Tampereen ammattikorkeakoulu

Tietoturvallinen ohjelmistokehitysprosessi

Ohjelmistoyritys vastaa uuden vuosikymmenen tietoturvavaatimuksiin

Juha Reiman

Opinnäytetyö
Lokakuu 2020
Tietojärjestelmäosaaminen, ylempi AMK

TIIVISTELMÄ

Tampereen ammattikorkeakoulu
Tietojärjestelmäosaaminen, ylempi AMK

REIMAN, JUHA:
Tietoturvallinen ohjelmistokehitysprosessi
Ohjelmistoyritys vastaa uuden vuosikymmenen tietoturvavaatimuksiin

Opinnäytetyö 97 sivua, joista liitteitä 14 sivua
Lokakuu 2020

Tietosuoja ja tietoturva ovat 2020-luvun muotisanoja ja syystä. Maailma on vuoden 1995 World Wide Web:n julkaisun jälkeen muuttunut enemmän ja enemmän teknologisesti yhdistetyksi. Samaan aikaan ohjelmistoteollisuuden perinteiset vahvaan määrittelyyn pohjaavat menetelmät ovat korvautuneet uusilla, jatkuvaan muutokseen kykenevillä ketterillä menetelmillä. Panostettaessa ketteryyteen joudutaan vahvistamaan huomattavasti aikaisempaa enemmän myös vaatimusten hallintaa, tai lopputuloksena on hallitsematonta eri suuntiin tapahtuvaa ohjelmistokehitystä. Niin teknologisesti yhdistynyt maailma, ketterät menetelmät ja vaatimusten hallintakin, antavat kaikki omat haasteensa modernille tietoturvalliselle ohjelmistokehitysprosessille.

Tämän opinnäytetyön toimeksiantaja oli ohjelmistoyritys Accountor HR Solutions Oy. Yritys toimittaa työelämän ohjelmistoalustaa, joka koostuu palkka-, henkilöstö- ja työvuoronhallintaohjelmistoista. Työn tarkoitus oli kartoittaa yrityksen kypsyystaso tuotekehityksen kokonaisprosessin osalta ja tavoitteena löytää tapoja tietoturvan parantamiseen kyseisessä prosessissa. Tutkimus toteutettiin tapaustutkimuksena, joka koostui haastatteluista, niiden analysoinneista sekä yrityksen aiemmin toteutettuun materiaaliin perehtymisestä ja päivittäisen työn havainnoinnista.

Tutkimuksessa löydettiin selkeää konsensusta tietoturvatyön toteuttamisen tärkeydestä sekä tapoja, joilla tietoturvaa voidaan kokonaisprosessissa parantaa. Tämän lisäksi tunnistettiin sellaisia rooleja organisaatiokarttaan, jotka pystyvät viemään tietoturva-ajattelua eteenpäin, ja osa-alueita, jotka vaativat kaikista eniten tekemistä, jotta voidaan todeta tuotekehityksen olevan niin tietoturvallista kuin mahdollista.

Tuotekehitysprosessin ollessa tarkkaan määritelty ja rooleilla vastuualueet selkeytetyt on pohja luotu tietoturvalliselle ohjelmistokehitykselle. Kun niihin lisätään organisaation omaksuma ketterän kehitysmallin mukainen toiminta ja selkeästi ja loogisesti ajateltu vaatimusten hallinta, voidaan todeta koko tuotekehityksen toimivan tietoturvallisesti ja omalta osaltaan varmistavan yrityksen menestystä halutuilla markkinoilla.

---

tietoturva, tietosuoja, ketterä, määrittely, vaatimus, ohjelmistokehitys

ABSTRACT

Tampereen ammattikorkeakoulu
Tampere University of Applied Sciences
Master'-s Degree Programme in Information Systems Competence

REIMAN, JUHA:
Information Secure Development Process
Software Company's Response to the Information Security of the New Decade

Master'-s thesis 97 pages, appendices 14 pages
October 2020

Information security and data privacy are 2020 buzzwords and for good reason. After the release of World Wide Web in 1995, the world has become increasingly technologically interconnected. This has risen the demand for information secure methods in software development. The need for constant change trough agile software development methods is also challenging the traditional way of taking care of the information security and requirements management.

The objective of this study is to find relationships between information secure software development, requirements management and agile development methods. This study was done as a case study that involved 14 interviews, getting acquainted with an ICT company's materials on subject, and observing the day-to-day working in the company. The theoretical section benchmarked methods of the software industry, and the scientific studies behind them.

The collected data were analyzed with qualitative methods in order to discover the connections between the topics of the study. Most of the interviewees believed that investing in information secure software development is an investment for the future. The findings indicate that creating information secure software is possible when given enough time and recourses.

---

information security, data privacy, agile, requirement, definition

SISÄLLYS

LYHENTEET JA TERMIT

Ketterä malli	Muutoksiin reagoiva projektimalli
Vesiputousmalli	Perinteinen projektimalli
Scrum	Yksi ketteristä projektimalleista
Kanban	Yksi ketteristä projektimalleista
Scrumban	Yksi ketteristä projektimalleista
Lean	Yksi ketteristä projektimalleista
Tuoteomistaja	Palvelun tai sovelluksen liiketoiminnallinen omistaja
Arkkitehti	Palvelun tai sovelluksen suunnittelija
Pyrähdys	Ajanjakso
Päivittäinen	Päivittäin toistuva palaveri ketterässä prosessissa
Katselmointi	Toistuva palaveri ketterässä prosessissa
Tietosuojavastaava	GDPR:n noudattamista valvova taho
DevOps	Kokonaisprosessin automatisointi
DevSecOps	Kokonaisprosessin automatisoinnin tietoturva
CI	Jatkuva integraatio
CD	Jatkuva asennus
Applikaatio	Mobiilisovellus
Rekisteri	GDPR Henkilötietoja sisältävä tietovarasto
Rekisterinpitäjä	GDPR:n määrittelemä rekisterin määrittelijä
Käsittelijä	GDPR:n määrittelemä rekisterin käsittelijä
Vaatimus	Ohjelmistolle esitettävä toiminnallinen tarve
Reunaehto	Ohjelmiston käyttöympäristön vaatimus

1 JOHDANTO

Todistamme tällä hetkellä yhä merkittävimmissä määrin maailmaa, jossa perinteiset asiakkaan omilla palvelimilla sijaitsevat, Internetiin kytkemättömät sovellukset jäävät historiaan ja tilalle tulee ohjelmistopalveluina pilvestä ajettavat sovellukset. Tämän lisäksi maailmaa on ottamassa haltuun sukupolvi, joka on tottunut sovellusten toimimiseen päätelaite riippumattomasti, Internet-selaimella ja erilaisilla applikaatioilla. Samaan aikaan viranomaiset tiukentavat otettaan kansalaistensa tietosuojasta, esimerkiksi Euroopan unionin tietosuoja asetuksen muodossa. Kaikki edellä mainittu aiheuttaa täysin uudenlaisen tilanteen myös yritysmyyntiin keskittyneille ohjelmistoyrityksille. Tietosuoja ja tietoturva ei enää olekaan kevyesti otettava aihealue, vaan sen ylläpitämiseen ja toteuttamiseen halutaan panostaa ja tällä tavalla pyrkiä estämään maineriskin tai erilaisten sanktioiden realisoituminen.

Samaan aikaan, kun ympäröivän maailman vaatimukset sovelluksista ja niiden toiminnasta monimutkaistuvat ja tehostuvat, myös asiakkaiden vaatimus sovelluskehityksestä tiukentuu. Enää ohjelmistokehittäjät eivät istu omassa erillisessä maailmassa, johon toimitetaan yksityiskohtainen määrittely toteutettavaksi ja josta vuosia projektin aloittamisen jälkeen toimitetaan asiakkaalle jotain hämärästi haluttua ohjelmistoa muistuttavaa, jonka käyttäminen on pahimmillaan maratonin viimeisiin kilometreihin verrattavaa taistelua. Nykypäivän asiakkaat olettavat saavansa ohjelmiston ominaisuuksiin sisällytetyt vaatimukset toteutukseen aina, oli ominaisuuden kehityksen aikataulu mitä tahansa. Tähän huutoon vastatakseen ohjelmistoala on kehittänyt käyttöönsä niin sanotut ketterät menetelmät. Näissä menetelmissä on sisäänkirjoitettuna käsitys, että siihen asti, kunnes on täysin valmista, kaikki voidaan vielä muuttaa.

Muutoksen mahdollisuus ja kehityksen nopeus näyttäytyvät peittelemättä vaatimusten hallinnan hankaluudessa. Vanhat Microsoft Excel ja Word listaukset vaatimuksista alkavat olla auttamatta liian hankalia ylläpitää ja sen myötä vaatimusten ylläpidon ammattimaiset työkalut yleistyvät jatkuvasti. Vaatimusten ollessa jatkuvien muutosten kourissa, ketterien menetelmien

varmistaessa tiheällä aikavälillä mahdollisuuden julkaista uusia versioita uusilla ominaisuuksilla, on vain luonnollista myös modernisoida ohjelmistokehityksen tekninen ympäristö ja automatisoida kaikki se mikä on automatisoitavissa moderneilla DevOps tekniikoilla. Tämä jatkuva mahdollisuus muutokseen ja alati monimutkaistuva tekninen ympäristö, on tietosuojan ja tietoturvan näkökulmasta suuri haaste ohjelmistoyritykselle. Lopulta kuitenkin kyseessä on niin tärkeä ja ohjelmistoyrityksen ydintekemistä koskettava haaste, että sen selättäminen on yritykselle jatkuvuuden ehto.

2 TAVOITTEET, TARKOITUS JA TUTKIMUSKYSYMYKSET

Opinnäytetyö on toteutettu Accountor HR Solutions Oy:n toimeksiannosta. Accountor HR Solutions Oy on yritys, jonka tuotevision mukainen tavoite on toimittaa Suomen paras työelämän ohjelmistoalusta. Käytännössä tämän toteutus tarkoittaa palkka-, henkilöstö- ja työvuorohallinnon ohjelmistojen kehitystä osaksi yhä tehokkaampia prosesseja ja kohti yhä tyytyväisempiä loppukäyttäjiä. Yrityksen toimialan takia, vaikka tuotteiden myynti kohdistuu vain toisille yrityksille eikä kuluttajakauppaa ole käytännössä olemassa, on yritys niin sanotun työntekijäkokemuksen digitalisaation eturintamassa.

Yrityksen kanssa määritellyssä kehittämistehtävässäni ”Asiakastarpeesta toimivaksi ominaisuudeksi” oli tarkoitus toteuttaa organisaatiolle käyttöön nykyaikainen tuotekehitysprosessi. Asiakaskentän jatkuvasti muuttuessa, tunnistettiin haasteelliseksi tarve oman prosessin kehittämiselle ja sen jatkuvalle parantamiselle. Samalla pyrittiin varmistamaan oman yrityksen säilyminen houkuttelevana työntekijöiden näkökulmasta. Prosessin tulisi lisäksi mahdollistaa tuotekehityksen skaalaamisen suuremmaksi ulkopuolisilla alihankintatiimeillä, sekä mahdollisilla fuusioilla ja yritysostoilla. Kehittämistehtävässä huomioitiin koko tuotekehitysketju annetusta asiakasvaatimuksesta tuotantokäyttöön toimitettuun ominaisuuteen asti. Tavoite oli arvioida ja pohtia onko käytössä oleva prosessi nykyaikainen ja toimiva. Tämä tuli toteuttaa nimenomaisesti vaatimusten hallinnan näkökulmasta eli miten yhdistää toimiva vaatimusten hallinta nykypäivän ketterässä mallissa toimivaan tuotekehitysprosessiin.

Opinnäytetyöni tarkoitus oli kattaa kehittämistehtävästä vaatimusten hallinnan ja ketterän kehitysprosessin osuus tutkimalla organisaation tämänhetkistä kypsyystasoa kyseisissä aihealueissa, sekä tuoda lisänä mukaan pohdintaa tietoturvan näkökulmasta. Opinnäytetyö on toteutettu kehittämistehtävästä erillisenä omana toteutuksenaan ja sen tavoite on antaa ymmärrys, miten vaatimusten hallinnassa ja tavoitellussa ketterässä ohjelmistokehitysprosessissa pystytään huomioimaan tietoturva. Minkälaisia rooleja ja vastuita se vaatii, sekä minkälaisia ylläpitoa helpottavia

toimintamalleja siihen löytyy. Näiden toimien jälkeen on oletettavaa, että toimeksiantajan tuotekehitysprosessi paitsi kestää nykypäivän asiakkaiden vaatimukset, myös antaa turvallisen selkänojan tietoturvan ja tietosuojan näkökulmasta.

Opinnäytetyöni tutkimusongelma muodostuu siis seuraavista kysymyksistä:

1) Miten tietoturvallisen ohjelmistokehityksen vaatimukset tulevat osaksi ketterää kehitysprosessia?
2) Mitä hyötyjä tai haasteita kehitystiimin jäsenet kokevat tietoturvallisessa ohjelmistokehityksessä olevan?
3) Miten luotua tietoturvallisen ohjelmistokehityksen mallia voidaan ylläpitää ja kehittää?

Opinnäytetyö itsessään siis eroaa alkuperäisestä kehittämistehtävästä tuoden mukaan näkökulman tietoturvasta. Kehittämistehtävän osalta on pohdittu enemmän vaatimusten hallinnan roolia ja moderneja ketteriä tuotekehitysprosesseja.
Kehittämistehtävän osalta pyrittiin löytämään vastaus seuraavanlaisiin kysymyksiin:

1) Miten vaatimusten hallinta tulisi toteuttaa osaksi ketterää kehitysprosessia?
2) Miten organisaatio saadaan toimimaan ketterän kehitysprosessin vaatimalla tasolla?

Tässä työssä on vastattu niin opinnäytetyön tutkimusongelmaan, kuin myös kehittämistehtävässä käsiteltäviin aihealueisiin, niiden ollessa keskenään limittyneitä ja toisiaan tukevia. Kehittämistehtävän tavoitteena oli tehostaa ja selkeyttää yrityksen ohjelmistokehitysprosessia haastatteluissa saatujen tulosten pohjalta, sekä toimia mallina nykyaikaisen ohjelmistokehitysprosessin perusteille.

3 OPINNÄYTETYÖN TUTKIMUSMENETELMÄT

Toimeksiantooni liittyvä kehittämistehtävä lähti tilanteesta, jossa organisaatiossa oli muutamia vuosia aikaisemmin siirrytty, käytännössä prosessittomasta muutaman henkilön osaamisen ylläpitämästä tuotekehityksestä, Scrum-mallin pohjalta luotuun ketterää mallintavaan tuotekehitysprosessiin. Prosessia jalkautettaessa ja käytännön kokemuksen kertyessä, oli huomattu tarve parantaa prosessin tiettyjä osa-alueita.

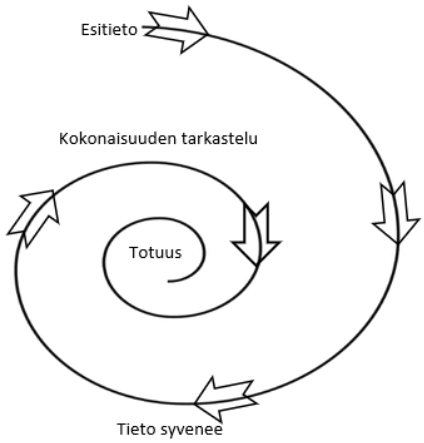
Yritykseen oli uuden mallin lanseerauksen mukana pesiytynyt ajattelumalli, ”tuotekehitysprosessi on nyt nykyaikainen Scrum ja se hoitaa homman”. Tämän ajattelun paikkansa pitävyyttä alettiin ajan kuluessa lisää ja lisää haastamaan. Lopulta suurimmaksi ongelmakohdaksi näytti hahmottuvan vaatimusten hallinta ja siihen liittyvät prosessin osat. Samaan aikaan yrityksen paikallisten muutostarpeiden kanssa, koko ohjelmistoteollisuus koki valtavaa siirtymää vanhoista lisenssimalleista kohti pilvipalveluja ja SaaS maailmaa. Nämä muutosvoimat yhdessä antoivat suuren paineen prosessin kehittämiseen siten, että kehityksessä huomioonotettaisiin vaatimusten hallinnan aiheuttamien muutosten lisäksi myös yhä enemmän koko prosessin tietoturva. Näin varmistettaisiin sekä asiakastarvetta tyydyttävä, että läpi koko kehitysputken tietoturvallinen ohjelmistopalvelu.

Tämän työn ja tutkimuksen tavoite oli saada selville, mikä on organisaation kypsyystaso tietoturvassa, ketterissä menetelmissä ja vaatimusten hallinnassa. Lisäksi haluttiin selvittää, miten jokaista osa-aluetta pystytään tulevaisuudessa kehittämään siten, että yrityksen maturiteetti aihealueiden osalta pystytään varmistamaan myös tulevaisuudessa. Menetelmäkartta tutkimukselleni löytyy taulukosta yksi.

TAULUKKO 1. Menetelmäkartta

Tieteenkäsitys	Hermeneuttinen
Tutkimustyyppi	Kvalitatiivinen
Tutkimusmenetelmä	Tapaustutkimus
Tiedonkeruutekniikat	Haastattelut, havainnointi, valmiit aineistot ja dokumentit

Tieteen filosofian kannalta ajateltuna, työ on lähimpänä hermeneuttista, joka on pohdiskelevaa tiedettä. Hermeneuttiseen ajatteluun yhdistetään usein kvalitatiivinen tutkimus, jossa tulokset pohjautuvat usein tutkijan omaan pohdintaan ja ajatteluun. Ominaista on myös tutkimusprosessin lähteminen tietystä oletuksesta ja päätyminen niin sanotun hermeneuttisen kehän avulla hiljalleen kohti totuutta. (Eskola & Suoranta 1998, 22.) Tätä olen tuonut esille Kuviossa yksi.



KUVIO 1. Hermeneuttinen kehä

Tutkimustyypit voidaan jakaa kahteen eri kategoriaan, tilastolliseen tai laadulliseen. Tilastollisissa eli kvantitatiivissa tutkimusmetodeissa pyritään saamaan suuren vastaajajoukon avulla esille eroja, eri tutkimukseen osallistujien ja niistä luotujen tutkimusyksiköiden välille (Alasuutari 2011, 26). Tutkimuseettisesti tilastollisissa menetelmissä ollaan varmemmalla pohjalla, koska tutkimuksessa käytetään paljon tieteen tulkintasääntöjä. Selkeitä ominaispiirteitä tilastolliselle tutkimukselle on kausaalisuhteiden esittäminen, tutkimuksen aloittaminen hypoteeseista ja teoriasta, päättelyn logiikan deduktiivisuus, sekä luvut ja tiedon objektiivisuus. (Kananen 2008, 27.) Tämän

lisäksi kvantitatiivisessa tutkimuksessa tulosten analysoinnissa yhteiset tulokset eivät anna johtolankoja tutkittavista ilmiöistä. Kaikille tutkimusyksiköille yhteiset asiat kuuluvat osaksi perusjoukkoa ja se, mikä on perusjoukossa, rajataan ulos tutkimuksissa tehtävistä yleistyksistä (Alasuutari 2011, 26).

Laadullisilla eli kvalitatiivisilla tutkimusmetodeilla tarkoitetaan käytännössä kaikkia niitä tutkimuksia, jotka eivät ole tilastollisten menetelmien mukaisia. Tutkimusprosessi on syklinen ja se ei sisällä tilastollisten menetelmien mukaisia tiukkoja tutkimussääntöjä. Aineisto ei koostu pelkistä luvuista, vaan tekstistä sen eri muodoissa (Kananen 2008, 24). Tutkimuksessa koostettuja aineistoja ja tutkimukseen osallistuneita tutkimusyksikköjä tarkastellaan kokonaisuutena ja pyritään selvittämään, mikä on mitattava kokonaisuus analyysin taustalla (Alasuutari 2011, 28). Kokonaisuudessa tulosten analyysi ei ole tutkimuksen viimeinen vaihe, vaan koko tutkimusprosessin osa, joka itsessään ohjaa kohti totuutta (Kananen 2008, 24). Toisin kuin kvantitatiivisessa menetelmässä, laadullisessa nimenomaisesti yritetään löytää tutkimuksessa esitetyn tulkinnan mukaisia yhteneväisyyksiä (Alasuutari 2011, 31). Ominaispiirteitä laadulliselle tutkimukselle on esimerkiksi tulkinta ja toimijan ymmärtäminen, tutkimuksen päättyminen hypoteeseihin ja teoriaan, päättelyn logiikan induktio ja kaiken tietynlainen subjektiivisuus, missä tutkija itse on instrumentti (Kananen 2008, 27).

Tässä työssä on haastateltavilta kerätty myös muutamia vastauksia, joita analysoidaan kvantitatiivisilla menetelmillä. Työ kuitenkin keskittyy laadulliseen tutkimukseen ja analysointiin kvalitatiivisilla perusteilla.

Tutkimukselle tähän mennessä tehtyjen rajausten ja linjausten pohjalta on tehtävä valinta muutaman yleisen menetelmän väliltä. Menetelmiä ovat esimerkiksi kokeellinen-, survey- ja tapaustutkimus. Näistä edellä mainituista vaihtoehdoista tutkimukseen valikoitui yksiselitteisesti tapaustutkimus, sen sisältämien pienen joukon tapauksien analysointiominaisuuksien takia. (Eskola & Suoranta 1998, 49.)

Yksinkertaisimmillaan laadullisen tutkimuksen aineistolla tarkoitetaan materiaalia, joka on vain tekstiä, sillä mistä teksti on alun perin peräisin, ei ole

merkitystä tutkimuksen kannalta. Esimerkkeinä eri menetelmistä ja tekniikoista voisi pitää haastatteluja, havainnointia, äänimateriaalia, kuvamateriaalia ja vastaavia. (Eskola & Suoranta 1998, 13.) Tiedonkeruumenetelminä tutkimuksessani käytin havainnointia, haastatteluja, sekä valmiita aineistoja ja dokumentteja. Havainnointiin sisältyy päivittäinen työskentelyni osana yrityksen tuotekehitysyksikköä. Tämän toiminnan perusteella lokeroin tekniikkani osaksi osallistuvaa havainnointia, jossa tutkija on osana tutkittavaa ympäristöä ja osallistuu sen toimintaan. (Kananen 2008, 70.) Haastatteluissa puolestani käytin puolistrukturoitua mallia, jossa kysymykset, haastattelutilanne ja haastatteluun käytetty aika, olivat kaikille identtiset, mutta esimerkki vastauksia tai niiden aihioita ei annettu (Eskola & Suoranta 1998, 64). Valmiiden aineistojen ja dokumenttien osalta tutkimukseen sisältyy tässä tapauksessa erilaiset yrityksellä olevat ja sille tuotetut tutkimuksen aihealueita koskevat dokumentit. Aineistoa tulkitsen niin sanottuna sekundaariaineistona eli muiden keräämänä, mutta itseni analysoimana. (Eskola & Suoranta 1998, 87.)

