Petteri Pyrrö

OHJELMISTOTUOTANTOPROSESSIN KEHITTÄMINEN INFOMATES SOFTWARE TECHNOLOGIES OY:SSÄ

Insinöörityö
Kajaanin ammattikorkeakoulu
Tekniikan ja liikenteen ala
Tietotekniikan koulutusohjelma
Kevät 2001

KAJAANIN AMMATTIKORKEAKOULU

INSINÖÖRITYÖ TIIVISTELMÄ

Osasto	Koulutusohjelma
Tekniikka ja liikenne	Tietotekniikka

Tekijä
Petteri Pyrrö

Työn nimi

Ohjelmistotuotantoprosessin kehittäminen Infomates Software Technologies Oy:ssä

Vaihtoehtoiset ammattiopinnot	Ohjaaja
	Jukka Heino
Aika Kevät 2001	Sivumäärä 62 + 47

Tiivistelmä

Tämän insinöörityön tarkoituksena on kehittää ohjelmistotuotantoprosessia Infomates Software Technologies Oy:ssä luomalla prosessille uusi rakenne ja toimintatavat.

Insinöörityön teoriaosassa käsitellään yleisesti ohjelmistotuotannon osa-alueita ja prosessimalleja. Malleja esitellään kolme: vesiputousmalli, spiraalimalli ja Rational Unified Process eli RUP-malli. Prosessin laatutason määrityksessä sivutaan CMM-kypsyysmallia (Capability Maturity Model). Lisäksi kerrotaan inkrementaalisen ja iteratiivisen prosessin eroavaisuuksista sekä erilaisten tuotantotiimien muodostamisesta.

Esittelyosassa esitellään Infomates Software Technologies Oy:n taustatiedot ja kuvataan vanha prosessi yleispiirteissään. Lisäksi kuvataan yhtiön liiketoiminnan erikoispiirteitä ja käytännön kokemuksia ohjelmistoprojekteista.

Jälkimmäisessä osassa määritetään vanhan prosessin kriittisimmät kehityskohteet ja luodaan uusi prosessimalli yhdistäen eri prosessimallien käytäntöjä yhtiön vanhan prosessin hyvien ominaisuuksien kanssa. Tiimirakenteita korjataan prosessin edellyttämään muotoon. Prosessin kehittyminen osoitetaan CMM-avainominaisuuksien avulla vertaamalla vanhan ja uuden prosessin avainominaisuuksia keskenään. Kehitystyö on vertauksen perusteella onnistunut: viemällä tässä työssä kuvatun mallin yhtiössä käytäntöön, nousee prosessin kypsyysaste CMM-laatutasossa ylöspäin yhden tason verran.

Luottamuksellinen 6.4.2001 – 5.4.2003
Kyllä

Hakusanat
Ohjelmistotuotanto, tuotantoprosessi

Säilytyspaikka
Opinnäytteen ohjaajan työhuone

KAJAANI OLYTECHNIC

ABSTRACT
FINAL YEAR PROJECT

Faculty	Degree programme
Faculty of Engineering	Information Technology

Author
Petteri Pyrrö

Title

Improving the Software Development Process in Infomates Software Technologies Ltd.

Optional professional studies	Instructor(s) / Supervisor(s)
	Jukka Heino
Date Spring 2001	Total number of pages 62 + 47

Abstract

The purpose of this final paper was to improve the software development process in Infomates Software Technologies Ltd. By developing a new process structure and activities.

The theory chapter covers the basics of software engineering and various software process models. The models presented are the Waterfall Model, the Spiral Model and the Rational Unified Process (RUP). The process quality issues are visited with the Capability Maturity Model (CMM). In addition, the differences between incremental and iterative development are presented, and a method for building developer teams is discussed.

The middle part presents the basic background information about Infomates Software Technologies Ltd. and the previous software process used in the company. The special features of the company's software business and some experiences from real-life software projects are also presented.

In the latter part of the project the most critical factors of the previous process in need of improving are defined. A new process is formed by combining features from the process models presented in the theory chapter with some good features from the previous process. The team structures are rebuilt in new form to comply with the new process structure. The improvement accomplished is shown by comparing the CMM Key Process Areas (KPAs) of the previous process with the new software process. Putting the new process described in the project in practise will increase the process maturity level one step higher.

Confidential 6.4.2001 – 5.4.2003
Yes

Keywords
Software engineering, software process, development process

Deposited at
Kajaani Polytechnic Library

Alkusanat

Työskenneltyäni vuoden ajan osittain opiskelujen ohella Infomates Oy:n tuotekehitysjohtajana, aiheen valinta insinöörityölle tuntui hyvinkin luontevalta. Varsinkin, kun työtehtäviin kuuluvat erilaiset kehitystehtävät ja ohjelmistoprojektien läpivienti. Kuluneen vuoden tuomat käytännön kokemukset ohjelmistoalan liiketoiminnasta, tekniikasta ja projekteista johtivat haluun kehittää kokonaisuutta paremman hallittavuuden aikaansaamiseksi. Myös aiemmat tapahtumat ja työtehtävät opiskelujen yhteydessä ovat antaneet näkökulmaa ja lisänneet itsetuntoa kehittämistehtävissä onnistumisesta; projektitoiminnasta ja johtamisesta olen saanut kokemusta toimiessani hallituksen puheenjohtajana ammattikorkeakoulun opiskelijayhdistyksessä ja Kajability Oy:ssä, teknistä ja asiantuntijanäkökulmaa sain kosolti Nokialta toimiessani siellä määrittelytehtävissä tuotekehitysyksikössä kesällä 1999.

Insinöörityön tekeminen ei silti ole ollut mikään helppo tehtävä. Aihepiirin rajaaminen, olennaisen löytäminen ja sovittaminen Infomates Oy:n tarpeisiin on ollut haastava tehtävä. Koko prosessin muodostaminen ja käytäntöön saaminen kestää kauemmin kuin sen kirjallinen kuvaaminen, ja varsinainen käytännön työ alkaakin tämän työn valmistumisen jälkeen. Pohjatyöllä on kuitenkin ratkaiseva merkitys prosessin kehittämisen kannalta.

Haluaisin esittää kiitokseni yliopettaja Jukka Heinolle insinöörityön ohjauksesta sekä sisältöön liittyvistä huomioista. Lisäksi haluan kiittää Infomates Oy:n suunnittelujohtaja Seppo Liuskia, samoin kuin toimitusjohtaja Kimmo Niskasta sekä ohjelmistoarkkitehti Aleksi Kalliota, heidän arvokkaista kommenteistaan ja näkemyksistään koko insinöörityön tuotantoprosessin ajalta.

Kajaanissa 5.4.2001

Petteri Pyrrö

SISÄLLYSLUETTELO

1 JOHDANTO

Tämän insinöörityön lähtökohtana oli tarve kehittää Infomates Oy:n ohjelmisto-tuotantoprosessia siten, että se vastaisi nykyaikaisia oliopohjaisen tuotannon vaatimuksia, mukautuisi projektin aikana tapahtuviin muutoksiin ja olisi lisäksi projektin koon mukaan skaalautuva. Nämä vaatimukset asettavat prosessin kehittämiselle suuria haasteita, ja vaatimuksien perusteella on selvää, että prosessin täytyy olla hyvinkin dynaaminen. Tavoitteena on muodostaa nämä kriteerit huomioiva puitekehys ohjelmistotuotanto-prosessille.

Koska Infomates Oy on vielä nuori yritys, toimintatavat eivät ole vielä täysin vakiintuneet. Lisäksi työntekijät ovat lähes kaikki alle kolme vuotta työkokemusta omaavia, joten henkilöstö ei ole ehtinyt ”aikaisemmassa elämässään” urautua mihinkään tiettyyn toimintamalliin. Toisaalta ohjelmistoalalle tyypillinen jatkuva muutos on tullut useimmille jo tutuksi. Suurta vastarintaa ei siten ole, päinvastoin; työntekijät ovat motivoituneita olosuhteiden ja toimintatapojen kehittämiseen. Ajankohta on siis oivallinen uusien toiminta-tapojen muodostamiselle ja läpiviennille sekä aiemmissa projekteissa käytettyjen menetel-mien arvioinnille.

Prosessin toimintatapojen muodostamisessa ei kuitenkaan pyritä urauttamaan tai rutinoimaan työmenetelmiä ja käytäntöjä enempää kuin on välttämätöntä. Oliopohjainen ohjelmistotuotanto on luonteeltaan iteratiivista, asiakasprojekteissa vaatimusmäärittelyt muuttuvat monesti projektin kuluessa, teknologian kehitys tuo uusia huomioitavia asioita prosessiin jne. Tuotantoprosessissa on kuitenkin oltava puitekehykset käytännöille, toimintatavoille ja menetelmille, jotta projekteja voidaan ylipäätään hallinnoida ja viedä läpi järkevästi sekä verrata tehtyjä projekteja yhteismitallisesti keskenään ja muodostaa mitattavia laatukriteerejä.

Ohjelmistotuotannon prosessien luomisen ja kehittämisen kanssa ovat painiskelleet pitkään monet ohjelmistoyritykset, yliopistoissa alaa on tutkittu ja kehitetty jo vuosia, jopa vuosikymmeniä. Nuoren yrityksen onkin hyvä ottaa oppia muiden kokemuksista. Lähde-materiaalin valinnassa on pyritty ottamaan huomioon kirjoittajien käytännön kokemus

ohjelmistoalalta ja toisaalta akateeminen, tutkimustuloksiin sekä koestettuihin teorioihin perustuva näkökulma.

Ohjelmistotuotanto on kokonaisuutena huomattavan laaja käsite ja kattaa varsinaisen teknisen tuotannon lisäksi paljon kaupallisia ja henkilöjohtamiseen liittyviä asioita. Tässä insinöörityössä on kuitenkin pyritty lähestymään ohjelmistotuotantoprosessia enemmän tuotekehityksen näkökulmasta, talouteen ja henkilöstöjohtamiseen liittyviä seikkoja on sivuttu vain silloin kun ne olennaisesti liittyvät ohjelmistotuotantoprosessiin ja sen eri vaiheisiin. Työn tarkoitus on luoda Infomates Oy:lle ohjelmistotuotantoprosessille uudet perusteet sekä toimia tuotannon johtohenkilöstön ja hallinnon tukimateriaalina.

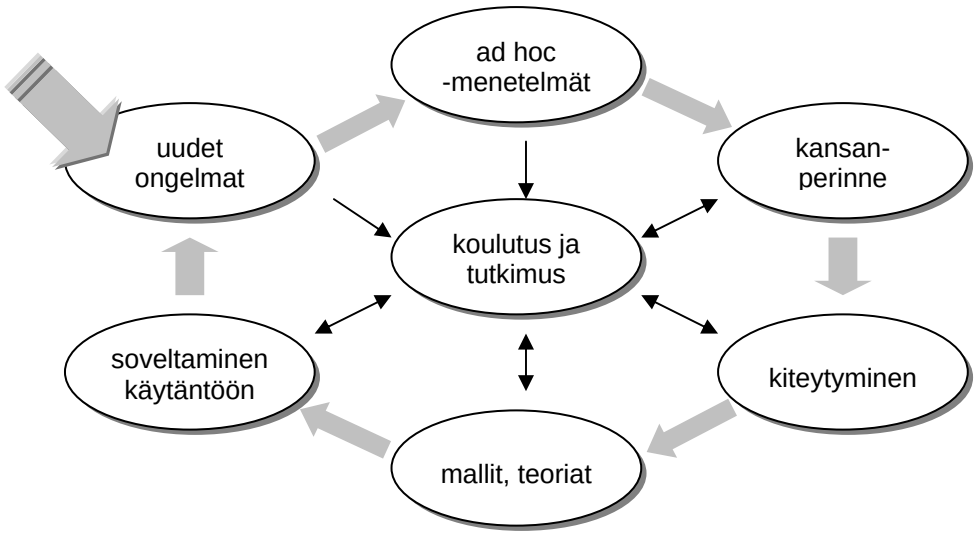
2 OHJELMISTOTUOTANNON TEORIAA

Kehittämistyön pohjustukseksi käydään aluksi läpi ohjelmistotuotannon perusteita sekä erilaisia ohjelmistotuotannon prosessimalleja. Lisäksi kuvataan inkrementaalisen ja iteratiivisen prosessin eroja sekä tiiminmuodostusmetodologia.

Käytetty terminologia voi osittain olla lukijalle uutta tai ainakin vaikuttaa siltä, koska lähdemateriaali oli enimmäkseen englanninkielistä, eikä kaikille termeille ole olemassa suoraa kotimaista vastinetta. Tällaisissa tapauksissa on pyritty tekemään asiaa kuvaava vapaa suomennos tai on käytetty pelkästään englanninkielistä nimikettä. Suomenkielisen termin tai suomennoksen ohessa on mainittu alkuperäinen englanninkielinen termi tärkeimpien termien osalta. Suomenkielisessäkin lähdemateriaalissa englanninkieliset vastineet oli mainittu. Tällä pyritään myös varmistamaan, että ohjelmistotuotannon teoriaan englannin kielellä perehtynyt lukija saa yhteneväisen käsityksen kulloisestakin asiakohdasta.

2.1 Ohjelmistotuotannon kehityksestä

Ohjelmistojen toimivuus ja käytettävyys on ollut puheenaiheena eri medioissa ja pöytäkeskusteluissa jo siitä lähtien, kun ensimmäiset ohjelmistosovellukset tehtiin. Ihmetellään usein, miksei ohjelmistoja voitaisi suunnitella ja toteuttaa yhtä huolellisesti ja rutiininomaisesti kuin esimerkiksi rakennuksia. Eräs syy lienee alan suhteellinen nuoruus (esimerkiksi rakennusalaan verrattuna). Mary Shaw [1] on todennut, että tekniikan alasta riippumatta kehitys noudattaa kuvan 1 mukaista kehämäistä mallia (suomennos ja selitteet ovat Haikalan ja Märijärven [2] kirjasta).



Kuva 1: Tekniikan yleistä kehittymistä kuvaava kehä [1]

Lähtökohtana tekniikan kehittymiselle ovat käytännön ongelmat, joihin etsitään ratkaisuja.

- Aluksi ongelman ratkaisuksi kelpaa mikä tahansa jo käytännössä koestettu toimiva tapa, eli sovelletaan ns. *ad hoc* -menetelmiä.
- Vähitellen löydetään ratkaisuja, jotka toimivat useammassa kuin yhdessä tapauksessa – näistä syntyy eräänlaista ”kansanperinnettä”.
- Kun kansanperinteenä välittyvä tietotaito kehittyy systemaattisemmaksi, se kiteytetään heuristiikoiksi ja työskentelymenetelmiksi.
- Vähitellen nämä menetelmät kehittyvät riittävän järjestelmällisiksi tukeakseen malleja ja teorioita niihin liittyvine formaaleine menetelmineen.
- Kun syntyneitä malleja ja teorioita sitten käytetään uusiin, entistä vaativampiin sovelluksiin, löytyy uusia ongelmia ja kokonaisia sovellusalueita, joille ei ole riittäviä ratkaisumalleja. Ympyrä siis sulkeutuu: on jälleen lähdettävä liikkeelle ad hoc –menetelmin.

Ohjelmistotuotantoon ei ole vielä syntynyt kovinkaan paljon puhtaasti alalle ominaisia ad hoc -menetelmiä, käytännöt ovat olleet muilta aloilta johdettuja – lähinnä erilaisia teollisuusalojen prosessimenetelmiä. Niitä on sitten sovellettu kulloiseenkin tilanteeseen ja tarpeeseen.

Brooks [3] kiteyttää paljon referoidussa artikkelissaan tuotantoprosessin eri vaiheiden merkityksen seuraavasti:

> Uskoakseni ohjelmiston tekemisen vaikeimmat osa-alueet ovat kokonaisuuden käsitteellisen rakenteen määrittely, suunnittelu ja testaus, ei varsinainen toteutustyö ja toteutustuloksen laadun testaus. Teemme itse asiassa silti syntaksivirheitä (ohjelmakoodia kirjoitettaessa); mutta ne ovat vain pisara meressä verrattuna useimpien järjestelmien käsitteellisiin virheisiin. *(Suomennos kirjoittajan).*

Brooksin kiteytys kertoo paljon prosessin alussa tehtävän työn merkityksestä ja siitä, minkälaiseen osaamiseen ohjelmistotuotannossa tulee ohjelmointiosaamisen lisäksi merkittävästi panostaa. Myöhemmissä luvuissa kuvataan erilaisia tuotantoprosesseja, niiden eri vaiheet ja vaiheiden sisältö – tämä kiteytys on hyvä pitää mielessä verrattaessa erilaisia prosessimenetelmiä toisiinsa.

2.1.1 Ohjelmistotuotannon erityispiirteitä

Jos ohjelmistotuotannon jonkinasteiselle kehittymättömyydelle löytyykin selitys alan nuoresta iästä, on ohjelmistoilla myös muutamia muista tekniikan aloista poikkeavia erityispiirteitä. Brooks mainitsee tällaisiksi ohjelmistojen luontaisen monimutkaisuuden (complexity), näkymättömyyden (invisibility), muunnettavuuden (changeability) ja yhdenmukaisuuden (conformity). Haikala ja Märijärvi lisäävät erityispiirteisiin vielä ainutkertaisuuden (uniqueness) ja skaalautumattomuuden (unscalability).

Monimutkaisuus. Ohjelmistotuotteet ovat luonnostaan monimutkaisia. Koska ongelmat joita ohjelmistot ratkovat ovat monimutkaisia, myös ohjelmistot ovat pakostakin monimutkaisia. Hyvälläkään ohjelmistosuunnittelulla monimutkaisuutta ei siis pystytä poistamaan – huonolla suunnittelulla sitä kylläkin pystyy lisäämään. Ohjelmistosuunnittelun keskeisin periaate onkin monimutkaisuuden minimointi ja hallinta pitämällä eri komponenttien väliset rajapinnat mahdollisimman vähäisinä, selkeinä ja yksinkertaisina. Ohjelmistotuotteen monimutkaisuus liittyy usein sen kokoon: suuret järjestelmät ovat yleensä myös monimutkaisia.

Näkymättömyys. Projektinhallinnan kannalta ohjelmistotyön hankalin ongelma on näkymättömyys: ohjelmistotyön keskeneräisistä tuotuoksista on hankala sanoa, mikä projektin valmiusaste on. Projekti saattaa esimerkiksi olla hyvin aikataulussa, vaikka puolet ajasta on jo käytetty, eikä riviäkään ohjelmakoodia ole vielä kirjoitettu. Vastaavasti projekti saattaa olla pahasti myöhässä, vaikka vasta puolet ajasta on käytetty ja järjestelmätestaus on jo alkanut. Projektinhallinnassa onkin tärkeää pyrkiä tekemään projektin todellinen eteneminen näkyväksi mm. erilaisten välietappien ja laadunvarmistustoimenpiteiden avulla.

Muunnettavuus. Ohjelmistoprojekteille on tyypillistä, että ohjelmistolle asetetut vaatimukset tarkentuvat, lisääntyvät tai muuttuvat projektin aikana. Myös kustannustekijät saattavat vaikuttaa muutostarpeisiin: ohjelmistotuotteen hinnan määrää yleensä tuote-kehityskustannukset – uusien ohjelmakopioiden tuottaminen on niihin verrattuna mitätön kustannustekijä.

Tyypillisesti myös ohjelmistosta löytyvät virheet, toimintaympäristössä tapahtuvat muutokset ja ympäristön muuttuvat vaatimukset aiheuttavat ohjelmistolle muutospaineita koko sen elinkaaren ajan. Huolellisella määrittelyllä ja suunnittelulla muutostarpeita voidaan vähentää, mutta tuskin juuri poistaa. Muun muassa tästä syystä ohjelmistojen ylläpidettävyys on tärkeä ominaisuus.

Yhdenmukaisuus. Ohjelmiston toteuttamistavoille ei ole mitään yhdenmukaista säännöstöä tai lakeja, on olemassa vain hyviä tapoja. Käytännössä ohjelmointitapoja on yhtä paljon kuin on ohjelmoijia. Ohjelmointikielet antavat ohjelmoijille runsaasti vapauksia erilaisten toteutustapojen käyttämiseen, jopa samaa ohjelmointikieltä käyttävät ohjelmoijat tuskin koskaan ratkaisevat annettua tehtävää tai ongelmaa samalla tavalla. Tästä seuraa, että eri ohjelmistojen osat koostuvat eri tavoilla toteutetuista komponenteista. Mitä useampi ihminen tekee projektissa ohjelmistokomponentteja, sen monimutkaisemmaksi järjestelmä ja sen hallinta käyvät.

Yhdenmukaisuutta pyritään aikaansaamaan käyttämällä ohjelmiston suunnittelussa ja toteutuksessa tarkoin ennalta määriteltyä ja sovittua arkkitehtuuria, työkaluja ja työ-menetelmiä. Lisäksi on sovittava lukuisista esitetyistä hyvistä toteutustavoista sopivin. Yrityksessä luodun ohjelmistotuotantoprosessin tai laatukäsikirjan tulee määrittää kaikki

nämä seikat, jotta yhdenmukaisella toimintatavalla voitaisiin toteuttaa edes jossain määrin yhdenmukaista jälkeä.

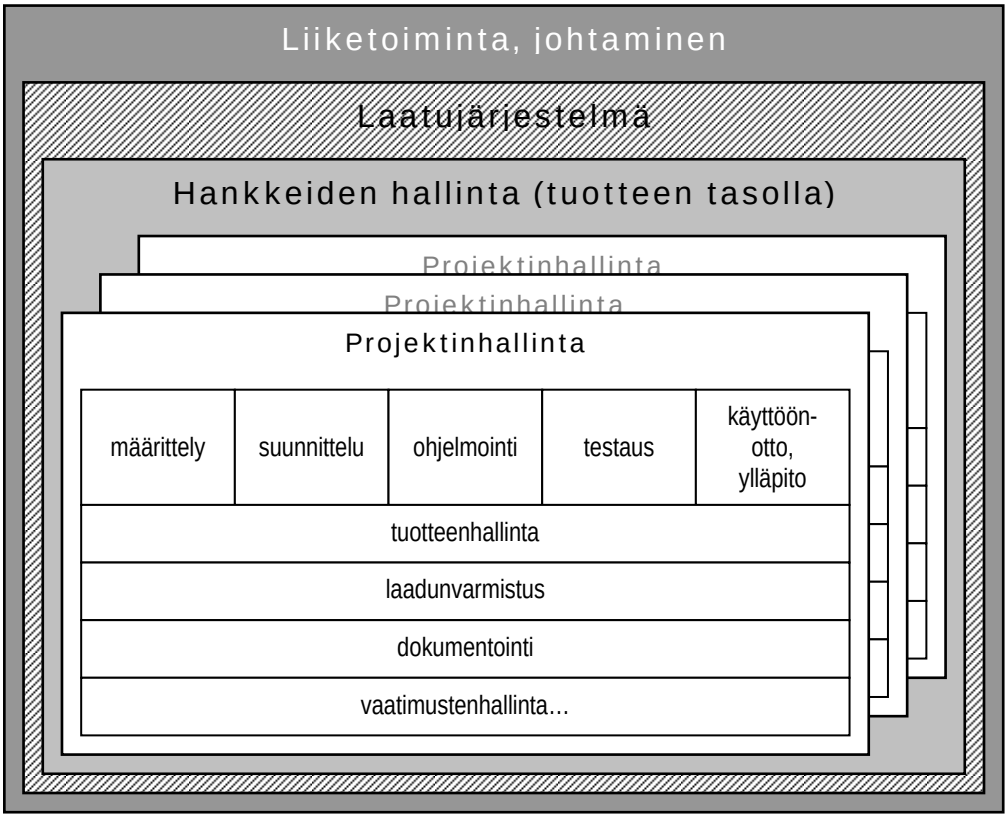
Ainutkertaisuus. Ohjelmistoprojekteista erityisesti asiakaskohtaiset, räätälöidyt sovellukset ovat paras tyyppiesimerkki – samantapaista ohjelmaa ei ehkä ole tehty koskaan aikaisemmin, eikä ehkä tehdä myöhemminkään. Ohjelmistoyrityksen sisäisissä tuotekehitysprojekteissa on kuitenkin usein asetettu tavoitteeksi toteuttaa osa komponenteista uudelleenkäytettäviksi tai yhteensopiviksi muiden tuoteperheen ohjelmistojen kanssa. Turhan usein tämä tavoite jää taka-alalle, koska uudelleenkäytettävyyttä ei osata suunnitteluvaiheessa ottaa riittävästi huomioon. Vasta oliopohjaiset menetelmät ovat tuoneet mukanaan todellisen mahdollisuuden komponenttien uudelleenkäytettävyyden suunnittelulle ja toteutukselle.

Skaalautumattomuus. Aiemmin hyväksi havaitut menetelmät eivät välttämättä toimi projektin koon kasvaessa. Pienehkön ohjelman tapauksessa jokainen projektiin osallistuvista omaa yleensä riittävän tarkan käsityksen koko käsillä olevasta ohjelmistosta, ja projekti saattaa onnistua erinomaisesti ilman dokumentaatiota, projektisuunnitelmaa ja kunnollista projektinohjausta. Ongelmien ilmaantuessa projekti saattaa olla pelastettavissa yhden projektitiimiin kuuluvan ”sankarisuorituksella”. Menetelmät tulee kuitenkin valita siten, että ne sopivat kulloinkin kyseessä olevan projektin kokoon. Tämä asettaa haasteita henkilöstön ammattitaidolle – yrityksen johdon ”työkalupakista” tulisi löytyä riittävä määrä sopivia työkaluja erikokoisiin tapauksiin, ja lisäksi pitäisi vielä osata käyttää jokaista työkalua sekä siirtää osaamista organisaation eri kerroksiin.

Myös ohjelmistokomponenttien skaalautuvuus on suuri haaste. Tämä liittyy ainutkertaisuuteen; kun suunnitellaan uudelleenkäytettäviä komponentteja, joudutaan miettimään, kuinka paljon voidaan havaita ja ottaa huomioon erilaisia uudelleenkäyttökohteita.

2.2 Ohjelmistotuotannon osa-alueet

Ohjelmistotuotanto jakautuu eri osa-alueisiin esimerkiksi kuvassa 2 esitetyllä tavalla. Lähestymistapa on perinteinen, ja ehkä erottelee eri osa-alueita liikaa toisistaan, mutta antaa kuitenkin ohjelmistotuotannon tärkeimmistä aihealueista kokonaiskuvan.



Kuva 2: Ohjelmistotuotannon osa-alueet [2]

Ohjelmistot toteutetaan projekteissa, jotka etenevät tiettyjen vaiheiden mukaisesti. Projektin vaiheet ovat yleensä ainakin *määrittely, suunnittelu, ohjelmointi* eli *toteutus, testaus* sekä *käyttöönotto ja ylläpito*. Projektiin liittyy myös koko sen elinkaaren ajan kestäviä tukitoimintoja, joista tärkeimpiä ovat tuotteenhallinta, laadunvarmistus ja dokumentointi. Tukitoimintojen määrä ja laajuus riippuu projektin koosta, isoissa projekteissa erilaisia tukitoimintoja on enemmän (kuten esimerkiksi vaatimustenhallinta ja riskienhallinta), pienemmissä ne on sisällytetty edellä mainittuun kolmeen tärkeimpään tukitoimintoon. Prosessin ja projektien vaihejakomalleja käsitellään tarkemmin jäljempänä.

Ohjelmistotuotantoprosessin tulisi luoda toimintatavat kuvan 2 mukaiselle kokonaisuudelle, painopisteen ollessa projektimenetelmien ja -käytäntöjen sekä tukitoimintojen ohjeistuksessa siten, että hankkeiden hallinta, laatunäkökohdat ja yrityksen liiketoiminta on otettu huomioon. Prosessin yksi merkittävä dimensio on yrityksen organisaatio ja sen adaptoiminen kokonaisuutta tukevaksi. Käytännössä tämä edellyttää linjaorganisaation ja projektiorganisaation osittaista yhteismallia. Erilaisia organisaatiomalleja on käsitelty jäljempänä.

Perinteisessä lähestymistavassa on myös ajateltu, että yrityksen toimintaa (ja tuotantoa) ohjaa laatujärjestelmä, joka määrittelee yrityksen toimintatavat ja toimintaprosessit. Tällä on haluttu korostaa laatuajattelun kattavuutta ja ohjaavaa roolia koko yrityksen toimintatavoissa. Tässä insinöörityössä lähestytään asiaa kuitenkin prosessinäkökulmasta, jossa ajatellaan laatutekijöiden ja -mittareiden kuuluvan erottamattomaksi osaksi tuotantoprosessia. Nopeasti ajateltuna voisi ajatella, ettei näkökulmissa ole mitään eroa. Ohjelmistoyrityksessä voi kuitenkin olla vaikeaa muodostaa tuotantoprosessia jonkun laatukäsikirjan pohjalta (esimerkiksi ISO 9001 tms.) – on luonnollisempaa optimoida itse prosessi ohjelmistotuotannollisista lähtökohdista ja sisällyttää laatukriteerit sen osiksi. Tällöin myös vältytään kahdelta erilliseltä järjestelmältä.

