TAMPEREEN TEKNILLINEN YLIOPISTO
TAMPERE UNIVERSITY OF TECHNOLOGY

ILKKA RANTANEN

OHJELMISTOKEHITYSPROSESSIN MÄÄRITTÄMINEN KYLMÄ-ALAN YRITYKSELLE

Diplomityö

Tarkastaja: professori Hannu Koivisto
Tarkastaja ja aihe hyväksytty
Teknisten tieteiden tiedekuntaneuvoston kokouksessa 3. huhtikuuta 2013

TIIVISTELMÄ

TAMPEREEN TEKNILLINEN YLIOPISTO
Automaatiotekniikan koulutusohjelma
RANTANEN, ILKKA: Ohjelmistokehitysprosessin määrittely kylmäalan yritykselle
Diplomityö, 56 sivua
Kesäkuu 2013
Pääaine: Oppivat järjestelmät
Tarkastaja: professori Hannu Koivisto
Avainsanat: Kylmäala, ohjelmistokehitys, ohjelmistokehitysmalli, ketterä ohjelmistokehitys, Scrum

Huurre Group Oy:n Huurre HOT on palvelu kylmäjärjestelmien etävalvontaan, etähallintaan sekä energiaoptimointiin. Palvelun käyttäjät voivat muun muassa nähdä reaaliaikaisen tilanteen kylmäjärjestelmistään sekä ohjata niitä Huurre HOT:n palvelusivujen kautta internetselaimella. Huurre HOT:n kehitys aloitettiin vuonna 2007 ja se lanseerattiin vuonna 2008. Vuonna 2009 se eriytettiin omaksi liiketoimintayksiköksi, josta lähtien palvelun asiakasmäärä on kasvanut tasaisesti.

Huurre HOT on rakennettu Tridium:n Niagara^{AX}-ohjelmistoalustan päälle. Rakentamisen on toteuttanut pääasiassa Huurre HOT:n kehityspartnerit, joiden kautta myös jatkokehitystä on toteutettu. Liiketoiminnan kasvaessa Huurre HOT:n henkilöstön määrä sekä tekninen osaaminen ovat lisääntyneet huomattavasti, jonka seurauksena myös kehitysstrategiaa on täsmennetty. Täsmennettyä kehitysstrategiaa varten tarvitaan hallittu ohjelmistokehitysprosessi.

Tämän työn tavoite on määritellä Huurre HOT:lle prosessi ohjelmistokehitykseen. Työssä tutustutaan ohjelmiston elinkaarimalliin sekä ohjelmistokehityksessä yleisesti käytettyihin vesiputousmalliin, spiraalimalliin, iteratiiviseen ja inkrementaaliseen malliin sekä ketterän kehityksen Scrum:iin. Myös muutamia ketterän ohjelmistokehityksen käytäntöjä esitellään paremman yleiskuvan saamiseksi ohjelmistokehityksestä.

Huurre HOT:n ohjelmistokehityksen nykytilanne analysoidaan ja sen kehitystarpeet tuodaan esiin. Analyysin jälkeen esiteltyjä ohjelmistokehitysmallien sekä ketterien käytäntöjen sopivuutta Huurre HOT:n tilanteeseen arvioidaan.

Työssä huomataan, että suurimmat ongelmat ohjelmistokehityksessä ovat olleet asiakkaiden vaatimusmäärittelyjen muuttumisessa, ohjelmistoprojektien seurannassa ja niiden venymisessä. Havaintojen pohjalta uudeksi ohjelmistokehitysmalliksi valitaan ketterän kehityksen Scrum, jossa tullaan hyödyntämään ketteristä käytännöistä erityisesti käyttäjätarinoita ja päiväpalavereita.

ABSTRACT

TAMPERE UNIVERSITY OF TECHNOLOGY
Master's Degree Programme in Automation Technology
RANTANEN, ILKKA: Defining a Software Development Process for a Refrigeration Company
Master of Science Thesis, 56 pages
June 2013
Major: Soft Computing
Examiner: Professor Hannu Koivisto
Keywords: Refrigeration sector, software development, software development life-cycle, agile software development, Scrum

Huurre HOT of Huurre Group Oy is a service for remote monitoring and management of refrigeration systems. Users of the service can access their own refrigeration systems via Huurre HOT website and monitor or control the systems with a web browser. Development of the service started 2007 and it was published 2008. Huurre HOT became a separate business unit 2009 and number of customers has increased steadily since.

Huurre HOT system has been built on Tridium's Niagara^{AX} software framework by Huurre HOT's development partners. All the software development for the system has been outsourced to the partners as well due to Huurre HOT's employees lack of knowledge of the system. While the business has grown the knowledge of the Huurre HOT's personnel has increased. It has resulted in changes in development strategy. The new strategy requires a controlled software development process for starting in-house software development.

The main goal for this thesis is to define a software development process for Huurre HOT. The thesis examines software development life-cycle, commonly used software development processes and a few agile software development techniques.

The current state of Huurre HOT's software development process is analysed, development needs of the process are identified and the examined software development processes and agile development techniques are evaluated for suitability.

The result of the analysis indicates that the biggest issues have been in customers changing definitions of software's requirements, monitoring the software project and keeping the project in schedule. Based on the results Scrum is selected as the best software development model for Huurre HOT in which especially use cases and daily meetings will be used.

ALKUSANAT

Haluan kiittää diplomityön ohjaajaa ja tarkastajaa Hannu Koivistoa työn aiheen tarkentamisesta, rakentavasta palautteesta sekä kannustamisesta työn tekemiseen. Haluan kiittää myös Huurre Group Oy:n Jukka Hellmania työn aktiivisesta ohjauksesta, rakentavasta palautteesta, uskosta työn edistymiseen sekä ajasta, jota olen saanut kiitettävästi työn kirjoittamiseen.

Lisäksi haluan kiittää opiskelukavereitani, joiden kanssa käydyt lukuisat keskustelut ovat selkeyttäneet ymmärrystä työni aihepiiristä ja auttaneet rentoutumaan kirjoittamisen ohella. Haluan kiittää myös tyttöystävääni kannustamisesta ja onnistuneesta motivoinnista työn tekemiseksi.

SISÄLLYS

TERMIT JA NIIDEN MÄÄRITELMÄT

AX Supervisor	Niagara^{AX}-ohjelmisto palvelimelle, jolla voidaan hallita usean JACE:n verkkoja.
BMS	Building Management System on tietokonepohjainen järjestelmä rakennuksen mekaanisten ja sähköisten laitteiden hallintaa ja valvontaa varten.
COP	COP (englanniksi Coefficient of Performance) on kylmäkerroin, joka kuvaa järjestelmän kykyä muuttaa sähköenergiaa kylmäenergiaksi.
HACCP	HACCP (englanniksi Hazard Analysis and Critical Control Points) on järjestelmä elintarvikehuoneiston omavalvontaan, jolla pyritään estämään mahdollisesti terveysvaaraa aiheuttavan tuotteen etenemisen kuluttajalle.
Huurre HOT	Huurre Group Oy:n palvelu kylmäjärjestelmien etävalvontaan ja etäkäyttöön.
JACE	JACE on teollisuustietokone Niagara^{AX}:lle, joka on suunniteltu useiden automaatioverkkojen integroimiseksi yhdenmukaiseksi järjestelmäksi.
Kylmälaitepositio	Kylmäkaluste, kalusteen osa tai kylmähuone, jossa säilytetään yhden tuoteryhmän elintarvikkeita
Niagara^{AX}	Tridium Inc:n modulaarinen ohjelmistokehitysalusta.
Omavalvonta	Elintarvikealan toimijan oma järjestelmä, jolla toimija pyrkii varmistamaan, että elintarvike, alkutuotantopaikka ja elintarvikehuoneisto sekä siellä harjoitettava toiminta täyttävät niille elintarvikemääräyksissä asetetut vaatimukset.
QNX	UNIX-pohjainen käyttöjärjestelmä, jonka päällä Niagara^{AX} toimii JACE:ssa.
RFID	RFID on radiotaajuinen etätunnistusmenetelmä tiedon etälukuun ja etätallentamiseen.
SVN	Lyhenne versionhallintajärjestelmästä Subversion, jonka tarkoitus on mahdollistaa esim. ohjelmistojen lähdekoodin muokkaamisen hajautetusti niin, että kaikkien muokkaajien työkopiot pysyvät ajan tasalla.
UPS	Uninterruptible Power Supply on järjestelmä tai laite, jonka tehtävä on taata tasainen virransyöttö lyhyissä katkoksissa ja syöttöjännitteen epätasaisuuksissa.

1 JOHDANTO

Huurre HOT on palvelu kylmäjärjestelmien etävalvontaan, etähallintaan ja energiaoptimointiin. Se toimii omana liiketoimintayksikkönä Huurre Group Oy:ssä, joka on elintarvikkeiden jakeluun ja säilytykseen ratkaisuja tarjoava kylmätekninen palvelutalo. Huurre Groupin toiminta on keskittynyt Pohjois- ja Itä-Eurooppaan ja sen liikevaihto vuonna 2011 oli 222 milj. €.

Huurre HOT -palvelun kehitys aloitettiin vuonna 2007 ja 2008 se aloitti toiminnan. Se eriytettiin omaksi liiketoimintayksiköksi vuonna 2009, josta lähtien työntekijöiden ja asiakkaiden määrä on kasvanut tasaisesti. Huurre HOT:sta on tullut tärkeä osa Huurre Group Oy:n strategiaa ja liiketoimintayksikkö raportoi konsernin johdolle.

Huurre HOT -palvelua halutaan kehittää jatkuvasti, koska Huurre HOT on tärkeä osa koko Groupin strategiaa. Suurin osa palvelun kehittämisestä liittyy palvelun teknisen alustan kehittämiseen, jolla mahdollistetaan uusien lisäpalveluiden tarjoamisen asiakkaille. Teknisen kehityksen toteutus on ollut täysin ulkoistettu Huurre HOT:n kehityspartnereille palvelun alkuvuosien aikana, mutta liiketoiminnan varmistamiseksi on päätetty tutkia mahdollisuutta aloittaa myös oma ohjelmistokehitys. Henkilökunnan osaamisen lisääntyminen, tarve kehityksen nopeuttamiselle, halu riippuvuuden vähentämiseksi kehityspartnereista sekä tavoite nykyisen kehitysprosessin parantamiselle ovat toimineet tärkeimpinä syinä tämän työn aloittamiselle.

Tässä työssä tutkitaan, miten ohjelmistokehitysprojektit voidaan viedä onnistuneesti läpi Huurre HOT:ssa. Työn tarkoitus on kehittää Huurre HOT:n tilanteeseen sopiva prosessikuvaus ohjelmistokehityksestä ja lisätä Huurre HOT:n yleistä tietoa ohjelmistokehityksestä. Työn 2. luvussa tutustutaan kylmäalaan, joka on Huurre HOT:n toimintaympäristö. Luvussa 3 määritellään lähtökohdat sekä tavoitteet tälle työlle. Luvussa 4 tutustutaan ohjelmiston elinkaarimalliin ja tuodaan esiin ohjelmistotuotannossa yleisesti käytetyt ohjelmistokehitysmallit vesiputousmalli, spiraalimalli, iteratiivinen ja inkrementaalinen malli sekä ketterän kehityksen Scrum. Luvussa myös selvitetään, minkälaisia ketterän kehityksen käytäntöjä on olemassa. Luvussa 5 analysoidaan ohjelmistokehitysmallien sekä ohjelmistokehityskäytäntöjen sopivuutta Huurre HOT:lle, jonka perusteella määritetään prosessi ohjelmistokehitykseen luvussa 6.

2 TAUSTAA KYLMÄALASTA

Tässä luvussa esitellään kylmäalaa yleisellä tasolla. Luvussa tuodaan esiin tuotteita ja palveluita, joita kylmäala tarjoaa. Kylmäalan tarjoamista etäpalveluista etävalvonta, etähallinta ja energiapalvelut käydään läpi tuotteita yksityiskohtaisemmin. Tarjolla olevia etäpalveluita sekä yksi omavalvontajärjestelmä esitellään markkinatilanteen yleiskuvan muodostamiseksi kohdeyrityksen Huurre Group Oy:n Huurre HOT -palvelun näkökulmasta.

2.1 Tuotteet ja palvelut

Kylmäalan tarjoamilla tuotteilla ja palveluilla on merkittävä rooli tämän päivän hyvinvointiyhteiskunnassa. Kylmäala on helppo yhdistää esimerkiksi elintarvikkeiden turvallisuuteen ja kesäisin miellyttäviin ilmastoituihin sisätiloihin ulkolämpötilan ollessa tukahduttavan kuuma. Perinteisesti kylmäala tarjoaa ratkaisuja asiakkaille, joilla on ongelmana liiallinen lämpö laitteessa, tilassa tai prosessissa.

2.1.1 Kylmäketju

Kylmäketjulla tarkoitetaan tuotteen lämpötilan ylläpitämistä valmistuspaikasta myyntipaikan kautta kuluttajalle. Kuva 2.1 esittää tyypillistä kylmäketjua. Tuote kulkee usein monen eri varaston läpi ja se saatetaan pakata ja purkaa useaan otteeseen ennen päätymistä lopulliseen myyntipaikkaan. Kylmäketjun hallinta onkin tärkeä tekijä tuotteen laadun takaamisessa elintarvike-, lääke- ja kemianteollisuudessa.

Kuva 2.1. *Tyypillinen kylmäketju.*

Kylmäketjun alkupäähän kylmäalan yritykset tarjoavat elementtirakenteisia tuotantotiloja lämpötilakriittisiin tuotantoympäristöihin, kuten lihan jatkokäsittelyyn teurastuksen jälkeen. Lääkealan yrityksille on tarjolla lämpötilan osalta tarkasti kontrolloituja puhdastilaratkaisuja. Skaala voi toimituksissa vaihdella yksittäisestä kylmähuoneesta aina valmiiksi rakennettuihin tuhansien kuutiometrien kylmävarastoihin.

Tuotantoympäristön kylmästä ja tiiviistä lähettämöstä tuotteet toimitetaan kylmä- tai pakkasvarastoihin. Tuotteen matkan aikana tuotteen halutaan pysyvän halutuissa lämpötilarajoissa. Kylmäkuljetustuotteiden turvallista kuljettamista varten tehdään mahdolliseksi käyttäen termolaatikoita, kylmäkontteja ja kylmäkuljetuskalustoa. Kuljetetut tuotteet puretaan myymälään saapumisen jälkeen myymälän kylmävarastoihin.

Myös tuotteen myyminen tulee hoitaa valvotusta lämpötilasta. Myymiseen yrityksillä on tarjolla erilaisia kylmäkalusteita. Kylmäkalusteiden tehtävä on pitää tuotteet oikeassa lämpötilassa mahdollisimman energiatehokkaasti ja taata, että asiakkaat pääsevät helposti tuotteisiin käsiksi.

Kylmäketjun loppupää jää asiakkaan vastuulle. Tuotteen oikea ja nopea kuljetus myymälästä esimerkiksi omaan jääkaappiin on tyypillisesti kylmäketjun heikoin lenkki. Riskien pienentämiseksi tuotteet pyritään pakkaamaan tuotteen säilymisen kannalta hyvin ja yritykset tarjoavat kylmälaukkuja ja -pusseja tuotteiden suojaamiseksi kuljetuksen ajalle.

2.1.2 Kylmäketjun valvonta

Onnistuneen kylmäketjun varmistaminen vaatii jatkuvaa valvontaa jokaisessa kylmäketjun vaiheessa. Kylmäketjun monitorointi on ainoa tapa varmistaa sen oleva validi. Mo-

nien tuotteiden valvonta on määrätty laissa: Esimerkiksi elintarvikkeiden kylmäketjun valvonta on määritelty Suomen elintarvikelaissa.

Lämpötilavalvonta on kylmäketjun valvonnan perusta. Lämpötilavalvonnassa tuotteen lähellä, tuotteessa kiinni tai tuotteen sisään työnnettynä olevan lämpötila-anturin lämpötila tallennetaan säännöllisin välein muistiin. Tallennetuista lämpötiloista nähdään, onko tuotteen lämpötila tai kalusteen olosuhdelämpötila pysynyt halutulla alueella ja onko kylmäketju rikkoutumaton.

Valvontajärjestelmät valvovat tuotteiden säilytystilojen lämpötiloja reaaliajassa: Kun säilytystilan lämpötila poikkeaa tavoitearvosta riittävästi, lämpötilahälytys aktivoidaan. Normaalisti hälytys välitetään kärkitietona vartiointiliikkeeseen, josta soitetaan huoltoyhtiölle. Uusimmat järjestelmät voivat lähettää hälytyksen sähköpostina tai tekstiviestinä halutun henkilön puhelimeen. Näin tuotteet voidaan pelastaa lämpenemiseltä tekemällä ajoissa korjaavia toimenpiteitä.

Elintarvikelain pykälässä 20 määrätään omavalvonnasta: “Elintarvikealan toimijan on laadittava kirjallinen suunnitelma omavalvonnasta (omavalvontasuunnitelma), noudatettava sitä ja pidettävä sen toteuttamisesta kirjaa.“ [1.] Tuotantotilan, kuljetuksen, varaston tai myymälän vastuuhenkilön on pidettävä huoli siitä, että lämpötila tulee kirjatuksi muistiin omavalvontasuunnitelman mukaan.

Omavalvonta voi olla yksinkertaisimmillaan lämpötilamittareiden manuaalista lukemista ja kirjaamista vihkoon. Se on kuitenkin erittäin työläs ja helposti unohtuva rutiini, jonka takia lämpötilojen tallennus hoidetaan usein automaattisesti.

2.1.3 Automaattiset omavalvontajärjestelmät

Useat automaatiojärjestelmien toimittajat tarjoavat omavalvontaan työkaluja. Kylmäjärjestelmän laitteet asennetaan normaalisti automaatioverkkoon. Useat kylmäjärjestelmät sisältävät päätelaitteen, jolla pystytään lukemaan kaikki kylmäjärjestelmän mittaamat lämpötilat. Kuvassa 2.1 on saksalaisen Eckelmann:n kylmäjärjestelmän päätelaite CI 3000, jonka kautta voidaan valvoa ja käyttää kaikkia automaatioverkossa olevia laitteita.