Haastattelujen kysymykset rakensin koskemaan tutkimuksen kolmea eri osa-aluetta. Kategorisoin kysymykset niin, että ensin käytiin läpi kahdeksan yleistä kysymystä, joiden avulla selvitettiin haastatellun asemaa, sekä ymmärrystä kaiken pohjana toimivasta ketterästä mallista. Tämän jälkeen siirryttiin neljän kysymyksen ajaksi vaatimustenhallintaan ja lopulta yhdentoista kysymyksen verran tietoturvan aihealueeseen. Haastateltavat valitsin siten, että koko yrityksen ohjelmistokehityksen linja johdosta tuotekehittäjään tuli katettua. Lisäksi haastateltavissa oli kaksi asiantuntijaa tietosuojan ja tietoturvan vastuualueilta, jotka eivät suoraan liittyneet ohjelmistokehitykseen. Haastattelut toteutettiin kolmen viikon ajanjakson aikana, useampi haastattelu päivässä, käyttäen Microsoft Teams työkalua. Haastattelut tallennettiin Teams:n omalla tallennus mahdollisuudella ja litteroitiin kaikkien haastattelujen valmistuttua. Haastateltuja pyydettiin olemaan mainitsematta haastattelun aihealueesta ja kysymyksistä organisaation sisällä, jotta saataisiin mahdollisimman todenmukainen kuva organisaation ymmärryksen tasosta kyseisestä aihealueesta.

Haastattelujen, havainnoinnin, yrityksessä toteutettujen sisäisten kehityshankkeiden sekä tuotetun dokumentaation pohjalta,

tutkimuskokonaisuus muodostaa selkeän kuvan tilasta, joka organisaatiossa tällä hetkellä on vallassa. Tämän opinnäytetyön mukanaan tuomalla tiedolla ja päättelyllä pystytään taas rakentamaan selkeä tavoitetila, mihin kehitysprosessia mahdollisesti halutaan viedä tulevaisuudessa.

4 TIETOTURVA

4.1 Tietoturva ja tietosuoja

Viimeisen kahdenkymmenen vuoden aikana tätä aikakautta on kutsuttu yhä enemmän tiedon aikakaudeksi. Vanhojen, kaikille tuttujen kuluttajatuotteiden lisäksi kaikkien tietoisuuteen on noussut yrityksiä kuten Facebook, Google ja Twitter. Näistä yrityksistä yksikään ei tuota tai myy mitään konkreettista tuotetta, vaan palveluja, joita rakennetaan käyttäjästä kerätyn tiedon avulla. (Martin & Talabis 2013, 1.)

Tietoon keskittyvien yritysten lisäksi myös kaikki perinteiset yritykset ovat heränneet tiedon arvoon. Pankit, vakuutusyhtiöt, terveydenhuoltoyritykset ja erilaiset tuotantoyritykset keräävät nykyään järjestelmällisesti asiakkaistaan ja käyttäjistään dataa, jota voidaan sitten käyttää hyväksi esimerkiksi lisäarvopalveluiden myynnissä. Kääntöpuolena tämä kaikki tieto on haluttua kauppatavaraa ja sen seurauksena jatkuvan laittoman tiedustelun kohteena. (Shriram & Venkataramanan 2017, 20.)

Ihmisistä kerätty tieto on arvokasta ja nykypäivänä, kun tietoa on kerättynä todella paljon, tarkoittaa tämä valitettavasti myös sitä, että tiedon hankkiminen laittomin keinoin on jatkuvaa ja alati kehittyvää toimintaa. Tämän seurauksena on vakiintunut kaksi erillistä termiä: tietoturva ja tietosuoja (Tietosuojavaltuutettu n.d.). Euroopassa ja sitä myöten myös Suomessa, tärkeimpänä tämän aihealueen saavutuksena voidaan pitää EU:n vuonna 2016 säätämää ja vuonna 2018 voimaan tullutta tietosuoja-asetusta: GDPR, jonka tärkein viesti on antaa ihmisistä kerätyn tiedon omistusoikeus takaisin ihmisille itselleen ja määritellä kovat sanktiot huonosti ylläpidetystä tietosuojasta. (Visma 2019.)

Suomessa EU:n tietosuoja-asetusta valvovan tietosuojavaltuutetun toimisto määrittelee tietoturvan ja tietosuojan eron niin, että tietoturva on yksi tapa huolehtia tietosuojasta. Tietosuojalla siis tarkoitetaan kokonaisvaltaisemmin ihmisten ja organisaatioiden oikeutta itsestä kerättyyn tietoon. Toisin sanoen tietosuoja kattaa myös tiedon turvaamisen sellaisia tahoja vastaan, joilla olisi

muuten pääsy itse tietoon. Esimerkiksi ohjelmiston pääkäyttäjällä on pääsy kaikkiin ohjelmistossa sijaitseviin tietoihin, mutta ei kuitenkaan oikeutta mennä niitä katsomaan tai hyväksikäyttää kyseisiä tietoja. (Tietosuojavaltuutettu n.d.)

Yhdysvalloissa puolestaan jo itse laki määrittelee tietoturvan tavaksi suojella tietoa ja estää tietojärjestelmiä ei sallitulta sisäänpääsyltä, käytöltä, julkistukselta häirinnältä, muokkaukselta tai tuhoamiselta (Martin & Talabis 2013, 20). Yleisemmin voidaan ajatella tietoturvan tarkoittavan käytännössä keinoja huolehtia tiedon luottamuksellisuudesta, eheydestä ja saatavuudesta. Tällöin puhutaan niin sanotusta CIA mallista, joka tulee englanninkielisistä sanoista confidentiality, integrity ja availability. (GeeksforGeeks 2020.)

Tietoturvan CIA mallia voidaan vielä laajentaa neljänteen osa-alueeseen, jäljitettävyyteen. Tällä neljännellä osa-alueella tarkoitetaan mahdollisuutta selvittää, mitä tiedon luottamuksellisuudelle, eheydelle tai saatavuudelle mahdollisessa tietoturvaloukkauksessa tapahtuu. Luottamuksellisuudella tarkoitetaan varmistusta, että tietoa käsittelevät vain ne, joilla on oikeus tietoon. Eheydellä tarkoitetaan varmistusta, ettei tieto muutu ilman tarkoitusta ja jos muuttuu, niin se ainakin tiedostetaan. Saatavuudella tarkoitetaan mahdollisuutta saada tietoa aina kun sitä tarvitaan. (Digi, ja väestötietovirasto 2013, 15.)

4.2 Tietoturvallinen ohjelmistokehitys

Yhdeksänkymmentäluvulla tapahtunut Internetselaimien yleistyminen ja World Wide Webin tulo yleiseen käyttöön, aiheutti sitä seuranneiden vuosien aikana täysin uudenlaisen ongelman tietokoneohjelmistoja kehittäville yrityksille. Aikaisemmin ohjelmistot olivat käytössä vain yritysten sisäisissä verkoissa ja hallinnoituna omilla palvelimilla. Internetin yleistymisen ja kehittymisen ansiosta ohjelmistot pystyvät sijaitsemaan missä vaan ja niitä käytetään Internetin yli toimivilla käyttöliittymillä tai Internetselaimilla. (Sommerville 2006, 718.)

Jatkuvasti kasvavaa tietoturvauhkaa vastaan ohjelmistoyrityksissä alettiin miettimään ja toteuttamaan niin kutsuttua tietoturvasuunnittelua osana tuotekehitystä. Tämän suunnittelun tarkoituksena on huolehtia, että järjestelmät

kehitetään kestämään erilaisia Internetin yli tulevia vihamielisiä hyökkäyksiä ja tietojen kalastelu yrityksiä. Lopulta tietoturvasuunnittelu laajeni itse sovelluksen tietoturvallisesta arkkitehtuurista koskemaan myös sitä ympäristöä, missä sovellusta käytetään ja ylläpidetään. (Sommerville 2006, 719.)

Tietoturvasuunnittelu voidaan tiivistää kahteen ajatukseen:

1. Sovelluksen tietoturvan pitää olla yksi osa-alue järjestelmäarkkitehtien osaamisessa ja työnkuvassa.
2. Sovellusympäristön tietoturvan tulee olla osa ympäristön rakentajien osaamista ja työnkuvaa.

Tietoturvaa ei voi ajatella yhtenä tehtävänä, joka tehdään jossain vaiheessa ohjelmistoprosessia, vaan sen pitää yllä mainitun mukaisesti läpileikata koko ohjelmistoyrityksen ja mielellään myös yrityksen asiakkaiden ja sidosryhmien osaaminen. (Sommerville 2006, 719.)

Sovelluksen tietoturvassa tärkeää on itse tiedon suojaaminen. Suojaamisessa varsinkin kryptaaminen, anonymisointi ja symbolisointi ovat tavat huolehtia tiedon säilymisestä turvallisesti vain sellaisten tahojen saatavissa, kenellä niihin on oikeus. Nämä tavat ovat vakiinnuttaneet asemaansa ja jalkautuneet käytettäviksi keinoiksi tietoturvan saavuttamisessa. (Shriram & Venkataramanan 2017, 27.)

Tiedon salaaminen eli kryptaaminen on todennäköisesti vanhin tapa suojata tietoa ja perustuu siihen, että ilman tiettyä etukäteen jaettua avainta, ei tieto ole luettavassa muodossa. Tiedon salaaminen on tehokas tapa suojata tietoa, mutta se aiheuttaa ongelmia suorituskyvyssä. Tämän lisäksi avaimen turvallisuudesta on huolehdittava läpi koko tiedon suojaamisen prosessin. Toisena tapana on tiedon anonymisointi. Tällä tarkoitetaan tapaa tehdä jaettavasta tiedosta sellaista, mitä voi ilman riskiä, vaikka väärät tahot saisivat datan käyttöönsä, jakaa toisille sovelluksille tai muille, jotka tietoa tai sen osia tarvitsevat. Ongelmana anonymisoinnissa on pystyä pitämään tällainen tieto käyttökelpoisena. Viimeisenä tiedon suojauksen elementtinä, on varsinkin finanssimaailmassa tuttu, tiedon symbolisointi. Tässä mallissa sensitiivinen tieto korvataan symboleilla, jotka ovat vain itse aidon sovelluksen käytössä. Tämä

tapa suojata tietoa on erittäin tehokas, mutta estää tiedon käytön sovelluksen ulkopuolella. (Shriram & Venkataramanan 2017, 27.)

Moni tietoturvan osa-alueista pitää huomioida niin sovelluksen, kuin sovellusympäristönkin tietoturvassa. Käyttäjä- ja käyttöoikeushallinnan tulee olla sovellukseen suunniteltu ja ympäristön pitää tukea todellisia ja standardin mukaisia autentikointitapoja. Tällä tavalla voidaan varmistaa, että vain oikeat käyttäjät pääsevät kiinni niihin tietoihin, mihin heillä on oikeus. Sovellus tulee suunnitella ja toteuttaa niin, että se tukee nykyaikaista automaattista tuotantoon julkaisua ja ylläpitoa. (Sommerville 2006, 720.)

Tämän lisäksi ympäristön pitää, väliohjelmat mukaan lukien, olla rakennettu siten, että ilmiselvät tietoturvariskit ovat vältetty. Myös ympäristön eri osien sisältämien ohjelmistojen säännöllinen päivitys on tärkeää tietoturvan ylläpitämiseksi. Ympäristössä tulee olla käytössä erilaisia nykyaikaisia tarkkailutyökaluja, joilla mahdollisia hyökkäyksiä huomataan. Myös tietojen aktiivinen varmuuskopiointi on yhdenlainen tietoturvatehtävä. (Hyrynsalmi, Leppänen & Rindell 2015, 62.) Taulukossa kaksi on tuotu esille erilaisia tietoturvasuunnittelussa huomioon otettavia ohjelmiston käytön alueita.

TAULUKKO 2. Tietoturvariskin sisältäviä sovellusympäristön alueita

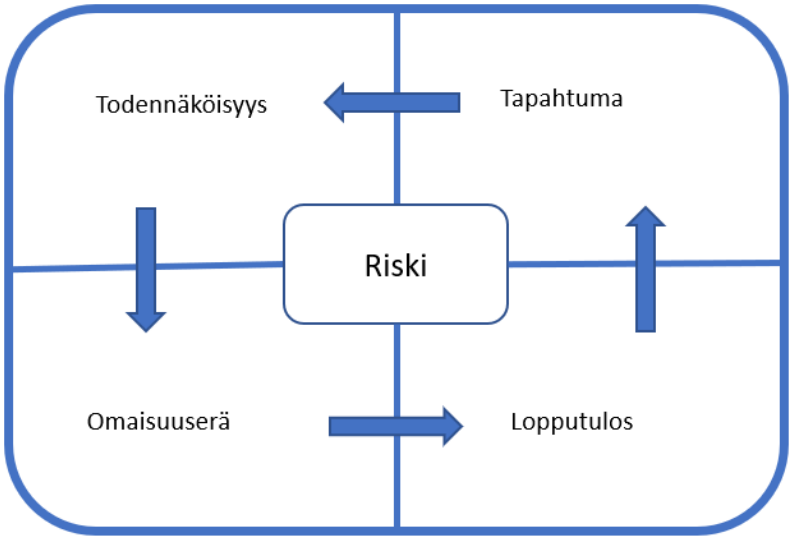
Sovellus
Uudelleenkäytettävät komponentit ja kirjastot
Väliohjelmistot
Tietokannan hallinta
Yleiset jaetussa käytössä olevat sovellukset
Käyttöjärjestelmä

Ohjelmiston ja sen toimintaympäristön tietoturvasuunnittelua tehtäessä on tärkeä ymmärtää erilaisia uhkia, mitä ohjelmiston käytöllä on olemassa. Nämä uhkat ja niiden mallinnukset voivat vaihdella, riippuen siitä minkälaisesta ohjelmistosta ja ympäristöstä on kyse, mutta pääpiirteittäin ne voidaan jakaa kolmeen eri kategoriaan:

1. Ohjelmiston sisältämän tiedon luottamuksellisuuden riski, jossa vihamielinen voima saa haltuunsa ohjelmiston sisältämää luottamuksellista tietoa, käyttäen jotain tapaa tai haavoittuvuutta ohjelmistossa.
2. Ohjelmiston sisältämän tiedon ehjyyden riski, jossa hyökkääjät pystyvät jollain tavalla tuhoamaan tietoa niin, että ohjelmiston käyttäminen ei ole enää tarkoituksenmukaista tai sen käyttöön saattaminen on hidasta ja kallista.
3. Ohjelmistoon pääsyn riski, jonka realisoituessa uhkaavat tahot ovat pystyneet estämään ohjelmiston oikeiden käyttäjien pääsyn ohjelmaan ja näin aiheuttamaan ongelmia.

Kaikki nämä riskit voivat realisoitua yhtä aikaa tai erikseen ja jokaisen riskin realisoituminen voi aiheuttaa myös seuraavan riskin realisoitumisen. Esimerkiksi laillisten käyttäjien pääsyn ollessa estetty ohjelmistoon, ei kyseisen käyttökatkon aikana pystytä lisäämään tai päivittämään järjestelmän sisältämiä tietoja ja näin järjestelmään tiedon ehjyys vaarantuu. (Sommerville 2006, 22.)

Riskissä itsessään kannattaa huomioida se, miten riski rakentuu. Riski sisältää neljä perusosaa, joista se kokonaisuudessaan muodostuu. Nämä neljä perusosaa ovat: tapahtuma, todennäköisyys, omaisuus ja lopputulos. Tietoturvasta puhuttaessa käytettäviä termejä taas on kolme, jotka käytännössä ovat täysin samoja asioita kuin riskissä yleisestikin. Termit ovat: uhka, heikkous ja vaikutus.
Uhka vastaa riskin tapahtumaa. Heikkouksilla tarkoitetaan riskin todennäköisyyttä. Vaikutuksen synonyyminä riskissä on lopputulos. Nämä kolme konseptia pitävät sisällään kaiken sen, mitä tietoturvariskin mallinnuksessa tulee ottaa huomioon. (Martin & Talabis 2013, 11.) Kuviossa kaksi on kuvattuna ne komponentit mistä perinteinen riski koostuu.



KUVIO 2. Riskin komponentit

Ohjelmistokehityksen osalta tietoturvariskit tulee arvioida ja käsitellä laajemmin, kuin mikä koskee vain kehitysyksikköä. Mahdollisista uhkista tulee rakentaa uhkamallinnus, jonka rakentamisen tulee olla osa kaikkia niitä prosessin alueita, joita tuotteen suunnitteleminen, toteuttaminen, myyminen tai käyttöönottaminen koskee. Tämän perusteella voidaan todeta, että tietoturvan kanssa toimiminen on aina läpileikkaus koko organisaatiosta ja sen toiminnasta. (Martin & Talabis 2013, 12.)

Luonnollinen aloitus tietoturvan uhkamallinnukselle on toteutettavien tuotteiden ja palveluiden pohtiminen liiketoiminta-alueiden lakeihin ja asetuksiin verraten. Toimittaessa Euroopan Unionin alueella, tulee luonnolliseksi osaksi tietysti artikla 2016/679:n mukainen yleinen tietosuoja-asetus eli GDPR. Tietosuoja-asetus itsessään on tuonut tietosuojaan joitain termejä, jotka lähes välittömästi vakiintuivat asianomaisten käyttöön. Näitä termejä selvennetään asetuksen neljännessä artiklassa. Ensinnäkin on olemassa rekisteri, jolla tarkoitetaan mahdollisuutta saada jäsenneltyjä henkilötietoja jonkun perusteen avulla. Rekisteri ei ole suoraan verrannollinen tietokantaan vaan käsitteenä laajempi. Toiseksi on olemassa rekisterin pitäjä. Tämä taho määrittelee millä perusteella rekisteriin tieto kerätään ja mitä tietoja sinne kerätään. Kolmanneksi on henkilötietojen käsittelijä, taho, jolla on pääsy rekisterin pitäjän rekisteriin, käsittelemään siellä olevia tietoja. Neljäntenä on rekisteröity, jolla tarkoitetaan

luonnollista henkilöä, jonka tietoja rekisterin pitäjä rekisteriin kerää ja joita mahdollisesti henkilötietojen käsittelijä käsittelee. (Asetus 2016/679/EU)

Tietosuoja-asetuksen tärkein anti on yhtenäistää koko Euroopan Unionin alueen käytännöt oikeiden henkilöiden henkilökohtaisen tiedon hallinnasta ja varmistaa kohtalaisen suurien sanktioiden avulla kaikkien yritysten, jotka henkilökohtaisia tietoja hallitsevat, noudattavan tietosuoja-asetuksen artikla viiden mukaisia periaatteita. Periaatteet sanovat, että tietoja on käsiteltävä hyvin, lainmukaisesti ja tietojen omistajalle läpinäkyvästi, sekä varmentaa, että kerätty tieto on kerätty oikeaa tarkoitusta varten eikä vain varmuuden vuoksi. Tiedon kerääjän on myös varmistettava, että tiedot ovat asianmukaisia ja olennaisia siihen tarkoitukseen, mihin niitä kerätään ja varmistettava, että tiedot säilytetään sellaisella tavalla, että niissä pystytään säilyttämään tietoturvan periaatteiden mukainen eheys ja luottamuksellisuus. (Asetus 2016/679/EU)

Rekisteröidylle tietosuoja-asetus antaa artiklassa 15 vain oikeuksia, ei velvollisuuksia. Rekisteröidylle annetaan oikeus saada aina tietoonsa, mitä tietoja hänestä on kerätty, mukaan lukien ne tahot keille tietoa on annettu tai näytetty. Rekisteröidyllä on siis oikeus saada tietää kaikki ne tahot, jotka ovat päässeet näkemään hänen henkilökohtaisia tietojansa. Tämän lisäksi rekisteröidyllä on oikeus saada varmistus, että kaikki se tieto, jota rekisterin pitäjä ei enää tarvitse mihinkään on pseydonymisoitu niin, että rekisteröityä ei voi niistä enää tunnistaa. Edellä mainitun lisäksi rekisteröidyllä on oikeus saada tietonsa poistettua rekisterin pitäjän rekisteristä ja mahdollisesti muokattavan, jos rekisteröity näkee tiedon olevan väärin. (Asetus 2016/679/EU) Artiklan 15 vaikutusaluetta on rajoitettu artiklassa 23 niin, että se ei koske sellaisia rekisterin pitäjiä ja rekistereitä, joiden ylläpidolla pyritään takaamaan kansallista turvallisuutta, puolustusta, yleistä turvallisuutta tai vastaavia julkisia toimia. (Asetus 2016/679/EU)

Rekisterin pitäjälle ja henkilötiedon käsittelijälle tietosuoja-asetus tuo useita erilaisia vastuita artiklassa 25. Näistä yksi tärkeimmistä on ajatus oletusarvoisesta tietosuojasta, joka ohjaa tietokoneohjelmistoja kehittäviä yrityksiä omaksumaan tietoturvan osaksi jokapäiväistä toimintaa ja näin mahdollistamaan myös rekisterin pitäjän tehokkaamman tietosuojasta

huolehtimisen. (Asetus 2016/679/EU) Erittäin suuri muutos aikaisempaan on myös tietosuoja-asetuksen määräämä tietosuojan vaikutustenarviointi, jota käsitellään artiklassa 38. Tällä arvioinnilla rekisterin pitäjän tulee ottaa huomioon tietosuoja aina, kun kerätty tieto sisältää sensitiivistä tietoa henkilöstä tai mahdollisuuden tunnistaa henkilö kerätystä tiedosta. (2016/679/EU) Suuri lisäys verrattuna aikaisempaan on myös tietosuoja-asetuksen määräys tietosuojavastaavasta. Tietosuojavastaava on taho, joka tulisi löytyä kaikista niistä yrityksistä, jotka toimivat tietosuoja-asetuksen mukaisina rekisterin pitäjinä tai henkilötietojen käsittelijöinä. Tietosuojavastaavan tarkoituksena on toimia tietosuoja-asetuksen asiantuntijana organisaatiossa, sekä linkkinä tietosuoja-asioissa rekisteröityjen ja tietosuojaviranomaisten välissä. (Asetus 2016/679/EU)

4.3 Tietoturva ohjelmistokehitysprosessissa

Suunniteltaessa tietokoneohjelmistoa, ei tietoturva ja tietosuoja voi olla vain yksi tehtävä tai ominaisuus, joka lisätään ohjelmistoon, vaan ohjelmiston koko tuotekehitys, käyttöönotto ja ylläpitoketju tulee toimia tietoturvallisten periaatteiden mukaisesti, vaikka luonnollisesti itse ohjelmiston suunnittelu tietoturvallisella tavalla on erittäin tärkeää. Asian voi ilmaista siten, että käyttöönoton ja tuotannon virheet tietoturvassa voivat pilata hyvin suunnitellun ohjelmiston tietoturvan, mutta huonosti tietoturvan puolesta suunniteltu ohjelmisto ei koskaan ole tietoturvallinen. (Sommerville 2006, 729.) Sommerville nostaa Software engineering kirjassaan esille, että ei ole olemassa mitään kaiken kattavaa ohjetta, kuinka suunnitella tietoturvallisia ohjelmistoja. Tämä johtuen siitä, että ohjelmistot ja niiden käyttöympäristöt ovat useasti hyvin erilaisia (2006, 730).

Siitä huolimatta, että kaiken kattavaa ohjetta ei ole olemassa, voidaan todeta olevan olemassa kymmenen erilaista kokonaisuutta, jotka huomioon ottamalla päästään huomattavasti tietoturvallisempaan kokonaisuuteen. Tämän listauksen kaltainen kokonaisuus on hyödyllinen varsinkin siksi, että organisaatioilla on tapana keskittyä lyhyen tähtäimen maaleihin, jolloin tietoturva on vain ylimääräinen, resursseja kuluttava ja tuottamaton

kokonaisuus. Lisäksi tämän kaltaisesta ylätasolla olevasta listauksesta voidaan tarvittaessa johtaa tarkempia huomioita ja kysymyksiä. (Sommerville 2006, 731.) Sommervillen kymmenen kokonaisuuden listaus on kuvattuna taulukossa kolme.