2.2.1 Ohjelmiston elinkaari

Ohjelmistotuotteella on elinkaari, kuten millä tahansa kaupallisella tuotteella. Elinkaari koostuu karkeasti ottaen tuotekehityksestä, myynnistä (aluksi voimakkaasti kasvava, myöhemmin tasaantuva), ylläpidosta ja markkinoilta poistamisesta. Ohjelmistotuotannon kannalta prosessi koskettaa lähinnä tuotekehitystä: kun tuotteen myynti on voitu aloittaa, on tuote (tai sen ensimmäinen myyntikelpoinen versio) valmistunut, ja näin ollen kulkenut tuotantoprosessin päähän. Usein ohjelmistotuotteiden elinkaarta jatketaan uusilla ohjelmistoversioilla, joihin on korjattu mahdollisia virheitä, lisätty toiminnallisuuksia ja parannettu käytettävyyttä.

Ohjelmistojen monistaminen ei ole merkittävä tuotannollinen toimenpide, ohjelmisto-teollisuudessa monistaminen on usein ulkoistettu jollekin toiselle yritykselle. Ylläpito on

puolestaan ohjelmistoyrityksessä kategorioitu tuotekehitystoiminnan ulkopuoliseksi toiminnaksi, jolle on usein määritetty oma henkilöstö tai kokonainen liiketoimintayksikkö. Niinpä erilaiset ohjelmistoprosessimallit keskittyvätkin tuotekehitykseen ja sen eri vaiheisiin jättäen ylläpidon ja tuotteen poiston mallien ulkopuolelle.

2.3 Prosessi

Ohjelmistoyrityksen liiketoimintaan sisältyy useita prosesseja. Esimerkiksi markkinointi, myynti, tuotanto ja koulutus muodostavat kukin oman prosessinsa, joilla kylläkin voi olla monia yhtymäkohtia. Walker Royce [4] on erotellut prosessit kolmeen tasoon seuraavasti:

- *Metaprosessi* käsittää yrityksen korkean tason sovitut toimintatavat, menetelmät ja käytännöt, joilla tähdätään ohjelmistokeskeiseen liiketoimintaan. Metaprosessin kohteita ovat yrityksen talous, pitkän tähtäimen strategiat ja tuotetun ohjelmiston ROI (Return On Investment) eli ohjelmistotuotantoon sijoitetun pääoman tuotto.
- *Makroprosessi* käsittää ohjelmistoprojektin sovitut toimintatavat, menetelmät ja käytännöt, joiden avulla ja joita noudattamalla tuotetaan tietyillä kustannus-, aikataulu- ja laatukriteereillä valmis ohjelmistotuote. Makroprosessin tarkoitus on luoda rajattu instanssi metaprosessista rajatulle projektille.
- *Mikroprosessi* käsittää projektitiimin sovitut toimintatavat, menetelmät ja käytännöt, joita noudattamalla ohjelmistoprojektissa saadaan konkreettisia tuotoksia. Mikroprosessin tarkoitus on saavuttaa projektin vaiheille asetetut tavoitteet riittävin toiminnallisuuksin riittävän laadukkaasti niin nopeasti ja taloudellisesti kuin on käytännöllistä.

Vaikkakin nämä kolme eri prosessitasoa ovatkin jonkin verran päällekkäisiä, niillä on erilaiset päämäärät, kohderyhmät, mittarit, huolenaiheet ja ajalliset kehykset (ks. taulukko 1).

Taulukko 1. Prosessin kolme tasoa attribuutteineen [4]

Attribuutti	Metaprosessi	Makroprosessi	Mikroprosessi
Aihe	Liiketoiminta-alue	Ohjelmistoprojekti	Iterointi (projektin vaiheessa)
Päämäärät	Liiketoiminnan tuottavuus Kilpailukyky	Projektin tuottavuus Riskienhallinta Projektin budjetti, aikataulu, laatu	Resurssienhallinta Riskiherkkyys Välietapin budjetti, aikataulu, laatu
Kohderyhmä	Osakkaat, asiakkaat Organisaation johto	Projektipäälliköt Vastaavat ohjelmisto-suunnittelijat	Aliprojektipäälliköt Ohjelmistosuunnittelijat
Mittarit	Projektin (onnistumisen) ennustettavuus Liikevaihto, markkinaosuus	Budjetin sisällä, aikataulussa Tärkeimmät välietapit onnistuneet Projektin työmäärä ja tuotokset	Budjetin sisällä, aikataulussa Tärkeimpien välietappien edistyminen Iterointiosuuksien työmäärä ja tuotokset
Huolenaihe	Byrokratia vs. standardisointi	Laatu vs. taloudellisuus	Sisältö vs. aikataulu
Aikakehys	6 – 12 kk	1 – useita vuosia	1 – 6 kk

On ehkä paikallaan valottaa projektin ja prosessin eroa sekä yhteneväisyyksiä. *Projekti* on kertaluonteinen tapahtuma(sarja), jolla on selkeä tavoite, tehtävät, aikataulu ja resurssit. *Prosessi* taas on jatkuvaluonteista, määrittää ja mallintaa sekä toimintatapoja että menetelmiä. Projektit viedään läpi jonkin prosessimallin mukaisesti, ja projektin eri vaiheissa voidaan hyödyntää erilaisiakin prosessimalleja.

Tässä insinöörityössä keskitytään erityisesti makroprosessin kehittämiseen. Roycen prosessitasot ovat paikoin toisiaan leikkaavia, ja joitakin meta- ja mikroprosessiin liittyviä asioita on sisällytetty silloin, kun niillä on tuotantoprosessin kehittämisen kannalta merkitystä.

2.3.1 Prosessin laatutason määrittäminen

Ohjelmistoprosessin laatutason voi määrittää Software Engineering Instituten (SEI) kehittämällä Capability Maturity Modelilla (CMM), josta tässä käytetään nimitystä

kypsyysmalli. Kypsyysmallissa on viisi tasoa, jotka kuvaavat ohjelmistotuotantoprosessin laatutasoa eli kypsyyttä (taulukko 2, koostettu ja suomennettu lähteestä [5]).

Taulukko 2. CMM-tasot ja avainominaisuudet [5]

CMM-taso	Avainominaisuudet
1. Alku. Ad hoc, jopa kaoottinen. Menestys riippuu täysin yksilösuorituksista.	• Ei ole
2. Toistettavissa. Perustason projektinhallintaa, jossa on määritetty sovelluksen toiminnallisuudet sekä projektin kustannukset ja aikataulu.	• Vaatimustenhallinta • Ohjelmistoprojektin suunnittelu • Ohjelmistoprojektin seuranta ja valvonta • Ohjelmistoalihankinnan hallinta • Tuotteiden laadunvarmistus • Muutoksienhallinta
3. Määritetty. Prosessin hallinnointi ja suunnittelu on dokumentoitu, standardisoitu ja integroitu. Kaikki projektit käyttävät hyväksyttyä ja räätälöityä versiota prosessista.	• Organisaation prosessisuuntautuneisuus • Organisaation prosessin määrittely • Koulutusohjelma • Tuotteenhallinta • Tuotesuunnittelu • Tiimityöskentely • Katselmoinnit
4. Hallittu. Prosessista ja tuotteiden laadusta kerätään tietoja eri mittareilla. Toimintatavat ovat itsestään selviä ja hallittuja.	• Kvantitatiivinen prosessinhallinta • Laadunhallinta
5. Optimoitavissa. Prosessia hiotaan ja parannetaan jatkuvasti eri laatumittareista saadun tiedon sekä uusin innovaatioiden ja tekniikoiden hyödyntämisen avulla.	• Virheiden estäminen • Teknologiamuutosten hallinta • Prosessimuutosten hallinta

Kypsyysmallissa voidaan havaita yhtenevyyttä aiemmin esitetyn tekniikan yleisen kehittymisen malliin (kuva 1, sivu 8). Ensimmäinen taso kuvaa tilannetta, jossa yrityksessä ei ole yhteistä, sovittua toimintatapaa, vaan kukin työntekijä tekee työtään omalla tavallaan, ilman yhteisiä säännöksiä tai seurantaa. Tällaisesta yrityksestä voisi olla ehkä esimerkkinä muutaman hengen taideteollinen yhteisö.

Toinen taso kuvastaa jossain määrin hallittua toimintaa, jossa on määritetty joitakin tuotannon kannalta kriittisimpiä asioita kuten tavoitteet, toiminnallisuudet, kustannukset, aikataulu ja resurssit. Useimmat aloittavat yritykset toimivat tällä tasolla.

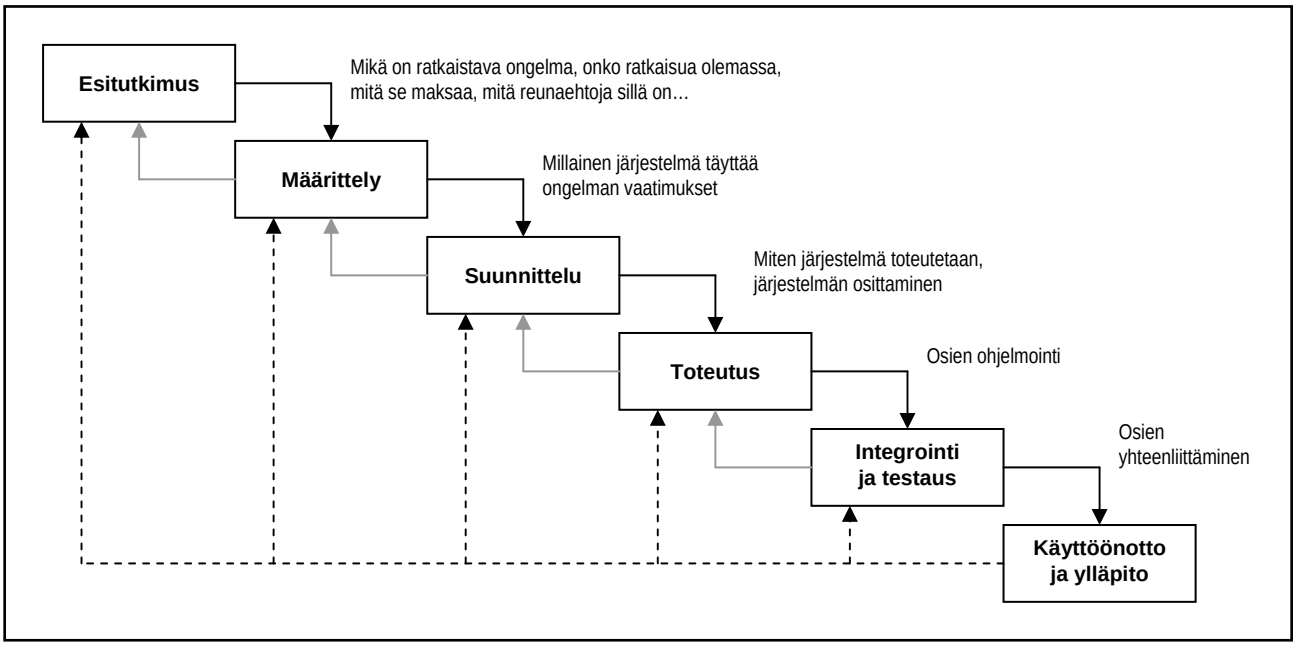
Kolmannessa tasossa toiminta on määritetty ja yhdenmukaistettu; lisäksi tuotantoa tukevia asioita on otettu huomioon, kuten esimerkiksi henkilöstön koulutus. Neljäs ja viides taso edustavat tilannetta, jossa yritys on jo toiminut pitkään kolmannella tasolla, hankkinut siitä kokemusta ja saadun kokemuksen avulla edelleen kehittänyt toimintatapojaan. Näillä tasoilla olevat yritykset ovat usein hankkineet laatusertifikaatin (esimerkiksi ISO 9001) ja ovat tyypillisesti toimineet alallaan useita vuosia tai vuosikymmeniä.

2.4 Prosessimalleja

Ohjelmistotuotannon juuret ovat Yhdysvalloissa, ja niinpä prosessimallit ovatkin kotoisin Yhdysvaltalaisten ohjelmistoalalla pitkään työskennelleiden insinöörien ja asiantuntijoiden piirustuspöydiltä. Lähtökohtana on ollut teollinen prosessi, josta on ajan saatossa kehitetty erityisesti ohjelmistotuotantoon paremmin soveltuvia menetelmiä. Ohjelmistotuotanto-prosessi kun poikkeaa paljolti liukuhihnaprosesseista, kuten luvussa 2.2.1 on todettu. Tässä luvussa esitellään vesiputousmalli eri variaatioineen, spiraalimalli ja RUP-malli.

2.4.1 Vesiputousmalli

Edellisessä luvussa esitetty ohjelmiston elinkaaren rajaus ulottuu Haikalan ja Märijärven [2] määritelmän mukaan pidemmälle: ohjelmiston *elinkaarella (life cycle)* tarkoitetaan aikaa, joka kuluu ohjelmiston kehittämisen aloittamisesta sen poistamiseen käytöstä. *Vaihejakomallilla* tarkoitetaan tapaa, jolla ohjelmiston kehitystyö tai koko elinkaari jaetaan vaiheisiin. Tavallisin vaihejakomalli on ns. vesiputousmalli (waterfall model), jonka eräs versio on esitetty kuvassa 3. Mallista on olemassa useita eri muunnelmia, mutta yleensä niistä voidaan erottaa ainakin määrittely-, suunnittelu- ja toteutusvaiheet. Määrittelyvaihetta edeltää usein esitutkimukseksi (feasibility study, preliminary analysis) tai tarve-kartoitukseksi (requirements study, requirements analysis) kutsuttu vaihe [2].



Kuva 3: Esimerkki vesiputousmallista [2]

Kaikkiin vaiheisiin liittyy laadunvarmistustoimenpiteitä, kuten tarkastuksia, katselmuksia ja testausta. Tarkastukset ja katselmukset ovat projektitiimin ja projektin muiden osapuolten keskinäisiä muodollisia kokouksia, ja niitä pidetään useampia projektin eri vaiheissa, tarpeen mukaan. Tarkastuksilla ja testauksella pyritään poistamaan toteutettavasta ohjelmistosta mahdollisia virheitä mahdollisimman varhaisessa vaiheessa sekä tarkistamaan, että eteneminen tapahtuu aikataulussa ja sisällöltään osapuolten kesken sovituissa puitteissa. Katselmuksia pidetään kunkin vaiheen päättyessä. Niissä todetaan projektin tilanne ja erityisesti se, että kaikki vaiheeseen liittyvät tavoitteet on saavutettu ja kaikki sovitut tuotokset on tuotettu (ns. *vaihetuotteet, artifacts, deliverables*) [2].

Vesiputousmallissa vaiheet ovat peräkkäisiä, eikä vaiheesta toiseen periaatteessa siirrytä, ennen kuin aiempi vaihe on suoritettu loppuun. Käytännössä projektin koosta ja resursseista riippuen joitakin vaiheita viedään läpi osittain rinnakkain. Vesiputousmallia sovellettaessa on tärkeää, että alussa tapahtuva määrittely- ja suunnittelutyö tehdään huolella, koska myöhemmässä vaiheessa tapahtuvat muutokset määrittelyissä voivat moninkertaistaa projektin läpivientiin kuluvan ajan ja viedä projektin taloudellisesti tappiolliseksi.

Roycen [4] mukaan viisi välttämätöntä edellytystä vesiputousmallin mukaisen prosessin toimivuudelle ovat

1. Tee määrittely- ja suunnittelutyö valmiiksi ennen toteutusta.
2. Ylläpidä kattavaa ja ajantasaista dokumentaatiota.
3. Tee homma toistamiseen, mikäli mahdollista.
4. Suunnittele, kontrolloi ja seuraa testausta.
5. Ota asiakas mukaan projektiin (involve the customer).

Esitutkimus (feasibility study, preliminary analysis) saa sysäyksen joko asiakaslähtöisestä tarpeesta tai yrityksen sisällä syntyneestä ideasta. Esitutkimuksessa kartoitetaan ja määritellään, mikä on ratkaistava ongelma (*ongelmakenttä* tai *-avaruus, problem space*), onko sille olemassa ratkaisua (*ratkaisukenttä* tai *-avaruus, solution space*), mitä toteuttaminen saa maksaa, ovatko resurssit riittävät jne. Esitutkimus vastaa kysymykseen miksi ohjelmisto tai järjestelmä tulisi tehdä (tai miksi sitä ei kannata tehdä), mutta ei ota kantaa sen toteuttamistapaan. Esitutkimus ei myöskään sisällä yksityiskohtaisia seikkoja, vaan esittää asiat laajemmalla tasolla ja pyrkii esittämään asiat yksinkertaisesti.

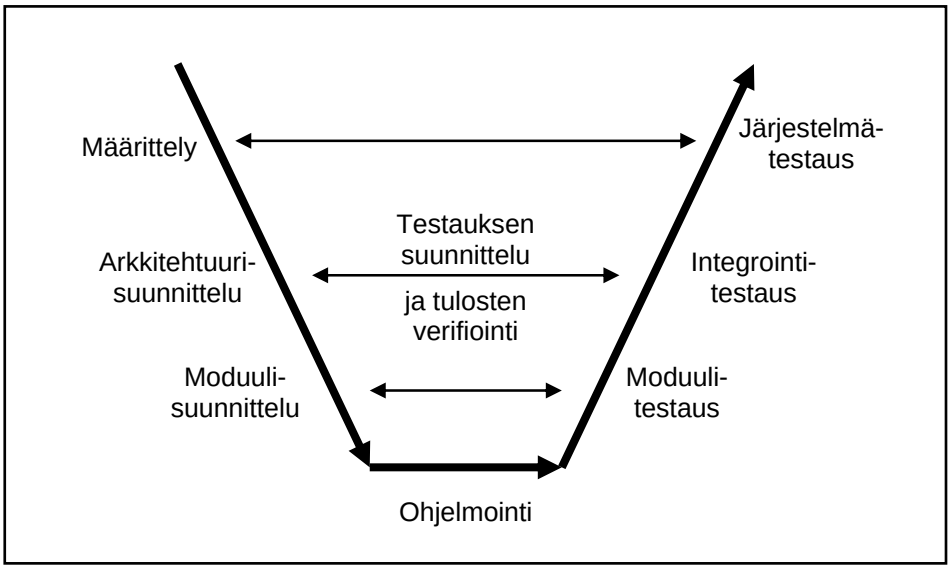
Määrittelyvaiheessa (requirements) kootaan asiakasvaatimukset ja analysoidaan niitä. Asiakasvaatimuksista johdetaan tekniset vaatimukset, jotka määrittelevät ohjelmiston teknisen toteutustavan. Termin *tekniset vaatimukset* lisäksi Haikala ja Märijärvi [2] kertovat käytettävän myös termejä *järjestelmävaatimukset, ohjelmistovaatimukset* ja *ominaisuudet (features)* enemmän tai vähemmän synonyymeinä. Määrittelyn tuloksena syntyy usein kaksi dokumenttia, vaatimusmäärittely (requirements specification, requirements analysis) ja toiminnallinen määrittely (functional specification, operational specification). Projektin koosta ja luonteesta riippuen nämä voidaan yhdistää yhdeksi dokumentiksi.

Suunnitteluvaiheessa (design, engineering) tehdään määrittelyissä kuvattujen toimintojen toteutuksen suunnittelu. Suunnitteluvaihe jaetaan usein kahteen (tai useampaan) tasoon, jälleen kerran projektin koon ja toteutettavan ohjelmiston monimutkaisuuden mukaan. Haikala ja Märijärvi nimittävät näitä kahta tasoa *arkkitehtuurisuunnitteluksi (architectural design)* ja *moduulisuunnitteluksi (module design, detailed design)*. Arkkitehtuurisuunnittelussa ohjelmisto jaetaan mahdollisimman itsenäisiin, toisistaan riippumattomiin osiin, moduuleihin. Moduulisuunnitteluvaiheessa suunnitellaan

jokaisen toteutettavan moduulin sisäinen rakenne. Oliopohjaisessa tuotannossa arkkitehtuurisuunnittelussa tehdään olioanalyysi, jossa määrittelyn pohjalta muodostetaan luokkakokonaisuudet ja erilaiset rajapinnat. Moduulisuunnittelussa vastaavasti suunnitellaan erilliset luokat ja rajapinnat tarkemmalla tasolla – sellaisella, että suunnitelman voi viedä tuotantoon *koodareille*, ohjelmiston toteuttajille. Oliosuuntautuneisuudesta on kerrottu luvussa 2.8.

Toteutusvaiheessa (implementation, coding, programming) kirjoitetaan ohjelmakoodi suunnittelussa tehtyjen mallien mukaan. Toteutusvaihe tuottaa ohjelmiston konkreettisesti näkyvän ja käytettävän osan eli itse ohjelmiston. Myös toteutus ositetaan usein toteutettavan ohjelmiston koon mukaisesti; yleensä erotellaan ainakin käyttöliittymän, sovelluslogiikan ja tietokantojen toteuttaminen omiksi osikseen.

Testauksen tarkoituksena on löytää ohjelmistosta virheitä. Testauksessa on useampia tasoja, joita kuvaa ns. V-malli (kuva 4). V-mallissa testaus jaetaan määrittelyn ja suunnittelun tasojen mukaisiin osiin: Määrittelytasoa vastaa järjestelmätestaus, arkkitehtuurisuunnittelutasoa integrointitestaus ja moduulisuunnittelua moduulitestaus. Moduulitestauksessa etsitään vikoja yksittäisistä moduuleista ja olioista, integrointitestauksessa moduulien yhteistoiminnasta ja järjestelmätestauksessa koko ohjelmiston eli järjestelmän toiminnoista ja suorituskyvystä.



Kuva 4: Testauksen V-malli [2]

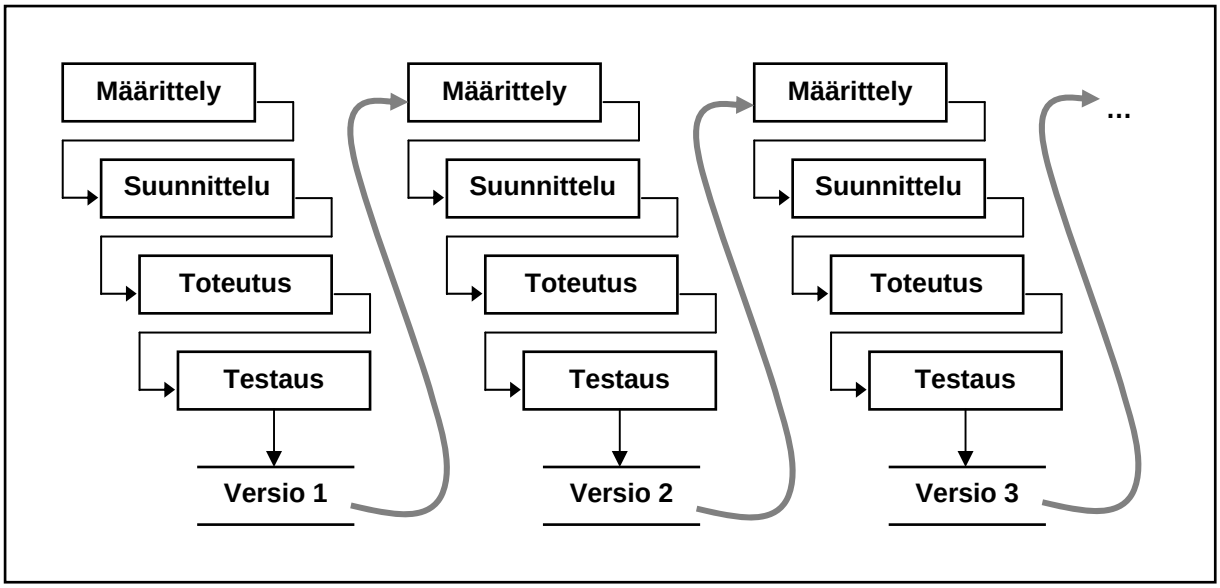
V-mallin mukaan kunkin tason testaus suunnitellaan jo määrittely- ja suunnittelutyön yhteydessä, eli järjestelmätestaus suunnitellaan osana ohjelmiston määrittelyä ja testauksessa verrataan valmista järjestelmää sen määrittelydokumentaatioon. Vastaavasti integrointitestaus suunnitellaan arkkitehtuurisuunnittelun yhteydessä ja moduulitestaus moduulisuunnittelun yhteydessä.

Ylläpito on Haikalan ja Märijärven mukaan asiakkaan ongelmien ratkomista, virheiden korjaamista, ohjelmiston muuttamista vaatimusten muuttuessa sekä uusien piirteiden lisäämistä. Ylläpito edellyttää palautekanavan luontia ohjelmiston toimittajan ja asiakkaan välille. Ylläpitoon siis kuuluu tuotetun ohjelmiston toimintakunnon ylläpitäminen, virheiden korjaaminen ja asiakkaiden avustaminen ongelmatilanteissa. Tämä vaihe tulkitaan usein myös itsenäiseksi, omaksi prosessikseen, koska sillä voi olla muista vaiheista poikkeavat resurssit ja käytännöt.

2.4.2 Evo-malli

Eräs sovellutus vesiputousmallista on ns. *evo-malli (evolutionary development)* ja eri protoilumallit. Evo-mallissa ensimmäisessä projektissa rakennetaan ydinjärjestelmä, jota sitten inkrementaalisesti eli rakentaen kehitetään edelleen. Malli siis muodostuu sarjasta toistuvia vesiputouksia, joista jokaisen tuloksena on uusilla ominaisuuksilla kasvatettu järjestelmä. Ohjelmistosta syntyy näin uusia versioita.

Käytännössä useimpien tuotekehityshankkeiden läpivienti tapahtuu evo-mallin mukaisesti. Uuden version kehitysprojekti käynnistyy seulomalla tärkeimmät työkohteet edellisestä versiosta saadun palautteen (virheet, toiveet uusista ominaisuuksista) perusteella. Nopeatahtisessa tuotekehityksessä version N+1 projekti käynnistyy jo ennen kuin versio N on valmis. Tällöin lähtökohdat projektille N+1 saadaan projektin N-1 palautteista sekä projektin N kokemuksista. Evo-mallin kaavio on kuvassa 5.

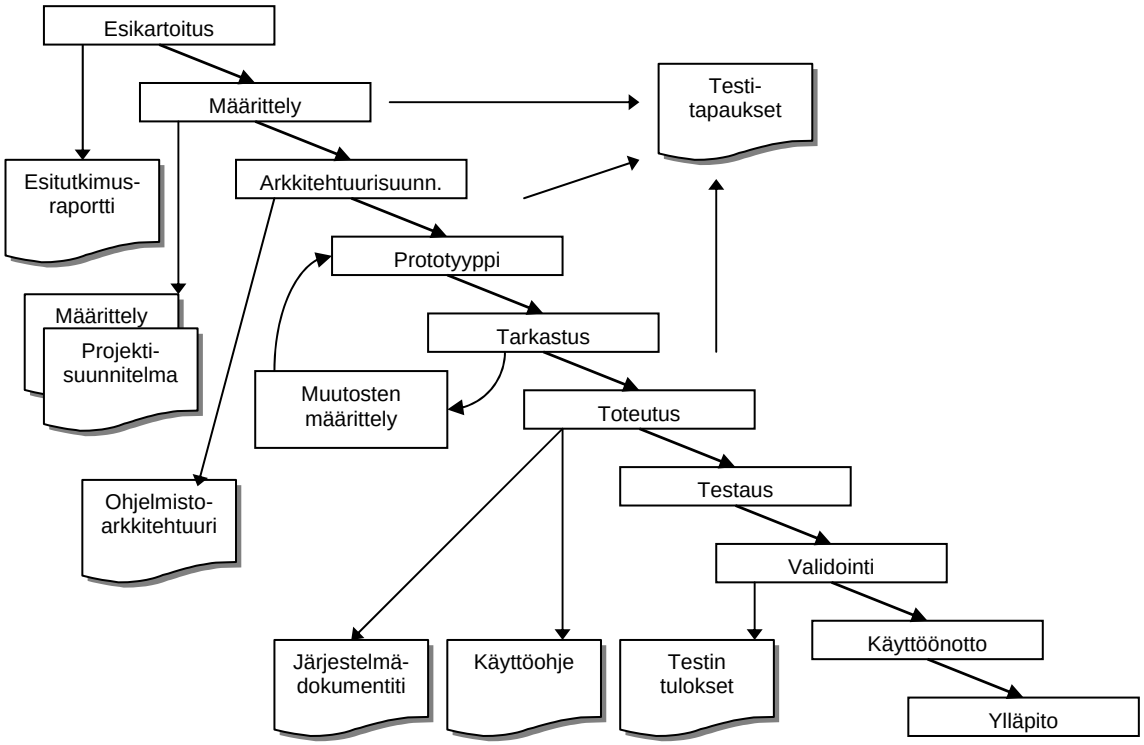


Kuva 5: Evo-malli [2]

Evo-mallia mukailee puolestaan ns. *protoilumalli*, jossa ensimmäisessä projektissa toteutetaan valmistettavan ohjelmiston prototyyppi. Prototyypit soveltuvat erityisesti uuden teknisen ratkaisun vaatimaan kokeilun tekemiseen tai tukemaan asiakasmäärittelytyötä. Toteutettavan ohjelmiston luonne, yrityksen valitsema toimintatapa ja prototyypin rakentamiseen käytettävä tekniikka määrittävät valmistuneen prototyypin pääkäyttövaihtoehdon. Pääkäyttövaihtoehdot ovat

1. Prototyypin valmistuttua sen perusteella määritellään toteutettava ohjelmisto, joka sitten toteutetaan alusta alkaen uudelleen.
2. Prototyyppi kehitetään valmiiksi ohjelmistoksi.

