokset ja työn edistymisen seuranta voidaan kohdistaa juuri oikeaan vaatimukseen.
- **Vaatimuksen esittäjä.** Vaatimuksen esittäjä on taho, joka vaatimuksen on esittänyt. Esittäjä voi olla esimerkiksi käyttäjä, ulkoisen liityntäpinnan tarjoaja tai järjestelmäasiantuntija.
- **Vaatimuksen kriittisyys.** Kriittisyys voidaan esittää esimerkiksi kolmeportaisena. 1 = pakollinen, 2 = hyödyllinen, 3= toivottu.
- **Vaatimuksen perustelu.** Lyhyt kuvaus miksi vaatimus on kriittisyysasteikolla määritetty millekin tasolle tai miksi vaatimus on yleensäkään olemassa.

(Suomidigi 2018, 21)

ID	VAATIMUS	VAATIMUKSEN ESITTÄJÄ	KRIITTISYYS	PERUSTELU
1.1	Käyttäjän on voitava vaihtaa salasanansa	Kalle Käyttäjä	2	Käyttömukavuus ja turvallisuus
1.2	Asiakaspalvelun on voitava vaihtaa asiakkaan salasana	Asiakaspalvelu	3	Asiakaspalvelun sujuvuus ja turvallisuus
2.1	Pankkiyhteys on oltava SSL salattu	Pankkipalvelu	1	Turvallisuus ja pankin käytäntö

Kuva 6. Esimerkki yksinkertaisesta vaatimusluettelosta.

2.3 Tietoturva

Tietoturvalla on erityisen merkittävä rooli nykyaikaisessa ohjelmistokehityksessä. Tästä syystä se on syytä käsitellä kokonaan omana kappaleenaan vaikka kuuluu kuitenkin teknisesti ja toiminnallisesti kahteen aiempaan. Vaatimusmäärittelyn tulee sisältää myös tietoturvaan liittyvät vaatimukset. Tietoturva otetaan huomioon kaikissa määrittelyn vaiheissa ja vaikuttaa kaikkeen suunnitteluun sekä ohjelmiston käyttöön.

Tehokkain tapa parantaa tietoturvallisuutta on ottaa se huomioon jo ohjelmiston suunnittelu- ja kehitysvaiheessa. Näin saadaan tehokkaimmin ennaltaehkäistyä tietomurtoja. Tietoturvallisuusvaatimusten määrittely on vaikeampaa kuin toiminnallisten vaatimusten. Lisäksi onnistuneen tietoturvan toteaminen on vaikemapaa kuin toiminnallisten vaatimusten. Yksinkertaisimmillaan tietoturvamäärityksissä määritellään miten järjestelmän tietoja ja palveluja voidaan suojata ulkoisilta hyökkääjiltä ja muilta haitallisilta tunkeutujilta. Vaatimusmäärittelyssä määritetään myös käyttäjien tunnistustapa ja muita teknisiä suojausperiaatteita ja käytäntöjä. (Kyberturvallisuuskeskus, Traficom 2018. 14)

Tietoturvallisen suunnittelun perusta on ottaa turvallisuus huomioon jo heti projektin alkaessa. Yksi tärkeimmistä tietoturvan osa-alueista on tietosuoja, jolla tässä yhteydessä tarkoitetaan henkilötietojen suojaamista. Henkilötietojen suojaamiseen on tietosuojalakiin (GDPR) perustuvia vaatimuksia. Suojaamisen lisäksi on huomioitava henkilötietojen erityisen tarkka lainsäädäntö henkilötietojen käsittelyssä ja ilmoitusvelvollisuudessa poikkeamatilanteessa. Tietosuojalaki on hyvin laaja kokonaisuus, johon on syytä tarkkaan tutustua tai konsultoida asiantuntijoita, jos tuotettavassa ohjelmistossa käsitellään henkilötietoja. Henkilötiedoiksi luetaan kaikki tiedot, joilla henkilö voidaan tunnistaa suoraan tai välillisesti. Joitain poikkeuksia lukuunottamatta kaikki järjestelmät sisältävät henkilötietoja. Pelkästään jo järjestelmään kirjautumiseen tarvittavat tunnistetiedot synnyttävät helposti henkilötietorekisterin.

Turvallisen suunnittelun periaatteet helpottavat suunnitelua ja parantavat lopputulosta. Periaatteisiin kuuluu niin teknisiä vaatimuksia kuin käytännön toimintatapojakin. Alla listattuna muutamia yksinkertaistettuja kohtia, joita suunnittelussa on hyvä huomioida.

- **Hyökkäyspinnan minimoiminen.** Hyökkäyspinnoilla tarkoitetaan järjestelmän komponentteja tai muita osia, joihin on pääsy verkon kautta tai fyysisesti. Näiden tarpeettomien hyökkäyspintojen poisto vähentää ”ovia”, joista joku voi tulla sisään. Se voi käytännössä tarkoittaa esimerkiksi palvelimella olevia ylimääräisiä ohjelmia tai fyysisen tietokoneen käytössä olevaa USB porttia.
- **Turvalliset oletusasetukset.** Täytyy olettaa, että käyttäjä ei ole tietoturvan asiantuntija. Ohjelmistossa olevat asetukset asetetaan oletusarvoisesti riittävän turvalliseksi.
- **Syötteenkäsittely.** Syötteillä tarkoitetaan kaikkea käyttäjän tai toisen järjestelmän syöttämää tietoa. Syötteet on aina tarkistettava ja varmistuttava, että niiden kautta ei voi tehdä ei-toivottuja toimenpiteitä järjestelmässä.
- **Erilliset tehtävät.** Tehtävien erottelemisella tarkoitetaan kriittisten tapahtumien hajauttamista eri palvelimille tai järjestelmille. Yksinkertaisena esimerkkinä dataa sisältävä palvelin ei tee automaattisia varmuuskopioita, vaan varmuuskopiot hoitaa toinen erillinen palvelin. Tällöin toiseen kohdistunut hyökkäys ei vaaranna datan säilymistä.
- **Mahdollisimman suppeat käyttöoikeudet.** Sekä käyttäjille, että toisille järjestelmille annetaan mahdollisimman vähän käyttöoikeusksia, kuitenkin niin, että kaikki tarvittavat tehtävät saadaan suoritettua. Käyttöoikeuksien minimointi koskee myös järjestelmien sisäisten prosessien toimintaa.
- **Syväsuojaus.** Syväsuojauksen periaatteena on rakentaa järjestelmäympäristö siten, että vaikka hyökkääjä läpäiseen yhden suojakerroksen, kuten palomuurin, sillä ei ole kuitenkaan vielä pääsyä koko järjestelmään. Mitä kriittisempi toiminto tai data on kyseessä, sen useamman suojauksen läpi hyökkääjän pitää onnistua pääsemään.
- **Turvallisuus vikatilanteissa.** Vaikka vikatilanteet lähtökohtaisesti pyritään estämään, pitää niihin silti varautua. Fyysiset tai ohjelmalliset komponentit saattavat vioittua syystä tai toisesta. Näihin tilanteisiin pitää varautua ja suunnitella tapa, miten järjestelmä reagoi näissä tilanteissa.
- **Ei luoteta liikaa ulkoisiin palveluihin.** Järjestelmään on hyvin usein kytketty ulkoisen tarjoajan palveluja. Niiden taustalla olevaa henkilöstä ja ei voida hallita tai niiden kautta voi jopa tulla hyökkäyksiä. Yksi tapa varautua tähän on antaa

ulkoiselle järjestelmälle mahdollisimman suppeat käyttöoikeudet, kuten jo aiemmin mainittiin.