TAULUKKO 3. Tietoturvallisen suunnittelun muistilista

1	Pohjaa tietoturvapäätökset tarkkaan tietoturvakäytäntöön
2	Vältä yksittäistä pistettä, jonka häiriö vaarantaa koko järjestelmän
3	Varmista epäkuntoon meno turvallisesti
4	Tasapainota tietoturva ja käytettävyys
5	Huomioi mahdollisuus käyttäjän manipulointiin
6	Käytä päällekkäisyyttä ja monimuotoisuutta vähentämään riskiä
7	Tarkista kaikkien syötteiden oikeellisuus
8	Lokeroi resurssit
9	Suunnittele käyttöönottoa varten
10	Suunnittele toimintaan palauttamista varten

Tietoturvakäytännöllä tarkoitetaan korkean tason päätöstä, jolla säädetään tietoturvan perusteet koko organisaation käyttöön. Sillä siis pyritään selittämään, mitä tietoturva organisaatiolle tarkoittaa. Periaatteessa tämä tarkoittaa sitä, että kaikki ne periaatteet, mitkä tietoturvakäytännössä on mainittu, tulisivat olla jollain tavalla osa järjestelmän suunnitteluun kohdistuvia vaatimuksia. Ketterissä kehitysmalleissa tämä voi olla hyvin hankalaa ylläpitää, koska ohjelmiston käytössä tarpeelliset ominaisuudet muuttuvat niin nopeasti. Tästä syystä arkkitehdeillä ja muilla ohjelmiston suunnittelusta vastaavilla tulisi olla niin hyvä käsitys tietoturvakäytännöstä, että se muodostaa suunnittelulle itsessään viitekehyksen, jonka sisällä päätökset voidaan tehdä. (Hyrynsalmi ym. 2016, 556.)

Tietokoneohjelmistoissa on aina järkevää pyrkiä lopputulokseen, missä mikään yksittäinen virhetilanne ei aiheuta koko järjestelmän alasajoa. Tämä sama periaate kannattaa ylläpitää myös tietoturvasuunnittelussa. Asian voi käsitellä siten, että ei anna olettaa vain yhden tietoturvaan liittyvän asian hoitavan koko ohjelmiston tietoturvaa. Pikemminkin tulisi jakaa vastuita eri resursseille ja

palveluille varmistaen täten, että tiedon eheys ja luottamuksellisuus säilyy myös tietoturvaloukkauksen sattuessa. (Sommerville 2006, 733.)
Tätä usean resurssin tai palvelun ajatusta on esimerkiksi oletus, että jokainen tietokoneohjelmisto kaatuu jossain vaiheessa käyttöänsä. Kaatumisen aiheuttava virhe voi olla itse ohjelmistossa, sen käyttämässä ympäristössä tai siinä käyttöjärjestelmässä, jota ohjelmisto toimiakseen käyttää. Vaihtoehtoja on useita, mutta lähes varmasti se virhe on olemassa ja jossain kohtaa käyttöä se tulee esiin. Kun lähes varmana tietona oletamme eteen tulevan tilanteen, että ohjelmisto kaatuu, on tämän tapahtuessa äärimmäisen tärkeää pitää ohjelman sisältämän tiedon tietoturva samalla tasolla, kuin ohjelmiston ollessa täysin toiminnassa. (Sommerville 2006, 733.)

Konkreettisen tietoturvan näkökulmasta yritykselle isoja asioita ovat nykyaikaisen tietokoneohjelmiston käytettävyysvaatimukset, jotta se voi menestyä jatkuvasti kovenevassa markkinassa. Tietoturvalle käytettävyys merkitsee yleensä jonkun tasoista vastakohtaa. Esimerkiksi uudelleen kirjautumiset tai varmistussalasanat ovat kaikki pois ohjelmiston yksinkertaisesta käytöstä, vaikeuttaen käyttäjän näkökulmasta työskentelyä. Jos ohjelmistosta tehdään tietoturvan nimissä liian vaikeita käyttää, voivat käyttäjät alkaa kehittämään ohjelmiston ulkopuolisia oikoreittejä ja lopputuloksena ohjelmistosta saadaan vielä tietoturvattomampi. (Sommerville 2006, 733.)

Perinteisesti on aina sanottu, että tietoturvan heikoin lenkki on ohjelmistojen käyttäjät. Käyttäjätunnukset ja salasanat saattavat olla kirjoitettuna paperille, niitä voidaan lainata tutulle, vaikka nimenomaisesti niin ei saisi tehdä ja käytetään samaa salasanaa kaikkialla riskeeraten kaikki järjestelmät joihin käyttäjätunnukset löytyvät. Käyttäjän manipulointi on myös vaikein uhka torjua, koska se on niin paljon ihmisistä riippuvainen. Parhaimpia keinoja tietoturvasuunnittelun kannalta on käyttää erilaisia valvontaohjelmistoja, sekä varmistaa että käyttöön tulevat autentikointimetodit eivät mahdollista tunnusten jakamista yleiseen käyttöön. (Sommerville 2006, 734.)

Päällekkäisyys tietoturvasuunnittelussa tarkoittaa, että samaa tietoa tai ohjelmiston osaa toistetaan ohjelmistossa useassa eri paikassa ja näin

varmistetaan, että yhden osan tietoturvan pettäessä tieto on edelleen olemassa ja mahdollisesti loppuohjelmiston käyttö on tietoturvaongelmasta huolimatta mahdollista. Monimuotoisuudella puolestaan tarkoitetaan, että ohjelmistokokonaisuuden eri osat voivat toimia erilaisilla alustoilla, sekä käyttää erilaisia teknologioita. Näin pyritään varmistamaan, että yhden teknologian tai alustan vaarantuminen ei kaada koko kokonaisuutta ja mahdollisesti tiedon eheys ja saatavuus pystytään tässäkin tapauksessa varmistamaan. (Sommerville 2006, 734.)

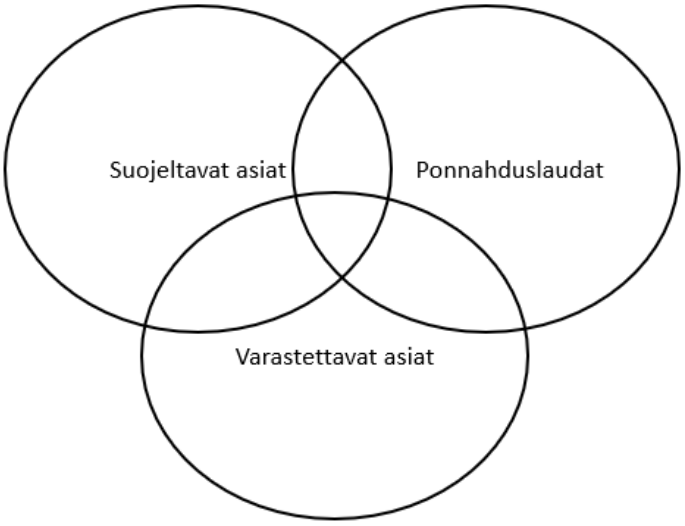
Yleisimmät niin sanotut Internetin yli tehdyt tietoturvahyökkäykset liittyvät jollain tavalla ohjelmistoon syötettäviin epätavallisiin syötteisiin, joilla pyritään saamaan ohjelmisto toimimaan tietoturvattomalla tavalla. Tällaisia voivat olla liian nopeasti toistetut syötteet, joilla saadaan palvelut kaatumaan tai mahdollisesti oikean syötteen tapainen, hyökkäykseen tarkoitettu ohjelmistokoodi, jonka ohjelmisto hyväksyy. Näiden välttämiseksi kaikki ohjelmiston käyttämät syötteet tulisi aina hyväksyä erilaisten varmistusten kautta. Tällaisia varmistuksia voivat olla esimerkiksi kenttien merkkimäärät ja erilaiset muotovaatimukset. (Sommerville 2006, 735.)

Tärkeitä asioita ohjelmistoa suunniteltaessa on myös huomioida alusta alkaen, että kaikkea ohjelmiston sisältämää tietoa ei kannata antaa käyttöön kaikille ohjelmiston käyttäjille tai ohjelmiston osille. On viisaampaa antaa aina käyttöön vain se tieto, mitä siinä osassa ohjelmistoa tarvitaan. Tämän avulla yhden käyttäjän vaarantuneet tunnukset tai yhden ohjelmiston tuottaman palvelun murtaminen, ei johda koko ohjelmiston ja sen sisältämän tiedon vaarantumiseen. (Sommerville 2006, 735.)

Kaikki ohjelmistot on tehty käyttöä varten ja jokainen ohjelmisto vaatii aina käyttöympäristön toimiakseen luotettavasti. Tästä syystä ohjelmiston suunnittelun pitäisi aina alkaa suunnittelusta käyttöönottoa varten. Mitä monimutkaisempi ja vaiherikkaampi käyttöönotto on, sitä enemmän sen aikana on mahdollisuus tehdä konfigurointivirhe, jonka seurauksena saattaa olla ympäristön ylläpitäjille näkymätön tietoturvariski. (Sommerville 2006, 735.)

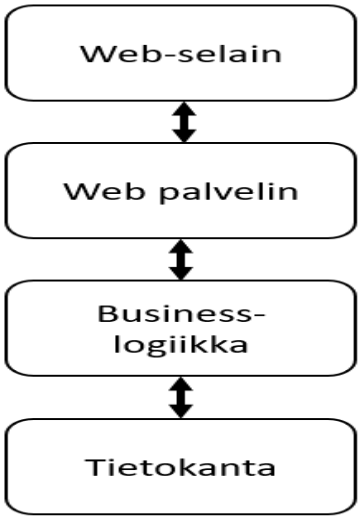
Noudattamalla kaikkia yllä olevia yhdeksää tietoturvallisen ohjelmistosuunnittelun kohtaa, voidaan pienentää huomattavasti tietoturvariskiä ohjelmiston kohdalla. Silti riski on aina olemassa, joten ehkäpä tärkein tietoturvaominaisuus on varmistaa, että riskin realisoituessa on aina oltava tie palauttaa ohjelmisto käyttöön eheällä ja luottamuksellisella tiedolla. (Sommerville 2006, 735.)

Kaiken tietoturvasuunnittelun pohjana on uhkamallinnus, jossa pyritään saamaan ensin käsitys yrityksen ja kehitettävän tietokoneohjelmiston varallisuuseristä. Kuviossa kolme on kuvattuna erilaisia kategorioita, jotka myös voivat olla päällekkäin yrityksen ajattelumaailmassa. Näistä kategorioista varastettavat asiat sisältävät lähtökohtaisesti sen tiedon, jotka ohjelmistossa on ja mikä antaa perustan hyökkääjille yrittää ohjelmistoon tunkeutua. Suojeltavat asiat puolestaan eivät välttämättä ole kaikki oikeaa tietoa, vaan voivat pitää sisällään myös esimerkiksi yrityksen maineen tai tietoturvaluokituksen. Ponnahduslaudat puolestaan pitävät sisällään itsessään yritykselle tai hyökkääjille arvottomia asioita, jotka kuitenkin aiheuttavat riskin varastettaville ja suojeltaville asioille. Ponnahduslautoja voivat olla esimerkiksi ohjelmistopalvelussa käytettävät kolmannen osapuolen ohjelmistot tai ympäristöt. (Shostack 2014, 42.)



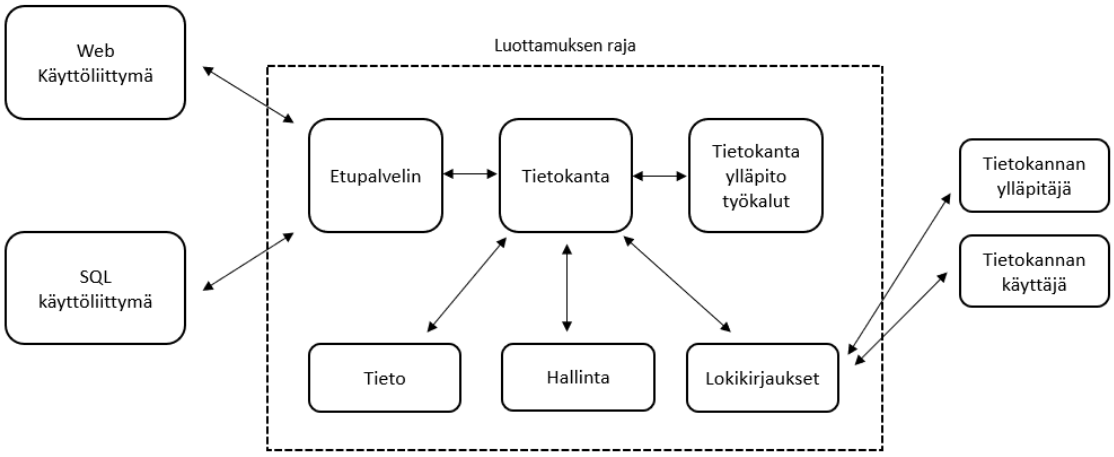
KUVIO 3. Varallisuuserät uhkamallinnuksessa

Uhkamallinnus on konkreettinen tapa lähestyä mahdollisia uhkia ja varautua niihin jo ohjelmistokehitysprosessin aikana. Näitä malleja on luotuna useita ja ne ovat lähes poikkeuksetta yritysten ensin itselleen luomia ja sitten eteenpäin jakamia. Yksinkertaisimmillaan uhkamallinnus voi olla taululle piirretty yksinkertainen arkkitehtuurikuva, jonka avulla pohditaan, mikä voi mennä vikaan. Kuviossa neljä olevassa arkkitehtuurikuvassa voidaan pohtia mitä tapahtuu, jos Web-selaimelta tulevat tiedot eivät olekaan aitoja, vaan hyökkääjän manipuloimia. Tai mitäpä jos hyökkääjällä onkin pääsy tietokantaan muokkaamaan tietoja, joita sitten varsinainen käsittelylogiikka käyttää? (Shostack 2014, 13.)



KUVIO 4. Yksinkertainen arkkitehtuurikuva

Arkkitehtuurikuvissa ja uhkamallinnuksessa on tärkeää ymmärtää, että työn tavoitteena on jatkuva tarkentaminen ja parantaminen siihen asti, kunnes ollaan halutussa tarkkuudessa. Se mikä alkaa yksinkertaisesta arkkitehtuuriluonnoksesta, tarkentuu ja parantuu sitä mukaa, mitä pidemmälle tietoturvasuunnittelu ja uhkamallinnus etenee. Kuviossa viisi on tuotu esille tarkemmalla tasolla oleva kuvaus ohjelmiston käyttöympäristöstä ja arkkitehtuurista. Tämän tarkemman tietoturvasuunnittelun ansiosta voidaan todeta, että minkä tahansa ei toivotun tahon pääsy luottamuksen rajan sisälle, antaa erittäin suuret oikeudet toimimiselle tiedon kanssa. Selkeä uhka tiedon turvallisuudelle ja eheydelle konkretisoituu. (Shostack 2014, 41.)



KUVIO 5. Yksityiskohtaisempi arkkitehtuurikuva

Uhkamallinnuksen lisäksi ohjelmistoteollisuus on luonut useita erilaisia viitekehyksiä mallintamaan tietoturvauhkia ja järjestäytynyt erilaisiin yhteisöihin, joissa tietoturvauhkia pohditaan ja niihin varautumisesta keskustellaan. Näistä viitekehyksistä ja malleista tunnetuimpia käsitellään tästä kappaleesta eteenpäin.

STRIDE on alun perin kahden henkilön, Loren Kohnfelderin ja Praerit Gargin 1990-luvulla kehittämä viitekehys, jonka nimi on akronyymi, kehyksen huomioon ottamista mahdollisista riskeistä. Nimen STRIDE muodostuminen käännöksineen on kuvattuna taulukossa neljä. (Shostack 2014, 61.)

TAULUKKO 4. STRIDE

Spoofing	Käyttäjän teeskentely
Tampering	Tietojen muuttaminen
Repudiation	Toiminnan kieltäminen
Information disclosure	Tiedon jakaminen oikeudettomalle
Denial of service	Resurssien kuluttaminen hyökkäyksellä
Elevation of privilege	Autorisoimattoman mahdollisuus käsitellä tietoa

STRIDE olettaa, että sitä käytettäessä malli ymmärretään ja tunnetaan, jolloin sen käyttämät taulukkomallit, termit ja ajatukset eivät aiheuta kehitysprosessiin suuria aikatauluviiveitä. STRIDE taulukkomalli on kuvattuna taulukossa viisi. Taulukon sarakkeessa ”Uhka” tarkoitetaan tietoturvauhkaa, josta myös

viitekehyksen nimitys on rakentunut. ”Vaarannettu ominaisuus” sarake tarkoittaa sitä ohjelmiston ominaisuutta, joka halutaan pitää kunnossa ja jota viereinen tietoturvauhka vaarantaa. ”Uhkan kuvaus” sarakkeeseen tarkennetaan, mikä tai mitkä tietoturvauhkat tähän kategoriaan kuuluvat. ”Tyypilliset uhrit” sarake puolestaan pitää sisällään ne osat ohjelmistoa tai sen käyttöä, jotka kyseisestä tietoturvauhkasta vaarantuvat. Viimeisenä sarakkeena on ”Esimerkit”, joihin lisätään kyseisestä tietoturvauhkasta kehitettyjä esimerkkitapauksia. Uhkan kuvaukset, tyypilliset uhrit ja esimerkit ovat niin ohjelmistokohtaisia, että niihin en ole sisällyttänyt taulukko on esimerkkiä. (Shostack 2014, 61.)

TAULUKKO 5. STRIDE taulukko

Uhka	Vaarannettu ominaisuus	Uhkan kuvaus	Tyypilliset uhrit	Esimerkit
Teeskentely	Autentikaatio			
Muokkaaminen	Tiedon eheys			
Kieltäminen	Lokioituminen			
Jakaminen	Luotettavuus			
Kuluttaminen	Käytettävyys			
Autorisoimattomuus	Autorisaatio			

Jokainen STRIDE mallin tietoturvauhka jaetaan omaan pienempään taulukkoon, johon kerätään esimerkkejä kyseisestä uhkasta. Taulukossa kuusi on esimerkkejä teeskentely uhkasta muodostettavista tietoturvauhkista eli erilaisia tapoja teeskennellä ohjelmistolle tai sen käyttäjille. Siinä missä esimerkeistä henkilön teeskenteleminen on enemmän käyttäjiin kohdistuvaa, ovat loput esimerkit teeskentelyä itse ohjelmistolle. Jos tämän kaltaisia tapauksia ei oteta tarpeeksi tarkasti huomioon, voi ohjelmisto luovuttaa lähes kaiken tietonsa hyökkäävälle taholle. (Shostack 2014, 65.)

TAULUKKO 6. Teeskentely esimerkit

Uhka esimerkki	Mitä hyökkääjä tekee	Muistiinpanot
Teeskennellä henkilöä	Muuttaa sähköpostin lähettäjän nimen	Mahdollisuus saada haltuun käyttäjän kirjautumistiedot
Teeskennellä palvelinta	Muuttaa käytettävää IP-osoitetta	
Teeskennellä prosessia	Luo tiedoston ennen oikeaa prosessia	

Muokkaamisen uhkalla tarkoitetaan, että jotain tietoa muokataan ohjelmiston tai käyttäjän sitä huomaamatta. Tämä tyypillisesti tehdään levyasemalla, verkossa tai muistissa. Levyasemalla voidaan muokata konfiguraatiotiedostoja tai ohjelman käyttämää tietokantaa. Verkossa puolestaan voidaan lisätä, muokata tai poistaa siirrettäviä paketteja. Taulukossa seitsemän on esimerkein kuvattu muokkaus uhan aiheuttamia tietoturvauhkia. (Shostack 2014, 65.)

TAULUKKO 7. Muokkaamisen esimerkit

Uhka esimerkki	Mitä hyökkääjä tekee	Muistiinpanot
Muokata tiedostoa	Muokata tiedostoa, jonka käyttöön luotetaan	Tarvitsee pääsyn tiedostopalvelimelle
Muokata muistia	Muokata ohjelmiston koodia muistissa	Erittäin vakavat seuraukset
Muokata verkkoa	Ohjata tietovirta omille palvelimille	

Kieltämisen uhka on käytännössä väite tietokoneohjelmiston käyttäjältä, että hän ei ole tehnyt tai ole syyllinen tehtyyn tietoturvarikkeeseen. Toisin kuin muissa uhkissa, kieltäminen voi olla myös totuuden mukaista, jolloin käyttäjä oikeasti ei ole tehnyt kyseistä asiaa ainakaan tiedostaen. Nykypäivänä järjestelmät ovat niin laajoja ja monimutkaisia, että käyttäjä ei aina tunnista ja muista kaikkea, mitä hän on järjestelmässä tehnyt. Kieltämisen uhkan tärkein torjunta on lokien kerääminen ja niiden analyysiin käytettävien työkalujen ylläpitäminen. Näiden lokien sotkeminen ja tuhoaminen on myös yksi

epärehellisten tahojen piiloutumistapa. Taulukossa kahdeksan on esimerkein kuvattu erilaisia kieltämisen uhkan aiheuttamia tietoturvauhkia. (Shostack 2014, 66.)

TAULUKKO 8. Kieltämisen esimerkit

Uhka esimerkki	Mitä hyökkääjä tekee	Muistiinpanot
Toiminnon kieltäminen	Väittää, että ei ole käyttänyt toimintoa	Ehkä hän ei oikeasti ole?
Lokien sotkeminen	Täyttää lokitiedoston valheellisella tiedolla	
Kieltää vastaanottaneensa	Väittää, että ei ole vastaanottanut lähetettyä tiedostoa	Onko joku varastanut matkalla?

Tiedon jakamisen uhka tarkoittaa tilannetta, missä joku käyttäjä tai muu henkilö saa kerättyä tietoa, jota hänellä ei ole oikeutta nähdä tai saada. Normaalisti nämä jakaantuvat kolmeen eri kategoriaan, joista on esimerkit taulukossa yhdeksän (Shostack 2014, 67). Tiedon jakaminen prosessista tarkoittaa käytännössä sitä, että ohjelmiston käyttämät prosessit vuotavat tietoa eteenpäin ja näin mahdollistavat tiedon luottamuksellisuuden rikkoutumisen. Tiedon jakaminen tietovarastosta tarkoittaa yksinkertaisimmillaan huonosti suunniteltuja salauksia ja käyttöoikeuksia tai vaihtoehtoisesti suojaamatta jätettyjä muistitikkuja. Tiedon jakaminen tietovirrasta puolestaan tarkoittaa tiedon hankkimista joko Internetissä liikkuvista tietovirroista tai mahdollisesti yhden ympäristön eri jaetuista resursseista. (Shostack 2014, 70.)