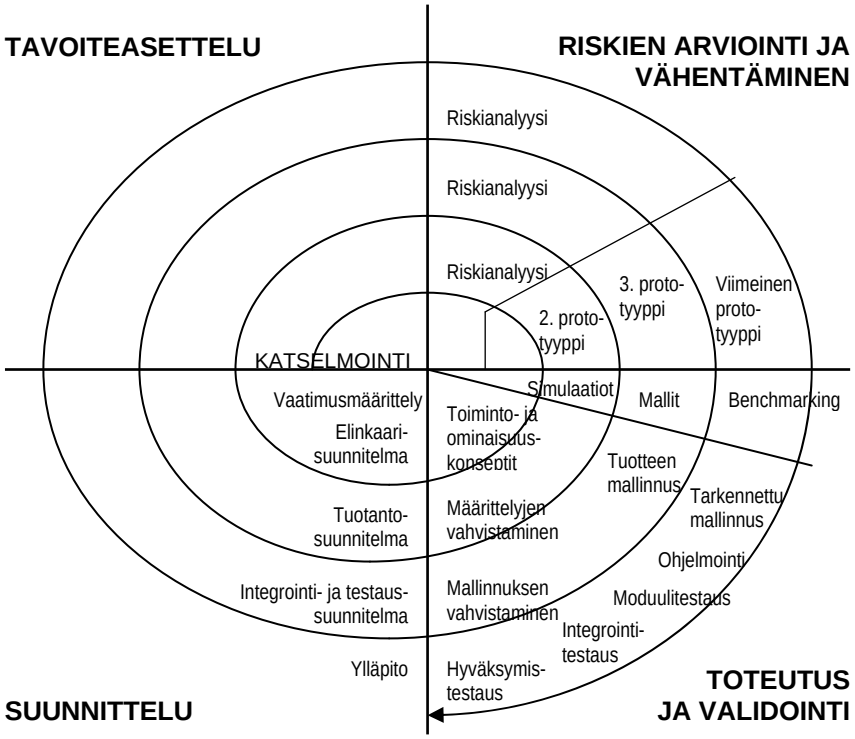
Kuvassa 5 esitetyn mallin voidaan ajatella kuvaavan ensimmäistä tapausta, kun versioiden tilalle ajatellaan ensin prototyyppiversiot ja sitten vasta varsinaisen tuotteen ensimmäinen versio. Toisen vaihtoehdon mukainen malli on kuvassa 6, jossa on myös esitetty prosessin aikana syntyvät kriittisimmät dokumentit. Protoilumalli pyrkii erityisesti varmistamaan ennen lopullista toteutusta, että ohjelmistossa on kaikki tarvittavat toiminnot ja että käyttöliittymä on asiakkaalle mieluinen.



Kuva 6: Esimerkki protoilumallista, mukailtu lähteestä [2]

2.4.3 Spiraalimalli

Spiraalimallin on luonut Barry Boehm vuonna 1986. Boehmin tavoitteena oli luoda prosessimalli, jossa riskienhallinta on huomioitu koko elinkaaren ajan. Se koostuu neljästä osasta: *tavoiteasettelu, riskien arviointi ja vähentäminen, toteutus ja validointi* sekä *suunnittelu* (kuva 7). Tässä mallissa prosessin katsotaan päättyvän käyttöönottoon; ylläpito on oma kokonaisuutensa, joka alkaa välittömästi tuotteen käyttöönoton jälkeen.



Kuva 7: Spiraalimalli [6]

Spiraalimallissa yksi vaihe edustaa yhtä täyttä 360° kierrosta, alkaen X-akselilta origon vasemmalta puolelta. Vaiheet etenevät origosta ulospäin spiraalikierteellä; kehän etäisyys origosta kuvaa kumuloituvia kustannuksia, joita projektin aikana syntyy tehdystä työstä ja siihen käytetystä ajasta.

Kukin vaihe alkaa aina **tavoiteasettelulla**, jota seuraa **riskien kartoitus ja vähentäminen** mm. erilaisten mallien ja prototyyppien avulla. Tätä seuraa **toteutusosa**, jossa ensimmäisillä kierroksilla tehdään käytännössä määrittely- ja suunnittelutyötä. Toteutusosan lopussa suoritetaan **validointi** eli vahvistetaan ja hyväksytään tulokset.

Ennen vaiheen päättymistä suunnitellaan seuraavan vaiheen sisältöä ja läpivientiä, jonka jälkeen kaikki vaiheen tuotokset katselmoidaan. Katselmoinnin tarkoituksena on vahvistaa kaikkien projektin osapuolien kesken edellisen vaiheen tulokset ja päättää etenemisestä seuraavaan vaiheeseen.

Malli on luonteeltaan inkrementaalinen eli kussakin vaiheessa sovellus kasvaa, siihen lisätään jotain uutta. Sovellusta rakennetaan lisäämällä siihen lisää ominaisuuksia ja toiminnallisuuksia vaihe vaiheelta. Inkrementaalisuudesta ja iteratiivisuudesta on kerrottu enemmän luvussa 2.6.

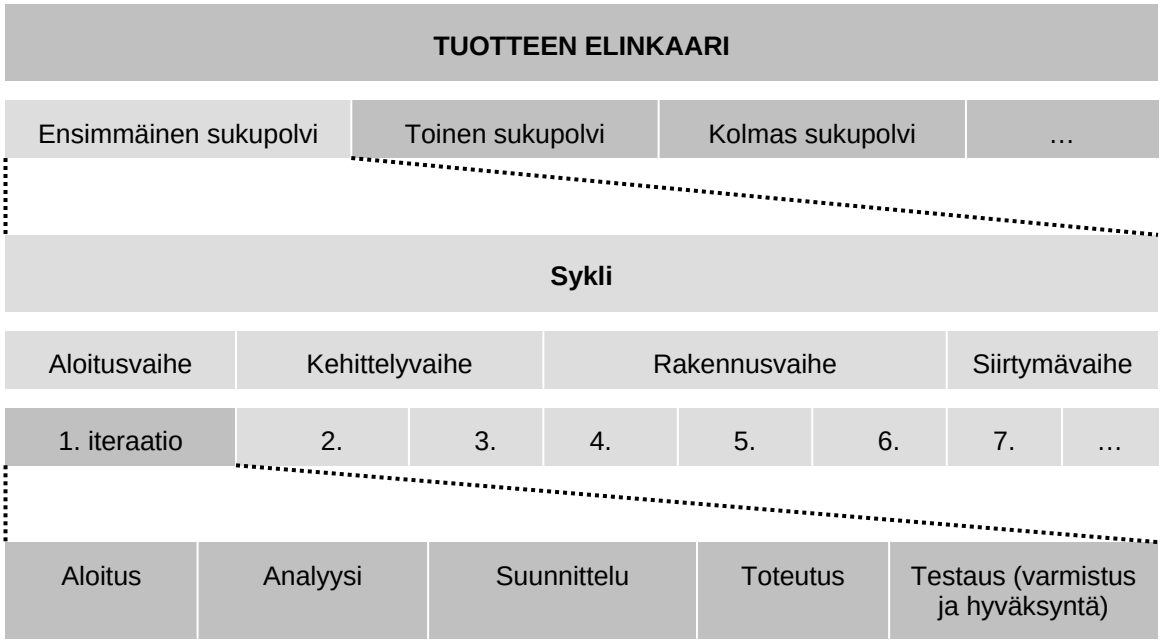
Spiraalimallin tarkoituksena ei ole määrittää kiinteää, yksityiskohtaista toimintaprosessia, vaan antaa kehys, jonka avulla voi projektikohtaisesti soveltaa muita prosessimalleja. Projektin koon ja erityisesti jäljellä olevien riskien mukaan voidaan toisen tai kolmannen kierroksen (määrittelyn tai suunnittelun vahvistamisen) jälkeen jatkaa Sommervillen [7] mukaan

- evo-mallilla, jos riskit ovat vielä suuria, tai
- vesiputousmallilla, jos jäljellä olevat riskitekijät voidaan ratkaista esimerkiksi prototyypin avulla, tai
- millä tahansa prosessimallilla tai niiden yhdistelmällä, jonka avulla jäljellä olevat haasteet ja riskit ratkotaan.

2.4.4 Rational Unified Process

Vesiputousmallia uudempi vaihejakomalli on *Rational Unified Process (RUP)*, jonka kehittäjiä ovat I. Jacobson, G. Booch ja J. Rumbaugh. Malli on nimetty kehittäjien työnantajan, Rational Software Corporationin mukaan. Koko prosessin perustana on iteratiivisuus.

RUP-mallissa elinkaaren ajatellaan koostuvan useista sykleistä, joista jokainen tuottaa sukupolven eli tuotteen uuden julkaistavan version. Jokainen sykli koostuu vaiheista ja jokainen vaihe puolestaan yhdestä tai useammasta iteraatiosta (kuva 8).



Kuva 8: RUP-malli [8]

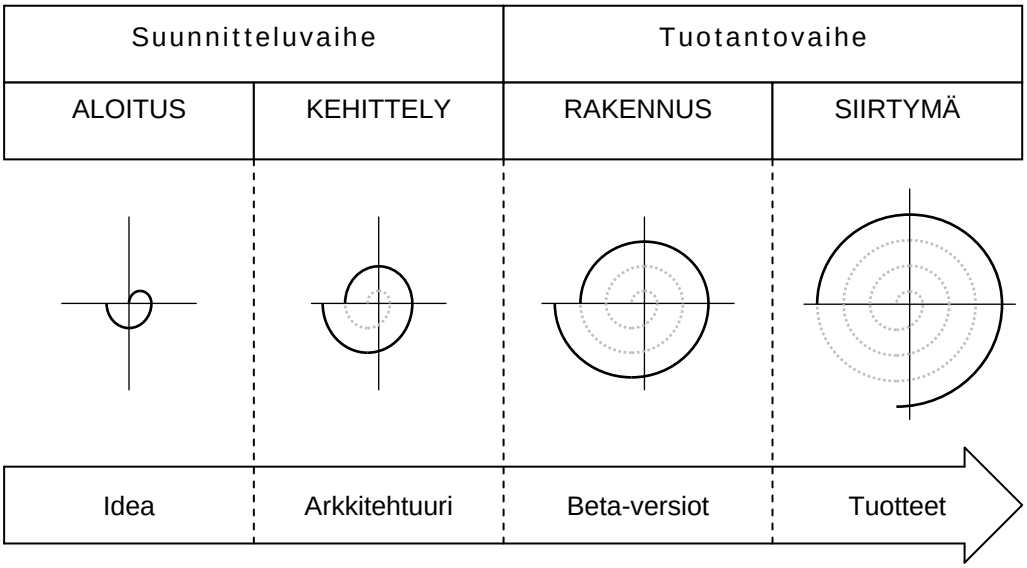
Syklit koostuvat neljästä päävaiheesta: *aloitusvaiheesta* (*inception), kehittelyvaiheesta (elaboration), rakennusvaiheesta (construction)* ja *siirtymävaiheesta (transition)*. Erikssonin ja Penkerin [9] mukaan RUP-malli on kehitystyön makroprosessi, joka on suunnattu sekä projektin johtajille että kehittäjille. Mikroprosessitoiminnat, kuten analyysi, suunnittelu, toteutus ja testaus ovat edelleen olemassa, mutta ne on sijoitettu osaksi makroprosessi-kehystä. Jokaisessa sukupolvisyklissä suoritettuja vaiheita tarkastellaan makronäkö-kulmasta eli hallinnollisesti. Nämä työvaiheet koostuvat samoista työvaiheista kuin olio-suuntautunut ohjelmistokehityskin – analyysi, suunnittelu, toteutus ja testaus – vaikkakin ne suoritetaan iteratiivisesti makroprosessin puitteissa.

Päävaiheiden sisältö on karkealla tasolla seuraava (tarkennettu sisältö liitteessä A):

- *Aloitus*. Määrittelee projektin laajuuden ja tavoitteet.
- *Kehittely*. Laajentaa aloitusvaiheen näkemystä. Projektin toteutettavuus tarkiste-taan, ja projekti suunnitellaan toimintojen ja resurssien osalta. Rakennettavan sovelluksen perustoiminnallisuus ja -arkkitehtuuri määritellään.
- *Rakennus*. Tuote kehitetään yksityiskohtaisesti useamman iteraatiosarjan aikana. Vaiheeseen kuuluu lisää analyysiä ja suunnittelua sekä itse ohjelmointi.

- *Siirtymä*. Toimittaa sovelluksen tai järjestelmän loppukäyttäjille mukaan lukien oheistoiminnot kuten markkinointi, paketointi, tuotetuki, dokumentointi ja koulutus.

Vaiheiden sisällössä voi havaita yhtäläisyyksiä spiraalimalliin. RUP-mallin ja spiraalimallin yhtäläisyyksiä onkin vertailtu kuvassa 9.



Kuva 9: RUP-malli ja spiraalimalli [4]

Spiraalin koolla Royce [4] on halunnut kuvata myös projektin inertian kasvamista suhteessa projektin vaiheeseen ajatellen tehdyn työn ja tuotettujen artefaktien määrää. Myöhemmissä vaiheissa reagointiaika kaikkiin suuriin muutoksiin (kuten esimerkiksi suuret arkkitehtuuri-, määrittely- tai organisaatiomuutokset) kasvaa voimakkaasti.

2.5 Prosessin artefaktit

Prosessin aikana sen eri vaiheissa syntyy erilaisia tuotoksia, *artefakteja*, jotka palvelevat monia eri tarkoituksia. Tässä luvussa esitellään artefaktin määritelmä ja tuotantoprosessin artefaktiryhmät RUP-mallin näkökulmasta.

2.5.1 Artefaktin määritelmä

Artefakti tarkoittaa ’tekoesinettä’. Artefakti on luonnonesineen vastakohta; artefakti on ihmiskätten työtä ja tietoisen (intentionaalisen) toiminnan tulosta, toisin kuin luonnon-esineet, jotka ovat sattumanvaraisten luonnonvoimien aikaansaannosta. Tuote, tuotos eli artefakti on ihmisen työn tulos, esimerkiksi käyttöesine, taideteos, vaate, talo, atk-ohjelma, palvelu, opetustapahtuma tai teatteriesitys.

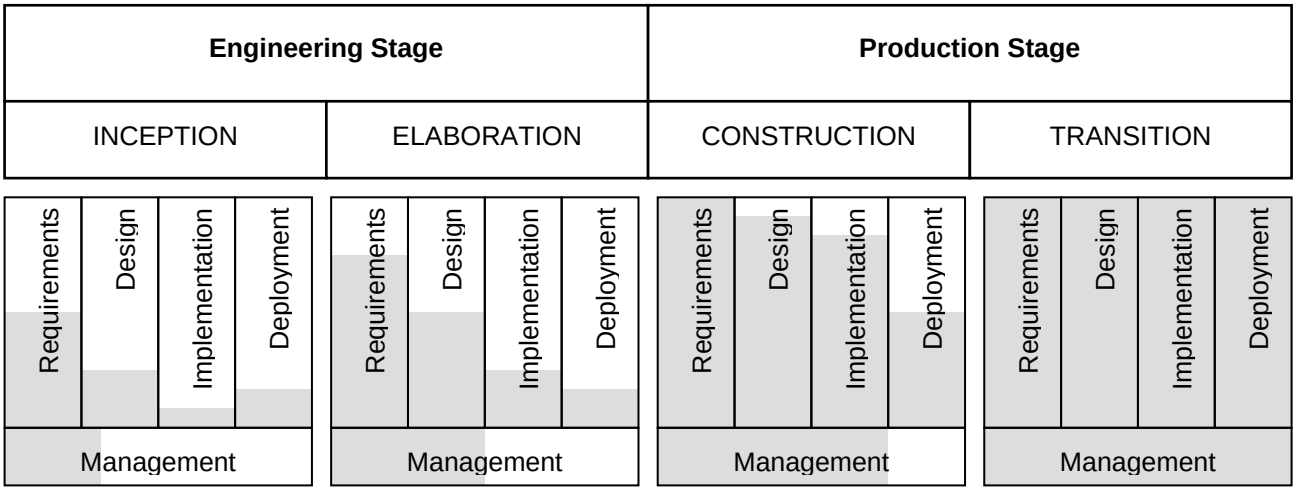
2.5.2 Artefaktiryhmät

Ohjelmistotuotantoprosessissa artefakteja ovat esimerkiksi vaatimusmäärittelydokumentti, arkkitehtuurikuvaus, testaussuunnitelma ja sovelluskomponentit. Artefaktit jakautuvat RUP-mallissa viiteen ryhmään: *hallinto-, vaatimus-, suunnittelu-, toteutus-* ja *lanseerausryhmään* (Eng. *management, requirements, design, implementation* ja *deployment*) [4]. Kuvassa 10 on periaatteellinen esitys kussakin ryhmässä tuotettavista artefakteista.

Vaatimusryhmä	**Suunnitteluryhmä**	**Toteutusryhmä**	**Lanseeraus-ryhmä**
1. Dokumentoitu visio, synopsis 2. Vaatimukset	1. Suunnittelumallit 2. Testausmalli 3. Arkkitehtuuri-kuvaus	1. Lähdekoodi 2. Valmiit ohjelma-komponentit	1. Valmis ohjelmisto 2. Käyttäjän opas
Hallintoryhmä			
Suunnitteluartefaktit		**Toimintoartefaktit**	
1. Työnjakokaavio 2. Business case 3. Lanseerausmäärittely 4. Ohjelmiston kehityssuunnitelma		5. Lanseerauskuvaukset 6. Tilannekatsaukset 7. Versionhallinta 8. Lanseerausdokumentit	

Kuva 10: Artefaktiryhmät ja niiden artefaktit [4]

Royce [4] on kuvannut myös artefaktiryhmien suhdetta prosessin etenemiseen kuvan 11 mukaisesti. Kuvassa tummennettu alue kuvaa kunkin artefaktiryhmän valmiusastetta.



Kuva 11: Artefaktiryhmien valmiusaste elinkaaren eri vaiheissa [4]

Merkittävää tässä näkökulmassa on se, että jo alkuvaiheessa mietitään valmistettavan sovelluksen kokonaisuutta. Visioartefaktien (määrittelydokumentit, projektikuvaus, projektisuunnitelma) tulee sisältää jo näkemys valmiista tuotteesta ja esitys toteutusarkkitehtuurista.

2.6 Inkrementaalinen vai iteratiivinen prosessi?

Inkrementaalisuuden ja iteratiivisuuden eroa on tarpeen täsmentää. Cockburn [10] erottelee inkrementit ja iteraatiot toisistaan seuraavasti:

- *Inkrementit (increments)* mahdollistavat tuotantoprosessin korjaamisen tai parantamisen.
- *Iteraatiot (iterations)* mahdollistavat tuotteen laadun parantamisen.

Inkrementaalisuus (incremental development) on rakentamista ”pala palalta”. Yksi ”palanen” on yksi inkrementti. Inkrementit voivat olla eri kokoisia ja niiden valmistusajat eripituisia. Kukin inkrementti tuo toteutusvaiheessa toteutettavaan sovellukseen lisää toiminnallisuuksia tai ominaisuuksia. Inkrementaalisessa prosessissa sovelluksen toteutus ositetaan, ja osat toteutetaan eri aikoina tai eri tahtiin. Kun kaikki osat ovat valmistuneet, ne integroidaan toimivaksi kokonaisuudeksi. Cockburnin määrite inkrementista tarkoittaa, että kunkin inkrementin toteutuksesta saa lisää kokemusta ja mahdollisesti oppii uutta,

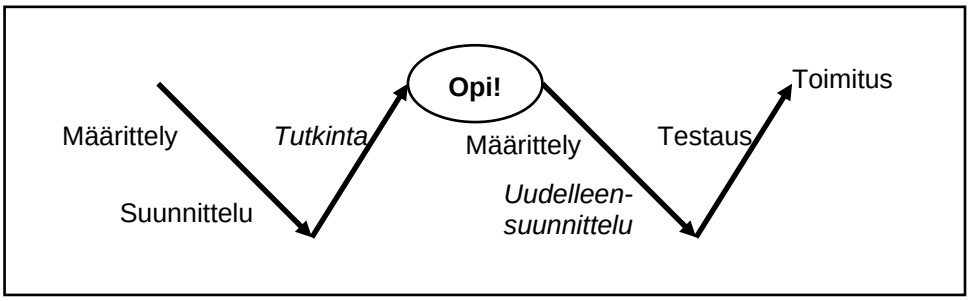
jolloin seuraavaan inkrementtiin siirryttäessä on mahdollista hyödyntää kokemuksia ja oppeja paremman tuloksen saavuttamiseksi.

Iteratiivisuus (iterative development) on kehittelyprosessi, jossa sovelluksen osiin tehdään muutoksia ja parannuksia. Iteraation tarkoitus on parantaa kohteen laatua; kohteena voivat niin määrittelyt, suunnittelumallit kuin toteutetut komponentitkin. Iteraatio tarkoittaa uudelleentyöstämistä: sovellukselle on esimerkiksi valittu jokin arkkitehtuuri tietyin perustein, mutta valintaa päätetään vielä harkita uudelleen mahdollisten uusien toiminnallisuusvaatimusten vuoksi. Tällöin arvioidaan, onko vaatimukset mahdollista toteuttaa jo valitulla arkkitehtuurilla vai harkitaanko muita vaihtoehtoja.

Iteroinnin käyttäminen on kuin kaksiteräinen miekka. Projektin riskejä voi vähentää tutkimalla iteraatioiden avulla, ovatko esimerkiksi määrittelyt riittäviä tai onko määrittelyjen pohjalta tehty mallinnus tarpeeksi tarkka. Tällä vältytään viallisen tai puutteellisen sovelluksen toimittamiselta. Toisaalta riskiä lisätään sillä, että oikein erityisesti halutaan löytää vikoja ja puutteita, jotka mahdollisesti aiheuttavat suuriakin muutostöitä ja voivat viivästyttää projektia. Katselmuksilla on tärkeä rooli tehtäessä valintoja suuntaan taikka toiseen – riippuu paljolti asiakkaasta ja valmistuvan sovelluksen merkittävyydestä yhtiölle, miten näissä tilanteissa toimitaan.

2.6.1 V-W -vaiheistus

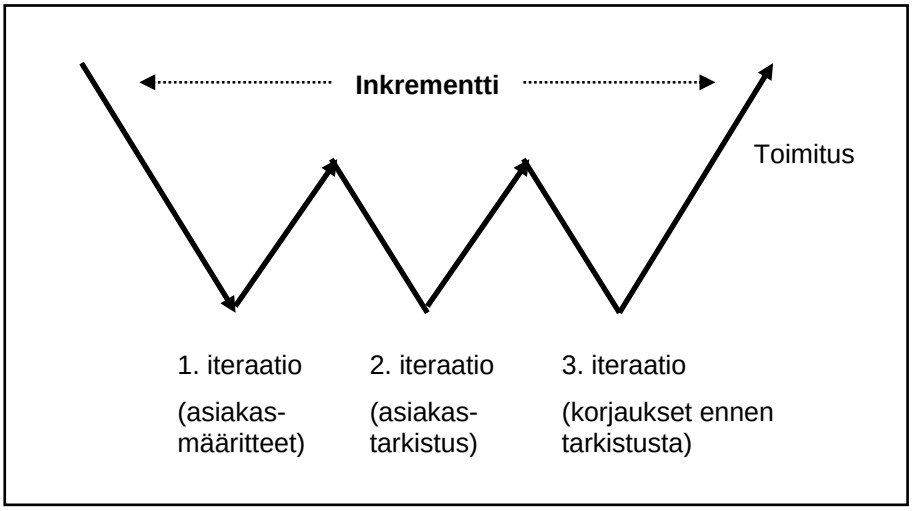
Tuotantoprosessi voi perustua inkrementaaliseen tai iteratiiviseen toimintatapaan, mutta toimintatavat on myös mahdollista yhdistää. Cockburn [10] on muodostanut yhdistelmästä V-W -vaiheistusmallin. Nimitys juontaa mallin piirtotapaan (kuva 12). Kun RUP-mallissa iteraatio koostuu viidestä toiminnosta (ks. kuva 8), koostuu V-W -vaiheistus neljästä perustoiminnosta: määrittely, suunnittelu, testaus ja toimitus.



Kuva 12: Iteraatio V-W -vaiheistuksessa [4]

V-W -vaiheistuksella on yhteys vesiputousmalliin siten, että yksittäinen V edustaa vesiputousmallin mukaisia päävaiheita. W tulee puolestaan siitä, että inkrementaalisessa prosessissa päävaiheita käydään läpi peräkkäin.

Inkrementeissä voi hyödyntää V-W –tyyppisiä iteraatiota esimerkiksi kuvan 13 mukaisella tavalla. Esimerkin lähtökohtana on tilanne, jossa kasataan kokoon vaatimusmäärittelyä.



Kuva 13: Iterointia inkrementissä [4]

Ensimmäisellä iteraatiolla kerätään asiakkaan kanssa yhdessä sovelluksen alustavat määritteet, joiden pohjalta tehdään ensimmäinen toiminnallisuusmäärittely. Se tarkistetaan asiakkaan kanssa (toinen iteraatio), ja asiakas huomaa määrittelyssä puutteita. Puutteet otetaan huomioon ja määrittelyjä korjataan. Kolmannen iteraation lopussa molemmat osapuolet toteavat määrittelyt riittäviksi, ja toiminnallisuusmäärittely hyväksytään.

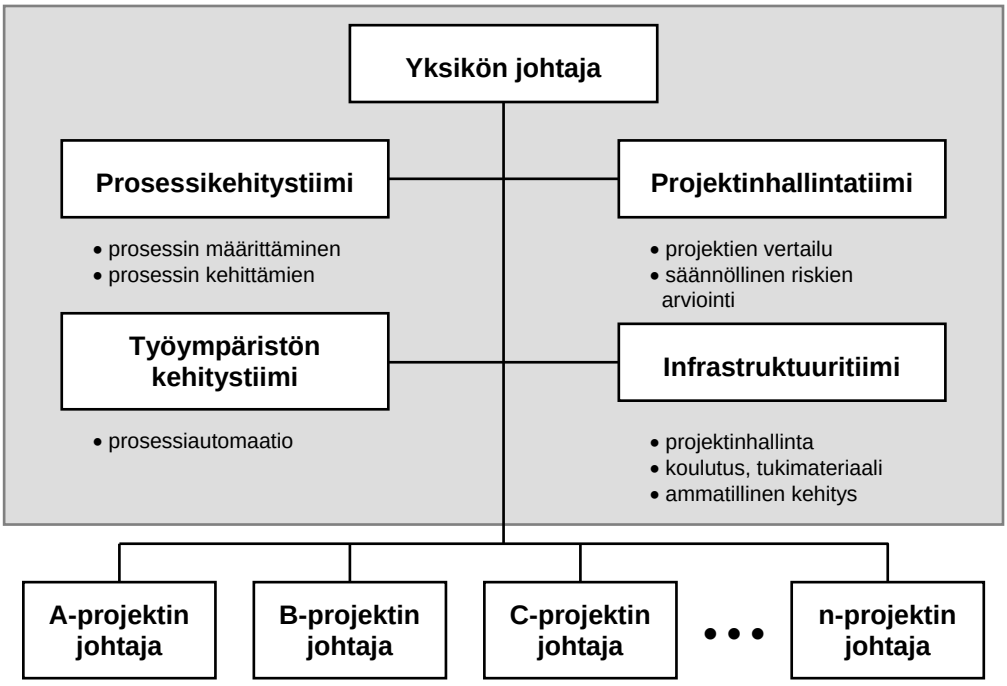
2.7 Organisaatiomallit ja roolit

Walker Royce [4] on tiivistänyt organisaation merkityksen seuraavasti:

> Organisaatiorakenteet muodostavat tiimien arkkitehtuurin. *(Suomennos kirjoittajan).*

Organisaatiolla on siis prosessin toiminnan kannalta samanarvoinen asema kuin teknologia-arkkitehtuurilla toteutettavassa sovelluksessa. Organisaation tarkoitus on määrittää tiimit, niiden tehtävät, tuotokset ja vastuualueet. Nämä ovat välttämättömiä perusrakenteita, jotka luovat yhdessä prosessimääritysten kanssa yrityksen tavan toimia.

Organisaatiomallit ovat yrityskohtaisia, ja kirjoissa esitetyt yleiset mallit harvoin soveltuvat suoraan yritysten käyttöön. Ne antavat kuitenkin suuntaviivoja organisaation luomiseen. Royce [4] jakaa ohjelmistoyrityksen organisaatiot kahteen kategoriaan: linjaorganisaatioon ja projektiorganisaatioon. Linjaorganisaatio muodostaa yrityksen hallinnollisen ja tukitoimintaosan, projektiorganisaatio puolestaan tuotannon projektikohtaisen organisaation. Roycen linjaorganisaatiomalli on esitetty kuvassa 14, projektiorganisaatiomalli on esitetty liitteessä B.