- **Vältetään salassapitoon perustuvaa turvallisuutta.** Salassapitoon perustuvalla turvallisuudella tarkoitetaan esimerkiksi salasanoja. Niin kauan kun salasana pysyy salaisena, järjestelmä on turvallinen. Tähän ei kuitenkaan voida aina luottaa, joten järjestelmässä olisi hyvä huomioida, ettei turvallisuus liikaa perustu siihen, että järjestelmän rakenne tai salaiset avaimet pysyvät salaisena. Lisäksi nämä salaisuudet on hyvä olla helposti vaihdettavia, jos ne kuitenkin pääsevät vuotamaan. Salassapitoon perustuvalla suojauksella tarkoitetaan myös lähdekoodin salassapitoa sekä järjestelmän sisäistä rakennetta.
- **Yksinkertainen järjestelmä.** Järjestelmän yksinkertaisuus helpottaa sen pitämisenä turvallisena. Yksinkertaisen järjestelmän turvallisuutta on myös helpompi arvioida. Yksinkertaisuudella saavutetaan myös kustannussäästöjä kehityksessä ja ylläpidossa. Myös mahdolliset tietoturvakorjaukset on helpompi suorittaa. Jos ohjelmistolle suoritetaan turvallisuustarkastus, on se huomattavasti halvempi suorittaa, jos järjestelmä on yksinkertainen ja on osoitettavissa helposti turvallisuudelle kriittiset komponentit.
- **Turvallisuusongelmien korjaaminen oikein.** Havaittaessa tietoturvahaavoittuvuus on se korjattava. Tietoturvallisuuskorjaukset on helpompi suorittaa, jos olemassa on hyvin suunniteltu kehitys- ja testaus- ja tuotantoonsiirtoprosessi. Korjauksissa pitää ottaa myös huomioon voiko muualla ohjelmistossa esiintyä sama haavoittuvuus.
- **Vain turvallisten ohjelmistokomponenttien valinta.** Ulkoisten järjestelmien lisäksi ohjelmakehityksessä käytetään usein ulkoisista lähteistä saatavia komponentteja, joko sellaisenaan tai pienillä muutoksilla. Näitä voi olla muun muassa tietokannat ja ohjelmakirjastot. Näistä on hyvä pitää komponenttiluetteloa versiotietoineen. Myös ulkoisiin komponentteihin voi tulla haavoittuvuuksia tai ne voivat vanhentua, jolloin ne pitää päivittää uusiin ja korjattuihin. Luettelon avulla ne on helpommin tunnistettavissa. Ennen kunkin komponentin integrointia on siihen tutustuttava erityisesti turvallisuuden kannalta.
  (Kyberturvallisuuskeskus, Traficom 2018. 19-25)

3 OHJELMISTOPROJEKTIN VAIHEET

Ohjelmistoprojektit sisältävät projektinhallinnallisesti pääosin samoja elementtejä kuin muidenkin alojen projektit. Ohjelmistoprojekteissa on kuitenkin erityispiirteitä, jotka on hyvä huomioida.

3.1 Projektin suunnittelu

Tyypillisesti ohjelmistotuotanto organisoidaan projektina. Tavoiteena on tuottaa asiakkaan määrittelemiin vaatimuksiin perustuva ohjelmisto. Asiakkaan eli tilaavan osapuolen tarve syntyy pääsääntöisesti liiketoiminnallisista tavoitteista. Toimittajan päämääränä on ratkaista asiakkaan ongelma tietoteknisesti. Kommunikointi asiakkaan ja toimittajan välillä on tärkeää, jotta ongelma tunnistetaan ja saadaan ratkaistuksi. (Haikala & Mikkonen 2011, 19)

Vaikka projektiin aktiivisesti osallistuvia voi olla vain muutamia henkilöitä ja projektin vaiheita tehdään usein päällekkäin ja sekalaisessa järjestyksessä, on kuitenkin hyvä tunnistaa missä projektin vaiheessa ollaan etenemässä ja mihin vaiheeseen kukin työtehtävä kuuluu. Tämä lisää projektin systemaattisuutta ja helpottaa hallintaa.

Ohjelmistokehitysprojektia valmistellessa on hyvä myös tunnistaa tilanne, jota voidaan kutsua korjausinvestoinniksi. Tällaisessa tilanteessa vanha ohjelmisto tai sen osa vaihdetaan vain siksi, koska se pitää siirtää alustalta toiselle tai nykyiseltä ohjelmalta loppuu tuki tai se vanhenee liiaksi. Tässä tilanteessa on hyvä käydä läpi alkuperäisen ohjelman päätarkoitus ja miten se voitaisiin tehdä entistä paremmin. Moni fyysinen prosessi ja toimintatapa on voinut mukautua sen mukaan mihin ohjelmisto on aiemmin antanut myöden. Korjausinvestoinnissa on mitä mainioin tilanne parantaa koko prosessia, jossa lähtokohtana on itse tehtävä suoritus, eikä niinkään mihin ohjelmisto pystyy. (Lehtimäki 2006, 8)

Tarkastellessa erityisesti pienten yritysten ohjelmistohankintaprojekteja, tarkoittaa se käytännössä ulkoistamista. Ulkoistaminen on yritysten normaali toimintatapa, joka on tarkoituksellinen valinta itse keskittyä ydinliiketoimintaan. Oman ydinliiketoiminnan ulkopuolelle jäävät tehtävät ulkoistetaan muille yrityksille. Usein sellaisille, jotka tekevät tätä nimenomaista työtä itse ydinliiketoimintanaan ja ovat hyviä ja tehokkaita niiden töiden

suorittamiseen. Ohjelmistokehitys on helposti niin kaukana omasta liiketoiminnasta, että se on luontevaa ulkoistaa ohjelmistokehtitysyrityksille. Yleinen viisaus on valita kumppaniksi oman yhtiön kokoisia kumppaneita, jos se vain on mahdollista.

Projektin aloituksessa on hyvä keskittyä projektisuunnitelman lisäksi myös projektin suunnittelemiseen. Nämä on hyvä tiedostaa eri asioiksi ja tehtäviksi, vaikka niissä samojen teemojen äärellä ollaankin. Hyvä projektisuunnitelmarunko pakottaa suunnittelemaan myös itse projektia.

Yksinkertainen tapa testata projektisuunnitelmaa on etsiä projektisuunnitelmasta vastaus näihin neljään kysymykseen. Jos johonkin näistä kysymyksistä ei saada tyydyttävää vastausta on suunnitelma ainakin osin puutteellinen. Voidaan tällöin ajatella, että vaikka projektisuunnitelma on tehty, itse projektia ei ole suunniteltu.

- **Mitkä ovat tämän projektin tehtävät?**
- **Kuka nämä tehtävät tulee tekemään?**
- **Milloin tehtävät tehdään ja missä järjestyksessä?**
- **Mitä tehdyistä tehtävistä syntyy lopputuloksena ja miten se edistää koko projektin valmistumista?**

(Lehtimäki 2006, 14)

Projektia suunniteltaessa käydään läpi myös yrityksessä jo ennalta olemassa oleva materiaali, joka voisi mahdollisesti liittyä käynnistyvään projektiin. Aiemmin samassa ympäristössä kohdattuihin ongelmiin ja niiden ratkaisuihin saattaa hyvinkin löytyä dokumentaatiota ja valmiita ratkaisuja, jotka auttavat myös tämän ohjelmiston tuotannossa tai projektin onnistumisessa. Tähänkin liittyy vanha viisaus, jonka mukaan samoja virheitä ei pidä tehdä kahdesti.

3.2 Projektityöskentely ja -organisaatio

Pienen yrityksen kohdalla resurssit näyttelevät suurta roolia, erityisesti suorat rahalliset kustannukset. Projektin venyminen tai kustannusten hallitsematon kasvu saattaa aiheuttaa ylitsepääsemättömiä ongelmia. Aikataulutukseen on tällöin hyvä liittää myös kustannuksiin liittyvää ennakointia, jotta rahoituksessa ei tule yllätyksiä.

Perinteisesti projekteille valitaan työryhmä, johon kuuluu hankkeen omistaja, projektipäällikkö ja johtoryhmä. Projektin omistaja on lähtökohtaisesti liiketoimintavastuussa

oleva yrityksen johtaja. Projektipäällikkö hoitaa projektin käytännön seurannasta ja ohjauksesta. Johtoryhmä koostuu henkilöistä, joilla on riittävä asiantuntemus alasta, jotta voivat toimia projektin koordinoijina. (Haikala & Mikkonen 2011, 33)

Pienen yrityksen projekteissa harvoin voidaan nimetä henkilöitä näiden tehtävien hoitamiseen. Projektityöryhmä voi useissa tapauksissa koostua jopa vain yhdestä henkilöstä, joka hoitaa kaikkien tyypillisten projektityöryhmän jäsenten toimenkuvat. Näissä tapauksissa voi olla tehokkaampaa hankkia asiaantunteva projektipäällikkö oman organisaation ulkopuolelta hanketta varten, jotta työmäärä tilaajalle ei kasva liian suureksi. Tilaajalla itsellään on kuitenkin oltava tarkka käsitys siitä, millaista ohjelmistoa ollaan teettämässä ja mitä toiminnallisuuksia tarvitaan.