Kuva 2.2. *Eckelmann:n kylmäjärjestelmän päätelaite CI 3000. [2.]*

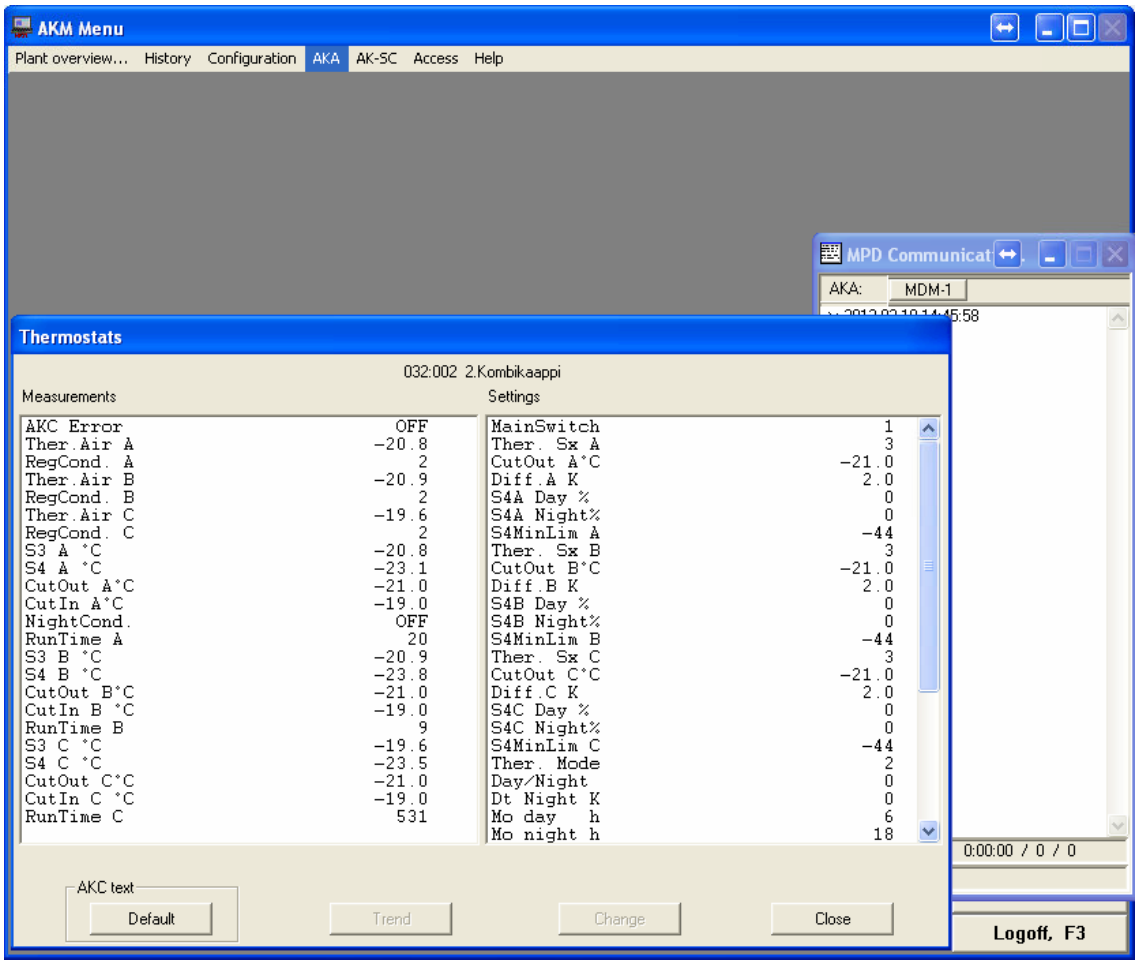
Päätelaitteen tehtävä ei ole ainoastaan lukea ja tallentaa lämpötiloja. Sen avulla kylmäjärjestelmän säätimet saadaan konfiguroitua keskitetysti. Päätelaitteella pystytään asettamaan mm. säätimien asetusarvot, ohjauksen tarkkuus, hälytysrajat sekä sulatusten aikataulu. Laitteelta on nopea katsoa kylmäjärjestelmän aktiiviset ja vanhat hälytykset ja normaalisti päätelaite välittää hälytyksen kärkitietona kiinteistön automaatiokeskukseen. Uusimmat päätelaitteet pystyvät välittämään hälytykset eteenpäin myös tekstiviestinä tai sähköpostilla halutuille vastaanottajille. Päätelaite on kylmäautomaatioverkon aivot ja erittäin kriittinen hälytysten ja yleisesti koko kylmäautomaatioverkon toimivuuden kannalta.

Suuressa osassa kylmäjärjestelmän päätelaitteissa on rajallinen määrä muistia lämpötilojen tallentamiseen, koska Suomessa olevat kylmäjärjestelmät ovat hyvin vanhoja. Viranomaisten vaatimaa omavalvontaa varten tallennetut lämpötilat on tulostettu esim. viikoittain päätelaitteesta ja tulosteet arkistoitu kansioihin. Kun päätelaiteeseen tulee toimintavika, siitä ei välttämättä lähde hälytystä eteenpäin. Silloin ei verkkoa valvo kukaan, eivätkä lämpötilat tallennu tarvittavalla tavalla.

2.1.4 Perinteiset etäpalvelut

Kylmäjärjestelmien huoltoyhtiöt ovat tarjonneet etäpalveluita jo 1990-luvulta lähtien. Etäpalvelut ovat perustuneet mahdollisuuteen muodostaa etäyhteys kylmäjärjestelmien päätelaitteisiin. Päätelaitteet ovat olleet tavallisesti tavoitettavissa modeemiyhteyksillä ja uudempiin laitteisiin saadaan yhteys internetin yli. Päätelaitteiden etäkäyttö vaatii aina lisensoidun ohjelmiston asentamisen käyttäjän PC:lle, eli monen eri kylmäjärjestelmän hallinta vaatii monta eri ohjelmistoa. Kuvassa 2.3 on tanskalaisen Danfossin

kylmäjärjestelmän ADAP-KOOL valvontaohjelmisto AKM, joka on hyvin yleisesti käytössä kylmäjärjestelmien huoltoyhtiöissä.



Kuva 2.3. *Danfossin valvontaohjelmisto AKM heidän kylmäjärjestelmään ADAP-KOOL.*

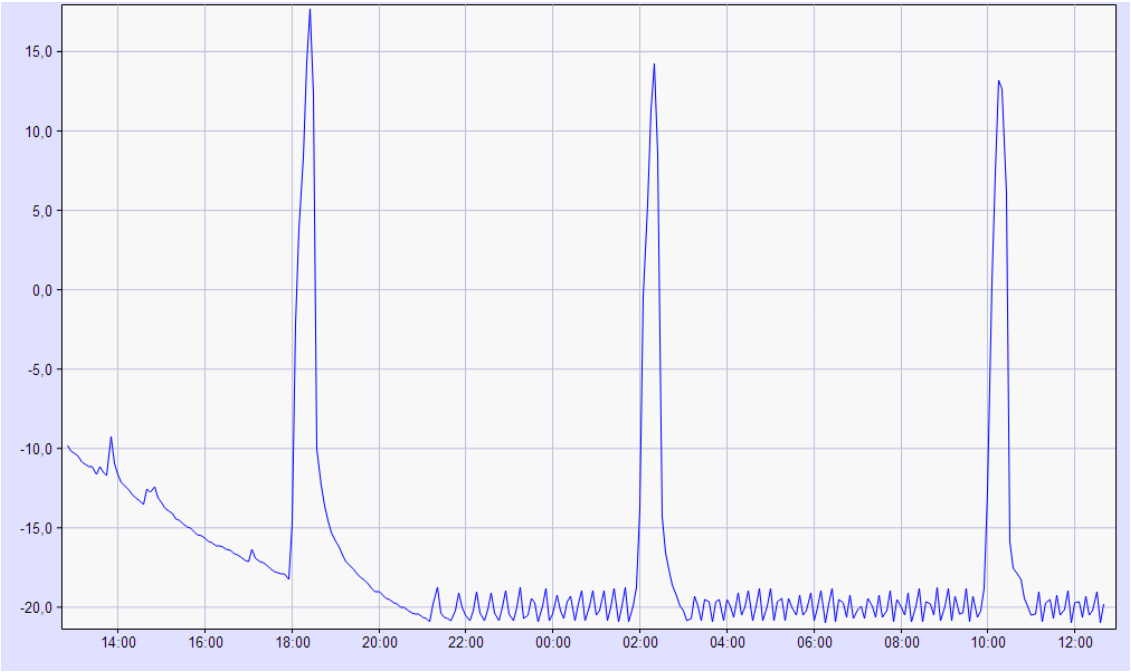
Kylmäjärjestelmien valmistajien ohjelmistot ovat yleisesti ottaen käytettävyydeltään erittäin huonoja. Etäyhteyden muodostaminen on hidasta ja kallista johtuen modeemien heikoista nopeuksista sekä niiden vaatimasta puhelinlinjoista. Lisäksi etäyhteyteen tarvittavat ohjelmistot ovat kalliita ja vaativat usein USB-lisenssiavaimen toimiakseen. Tavallista on, että käyttäjältä puuttuu tarvittava ohjelmisto tai USB-lisenssiavain työn suorittamiseen etänä. Lisäksi käyttäjän on vaikea hallita useita ohjelmistoja. Kun etäyhteyttä ei välttämättä osata käyttää tai yhteys modeemiin ei toimi, aiheutuu hälytysten tarkistamisesta paljon kustannuksia. Suurin osa hälytyksistä ei ole kriittisiä, vaan ne voidaan hoitaa etänä tai niiden korjaus voidaan siirtää vaikka seuraavalle päivälle.

2.1.5 Uudet etäpalvelut

Rakennusautomaation etäpalveluiden yleistymisen myötä myös kylmäalalle on tullut tarjolle etäpalveluita 2000-luvun loppupuolella. Kylmäalalla etäpalveluilla pyritään poistamaan omavalvonnan työtaakka tallentamalla ja arkistoimalla lämpötilat palvelun-

tarjoajan palvelimille. Lämpötilojen katselu ja viranomaisten vaatimat lämpötilaraportit ovat helposti saatavilla palvelimilta yksinkertaisesti käyttäen verkkoselainta. Tämä poistaa monimutkaisten ohjelmistojen opettelun ja mahdollistaa tallennetun datan paremman hyödyntämisen. Tallennettuun dataan pohjautuen pystytään tarjoamaan asiakkaille myös lisäpalveluita.

Tärkein etävalvontapalvelu kylmäalalla on lämpötilavalvontapalvelu. Lämpötilavalvontapalvelussa palveluntarjoaja huolehtii lämpötilojen tallentamisesta. Lämpötilat tallennetaan sovitun ajan välein tietokantaan ja niitä säilytetään sopimuksessa määritetyn aikajakson ajan. Kyseinen palvelu poistaa omavalvonnan tarpeen asiakkaalta ja säästää asiakkaan aikaa tärkeämpiä työtehtäviä varten. Lämpötilatiedot ovat asiakkaan saatavilla yleensä yksinkertaisesti verkkoselaimen kautta. Verkkosivuilla data on saatavilla taulukoina tai visuaalisesti käyränä, kuten kuvassa 2.4. Kuvaajasta saa nopealla vilkaisulla selville, onko lämpötila käyttäytynyt halutulla tavalla. Esimerkiksi kuvan 2.4 käyrän käyttäytymisestä selviää, että kaluste ei ole toiminut normaalisti koko päivää. Myös laitteessa suoritetut sulatukset näkyvät selvästi käyrässä korkeina lämpötilapiikkeinä.



Kuva 2.4. *Erään kylmälaiteposition lämpötiladataa.*

Uudenaikainen lämpötilojen esitystapa on kuvan 2.5 tapainen taulukko, joka korostaa väreillä liian lämmintä tai kylmää lämpötilaa. Kyseinen raportti on käytettävissä osana HACCP-raporttia. HACCP tulee sanoista hazard analysis critical control point ja sillä tarkoitetaan vaarojen analysointia ja kriittisten pisteiden hallintaa tuotteiden valvonnassa. HACCP-raportin pitää vastata seuraaviin kysymyksiin:

- mitä valvotaan
- millä tavoin
- miten usein
- mitkä ovat raja-arvot
- mitä tehdään, jos raja-arvo ylittyy eli ohjeet korjaaville toimille
- miten havainnot kirjataan ja
- missä kirjattuja havaintotietoja säilytetään ja kuinka kauan. [3.]

HACCP Report

defrost | -5.0 | -4.0 | -3.0 | -2.0 | SP | 2.0 | 3.0 | 4.0 | 5.0 Algorithm: Average

Date	Time	SP	Pos. 21 Leikkeet 1 Leikkeleet A
13.05.2013	01:00	3,0	3,7
13.05.2013	02:00	3,0	3,7
13.05.2013	03:00	3,0	3,9
13.05.2013	04:00	3,0	3,9
13.05.2013	05:00	3,0	3,5
13.05.2013	06:00	3,0	4,0
13.05.2013	07:00	3,0	3,6
13.05.2013	08:00	3,0	6,2
13.05.2013	09:00	3,0	3,9
13.05.2013	10:00	3,0	4,0
13.05.2013	11:00	3,0	4,0
13.05.2013	12:00	3,0	3,8
13.05.2013	13:00	3,0	3,7
13.05.2013	14:00	3,0	3,9
13.05.2013	15:00	3,0	3,8
13.05.2013	16:00	3,0	4,0
13.05.2013	17:00	3,0	3,8
13.05.2013	18:00	3,0	6,0
13.05.2013	19:00	3,0	3,5
13.05.2013	20:00	3,0	4,0
13.05.2013	21:00	3,0	3,8
13.05.2013	22:00	3,0	4,0
13.05.2013	23:00	3,0	3,8

Kuva 2.5. *Osa erään position uudenaikaisesta HACCP-raportista.*

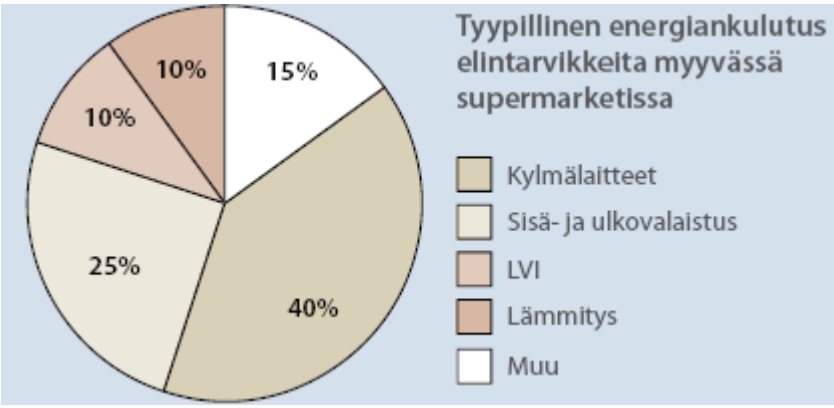
Lämpötilanvalvontaan on tarjolla lisäpalveluna hälytyspalvelu, jossa kylmäjärjestelmän hälytykset välitetään verkkosivuille, sähköposteihin ja matkapuhelimiin. Kylmäjärjestelmästä lähtee usein paljon erilaisia hälytyksiä, joista suuri osa voidaan jättää huomiotta tai hoitaa myöhemmin. Palvelu hälytysten esianalysointiin etänä tehostaa

huoltopalvelun toimintaa ja säästää suuria summia rahaa asiakkailta, kun esimerkiksi turhia korjauskäyntejä ei tarvitse tehdä keskellä yötä.

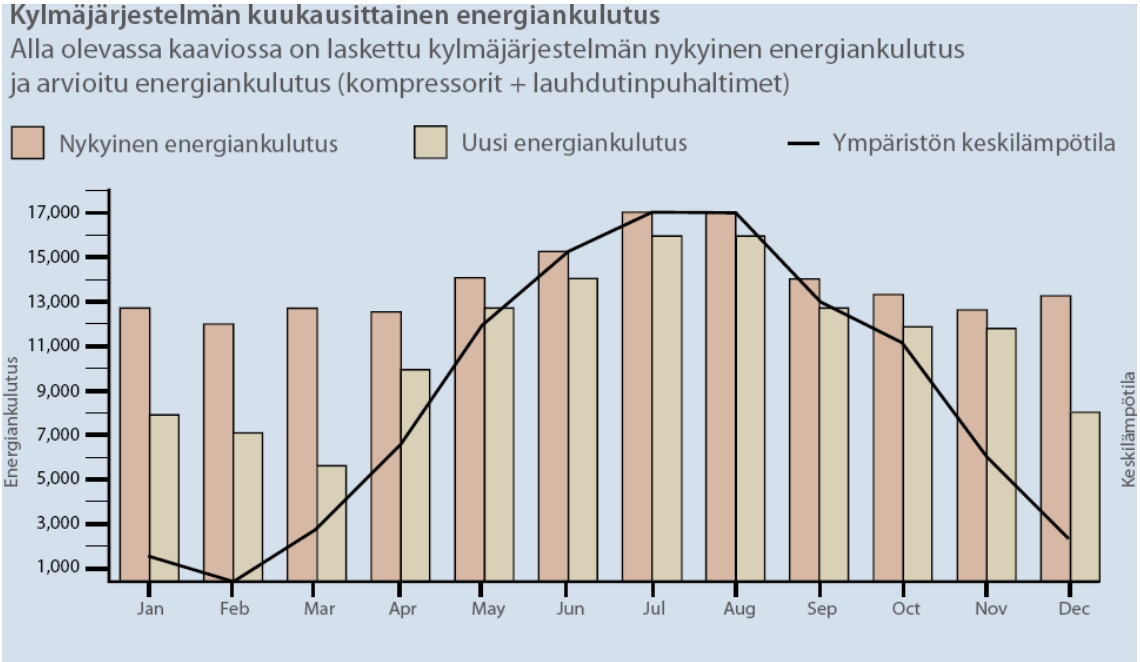
Tallennetuista lämpötilatiedoista on hyötyä kylmäjärjestelmän huollolle. Lämpötilojen pohjalta on mahdollista päätellä viat etänä ilman kohteeseen menemistä. Näin korjauksen tekevää henkilöä voidaan informoida todennäköisistä vioista etukäteen, tai parhaassa tapauksessa ongelma voidaan korjata täysin etätyönä. Lisäksi lämpötilojen systemaattinen analysointi voi paljastaa mahdollisia vikoja jo ennen niistä aiheutuvaa haittaa. Esimerkiksi lämpötilan nouseva trendi voi kertoa, että kaluste alkaa jäätyä.