TAULUKKO 9. Tiedon jakamisen esimerkit

Uhka esimerkki	Mitä hyökkääjä tekee	Muistiinpanot
Tiedon jakaminen vastoin prosessia	Kerää tietoa virheilmoituksesta	
Tiedon jakaminen vastoin tietovarastoja	Käyttää hyväksi heikkoa tietokannan pääsynhallintaa	
Tiedon jakaminen vastoin tietovirtaa	Lukee tietoa verkosta	

Kuluttamisen uhka on STRIDE mallissa ainoa uhka, joka itsessään on jo tietoturvauhka eli niin sanottu Denial-of-Service hyökkäys. Tämän mallisella hyökkäyksellä pyritään kuormittamaan käyttöympäristö niin pahasti, että ohjelmiston käyttämisestä ei tule mitään ja näin mahdollisesti paljastamaan tietoturva-aukkoja ympäristössä. DoS hyökkäykset voidaan jakaa kahteen eri kategoriaan. Ensimmäisessä kategoriassa hyökkäykset ovat voimassa vain sen ajan, kun itse hyökkäys on käynnissä. Toisessa kategoriassa hyökkäyksen päättäminen vaatii järjestelmän uudelleen käynnistämisen tai vastaavan järjestelmän käyttöä haittaavan toiminnan. Taulukossa kymmenen on esimerkein kuvattu erilaisia DoS hyökkäyksiä. (Shostack 2014, 70.)

TAULUKKO 10. Kuluttamisen esimerkit

Uhka esimerkki	Mitä hyökkääjä tekee	Muistiinpanot
DoS prosessia vastaan	Käyttää kaiken muistin	
DoS tietovarastoa vastaan	Täyttää tietovaraston tiedostoilla	
DoS tietovirtaa vastaan	Käyttää verkkoresurssit	

Autorisoimattomuuden uhkalla tarkoitetaan sitä, että jonkun annetaan tehdä sellaista, mihin hänellä ei ohjelmistossa pitäisi olla oikeutta. Autorisoimattomuuden voi jälleen jakaa kahteen eri kategoriaan. Ensinnäkin on mahdollista toimia käyttäen hyväksi korruptoitunutta prosessia. Tässä kategoriassa hyökkääjä saa jonkun heikkouden kautta itselleen oikeudet toimia tasolla, jossa hänellä ei pitäisi olla oikeuksia. Tämän tason kautta hyökkääjä voi

korottaa itselleen vielä laajemmat oikeudet, joilla hän saa vielä enemmän tietoa hallintaansa. Toisena kategoriana on autorisoinnin epäonnistuminen eli mahdollinen virhe koodissa, jonka vuoksi autorisointia ei esimerkiksi tarkastetakaan joka kerta ja hyökkääjä saa kyseisen tietyn oikeuden hallintaansa. Ohjelman omien oikeuksien lisäksi on mahdollista, että hyökkääjä saa hallintaansa jonkun kolmannen osapuolen ohjelmiston ja näin saa kyseisen ohjelmiston käyttämät oikeudet käyttöönsä. Autorisoimattomuuden uhkan sisältämiä tietoturvauhka esimerkkejä on listattuna taulukossa 11. (Shostack 2014, 71.)

TAULUKKO 11. Autorisoimattomuuden esimerkit

Uhka esimerkki	Mitä hyökkääjä tekee	Muistiinpanot
Autorisoimattomuus korruptoituneella prosessilla	Lähettää syötteitä, joita koodi ei osaa käsitellä	
Autorisoimattomuus autorisoinnin koodivirheen seurauksena		
Autorisoimattomuus tiedon muokkauksella	Muokkaa tietoja levyllä niin, että ohjelmisto käyttää tiedostoja johonkin muuhun kuin alun perin oli tarkoitus	

OWASP tulee sanoista Open Web Application Security Project. OWASP on voittoa tavoittelematon projekti, jonka tavoite on parantaa sovellusten tietoturvaa jatkuvasti. OWASP on vapaan lähdekoodin yhteisö, joka kehittää kaiken toimintaansa liittyvän heihin kuuluvan yhteisön pohjalta. Järjestö julkaisee vapaan lähdekoodin tietoturvaan liittyviä ohjelmistoja, sekä järjestää tietoturvaan liittyviä koulutuksia ympäri maailmaa. (OWASP Foundation 2020)

Ehkä tärkein ja käytetyin tieto järjestöltä on OWASP top ten, joka listaa vuosittain kymmenen ajankohtaisinta tietoturvauhkaa. Lista on uhkamallintajalle mielenkiintoinen kokonaisuus, koska se on todella hyvän mittainen ja listalle

kerätyt uhkat ovat todennäköisyyden ja vahingollisuuden osalta hyvin tasapainotettuja. Kaikista uhkista löytyy hyvin pohjatietoa ja yksityiskohtaista ohjeistusta sekä tietoa, jonka pohjalta tietoturva arkkitehdit voivat päätellä onko kehitettävä ohjelmisto altis hyökkäykselle ja kuinka hyökkäys mahdollisesti pystytään estämään. (Shosack 2014, 94.) Taulukossa 12 on listattuna OWASP top ten vuodelta 2020. (OWASP Foundation 2020)

TAULUKKO 12. OWASP top ten

Uhka	Lisätietoja
Injektio	Esimerkiksi, SQL, NoSQL, OS ja LDAP injektio
Autentikointivirhe	Sovelluksen autentikointi ja sessionhallinta toimivat virheellisesti ja aiheuttavat vaarantumisen
Sensitiivisen tiedon vuoto	Sovellukset ja rajapinnat eivät suojele tarpeeksi niissä liikutettavaa sensitiivistä tietoa
XML:n ulkopuoliset entiteetit	Vanhat tai huonosti konfiguroidut XML käsittelijät aiheuttavat riskin
Käyttöoikeushallintavirhe	Käyttöoikeushallinnan huono suunnittelu aiheuttaa tietoturvariskin
Tietoturvakonfiguraatiovirhe	Huonosti konfiguroitu ympäristö tai päivittämättömät käyttöjärjestelmät aiheuttavat tietoturvariskin
Web-sivuille ristiin kirjoitettu XSS	Tietoturvariski ilmenee, kun toimivat XSS hyväksynnät uusilta web-sivuilta puuttuvat
Tietoturvaton sarjoituksen poisto	Tietoturvaton sarjoituksen poisto johtaa usein etäkoodin ajamiseen ja aiheuttaa tietoturvariskin
Tietoturvattomien komponenttien käyttö	Tietoturvariskin sisältävää komponenttia voidaan hyväksikäyttää koko sovelluksen vaarantamiseksi
Riittämätön valvonta ja lokitus	Riittämätön ohjelmallinen valvonta ja lokitus aiheuttavat ongelmia realisoituneen tietoturvariskin tutkimisessa

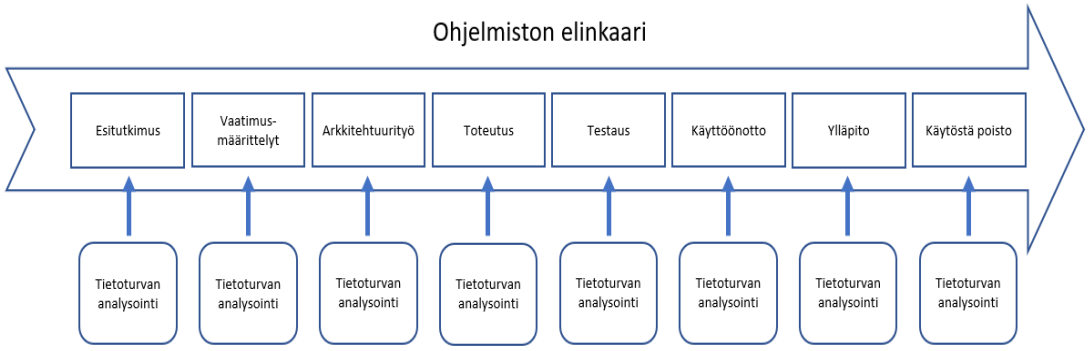
CIS tulee sanoista Center for Internet Security. CIS on myös voittoa tavoittelematon järjestö, jonka tarkoituksena on tuottaa esikuva-analyysi tietoturvallisista ohjelmistoympäristökonfiguroinneista. CIS on varsinkin ympäristöjen asentamisen ja konfiguroinnin ammattilaisten järjestö, josta voi hakea valmiiksi toteutettuja esikuva-analyysejä omaan käyttöön. (Center for Internet Security 2020)

Suomessa varsinkin julkisella sektorilla Valtiovarainministeriö on taho, joka huolehtii tietoteknisten ja siihen liittyvien toimintojen kehityksestä ja valvonnasta. Vahti puolestaan on valtiovarainministeriön asettama johtoryhmä, jonka tavoitteena on ylläpitää ja kehittää yhteiskunnan tietoturvallisuuden parantamiseen keskittyvää VAHTI-ohjeistusta. (Valtiovarainministeriö n.d.) Ohjelmistokehityksessä toimivan yksityisen sektorin yrityksen tärkeimpänä osa-alueena voidaan nähdä VAHTI-ohjeistukset, jotka kaikki on laadittu eri Valtiovarainministeriön alaisten johtoryhmien toimeenpanemana ja valvomana. Ohjeistukset ovat julkisesti kaikkien organisaatioiden käytettävissä. (Digi- ja väestötietovirasto 2020) Ohjeistus itsessään on konkreettinen vastaus Suomen valtionhallinnolta Tietoturva- ja kyberuhkien aiheuttamaan yhteiskunnalliseen riskiin. (Digi- ja väestötietovirasto 2020) Ohjeistukset sisältävät tällä hetkellä 21 erilaista ohjekokonaisuutta. Tämän työn osalta käsitellään vain ohjetta 1/2013 Sovelluskehityksen tietoturvaohje.

VAHTI 1/2013 Sovelluskehityksen tietoturvaohjeen tavoitteena on mahdollistaa tietoturvallisuusasetuksen mukainen tietoturvan perustaso valtionhallinnon organisaatioissa käytössä olevissa sovelluksissa, koska tietoturvan taso sovelluksissa on yksi kyberturvallisuuden edellytyksistä. Ohjeistus on tehty tavoitteenaan toimia ohjelmistokehitysprosessin tietoturvaohjeena ja varmistaa, että tarvittavat tietoturvavaatimukset ovat ohjelmistokehityksessä käyttöönotettuna. (Digi- ja väestötietojärjestelmä 2020)

Ohjeistus listaa erittäin tärkeinä tietoturvassa huomioonotettavina asioina esimerkiksi tietoturvallisuuden huomioonottamisen ohjelmistokehityksen kaikissa vaiheissa, ohjelmistokehittäjien ja varsinkin arkkitehtuurista vastaavien tietoturvaosaamisen tason, sekä niin sanottujen kolmannen osapuolen ohjelmistojen ja komponenttien tietoturvatason varmistamisen. Näiden lisäksi

tulisi organisaatiossa varmistaa, että tietoturva todellakin on osana prosessia sen jokaisessa vaiheessa erillisinä tarkistuspisteinä ja huolehtia sovelluksen riskianalyysin valmistumisesta ja ylläpidosta koko ohjelmistokehityksen ajan. (Vahti 1/2013 2013, 17.) Kuviossa kuusi on tuotu esille, kuinka tietoturvan tarkastuspiste kuuluu ohjelmiston elinkaaren jokaiseen vaiheeseen.



KUVIO 6. Tietoturva ohjelmiston elinkaaren aikana

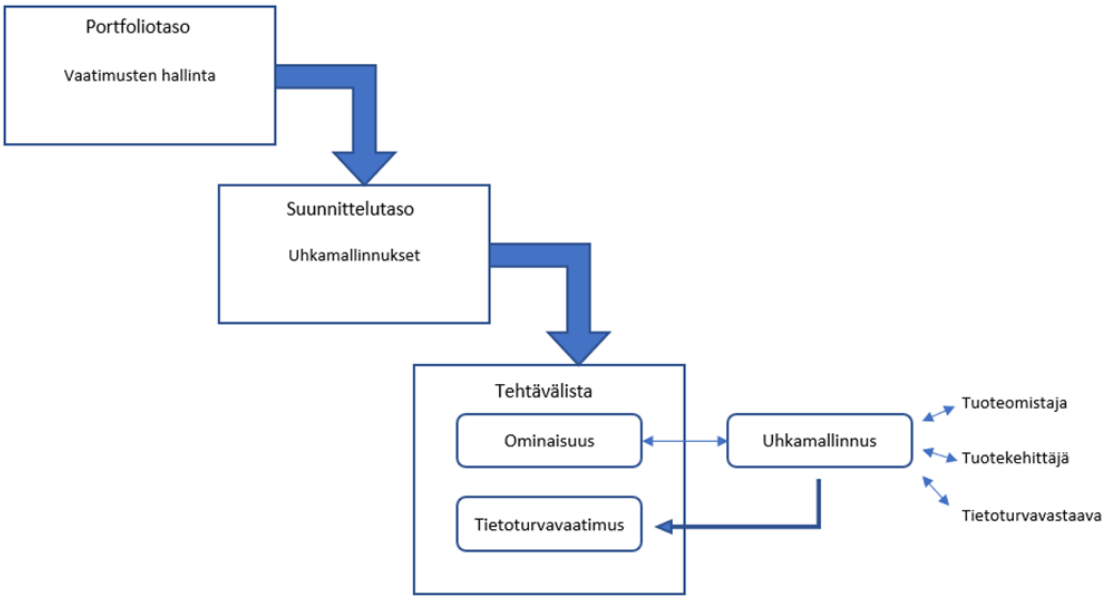
VAHTI 1/2013 listaa myös tietynlaisen minimivaatimuksen sille dokumentaatiolle, mitä ohjelmistosta tarvitaan, jos tietoturva halutaan säilyttää läpi ohjelmiston elinkaaren. Dokumentit tulee olla myös ajan tasalla ja säilöttynä niin, että tarvittavilla henkilöillä on niihin pääsy (Vahti 1/2013 2013, 25). Taulukossa 13 on listattuna Vahti 1/2013 ohjeistuksessa suositeltu dokumentaatio.

TAULUKKO 13. Vahti 1/2013 suositeltu dokumentaatio

1	Määrittely sovelluksen tietosisällöstä ja tarkoituksesta
2	Tietoturvatason määrittely
3	Tietoturvariskien kartoitus
4	Toiminnallinen määrittely
5	Tietoturva-arkkitehtuurin mukainen vaatimusmäärittely
6	Komponenttiriippuvuuksien dokumentti
7	Noudatettavat lait, standardit ja normit
8	Turvakuvaus
9	Elinkaaren hallintasuunnitelma
10	Kolmannen osapuolen ohjelmistojen tietoturvavalintakriteerit
11	Tietoturvan testisuunnitelma
12	Tietoturvatestien raportit
13	Asennusdokumentaatio
14	Käyttöönottosuunnitelma
15	Ylläpitosuunnitelma
16	Jatkuvuus- ja toipumissuunnitelma

Kokonaisuudessaan Vahti 1/2013 antaa hyvän suomalaisen viitekehyksen sovelluskehitykseen varsinkin julkiselle sektorille, mutta myös monelta osin sovellettavaksi yksityisen sektorin käyttöön. Tämän lisäksi viranomaistaholta ohjeistusta on saatavissa esimerkiksi Digi- ja väestötietoviraston julkaisemassa Turvallisen sovelluskehityksen käsikirjassa. Käsikirja sisältää käytännön ohjeistusta ohjelmistokehitystä tekeville organisaatioille siitä, kuinka tietoturvallista ohjelmistokehitystä voidaan konkreettisesti toteuttaa. Käsikirja luo tietoturvalle ja ohjelmistokehitykselle hyvin tiukat ehdot onnistuakseen. Ensinnäkin käsikirja olettaa, että ohjelmistokehitys on nykyaikaisen ketterän kehitysmallin mukaista, toisin sanoen ymmärretään, että perinteisellä vesiputousmallilla ei lopulta pystytä toimimaan tietoturvallisesti. Ketterän kehitysmallin edellytys johtaa myös vaatimukseen nykyaikaisen ketterän kehitysmallin mukaisesta autonomisesta tiimistä, joka tuoteomistaja mukaan lukien, vastaa tietoturvasta. Lopuksi käsikirjassa huomioidaan, että käyttöön tarvitaan moderni jatkuva toimitusputki, jonka avulla toimitusketjussa kaikki mikä voidaan automatisoida, on automatisoitu. (Digi- ja väestötietovirasto 2020, 5.)

Ketterien menetelmien automatisoidun toimitusketjun lisäksi oletetaan, että käytössä on moderni vaatimustenhallintajärjestelmä, missä pystytään ylläpitämään esimerkiksi erilaisia tietoturvavaatimuksia. Tämän vaatimushallintajärjestelmän täytyy myös mahdollistaa erilaiset hierarkiat ja linkitykset, joiden avulla tietoturva ja tietosuoja pidetään näkyvänä läpi tuotekehitysprosessin. (Digi- ja väestötietovirasto 2020, 5.) Tätä vaatimusten hallinnan ja eri tasojen linkitystä on pyritty selkeyttämään Kuviossa seitsemän.



KUVIO 7. Tietoturva tuotekehitys prosessissa

4.4 Tietoturva tuotekehityksessä

Nykypäivänä tietoturva nähdään tärkeänä osana ohjelmistokehitystä ja se näkyy myös tähän työhön tehdyissä haastatteluissa. Haastatelluista lähes kaikki olivat sitä mieltä, että tietoturvan pitää olla prioriteeteistä korkeimpia, kun uutta ohjelmistoa rakennetaan, vanhoihin ohjelmistoihin päivitetään ominaisuuksia tai vaihdetaan teknologioita.

Organisaatiossa ollaan kuitenkin vasta tämän työn alkuvaiheessa ja tietoturvan konkreettinen parantaminen on lähinnä ajatuksia siitä, miten asiat voisivat olla. Erilaisia ohjausryhmiä kootaan kasaan ja suunnitellaan töitä ja niiden aloittamista. Tietoturvaa, kuten muitakin ydintekemisen reunalla olevia asioita,

leimaa samanlainen ”konserni hoitaa”- ajattelu, kuin isoissa organisaatioissa ja yrityksissä useasti vallitsee. Ei oteta tiettyä ratkaisumallia itse käyttöön, koska ajatellaan, että jostain tulee kuitenkin taho, joka asian määrittelee ja hoitaa. Haastatteluissa tässä asiassa oli selkeästi eroa siinä, kuinka lähellä konsernin hallinnon tasoa vastaaja oli. Erilaiset johtotahot ja konsernin henkilökunta olivat lähtökohtaisesti sitä mieltä, että tietoturva täytyy nitoa kiinni osaksi kaikkien roolien työtä. Tuotekehitysyksikössä taas nähtiin enemmän tarpeelliseksi kohdistettu rooli ja osoitettu henkilö, joka hoitaa tietoturvaan liittyvät asiat tai ainakin huolehtii asian edistämisestä kaikkialla.

Totuus tässäkin asiassa löytyy mielestäni puolesta välistä. Tietty rooli, jonka vastuulla on pitää huolta eri viitekehyksistä ja niiden nivomisesta osaksi prosessia varmasti tarvitaan, mutta lopulta se on jokainen työntekijä itse, joka omalla työllään ratkaisee, miten tietoturvasta huolehtiminen lopulta onnistuu. Tutkimuksessa kävi selvästi ilmi se, minkä huomattava osa haastatelluista totesikin, että tietoturvan kohottamiseen yrityksessä vaaditaan kaikkien prosessissa jollain tavalla mukana olevien tietoturva osaamisen nostoa. Tämä oli haastateltujen lähes konsensusmainen tahtotila.

Omien vastuiden hämäryys tietoturva-asioissa oli myös erittäin yleinen havainto koskien koko organisaatiota. Ymmärrettävästi johto, jonka vastuu on laajempi, ymmärsi olevansa asioista vastuussa, mutta organisaatiohierarkian alemmilla tasoilla, ei esimerkiksi arkkitehdeille ollut selkeää, miten tietoturva-asioita tulisi ottaa huomioon. Useat kohdat olivat lähes täysin henkilöiden omien harrastusten tai yleisen mielenkiinnon perusteella hoidettuja. Tästä voidaan tehdä päätelmä, että tietoturvaongelmien vähyys yrityksessä voi hyvinkin johtua erittäin ammattitaitoisesta henkilökunnasta.

Yksi huomionarvoinen asia, joka nousi esiin niin tutkimuksessa, kuin muutenkin yritystä havainnoidessa, oli ymmärrys EU:n tietosuoja-asetuksen vaatimasta tarkkuudesta. Asetuksen kovat sanktiot oli selkeästi huomioitu ja ymmärretty ja siihen tarvittavia ohjaavia dokumentteja, sopimustekstejä ja erilaisia läpikäyntejä oli hyvinkin paljon. Tämä oli monella kohtaa konkreettinen esimerkki siitä, että jos tietosuojan ja tietoturvan halutaan olevan kohdallaan, niin se on mahdollista toteuttaa hyvin.

Huolestuttavaa organisaation kannalta oli tuotekehityksen prosessista ja vaatimusten määrittelyistä vastaavien henkilöiden ohut ymmärrys erilaisista tietoturvan käsitteistä tai esikuva-analyyseista. Löytyi esimerkiksi yksi tuoteomistaja, jolla ei ollut käsitystä sen enempää Vahti kuin OWASP ohjeistuksista. Vaikka voidaan aina kehittyvän organisaation osalta miettiä, mitä kaikkea eri roolien tulisikaan tietää, niin tietynlainen oman alan tuntemus ja siihen kohdistuva kiinnostus tarvittaisiin vähintään silloin, kun kyseessä on koko prosessin yksi tärkeimpiä avainrooleja.

Viimeinen tutkimuksesta esille nostettava ajatus on ehdottomasti tietynlaisen pohjatason hakeminen eli mikä on tietosuojan ja tietoturvan osoitettu taso, minkä alle ei missään nimessä haluta mennä. Tähän yksi vaihtoehto voisi olla OWASP yhdistyksen asettamat standardit, sekä Microsoftin omat tietoturva foorumit. Tämän työn tutkimushavainnoinnin jälkeen, lähtisin alkuun suosittelemaan turvallisen sovelluskehityksen käsikirjan sekä Vahti luokituksen haltuun ottamista. Tämä yhdistelmä muodostaisi selkeästi kaikista tehokkaimman tietoturva viitekehyksen, kun toimitaan Suomessa ja suomeksi.

5 TIETOTURVA JA KETTERÄT MENETELMÄT

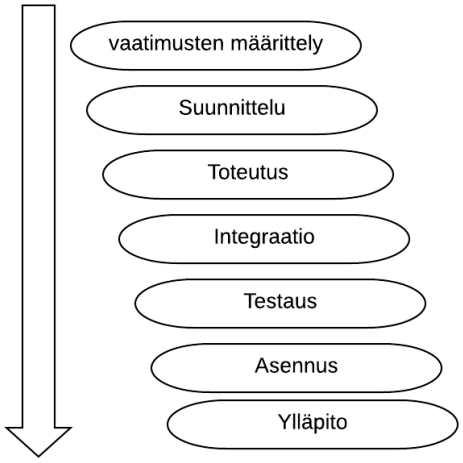
5.1 Ketterät menetelmät

Ketterällä kehityksellä tarkoitetaan viitekehystä, jonka avulla ohjelmistokehityksessä pystytään tuottamaan liiketoiminta-arvoa mahdollisimman aikaisessa vaiheessa. Menetelmä tukee inkrementaalista ominaisuuksien ja prosessin kehittymistä, sekä mahdollisuutta muokata ominaisuuksien vaatimuksia kehityksen ollessa käynnissä. Tämän kaiken avulla organisaatiolla on kyky tuottaa tuotteita ja palveluita, jotka ovat mahdollisimman lähellä asiakkaan tarvetta. (Layton & Ostermiller 2017, 7.)