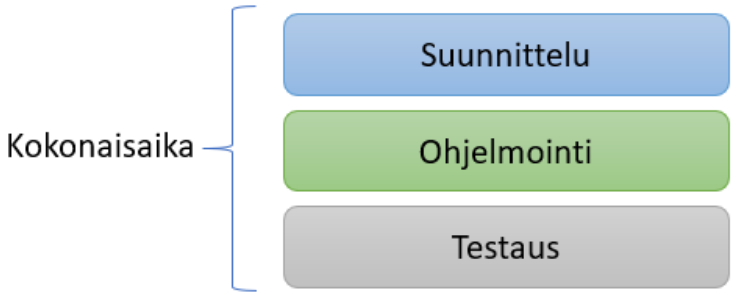
Projektipäällikön lisäksi tärkeitä rooleja ohjelmiston tuottamisessa ovat tekninen pääarkkitehti ja toiminnallisuuksista vastaava suunnittelija. Pienessä projektissa näistä toinen voi olla projektipäällikkö tai toisaalta projektipäällikkö voi olla kumpikin näistä. Näiden roolien takana olevien henkilöiden on ymmärrettävä syntyvän järjestelmän rakenne sekä toiminnallinen laajuus. (Lehtimäki 2006, 15)

3.3 Aikataulutus

Ohjelmistoprojektille, kuten mille tahansa projektille, luodaan aikataulu. Aikataulutuksessa on hyvä huomioida eri vaiheiden resurssitarpeita, toteutustapoja ja riskejä. Aikataulua on hyvä päivittää säännöllisesti projektin edetessä. (Haikala & Mikkonen 2011, 19)

Luotaessa ohjelmistoprojektin aikataulua on syytä tehdä se yhdessä ohjelmistokehittäjien ja muun tuotantoon osallistujien kanssa. Kuten muissakin projekteissa, mitä paremmin ja pidemmälle suunnitelmat on tehty sen tarkempi aikataulu saadaan tehtyä. Kokemuksesta voidaan sanoa, että ohjemistojen tuotannossa tilaaja helposti arvioi työn määrän väärin. Joidenkin ominaisuuksien kohdalla tilaaja olettaa sen vaativan vain vähän työtä, mutta todellisuudessa työtä on paljon. Toisaalta taas joidenkin toimintojen rakentamisen oletetaan vaativan paljonkin työtä, mutta todellisuudessa työ on melko pieni ja nopea. Aikataulu keskeinen osa on työmääräarvio. Todellisen työmäärän arvion voi parhaiten tehdä työn toteuttaja tai suunnittelija, jolla on selkeä käsitys siitä miten eri toiminnot käytännössä tehdään. Toinen osuus on suunnitella ja luoda aikataulu miten työmäärä sovitetaan koko projekti aikatauluun ja muiden tuotantoprojektien joukkoon.

Aikataulullisesti yksinkertaisetettuna ohjelmistontuotantoprosessi voidaan käsitellä kolmena eri kokonaisuutena, suunnittelu, ohjelmointi ja testaus. Tätä kutsutaan kolmasosien säännöksi. Tässä säännössä jokaista tehtyä koodinkirjoitustuntia kohden tehdään tunti suunnittelua sekä tunti testausta ja varmistusta. Kolmasosien sääntö kertoo havannollisesti sen, että jos kesken projektin tuodaan tehtäväksi lisää ominaisuuksia tai vaatimuksia, pelkkä ohjelmistokehittäjän lisäaika ei riitä. Samalla pitää myös suunnitteluun ja testaukseen tuoda sama määrä työresurssia, jotta aikataulussa on edelleen mahdollista pysyä. (Berkun 2006, 32-34)



Kuva 7. Yksinkertaistettuna aikataulun kolmasosien sääntö (Berkun 2006, 32-34)

Suuremmassa projektissa aikataulu on hyvä paloitella pienempiin osiin tai vaiheisiin, jolloin näitä kolmasosien sääntöön perustuvia aikataulutuksia tehdään useampi. Nämä vaiheet sisältävät kukin suunnittelua, ohjelmointia ja testausta. Lisäksi useampaan vaiheeseen jaettuun ohjelmistoprojektin alkuun on syytä varata alkuvaiheen suunnittelua, jolloin tutustutaan esimerkiksi asiakkaisiin ja liiketoimintaympäristöön. (Berkun 2006, 36-37)

Kun projektin aikautaulua pilkotaan työtehtäviin, sopivan tarkkuuden katsotaan olevan 40 tuntia. Eli työaikana noin yksi viikko. Suuremmat tehtävä on hyvä jakaa paloiksi. Pienempiin osiin jaettuja tehtäviä on helpompi ohjata ja seurata.

3.4 Projektin seurannan työvälineet

Projektinhallintaan on saatavilla suuri määrä erilaisia ohjelmistoja ja työkaluja. Yksikertaisimmillaan ja hyvin pienissä projekteissa seurantaa voi tehdä ihan perus taulukkolaskentaohjelmalla, mutta pääosin projekteissa vaatimukset kasvavat ja on hyvä ottaa käyttöön jokin siihen suunniteltu ohjelmisto.

Vaikka projekti on usein kertaluontoinen, sopivaan projektinhallintaohjelmiston valintaan kannattaa kiinnittää huomioita. Projektinhallintaohjelmistot sisältävät esimerkiksi ominaisuuksia projektin seurantaan ja aikatauluun, dokumenttien säilytykseen sekä projektin sisäiseen ja ulkoiseen viestintään. Yksi tärkeimmistä projektinhallintaohjelmistojen ominaisuuksista on pitää kaikki tieto samassa paikassa, josta kaikki niitä tarvitsevat pääsevät niitä tarkastelemaan. Näissä ohjelmistoissa on usein myös toiiminnallisuuksia jakaa dokumentteja projektiorganisaation ulkopuolelle. (Agendium Ltd n.d.)

Useimmat nykyiset projektinhallintaohjelmistot ovat pilvipalveluita, jotka jo luonnostaan mahdollistavat helpon pääsyn ohjelmaan mistä tahansa. Projektinhallintaohjemiston valinnassa on hyvä kiinnittää huomiota tarvittaviin ominaisuuksiin, jottei makseta turhista ominaisuuksista. Pienen kokoluokan projekteissa harvoin tarvitaan budetointityökaluja tai edes työvoiman resursoinnin työkaluja, jolloin nämä ominaisuudet nostavat turhaan ohjelmiston kustannusta.

Pienenkin ohjelmistoprojektin hallintatyökalu pitäisi kuitenkin olla tavoitettavissa siten, että kaikki projektiin osallistuvat pääsevät siihen helposti. Projekti pitäisi olla myös helposti aikataulutettavissa ja edistymistä pitäisi olla helppo seurata. Lähtökohtaiseti ohjelmistoyrityksillä on käytössään jo jokin tuotannonhallintaohjelmisto. Tällöin helpoin tapa on saada suoraan sopivat käyttöoikeudet kaikille projektiin osallistuville tähän järjestelmään, jos se teknisesti on mahdollista. Lisäksi saattaa silti tulla tarvetta oman sisäisen kehitysprojektin hallinnalle. Tässä tapauksessa haasteeksi saattaa tulla tiedon pirstaloituminen moneen eri hallintaohjelmaan.

4 OHJELMISTON SUUNNITTELUN JA TOTEUTUKSEN PERUSTEITA

Ohjelmistotuotanto voidaan jakaa eri vaiheisiin tai alaprojekteihin. Jako on tärkeä erityisesti aikatalutuksen kannalta. Eri vaiheissa myös tarvitaan eri henkilöresursseja, jolloin resurssien jako on helpompaa, jos projekti on jaettu selkeisiin vaiheisiin.

4.1 Esitutkimus, vaatimusmäärittely ja toiminnallinen määrittely

Esitutkimuksessa kartoitetaan vaihtoehdot, miten ohjelmisto voidaan tuottaa. Tuotanto voidaan esimerkiksi tehdä itse, teettää alihankintana tai ostaa valmis ohjelmisto. Esitutkimuksessa voidaan tehtdä alustavia määrittelyjä, tarveanalyysejä sekä riskien ja kannattavuuden arviointia. Koska asiakas useimmiten tietää oman liiketoimintansa ympäristön ja tarpeet paremmin kokonaisuutena kuin toimittaja, on tilaajan yleensä hyvä tehdä tämä vaihe itse tai yhteistyössä asiantuntijayrityksen kanssa. Määrittelyssä myös kuvataan syntyvän tuotteen toiminnallinen lopputulos. (Haikala & Mikkonen 2011, 20-21)

Ohjelmiston haluttujen toiminnallisuuksien lista saattaa helposti paisua suureksi, kun mukaan keksitään kaikki vaatimusmäärittelyn edetessä mieleen tulevat toiminallisuudet. Osa näistä toiminnallisuuksista on kuitenkin kokonaisuuden kannalta enemmän projektin työmäärää ja kustannuksia rasittavia kuin lopputuotteen kannalta merkityksellisiä toimintoja.

Tominnallisuuksien listauksen jälkeen, voidaan muutamalla yksinkertaisella tavalla karsia osia pois. Periaatteena on saada eri tavoin karsittua pois vähemmän tärkeät ominaisuudet.

- **Älä tee -lista**. Pakota itsesi keksimään jokaista toteutettavaa ominaisuutta kohti yksi ominaisuus, jota ei toteuteta.
- **”Pelaa pokeria”.** Pelataan näennäisesti pokeria projektin avainhenkilöiden kesken. Henkilöt saavat panostaa eri toiminnallisuuksiin. Eniten panoksia saaneet toiminnallisuudet toteutetaan ja vähiten saaneet jätetään pois tai siirretään mahdollisesti myöhemmin toteutettavaksi.
- **Product boxing.** Tuotettavalle ohjelmistolle tai ohjelmistotuotteelle suunnitellaan kuvitteellinen myyntipakkaus. Myyntipakkauksessa listataan periteisesti tuotteen

parhaat ja tärkeimmät ominaisuudet. Tämä tuo esiin oman tuotettavan ohjelmistosi tärkeimmät ominaisuudet. Muiden ominaisuuksien tärkeys kyseenalaistetaan.