2.1.6 Energiapalvelut

Kylmäalan asiakkaat ovat kiinnostuneita energiapalveluista, koska kylmäjärjestelmät kuluttavat paljon energiaa. Kylmäjärjestelmä voi kuluttaa koko kiinteistön energiasta 40-60 %, joten kylmäjärjestelmän käyttö voi aiheuttaa miljoonien eurojen kustannukset vuodessa. Kuvassa 2.6 on kuvattu tyypillisen elintarvikkeita myyvän supermarketin energiankulutuksen jakautumista Danfossin mukaan ja kuvat 2.7 esittää erään keskikokoisen elintarvikemyymälän kuukausikohtaista energiankulutusta. [4.]

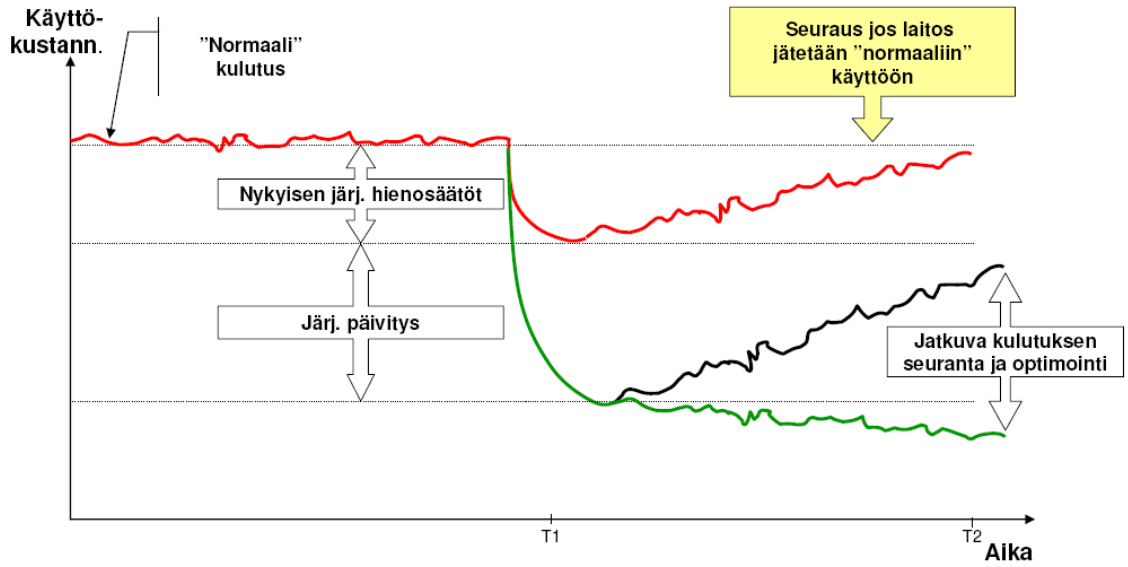


Kuva 2.6. *Supermarketin energiakustannukset eriteltynä. [4.]*



Kuva 2.7. *Erään kaupan kylmäjärjestelmän energiankulutusten vertailua kuukausikohtaisesti. [4.]*

Kylmäjärjestelmä on monimutkainen kokonaisuus, jossa on paljon säädettävää. Kun kylmäprosessin dataa sekä energiatietoa tallennetaan muistiin, voidaan niiden perusteella hienosäätää kylmäjärjestelmä mahdollisimman energiatehokkaaksi. Näillä säädöillä voidaan energiatehokkuutta parantaa yli 15 % riippuen kylmälaitoksen nykyisistä asetuksista. [4.] Jotta kylmälaitoksen energiatehokkuus saadaan pidettyä yllä, on energian kulutusta valvottava. Ilman valvontaa energiankulutus lähtee helposti nousuun kuvan 2.8 osoittamalla tavalla.



Kuva 2.8. *Energiankulutuksen seurannalla pidetään kulutus alhaalla. [4.]*

Kylmäalalla esimerkiksi suuret myymäläketjut haluavat vertailla myymälöidensä energiatehokkuutta. Maailmassa ei ole kahta samanlaista kylmälaitosta, joten niiden vertaileminen tuottaa ongelmia. COP:n avulla on hyvin helppo vertailla erilaisten kylmälaitosten tehokkuutta. COP muodostuu sanoista coefficient of performance ja se kuvaa, kuinka tehokkaasti kylmäjärjestelmä muuttaa sähköenergiaa lämpöenergiaksi. [5.] Etenkin suurten kauppaketjujen johtajille COP antaa työkalun tehokkuuden vertailuun. Kaava 2.1 määrittää, miten COP saadaan lasketuksi.

$$\mathrm{COP} = \frac{\mathrm{Q}}{\mathrm{W}} \tag{2.1}$$

Kaavassa Q on kylmäjärjestelmän poistama energia ja W kylmäjärjestelmän kompressorien käyttämä energia.

2.2 Toimijat etävalvonnassa

Kylmäalalla ei ole Pohjois-Euroopassa montaa yritystä, jotka tarjoavat etäpalveluita, vaan suurin osa etäpalveluiden tarjoajista on kylmäalan ulkopuolelta. Normaalia on, että kylmäjärjestelmiä voidaan lukea etänä, mutta varsinaisia etäpalveluita ei ole saatavilla. Tässä kappaleessa esitellään kohdeyrityksen Huurre Group etävalvontapalvelu Huurre HOT, sekä sen varteenotettavat kilpailijat Pohjois-Euroopassa.

2.2.1 Huurre HOT

Huurre HOT on kylmälaitteiden etävalvonta- ja etäohjauspalvelu. [6] Huurre Group Oy:n omistamalla Huurre HOT:lla on asiakkaina supermarketteja, vähittäiskauppoja, hotelleita, ravintoloita, suurkeittiöitä, lääkealan yrityksiä sekä kylmälaitoksia. Palvelun asiakkaat pystyvät katselemaan reaaliaikaista lämpötilatietoa omista kohteistaan kirjauduttuaan Huurre HOT -palvelun internetsivuille. Palvelun käyttäjällä on mahdollisuus tutkia historiatietoa lämpötiloista valvonnassa olevista kylmäkalusteista ja ladata erilaisia lämpötilojen laatua kuvaavia raportteja. Palvelusivuilta käyttäjä voi nähdä kaikki kylmälaiteverkon hälytykset. Esianalysoidut kylmäjärjestelmän hälytykset voidaan lähettää eteenpäin sähköpostitse ja tekstiviestein.

Kaikkien Huurre HOT:n asiakkaiden käytössä on Huurre HOT Center -palvelukeskus, jossa työskentelee usean eri tekniikan ja asiakaspalvelualan ammattitaitoisia henkilöitä. Palvelukeskus auttaa niin sisäisiä kuin ulkoisia asiakkaita kaikissa kylmäjärjestelmiin liittyvissä ongelmissa, informoi asiakasta mahdollisista kylmäjärjestelmän vioista, tilaa ja seuraa huoltotöitä, kouluttaa asiakkaita sekä raportoi asiakkaalle kylmälaitoksen tilasta ja tehdyistä toimenpiteistä. Palvelukeskuksen kautta asiakkailla on myös mahdollisuus saada käyttöönsä vuorokauden ympäri päivystävä teknikko.

Huurre HOT Center -palvelukeskuksessa energia-analyysipalvelun asiakkaan kylmäjärjestelmää optimoidaan mahdollisimman energiatehokkaaksi etsimällä sille parhaat mahdolliset parametrit energiatehokkuuden kannalta ja seuraamalla sen käyttäytymistä.

Energia-analyysipalvelun avulla saadaan energian säästöjä ilman suuria investointeja. Säästöt ovat tyypillisesti 5-20 %.

Kylmälaitosten vertailuun Huurre HOT tarjoaa erilaisia raportteja. Energiatehokkuutta voidaan vertailla perinteisillä yksiköillä, kuten $\mathrm{kWh/m^2}$ tai kWh/m^3, mutta myös erityisesti vähittäiskauppojen kylmäjärjestelmien tehokkuuden vertailuun tarkoitetulla yksiköllä $\mathrm{kWh/m}$. Metrit viimeiseen yksikköön saadaan muuntamalla kylmälaitoksessa olevat kylmäkalusteet ja kylmähuoneet metreiksi niiden koon ja höyrystimen perusteella.

2.2.2 IWMAC

Norjalainen IWMAC on web-pohjainen etävalvonta- ja etähallintajärjestelmä kaiken tyyppisille laitoksille. Heidän asiakkaat muodostuvat muun muassa supermarketeista, rakennuksista, polttoaineasemista sekä kylmälaitoksista. IWMAC:n käyttö ei siis rajoitu vain kylmäjärjestelmiin, vaan he tarjoavat järjestelmän myös kiinteistön valvontaan sekä hallintaan.

Kylmäalan asiakkaille IWMAC tarjoaa lämpötilojen ja muiden laiteparametrien tallentamisen omille palvelimilleen, HACCP-raportit, hälytyspalvelun sekä energiankulutusraportit internetsivujen kautta.

IWMAC on lähempänä teknistä alustaa kuin palvelua. IWMAC:lla ei ole Huurre HOT:n kaltaista palvelukeskusta, vaan heillä on ainoastaan ympäri vuorokauden toimiva hälytyskeskus. Asiakkaan halutessa hälytyskeskus analysoi keskukseen saapuvat hälytykset ennen kuin ne lähetetään eteenpäin asiakkaalle. Merkittävä osa saapuvista hälytyksistä voidaan luokitella ei-kriittisiksi ja vältytään turhilta hälytyksiltä ja huoltokäynneiltä. [7.]

2.2.3 e*Service

Carrier on ilmastoinnin, lämmityksen sekä kylmätekniikan toimittaja ja teknologian kehittäjä. Carrierin etäpalvelut tunnetaan nimellä e*Service ja se on käytössä useissa Euroopan maissa.

Etävalvontapalveluun kuuluu kylmäjärjestelmän etävalvonta, hälytyspalvelu hälytysten analysointiin ja turhien hälytysten suodattamiseen sekä ympäri vuorokauden päivystävä asiantunteva teknikko, joka voi tehdä tarvittaessa etäkorjauksia. Web-pohjaisena palveluna e*Service tarjoaa raportit lämpötiloille, energiankulutukselle sekä hälytyksille. Asiakkaan on mahdollista ladata itselleen kaikki tallennettu data. Energiapalvelussa analysoidaan kylmäjärjestelmän tehokkuutta, optimoidaan järjestelmän suorituskykyä sekä verrataan energiankulutusta tavoitearvoihin. [8.]

Kuvassa 2.9 on Carrierin etäpalvelukeskus, joka sijaitsee Saksan Erfurtissa. Etäpalvelukeskus ei pysty tarjoamaan tukea esimerkiksi suomeksi tai ruotsiksi, vaan asiakkaiden on hallittava saksan tai englannin kieli asiointia varten.

Kuva 2.9. *Carrier Monitoring Center. [9.]*

2.2.4 TempNet

Sensire, vanhalta nimeltään Controlmatic Oy Ltd, on vuonna 2001 perustettu suomalainen etätiedonkeräykseen erikoistunut yritys. Sensire:llä on lämpötilojen omavalvontaan tarkoitettu pilvipalvelu nimeltään TempNet ja se on keskittynyt pääasiassa kylmäalan tuotteiden kuljetuksen valvontaan. TempNet:n asiakkaita ovat mm. Suomen Kiitoautot, Uudenmaan Pikakuljetus, Havi Logistics, Scandic Trans, Abloy, ja Posten Åland.

TempNet on lämpötilojen automaattinen ja langaton omavalvontajärjestelmä pilvipalveluna. TempNet:n käyttöliittymää käytetään internetselaimella, eikä erillisiä työasemasovelluksia tai tietokoneita tarvita. TempNet:n käyttöliittymän kautta asiakas pääsee käsiksi Sensire:n palvelimilla oleviin tallennettuihin lämpötiloihin. Käyttöliittymän kautta asiakas pystyy hallitsemaan asiakastietoja ja mittauskohteita. Omavalvonnan hälytysrajat ovat aseteltavissa käyttöliittymän kautta, jolla myös hallitaan tekstiviestihälytysten piirissä olevia mittauskohteita sekä hälytysten vastaanottajia. Lisäksi asiakas voi ladata koosteraportteja lämpötiloista sekä muista olosuhdetiedoista. TempNet:iin on mahdollista liittää muita kuin lämpötila-antureita. Sensire:n valikoimaan sisältyy sensoreita myös sähkönkulutuksen, ruokakuljetusten, kosteuden ja hiilidioksidin mittaukseen.

TempNet Mobile tarjoaa langattoman ratkaisun lämpötilojen mittaukseen lämpötilakriittisten lääke- ja elintarvikekuljetusten aikana. TempNet Mobile koostuu webkäyttöliittymästä, tukiasemasta GPS-paikannuksella, lämpötila-antureista, ajoneuvonäytöstä sekä käsipäätteistä asiakkaan tarpeen mukaan. Sensoreilla mitatut tiedot siirtyvät automaattisesti tukiaseman kautta varmennettuun tietokantaan ja hälytyksiä voidaan lähettää reaaliaikaisesti kesken kuljetusten. Ajoneuvonäytön kautta voidaan kirjata työ-

lajit, esim. ajo, tauko, lastaus ja odotus. Näistä tiedoista voidaan tulostaa koosteraportit käyttöliittymän kautta. Lisäksi TempNet Mobilen käyttöliittymästä nähdään kuljetusautojen sijainnit, niiden ajamat reitit, lämpötilat saapuessa ja lähtiessä.

TempNet Handy on langaton käsipääte lämpötilojen tallennukseen. Järjestelmään kuuluu tukiasema ja käsipääte. Käsipääte tunnistaa mittaajan ja mittauskohteen joko RFID- tai viivakoodinlukijalla, mittaa kohteen lämpötilan ja tiedot siirtyvät automaattisesti tietokantaan. Lisäksi on tarjolla 3G-käsipääte, joka pystyy toimimaan ilman tukiasemaa lähettämällä mittaustiedot suoraan tietokantaan. Tietokannasta käyttäjän on mahdollista tulostaa lämpötilaraportit netistä.

TempNet on tarjolla kahtena erilaisena palveluna. Tarjolla on palvelu kiinteällä kuukausimaksulla, sekä investointimallina maksamalla laitteet ja kiinteää kuukausimaksua tietojen säilytyksen ylläpidosta. TempNet on kattava omavalvontajärjestelmä koko kylmäketjulle. [10.]

2.2.5 Resource Data Management

Resource Data Management, tästä eteenpäin RDM, on vuonna 2000 perustettu, nyt jo suuri kansainvälinen yritys, joka suunnittelee ja valmistaa ohjausjärjestelmiä rakennusten, energiankulutuksen ja jäähdytyksen hallintaan. Toimilaitteiden valmistamisen lisäksi RDM tarjoaa etähallinta- ja etävalvontapalveluita omasta palvelukeskuksesta. Etäpalveluita on tarjolla yksinkertaisesta hälytyspalvelusta koko laitoksen tai kiinteistön hallintaan.

RDM:n etäpalveluihin kuuluu mittausdatan tallennus, hälytyspalvelu, laitteiden etäohjaus ja yhteenvetoraportit. Lisäksi RDM pystyy tarkkailemaan laitteiden ja järjestelmien sekä työntekijöiden suorituskykyä. Asiakkaat voivat tarkastella, valvoa ja ohjata oman laitoksen tilaa internetin kautta. RDM voi tilata huoltotöitä perustuen kohteesta kerättyyn dataan ja järjestää vuosihuoltoja. Lisäksi RDM tarjoaa mielellään räätälöityjä palveluita asiakkaan tarpeisiin. [11.] RDM:n jälleenmyyjä on pohjoismaissa ruotsalainen Woodley, joka tarjoaa valvonta- ja hallintaratkaisuja kylmäalalla. [12]

2.2.6 Yhteenveto

Taulukkoon 2.1 on koottu yhteen avainasiat edellisissä kappaleissa esitellyistä palveluista. Taulukosta näkee selvästi, miten omavalvontajärjestelmät, joista mukana on TempNet, eroaa etäpalveluista. Niin etäpalveluista kuin omavalvontajärjestelmistäkin löytyy raportit sekä hälytysten lähettäminen, mutta omavalvontajärjestelmät eivät pysty tietojen analysointiin.

	Huurre HOT	e*Service	RDM	IWMAC	TempNet
Omavalvonta	X	X	X	X	X
HACCP ym. raportit	X	X	X	X	X
Hälytysten lähetys	X	X	X	X	X
Hälytysten analysointi	X	X	X	X	
Hälytyskeskus	X	X	X	X	
Etähallinta	X	X	X	X	
Energiaoptimointi	X	X	X	X	
Selainpohjainen	X	X	X		X
Aktiivinen etävalvonta	X	X	X		
BMS			X	X	
Käsipääte					X
COP	X				

Taulukko 2.1. *Yhteenveto esitellyistä palveluista.*

Kaikki palvelut ovat käytettävissä web-selaimella. Muut erot etäpalveluiden välillä liittyvät lisäpalveluihin. RDM sekä IWMAC toimivat kylmäautomaatioverkkojen lisäksi koko kiinteistön hallintajärjestelminä. TempNet tarjoaa järjestelmiinsä käsipäätteen. Ainoastaan Huurre HOT pystyy laskemaan kylmäjärjestelmän COP:n.