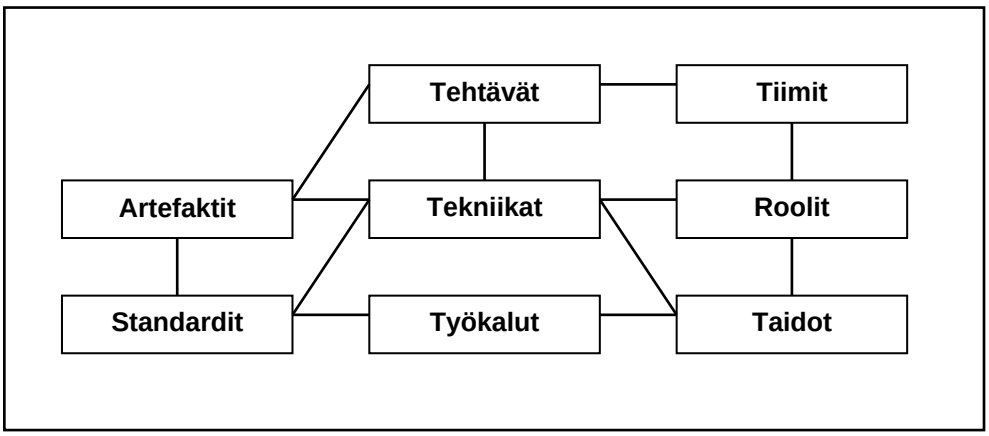


Kuva 14: Ohjelmistotuotantoyksikön linjaorganisaatio ja perusroolit [4]

Roycen mallit ovat karkeasti yleistettyjä – hänen tavoitteenaan on ollut luoda perusmalli, josta voi räätälöidä yrityskohtaisia perusorganisaatioita poistamalla, lisäämällä ja yhdistelemällä eri tehtäviä ja rooleja. Kahdenkymmenen työntekijän ohjelmistotalossa voi jopa yksi henkilö hoitaa kaikki linjaorganisaation roolit, kun taas 10 000 työntekijän telekommunikaatioalan yhtiössä voidaan tarvita satoja ihmisiä muodostamaan riittävän tehokas ohjelmistotuotantoyksikkö.

2.7.1 Roolit organisaatiossa: iso-M ja pieni-m

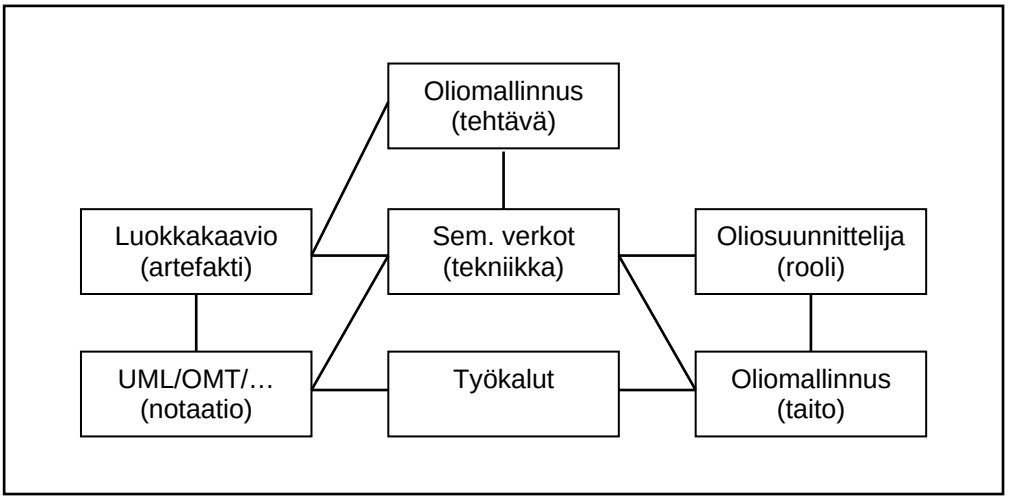
Cockburnin [10] iso-M -metodologia määrittää erään tavan suunnitella henkilöstön ja tiimien kokoonpanoa. Henkilöstö- ja tiimisuunnittelulla pyritään luomaan vastauksia sellaisiin kysymyksiin kuin ”Kuka tekee mitäkin tehtävää? Mitä artefakteja kenenkin tulee tuottaa? Kuka tarvitsee tuotettuja tuloksia? Mitä vuorovaikutuksia tiimien ja henkilöiden välillä on? Mitä työkaluja käytetään? Mitä toimintatapaa noudatetaan?”. Iso-M metodologia on tarkoitettu makro- ja metatason suunnitteluun, jossa pääasiallisesti pohditaan, mitä ominaisuuksia yrityksellä ja yrityksen henkilöstöllä on oltava, jotta ohjelmistoprojekteja voidaan viedä läpi. Iso-M -metodologiaa kuvaa kuva 15.



Kuva 15: Iso-M -metodologia kohdistuu kaikkiin työkuvauksiin [10]

Lähempänä mikrotasoa on pieni-m -metodologia. Se on suunnattu yksittäisten työtehtävien ominaisuuksien suunnitteluun, ja sen avulla voi esimerkiksi muodostaa työntekijöistä henkilöprofiileja. Profiilien avulla voi sitten sijoittaa henkilöitä sopiviin tehtäviin ja tiimeihin. Samalla myös selviää, jos organisaatiossa on jonkin tietyn osaamisen

vajausta tai jos organisaatiossa on sijoitettu työntekijöitä vääriin tehtäviin. Esimerkki oliosuunnittelijan profiilista kuvassa 16, lista esimerkkirooleista liitteessä I.



Kuva 16: Pieni-m -metodologia kohdistuu yksittäiseen työkuvaukseen [10]

Roolit. Roolit ovat kuin työkuvaukset, jotka laittaisit työnhakuilmoituksiin hakiessasi uusia työntekijöitä: projektipäällikkö, määrittelyiden kokooja, liiketoimintamallien suunnittelija, luokkasuunnittelija, ohjelmistosuunnittelija, ohjelmoija jne.

Taidot. Ne taidot, kyvyt ja ominaisuudet, joita odotat niillä ihmisillä olevan, jotka vastaavat työnhakuilmoituksiin: ohjelmointiosaamista, työkalujen tuntemusta jne.

Tekniikat. Ne tekniikat, joita odotat työntekijöiden työssään käyttävän: UML-mallinnus, käyttötapausmallinnus, Java-ohjelmointi jne.

Työkalut. Mitä työkaluja työntekijät työssään käyttävät, joko oman tekniikkansa piirissä tai tuottaakseen standardin mukaisia artefakteja.

Tiimit ja tehtävämääritteet. Kuinka ryhmittelet ihmisiä ja asetat heitä eri rooleihin. Esimerkiksi tehtävämäärite *liiketoiminnan analysoija-suunnittelija (business analyst-designer)* sisältää roolit määrittelyiden kokooja, määrittelyiden analysoija ja liiketoiminta-mallien suunnittelija. *Suunnittelija-ohjelmoija (designer-programmer)* puolestaan koostuu rooleista luokkasuunnittelija, ohjelmoija, dokumentoija ja testaaja. Jokaisessa *suunnittelu-tiimissä (design team)* on yksi liiketoiminnan analysoija-suunnittelija, kaksi tai kolme

suunnittelija-ohjelmoijaa, yksi tietokantasuunnittelija, yksi järjestelmäsuunnittelija ja yksi järjestelmädokumentoija. *Infra-suunnittelutiimissä (infrastructure design team)* taas on yksi pääsuunnittelija ja kahdesta neljään suunnittelija-ohjelmoijaa.

Artefaktit (artifacts, deliverables). Esimerkiksi määrittelydokumentit, luokkakaaviot, testaussuunnitelma – eli niitä tuotoksia, joita haluat kunkin henkilön tai tiimin toimittavan toiselle henkilölle tai tiimille.

Standardit. Määrittävät, mikä on sallittua tai kiellettyä suunnittelussa ja artefakteissa. Näitä ovat mm. notaatio, projektikäytännöt ja laatukriteerit. Notaatio määrittää yhteisesti sovitun esitystavan esimerkiksi seuraavissa tapauksissa:

- Käyttötapauskuvausten esitystapa toiminnallisuusmäärittelyissä
- Oliomallinnuksen esitystapa (UML, OMT, Shaeler-Mellor tms.)
- Ohjelmointikieli ja sallitut variantit

Projektikäytännöt määrittävät esimerkiksi seuraavanlaisia asioita:

- Käyttötapauskuvauksissa otsikkona käyttötapauksen nimi
- Many-to-many -relaatio kielletty ER-mallinnuksessa
- Friend-luokkatyypin käyttö kielletty C++ -ohjelmoinnissa

Laatukriteerit voivat määrittää tai pelkästään tuoda tiettäväksi mm. seuraavia seikkoja:

- Valmiusaste suhteessa määrittelyihin, mallinnukseen, dokumentointiin ja ohjelmointipohjiin (programming templates)
- Julkaistujen suunnittelumallien (design patterns) käyttö suunnittelussa

Tehtävät. Ne tiimien tehtävät ja välietapit, joiden tuloksena syntyy kunkin vaiheen edellyttämät artefaktit ja lopulta valmis tuote.

2.8 Oliosuuntautuneisuudesta

Cockburn [10] määrittää *oliosuuntautuneisuuden (object orientation, OO)* seuraavasti:

> Oliosuuntautuneisuus on sekä teknologia että tapa ajatella: ohjelmiston paketointi-teknologia (kapselointiperiaate) ja tapa ajatella ohjelmointia. Olioteknologian kaksi päämäärää ovat muutosten vaikutusten minimointi ja järkevän liiketoimintamallin luominen. Ensimmäisen päämäärän saavuttaminen edellyttää avointa ja runsasta kommunikointia ohjelmiston tuotantotiimissä. Toisen päämäärän merkittävä etu on siinä, että tuotannon henkilöstö ymmärtää paremmin käyttäjiä ja yrityksen kaupallisen puolen henkilöstöä, ja käyttäjät sekä kaupallinen henkilöstö ymmärtävät paremmin tuotannon henkilöstöä. Molemmat päämäärät edellyttävät ehkä normaalista ohjelmistosuunnittelijasta poikkeavaa persoonallisuutta; täytyy kyetä elämään muutoksessa, ymmärtää ja muodostaa abstraktioita sekä osata kommunikoida ihmisten kanssa. Tällainen muutos ohjelmisto-suunnittelijoiden persoonallisuudessa tuo lisäpotkua projekteihin. *(Suomennos kirjoittajan).*

Oliosuuntautuneisuudella on siis se ero rakenteiseen ohjelmointiin, että oliomenetelmin rakennettuja järjestelmiä on helpompi kuvata ja ymmärtää. Toisaalta se asettaa rakenteiseen ohjelmointiin ja siihen liittyviin menetelmiin tottuneille haasteita: ajattelutapaa täytyy muuttaa, ja ihmisten kanssa kommunikointi lisääntyy. Oliosuuntautuneisuus on vastakohta ”koppikoodaamiselle” – ohjelmoijien ja ohjelmistosuunnittelijoiden tulee keskustella keskenään sekä määrittelytiimin ja usein myös asiakkaidenkin kanssa.

Ohjelmistotuotannon kannalta olioteknologian monellakin tavalla tärkeä ominaisuus on *uudelleenkäytettävyys*. Se tarkoittaa yksikertaisesti sitä, että kerran luotua komponenttia voi käyttää useassa eri sovelluksessa jotumatta luomaan sovelluskohtaisesti samaa toimintoa uudelleen. Ominaisuus ei tosin sisälly teknologiaan ihan automaattisesti. Uudelleenkäytettävien komponenttien luonti vaatii olioajattelun syvällistä ymmärtämystä. Oliosuunnittelussa on tärkeää osata luoda *abstraktioita* eli yleistyksiä. Uudelleenkäytettäviä komponentteja voi luoda abstrahoimalla. Infomates Oy:n ohjelmistoarkkitehti Aleksi Kallio onkin kiteyttänyt olio-ohjelmoinnin kolme tärkeintä menetelmää eräänlaiseksi kolmiyhteydeksi: kapselointi, uudelleenkäytettävyys ja abstraktio.

3 OHJELMISTOTUOTANTO INFOMATES OY:SSÄ

Infomates Oy on ohjelmistoja valmistava yritys, jonka tuotanto pohjautuu olio-ohjelmointiin ja siihen liittyviin menetelmiin. Tuotanto on toiminut projektipohjaisesti. Projektien menetelmät ovat peräisin yrityksen johdon mukanaan tuomasta aiemmassa työssä hankitusta käytännön kokemuksesta, opinnoista ammattikorkeakoulussa sekä ulkopuolelta hankitusta teoriamateriaalista. Tässä luvussa esitellään tarkemmin Infomates Oy, tähän saakka käytössä ollut ohjelmistotuotantoprosessi, yrityksen henkilöstö ja organisaatio sekä muutamia käytännön kokemuksia tehdyistä projekteista.

3.1 Infomates Oy

Infomates Oy on perustettu 29.2.2000 ja yrityksessä työskentelee 15 henkilöä. Yritys valmistaa päätelaiteriippumattomia ohjelmistosovelluksia, joka käytännössä tarkoittaa Internet-selainohjelmalla käytettäviä sovelluksia sekä erilaisia mobiilipalveluita. Tuotteet koostuvat asiakaskohtaisista ratkaisuista ja SMS-palveluista, jotka ovat luonteeltaan enemmän pakettituotteita. Kaikki tuotteet ovat hankittavissa sovellusvuokrausperiaatteella.

3.1.1 Sovellusten arkkitehtuuri

Päätelaiteriippumattomuus ja sovellusvuokraus asettavat tiettyjä kriteerejä sovellusten toteuttamiseen. Erilaiset päätelaitteet tarkoittavat sovelluksen kannalta erilaisia käyttöliittymiä, sovellusvuokraus puolestaan mahdollisuutta käyttää sovellusta jopa miltä tahansa Internetiin kytketyltä tietokoneelta. Lisäksi asiakaskohtaisissa sovelluksissa on hyvin usein tarve integroida jo olemassa olevia ohjelmistoja osaksi toiminnallisuutta tai vähintäänkin mahdollistaa tietojen siirto eri sovellusten välillä.

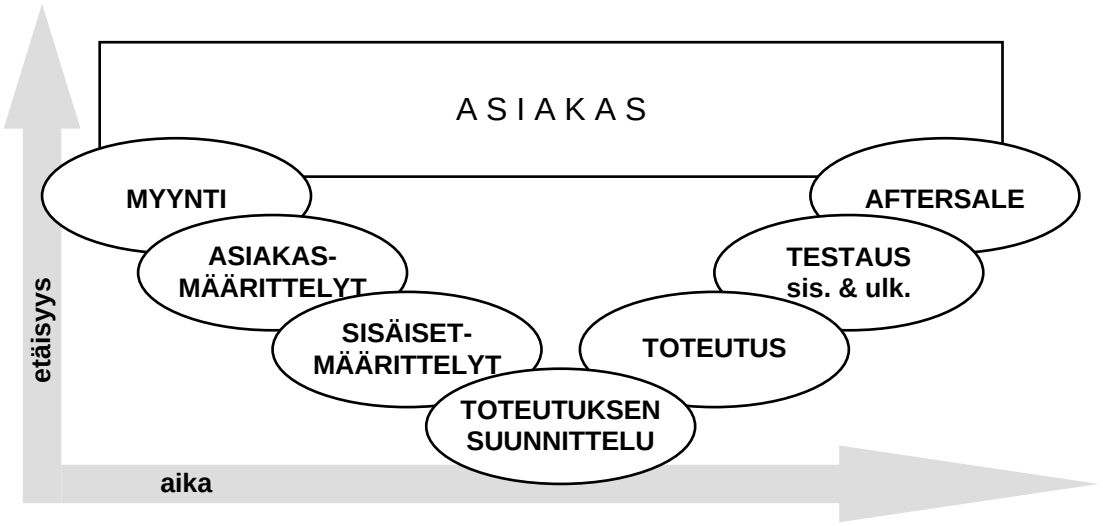
Tällaiset sovellukset täytyy toteuttaa siten, että käyttöliittymä, sovelluslogiikka ja tiedot on eroteltu omiksi lohkoikseen, ja eri kerrosten välillä on tarkkaan määrätyt rajapinnat. Kyseessä on siis kolmi- tai nelitasoarkkitehtuuri (kuva 17). Kuvattu arkkitehtuuri ei ole kirjoittajan omaa keksintöä, vaan kyseessä on yleisesti tunnettu ohjelmistoarkkitehtuuri.

Kuva 17: Nelitasoarkkitehtuuri.
Kolmitasoarkkitehtuurissa ei määritellä sisältökerrosta.

Kolmitasoarkkitehtuuri koostuu käyttöliittymä-, sovellus- ja tietokerroksesta. Nelitasoarkkitehtuurissa myös sisältö katsotaan omaksi kerroksekseen – sisältö muokataan kullekin käyttöliittymälle sopivaksi. Käyttöliittymä on käyttäjän rajapinta sovellukseen, ja tavallinen käyttäjä assosioi (yhdistää) käyttämänsä sovelluksen toiminnallisuudet juuri tähän kerrokseen. Varsinainen toiminnallisuus tapahtuu kuitenkin sovelluskerroksessa. Olio-ohjelmoinnissa tietokerros on näkymätön. Tietokerros ei toimi tietovarastona, johon tehdään operaatioita, vaan tarjoaa luoduille olioille pysyvyyttä eli persistenssiä.

3.2 Ohjelmistotuotantoprosessi

Infomates Oy:n tuotantoprosessi on mukaillut pitkälti vesiputousmallia. Alkuvaiheessa yrityksen tuotanto koostui lähes yksinomaan asiakasprojekteista, joissa asiakkaalle toteutetaan räätälöity sovellus tiettyyn erikoistarpeeseen. Prosessi on muotoiltu näistä lähtökohdista. Prosessikaavio on kuvattu kuvassa 18.



Kuva 18: Infomates Oy:n prosessi asiakasprojekteissa

Projektin käynnistymisen lähtökohtana on ollut tapaaminen asiakkaan kanssa. Asiakkaalla on ollut tarve tietynlaiselle ohjelmistotyökalulle, jolloin työkalun ominaisuuksia on ryhdytty määrittelemään yhdessä asiakkaan kanssa. Kuvassa on pyritty hahmottamaan kunkin vaiheen etäisyyttä asiakkaasta, toisaalta on myös haluttu korostaa asiakkaan mukanaoloa tietyissä kriittisissä vaiheissa. Vaiheiden limittämisellä on haluttu painottaa osin rinnakkaisia työvaiheita projektin edetessä.

Näkyvin eroavaisuus teoriaosassa esitettyyn kuvan 3 malliin on määrittelyvaiheen jakaminen kahteen osaan, asiakasmäärittelyyn ja sisäiseen määrittelyyn. Tällä on haluttu erityisesti korostaa asiakaslähtöisyyttä ottamalla asiakas mukaan tuotteen kehittämiseen heti alkuvaiheessa. Asiakkaalla on mahdollisesti selkeä tarve saada jokin asia hoidettua omassa yrityksessään paremmin, mutta ei välttämättä tiedä informaatioteknologian tuomista ratkaisumalleista juuri mitään. Tällöin on tärkeää saattaa yhteen asiakkaan osaaminen ja kokemus omalta toimialaltaan, kertoa tekniikan tarjoamista

mahdollisuuksista ja ryhtyä yhteistyössä kartoittamaan ja täsmentämään tarpeita ja vaatimuksia uuden työkalun toteuttamiseksi.

Liitteessä C on kuvattu sama prosessi, mutta kuvaus on yleisempi ja pyrkii ottamaan huomioon asiakasprojektin lisäksi yrityksen oman tuotekehityksen. Yhtiön oma tuote-kehitys alkoi vasta hieman asiakasprojekteihin ryhtymisen jälkeen. Oman tuotteen kehittelyhän alkaa ideasta; Infomates Oy:ssä ideat kirjataan ideadokumentiksi, joita käsitellään säännöllisesti ideapalavereissa. Tällöin ideoihin syvennytään tarkemmin, arvioidaan niiden uutuus- ja markkina-arvo sekä toteuttamiskelpoisuus. Vasta tämän jälkeen ryhdytään määrittelytyöhön, mistä eteenpäin prosessin kulku onkin asiakas-projektin kulkua vastaava.

3.2.1 Prosessin artefaktit

Infomates Oy:n tuotantoprosessin aikana on määritelty tuotettavan vaiheittain taulukossa 3 mainitut artefaktit eli tuotokset.

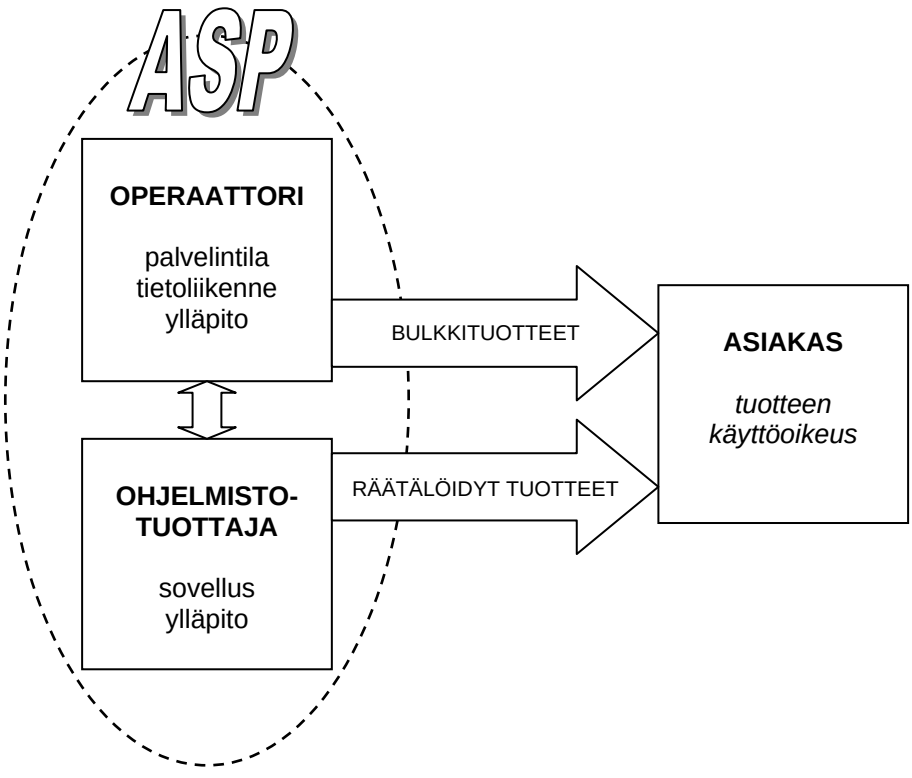
Taulukko 3. Infomates Oy:n tuotantoprosessin artefaktit

Projektin vaihe	Artefaktit
Myynti- tai ideavaihe	• Tarjous • Tuotekuvaus • Asiakaspalaverimuistiot • Ideadokumentti
Määrittely	• Projektisuunnitelma • Projektisopimus • Vaatimus- ja toiminnallisuusmäärittely • Käyttöliittymäprototyyppi • Määrittelypalaverimuistiot
Suunnittelu	• Teknologiakuvaus • Tietokantasuunnitelma • Luokkakaaviot
Toteutus	• Sovelluskomponentit • Tietokanta • Käyttöliittymä • Dokumentoitu lähdekoodi (JavaDoc) • Ohjelmakoodin katselmusmuistiot
Testaus	• Testitapauslomake • Testiraportit • Testauspalaverimuistiot
Aftersale	• Luovutuspöytäkirja • Käyttöohjeet • Ylläpitopäiväkirja

3.3 Sovellusvuokraus – ASP

Sovellusvuokraus (Application Service Providing) on liiketoiminnan muoto, jossa ohjelmiston ostamisen sijasta asiakas maksaa tuotteen käytöstä tai käyttöoikeudesta. Tämä on käytännössä mahdollista tietoverkon yli käytettävien ohjelmistojen kanssa. Tyypillistä sovellusvuokrausohjelmistoille (ASP-sovelluksille) onkin, että niitä käytetään Internet-selainohjelmalla. Sovellusvuokrausohjelmistoja tarjoavatkin yritykset tai yritys-yhteistyöyhteisöt, jotka tarjoavat lisäksi tietoliikenneyhteyksiä ja Internet-palveluita. Tällaista yritystä tai yritysryhmää kutsutaan ASP-palveluntarjoajaksi (Application Service Provider).

Infomates Oy:n kumppanina ASP-liiketoiminnassa onkin Kajaanin Puhelinosuuskunta, joka tarjoaa juuri edellä mainittuja tuotteita ja palveluita. Arvoketju on erityyppisille ASP-sovelluksille kuvassa 19 esitetyn mukainen.



Kuva 19: Infomates Oy:n ja KPO:n sovellusvuokrausliiketoiminta

Kuvassa mainitut bulkkituotteet ovat monistettavia massatuotteita, joita voidaan myydä sellaisenaan tai hyvin pienillä muutoksilla useille asiakkaille. Näiden myyntiä hoitaa operaattori, kun taas asiakaskohtaisia eli ns. räätälöityjä sovelluksia myy pääsääntöisesti

ohjelmistotuottaja. Asiakkaita ovat yritykset, sillä Infomates Oy:n tuotevalikoimaan ei sisälly kuluttajatuotteita.

3.3.1 Sidokset ohjelmistotuotantoprosessiin

Teknisestä näkökulmasta ASP-liiketoiminnalla on ohjelmistotuotantoprosessiin sellainen huomionarvoinen vaikutus, että kaikkien sovellusvuokrattavien ohjelmistojen tuotekehitys-projektissa on merkittävänä osapuolena operaattori, joka tarjoaa teknisen alustan sovelluksille. Prosessin määrittely- ja suunnitteluvaiheessa tulee ottaa huomioon sovelluksen vaatimat tietoliikennerajapinnat, suorituskapasiteetti ja levytila, jotka kaikki ovat alustan ominaisuuksiin liittyviä vaatimuksia.

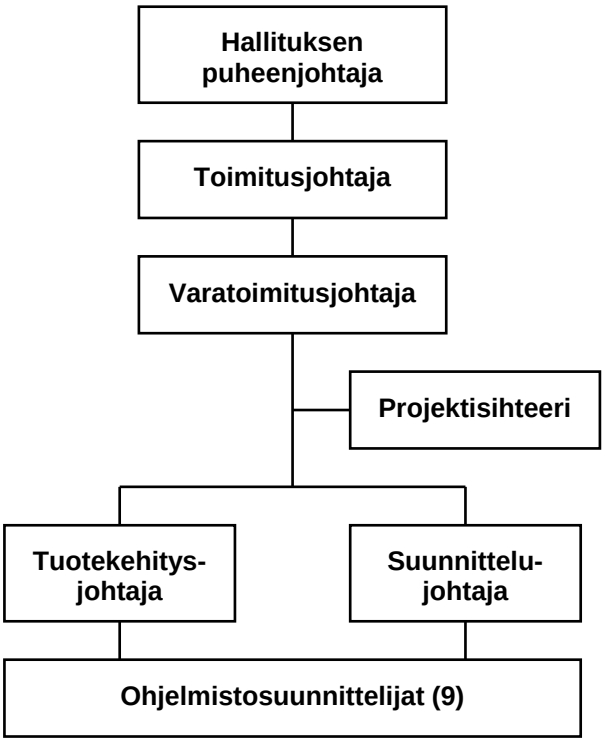
Toinen vaikutus ASP-yhteistyössä on bulkkituotteiden esitutkimustyö, jota tehdään yhteisesti. Kun operaattori toimii jälleenmyyjänä, on tärkeää saada näkemys myös heiltä tuotteen tarvittavista ominaisuuksista ja käyttäjäkohderyhmästä. Operaattorilla on pitempiaikaista kokemusta tekniseltä toimialalta – asiakkaiden tarpeita osataan ennakoida paremmin kuin nuoressa ohjelmistoyrityksessä.

3.4 Organisaatio ja henkilöstö

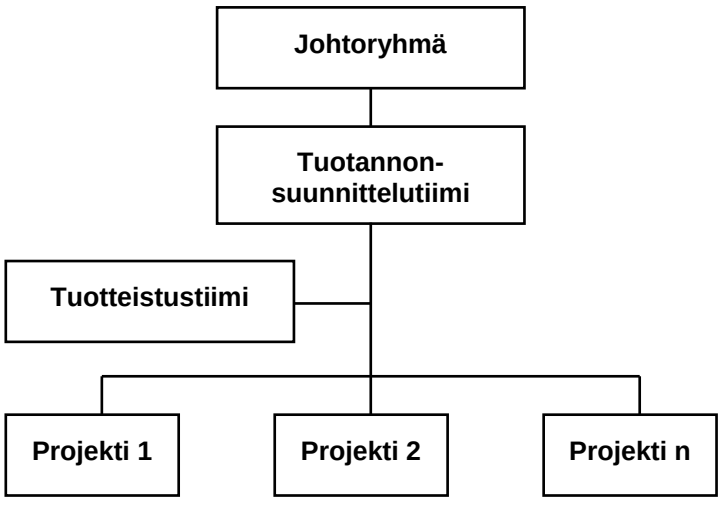
Yhtiön 15 työntekijää muodostivat vuonna 2000 kuvan 20 mukaisen linjaorganisaation. Hallinnon ja markkinoinnin tehtävissä työskenteli neljä henkilöä, määrittely- ja projektin-ohjaustehtävissä kaksi henkilöä ja ohjelmistosuunnittelu- ja tuotantotehtävissä yhdeksän. Linjaorganisaation merkitys on ollut hallinnollinen; projekteilla on ollut oma organisaatio-nsa.