- **Keilaratastrategia.** Valitaan suppea kohderyhmä tai käyttötarkoitus ja toteutetaan ensin siihen vaadittavat ominaisuudet. Loput ominaisuudet hylätään tai siirretään myöhemmin toteutettavaksi.

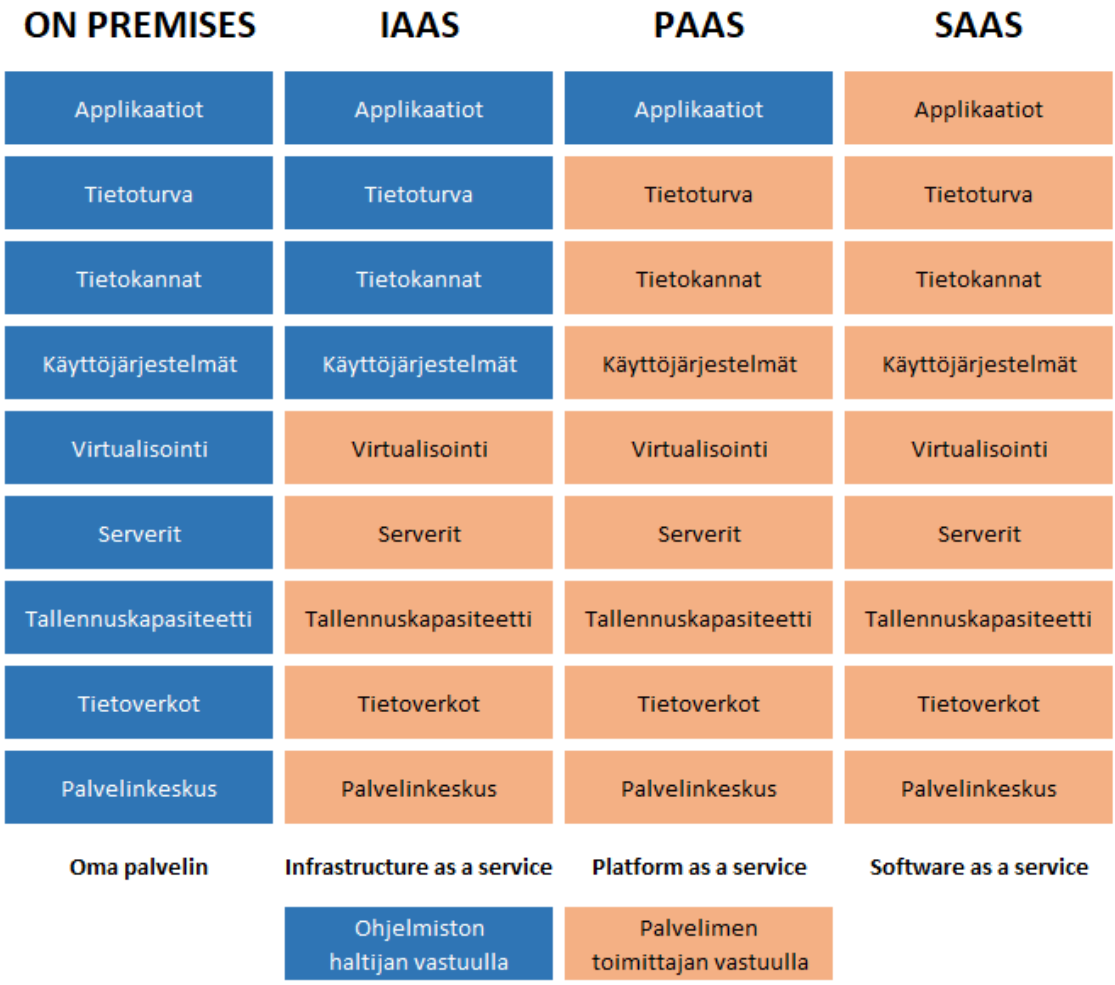
(Järvenpää & Kovanen 2018, 16)

4.2 Palvelinympäristön valinta

Itse ohjelmiston tuottamisen lisäksi on valittava mistä ohjelmisto tarjotaan käyttäjilleen eli missä ohjelmistoa suoritetaan. Ohjelmisto voi olla yksittäisiin tietokoneisiin paikallisesti asennettava ohjelmisto, jolloin valinta on helppo. Näissä tapauksissa ohjelma asennetaan tapauskohtaisesti niihin tietokoneisiin, joissa sitä tarvitaan. Kuitenkin suuri osa nykyohjelmistoista suoritetaan ja tarjotaan ulkoiselta palvelimelta ja ohjelmiston käyttäjät käyttävät sitä sisäverkon tai internetin välityksellä. Esimerkiksi kaikki nettiselaimella hallittavat ohjelmistot tarjotaan erilliseltä palvelimelta, näistä puhutaan yleisesti pilvipalveluina.

Joissain tapauksissa ohjelmistoa voidaan tarjota myös yrityksen sisäverkossa olevalta palvelimelta, erityisesti jos ohjelmisto on pääosin tarkoitettu vain sisäiseen käyttöön. Näissä tapauksissa pääsy palveluun ulkoverkosta voidaan jopa kokonaan estää, joka parantaa osaltaan tietoturvaa. Pilvipalveluiden yhteydessä yleisimmin käytetyt termit on hyvä tietää vähintään pääpiirteittäin. Ne ovat merkityksellisiä strategisesti ja myös taloudellisesti.

Pilvipalveluiden yhteydessä yleisimmät termit ovat IaaS, PaaS ja SaaS. Myös *on premises* -nimitystä käytetään silloin, kun kyse on yrityksen tiloissa olevasta omasta palvelimesta tai palvelinkokonaisuudesta. Näiden eri nimitysten alle pohjimmiltaan kätkeytyy palvelimen ulkoistamisen aste ja palvelinten ja sen sisäisten toiminnallisuuksien ylläpidon vastuun jako. Palvelinten ylläpidon tarjoaja ei tarvitse olla sama kuin ohjelmiston tekijä tai ohjelmiston ylläpitäjä, eikä usein näin olekkaan.



Kuva 8. Pilvipalveluiden yleisimmät muodot, mukaillen Nebula Inmics Oy:tä

- **Iaas (Infrastucture as a service)** tarkoittaa infrastrukstuuria palveluna. Käytännössä yrityksellä on oma palvelin, joka on sijoitettu ylläpitoa tajoavan yrityksen palvelinsaliin. Palvelinsalin ylläpitäjä huolehtii palvelimen fyysisestä ylläpidosta, kuten katkeamaton sähkönsyöttö sekä jäähdytys. Usein myös palvelinten fyysinen huolto on palvelinsalin tarjoajan vastuulla, esimerkiksi tilanteissa, joissa palvelimen osia pitää vaihtaa rikkoituneen tilalle.
  Tässä mallissa kaikki palvelinten ohjelmallinen hallinta ja ylläpito on tilaajan eli palvelinlaitteen omistajan vastuulla. IaaS-malli sopii parhaiten yrityksille, joilla on riittävästi sisäistä IT-osaamista palvelinten ylläpitoon.
- **PaaS (Platform as a Service)** tarkoittaa alustapalvelua ohjelmistojen kehittämiseen. PaaS-mallissa ohjelmistokehittäjille tarjotaan valmis alusta, johon tarjottavat palvelut asennetaan ohjelmistokehittäjien toimesta. Tässä mallissa

huomioitavaa on se, että palvelimen käyttöjärjestelmän ja tietokantapalvelimen ylläpitovastuu on palvelun tarjoajalla.

- **SaaS (Software as a Service)** tarkoittaa ohjelmistoa palveluna. Se on tutuin pilvipalvelumuoto. Tässä mallissa ohjelmiston toimittaja vastaa kokonaisuudessaan palvelun toimivuudesta. SaaS palvelusta maksetaan usein vain käytön mukaan. Käyttäjälle jää vastuu vain käyttäjäoikeuksien hallinnasta ja lakisääteisistä velvoitteista ohjelmiston käytössä.
  (Inmics Nebula Oy, 2018)

Ohjelmistokehitysprojektin ollessa räätälöity vahvasti vain tilaajan tarpeisiin, on luontevin malli PaaS, jolloin palvelimen käyttöjärjestelmän ja tietokantapalvelinten ylläpidon ja päivitysten vastuu voidaan siirtää palvelinsalin ylläpitäjälle. Erityisesti käyttöjärjestelmäpäivitysten ajaminen voidaan katsoa tietoturvallisuuden kannalta hyvin tärkeäksi, joten sen säännöllisestä tekemisestä voidaan tällöin varmistua.