Nykypäivänä edelleen yleisin tapa on valvoa lämpötiloja omatoimisesti omavalvonnan avulla. Etäpalvelut kuitenkin mahdollistavat suurienkin säästöjen saavuttamisen ilman, että asiakkaan tarvitsee itse perehtyä tarkasti kylmäjärjestelmien teknisiin ratkaisuihin. Etäpalvelut säästävät asiakkaan aikaa, jolloin voidaan keskittyä laitoksen muihin töihin. Etäpalveluiden käyttäjäkunta onkin ollut kasvussa jo monta vuotta.

3 LÄHTÖKOHDAT JA TAVOITTEET

Huurre HOT:n liiketoiminta vastaa Huurre HOT -palvelun ylläpidosta, kehittämisestä, myynnin edistämisestä sekä asiakaspalvelusta, joka sisältää yhteydenpitoa asiakkaisiin, asiakkaiden koulutusta, asiakaspyyntöjen käsittelyä, kylmäjärjestelmien hälytysten esianalysointia, vikojen ennaltaehkäisyä ja huoltotöiden tilaamista.

Palvelun ylläpitäminen on tekniseltä puolelta palvelimien tilan valvomista, palvelimien tietoturvasta huolehtimista, muutostöiden suorittamista, mahdollisten ongelmien korjausta sekä järjestelmäpäivitysten asentamista. Ylläpidon tekninen puoli hoitaa myös uusien kohteiden konfiguroinnin Huurre HOT -palveluun. Lisäksi ylläpitoon kuuluu Huurre HOT -palvelun Huurre HOT Center, joka tarjoaa apua kylmäjärjestelmien konfigurointiin, Huurre HOT -palvelusivujen käyttöön sekä tietoliikenneongelmien ratkaisemiseen.

Huurre HOT:n kehittämiseen sisältyy uusien laitteiden integrointia osaksi järjestelmää, uusien toimintojen määrittämistä ja kehittämistä, ongelmien analysointia sekä palvelusivujen käytettävyyden parantamista.
Huurre Group on kehittänyt lukuisia kylmäalan tuotteita useiden vuosien ajan ja sillä on hyvin toimiva tuotekehitys. Tuotekehitys on noudattanut vahvasti vesiputousmallia. Huurre HOT on tuonut tuotekehitykseen mukaan myös ohjelmistokehityksen. Koska Huurteella ei ole kokemusta ohjelmistokehityksestä, on sitä lähdetty toteuttamaan hyvin tunnetun vesiputousmallin mukaan.

Huurre HOT -palvelun tekninen tuotekehitys on ohjelmistokehitystä, johon on olemassa erilaisia ohjelmistokehitysmalleja. Tämän työn tavoite on tutustua eri ohjelmistokehitysmalleihin, tutkia niiden sopivuutta Huurre HOT:n tilanteeseen ja määrittää selkeä ohjelmistokehitysprosessi Huurre HOT:lle.

4 TEORIAA OHJELMISTOKEHITYKSESTÄ

Ohjelmistotuotanto on yhteisnimitys niille työnteon ja työnjohdon menetelmille, joita käytetään tietokoneohjelmien ja -ohjelmistojen tuottamiseen. Tietokoneohjelmistojen kehittämisen systemaattista käsittelyä varten on luotu malleja ohjelmistojen valmistusprosesseista elinkaarimallin mukaisesti. Elinkaarimallissa ohjelmistokehitys on aikaan sidottu prosessi, jossa ohjelmiston varsinainen tekninen valmistus on vain osa kokonaisketjua. [16.]

Tässä kappaleessa esitellään ohjelmiston elinkaaren vaiheet sekä ohjelmistokehityksessä käytettyjä malleja. Kappaleen tavoite on antaa yleiskuva ohjelmistojen kehittämisestä ja erilaisista olemassa olevista ohjelmistokehitysmalleista. Kaikkia olemassa olevia malleja ei pyritä käsittelemään. Valitut ohjelmistokehitysmallit pyritään kuvaamaan yksityiskohtaisesti sekä niiden heikkoudet ja vahvuudet tuodaan esiin.

4.1 Ohjelmiston elinkaaren vaiheet

Ohjelmiston elinkaarella tarkoitetaan aikaa, joka käytetään ohjelmiston kehittämisen aloittamisesta sen käytöstä poistumiseen saakka. [17.] Ohjelmiston elinkaari jakaantuu kahteen pääluokkaan: kehitykseen ja ylläpitoon. Kehitykseen kuuluu vaatimusten määrittely, suunnittelu, toteutus, integrointi, testaus sekä julkaisu ja käyttöönotto.

4.1.1 Kehitys

Ohjelmiston elinkaari alkaa vaatimusten määrittelystä. Vaatimusten määrittelyssä kuvataan ohjelmistoprojektin tavoitteet ja vaatimukset, jotka valmiin järjestelmän tulisi täyttää. Vaatimusmäärittelyssä ei oteta millään tavoin kantaa siihen, miten tavoitteet voidaan saavuttaa. Vaatimusten määrittely tehdään tiiviissä yhteistyössä ohjelmiston asiakkaan kanssa, jossa asiakkaan turhien toteutusteknisten vaatimusten sivuuttaminen on tärkeää. [16, s. 78-81.]

Seuraava vaihe on suunnittelu. Suunnittelu koostuu kahdesta vaiheesta: toiminnallisesta määrittelystä sekä teknisestä määrittelystä. Toiminnallisessa määrittelyssä kirjoitetaan määrittelydokumentti, jossa kuvataan kaikki järjestelmän toteuttamat toiminnot ja liitokset järjestelmän ulkopuolelle. Kuvauksesta käy ilmi kaikki, mitä järjestelmällä voi tehdä ja miten ne voidaan tehdä. Määrittelyyn sisältyy esimerkiksi kuvaus käyttöliittymän ulkoasusta, valikoista ja asetuksista. Ideaalinen toiminnallinen määrittely ei jätä järjestelmän toiminnan kannalta mitään epäselväksi seuraavissa vaiheissa. Vaikka siihen

on käytännössä mahdotonta päästä erityisesti suuria järjestelmiä toteutettaessa, on siihen hyvä pyrkiä.

Tekninen määrittely, eli arkkitehtuurisuunnittelu, on toiminnallisen määrittelyn jälkeinen vaihe. Teknisen määrittelyn tavoite on kuvata ohjelmiston arkkitehtuuri tarkasti pohjautuen edellisen vaiheen toiminnallisuuden määrittelyyn. Tekniseen määrittelyyn sisältyy esimerkiksi käytettävät ohjelmointikielet, tarvittavat ohjelmistokomponentit ja niiden rakenteet, käytettävät tietorakenteet ja niiden väliset sovellusrajapinnat.

Arkkitehtuurin suunnitteluun on olemassa suunnittelumalleja, joiden tarjoamat valmiit ratkaisut suunnitteluongelmiin nopeuttavat prosessia sekä lisäävät toimintavarmuutta. Suunnittelumalleja ovat esimerkiksi epäkonkreettinen tehdas ja rakentajarajapinta. Riippuen ohjelmiston koosta ja monimutkaisuudesta, voidaan kuvaus kirjoittaa kerroksittain. Esimerkiksi ylimmässä kerroksessa voidaan kuvata palvelinympäristö, asiakasliityntä, tietovarastot ja niiden yhteys ulkopuolisiin järjestelmiin. Seuraavalla tasolla kuvataan komponentit pilkottuina pienempiin kokonaisuuksiin, joista käy ilmi niiden tarkempi määrittely. [16, s. 81-83.]

Kun suunnittelu on valmis, siirrytään toteutusvaiheeseen. Toteutukseen kuuluu varsinainen lähdekoodin kirjoittaminen ja tarvittavien oheiskomponenttien tuottaminen, joita voivat olla esimerkiksi grafiikka ja äänet. Jos edelliset vaiheet on toteutettu erinomaisesti, voi toteutukseen kuluva aika olla vain pieni osa koko ohjelmistoprojektiin käytetystä ajasta. Toteutusvaiheen jälkeen käytössä tulisi olla ajettava ohjelmisto.

Integroinnissa ohjelmapalaset yhdistetään toimivaksi kokonaisuudeksi. Integrointivaiheeseen kuuluu moduulien välisen kommunikaation testaaminen eri tilanteissa niin kattavasti kuin mahdollista. Järjestelmätestaukseen ja käytettävyystestaukseen voidaan siirtyä vasta, kun rajapintojen väliset toiminnot ovat tyydyttävällä tasolla.

Testausvaiheessa viimeistellään ohjelmisto julkaisua ja käyttöönottoa varten. Testaaminen koostuu testimenetelmien suunnittelusta, erilaisten testidokumenttien kirjoittamisesta sekä testiympäristöjen rakentamisesta. Testitulosten pohjalta ohjelmistoon tehdään tarvittavia muutoksia.

Testauksen alpha-vaiheessa kokeillaan luotujen toimintojen kykyä toimia oikein käytännön tilanteissa. Alpha-vaiheessa kirjoitetaan myös dokumentaatio sisäisen lähdekoodin toiminnasta. Testauksen beta-vaiheessa ohjelmistosta etsitään ja korjataan siihen jääneet ohjelmistovirheet. Ohjelmistovirheitä syntyy mm. beta-vaiheen aikana tehdyistä muutoksista sekä huolimattomuudesta ohjelmoinnissa ja beta-testaus onkin yksi eniten aikaa vaativimmista vaiheista ohjelmistonkehityksessä.

Virheitä löytyy lähes kaikista ohjelmistoista. Ohjelmistoyritykselle onkin erittäin tärkeää löytää ja korjata ohjelmistovirheet itse. Jos asiakas löytää ne ja kertoo huonoista käyttökokemuksista muille mahdollisille asiakkaille, saa ohjelmisto huonoa mainetta ja myynti voi kärsiä. Ohjelmiston testaamiselle ei pidä jättää kaikkea vastuuta hyvästä laadusta, vaan laatuajattelu on oltava mukana ensimmäisestä ohjelmistoprojektin vaiheesta lähtien. Tutkitusti testausvaiheessa ei löydetä käytännössä koskaan kaikkia virheitä, jonka lisäksi virheiden korjaaminen voi luoda uusia ohjelmistovirheitä. [16, s. 283-302.]

Kun ohjelmistossa on tietty määrä toiminnallisuuksia, sitä voidaan pitää julkaisukelpoisena. Julkaisukelpoisessa versiossa ohjelmisto on käytettävä ja sen ominaisuudet toimivat virheettömästi. Julkaisua varten ohjelmiston osat ja dokumentaatio kootaan yhdeksi paketiksi. Se, että julkaistaanko ohjelmistoversio, riippuu kaikkien ohjelmistoa kehittävien tahojen tyytyväisyydestä. Joskus on kuitenkin tärkeämpää saada ohjelmisto valmiiksi ajallaan kuin toteuttaa kaikki laatuvaatimukset.

Julkisesti saatavat ohjelmistot yleensä julkaistaan. Kun ohjelmisto on rakennettu vain tietyille asiakkaille, julkaisun sijaan tehdään käyttöönotto. Käyttöönotossa julkaisukelpoiseksi koottu ohjelmistopaketti asennetaan suunnitellusti halutuille laitteille. Käyttöönottoon liittyy usein laiteasennuksia, käyttöympäristön ja tietoliikenneyhteyksien konfigurointia sekä käyttäjien kouluttamista. [16, s. 41.]

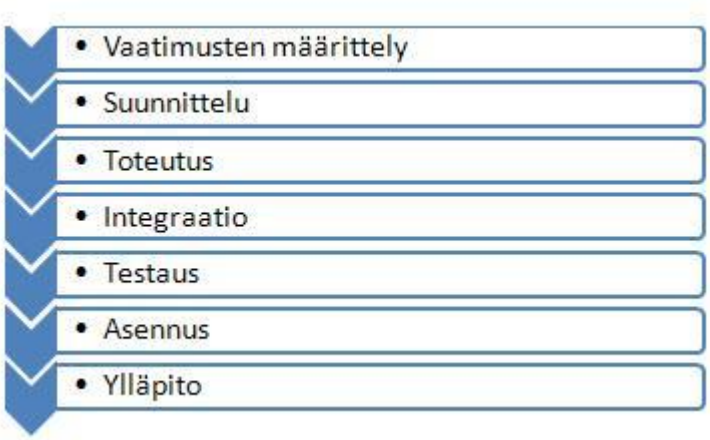
4.1.2 Ylläpito

Ohjelmiston ylläpidon tehtävä on pitää ohjelmisto toimintakuntoisena. Siihen sisältyy kaikki toimenpiteet, joilla ohjelmiston käyttäjät saadaan pidettyä tyytyväisinä. Kun uusia ohjelmistovirheitä löydetään, uusia toiminnallisuuksia tai vaatimuksia ilmaantuu, on ylläpidon tehtävä huolehtia toimenpiteistä niiden korjaamiseksi. Ylläpidon toimenpiteitä ovat esimerkiksi ohjelmistovirheiden raportointi ohjelmistotuotantoon ja uusien versioiden päivittäminen vanhojen tilalle. [16, s. 41.]

4.2 Vesiputousmalli

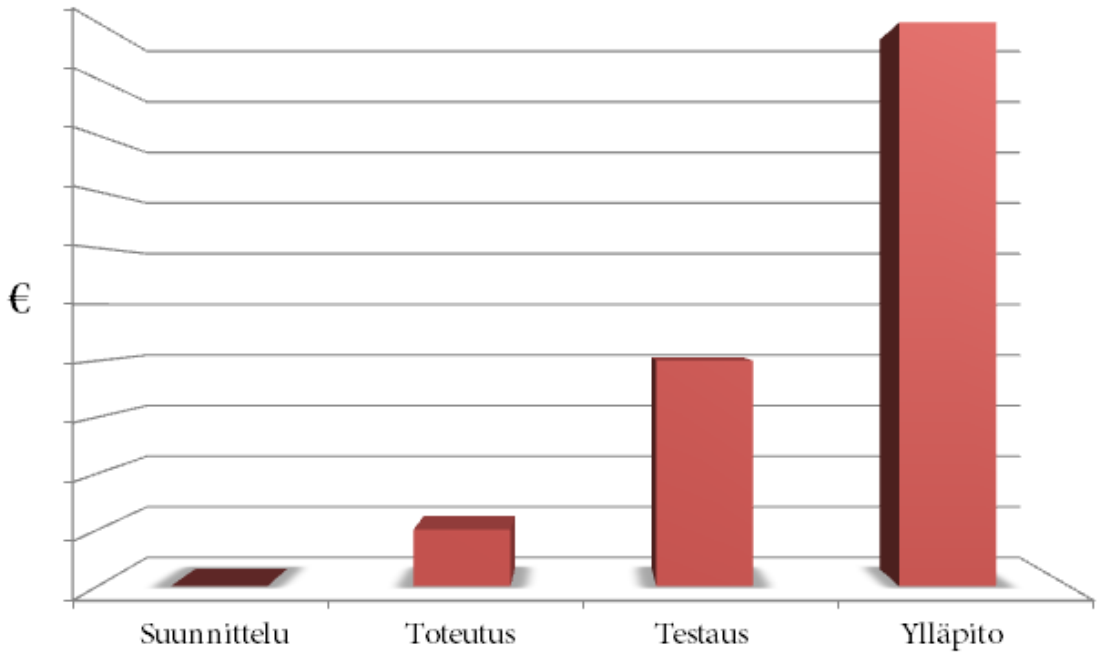
Vesiputousmalli on mielletty ensimmäiseksi varsinaiseksi prosessimalliksi ohjelmistotuotannossa. Winston Royce kirjoitti artikkelin “Managing the Development of Large Software Systems” vuonna 1970, jossa vesiputousmallin periaatteet esiteltiin. [18.] Koska varsinaisia ohjelmistotuotantomalleja ei tuohon aikaan vielä ollut, sai malli juuret valmistusprosesseista. Valmistusprosessit ovat hyvin strukturoituja ja vaiheistettuja, joissa muutoksien tekeminen edellisiin prosessin vaiheisiin oli erittäin kallista tai mahdotonta.

Roycen vesiputousmalli jakaa suunnittelu- ja toteutusprosessin lineaarisiin vaiheisiin, jossa prosessi etenee vaihe vaiheelta alaspäin kuin vesi vesiputouksessa. Malli sisältää kaikki tyypilliseen ohjelmistokehitykseen kuuluvat vaiheet, jotka ovat kuvattuna järjestyksessä kuvassa 4.1.



Kuva 4.1. *Vesiputousmallin vaiheet järjestyksessä.*

Ohjelmiston huolellinen suunnittelu sen elinkaaren alussa johtaa yleensä suuriin säästöihin projektin myöhemmissä vaiheissa. On olemassa tutkimuksia, joiden mukaan ohjelmistovirheen löytäminen vaatimusten määrittely- tai suunnitteluvaiheessa on rahallisesti, työmäärällisesti sekä ajallisesti halvempaa korjata myöhempiin vaiheisiin verrattuna. Kuva 4.2. havainnollistaa ohjelmistovirheiden korjaamisen kustannusten nousua vaihe vaiheelta. Arvioidaan, että kustannukset voivat nousta jopa yli 150-kertaisiksi, kun virheellistä ohjelmistoa pyritään ylläpitämään. [19.]



Kuva 4.2. *Ohjelmistovirheiden korjaamisen kustannukset eri projektin vaiheissa.*

Vesiputousmallissa on nimenomaan kyse varhaisen suunnittelun korostetusta merkityksestä. Ajan käyttäminen projektin varhaisessa vaiheessa vaatimusten määrittelyyn sekä suunnitteluun säästää aikaa, rahaa ja työmäärää projektin myöhemmissä vaiheissa.

Vesiputousmallin toimivuus edellyttää, että jokainen vaihe on toteutettu täydellisesti oikein ja täysin valmis ennen seuraavaan vaiheen siirtymistä.

4.2.1 Vesiputousmallin edut

Vesiputousmalli on ollut kauan käytössä, joten sen hyvät puolet tunnetaan erittäin hyvin. Vesiputousmallin etuna on mallin selkeä ja helppo omaksuttavuus, eli se tarjoaa ohjelmistokehitysprojektiin selkeän ja helposti ymmärrettävän ja hallittavan lähtökohdan. Ohjelmistokehitysprojektin etenemisen seuranta on helppoa, koska projektin etenemisen näkee katsomalla missä vesiputousmallin vaiheessa ollaan menossa.

