KETTERÄ VAATIMUSMÄÄRITTELYPROSESSI

HAMK
HÄMEEN AMMATTIKORKEAKOULU
HÄME UNIVERSITY OF APPLIED SCIENCES

Ammattikorkeakoulututkinnon opinnäytetyö

Tietojenkäsittelyn koulutus

Kevät 2022

Päivi Paakkari

HAMK
HÄMEEN AMMATTIKORKEAKOULU
HÄME UNIVERSITY OF APPLIED SCIENCES

Tietojenkäsittelyn koulutus Tiivistelmä
Tekijä Päivi Paakkari Vuosi 2022
Työn nimi Ketterä vaatimusmäärittelyprosessi
Ohjaaja Lasse Seppänen

TIIVISTELMÄ

Opinnäytetyön tavoitteena oli tutkia, millainen on ketterän vaatimusmäärittelyn prosessi, millainen on ketterän vaatimusmäärittelyn rooli ohjelmistokehityksen eri vaiheissa sekä miten ketterää vaatimusmäärittelyä voi ohjeistaa. Opinnäytetyön toimeksiantaja oli Tietoevry Health.

Opinnäytetyön tietopohjassa tarkastellaan ketteriä ohjelmistoprojektimalleja, prosessien mallintamista ja vaatimusmäärittelyn teoriaa. Tietopohjassa kuvataan myös esimerkkimäärittelyyn liittyviä viranomaisvaatimuksia, kestävän kehityksen vaatimuksia sekä terveysalan ohjelmistoja koskevia ohjelmistostandardeja. Opinnäytetyö on toiminnallinen. Opinnäytetyön käytännön osassa toteutettiin ohjeistus ketterälle vaatimusmäärittelylle ja kaksi esimerkkimäärittelyä. Toimeksiantajan asiantuntijat arvioivat ketterän määrittelyprosessin ohjeistuksen soveltuvuuden tarkoitukseensa.

Opinnäytetyön tietopohjan ja käytännön työn pohjalta voidaan todeta, että ketterällä vaatimusmäärittelyllä on oma, tärkeä roolinsa koko ohjelmistokehityksen ajan. Sen vuoksi määrittelyn ja suunnittelun roolia ketterässä ohjelmistokehityksessä tulee ohjeistaa. Opinnäytetyönä laadittu ohjeistus sai hyvät arvioinnit ja yritys on tyytyväinen työn tuloksiin.

Avainsanat Ketterä ohjelmistokehitys, Vaatimusmäärittely, Suunnittelu, Prosessi
Sivut 63 sivua ja liitteitä 25 sivua

HAMK
HÄMEEN AMMATTIKORKEAKOULU
HÄME UNIVERSITY OF APPLIED SCIENCES

Degree Programme in Business Information Technology
Author Päivi Paakkari
Subject The Agile Definition Process
Supervisors Lasse Seppänen

Abstract
Year 2022

ABSTRACT

The purpose of the thesis was to study the process of agile requirements definition, the role of agile requirements definition in different phases of software development, and how agile requirements definition can be instructed. The thesis was commissioned by Tietoevry Health.

The theoretical part of the thesis observers agile software project models, process modeling and requirements theory. The theoretical part also describes regulatory requirements related to example definition, sustainability requirements, and software standards for health care software. The thesis is functional. In the practical part of the thesis, guidelines for agile requirements definition and two example definitions were implemented. The client's specialists evaluated the suitability of the guidelines for the agile definition process for its purpose.

Based on the theoretical part and practical work of the thesis, it can be stated that agile requirements definition has its own important role throughout software development. Therefore, the role of definition and design in agile software development should be guided. The guidelines prepared as a thesis received good evaluations and the company is satisfied with the results of the work.

Keywords Agile Software Development, Requirements Specification, Design, Process

Pages 63 pages and appendices 25 pages

Sanasto

Agile	Ketterä, esimerkiksi ketterä ohjelmistokehitys, agile software development.
BPMN	Business Process Model and Notation, liiketoimintaprosessien mallinnustapa.
Bug	Virhe, virheenkorjauksen tehtävä kehitysjonolla.
Backlog	Kehitysjono, joka voi koostua useista saman- tai erityyppisistä työtehtävistä.
Change request	Muutospyyntö sovellukseen tai ohjelmistoon. Yksi työtehtävätyyppi kehitysjonolla.
Epic	Kertomus. Yksi työtehtävätyyppi kehitysjonolla.
Feature	Ominaisuus, toiminnallisuus, piirre. Yksi työtehtävätyyppi kehitysjonolla.
ICT	Information and Communication Technology, tieto- ja viestintätekniikka tai lyhyemmin tietotekniikka.
Iteraatio	Sykli, toisto.
Kanban	Kanban-taululla visualisoidaan työn etenemistä tehtäväkorteilla, jotka sijoitetaan esimerkiksi sarakkeisiin Tekemättä, Työn alla ja Valmis sen mukaan, missä vaiheessa tehtävän toteutus on.
Lokalisoida	Kääntää ja sovittaa tuote paikalliseen käyttöön.
Notaatio	Tietojärjestelmän mallinnuskielessä käytettävä merkintätapa.
OMG	Object Management Group, kansainvälinen teknologiastandardien konsortio.
STM	Sosiaali- ja terveysministeriö.
Story	Tarina. Yksi työtehtävätyyppi kehitysjonolla.
THL	Terveyden ja hyvinvoinnin laitos.
UML	Unified Modelling Language, ohjelmistoprosessien mallinnustapa.
Valvira	Sosiaali- ja terveysalan lupa- ja valvontavirasto.
Work item	Työtehtävä kehitysjonolla, useita tyyppejä, esimerkiksi feature, story.

Sisällys

Kuvat

Liitteet

Liite 1: Aineistonhallintasuunnitelma

Liite 2: Ketterä määrittelyprosessi, tekstimuotoinen aineisto

Liite 3: Ketterä määrittelyprosessi, diasarja

Liite 4: Palautekyselylomake: Ketterä määrittelyprosessi -aineisto

1 Johdanto

Asiakkaan vaatimusten kerääminen, analysointi ja dokumentointi vaatimuksiksi eli vaatimusmäärittely on tietojärjestelmän kehittämisen perusta. Vaatimukset kuvaavat miten käyttäjän on tarkoitus käyttää järjestelmää, mitä hyötyä järjestelmästä on ja mitä ominaisuuksia järjestelmässä tulee olla. Määrittelyn lähtökohta on asiakasorganisaation ja käyttäjien tarpeen tai ongelman ymmärtäminen. Määrittelyn pohjalta tehdään tarkemmat suunnitelmat mm. arkkitehtuurista ja käyttöliittymistä.

Ketterissä tuotekehitysmenetelmissä määrittelykin tehdään ketterästi, ensin yleisellä tasolla ja siitä kehitystyön edetessä vaiheittain tarkentaen. Määrittelyssä huomioidaan myös järjestelmää koskeva lainsäädäntö ja viranomaisohjeistus. Vaatimusmäärittelyn lopputuloksena on dokumentaatio, jota hyödynnetään ja tarkennetaan kehitysvaiheen lisäksi sovelluksen verifioinnissa ja validoinnissa. Määrittely tukee myös käyttö- ja asennusohjeiden laatimista.

Tämän opinnäytetyön on tilannut työnantajani Tietoevry Oyj Health. Työn tarkoituksena on kuvata määrittelyn prosessi niin, että tilaaja voi käyttää sitä määrittelytyötä tekevien työntekijöiden perehdytyksessä. Opinnäytetyö on toiminnallinen ja sen ohessa on toteutettu erikoissairaanhoidon hoitoonpääsyn seurantaa koskeva vaatimusmäärittely, jonka valmistumisprosessi kuvataan työn toteutusosassa. Itse vaatimusmäärittelyn sisältö on rajattu työn ulkopuolelle.

Tutkimuskysymykset:

- Millainen on ketterän vaatimusmäärittelyn prosessi?
- Millainen on ketterän vaatimusmäärittelyn rooli ohjelmistokehityksen eri vaiheissa?
- Miten ketterää vaatimusmäärittelyä voi ohjeistaa?

2 Kestävä kehitys ICT-alalla

Kestävä kehitys yritystoiminnassa lähtee liiketoiminnan arvoihin ja periaatteisiin perustuvasta lähestymistavasta. Yhdistyneet Kansakunnat on määritellyt tämän lähestymistavan toiminnaksi, joka täyttää vähintään perusvalvollisuudet ihmisoikeuksien, työvoiman, ympäristön ja korruption vastaisuuden suhteen kaikkialla missä yritys toimii. (United Nations Global Compact, n.d.)

Lähestymistapa on koottu kymmeneksi periaatteeksi, joista kolme koskee ympäristöä ja seitsemän ihmisoikeuksia, työvoimaa ja korruption torjuntaa. (United Nations Global Compact, n.d.)

Kolme ympäristöperiaatetta ovat:

- Yrityksien on tuettava ennalta varautuvaa lähestymistapaa ympäristöhaasteisiin
- Yritysten on toteutettava aloitteita ympäristövastuun lisäämiseksi
- Yritysten on rohkaistava ympäristöystävällisten teknologioiden kehittämistä ja levittämistä. (United Nations Global Compact, n.d.)

Ympäristöperiaatteet kytkeytyvät läheisesti kymmenessä periaatteessa ilmaistuihin ihmisoikeuksien, työvoiman ja korruption torjunnan periaatteisiin. Kun yritykset huolehtivat perusvelvollisuuksistaan ihmisiä ja planeettaa kohtaan näiden periaatteiden avulla, ne myös luovat pohjaa pitkän aikavälin menestykselle. (United Nations Global Compact, n.d.)

Suomen Liikenne- ja viestintäministeriö julkaisi vuonna 2021 kansallisen ICT-alan ilmasto- ja ympäristöstrategian, joka pyrkii edistämään ekologisesti kestävää digitalisaatiota ja samalla tukemaan kansallisten ilmasto- ja ympäristötavoitteiden saavuttamista. (LVM, 2021) Strategian tavoitteet on jaettu kuuteen osaan: ICT-infrastruktuurin ilmasto- ja ympäristöystävällisyyteen, datatalouden ilmasto- ja ympäristöystävällisyyteen, kestäviin materiaalivirtoihin ja kiertotalouteen, tietopohjan laajentamiseen ja mittaamisen kehittämiseen, kuluttajien tietoisuuden ja osaamisen lisäämiseen sekä nousevien teknologioiden hyödyntämiseeen ja haasteisiin vastaamiseen. (LVM, 2021)

ICT-infrastruktuurin ilmasto- ja ympäristöystävällisyyden toimenpiteistä nousevat erityisesti esille energiatehokkuus ja datakeskuksien hukkalämmön hyödyntäminen. Datatalouden ilmasto- ja ympäristöystävällisyyden toimenpiteissä korostuvat energianäkökohtien huomioiminen ohjelmistojen ja palvelujen suunnittelussa ja hankinnassa. Kestävissä materiaalivirroissa ja kiertotaloudessa keskeisiä toimenpiteitä ovat laitteiden käyttöiän pidentäminen ja kierrätyksen tehostaminen. Tietopohjan laajentamisen ja mittaamisen kehittämisen toimenpiteet tarkoittavat ICT-alan ilmasto- ja ympäristövaikutuksia koskevan datan tekemistä läpinäkyväksi ja yhteismitalliseksi esimerkiksi energiankulutuksen ja materiaalivirtojen osalta. Kuluttajien tietoisuuden ja osaamisen lisäämistä toteutetaan mm. laitteiden ja palvelujen energiatehokkaan käytön opetuksella. Nousevien teknologioiden kuten lohkoketjujen ja kvanttiteknologian hyödyntäminen ja haasteisiin vastaaminen tarkoittaa ymmärryksen lisäämistä niiden ympäristövaikutuksista ja niiden ekologisesti kestävää hyödyntämistä. (LVM, 2021)

Yrityksiltä odotetaan kestävän kehityksen huomiointia liiketoiminnassaan. Sidosryhmät odottavat yrityksiltä ympäristöllisesti, sosiaalisesti ja taloudellisesti vastuullista toimintaa. Lisäksi yrityksen tulee tuottaa ratkaisuja, jotka edistävät sen asiakkaiden ja koko yhteiskunnan kestävän kehityksen tavoitteiden saavuttamista. (Tietoevry, s 4)

Työnantajani Tietoevryn *Sustainability Report 2021*:ssä eli Kestävän kehityksen raportissa 2021 kuvataan Sustainability dashboard -osassa, kuinka hiilettömän energian osuus yhtiön datakeskuksissa ja toimistoissa on kasvanut viime vuosina 92 %:iin ja tavoitteena on 100 % vuonna 2023. Työasemien kierrätysaste on 70-86 % ja kierrätyksen tavoite on 100 % vuonna 2023. Raportissa kuvataan myös muiden kestävän kehityksen vaatimusten täyttäminen. (Tietoevry, s 8) Lisäksi kaikki toimistot ja datakeskukset ovat ottaneet käyttöön ISO 14001-sertifioidun ympäristöhallintajärjestelmän (Environmental Management System). (Tietoevry, s 18).

Kestävän kehityksen näkökulma tulee huomioida myös määrittely- ja suunnittelutyön aikana. Suunnitellun ratkaisun tulee olla linjassa kestävän kehityksen periaatteiden kanssa. Esimerkiksi Helsingissä julkisen liikenteen hiilidioksidipäästöjä on voitu vähentää analysoimalla liikenteestä kertynyttä dataa ja optimoimalla aikatauluja, reittejä ja polttoaineen kulutusta. (Tietoevry, s 27)

3 Erikoissairaanhoidon hoitoonpääsyn seuranta

Erikoissairaanhoidolla tarkoitetaan ”lääketieteen ja hammaslääketieteen erikoisalojen mukaisia sairauksien ehkäisyyn, tutkimiseen, hoitoon, ensihoitoon, päivystykseen ja lääkinnälliseen kuntoutukseen kuuluvia terveydenhuollon palveluja” (Terveydenhuoltolaki 2010/1326 §3). Julkisen sektorin erikoissairaanhoidon palveluja annetaan sairaanhoitopiireissä ja erikoislääkärijohtoisissa terveyskeskuksissa. (Tuominen ym., 2022, s. 8, s.10)

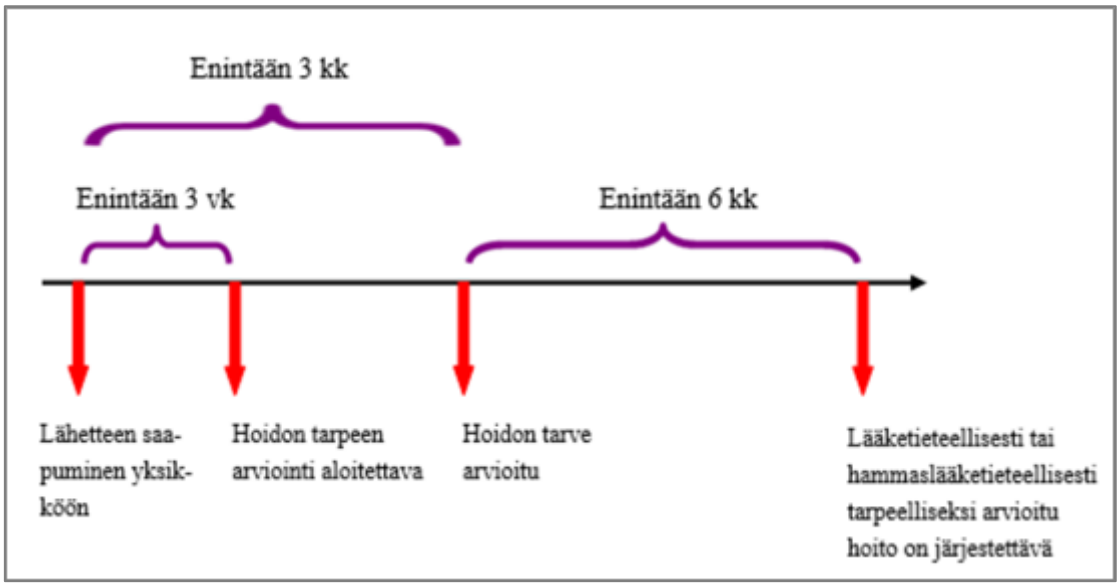
Hoitoonpääsyn seuranta perustuu Terveydenhuoltolakiin. Erikoissairaanhoidon hoitoonpääsyn tiedot kerätään ”sairaanhoitopiirien järjestämästä erikoissairaanhoidosta ja perusterveydenhuollon yhteydessä toteutettavasta erikoissairaanhoidosta.” Hoitoonpääsyn seuranta koskee kiireetöntä erikoissairaanhoitoa. (Tuominen ym., 2022, s. 7)

Potilaan ottaminen sairaalaan kiireettömään hoitoon edellyttää aina lääkärin lähetettä. Hoidon tarpeen arviointi on aloitettava kolmen viikon kuluessa lähetteen saapumisesta sairaalaan. (Terveydenhuoltolaki 2010/1326, §52) Hoidon tarpeen arviointi voidaan tehdä saapuneen lähetteen käsittelyn yhteydessä tai erillisellä arviointikäynnillä. (Tuominen ym., 2022, s. 18)

Tarpeelliseksi katsottu hoito on järjestettävä viimeistään kuuden kuukauden kuluessa hoidon tarpeen toteamisesta (Terveydenhuoltolaki 2010/1326, §52, 3. mom). Hoito voi olla poliklinikka- tai vuodeosastohoitoa, päiväkirurgiaa, muu leikkaus tai toimenpide tai lääkinnällistä kuntoutusta. (Tuominen ym., 2022, s. 19)

Kuva 1 esitetään somaattisen erikoissairaanhoidon ja aikuispsykiatrian hoitoonpääsyn enimmäisajat.

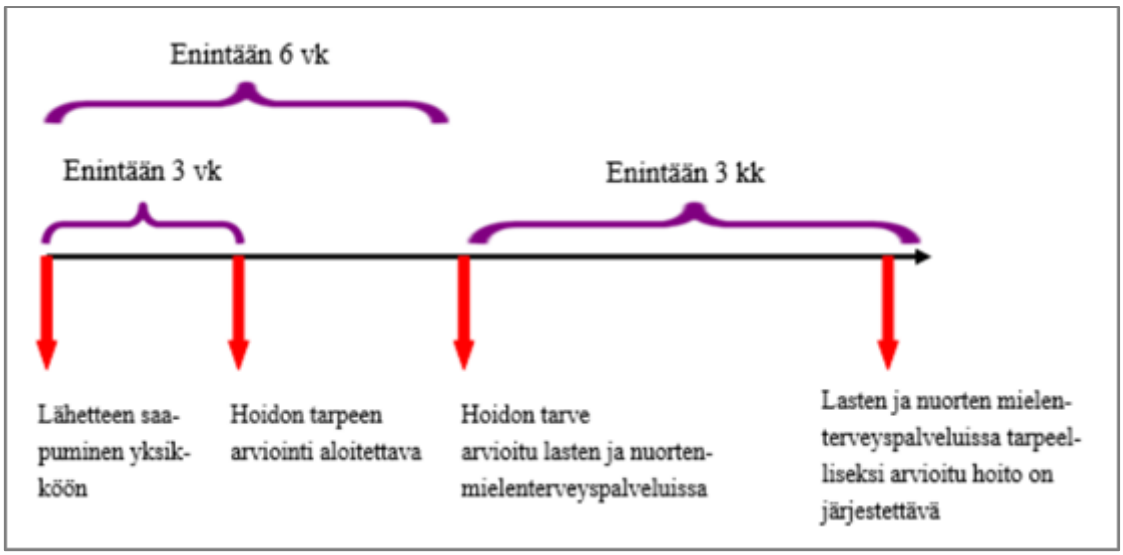
Kuva 1 Terveydenhuoltolain mukaiset hoitoonpääsyn enimmäisajat erikoissairaanhoidossa (Tuominen ym., 2022, s. 11)



Lasten ja nuorten (alle 23-vuotiaiden) mielenterveyspalveluissa hoidon tarpeen arvioinnin edellyttämät tutkimukset ja erikoislääkärin arviointi on toteutettava nopeammin, kuudessa viikossa lähetteen saapumisesta. Hoidon tulee alkaa kolmessa kuukaudessa hoidon tarpeen toteamisesta. (STM, n.d.)

Kuva 2 esitetään lasten ja nuorten mielenterveyspalvelujen hoitoonpääsyn enimmäisajat erikoissairaanhoidossa.

Kuva 2 Terveydenhuoltolain mukaiset hoitoonpääsyn enimmäisajat lasten ja nuorten mielenterveyspalveluissa (Tuominen ym., 2022, s. 12)



Hoitoonpääsyn tilanne raportoidaan Terveyden ja hyvinvoinnin laitokselle THL:lle kuukausittain aina kuukauden viimeisen päivän tilanteesta seuraavan kuukauden 15. päivään mennessä. (Tuominen ym., 2022, s. 6)

Tiedot raportoidaan poikkileikkauspäivän tilanteiden mukaisina taulukkoraportteina. Raporteilla ilmoitetaan mm. hoitoa odottavien lukumäärä ja odotusajat, hoidon tarpeen arviointia odottavien lukumäärät ja odotusajat sekä saapuneet lähetteet ja niiden käsittelyajat. Lisäksi ilmoitetaan yleisimpiin leikkauksiin ja konservatiivisiin hoitoihin odottavien lukumäärä ja odotusajat, myös toteutuneiden hoitojen osalta. Nuorten mielenterveyspotilaiden osalta ilmoitetaan alle 23-vuotiaiden lasten ja nuorten mielenterveyspalvelujen hoitoa odottavien lukumäärä ja odotusajat sekä alle 23-vuotiaiden lasten ja nuorten mielenterveyspalvelujen hoidon tarpeen arviointia odottavien lukumäärä ja odotusajat. (Tuominen ym., 2022, ss. 26–41)

Sosiaali- ja terveysministeriön ohjeiden mukaan kunnan tai sairaanhoitopiirin on julkaistava tiedot kiireettömään hoitoon pääsyn odotusajoista web-sivuillaan vähintään neljän kuukauden välein. Terveydenhuollon viranomaisten työnjaon mukaisesti Terveyden ja hyvinvoinnin laitos THL vastaanottaa hoitoonpääsytiedot erikoissairaanhoidon toimijoilta, ja aluehallintovirastot ja Valvira valvovat hoitoon pääsyn toteutumista. (STM n.d.)

4 Prosessien kuvaaminen ja mallintaminen

Kielitoimiston sanakirjassa prosessi määritellään tapahtumasarjaksi tai kehityskuluksi (Kielitoimiston sanakirja, n.d.). Yksi yleisimmistä prosessin määritelmistä löytyy mm. Cambridgen sanakirjasta: ” a series of actions that you take in order to achieve a result ” (Tietyn lopputuloksen saavuttamiseksi suoritettava sarja toimenpiteitä). (Cambridge Dictionary, n.d.). Tästä määritelmästä on olemassa lukuisia eri muunnelmia.

Termiä prosessin kuvaaminen käytetään usein tilanteessa, jossa tarkastellaan prosessia yksityiskohtaisesti, ja termiä prosessin mallintaminen, kun prosessia tarkastellaan ylätasolla ja käyttäen mahdollisesti muitakin analyysitapoja kuin prosessikuvausta. Kuvaaminen ja mallintaminen viittaavat visuaaliseen esittämistapaan, jota teksti voi täydentää, kuten esimerkiksi kaavioihin. Termejä käytetään myös toistensa synonyymeinä. (Martinsuo & Blomqvist, 2010, ss 8-15)

Prosessien mallintaminen (tai kuvaaminen) on olennainen osa vaatimusmäärittelyä ja ohjelmiston suunnittelua. Mallintamista voidaan ajatella ohjelmiston toiminnan piirustuksina. Mallien avulla pyritään varmistamaan, että loppukäyttäjien tarpeet ovat huomioitu oikein, tekniset vaatimukset saavutetaan ja kokonaisuus on liiketoiminnallisesti järkevä. (Unified Modeling Language UML., n.d.)

4.1 Business Process Model and Notation

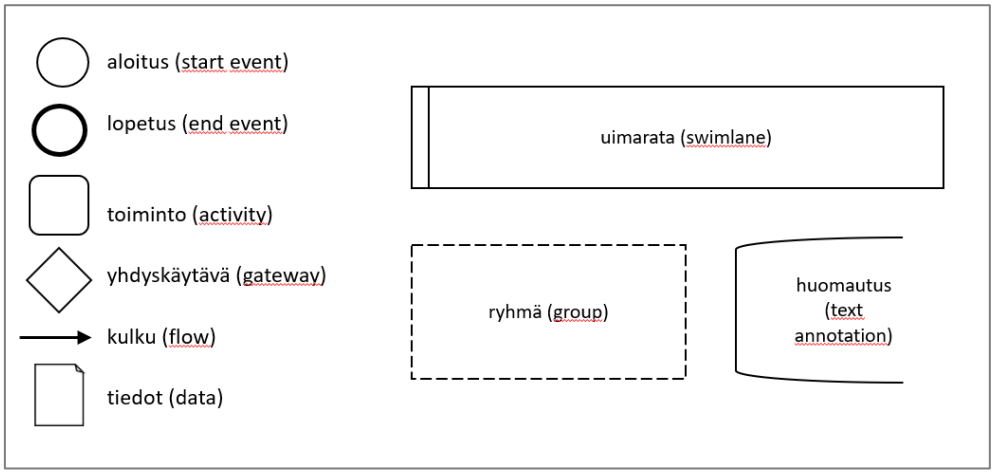
Business Process Model and Notation BPMN on kymmenisen vuotta sitten kehitetty graafinen prosessikuvausmenetelmä, jolla on standardin asema. Pari vuotta sitten esimerkiksi Suomidigin sivuilla Julkisen hallinnon neuvottelukunnan JHS-suosituksessa 152 viitattiin BPMN:n prosessikuvausohjeisiin. (Suomidigi, 2020). Kuvausmenetelmää ylläpitää Object Management Group. (Object Management Group, n.d. -a)

BPMN antaa työkalut prosessien kuvaamistavan yhtenäistämiseen. Organisaatiot voivat kuvata omia prosessejaan yhtenäisellä tavalla sekä viestiä standardoidulla tavalla eri toimijoiden kesken. (Object Management Group, n.d. -a)

BPMN-vuokaavion peruselementtejä ovat tapahtumat (events), toiminto (activity), yhdyskäytävä (gateway) ja kulku (flow). Lisäksi vuokaavioon liittyy data (tiedot), artifact (”häiriö”, selittävä lisäosa kuten ryhmä tai huomautus) ja swimlane (uimarata). (Object Management Group, n.d. -b)

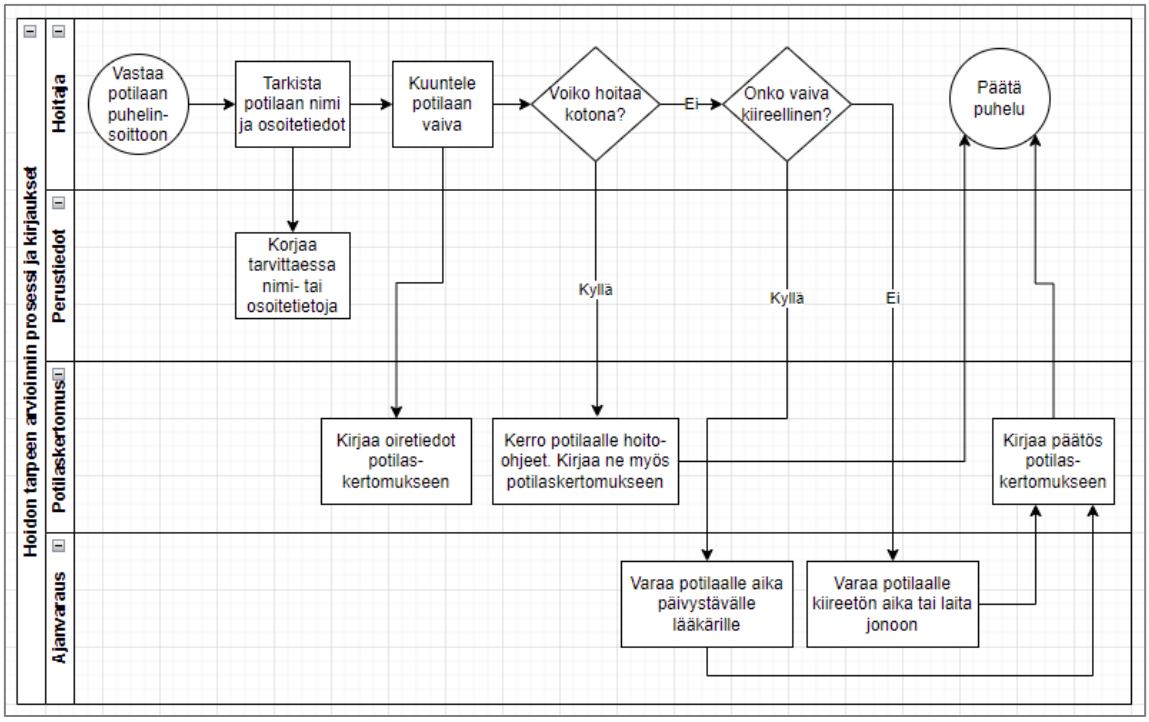
Kuva 3 on esitetty BPMN-vuokaavion peruselementtejä. Vuokaaviossa on tyypillisesti ainakin kaksi tapahtumaa, aloitus ja lopetus. Niiden väliin kuvataan tarkasteltava prosessi. (Object Management Group, n.d. -b)

Kuva 3 BPMN-vuokaavion peruselementtejä



Kuva 4 on esimerkki prosessikuvauksesta, jossa on käytetty uimaratarakennetta havainnollistamaan mitä käyttäjä tekee ja mitä tapahtuu eri sovelluksissa. Kuvan prosessissa potilas ottaa yhteyttä terveyskeskukseen, jossa hoitaja tekee puhelimessa (perusterveydenhuollon) hoidon tarpeen arvioinnin ja sopii jatkohoidon potilaan kanssa.