Linjaorganisaatiota on hyödynnetty muodostettaessa operatiivisia tiimejä. Alunperin tiimejä oli vain yksi: tuotannonsuunnittelutiimi, johon kuuluivat toimitusjohtaja, varatoimitusjohtaja, tuotekehitysjohtaja ja suunnittelujohtaja. Myöhemmin tämä tiimi nimettiin johtoryhmäksi ja muodostettiin uusi tuotannonsuunnittelutiimi, johon kuuluvat tuotekehitysjohtaja,

suunnittelujohtaja ja kolme ohjelmistosuunnittelijaa. Tiimiorganisaatiokaavio on kuvattu kuvassa 21.



Kuva 20: Infomates Oy:n linjaorganisaatiokaavio



Kuva 21: Infomates Oy:n tiimiorganisaatiokaavio

Kuvassa 21 esiintyy myös tuotteistustiimi. Tiimin tehtävänä on huolehtia tuoteidea-lähtöisten projektien kaupallisesta tuotteistamisesta. Tämä tiimi on suhteellisen nuori, kirjoittamishetkellä se on ollut olemassa vasta neljä kuukautta ja on ollut hieman erillinen

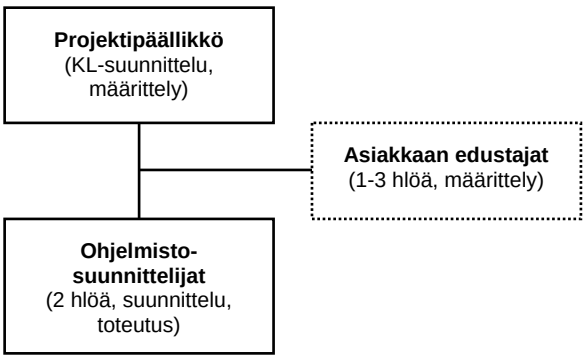
osa prosessissa ja projekteissa. Tuotteistustiimiin palataan luvussa 4.2.1. Projekteissa organisaation ovat muodostaneet projektitiimit. Luvussa 3.5 on kuvattu muutama käytännön projekti, ja kunkin kuvauksen yhteydessä on esitetty kaavio tiimin kokoonpanosta.

3.5 Käytännön projektikokemuksia

Käytännön projekteissa prosessin seuraaminen on onnistunut vaihtelevasti. Tässä luvussa kuvataan kolme aitoa tapausta (yhtiössä on tehty enemmänkin projekteja, mutta näissä kolmessa on havaittavissa selkeitä kehitystarpeita ja ne ovat kaikki tapauksina erilaisia). Asiakkaiden nimet ja liiketoiminnallisesti luottamukselliset asiat on jätetty kuvauksista pois.

3.5.1 Case A

Asiakasprojektin alkuvaiheessa luotiin vaatimusmäärittely yhdessä asiakkaan kanssa ja haettiin yhteistä näkemystä toteutettaviin toiminnallisuuksiin käyttöliittymäprototyypin avulla, mutta projektisopimuksessa ei kuitenkaan sovittu toiminnallisuuksista riittävän yksityiskohtaisesti. Tällöin jouduttiin tekemään suuriakin muutoksia jälkikäteen, ja toimitusaika viivästyi alkuperäisestä. Projektitiimissä oli mukana koko projektin ajan vähintään kolme henkilöä, ja määrittelyissä lisäksi asiakkaan edustajia (kuva 22). Sovellus on toimitettu ja se on käytössä asiakkaalla.

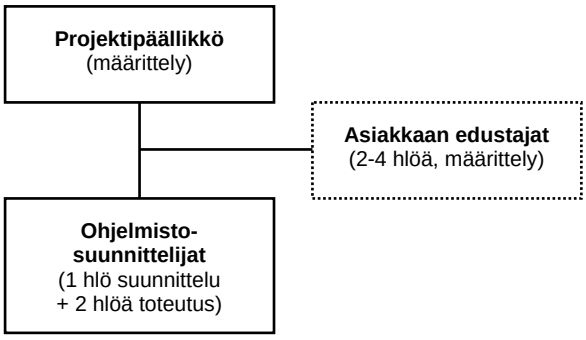


Kuva 22: Case A:n projektitiimi

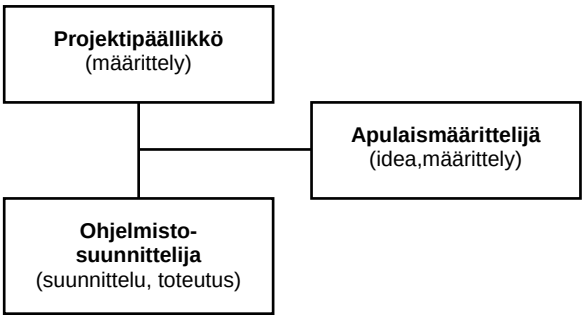
3.5.2 Case B

Toisessa asiakasprojektissa alkuvaihe käynnistyi hyvin; yhteisiä määrittelypalavereita pidettiin säännöllisesti ja läsnä oli useampia asiakkaan edustajia. Ongelmaksi muodostuivat kuitenkin projektin rahoitus ja määrittelytyö. Asiakkaalla ei ollut erityistä teknistä tuntemusta niistä mahdollisuuksista, joita sovelluksessa käytettävällä teknologialla oli mahdollista saavuttaa. Toisaalta asiakkaalle esiteltiin yhtiön omaa näkemystä sovelluksen ominaisuuksista, ja asiakkaan edustajat luottivat siihen, että yhtiöllä on riittävän selkeä kuva tarvittavista toiminnallisuuksista ja ominaisuuksista. Määrittelyn apuna käytettiin jonkin verran käyttöliittymäprototyyppiä.

Määrittelyissä oli lähtökohtainen ongelma: yhtiö oli asennoitunut tekemään asiakaskohtaisen sovelluksen, kun taas asiakas odotti yleistä ratkaisua, jota ehkä myöhemmin ensimmäisten käyttökokemusten jälkeen olisi räätälöity enemmän. Odotukset eivät siis kohdanneet. Projekti keskeytyi sittemmin rahoitusongelmiin. Tiimissä oli sen alkuvaiheessa projektipäällikön lisäksi yksi henkilö määrittelijä/suunnittelija/toteuttajana. Myöhemmin tiimiin lisättiin kaksi ohjelmistosuunnittelijaa lisää (kuva 23a). Määrittelyissä oli mukana asiakkaan edustajia.



Kuva 23a: Case B:n projektitiimi



Kuva 23b: Case C:n projektitiimi

3.5.3 Case C

Yhtiön oma tuotekehitysprojekti sai lähtösysäyksen näennäisen yksinkertaisesta ideasta. Idea vaikutti sen verran yksinkertaiselta toteuttaa, että toteutustiimiin valittiin vain yksi

henkilö projektipäällikön lisäksi (kuva 23b) ja aikatauluksi arvioitiin 2-3 kuukautta. Toteutus aloitettiin käytännössä jo ideadokumentin ja kahden sisäisen määrittelypalaverin pohjalta. Ensimmäinen prototyyppi valmistuikin nopeasti, ja koska kyseessä oli bulkkituote, testaukseen päätettiin ottaa mukaan operaattorin edustajia.

Testauksessa ja myöhemmissä muissa palavereissa tuli esille kosolti ohjelmallisia virheitä ja toiminnallisuuspuutteita. Operaattorin edustajalta tuli merkittäviä muutosehdotuksia. Kaikki muutokset kirjattiin, korjaus- ja jatkokehitystyötä jatkettiin. Yksinkertainen idea oli kuitenkin jo laajentunut merkittävästi, ja tehdyt arkkitehtuuri- ja rajapintavalinnat eivät täysin tukeneet asetettuja uusia vaatimuksia. Projektin tuloksena valmistuu määritysten mukainen kaupallinen bulkkituote, mutta aikataulu venyi kuukaudella, ja tuotteen jatkokehitys täytyy seuraavaan versioon aloittaa alusta useiden teknisten rajapintojen toiminnan parantamiseksi.

4 OHJELMISTOTUOTANNON KEHITTÄMINEN

Kehittämistyön lähtökohtina voidaan pitää prosessille annettuja vaatimuksia; oliopohjaisen tuotannon tuomat vaatimukset, projektin aikana tapahtuviin muutoksiin mukautuminen, projektin ja organisaation koon mukaan skaalautuva, sisältää sellaisia mittareita, joiden avulla eri projekteja voi riittävällä tarkkuudella verrata toisiinsa. Selkeästi tavoitteena on makroprosessin määrittäminen (makroprosessista on kerrottu luvussa 2.3).

4.1 Vanhan prosessin kehittämiskohteet

Infomates Oy:n aiemmassa prosessissa voi havaita kehittämiskohteita useilla eri tasoilla. Vaikka prosessi on periaatetasolla määritetty, määritykset eivät ole tarpeeksi tarkkoja, ja kriittisiä asioita on jäänyt määrittämättä. Merkittävimmät kehitystavoitteet on lueteltu taulukossa 4.

Taulukko 4. Prosessin kehittämistavoitteet ja -toimenpiteet

KEHITTÄMISTAVOITE	KEHITTÄMISTOIMENPITEET
Prosessin tarkentaminen.	Sovelletaan eri prosessimallien ominaisuuksia yhtiön tarpeisiin (erityyppiset projektit).
Prosessin artefaktien määrittäminen.	Määritetään prosessin eri vaiheissa tuotettavat artefaktit.
Tiimien ja roolien selkiyttäminen.	Määritetään perustiimit ja –roolit, joiden avulla työntekijöitä voidaan sijoittaa eri tehtäviin.

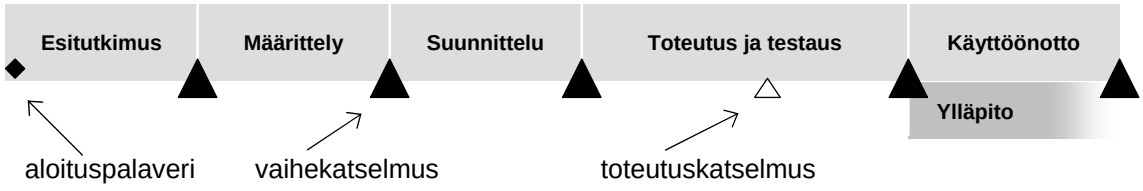
Käytetystä prosessista löytyy myös toimiviakin käytäntöjä, joita kannattaa sisällyttää myös uudistettuun prosessiin. Säilytettäviä käytäntöjä ovat määrittelyvaiheen jakaminen asiakasmäärittelyihin ja sisäisiin määrittelyihin sekä testausvaiheen jakaminen sisäiseen ja ulkoiseen testaukseen. Lisäksi käyttöliittymäprototyypin käyttö on todettu projekteissa erityisen tehokkaaksi määrittelyn ja suunnittelun apuvälineeksi.

4.2 Uudistettu prosessi

Infomates Oy:n tapauksessa on hyödyllistä yhdistää piirteitä kaikista kolmesta teoriaosassa esitetystä prosessimallista: perinteisestä vesiputousmallista, spiraalimallista ja RUP-mallista eli modernista prosessista. Kaikille prosesseille yhteinen nimittäjä on joka tapauksessa prosessikulku esitutkimuksesta käyttöönottoon (vesiputousmallissa lisäksi ylläpito), RUP-mallissa vaiheita on turhankin paljon supistettu. Spiraalimalli taas puolestaan avautuu hyvin lähelle vesiputousmallia, kun kierre ”venytetään” vaakasuoraksi. Spiraalimallin fokus on kuitenkin nelivaiheisessa riskienhallinnassa, ja varsinaisessa prosessikulussa voi käyttää haluamaansa menetelmää, kuten luvussa 2.4.3 on mainittu.

Iterointia voi tehdä kussakin vaiheessa, mikäli tarpeellista. Tärkeintä on arvioida inkrementtien määrä määrittely-, suunnittelu- ja toteutusvaiheelle; inkrementtien sisällä voidaan tarvittaessa tehdä iteraatioita, kuten luvussa 2.6.1 on kuvattu.

Infomates Oy:n ohjelmistotuotantoprosessissa vaiheita voi vähentää vesiputousmallissa esitetyistä ja hieman lisätä RUP-malliin nähden. Riittävät vaiheet ovat *esitutkimus*, *määrittely*, *suunnittelu*, *toteutus ja testaus* sekä *käyttöönotto* (kuva 24). Ylläpito muodostaa oman prosessinsa ja se alkaa välittömästi käyttöönoton yhteydessä. Ylläpitoprosessin sisältöön ei ole tässä työssä otettu kantaa.



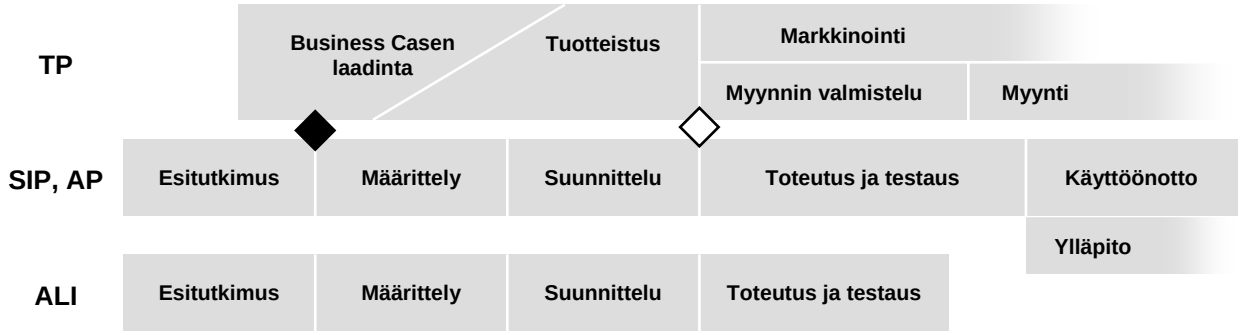
Kuva 24: Infomates Oy:n uusi prosessikaavio ja välietapit

Kuvaan on merkitty myös tärkeimmät välietapit, joita ovat esitutkimuksen aloituspalaveri ja kunkin vaiheen lopussa pidettävät vaihekatselmukset. Kullakin vaiheella on myös omat välietappinsa, mutta ne ovat projektikohtaisia ja liittyvät eri artefaktien ja osa-artefaktien tuottamiseen, joten ne on makrotasonäkymästä jätetty pois. Tarkennettu prosessikaavio on kuvattu liitteessä D, johon on lisätty näkyviin myös prosessin vaiheisiin liittyvät artefaktit ja ohjedokumentit sekä tärkeimmät prosessikulkuun vaikuttavat välietapit ja tapahtumat. Välietappien katselmuksien eli *vaihekatselmuksien* tarkistuslistat löytyvät liitteestä E.

4.2.1 Prosessin muoto erityyppisissä projekteissa

Alihankintatyönä tehtävät sovellukset noudattavat hieman erilaista prosessikulkua kuin yrityksen oman tuotekehityksen projektit. Alihankintatöissä sovelluksen perusmäärittely on usein ennalta annettu, kun taas omissa tuotekehitysprojekteissa kaikki määrittelyt on tehtävä itse. *Alihankintaprojekteissa (ALI)* osallistutaan usein määrittelyn syventämiseen, mutta päävastuu toteutuksen sisällöstä on päämiehellä eli työn tilaajalla. Projektin vaiheita on vähemmän, sillä käyttöönottovaihe on tarpeeton – valmistetut ohjelmistokomponentit toimitetaan testausvaiheen jälkeen tilaajalle (tai on jo toimitettu tilaajan ympäristöön, lopputestaus on suoritettu tällöin paikan päällä).

Omat tuotekehitysprojektitkin jakautuvat kahteen kategoriaan: asiakaslähtöisiin tuotekehitysprojekteihin eli *asiakasprojekteihin (AP)* ja *sisäisiin tuotekehitysprojekteihin (SIP)*. Asiakasprojekteissa asiakkaan edustajia on mukana määrittelytyössä ja projektien etenemiseen vaikuttavat aloitus-vaiheessa voimakkaasti erilaiset liiketoiminnalliset asiat, kuten toteutettavan sovelluksen hinta, sisällön rajaaminen ja toimitusaikataulu. Sisäisissä projekteissa liiketoiminnallisilla asioilla on erilainen vaikutus ja tarkoitus: ohjelmistotuotannon rinnalla alkaa *tuotteistusprosessi (TP)*, jonka tavoitteena on luoda tuotteen kaupallinen infrastruktuuri (kuva 25). Kaupallisen infrastruktuurin katsotaan tässä sisältävän mm. tuotteen liiketoimintasuunnitelman, esitemateriaalin, markkinointisuunnitelman ja myyntisopimukset.



Kuva 25: Tuotteistusprosessi ja projektityypit

Nyt kun tuotteistus on määritetty omaksi prosessikseen, tulee tuotteistustiimille (kuva 21 sivulla 43) selkeämpi rooli. Samalla selkiytyy yhteys ohjelmistotuotantoprosessiin. Koska jokainen projekti arvioidaan ensin liiketoiminnallisesta näkökulmasta (eli arvioidaan, onko projekti liiketaloudellisesti kannattava; onko tuotteelle markkinoita, maksetaanko tuotteesta tarpeeksi jne.), on tärkeää tehdä päätökset projektin etenemisestä mahdollisimman aikaisessa vaiheessa, ettei sellaisia kustannuksia pääse syntymään, joille ei ole olemassa maksajaa. Kuvassa 25 on mustalla vinoneliöllä merkitty se kohta projektissa, jossa tuotteistustiimi yhdessä esitutkintatiimin kanssa tekee päätöksen, ryhdytäänkö projektiin vai ei.

Toinen erittäin kriittinen välietappi on merkitty valkoisella vinoneliöllä. Kun tuotanto-prosessissa ollaan vaiheessa, jossa tehdään päätös toteutusvaiheeseen siirtymisestä, tehdään samalla päätös tuotteen kohtalosta – tehdäänkö tuotetta vaiko ei. Tuotteistus-prosessi on synkronoitu tähän välietappiin, sillä markkinointia ja myynnin valmistelua ei kannata aloittaa, jos tuotetta ei syystä tai toisesta aiotakaan valmistaa loppuun saakka.

Asiakasprojekteissa on tavoitteena tuoda mukaan oma tuotteistusprosessi silloin, kun näköpiirissä on toteutettavan sovelluksen laajempi kaupallinen hyödyntäminen. Nämä mahdollisuudet ovat olemassa, kun asiakkaan tilaamalla erikoistarpeeseen valmistetulla sovelluksella on joko sellaisenaan tai pienin muutoksin olemassa muita potentiaalisia asiakkaita. Tällöin haasteena on erottaa määrittelyvaiheessa yleisluontoiset toiminnallisuu-det ja ominaisuudet asiakaskohtaisista erikoistoiminnallisuuksista ja -ominaisuuksista.

4.2.2 Artefaktit

Prosessin aikana syntyviä tuotteita on jaoteltu projektityypeittäin ja vaiheittain liitteessä F. Lisäksi taulukossa on mainittu erikseen työnjohdon tuottamat artefaktit.

4.2.3 Oliosuuntautuneisuuden huomiointi

Oliosuuntautuneisuus ja oliopohjaiset menetelmät huomioidaan makrotasolla lähinnä mahdollistavilla toimenpiteillä. Mikrotasolla tehdään yksityiskohtaisempia valintoja. Erään työmenetelmän valinta kuitenkin vaikuttaa koko prosessiin: mallinnustekniikan valinta.

Ottamalla käyttöön UML-mallinnus (Unified Modeling Language, määrittelee oliopohjaisen mallinnusnotaation ja menetelmät) voidaan yhtenäistää tuotantoprosessin eri vaiheissa syntyvä dokumentaatio. Määrittelyvaiheen tehtäviin on sisällytetty käyttötapausten mallinnus UML:n käyttötapausmallinnusta hyödyntäen. UML:ää käytetään myös suunnitteluvaiheessa arkkitehtuuri-, komponentti- ja luokkakaavio-suunnittelussa.

Koska oliopohjaisten menetelmien käyttö vaatii paljon ihmisten keskinäistä vuoro-vaikutusta, on prosessin eräänä painopisteenä pyritty lisäksi huomioimaan, että tarvittava yhteinen keskustelufoorumi on olemassa (teknologiatiimi, esitellään luvussa 4.4.4) ja että erityisesti suunnitteluvaiheen aikana pidetään riittävästi palavereita.

4.2.4 Mukautuminen muutoksiin

Paras tapa mukautua muutoksiin on niiden ennaltaehkäisy. Siksi prosessissa painotetaan asiakkaan, tarkemmin *käyttäjän*, mukanaoloa määrittelyvaiheessa. Prototyypin avulla pyritään kaivamaan ominaisuus- ja toiminnallisuusvaatimuksia irti myös sellaisilta käyttäjiltä, joilla ei ehkä ole abstraktia hahmotuskykyä, vaan sen sijasta havaitsevat erilaisia tarpeita käytännön kautta. Onpa esitutkimusvaiheen erääksi arviointikriteeriksi (liitteessä G) asetettu kysymys ”Tarvitseeko asiakas konsultointia ennen määrittelyjen aloittamista?”.

Edellä mainittu arviointikriteeri ei ole mikään vitsi tai hieno ”kikka”, jolla yritetään rahastaa asiakasta lisää. On olemassa tapauksia, joissa asiakkaalla ei ole aiempaa kokemusta tietotekniikan käytöstä työssään tai liiketoiminnassaan, tai tieto on huomattavan vanhentunutta. Näissä tapauksissa on projektin menestymisen kannalta viisaampaa ensin kouluttaa asiakasta, jotta hän toisaalta näkee ja ymmärtää omia tarpeitaan paremmin ja

toisaalta osaa vaatia ohjelmistolta niitä asioita, joilla on aidosti merkitystä työn helpottamisessa tai tehostamisessa.

On muutoksiin varauduttu muutoinkin. Asiakasprojekteissa asiakas otetaan mukaan kaikkiin kriittisimpiin vaihekatselmuksiin (määrittely- ja suunnitteluvaiheen katselmukset). Testauksesta osa suoritetaan yhdessä asiakkaan kanssa. Tällöin tavoitteena on tutkia enää lähinnä käytettävyyteen ja ulkoasuun liittyviä seikkoja; mikäli testausvaiheessa tulee esille suuria toiminnallisia vikoja, on yhteydenpitoa asiakkaaseen ollut liian vähän tai määrittely on ollut puutteellista.

4.2.5 Laadun arviointi

Prosessin ja projektin tuotosten laatua voi arvioida useilla kriteereillä. Laatujärjestelmän eli laatukäsikirjan luominen yritykselle on kokonaisuutena jo diplomityön luokkaa, joten sellaista ei ole tähän työhön pyrittykään sisällyttämään. Toisaalta tavoitteena oli myös laatuajattelun ja mittareiden luominen itse prosessiin eikä erilliseksi järjestelmäksi. Tässä vaiheessa prosessin ja projektien arviointiin riittänevät taulukossa 5 luetellut laatukriteerit.

Taulukko 5. Prosessin laatumittarit

MITTARI	MITTAUSMENETELMÄ
Asiakastyytyväisyys.	Luodaan palaute- ja haastattelulomake, jossa on pisteytettyjä kysymys- ja vastausvaihtoehtoja. Kerätään lomakkeella palautetta säännöllisesti.
Projektin kannattavuus.	Lasketaan projektin tuotto vertaamalla toteutuneita kustannuksia budjetoituun (siihen budjettiin, jonka mukaan asiakasta on laskutettu), eli kustannusten A/T-suhde (arvion suhde toteutumaan).
Työmäärän A/T-suhde.	Lasketaan kaikkien vaiheiden työmäärien A/T-suhde erikseen ja yhdessä.
Aikataulu.	Projektin keston A/T-suhde.
Uudelleenkäytettävyyssuhde (UKS).	UKS = lopullisessa tuotteessa olevien olemassa olevien, aiemmin luotujen komponenttien määrä suhteessa sovelluksen kaikkiin komponentteihin.
Geneerisyyssuhde (GS).	GS = projektissa toteutettujen, tiettävästi uudelleenkäytettävien komponenttien suhde kaikkiin projektissa toteutettuihin komponentteihin.

4.3 Roolit ja tiimit

Roolien ja tehtävien muodostamiseen on hyvä käyttää luvussa 3 esitettyä iso-M ja pieni-m -metodologiaa. Ensin pohditaan, minkälaisia tiimejä ohjelmistotuotantoprosessissa kokonaisuutena tarvitaan. Sitten mietitään tiimeissä tarvittavat roolit ja muodostetaan näistä roolikohtaisia taulukoita pieni-m -metodologialla. Kun roolit on taulukoitu, voidaan niitä verrata tiimitaulukoihin, joita puolestaan verrataan sitten prosessikaavion ja tarkastetaan, onko kaikki tarpeellinen huomioitu.

Taulukossa 6 on esimerkki roolin muodostamisesta. Liitteessä H ovat Infomates Oy:n prosessissa määritetyt tiimit ja roolit. Järjestystä on Cockburnin mallista muutettu siten, että roolitaulukoissa rooli on asetettu ylimmäksi ja kirjoitettu vahvennetulla kirjasimella. Näin taulukot ovat löydettävissä ja käsiteltävissä luontevammin. Tiimejä on käsitelty tarkemmin seuraavassa luvussa.

Taulukko 6. Roolin muodostaminen pieni-m –metodologiaa hyödyntämällä, esimerkkinä käyttöliittymäsuunnittelija.

ROOLI Käyttöliittymäsuunnittelija		
Tuotokset	**Tekniikat, menetelmät**	**Tehtävät**
Käyttöliittymämallit Käyttöliittymät	Piirtäminen, mallinnus	Käyttöliittymän suunnittelu Käyttöliittymän toteutus
Käytettävät standardit	**Tarvittavat työkalut**	**Tarvittavat taidot**
Yhtiön sovitut käyttöliittymästandardit HTML, XML, CSS	Helppokäyttöinen, riittävän laadukas piirto-ohjelmisto (MS Powerpoint, CorelDraw) Riittävän laadukas kuvankäsittelyohjelmisto (Adobe Photoshop tms.) HTML-editori	Käyttöliittymäsuunnittelu Käytettävyys Visualisointi HTML, XSL, CSS

4.3.1 Tiimien työnjako

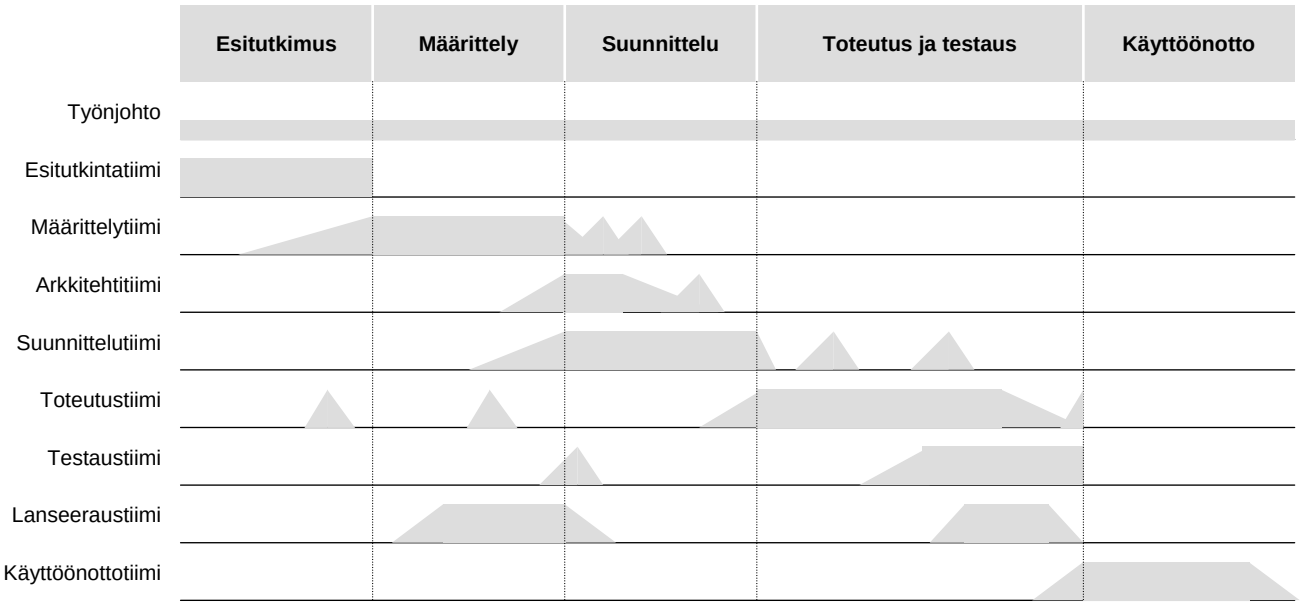
Tiimit on järkevää jaotella ja nimetä vaiheiden mukaisesti, koska myös tehtävät ja tuotokset ovat vaiheissa erilaiset. Jako ei perustu henkilövalintoihin, vaan roolivalintoihin. Tämä mahdollistaa tiimien skaalautuvuuden sekä työntekijöiden sijoittamisen osaamisen ja resurssien mukaan useampiin tiimeihin samanaikaisesti. Pienessä projektissa voidaan esimerkiksi yhdistää esitutkimus-, määrittely- ja testaustiimi yhdeksi tiimiksi, samoin arkkitehti- ja suunnittelutiimi sekä lanseeraus- ja käyttöönottoryhmä. Tällaisessa projektissa tiimejä olisi yhteensä viisi kahdeksan sijasta (kaikki tiimit).