Vesiputousmallin kattava dokumentointi mahdollistaa esim. uuden työntekijän helpon liittymisen projektiin missä tahansa projektin vaiheessa. Työntekijöitä voidaan nopeasti siirrellä sinne, missä resursseja tarvitaan. Kun työntekijä päättää lähteä projektista ennen sen valmistumista, ei tieto katoa työntekijän mukana, vaan se säilyy dokumentaatiossa. Vesiputousmallia käyttävät projektit eivät siis ole tarkasti sidottuja projektin työntekijöihin.

Lisäksi vesiputousmallille on kehitetty laaja teoria- ja työkalutuki, eikä malli vaadi projektin osallistujilta mitään erityistaitoja. Vesiputousmalli sopii hyvin hierarkkisiin organisaatioihin olemalla itsekin hyvän järjestelmällinen. Malli ei rajoita eikä vaadi mitään erityisiä tekniikoita, joten se on vapaasti räätälöitävissä. Kun työntekijät haluavat välttää jatkuvaa ryhmätyötä ja isoja projektipalavereita, toimii malli hyvin myös sellaisten tekijöiden kanssa. Vesiputousmallin avulla voidaan antaa asiakkaalle selkeä ja ymmärrettävä kustannusarvion jo projektin alussa, eikä kiireisen asiakkaan tarvitse välttämättä osallistua projektityöhön työn edetessä. [20.]

4.2.2 Vesiputousmallin kritiikki

Vesiputousmalli pohjautuu täydelliseen suunnitteluun. Toteutusongelmien ennakointi ei projektin suunnittelijoilta välttämättä aina onnistu, jolloin joudutaan keksimään uusi ratkaisumalli. Sen käyttäminen vaatii usein uutta suunnittelua. Yleisesti uskotaan olevan mahdotonta suunnitella mitään suurempaa ohjelmistoa etukäteen siten, että aiempiin vaiheisiin ei tarvitsisi palata.

Usein ohjelmistotuotannossa asiakkaat eivät osaa tarkasti määritellä omia vaatimuksiaan. Heidän käsitys vaatimuksista selkeytyy vasta, kun he ovat päässeet kokeilemaan jollain tasolla toimivaa prototyyppiä. Ei myöskään ole epätavallista, että asiakas muuttaa vaatimuksiaan kesken projektin. Vesiputousmallissa se tarkoittaa sitä, että suuri osa vaatimusten määrittely- sekä suunnitteluvaiheeseen käytetystä työstä joudutaan hylkäämään ja tekemään uudelleen.

Vesiputousmallia sanotaan epäsoveliaaksi ohjelmistoprojektiin, jossa projektin probleema muuttuu jatkuvasti vaatimusten muuttumisesta ja itse probleemasta tehtyjen uusien havaintojen takia. Näiden muutosten tekemisen helpottamiseksi voidaan ohjelmistoa etukäteen valmistella mahdollisia päivityksiä varten muuttamalla sen rakennetta

tai hyödyntämällä modulaarisuutta, joka parantaa ohjelmiston joustavuutta suunnittelun suhteen. Molemmat vaativat kuitenkin osaamista sekä aikaa. [20.]

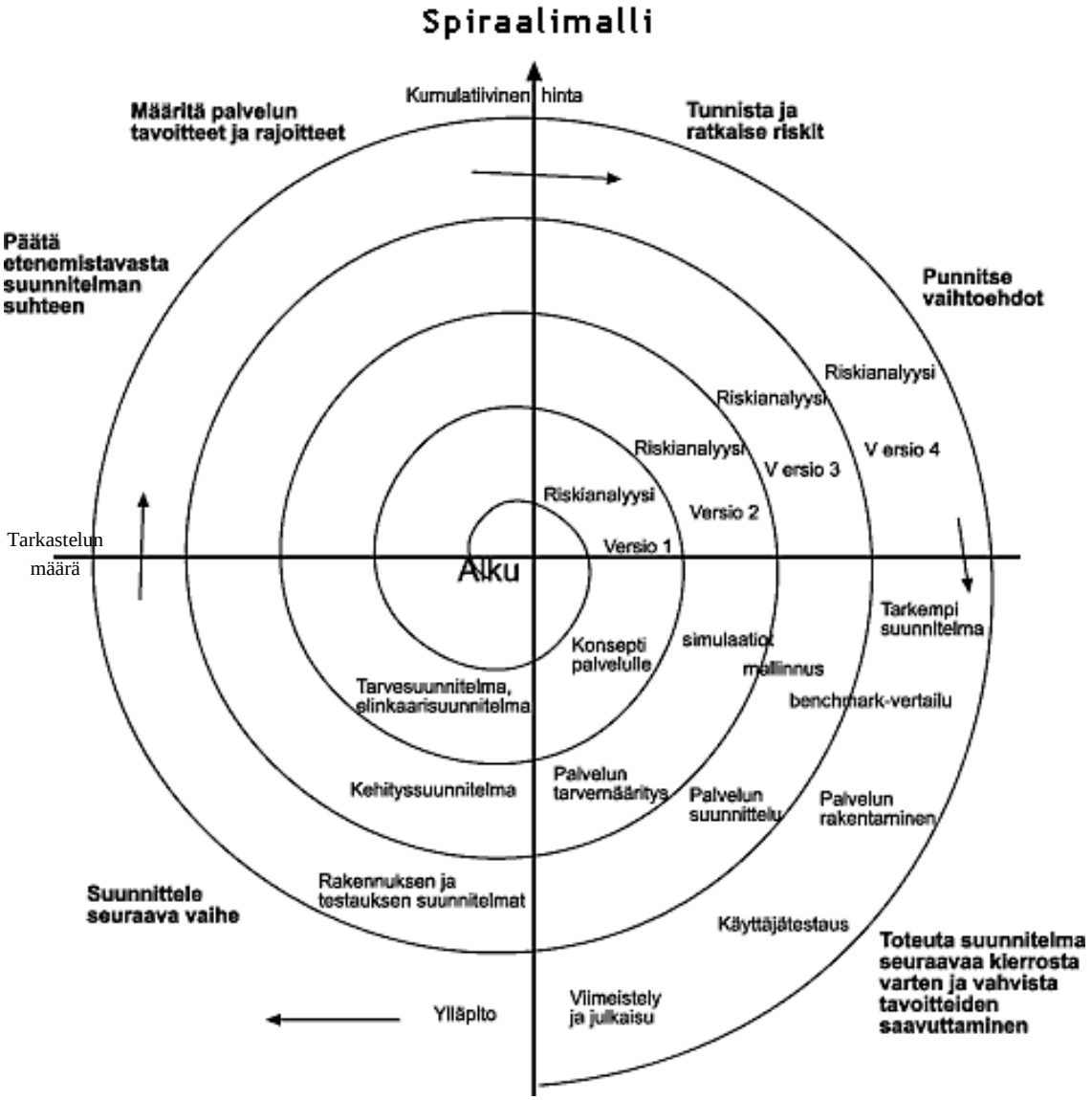
4.2.3 Vesiputousmallin variaatiot

Vesiputousmallin ongelmien ratkaisemiseksi on olemassa muokattuja vesiputousmalleja. Ne pyrkivät poistamaan puhtaan vesiputousmallin saamaa kritiikkiä. Kaikki ohjelmistosuunnittelumallit yhtenevät osittain vesiputousmalliin, mutta ne pyrkivät usein tarjoamaan enemmän joustavuutta suunnitteluprosessiin. Joustavuus mahdollistaa siirtymisen seuraavaan vaiheeseen ilman että edellistä on toteutettu täydellisesti ja se pyrkii mahdollistamaan helpomman muutoksen tekemisen edellisiin vaiheisiin kuin puhdas vesiputousmalli. [18]

4.3 Spiraalimalli

Spiraalimalli on ohjelmistokehitysmalli, joka yhdistää iteratiivisen vaatimusmäärittelyn sekä lineaaristen mallien systemaattisen lähestymistavan. Malli pitää sisällään samoja vaiheita kuin vesiputousmalli, mutta spiraalimallissa edetään iteroiden useiden toistuvien syklien avulla.

Spiraalimalli siis yhdistää iteratiivisen kehityksen vesiputousmallin järjestelmälliseen ja kontrolloituun aspektiin. Sillä mahdollistetaan tuotteen parantaminen jokaisen spiraalin kierroksen aikana. Spiraalimalliin kuuluu selvästi myös riskienhallinta. Suurten teknisten- sekä johtamistason riskien tunnistaminen ja niiden määrätietoinen minimointi auttaa pitämään ohjelmistokehitysprosessin hallinnassa. [21.] Kuva 4.3 kuvaa spiraalimallin mukaista etenemistä.



Kuva 4.3. *Prosessikuvaus spiraalimallista. [22.]*

Spiraalimallin perusta on jatkuva tuotteen jalostaminen kierroksittain. Spiraalimallia noudattava projekti aloitetaan spiraalin keskeltä ja jokaisella kierroksella käydään läpi tehtäväalueet. Tehtäväalueet ovat

- tavoitteiden, vaihtoehtojen ja rajoitteiden määrittely
- vaihtoehtojen arviointi ja riskien tunnistaminen ja ratkaiseminen
- toteuttaminen ja testaaminen sekä
- seuraavan kierroksen suunnittelu.

Jokaisen syklin tuotos on aina edellisen syklin tuotoksen jatke. Vaatimusten määrittelyä, suunnittelua ja riskien analysointia tehdään useita kertoja. Lopullinen suunnitelma, implementaatio, integraatio sekä testaus tapahtuvat vasta viimeisellä spiraalin kierroksella. Spiraalimalli käyttää monia samoja vaiheita, jotka löytyvät vesiputousmallista,

erityisesti samassa järjestyksessä, mutta ne erotellaan toisistaan suunnittelulla, riskien arvioinnilla sekä prototyyppien sekä simulaatioiden rakentamisella.

Spiraalimallissa dokumentaatiota tuotetaan vain, kun sitä tarvitaan. Dokumentaatiossa käsitellään prosessin sen hetkisen tuotteen informaatiota. Kaikkea dokumentaatiota ei siis luoda vesiputousmallin tapaan projektin alussa, vaan ideana on, että dokumentaatio kuvaa vain sen hetkistä tuotetta, jonka on tarkoitus jatkuvasti päivittyä käyttäjien arviointien perusteella. [23, s. 56-57.]

4.3.1 Mallin hyvät puolet

Spiraalimalli mahdollistaa uusien ominaisuuksien lisäämisen tuotteeseen, kun niistä tulee käytettäviä tai ne tiedostetaan. Samaa periaatetta noudatetaan useita ohjelmistoversioita käyttävissä menettelytavoissa ja se mahdollistaa sujuvan siirtymisen tuotteen ylläpidon toimenpiteisiin, koska spiraalimalli soveltuu erittäin hyvin ylläpidon malliksi. Malli soveltuu erityisesti sellaisten palveluiden ja tuotteiden suunnitteluun, joissa perusratkaisu, eli mitä oikeastaan on tarkoitus tehdä, ei ole täysin selvä. Prosessin edetessä ongelman esittämisen ja ratkaisun sykleinä hioutuu valmis ratkaisuehdotus, jota voidaan analysoida tarkemmin. [24.]

Spiraalimallissa käyttäjät tuodaan mukaan projektiin jo alkuvaiheessa. Projekteissa, joiden avainasemassa on käyttöliittymä ja käytettävyys, käyttäjien osallistuminen auttaa projektin onnistumisessa huomattavasti. Lisäksi spriaalimallilla saadaan usein toimitettua nopeasti käyttäjälle käyttäjän haluamia toiminnallisuuksia, vaikka koko tuote ei vielä olisi valmis.

Ohjelmistoprojektit sisältävät useita eri alueiden riskejä, kuten projektin budjetin ylitys, muuttuvat vaatimukset, projektin tärkeiden työntekijöiden menettäminen, mahdolliset viiveet, muiden kehittäjien kilpailu ja teknologiset läpimurrot, jotka voivat tehdä projektista vanhentuneen. Spiraalimallin pääpaino onkin riskien minimoimisessa ja se sopii siihen mainiosti mallin jokaisella kierroksella tehtävän riskianalyysien ansiosta.

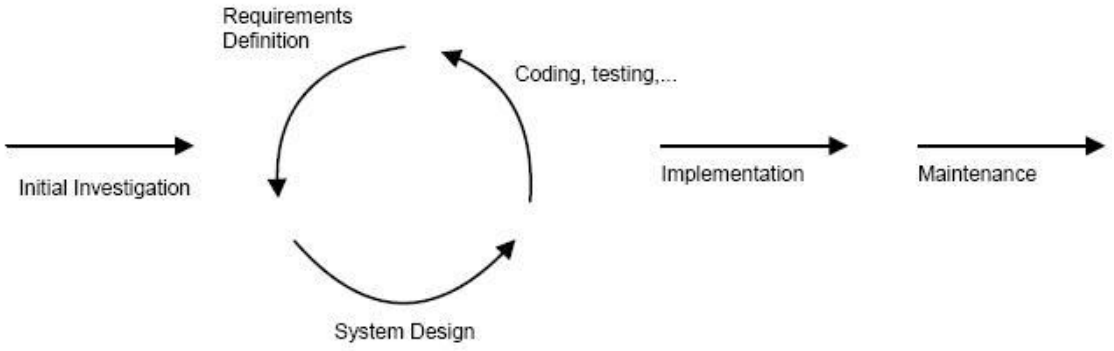
4.3.2 Mallin huonot puolet

Spiraalimallin sanotaan toimivan parhaiten suurille ja monimutkaisille projekteille, joiden kustannukset ovat myös korkeita. Pienille projekteille malli on liian monimutkainen ja se voi hidastaa tuotteen valmistumista. Monimutkaisen spiraalimallin tarkka seuraaminen on työlästä ja etenkin projektin riskien analysointi ja arviointi vaatii työntekijältä erityistaitoja. Projektin onnistuminen riippuukin pitkälti riskien analysointivaiheen onnistumisesta. Jos esimerkiksi riski arvioidaan väärin, se saattaa seuraavilla sykleillä moninkertaistua.

Usein projekteissa luotuja prototyyppejä ei voi käyttää muissa projekteissa käyttäjien toiveiden ja niiden takia tehtävien muutoksien jälkeen. Budjetin ja aikataulun pitäminen käyttäen spiraalimallia on hankalaa. Dokumentaation suuren määrän tarve projektin välivaiheissa tekee projektinhallinnasta monimutkaista. [21; 25; 26.]

4.4 Iteratiivinen ja inkrementaalinen malli

Iteratiivinen ja inkrementaalinen malli on ohjelmistotuotantomalli, joka on iteratiivisen mallin ja inkrementaalisen mallin yhdistelmä. Iteratiivisen ja inkrementaalisen mallin perusidea on kehittää järjestelmä toistuvien syklien kautta, eli iteratiivisesti, ja pienissä osissa kerrallaan, eli inkrementaalisesti. Näin ohjelmistokehittäjät voivat ottaa hyödyn irti edellisten versioiden kehityksessä sekä järjestelmän käytöstä opituista asioita. Käyttö mahdollistetaan toteuttamalla prosessin alussa yksinkertainen versio ohjelmistosta, joka täyttää vain osan ohjelmiston vaatimuksista. Yksinkertaista versiota parannellaan iteratiivisesti, kunnes koko järjestelmä on toteutettu. Jokaisessa iteraatiossa muutetaan suunnitelmia sekä lisätään uusia toiminnallisuuksia. Kuva 4.4. esittää iteratiivisen mallin toimintaperiaatetta.



Kuva 4.4. *Iteratiivisen kehityksen perusperiaate. [27]*

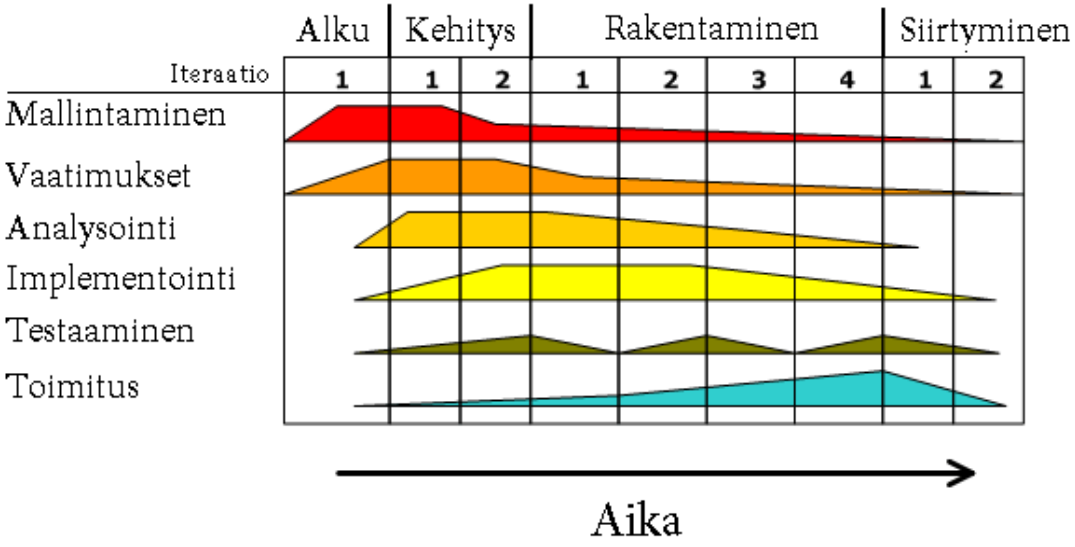
Prosessissa itsessään on kolme askelta: alustaminen, iteraatioaskel ja projektin hallintalistan päivitys. Alustamisessa tehdään järjestelmästä perusversio. Perusversion tarkoitus on päästää käyttäjä käsiksi yksinkertaiseen versioon ohjelmistosta, jonka avulla ongelman avainasiat nähdään selvästi ja ymmärretään, miten tuote pitäisi implementoida järkevästi ja helposti.