Kuva 4 Esimerkki BPMN:llä kuvatusta prosessista



BPMN:n sivuilta on luettavissa ja ladattavissa englanninkielinen BPMN-pikaopas, jossa ohjeistetaan notaation perusmerkinnät. Sivuilta löytyy myös esimerkkejä notaation käytöstä. (Object Management Group, n.d. -b) Notaatioon voi perehtyä syvällisemmin BPMN-sivuilta löytyvän koulutuksen avulla. Kuvaustavasta voi suorittaa myös sertifikaatin. (Object Management Group, n.d. -a)

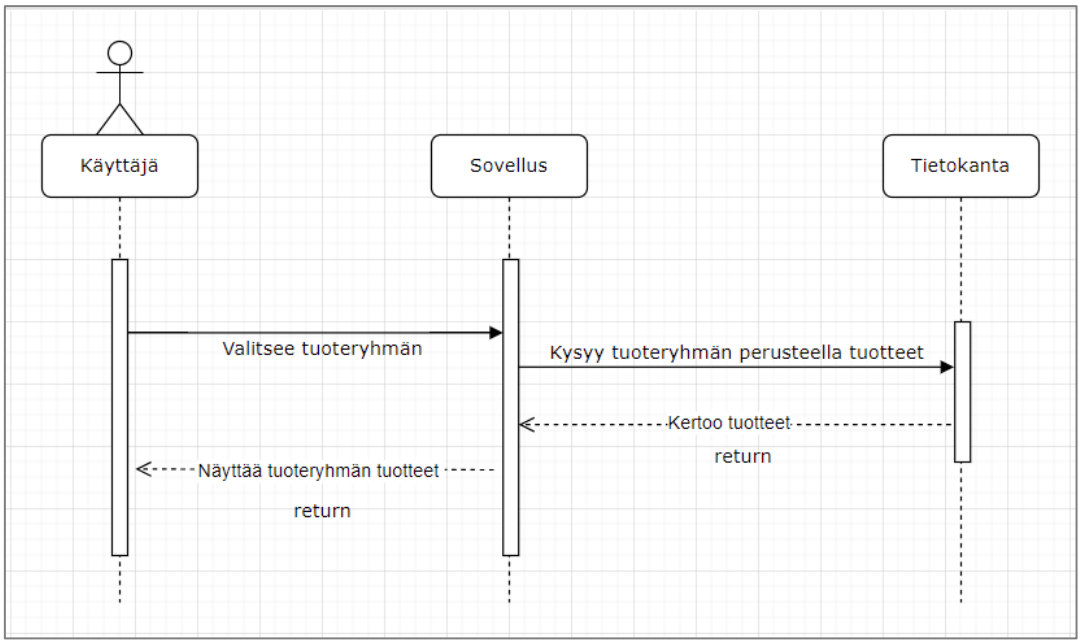
4.2 Unified Modeling Language

Unified Modeling Language (UML) on standardoitu mallinnustapa, jolla voidaan määritellä, visualisoida ja dokumentoida ohjelmiston ominaisuuksia, myös ohjelmiston arkkitehtuuria. UML syntyi parikymmentä vuotta sitten USA:ssa, kun hajallaan olevia mallinnustapoja haluttiin yhtenäistää ohjelmistotuotannon tarpeisiin. Kuvausmenetelmää ylläpitää Object Management Group. (Unified Modeling Language UML., n.d.)

UML-kaavioita on kolmea päätyyppiä ja kolmeatoista eri alatyyppiä. Tavallisimmin käytettyjä ovat Rakennekaavion (structure diagram) alatyypit Luokkakaavio (class diagram)(Kuva 6, Kuva 7) ja Oliokaavio (object diagram). Käyttäytymiskaavion (Behaviour diagram) tavallisimpia alatyyppejä ovat Aktiviteettikaavio (Activity diagram) ja käyttötapauskaavio

(Use Case diagram)(Kuva 14) sekä Vuorovaikutuskaavion (Interaction diagram) alatyyppi Sekvenssikaavio (Sequence diagram) (Kuva 5) (Unified Modeling Language UML., n.d.)

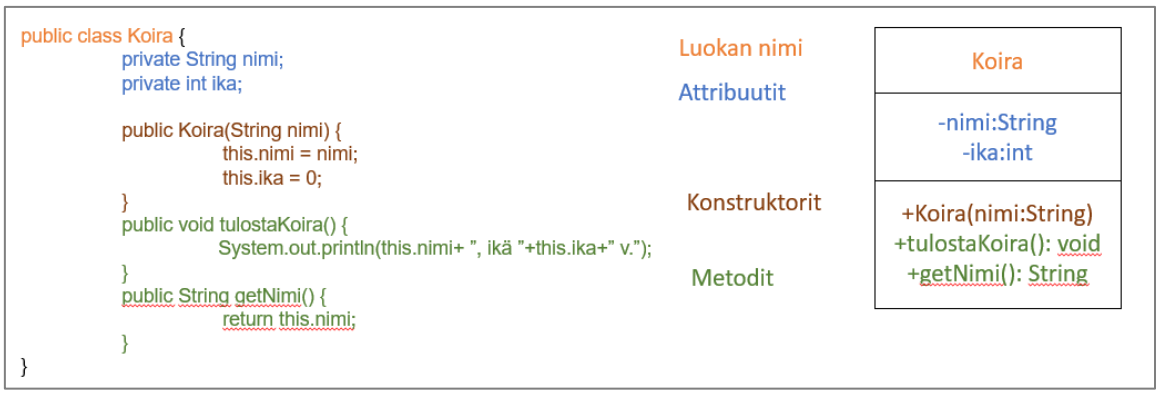
Kuva 5 Esimerkki UML:lla tehdystä sekvenssikaaviosta



UML-kaaviossa voi luoda sisäkkäisiä kaaviorakenteita, koska lähes kaikki kaavioiden rakennustarpeet ovat luokittelijoita. Silloin luokkia voidaan upottaa niitä hallitsevan komponentin sisään ja porautua kuvauksessa tarkastelemaan niitä tarkemmin. Esimerkiksi jonkin komponentin käyttäytyminen voidaan määrittää sekvenssikaavion (Kuva 5) avulla. (Unified Modeling Language UML., n.d.)

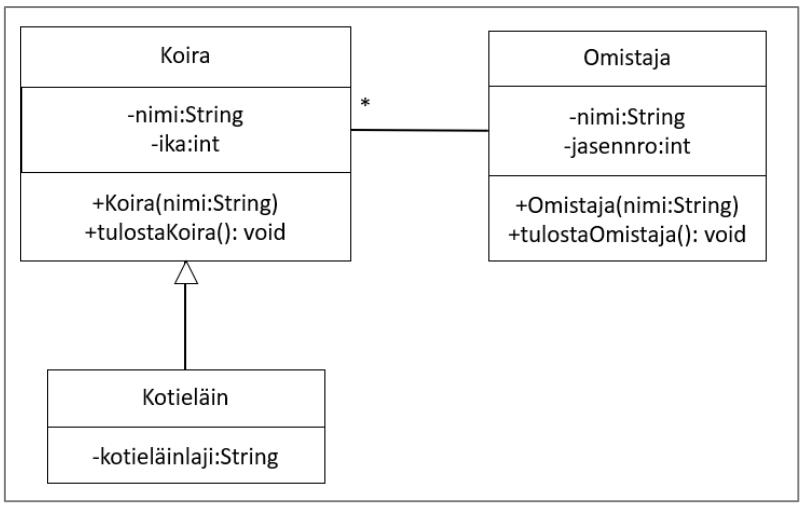
Kuva 6 on värien avulla kuvattu olio-ohjelmoinnin koodin osien ja UML-luokan osien vastaavuuksia, eli mitkä UML-kaavion luokan eri osat vastaavat koodin osia Luokan nimi, Attribuutit, Konstruktorit ja Metodit. Miinusmerkki attribuutin edessä tarkoittaa privatea, plusmerkki konstruktorin ja metodin edessä publicia. (Helsingin yliopiston tietojenkäsittelytieteen osasto, 2020)

Kuva 6 Esimerkki UML:lla tehdystä luokkakaaviosta.



Kuva 7 on kuvattu luokka Koira, joita voi olla luokalla Omistaja rajaton määrä sekä luokka Kotieläin, joka perii luokan Koira. Koiran ja Omistajan välinen nuoleton viiva tarkoittaa, että omistaja tuntee koiransa ja koira omistajansa.

Kuva 7 Esimerkki UML:lla tehdystä luokkakaaviosta. Luokkien välisiä suhteita.



Joitakin UML-kuvauksien tyyppejä voidaan luoda lähdekoodin perusteella, ja toisinpäin. UML-kuvauksen perusteella voidaan luoda myös testitapauslistoja. (Unified Modeling Language UML., n.d.)

Markkinoilla on useita UML-pohjaisia työkaluja, joilla voi suunnitella ja analysoida sovellusratkaisuja ja esittää tulokset UML-kaaviotyyppien avulla. Esimerkiksi Oraclen JDeveloper käyttää UML-mallinnusta. (Oracle Data Sheet, n.d.)

5 Tuotekehityksen projektimallit

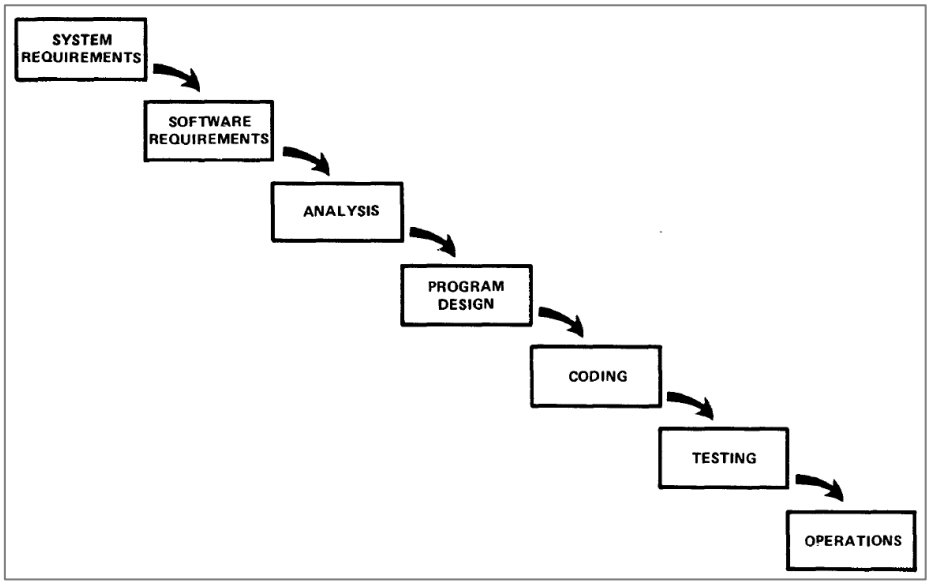
Tuotekehityksen projektimalleilla tarkoitetaan tässä ohjelmistojen kehittämisessä käytettäviä projektimalleja. Projektimalleista on valittu tähän muutamia tunnetuimpia ja käytetyimpiä, mutta malleja ja niiden soveltamistapoja on olemassa paljon enemmän.

5.1 Vesiputousmalli

Vesiputousmalli on vanhin ohjelmistotuotannon prosessimalli. Winston Royce esitteli sen artikkelissaan Managing the Development of Large Software Systems v 1970 (Royce, 1970). Vesiputousmallissa tuotekehitys etenee vaiheittain, ja seuraava vaihe alkaa, kun edellinen loppuu. Malli alkaa vaatimusmärittelyllä, jota seuraa arkkitehtuurisuunnittelu ja sovelluksen suunnittelu. Kun suunnitelmat ovat valmiit, alkaa koodausvaihe ja sen jälkeen testausvaihe. Vaiheisiin sisältyy useita alivaiheita, monikertaisia tarkistuksia ja tarkkaa dokumentointia. (Luukkainen & Ilves, 2021a)

Kuva 8 on esitetty Roycen artikkelin yksinkertaisin vesiputousmalli. Royce itsekin kritisoi mallia samassa artikkelissa. Malli sai kuitenkin suuren suosion.

Kuva 8 Roycen vesiputousmalli



Vesiputousmallin suurimpia heikkouksia on sen kankeus vastata muutostarpeisiin, joita tulee asiakkaan taholta, toimintaympäristön taholta, vaatimusten puuttumisen tai vaatimusten väärinymmärryksien vuoksi. Kun laadunhallinta eli testaus tapahtuu vasta loppuvaiheessa, tarvittavat korjaukset vievät ennakoimattoman määrän aikaa ja tulevat kalliiksi. (Luukkainen & Ilves, 2021a)

Vesiputousmallin ongelmiin on vastattu erilaisilla ketterillä menetelmillä mutta niiden käyttöönotto ei ole aina ollut helppoa. (Ahokas, 2022). Vesiputousmallin on arvioitu olevan edelleen käytetyin kehitysmalli (Chowdhury ym.,2017, s 2).

5.2 Agile

Agile eli ketterä ohjelmistokehitys alkoi muotoutua parikymmentä vuotta sitten Utahissa Yhdysvalloissa, kun seitsemäntoista silloisiin ohjelmistokehityksen käytäntöihin turhautunutta ammattilaista kokoontui viikonlopuksi miettimään, miten toimintaa ja ennen kaikkea asiakkaiden tyytyväisyyttä voisi kohentaa. Perinteistä vesiputousmallia kevyempiä ohjelmistokehityksen käytäntöjä oli jo olemassa mutta niiden käyttö oli hajanaista. (Agile Alliance, n.d.).

Tuossa kokoontumisessa syntyi ketteryyden periaatteet kokoava Agile manifesti ja myöhemmin kuvattiin sen takana olevat kaksitoista periaatetta. Ketterä ohjelmistokehitys on yläkäsite, jonka alla on erilaisia viitekehyksiä ja käytäntöjä. Niille on yhteistä ketterän kehityksen manifestissa ja sen taustalla olevissa 12 periaatteessa ilmaistut arvot. (Agile Alliance, n.d.).

”Ketteryys (Agile) on projektinhallinnan ja ohjelmistokehityksen iteratiivinen, työvaiheita toistava lähestymistapa, joka auttaa tiimejä luomaan arvoa asiakkailleen nopeammin (…).” Ketterässä työskentelytavassa tiimi toimittaa työn tulokset pienissä mutta toimivissa erissä. Ketteryyteen kuuluu myös vaatimuksien, suunnitelmien ja tuloksien jatkuva arviointi. Sen vuoksi tiimeillä on valmiudet reagoida muutoksiin nopeasti. (Atlassian Agile Coach, n.d. -a)

Ketterä lähestymistapa ohjelmistokehitykseen arvostaa työntekijöitä ja heidän yhteistyötään. Ratkaisut kehittyvät yhteistyön avulla, kun itseohjautuvat ja monitaitoiset tiimit hyödyntävät omaan toimintaympäristöönsä sopivia käytäntöjä. (Agile Alliance, n.d.)

Ketterän ohjelmistokehityksen manifesti kuuluu seuraavasti:

> Löydämme parempia tapoja kehittää ohjelmistoja, kun teemme sitä itse ja autamme siinä muita.
>
> Kokemuksemme perusteella olemme päätyneet arvostamaan:
>
> Yksilöitä ja kanssakäymistä enemmän kuin prosesseja ja työkaluja.
>
> Toimivaa ohjelmistoa enemmän kuin kattavaa dokumentaatiota.
>
> Asiakasyhteistyötä enemmän kuin sopimusneuvotteluja.
>
> Vastaamista muutokseen enemmän kuin suunnitelman seuraamista.
>
> Oikeanpuoleisillakin asioilla on arvoa, mutta vasemmanpuoleisia arvostamme enemmän. (Agile Alliance, n.d.).

Agilen manifestin takana olevat kaksitoista periaatetta koskevat asiakkaan tyytyväisenä pitämistä säännöllisillä toimituksilla, muuttuvien vaatimusten vastaanottamista myös myöhäisessä vaiheessa, toimivien ohjelmistoversioiden toimittamista säännöllisesti muutaman viikon välein, yhteistyötä yrityksen liiketoiminnasta vastaavien kanssa, motivoituneiden yksilöiden ympärille rakennettavia projekteja, kasvotusten tapahtuvaa tiedonvälitystä, edistyksen mittaamista toimivalla ohjelmistolla, keskittymistä työtavan kestävyyteen, ohjelmiston tekniseen laatuun ja hyvään rakenteeseen, keskittymistä yksinkertaisuuteen, tekemättä jätettävän työn maksimointiin, itseohjautuvien tiimien suunnitelmiin ja tiimin työskentelyn säännölliseen itsearviointiin. Agile-ajattelu mullisti ohjelmistokehityksen ja sen pohjalta on innovoitu useita erilaisia viitekehyksiä ohjelmistokehityksen tueksi. (Atlassian Agile Coach, n.d.-b)

5.3 Scrum

Scrum on yksi tunnetuimpia ketterän ohjelmistokehityksen viitekehyksiä. Rugby-termiä scrum ei ole tässä yhteydessä suomennettu. Scrum on kehitetty ohjelmistokehityksen projektinhallintaan mutta sitä voi käyttää muissakin projekteissa. Scrumin ytimessä ovat monitaitoiset scrum-tiimit, jotka organisoivat itsenäisesti työtään, oppivat kokemuksistaan ja pyrkivät jatkuvasti parantamaan suoritustaan. (Drumond, C.,n.d.)

Scrum-projektissa on kolme roolia: tuoteomistajan, scrum masterin ja kehitystiimin jäsenen roolit. Roolit eivät ole ammattinimikkeitä tai titteleitä vaan kuvaavat henkilön vastuualuetta scrum-kehityksen aikana. (West, D.,n.d.)

Tuoteomistajalla tulee olla selkeä näkemys tuotteen kehittämisestä. Tuoteomistajan tulee ymmärtää asiakkaiden tarpeet ja hän on myös liiketoiminnan edustaja tiimin suuntaan. Tuoteomistajan vastuulla on organisoida, että asiakkaiden vaatimukset kootaan ja priorisoidaan sekä pilkotaan kehitysjonoksi (backlog), joka esitellään tiimille. Tuoteomistaja ei kuulu kehitystiimiin mutta tiimillä ja hänellä tulee olla hyvä keskinäinen luottamus toisiinsa. (West, n.d.)

Scrum master on kehitystiimin jäsen ja tiimiä palveleva vetäjä, jonka tehtävä on pitää huolta siitä, että ohjelmistokehitys tapahtuu scrumin periaatteiden mukaan. Scrum master huolehtii siitä, että tiimin työn eteneminen on läpinäkyvää kaikille. Esimerkiksi hän organisoi kehitysjonon tehtävien visualisoinnin kanban-taulun avulla, järjestää päivittäiset scrum-palaverit, suojaa tiimin työrauhaa ulkoa tulevilta keskeytyksiltä ja poistaa tiimin työskentelyn esteitä. Lyhyesti ilmaisten, scrum master pitää scrum-prosessin käynnissä ja liimaa sen osaset yhteen. (West, n.d.)

Kehitystiimi ja sen jäsenet muodostavat itsenäisen ja monitaitoisen tiimin, jonka tehtävänä on toteuttaa ne kehitysjonon tehtävät, jotka tiimi valitsee tehtäväkseen. Tiimin suositeltu koko on 5–10 henkilöä. Kehitystiimiin voi kuulua monenlaisia osaajia kuten arkkitehteja, käyttöliittymäsuunnittelijoita, tuotekehittäjiä, ja testaajia. Tiimin jäsenillä tulee olla valmius toimia itse useammassa roolissa tarvittaessa ja oppia koko ajan lisää. (West, n.d.)

Scrum-projektissa työskentely tapahtuu lyhyissä, viikon tai kahden mittaisissa sprinteissä. Sprintin alussa, sprintin suunnittelussa, tiimi valitsee tuoteomistajan esittelemästä tuotteen kehitysjonosta ne kehitystehtävät, jotka se uskoo voivansa toteuttaa sprintin aikana. Tiimi tekee tehtävistä työmääräarviot ja tarvittaessa pilkkoo niitä pienemmiksi tehtäviksi. Tätä kehitysjonoa kutsutaan sprintin kehitysjonoksi. (Rehkopf, n.d.)

Sprintin aikana tiimillä on joka päivä lyhyt, noin viidentoista minuutin päivittäispalaveri, (daily), jossa jokainen tiimin jäsen kertoo vuorollaan mitä hän on tehnyt edellisenä päivänä, mitä hän aikoo tehdä tänään ja onko tekemisessä mitään ongelmia. Näin tiimin työskentely tulee läpinäkyväksi kaikille. (Rehkopf, n.d.)

Sprintin päätteeksi tiimillä tulee olla toimiva ja toimitettavissa oleva sovellusosio. Työn tuloksen tulee myös täyttää aikaisemmin laadittu ”valmiin määritelmä” (Definition of Done), johon voi kuulua esimerkiksi dokumentteja kuten asennusohjeet. Sprintin lopussa järjestetään katselmointi (sprint review), jossa tiimi esittelee tuotoksen ja vastaa kysymyksiin. Sprintin päätteeksi tiimi järjestää usein myös sisäisen arviointipalaverin, retrospektiivin, jossa käydään läpi mitä opittiin, mikä meni hyvin ja mitä voitaisiin parantaa (Rehkopf, n.d.).

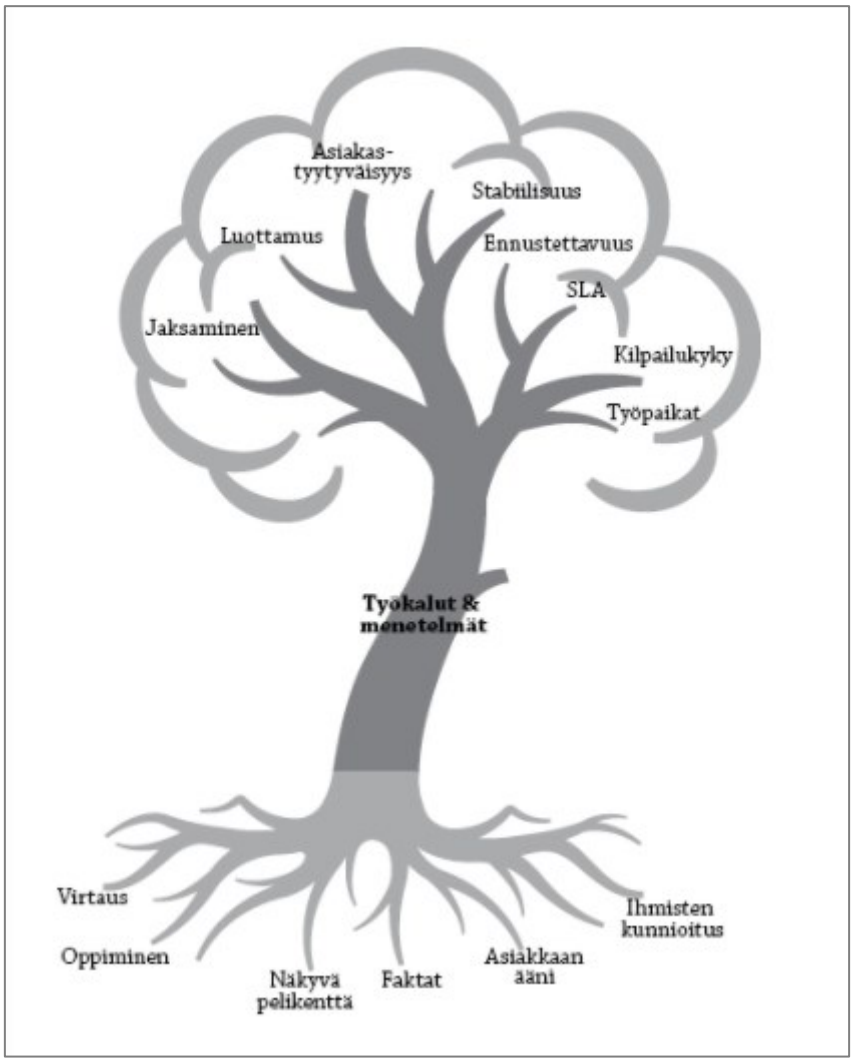
5.4 Lean

Kirjassaan *Lean asiantuntijatyön johtamisessa* Sari Torkkola kertoo, kuinka lean-johtamisessa tavoitellaan työn sujuvaa etenemistä, virtausta. Virtauksen kolme pahinta vihollista ovat vaihtelu, ylikuormitus ja hukka. Päämäärään pääsemiseksi nämä esteet tulee poistaa. (Torkkola, 2015, s.23)

Leanilla voidaan ajatella olevan myös laajempaa, jatkuvaan kehittämiseen kannustavaa merkitystä. ”Lean-nimitys viittaa johtamisen filosofiaan, ajattelutapaan, menetelmiin ja työkaluihin, jotka keskittyvät erilaisten ’turhuuksien’ poistamiseen työ- ja tuotantoprosesseissa. Tavoitteena on jatkuva kehittäminen laadun parantamiseksi ja samaan aikaan kevyt ja joustava työn tai tuotannon malli. ” (Mikkonen-Craig, H., 2020).

Torkkola kuvaa Lean-ajattelua puuna (Kuva 9), jossa juuristossa ovat periaatteet, rungossa näkyvät toimenpiteet ja työkalut ja oksistossa hedelmät. (Torkkola, 2015, s. 219)

Kuva 9 Lean-ajattelun puu (Torkkola, 2015, s. 219)



Torkkola listaa lean-ajattelun periaatteet seuraavasti:

Periaate 1: Virtaus on päämäärä.

Periaate 2: Oppiminen on tärkeämpää kuin suorittaminen.

Periaate 3: Tilannekuva visualisoidaan kaikille näkyväksi.

Periaate 4: Päätökset tehdään tosiasioiden pohjalta.

Periaate 5: Asiakkaan ääni antaa suunnan.

Periaate 6: Ihmisten kunnioittaminen on lähtökohta. (Torkkola, 2015, s. 220)

Leanin keskeisiä käsitteitä ovat mm.

- Arvoa lisäävä toiminta: arvoa lisäävää toimintaa ovat asiakkaan näkökulmasta kaikki ne aktiviteetit, jotka muokkaavat tuotetta tai palvelua eteenpäin kohti sitä, mitä asiakas on tilannut.