Kehittämistehtävän toimeksiantoyrityksen toimiessa ketteristä menetelmistä pääsääntöisesti Scrum-viitekehyksessä, tässä työssä käsitellään, kappaletta 3.1.1 ketterän kehityksen historia osuutta lukuun ottamatta, ketteriä menetelmiä Scrum-viitekehystä vasten ja muihin menetelmiin viitataan vain tarvittaessa.

5.1.1 Ketterän kehityksen historia

Erilaisten modernien projektimallien historian voidaan katsoa alkaneen toisen maailmansodan jälkeisestä modernisoituvasta maailmasta, jossa rakennettiin nopeasti uudistuvien yhteiskuntien lisäksi ensimmäisiä kaupalliseen käyttöön toteutettuja tietokoneita ja näiden tarvitsemia ohjelmistoja. Näissä projektimalleissa hahmoteltujen periaatteiden pohjalta luotiin niin kutsuttu vesiputousmalli, jonka nimikin kuvaa, kuinka tämän kaltaisessa projektissa liikutaan ylhäältä alaspäin, siirtyen seuraavaan vaiheeseen vasta edellisen vaiheen valmistumisen jälkeen. Tätä mallia kuvattaessa prosessikaaviolla, se näyttää hieman vesiputoukselta. (Layton & Ostermiller 2017, 8.) Vesiputousmallin prosessikaavio on kuvattuna Kuviossa kahdeksan.



KUVIO 8. Vesiputousmallin prosessikaavio

Vesiputousmallissa prosessin tärkein osa on heti ensimmäisenä löytyvä ”vaatimusten määrittely”, jonka aikana valetaan pohjaa koko myöhemmälle toiminnalle. Tavoitteena on saada selville vastaus vähintään kahteen kysymykseen, ”mitä halutaan” ja ”miksi halutaan?” (Kähönen 2015, 5). Vesiputousmallissa tämä vaihe on niin tärkeä, että sitä jatketaan siihen asti, että yhteisymmärrys käyttäjien vaatimuksista ominaisuuksille on saavutettu. Yhteisymmärryksen tueksi tuotetaan tarkan tason kirjallinen dokumentti, jonka tavoite on avata vaatimukset tuotekehitykselle mahdollisimman selkeästi. (Kähönen 2016, 6)

Vesiputousmallin haasteet liittyvät suurimmalta osin juuri tähän vaatimusten määrittely vaiheeseen. Koska vaihe on ensimmäisenä, siihen pohjataan kaikki työmäärät, aikataulut ja hinnat, joita projektissa tai tuotekehityksessä käytetään. Ominaisuuden tilaajilta eli lopulta sovelluksen käyttäjiltä, on erittäin paljon vaadittu, että he ymmärtävät jo tässä vaiheessa kaikki ne vaatimukset, joita he tarvitsevat lopputuotteen ominaisuuksissa. Tämän lisäksi vesiputousmallissa myös tuotteen kehitys suunnitellaan pääsääntöisesti niin, että toimiva versio sovelluksesta on olemassa vasta kehityssyklin aivan loppupuolella. (Pressman 2010, 41.)

Vesiputousmallia on pyritty parantamaan tekemällä siitä iteratiivisempaa, esimerkiksi palaamalla useammin määrittelyihin tai lisäämällä vuorovaikutusta loppukäyttäjien kanssa. Johtuen mallin sisäänkirjoitetusta lineaarisesta mallista,

tämä yleensä aiheuttaa sekaannuksia koko kehityssyklin toiminnassa ja tätä kautta hidastaa tuotteen kehittämistä, nostaa kustannuksia ja tekee lopputuloksesta valjun kompromissin. (Pressman 2010, 40.)

Vesiputousmallissa jatkuvasti toistuvien samojen ongelmien takia, alkoi ohjelmistoteollisuuteen ilmestymään 1990-luvun alkupuolelta lähtien erilaisia kevyitä kehitysmalleja. Nämä mallit painottivat omilla tavoillaan iteratiivisuutta, sekä pyrkivät lähestymään koko tuotekehitysprosessia uudella, ihmisiä painottavalla mallilla, pyrkimyksenä korjaamaan vesiputousmallin ohjelmistoteollisuudelle aiheuttamia ongelmia. (Haikala & Mikkonen 2011, 42.)

Näistä kevyistä kehitysmaleista tunnetuin ja käytännössä kaiken ketterän kehityksen pohjana käytetty malli on XP eli äärimmäinen ohjelmointi. XP kehitettiin 1990-luvun alussa auttamaan ohjelmistokehittäjiä vesiputousmallissa ilmenneissä ongelmissa. Käytännössä mallissa lueteltiin erilaisia kehittäjien näkökulmasta hyödyllisiä arvoja ja periaatteita, joilla saataisiin ohjelmistokehitystä toimivammaksi, sekä asiakasta eli sovelluksen loppukäyttäjää enemmän palvelevaksi. Äärimmäisen ohjelmoinnin prosessissa toteutetut arvot johtivat lopulta Agile Manifeston eli ketterän julistuksen luomiseen. (Measey, Wolf, Berridge, Gray, Levy, Oliver, Roberts, Short & Wilmhurst 2015, 125.)

Vuonna 2001, seitsemäntoista näitä kevyitä iteratiivisia malleja suosivaa ohjelmistokehittäjää kerääntyi hiihtomajaan Utahissa, Yhdysvalloissa viettämään aikaa, sekä keskustelemaan ohjelmistokehityksen ongelmista (Haikala & Mikkonen 2011, 43). Tuohon asti valtavirran suosimat prosessimallit ohjelmistokehityksessä olivat erilaisia vesiputousmalliin pohjaavia prosesseja. Karrikoiden sanoen, maailma oli hyväksynyt, että ohjelmistokehitys on kallista ja se mitä toteutetaan, harvemmin on sitä, mitä halutaan ja tarvitaan. (Haikala & Mikkonen 2011, 43.)

Tuo joukko ohjelmistokehittäjiä edusti erilaisia kevyitä projektimalleja, joista ei ollut muodostunut vielä valtavirtaa ohjelmistoteollisuuteen. Näiden henkilöiden tapaamisen seurauksena julkaistiin niin sanotut ketterän kehityksen periaatteet. Nämä 12 ketterän julistuksen periaatetta ovat:

1) Tärkein tavoitteemme on tyydyttää asiakas toimittamalla tämän tarpeet täyttäviä versioita ohjelmistosta aikaisessa vaiheessa ja säännöllisesti.
2) Otamme vastaan muuttuvat vaatimukset myös kehityksen myöhäisessä vaiheessa. Ketterät menetelmät hyödyntävät muutosta asiakkaan kilpailukyvyn edistämiseksi.
3) Toimitamme versioita toimivasta ohjelmistosta säännöllisesti, parin viikon tai kuukauden välein ja suosimme lyhyempää aikaväliä.
4) Liiketoiminnan edustajien ja ohjelmistokehittäjien tulee työskennellä yhdessä päivittäin koko projektin ajan.
5) Rakennamme projektit motivoituneiden yksilöiden ympärille. Annamme heille puitteet ja tuen, jonka he tarvitsevat ja luotamme siihen, että he saavat työn tehtyä.
6) Tehokkain ja toimivin tapa tiedon välittämiseksi kehitystiimille ja tiimin jäsenten kesken on kasvokkain käytävä keskustelu.
7) Toimiva ohjelmisto on edistymisen ensisijainen mittari.
8) Ketterät menetelmät kannustavat kestävään toimintatapaan. Hankkeen omistajien, kehittäjien ja ohjelmiston käyttäjien tulisi pystyä ylläpitämään työtahtinsa hamaan tulevaisuuteen.
9) Teknisen laadun ja ohjelmiston hyvän rakenteen jatkuva huomiointi edesauttaa ketteryyttä.
10) Yksinkertaisuus - tekemättä jätettävän työn maksimointi - on oleellista.
11) Parhaat arkkitehtuurit, vaatimukset ja suunnitelmat syntyvät itseorganisoituvissa tiimeissä.
12) Tiimi tarkastelee säännöllisesti, kuinka parantaa tehokkuuttaan, ja mukauttaa toimintaansa sen mukaisesti.
(Lasse Koskela n.d.)

Näiden periaatteiden kirjaamisen lisäksi, perustettiin ketteryyttä ohjelmistoteollisuuteen ajava Agile Alliance järjestö, sekä kirjoitettiin niin sanottu Agile Manifesto, eli ketterä julistus, joka toimi perustetun järjestön peruskirjana (Haikala & Mikkonen 2011, 44).

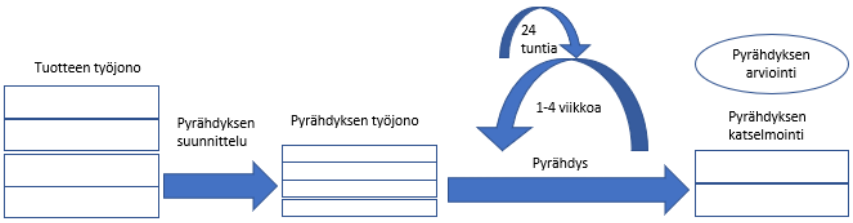
Sanatarkasti Agile Manifesto menee näin: ”Löydämme parempia tapoja tehdä ohjelmistokehitystä, kun teemme sitä itse ja autamme muita siinä. Kokemuksemme perusteella arvostamme:

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

Jälkimmäisilläkin asioilla on arvoa, mutta arvostamme ensiksi mainittuja enemmän.” (Lasse Koskela n.d.)

Ketterän julistuksen julkaisusta voidaan katsoa lopullisesti alkaneen ketterien menetelmien aika ohjelmistokehityksessä. Suurin osa ennen julistusta kehitetyistä malleista on hiipunut ajan myötä pois ja tilalle on tullut ketterien mallien periaatteita ja julistusta yhdistelevä Scrum. Yhä merkittävämpää asemaa ottavat myös seuraavat menetelmät: Kanban, erilaiset Scrumban-mallit, jotka yhdistelevät Scrum- ja Kanban-malleja, varsinkin isoille organisaatioille soveltuva SAFe, sekä kaiken taustalle tehokkuutta ohjaava Lean ohjelmistokehitysmalli. (Measey ym. 2015, 4.)

Scrum, joka on ketteristä malleista kaikista laajimmalle levinnyt, on alun perin Japanista ja mainittu ensimmäistä kertaa artikkelissa jo vuonna 1986 (Haikala & Mikkonen. 2011. 46). Scrumin aikakauden katsotaan silti laajemmin alkaneen vuodesta 1995, kun Yhdysvaltalaiset Jeff Sutherland ja Ken Schwaber kehittivät siitä ohjelmistotuotantoon sopivan mallin (Measey ym. 2015, 132). Scrum on ketteristä menetelmistä kaikista jäykin ja ainoa, jossa on tarkkaan piirretty prosessinkulku. Tuo prosessinkulku on kuvattuna Kuviossa yhdeksän.

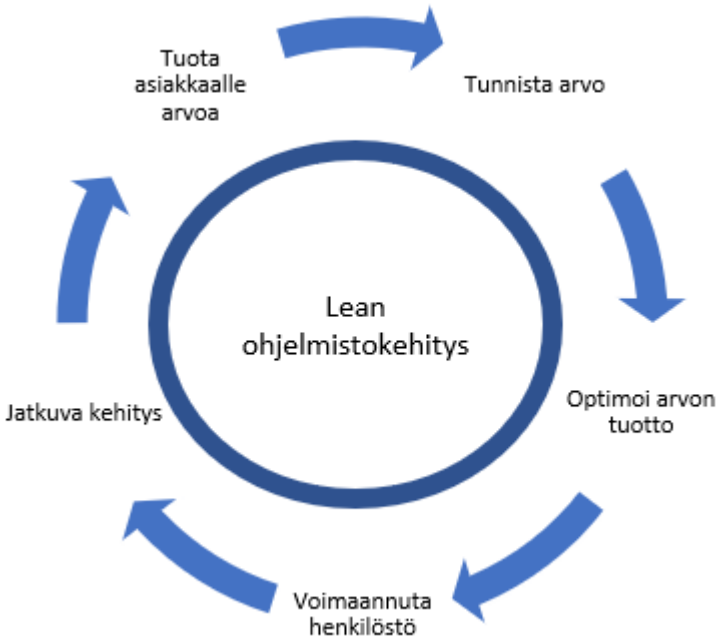


KUVIO 9. Scrum prosessi

Lean ohjelmistokehitys tuli laajempaan tietoisuuteen vuonna 2003 Mary ja Tom Poppendieckien tekemän työn pohjalta. Mallin pohjana toimi Toyotan kehittämä teollisen tuotannon tehokkuuteen keskittyvä Lean malli, joista Mary ja Tom muokkasivat periaatteita osaksi ohjelmistokehitysprosessia. Lean ohjelmistokehityksessä kehitystiimille annetaan viisi avainkysymystä, joiden avulla selvitetään niin sanottu Lean kyvykkyys. (Measey ym. 2015, 153.)

1) Onko tuotekehitystiimi kokonaisvaltainen eli sisältääkö se kaiken tarvittavan loppukäyttäjänarvon tuottamiseen?
2) Ymmärtävätkö kaikki tuotekehitystiimin jäsenet, mitä asiakkaat oikeasti arvostavat?
3) Onko tuotekehitystiimin fokus inkrementaalisen arvon tuottamisessa asiakkaalle?
4) Tuottaako tuotekehitystiimi arvoa luotettavasti ja toisteisesti?
5) Löytyykö tuotekehitystiimistä johtaja, joka välittää syvällisesti asiakkaasta ja asiakkaan ongelmista, sekä johtaja, joka välittää syvällisesti sovelluksen teknisestä yhtenäisyydestä?

Lean eroaa Scrumista siinä, että se ei ole itsessään piirretty prosessi vaan enemmän ajattelutapa olemassa olevan prosessin käyttämiseen ja tehostamiseen (Haikala & Mikkonen 2011, 54). Tätä ajattelutapaa on pyritty tuomaan esille Kuviossa kymmenen.



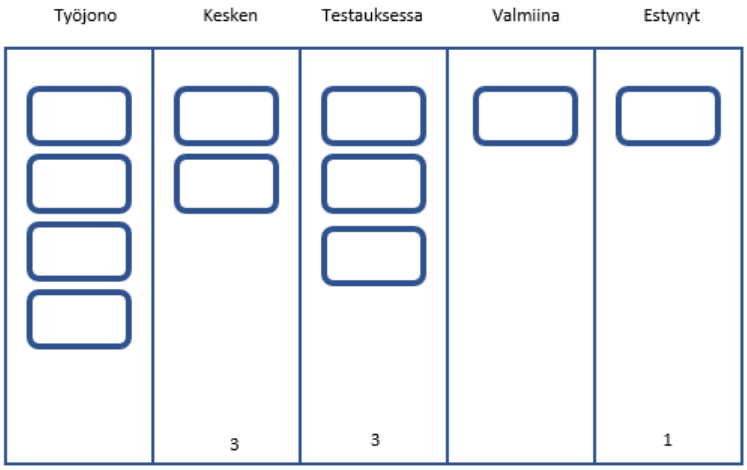
KUVIO 10. Lean ajattelu

Kanban on mallina yksinkertainen ja pohjautuu Lean ajattelun visualisoimiseen. Malli ei sisällä mitään määrättyjä rooleja tai vakioitua prosessia, jonka takia Kanban-mallia ei sinällään pidetä edes omana tuotekehitysprosessina, vaan enemmänkin vaihtoehtoisena polkuna ketterään toimintaan. Mallin pohjimmainen tarkoitus on visualisoida ja esittää tuotekehitysprosessin sen hetkinen tehokkuus ja antaa kipinä jatkuvaan tehostamiseen ja parantamiseen. (Measey ym. 2015, 148.) Mallia ohjaa kuusi perusperiaatetta, jotka ovat

1) Työnkulku pitää olla visualisoitu.
2) Kesken olevilla töillä tulee olla enimmäismäärä.
3) Käytäntöjen tulee olla täsmällisiä.
4) Läpimenoaika vaiheesta toiseen tulee mitata.
5) Palaute kanavat tulevat olla käyttöönotettuna.
6) Parannetaan yhdessä, kehitytään kokeilemalla.

Näillä periaatteilla tuotekehityssykliä viedään eteenpäin ilman mitään tarkemmin määriteltyä prosessia tai suunniteltua aikataulua. Kun tietyssä vaiheessa on tilaa, voidaan kyseiselle vaiheelle ottaa uusi työ ja kun työ on kyseisen vaiheen mukaisesti valmis, siirretään se seuraavaan vaiheeseen. Taulussa olevat vaiheet voidaan päättää tuotekehitystiimissä, kuin myös jokaisen vaiheen

enimmäistyömäärä. (Haikala & Mikkonen 2011, 55.) Kanban-taulun esimerkkimalli löytyy kuviosta 11.



KUVIO 11. Kanban taulu

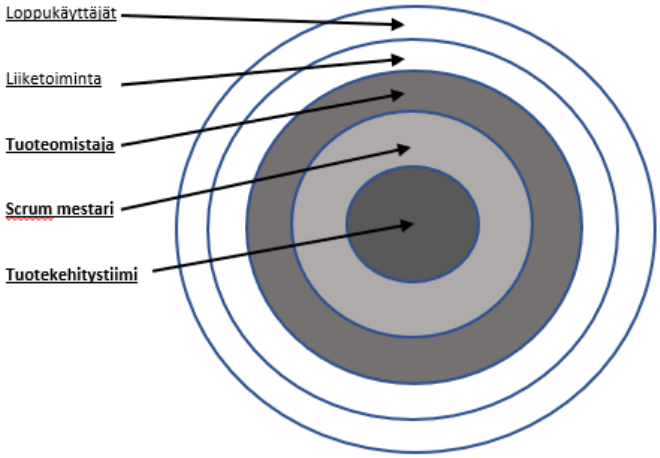
Scrumban on tapa, jolla organisaatiot yhdistelevät Scrum- ja Kanban-malleja saavuttaakseen molempien mallien parhaat puolet. Yleisin yhdistelevä malli tästä on tapa käyttää Kanban-taulua Scrum pyrähdyksen sisällä tehtävän työn visualisoimiseen. Muita mahdollisia tapoja on pyörittää eri kehitysprojekteja Scrum-mallissa, mutta itse ylätason kokonaishanketta Kanban-mallissa. Vaihtoehtoisesti voi myös eriyttää isot ja pyrähdykseen sopimattomat työt Kanban-malliin ja muutoin käyttää Scrum-mallia. (Measey ym. 2015, 152.)

Scrumbut on laajasti käytetty termi mallista, jossa puritaaninen Scrum on mukautettu organisaation käyttöön. Mahdollisia variaatioita voivat olla esimerkiksi päivittäisten kokousten muuttaminen viikoittaisiksi, pyrähdysten vakiopituudesta luopuminen, arviointipalaverin puuttuminen tai usean tuoteomistajan käyttäminen. Tämän tyyppiset toiminnat luetaan Scrum-malliin sopimattomiksi käytännöiksi. (Haikala & Mikkonen 2011, 52.)

5.1.2 Scrum tiimi

Scrum-tiimissä tarvittavat roolit on erikseen Scrum-mallin perusteissa määritelty. Toimiakseen Scrum tarvitsee kolme erilaista roolia, Tuotekehittäjän,

Scrum-mestarin ja Tuoteomistajan. Näillä kolmella eri roolilla on kokonaisvaltainen vastuu arvon tuottamisesta loppukäyttäjille, asiakkaille ja liiketoiminnalle. (Haikala & Mikkonen 2011, 48.) Scrum-tiimin sijaintia organisaatiossa on kuvattu Kuviossa 12.



KUVIO 12. Scrum tiimi organisaatiossa

Scrum-mestari on kehitystiimin palvelija, jonka tehtävänä on huolehtia, että tiimi saa toteuttaa työnsä rauhassa ja toisaalta vastaa tiimin tuloksesta tuoteomistajalle. Scrum-mestaria on myös kutsuttu projektipäälliköksi ilman mitään valtaa eli niin sanotuksi palvelijajohtajaksi. (Measey ym. 2015, 49.) Scrum-mestarin tärkeimpiin tehtäviin kuuluu kehityksen esteiden poistaminen, sekä prosessin ylläpitäminen ja valvominen. Scrum-mestarin tulee varmistaa toteutettujen töiden sääntöjenmukaisuus tuotekehitystiimin yhteisesti hyväksyttyjen periaatteiden perusteella. (Haikala & Mikkonen 2011, 49.)

Tuotekehitystiimin ulkopuolella Scrum-mestari toimii tuotekehityksen esteitä poistavana ja Scrum-prosessin jatkuvaa parantamista ajavana voimana. On siis hänen tehtävänsä varmistaa, että koko organisaatio tukee tuotekehityksen valitsemaa Scrum-prosessia ja toimii sen edellyttämällä tavalla. (Measey ym. 2015, 133.) Taulukossa 14 löytyy kaikki Scrum-mestarin tehtävät.

TAULUKKO 14. Scrum mestarin tehtävät

1	Scrum-prosessin onnistuminen
2	Scrum-käytäntöjen ja roolien perustaminen
3	Scrum-tiimin suojeleminen ulkopuoliselta häiriöltä
4	Scrum-tiimin toiminnallisuuden ja tuotannollisuuden varmistaminen
5	Yhteistyön varmistaminen eri roolien ja toimintojen välissä
6	Scrum-prosessin ylläpitäminen eri palaverien muodossa
7	Organisaation johtaminen ja valmentaminen kohti parempaa Scrum-prosessia
8	Tuoteomistajan avustaminen työjonon ylläpitämisessä

Tuoteomistaja vastaa tuotteen liiketoiminnallisesta onnistumisesta ja toimii rajapintana tuotekehityksen ja liiketoiminnan, sekä muiden sidosryhmien välissä. Tuoteomistajan tärkeimpänä tehtävänä on positiostaan käsin, kerätä kaikki saatavilla oleva tieto erilaisista sovellukseen kohdistetuista vaatimuksista ja jalostaa niistä tuotekehitystiimille prioriteettijärjestyksessä oleva työlista. (Haikala & Mikkonen 2011, 48.) On täysin tuoteomistajan vastuulla, että tuotteella on selkeä visio ja että kaikki tuotekehitystiimin työlistalla olevat artikkelit varmasti vievät kohti tuon vision saavuttamista (Measey ym. 2015, 133). Liiketoiminnalle tuoteomistaja vastaa puolestaan mahdollisten tuotekehitysbudjettien pitävyydestä, tuotteen hinnoittelun riittävyydestä, sekä esimerkiksi tuotteen dokumentaation ajantasaisuudesta (Haikala & Mikkonen 2011, 49). Taulukosta 15 löytyvät kaikki tuoteomistajan tehtävät.