Iteraatioaskeleessa tehdään uudelleensuunnittelu ja sen pohjalta uusi implementaatio. Implementaation tulee olla yksinkertainen, suoraviivainen ja modulaarinen soveltuen hyvin mahdollisiin tuleviin uusiin muutoksiin ja suunnitelmiin. Suunnitelmien tarkkuus määräytyy täysin projektin mukaan: Kevyet projektit eivät tarvitse suunnitelman dokumentointia juurikaan, vaan lähdekoodi voi toimia suurena osana järjestelmän dokumentaationa, kun taas kriittiset projektit vaativat iteraation täydellisen dokumentaation, joihin kuuluu esimerkiksi ilmailualaan liittyvät projektit.

Projektin hallintalistan avulla ohjataan iteraatioprosessia. Projektin hallintalista sisältää kaikki tehtävät, jotka pitää hoitaa. Uudet toiminnot, jotka pitää implementoida, sekä uudelleensuunnittelua tarvitsevat ratkaisut, ovat seurannassa hallintalistalla. Projektin hallintalistaa uudistetaan jatkuvasti analysointivaiheen tuloksena.

Inkrementaalisessa kehittämisessä järjestelmän toiminnallisuudet jaetaan osioihin. Jokainen osio kulkee kaikkien ohjelmistotuotannon vaiheiden läpi, eli vaatimusten määrittelystä aina käyttöönottoon asti. Jokainen tällainen prosessi voidaan jakaa neljään vaiheeseen:

- Alkuvaiheessa määritellään osion laajuus, toiminnalliset vaatimukset sekä riskit.
- Kehitysvaiheessa luodaan ohjelman toimiva arkkitehtuuri, joka pienentää suurimmat riskit ja täyttää järjestelmän ei-toiminnalliset vaatimukset.
- Rakentamisvaiheessa järjestelmän arkkitehtuuri täydennetään inkrementaalisesti valmiiksi.
- Siirtymisvaiheessa järjestelmä toimitetaan toimintaympäristöön.



Kuva 4.5. *Iteratiivisen ja inkrementaalisen mallin vaiheet. [28]*

Jokainen vaihe voidaan jakaa yhteen tai useampaan iteraatioon, kuten kuvassa 4.5. on tehty. Kuvan värillisten palkkien paksuudesta nähdään, että alkuvaiheessa suurin työ tehdään mallintamisessa ja vaatimusten määrittelyssä, kun rakentamisvaiheessa analysoinnilla, implementoinnilla ja testaamisella on suurempi rooli. [29.]

4.4.1 Mallin hyvät puolet

Ohjelmiston arkkitehtuurin rakentaminen iteratiivisesti ja inkrementaalisesti antaa kehittäjille mahdollisuuden huomata puutteet ja ongelmat arkkitehtuurissa ja tehdä tarvittaessa suuriakin muutoksia ohjelmiston arkkitehtuuriin prosessin alkuvaiheissa. Se on huomattavasti edullisempaa suhteessa muutosten tekemiseen projektin loppuvaiheessa.

On erittäin yleistä, että ohjelmiston vaatimukset muuttuvat kesken projektin. Usein asiakkaan on vaikea ymmärtää koko järjestelmä, kun he voivat tutustua siihen pelkästään dokumentaation avulla. Iteratiivinen ja inkrementaalinen malli jakaa projektin osiin, joista jokainen osa on toimiva versio jostain järjestelmän osasta. Keskittymällä niihin, asiakkaan ja kehittäjien on helppo neuvotella jatkuvasti järjestelmän vaatimuksista, jolloin vältytään turhalta etukäteensuunnittelulta.

Inkrementaalisuudella tuodaan järjestelmään jatkuvasti uusia toimintoja ja parannuksia. Se antaa asiakkaalle mahdollisuuden seurata projektin etenemistä selkeästi toteutettujen toimintojen pohjalta. Kun uusi toiminto tuodaan järjestelmään yksi kerrallaan, kehittäjät pystyvät paikantamaan ongelmia tuottavat toiminnot helposti ja tarvittaessa suunnittelemaan ne uudelleen alusta asti.

Iteratiivinen ja inkrementaalinen malli antaa projektitiimille mahdollisuuden keskittyä minimoimaan suurimmat riskit jo projektin elinkaaren alussa. Mallin mukaan kehittäjien on hallittava iteraatioita perustuen jatkuvaan riskien arviointiin ja analysointiin, jolloin seuraavassa iteraatiossa on aina vähemmän riskejä kuin edeltävässä iteraatiossa. [30.]

4.4.2 Mallin huonot puolet

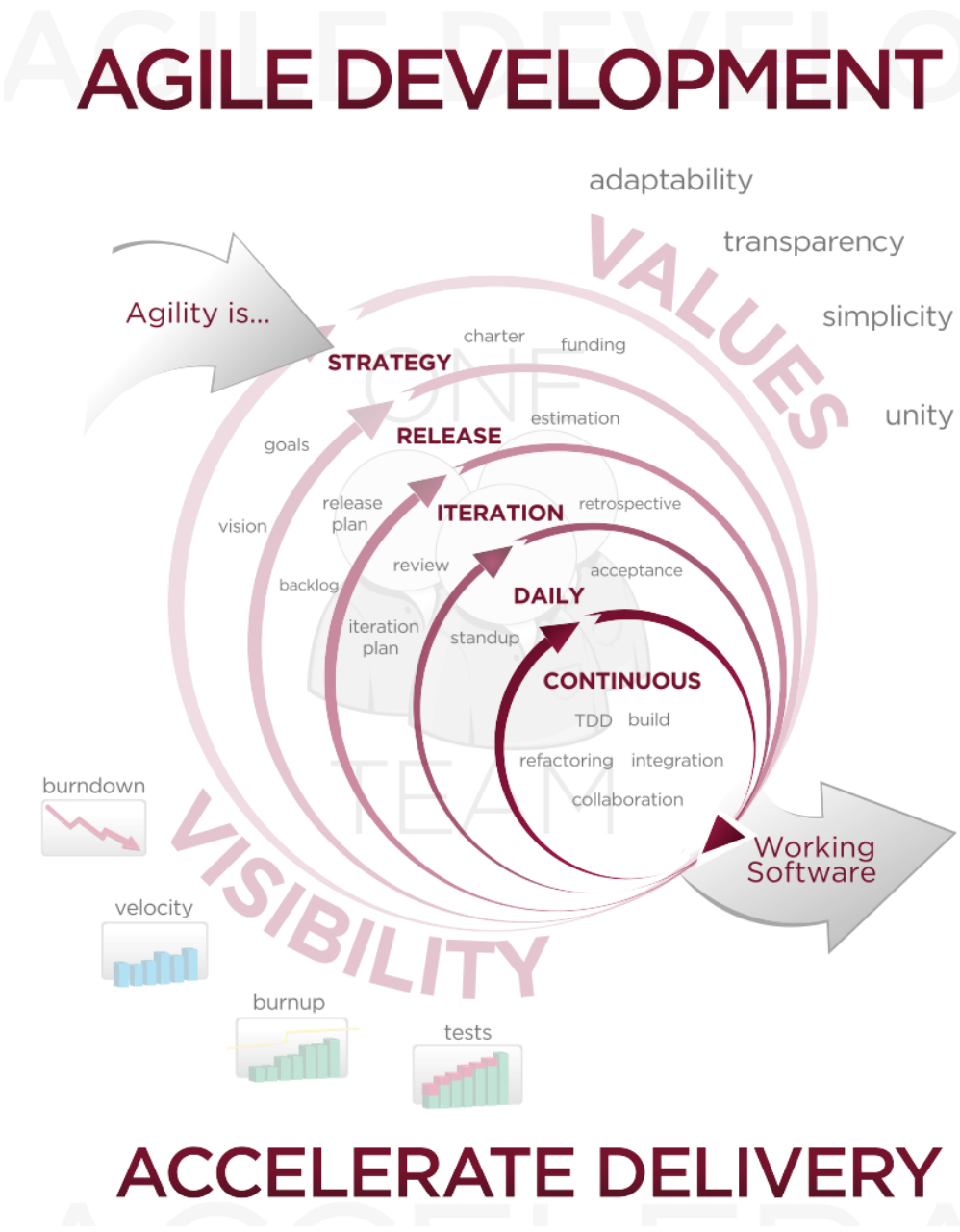
Iteratiivisen mallin käyttö voi saada projektin jumiin silmukkaan. Jos jatkuvasti joudutaan ongelmien löytymisen takia palaamaan takaisin suunnittelemaan, korjaamaan, implementoimaan ja taas testaamaan, voi budjetti ja aikataulu ylittyä.

Malli vaatii tuotteen käyttäjiltä aktiivista osallistumista. Jos käyttäjät eivät osallistu, on mahdollista, että malli on turhan monimutkainen projektia varten. Vaikka käyttäjien osallistuminen on hyvä asia projektin kannalta, vaatii se myös huomattavan määrän aikaa projektin henkilökunnalta ja mahdollisesti aiheuttaa viiveitä. Myös asiakkaan epäviralliset pyynnöt jokaisen vaiheen jälkeen voi aiheuttaa sekaannusta ja johtaa vaatimusten lisääntymiseen. [31.]

Suuri dokumentaation määrä saattaa aiheuttaa ongelmia, kuten myös järjestelmän toimintojen ja ominaisuuksien jako pienempiin osuuksiin. Iteraatioiden kehityksessä on jatkuvasti pidettävä mielessä integrointi, koska muuten osia ei saada yhdistettyä kokonaisuuteen. [32.]

4.5 Ketterä ohjelmistokehitys

Ketterä ohjelmistokehitys (engl. agile software development) on iteratiiviseen ja inkrementaaliseen ohjelmistokehitykseen pohjautuva joukko ohjelmistokehitysmenetelmiä, jossa vaatimukset ja ratkaisut kehittyvät itseorganisoituneiden ja eri toimintoihin keskittyvien työryhmien yhteistyönä. Se korostaa adaptiivista suunnittelua, evolutiivista kehitystä ja käyttöönottoa, aikasidonnaisten iteraatioiden hyödyntämistä ja rohkaisee nopeaan ja joustavaan muutoksiin reagoimiseen. Kuva 4.6. esittää visuaalisesti ketterää ohjelmistokehitystä.



Kuva 4.6. *Ketterä ohjelmistokehitys kuvattuna visuaalisesti. [33.]*

Ketterän kehityksen maailmassa lähdetään siitä, että ei ole vain yhtä oikeaa tapaa saavuttaa haluttua lopputulosta. Ketterät menetelmät ovatkin harvoin tarkasti määriteltyjä ohjeistoja eri tilanteiden toimintaohjeiksi. Ketterästä ohjelmistokehityksestä on julkaistu manifesti Agile Manifesto, jota pidetään ketterän kehityksen perusmääritelmänä. Manifestin on julkaissut 17 ketterän kehityksen puolestapuhujaa, joiden tarkoitus oli luoda yhteistä pohjaa ketterille menetelmille ja edistää näin ketterän menetelmän idean leviämistä.

Manifestin sisältö on seuraava:
“Löydämme parempia tapoja tehdä ohjelmistokehitystä, kun teemme sitä itse ja autamme muita siinä. Kokemuksemme perusteella arvostamme:

- Yksilöitä ja kanssakäymistä enemmän kuin menetelmiä ja työkaluja
- Toimivaa ohjelmistoa enemmän kuin kattavaa dokumentaatiota
- Asiakasyhteistyötä enemmän kuin sopimusneuvotteluja
- Vastaamista muutokseen enemmän kuin pitäytymistä suunnitelmassa

Jälkimmäisilläkin asioilla on arvoa, mutta arvostamme ensiksi mainittuja enemmän.” [34]

4.5.1 Ketterien kehitysmenetelmien yleiset periaatteet

Ketterästä kehityksestä on olemassa useita käytäntöjä, mutta niistä kaikista löytyy samoja piirteitä. Ketterän kehityksen ideologiassa lähdetään siitä, että kaikkien vaatimusten kirjaaminen etukäteen ei usein onnistu. Vaikka siinä jollain tasolla onnistuttaisiin, projektin aikana on todennäköistä löytää määrittelystä puutteita tai tehdä määrittelyyn muutoksia. Ennakkomäärittelyihin ei siinä tapauksessa ole järkevää käyttää liikaa työtunteja. Mitä vähemmän toteutettavasta ohjelmistosta osataan sanoa ennakkoon, sitä todennäköisemmin projektissa kannattaa soveltaa ketteriä menetelmiä.

Toinen keskeinen asia ketterässä kehityksessä on työskentelyn keskittäminen korkeimman prioriteetin ominaisuuden tekemiseen. Kun siihen ei kiinnitetä huomiota, toteutetaan usein ensin helppoja toimintoja, joilla projektin edistymistä voidaan osoittaa. Tällöin vaikeimmat ja riskeiltään suurimmat toiminnot jäävät projektin loppuun. Yleensä juuri vaikeimmat toiminnot tuovat asiakkaalle suurimman hyödyn. Vaikeisiin toimintoihin keskittyminen parantaa myös projektin ennustettavuutta.

Kolmas periaate ketterissä menetelmissä on iteratiivisuus ja inkrementaalisuus. Useat projektin ongelmat paljastuvat vasta käyttöönottovaiheessa. Jakamalla projekti pienempiin osiin siten, että ohjelmiston elinkaari tulee käydyksi läpi useita kertoja, paljastaa mahdolliset ongelmat jo sen alkuvaiheessa.

Perinteisesti, kuten vesiputousmallissa, ohjelmiston testaus jätetään projektin loppuun, jolloin ei enää olisi aikaa tehdä korjauksia. Jatkuva laadunvarmistus ja testaus kuuluvat ketterään kehitykseen alusta loppuun. Määräajan umpeutuessa, kun ohjelmiston osasta on osio valmiina, on valmis osio testattu ja hyväksi todettu.

Kaikissa ketterissä menetelmissä ennakkomäärittely suositellaan tehtäväksi keskittyen käyttäjätarinoihin teknisten yksityiskohtien sijaan. Eroja perinteisiin vaatimusmäärittelytekniikoihin ei juurikaan ole, mutta olennainen poikkeus on ennakkomäärittelyprosessin pitäminen laajuudeltaan hallinnassa: Käyttäjätarinoiden tulisi liittyä liiketoiminnallisesti tärkeisiin asioihin, eikä järjestelmän yksityiskohtaiseen toiminnan kuvaamiseen.

Yhteenvetona ketterän kehityksen yleisistä periaatteista voidaan sanoa, että ketterien menetelmien yleisesti tiedossa olevat vahvuudet ovat etenkin toteutusvaiheen tekemisen tehostamisessa. Ketterät menetelmät huolehtivat siitä, ettei toteutusvaiheessa ajauduta väärille urille tai käytettäisi rahoja epäolennaisten toimintojen hiomiseen. Ketterän kehi-

tyksen tärkeässä roolissa ovat myös monenlaiset jatkuvan testauksen ja laadunvarmistuksen toimintatavat. Nämä varmistavat sen, että mitä tahansa saadaankin valmiiksi, niin lopputulos on toimivaa. [35.]

4.5.2 Ketterien kehitysmenetelmien yleiset käytännöt

Päiväpalaveri

Kaikki ketterän kehityksen menetelmät pitävät suoraa keskustelua kasvotusten dokumenttien kirjoittamista tärkeämpänä. Monissa implementaatioissa työryhmän kesken pidetään päivittäin palaveri, jossa jokainen käy läpi mitä edellisenä päivänä on tehnyt, mitä aikoo tehdä tänään ja minkälaisia esteitä kehitystyöllä on näköpiirissä. On tärkeää, että myös asiakkaan edustaja on mukana palavereissa. Palaverit pyritään pitämään lyhyinä ja normaalia on, että ne pidetään seisaaltaan. Silloin palaverit pysyvät maksimissaan noin 15 minuutin mittaisina. Palaverien tarkoitus on löytää mahdollisia ongelmia ja tuoda ne kaikkien tietoisuuteen. [36.]

Käyttäjätarinat

Ketterässä kehityksessä ei yleensä tehdä täydellistä vaatimusmäärittelyä etukäteen, vaan vaatimukset muotoutuvat projektin edetessä. Projektin käynnistämiseksi voidaan hyödyntää käyttäjätarinoita. Käyttäjätarina on yksi selkeä toiminnallinen vaatimus, johon järjestelmän on toteutettava. Tarinoiden teksti pyritään pitämään erittäin lyhyenä, mutta niistä on käytävä ilmi kuka pystyy tekemään mitä ja miksi.

Usein ketterän kehityksen työryhmät työskentelevät samoissa tiloissa avokonttoreissa. Työryhmään kuuluu jokainen, joka tarvitaan ohjelmiston valmiiksi saamiseen, joten ohjelmoijien lisäksi työryhmään kuuluu myös asiakkaan edustaja. Asiakkaan edustajat ovat tärkeitä, koska heitä tarvitaan vaatimusmäärittelyjen yhteydessä käyttäjätarinoiden luomisessa. He voivat olla esim. tuotepäälliköitä, liiketoiminta-analyytikkoja tai varsinaisia käyttäjiä.

Käyttäjätarinoita pyritään kirjoittamaan mahdollisimman suuri määrä ottamatta kantaa niiden toteuttamiseen. Hyvillä käyttäjätarinoilla on yhteisiä piirteitä:

- Riippumattomuus: Tarinoiden riippuvuus toisistaan voi vaikeuttaa priorisointia sekä toteutukseen vaaditun työajan arviointia.
- Neuvoteltavuus: Tarinoissa pitää olla joustamisen varaa, eivätkä ne ole sopimuksia.
- Arvokkuus: Tarinan pitää tuoda lisäarvoa käyttäjälle.
- Arvioitavuus: Tarinoita pitää pystyä arvioimaan, koska suunnittelu pohjautuu niihin.
- Pienuus: Pitkät tarinat ovat monimutkaisia ja riskialttiita. Niitä on vaikea tehdä ja tulkita. Usein ne voidaan jakaa pienempiin osatarinoihin.
- Testattavuus: Tarinoita on pystyttävä testaamaan.

Käyttäjätarinat eivät aina ole oikea ratkaisu kaikkiin tilanteisiin. Hyvin suurissa projekteissa tarinoita voi muodostua niin suuri määrä, ettei niiden käsittely ole mahdollista. Ongelman ratkaisemiseksi voidaan luoda korkean tason tarinoita tai ryhmitellä tarinoita teemoittain.