- Arvoa lisäämätön toiminta eli hukka: hukka on toimintaa, jolla ei asiakkaan näkökulmasta ole arvoa. Tällaisia ovat esimerkiksi varastointi ja jonottaminen.

- Jatkuva virtaus: jatkuva virtaus tarkoittaa, että tavarat ja puolivalmisteet liikkuvat tuotantojärjestelmässä ilman, että niitä varastoidaan prosessin aikana tai kootaan yhteen tuotantoeriksi.

- Pullonkaula: pullonkaula on prosessin hitain vaihe, joka hidastaa koko prosessin etenemistä.

- Ongelmanratkaisu: ongelmanratkaisu on käytäntö, jossa määritellään ratkaisut tunnistettuihin ongelmiin. Sen avulla mahdollistetaan parannus- tai kehittämistoimenpiteet prosessin, tuotteen tai palvelun osalta. Ongelmanratkaisu voi perustua joko loogiseen päättelyyn tai luoviin ongelmanratkaisun menetelmiin ja joissain tilanteissa myös niiden yhdistelmään.

- Juuri oikeaan tarpeeseen: juuri oikeaan tarpeeseen tarkoittaa, että asiakas saa tarvitsemansa palvelun, jolla on haluttu vaikuttavuus ja juuri silloin, kun hän sitä tarvitsee (LeanThinking, n.d.)

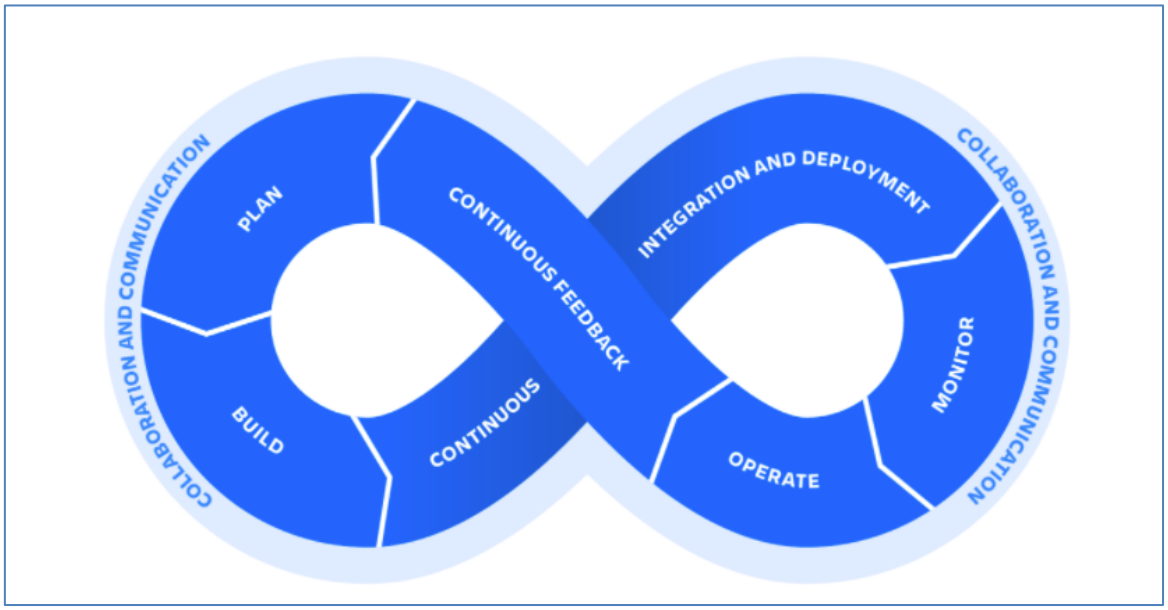
5.5 DevOps

DevOps on ohjelmistokehityksen toimintamalli, joka tarkastelee kehittämisen, testauksen, toimituksen ja ylläpidon prosesseja yhtenä kokonaisuutena. Malli korostaa tuotekehityksen (Development) sekä toimituksen ja ylläpidon (Operations) yhteistyötä. Määrittely- ja suunnitteluvaiheet malli sivuuttaa lyhyesti, usein feedback-termillä. DevOps-mallissa

keskitytään automaattiseen, laaja-alaiseen laadunvalvontaan ja mittaamiseen kehitysprosessin joka vaiheessa, jatkuvaan integraatioon ja jatkuvaan toimitukseen. (Felsen, N., 2017, s 16, s19)

DevOps-tiimiin kuuluu kehittäjiä ja ylläpitäjiä, joilla on yhteinen tavoite nopeuttaa ja parantaa ohjelmiston käyttöönottoa ja kehittää ohjelmiston laatua. Joskus tiimi voi työskennellä yhdessä tuotteen koko elinkaaren ajan. DevOps-toimintamallia kuvataan usein kahdeksikkosilmukalla, joka havainnollistaa jatkuvaa yhteistyötä ja jatkuvaa parantamisen tarvetta ohjelmiston koko elinkaaren ajan. Tuotteen DevOps-elinkaari rakentuu kuudesta (Atlassian, n.d.)(Kuva 10) tai joissakin lähteissä kahdeksasta (Gunja, 2021) eri prosessista.

Kuva 10 DevOps elinkaari



DevOpsin vasemmanpuoleisen silmukan vaiheet (jatkuva palaute, suunnittelu, kehitys ja testaus, jatkuva integrointi ja toimitus) koskevat tuotteen kehitysvaihetta. Oikeanpuoleisen silmukan vaiheet (jatkuva integrointi ja toimitus, seuranta ja ylläpito sekä jatkuva palaute) koskevat tuotteen operaatiovaihetta. Kaikkien vaiheiden aikana tiimin yhteistyö ja kommunikaatio on välttämätöntä toimituksen yhdenmukaisuuden, laadun ja nopeuden ylläpitämiseksi. (Atlassian, n.d.)

DevOps-tiimit käyttävät omia työkaluohjelmistojaan, joiden avulle ne voivat toteuttaa jatkuvaa integrointia, jatkuvaa toimitusta, automaatiota ja yhteistyötä. (Atlassian, n.d.)

DevOps-toimintamallia varten on kehitetty myös omia pilvipalveluympäristöjään, joista ohjelmistoyritys voi helposti hankkia käyttöönsä kaiken tarpeellisen. (Felsen, N., 2017, s 19)

5.6 SAFe (© Scaled Agile, Inc.)

SAFe on lyhenne sanoista Scaled Agile Framework, jonka voisi suomentaa skaalautuvaksi ketteräksi viitekehykseksi. SAFe for Lean Enterprises -viitekehyksen tavoitteena on edistää digitaalisen liiketoiminnan ketteryyttä ja palvelujen kehittämistä. SAFe-viitekehys laajentaa ketterän ohjelmistokehityksen logiikkaa yritysten muihinkin toimintoihin kuten tuotevalikoiman hallintaan ja rahoitukseen. SAFe-materiaalin oikeudet omistaa Scaled Agile, Inc. (Scaled Agile, Inc., 2021 a).

SAFen viitekehys yhdistää Leanin, Agilen ja DevOpsin menetelmät yhtenäiseksi toimintamalliksi, jonka avulla pyritään tuottamaan ohjelmistoja ja palveluita nopeammin, ennakoitavammin ja laadukkaammin. SAFe korostaa asiakaslähtöisyyttä, ymmärrystä siitä miten tehdyt päätökset vaikuttavat asiakkaaseen. Asiakas on ”se, joka käyttää työni tulosta saadakseen siitä hyötyä”. Asiakasta varten tulee rakentaa kokonaisratkaisuja ja ymmärtää asiakassuhteen elinkaaren arvo. (Rusanen, 2022)

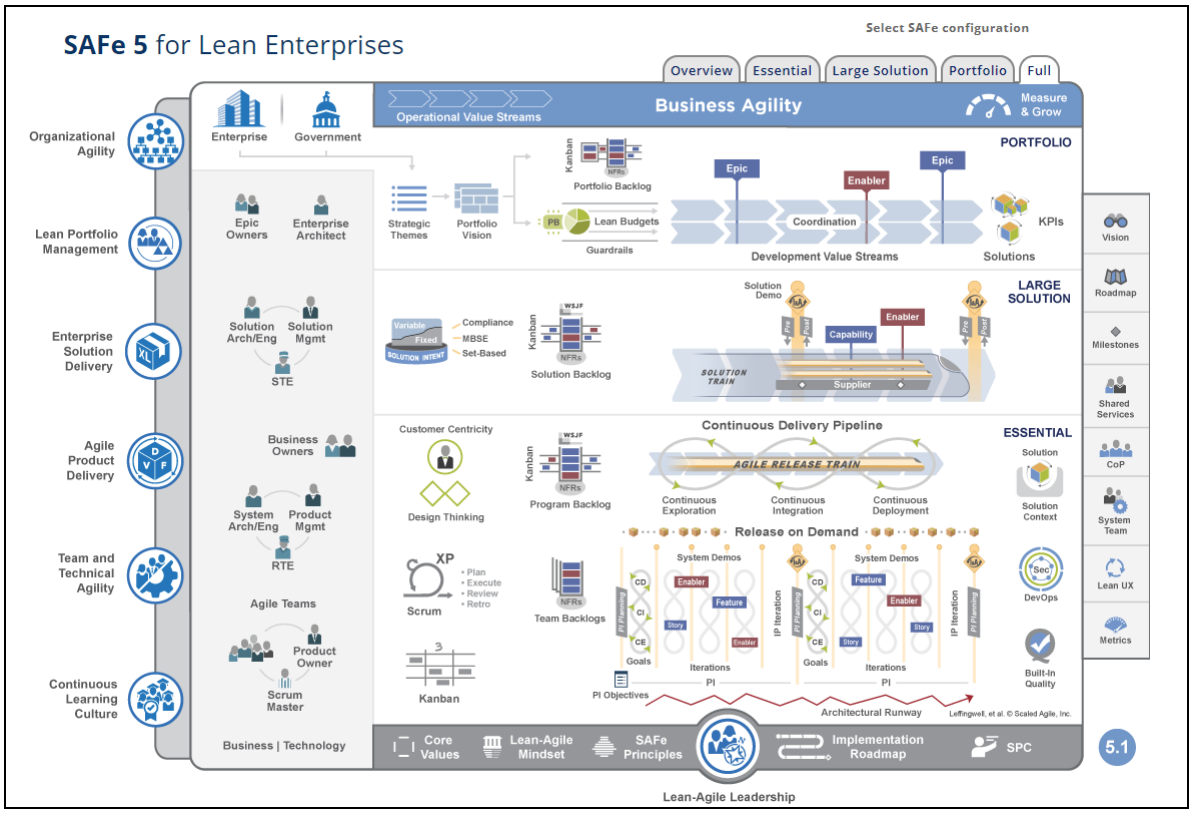
SAFen periaatteet ovat:

- Ota taloudellinen näkökulma
- Hyödynnä systeemiajattelua
- Varaudu vaihteluun, huolehdi vaihtoehdoista
- Kehitä vaiheittain nopeiden, toisiaan täydentävien lisäosien avulla
- Määritä ajankohdat järjestelmän kehityksen objektiiviselle arvioinnille
- Visualisoi ja rajoita tarvittavan työn määrää, pienennä eräkohtaista työmäärää ja hallitse jonojen pituutta.
- Rytmitä ja synkronoi työtä toimialueiden yli ulottuvalla suunnittelulla
- Kartuta tietotyöntekijöiden sisäistä motivaatiota
- Hajauta päätöksenteko
- Rakenna organisaatio arvon ympärille (Scaled Agile, Inc., 2021 b).

SAFe on muokattava ja skaalautuva malli, joka tukee neljällä erilaisella kokoonpanolla eri kokoisia ohjelmistotuotantoratkaisuja muutaman tiimin sovelluksista aina vaativiin ja monimutkaisiin, satojen työntekijöiden tuotantojärjestelmiin. Mallissa kuvataan tarkasti kehitystyöhön liittyvät roolit ja vastuut. SAFe-viitekehys sisältää ohjeet myös sen itsensä käyttöönottoon. SAFe-viitekehystä päivitetään jatkuvasti palautteen perusteella. Tämänhetkinen versio on 5.1. (Scaled Agile, Inc., 2021 a)

Kuva 11 esitetään SAFe-viitekehyksen keskeiset elementit (© Scaled Agile, Inc). Kuvan alareunassa on SAFe-mallin perusta, eli ydinarvot (core values), Lean-Agile ajattelutapa (Lean-Agile Mindset) sekä SAFen edellä mainitut periaatteet. Perustaan kuuluu myös käyttöönoton tiekartta (Implementation Roadmap) sertifioidun SAFe-ohjelman konsultin (SPC) opastamana. Lean-Agile -johtajuus (Lean-Agile Leadership) on kuvattu ydinkompetenssien yhteydessä. (Scaled Agile, Inc., 2021 a)

Kuva 11 SAFe-viitekehys Full, (Scaled Agile, Inc., 2021 a)

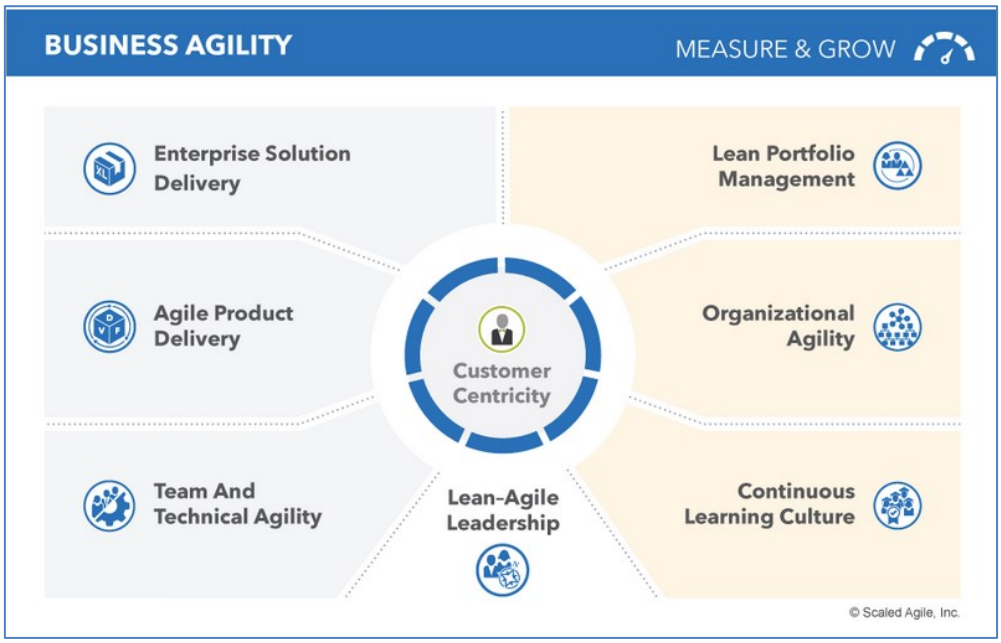


Kuvan vasemmassa reunassa ovat ketterän liiketoiminnan ydinkompetenssit (ydinosaamisalueet): organisatorinen ketteryys (Organizational Agility), lean-portfolion hallinta (Lean Portfolio Management), lean-yritysratkaisujen käyttöönotto (Enterprise Solution Delivery), ketterä asiakaskeskeinen kehitysprosessi (Agile Production Delivery),

tiimien ja teknologinen ketteryys (Team and Technical Agility) ja jatkuvan oppimisen kulttuuri (Continuous Learning Culture). Seitsemäs ydinkompetenssi on organisaation Lean-Agile -johtajuus. Kaikki kompetenssit liittyvät asiakaskeskeisyyteen.

Kuva 12 esitetään ydinkompetenssit tiivistetysti (© Scaled Agile, Inc). Asiakaskeskeisyys on kompetenssien keskeinen elementti. Malli painottaa myös ketterän johtamistavan merkitystä (Scaled Agile, Inc., 2021 c).

Kuva 12 Ketterän liiketoiminnan ydinkompetenssit, (Scaled Agile, Inc., 2021 c)



SAFe-viitekehys Full -kuvan (Kuva 11) alaosassa, Välttämättömät (Essential)-alueella on kuvattu ketterän tuotekehityksen käytännöt, jotka ovat tuttuja Scrum ja Devops -malleista ja palvelumuotoilusta. Keskeinen organisatorinen työkalu SAFessa on julkaisujuna (Agile Release Train, ART). Julkaisujunat ovat lähtökohtaisesti monialaisia ja niillä on kaikki ominaisuudet – henkilöstö, ohjelmistot, laitteistot, laiteohjelmistot ja muut –, joita tarvitaan ratkaisujen määrittämiseen, toteuttamiseen, testaamiseen, käyttöönottoon, julkaisuun ja soveltuvin osin myös käyttöön. (Scaled Agile, Inc., 2021 d).

Julkaisujuna toimii 8–12 viikon toistuvissa jaksoissa eli inkrementeissä (PI, Program increment). Inkrementin puitteissa ketterät tiimit rakentavat tuotteesta seuraavan version. Inkrementti koostuu tyypillisesti neljästä tai viidestä sprintistä. Yhden sprintin kesto on kaksi tai enintään kolme viikkoa. (Rusanen, 2022)

Isot sovellukset/ratkaisut -alueella (Large solutions) yksittäiset julkaisujunat yhdistyvät sovelluskokonaisuuden yhteensovittavaksi järjestelmäjunaksi. Järjestelmäjunat vaativat julkaisujunien yhteensovittamisen lisäksi yhteistä visiota ja yhteistä tiekarttaa. Isojen ratkaisujen hallintaan tarvitaan myös julkaisujunien kehitysjonot (backlogit) yhdistävä järjestelmäbacklog. (Scaled Agile, Inc., 2021 e)

Kuva 11 Portfolio-alueen ominaisuuksia tarvitaan silloin, kun suurista sovelluskokonaisuuksista kehitetään laajoja järjestelmiä ja ekosysteemejä. Portfolioajattelussa lähdetään stategisista teemoista, arvovirroista ja portfoliovisiosta. Lean budjetointi rakennetaan arvovirtojen ympärille. (Scaled Agile, Inc., 2021 f).

SAFe tuo mukanaan muutamia uusia rooleja Scrum-roolien rinnalle. RTE, Release Train Engineer toimii ikään kuin julkaisujunan Scrum Masterin roolissa eli tukee ja ohjaa tiimejä niiden tehtävissä. Tuotehallinta (Production Management) toimii tuoteomistajan roolissa julkaisujunan tasolla. Järjestelmäarkkitehti (System Architect tai System Engineer) vastaa julkaisujunan tason arkkitehtuuriasioista. Yhdessä nämä roolit ohjaavat julkaisujunan etenemistä. (Rusanen, 2022; Scaled Agile, Inc. 2021 g)

Large solution -tasolla mukaan tulee sovelluskokonaisuuden ohjaamiseen liittyvät Solution Train Engineer, Solution Management ja Solution Architect tai Solution Engineer. Lisäksi Portfolio-tasolla voi olla portfoliota koordinoivat Epic Owners ja teknologisesta strategiasta vastaava Enterprise Architect. (Rusanen, 2022; Scaled Agile, Inc. 2021 h)

SAFen käyttöönottoprosessi yrityksessä on kuvattu kahdentoista askeleen prosessina, joka etenee neljässä vaiheessa: 1) ohjaavan koalition rakentaminen, 2) käyttöönoton suunnittelu, 3) julkaisujunien, sovellusjunien ja portfolion käyttöönotto sekä 4) ylläpito ja parantaminen. Jokaiseen neljään vaiheeseen sisältyy omat koulutuspakettinsa. (Scaled Agile, Inc. 2021 i)

6 Vaatimusmäärittely

Vaatimusmäärittelyllä määritellään asiakkaan vaatimukset rakennettavalle ohjelmistolle. Vaatimukset voivat olla toiminnallisia, jolloin ohjelmistolla pitää pystyä tekemään määriteltyjä asioita. Vaatimukset voivat olla myös ei-toiminnallisia, eli ohjelmiston tulee täyttää esimerkiksi suorituskyky- tai tietoturvavaatimukset. (Luukkainen & Ilves, 2021 b)

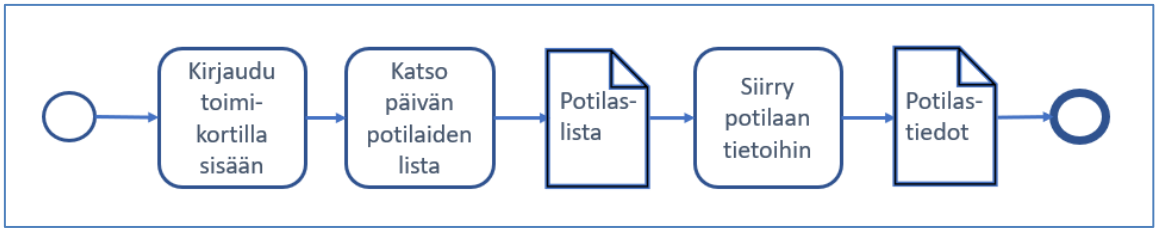
Luukkaisen ja Ilveksen (2021 b) mukaan ”Lineaarisissa prosessimalleissa, eli vesiputousmallissa vaatimusmäärittely tehdään kokonaisuudessaan ennen ohjelmiston suunnittelua ja toteutusta. Iteratiivisessa ohjelmistokehityksessä vaatimusmäärittelyä taas tapahtuu vähän kerrassaan ohjelmiston toiminnallisuuden kasvamisen myötä. ”

Vaatimusmäärittelyä voidaan tehdä monella eri menetelmällä kuten haastattelujen, työpajojen, käyttöliittymäkuvien tai prototyyppien avulla. Määrittelytapaan vaikuttavat mm. kehitettävä ohjelmisto, käytettävä prosessimalli ja kehitysorganisaation käytännöt. Määrittelytavasta riippumatta asiakkaan edustajien tulee olla aktiivisesti mukana vaatimusmäärittelyssä, jotta asiakkaan tarpeet ja vaatimukset tulevat ymmärretyiksi oikein. (Luukkainen & Ilves, 2021 b)

Toiminnalliset vaatimukset kuvaavat järjestelmän käyttöä, mitä ohjelmistolla tulee voida tehdä. Toiminnallisen vaatimuksen esimerkissä terveydenhuollon järjestelmässä vastaanottoa pitävän ammattilaisen tulee voida kirjautua toimikortillaan sisään, nähdä päivän potilaiden listan ja siirtyä listalta potilaan tietoihin ennen potilaan kutsumista huoneeseen. Toiminnalliset vaatimukset voidaan kuvata monin tavoin, kuten BPMN-prosessikaavioina, UML-käyttötapauskaavioina tai ”käyttäjän äänen” mukaisina, kirjoitettuina käyttäjätarinoina.

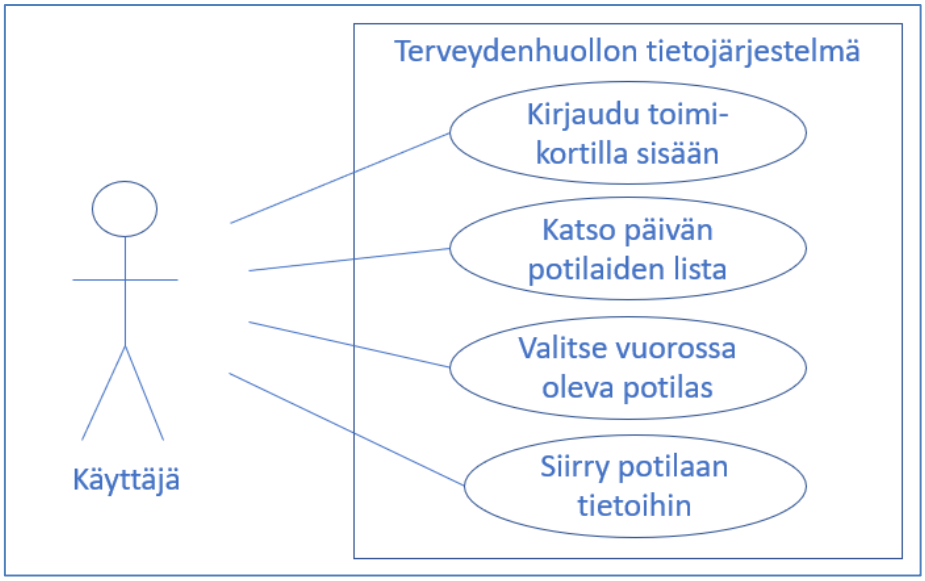
Kuva 13 esitetään käyttäjän toiminta ja sovelluksen näyttämä informaatio prosessikaaviona. Pyörylät ovat alku- ja loppumerkkejä, toiminta kulmistaan pyöristetty nelikulmio ja data lomakesymboli. Etenemissuunta kerrotaan nuolilla.

Kuva 13 Esimerkki BPMN prosessikaaviosta



Kuva 14 esitetään sama käyttäjän toiminta UML-käyttötapausmuodossa. Käyttäjän toiminta on kuvattu Terveydenhuollon tietojärjestelmän kehyksien sisään. Toiminnan vaiheet etenevät ylhäältä alas.

Kuva 14 Esimerkki UML-käyttötapauskaaviosta



Kuva 15 esitetään sama käyttäjän toiminta käyttäjätarinana. Toiminnallisille vaatimuksille on tyypillistä, että ne kuvataan muodossa, jossa jokin käyttäjärooli toteuttaa jonkin yksittäisen käyttötapauksen. Käyttäjätarinan rakenne pyritään pitämään vakiona: Käyttäjänä <rooli> haluan <toiminta> jotta <hyöty> (As a *type of user*, I want *functionality* so that *business value*). (Luukkainen & Ilves, 2021 b)

Kuva 15 Esimerkki kirjoitetusta käyttötapauksesta

Terveydenhuollon ammattilaisena haluan
kirjautua järjestelmään sisään organisaationi tunnistautumistavan avulla,
nähdä päivän potilaiden listan ja
siirtyä suoraan listalta potilaan tietoihin,
jotta voin pitää vastaanottoa päivän potilaiden listan kautta.