TAULUKKO 15. Tuoteomistajan tehtävät

1	Töiden suhteellisen arvon määritteleminen
2	Liiketoiminnan ja tuotekehitystiimin visioiden yhteensovittaminen
3	Käytettävissä olevan budjetin määritteleminen
4	Pyrähdysten ja versioiden tavoitteiden asetanta
5	Osallistuminen pyrähdyksen ja version suunnittelupalavereihin
6	Töiden tarkentaminen ja päätösten tekeminen tiimin kanssa reaaliajassa
7	Töiden hyväksyminen
8	Pyrähdyksen ja version hyväksyminen
9	Julkaisupäivän päättäminen
10	Tuotteen ominaisuuksien päättäminen
11	Työjonon järjestäminen priorisoituun järjestykseen
12	Töiden tarkentaminen ja pyrähdykseen otettujen töiden priorisoiminen
13	Tuotteen takaisinmaksun varmistaminen
14	Varmistaa, että tuote vastaa sitä mitä käyttäjät tuotteelta haluavat

Tuotekehittäjä on termi, joka käsittää kaikki Scrum-tiimin jäsenet välittämättä siitä, mikä on heidän tarkempi osaamisalueensa. Tästä syystä kaikki ohjelmoijat, testaajat, arkkitehdit, käyttöliittymäsuunnittelijat tai muut osaamiset, joita tuotekehityksessä tarvitaan tuotteen tekemistä varten, on kerätty yhden tasavertaisen termin alle. Näin Scrum-malli pyrkii varmistamaan periaatetta kaikkien kehitykseen osallistuvien tasavertaisuudesta ja tarpeellisuudesta. Suositeltava koko Scrum-tiimille on 5-9 henkeä. (Haikala & Mikkonen 2011, 49.)

Tämä tasavertaisuuden periaate on myös Scrum-mallin suurin ero perinteiseen vesiputous, ja muihin ”käske ja valvo” -malleihin. Tuotekehitystiimi on itse vastuussa tekemänsä työn laadusta ja määrästä ja tuon vastuun myötä tuotekehitystiimillä on paljon valtaa työhönsä liittyvissä asioissa. Esimerkiksi tuotekehitystiimi itse päättää työskentelykäytännöistään, kuinka paljon työtä jokaisessa tehtävässä työssä on, kuinka paljon työtä yhden pyrähdyksen aikana tehdään, milloin työ on valmis ja mitä töitä kukakin tuotekehitystiimin jäsen tekee. (Measey ym. 2015, 135.) Tuotekehitystiimin tehtävät löytyvät taulukosta 16.

TAULUKKO 16. Tuotekehitystiimin tehtävät

1	Käydä tuoteomistajan kanssa työt läpi ja varmistaa molemminpuolinen ymmärrys
2	Määritellä muuttuvaa arkkitehtuuria ja varmistaa laadun pysyminen halutulla tasolla
3	Itseohjautuva, kaiken tarpeellisen sisältävä tiimi ilman annettuja rooleja
4	Sisältää 5 – 9 henkeä, joiden osaaminen riittää töiden valmistumiseen
5	Huolehtii palavereiden pitämisestä pääsääntöisesti kasvotusten
6	Huolehtia tehtävien luomisesta töille ja niiden toteuttamisesta
7	Valta tehdä mitä vain, jotta tavoitellut työt valmistuvat
8	Esitellä valmistuneet työt tuoteomistajalle ja sidosryhmille
9	Oikeus tehdä mitä vain asetettujen rajojen ja standardien sisällä, saavuttaakseen pyrähdyksen ja version tavoitteet

5.1.3 Scrum prosessi

Scrum-mallissa tuotekehitystä tapahtuu pyrähdyksittäin. Pyrähdyksen pituus on tuotekehitystiimin itse päättämä, mutta Scrum-malli suosittelee 2-4 viikkoa pitkiä pyrähdyksiä (Measey ym. 2015, 135). Valitun pyrähdyksen pituuden perusteella määräytyy myös, päivittäispalaveria lukuun ottamatta, pyrähdykseen kuuluvien palaverien pituudet (Scrumguides.org 2017, 9).

Pyrähdys alkaa suunnittelupalaverilla, jonka pituus riippuu valitusta pyrähdyksen pituudesta, käytännössä aikaa tulisi varata palaverin molemmille osioille yksi tunti per pyrähdyksen viikko (Scrumguides.org 2017, 9). Suunnittelupalaveri jaetaan kahteen toiminnalliseen osioon ja tällöin esimerkiksi käytettäessä kahden viikon pyrähdystä, varataan palaveriaikaa kaksi tuntia molempiin osioihin (Hundermark 2014, 23).

Osiossa yksi tuotekehitystiimi ja tuoteomistaja keskustelevat Scrum-mestarin johdolla, mikä olisi realistinen pyrähdyksen tavoite, palaveri siis pyrkii vastaamaan kysymykseen ”mitä pystymme toteuttamaan”. Tuoteomistajan priorisoima tuotteen kehitysjono toimii pohjana suunnittelupalaverin työlle. Jos tehtävien määritykset ja vaatimukset ovat tiimin asettamien standardien tasolla, niin tiimin jäsenet antavat arvion kuinka paljon tiimin kykyä tuottaa asioita työ kuluttaa ja ottavat sen pyrähdykseen mukaan. Tätä toistetaan niin monta kertaa, että tuotekehitystiimillä on omasta mielestään tarpeeksi töitä pyrähdyksen ajalle. Jos taas määrityksissä on puutteita, pyritään niitä parantamaan jo palaverin aikana. Tilanteissa, joissa puutteet vaatimuksissa ovat liian suuria, voi tiimi kieltäytyä toteuttamasta tehtävää. Vaihtoehtoisesti työlistaan voidaan lisätä tehtäviä, joiden tarkoitus on määritellä kyseistä tehtävää, sillä nämäkin tehtävät käyttävät tiimin tuotekehitysaikaa. (Hundermark 2014, 22.)

Osiossa kaksi tuotekehitystiimi pohtii tapoja ja ratkaisumalleja pyrähdykseen valitsemilleen töille, tämä palaveri siis pyrkii vastaamaan kysymykseen ”miten pystymme toteuttamaan”. Palaverin aikana on tavoite saada toteutettua tietynlainen ylätason arkkitehtuuri pyrähdyksessä toteutettaville ominaisuuksille. Scrum-mestari varmistaa, että erityisesti tuoteomistaja, mutta myös muut tarvittavat sidosryhmät ovat tavoitettavissa palaverin aikana. (Hundermark 2014, 23.)

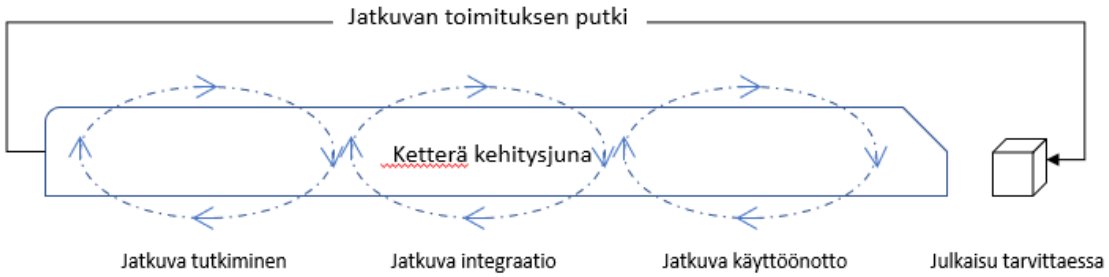
Päivittäispalaveri on riippumatta pyrähdyksen pituudesta aina 15 minuutin mittainen. Palaveriin osallistuvat tuotekehitystiimi sekä palaverin vetäjä, Scrum-mestari. Tuoteomistaja saa osallistua palaveriin, mutta vain kuuntelemaan ja vastaamaan, hänellä ei siis ole oikeutta kysyä esimerkiksi töiden edistymisestä. Palaverin runko rakentuu kolmelle kysymykselle. ”Mitä olet tehnyt edellisen päivittäispalaverin jälkeen”, ”Mitä aiot tehdä seuraavaksi”, ”Onko tiedossa tekemiselle esteitä”. Päivittäispalaveri on ketterässä mallissa erittäin tärkeä, koska se ylläpitää ajatusta aina uuden suunnitelman tekemiseksi 24 tunnin ajaksi. (Haikala & Mikkonen 2011, 50.)

Pyrähdyksen tekeminen päättyy katselmointipalaveriin, jonka kesto määräytyy valitusta pyrähdyksen pituudesta. Käytännössä varataan yksi tunti per pyrähdyksen viikko. Palaveriin osallistuvat tuotekehitystiimin, tuoteomistajan ja Scrum-mestarin lisäksi kaikki halukkaat sisäiset sidosryhmät. Palaverin tarkoituksena on antaa tuotekehitystiimin esitellä kaikille halukkaille toteutetut toiminnallisuudet ja näin varmistaa, että kaikki ymmärtävät ominaisuudet samalla tavalla, sekä kerätä tietoa ja palautetta tulevia toiminnallisuuksia varten. (Hundermark 2014, 25.)
Pyrähdys päättyy kokonaisuudessaan pyrähdyksen arviointipalaveriin, jonka kesto jälleen määräytyy valitusta pyrähdyksen pituudesta, käytännössä varataan enimmillään yksi tunti per pyrähdyksen viikko. Palaveriin osallistuvat tuotekehitystiimi, tuoteomistaja, sekä Scrum-mestari. Palaverin tarkoituksena on vastata kahteen kysymykseen: ”Mikä meni pyrähdyksessä hyvin” ja ”Mitä voimme seuraavassa pyrähdyksessä parantaa”. Vastauksia haetaan niin, että kaikki palaveriin osallistuvat vastaavat näihin kysymyksiin vuorollaan ja vastauksista koostetaan tehtävälista Scrum-mestarille ja muille asianomaisille. (Haikala & Mikkonen 2011, 51.) Arviointipalaverin tarkoitus on siis antaa tiimille runko jatkuvan kehityksen malliin, jonka tavoitteena on aina seuraavalla pyrähdyksellä olla parempi, kuin mitä oltiin edellisellä pyrähdyksellä (Hundermark 2014, 26).

5.1.4 SAFe, mitä se on?

SAFe:n erottaa muista yleisimmistä ketteristä menetelmistä sen suuntaus koko organisaatioon. Siinä missä Scrum, Kanban ja vastaavat olettavat itseohjautuvien tiimien ja muutamien pakollisten roolien riittävän kaikkeen ohjelmistojen hallintaan ja kehitykseen liittyvään työhön, SAFe ymmärtää, että organisaation ollessa tarpeeksi iso, näin ei ole. SAFe:n kehittäjä Dean Leffingwell on sanonut vapaasti suomentaen ”Elämme ohjelmistojen aikakautta, jossa jokainen yritys on ohjelmistoyritys. Ketteryys ei ole vaihtoehto tai vain tiimien ohjenuora, se on liiketoiminnan edellytys.” (Scaled Agile Inc 2020.)

SAFe:ssa koko organisaation ketteryys pohjautuu organisaation jakamiseen karkeasti kolmeen osaan, joista jokaisen täytyy omaksua ketteriä periaatteita toimintaansa, jotta koko organisaation ketteryys tulee toimivaksi. (Measey, ym. 2015. 160.) Kolme olemassa olevaa tasoa ovat, portfoliotaso, ohjelmataso, sekä itseohjautuvien tiimien taso. Tämän jakamisen seurauksena myös kehitettävien sovellusten uudet ominaisuudet on käsiteltävä seuraavilla kolmella tasolla: eeppisellä tasolla, ominaisuuden tasolla ja käyttäjätarina tasolla. Näiden kolmen tason yhteistoiminta mahdollistaa SAFe-mallissa niin sanotun ketterän kehitysjunan jatkuvan liikkeen eteenpäin. (Scaled Agile Inc 2020.) Kuviossa 13 on kuvattuna ketterän junan liike.

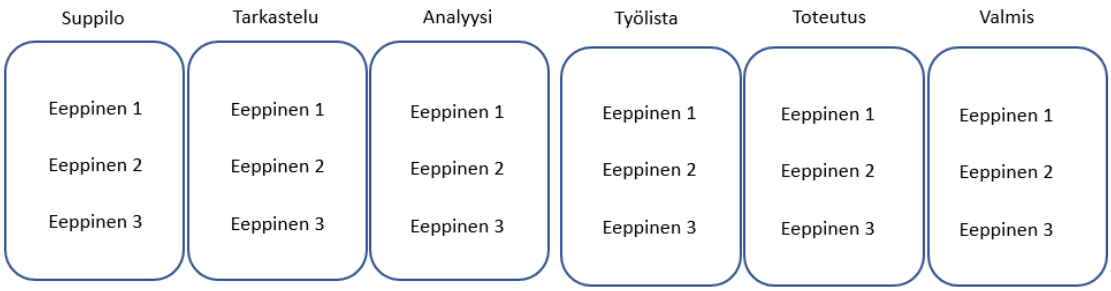


KUVIO 13. Ketterä kehitysjuna

Portfolio tasolla SAFe käsittää kahden tärkeän kokonaisuuden hallinnan. Kehitettäviin sovelluksiin tuotettavien ominaisuuksien eeppisen tason suunnittelun, sekä niiden rahoittamisen. Portfolio tasolla ohjelmistot ja niihin liitettävät palvelut muodostavat seurattavia eeppisiä tasoja. Eeppisen tason ominaisuuskokonaisuuksia hallitaan Kanban-mallisesti portfolio työlistassa. Kanban-prosessi voi sisältää useita eri vaiheita, jotka organisaatio voi itse päättää, mutta SAFe suosittelee kuusiportaista taulua. Tärkeimpänä tehtävänä

on varmistaa, että eri vaiheiden jälkeen portfoliotyölista sisältää eeppisen tason ominaisuuskokonaisuudet sillä tasolla, että ne ovat otettavissa mukaan ohjelmatasolle inkrementteihin. Tärkeitä rooleja portfolio tasolla on eeppisten ominaisuuskokonaisuuksien omistajat, ohjelmaportfolion johto, sekä kokonaisarkkitehti. (Scaled Agile Inc 2020.)

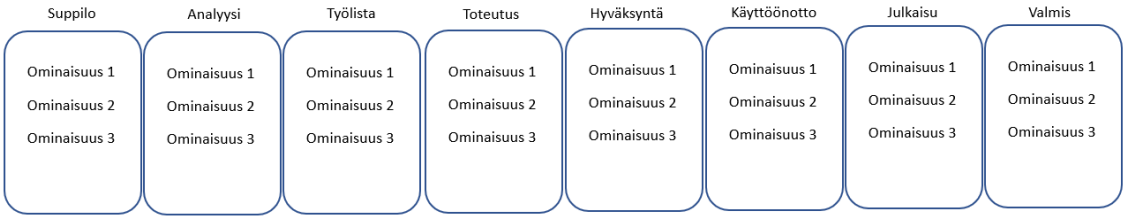
Portfolio Kanban muodostuu kuusiportaisesta taulusta, johon kaikki eeppiset tasot nostetaan ja dokumentoidaan. Ensimmäinen porras on suppilo, johon kerätään kaikki ideat ja ajatukset, joista voidaan aloittaa luomaan arvoa. Tästä eeppiset tasot siirretään tarkasteluportaalle, jossa tavoitteena on hahmotella kokonaisuuden arvo. Tämän jälkeen analyysiportaalla pyritään miettimään toteutusta jo sillä tasolla, että eeppisille tasoille tulee hinta. Analyysiportaalla toteutetaan myös niin kutsuttu ”tehdään/ei tehdä”-päätös, jonka seurauksena eeppiset tasot voidaan siirtää portfolio työlistaan. Toteutusportaassa eeppiset tasot on otettu käytännössä mukaan tuotekehitysjunaan ja viimein viimeisellä portaalla eeppiset tasot ovat valmiit ja tuottavat arvoa yritykselle. (Scaled Agile Inc 2020.) Kuviossa 14 on kuvattuna Kanban-taulu, jota käytetään portfoliotasolla.



KUVIO 14. Portfoliotason kanban

Ohjelmataso on portfolion ja tiimitason välissä oleva taso, jossa ylläpidetään ketterän julkaisujunan liikettä. Myös ohjelmatasolla kehitystä voidaan hallita Kanban-taululla, jolle SAFe suosittelee kahdeksaa porrasta. Kanban-taulun ensimmäinen porras on jälleen suppilo, missä voi olla mitä tahansa ideoita koskien ohjelman ominaisuuksia tai vaatimuksia. Tätä seuraa analyysiporras, jossa ominaisuudelle kirjoitetaan vaatimuksia, hahmotellaan työmäärää ja tarkastellaan, onko se linjassa annetun vision kanssa. Kaiken ollessa kunnossa,

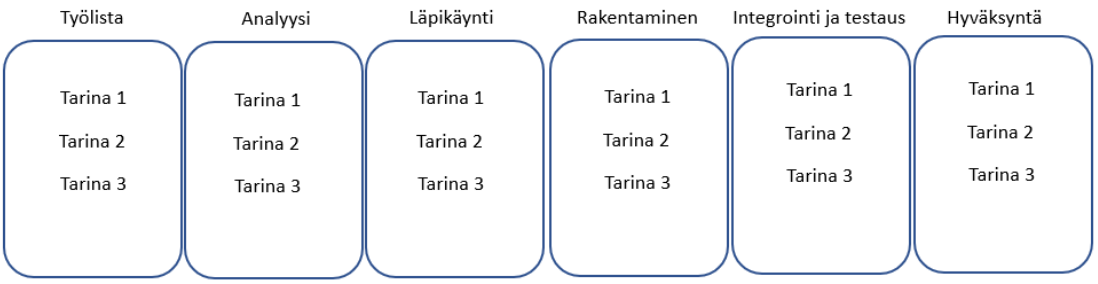
työ siirretään seuraavalle portaalle eli työlistalle. Toteutusportaalla ketterä kehitysjuna ottaa aina työlistalta ylimpänä olevat ominaisuudet, jotka sitten pilkotaan itseohjautuvien tiimien tasolle tarinoiksi ja asetellaan iteraatioihin. Ominaisuuden valmistuessa se siirretään hyväksyntäportaalle, jossa ominaisuus esitellään sidosryhmille ja hyväksytetään. Hyväksynnän jälkeen ominaisuus siirretään julkaisuportaalle, jossa huolehditaan ominaisuuden julkaisusta asiakkaiden käyttöön. Lopulta ominaisuus siirtyy valmis-portaalle, johon se dokumentoituu tulevaisuutta varten. (Scaled Agile Inc 2020.) Kuviossa 15 on kuvattuna Kanban-taulu, jota käytetään ohjelmatasolla.



KUVIO 15. Ohjelmatason Kanban

SAFe olettaa itseohjautuvien tiimien käytössä olevan jonkun ketterän ohjelmistonkehitysmenetelmän, sekä siihen tarvittavat työkalut ja konseptit. Soveltuvia malleja ovat esimerkiksi XP, Scrum, Kanban ja näiden yhdistelmät. Omana lisänään näihin malleihin SAFe tuo ymmärryksen tiimin toiminnasta ohjelmatasolla, ajatuksen kokonaisuuden jatkuvasta virrasta, sekä rajoitteet keskeneräisten töiden määrälle. (Measey ym. 2015, 160.)

Itseohjautuvien tiimien Kanban-malliksi SAFe suosittelee kuusiportaista mallia, jossa työlistaporras sisältää kaikki ne käyttäjätarinat, mitä ohjelmatasolta tulevista ominaisuuksista on ideoitu. Analyysiportaalla nuo käyttäjätarinat käydään tarkemmin läpi ja niistä muodostetaan ymmärrettäviä kokonaisuuksia, jotka sitten tutkitaan läpikäyntiportaalla. Käyttäjätarinoiden tuotantoon hyväksynnän jälkeen alkaa niiden rakentaminen, integrointi ja testaus Näillä portailla tarinat liikkuvat sen mukaan, miten niiden käytännön toteuttaminen edistyy. Lopulta valmiit käyttäjätarinat hyväksytetään sidosryhmillä hyväksyntäportaalla. (Measey ym 2015, 164.) Kuviossa 16 on kuvattuna Kanban-taulu, jota käytetään tuotekehitystiimin tasolla.



KUVIO 16. Itseohjautuvien tiimien Kanban

5.2 Tietoturva osana ketteriä menetelmiä

Ketterät menetelmät ovat ajansaatossa rakentuneet tarpeesta pystyä tekemään muutoksia valmistuvaan lopputuotteeseen sitä mukaa, kun loppukäyttäjät alkavat ymmärtää mitä he haluavat. Tämä antautuminen jatkuvalle muutokselle on se vahvuus, joka todellisessa ketterässä organisaatiossa on. Siinä missä vesiputousmallin arkkitehti antaa vastauksena pitkät toimitusajat ja kovat hinnat jokaiselle muutokselle, ketterän mallin arkkitehti löytää ratkaisun toteuttaa järkevästi ohjelma juurikin niillä ominaisuuksilla, jotka käyttäjä haluaa. Ketterät menetelmät eivät siis kannusta dokumentoimattomuuteen tai suunnittelemattomien muutoksien tekemiseen, kuten välillä virheellisesti luullaan, vaan ainoastaan jatkuvaan valmiuteen tehdä muutos. Tietoturva ketterissä menetelmissä taas tarkoittaa sitä, että missä tahansa kohtaa prosessia ilmeneekään jonkun sidosryhmän aiheuttama muutostarve, on olemassa toimivat mekanismit myös tietoturvan varmistamiseksi. Toisin sanoen ketterät menetelmät pakottavat organisaation omaksumaan koko tietoturvakulttuurin, eikä vain liimaamaan tietoturvaa osaksi prosessia.

Tutkimuksen kohteena olevan organisaation malli ketterästä kehityksestä ei ole suoraan mikään teoreettinen viitekehys. Malli on muotoutunut omaksi kokonaisuudekseen itse lanseeratusta Scrum-mallista ja konsernin projektisalkun hallinnassa suosimasta SAFe-versiosta. Tämän lopputuloksena on tietynlainen tiimien ja aliorganisaatioiden Scrumbut, joka sinänsä kyllä vastaa ketterän mallin vaatimuksiin tarpeeksi tehokkaasti toimiakseen modernin ohjelmistokehitysmallin mukaisesti.

Kaikissa eri ketterissä menetelmissä on annettuna erilaisia rooleja, joilla on vastuunsa, esimerkkinä Scrum-mallin tuoteomistaja tai SAFe-mallin ketterän junan insinööri. Vaikkakin noille rooleille voidaan sinänsä vastuuttaa myös tietoturvaan liittyviä asioita oli haastattelujen perusteella ilmeistä, että tuo vastuu halutaan erilliselle roolille, joka hoitaa tietoturvan sisällyttämisen kaikkiin prosessin eri vaiheisiin. Haastateltujen joukossa tästä roolituksesta oltiin jotakuinkin samaa mieltä, vain yksi haastatelluista oli ehdottomasti sitä mieltä, että vastuut pitäisi sisällyttää olemassa oleviin rooleihin. Vaikka jossain kohtaa voitaisiinkin miettiä tietoturvasta vastuullisen nimettyä roolitusta, niin tärkeintä on kuitenkin omaksua tietoturva osaksi kaikkea ajattelua.

Tutkimuksen kohteena olevan organisaation osalta löytyi haastatteluissa haastateltavien väliltä huomattavia eroja ketterien menetelmien ymmärryksessä. Haastateltavista muutamat tunsivat ketterät menetelmät erittäin hyvin, mutta osa ei käsittänyt lainkaan, mitä tarkoittaa koko käsite ketterät menetelmät, saati osanneet nimetä perusmalleja ollenkaan. Esimerkkinä muutama haastateltava oli sitä mieltä, että vesiputousmalli on yksi ketteristä menetelmistä, kun taas muutama henkilö tunsi, XP-, Scrum-, Kanban- ja SAFe-mallit ja jopa niihin kuuluvia roolituksia. Vaikka suurin osa haastatelluista olikin tietämyksensä puolesta näiden ääripäiden välissä, on huolestuttavaa näin perustason tietämyksen puute ohjelmistoyrityksen henkilöstössä.