SaaS malli sopii kokonaisuuksiin, jossa ohjelmistoa ja sen käyttöoikeutta myydään usealle eri organisaatiolle tai kuluttajalle. Tällöin vastuu palvelun toiminnasta on kokonaan ohjelmiston omistajalla eli pääsääntöisesti myyjällä.

On premises mallin hyvinä puolina voidaan pitää nopeaa verkkoyhteyttä sisäverkossa oleville muille tietokoneille. Tämä korostuu erityisesti, jos palvelimelta käytetään suuria tiedostoja tai viive pitää olla mahdollisimman lyhyt. Huonona puolena omissa tiloissa sijaitsevalle palvelimelle voidaan pitää täyttä omistajan vastuuta palvelimen elinkaaren tarvitsemista huolloista ja päivityksistä. Tätäkin voidaan toki helpottaa eri asteisilla ylläpitosopimuksilla alan tarjoajien kanssa. Laitetilan turvallisuus on usein myös huonompi kuin alaan erikoistuneella palveluntarjoajalla.

4.3 Toteutusprojekti

Toteutusprojektin alussa tehdään tarkempi tekninen määritys, yksityiskohtaisempi suunnittelu sekä suunnitelma tuotannon vaiheista. Tarkempia suunnitelmia tehdään muun muassa suunnittelusta, itse ohjelmoinnista ja testauksesta. Toteutusprojektiin sisältyy usein monia alaprojekteja, kuten hankinta-, käyttöönotto- ja koulutusprojekteja. Osa näistä osaprojekteista voidaan toteuttaa myös itse. Näiden ohella toteutukseen voi liittyä muitakin tehtäviä, kuten kilpailutusta, ylläpitoa ja käyttötukea. (Haikala & Mikkonen 2011, 21).

Pienen yrityksen kannalta ei ole useinkaan tarkoituksenmukaista hankkia lisäresursseja omaan organisaatioon näiden osaprojektien ja tehtävien toteuttamiseen. Joihinkin tehtäviin resurssi löytyy luontaisesti, mutta puuttuva resurssi on kustannustehokasta ostaa alihankintana, koska tarve on usein lyhytaikaista.

Muistakin tuotannoista tuttu termi ja periaate Minium Viable Product (MVP), sopii käytettäväksi myös ohjelmistotuotantoon. Käytännössä tämä tarkoittaa sitä, että valmistetaan tuotteesta, eli tässä tapauksessa ohjelmistosta, mahdollisimman pienellä vaivalla toimiva versio, joka kuitenkin sisältää toiminnalliset pääosat. Tällä versiolla voidaan varmentaa, että ratkaisut ja tehdyt toiminnot ovat oikeita. MVP version olisi hyvä sisältää erityisesti kaikki kriittisimmät toiminnot, sekä kehitysprojektin riskipitoisimmat osat. Näitä riskipitoisia ja vaikeammin ratkaistavia osia ja toimintoja tulee herkästi työnnettyä projektin loppuvaiheeseen, jotka kuitenkin voivat olla koko projektin onnistumisen kannalta tärkeä ratkaista ajoissa. (Järvenpää & Kovanen 2018, 14)

Tuotettavan ohjelmiston ja siihen liittyvän liiketoiminnan ollessa kokonaan uusi ja sisältäessä paljon uutta teknistä sisältöä, voisi ennen varsinaista tuotantoa tulla kyseseen konseptitodistus. Tutummin tätä kutsutaan nimellä Proof of concept (PoC). Nimitys ja toimintatapa on tuttu muyös muilta teollisuuden aloilta. Ideana on tehdä mahdollisimman yksinkertainen prototyyppi, jonka on tarkoitus todistaa idean toteuttamiskelposuus. Erityisen hyödyllinen PoC on tilanteissa joissa ohjelmistoa ollaan tekemässä olemassa oleviin vanhoihin laitteisiin tai laiteympäristö on muuten erikoisempi. Myös kuluttajille suunnattua uudenlaista palvelua voisi olla hyödyllista ”koeponnistaa” kevyellä testiversiolla.

4.4 Testaus

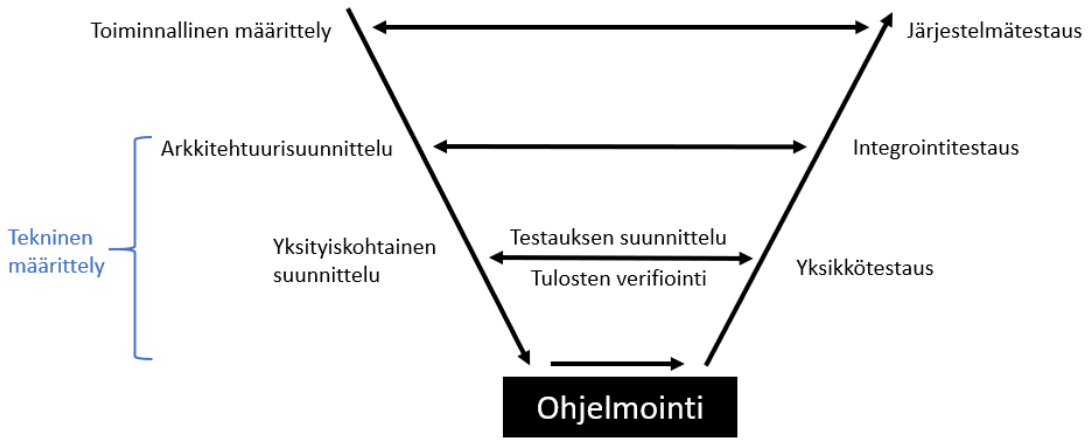
Ohjelmiston tuotannon aikana ja erityisesti loppuvaiheessa tehdään testausta, jolla varmennetaan tehdyn ohjelmiston virheetön toiminta eri tilanteissa. Testausta varten pystytetään yleisesti testiympäristö, joka vastaa ominaisuuksiltaan tuotantoon otettavaa ohjelmistoa ja ohjelmistoympäristöä. Testiympäristö on tärkeä myös käyttöönoton jälkeen, jotta myöhemmin tuotantoympäristössä esiin tulevat vikatilanteet voidaan toistaa, paikantaa ja korjata hallitusti.

Testauksella tarkistetaan tehdyn järjestelmän vastaavan suunniteltua sekä määritettyjä vaatimuksia. Toiminnallisten ominaisuuksien lisäksi ohjelmiston turvallisuutta pitää testata. Turvallisuutta voi testata esimerkiksi penetraatiotestauksella, jossa yritetään

tuntkentua palveluun tai järjestelmään ja sitä kautta havaitsemaan tietoturva-aukot. Toinen tärkeä turvallisuuteen liittyvä testausmenetelmä on stressitestaus, jolla pyritään luomaan järjestelmään suuri kuormitus, joka paljastaa heikoimmat kohdat ja ne voidaan korjata tai suojata. Stressitestiin voi kuulua myös verkkoyhteyden katkeamisen testausta tai sähkökatkoksen aiheuttama poikkeustilanne. (Kyberturvallisuuskeskus, Traficom 2018. 30-32)

Testaus suoritetaan systemaattisesti ja suunnitellusti. Testausta on hyvä suorittaa myös joku muu kuin itse ohjelmiston tekijä. Tekijän testaus saattaa kohdistua enemmän toiminnallisuuden todentamiseen kuin vikakohtien etsimiseen. (Haikala & Mikkonen 2011, 205)

Testauksen periaatetta voidaan kuvata V-mallilla. V-mallissa testausta tehdään testaustasoa vastaavalla suunnittelutasolla. Alihankintana tehtävän ohjelmiston toiminnallinen määrittely tehdään luontevimmin tilaajan johdolla, joten tilaajan tehtävä on ainakin osin tehdä järjestelmätestausta. Järjestelmätestauksessa testataan koko järjestelmän toimintaa ja miten se vastaa toiminnallista määrittelyä. Tekninen määrittely on usein ohjelman kehittäjän vastuulla, jolloin integrointi- ja yksikkötestaus voidaan jättää lähinnä ohjelmistokehittäjän vastuulle. (Haikala & Mikkonen 2011, 206)



Kuva 9. Testauksen V-malli (Haikala & Mikkonen 2011, 207)

4.5 Käyttöönotto ja elinkaaren hallinta

Testauksien ja varmistusten jälkeen siirrytään käyttöönottovaiheeseen. Käyttöönotto saatta olla hyvä erottaa kokonaan omaksi projektikseen, jos se on suuritöinen. Käyttöönotto arvioidaan usein helpommaksi ja vähemmän resurssia vaativaksi kuin se oikeasti onkaan. Huomioon on otettava asioita, kuten koulutus, käyttäjätuki, kayttöohjeiden laadinta ja muutosvastarinnan voittaminen. Ohjelmallisen tuotantoympäriston pystytys ja datan tuonti mahdollisista vanhoista järjestelmistä pitää myös laskea käyttöönottovaiheen töihin. Jos mahdollista, voidaan aloittaa pilottiasiakkaalla tai pilottikäyttäjällä. Voidaan myös valita jokin asiakasryhmä, jolla uuden ohjelmiston toimivuutta testataan. (Juvonen 2018, 103-104)