Ei-toiminnalliset vaatimukset koskevat usein koko järjestelmää ja ne tulee ottaa alusta asti huomioon järjestelmää suunniteltaessa. Ei-toiminnalliset vaatimukset voidaan jakaa kahteen ryhmään: laatuvaatimuksiin ja toimintaympäristön rajoitteisiin. (Luukkainen & Ilves, 2021 b)

Laatuvaatimukset toisaalta ohjaavat ja toisaalta rajoittavat järjestelmää. Tällaisia ovat esimerkiksi tietoturva kuten pääsynhallinta, suorituskyky kuten sovelluksen toiminnan ripeys, ja skaalautuvuus, kuten voiko käyttäjiä olla yhtä aikaa kymmeniä vai tuhansia. (Luukkainen & Ilves, 2021 b) Toimintaympäristön rajoitteisiin kuuluvat esimerkiksi käyttöympäristö kuten mobiilisovellus tai työpöytäsovellus, mahdolliset tarvittavat integraatiot muihin järjestelmiin sekä lainsäädäntö ja standardit kuten EU:n tietosuojalainsäädäntö. (Luukkainen & Ilves, 2021 b)

6.1 Vaatimusmäärittelyn vaiheet

Vaatimusmäärittely jaetaan perinteisesti useampaan vaiheeseen. Määrittely aloitetaan kartoitusvaiheella, jossa selvitetään ohjelmiston sidosryhmät, eli tahot, jotka ovat suoraan tai välillisesti tekemisissä ohjelmiston kanssa. Sidosryhmiä voivat olla esimerkiksi järjestelmän loppukäyttäjät, käyttötuesta ja ylläpidosta vastaavat, tietohallinto ja organisaation johto. (Luukkainen & Ilves, 2021 b)

Sidosryhmien edustajia haastatellaan ja heidän kanssaan voidaan pitää työpajoja. Asiakkaalla voi olla käyttäjiltä kerättyjä vaatimuksia, tai vaatimuksia voidaan koota havainnoimalla loppukäyttäjän työskentelyä. Järjestelmälle voidaan löytää erilaisia käyttäjärooleja ja näille tyypillisiä käyttötapauksia. (Luukkainen & Ilves, 2021 b)

Seuraavaksi kerätyt vaatimukset dokumentoidaan ja analysoidaan. Dokumentaatiossa kuvataan mm. miten sovelluksen tai sen osien tulee toimia. Sitten vaatimukset analysoidaan. Kattavatko vaatimukset kaikki mahdolliset käyttötapaukset? Onko vaatimuksissa keskinäisiä ristiriitoja? Onko vaatimukset kuvattu niin, että niiden toteutuminen on testattavissa? Ovatko vaatimukset ylipäätään toteutettavissa ja taloudellisesti järkeviä? (Luukkainen & Ilves, 2021 b)

”Erityisesti vesiputousmallia käyttäessä vaatimusdokumentti toimii oleellisena osana asiakkaan ja ohjelmiston kehittäjien välisessä sopimuksessa. Sovelluksen hinta perustuu vaatimusmäärittelyssä kuvattuun toiminnallisuuteen, ja jos asiakas muuttaakin mieltään, saattaa siitä tulla lisäkustannuksia.” (Luukkainen & Ilves, 2021 b)

Vaatimuksien validointi tarkoittaa sen varmistamista, että dokumentoidut vaatimukset todella vastaavat asiakkaan käsitystä järjestelmältä vaadittavista ominaisuuksista. Tässä voidaan käyttää apuna myös esimerkiksi käyttöliittymäkuvia tai prototyyppejä. (Luukkainen & Ilves, 2021 b)

Vaatimuksia tulee myös hallinnoida, eli priorisoida, muokata ja tarkentaa kehitysjonolla. Uusia vaatimuksia voidaan myös lisätä tai poistaa tarpeettomia vaatimuksia kehitysjonolla työn edetessä. (Luukkainen & Ilves, 2021 b)

6.2 Vaatimusmäärittely ja palvelumuotoilu

”Palvelumuotoilu on muotoiluajatteluun perustuva muotoilun osaamisala, joka on erikoistunut palvelujen, asiakas- ja työntekijäkokemusten sekä palveluliiketoiminnan ihmislähtöiseen kehittämiseen. Palvelun käyttäjä missä tahansa roolissa – asiakkaana, asiakaspalvelijana tai yhteistyökumppanina – on palvelumuotoilussa kaiken kehittämisen keskipiste.” (Koivisto ym., 2019, s 34)

Kehittäminen palvelumuotoilun avulla perustuu muotoiluajatteluun, Design thinking. "Muotoiluajattelulla tarkoitetaan ihmislähtöistä innovaatioprosessia, jossa pyritään yhdistämään se, mikä ihmisille on haluttavaa siihen, mikä teknologisesti on toteutettavissa, ja mikä taloudellisesti on kannattavaa." (Koivisto ym., 2019, s 35).

Toinen keskeinen periaate on ”laajentaa muotoilun toiminta-alue tuotekeskeisyydestä kokonaisvaltaisten systeemien sekä ajassa tapahtuvien kokemusten ja prosessien kehittämiseen.” Tällöin esimerkiksi digitaalisia tuotteita ei enää nähdä ohjelmistotuotteina vaan palveluina, joihin liittyy käyttökokemuksia ja prosesseja. (Koivisto ym., 2019, s 34)

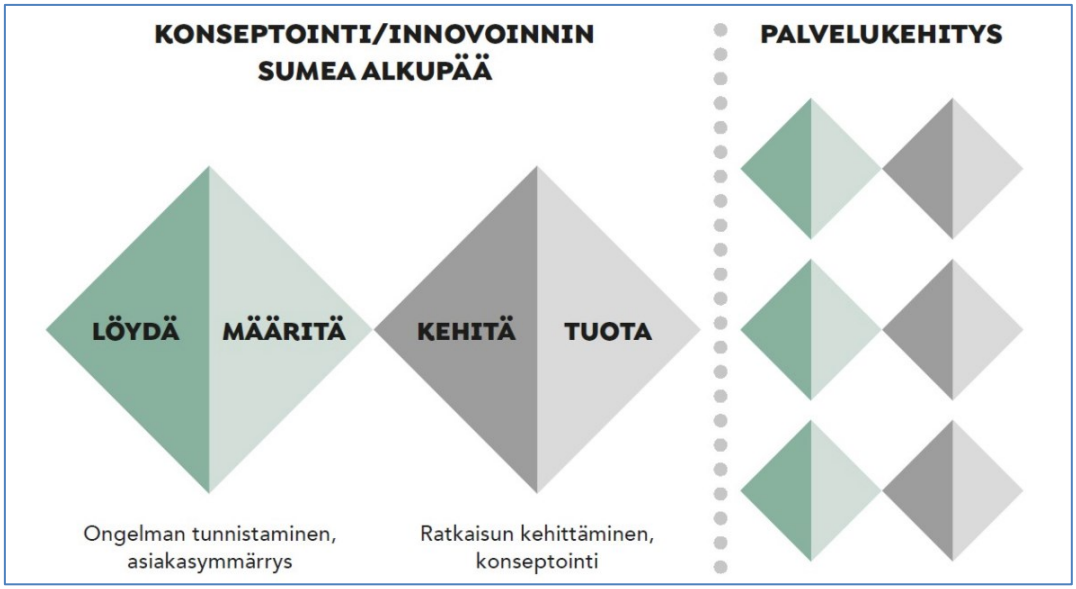
Palvelumuotoilun prosessia voi hyödyntää digitaalisten palvelujen kuten tietokoneohjelmien vaatimusmäärittelyssä. Samalla vaatimusmäärittelyssä käytettävissä olevien työkalujen määrä kasvaa. Konseptoinnin avulla voidaan vaatimusmäärittelyssä edetä vaiheittain, ketterien kehitysmenetelmien rytmin mukaisesti. (Koivisto ym., 2019, s 74)

Palvelumuotoilulla on oma prosessinsa ja laaja työkalupakki menetelmiä. Palvelumuotoilun prosessi esitetään usein Tuplatimantti-mallin avulla, jossa on kaksi vaihetta, timanttia. Ensimmäinen timantti on Ongelman tunnistaminen -timantti, jonka alussa pyritään löytämään ja määrittämään oikea ongelma tai arvonluonnin mahdollisuuksia. Toisessa vaiheessa ideoidaan tunnistettuun ongelmaan tai mahdollisuuksiin erilaisia ratkaisumalleja tai konsepteja, joita arvioidaan ja testataan asiakkailla. Tavoitteena on tuottaa palvelusta konsepti, jonka pohjalta voidaan tehdä päätös mahdollisesta toteutuksesta.

Tuplatimanttivaiheet voivat toistua isoissa projekteissa esimerkiksi niin, että jos alussa on syntynyt uudenlainen palvelukonsepti, joka vaatii toteutuakseen sekä fyysisiä että digitaalisia palveluympäristöjä, eri palveluja voidaan lähteä viemään eteenpäin omina kehittämisprojekteinaan. (Koivisto ym., 2019, s 47)

Kuva 16 Tuplatimantti-konseptoinnin jälkeen eri palvelujen kehitys etenevät omina prosesseinaan. Kuvan ensimmäisessä tuplatimantti-prosessissa tunnistetaan ongelma ja saadaan asiakasymmärrystä ja sen jälkeen kehitetään ratakaisumalli konseptoinnin avulla. Sen jälkeen eri palvelujen kehittämisessä voidaan hyödyntää samaa prosessirakennetta.

Kuva 16 Palvelumuotoilun konseptointi- ja palvelukehitysprosessi



Palvelumuotoilu laajentaa asiakaslähtöisyyden ideaa palvelun kehittämisestä palvelua tarjoavaan yritykseen itseensä. ”Jos yritys haluaa ottaa palvelumuotoilun haltuun omassa toiminnassaan kestävällä ja vaikuttavalla tavalla, vaatii tämä usein muutosprosessia yrityksen kulttuurissa ja toimintatavoissa. Useinkaan ei riitä, että yritys opettelee vain palvelumuotoiluprosessin ja -menetelmien käytön, vaan tarvitaan kokonaisvaltaisempi, yrityskulttuuria ja koko yritystä – johdosta operatiiviseen palvelutuotantoon – koskeva muutos organisaatiolähtöisestä yrityksestä aidosti asiakaslähtöiseksi. Tämä tarkoittaa myös uudenlaista otetta innovaatio- ja kehittämistoiminnassa, jossa organisaatio- tai asiantuntijalähtöisen kehittämisen tulee väistyä asiakas- ja käyttäjälähtöisen kehittämisen tieltä.” (Koivisto ym., 2019, s 164)

6.3 Terveydenhuollon järjestelmiä koskevat standardit

Euroopan unionin lääkinnällisten laitteiden asetus MDR (Medical Devices Regulation) tuli voimaan v. 2021. Se korvaa aikaisemman direktiivin MDD:n (Medical Device Directive). MDR-asetus koskee lääkinnällisiä laitteita kuten infuusiopumppuja ja röntgenlaitteita mutta tietyin edellytyksin myös potilastietojärjestelmiä. (Fimea, 2021). Toinen keskeinen potilastietojärjestelmiin liittyvä säädös on Laki sosiaali- ja terveydenhuollon asiakastietojen sähköisestä käsittelystä (784/2021). Terveydenhuollon potilastietojen käsittelyyn

tarkoitettujen tietojärjestelmien olennaisten vaatimusten toteutumista valvoo Valvira (Valvira, 2022).

Kansallisten ja EU-säädösten vaatimustenmukaisuuden saavuttamisen tukena toimivat kansainväliset standardit. ”Standardit ovat julkaisuja, joihin on kirjattu yhteisesti sovittuja vaatimuksia, suosituksia tai vaikkapa ominaisuuksia tuotteille ja niiden valmistukselle tai testaukselle sekä järjestelmille tai palveluille.” (Suomen Standardoimisliitto SFS, n.d.)

Tässä esitellään kaksi potilastietojärjestelmän vaatimusmäärittelyyn liittyvää standardia: ISO/IEC 62304:2006 (2006). Medical device software – Software life cycle processes sekä IEC 82304-1(2016) Health software – Part 1: General requirements for product safety. Standardien kuvauksessa pyritään antamaan kuvan standardien sisällöllisistä vaatimuksista ja ohitetaan tarkat termien määrittelyt.

6.3.1 Medical device software – Software life cycle processes (IEC 62304)

Medical device software eli MDS-standardi koskee ohjelmiston kehittämistä ja ylläpitoa silloin, kun ohjelmisto itsessään on lääkinnällinen laite, sekä silloin, kun ohjelmisto on sulautettu tai muuten olennainen osa lääkinnällistä laitetta (ISO/IEC 62304/2006, s. 9). Standardissa kuvataan ohjelmiston kehittämisprosessin ja ylläpitoprosessin vaatimukset (ISO/IEC 62304/2006, s. 9). Standardin perusoletuksiin kuuluu, että ohjelmiston kehittämiseen ja ylläpitoon liittyy laadunhallintajärjestelmä ja riskienhallintajärjestelmä standardin ISO 14971 mukaisesti (ISO/IEC 62304/2006, s. 15). Lisäksi ohjelmiston valmistajan tulee määritellä ohjelmistolle turvallisuusluokka A, B tai C sen mukaan, kuinka vakavan terveyshaitan ohjelmisto voi aiheuttaa potilaalle, käyttäjälle tai muille ihmisille. A on lievin, C vakavin haittaluokka. Eri luokkia koskee hieman eri vaatimukset. (ISO/IEC 62304/2006, s. 15)

MDS-standardin alkupuoli koskee ohjelmiston kehitysprosessia, kuten kehityssuunnitelmaa, vaatimusmäärittelyä, arkkitehtuuria ja yksityiskohtaista suunnittelua. Loppupuoli koskee ohjelmiston ylläpitoa ja ongelmien ratkaisemista.

Kehitysprosessia koskevassa osuudessa standardi edellyttää, että ohjelmistolla on tietyt vaatimukset täyttävä kehityssuunnitelma. Valmistajan tulee laatia ohjelmiston laajuuden kannalta asianmukainen, kirjallinen kehityssuunnitelma / -suunnitelmat, jossa määritetään

kehittämisen elinkaarimalli tai kehittämisen prosessi ja vaiheistus. Kehityssuunnitelmassa tulee kuvata kehittämiseen liittyvät tehtävät eri vaiheissa, ominaisuuksien jäljitettävyys, konfigurointi, testaus ja riskienhallinta. Lisäksi on kuvattava kehityksessä ja ongelmien ratkaisussa käytettävät tukiohjelmistot elinkaaren eri vaiheissa. Kehityssuunnitelma tulee pitää ajan tasalla.

Ohjelmiston kehityssuunnitelman tulee noudattaa koko järjestelmän kehityssuunnitelmaa. Ohjelmiston kehityssuunnitelmassa tulee kuvata käytetyt standardit, metodit ja työkalut. Suunnitelmaan sisältyy mahdollisten sovellus- tai laiteintegraatioiden suunnitelma, verifioinnin (todentamisen) suunnitelma, riskienhallinta- ja dokumentaatiosuunnitelmat. Lisäksi suunnitellaan tarvittavat konfiguraatiot ja kehitysvälineet. (ISO/IEC 62304/2006, ss. 16–18)

Valmistajan on määritettävä ohjelmiston vaatimukset erillään mahdollisista järjestelmävaatimuksista. Ohjelmistovaatimuksiin tulee sisältyä mm. toiminnalliset ja suorituskykyvaatimukset, syötteiden ja tuotoksien kuvaukset, rajapinnat muihin järjestelmiin, herätteet ja hälytykset sekä turvallisuusvaatimukset. Ohjelmistovaatimuksia ovat myös koulutusta ja tarkkuutta vaativat käytettävyysasiat, vaatimukset tietokannalle ja ohjelman käsittelemälle datalle. Ohjelmistolta edellytetään myös asennus- ja hyväksyntävaatimuksia, käyttäjädokumentaatiota, ylläpitovaatimuksia käyttäjille sekä lainsäädännön asettamien vaatimuksien noudattamista. (ISO/IEC 62304/2006, ss. 18–20)

Valmistajan tulee luoda ohjelmistovaatimusten perusteella arkkitehtuurisuunnitelma, joka määrittelee ohjelmiston rakenteen ja komponentit. Valmistajan tulee dokumentoida sekä ohjelmiston ulkoiset että sisäiset, komponenttien väliset rajapinnat. Arkkitehtuurisuunnitelmassa tulee huomioida tekniset laitteisto- ja ohjelmistovaatimukset kuten tarvittavan muistin määrä tai näytön ominaisuudet. Arkkitehtuurisuunnitelmassa tulee huomioida myös riskien hallinta. (ISO/IEC 62304/2006, ss. 20–21)

Valmistajan tulee luoda ja dokumentoida yksityiskohtainen kehityssuunnitelma jokaiselle ohjelmistoon kuuluvalle sovellukselle tai sovelluksen osalle. Yksityiskohtaisen suunnittelun vaatimus koskee myös ohjelmiston ulkoisia ja sisäisiä rajapintoja. Yksityiskohtaisen

suunnittelun tulee noudattaa ohjelmiston arkkitehtuurisuunnitelmaa. (ISO/IEC 62304/2006, s. 21)

Kehitysprosessiin sisältyy myös varsinainen toteutustyö, sen seuranta, integraatioiden hallinta ja testaus. Ohjelmiston toteutukseen tulee liittyä hyväksyntäkriteerit ja kriteereiden arviointi- ja tarkistusmenettely. Hyväksyntäkriteerien tulee perustua ohjelmisto- ja riskienhallinnan vaatimuksiin. (ISO/IEC 62304/2006, ss. 21–22) Jos ohjelmistossa on integraatioita muihin sovelluksiin tai laitteistoihin, tarkistusmenettelyyn tulee sisältyä myös näiden integraatioiden toiminnan tarkistus ja testaus integraatiosuunnitelman mukaisesti. (ISO/IEC 62304/2006, ss. 22–23)

Järjestelmän testaamisen vaatimuksiin sisältyy testitapausten luominen, löydettyjen ongelmien ratkaisuprosessi ja uusintatestaus korjausten jälkeen. Testitapausten tulee olla asianmukaisia, kattavia, noudattaa vaatimusmäärityksiä ja niillä tulee olla selkeät hyväksymis-/hylkäämiskriteerit. Testitapaukset ja niiden tulokset tulee dokumentoida. (ISO/IEC 62304/2006, s. 24)

Kehitysprosessin vaatimuksia asetetaan myös ohjelmiston julkaisulle ja jakelulle. Ennen ohjelmiston julkaisua valmistajan tulee varmistaa, että ohjelmiston arviointi- ja testausvaihe on valmis sekä kehityksen kehitysjono suoritettu. Tiedossa olevien virheiden tulee olla dokumentoituja ja arvioituja.

Ohjelmiston jakelun ja toimituksen tulee olla luotettavaa ja toistettavissa. Julkaistut versiot tulee dokumentoida, kuten niiden valmistustavat. Ohjelmisto tulee voida arkistoida. (ISO/IEC 62304/2006, s. 25)

MDS-standardin loppupuoli koskee ohjelmiston ylläpitoa, riskien hallintaa, konfiguraatioiden hallintaa ja ongelmien ratkaisuprosessia. (ISO/IEC 62304/2006, ss. 26–33). Niitä ei kuvata tässä tarkemmin.

6.3.2 Health software – Part 1: General requirements for product safety (IEC 82304-1)

Standardi 82304 on kaksiosainen; IEC 82304-1:2016 Health software – Part 1: General requirements for product safety käsittelee nimensä mukaisesti terveyssovellusten yleisiä

turvallisuusvaatimuksia ja toinen osa ISO/TS 82304 -2:2021 Health software – Part 2: Health and wellness apps – Quality and reliability käsittelee mobiileja terveys- ja hyvinvointisovelluksia. (IEC 82304/1:2016)

Koska opinnäytetyön liittyy ammattilaisille tarkoitettua työasemasovellus, tässä käsitellään vain ensimmäistä osaa. Terveyssovelluksella tai terveysohjelmistolla tarkoitetaan tässä yhteydessä terveydenhuollon ammattilaisten tai kansalaisten käyttämää sovellusta tai ohjelmistoa, joka on kehitetty ihmisen terveystietojen hallinnointia varten.

Standardi General requirements for product safety koskee terveyteen liittyviä ohjelmistotuotteita, jotka on suunniteltu toimimaan yleisimmillä sovellusalustoilla, eli työasemasovelluksia. Standardissa painotetaan valmistajille asetettavia vaatimuksia. Standardi kattaa terveysohjelmiston koko elinkaaren suunnittelusta ja kehityksestä validointiin (kelpuutukseen) sekä ylläpitoon ja hävittämiseen asti (IEC 82304/1:2016 s 6).

Yleisinä vaatimuksina terveysohjelmiston valmistajan tulee määrittää ja dokumentoida ohjelmiston käyttötarkoitus, käyttäjäprofiili ja käyttöympäristö. Lisäksi valmistajan tulee laatia ja dokumentoida ohjelmiston riskiarvio ja riskien hallintasuunnitelma. (IEC 82304/1:2016 ss. 10–11)

Standardissa on useita ohjelmiston käyttöön liittyviä vaatimuksia. Ohjelmiston valmistajan tulee määrittää ja dokumentoida miten ohjelmistoa on tarkoitus käyttää sekä käyttöä koskevat vaatimukset. Valmistajan on määritettävä liittymä-/rajapintavaatimukset mukaan lukien käyttöliittymän vaatimukset. Liittymillä ei tässä tarkoiteta teknisiä rajapintoja vaan teknisiä tarpeita, kuten esimerkiksi ”Näytöllä olevien tietojen tulee olla luettavissa 3 metrin etäisyydeltä päivystysyksikössä”. Valmistajan tulee määrittää ja dokumentoida vaatimukset, miten ohjelmisto tulee suojata muilta sovelluksilta, jotka käyttävät samaa laiteympäristöä, ja niiden tahattomalta tai tahalliselta vaikutukselta. (IEC 82304/1:2016 s. 11)

Valmistajan tulee määrittää ja dokumentoida tietosuoja- ja tietoturvallisuusvaatimukset kuten käyttövaltuudet, käyttäjän tunnistaminen, terveystietojen eheys ja oikeellisuus sekä suojautuminen pahantahtoisilta aikeilta. Määritys- ja dokumentaatiovaatimus koskee myös sovelluksen mukana tuleva dokumentaatiota kuten käyttäjän oppaita. (IEC 82304/1:2016 s. 11)

Valmistajan tulee määrittää ja dokumentoida ohjelmiston tarvitseman käyttötuen vaatimukset, kuten

- sovelluspäivitykset, jotka säilyttävät tietojen eheyden ja yhteensopivuuden aikaisempien versioiden kanssa,
- palautukset edelliseen versioon päivityksen jälkeen,
- oikea-aikaiset turvapäivitykset ja -korjaukset,
- ohjelmiston jakelu- ja toimitusmekanismit, jossa asennuksen eheys on varmistettu,
- tietojen poistamisen aktiivikäytöstä tai tietojen pois pyyhkimisen sekä
- tietojen siirron ja/tai säilyttämisen. (IEC 82304/1:2016 s. 11)

Ohjelmiston valmistajan tulee varmistaa, että terveysohjelmiston käyttöä koskevat vaatimukset on kuvattu jo järjestelmän käyttövaatimuksissa niin, että valmistaja pystyy osoittamaan vaatimusten täyttymisen. Valmistajan tulee kirjata vaatimusten verifiointi eli todentaa, että tuote vastaa vaatimuksia. Valmistajan tulee varmistaa, että käyttöä koskevat vaatimukset myös päivitetään tarpeen mukaan esimerkiksi käytön vaatimusten verifioinnin tai validoinnin seurauksena. (IEC 82304/1:2016 s. 11-12)

Ohjelmiston valmistajan tulee määrittää ja dokumentoida terveysohjelmiston järjestelmävaatimukset. Vaatimukset sisältävät ohjelmiston tarkoitetun käytön vaatimukset sekä vaatimukset yhteen toimivuudesta, lokalisoinnista ja tuesta määritetyille kielille. Ne sisältävät riskienhallintasuunnitelman, joka perustuu alkuperäiseen riskien arviointiin. Vaatimuksiin kuuluvat myös käyttöliittymämäärittelyt sekä terveysohjelmiston ohjelmisto- ja laitealustoja koskevat vaatimukset, jotta ohjelmisto toimii odotetulla tavalla odotetulla kuormituksella ja vaaditulla suorituskykytasolla. Vaatimuksiin sisältyy myös turvallisuusvaatimuksia, miten turvatoimia ajastetaan sekä miten turvallisuuden vaarantuminen havaitaan, tunnistetaan, rekisteröidään ja torjutaan järjestelmän normaalin käytön aikana. Turvallisuusvaatimuksiin sisältyy myös ydintoiminnallisuuksien suojaaminen tilanteessa, jossa järjestelmän turvallisuus on vaarantunut, sekä järjestelmän tietojen ja konfiguraation palauttamistapa, jonka tekee auktorisoitu käyttäjä.

Ohjelmiston valmistajan tulee varmistaa, että terveysohjelmiston järjestelmävaatimukset ovat yksilöityjä, keskenään ristiriidattomia, selkeitä ja kuvattu niin, että ne toimivat järjestelmätestien testauskriteereinä. Vaatimusten verifiointi tulee kirjata. Valmistajan tulee varmistaa, että järjestelmävaatimukset myös päivitetään tarpeen mukaan esimerkiksi käytön vaatimusten verifioinnin tai validoinnin seurauksena.

Terveysohjelmiston elinkaarivaatimuksissa on tiedostettu, että niiden kaikkien osien kannalta ei aina ole mahdollista noudattaa standardoituja prosesseja esimerkiksi vanhojen osajärjestelmien vuoksi. Tällöin valmistajan tulee tehdä riskienhallintasuunnitelma ns. jäännösriskien osalta. (IEC 82304/1:2016 ss. 12–13)

Validoinnilla eli kelpuutuksella varmistetaan, että valmistetaan aiotun tarkoituksen mukaista tuotetta. Valmistajan tulee tehdä validointisuunnitelma, jossa määritetään validoinnin kohde ja siihen soveltuvat validointimenetelmät sekä kuvataan mahdolliset rajoitteet, joilla voi olla vaikutusta validointiin. Valmistajan tulee valita validointiin asianmukaiset menetelmät, työnkulut ja hyväksymiskriteerit. Valmistajan tulee määrittää myös validointia tukevat ympäristötekijät kuten käyttöjärjestelmä, laite- ja ohjelmistoympäristö, joita validointi vaatii.

Ennen validointia valmistajan tulee määrittää myös validoinnin tekevän tiimin pätevyysvaatimukset sekä mahdollinen koulutustarve. Koulutus tulee olla suoritettu ennen validointia. Validointitiimin riippumattomuus suunnittelutiimistä tulee myös sisällyttää suunnitelmaan. Valmistajan tulee vahvistaa valmius validointiin, kun validointisuunnitelma on laadittu, pätevä validointitiimi on asetettu ja ohjelmisto on toimitusvalmis validoitavilta osiltaan. (IEC 82304/1:2016 s. 13)

Tiimi validoi ohjelmiston sen suunnitellussa käyttöympäristössä validointisuunnitelman mukaisesti. Jos validointisuunnitelmasta on tarpeen poiketa, asia tulee perustella validointiraportissa. Jos ohjelmistosta löytyy poikkeavuuksia, ne ratkaistaan erillisen ongelmaratkaisuprosessin mukaisesti. Jos ongelmanratkaisuprosessi johtaa ohjelmiston muuttamiseen, validointi tehdään riittävässä laajuudessa uudestaan.

Validointitiimi tekee ohjelmiston validointiraportin, jossa kuvataan todisteellisesti, miten käyttövaatimukset ovat yhteydessä validoinnin tuloksiin, miten ohjelmisto täyttää

käyttövaatimukset sekä kuvataan ohjelmiston hyväksyttävissä oleva jäännösriski. (IEC 82304/1:2016 ss. 13–14)

Muita vaatimuksia, joita standardi 82304 käsittelee ovat terveysohjelmiston tunnistetiedot, kuten ohjelmiston ja sovellusten nimi- ja versiotiedot sekä vaadittavaa dokumentaatio, jonka tulee sisältää vähintään valmistajan yhteystiedot, ohjelmiston tunnistetiedot, tarvittavat käyttöohjeet ja tekninen kuvaus. Dokumentaatiossa tulee kuvata myös käyttäjiltä vaadittava perehtyneisyys ja vaatimukset laite- ja ohjelmistoympäristölle. Dokumentaation tulee olla käyttäjien ymmärtämässä muodossa. (IEC 82304/1:2016 ss. 14–15)

Käyttöohjeiden tulee sisältää kaikki käyttöön tarvittava tieto mukaan lukien tarvittavat asennusohjeet. Ohjeiden tulee sisältää tarkoitetun käytön kuvaus, lyhyt ohjelmiston ja sen keskeisimpien ominaisuuksien kuvaus, turvaohjeet ja mahdolliset rajoitukset ja varoitukset. Käyttöohjeisiin tulee sisältyä myös asennustiedot kuten kuka asennuksen voi tehdä, asennuksen aikaiset turvaohjeet, tiedot riippuvuuksista muihin sovelluksiin, konfiguroinnista, rajapinnoista, sovellusalustoista sekä varsinaiset asennusohjeet. Ohjeiden tulee sisältää ohjeet mm ohjelmiston käynnistämisestä, käytöstä, ohjelmiston virheilmoituksista ja huomautuksista, ohjelmiston sulkemisesta sekä mitä pitää huomioida, kun ohjelmiston käyttö lopetetaan. Lisäksi tekniselle kuvaukselle on omat vaatimuksensa. (IEC 82304/1:2016 ss. 15–18)

Standardissa on vaatimuksia myös ohjelmiston ylläpitovaiheelle. Ohjelmiston uudelleenvalidointi tulee tehdä, jos ohjelmistoa on ylläpidon aikana jouduttu muokkaamaan merkittävässä laajuudessa. Myös validointisuunnitelma tulee päivittää vastaavasti.