Taulukossa 7 on esimerkki tiimin muodostamisesta. Taulukkoon on lueteltu tiimiin kuuluvat roolit, tiimin työssään tarvitsemat artefaktit, tiimin tuottamat artefaktit sekä tiimin tehtävät. Liitteestä H löytyvät kaikki Infomates Oy:n prosessissa määritetyt tiimit ja roolit.

Taulukko 7. Tiimin koostaminen vapaasti iso-M –metodologiaa soveltaen

MÄÄRITTELYTIIMI		
Roolit	**Tarvittavat artefaktit**	**Tuotettavat artefaktit**
Määrittelijä Bisnesasiantuntija Käyttäjä (asiakas)	Asiakasvaatimukset	Vaatimusmäärittely Toiminnallisuusmäärittely Käyttötapaukset
Tehtävät		
Määrittelyjen kerääminen. Tähän tehtävään sisältyy käyttäjien eli asiakkaan käyttäjien haastatteleminen hyödyntäen yhtiössä sovittuja käytäntöjä. Käyttötapausten koostaminen. Vaatimus- ja toiminnallisuusmäärittelyn laatiminen. Mahdollisten lisämäärittelyjen tekeminen suunnitteluvaiheen aikana.		

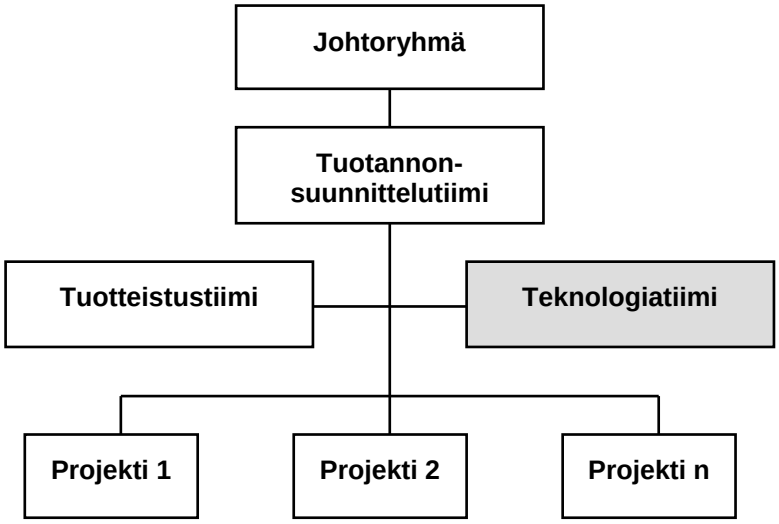
Kuvassa 26 on puolestaan kuvattu eri tiimien ja työnjohdon työmäärää prosessin eri vaiheissa. Työ jaksottuu tiimikohtaisesti portaittain; tämä mahdollistaa uuden tuotesukupolven tai kokonaan uuden projektin aloittamisen viimeistään meneillään olevan määrittelyvaiheen päätyttyä, sillä esitutkimus- ja määrittelytiimien resurssit vapautuvat käyttöön.



Kuva 26: Työnjohdon ja tiimien työn jakautuminen projektin aikana

4.3.2 Muita organisaatiouudistuksia

Linjaorganisaatio on nykyisellään toimiva, ja johtoryhmä sekä tuotannonsuunnittelutiimi ovat riittäviä hallinnollisia elimiä projektienhallintaan. Projekteihin on jo määritetty riittävästi tiimejä. Tekninen kehitys, laadun parantaminen ja asiakasvaatimukset kuitenkin edellyttävät foorumin, jossa puidaan käytettäviä teknisiä ratkaisuja ja mahdollisuuksia. Organisaatiossa tarvitaan siis prosessia tukevia tiimejä, tärkeimpänä *teknologiatiimi* (kuva 27).



Kuva 27: Teknologiatiimi lisättynä tiimiorganisaatiokaavioon

Teknologiatiimin tehtävänä on kehittää uudelleenkäytettävyyteen tähtääviä toimintatapoja kehittämällä yrityksen tuotannossa käytettäviä teknologioita, oliomenetelmiä ja näihin liittyvää infrastruktuuria ja järjestämällä yhtiön sisäisiä koulutuksia hyvistä suunnittelukäytännöistä. Teknologiatiimi voi myös muutoinkin toimia ”training centerinä” eli sisäisten koulutusten järjestäjänä. Käytännön kehitystyö tapahtuu mm. arvioimalla projekteissa käytettyjen toteutusratkaisujen soveltuvuutta toteutetun ohjelmiston toimintaympäristöön, järjestämällä suunnittelu- ja toimintatapakatselmuksia sekä järjestämällä teemapäiviä, pienseminaareja ja koulutuksia.

4.3.3 Vertailu kypsyysmalliin

Kehittämistyön viimeisenä asiakohtana on hyvä verrata tuloksia teoriaosassa esitettyyn kypsyysmalliin. Vertailua ei voi tehdä objektiivisesti eikä millään mainittavilla tieteellisillä perusteilla, sillä mittaustietoja aiemmista projekteista ei ole olemassa riittävästi järkevän tuloksen saamiseksi. Vertailu on tehty puhtaasti ”näppituntumalla” perustuen kirjoittajan omakohtaisiin käytännön kokemuksiin yhtiön tuotantoprosessista ja sen toimivuudesta, joten se on lisäksi erittäin subjektiivinen. Lähtötilannetta kuvaa taulukko 8, jossa avainominaisuuksista vahvennetut olivat olemassa kirjoittajan mielestä kohtuullisella tasolla ennen kehitystyön aloittamista.

Taulukko 8. CMM-avainominaisuudet lähtötilanteessa ennen kehittämistyön aloittamista (korostettu vahvennetulla kirjasimella)

CMM-taso	Avainominaisuudet
1. Alku. Ad hoc, jopa kaoottinen. Menestys riippuu täysin yksilösuorituksista.	• Ei ole
2. Toistettavissa. Perustason projektinhallintaa, jossa on määritetty sovelluksen toiminnallisuudet sekä projektin kustannukset ja aikataulu.	• **Vaatimustenhallinta** • **Ohjelmistoprojektin suunnittelu** • **Ohjelmistoprojektin seuranta ja valvonta** • **Ohjelmistoalihankinnan hallinta** • **Tuotteiden laadunvarmistus** • Muutoksienhallinta
3. Määritetty. Prosessin hallinnointi ja suunnittelu on dokumentoitu, standardisoitu ja integroitu. Kaikki projektit käyttävät hyväksyttyä ja räätälöityä versiota prosessista.	• **Organisaation prosessisuuntautuneisuus** • Organisaation prosessin määrittely • **Koulutusohjelma** • Tuotteenhallinta • Tuotesuunnittelu • Tiimityöskentely • Katselmoinnit
4. Hallittu. Prosessista ja tuotteiden laadusta kerätään tietoja eri mittareilla. Toimintatavat ovat itsestään selviä ja hallittuja.	• Kvantitatiivinen prosessinhallinta • Laadunhallinta
5. Optimoitavissa. Prosessia hiotaan ja parannetaan jatkuvasti eri laatumittareista saadun tiedon sekä uusin innovaatioiden ja tekniikoiden hyödyntämisen avulla.	• Virheiden estäminen • Teknologiamuutosten hallinta • Prosessimuutosten hallinta

Yhtiön projektit ovat olleet kutakuinkin toistettavissa (taso 2), muutamia projekteja on viety onnistuneesti läpi käyttäen silloista prosessia. Tosin muutoksienhallinta on joissakin tapauksissa ontunut. Lisäksi lähtötilanteessa on oltu prosessisuuntautuneita, ja sisäisiä koulutuksia on pidetty yhtiön alkumetreiltä lähtien. Henkilöstölle on hankittu myös runsaasti ulkopuolista koulutusta.

Tarkastellaan seuraavaksi kehittämistyössä luotuja lähtökohtia uusien avainominaisuuksien saavuttamiseksi (ei kai voine tässä vaiheessa esittää väittämää, että ”olisi saavutettu” – aika näyttää, kunhan muutama projekti on uuden prosessin mukaisesti saatettu päätökseen). Taulukossa 9 aiemmin saavutetut avainominaisuudet on hieman häivytetty vaaleammaksi, ja uuden prosessin avulla saavutettavat on korostettu vahvennuksella ja alleviivauksella.

Taulukko 9. Saavutettavissa olevat CMM-avainominaisuudet (korostettu vahvennetulla kirjasimella, lähtötilanteen ominaisuudet vaalennettu)

CMM-taso	Avainominaisuudet
1. Alku. Ad hoc, jopa kaoottinen. Menestys riippuu täysin yksilösuorituksista.	• Ei ole
2. Toistettavissa. Perustason projektinhallintaa, jossa on määritetty sovelluksen toiminnallisuudet sekä projektin kustannukset ja aikataulu.	• **Vaatimustenhallinta** • **Ohjelmistoprojektin suunnittelu** • **Ohjelmistoprojektin seuranta ja valvonta** • **Ohjelmistoalihankinnan hallinta** • **Tuotteiden laadunvarmistus** • **Muutoksienhallinta**
3. Määritetty. Prosessin hallinnointi ja suunnittelu on dokumentoitu, standardisoitu ja integroitu. Kaikki projektit käyttävät hyväksyttyä ja räätälöityä versiota prosessista.	• **Organisaation prosessisuuntautuneisuus** • **Organisaation prosessin määrittely** • **Koulutusohjelma** • Tuotteenhallinta • **Tuotesuunnittelu** • **Tiimityöskentely** • **Katselmoinnit**
4. Hallittu. Prosessista ja tuotteiden laadusta kerätään tietoja eri mittareilla. Toimintatavat ovat itsestään selviä ja hallittuja.	• Kvantitatiivinen prosessinhallinta • Laadunhallinta
5. Optimoitavissa. Prosessia hiotaan ja parannetaan jatkuvasti eri laatumittareista saadun tiedon sekä uusin innovaatioiden ja tekniikoiden hyödyntämisen avulla.	• **Virheiden estäminen** • Teknologiamuutosten hallinta • Prosessimuutosten hallinta

Saavutettavissa ovat siis muutoksienhallinta, prosessin määrittely, tuotesuunnittelu, tiimityöskentely, katselmoinnit ja virheiden estäminen. Muutoksienhallinnan menetelmiä on esitelty luvussa 4.2.4, tämän kehittämistyön tavoitteena on ollut prosessin kehittäminen eli prosessin määrittely (luku 4.2 sisältää uuden prosessin määritteet), tuotesuunnittelu mahdollistuu erillisen tuotteistamisprosessin (luku 4.2.1) avulla, tiimit ja roolit on määritetty luvussa 4.3.1, katselmoinnit suoritetaan jokaisen vaiheen päätteeksi ja lisäksi suoritetaan määrittely- suunnittelu- ja ohjelmakoodikatselmuksia, virheiden estäminen on osaltaan huomioitu muutostenhallinnassa ennakoivilla toimenpiteillä (joita on esitelty luvussa 4.2.4).

5 JOHTOPÄÄTÖKSET

Pienen insinööritoimiston ohjelmistotuotannolla ja globaalin ohjelmistotalon ohjelmisto-tuotannolla on käytännön tasolla paljonkin eroavaisuuksia, mutta peruspuitteet ovat kuitenkin samat – määrittely ja suunnittelu on tehtävä huolella ennen toteutusta, asiakkaan vaatimukset on ensisijaisesti huomioitava, tuotteen markkinakelpoisuuden huomiointi on liiketoiminnan kannalta elintärkeää. Kyse on lähinnä siitä, kuinka monta roolia ja tehtävää yhden työntekijän harteille kertyy, ja se puolestaan määrittelee tuotantokapasiteetin suuruuden. Tätä tuotantokapasiteettia voidaan ohjata tuotantoprosessin muodostamien menetelmien ja käytäntöjen avulla. Jos tuotantoprosessi on kunnossa ja se on yrityksessä viety käytäntöön, voi pienilläkin resursseilla saavuttaa suuriakin lopputuloksia. Prosessissa määritettyjen erilaisten projektityyppien hyödyntäminen antaa yritykselle mahdollisuuden mukautua joustavammin olosuhteiden muutoksiin, ja erityisesti alihankintaa harjoittavalle yritykselle se tuo juuri sitä uskottavuutta, jota alihankintatyön tilaajat edellyttävät. Jos prosessi vielä auditoidaan laatu- ja standardoimisjärjestön edustajan toimesta ja se saavuttaa kansainvälisesti hyväksytyn laatutason, yrityksen uskottavuus ammattimaisena toimijana kasvaa entisestään. Työtä auditointitasolle pääsemiseksi kuitenkin vaaditaan vielä paljon.

Infomates Oy on kotimaisessa ympäristössä hieman keskikokoista pienempi ohjelmistotalo, jolla on tavoitteena kansainvälistyä. Tavoitteiden saavuttamiseksi yrityksen tulee kasvaa – joko yksin tai verkostoitumalla muiden ohjelmistotalojen kanssa. Molemmissa tapauksissa henkilöstön määrä kasvaa, ja projekteihin sitoutuneiden osapuolien lukumäärä kasvaa. Kriteereinä ohjelmistotuotantoprosessin kehittämiselle olivatkin joustavuuden ja skaalautuvuuden huomioiminen, riittävän yhteismitallisen metriikan luominen eri projektien vertailun mahdollistamiseksi sekä käytettävän tuotantoteknologian huomioon ottaminen ja optimointi prosessin kannalta.

Esitetty malli Infomates Oy:n uudeksi ohjelmistotuotantoprosessiksi pohjautuu osin perinteiseen tuotantoprosessiin ja osin iteratiiviseen, moderniin tuotantoprosessiin. Yrityksen oma tuotekehitys ja asiakasprojektit on pyritty huomioimaan riittävällä tasolla, ja lisäksi on määritetty prosessi myös alihankintaprojekteille. Kehittämistyö on tehty lähes yksinomaan makroprosessinäkökulmasta ja tärkeä jatko työlle onkin mikroprosessien

määrittäminen sekä nyt määritettyjen menetelmien ja toimintatapojen saattaminen käytäntöön. Tärkeintä on, että sovitetaan menetelmä ratkaistavaan ongelmaan, eikä päinvastoin – tämä on ehkä spiraalimallin tarjoama paras näkökulma.

Vaikkakin luvussa 4.4.5 todettiin yhtiön jo pitkälti toimivan kypsyysmallin tasolla 2, se ei kuitenkaan tarkoita, etteikö myös siihen kuuluvia avainominaisuuksia tulisi kehittää edelleen. Uusien aluevaltauksien ohella jo saavutettuja asioita ei saa päästää unohtumaan. Päinvastoin, osassa vanhan mallin mukaan läpiviedyistä projekteista monet perusasiat ovat päässeet unohtumaan. Tavoite on kuitenkin vähentää ”kantapään kautta oppimista”, sillä se tulee liiketoiminnassa kalliiksi.

Itse prosessin määrittämisellä ei ole niin suurta vaikutusta kuin toimintatapojen luomisella. Prosessi määrittää kyllä toimintatavat, mutta käytäntöön vieminen on johtamistyötä. Toimintatavoista tulee tapa toimia vasta, kun työntekijät noudattavat prosessissa määritettyjä toimintatapoja. Johtamisoppejakin on siis työn aikana syntynyt: selkeästi johtaminen on tavoitteiden asettamista, toimintatapojen määrittämistä, tehtävien jakamista ja näiden kaikkien seurantaa ja ohjausta.

Tulevaisuus näyttää, kuinka prosessi toimii käytännössä. Ratkaisevana tekijänä tässäkin ovat viime kädessä sen toimeenpanijat, ihmiset.

LÄHDELUETTELO

1 Shaw, M. : Prospects for an Engineering Discipline of Software, IEEE Software (November 1990) pp. 15-24.

2 Haikala, I. , Märijärvi, J. : Ohjelmistotuotanto (7. painos), Suomen ATK-kustannus 2000

3 Brooks, Frederick P. : ”No Silver Bullet: Essence and Accidents of Software Engineering”, Computer Vol. 20 No. 4 (April 1987) pp. 10-19. Internetissä http://www.student.math.uwaterloo.ca/~cs212/resource/Articles/SilverBullet.html

4 Royce, W. : Software Project Management, Addison-Wesley 1998

5 Paulk, M. et al. The Capability Maturity Model: Guidelines for Improving the Software Process, Addison-Wesley, 1995.

6 Boehm, B. : "A Spiral Model of Software Development and Enhancement", ACM Software Engineering Notes (August 1986) pp.14-24.

7 Sommerville, I. : Software Engineering (viides painos), Addison-Wesley, 1996

8 Jacobson, I. , Booch, G. , Rumbaugh, J. : The Unified Software Development Process. Addison-Wesley 1998

9 Eriksson, H-E. , Penker, M. : UML, IT Press, 2000

10 Cockburn, A. : Surviving Object-Oriented Projects, Addison-Wesley 1998

LIITELUETTELO

LIITE A

RUP-PROSESSIN PÄÄVAIHEIDEN TAVOITTEET, TEHTÄVÄT JA ARVIOINTIKRITEERIT

Walker Royce [4] on määrittänyt RUP-prosessin päävaiheiden tärkeimmät tavoitteet, olennaisimmat tehtävät ja tärkeimmät arviointikriteerit.

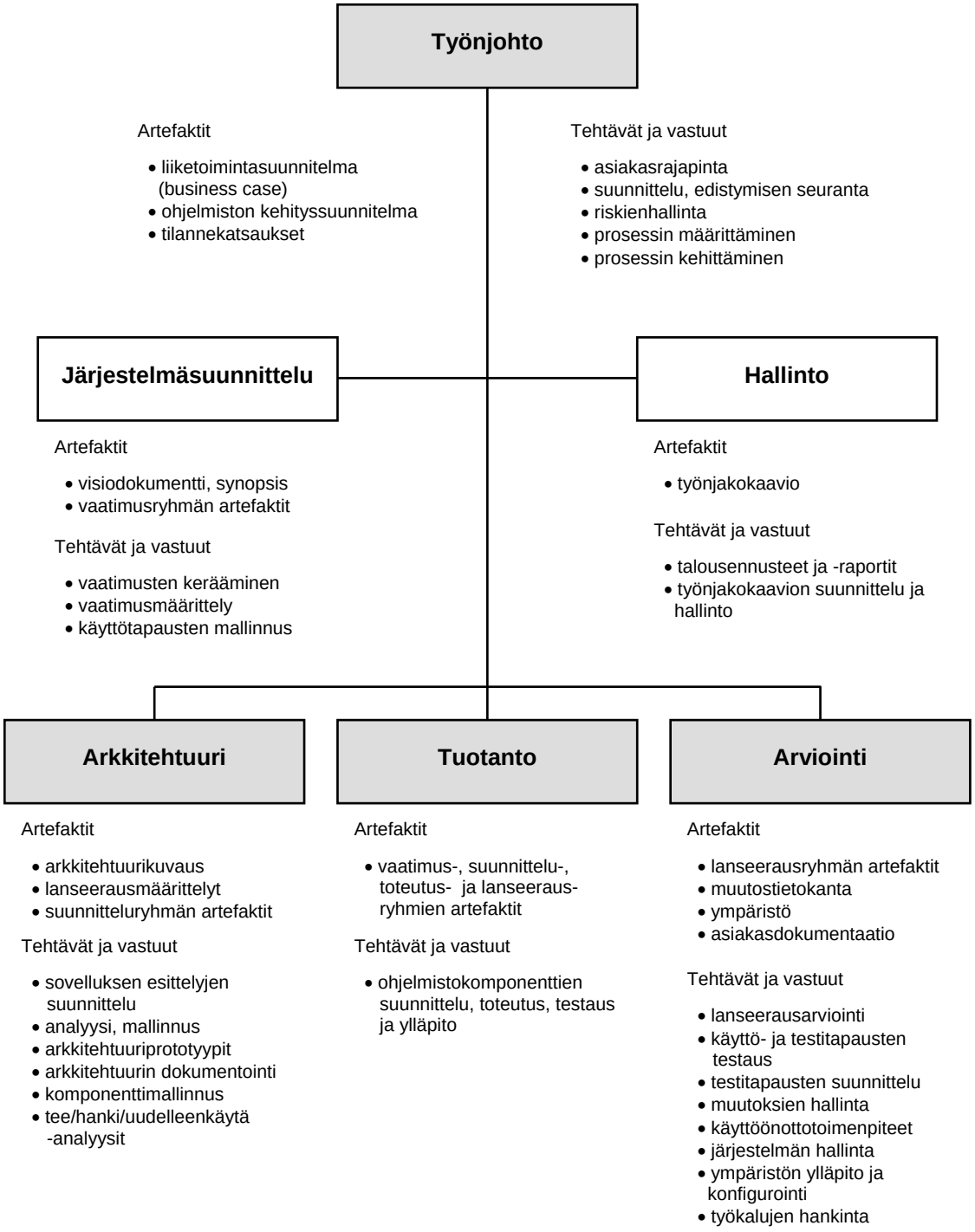
ALOITUSVAIHE	
Tärkeimmät tavoitteet	Projektissa toteutettavan sovelluksen (ohjelmiston, järjestelmän) puitteet sisältäen toiminnallisuuskonseptin, hyväksymiskriteerit ja selkeä yhteisymmärrys siitä, mitä valmistettavaan tuotteeseen sisällytetään ja mitä ei. Sovelluksen kannalta krittisimpien käyttötapausten erottelu sekä korkean tason synopsis, joissa määritetään suunnittelun suuntaviivat. Esitys vähintään yhdestä käyttökelpoisesta arkkitehtuurista, jolla synopsiksessa ja käyttötapauksissa kuvattu järjestelmä voidaan totoeuttaa. Koko projektin kustannusten ja aikataulun arvio (sisältäen arviot kehittelyvaihetta varten). Potentiaalisten riskien arviointi.
Olennaisimmat tehtävät	Projektin puitteiden laatiminen sanalliseen muotoon. Tähän tehtävään sisältyy vaatimusten ja toiminnallisen konseptin laatiminen käyttäjän näkökulmasta. Syntyvän dokumentin tulee sisältää riittävästi informaatiota ongelmakentän (problem space) määrittelemiseksi ja myöhemmin lopputuotteen valmiiksi toteamiseen. Arkkitehtuurin muodostaminen. Arvioidaan suunnittelun suuntaviivat, ongelmakentän epäselvyydet ja saatavilla olevat ratkaisukentän (solution space) työkalut sekä menetelmät (teknologiat, olemassa olevat komponentit). Sellaisen dokumentin koostaminen, josta käy ilmi vähintään yksi toteuttamiskelpoinen arkkitehtuuri sekä määrittelyt itse toteutettavista ja ostettavista osioista, jotta karkean tason kustannus-, aikataulu- ja resurssiarviot voidaan tehdä. Business casen eli tuotekohtaisen liiketoimintasuunnitelman laatiminen. Arvioidaan suuntaviivat riskienhallintaan, henkilöstömäärään, iteraatiosuunnitelmiin, kustannuksiin, aikatauluun ja odotettuun tuottavuuteen. Tehdään tuotekehitystä tukevaa infrastruktuuria (työkalut yms.) koskevat päätökset.
Tärkeimmät arviointikriteerit	Ovatko kaikki osapuolet yksimielisiä projektin puitteista sekä kustannus- ja aikatauluarviosta? Onko vaatimukset ymmärretty? Ovatko osapuolet yksimielisiä kuvatuista käyttötapauksista? Ovatko kustannus- ja aikatauluarviot, prioriteetit, riskit ja käytettävät menetelmät uskottavia? Kattaako ja kuvaako tehty arkkitehtuuriprototyyppi edellä mainitut kriteerit? (Arkkitehtuuriprototyypin suurin arvo on siinä, että sen avulla sovelluksen puitteet voidaan ymmärtää ja arvioida sen luoneen tiimin uskottavuus ja kyky ratkaista teknisiä ongelmia).

KEHITTELYVAIHE	
Tärkeimmät tavoitteet	Arkkitehtuurin puitteiden ja sisällön määrittäminen niin pian kuin käytännöllistä. Vision määrittäminen. Tarkan suunnitelman luonti rakennusvaiheen läpiviemiseksi. Vakuuttuminen siitä, että kaikki suunnitelmat tukevat visiota järkevin kustannuksin ja järkevässä aikataulussa.
Olennaisimmat tehtävät	Vision tarkka määritteminen. Tähän tehtävään sisältyy arkkitehtuuriin ja suunnitteluun liittyviin päätöksiin vaikuttavien järjestelmän käyttötapausten yksityiskohtainen ymmärtäminen. Prosessin ja infrastruktuurin tarkka määrittäminen. Tuotantoprosessin ja välietappien sekä niiden arviointikriteereiden muodostaminen. Arkkitehtuurin viimeistely ja komponenttien valinta. Tuotantosuunnitelmaan kuuluvat komponentit arvioidaan ja niiden osto-valmistussuhteet on sillä tasolla määritetty, että tuotantovaiheen kustannukset ja aikataulu voidaan määrittää luotettavasti. Valitut komponentit integroidaan ja verrataan suunnitelmaa synopsikseen. Näistä tehtävistä saadut tulokset voivat hyvinkin johtaa arkkitehtuurin uudelleen suunnitteluun, jos vaihtoehtoisia, toteutuksen kannalta parempia menetelmiä on löydetty tai jos vaatimusmäärittelyt ovat muuttuneet. Esityskelpoisen arkkitehtuuriprototyypin valmistaminen.
Tärkeimmät arviointikriteerit	Onko visio stabiili? Onko valittu arkkitehtuuri stabiili? Ovatko kustannus- ja aikatauluarviot, prioriteetit, riskit ja käytettävät menetelmät uskottavia? Kattaako ja kuvaako tehty arkkitehtuuriprototyyppi edellä mainitut kriteerit? (Arkkitehtuuriprototyypin suurin arvo on siinä, että sen avulla sovelluksen puitteet voidaan ymmärtää ja arvioida sen luoneen tiimin uskottavuus ja kyky ratkaista teknisiä ongelmia). Käykö prototyypin esittelystä ilmi, että suurimmat riskit on huomioitu ja uskottavasti ratkaistu? Onko tuotantosuunnitelma riittävän tarkka? Tukeutuuko se uskottaviin arvioihin? Ovatko kaikki osapuolet yksimielisiä siitä, että esitetty visio voidaan saavuttaa viemällä toteutussuunnitelma käytäntöön esitetyn arkkitehtuurin mukaisesti? Ovatko todelliset resurssikustannukset verrattuna suunniteluihin kustannuksiin hyväksyttäviä?

RAKENNUSVAIHE	
Tärkeimmät tavoitteet	Tuotantokustannusten minimointi optimoimalla resurssit ja välttämällä ylimääräistä tai toistuvaa työtä (scrap and rework). Riittävän laadun saavuttaminen niin nopeasti kuin käytännöllistä. Käyttökelpoisten sovellusversioiden (ns. alfa- ja betaversiot) tuottaminen niin nopeasti kuin käytännöllistä.
Olennaisimmat tehtävät	Resurssienhallinta, työnjohto sekä prosessin optimointi. Kaikkien tarvittavien komponenttien tuottaminen ja testaaminen asetettujen kriteereiden mukaisesti. Tuoteversioiden valmiusasteen arviointi visiossa määritettyjen hyväksymiskriteereiden perusteella.
Tärkeimmät arviointikriteerit	Ovatko tuotteen ominaisuudet sillä tasolla, että se voidaan toimittaa loppukäyttäjille? (Olemassa olevat puutteet eivät ole seuraavan julkaistavan version tekemisen tarkoituksen esteitä). Ovatko kaikki osapuolet valmiita tuotteen toimittamiselle loppukäyttäjille? Ovatko todelliset resurssikustannukset verrattuna suunniteluihin kustannuksiin hyväksyttäviä?
SIIRTÄMISVAIHE	
Tärkeimmät tavoitteet	Käyttäjien omatoimisuuden saavuttaminen (tuotteen käyttöön liittyen). Osapuolien keskinäisen yksimielisyyden saavuttaminen siitä, että tuotannon tavoitteet on saavutettu asetettujen kriteereiden ja vision mukaisesti. Viimeisten tuotteeseen liittyvien tehtävien loppunvienti niin nopeasti ja kustannustehokkaasti kuin käytännöllistä.
Olennaisimmat tehtävät	Rinnakkaisten tuotantoprosessien synkronointi ja yhdistäminen yhdenmukaiseksi tuotantoprosessiksi Lanseeraukseen liittyvien tehtävien valmistelu ja toteutus (siirtyminen vanhasta järjestelmästä uuteen, mahdollinen kaupallisen tuotepaketin tekeminen, myynnin tukimateriaalin koostaminen, myyntihenkilöstön koulutus). Lanseeraussuunnitelman arviointi vaatimussetin hyväksymiskriteereiden ja vision mukaan
Tärkeimmät arviointikriteerit	Onko käyttäjä (asiakas) tyytyväinen? Ovatko todelliset resurssikustannukset verrattuna suunniteltuihin kustannuksiin hyväksyttäviä?