Elinkaaren hallinnassa on erityisen tärkeää tiedostaa, että myöskään ohjelmistot eivät toimi vuosia ilman ylläpitoa. Tämä seikka on huomioitava, kun suunnitteluvaiheessa tehdään valintoja. Teknologioiden valinnoissa on hyvä keskittyä alustoihin ja ratkaisuihin, joille on ennustettavissa pitkä elinkaari. Tämäkään ei aina kuitenkaan riitä, jolloin hyvän tuki- ja ylläpitokumppanin merkitys tulee erityisen tärkeäksi. (Alfame Oy 2015, 13)

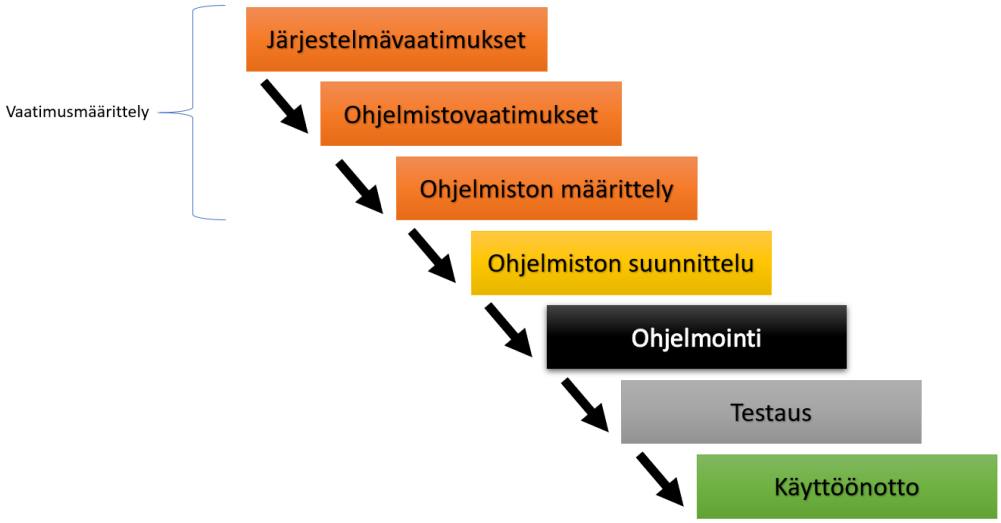
Tuotantoprojektin aikana, mutta viimeistään käyttöönotossa tuotantojärjestelmiä ja niiden tietokantoja on ryhdyttävä varmuuskopioimaan. Varmuuskopion ottaminen on hyvä olla automaattista ja tapahduttava riittävän usein. Tiheys riippuu paljolti syntyvän datan määrästä sekä kriittisyydestä. Varmuuskopioita voidaan myös ottaa reaaliaikaisesti esimerkisi kahdennettuun ja peilattuun tietokantaan, jolloin tieto tallennetaan kahteen tietokantaan samanaikaisesti. Varmuuskopioiden on aina sijaittava fyysisesti eri palvelimilla, mieliummin kokonaan eri rakennuksessa. Erittäin kriittisessä datan tallennuksessa voidaan käyttää myös disaster logia, joka voi tarkoittaa esimerkiksi samanaikaista tekstimuotoista tallennusta syntyvästä datasta. Tällaisesta varmennuksesta tiedon palautus on kuitenkin työlästä.

5 OHJELMISTOPROJEKTIEN TOTEUTUSMENETELMÄT

Ohjelmistotoimittajat ovat yrityksiä, joiden tuotteena on myydä ohjelmistokehitystä. Toimittajilla on omat sisäiset prosessinsa ja toimintatavat asiakkaan tilaaman tuotteen valmistamiseen. Ohjelmistoyritykset käyttävät alalle ominaisia prosesseja ja tuotannonhallintaa. Tilaajan on hyvä tietää perusteet alihankkijan sisäisen tuotannon ohjauksesta, tästä saattaa olla hyötyä projektin hallinnassa. Yleisimmät tuotantoprosessityypit voidaan jakaa kahteen pääryhmään, vesiputousmalliin ja ketterään kehitykseen. Vaikka tilaaja ei yleisesti suoraan itse tuotantoprosessiin osallistu, on hyödyllistä ymmärtää näiden kahden pääperiaatteet sekä niiden erot sekä muita perusperiaatteita ohjelmistokehitysen tuotantotavoista ja projektimalleista.

5.1 Vesiputousmalli

Vesiputousmallia voidaan pitää totetusmenetelmuien kantaisänä. Nimi kuvaa hyvin sitä tapaa, jolla malli toimii. Vesiputousmallissa tuotannon vaiheet seuraavat toisiaan eikä aikaisempiin vaiheisiin ole tarkoitus palata. Tämän mallin hyvä puoli on selkeys ja yksinkertaisuus. Huono puoli on mentelmän kankeus ja se elämän tosiasia, että kaikki ei mene kerralla oikein ja ongelmia huomataan vasta jälkikäteen, tällöin olisi tarve palata aiempaan vaiheeseen. (Juvonen 2018, 16)



Kuva 10. Vesiputousmallin havainnekuva (mukaillen Juvonen 2018, 16)

5.2 Ketterät menetelmät

Ketteriin menetelmiin tutustuessa ei voi olla törmäämättä Agile Alliancen antamaan Agile manifestoon, jossa joukko ohjelmistokehittäjiä loivat perusteet ketterämmälle ohjelmistokehitykselle yhteiskokouksessaan Utahissa vuonna 2001. (The Agile Alliance, 2001)

Kuva 11. Agile Manifesto (The Agile Alliance, 2001)

Yleisin ketterä menetelmä on nimeltään Scrum. Scrum noudattaa periaatteiltaan ketterän kehityksen periaatteita. Ketteryyden lisäksi luonteeseen kuuluu myös jatkuva kehittyminen. Scrumin pääperiaate rakentuu kolmen roolin sekä 1-4 viikon mittaisista sprinteistä. Sprinteissä on tarkoitus saada valmiiksi jotain kokonaisuusksia tai toimivia osakokonaisuuksia. Scrumissa työtä tehdään toiminto kerrallaan ja suunnittelu vasta alkaneen sprintin sisällä. (Juvonen 2018, 16)

Scrumin kolme eri roolia voi tulla tutuksi myös ohjelmiston tilaajalle, vaikka ei suoranaisesti osallistu tuotantoprosessiin. Tällöin on hyvä tietää, miten roolitus ja sprint-työtapa Scrumissa toimii.

- **Tuotteen omistaja** on asiakkaan edustaja. Omistaja tekee työlistoja ja ominaisuuslistoja yhdessä tilaajan kanssa. Omistajan ei tarvitse osata itse ohjelmointia, mutta tekninen ymmärrys ja vuorovaikutustaidot ovat tärkeitä.

- **Scrum mestari** on vastuussa tuotantotiimin toimintaedellytyksistä, laitteista ja muista työtä haittaavien esteiden poistamisesta. Scrum mestari vastaa myös scrum prosessin noudattamisesta ja hänen on oltava scrum prosessin asiantuntija. Scrum mestari ei kuitenkaan ole projektipäällikkö tai tiiminvetäjä.
- **Tiimi** on se osa ryhmää, jotka tekevät itse suorittavan työn. Scrumissa tiimin on tarkoitus olla itseohjautuva ryhmä. Tiimi valitsee itse työlistalta suoritettavat tehtävät, prioriteetti huomioiden. Tiimin sisällä työtehtävät jaetaan tasaisesti, lopulta käytännössä osaamiskyvyt huomioiden.