Ohjelmiston valmistajan tulee tiedottaa käyttäjiä ja asiaan liittyviä organisaatioita ilmenneistä turvallisuusongelmista ja -haavoittuvuuksista sekä turvallisuuteen vaikuttavista säädös- ja viranomaisohjeistusmuutoksista ylläpidon aikana. Ohjelmiston valmistajan tulee tiedottaa ohjelmiston uusista versioista, sen uusista ominaisuuksista ja korjatuista virheistä sekä muutosten mahdollisista vaikutuksista tietosuojaan tai -turvaan, tunnistukseen tai dokumentaatioon. Samoin ohjelmiston käytön päättämisen ohjeistus ja asennuksen poistamisen ohjeistus kuuluu ylläpitovaiheen vaatimuksiin. (IEC 82304/1:2016 ss. 18–19)

6.4 Vaatimusmäärittely ketterässä ohjelmistokehityksessä

Ketterässä ohjelmistokehityksessä myös vaatimusmäärittely tehdään ketterien menetelmien mukaisesti. Vaatimusmäärittely sulautuu osaksi kehitysjonon rakentamista. (DevOps, n.d.) Vaatimukset voidaan kerätä ylätasolla kehitysprosessin alussa. Sen jälkeen vaatimusmäärittelyä tarkennetaan ja täydennetään iteratiivisesti kehitysprosessin aikana. (DevOps, n.d.) Määrittelylle varataan myös oma sprinttiaikansa. (Peters, S., n.d.)

Vaatimusmäärittelyä käytetään koko kehityskaaren ajan verifioinnissa ja validoinnissa (Peters, S., n.d.). Dokumentaation asema ei ole yhtä keskeinen kuin vesiputousmallissa, mutta riittävää dokumentaatiota pidetään edelleen arvokkaana (Agile Alliance, n.d.).

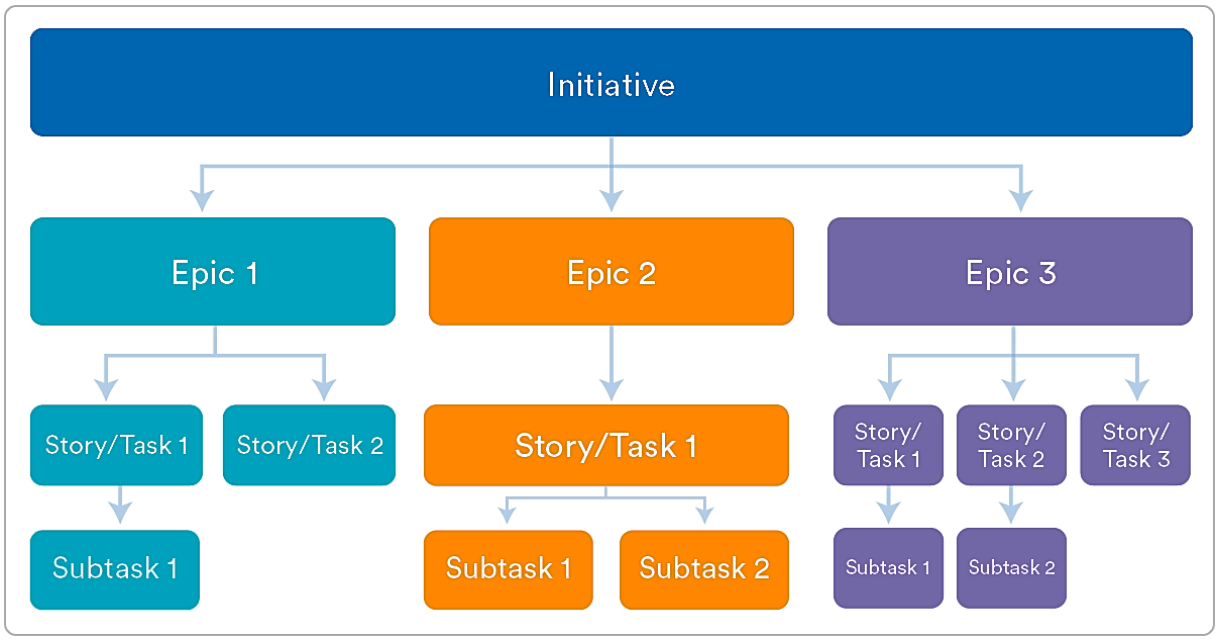
Ohjelmistoja tuottavalla yrityksellä tulee olla ohjelmiston kehityssuunnitelma. Ohjelmiston kehityssuunnitelma kuvaa ohjelmiston ja siihen kuuluvien sovelluksien suunnittelu- ja kehitystoiminnan vaiheet. Suunnitelma määrittelee myös kuka vastaa kunkin vaiheen toteutuksesta. (Scaled Agile, Inc., 2021 c)

Tuoteomistajalla tulee olla näkemys tuotteen kehityssuunnasta. Vaatimusmäärittely ja tuotteen suunnittelu tehdään yhteistyössä sidosryhmien kuten asiakkaan / käyttäjien, designerin, tuoteomistajan ja tuotekehityksen kanssa. (Peters, S., n.d.)

Ketterissä menetelmissä kehitysjono eli backlog on hierarkkinen ja käytetyt tasot vaihtelevat eri lähestymistavoissa. Ylimmällä tasolla ovat initiativet (aloitteet), ylätason kuvaukset sovelluksesta tai järjestelmästä. Initiativet koostuvat epiceistä (kertomuksista), jotka ovat yhdelle tiimille tarkoitettuja kuvauksia, joiden toteuttaminen vie kuukaudesta kolmeen kuukauteen. (Rehkopf, M., n.d.-b)

Epicien alla ovat storyt (käyttäjätarinat), joissa epicit on jaettu sprinttiin hyvin mahtuviksi kokonaisuuksiksi, esimerkiksi muutaman henkilötyöpäivän kokoisiksi tehtäviksi. User storyjen alapuolella ovat subtaskit (tehtävät), joiden toteuttaminen vie aikaa parista tunnista pariin päivään. (Rehkopf, M., n.d.-b)

Kuva 17 Agilen backlogin rakenne (Rehkopf, M., n.d.-b)



Kuva 17 havainnollistaa kehitysjonon hierarkkista rakennetta. Kun tuoteomistaja on organisoinut, että vaatimukset on kerätty, priorisoitu ja kuvattu kertomuksiksi (epics) ja käyttäjätarinoiksi (stories), kehitystiimi ja scrum master tekevät työmääräarviot ja poimivat sopivan määrän tarinoita kehitysjonolleen (backlog). Tiimi pilkkoo tarinat tarvittaessa pienemmiksi. Storeille muodostetaan myös taskit. Sen jälkeen työ jatkuu varsinaisella ohjelmoinnilla ja siihen liittyvillä tarkistus- ja testaustehtävillä. (Rehkopf, M., n.d. -a)

Tuotekehityksen sekä tuotteen toimituksen ja ylläpidon yhteistyö on tärkeää. Ylläpito saa arvokasta tietoa ja tukea tuotekehitykseltä. Käytössä olevan tuotteen asiakaspalautetta hyödynnetään jatkokehityksessä. (DevOps, n.d.)

6.4.1 User story mapping

Agilessa vaatimusmäärittelyn pilkkomisessa on myös riskejä. Kun vaatimukset pilkotaan eri puolille kehitysjonoa, on vaarana, että näkemys kokonaisuudesta ei välity tuotekehittäjille. Yhtenä ratkaisuna tähän on kehitetty User Story Mapping -menetelmä. (Patton, J., 2022)

Story mapping voidaan rakentaa sähköisesti tai post-it-lapuilla ja se tehdään yleensä ryhmissä. Ensin kirjoitetaan lyhyt tuotteen tai ominaisuuden kuvaus, (big story) ja annetaan

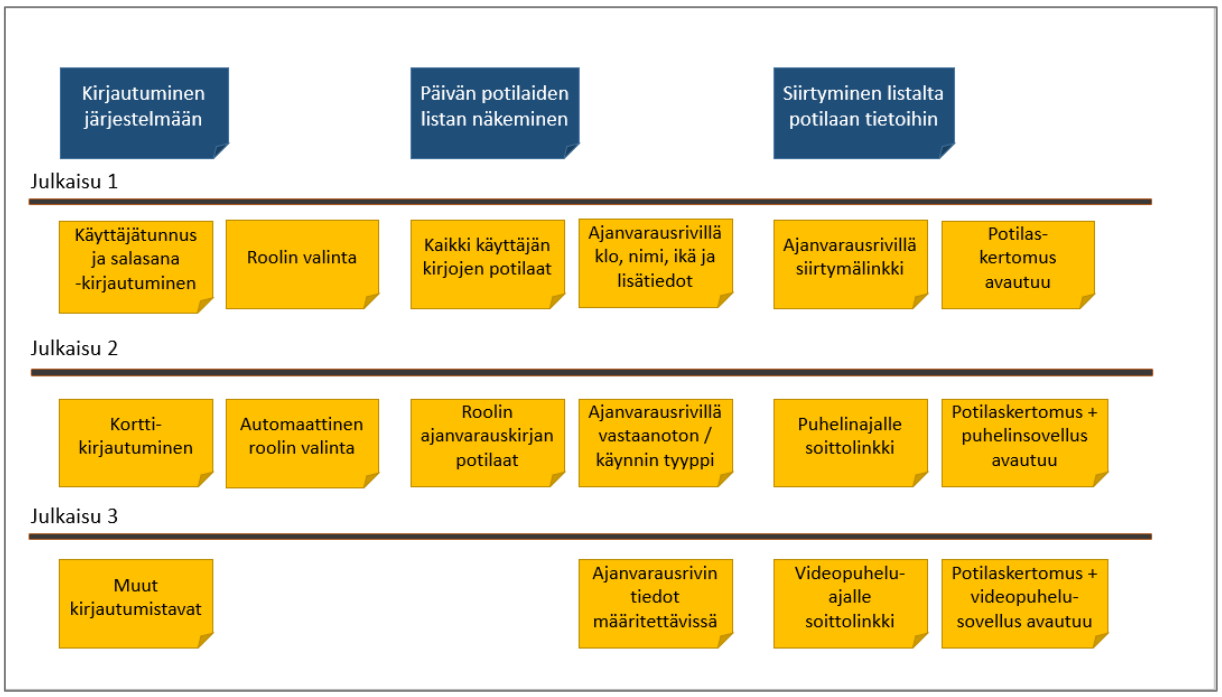
tuotteelle nimi, (what), nimetään käyttäjä, (who) ja kerrotaan mitä hyötyä tuotteesta on omalle organisaatiolle ja asiakkaille, (why). (Patton, J., 2015)

Sen jälkeen rakennetaan yleiskuva, (big picture) kirjoittamalla se lapuille riviin vasemmalta oikealle. Kuvataan ylätasolla tuotteen koko työnkulku, mitä tuotteen menestyksen kannalta kriittisin käyttäjäryhmä tekee, mitkä ovat keskeisimmät toiminnallisuudet. Lopuksi lisätään muita keskeisiä käyttäjäryhmiä. (Patton, J., 2015)

Tutkimisvaiheessa (explore) kukin toiminnallisuus pilkotaan pienemmiksi tehtäviksi sen alapuolelle. Tässä vaiheessa ideoidaan käyttöliittymien piirteitä, uusia tapoja toteuttaa toiminnallisuus, käyttäjän toiminnan variaatioita ja poikkeuksia sekä muita yksityiskohtia. Toiminnallisuuden alle lisätään uusi lappuja, jaetaan niitä useammaksi lapuksi, järjestetään lappuja tärkeimmät ylimmäksi. (Patton, J., 2015)

Julkaisun suunnitteluvaiheessa, (slice out viable releases) laput ryhmitellään pienimmiksi mahdollisiksi julkaistaviksi kokonaisuuksiksi. Kullekin julkaisulle kerrotaan sen tavoite ja vaikutus kokonaisuuteen. Julkaisulle määritetään, miten sen onnistuminen voidaan mitata esimerkiksi käyttäjän toiminnasta. (Patton, J., 2015)

Kuva 18 Esimerkki suunnitteluvaiheen User story mappingista



Kuva 18 esimerkki voisi olla taululle liimalapuista tehty suunnitelma. Viimeisessä vaiheessa, julkaisustrategian suunnittelussa, (slice out a development strategy), suunnitellaan käyttäjätarinoiden uudelleenmäärittelyyn tarvittava työ, käydään kehittäjien ja testaajien kanssa työpajassa läpi työn yksityiskohdat ja sovitaan hyväksyntäkriteerit, suunnitellaan kehitys- ja testaustyö sekä miten verifioidaan luotu toimiva julkaisu. (Patton, J., 2015)

Jokaisen julkaisun työvaiheet jaetaan kolmeen osaan. Ensimmäinen osa on yksinkertaisin mahdollinen toimiva versio sovelluksesta (walking skeleton). Sen avulla voi esitellä asiakkaille sovelluksen ominaisuuksia. Sen avulla pääsee myös jo aloittamaan sovelluksen suorituskyvyn ja skaalautuvuuden testausta. (Patton, J., 2015)

Toisessa vaiheessa kehitetään julkaisun päätoiminnallisuuksia, testataan niitä yhteistyössä asiakkaiden kanssa ja tarkennetaan niitä palautteen perusteella. Suorituskyvyn ja skaalautuvuuden testaus jatkuu myös. (Patton, J., 2015)

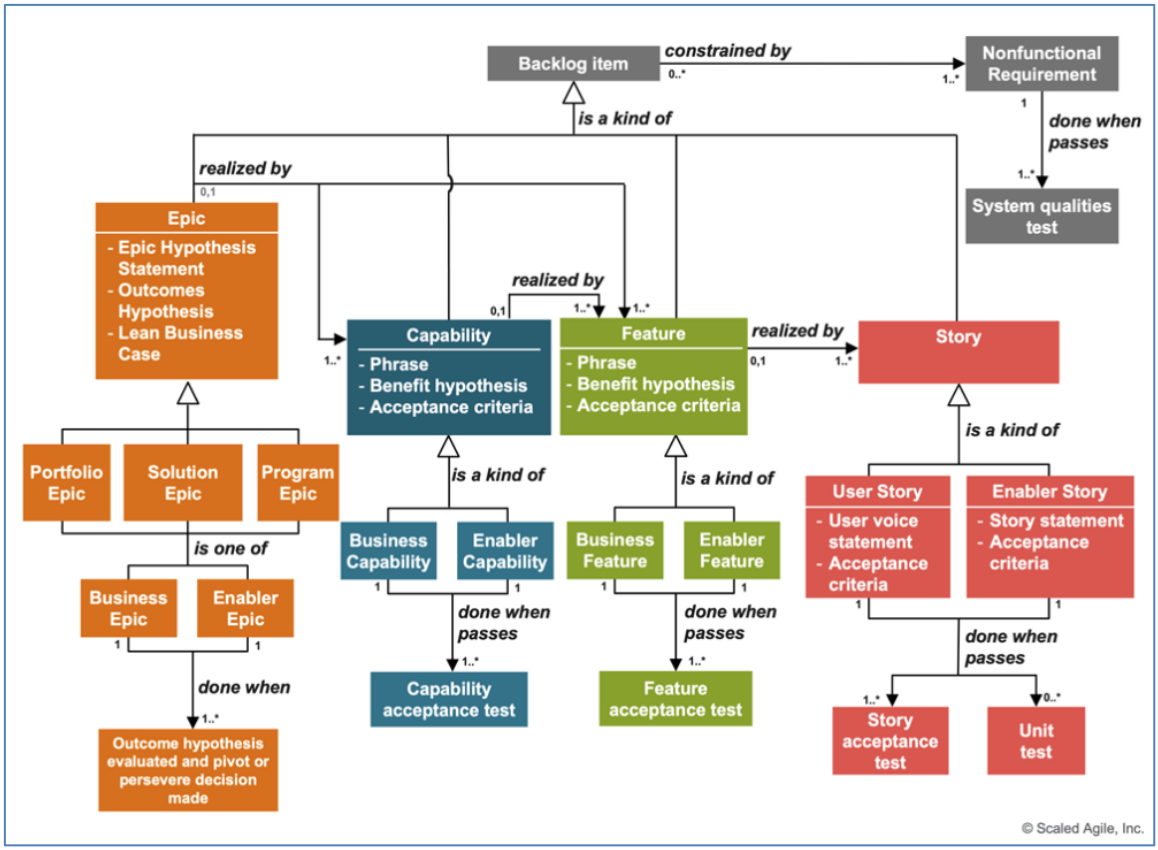
Kolmannessa vaiheessa sovellus viimeistellään julkaisua varten. Sovelluksen julkaisuvalmiutta arvioidaan vertaamalla sitä julkaisustrategian tavoitteisiin. (Patton, J., 2015)

6.4.2 Vaatimusten hallintamalli SAFen mukaan

SAFen vaatimusten hallintamalli SAFe Requirements Model (© Scaled Agile, Inc.) on pitkälle kehitetty versio ketteristä menetelmistä. SAFe-mallissa järjestelmän vaatimukset kuvataan muutamien pääelementtien kautta: Kertomukset (Epics), Kyvykkyydet (Capabilities), Ominaisuudet (Features) ja Tarinat (Stories) sekä Ei-toiminnalliset vaatimukset (Non-functional requirements). (Scaled Agile, Inc., 2021 k).

Kuva 19 malli näyttää monimutkaiselta, mutta sen keskeisimmät elementit ovat tuttuja ketterän kehitysjonon osia. SAFessa kehitysjonon elementtien sisällöt on määritelty tarkasti.

Kuva 19 SAFe Requirements Model. (Scaled Agile, Inc., 2021 k)



Mallissa Kertomus (Epic) on laajin ylätason kokonaisuus, joka voi kattaa kokonaisen sovelluksen tai merkittävän osan siitä. Kertomukselle voidaan osoittaa myös budjetti. Sen vuoksi epicit vaativat tuotehallinnan hyväksynnän. Kertomukselle määritetään myös MVP, Minimum Viable Product, pienin toimitettavissa oleva tuote. Kertomuksen kuvaus (hypothesis statement) kirjoitetaan määrämuotoiselle pohjalle, joka toimii sekä liiketaloudellisen (business case) että tuotteen (outcome) arvion perusteluna. (Scaled Agile, Inc., 2021 l) Esimerkki kertomuksesta voisi olla kuvitteellinen Sydänpotilaan seuranta -sovellus.

Ominaisuus (Feature) on toiminnallinen kokonaisuus, joka täyttää jonkun sidosryhmän tarpeen. Jokainen ominaisuus sisältää lyhyen kuvauksen sisällöstä (phrase), hyötyhypoteesin (benefit hypotesis) ja hyväksyntäkriteerit (acceptance criteria). Featuret laaditaan niin, että ne voidaan toimittaa yhdellä julkaisujunalla (Agile Release Train) yhden inkrementin aikana. (Scaled Agile, Inc., 2021 m)

Kyvykkyys (Capability) on piirre tai toiminnallisuus, joka tyypillisesti kattaa useamman julkaisujunan alueen. Siksi kyvykkyys jaetaan useammaksi featureksi. Muutoin kyvykkyyden sisältö rakentuu samalla tavalla kuin ominaisuuden (featuren) sisältö. Kaikkiin ominaisuuksiin ei kuitenkaan liity kyvykkyyttä. (Scaled Agile, Inc., 2021 m) Esimerkki kyvykkyydestä voisi olla seuraava: ”Potilaan tulee nähdä mobiilisovelluksen kautta samat tallennetut verenpaineenmittaustiedot, jotka ammattilainen näkee sovelluksen pilvipalvelun kautta”. Tämä jaettaisiin useampaan ominaisuuteen mobiilisovelluksen julkaisujunaan ja pilvipalvelujen julkaisujunaan. (Scaled Agile, Inc., 2021 m) Esimerkki itsenäisestä featuresta voisi olla ”Käyttäjä näkee verenpainemittausten ylä- ja alapaineen trendit valitsemallaan aikavälillä, kuten viikko, kuukausi tai vuosi”.

Storyt (tarinat) ovat lyhyitä, yksinkertaisia toiminnallisuuksien kuvauksia, jotka on yleensä kerrottu käyttäjän näkökulmasta. Tarinat on mitoitettu siten, että ne voidaan toteuttaa yhdessä iteraatiossa. Siten ne sopivat inkrementaaliseen kehittämissykliin. (Scaled Agile, Inc., 2021 n) Esimerkki tarinasta olisi ”Hoitajana haluan voida muokata mallipohjasta luotua verenpaineenmittausohjetta, jotta voin ohjata ja neuvoa potilaita yksilöllisesti”.

Tarinat tarjoavat sopivasti tietoa sekä liiketoiminta- että teknisille ihmisille, jotta he ymmärtävät tarkoituksen. Tarinan yksityiskohtia lykätään, kunnes tarina on valmis toteutettavaksi. Hyväksymiskriteerien ja -testien ansiosta tarinat tarkentuvat, mikä auttaa lopputuloksen laadun varmistamisessa. (Scaled Agile, Inc., 2021 k)

Enablerit eli mahdollistajat ovat tarinoiden (stories), ominaisuuksien (features) ja Kyvykkyyksien (capabilities) tyyppejä, jotka tukevat esimerkiksi toiminnallisten tai arkkitehtuurin vaatimusten toteuttamista. (Scaled Agile, Inc., 2021 o) Mahdollistajatarina (Enabler story) voisi olla esimerkiksi ”Käyttöliittymän fonttiperheen valinta”.

Ei-toiminnalliset vaatimukset (Non-functional requirements, NFRs) koskevat järjestelmän ominaisuuksia kuten skaalautuvuutta, suorituskykyä tai tietoturvaa. Ne asettavat usein myös rajoituksia järjestelmän suunnittelulle. NFR:t ovat yhtä kriittisiä vaatimuksia kuin toiminnallisetkin ja niiden avulla varmistetaan järjestelmän tehokkuus ja käytettävyys. Niiden tulee olla myös sopivalla tasolla; ylimitoitetut vaatimukset tulevat kalliiksi, alimitoitetut vaatimukset voivat estää järjestelmän aiotun käytön. Esimerkiksi jos valitulla

teknologialla järjestelmän suorituskyky on suunniteltu parille tuhannelle käyttäjälle, ratkaisu voi olla liian kallis muutaman kymmenen käyttäjän ympäristöihin, mutta täysin alimitoitettu ja sen vuoksi hidas kymmenen tuhannen käyttäjän ympäristössä. (Scaled Agile, Inc., 2021 p)

SAFen vaatimusten hallintamalli SAFe Requirements Model kuvaa yksityiskohtaisesti miten, milloin ja millä tarkkuudella vaatimukset kuvataan kehitysjonolle. Se kuvaa miten vaatimuksia iteratiivisesti tarkennetaan ja miten vaatimuksia käytetään toteutusvaiheessa. Malli kattaa vaatimusten hyödyntämisen aina testausvaiheeseen asti. (Scaled Agile, Inc., 2021 k)

7 Vaatimusmäärittelyn prosessin kuvauksen tavoite ja tarkoitus

Opinnäytetyössä on tavoitteena kuvata vaatimusmäärittelyn prosessia ketterässä ohjelmistokehityksessä siten, että määrittelyn rooli nousee näkyviin ohjelmistokehityksen kokonaisuudesta. Vaatimusmäärittely on osa ketterää ohjelmistokehitystä, eikä se ole enää oma vaiheensa kuten vesiputousmallissa. Vaatimusmäärittely työtehtävänä vaatii määrittelytyön prosessin tuntemista. Sen vuoksi tässä opinnäytetyössä ja sen tuloksena syntyneessä aineistossa tarkastellaan vaatimusmäärittelyä juuri työtehtävän näkökulmasta. Tavoitteena on ollut luoda helppokäyttöinen ohjeistus määrittelytyötä tekevien henkilöiden perehdytykseen.

Materiaalissa nojataan teoriaosassa esiteltyihin ketterän ohjelmistokehityksen prosessimalleihin, erityisesti SAFeen. Lisäksi hyödynnetään teoriaosuudessa esiteltyjä asiantuntijatyön johtamiseen liittyvää lean-teoriaa sekä palvelumuotoilun ja kestävän kehityksen periaatteita. Materiaalia tullaan käyttämään Tietoevry Healthin määrittelytyötä tekevien asiantuntijoiden sisäisen koulutuksen tukena.

Opinnäytetyötä varten oli tehtävänä määritellä potilastietojärjestelmään erikoissairaanhoidon hoitoonpääsyn seurantaan liittyviä asioita. Tavoitteena oli, että konkreettinen ja dokumentoitu määrittelytehtävä täydentää prosessin kuvaamista teorian lisänä. Aihepiiriksi hoitoonpääsy valikoitui ajankohtaisuutensa vuoksi, koska uusi opas ilmestyi alkuvuodesta.

Määrittelyn kohteena oli kaksi eri kokonaisuutta. Ensimmäinen, pienempi kokonaisuus oli selvittää ja määritellä mitä muutostarpeita tämän vuoden alussa julkaistu uusi erikoissairaanhoidon hoitoonpääsyn seurannan ohje mahdollisesti tuo Lifecaren nykyversioon. Toinen, isompi tutkimis- ja määrittelytehtävä oli selvittää miten Lifecaressa erikoissairaanhoidon hoitoonpääsyn seuranta järjestetään, kun ajanvaraussovellus ja jononhallintasovellus uudistuvat täysin lähiversioissa.

Molemmat määrittelytehtävät ovat tehtävinä aika tavallisia mutta keskenään hyvin erilaisia. Ei ole ollenkaan harvinaista, että potilastietojärjestelmää koskevat viranomaisohjeistukset muuttuvat ja ohjelmiston ominaisuuksia muutetaan sen mukaan. Pienemmässä tehtävässä

oli kyse tästä. Sekään ei ole harvinaista, että ison järjestelmän osien määrittelyä tehdään lohkoittain niin, että kaikkia muita liittyviä osioita ei ole vielä määritelty kuin ylätasolla. Tämä oli isomman määrittelytehtävän lähtökohta. Kumpikin tehtävä poikkeaa tavanomaisista määrittelytyön prosessin kuvauksista. Opinnäytetyön toteutusosassa käytetään määrittelytehtäviä esimerkkeinä tuomassa konkretiaa määrittelytyön prosessiin.

Opinnäytetyön tuloksena on syntynyt diasarja Ketterä_määrittelyprosessi.ppt (Liite 2) sekä samanniminen tekstimuotoinen ohjeistus (Liite 3). Samansisältöiset ohjeistukset on muotoiltu niin, että ne toimivat uusien määrittelijöiden perehdytyksen tukimateriaalina. Diamateriaali on käyty läpi kokeneiden vaatimusmäärittelijöiden kanssa ja pyydetty heiltä palautetta siitä, miten se heidän arvionsa mukaan soveltuu määrittelijätehtäviin perehdytykseen. Tekstimuotoista materiaalia on arvioitu omana, itsenäisenä perehdytysmateriaalina samasta näkökulmasta. Läpikäynnistä ja materiaalista saatujen palautteen perusteella diasarjaa ja tekstidokumenttia on vielä korjattu.

Ohjeistus ei ole varsinainen koulutusmateriaali, koska aineistoa laadittaessa on keskitytty enemmän konkreettisen prosessin kuvaamiseen kuin siihen, miten materiaali parhaiten toimisi kouluttamisen näkökulmasta. Aihepiirin kouluttaja voi käyttää materiaalia perehdytyksen tukena harkintansa mukaan.

Opinnäytetyön toteutusosan tausta-aineisto koostuu päiväkirjamerkinnöistä sekä palautteista, joka on kerätty diasarjan ja tekstimuotoisen materiaalin arvioinneista. Aineistonhallintasuunnitelma on kuvattu Liite 1. Määrittelytehtävistä syntyneet varsinaiset vaatimusmäärittelydokumentit eivät sisälly opinnäytetyöhön.

8 Esimerkkimäärittelyjen prosessit

Tässä luvussa kuvataan miten kaksi opinnäytetyöhön liittyvää määrittelytehtävää käytännössä etenivät. Käytän esimerkeissä samaa jäsentelyä kuin määrittelyprosessin kuvausmateriaalissa, eli kuvaan määrittelytehtävien vaiheet samojen otsikoiden alla. Vaatimusmäärittelyjä tehdessäni pidin päiväkirjaa, jotta sain tallennettua työskentelyn vaiheita tarkemmalla tasolla kuin mitä tuotteiden kehityslistoille tallennetaan.