LIITE B

PERUSPROJEKTIORGANISAATIO



Perusprojektiorganisaatio [4]

LIITE C

PROJEKTIN ETENEMINEN TUOTANTOPROSESSIN JA DOKUMENTOINNIN NÄKÖKULMASTA

PROJEKTIN ETENEMINEN

TUOTANTOPROSESSIN JA DOKUMENTOINNIN NÄKÖKULMASTA

INFOMATES
SOFTWARE TECHNOLOGIES

Ennen tuotantoa

IDEAKESKUSTELU

Projekti alkaa aina ideasta; joko Infomates Oy:n sisäisestä tai asiakkaan tarpeesta. Idea voi syntyä jopa yritysesittelykierroksella.

MÄÄRITTELYT

Ideakeskustelut johtavat määrittelykeskusteluun. Tällöin määritellään tuotteen/ratkaisun ominaisuudet ja toiminnallisuudet sekä se, onko idea toteutettavissa Infomate Oy:n tekniikalla.

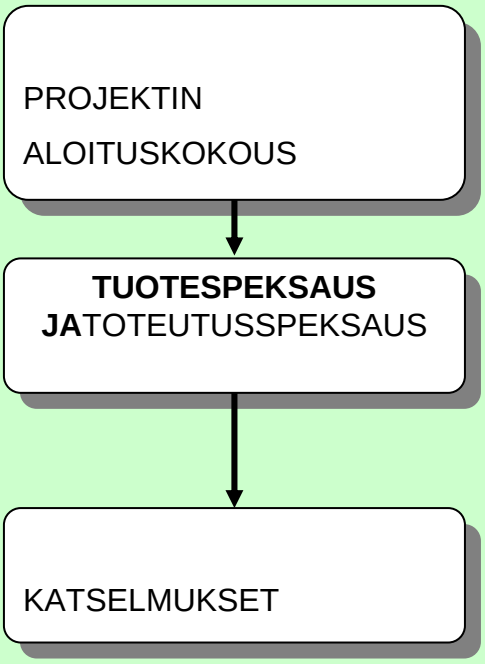
TARJOUS

Mikäli idea on toteutettavissa ja resursoitavissa, tehdään asiakkaalle tarjous.

SOPIMUS JA

Kun tarjous on hyväksytty, tehdään tarvittaessa sopimus ja tarkennettu projektisuunnitelma.

Tuotanto

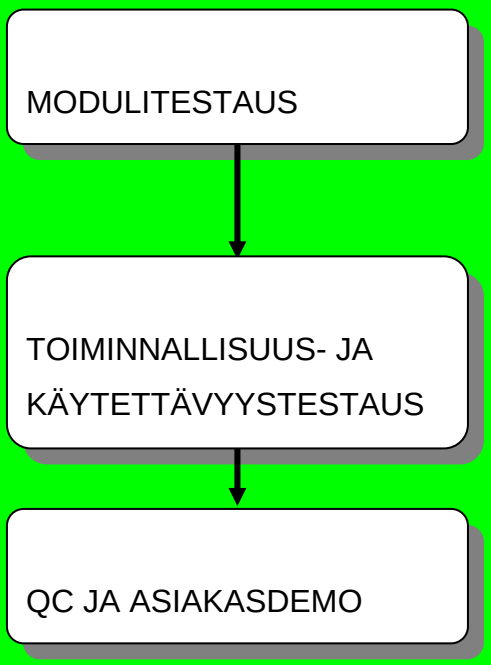


Tuotanto alkaa aloituskokouksessa, jossa on läsnä myös tilaajan edustaja. Tekijät on usein valittu jo määrittelyvaiheessa. Pienemmissä projekteissa aloituskokous ei ole välttämätön.

Tuotekuvausta teknisesti tarkempi on tuotespeksi, jossa toiminnallisuuksia on kuvattu syvemmin. Toteutusspeksi sisältää esimerkiksi UML- ja ER-mallinnukset.

Tuotannossa pidetään säännöllisesti katselmuksia ja aina jokaisen vaiheen päätteeksi. Vaiheenpäätöskatselmukseen saattaa osallistua myös tilaaja.

Testaus

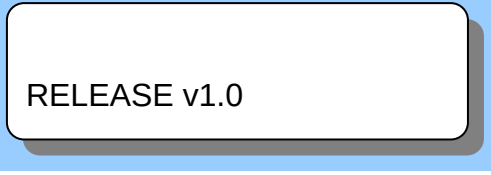


Tuotettua koodia tulee testata jatkuvasti. Kunkin modulin kehittäjä on vastuussa modulinsa toimivuudesta laadittujen speksien mukaisesti.

Ohjelmiston toimivuutta testataan myös kokonaisuutena. Testikatselmukset ovat tärkeä paikka kehitys- ja muutoskeskusteluille. Käytettävyystestaus voi olla omaa tai ulkoistettua (TKK).

Kun rutiinitestaukset on suoritettu, pidetään laadunvarmistuskatselmus. Tarvittaessa ennen luovutusta tai sen yhteydessä järjestetään asiakasdemo toimivalla tuotteella.

Luovutus tilaajalle



Luovutuksen yhteydessä pidetään projektin päätöspalaveri ja toimitetaan asiakkaalle tuote sekä tuotteeseen liittyvä oheismateriaali.

LIITE D

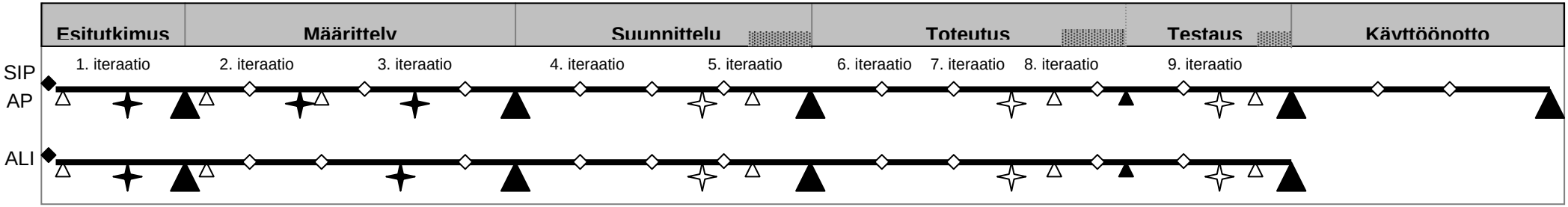
MAKROTASON PROSESSIKAAVIO

Makrotason prosessikaavio

Petteri Pyrrö 2.4.2001

Ohjedokumentit

- Projektin ohjausohje
- Ideadokumentin käyttöohje
- Määrittelyohje
- UML-mallinnusohje
- Infomates Oy:n arkkitehtuurin perusteet
- Ohjeistus oliosuunnitteluun
- ASP-infrastruktuuri
- Design Patterns
- Oliosuunnittelun tarkistuslista
- Ohjeistus ohjelmointiin (ohjelmointikäytännöt Infomates Oy:ssä)
- Ohjeistus testaukseen
- Opas käyttöohjeen tekemiseen
- Ylläpito-opas
- Tuotetukikansion koostamisohje



Vaiheartefaktit

YLEI-SET	Projektisuunnitelma 1. prototyyppi	Vaatimus- ja toiminnallisuusmäärittely 2. prototyyppi (tai useampi)	Arkkitehtuurikuvaus Toteutussuunnitelma	Sovelluskomponentit Komponenttitestausraportit Testaussuunnitelma	Valmis sovellus Testausraportit	Luovutuspöytäkirja
SIP	Ideakats. muistio				Käyttöohje Tuotetukikansio	
AP	Projektikuvaus (=Tuotekuvaus AP:ssa) Projektisopimus		Edistymisraportti	Edistymisraportti		Koulutuspalaute
ALI			Edistymisraportti	Edistymisraportti Integrointisuunnitelma		

◇ = Seurantapalaveri
= Mahd. muut esittelyajankohdat
= Vaihekatselmus
= Prototyypin esittely
= Suunnitelman luonti/tarkistus
SIP = Sisäinen tk-projekti
AP = Asiakasprojekti
ALI = Alihankintaprojekti
◆ = Aloituspalaveri

LIITE E

VAIHEKATSELMUSTEN TARKISTUSLISTAT

IDEAKATSELMUKSEN D.M.2001 TARKISTUSLISTA

Katselmuksen tarkoitus

Ideakatselmuksen tarkoituksena on käsitellä uudet tuoteideat ja päättää niiden hyödyntämisestä.

Idean XXX käsittely

Onko ideassa esitetyllä tuotteella, palvelulla tai menettelyllä olemassa kilpaileva tuote, palvelu tai vaihtoehtoinen menettely?

Kenelle ideasta on hyötyä?

Onko idea patentoitavissa?

Kuka tuotetta, palvelua tai menettelyä käyttää?

Kuka tuotteesta tai palvelusta maksaa?

Millä teknologialla idea on toteutettavissa?

Mitä toteuttaminen vaatii (arvio kustannuksista, työmäärästä, aikataulusta)?

Sopiiko idea Infomates Oy:n liiketoimintakonseptiin?

Ryhdytäänkö idean käsittelyssä jatkotoimenpiteisiin?

Jos jatkotoimenpiteisiin ryhdytään, mitä ne ovat? Aikataulu?

PROJEKTIN X MÄÄRITTELYKATSELMUKSEN TARKISTUSLISTA

Katselmuksen tarkoitus

Määrittelykatselmuksen tarkoituksena on käydä läpi toteutettavan sovelluksen määrittelydokumentaatio ja varmistaa, että kaikki asiakasvaatimukset on huomioitu ja että projektin osapuolet ovat yksimielisiä määrittelyn pohjalta toteutettavan sovelluksen ominaisuuksista ja toiminnallisuuksista sekä päättää etenemisestä seuraavaan vaiheeseen.

Määrittelydokumentin käsittely

Yleiset tarkistukset

Onko määrittelydokumentissa kaikki siihen kuuluvat osiot?

- Tähän määrittelydokumentin sisällysluettelo

Asiakasprojektin tarkistukset

Ovatko toiminnallisuudet asiakkaan tarpeiden mukaisia?

Ovatko kaikki toiminnallisuudet ja ominaisuudet toteutettavissa?

Toteutetaanko kaikki toiminnallisuudet ja ominaisuudet ensimmäisessä ohjelmistoversiossa?

Onko vaatimukset ymmärretty? Ovatko osapuolet yksimielisiä kuvatuista käyttötapauksista?

Sisäisen tuotekehitysprojektin tarkistukset

Tuottavatko toiminnallisuudet ideadokumentissa kuvatun kokonaisuuden kaikki tarpeet?

Ovatko kaikki toiminnallisuudet ja ominaisuudet toteutettavissa?

Toteutetaanko kaikki toiminnallisuudet ja ominaisuudet ensimmäisessä ohjelmistoversiossa?

Mitä palautetta on saatu tuotteistustiimiltä?

Onko sovellus edelleenkin todettu tarpeelliseksi ja kannattavaksi toteuttaa?

Alihankintaprojektin tarkistukset

Onko vaatimukset ymmärretty? Ovatko osapuolet yksimielisiä kuvatuista käyttötapauksista?

Määrittelyvaiheen tila

Onko määrittely toteutunut aikataulussa?

Mikä on määrittelyvaiheeseen käytetty työmäärä? Vastaako se suunniteltua?

Voidaanko edetä sovelluksen suunnitteluvaiheeseen?

Mikäli suunnitteluvaiheeseen ei voida edetä, mitkä ovat esteenä olevat tekijät?

Mikä on arvio lisämäärittelytyön, korjaustyön tai muun esteen aiheuttaman viiveen kestoajasta

PROJEKTIN X SUUNNITTELUKATSELMUKSEN TARKISTUSLISTA

Katselmuksen tarkoitus

Suunnittelukatselmuksen tarkoituksena on käydä läpi toteutettavan sovelluksen arkkitehtuuri ja toteutussuunnitelma sekä varmistaa, että kaikki määrittelydokumenteissa esitetyt toiminnallisuudet ja ominaisuudet on huomioitu. Lisäksi on varmistettava, että projektin osapuolet ovat yksimielisiä suunnittelun pohjalta toteutettavan sovelluksen suorituskyvystä, rakenteesta ja niistä toteutusmenetelmistä, joita toteutussuunnitelmassa on esitetty.

Katselmuksen tärkein päätös on, siirrytäänkö toteutusvaiheeseen vai jatketaanko suunnitteluartefaktien iterointia.

Suunnitteluartefaktien käsittely

Yleiset tarkistukset

Ovatko kaikki suunnitteluvaiheen artefaktit valmiit?

- Arkkitehtuurikuvaus
- Toteutussuunnitelma

Mitä palautetta on saatu tuotteistustiimiltä? Onko näköpiirissä liiketoiminnallisista seikoista johtuvia muutoksia?

Asiakasprojektin tarkistukset

Onko asiakkaalta tullut muutosesityksiä? Onko ne huomiotu suunnitteluartefakteissa?

Onko asiakas hyväksynyt viimeisimmän käyttöliittymäprototyypin?

Onko mahdolliset esille tulleet muutokset huomioitu suunnitteluartefakteissa?

Sisäisen tuotekehitysprojektin tarkistukset

Onko sovellus edelleenkin todettu tarpeelliseksi ja kannattavaksi toteuttaa?

Alihankintaprojektin tarkistukset

Onko tilaaja toimittanut muutosesityksiä?

Mikä on tilaajan näkemys tuotantovaiheeseen siirtymisestä?

Arkkitehtuurikuvauksen käsittely

Onko arkkitehtuurikuvauksen sisältö OK?

- (Tähän sisällysluettelo)

Onko määrittelyvaiheen artefakteissa esitetyt toiminnallisuudet ja ominaisuudet huomioitu?

Ovatko projektin osapuolet yksimielisiä arkkitehtuurin pohjalta toteutettavan sovelluksen rakenteesta ja suorituskyvystä?

Onko toteutustiimillä riittävät tiedot ja taidot arkkitehtuurissa esitettyjen teknologioiden käytöstä? Tarvitaanko koulutusta?

Toteutussuunnitelman käsittely

Onko toteutussuunnitelman sisältö OK?

- (Tähän sisällysluettelo)

Ovatko projektin osapuolet yksimielisiä niistä toteutusmenetelmistä, joita toteutussuunnitelmassa on esitetty?

Onko toteutussuunnitelmassa esitetty työnjako järkevä?

Suunnittelevaiheen tila

Onko suunnittelu toteutunut aikataulussa?

Mikä on suunnitteluvaiheeseen käytetty työmäärä? Vastaako se suunniteltua?

Voidaanko edetä projektin toteutusvaiheeseen?

Mikäli toteutusvaiheeseen ei voida edetä, mitkä ovat esteenä olevat tekijät?

Mikä on arvio lisäsuunnittelutyön, korjaustyön tai muun esteen aiheuttaman viiveen kestoajasta?

PROJEKTIN X TESTAUSKATSELMUKSEN TARKISTUSLISTA

Katselmuksen tarkoitus

Testauskatselmuksen tarkoituksena on käydä läpi sovelluksen toiminnallisuudet ja ominaisuudet ja varmistaa, että kaikki määrittelydokumenteissa esitetyt toiminnallisuudet ja ominaisuudet ovat olemassa.. Lisäksi on varmistettava, että projektin osapuolet ovat yksimielisiä siitä, että sovellus vastaa asetettuja tavoitteita ja soveltuu käyttötarkoitukseensa.

Katselmuksen tärkein päätös on, lanseerataanko tuote vai joudutaanko vielä iteroimaan.

Testausartefaktien käsittely

Yleiset tarkistukset

Ovatko kaikki tuotanto- ja testausvaiheen artefaktit valmiit?

- Sovelluksen komponentit
- Testausraportti (viimeisin, jossa ei ole enää todettu virheitä)
- Dokumentoitu lähdekoodi, JavaDoc

Onko versionhallinta ajan tasalla?

Ovatko projektin osapuolet yksimielisiä sovelluksen valmiusasteesta?

Asiakasprojektin tarkistukset

Onko asiakkaalta tullut muutosesityksiä? Onko ne huomiotu viimeisimmässä sovellus-versiossa?

Sisäisen tuotekehitysprojektin tarkistukset

Alihankintaprojektin tarkistukset

Onko tilaaja toimittanut muutosesityksiä? Onko ne huomiotu viimeisimmässä sovellus-versiossa?

Onko sovellus integroitavissa tässä vaiheessa tilaajan järjestelmään?

Mikä on tilaajan näkemys sovelluksen lanseeramisvalmiudesta? Entä aikataulusta?

Testausraportin käsittely

Onko testauksen aikana ilmennyt muutosesityksiä? Montako?

Montako virhetapausta kaikkiaan ilmoitettiin testauksen aikana?

Testausvaiheen tila

Onko toteutus ja testaus toteutunut aikataulussa?

Mikä on vaiheeseen käytetty työmäärä? Vastaako se suunniteltua?

Voidaanko tuote lanseerata?

Mikäli tuotetta ei vielä voida lanseerata, mitkä ovat esteenä olevat tekijät?

Mikä on arvio korjaus- tai muutostyön tai muun esteen aiheuttaman viiveen kestosta?

LIITE F

ARTEFAKTIT UUDISTETUSSA TUOTANTOPROSESSISSA

Artefaktit Infomates Oy:n uudistetussa tuotantoprosessissa			
VAIHE	**Sisäinen tuotekehitysprojekti**	**Asiakasprojekti**	**Alihankintaprojekti**
ESI-TUTKIMUS	Ideakatselmuksen muistio Projektisuunnitelma 1. prototyyppi (jos mahdollista)	Projektisopimus Projektisuunnitelma Tuotekuvaus 1. prototyyppi	Projektisopimus Projektisuunnitelma 1. prototyyppi (mikäli tarpeen)
MÄÄRITTELY	Vaatimusmäärittely Toiminnallisuusmäärittely Arkkitehtuurikuvaus 1. tai 2. Prototyyppi (useampi, jos tarpeen)	Vaatimusmäärittely Toiminnallisuusmäärittely Arkkitehtuurikuvaus 2. tai useampi prototyyppi	Vaatimusmäärittely Toiminnallisuusmäärittely Arkkitehtuurikuvaus 1. prototyyppi (useampi, jos tarpeen)
SUUNNITTELU	Arkkitethtuurikuvaus (sisältäen mallin) Toteutussuunnitelma	Arkkitethtuurikuvaus (sisältäen mallin) Toteutussuunnitelma Edistymisraportti	
TOTEUTUS	Sovelluskomponentit JavaDoc	Sovelluskomponentit JavaDoc Edistymisraportti (asiakkaalle)	Sovelluskomponentit JavaDoc Edistymisraportti (tilaajalle) Integrointisuunnitelma
TESTAUS	Valmis sovellus Testausraportit Tuotetukikansio Käyttöohje		Valmis sovellus Testausraportit Tuotetukikansio Käyttöohje Luovutuspöytäkirja
KÄYTTÖÖN-OTTO	-	Koulutuspalaute Luovutuspöytäkirja	-
TYÖN-JOHTO	Projektisuunnitelma Vaiheraportit Seurantapalaverimuistiot Projektiraportti		

LIITE G

PROJEKTIN VAIHEIDEN TAVOITTEET, TEHTÄVÄT JA ARVIOINTIKRITEERIT

PROJEKTIN VAIHEIDEN TAVOITTEET, TEHTÄVÄT JA ARVIOINTIKRITEERIT

Tässä dokumentissa on taulukoitu Infomates Oy:n ohjelmistotuotantoprosessin vaiheiden tärkeimmät tavoitteet, olennaisimmat tehtävät ja tärkeimmät arviointikriteerit kaikissa kolmessa projektityypissä: sisäisessä tuotekehitysprojektissa (SIP), asiakas-projektissa (AP) ja alihankintaprojektissa (ALI). Edellämainittujen projektityyppien yläpuolella olevaan kenttään on kirjattu kaikille projektityypeille yhteiset seikat.

ESITUTKIMUSVAIHE	
Tärkeimmät tavoitteet	Päätös projektin aloittamisesta. Koko projektin työmäärän, resurssien ja aikataulun arviointi. Arvio projektin kannattavuudesta. Osapuolten välinen yhteisymmärrys projektin tavoitteista. Potentiaalisten riskien arviointi. SIP Arvio tuotoksen luonteesta: onko se sovitettavissa johonkin yhtiön tuoteperheeseen? AP Arvio projektissa syntyvän sovelluksen luonteesta: onko saatavissa geneerinen tuote eli muillekin vastaaville asiakkaille tai markkinasegmentille soveltuva tuote, vai onko kysessä erikoissovellus? Arvio tuotoksen luonteesta: onko se sovitettavissa johonkin yhtiön tuoteperheeseen? ALI Arvio yhtiön kyvystä ryhtyä projektiin.
Olennaisimmat tehtävät	Projektisuunnitelman laadinta. Projektikuvauksessa kerrotaan projektin tavoitteet, osapuolet, osapuolten tehtävät ja projektista saatavat hyödyt sekä arvioidaan työmäärä, resurssit ja aikataulu. SIP - AP Tuotekuvauksen laadinta. Sopimuksen laadinta määrittelyvaiheesta. Mikäli tarjotaan jo koko projektia, projektisopimuksen laadinta. Käyttöliittymäprototyypin laadinta tässä vaiheessa, mikäli asiakas sitä edellyttää ennen projektin aloittamista.

	ALI
	Selvitys yhtiön resurssien riittävyydestä.
Tärkeimmät arviointikriteerit	Ovatko kaikki osapuolet yksimielisiä projektisuunnitelman sisällöstä eli tavoitteista, tehtävistä, hyödyistä sekä työmäärä-, resurssi- ja aikatauluarvioista?
	Ovatko kaikki osapuolet yksimielisiä alustavasta kustannusarviosta?
	Ovatko kustannus- ja aikatauluarviot, prioriteetit, riskit sekä käytettävät menetelmät uskottavia?
	SIP
	Ideakatselmuksen tarkistuslistan arviointikriteerit.
	AP
	Onko tuotteesta mahdollista saada geneerinen?
	Kenelle tuotosten IPR:t (tuote- ja tekijänoikeudet) kuuluvat?
	Onko tuotettava sovellus sovitettavissa johonkin yhtiön tuoteperheeseen?
	Onko tarjous määrittelytyöstä laadittu realististen kustannusarvioiden pohjalta?
	Ovatko kaikki osapuolet yksimielisiä tuotekuvauksen sisällöstä?
	Tarviiko asiakas konsultointia ennen määrittelyjen aloittamista? Joissakin tapauksissa voi kysessä olla tilanne, että asiakkaalla ei ole aiempaa kokemusta tietotekniikan hyödyntämisestä työssään tai liiketoiminnassaan.
	Mikäli tarjotaan jo koko projektia, ovatko työmäärä-, resurssi-, aikataulu- ja kustannusarviot uskottavia?
	Mikäli tarjotaan koko projektia, ovatko osapuolet yksimielisiä projektisopimuksen sisällöstä?
	Mikäli käyttöliittymäprototyyppi on laadittu, on arvioitava, onko prototyypissä esitettynä asiakkaan odottamat tärkeimmät toiminnallisuudet ja ominaisuudet?
	ALI
	Kattaako alihankintaprojektin tilaajan tarjous arvioidut projektikustannukset riittävällä katteella?
	Kenelle alihankintaprojektista syntyvien tuotosten IPR:t kuuluvat?
	Voidaanko ja saadaanko alihankintaprojektissa syntyviä tuotoksia hyödyntää yhtiön muissa sovelluksissa, projekteissa tai toiminnassa?
	Missä määrin yhtiön täytyy osallistua inegrointityöhön?
	Mitkä ovat yhtiön vastuut tuotosten toiminnallisuudesta?

MÄÄRITTELYVAIHE	
Tärkeimmät tavoitteet	Osapuolten välinen yhteisymmärrys projektin tuotosten sisällöstä. Määrittelyjen kerääminen niin pian ja tarkasti kuin käytännöllistä. Koko projektin tavoitteiden, työmäärän, resurssien ja aikataulun tarkistus. Potentiaalisten riskien arviointi ja minimointi. Muutostenhallinta. SIP - AP Toteutettavan sovelluksen toiminnallisuuksien ja ominaisuuksien määrittely geneerisiksi, mikäli mahdollista. Tuotosten sovittaminen yhtiön tuoteperheeseen, mikäli mahdollista. ALI -
Olennaisimmat tehtävät	Vaatimus- ja toiminnallisuusmäärittelyn laadinta. Määrittelydokumentaatiossa kuvataan tuotteen käyttötarkoitus ja käyttäjät, kerrotaan kaikki vaadittavat ominaisuudet, toiminnallisuudet, kriittisimmät käyttötapaukset, sidokset, kapasiteetti- ja ympäristövaatimukset sekä tekniset ja lailliset rajoitteet. Alustava esitys käytettävästä arkkitehtuurista (erityisesti, mikäli poiketaan yhtiön perinteisestä sovellusarkkitehtuurista). Kattavan käyttöliittymäprototyypin laadinta. Prototyypillä täytyy pystyä esittämään kriittisimpien käyttötapausten kulku sovelluksessa ja kriittisimpien tietojen käsittely. SIP Kriittisimpien käyttötapausten läpikäyminen käyttöliittymäprototyypin avulla. AP Vaatimusten kerääminen yhdessä asiakkaan kanssa. Kriittisimpien käyttötapausten läpikäyminen käyttöliittymäprototyypin avulla sovelluksen tulevien käyttäjien kanssa. ALI Määrittelyjen kerääminen ja tarkistaminen yhdessä tilaajan kanssa.

Tärkeimmät arviointikriteerit	Ovatko kaikki osapuolet yksimielisiä määrittelydokumentaation sisällöstä eli käyttötarkoituksesta, käyttäjistä, käyttötapauksista, toiminnallisuuksista, ominaisuuksista, sidoksista, kapasiteetti- ja ympäristövaatimuksista sekä rajoitteista?
	Onko projektin tavoite stabiili?
	Ovatko kaikki osapuolet edellenkin yksimielisiä tuotosten sisällöstä?
	Ovatko kaikki osapuolet yksimielisiä tarkennetusta kustannusarviosta?
	Ovatko kustannus- ja aikatauluarviot, prioriteetit, riskit sekä käytettävät menetelmät edelleenkin uskottavia?
	Käykö prototyypin esittelystä ilmi, että suurimmat riskit on huomioitu?
	Onko prototyypissä huomioitu kriittisimmät käyttötapaukset ja tietojen oikea käsittely?
	Onko viimeisimmät mahdolliset muutokset huomioitu? Onko muutoksilla kriittisiä vaikutuksia työmääriin, resursseihin tai aikatauluun?
	Muut määrittelykatselmuksen tarkistuslistassa esitetyt arviointikriteerit.
	SIP
	-
	AP
	Onko prototyypissä esitetty projektisopimukseen sisällytetyt toiminnallisuudet ja ominaisuudet?
	Ovatko kaikki osapuolet yksimielisiä prototyypin käytettävyydestä?
	Voidaanko mahdolliset muutokset sisällyttää sopimukseen?
	ALI
	Voidaanko mahdolliset muutokset sisällyttää sopimukseen?

SUUNNITTELUVAIHE	
Tärkeimmät tavoitteet	Luoda käytettävissä olevien tietojen nojalla kattava arkkitehtuuri- ja komponenttitason suunnitelma, jonka pohjalta sovellus voidaan toteuttaa niin pian kuin käytännöllistä. Tarkan toteutussuunnitelman luonti toteutusvaiheen läpiviemiseksi. Vakuuttuminen siitä, että kaikki suunnitelmat tukevat asetettuja tavoitteita järkevin kustannuksin ja järkevässä aikataulussa. Potentiaalisten riskien arviointi ja minimointi. Muutostenhallinta. SIP - AP - ALI -
Olennaisimmat tehtävät	Tavoitteiden tarkka määrittäminen. Tähän sisältyy arkkitehtuuriin ja suunnitteluun liittyviin valintoihin ja päätöksiin vaikuttavien sovelluksen käyttötapausten ymmärtäminen ja kokonaisuuden hahmottaminen. Prosessin ja infrastruktuurin tarkka määrittäminen. Tuotantoprosessin, välietappien ja niiden arviointikriteerien muodostaminen. Arkkitehtuurikuvauksen laadinta. Esityskelpoisen arkkitehtuuriprototyypin valmistaminen. Tarkoitus on osoittaa erityisesti uusien ideoiden ja ratkaisujen toimivuus. Toteutussuunnitelman laadinta. Olemassa olevien komponenttien mahdollisimman kattava hyödyntäminen. Päätökset itse toteutettavista ja mahdollisesti ostettavista komponenteista tai sovittimista. SIP - AP - ALI Integrointisuunnitelman valmistelut.