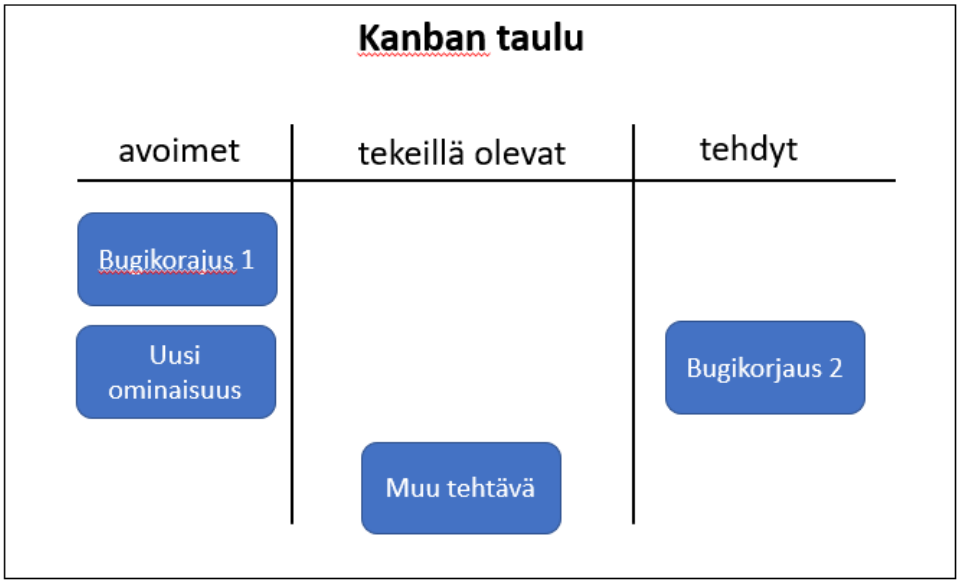
(Juvonen 2018, 16)

Kuva 12. Yksittäisen sprintin kulku (Juvonen 2018, 20)

Yksi muista mainittavista tuotannon ohjaustavoista on muualtakin teollisuudesta tuttu **Kanban** malli. Kanban on yksinkertainen mentelmä, jossa työvaiheilla tai ominaisuuksilla voi olla kolme eri statusta. Statusta muutetaan työn edistymisen mukaan.

- avoimet tehtävät
- tekeillä olevat tehtävät
- tehdyt tehtävät

Ohjelmistojen kehityksessä tämä on erityisen käytännöllinen ylläpitovaiheessa. Ylläpitovaiheessa työt voidaan ottaa sisään tiketteinä, joka on käytännössä yksi työtehtävä tai hoidettava asiakastapahtuma. Ohjelmistotuotannossa tiketit voidaan jakaa eri tyyppisiin tikettehin, kuten bugi-, eli virhekorjauksiin, uusin ominaisuuksiin tai yleisiin tehtäviin. Tiketit voidaan jakaa tuotantotiimeille tai yksittäisille työntekijöille, jolloin tiimien ja tekijöiden työkuormaa ja työjonoa voidaan seurata. (Juvonen 2018, 16)



Kuva 13. Yksinkertaistettu Kanban-taulu

Kanban taulu on yleinen kaikissa sähköisissä projektinhallinta-ohjelmistoissa. Kanban taulun periaate ja tikentöintijärjestelmä sopivat hyvin yhteen. Ohjelmistoa teettävän yrityksen kannalta on hyvä, jos pääsy tuotannon ohjaukseen ainakin näiltä osin myös teettävällä yrityksellä. Tällöin projektin etenemisestä saadaan tarkempi kuva sekä reaaliaikanen tieto. Teettävän yrityksen osallistuessa enemmän itse tuotantoon ja suunnitteluun, tarvitaan oikeus myös tehdä uusi tikettejä.

6 DOKUMENTOINTI JA SOPIMUKSET

Dokumentointi on tärkeä osa missä tahansa tuotannossa. Dokumentoinnilla tarkoitetaan mahdollisimman tarkkaa kirjallista kuvaamista, jossa ohjelmiston toiminta ja sisältö olisi kuvattu yksiselitteisesti väärinymmärrysten välttämiseksi. Käytännössä kaikki suunnitelmat ja valmiit tuotokset ja niiden tuotannon aikana kertyneet dokumenti ovat osa dokumentointia. Dokumentoinnin osana voidaan pitää myös yritysten välisiä sopimuksia.

6.1 Dokumentointi

Organisaatiossa on usein jo ennalta dokumentointu järjestelmiä ja käytettyjä prosesseja. Dokumentoinnin perustana on tällöin hyvä käyttää jo olemassa olevaa dokumentointia. Jos dokumentointia ei ole olemassa, olisi tässä samassa yhteydessä hyvä tehdä myös kuvaus järjestelmistä ja prosesseista, joihin tuotettava ohjelmisto liittyy. Dokumentointi on tärkeää hankitavaiheen lisäksi myös ohjelmiston käyttöönoton yhteydessä ja jatkossa ylläpitovaiheessa. Dokumentoinnista on erityisen paljon apua vikatilanteiden paikannuksessa ja korjauksessa. On hyvä ottaa myös huomioon asiaan liittyvien henkilöiden mahdollinen vaihtuminen jopa projektin aikana, jolloin hyvä dokumentointi auttaa uusien henkilöiden perehtymisessä ohjelmistoon ja sen toimintaan. (Alfame Oy 2015, 6)

Tärkeää on dokumentoida myös projektin aikana, ei ainoastaan suunnitteluvaiheessa ja valmistumisen yhteydessä. Myös muuttuneet suunnitelmat ja määrittely on syytä kirjata heti. Projektin aikana syntynyt dokumentaatio auttaa esimerkiksi käyttäjien perehdyttämisessä sekä käyttöohjeiden tekemisessä. Tarkka tekninen dokumentaatio voi olla joillain aloilla läkisääteinen ja auditoidaan määrävälein. Dokumenteista voidaan myös varmentaa, että järjestelmä on tehty säännösten mukaan.

6.2 Sopimukset

Tilaajan ja toimittajan välille tehdään aina kirjallinen sopimus. Sopimuksen tekoa ja sopimusehtojen määrittämistä helpottamaan on olemassa IT-alalle tehdyt sopimusehtoliitteet ja yleiset sopimusehdot. Näiden käyttö helpottaa huomattavasti sopimuksien tekoa ja auttaa ymärtämään IT-alan sopimusten sisältöä. Yksi tärkeimmistä sopimuskohdista on syntyvän ohjelmiston ja sen osien immateriaalioikeuksien määrittäminen.

Immateriaalioikeudet kuuluvat lähtökohtaisesti tilaajalle eli maksajalle. Huomioitavaa on myös erillisen sopimuksen tekeminen ylläpitovaiheesta varsinaisen ohjelmistotuotannon jälkeen.

IT2018 YSE sopimusehdot sisältävät yleisiä pelisääntöjä koskien IT palveluita. Saatavilla on yleisten ehtojen lisäksi 9 erityisehtoja sisältävää sopimusehtoliitettä eri tarpeisiin. Sopimusehtojen lisäksi IT2018 sisältää yleisimmät sopimusmallit eri IT projektien tarpeisiin. Sopimusehdot sekä siihen liittyvä muu materiaali on maksullista. Lisenssi ehtojen käyttöön myydään vuosilisenssinä. IT2018 ehdot on yleisesti tunnettu ja käytännössä ainoa toimija Suomessa. Ehdot on satavilla myös englanniksi. IT-alan sopimusehdot on laadittu ohjelmistojen toimittajayritykset sekä niiden asiakkaita edustavat järjestöt yhdessä, jolloin kummankin osapuolen edut on huomioitu. Sopimusehtoja uusitaan määrävälein. Uudet sopimusehdot julkaistaan noin kolmen vuoden välein. (Keskuskauppakamari 2021)

IT2018 sopimusehtojen melko korkeasta hinnasta huolimatta sen hankinta on todennäisesti silti kannattavaa. Pääosin ohjelmistoyrityksillä on sopimusehtojen käyttöoikeus entuudestaan olemassa, jolloin tilaajan ei lisenssiä tarvitse ostaa erikseen. Projektin tilaajan on kuitenkin hyvä tutustua kaikkiin IT2018 olemassa oleviin dokumentteihin, jolloin syntyy parempi käsitys alan periaatteista ja toimittajalta osataan vaatia oikean tyyppisiä sopimuksia.

Hankintahetkellä ja alihankkijan valinnan yhteydessä on kiinnitettävä huomioita myös immateriaalioikeuksiin. Jos oikeudet tuotettavaan ohjelmistoon jäävät työn suorittavalle osapuolelle, voi jatkokehitys ja ohjelmistotuotannon myöhemmän vaiheen kilpailutus olla ongelmallista. Immateriaalioikeus ohjelmistoon määrittää esimerkiksi sen, kenellä alkuperäinen lähdekoodi ohjelmistosta on. Immateriaalioikeudet olisi hyvä mahdollisuuksien mukaan aina pyrittävä pitämään omassa organisaatiossa, joka mahdollistaa muun muassa jatkokehityksen uuden alihankkijan kanssa, jos tarve näin vaatii. (Alfame Oy 2015, 12)

Lähtökohta ohjelmiston tuotantoon toimittajan kannalta ovat asiakkaalta tulevat vaatimukset. Mitä tarkemmin tilaaja on valmiin tuotteen toiminnot määritellyt, sen suoraviivaisemmin projektia voidaan toteuttaa. Käytännössä projektin aikana tulee kuitenkin runsaasti muutoksia ja lisäyksiä. Näihin on syytä varautua jo aloitusvaiheessa. Jos kyseessä on huomattavan turvallisuuskriittinen sovellus, määrittely tehdään erityisen tarkaksi ja projektin aikana tapahtuvat määrittelyn muutokset tehdään ohjeistetusti ja hallitusti.