8.1 Vaatimusmäärittelyn käynnistyminen

Erikoissairaanhoidon hoitoonpääsyn seurantatuotteen kehittämisestä vastaava tuoteomistaja osoitti minulle vaatimusmäärittelytehtävän, koska asiaa koskevat viranomaisohjeet olivat juuri muuttuneet. Tehtävänä oli perehtyä juuri julkaistuun erikoissairaanhoidon hoitoonpääsyn seurannan oppaaseen ja määritellä tuleeko sen pohjalta uusia muutostarpeita Lifecaren nykyversioon.

Kun ensimmäinen tehtävä oli valmis, sain toisen, samaan aiheeseen liityvän määrittelytehtävän. Koska vaatimukset olivat nyt tuttuja ja koska uuden ajanvaraussovelluksen toteutus oli jo pitkällä, sain tehtäväkseni määritellä miten erikossairaanhoidon hoitoonpääsyn seurannan tarvitsemat ominaisuudet ratkaistaisiin tulevassa Lifecaren versiossa.

8.2 Vaatimusten kerääminen

Ensimmäisen määrittelyn pohjaksi vertasin uutta opasta edelliseen oppaaseen sivu sivulta ja kokosin kaikki eroavuudet erilliseen dokumenttiin. Sen jälkeen kävimme yhdessä toisen asiantuntijan kanssa ne kohta kohdalta läpi. Löysimme vain neljä muuttunutta kohtaa, joista kaksi vaativinta oli jo toteutettu Lifecareen asiakastoiveiden perusteella. Loput kaksi olivat suhteellisen pieniä toteutettavia.

Toista määrittelyä varten viranomaisvaatimukset olivat jo tiedossa. Aiheeseen liittyviä asiakkaiden vaatimuksia löytyi sekä tuotekehitystoivomuksien että virheilmoituksien listoilta.

Olin lisäksi parin viime vuoden aikana keskustellut aiheesta eri yhteyksissä asiakkaiden kanssa ja saanut käsityksen tämän hetken ongelmista.

Ensimmäiseksi selvitin oliko muihin hoitoonpääsyyn liittyviin keskeisiin sovelluksiin suunnitteilla lähiaikojen muutoksia ja jos oli, niin minkälaisia. Sain niistä tiedot varsin nopeasti. Sitten otin yhteyttä arkkitehtiin. Uutta ajanvaraussovellusta suunnitteleva arkkitehti oli miettinyt erästä tietokantarakennetta, josta voisi olla tässä hyötyä. Kävimme ideaa lyhyesti läpi.

Uuden ajanvarauksen erikoissairaanhoidon ominaisuuksia oli jo osittain määritelty. Tämä määrittely piti rakentaa niin, että se ei ole ristiriidassa aikaisempien määrittelyjen kanssa. Uudella ajanvaraussovelluksella on asiakasryhmä, jolle kehitettäviä ominaisuuksia on esitelty aikaisemminkin. Se oli minulle ykkösvaihtoehto asiakaspalautteen antajaksi. Toinen vaihtoehto oli kysyä muutamalta aihepiiriin perehtyneeltä asiakkaiden edustajalta olisivatko he kiinnostuneita kommentoimaan määrittelyä. Tiesin myös, että ennen asiakkaiden puheille lähtöä voisin käydä ideoita läpi oman talon asiantuntijoiden kanssa. Saisin heiltä apua myös määrittelyn käyttöliittymäsuunnitelmien havainnollistamisessa asiakkaille.

8.3 Vaatimusmäärittelyn tarkentaminen

Ensimmäisessä määrittelyssä tein kahdesta puuttuvasta piirteestä kehitysjonolle kaksi ominaisuutta (featurea), ensimmäisen koodistojen ylläpidolle ja toisen raporttien ylläpidolle. Molempien määrittelyksi riitti muutama muutaman lauseen pituinen kuvaus. Raporttiasiasta pidettiin vielä lyhyt palaveri, jossa tarkennettiin miten toteutus tehdään.

Toisessa määrittelytehtävässä kokosin käytettävissä olevan aineiston ja lähdin sen pohjalta luonnostelemaan Word-dokumenttia. Tietokantaratkaisua hahmottelin Excel-sovelluksella, jotta saisin tietosisällöt kohdalleen. Välillä konsultoin ajanvaraussovelluksen tuoteomistajaa sekä arkkitehtia eteninkö oikeaan suuntaan ja sain hyviä lisäevästyksiä molemmilta.

Kun tietosisältö alkoi olla suunniteltu, aloin hahmotella käyttöliittymien vaatimuksia. Mitä tietoja tarvittaisiin uuteen ajanvaraukseen, että käyttäjä pystyisi tekemään tarvittavat ajanvaraukset niin, että hoitoonpääsyn vaiheet olisivat mahdollisimman läpinäkyvät?

Millaisia näkymiä vastuukäyttäjä tarvitsisi tarkistaessaan kertyneiden hoitoonpääsytietojen tilannetta? Miten työnkuluista saataisiin käyttäjille optimaalisia? Näihin kysymyksiin pyrin löytämään vastauksia.

Kun keskustelin arkkitehdin kanssa tietokantaratkaisusta, selvisi, että tietojen haku tietokannasta tulisi olemaan nopeaa ja tehokasta. Minulla jäi kuitenkin kysymättä millainen palvelimien ja työasemien energiankulutus tulisi liittymään tähän ja koko uudistuvan ohjelmiston teknisiin ratkaisuihin. Ohjelmistoa uusitaan asteittain, sovellus sovellukselta, ja uusilla teknologioilla on uusia mahdollisuuksia vaikuttaa energiankulutukseen.

8.4 Valmis vaatimusmäärittely

Ensimmäisestä tehtävästä tein parin sivun määrittelydokumentin Wordilla tehtävänannon liitteeksi. Se katselmoitiin kahteen kertaan, ensin konsultoimani asiantuntijan kanssa ja sen jälkeen tuoteomistajan kanssa. Lisäksi liitin eroavaisuusdokumentin samalle tehtävänannolle.

Toisesta tehtävästä syntyi huomattavasti laajempi määrittelydokumentti. Määrittely sisältää tietokanta- ja käyttöliittymävaatimuksia sekä yksilöityjä vaatimuksia usealle sovellukselle. Määrittelyn alussa on myös tiivis johdanto-osa viranomaisvaatimuksista aihepiiriin tutustumisen helpottamiseksi.

Vaatimusmäärittely on katselmoitu asiantuntijaryhmässä nyt kertaalleen ja sitä pitää vielä täydentää. Ennen asiakkaille esittelyä siihen pitää mm. piirtää käyttöliittymäluonnoksia. Varsinaisen käyttöliittymäsuunnittelun tekee käyttöliittymäsuunnittelija mutta esimerkiksi asiakaspalavereiden havainnollistamistarpeisiin kuvia voivat piirtää muutkin suunnittelijat. Toinen vaatimusmäärittely on tällä hetkellä vielä kesken.

8.5 Erityistilanteita

Erityistilanteilla tarkoitetaan tässä käytännön määrittelytyössä vastaan tulevia määrittelytarpeita, jotka voivat poiketa paljonkin oppikirjaesimerkkien prosesseista. Esimerkiksi vaatimusmäärittelyyn voi liittyä paljon reunaehtoja. Tavallisimpia reunaehtoja

ovat olemassa olevat sovellukset tai määrittelyn kohteena olevan sovelluksen muut osat, jotka muodostavat rajoitteita. Määrittelyn liikkumavara saattaa olla niukka, jos sen tulee koskea vain yhtä sovellusta tai sen osaa.

Ensimmäisen määrittelytehtävän tapauksessa, jos muutostarve olisi ollut laajempi, määrittelyssä olisi helposti päädytty asettamaan vaatimuksia usealle sovellukselle. Kun reunaehtoja on paljon, yksi mahdollisuus on miettiä ratkaisua riittävän suuressa ryhmässä usean näkökulman kautta.

Toinen, päinvastainen määrittelyyn vaikuttava tekijä voi olla reunaehtojen ja riippuvuuksien puuttuminen. Määrittelyä tehtäessä ei ole vielä tietoa, miten jokin määrittelyyn liittyvä asia tullaan ratkaisemaan. Tämä oli lisähaasteena toisessa määrittelytehtävässä, koska erikoissairaanhoidon kaikkia ajanvarausominaisuuksia ei vielä ole määritelty. Ratkaisuna pyrin kuvaamaan hoitoonpääsyyn liittyvien vaatimusten lisäksi myös ne vaatimukset, jotka tämä määrittely asettaa myöhemmille määrittelyille.

9 Perehdytysmateriaalin kirjoittaminen ja testaaminen

Kirjoitin kaksi samansisältöistä perehdytysmateriaalia eri käyttötarkoituksiin. Diasarjamuotoisen perehdytysmateriaalin on tarkoitus toimia koulutustilanteessa koluttajan tukena. Tekstimateriaalin on tarkoitus toimia itseopiskelumateriaalina tai wiki-sivun pohjamateriaalina.

Diasarjassa on käytetty keskeisistä termeistä niiden englanninkielisiä nimityksiä, koska ne ovat alan ammattitermejä. Tekstimuotoisessa ohjeistuksessa samat termit on lisäksi suomennettu. Suomennokset eivät olisi mahtuneet dioihin ja olisivat voineet lisäksi herättää hilpeyttä.

9.1 Materiaalin näkökulmien ja rakenteen valinnat

Perehdytysmateriaalit on kirjoitettu vaatimusmäärittelijän tehtävän etenemisen näkökulmasta. Etenemisjärjestyksen mukaisesta kuvauksesta on ehkä eniten hyötyä alkuvaiheiden kuvauksessa. Tavoitteena on kuvata, miten vaatimusmäärittelyssä pääsee alkuun, mitä asioita ainakin on hyvä huomioida.

Keruuvaiheeseen valitsin kolme vähän erilaista näkökulmaa, joissa asiat tapahtuvat yhtä aikaa. Vaatimusten kerääminen, analysointi ja dokumentointi tapahtuvat alkuvaiheen jälkeen iteratiivisesti. Samoja työvaiheita toistetaan, kun siirrytään määriteltävästä sovelluksen lohkosta toiseen. Tavoitteena on kuvata vaatimusten keruussa tarvittavia erilaisia lähestymistapoja. Lähestymistavat ovat asiakaskommunikaatio, vaatimusten tunnistaminen ja vaatimusten analysointi.

Vaatimusten viennistä kehitysjonolle on olemassa paljon hyviä organisaatiokohtaisia ohjeistuksia, siksi kuvaan siitä vain keskeisimmät asiat. Vaatimusmäärittelyn iteratiivisesta tarkennuksesta, sen hyvistä puolista ja haasteista aina testausvaiheeseen asti, löytyy myös teoriaosuudesta kuvauksia, joita olen yrittänyt tiivistää tähän.

Vaatimusmäärittelyn päättämisvaiheen kuvauksessa pyrin tuomaan esille sen kaksijakoisuuden. Vaatimusmäärittely tulee saada sillä tasolla valmiiksi, että sovelluksesta on

olemassa yleiskuva. Toisaalta vaatimusmäärittelyn iteratiivisuuden vuoksi määrittelyä tarkennetaan koko kehitysvaiheen ajan. Eri määrittelytavat täydentävät ketterästi toisiaan.

Viimeistään vaatimusmäärittelyn päättämisvaiheessa vaatimusmäärittely katselmoidaan ja näin sen laatu pystytään varmistamaan. Lisäksi kuvasin muutamia käytännössä vastaan tulleita erityistilanteita.

9.2 Materiaalien palautteet

Pyysin kahta kokenutta vaatimusmäärittelijäkollegaa arvioimaan diasarjaa siitä näkökulmasta, soveltuuko se heidän mielestään uusien vaatimusmäärittelijöiden perehdytykseen. Kävimme diasarjan läpi tunnin mittaisessa palaverissa, jonka päätteeksi pyysin heiltä palautelomakkeella (Liite 4) yleisarvioinnin ohjeistuksen käyttökelpoisuudesta sekä parantamisehdotuksia.

Pyysin kahta muuta kokenutta vaatimusmäärittelijäkollegaa arvioimaan tekstimateriaalia siitä näkökulmasta, soveltuuko se heidän mielestään uusien vaatimusmäärittelijöiden perehdytykseen itseopiskeluna. Pyysin myös heiltä palautelomakkeella (Liite 4) yleisarvioinnin ohjeistuksen käyttökelpoisuudesta sekä parantamisehdotuksia.

9.2.1 Diasarjan palautteet

Kummankin arvioijan mielestä materiaali sopii tarkoitukseensa erittäin hyvin. Kysymykseen, miten materiaalia voisi mielestäsi parantaa, he vastasivat seuraavasti:

Palaute 1

> Mielestäni materiaalissa on pääasiat huomioitu ja olisin itse hyötynyt aloittaessani tällaisesta läpikäynnistä ihan yleisellä tasolla. Olisi selkiyttänyt sitä, mitä minulta vaaditaan ja mikä on minun vastuuni ja työ tuotekehitysprosessissa. Toki yrityskohtaisia yksityiskohtia on vaikea sisällyttää yleiseen materiaaliin, joten tarkennuksia varmaan tarvitaan tapauskohtaisesti.

Ehkä olisi voinut olla myös mietitty jotain yleisiä sudenkuoppia. Jotain esimerkkejä, mitä tulee mieleen: Tehdään tekniikan ehdoilla eikä aidosti käyttäjille. Keskitytään vain omaan määriteltävään tuotteeseen eikä huomioida riippuvuuksia muihin tuotteisiin, mikä voi näkyä käyttäjille.

Palaute 2

Materiaali palvelee sekä uusien määrittelytyötä tekevien henkilöiden perehdyttämistä että kertauksena ja keskustelun pohjaksi kokeneemmille määrittelytyön tekijöille.

Riippuvuuksien havainnoimisesta, kuvaamisesta ja seurannasta voisi kertoa materiaalissa lisää. Varsinkin isoissa järjestelmissä riippuvuuksista aiheutuu monesti yllätyksiä ja ongelmia.

Tuotekehityksen aikaiset demot – sekä sisäiset että asiakkaille suunnatut – olisi hyvä olla myös mukana. Iteratiivisen kehityksen mukaisesti usein toistuvissa demoissa saadaan arvokasta tietoa siitä, että ollaan menossa oikeaan suuntaan. Demot voivat sisältää paitsi toteutuksen demoamista, myös määrittelyn läpikäyntiä.

9.2.2 Tekstimateriaalin palautteet

Toisen arvioijan mielestä materiaali sopii tarkoitukseensa hyvin ja toisen mielestä erittäin hyvin. Kysymykseen, miten materiaalia voisi mielestäsi parantaa, he vastasivat seuraavasti:

Palaute 3

Aineistoin yleisluontoisuuteen voisi kiinnittää vielä enemmän huomiota esim. asiakkaasta puhuessa pitää muistaa, että asiakas voi olla sisäinen tai ulkoinen. Ennen määrittelyprosessin aloittamista tehtävänannon pitää olla selvä ja tähän kannattaa käyttää sopivasti aikaa. Tehtävänannon sekavuus voi olla ensimmäinen kohta, missä voidaan mennä ”ojaan”. Määrittelyprosessin

aikana voi tulla eteen yllätyksiä ja se on täysin normaalia, tällöin kommunikaation tärkeys osapuolien välillä nousee esiin.

Toisinaan yllätyksen tullessa vastaan, pitää kääntää suuntaa tai joskus jopa kehitys keskeytyy ja näitä ei tule pelätä. Ohjeessa voisi ehkä hieman enemmän siitä, että määrittelyä on monenlaista. Liiketoiminnallinen määrittely, toiminnallinen määrittely, tekninen määrittely. Isossa kokonaisuudessa näiden tekemiseen voi tarvita useita henkilöitä ja pienessä kehityksessä yksi henkilö voi hoitaa kaiken.

Määrittelyssä on hyvä sopia rajauksista, kuten dokumentissa mainittiin. Työmäärä arvioissa on hyvä tuoda selkeäsi esiin mitä on määritelty toimivaksi ja mitä ei ole määritelty. Tämä helpottaa ristiriitatilanteissa keskustelun ohjaamista oikeaan suuntaan. Joskus asiakkaan oletus ja toimittajan oletus eivät ole sama.

Kaiken kaikkiaan ohje on hyvä perehdytysmateriaali uudemmille määrittelijöille ja muistutuksena myös kokeneemmille. Näen dokumentin hyödyllisenä ja minun mielestä se on hyvä lisä perehdytysmateriaaleihin.

Palaute 4

Luin tämän ajatuksella ja en keksi mitään parannettavaa. Tämä oli erittäin hyvä yleinen prosessikuvaus ja kuten tekstissä mainittiin, niin usein tai oikeastaan aina tähän liittyy käytännöt, joita määrittelytyön tekijän työympäristössä on hyväksi havaittu.

9.2.3 Materiaalien korjaukset

Tein dia- ja tekstimateriaaliin palautteen perusteella seuraavat korjaukset:

Lisäsin materiaaliin oman kappaleen ja dian määrittelyn suodenkuopista. Lisäsin maininnat riippuvuuksien havaitsemisesta, kuvaamisesta ja seurannasta. Samoin lisäsin mainintoja asiaksdemoista ja niiden hyödyistä sopiviin vaiheisiin.

Lisäsin maininnan sisäisestä tai ulkoisesta asiakkaasta ja tehtävänannon selkeyttämisen merkityksestä. Täydensin myös monenlaiset määrittelyt, toteutuksen yllätykset ja suunnan muuttumiset aineistoihin. Myös työmääräarviot on lisätty materiaaliin.

Selkeytin diasarjaa jakamalla pari täyteen ahdettua diaa kahdeksi ja tekemällä muutamia pieniä tarkennuksia sanamuotoihin. Päivitin tekstimuotoisen materiaalin ja diasarjan samansisältöisiksi. Yhtenäistin myös niiden jäsentelyt.

10 Johtopäätökset ja pohdinta

Opinnäytetyön käytännön osassa toteutettiin vaatimusmäärittelyn prosessin kuvaus ja ohjeistus sekä diasarjana että tekstidokumenttina eri käyttötarkoituksiin. Kuvaus luotiin sekä teoriaosuuden että samaan aikaan palkkatyössä toteutettujen kahden vaatimusmäärittelytehtävän päiväkirjamerkintöjen pohjalta. Opinnäytetyön teoriaosassa tarkasteltiin eri kehitysprojektimalleja, prosessien kuvaamistapoja ja vaatimusmäärittelyyn liittyvää teoriaa erityisesti ketterässä ohjelmistokehityksessä.

Vaatimusmäärittelyllä on merkittävä rooli lähes koko tuotekehitysprosessin ajan. Vaatimusmäärittelyn avulla ohjelmisto pystyy vastaamaan asiakkaan tarpeisiin, täyttämään kulloinkin voimassa oleva viranomais- ja standardien vaatimukset sekä tukemaan kannattavaa liiketoimintaa.

Vastaus opinnäytetyön ensimmäiseen tutkimuskysymykseen, millainen on ketterän vaatimusmäärittelyn prosessi, kuvataan yksityiskohtaisesti Ketterä määrittelyprosessi -aineiston kuvauksessa. Tiivistettynä vaiheet ovat 1) Vaatimusmäärittelyn käynnistäminen, 2) Vaatimusten kerääminen, 3) Vaatimusten tarkentaminen, 4) Valmis vaatimusmäärittely sekä 5) Sudenkuoppia ja erityistilanteita. Vaatimusmäärittelyn prosessi etenee kehitysprosessin edellä ja sitä palvellen.

Toinen tutkimuskysymys koskee ketterän vaatimusmäärittelyn roolia ohjelmistokehityksen eri vaiheissa. Ketterän vaatimusmäärittelyn rooli muuttuu ohjelmiston kehityksen edetessä. Alussa vaatimusmäärittelyä ja suunnittelua tehdään yleisellä tasolla ja määrittelyä validoidaan asiakkaan kanssa. Kun prioriteetit ovat selvillä, ensimmäisiä lohkoja määritellään tarkemmalle tasolle niin, että ohjelmointityö pääsee alkamaan. Valmistuvia versioita esitellään asiakkaille ja saadaan palautetta. Koko ajan vaatimusmäärittelyä tarkennetaan eri tasoilla kierros kierrokselta. Ylätason vaatimukset voivat laajentua, niiden alle voi tulla uusia vaatimuksia tai vanhoja poistua asiakaspalautteen myöstä. Toteutustasolla tilanne voi elää todella paljon.

Ketterässä ohjelmistokehityksessä vaatimusmäärittelyn haasteet ovat erilaisia kuin perinteisessä vesiputousmallissa. Ylätasolla vaatimusten tulee olla lohkottavissa ja

priorisoitavissa toteuttamiskelpoisiin osiin. Kun määrittelyä ei tehdä alussa liian yksityiskohtaisesti vaan yleisemmällä tasolla, vaatimusten muutokset on helpompi ottaa huomioon. Toteutustasolla vaatimusten tulee olla joustavasti päivitettävissä, tarkennettavissa ja pilkottavissa. Lisäksi määrittelyä ja suunnittelua tekevät eri henkilöt eri vaiheissa ja sen vuoksi tiedon välittyminen on keskeisen tärkeää. Kokonaisuuden hallinta on haasteellista.

Vaatimusmäärittely pyrkii pysyttelemään aina pari askelta edellä sovelluskehittäjien tarpeita, jotta sovelluksen kehitysjonolla riittää valittavaa. Samaan aikaan validoinnin ja verifioinnin tulee löytää määrittelystä, suunnittelusta ja toteutuksesta ne piirteet, joihin testauksen läpivientiä verrataan. Vaatimusmäärittelyllä ja suunnittelulla on monipuolinen kehitysprosessia palveleva rooli ketterän ohjelmistokehityksen eri vaiheissa.

Opinnäytetyön kysymykseen, miten ketterää vaatimusmäärittelyä voi ohjeistaa, vastataan vaatimusmäärittelylle itselleen asetettavien vaatimusten pohjalta. Ketterän ohjelmistokehityksen teoriat eivät anna selkeitä ohjeita siihen, miten vaatimusmäärittely tulisi toteuttaa vaan ainoastaan asiakkaan näkökulmaa ja palautetta painottavan viitekehyksen. Vaatimuksien keräämisessä, analysoinnissa ja dokumentaatiossa on kuitenkin tiettyjä huomioitavia asioita ja ohjelmistokehityksen prosessi itsessään asettaa määrittelylle ja suunnittelulle omia vaatimuksiaan. Mitä suuremmasta ohjelmistoalan yrityksestä on kysymys, sitä hallitumpaa ohjelmistotuotannon on oltava ja silloin myös vaatimusmäärittelyltä vaaditaan asianmukaista tukea kehitysprosessille. Nämä vaatimusten kriteerit ovat koulutettavissa ja ohjeistettavissa. Opinnäytetyössä laaditut diasarjamuotoiset ohjeet ja tekstimuotoiset ohjeet olivat molemmat toimivia ratkaisuja.

Opinnäytetyön tekemisessä on ollut omat ajalliset haasteensa, koska olen opiskellut työn ohessa. Toisaalta oli antoisaa tehdä opinnäytetyötä, jonka tuloksia pystyn soveltamaan omassa työssäni myös jatkossa. Syväsukellus vaatimusmäärittelyn prosessiin lisäsi ymmärrystäni määrittelyn ja suunnittelun monipuolisuudesta ja toisaalta monista erilaisista vaatimuksista, mitä niille asetetaan ohjelmistokehityksen aikana. Samoin ymmärrykseni kasvoi koko kehitysprosessista ja sen keskinäisistä riippuvuuksista. Ketterä määrittelyprosessi on ketterän kehityksen palvelemista koko kehitysprosessin ajan.

11 Yhteenveto

Opinnäytetyön tavoitteena oli tutkia, millainen on ketterän vaatimusmäärittelyn prosessi, millainen on ketterän vaatimusmäärittelyn rooli ohjelmistokehityksen eri vaiheissa sekä miten ketterää vaatimusmäärittelyä voi ohjeistaa. Opinnäytetyön tutkimuskysymyksiin vastaaminen onnistui hyvin. Ketterän määrittelyn vaiheet ovat löydettävissä kehitysprosessista, ja joiltakin osin vaatimusmäärittely jatkuu koko kehitysprosessin ajan. Ketterässä ohjelmistokehityksessä määrittelyllä ja suunnittelulla on oma, tärkeä roolinsa, joka palvelee ja luo edellytyksiä onnistuneelle sovelluskehittämiselle. Määrittelyä voi hyvin ohjeistaa esimerkiksi diasarjan tai tekstimateriaalin avulla.

Opinnäytetyön käytännön osuudessa toteutettiin ohjeistus ketterästä vaatimusmäärittelystä kahdessa eri muodossa, koulutustilaisuudessa käytettävänä diasarjana ja itseopiskelua tukevana tekstiaineistona. Käytännön osuudessa toteutettiin myös kaksi oikean työtehtävän vaatimusmäärittelyä, joiden kokemuksia hyödynnettiin ohjeistuksen laatimisessa. Toinen vaatimusmäärittelyistä tulee viimeistelyvaiheeseen opinnäytetyön valmistumisen jälkeen.

Opinnäytetyö onnistui mielestäni tavoittelemallani tavalla ja ymmärrykseni työn aihepiiristä kasvoi. Alkuvaiheen määrittely ja siihen liittyvät asiat olivat entuudestaan jossain määrin tuttuja. Määrittelyn ja suunnittelun rooli ohjelmistokehityksen koko prosessin aikana avautuu minulle nyt opinnäytetyön jälkeen ihan uudella tavalla.

Opinnäytetyön tulosta, Ketterä määrittelyprosessi -materiaalia tullaan käyttämään työpaikallani uusien vaatimusmäärittelijöiden perehdytyksessä. Jatkossa olisi mielenkiintoista tutkia tarkemmin määrittelyä testivetoisessa kehityksessä (test driven development).

Kiitän työnantajaani Tietoevry Healthia tuesta opiskelulleni ja opinnäytetyölleni.

Lähteet

Agile Alliance, (n.d.). *A Short History of Agile*. https://www.agilealliance.org/agile101/

Ahokas, K. (2022). *”Kokeilimme scrumia, mutta projekti epäonnistui” – miksi suosituin ketterä menetelmä on vaikea hallita?* Tivi. https://www.tivi.fi/uutiset/kokeilimme-scrumia-mutta-projekti-epaonnistui-miksi-suosituin-kettera-menetelma-on-vaikea-hallita/adaa871c-4080-4449-beb8-5bb6a61b8d59

Atlassian, (n.d.). *DevOps: Breaking the development-operations barrier*. https://www.atlassian.com/devops

Atlassian Agile Coach, (n.d.-a). *What is Agile?* https://www.atlassian.com/agile

Atlassian Agile Coach, (n.d.-b) *Agile Subway Map.* https://www.agilealliance.org/agile101/subway-map-to-agile-practices/

Cambridge Dictionary, (n.d.) Hakusana *Process*. https://dictionary.cambridge.org/dictionary/english/process

Chowdhury, E., Bhowmik, A., Hasan, H. & Rahim, S.(2017). *Analysis of the Veracities of Industry Used Software Development Life Cycle Methodologies*. https://arxiv.org/ftp/arxiv/papers/1805/1805.08631.pdf

DevOps, (n.d.). *DevOps principles. Customer-centric action*. https://www.atlassian.com/devops/what-is-devops

Drumond, C. (n.d.).*What is Scrum?* Atlassian Agile Coach https://www.atlassian.com/agile/scrum

Felsen, N. (2017). *Effective DevOps with AWS*. Packt Publishing, Limited. ProQuest Ebook Central. http://ebookcentral.proquest.com/lib/hamk-ebooks/detail.action?docID=4933227.