Käyttäjätarinat keskittyvät parantamaan työryhmän sisäistä viestintää. Dokumentaation ja formaalin viestinnän tarve syntyy, kun vaatimuksia on tarpeen kommunikoida useille eri tahoille, esimerkiksi monitoimittajaprojekteissa. [36.]

Jatkuva integraatio

Ketterässä kehityksessä työt pilkotaan inkrementteihin. Sen lisäksi iteraatiot pyritään pitämään lyhyinä 1-4 viikon sykleinä. Jokainen työryhmässä tehtävä iteraatio pitää sisällään kaikki pienen ohjelmistoprojektin julkaisuun tarvittavat tehtävät: projektisuunnittelun, vaatimusanalyysin, ohjelmistosuunnittelun, koodauksen, testauksen sekä dokumentoinnin. Iteraation lopussa tuotos integroidaan osaksi koko ohjelmistoa ja se esitellään sidosryhmille.

Keskivaiheen integroinneilla vältytään lopun integraatiorypistykseltä, josta voi helposti tulla hyvin pitkä ja vaikea. Keskivaiheen integrointia kutsutaan jatkuvaksi integraatioksi. Jatkuva integraatio säästää projektin ikäviltä yllätyksiltä projektin päätösvaiheessa ja se myös minimoi projektin kokonaisriskiä. Jatkuvaan integraatioon kuuluu kaikki toimenpiteet, jotka kuuluvat tuotteen julkaisuun. Se voi sisältää esimerkiksi vain tuoreimman ohjelmistoversion noutamisen versionhallinnasta, kääntämisen ja lopputuotteen siirtäminen asennushakemistoon. Toisessa projektissa siihen voi kuulua edellä mainittujen lisäksi verkkopalvelimen alustaminen, web-palvelujen asentaminen ja käyttöohjeiden kirjoittaminen.

Ketterässä kehityksessä edistymisen mittarina toimii toimiva ohjelmisto, joka yhdessä työryhmien sisäisen henkilökohtaisen kommunikaation kanssa tuottaa huomattavasti vähemmän kirjoitettua dokumentaatiota kuin muut ohjelmistokehitysmallit. Toimiva ohjelmisto edistymisen mittarina ajaa sidosryhmät priorisoimaan omia vaatimuksia ja toiveita. [36.]

Jälkitarkastelu

Ketterään kehitykseen kuuluu jälkitarkastelu. Jälkitarkastelun idea on pysähtyä miettimään nykyisiä käytäntöjä ja toimintatapoja erityisesti jokaisen toimituksen jälkeen. Jälkitarkasteluissa mietitään, olivatko valitut käytännöt ja toimintatavat parhaita mahdollisia ja olisiko niissä parannettavaa. Jälkitarkastelupalaveri pidetään normaalisti projektin päättymisen jälkeen ja niissä tuodaan esiin projektin aikana saadut kokemukset mahdollista myöhempää hyödyntämistä varten. [36.]

Yhteisomistus

Kun työskennellään tiimissä, jossa on enemmän kuin yksi ohjelmoija, on erittäin tärkeää, että koodin omistajuus on selvillä. Perinteisessä luokkakohtaisessa omistajuudessa vain yksi henkilö saa tehdä muutoksia omistuksessaan oleviin luokkiin. Se saattaa hidastaa työn edistymistä, kun aina luokkaan tehtävä muutokset pitää tehdä yhden henkilön kautta. Yhteisomistajuudessa kaikki koodi on koko ohjelmointityöryhmän yhteisessä omistuksessa, jolloin kaikki jäsenet saavat tehdä muutoksia mihin luokkaan tahansa. Yhteisomistajuuden pitäisi nopeuttaa työntekoa, kun muutoksen tekeminen ei ole yksittäisestä ihmisestä kiinni. Malli vaatii toisaalta myös hyvää tiimihenkeä, sitoutumista ja vastuun ottamista koko tuotteesta, koska yhteisomistajuudessa komponentissa oleva virhe on kaikkien vastuulla. [36.]

Tasainen tahti

Ohjelmointi on ajatustyötä, joka rasittaa aivoja enemmän kuin rutiiniluontoinen ruumiillinen työ. Ylitöiden tekeminen ei pidemmän päälle nopeuta työn edistymistä, koska ihminen jaksaa tehdä pitempää päivää tuottavasti vain muutaman päivän tai viikon. Sen jälkeen väsymys alkaa heikentää suorituskykyä. Tasaisella tahdilla tarkoitetaan työtahdin pitämistä sellaisena, jota ihminen voi jatkaa loputtomiin. Sillä myös voidaan varmistaa, että työryhmä toimii optimaalisesti pitemmällä aikavälillä. [36.]

Pariohjelmointi

Pariohjelmoinnissa kaksi kehittäjää tekee yhdessä samaa tehtävää. Toinen toimii pääohjelmoijana ja toinen seuraa vierestä. Sen positiivisista vaikutuksista ohjelmiston laatuun on näyttöä ja sitä on tutkittu opiskelijaryhmissä. Pariohjelmointi lisää ohjelmiston laatua ja kehittäjien ymmärrystä ohjelmistosta. Kun ohjelmointi suunnitellaan ja tehdään yhdessä, työ tulee tehtyä huolellisemmin eikä koodiin kirjoiteta niin paljon virheitä kuin yksin ohjelmoitaessa. Etenkin projektin vaikeimmassa vaiheessa, suunnittelussa, pariohjelmoinnista saadaan eniten hyötyä. [37.]

Vaikka pariohjelmointi saattaa kuulostaa erittäin tehottomalta tavalta ohjelmoida ja kaksi ohjelmoijaa saa erikseen enemmän koodia aikaiseksi, pariohjelmoinnin tuloksena tuotettu koodi on moninkertaisesti laadultaan parempaa. Se mahdollistaa paremman koodin ylläpidettävyyden ja muokattavuuden, joilla voidaan saavuttaa säästöjä tulevaisuudessa.

Pariohjelmoinnilla on sosiaalisia vaikutuksia. Työntekijöiden keskittyminen on parempaa verrattuna yksin työskentelyyn, koska joku toinen on mukana seuraamassa työn etenemistä. Yksin työskentelevän on helpompi käydä Facebookissa tai Twitterissä ja häntä on helpompi häiritä kuin kahta tiiviisti työhön keskittyvää henkilöä.

Pariohjelmoinnissa ohjelmoijat jakavat osaamistaan ja näin kehittävät taitojaan. Pikanäppäinoikotiet sekä erilaiset uudet tavat tehdä asioita leviävät nopeasti henkilöstön sisällä, kun pareja vaihdetaan säännöllisesti. [38.]

Muita käytäntöjä

Koska ketterässä kehityksessä oletetaan, että vaatimukset voivat muuttua milloin tahansa, on koodi oltava erityisen ylläpidettävää. Refaktoroinnilla tehdään ohjelmakoodi helposti ymmärrettäväksi ja ylläpidettäväksi, vaikka työn tuloksena varsinainen toiminnallisuus ei muutukaan. [36.]

Ketterässä kehityksessä pyritään välttämään lopun testaa ja korjaa -vaihe testivetoisella kehityksellä. Se on suunnittelumenetelmä, jossa ensin luodaan testitapaus ja vasta sen jälkeen kehitetään ohjelmaa niin, että se läpäisee luodun testin. Tekniikan avulla pyritään välttymään virheiden kiertelyltä ja projektin venymiseltä. Samalla saadaan jatkuvasti kehittyvä testiverkosto, jonka varassa uusien toimintojen kehittäminen ja virheiden korjaaminen on huomattavasti turvallisempaa. [36.]

Yksikkötestauksella tarkoitetaan matalan tason testausta. Sillä pyritään varmistamaan koodin toimivuus metoditasolla. Jokaiselle funktiolle kirjoitetaan testitapaukset, joista funktion pitää suoriutua. Usein yksikkötestit ovat automaattinen osa jatkuvaa integraatiota ja yksikkötestauksen avulla voidaan varmistaa, että ohjelmiston osaan tehty muutos ei aiheuta ongelmia muissa ohjelmiston osissa. [36.]

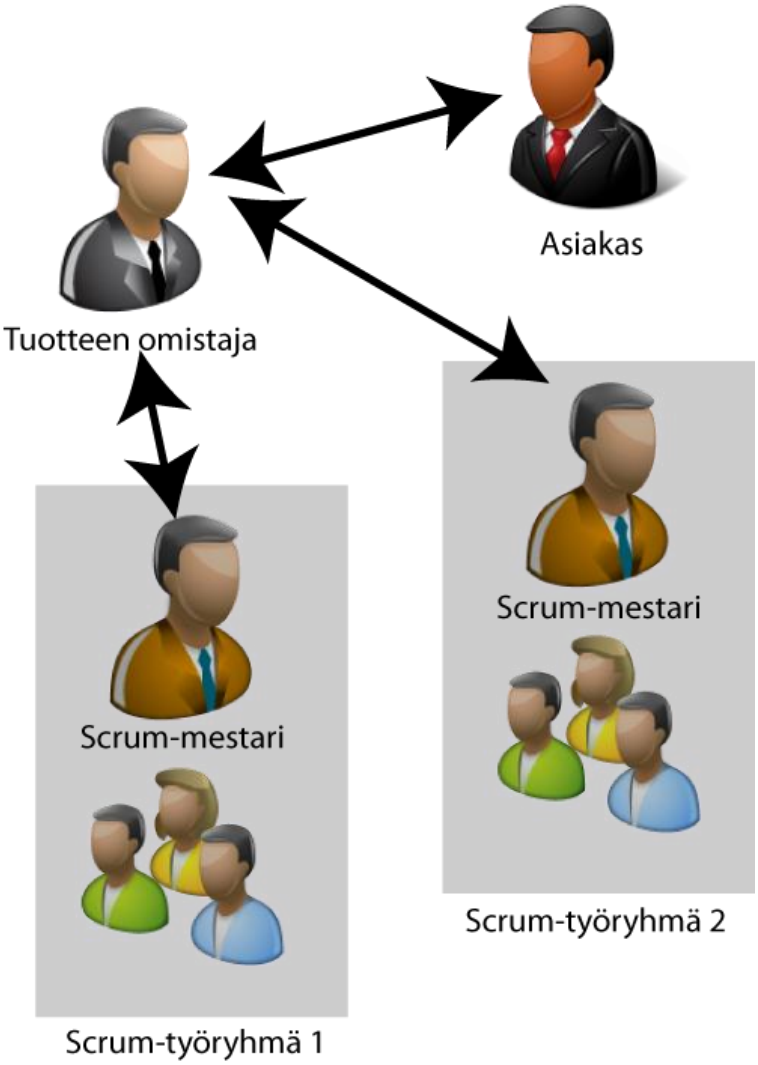
4.6 Scrum

Scrum on yksi käytetyimmistä ketterän kehityksen malleista, joka tarjoaa prosessin ohjelmistokehitysprojektin ohjaukseen. Scrum:ssa ei oteta kantaa matalan tason insinöörikäytäntöihin, vaan siinä keskitytään pääasiassa projektin vaiheistamiseen ja jatkuvaan kontrolliin projektin etenemisessä.

Muiden ketterien ohjelmistokehitysmallien tavoin Scrum:ssa ohjelmistokehitys jaetaan erimittaisiin sykleihin. Sykleistä tärkein on sprintti. Sprintillä tarkoitetaan yhtä kehitysjaksoa, jonka jälkeen tuote on ainakin periaatteessa valmis julkaisua varten. Tyypillinen kesto sprintille on 30 päivää, mutta luonnollisesti pituus voi vaihdella organisaation tarpeiden mukaan. Kehitysjakson päätteeksi järjestetään sprinttikatselmus, jossa kehitystiimi esittelee sprintin konkreettiset saavutukset tuoteomistajille sekä mahdollisille sidosryhmien edustajille palautteen saamiseksi ja ymmärryksen lisäämiseksi. Ennen seuraavan sprintin käynnistämistä pidetään vielä sprintin jälkitarkastelu. [39.]

4.6.1 Scrum:n roolit

Vesiputousmallin mukaisissa projekteissa on usein määrittelijä, suunnittelija, ohjelmoija, testaaja ja projektipäällikkö. Heitä voi olla useita henkilöitä projektipäällikköä lukuun ottamatta. Projekteissa, joissa Scrum on käytössä, on vain kolme eri roolia: Tuotteen omistaja, Scrum-mestari sekä Scrum-työryhmä. Scrum:n roolit on kuvattu kuvassa 4.7.



Kuva 4.7. *Scrum:ssa käytetyt työntekijöiden roolit ja niiden suhteet toisiinsa.*

Scrum:ssa käytetään yhtä tai useampaa Scrum-työryhmää. Työryhmä koostuu tuoteomistajasta, Scrum-mestarista ja kehitystyöryhmästä. Scrum-työryhmä päättää sprintin tavoitteet ja tehtävät sekä vastaa yhdessä tavoitteisiin pääsemisestä. Työryhmä on itseohjautuva ja monitaitoinen ja se päättää itse omista työmenetelmistään.

Tuoteomistaja on henkilö, joka viime kädessä vastaa tuotteen ominaisuuksista. Ohjelmistokehitysprosesseissa tuoteomistaja on tavallisesti tuotepäällikkö ja asiakasprojekteissa asiakkaan edustaja. Tuotteen omistaja tekee kaikki päätökset tuotteen ominaisuuksista ja toiminnallisuuksiin vaikuttavista tekijöistä.

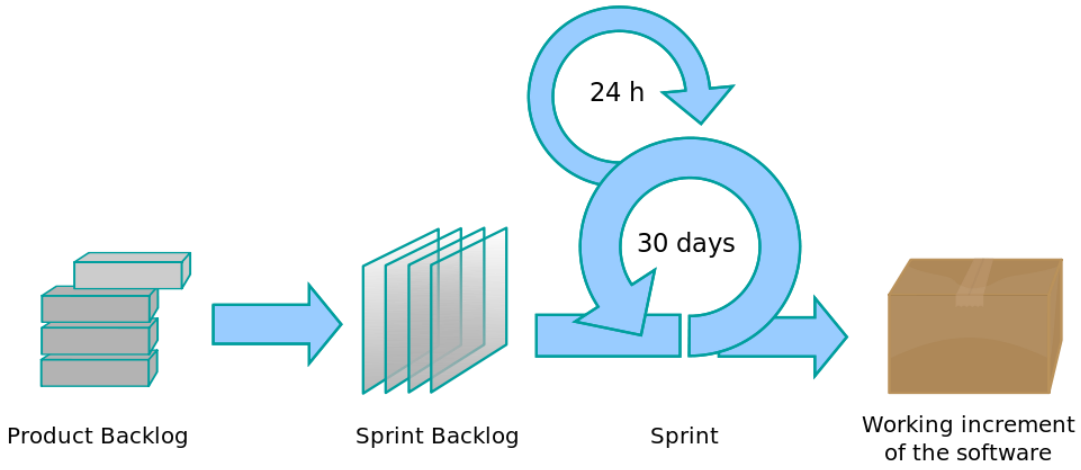
Scrum-mestari huolehtii siitä, että työryhmä voi tehdä työtään optimaalisella tavalla. Työryhmän työntekijät raportoivat mestarille päivittäin ongelmista, jotka hidastavat töiden sujumista. Mestarin tehtävä on ratkoa raportoidut ongelmat. Lisäksi mestari johtaa päivittäiset päiväpalaverit ja vastaa siitä, että Scrum:a noudatetaan oikein.

Työryhmään kuuluvat kaikki projektin kanssa tekemisissä olevat henkilöt. Työryhmän sisällä kaikki tekevät kaikkensa projektin edistämiseksi ja olennaista on, että työ-

ryhmä vastaa itse yhteisöllisesti tehtävien jaosta. Silloin ei ajauduta työtehtävien siirtelyyn henkilöltä toiselle. [39.]

4.6.2 Scrum:n tuotokset

Kuva 4.8. esittää Scrum-prosessia. Tuotteen kehitysjono (engl. Product Backlog) on järjestetty lista kaikista tuotteeseen toteutettavista vaatimuksista ja muutoksista. Kehitysjono uudistuu jatkuvasti tuotteen ja ympäristön kehittyessä. Kehitysjonosta huolehtiminen kuuluu tuoteomistajalle.



Kuva 4.8. *Scrum-prosessi. [40]*

Sprintin tehtävälista (engl. Sprint Backlog) muodostuu sprintin suunnittelupalaverissa valituista tuotteen kehitysjonon kohdista ja niiden toteuttamissuunnitelmasta. Tehtävälista toimii kehitystiimin ennusteena seuraavan version toiminnallisuuksista sekä tarvittavista tehtävistä niiden toteuttamiseksi. Sprintin aikana tuoteomistaja ei voi vaihtaa jo sprinttiin valittuja tuotteen kehitysjonon kohtia toisiin, mutta kehitystiimi voi halutessaan muuttaa, lisätä tai poistaa sprintin tehtävälistan tehtäviä varmistaakseen, että sprintin tavoite täyttyy.

Tuoteversio (engl. Increment) on kaikkien aikaisempien ja nykyisen sprintin yhdistelmä, jossa on kaikki tuotteen kehitysjonon jo toteutetut kohdat. Sprintin lopussa valmistuneen tuoteversion tulee täyttää Scrum-työryhmän määritelmät ja olla käyttökelpoisessa kunnossa riippumatta siitä, julkaiseeko tuoteomistaja sen. [41, s. 219.]

4.6.3 Scrum:n tapahtumat

Scrum:ssa tapahtumat ovat määritelty etukäteen. Niillä pyritään luomaan säännöllisyyttä työskentelyyn sekä eliminoimaan muiden palaverien tarpeellisuus. Kaikilla Scrum:n tapahtumilla on enimmäisaika. Tapahtuminen rajoittaminen ajallisesti mahdollistaa riit-

tävän ajan varaamisen suunnittelulle ilman, että suunnittelussa tehdään tarpeetonta työtä.