Vahti 1/2013 ohjeistukseen pohjautuva Digi- ja väestötietokeskuksen tietoturvallisen sovelluskehityksen käsikirja olettaa tuotekehityksen noudattavan jotain ketterää menetelmää pystyäkseen oikeasti vastaamaan tietoturvaan. Suurimpana syynä tähän on ketterien mallien sisäänkirjoitettu tiimin autonomia, sekä tahtotila prosessin osien automatisointiin ja jatkuvaan parantamiseen. Kehitystiimin autonomialla tarkoitetaan tässä tilanteessa sitä, että tiimillä on jopa tietynlainen liiketoimintavastuu sovelluksen tietoturvasta. Erittäin tärkeänä nähdään tuoteomistajan vastuu ja valta tuotteen kehittämisessä ja eri töiden priorisoinneissa. Tuoteomistajan tietoturvapäätösten yli ei saa kävellä edes esimerkiksi ohjausryhmän päätöksellä. Prosessin osien automatisoinnilla puolestaan haetaan prosessin tukea automaation rakentamiselle, tässä jälleen ketterän ajattelun ja menetelmän sisäänkirjoitetut lainalaisuudet antavat tukea.

Luonnollisesti myös inhimillisten virheiden vähentyessä, niistä johtuvat tietoturvapoikkeamat vähentyvät.

Tietoturvallisen sovelluskehityksen käsikirjan ketterän kehitysmallin vaatimukset tunnistaen voidaan todeta, että organisaation ketterien mallien ymmärrys ei ole niin korkealla, että tehokas tietoturvallinen prosessimalli olisi mahdollista. Mahdollisuus on luoda rooli, joka huolehtii tietoturvasta prosessin jokaisessa kohdassa, mutta jos halutaan esimerkiksi jollain uhkamallinnusskenaariolla toimiva autonominen tiimi ratkomaan kyseisen tiimin rakentamien sovellusten tietoturvaa, on ensimmäinen askel rakentaa organisaatiolle ymmärrys, mitä tarkoittaa olla ketterä.

6 TIETOTURVA JA VAATIMUSTENKÄSITTELY

6.1 Mitä vaatimukset ovat?

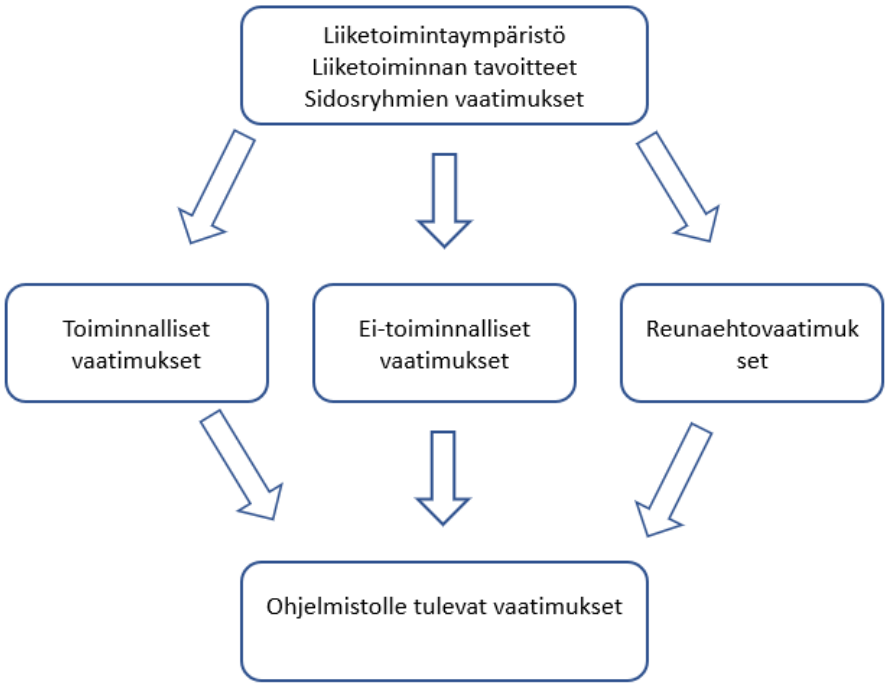
Pressman Roger kirjoittaa kirjassaan Software engineering vapaasti suomentaen seuraavan ”Se on pahin painajaisesi. Asiakas kävelee huoneeseen, istuu alas, katsoo sinua suoraan silmiin ja sanoo: ”Minä tiedän, että sinä ajattelet ymmärtäväsi mitä minä sanoin, mutta se mitä sinä et ymmärrä on se, että se mitä minä sanoin ei ollut se mitä minä tarkoitin”.” Vaatimusten hallintaa pidetään kiistatta ohjelmistokehityksen vaikeimpana osa-alueena. Se on lähtökohta ohjelmiston kehittämiselle ja se on lähestulkoon aina syyllinen kehityksen epäonnistumiselle. (2010, 119.)

Ohjelmistokehityksen voisi hahmotella olevan ohjelmiston tarvetta määrittelevä kuvausprosessi, joka aloitetaan ylätason tarvekuvauksesta ja jota sen jälkeen tarkennetaan kierros kierrokselta, kunnes saavutetaan ohjelmakoodintaso, jossa ohjelmoija kuvaa vaadittujen ominaisuuksien toimintaa ohjelmointikielen avulla. (Haikala & Mikkonen 2011, 62.) Vaatimukset siis ovat toteutettavalle ohjelmistolle asetettuja kuvauksia toiminnallisuuksista, joita ohjelmistoon halutaan, sekä laatuominaisuuksista, joilla varmistetaan ohjelmiston käytettävyyttä ja kustannustehokkuutta. (Pressman 2010, 120).

Ohjelmistolle määritellyt erilaisten ominaisuuksien vaatimukset tulevat aina liiketoiminnan tarpeista. Ensinnäkin siitä minkälaisessa liiketoimintaympäristössä toimintaa harjoitetaan, sekä mitä lakeja ja viranomaismääräyksiä kyseisellä toimialalla tulee ottaa huomioon ohjelmaa kehitettäessä. Toisekseen tarvitaan käsitys siitä, mitä asiakkaat ja muut huomioonotettavat sidosryhmät kehitettävältä ohjelmistolta odottavat ja lopuksi tietenkin päätös siitä, mitä liiketoiminta itse haluaa markkinoille tarjota. (Haikala & Mikkonen 2011, 62.)

Vaatimukset voidaan jakaa kolmeen eri kategoriaan, toiminnallisiin-, ei-toiminnallisiin- ja reunaehtovaatimuksiin. Toiminnallisilla vaatimuksilla tarkoitetaan ominaisuuksia, joita ohjelmassa tulee olla. Näitä vaatimuksia voidaan kuvata käyttäjätarinoiden, käyttötapausten ja erilaisten

käyttöliittymämallien kautta. Esimerkkeinä tällaisista vaatimuksista ovat mahdollisuus nähdä oman pankin mobiilisovelluksessa oman pankkitilinsä saldo tai puhelinlaskun summa. Näiden vaatimusten tavoitteena on siis varmistaa, että ohjelmiston toiminta on käyttäjien toivomaa. Ei-toiminnallisilla vaatimuksilla tarkoitetaan loppukäyttäjälle näkymättömiä, mutta käyttökokemuksessa merkitseviä vaatimuksia. Näitä voivat olla esimerkiksi ohjelmiston käyttönopeus, skaalautuvuus, ylläpidettävyys ja tietoturva. Reunaehtovaatimuksilla taas tarkoitetaan ohjelmiston käyttöympäristöön liittyviä vaatimuksia. Tällaisia ovat esimerkiksi vaatimukset pystyä ylläpitämään ohjelmistoa Linux-käyttöjärjestelmällä toimivilla palvelimilla ja pystyä käyttämään ohjelmistoa Windows käyttöjärjestelmällä toimivalla tietokoneella, sekä Android käyttöjärjestelmällä toimivalla älypuhelimella. Lopulta ohjelmistolle syntyy joukko vaatimuksia, joiden perustana ovat toiminnalliset vaatimukset, joita sitten ei-toiminnalliset- ja reunaehtovaatimukset muokkaavat tarpeensa mukaan. (Haikala & Mikkonen 2011, 61.) Kuviossa 17 on kuvattuna vaatimusten muodostuminen ohjelmistokehityksessä.



KUVIO 17. Vaatimusten muodostuminen

Haikala ja Mikkonen toteavat Ohjelmistokehityksen perusteet kirjassaan, että ohjelmistokehitys on käytännössä vain eritasoisten määrittelyjen luomista. Määrittely alkaa ylätason suunnitelmista ja lopulta päättyy ohjelmointikieleen,

joka määrittelee käyttöjärjestelmälle, kuinka ohjelman tulee toimia. Tästä syystä ensimmäinen versio määrittelystä, jonka perusteella esimerkiksi sopimukset ohjelmistokehityksestä rakennetaan, voi olla hyvinkin vapaamuotoista. (Haikala & Mikkonen 2011, 70.)

Vaatimusmäärittelyissä käytettäviä malleja ja dokumentaatioita on kehitetty ohjelmistokehityksen historiassa useita. Se mikä palvelee ohjelmistokehityksen kuvaamisessa parhaiten, määräytyy kokonaisuuden perusteella. Mobiilipankkisovelluksen ja ydinvoimalan autonomisen hälytysjärjestelmän kehittämisessä on hyvin vähän yhteneväisyyksiä ohjelmistokehitysprosessissa ja sen validoinnissa. Kaikkien vaatimusmäärittelyiden kuvausmallien tavoite on kuitenkin saada vaatimukset kuvattua niin hyvin, että ne pysyvät ymmärrettävinä koko tuotekehityksen prosessin ajan ja näin varmistava lopputuotteen olevan mahdollisimman lähellä haluttua ohjelmistoa. (Lopez 2011, 81.)

Yleisin ja helppotajuisin tapa toteuttaa vaatimusmäärittely on luonnollisen kielen avulla. Tämä tarkoittaa yksinkertaisesti sitä, että vaatimukset kuvataan vaatimusmäärittelydokumenttiin kirjoittaen luonnollisella kielellä siitä tarvittavat tiedot ylös (Pressman 2010, 120). Luonnollisen kielen ehdoton etu on loppukäyttäjän ymmärrys sen käytöstä, luonnollisen kielen avulla voidaan siis varmistaa, että vaatimusten esittäjät saavat esitettyä toiveensa tavalla, joka on heille tuttua. Ongelmana luonnollisessa kielessä on vaatimusten tulkinnan mahdollisuus, mitä tarkemmalle tasolle vaatimusmäärittelyssä mennään. (Lopez 2011, 80.)

Luonnollinen kieli riittää kehitysprosessin tiettyyn pisteeseen asti. Siinä vaiheessa, kun aletaan törmäämään enemmän ja enemmän tulkintaongelmiin, tulee ruveta toteuttamaan määrittelyä tarkemmin, käyttäen erilaisia tarkempia ja määrämuotoisempia mallinnuksia. Näistä yleisin ja vuosien saatossa standardiksi muodostunut on yleinen mallinnus kieli eli UML. (Pressman 2010, 54.)

UML pyrkii mallintamaan graafisesti ja mahdollisimman yksityiskohtaisella tasolla sen, mitä ohjelmisto pitää sisällään ja mitä sitä käytettäessä tapahtuu. Mallinnus sisältää kolmetoista erilaista kaaviomallia, jotka jaetaan rakenne- ja

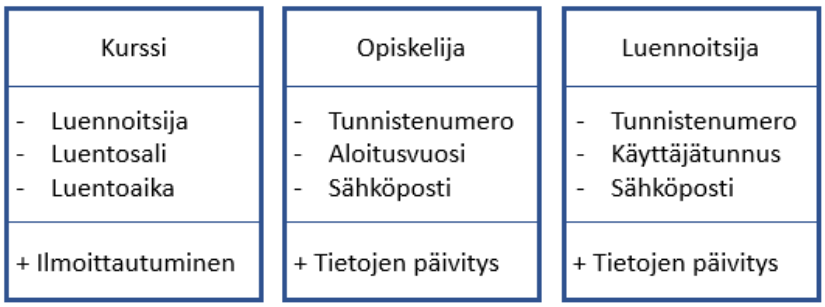
käyttäytymiskaavioihin. Kaavioista seitsemän kuuluvat rakennekaavioihin ja kuusi käyttäytymiskaavioihin. Tämän lisäksi käyttäytymiskaaviosta voidaan eriyttää vielä vuorovaikutuskaaviot. (Haikala & Mikkonen 2011, 74.) Taulukossa 17 on listattuna UML-kaaviomallit.

TAULUKKO 17. UML-kaaviomallit

Rakennekaaviot	
1	Luokkakaavio
2	Oliokaavio
3	Pakkauskaavio
4	Sijoittelukaavio
5	Koostekaavio
6	Komponenttikaavio
Käyttäytymiskaaviot	
7	Aktiviteettikaavio
8	Käyttötapauskaavio
9	Tilakaavio
Vuorovaikutuskaaviot	
10	Ajoituskaavio
11	Kokoava vuorovaikutuskaavio
12	Kommunikointikaavio
13	Sekvenssikaavio

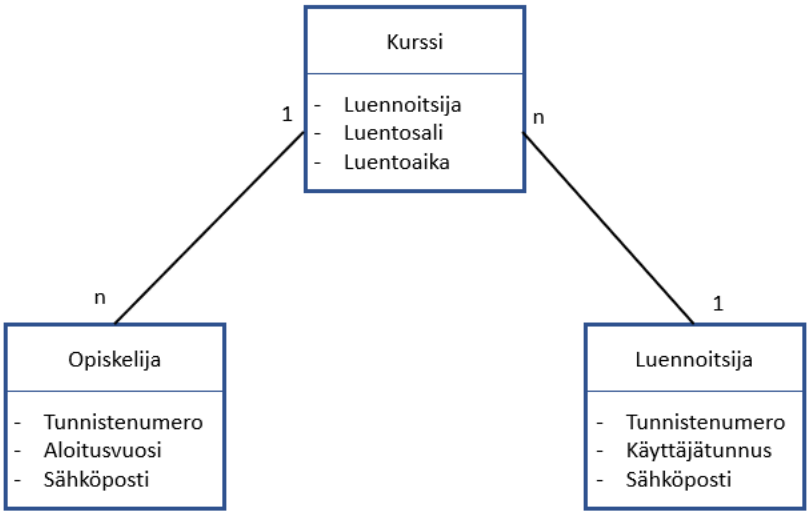
UML-kaavioiden käytännön ongelma on niiden tekninen tapa kuvata asioita kohtalaisen monimutkaisella tavalla. Siinä missä luonnollisessa kielessä vaatimusten antaja käytti tekniikkaa, jonka hän varmuudella ymmärsi ja ohjelmistokehittäjillä oli riski ymmärtää asiat eri lailla, UML kääntää tämän asetelman ylösalaisin. Tuotekehittäjien puolesta UML kertoo käytännössä kaiken, mitä ohjelmiston tekemiseen tarvitaan, mutta taas ohjelmiston loppukäyttäjälle se ei kerro juurikaan mitään monimutkaisuutensa takia. Tästä syystä mallinnuksesta käytetään lähinnä kahta eri kaaviomallia, luokkakaaviota ja käyttötapauskaaviota. (Hood, Fichtinger, Pautz & Wiedemann 2008, 53.)

Luokkakaavio koostuu visuaalisesti laatikoista, jotka rakentuvat kolmesta osiosta. Ylimpänä on luokan nimi, sekä mahdolliset lisätiedot. Nimen alapuolella on kyseisen luokan sisältämät tiedot ja niiden lisätiedot. Alimmaisena on kyseisen luokan toiminnot eli mitä funktioita luokalla on. (Haikala & Mikkonen 2010, 85.) Kuviossa 18 on esimerkinomaisesti luokkakaavion luokkia.



KUVIO 18. Erilaisia luokkia

Luokkakaavioita voidaan käyttää monella eri tavalla ja yksi yleisimmistä tavoista on esittää eri luokkien suhteet ja näin selkeyttää ohjelmistossa tapahtuvia asioita. Kuviossa 19 olen pyrkinyt tuomaan esille kolmen erilaisen luokan keskinäiset suhteet.

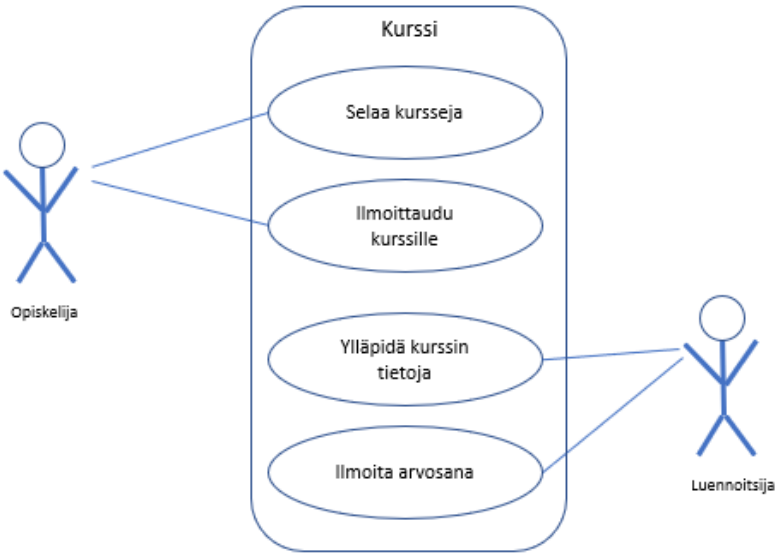


KUVIO 19. Luokkien suhteet

UML-mallinnuksesta toinen useasti käytetty kaaviomalli on käyttötapauskaavio. Tämän kaavion tarkoitus on käytännössä kertoa, mitä toiminnallisuuksia jokaiselle luokalle löytyy. Kuviossa 20 olen pyrkinyt näyttämään, kuinka erilaiset

aktorit käyttävät luokkia ja muodostavat näin erilaisia käyttötapauksia. Käyttötapauskaavion lisäksi käyttötapaukset dokumentoidaan myös luonnollisella kielellä. Tämä on ohjelmiston loppukäyttäjälle helpoin tapa saada mahdollisimman tarkkaa tietoa siitä, mitä ohjelmistolla on tarkoitus pystyä tekemään ja tuotekehitykselle varmuus minkälaisiin vaihtoehtoisiin käyttömahdollisuuksiin ohjelmiston vähintään pitää pystyä. (Haikala & Mikkonen 2011, 80.)

Käyttötapausten dokumentoinnissa on suositeltavaa toteuttaa ainakin seuraavat asiat. Käyttötapauksella tulee olla selkeä luetteloiva koodi, kuvaava nimi ja sen pitää kertoa, mitä aktoreita käyttötapauksessa on. Lisäksi tarvitaan itse käyttötapaus eli kuvaus mitä ohjelmistolla on tarkoitus tehdä ja sen tekemisen lopputulos eli mitä kyseinen kuvaus saa ohjelmistossa aikaan. (Haikala & Mikkonen 2011, 80.)



KUVIO 20. Käyttötapauskaavio

Pelkkä visuaalinen kaavio ei välttämättä avaa katsojalle käyttötapausta tarpeeksi tarkasti ja tästä syystä on myös suositeltavaa toteuttaa määrämittainen taulukko käyttötapauksesta, niin että siinä on tekstin avulla kerrottu, mitä käyttötapauksessa tapahtuu. Kuvion 20 käyttötapaus on avattuna taulukoissa 18 ja 19.

TAULUKKO 18. Käyttötapaus esimerkki 1

Käyttötapaus 1	Kurssille ilmoittautuminen
Käyttäjä	Opiskelija
Kuvaus	Opiskelija valitsee käyttöliittymästä haluamansa kurssin ja täyttää tarvittavat tiedot. Tämän jälkeen opiskelija lähettää kurssi-ilmoittautumisen hyväksyttäväksi.
Lopputulos	Kurssi-ilmoittautuminen on lähetetty hyväksyttäväksi ja opiskelija on saanut onnistuneesta lähetyksestä tiedon sähköpostiinsa.

TAULUKKO 19. Käyttötapaus esimerkki 2

Käyttötapaus 2	Kurssi-ilmoittautumisen hyväksyminen
Käyttäjä	Luennoitsija
Kuvaus	Luennoitsija valitsee käyttöliittymästä, valitsemansa kurssin alta, ilmoittautuneen opiskelijan ja hyväksyy tämän ilmoituksen.
Lopputulos	Opiskelija on hyväksytty kurssille ja hän on saanut hyväksynnästä sähköpostin. Hyväksytty opiskelija on näkyvillä kurssin osallistujat listauksessa.

Käyttäjätarinat, jotka eivät kuulu enää UML-mallinnukseen, ovat yksinkertaistettuja käyttötapauksia, joiden tarkoituksena on olla tarpeeksi helppoja ja yksinkertaisia toteuttaa nopeasti ja pienimmällä mahdollisella dokumentaatiolla. Tämän ansiosta käyttäjätarinoilla pystytään mallintamaan ohjelmistoon tulevia vaatimuksia erilaisissa ketterän kehityksen malleissa, joissa muutokset vaatimuksissa ovat mahdollisia jopa päivittäin. (Beyer 2010, 41.)

Käyttäjätarinat pelkästään eivät kerro kehittäjille tarpeeksi yksityiskohtaisesti sitä, mitä pitää toteuttaa, mutta niissä kerrottu käyttäjätarina toimii pohjana jatkojalostukselle, jossa tarina pilkotaan pienemmiksi ja tarkemmiksi töiksi kehittäjiä varten. (Beyer 2010, 41.) Taulukoissa 20 ja 21 on kerrottuna taulukoiden 18 ja 19 käyttötapaukset käyttäjätarinamuodossa.

TAULUKKO 20. Käyttäjätarina esimerkki 1

Käyttäjätarina 1	Opiskelijana haluan pystyä selailemaan valittavia kursseja, valitsemaan niistä haluamani ja ilmoittautumaan sille.

TAULUKKO 21. Käyttäjätarina esimerkki 2

Käyttäjätarina 2	Luennoitsijana haluan pystyä selailemaan kurssilleni ilmoittautuneita opiskelijoita ja hyväksymään heidän ilmoittautumisensa.

Vaatimusta on mahdollisuus lähestyä myös käyttöliittymäsuunnitelman kautta, koska käytännössä kaikki ohjelmistot nykypäivänä sisältävät jossain muodossa käyttöliittymän, jota ihminen voi käyttää. Käyttöliittymäsuunnitelmassa ohjelmistosta luodaan visuaalinen ja mahdollisesti jopa interaktiivinen malli, jonka avulla loppukäyttäjät pääsevät näkemään miltä lopullinen ohjelmisto näyttää ja miten sitä käytetään. (Pressman 2010, 313.)

Ketterät menetelmät aiheuttavat omat ongelmansa vaatimusten käsittelyn eri vaiheisiin, lupaamalla sisäänkirjoitetusti mahdollisuuden muuttaa kaikkea sitä mukaan, kun projekti etenee ja tieto kaikilla osapuolilla lisääntyy (Beyer 2010, 11).
Ketterät menetelmät eivät kuitenkaan lähtökohtaisesti ota kantaa, miten tämä muutosten mahdollisuus tulisi toteuttaa ja millä tavoin muutosten aikana pystytään pitämään huolta jäljitettävyydestä ja varmistamaan, että ohjelmisto lopulta ratkaisee sen ongelman, mihin sitä ollaan kehittämässä. (Beyer 2010, 11).