Fimea, (2021). *Lääkinnällisiin laitteisiin liittyvä lainsäädäntö*. https://www.fimea.fi/laakinnalliset_laitteet/laakinnallisiin-laitteisiin-liittyva-lainsaadanto

Gacne Denis, Ringuette, Simon (noudettu 15.1.2022). *Business Process Model and Notation. BPMN Quick Guide*. Second edition. Object Management Group. https://www.bpmn.org/

Gunja, S. (2021). *What is DevOps? Unpacking the rise of an IT cultural revolution.* Dynatrace. https://www.dynatrace.com/news/blog/what-is-devops/

Helsingin yliopiston tietojenkäsittelytieteen osasto, (2020). *Ohjelmoinnin MOOC 2020. Osa 11 Luokkakaaviot.* https://ohjelmointi-20.mooc.fi/osa-11/1-luokkakaaviot

ISO/IEC 62304:2006 (2006). *Medical device software – Software life cycle processes*. The International Electrotechnical Commission (IEC) https://www.iso.org/standard/38421.html

IEC 82304-1(2016) *Health software – Part 1: General requirements for product safety.* The International Electrotechnical Commission (IEC) https://www.iso.org/standard/59543.html

Koivisto, M., Säynäjäkangas, J. Forsberg, S. (2019). *Palvelumuotoilun bisneskirja*. Alma Talent. https://hamk.finna.fi/Record/vanaicat.134302#toc

JUHTA – Julkisen hallinnon tietohallinnon neuvottelukunta. *JHS 152 Prosessien kuvaaminen*. Versio: 5.10.2012. Julkaistu: 13.12.2002. Suomidigi. https://www.suomidigi.fi/ohjeet-ja-tuki/jhs-suositukset/jhs-152-prosessien-kuvaaminen

Kielitoimiston sanakirja. (n.d.). Hakusana *Prosessi*. https://www.kielitoimistonsanakirja.fi/#/prosessi

LeanThinking, (n.d.). *LEAN-sanaston sisällysluettelo*. https://leanthinking.fi/sanasto/heijunka/

Luukkainen, M. & Ilves, K. (2021a). *Ohjelmistotuotanto 2021, Osa 1 Ohjelmistotuotanto ja sen osa-alueet.* Helsingin yliopisto. https://ohjelmistotuotanto-hy.github.io/osa1/

Luukkainen, M. & Ilves, K. (2021b). *Ohjelmistotuotanto 2021, Osa 2 Vaatimusmäärittely.* Helsingin yliopisto. https://ohjelmistotuotanto-hy.github.io/osa2/

Liikenne ja Viestintäministeriö LVM (2021). *ICT-alan ilmasto- ja ympäristöstrategia*. Liikenne- ja viestintäministeriön julkaisuja 2021:4. http://urn.fi/URN:ISBN:978-952-243-587-3

Martinsuo, M. & Blomqvist, M. (2010). *Prosessien mallintaminen osana toiminnan kehittämistä*. Tampereen teknillinen yliopisto. Teknis-taloudellinen tiedekunta. Opetusmoniste 2. https://trepo.tuni.fi//bitstream/handle/10024/128389/prosessien_mallintaminen.pdf?sequence=1

Mikkonen-Craig, H. (2020). *Lean, liinaus ja lean-johtaminen* 4/2020 Kielikello https://www.kielikello.fi/-/lean-liinaus-ja-lean-johtaminen

Object Management Group, (n.d. -a). *Business Process Model and Notation*. https://www.bpmn.org/

Object Management Group, (n.d. -b) *BPMN Quick Guide*. http://www.bpmnquickguide.com/getit.html

Oracle Data Sheet (n.d.). *Oracle JDeveloper*. jdeveloper11g-datasheet-1-133040.pdf. Haettu 20.2.2022 osoitteesta https://www.oracle.com/application-development/technologies/jdeveloper.html

Patton, J. (2022). *User Story Mapping*. https://www.jpattonassociates.com/story-mapping/

Patton, J. (2015). *Story Map Concepts*. http://www.jpattonassociates.com/wp-content/uploads/2015/03/story_mapping.pdf

Peters, S. (n.d.). *Agile Design. Design: agile, just like development*. Atlassian Agile Coach. https://www.atlassian.com/agile/design

Rehkopf, M. (n.d. -a). *Sprints*. Atlassian Agile Coach. https://www.atlassian.com/agile/scrum/sprints

Rehkopf, M. (n.d. -b). *Stories, epics, and initiatives*. https://www.atlassian.com/agile/project-management/epics-stories-themes

Royce, Winston W. (1970). *Managing the Development of Large Software Systems*. http://www-scf.usc.edu/~csci201/lectures/Lecture11/royce1970.pdf

Rusanen, A. (2022). *Mikä on SAFe (Scaled Agile Framework) -webinaari* 9.3.2022. Tieturi. https://www.tieturi.fi/webinaari/mika-on-safe-scaled-agile-framework/

Scaled Agile, Inc. (2021 a). *SAFe 5 for Lean Enterprises*. https://www.scaledagileframework.com/safe-for-lean-enterprises/

Scaled Agile, Inc. (2021 b). *SAFe Lean-Agile Principles*. (https://www.scaledagileframework.com/safe-lean-agile-principles/

Scaled Agile, Inc. (2021 c). *Business Agility*. https://www.scaledagileframework.com/business-agility/

Scaled Agile, Inc. (2021 d). *Agile Release Train*. https://www.scaledagileframework.com/agile-release-train/

Scaled Agile, Inc. (2021 e). *Solution Train*. https://www.scaledagileframework.com/solution-train/

Scaled Agile, Inc. (2021 f). *Lean Portfolio Management*. https://www.scaledagileframework.com/lean-portfolio-management/

Scaled Agile, Inc. (2021 g). *Essential SAFe*. https://www.scaledagileframework.com/essential-safe/

Scaled Agile, Inc. (2021 h). *Large solution SAFe*. https://www.scaledagileframework.com/large-solution-safe/

Scaled Agile, Inc. (2021 i). *SAFe Implementation Roadmap*. https://www.scaledagileframework.com/implementation-roadmap/

Scaled Agile, Inc. (2021 k). *SAFe Requirements Model*. https://www.scaledagileframework.com/safe-requirements-model/

Scaled Agile, Inc. (2021 l). *Epics*. https://www.scaledagileframework.com/epic/

Scaled Agile, Inc. (2021 m). *Features and Capabilities*. https://www.scaledagileframework.com/features-and-capabilities/

Scaled Agile, Inc. (2021 n). *Story*. https://www.scaledagileframework.com/story/

Scaled Agile, Inc. (2021 o). *Enablers*. https://www.scaledagileframework.com/enablers/

Scaled Agile, Inc. (2021 p). *Nonfunctional Requirements*. https://www.scaledagileframework.com/nonfunctional-requirements/

Sosiaali- ja terveysministeriö STM. (n.d .). *Hoitoon pääsy (hoitotakuu)*. https://stm.fi/hoitotakuu

Suomen Standardoimisliitto SFS, (n.d.). *Mitä standardi tarkoittaa?* https://sfs.fi/standardeista/mika-on-standardi/

Terveydenhuoltolaki 2010/1326. https://www.finlex.fi/fi/laki/ajantasa/2010/20101326

Tietoevry Oyj (2022). *Sustainability report 2021*. https://www.tietoevry.com/en/SysSiteAssets/files/sustainability/tietoevry2021_sustainability_report.pdf

Torkkola, Sari (2015) *Lean asiantuntijatyön johtamisessa*. Alma Talent. https://ezproxy.hamk.fi/login?url=https://verkkokirjahylly.almatalent.fi/teos/15jo424908

Tuominen, P., Laitinen, U. & Häkkinen, P. (2022). *Erikoissairaanhoidon hoitoonpääsyn seuranta. Määrittelyt ja ohjeistus 2021*. Terveyden ja hyvinvoinnin laitos. https://www.julkari.fi/bitstream/handle/10024/143740/URN_ISBN_978-952-343-805-7.pdf?sequence=1&isAllowed=y

Unified Modeling Language UML. (n.d.). *What is UML. Introduction to OMF's Unified Modeling Language (UML)*. https://www.uml.org/what-is-uml.htm

United Nations Global Compact. (n.d.). *The Ten Principles of the UN Global Compact*. https://www.unglobalcompact.org/what-is-gc/mission/principles

Valvira, (2022). *Sosiaali- ja terveydenhuollon tietojärjestelmät*. https://www.valvira.fi/terveydenhuolto/sosiaali-ja-terveydenhuollon-tietojarjestelmat

West, D. (n.d.). *Scrum roles and the truth about job titles in scrum*. Atlassian Agile Coach. https://www.atlassian.com/agile/scrum/roles

Liite 1: Aineistonhallintasuunnitelma

Opinnäytetyön laatimisen yhteydessä syntyy aineisto, joka sisältää perehdytysmateriaalin arviointipalautteita, päiväkirjamerkintöjä ja opinnäytetyön versioita apudokumentteineen.

Opinnäytetyötä varten palaute perehdytysmateriaaleista pyydetään osittain vapaamuotoisena. Palautteen antajille kerrotaan, että antamalla palautetta hän hyväksyy, että analysoitua palautetta hyödynnetään opinnäytetyössä. Vastaajien henkilötietoja tai muita tunnistetietoja ei tallenneta missään vaiheessa. Vaatimusmäärittelyjen aikana pidetään myös päiväkirjaa, johon kirjataan tietoja vaatimusmäärittelyjen edistymisestä.

Kaikkia opinnäytetyön laatimisen yhteydessä syntyneitä aineistoja ja opinnäytetyön versioita säilytetään tekijän tietokoneen D-asemalla, josta tehdään säännöllisesti työstön aikana varmuuskopioita ulkoiselle kovalevylle. Materiaaleja säilytetään ulkoisella kovalevyllä vähintään yhden vuoden opinnäytetyön valmistumisesta. Tämän jälkeen aineistot tuhotaan.

Opinnäytetyön tuloksena syntyneen perehdytysaineiston omistaa Tietoevry Health.

Liite 2: Ketterä määrittelyprosessi, tekstimuotoinen aineisto

Ketterä määrittelyprosessi

Tämä ketterän vaatimusmäärittelyn prosessin kuvaus on tarkoitettu määrittelytehtäviin perehtyvälle asiantuntijalle oman opiskelun tueksi. Selostuksen tavoitteena on kuvata ja ohjeistaa määrittelyn ja suunnittelun prosessi, miten se etenee ketterän ohjelmistokehityksen vaiheiden mukana. Ohjeistus on yleisluonteinen; sen lisäksi tarvitaan koulutusta ”talon tavoista”.

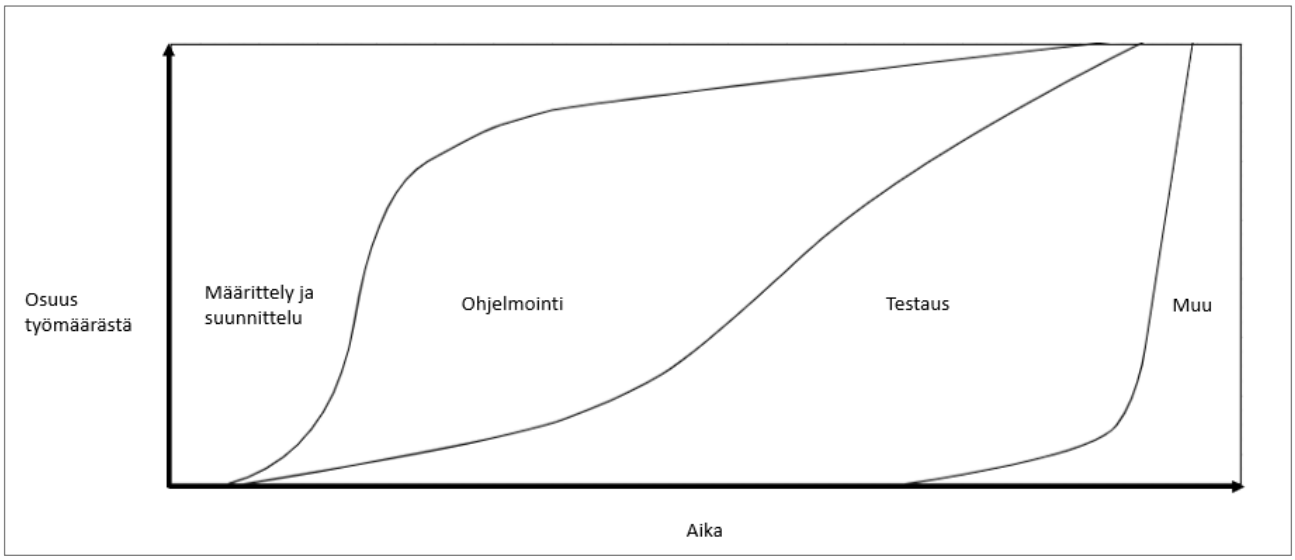
Sovelluksen vaatimusmäärittelyssä ja suunnittelussa asiakkaan vaatimukset sovellukselle kootaan ja prosessoidaan niin, että tuotekehitys ja testaus voivat tukeutua määrittelyihin ja tuottaa toimivan, vaatimukset täyttävän sovelluksen, joka on myös liiketaloudellisesti järkevä.

Sovelluksen kehittämiseen liittyy monenlaista määrittelyä, kuten liiketoiminnallista, toiminnallista ja teknistä määrittelyä. Tässä ohjeessa katsotaan vaatimusmäärittelyprosessia lähinnä toiminnallisen määrittelyn näkökulmasta.

Ketterissä ohjelmistokehityksen menetelmissä myös määrittely ja suunnittelu tehdään ketterästi. Määrittelyllä ja suunnittelulle varataan omat sprinttiaikansa. Määrittely ja suunnittelu nivoutuvat osaksi kehitysprosessin iteratiivista etenemistä, ne eivät ole itsenäisiä vaiheita. Vaatimusmäärittelyyn ja suunnitteluun osallistuu useita ammattilaisia kehitysprosessin eri vaiheissa; myös määrittely on tiimityötä.

Kuva 20 on esitetty miten sovelluskehityksen vaiheet etenevät ja miten eri vaiheiden suhteellinen työmäärä vaihtelee ajan kuluessa. Kuvio ei perustu mitattuihin työmääriin vaan pyrkii pelkästään havainnollistamaan eri vaiheiden lomittumista iteratiivisessa toteutuksessa.

Kuva 20 Vaatimusmäärittelyn suhde muihin ohjelmistokehityksen vaiheisiin.



Vaatimusmäärittely- ja suunnittelutyö painottuu kehitysprosessin alkuvaiheeseen mutta sitä tehdään lähes koko kehitysprosessin ajan. Ohjelmointivaiheeseen sisältyy koodaus- ja vastaavia tehtäviä. Testausvaiheella tarkoitetaan validointia ja verifiointia sekä muulla vaiheella tarvittavien tuotedokumenttien laatimista ja muita viimeistelytehtäviä.

1. Vaatimusmäärittelyn käynnistäminen

Vaatimusmäärittelyn tehtävänanto tulee tuotteen omistajalta. Tuotteen omistaja antaa vaatimusmäärittelijälle tehtäväksi määritellä tietyn sovelluksen tai sovellusalueen tai ominaisuuden sovellukseen. Tyypillisesti sovellus on osa jotakin ohjelmistokokonaisuutta. Vaatimusmäärittelylle annetaan myös aikataulu.

Tehtävä annetaan usein yhdelle tai useammalle aihepiirin asiantuntijalle tai asiantuntijaksi kehittyvälle työntekijälle. Ammattinimikkeellä ei yleensä ole merkitystä.

Tehtävänannossa sovitaan muoto, millä määrittely tehdään. Määrittely voi olla tiivis, määrämuotoinen kuvaus kehitysjonon ylätason elementillä kuten kertomuksella (epicillä) tai määrämuotoisella dokumentilla. Määrittely voi olla myös vapaamuotoinen kuvaus kertomuksella (epicillä), erillisellä dokumentilla tai vaikkapa wiki-sivulla.

Tehtävänannossa kerrotaan määrittelyn syy tai käyttötarkoitus. Onko tarkoitus määritellä virheenkorjaus vai asiakaspalautteen perusteella tehtäviä muutoksia (change requests)? Onko määrittelytarpeen alkuunpanijana lainsäädännön tai viranomaisohjeiden muuttuminen? Määritelläänkö sovelluksen laajennus vai kokonaan uusi sovellus? Tässä kuvauksessa oletetaan, että kyseessä on yksittäisen sovelluksen suunnittelu.

Tehtävänantoa on edeltänyt liiketoimintajohdon ja tuotehallinnan tekemä hyöty- tarve- ja kannattavuusanalyysi omine vaiheineen. Taustalla on mm. tuotestrategia ja tuotevalikoiman kehitysjono (backlog), teknologiset linjaukset sekä julkaisusuunnitelma.

Ennen määrittelyn aloittamista tehtävänannon selkiyttämiseen kannattaa käyttää aikaa. Tarvittaessa määrittelytehtävän antajalta voi pyytää lisätietoja.

Vaatimusmäärittely aloitetaan perehtymällä tehtävänantoon. Selvitä kuka ja millainen asiakas on kyseessä. Asiakas voi olla organisaation sisäinen tai ulkoinen. Ota selvää millainen sopimus määrittelystä tai projektista on tehty asiakkaan kanssa.

Selvitä aihealueen perustiedot. Ota selvää, mikä on sovelluksen käyttötarkoitus, mitä sovelluksen avulla halutaan saavuttaa tai välttää. Selvitä mitkä ovat sovelluksen keskeiset, tavoitellut toiminnallisuudet. Myös aihealueen rajaus on hyvä sopia.

Sovellukseen saattaa liittyä velvoittavaa lainsäädäntöä, viranomaisohjeita tai standardeja. Huomioi myös tekniset reunaehdot kuten päätelaitteet, joilla sovellusta tullaan käyttämään. Huomaa, että valittuun teknologiaan voi sisältyä myös uusia mahdollisuuksia, kuten ympäristönäkökohtia energiankäytössä tai sovelluksen toiminnallisuuksissa.

Jos sovelluksella on rajapintoja useisiin muihin sovelluksiin tai sitä käytetään yhdessä useiden sovellusten kanssa, määrittelyn alkuvaiheessa voi pitää myös työpajan yhdessä näiden sovellusten asiantuntijoiden kanssa. Työpajan tavoitteena on koota hyvät lähtötiedot ja kertyneet vaatimukset/asiakastoiveet useasta eri näkökulmasta. Työpajan vetäjäksi voi pyytää jonkun työpajatyöskentelyn asiantuntijan. Työpajan avulla saat koottua hyvät pohjatiedot määritettävästä sovelluksesta ja mitä vaatimuksia muut sovellukset sille asettavat. Yhdessä havaitaan paremmin myös riippuvuudet, joita varsinkin isoissa järjestelmissä voi olla runsaasti. Työpajan avulla

löydetään myös lyhyessä ajassa keskeisimpiä kehityskohteita senhetkisen tiedon mukaan. Kehityskohteista saa hyviä keskustelunavauksia, kun ominaisuuksia määritellään asiakkaan kanssa. Työpajan työryhmä voi myöhemmin toimia myös hyvänä tukiryhmänä määrittelytyössä.

Tekniset reunaehdot ovat määrittelyssä huomioitavia, usein ei-toiminnallisia ominaisuuksia, jotka ohjelmistoarkkitehti määrittelee tai on aikaisemmin määritellyt. Perustavalaatuisia reunaehtoja ovat esimerkiksi onko sovellus asennettavissa paikallisesti vai tarjotaanko se pilvipalveluna, millaiset verkkoratkaisut ohjelmisto vaatii, millä päätelaitteilla ohjelmistoa voi käyttää, mikä on ohjelmiston perusjärjestelmä. Ohjelmistokohtaisia ovat usein myös suorituskykymitoitus, järjestelmän skaalautuvuus, tietosuoja- ja tietoturvaratkaisut sekä jakelu- ja asennusasiat. Sovellusta itseään voi rajoittaa riippuvuus muista järjestelmistä tai integraatioista. Sovelluskohtainen tekninen suunnittelu tai arkkitehtuurisuunnittelu on hyvä tehdä yhteistyössä toiminnallisten vaatimuksien määrittelyn kanssa.

2. Vaatimusten kerääminen

Vaatimusten keräämistä voidaan tarkastella kolmesta eri näkökulmasta: asiakaskommunikaation kautta, vaatimusten tunnistamisen kautta ja vaatimusten käsittelyn kautta. Näkökulmat lomittuvat keskenään käytännön työssä.

Selvitä ketkä ovat asiakkaita, eli mitkä ovat ne sidosryhmät, jotka ovat eri tavoin tekemisissä määriteltävän sovelluksen kanssa. Asiakaskommunikaatiossa on tavoitteena saada hyvä keskusteluyhteys asiakkaan kanssa, jotta voi oppia tunnistamaan ja syvällisesti ymmärtämään asiakkaan tarpeet ja vaatimukset kehitettävälle sovellukselle. Asiakaskommunikaatiota voi helpottaa, jos määrittelyprosessin alussa käydään asiakkaan kanssa yhdessä läpi määrittely- ja kehitysprosessin käytännöt. Näin varmistetaan, että myös ne asiakkaan edustajat, jotka eivät ole ICT-alan ammattilaisia vaan oman toimialansa asiantuntijoita, tuntevat prosessin menettelyt.

Sovi vaatimusten keräystavat, esimerkiksi menettekö seuraamaan työskentelyä paikan päälle vai haastatteletteko käyttäjiä. Kun demoversio tai ensimmäinen versio on valmistunut, sen avulla voi tehdä käytettävyystestausta yhdessä käyttäjien kanssa. Palvelumuotoilusta tuttu Tuplatimantti-konseptointi on myös hyvä työkalu vaatimusten keräämiseen ja käsittelyyn.

Sovi määrittelyn, suunnitelmien ja sovellusversioiden esittelytavat. Asiakaspalavereissa vaatimuksia ja ratkaisuja voi mallintaa esimerkiksi näytönkuvilla tai prototyyppiversioilla.

Käy vaatimukset ja valmiit ratkaisut toistuvasti läpi asiakkaan kanssa määrittely- ja kehitysprosessin aikana. Näin varmistat, että teette oikeita asioita ja samalla sitoutat asiakasta sovellukseen. Huomioi myös, että matkan varrella tulee aina yllätyksiä. Hyvä keskusteluyhteys asiakkaan kanssa on silloin hyvin tärkeää.

Asiakasvaatimuksien tunnistaminen alkaa asiakkaan odotuksien, tarpeiden ja reunaehtojen selvittämisellä. Asiakasvaatimuksien ymmärtämiseksi pitää ymmärtää myös asiakkaan liiketoimintaprosessit ja -tavoitteet, joihin sovellus liittyy. Ota selvää, miten sovelluksen ratkaisema asia hoidetaan tällä hetkellä ja onko siinä ongelmia. Selvitä ketkä tulevat käyttämään sovellusta, mitkä ovat keskeisimmät käyttäjäpersoonat/käyttäjätyypit, tavallisimmat käyttötapaukset ja niiden työnkulut. Tutki mitä vaatimuksia käytölle ja käyttöliittymälle voidaan asettaa.

Vaatimukset ja ratkaisut mallinnetaan ja demostroidaan asiakkaalle toistuvasti esimerkiksi näytönkuvien, prototyyppien ja valmiiden osien esittelyjen avulla. Demojen palaute kertoo ollaanko menossa oikeaan suuntaan. Demojen palautteesta saadaan myös uusia vaatimuksia tai päivityksiä olemassa oleviin vaatimuksiin.

Asiakasvaatimusten käsittely tarkoittaa vaatimusten kokoamista, analysointia ja dokumentointia. Vaatimukset on hyvä kuvata tekstin ohella myös visuaalisesti, kuvien ja kaavioiden avulla. Sisällytä vaatimuksiin toiminnalliset ja ei-toiminnalliset vaatimukset. Kuvaa ja analysoi myös esille tulleet riskit, riippuvuudet ja rajoitukset.

Asiakasvaatimukset tulee validoida, varmistaa, että vaatimukset on ymmärretty oikein. Vaatimukset eivät saa olla ristiriidassa keskenään vaan ne täytyy synkronoida.

Vaatimusmäärittelystä tehdään tarvittaessa koostedokumentti. Vaatimusmäärittelydokumentissa voidaan vaatimusten lisäksi taustoittaa aihepiiriä ja vaatimuksia, varsinkin isommissa ja/tai monimutkaisemmissa määrittelyissä. Tällöin vaatimusmäärittelydokumenttia voi käyttää suunnittelijoiden, kehittäjien ja testaajien aiheeseen perehtymisen apuna.

Vaatimukset priorisoidaan ensin määrittelyn yhteydessä ja myöhemmin tuoteomistajan tai kehitysprojektin toimesta vaatimusten hallinnointiin liittyen.

Ohjelmistoja tuottavalla organisaatiolla on sovitut käytännöt siitä, kuinka monta ja minkä nimisiä portaita kehitysjonoilla (backlogeilla) on ja mikä niiden rooli on tuotekehitysprosessissa. Tässä käytetään neliportaista asteikkoa: kertomus (epic) – ominaisuus (feature) – tarina (story) – tehtävä (task).

Yksittäinen vaatimus voi olla iso ylätason vaatimus kertomuksella tai pieni vaatimus tarinalla tai jotain siltä väliltä. Yksittäiset vaatimukset voidaan kuvata vaatimusmäärittelydokumentissa mutta ennen kaikkea ne viedään kehitysjonolle, sen sopivalle tasolle. Yksittäinen vaatimus voi siis olla kertomus, ominaisuus tai tarina. Kehitysjonolle vienti tehdään määrittelyn lopulla tai sen jälkeen. Vaatimus voi olla tyypiltään toiminnallinen, ei-toiminnallinen tai mahdollistava. Kehitysjonolla vaatimus saa yksilöivän id-numeron, prioriteetin ja linkitykset mihin toiminnallisuuteen se kuuluu. Myöhemmin se saa työmääräarvion ja aikataulun: inkrementin ja sprintin.

Kehitysjonon eri portaiden vaatimuksilla tulee olla valmiin määritelmä, Definition of Done, eli kriteerit siitä, miten riittävän hyvä vaatimus kirjoitetaan. Yksittäisen vaatimuksen sisältö on toivottavaa kuvata käyttäjän näkökulmasta aina kun mahdollista. Esimerkiksi: ”Käyttäjänä haluan siirtyä potilaslistalta potilaan nimen tai muun tunnisteen kautta potilaan kertomustietoihin, jotta voin pitää vastaanottoa potilaslistan kautta.” Yksittäisen vaatimuksen tulee olla selkeä ja yksiselitteinen. Se tulee olla testauksella todennettavissa. Sen tulee olla jäljitettävissä, mistä mahdollisesta ylätason vaatimuksesta se polveutuu ja mitä mahdollisia muutoksia se on läpikäynyt.

Kun uudesta sovelluksesta luodaan kehitysjonoa, mahdollinen vaatimusmäärittelyn kooste viedään yhdelle kertomukselle (epic) määrämuotoisena tai liitteeksi. Kertomuksia voi olla sovellusta kohti useampi. Priorisoidut vaatimukset viedään kertomusten alle ominaisuuksina (feature). Tästä saadaan sovelluksen kehitysjono.

Inkrementtisuunnittelun yhteydessä tiimi valitsee omalle kehitysjonolleen ominaisuudet (feature), jotka se pilkkoo kehitystyön kannalta sopivan kokoisiksi tarinoiksi (story). Tarvittaessa eri tehtävät pilkotaan vielä tehtäviksi (task) niiden alle. Tästä syntyy tiimin kehitysjono.

3. Vaatimusmäärittelyn tarkentaminen

Toteutusvaiheessa, kun ohjelmaa kehitetään paloittain, tehdään vaatimusten tarkennuksia, priorisointia, karsintaa, muokkaamista ja uusia vaatimuksia. Toteutuksessa saatetaan huomata, että asioita voidaan toteuttaa paremmin kuin määrittelyssä on suunniteltu. Toteutuksessa huomataan myös, jos vaatimusmäärittelystä puuttuu jotakin tai jos siinä on jotakin turhaa. Toteutuksen tulee kuitenkin pysyä ylätason vaatimusten puitteissa.

Toteutuksen aikana tehdään myös riippuvuuksien ja riskien havainnointia, kuvaamista ja seurantaa. Toteutuksen aikana pidetään toistuvasti asiakasdemoja. Asiakasdemoista saatu palaute on erityisen arvokasta työn validoinnin kannalta. Joskus kehityksen aikana tulee yllätyksiä. Niitä ei kannata pelätä vaan ne kuuluvat asiaan. Joskus voidaan joutua muuttamaan kehityksen suuntaa, tai kehitykseen voi tulla keskeytyksiä.