Tärkeimmät arviointikriteerit	Eteneekö suunnittelu aikataulussa?
	Vastaavatko työmäärät suunniteltua?
	Onko projektin tavoite stabiili?
	Onko valittu arkkitehtuuri stabiili?
	Ovatko kustannus- ja aikatauluarviot, prioriteetit, riskit ja käytettävät menetelmät uskottavia?
	Kattaako ja kuvaako tehty arkkitehtuuriprototyyppi edellä mainitut kriteerit? (Arkkitehtuuriprototyypistä suurin saatava hyöty on se, että sen avulla sovelluksen puitteet voidaan ymmärtää sekä arvioida prototyypin luoneen tiimin uskottavuus ja kyky ratkaista teknisiä ongelmia).
	Käykö prototyypin esittelystä ilmi, että suurimmat riskit on huomioitu ja uskottavasti ratkaistu?
	Onko tuotantosuunnitelma riittävän tarkka? Tukeutuuko se uskottaviin arvioihin?
	Ovatko kaikki osapuolet yksimielisiä siitä, että esitetyt tavoitteet ja suunnitelmat voidaan saavuttaa viemällä toteutussuunnitelma käytäntöön esitetyn arkkitehtuurin mukaisesti?
	Onko viimeisimmät mahdolliset muutokset huomioitu? Onko muutoksilla kriittisiä vaikutuksia työmääriin, resursseihin tai aikatauluun?
	Ovatko todelliset resurssikustannukset suunniteltuihin verrattuna hyväksyttäviä?
	Muut suunnittelukatselmuksen tarkistuslistassa esitetyt arviointikriteerit.
	SIP
	-
	AP
	Voidaanko mahdolliset muutokset sisällyttää sopimukseen?
	ALI
	Onko kaikilla osapuolilla yhteinen näkemys integrointirajapinnoista ja niiden toiminnallisuudesta?
	Voidaanko mahdolliset muutokset sisällyttää sopimukseen?

TOTEUTUSVAIHE	
Tärkeimmät tavoitteet	Tuotantokustannusten minimointi optimoimalla käytettävissä olevat resurssit ja välttämällä ylimääräistä työtä. Riittävän laadun saavuttaminen niin nopasti kuin käytännöllistä. Käyttökelpoisten sovellusversioiden (alfa- ja betaversioiden) tuottaminen niin nopeasti kuin käytännöllistä. Potentiaalisten riskien arviointi ja minimointi. Muutostenhallinta. SIP - AP - ALI -
Olennaisimmat tehtävät	Resurssienhallinta, työnjohto sekä prosessin optimointi. Sovelluskomponenttien toteutus. Toteutuksen laadun tarkkailu teknologiakatselmuksien avulla. Komponenttitestaus. SIP AP Raportointi edistymisestä asiakkaalle. ALI Raportointi edistymisestä tilaajalle. Integrointisuunnitelman toteutus.
Tärkeimmät arviointikriteerit	Eteneekö toteutus aikataulussa? Vastaavatko työmäärät suunniteltua? Onko komponenttien valmiusaste riittävä järjestelmätestaukseen siirtymiseksi? Onko komponentit toteutettu valitun arkkitehtuurin edellyttämällä tavalla? Ovatko komponentit uudelleenkäytettäviä? SIP - AP ALI -

TESTAUSVAIHE	
Tärkeimmät tavoitteet	Sovelluksen saattaminen lanseerausvalmiiksi.
	Osapuolten keskinäisen yksimielisyyden saavuttaminen siitä, että tuotannon tavoitteet on saavutettu asetettujen kriteereiden ja tavoitteiden mukaisesti.
	SIP
	-
	AP
	ALI
	-
Olennaisimmat tehtävät	Testaussuunitelman laadinta.
	Sovelluksen siirto ASP-alustalle.
	Sovelluksen kokonaistoiminnan perusteellinen testaaminen.
	Käyttöönottovaiheen valmisteleminen.
	Ylläpidon valmisteleminen. Tähän kuuluvat ylläpitohenkilöiden valinta ja tarvittavien ylläpitotoimien suunnittelu. (Ylläpito alkaa välittömästi lanseerauksen jälkeen)!
	SIP
	Sovelluksen käyttöohjeen toteutus.
	Sovelluksen tuotetukikansion koostaminen.
	AP
	Sovelluksen asentaminen asiakkaan ympäristöön, mikäli kysessä ei ole ASP-sovellus.
	Sovelluksen käyttöohjeen toteutus.
	Sovelluksen tuotetukikansion koostaminen.
	ALI
	Sovelluksen tai järjestelmän integroiminen tilaajan ympäristöön.

Tärkeimmät arviointikriteerit	Eteneekö testaus aikataulussa?
	Vastaavatko työmäärät suunniteltua?
	Vastaako tuote määrittelydokumenttien kuvausta?
	Yltävätkö tuotteen ominaisuudet tavoitetasolle?
	Ovatko kaikki osapuolet yksimielisiä tuotosten valmiusasteesta?
	Ovatko käyttöönottovaiheen valmistelut kunnossa – onko käyttöohje ja tuotetukikansio valmis?
	Onko ylläpitotoiminnot valmisteltu?
	Täyttääkö tuote yhtiön laatukriteerit?
	Voidaanko tuote lanseerata?
	Ovatko kaikki osapuolet valmiita lanseeraukseen?
	Vastaavatko todelliset tuotantokustannukset ennakoituja kustannuksia?
	Muut testauskatselmuksen tarkistuslistassa esitetyt arviointikriteerit.
	SIP
	Onko tuotteen käyttöönottoon kuuluvat valmistelut valmiina samanaikaisesti tuotteen kanssa?
	Voidaanko tuotteen myynti aloittaa?
	AP
	Onko asiakas tyytyväinen?
	Onko tuotteen käyttöönottoon kuuluvat valmistelut valmiina samanaikaisesti tuotteen kanssa?
	ALI
	Onko tilaaja tyytyväinen?
	Onnistuiko integrointi suunnitellulla tavalla?
	Onko asiakas tyytyväinen?
	Vastaavatko projektin todelliset kustannukset ennakoituja kustannuksia?
	Ovatko todelliset kustannukset verrattuna suunniteltuihin kustannuksiin hyväksyttäviä?
	Vastaavatko toteutuneet työmäärät suunniteltua?

KÄYTTÖÖNOTTOVAIHE	
Tärkeimmät tavoitteet	Projektin tuotosten lanseeraus eli toimittaminen asiakkaille.
	SIP
	AP
	Tukea asiakasta saamaan sovelluksesta odotettu hyöty.
	ALI
	-
Olennaisimmat tehtävät	Tuotosten lanseeraus eli toimittaminen käyttäjille.
	SIP
	Tuotteen myyntityön aloittaminen.
	AP
	Tarvittavien asiakastunnusten luonti. Käyttöönottokoulutus. Tuotetukikansion esittely.
	ALI
	-
Tärkeimmät arviointikriteerit	Onko asiakas tyytyväinen? Vastaavatko projektin todelliset kustannukset ennakoituja kustannuksia? Ovatko todelliset kustannukset verrattuna suunniteltuihin kustannuksiin hyväksyttäviä? Vastaavatko toteutuneet työmäärät suunniteltua? Muut käyttöönottokatselmuksen tarkistuslistassa esitetyt arviointikriteerit.
	SIP
	Onko tuotteen myynti vastannut odotuksia?
	AP
	Onko asiakas- ja käyttäjätunnuksia luotu riittävästi? Onnistuiko käyttökoulutus – osaavatko käyttäjät käyttää sovellusta? Osaavatko käyttäjät käyttää tuotetukikansiota?
	ALI
	-

LIITE H

OHJELMISTOTUOTANTOPROSESSISSA TARVITTAVAT TIIMIT JA ROOLIT

OHJELMISTOTUOTANTOPROSESSISSA TARVITTAVAT TIIMIT JA ROOLIT

Tähän dokumenttiin on koostettu Alistair Cockburnin kehittämällä iso-M ja pieni m -metodologialla ohjelmistoprojektissa ja ohjelmistotuotantoprosessissa tarvittavia rooleja. Roolit on koostettu kukin omaan taulukkoonsa. Lisäksi on tehty tiimikohtainen taulukko, josta näkyvät tiimissä tarvittavat roolit, tiimin tuotokset sekä tiimin tarvitsemat artefaktit.

ESITUTKINTATIIMI		
Roolit	**Tarvittavat artefaktit**	**Tuotettavat artefaktit**
Määrittelijä Bisnesasiantuntija Projektipäällikkö Ohjelmistoarkkitehti	-	Ideakatselmuksen muistio Projektikuvaus Talousarvio Tarjous (tai sopimus)
Tehtävät		
Projektikuvauksen tekeminen. Projektikuvaus sisältää arvion projektin työmääristä, resurssitarpeista, toteutusaikatalulusta, riskeistä ja kustannuksista. Projektikuvauksen liitteeksi projektin alustava talousarvio (sisäisessä tuotekehitysprojektissa ja asiakasprojektissa). Tarjouksen laatiminen asiakkaalle (asiakasprojekti). Ideakatselmuksen järjestäminen (sisäinen tuotekehitysprojekti).		

MÄÄRITTELYTIIMI		
Roolit	**Tarvittavat artefaktit**	**Tuotettavat artefaktit**
Määrittelijä Bisnesasiantuntija Käyttäjä (asiakas)	Asiakasvaatimukset	Vaatimusmäärittely Toiminnallisuusmäärittely Käyttötapaukset
Tehtävät		
Määrittelyjen kerääminen. Tähän tehtävään sisältyy käyttäjien eli asiakkaan käyttäjien haastatteleminen hyödyntäen yhtiössä sovittuja käytäntöjä. Käyttötapausten koostaminen. Vaatimus- ja toiminnallisuusmäärittelyn laatiminen. Mahdollisten lisämäärittelyjen tekeminen suunnitteluvaiheen aikana.		

ARKKITEHTITIIMI		
Roolit	**Tarvittavat artefaktit**	**Tuotettavat artefaktit**
Ohjelmistoarkkitehti Järjestelmäarkkitehti Ohjelmistosuunnittelija Tietokantasuunnittelija Määrittelijä	Vaatimusmäärittely Toiminnallisuusmäärittely	Komponenttikaaviot Collaboration diagram Deployment diagram Järjestelmäkaavio
Tehtävät		
Alustavan esityksen laatiminen toteutettavan sovelluksen arkkitehtuurista määrittelyvaiheessa. Olioanalyysin suorittaminen määrittelytiimin artefakteille (toiminnallisuus- ja vaatimusmäärittely sekä käyttötapaukset). Esityskelpoisen arkkitehtuuriprototyypin valmistaminen. Tarkoitus on osoittaa erityisesti uusien ideoiden ja ratkaisujen toimivuus.		

SUUNNITTELUTIIMI		
Roolit	**Tarvittavat artefaktit**	**Tuotettavat artefaktit**
Ohjelmistoarkkitehti Ohjelmistosuunnittelija Tietokantasuunnittelija	Komponenttikaaviot Collaboration diagram Deployment diagram Järjestelmäkaavio Vaatimusmäärittely Toiminnallisuusmäärittely	Toteutussuunnitelma
Arkkitehtitiimin artefaktien analysointi. Tavoitteiden tarkka määrittäminen. Tähän sisältyy arkkitehtuuriin ja suunnitteluun liittyviin valintoihin ja päätöksiin vaikuttavien sovelluksen käyttötapausten ymmärtäminen ja kokonaisuuden hahmottaminen. Toteutussuunnitelman laadinta. Olemassa olevien komponenttien mahdollisimman kattava hyödyntäminen. Päätökset itse toteutettavista ja mahdollisesti ostettavista komponenteista tai sovittimista.		

TOTEUTUSTIIMI		
Roolit	**Tarvittavat artefaktit**	**Tuotettavat artefaktit**
Ohjelmistosuunnittelija Tietokantasuunnittelija Käyttöliittymäsuunnittelija Ohjelmoija	Toteutussuunnitelma Komponenttikaaviot Collaboration diagram Deployment diagram Järjestelmäkaavio	Prototyypit Luokkakaavio Ohjelmistokomponentit JavaDoc Tietokannat Käyttöliittymä
Tehtävät		
Prototyyppien toteuttaminen esitutkimus-, määrittely- ja suunnitteluvaiheessa. Sovelluskomponenttien suunnittelu. Sovelluskomponenttien toteutus mukaan lukien tietokannat, käyttöliittymä ja erilaiset rajapintakomponentit (esimerkiksi tietoliikennekomponentit). Toteutuksen laadun tarkkailu teknologiakatselmuksien avulla. Komponenttitestaus. Sovelluksen siirto ASP-alustalle.		

TESTAUSTIIMI		
Roolit	**Tarvittavat artefaktit**	**Tuotettavat artefaktit**
Testaaja Käyttäjä Määrittelijä	Käyttöohje Käyttötapaukset	Testaussuunnitelma Testausraportti Muutosehdotukset
Tehtävät		
Testaussuunitelman laadinta. Sovelluksen kokonaistoiminnan perusteellinen testaaminen. Testausraportin ja mahdollisten muutosehdotusten koostaminen.		

LANSEERAUSTIIMI		
Roolit	**Tarvittavat artefaktit**	**Tuotettavat artefaktit**
Dokumentoija Käyttöönottokouluttaja Bisnesasiantuntija	Sovellus	Käyttöohje Tuotetukikansio Käyttöönoton koulutussuunnitelma
Tehtävät		
Käyttöohjeen laadintaan liittyvän sisällön kerääminen määrittelyvaiheessa määrittelydokumentaatiosta. Käyttöohjeen laatiminen. Tuotetukikansion koostaminen. Käyttöönoton koulutussuunnitelman laatiminen. Ylläpidon valmisteleminen. Tähän kuuluvat ylläpitohenkilöiden valinta ja tarvittavien ylläpitotoimien suunnittelu yhdessä valittujen ylläpitohenkilöiden ja toteutustiimin kanssa. (Ylläpito alkaa välittömästi lanseerauksen jälkeen)!		

KÄYTTÖÖNOTTOTIIMI		
Roolit	**Tarvittavat artefaktit**	**Tuotettavat artefaktit**
Käyttöönottokouluttaja	Käyttöönoton koulutussuunnitelma Käyttöohje Tuotetukikansio Sovellus	Käyttöönottokoulutus
Tehtävät		
Käyttöönottokoulutus. Tuotetukikansion esittely.		

ROOLI		
Bisnesasiantuntija		
Tuotokset	**Tekniikat, menetelmät**	**Tehtävät**
Sovelluksen liiketoimintasuunnitelma	Haastattelut Neuvottelut, neuvottelutekniikat	Asiakkaan tarpeiden kerääminen Sovelluksen liiketoimintasuunnitelman toteutus
Käytettävät standardit	**Tarvittavat työkalut**	**Tarvittavat taidot**
Liiketoimintasäädökset Yhtiön liiketoimintatavat	Riittävän laadukas tekstinkäsittelyohjelmisto (MS Word) Riittävän laadukas taulukkolaskentasovellus (MS Excel)	Liiketoiminta Kommunikointi

ROOLI		
Dokumentoija		
Tuotokset	**Tekniikat, menetelmät**	**Tehtävät**
Käyttöohje	Sovelluksen käyttäminen ohjeiden mukaan Sovelluksen käyttäminen tahallisesti väärin	Asiakasdokumentointi
Käytettävät standardit	**Tarvittavat työkalut**	**Tarvittavat taidot**
Yhtiön sovitut dokumentointimenetelmät	Sovellus Riittävän laadukas tekstinkäsittelyohjelmisto (MS Word) PDF-muunnin (Adobe Acrobat)	Sovelluksen tuntemus Äidinkielen hallinta Dokumentointi Tarkkuus

ROOLI		
Järjestelmäarkkitehti		
Tuotokset	**Tekniikat, menetelmät**	**Tehtävät**
Järjestelmäkaavio	Semanttinen piirtäminen ja mallinnus Semanttinen kirjoittaminen	Tietojärjestelmän valinta Tietojärjestelmän suunnittelu
Käytettävät standardit	**Tarvittavat työkalut**	**Tarvittavat taidot**
UML, SQL, LDAP	UML-mallinnustyökalu (Argo/UML, MagicDraw) tai riittävän laadukas piirto-ohjelmisto (MS Word) Tietokantapalvelin Hakemistopalvelin	UML Tietojärjestelmät: tietokannat (SQL, Oracle, MySQL), hakemisto-palvelimet (LDAP) Visualisointi Kommunikointi

ROOLI		
Käyttöliittymäsuunnittelija		
Tuotokset	**Tekniikat, menetelmät**	**Tehtävät**
Käyttöliittymämallit Käyttöliittymät	Piirtäminen, mallinnus	Käyttöliittymän suunnittelu Käyttöliittymän toteutus
Käytettävät standardit	**Tarvittavat työkalut**	**Tarvittavat taidot**
Yhtiön sovitut käyttöliittymästandardit HTML, XML, CSS	Helppokäyttöinen, riittävän laadukas piirto-ohjelmisto (MS Powerpoint, CorelDraw) Riittävän laadukas kuvankäsittelyohjelmisto (Adobe Photoshop tms.) HTML-editori	Käyttöliittymäsuunnittelu Käytettävyys Visualisointi HTML, XSL, CSS

ROOLI		
Määrittelijä		
Tuotokset	**Tekniikat, menetelmät**	**Tehtävät**
Vaatimusmäärittely Toiminnallisuusmäärittely Käyttötapaukset Käyttötapauskaaviot	Haastattelut Semanttinen piirtäminen, mallinnus Semanttinen kirjoittaminen	Vaatimusten kerääminen Toiminnallisuuksien ja ominaisuuksien määritteleminen
Käytettävät standardit	**Tarvittavat työkalut**	**Tarvittavat taidot**
FDD UML	Riittävän laadukas tekstinkäsittelyohjelmisto (MS Word) UML-mallinnustyökalu (Argo/UML, MagicDraw) tai riittävän laadukas piirto-ohjelmisto (MS Word)	Kommunikointi Verbaalisuus Visualisointi Tarkkuus UML behavioral diagrams

ROOLI		
Ohjelmistoarkkitehti		
Tuotokset	**Tekniikat, menetelmät**	**Tehtävät**
Toteutussuunnitelma Komponenttikaaviot Collaboration diagram Deployment diagram	Semanttinen piirtäminen ja mallinnus Semanttinen kirjoittaminen	Arkkitehtuurin valinta Arkkitehtuurin suunnittelu
Käytettävät standardit	**Tarvittavat työkalut**	**Tarvittavat taidot**
UML Yhtiön sovitut Design Patternsit Yhtiön sovitut ohjelmointi-menetelmät	Riittävän laadukas tekstinkäsittelyohjelmisto (MS Word) UML-mallinnustyökalu (Argo/UML, MagicDraw) tai riittävän laadukas piirto-ohjelmisto (MS Word)	Oliosuunnittelu, olioanalyysi UML Abstraktioiden hahmottaminen ja luominen Design Patterns Kommunikointi

ROOLI		
Ohjelmistosuunnittelija		
Tuotokset	**Tekniikat, menetelmät**	**Tehtävät**
Ohjelmistokomponentit Luokkakaaviot JavaDoc	Standardin mukainen ohjelmointi Semanttinen piirtäminen	Ohjelmistokomponenttien toteutus Ohjelmistokomponenttien suunnittelu
Käytettävät standardit	**Tarvittavat työkalut**	**Tarvittavat taidot**
Java UML Yhtiön sovitut ohjelmointi-menetelmät	IDE (Kawa) UML-mallinnustyökalu (Argo/UML, MagicDraw) tai riittävän laadukas piirto-ohjelmisto (MS Word)	Olio-ohjelmointi Java UML

ROOLI		
Projektipäällikkö		
Tuotokset	**Tekniikat, menetelmät**	**Tehtävät**
Projektisuunnitelma Työnjakokaavio Tilannekatsaukset	Johtamismenetelmät Palaverit	Projektinhallinta Seurantapalaverien järjestäminen ja johtaminen Katselmusten järjestäminen ja johtaminen
Käytettävät standardit	**Tarvittavat työkalut**	**Tarvittavat taidot**
Yhtiössä määritelty prosessi Yhtiön sovitut projektimenetelmät	Riittävän laadukas tekstinkäsittelyohjelmisto (MS Word) Projektinhallintaohjelmisto (MS Project)	Johtamistaidot Kommunikointi Tarkkuus

ROOLI		
Testaaja		
Tuotokset	**Tekniikat, menetelmät**	**Tehtävät**
Testausraportti	Sovelluksen käyttäminen ohjeiden mukaan Sovelluksen käyttäminen tahallisesti väärin	Sovelluksen testaaminen
Käytettävät standardit	**Tarvittavat työkalut**	**Tarvittavat taidot**
Yhtiön sovitut testaus-menetelmät	Sovellus	Sovelluksen tuntemus Tarkkuus

ROOLI		
Tietokantasuunnittelija		
Tuotokset	**Tekniikat, menetelmät**	**Tehtävät**
Tietokannat Luokkakaaviot JavaDoc	Standardin mukainen tietokantaohjelmointi Semanttinen piirtäminen	Tietokantojen toteutus Titetokantojen suunnittelu
Käytettävät standardit	**Tarvittavat työkalut**	**Tarvittavat taidot**
SQL, Java, UML Yhtiön sovitut ohjelmointi-menetelmät	MySQL, Oracle IDE (Kawa) UML-mallinnustyökalu (Argo/UML, MagicDraw) tai riittävän laadukas piirto-ohjelmisto (MS Word)	SQL, JDBC Java, EJB

ROOLI		
Käyttäjä (asiakas)		
Tuotokset Ideat Käyttökokemukset Palaute	**Tekniikat, menetelmät** Sovelluksen normaali käyttö Osallistuminen määrittelyihin ja testaukseen	**Tehtävät** Määrittely Sovelluksen testaus
Käytettävät standardit Ei ole.	**Tarvittavat työkalut** Sovellus	**Tarvittavat taidot** Oman alan tuntemus Oma työtehtävä Aiemman vastaavan sovelluksen käyttökokemus

ROOLI		
Käyttöönottokouluttaja		
Tuotokset Koulutussuunnitelma Koulutusmateriaali Koulutus	**Tekniikat, menetelmät** Pedanttiset ja didaktiset menetelmät	**Tehtävät** Käyttäjien koulutus
Käytettävät standardit Ei ole.	**Tarvittavat työkalut** Riittävän laadukas tekstinkäsittelyohjelmisto (MS Word) Presentaation luontityökalu (MS Powerpoint) Presentaatiovälineet (dataprojektori, tietokone) Sovellus	**Tarvittavat taidot** Esiintymistaidot Kommunikointi Sovelluksen riittävä tuntemus

ROOLI		
Ohjelmoija		
Tuotokset Ohjelmistokomponentit JavaDoc	**Tekniikat, menetelmät** Standardin mukainen ohjelmointi	**Tehtävät** Ohjelmistokomponenttien toteutus
Käytettävät standardit Java Yhtiön sovitut ohjelmointi-menetelmät	**Tarvittavat työkalut** IDE (Kawa)	**Tarvittavat taidot** Olio-ohjelmointi Java UML

LIITE I

ROOLIT, TAIDOT JA TEKNIIKAT

Alistair Cockburnin [10] esimerkinomaisia roolikuvauksia (suom. kirjoittajan).

Rooli	Selite	Taidot	Tekniikat
Bisnes-ekspertti	Kokenut liikemies. Tietää, kuinka bisnes toimii, ja osaa vastata mm. kysymyksiin kuinka hommat tehdään, mikä on pysyvää, mikä muuttuvaa, mitä olennaista kuhunkin konseptiin kuuluu sekä tuntee toimintatavat ja termit.	Bisnes, liiketoimintasuunnitelmat	Ei mitään (uuden ohjelmiston luontiin liittyen).
Käyttäjä-ekspertti	Kokenut käyttäjä. Tietää, kuinka uuden ohjelmiston tulee toimia käyttäjän näkökulmasta. Osaa kertoa, mitkä sovelluksen toiminnoista ovat kriittisiä, mitä voidaan jättää pois, mitä oikoteitä voidaan käyttää ja mitä vaihtoehtoja eri toiminnoille voidaan luoda. EI ole IT-ammattilainen, joka on jutellut käyttäjien kanssa, ei johtaja, joka on tilannut ohjelmiston, ei ostopäällikkö. Kohdekäyttäjäryhmästä voi löytyä useita käyttäjäeksperttejä.	Käyttäjän omaan työhön liittyvät taidot. Kyky artikuloida toiminteita, tehtäviä ja prioriteetteja.	Ei mitään (uuden ohjelmiston luontiin liittyen).
Tekninen avustaja	Tietää, kuinka johtaa ryhmäkeskusteluja, oli sitten kyseessä määrittely- tai suunnittelupalaveri tai katselmus.	Ryhmätyötaidot.	JAD (Joint Application Design) tai CRC-korttitekniikka, katselmuksen valmistelut
Vaatimus-analyytikko	Tietää bisneksestä sen verran, että kykenee tutkimaan asiakkaan vaatimuksia sekä lisäksi tekniikkaa sen verran, että kykenee etsimään vaihtoehtoisia vaatimuksia, jos toteutus tai ratkaisu tuntuu liian hankalalta. Tutkii ja kirjaa ylös uuden järjestelmän rajapinnat, nykyiset ja suunnitellut.	Kommunikointitaidot, huolellisuus.	”maalaisjärki”, domain object modeling (etsii substantiivit ja verbit, karsii niitä, tarkistaa liiketoiminnaliset aspektit, soveltaa kardinaliteettisääntöjä)
Arkkitehti	Osaa suunnitella järjestelmäkokonaisuuksia. Arkkitehti vastaa järjestelmän komponenttien osittelusta, rajapinnoista, suorituskykyvaatimuksista sekä sovelluksen selkeäpiirteisyydestä ja toimivuudesta. Toimii uudelleenkäytettävyyden puolestapuhujana. Työskentelee projektipäällikön kanssa projektisuunnitelman suunnittelussa ja priorisoinnissa. Tekee korkean tason malleja ja suunnitelmia.	Kyky arvioida ja mallintaa koko järestelmä. Kyky tehdä laajoja sekä yksityiskohtaisia teknisiä ratkaisuja ja päätöksiä.	Mallinnustekniikat, systeemianalyysi.
Projekti-päällikkö	Osaa kerätä ja integroida projektin eri osapuolilta saatavaa informaatiota ja koostaa niiden pohjalta projektisuunnitelman. Kykenee estämään ”toiminnallisuustulvan” ja muut projektin läpivientiä uhkaavat tekijät. Vastaa prosessista muista roolilähteistä saadun tiedon perusteella.	Motivaatio, seuranta ja tarkkailu, kommunikointi, suunnittelu.	Projektin ennakointi, johtamistekniikat, tiimien rakentaminen

Vastaava suunnittelija	Osaa luoda komponenttikehyksiä (frameworks), erottaa heikon toteutusmallin vahvasta, kykenee valvomaan ja valmentamaan muita ohjelmoijia masentamatta heitä. Suunnittelee komponentteja ja alijärjestelmiä soveltaen mallinnustekniikkaa vaatimusmäärittelyihin. Yleensä (muttei välttämättä) tiimin paras ohjelmoija.	Framework-suunnittelu, luokkasuunnittelu, kommunikointitaidot, ohjelmointi.	Erilaiset mallinnus- ja ohjelmointitekniikat
Ohjelmoija	Kykenee suunnittelemaan ja toteuttamaan luokan tai luokkakirjaston annetun määrittelyn ja luokkakaavion (tai muun kaavion) perusteella. Voi kyetä suunnittelemaan komponenttikehyksiä. Voi olla ekspertti, jolla ei juuri ole kiinnostusta kommunikointiin ja opetukseen.	Luokkasuunnittelu, ohjelmointi, framework-suunnittelu.	Erilaiset mallinnus- ja ohjelmointitekniikat
Design mentor	Vastaavat taidot kuin vastaavalla suunnittelijalla, mutta vahvemmat kommunikointitaidot. Vastuualueena tiimien mallinnus- ja ohjelmointitason kohottaminen ja nuorempien ohjelmoijien valmentaminen.	Kyky vertailla ja selventää eri suunnittelu- ja mallinnusvaihtoehtoja, Framework-suunnittelu, luokkasuunnittelu, kommunikointitaidot, ohjelmointi.	Valmennustaidot, ryhmätyötaidot, erilaiset mallinnus- ja ohjelmointitekniikat
Dokumentoija	Luo ”ulkopuolisen” dokumentaation kuten käyttöliittymä- ja testausspesifikaatiot sekä luonnoksen sovelluksen käyttöoppaasta.	Dokumentointitaidot, OO-käsitteiden tuntemus	
Testaaja	Osaa luoda ja suorittaa testitapauksia joko annetun vaatimusmäärittelyn tai käyttöliittymäspesifikaation pohjalta. Luo syklisiä järjestelmätestaussarjoja (regressiotestit).	Testitapausten ja -sarjojen luonti.	Erilaiset testaustekniikat.
Käyttöliittymäsuunnittelija	Osaa luoda helppokäyttöisiä käyttöliittymiä. Tuntee tai kykenee oppimaan käyttöliittymästandardit. Ajaa yksinkertaisuuden ja yhdenmukaisuuden periaatteita suunnittelutyössään.	Kykyenee tunnistamaan käyttäjän työskentelytavat ja tarpeet sekä arvioimaan ja testaamaan käyttöliittymää.	Käyttäjälähtöinen suunnittelu, käyttöliittymämallinnus, video- ja kyselypohjainen testaus