Tuotteen kehitysjonon muodostaminen

Ensimmäinen tapahtuma on tuotteen kehitysjonon muodostaminen. Kehitysjonoon kirjataan kaikki valmiiseen tuotteeseen tilatut ominaisuudet sekä työmääräarviot. Tuotteen omistaja priorisoi kehitysjonon, mutta kehitystyöryhmä antaa lopullisen työmääräarvion jokaisesta listan asiasta. Kehitysjonon muodostaminen on toistuva prosessi, jossa tuoteomistaja ja Scrum-työryhmä lisäävät, päivittävät ja arvioivat yhteistyössä kehitysjonoa projektin edetessä. [39.]

Sprintin suunnittelu

Tuotteen työlistan muodostamisen jälkeen pidetään sprintin suunnittelupalaveri. Suunnittelupalaverissa suunnitellaan sprintin toteutettavat toiminnallisuudet. Suunnittelupalaveriin osallistuu koko Scrum-työryhmä ja palaverin pituus riippuu sprintin pituudesta. [39.]

Sprintti

Sprintti on Scrum:n tärkein tapahtuma. 1-4 viikon mittaisen sprintin aikana toteutetaan siihen kuuluviksi valitut toiminnallisuudet. Toteutettujen toiminnallisuuksien tulee olla käyttökelpoisia ja mahdollisesti myös julkaisukelpoisia. Uusi sprintti käynnistyy välittömästi edellisen päätyttyä ja tätä jatketaan, kunnes kaikki halutut toiminnallisuudet ovat kehitetty. Sprintin aikana vaatimusten muuttamista ei sallita ja Scrum-työryhmällä on vapaus tehdä tarpeelliseksi katsottuja toimenpiteitä sovitun päämäärän saavuttamiseksi. Scrum-työryhmällä on vapaus organisoida itsensä parhaalla katsomallaan tavalla. [39.]

Päiväpalaveri

Päivittäin sprintin aikana pidetään lyhyt päiväpalaveri. Sen tarkoitus on antaa Scrum-työryhmälle tilaisuus synkronoida keskinäiset työt ja suunnitella seuraavan päivän työtehtävät. Tarkastelemalla edellisen päivän työtä ja ennustamalla seuraavaa päivää saadaan selville missä työvaiheessa kukin on. Päiväpalaverissa Scrum-työryhmän jäsenet kertovat mitä he ovat tehneet viime palaverin jälkeen, mitä he aikovat tehdä ennen seuraavaa päiväpalaveria ja onko työn etenemisellä mitään esteitä. Päiväpalaverin tarkoitus ei ole raportointi, vaan työpäivän työpanoksen optimointi sprintin tavoitteiden saavuttamiseksi. [39.]

Sprintin katselmointi

Sprintin lopussa pidetään sprinttikatselmus. Katselmuksessa tarkastellaan kehitettyä tuoteversiota ja mahdollisesti muutetaan kehitysjonoa. Scrum-työryhmä esittelee tuotteen omistajalle, mitä sprintissä saatiin kehitettyä ja saa palautetta tuotedemosta. Kokemusten ja tulosten pohjalta palaveriin osallistujat keskustelevat seuraavista kehitystöistä ja luovat realistisen pohjan seuraavan sprintin suunnittelupalaverille. [39.]

Sprintin jälkitarkastelu

Sprintin jälkeen pidetään sprintin jälkitarkastelu. Siinä Scrum-työryhmä, Scrum-mestari sekä tuoteomistaja arvioivat päättyneen sprintin. Jälkitarkastelussa jokainen jäsen kertoo, mikä sprintissä meni hyvin ja missä olisi parannettavaa. Lopuksi osallistujat priorisoivat mitkä muutokset pyritään toteuttamaan seuraavan sprintin aikana. Jälkitarkastelun jälkeen siirrytään takaisin sprintin suunnitteluun. [39.]

4.6.4 Scrum:n haasteet

Scrum ei takaa ohjelmistokehityksen onnistumista, vaan Scrum:n käyttöönotto saattaa epäonnistua. Käyttöönoton epäonnistumiseen on useita syitä, joita on havaittu tutkimalla epäonnistuneita ketterin menetelmien ohjelmistoprojekteja.

Ketterien menetelmien nopea ja laaja käyttöönotto saattaa keskeyttää tai ainakin hidastaa ohjelmistokehitystä. Käyttöönotossa työntekijät joutuvat opettelemaan menetelmän uudet toimintatavat, eikä niiden sisäistäminen välttämättä onnistu lyhyessä ajassa. Kun menetelmän perusteita ei hallita, käyttöönotosta aiheutuu sekaannuksia ja virheellisiä tulkintoja menetelmien käytännöistä. Käyttöönotto tulisi tehdä menetelmän perusteiden sisäistämisen jälkeen selkeinä vaiheina porrastetusti.

Vesiputousmallista opitut käytännöt saattavat vaikuttaa ketterien menetelmien käytössä. Uusissa projekteissa saatetaan yrittää määritellä vaatimusmäärittelyt totuttuun tapaan hyvin yksityiskohtaisesti ennen toteutusvaihetta. Silloin ei ole ymmärretty menetelmän rohkaisevan vaatimusmuutoksien hyväksymiseen ja niihin reagoimiseen.

Sprintit saatetaan määritellä liian pitkiksi, jolloin ajankäytön tehokkuus huononee. Se myös yleensä tarkoittaa sitä, että iteraation työtehtävien työmääräarviointi ei ole onnistunut. Iteraatiolle saatetaan myös suunnitella liian suuret tavoitteet, jolloin se alkaa muistuttamaan perinteistä vesiputousmallia.

Laadunhallinta ja testaus jäävät helposti liian pienelle huomiolle ketterässä kehityksessä, koska ne eivät varsinaisesti tuota mitään uutta toiminnallisuutta ohjelmistoon. Jos ne jätetään ainoastaan viimeiselle iteraatiokierrokselle, alkaa ohjelmistokehitys muistuttamaan vesiputousmallia. Se muodostaa usein ohjelmistoprojektin loppuun työlään, kalliin ja aikaa vievän testaa ja korjaa -vaiheen.

Sprintin jälkitarkastelu auttaa työryhmää parantamaan toimintatapojaan. Sen väliin jättäminen saattaa aiheuttaa ohjelmistokehityksen epäonnistumisen, jos alussa vääriä toimintatapoja ei saada korjattua projektin aikana. [42.]

5 NYKYTILANNEANALYYSI

Tässä luvussa tutkitaan, miten Huurre HOT:ssa on toteutettu ohjelmistokehitystä, mitä kehitettävää siinä on ja miten luvussa 4 esitellyt ohjelmistokehitysmallit ja ohjelmisto-käytännöt soveltuvat Huurre HOT:n tilanteeseen.

6 TULOKSET JA NIIDEN TARKASTELU

Tässä luvussa määritellään ohjelmistoprosessi Huurre HOT:lle perustuen luvussa 5 tehtyihin havaintoihin. Määrittelyä varten prosessista piirretään kuva ja sen kaikki kohdat käydään läpi.

7 YHTEENVETO

Huurre Group Oy:n liiketoimintayksikkö Huurre HOT on ylläpitänyt ja kehittänyt tarjoamaansa palvelua jo muutaman vuoden ajan, joiden aikana palvelun asiakasmäärä on kasvanut merkittävästi. Kasvun seurauksena järjestelmän teknisen alustan on pystyttävä käsittelemään suurempi määrä tietoa ja järjestelmän hallittavuus on vaikeampaa kuin aikaisemmin. Lisäksi palvelun käyttäjämäärän kasvu on lisännyt palautetta sisäisiltä ja ulkoisilta asiakkailta. Niin tekniset vaatimukset kuin asiakkaiden palaute ovat lisänneet kehityspyyntöjen määrää huomattavasti.

Huurre HOT:n strategisena tavoitteena on pitkään ollut ohjelmistokehityksen vankentaminen, jonka yhtenä osa-alueena on ollut oman ohjelmistokehityksen tutkiminen. Tutkinnan tuloksena on huomattu, että henkilöstön määrä sekä osaamisen taso ovat kehittyneet tasolle, joka mahdollistaa teknisen alustan ohjelmistokehittämisen myös yhtiön sisällä kehityspartnereiden sijaan. Oman ohjelmistokehityksen käynnistämiseksi ja hallitsemiseksi ohjelmistokehitysprosessin määrittely oli ajankohtaista.

Ohjelmistokehitysmalleista ketterän kehityksen Scrum oli selvästi sopivin malli Huurre HOT:n ohjelmistokehitykseen. Scrum:iin pohjautuvaan ohjelmistokehitysprosessiin päädyttiin analysoimalla käytössä olleita toimintatapoja, tuomalla niissä havaittuja ongelmia esiin sekä tutkimalla eri mallien sopivuutta Huurre HOT:n tilanteeseen.

Määritelty ohjelmistokehitysprosessi tullaan ottamaan Huurre HOT:n kehityksen käyttöön tulevissa projekteissa. Tällä työllä saatiin myös lisättyä Huurre HOT organisaation tietoa ohjelmistokehityksestä.

LÄHTEET

[1] Finlex. Elintarvikelaki. [WWW]. [viitattu 5.5.2013]. Saatavissa: http://www.finlex.fi/fi/laki/ajantasa/2006/20060023.

[2] Eckelmann. E*LDS CI 3000 / CI 3100 Store Computer. [WWW]. [viitattu 5.5.2013]. Saatavissa: http://www.eckelmann.de/fileadmin/user_upload/ downloads/Kaeltetechnik/EN/E_LDS_CI3000_CI3100_EN.pdf.

[3] Evira. HACCP. [viitattu 14.5.2013]. Saatavissa: http://www.evira.fi/portal/ fi/tietoa+evirasta/asiakokonaisuudet/omavalvonta/haccp/.

[4] Danfoss. ADAP-KOOL – Next Generation ENERGIAOPTIMOINTI. [WWW]. [viitattu 5.5.2013]. Saatavissa: http://www.combicool.fi/assets/files/ ohjeet/danfoss/AK-NG/Energiaoptimointi.pdf.

[5] RefGroup Oy. Energian säästö ja lämpökertoimet. [WWW]. [viitattu 5.5.2013]. Saatavissa: http://www.ilmalampopumput.fi/fi/mika-ihmeen-lampopumppu/ energian-saasto.

[6] Huurre Group. Huurre HOT. [viitattu 14.5.2013]. Saatavissa: http://www.huurrehot.com/fi.

[7] IWMAC. [WWW]. [viitattu 5.5.2013]. Saatavissa: http://www.iwmac.eu.

[8] Carrier Refrigeration. Remote service. [WWW]. [viitattu 5.5.2013]. Saatavissa: http://www.carrier-refrigeration.com/Remote-service.470.0.html.

[9] Carrier Refrigeration. e Service. [WWW]. [viitattu 5.5.2013]. Saatavissa: http://www.carrierrefrigerationservice.co.uk/acatalog/e_service.pdf.

[10] Sensire. TempNet. [WWW]. [viitattu 5.5.2013]. Saatavissa: http://www.sensire.fi/.

[11] Resource Data Management. Remote Monitoring. [WWW]. [viitattu 5.5.2013]. Saatavissa: http://www.resourcedm.com/RemoteMonitoring.aspx.

[12] Resource Data Management. Remote Monitoring. [WWW]. [viitattu 5.5.2013]. Saatavissa: http://www.woodley.se/wordpress/category/produkter/temperaturovervakning.

[13] Tridium Inc. Niagara^{AX}. [WWW]. [viitattu 5.5.2013]. Saatavissa: http://www.tridium.com/cs/products_/_services/niagaraax.

[14] Tridium Inc. Niagara^{AX}. [WWW]. [viitattu 5.5.2013]. Saatavissa: http://www.niagaraax.com/cs/products_/_services/niagara_framework_architecture_diagram.

[15] Tridium Inc. JACE. [WWW]. [viitattu 5.5.2013]. Saatavissa: http://www.tridium.com/cs/products_/_services/jace.

[16] Haikala, I., Märijärvi, J. 2006. Ohjelmistotuotanto. 11. painos. Helsinki. Talentum. 440 p.

[17] Ala-Mutka, K., Rintala, M., Savikko, V., Palviainen, J. Ohjelmiston elinkaari. Tietotekniikan peruskurssi. [WWW]. [viitattu 5.5.2013]. Saatavissa: http://www.cs.tut.fi/etaopetus/titepk/luku21/elinkaari.html.

[18] Taina J. 2009. Suunnittelmakeskeiset prosessit. Ohjelmistoprosessit ja ohjelmistojen laatu. [WWW]. [viitattu 5.5.2013]. Saatavissa: http://www.cs.helsinki.fi/u/taina/opol/k-2009/pdf/luku-6_2.pdf

[19] The Incredible Rate of Diminishing Returns of Fixing Software Bugs. [WWW]. [viitattu 5.5.2013]. Saatavissa: http://superwebdeveloper.com/2009/11/25/the-incredible-rate-of-diminishing-returns-of-fixing-software-bugs/

[20] What is Waterfall model- advantages, disadvantages and when to use it?. [WWW]. [viitattu 5.5.2013]. Saatavissa: http://istqbexamcertification.com/what-is-waterfall-model-advantages-disadvantages-and-when-to-use-it.

[21] What is Spiral model- advantages, disadvantages and when to use it?. [WWW]. [viitattu 5.5.2013]. Saatavissa: http://istqbexamcertification.com/what-is-spiral-model-advantages-disadvantages-and-when-to-use-it.

[22] Web-käytettävyys: 2.4. Suunnittelu. [WWW]. [viitattu 5.5.2013]. Saatavissa: http://www.uiah.fi/mediastudio/survey4/24.html.

[23] NASA. 2004. NASA Software Safety Guidebook. NASA-GB-8719. [WWW]. [viitattu 5.5.2013]. Saatavissa: http://www.hq.nasa.gov/office/codeq/doctree/871913.pdf.

[24] Ohjelmistotuotannon mallit. [WWW]. [viitattu 5.5.2013]. Saatavissa: http://hlab.ee.tut.fi/hmopetus/vpsist-oppimateriaali/4-menetelmia-ja-malleja/4-3-suunnittelumalleja/4-3-1-ohjelmistotuotannon-malli.

[25] Spiral Model: Advantages and Disadvantages. [WWW]. [viitattu 5.5.2013]. Saatavissa: http://www.ianswer4u.com/2011/12/spiral-model-advantages-and.html#axzz2SRiZaRaE.

[26] Spiral Model Advantages and Disadvantages. [WWW]. [viitattu 5.5.2013]. Saatavissa: http://www.buzzle.com/articles/spiral-model-advantages-and-disadvantages.html.

[27] System Development Concept (Part I). [WWW]. [viitattu 5.5.2013]. Saatavissa: http://dewaclass09.blogspot.fi/2009/03/system-development-concept-part-i.html.

[28] Software Development. [WWW]. [viitattu 5.5.2013]. Saatavissa: http://www.estenda.com/servicesSoftwareDevelopment.shtml.

[29] Cockburn, A. Using Both Incremental and Iterative Development. CrossTalk, The Journal of Defense Software Engineering. [WWW]. [viitattu 5.5.2013]. Saatavissa: http://www.crosstalkonline.org/storage/issue-archives/2008/200805/200805-Cockburn.pdf.

[30] Kendall, S. 2001. Overview of the Unified Process. [WWW]. [viitattu 5.5.2013]. Saatavissa: http://www.informit.com/articles/article.aspx?p=24671&seqNum=6shtml.

[31] Advantage and disadvantage of iterative development?. [WWW]. [viitattu 5.5.2013]. Saatavissa: http://wiki.answers.com/Q/Advantage_and_disadvantage_of_iterative_development.

[32] Sami, M. Software Development Life Cycle Models and Methodologies. [WWW]. [viitattu 5.5.2013]. Saatavissa: http://melsatar.wordpress.com/2012/03/15/software-development-life-cycle-models-and-methodologies.

[33] VersionOne. Agile Development. [WWW]. [viitattu 5.5.2013]. Saatavissa: http://www.versionone.com/pdf/agileposter.pdf.

[34] Agile Manifesto. [WWW]. [viitattu 5.1.2013]. Saatavissa: http://agilemanifesto.org/iso/fi.

[35] Poimala, S., Tolvanen, P. Ketteryys haltuun. Osa 1: Ketterän kehityksen yleiset periaatteet. [WWW]. [viitattu 5.5.2013]. Saatavissa: http://www.meteoriitti.com/fi-FI/tiedotteet/ajankohtaista/ketteryys-haltuun-ketteran-kehityksen-yleiset-periaatteet.

[36] Poimala, S., Tolvanen, P. Ketteryys haltuun. Osa 2: Yleisimmät ketterät käytännöt. [WWW]. [viitattu 5.5.2013]. Saatavissa: http://www.meteoriitti.com/fi-FI/tiedotteet/ajankohtaista/ketteryys-haltuun-ketteran-kehityksen-yleiset-periaatteet.

[37] Pariohjelmointi parantaa ohjelmistojen laatua teollisuusyrityksissä. [WWW]. [viitattu 5.5.2013]. Saatavissa: http://www.aalto.fi/fi/current/news/view/2011-12-19-003/.

[38] Vanhanen, J. Empirical Assessment of the Adoption, Use, and Effects of Pair Programming. Doctoral dissertation. Espoo 2011. Aalto University Schools of Technology.

[39] Poimala, S., Tolvanen, P. Ketteryys haltuun. Osa 3: Scrum pähkinänkuoressa. [WWW]. [viitattu 5.5.2013]. Saatavissa: http://www.meteoriitti.com/fi-FI/tiedotteet/ajankohtaista/ketteryys-haltuun-scrum-pahkinankuoressa.

[40] Lakeworks. Scrum process. [viitattu 5.5.2013]. Saatavissa: http://commons.wikimedia.org/wiki/File:Scrum_process.svg.

[41] Williams, L. 2007. A Survey of Agile Development Methodologies. [viitattu 5.5.2013]. Saatavissa: http://agile.csc.ncsu.edu/SEMaterials/AgileMethods.pdf.

[42] Markkula, L. Ketterien ohjelmistokehitysmenetelmien soveltaminen ja etujen arvioiminen eräässä projektissa. Diplomityö. Tampere 2009. Tampereen teknillinen yliopisto. 51 p.