Toteutuksen dokumentointi, edes lyhyesti, on tärkeää. Testitapaukset perustuvat määrittelyn ja toteutuksen dokumentaatioon, samoin tuotedokumentit. Lisäksi määrittelyn ja toteutuksen tulee säilyttää validiteettinsa ja jäljitettävyytensä.

Ketterissä menetelmissä ketterä määrittely ja suunnittelu tarkoittaa ennen kaikkea joustavaa tapaa tarkentaa ylätason määrittelyjä lähempänä toteutusajankohtaa. Ominaisuudet on määritelty ylätasolla, sovelluksen kehitysjono on olemassa vähintään kertomustasolla (epic). Tarkentavat määrittelyt tehdään toteutuksen prioriteetin mukaan. Tällöin minimoidaan turha työ; määritellään tarpeen mukaan ja iteratiivisesti.

Vaatimuksien tarkentamiselle varataan oma sprinttiaikansa. Tavoitteena on, että sovelluksen kehitysjonolla olisi tiimille aina riittävästi valmista, yksityiskohtaista määrittelyä inkrementtisuunnittelua varten. Tällöin määrittely – suunnittelu – toteutus – testaus -virtaus etenee sujuvasti.

Vaatimusmäärittelyllä on oma roolinsa vielä testausvaiheessakin. Testitapaukset luodaan usein vaatimuksien ja toteutuksen pohjalta. Testitapauksille tulee löytyä vastineet vaatimusmäärittelystä, sen eri vaiheista. Lisäksi testitapauksien tulee kattaa kaikki toteutetut vaatimukset.

Testausvaiheessa sovelluksen ominaisuudet validoidaan, eli tarkistetaan, että on tehty oikeita asioita. Validointi tehdään yhdessä asiakkaan kanssa. Testauksen avulla sovelluksen ominaisuudet myös verifioidaan, eli tarkistetaan, että asiat toimivat oikein, että sovellukseen ei ole jäänyt julkaisua estäviä bugeja. Riskit ja riippuvuudet tulee huomioida myös testausvaiheessa.

Ketterissä menetelmissä käytettävällä ketterällä määrittelyllä ja suunnittelulla on myös varjopuolensa. Iteratiivisen määrittelyn ongelmana on helposti kokonaiskuvan katoaminen, erityisesti silloin, kun toteuttavia tiimejä on useita. Kun määritellään yksityiskohtia, unohtuu esimerkiksi helposti, miten samankaltaisia toiminnallisuuksia on määritelty toisaalla ohjelmistossa. Sovelluksen ja ohjelmiston logiikan tulisi kuitenkin toimia kauttaaltaan samalla tavalla. Esimerkiksi kohdetta hiirellä vedettäessä lähtöpaikan ja kohdepaikan korostuksien tulee olla yhteneväiset koko sovelluksessa, tai terminologian yhteneväinen koko sovelluksessa. Käyttökokemussuunnittelun linjaukset auttavat määrittelyn yhdenmukaisuudessa ja niillä on sen vuoksi määrittelyjen tarkentamisessa tärkeä rooli.

4. Valmis vaatimusmäärittely

Milloin määrittely on sitten valmis? Alun tehtävänannossa vaatimusmäärittelylle on annettu aikataulu ja muut reunaehdot. Siihen mennessä sovitun tasoinen määrittely tulee olla tehtynä. Myös vaatimusmäärittelyllä on hyvä olla valmiin määrittely, Definition of Done, jota vasten se on suositeltavaa myös katselmoida.

Kuitenkin vaatimusmäärittely, sen tarkentaminen ja muu hyödyntäminen jatkuu iteratiivisesti siihen asti, kunnes kaikki testitapaukset on kuvattu. Vaatimusmäärittelyä voi hyödyntää myös tuotedokumentaatiossa ja ylläpidon aikana.

5. Sudenkuoppia ja erityistilanteita

Vaatimusmäärittelyyn liittyy muutamia sudenkuoppia. Määrittely ja toteutus voidaan tehdä tekniikan ehdoilla eikä aidosti käyttäjille. Tulos ei silloin ole käyttäjien kannalta paras mahdollinen. Määrittelyssä voidaan keskittyä vain omaan tuotteeseen. Silloin riippuvuuksien havainnointi, kuvaaminen ja seuranta voi jäädä puutteelliseksi ja yllätykset ovat väistämättömiä. Jos vaatimusmäärittely on ollut laaja, ja sen perusteella määritetään rajattu toteutus

työmääräarvioineen, voi käydä niin, että toteutuksen kuvauksessa ei kerrota selkeästi, mitä toiminnallisuuksia määrittelystä on tarkoitus toteuttaa ja mitä ei sillä työmäärällä. Silloin asiakas voi pettyä tulokseen.

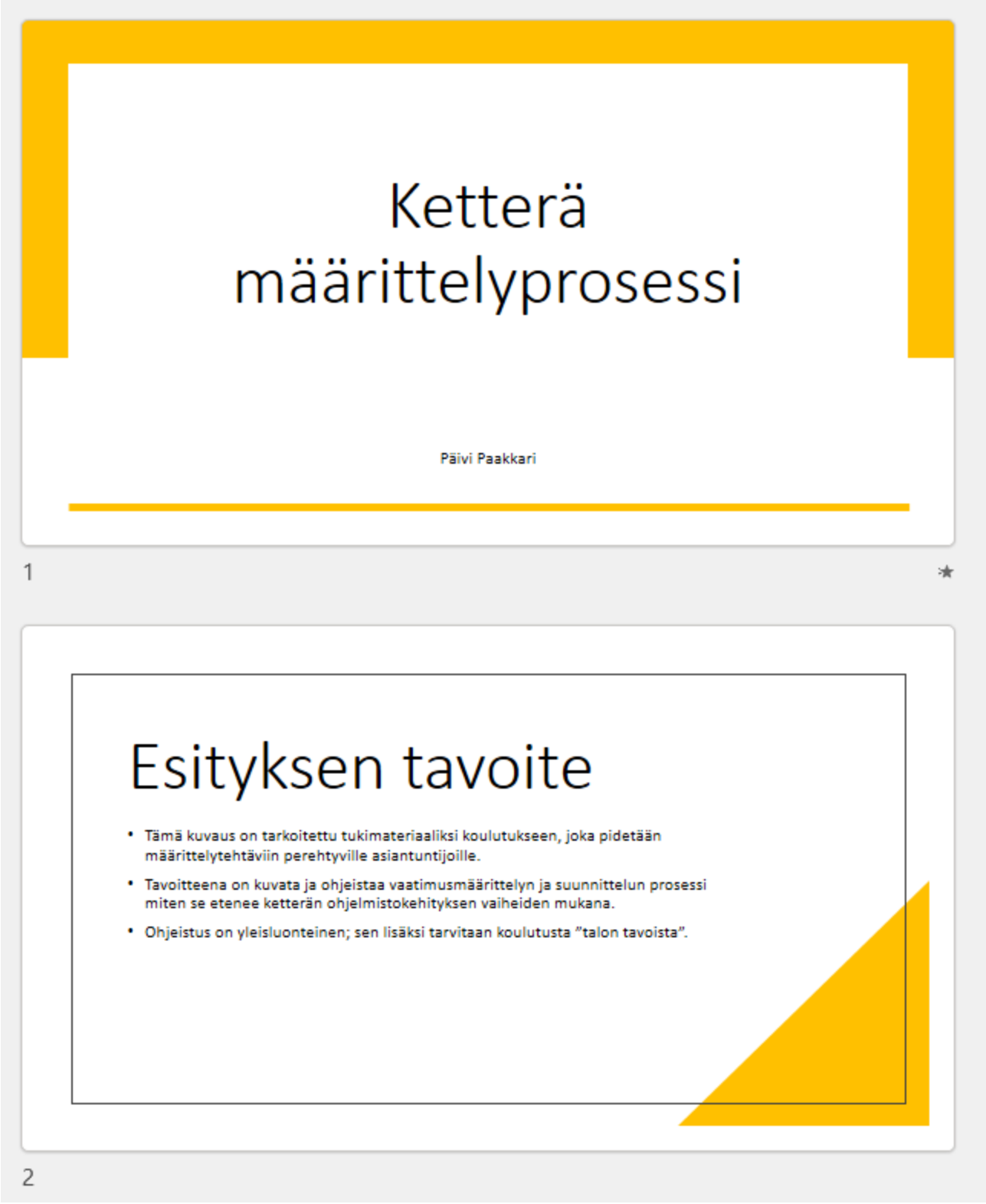
Vaatimusmäärittelyssä tulee usein vastaan erityistilanteita. Vaatimusmäärittelytehtävä voi koskea sovellusta tai sovelluksen osaa, jolla on todella paljon riippuvuuksia joka suuntaan niin, että vaatimusmäärittelylle jää vain vähän liikkumavaraa. Tällöin voi olla kyse esimerkiksi virheenkorjauksesta tai muuttuneista viranomaisvaatimuksista. Usein muutos pyritään rajaamaan mahdollisimman pieneen sovellusten lukumäärään, mieluiten vain yhteen. Tällöin voi auttaa, jos määrittelijä pystyy tutkimaan tilanteen perusteellisesti, kuvaamaan riippuvuudet ja pohtimaan ratkaisua useamman henkilön ryhmässä esimerkiksi tuotekehittäjien kanssa.

Vaatimusmäärittelytehtävä voi koskea myös sovellusta tai sovelluksen osaa, jonka mahdollisista riippuvuuksista ei vielä ole tarkkaa tietoa. Määrittelyä tehtäessä ei ole vielä tiedossa, miten jokin määrittelyyn liittyvä asia tullaan ratkaisemaan. Määrittelijä joutuu määrittelemään kokonaisuutta, josta osa on tavallaan ilmassa. Silloin voi vain määritellä mitä ominaisuuksia puuttuvasta osasta pitää löytyä nyt määriteltävän sovelluksen tarpeisiin sitten, kun puuttuvan osan määrittelyä viedään eteenpäin.

Kiireelliset tai kriittiset virheet vaativat joskus korjauksen kiireellistä määrittelyä. Tällöin määrittelytiimiin kannattaa koota mahdollisimman monipuolinen ryhmän osaajia aiheen ympäriltä, jotta eri näkökulmat tulevat huomioitua. Ratkaisun määrittelyn katselmointi kannattaa tehdä kiireestä huolimatta.

Tuotekehitysprosessi voi joskus pysähtyä siihen vaiheeseen, kun vaatimusmäärittely on saatu valmiiksi ja katselmoitu. Vaatimusmäärittely laitetaan silloin hyllylle odottamaan kehitysprosessin käynnistymistä. Jos ja kun kehitysprosessi käynnistyy, vaatimusmäärittely voidaan tarkistaa ja päivittää ennen kuin se viedään kehitysjonolla pitemmälle. Tilanteessa pitää huomioida myös, miten asia kommunikoidaan asiakkaalle.

Liite 3: Ketterä määrittelyprosessi, diasarja



Vaatimusmäärittely

- Sovelluksen vaatimusmäärittelyssä ja suunnittelussa asiakkaan vaatimukset sovellukselle kootaan ja prosessoidaan niin, että tuotekehitys ja testaus voivat tukeutua määrittelyihin ja tuottaa toimivan, vaatimukset täyttävän sovelluksen, joka on myös liiketaloudellisesti järkevä.
- Sovelluksen kehittämiseen liittyy monenlaista määrittelyä: liiketoiminnallista, toiminnallista ja teknistä määrittelyä.
- Tässä ohjeessa katsotaan vaatimusmäärittelyprosessia lähinnä toiminnallisen määrittelyn näkökulmasta.

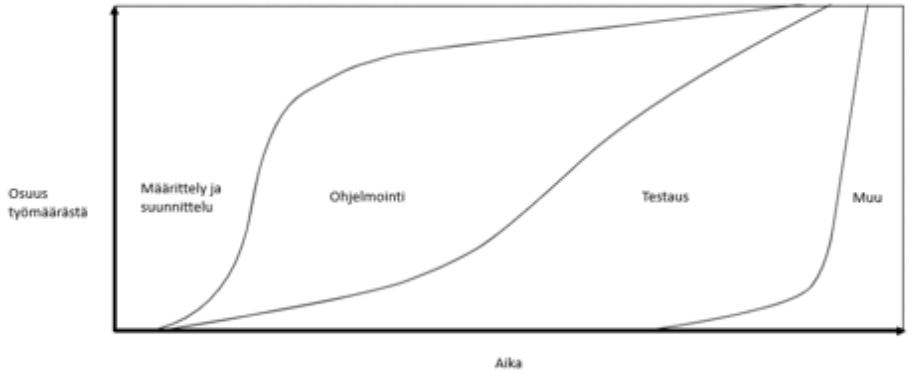
3

Vaatimusmäärittely

- Ketterissä menetelmissä myös määrittely ja suunnittelu tehdään ketterästi:
  - Inkrementeissä / sprinteissä.
  - Osana kehitysprosessin iteratiivista etenemistä (ei itsenäinen vaihe).
- Vaatimusmäärittelyyn ja suunnitteluun osallistuu useita ammattilaisia kehitysprosessin eri vaiheissa
  - Tiimityötä!

4

Vaatimusmäärittelyn suhde muihin ohjelmistokehityksen vaiheisiin



Kuvio pyrkii havainnollistamaan eri vaiheiden lomittumista iteratiivisessa toteutuksessa

5

Vaatimusmäärittelyn käynnistäminen

6

Vaatimusmäärittely

- Tehtävänanto tulee tuotteen omistajalta:
  - määriteltävä sovellus / -alue tai asia
  - osana ohjelmistoa xx
  - aikataulu
- Tehtävä annetaan aihepiirin asiantuntijalle / -asiantuntijoille tai sellaiseksi kehittyvälle
- Tehtävänannossa sovitaan myös määrittelyn muoto
  - Tiivis määrämuotoinen kuvaus epicillä / featurella?
  - Lyhyt vapaamuotoinen kuvaus epicillä / featurella?
  - Määrämuotoinen tai vapaamuotoinen erillinen dokumentti?
  - Muu organisaatiossa sovittu tapa, esimerkiksi wiki-sivu?

7

Vaatimusmäärittely

- Vaatimusmäärittelyn syy tai käyttötarkoitus?
  - Määrittelyä vaativa virheenkorjaus?
  - Asiakaspalautteen perusteella tehtävät muutokset (change requests)?
  - Viranomaisohjeet / säädökset muuttuneet?
  - Sovelluksen laajennus tai uusi sovellus?
- Tehtävänantoa on edeltänyt tuotehallinnan tekemä hyöty-, tarve- ja kannattavuusanalyysi omine vaiheineen
  - Tuotestrategia, tuotevalikoiman kehitysjono (Portfolion backlog)
  - Teknologiset linjaukset, julkaisusuunnitelmat
- Ennen määrittelyn aloittamista tehtävänannon selkeyttämiseen kannattaa käyttää aikaa

8

Perehtymisvaihe

- Kuka ja millainen asiakas? Sisäinen vai ulkoinen?
- Minkälainen sopimus on asiakkaan kanssa?
- Aihealueen perustietojen selvittäminen
- Sovelluksen käyttötarkoitus
- Mitä sovelluksen avulla halutaan saavuttaa tai välttää?
- Tavoitellut keskeiset toiminnallisuudet
- Aihealueen rajaus
- Lainsäädäntö, viranomaisohjeet, standardit
- Tekniset reunaehdot kuten päätelaitteet
- Tekniset mahdollisuudet kuten ympäristönäkökohdat ja energiankulutuksen vähentäminen

9

Työpaja

- Onko määriteltävällä sovelluksella useita rajapintoja tai riippuvuuksia muihin sovelluksiin?

➢ Ohjelmistoyrityksen sisäinen työpaja (workshop)
  - Kootaan yhteen asiaan perehtyneet asiantuntijat
  - Kootaan lähtötiedot ja kertyneet vaatimukset useasta eri näkökulmasta
  - Havaitaan riippuvuudet, joita isoissa järjestelmissä usein runsaasti
  - Etsitään keskeisimmät kehityskohteet sen hetkisen tiedon mukaan

- Vetäjäksi työpajatyöskentelyn asiantuntija
- Vaatimusmäärittelijät saavat hyvät pohjatiedot ja tukiryhmän työlleen

10

Tekniset reunaehdot

- Huomioidaan määrittelyn yhteydessä
- Usein ei-toiminnallisia

- Perustavanlaatuiset, kuten
  - Paikallinen vai pilvi
  - Verkkoratkaisut
  - Päätelaitteet
  - Perusjärjestelmä

- Ohjelmistokohtaiset, kuten
  - Suorituskyky
  - Skaalautuvuus
  - Tietosuoja / -turva
  - Jakelu ja asennus

- Sovelluskohtaiset, kuten
  - Riippuvuus muista sovelluksista
  - Riippuvuus integraatioista
  - Tekninen / arkkitehtuurin suunnittelu yhtä aikaa!

11

Vaatimusten kerääminen

12

Kolme näkökulmaa vaatimusten keräämiseen

- Asiakaskommunikaatio
- Vaatimusten tunnistaminen
- Vaatimusten analysointi
- Lomittuvat keskenään käytännön työssä

13

Asiakaskommunikaatio

- Ketkä ovat asiakkaita: sidosryhmät, jotka ovat tekemisissä määriteltävän sovelluksen kanssa.
- Tavoitteena on saada hyvä keskusteluyhteys asiakkaan kanssa, jotta voi oppia tunnistamaan ja ymmärtämään asiakkaan tarpeet ja vaatimukset kehitettävälle sovellukselle.
- Alussa sovitaan asiakkaan kanssa määrittelyprosessin ja kehitysprosessin käytännöt.
  - Varmistetaan että myös muut kuin ICT-alan ihmiset tuntevat menettelyt.
- Sovitaan vaatimusten keräystavat, esim. työskentelyn seuraaminen, haastattelut, käytettävyystestaukset, tuplatimantti-konseptointi.
- Sovitaan suunnitelmien ja sovellusversioiden esittelytavat.
- Vaatimusten ja valmiiden ratkaisujen iteratiivinen läpikäynti asiakkaan kanssa useita kertoja - validointi, sitouttaminen.
- Yllätyksiä tulee matkan varrella - hyvä keskusteluyhteys silloin tärkeä.

14

Asiakasvaatimusten tunnistaminen

- Asiakkaan odotukset, tarpeet ja reunaehdot
- Liiketoimintaprosessit ja -tavoitteet, joihin sovellus liittyy
- Miten asia hoidetaan tällä hetkellä? Onko ongelmia?
- Sovelluksen käyttäjät
- Sovelluksen käyttötapaukset ja työnkulut
- Käytön ja käyttöliittymän UX-vaatimukset
- Vaatimusten ja ratkaisujen mallintaminen ja demonstrointi asiakkaalle toistuvasti, esim. näytönkuvat, prototyyppiversiot, valmiit osiot
- Demojen palaute kertoo ollaanko menossa oikeaan suuntaan
- Palautteen muotoilu vaatimuksiksi / vaatimusten päivittäminen

15

Asiakasvaatimusten käsittely

- Vaatimusten kokoaminen, analysointi ja dokumentointi
  - Tekstin ohella myös kuvat ja kaaviot
  - Toiminnalliset ja ei-toiminnalliset vaatimukset
  - Riskit, riippuvuudet ja rajoitteet
- Vaatimusten validointi ja synkronointi
  - Onko ymmärretty asiat oikein?
  - Onko vaatimuksissa ristiriitoja?
- Vaatimusmäärittelyn koostedokumentti
  - Vaatimusten lisäksi taustoittaminen
  - Isommat ja/tai monimutkaisemmat määrittelyt
  - Suunnittelijoiden, kehittäjien ja testaajien perehtyminen aihealueeseen
- Vaatimusten priorisointi ja hallinnointi
  - Tuoteomistaja, kehitysprojekti

16

Yksittäinen vaatimus

- Organisaatiossa on sovitut käytännöt backlogilla käytettävistä portaista
  - tässä käytetään neliportaista asteikkoa: epic - feature - story - task
- Yksittäinen vaatimus luodaan backlogille sopivalle tasolle määrittelyn lopulla/jälkeen:
  - Esimerkiksi epic, feature, story
  - Toiminnallinen, ei-toiminnallinen, mahdollistava
  - ID, prioriteetti, mihin kokonaisuuteen liittyy
  - Saa myöhemmin inkrementin, sprintin, työmääräarvion
  - Vaatimuksen kuvaus mieluiten käyttäjän näkökulmasta:
    - Esimerkiksi "Käyttäjän haluan siirtyä potilaslistalta potilaan nimen tai muun tunnisteen kautta potilaan kertomustietoihin, jotta voin pitää vastaanottoa listan kautta"
- Selkeä, yksiselitteinen, todennettavissa ja jäljitettävissä
- Definition of Done (DoD), valmiin määritelmä eri tasojen vaatimuksilla

17

Backlogien luominen

- Sovelluksen backlogin luominen
  - Vaatimusmäärittelyn kooste viedään epicille: lyhyt teksti tai dokumentti liitteeksi
  - Priorisoidut vaatimukset viedään featureiksi epicin alle
- Tiimin backlogin luominen
  - Featuret pilkotaan storyiksi inkrementtisuunnittelun yhteydessä
    - Storyjen alle taskit tarvittaessa

18

Vaatimusmäärittelyn tarkentaminen

19

Toteutus

- Kun ohjelmaa kehitetään, tehdään vaatimusten tarkennuksia, priorisointia, karsintaa, muokkaamista ja uusia vaatimuksia
- Toteutuksessa saatetaan huomata, että asioita voidaan toteuttaa paremmin kuin vaatimusmäärittelyssä on suunniteltu
- Toteutuksessa huomataan, jos vaatimusmäärittelystä puuttuu asioita tai jos siinä on jotain turhaa.
- Toteutuksen tulee kuitenkin pysyä ylätason vaatimusten puitteissa
- Riippuvuuksien ja riskien havainnointi, kuvaaminen ja seuranta
- Toistuvat asiakasdemot!
- Yllätyksiä tulee, kuuluvat asiaan, ei tarvitse pelätä
- Joskus suuntaa voi joutua muuttamaan tai tulee keskeytyksiä
- Toteutuksen dokumentointi - lyhyestikin - on tärkeää!
  - Testitapaukset, tuotedokumentit kuten asennusohjeet, konfiguraatio-ohjeet
  - Validiteetti, jäljitettävyys

20

Iteratiivinen määrittely, pros

- Iteratiivisen määrittelyn etuna on määrittelyn joustava tarkentaminen lähempänä toteutusaikaa
  - Ominaisuuksia on määritelty ylätasolla
  - Sovelluksen backlog on olemassa vähintään epic-tasolla
  - Tarkennukset tehdään ominaisuuksien kehittämisen prioriteetin mukaan
  - Tarkennukset, featuret, tehdään joustavasti lähempänä suunniteltua toteutusajankohtaa - turhan työn minimointi
  - Vaatimuksien tarkennuksille varataan oma sprinttiaikansa
  - Tavoitteena on, että sovelluksen backlogilla olisi tiimille aina riittävästi valmista inkrementtisuunnittelua varten
  - Määrittely - Suunnittelu - Toteutus - Testaus -virtauksen sujuvuus!

21

Testaus

- Testitapaukset
  - Testitapaukset luodaan vaatimuksien ja toteutuksen pohjalta
  - Testitapauksille tulee löytyä vastineet vaatimusmäärittelyn eri vaiheista
  - Testitapauksien tulee kattaa kaikki toteutetut vaatimukset
- Validointi
  - Tehty oikeita asioita
  - Validoidaan usein asiakkaan kanssa
- Verifiointi
  - Tehty asioita oikein
  - Buginmetsästystä
- Riskien ja riippuvuuksien huomiointi myös testauksessa!

22

Iteratiivinen määrittely, cons

- Iteratiivisen määrittelyn ongelmana on helposti kokonaiskuvan katoaminen, varsinkin, kun tiimejä on useita
- Kun määritellään yksityiskohtia, unohtuu helposti miten samantyyppisiä toiminnallisuuksia on määritelty toisaalla.
  - Sovelluksen logiikan pitäisi toimia kauttaaltaan samalla tavalla
  - Esimerkiksi kohdetta hiirellä vedettäessä lähtöpaikan ja kohdepaikan korostuksien tulee olla yhteneväiset; samaa tarkoittavien termien tulee käyttöliittymissä olla samanlaiset kautta ohjelmiston,
- UX-linjaukset!

23

Milloin määrittely on valmis?

- Vaatimusmäärittelylle on annettu aikataulu
- Siihen mennessä sovitun tasoinen määrittely tulee olla tehtynä
- Vaatimusmäärittelyn DoD ja katselmointi
- Periaatteessa vaatimusmäärittely ja/tai sen hyödyntäminen jatkuu siihen asti, kunnes kaikki testitapaukset on kuvattu tai pitemmällekin
- Vaatimusmäärittelyä voi hyödyntää myös tuotedokumentaatiossa ja ylläpidon aikana

24

Sudenkuoppia

- Määritellään ja toteutetaan tekniikan ehdoilla eikä aidosti käyttäjille
  - Tulos ei käyttäjän kannalta paras mahdollinen
- Keskitytään vain omaan määriteltävään tuotteeseen
  - Riippuvuuksien havainnointi, kuvaaminen ja seuranta jää puutteelliseksi, yllätyksiä tiedossa
- Jos vaatimusmäärittely on ollut laaja, ja sen perusteella määritetään rajattu toteutus työmääräarvioineen, voi käydä niin, että toteutuksen kuvauksessa kerrota selkeästi, mitä toiminnallisuuksia määrittelystä on tarkoitus toteuttaa ja mitä ei sillä työmäärällä.
  - Asiakas voi pettyä tulokseen.

25

Erityistilanteita

- Määrittelyyn liittyy paljon riippuvuuksia kuten useita sovelluksia integraatioineen
  - Tilanteen tutkiminen perin pohjin
  - Riippuvuuksien kuvaaminen
  - Asiantuntijoiden kuten tuotekehitystiimin arvio
- Riippuvuuksista ei vielä tietoa
  - Määrittelyä tehdessä ei vielä ole tiedossa miten joku määriteltävään ominaisuuteen liittyvä asia / sovelluksen osa tullaan ratkaisemaan
  - Voi määritellä mitä ominaisuuksia puuttuvasta osasta pitää löytyä määriteltävän ominaisuuden tarpeisiin sitten, kun osan määrittelyä viedään eteenpäin

26

Erityistilanteita

- Kiireellinen määrittely
  - Kiireellinen tai kriittinen virhe tuotannossa
  - Monipuolinen virheenkorjauksen määrittelytiimi asian ympärille
  - Ratkaisun määrittelyn katselmointi
- Vaatimusmäärittely jää hyllylle
  - Vaatimusmäärittely katselmoitu ja valmis mutta kehittämisprojektia ei käynnistetä
  - Jos ja kun joskus käynnistyy, vaatimusmäärittely voidaan tarkistaa ja päivittää ennen kuin se viedään backlogille
  - Asiakaskommunikaatio!

27

Kiitos!

28

Liite 4: Palautekyselylomake: Ketterä määrittelyprosessi -aineisto

Kyselyn tarkoituksena on kerätä palautetta Ketterä määrittelyprosessi -ohjeistuksen käyttökelpoisuudesta uusien vaatimusmäärittelijöiden perehdytyksessä. Aineistoon ei sisälly ”talon tapojen” perehdytystä, ne ajatellaan koulutettavan erikseen. Kysely on osa opinnäytetyötä, jossa tutkitaan ketterän vaatimusmäärittelyn ohjeistamista.

Arvioitava materiaali: ☐ Diasarja

☐ Tekstiaineisto

☐ Suostun, että vastauksiani hyödynnetään Päivi Paakkarin opinnäytetyössä.

Kyselyn vastaukset käsitellään nimettöminä eikä vastaajan henkilötietoja tai muita tunnistetietoja tallenneta.

Kysymykset

Arvioitava materiaali soveltuu mielestäni uuden vaatimusmäärittelijän perehdytykseen (rengasta oikea vaihtoehto)

1. Erittäin huonosti
2. Huonosti
3. Kohtalaisesti
4. Hyvin
5. Erittäin hyvin

Miten materiaalia voisi mielestäsi parantaa?