Projektin aikaisista muutoksista on hyvä sopia osapuolien kesken jo ennakkoon. Vastuu muutoksien aiheuttamista kuluista vaihtelee suuresti sen mukaan, tehdäänkö työtä kiinteällä projektihinnalla tai suoraan tuntilaskutuksella. Lisäksi on hyvä sopia miten toimitaan selvissä vikakorjauksissa tai toisaalta kokonaan uusissa toiminnallisuuksissa, joita projektin aikana on ilmennyt. (Haikala & Mikkonen 2011, 22)

7 YHTEENVETO

Opinnäytetyön aiheena oli pienen yrityksen ohjelmistotuotanto kokonaan tai osittain ulkoistettuna. Tavoitteena oli kehittää projektiosaamista ja onnistumisen mahdollisuutta ulkoistetuissa ohjelmistoprojekteissa. Tämä työ on jaettu kuuteen eri toiminalliseen kokonaisuuteen tai työn vaiheeseen.

Työ aloitettiin tutustumalla liiketoiminnallisiin ja taloudellisiin lähtökohtiin, joiden pitää täyttyä, jotta projektiin ryhtyminen on yleensäkään järkevää ja perusteltua. Toiseksi käsiteltiin vaatimusten määrittelyä, jolla selvitetään ja suunnitellaan mitä kaikkea tuotettavan ohjelmiston pitää sisältää ja pystyä käsittelemään sekä mitä toiminallisuusksia tarvitaan. Kolmantena käsiteltiin projektia itse projektityön kannalta ja selvitettiin periaatteita ohjelmistoprojektin tekemisestä onnistuneesti. Neljännessä luvussa käsiteltiin ohjelmistojen suunnittelun yleisiä periaatteita ja siihen liittyviä sääntöjä, työtapoja ja eri vaiheiden teknisiä valintoja. Neljäs luku on käsitelty käytännön tason toimina kronologisessa järjestyksessä, joita ohjelmistoprojektissa on tehtävä ja otettava huomioon. Viides luku käsittelee itse ohjelmiston tuotantoa ohjelmistokehittäjien kannalta sekä miten tuotantoa ohjataan eri tavoilla. Kuudes luku käsittelee ohjelmistotuotannon sopimuksia sekä ohjelmiston dokumentointia työn aikana.

Kuhunkin yllämainittuun osa-alueeseen on koottu perusasioita yksikertaistetulla ja mahdollisimman ymmärrettävällä tavalla, jotta ohjelmistotuotannosta melko tietämätönkin voisi niitä hyödyntää. Tämä helppo ymmärrettävyys on yksi tämän työn perusperiaatteista.

Tarkemmin aiheeseen ja sen kirjallisuuteen perehdyttäessä voidaan todeta, että ohjelmistokehitysprojekti kokonaisuudessaan ei ole mahdottomasti erilainen kuin muutkaan tuotekehitys- tai tuotteen valmistusprosessit. Alalla on kuitenin paljon eri toimintatapoja ja hienolta kuulostavia termejä, joita tässä työssä on tarkoituksella tuotu ymmärrettävämpään muotoon.

Työn aikana myös ymmärrys projektien hallintaan ja projektityöskentelyyn parani. Sen merkitys on osoittautunut melko suureksi. Projektityöskentely sinänsä ei juuri eroa muidenkaan alojen projekteista. Samat projektihallinan perusperiaatteet pätevät myös ohjelmistokehitykseen kuin muihinkin aloihin. Eroa löytyy muun muassa projektihallinnan

työkaluista, jotka on usein tehty silmälläpitäen juuri määrättyä teollisuuden alaa, kuten tässä tapauksessa ohjelmistokehitystä.

Vaikka IT-järjestelmät ja tekniikat yleisesti ottaen kehittyvät nopeasti, kuitenkin peruspilarit pysyvät melko hyvin muuttumattomina. Osa lähdemateriaaleista on yli kymmenen vuotta vanhoja, mutta periaatteet ovat silti melko yhteneväsiä uudempien lähteiden kanssa.

Ohjelmistokehityksen ja ohjelmistojen erityispiirteenä voidaan pitää sitä, että ohjelmistoala ja IT-järjestelmät liittyvät enenevissä määrin kaikkiin muun teollisuuden ja palvelun aloihin. Ei ole enää alaa, jossa ei olisi käytössä jotain ohjelmistotuotetta tai sulautettuja ohjelmistoja. Osassa perinteisistä aloista IT-järjestelmät ja ohjelmistot ovat jo määräävässä asemassa päivittäisessä työssä ja elintärkeitä koko tuotannon ja toiminnan kannalta.

Pienissä yrityksissä ohjelmistokehityksen teettäminen tai ohjelmistojen hankinta tulee varmasti ajankohtaiseksi kaikille jossain vaiheessa, useimmille se on arkipäivää. Tällöin perusteiden ymmärtäminen ennen projektiin aloitusta on ensiarvoisen tärkeää. Vaikka tilaajan ei tarvitse syvällisesti ohjelmistotuotantoa ymmärtää, onnistuminen on huomattavasti todennäköisempää, jos kaikki tieto ei tule uutena tai myyjän markkinointiviestinnän kautta. Jos yrityksen sisällä ei ole edes nimellisesti alaan liittyvää kokemusta tai harrastuneisuutta, on hyvä hankkia sitä puolueettomalta kolmannelta osapuolelta ennen projektiin ryhtymistä.

Työn aikana on tarkentunut selvästi se, kuinka tärkeä ennalta suunnittelu on ohjelmistoprojektin onnistumisen kannalta. Oli se sitten projektin hallinta tai ohjelman käyttöliittymä tai mitä tahansa siltä väliltä. Huolellisella suunnittelulla, yksityiskohtaisella dokumentoinnilla, tarkalla organisoinnilla ja hyvällä testaamisella voidaan parhaassa tapauksessa saada projekti päätökseen onnistuneesti ja aikatalussa.

LÄHTEET

Haikala I. & Mikkonen T. 2011. Ohjelmistotuotannon käytännöt. Talentum Media Oy

The Agile Alliance 2001. https://agilemanifesto.org/

Alfame Oy 2015. 12 Vinkkiä ohjelmistokehityksen ostajalle. Viitattu 1.2.2021. https://www.alfame.com/hs-fs/hub/395117/file-2654455542-pdf/files/12_vinkkia_ohjelmistokehityksen_ostajalle_Alfame.pdf

Huuhka Terttu 2017. Tehokkaan hankinnan työkalut. Books On Demand

Järvenpää J. & Kovanen P. 2018. Ohjelmistokehityksen ostaja pikaopas. Vincit Oy

Inmics Nebula Oy 2018. Pilven monet kasvot – Iaas, Paas ja Saas. Viitattu 5.4.2021 https://www.inmicsnebula.fi/fi/blogi/pilven-monet-kasvot-iaas-paas-ja-saas

Keskuskauppakamari 2021. IT ehdot 2018. Viitattu 1.5.2021 https://it-ehdot.fi/

FinnLectura 2013. Ketterää kehitystä. Useita kirjoittajia

Scott Berkun 2006. Projektihallinnan taito. Readme.fi

Alfame Oy n.d. Vaatimusmäärittely ketterässä ohjelmistokehityksessä. Viitattu 13.5.2021 https://www.alfame.com/hubfs/files/Vaatimusma%CC%88a%CC%88rittely%20kettera%CC%88ssa%CC%88%20ohjelmistokehityksessa%CC%88%20-opas.pdf

Juvonen Rami 2018. Ohjelmistoprojektin sudenkuopat ja miten ne vältetään. Books on Demand

Lehtimäki Timo 2006. Ohjelmistoprojektit käytännössä. Readme.fi

Open Source Initiative. The Open Source definition**.** Viitattu 25.9.2021 https://opensource.org/docs/osd

Object Management Group 2017. Viitattu 25.9.2021 https://www.omg.org/spec/UML/2.5.1/PDF

Forselius Pekka 2013. Onnistunut tietojärjestelmän hankinta. Talemtum media Oy.

Suomidigi 2018. ICT-palvelujen kehittäminen: Vaatimusmäärittely. Viitattu 4.10.2021 https://www.suomidigi.fi/sites/default/files/2020-06/JHS173.doc

Kyberturvallisuuskeskus, Traficom 2018. Turvallinen tuotekehitys. Viitattu 4.10.2021 https://www.kyberturvallisuuskeskus.fi/sites/default/files/media/publication/Turvallinen_tuotekehitys_Suomi_J003_2018.pdf

Agendium Ltd n.d. Projektityön digiopas. Viitattu 5.1.2021 https://www.agendium.com/projektinhallinta/miksi-projektityokaluja-tarvitaan