Vaatimusten lopullisen oikean tiedon tullessa aina ohjelmiston loppukäyttäjiltä, on määritysten tekemisessä ja muuttamisessa aina oltava mukana mahdollisuuksien mukaan, joko loppukäyttäjät ja jonkunlaiset loppukäyttäjien toimintaa simuloivat roolit, esimerkiksi käyttöliittymäsuunnittelijat. Tämän toimintamallin avulla vaatimukset eivät ole vain sitä, mitä tuoteomistaja ja kehittäjät yhdessä keksivät palavereissa. (Beyer 2010, 11.)

Beyer Hans esittää kirjassaan User centered agile methods muutaman huomion nykyaikaisen ketterän mallin mukaan toimivan tuotekehityksen haasteista loppukäyttäjävaatimusten kanssa.

1) Käyttäjät eivät osaa kertoa mitä tekevät
   - Käyttäjät eivät siis tiedä miten oikeasti kuvaillaan ongelmaa, johon ohjelmistoa ollaan luomassa. Ajatukset harhailevat korjaamaan pieniä yksityiskohtia sieltä täältä, joiden osalta ollaan varmoja ja isoihin epäselviin kokonaisuuksiin ei kosketa.

1) Käyttäjät haluavat olla avuliaita
   - Käyttäjät haluavat olla apuna kehittämässä järjestelmää, mutta tämä voi johtaa siihen, että käyttäjät kertovat pieniä korjauksia sinne tänne, kun heidän pitäisi sanoa, että koko järjestelmä ei vain tue heidän toiveitaan.

2) Käyttäjät eivät ole käytettävissä
   - Ei ole yllätys, että loppukäyttäjillä on usein omia töitään ja tekemisiään, joten he eivät voi olla käytettävissä tuotekehitykselle jatkuvasti ja mitä enemmän he ovat tuotekehityksen käytettävissä, sitä vähemmän he kerkeävät työskentelemään sen alkuperäisen ongelman kanssa.

3) Sijaiskäyttäjät eivät ole käyttäjiä
   - Yrityksen loppukäyttäjien tilalle asettamat roolit eivät ole oikeita käyttäjiä ja eivät voi tietää loppukäyttäjien vaatimuksia. Tämä koskee myös tuoteomistajaa.

4) Markkinointitutkimukset eivät kerää suunnitteludataa
   - Kun liiketoiminnot haluavat kerätä käyttäjädataa, ne tekevät markkinatutkimuksen ja päättelevät siitä. Tässä täytyy pitää mielessä, että nämä tutkimukset eivät kerää dataa käyttäjien ongelmista vaan toiveista ylätasolla.

(2010, 19.)

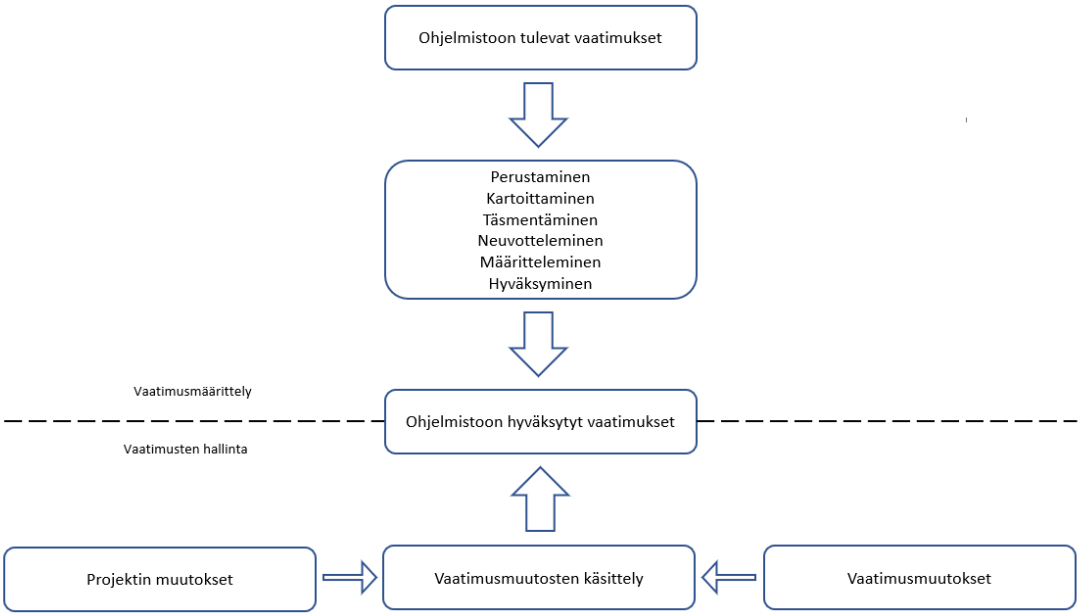
6.2 Vaatimusten määrittely

Ohjelmistoprojektin vaatimusten voidaan katsoa aina noudattavan seitsemää askelta, jotka Pressman kirjoittaa Software engineering kirjassaan olevan perustaminen, kartoittaminen, täsmentäminen, neuvotteleminen, määritteleminen, hyväksyminen ja lopulta hallinta (2010, 121).

Kaikki alkaa perustamisesta, jossa liiketoiminta eri sidosryhmineen hahmottelee liiketoimintaidean tarpeeksi syvällisellä tasolla, jotta siihen pystytään tekemään toimiva kuvaus projektin laajuudesta (Hood ym 2008, 32). Perustamisen tarkoitus siis on saada organisaatiolle tietoon perusymmärrys ongelmasta, tietoon ne henkilöt, jotka haluavat ongelmaan ratkaisua ja ymmärryksen halutun sovelluksen luonteesta. Lisäksi tulee mahdollistaa tehokas kommunikaatio eri sidosryhmien välillä heti alusta alkaen (Pressman 2010, 121). Kartoittamisen tarkoitus on selventää ohjelmistoa tekeville henkilöille, mitä loppukäyttäjät ohjelmistolta tarvitsevat oman liiketoimintansa tai arkensa avuksi. Tämä kuulostaa yksinkertaiselta, mutta ei missään nimessä sitä ole. Itse asiassa vaatimusten kartoituksen on sanottu olevan yhtä helppoa, kuin paimentaa perhosparvea (Hood ym 2008, 52). Ongelmia aiheuttavia osa-alueita on ensinnäkin sovelluksen laajuus, eli mitä kaikkea ohjelmistoon pystytään annetussa aikaikkunassa ja olemassa olevilla resursseilla tekemään. Toisekseen molemminpuolinen ymmärrys, mitä projektin on tarkoitus toteuttaa voi puuttua. Loppukäyttäjät eivät ymmärrä sovelluksen tarvitsemaa teknistä ympäristöä ja tekijät eivät ymmärrä loppukäyttäjän käyttöympäristöä. Kolmantena ja viimeisenä ongelmakohtana on vaatimusten epävakaisuus, toisin sanoen vaatimukset aina muuttuvat projektin edetessä. (Pressman 2010, 122.)

Vaatimusten kartoittamisen jälkeen on edessä vaatimusten tarkentaminen. Tässä vaiheessa sidosryhmiltä ja loppukäyttäjiltä kerätty tieto jalostetaan erimallisiksi skenaarioiksi ja esimerkiksi käyttäjätarinoiksi, joilla pyritään selkeyttämään vaatimuksia niin, että kaikki tahot ymmärtävät ne samanlailla. Lisäksi tavoitellaan ymmärrystä eri vaatimusten suhteista ja prioriteeteistä. (Pressman 2010, 122.) Neljännessä askeleessa otetaan käytännössä ensimmäinen askel kohti ohjelmiston kehitystä eli neuvotellaan saavutetulla tiedolla, miten sovellusta aiotaan rakentaa. Tähän asti asiat ovat olleet enemmän idealismin tasolla, mutta tässä vaiheessa mukaan tulee realismia niin resurssien, kuin maksimihinnoittelunkin puolesta. Tavoitteena on neuvotella ja käydä läpi kaikkien sidosryhmien kanssa vaatimukset niin, että niitä parannetaan, yhdistellään ja turhat poistetaan kompromissien mukaisesti (Pressman 2010, 122.)

Kaiken edellä mainitun jälkeen projekti käytännössä alkaa. Tavoitteena on kaikessa yksinkertaisuudessa määritellä vaatimukset niin tarkasti, kuin se on järkevällä työmäärällä mahdollista. Määrittelyä voidaan toteuttaa usealla eri tavalla. Voidaan kirjoittaa erilaisia käyttötapauskuvauksia, joihin lisätään erilaisia prosessikaavioita ja graafisia malleja. Ohjelmistosta voidaan toteuttaa prototyyppi tai käyttöliittymämallinnus. Voidaan myös käyttää kaikkia edellä mainittuja yhdessä. Tavoitteena on, että tämän vaiheen jälkeen kaikki ymmärtävät mihin suuntaan ohjelmiston kehitys on etenemässä. (Hood ym 2008, 53.) Viimeisenä vaiheena ennen toteutuksen alkua on validointi, jossa käydään sidosryhmien kanssa läpi kaikki vaatimuksista toteutettu dokumentointi ja tarkastellaan, onko se hyväksyttävän muotoista ja kaikille osapuolille selkeää (Pressman 2010, 124). Kuviossa 21 on selkeytetty, kuinka vaatimusten määrittely ja hallinta eroavat toisistaan.



KUVIO 21. Vaatimusten jakautuminen

6.3 Vaatimusten hallinta

Vaatimusten hallintaa voisi kuvata kaikkien niiden toimenpiteiden summaksi, jotka tapahtuvat vaatimusten määrittelyjen jälkeen. Vaatimusten hallinnalla tarkoitetaan siis mekanismia, jossa ei keskitytä uusien projektien uusiin vaatimuksiin, vaan pyritään toteuttamaan hallittua muutosta olemassa oleviin

vaatimuksiin. Tavoitteena on, että vaatimuksille annettu arvo pysyy koko kehitysputken ajan, siitä hetkestä lähtien, kun vaatimus on ensimmäistä kertaa kartoitettu ja dokumentoitu (Hood ym 2008, 59.)

Muutokset vaatimuksissa johtavat lähes väistämättä muutokseen työmäärässä, budjetissa tai aikataulussa ja koska ohjelmistotuotannossa voidaan aina lähteä oletuksesta, että vaatimukset muuttuvat, niin vaatimusten hallinnan perimmäinen tarkoitus on mahdollistaa tuo muutos niin, että laadullisiin ja liiketoiminnallisiin tavoitteisiin päästään mahdollisimman useasti. (Haikala & Mikkonen 2011, 67.)

6.4 Tietoturva ja vaatimukset

Vaatimusten hallintaa on useasti kuvattu ohjelmistokehityksen vaikeimpana osa-alueena ja yleensä vaatimusten hallinnassa esiintyviä virheitä lähdetään korjaamaan panostamalla vaatimusten määrittelyyn. Tämä on hyvin harvoin oikea tie, koska ongelmaa hyvin harvoin aiheuttaa esitetty vaatimus, vaan esitetyn vaatimuksen väärä tulkinta ja siitä johtuva muutos prosessin myöhäisemmässä vaiheessa.

Vaatimusten muutokset ja siitä johtuva vaatimusten hallinta on avainroolissa, kun mietitään tietoturvaa ohjelmistokehitysprosessissa. Tietoturva vaatimusten hallinnassa voi tarkoittaa kahta eri lähestymistapaa, toinen on mahdollisuus luoda prosessissa ohjelmistoon tietoturvavaatimuksia, jotka sitten kuljetetaan osaksi ohjelmistoa, kuten muutkin vaatimusten hallinnan läpi tulevat ominaisuudet. Toinen osa prosessia on eri ominaisuuksille määritellyt tietoturvavaatimukset, toisin sanoen kaikkien vaatimusten määrittelyissä ja muutoksissa pitäisi ottaa huomioon ennalta määritellyt tietoturvavaatimukset. Haastatteluissa vaatimusten hallinta nousi esille organisaation isoimpana ongelmakenttänä puhuttaessa niin tietoturvasta, kuin yleisestikin tuotekehityksestä. Tämä on odotettavaa, koska kuten aikaisemminkin on mainittu, nähdään vaatimusten hallinta yleisesti ohjelmistokehityksen vaikeimpana osa-alueena.

Haastatteluissa vaatimusten hallinnan tärkeimpänä lähtökohtana pidettiin toimintaan käytettävää järjestelmää, jossa toimiakseen vaatimusten hallintaan tulee olla vähintään kolme avainominaisuutta. Ensimmäiseksi järjestelmän tulee olla lähtökohdiltaan hierarkkinen, eli ominaisuudet ja niiden vaatimukset tulee pystyä sitomaan toisiinsa kiinni siten, että järjestelmä on loogisesti ymmärrettävissä. Toisekseen järjestelmässä tulee olla mahdollisuus linkittää ominaisuuksia ja niiden vaatimuksia toisiinsa myös yli ominaisuuksien ja näin varmistaa, että kaikki vaatimukset tulevat huomioiduiksi myös muissa ominaisuuksissa, joihin kyseiset vaatimukset vaikuttavat. Kolmantena kohtana haastatteluissa tuli esille tarve pystyä luokittelemaan erilaisilla merkinnöillä vaatimuksia ja näin pystyä nopeasti tarkastamaan, mitkä asiat ovat tietoturvaliitännäisiä.

Nämä samat kolme ominaisuutta olivat vaatimuksena Digi- ja väestötietokeskuksen turvallisen sovelluskehityksen ohjeistuksessa. Tässä voidaan siis todeta olevan tietyntasoinen esikuva-analyysi siitä, mitä vaatimusten hallinnan tulee pitää sisällään toimiakseen. Huomioitavaa kuitenkin on yhden haastatellun kommentti ”En tiedä ketä palvelisi se linkitettyjen vaatimusten ja merkintöjen neliulotteinen hämähäkinverkko”, tämän vastauksen tulkintana on, että vaatimusten hallinta on yksinkertaistettava ja tasapäistettävä niin pitkälle, että se palvelee ketterää kehitysmallia ja eri osia organisaatiossa.

Kysyttäessä muuttuvista vaatimuksista eri vaiheessa tuotekehitysprosessia oli vastaukset hyvin linjassaan. Haastatellut olivat lähes yksimielisiä siitä, että väärillä vaatimuksilla tehtäviä ominaisuuksia ei kannata julkaista. Sen sijaan vastauksissa oli havaittavia eroja sen perusteella, kuinka lähellä asiakasrajapintaa henkilö toimi. Tuotekehitystiimissä oli vahva konsensus reagoida tilanteeseen viivästyttämällä versiojulkaisua, kun taas tiimien ulkopuolella oli ajatuksia erilaisista ominaisuuksien pois jättämisistä ja asiakkaan konsultoimisesta.

7 TUTKIMUKSEN ANALYSOINTI

Tämän opinnäytetyön tavoitteena on ollut saada ymmärrys ketterän ohjelmistokehityksen lainalaisuuksista, sekä tarjota malli nykyaikaisen tietoturva-ajattelun sitomiseen käytössä olevaan prosessiin. Tätä tavoitetta varten tässä työssä on avattu laaja-alaisesti ohjelmistokehitystä koskevaa teoriaa ja parhaita käytäntöjä. Tätä samaa tavoitetta kohti pyrittiin pääsemään tehdyn tutkimuksen avulla.

Tutkimuksen tärkein osa oli henkilöhaastattelut, jotka toteutettiin tammikuussa vuonna 2020 valituille henkilöille organisaatiosta. Haastatellut valittiin niin, että saatiin mahdollisimman kattava edustus opinnäytetyön aiheena olevan tietoturvallisen tuotekehitysprosessin alueelta. Tavoitteena oli siis saada ajantasainen kuva organisaation kyvykkyydestä toimia tuotekehityksessä ja varsinkin tietoturvan hoitamisesta siinä. Haastattelut veivät yhteensä kolme viikkoa, jonka jälkeen kaikki neljätoista haastateltavaa olivat haastateltu.

Haastatelluilla ei ollut haastattelun aluksi tietoa kysymyksistä tai niiden aihealueista. Haastattelut aloitettiin käymällä läpi yleiset asiat, tutkimuskysymykset ja tutkimuksen eettiset säännöt. Tämän jälkeen avattiin kysymysten aihealueet ja kerrottiin, kuinka monta kysymystä per aihe alue on. Lisäksi kerrottiin, että haastatteluissa jokaiselle haastatellulle annetaan yksi tunti aikaa vastata kaikkiin 24 kysymykseen. Haastattelut pidettiin Microsoft Teams työkalulla, jonka avulla myös kaikki haastattelut tallennettiin Microsoft Stream palveluun. Haastatteluissa, sekä tätä myötä tallenteissa jaettiin Microsoft Powerpoint työkalulla käsiteltävä kysymys, joten haastattelija ja haastateltava eivät nähneet toisiaan haastattelujen aikana. Haastattelut pysyivät kaikkien haastateltujen osalta samana, lukuun ottamatta kysymystä 22, jonka haastattelija totesi turhaksi kolmannen haastattelun jälkeen, johon mennessä kaikissa kolmessa haastattelussa kysymykseen oli tyhjentävästi vastattu jo kysymyksen 21 aikana. Tämän jälkeen haastatelluille annettiin mahdollisuus tarkentaa kysymyksen 21 vastausta kysymyksen 22 aikana jos niin halusivat, mutta kyseiseen kysymykseen ei käytännössä vastattu kenenkään haastateltavan toimesta.

Haastatteluissa haastattelija teki muistiinpanoja jokaisesta haastateltavasta per kysymys, tavoitteena nostaa tärkeimpiä asiasanoja esille ja näin valmistella tulevaa haastattelujen litterointia ja analysointia. Asiasanat kerättiin Microsoft OneNote työkaluun ja siirrettiin siitä Microsoft Excel työkaluun, jossa samojen asiasanojen toistuvuutta per kysymys analysoitiin. Kaikkien haastattelujen ollessa valmiina, haastattelut litteroitiin referoivan litteroinnin mallilla kuuntelemalla ne tarpeeksi monta kertaa Microsoft Stream työkalusta ja kirjoittamalla Microsoft OneNote työkaluun ranskalaisin viivoin eriteltynä tärkeimpiä huomioita. Litteroinnin valmistumisen jälkeen auki kirjoitetut vastaukset kysymyksiin siirrettiin Microsoft Excel työkaluun, jossa jokainen kysymys muodosti oman sarakkeen, jonka alle haastattelujärjestyksessä jokaisen haastateltavan vastaukset tallennettiin. Näin luotiin näkymä kaikkiin vastauksiin kysymyksittäin ja pystyttiin hakemaan eroja ja yhtäläisyyksiä vastauksista. Tämän jälkeen vastauksia luokiteltiin värikoodeittain niin, että kysymyksen sarakkeessa olevista 14 eri vastaajan solusta samankaltaisuuksia vastaajien joukossa värjättiin samalla värillä ja jos vastauksessa oli paljon hajontaa, pyrittiin luokittelun avulla löytämään yleisempi mielipide ja huomioimaan täysin poikkeavat vastaukset.

Toteutetulla analyysillä pystyttiin aineistosta päättelemään useita tämän opinnäytetyön aihealueena olevia asioita. Haastatelluista kaikki olivat sitä mieltä, että heillä on käsitys, kuinka he pystyvät vaikuttamaan tuotekehitykseen oman työnsä kautta, tämä on luonnollisesti ohjelmistoyritykselle elintärkeää ja kertoo organisaatiolla olevan ainakin tässä tapauksessa oikeankaltaisia rooleja. Toinen selkeä, mutta ei niin positiivinen huomio tuloksista oli todella vahva polarisaatio niin ketterien menetelmien, kuin niiden sisältämien roolien tuntemisessa. Voidaan todeta, että osa haastatelluista tunsi lähes kaiken, mutta suurella osalla oli vakavia haasteita mallien tuntemisessa. Edelleen vahvaa osaamista organisaatio pystyi näyttämään tietoturvan ja tietosuojan erittelyllä, jossa lähes kaikki haastateltavat tiesivät mistä on kyse. Eräskin haastateltava totesi: *Tietoturva on tapoja suojata tietoa ja tietosuoja on kuvaus tiedon käytöstä ja siitä mitä tiedolla tehdään*. Muutama haastatelluista esitti tämän asian juuri päinvastoin, mutta tämä voidaan selittää haastattelutilanteen jännityksellä.

Keskityttäessä vaatimusten hallintaa koskeviin kysymyksiin pystyttiin myös tulosten analysoinnista päättelemään tiettyjä asioita. Ensinnäkin voitiin todeta kaikkien haastatelluiden olevan sitä mieltä, että vaatimusten hallinta vaatii toimiakseen tiketöintijärjestelmän, joka tukee myös erilaisten vaatimusten hierarkiaa ja linkitystä toisiinsa. Haastatellut olivat myös hyvin yksimielisiä siitä, että vaatimusten hallinta ei ole vain käytettävä ohjelmisto, vaan vaatii toimiakseen organisaation, joka tukee vaatimusten tulkintaa, auki kirjoittamista ja kokonaishallintaa. Eräs haastatelluista ilmaisi asian ponnekkaasti toteamalla: *Ilman asiakkaan ja kehityksen välistä analysointikerrosta ei voida onnistua*. Puhuttaessa tarkemmin vaatimuksiin ja sitä kautta ominaisuuksiin tulevista muutoksista, näkyi tuloksissa erittäin vahvasti eroa tuotekehityksen ja liiketoiminnan välillä. Siinä missä tuotekehityksen vastauksissa tehtäviin muutoksiin ja muutettaviin julkaisupäiviin kanta oli hyvinkin tiukka, niin liiketoiminnan edustajilla se oli selkeästi enemmän ratkaisuhaluinen ja asiakkaita palveleva. Se mikä tämänkin aihealueen vastauksista voidaan päätellä, on ketterien menetelmien perimmäisen tarkoituksen ymmärryksen puute, eli jatkuva valmius muutokseen on se mihin organisaation pitäisi olla aina valmis. Haastatelluista vain yksi otti esille mahdollisuuden kasvattaa DevOps maturiteettia, hyväksyä muutos ja julkaista uusia versioita tarpeeksi usein, jotta muutokset tulevat huomioiduiksi aina kun niille on oikea tarve. Lopulta kuitenkin ohjelmistoa tehdään aina käyttäjiä varten, kuten yksi haastatelluista sanoi: *jos olet epävarma, niin kuuntele asiakasta*.

Tämän opinnäytetyön näkökulma tuotekehityksen kokonaisprosessiin oli tietoturva ja siitä myös oli lukumääräisesti eniten kysymyksiä. Näiden vastausten analysointi myös osoitti eniten muutoksen ja parannuksen tarvetta eri osiin organisaatiota. Yleisesti voidaan tulkita, että ymmärrys tietoturvan käytännön malleista ei ollut riittävä, mutta käsitys niiden tärkeydestä oli selkeä. Tämän osuuden haastatteluissa myös tuli selkeästi esiin tarve huolehtia teknisten roolien tarpeeksi suuresta resurssoinnista. Käsiteltävät osa-alueet ovat vaikeita ja varsinkin aikaa vieviä ohjelmistokehityksessä, kuten eräs haastatelluista sanoi: *Malli, jossa koodataan ominaisuus ja liimataan päälle laatu, sekä tietoturva ei tule koskaan toimimaan*. Tämän osa-alueen kysymyksissä oli organisaation kannalta hienointa se, että haastatelluista 93% halusi olla mukana kehittämässä

tietoturvallista ohjelmistokehitystä ja sama määrä myös jatkuvasti sitä parantamassa omien organisaatiorooliensa mukaisesti.

Haastatteluissa tuli varsinkin tuotekehityksen osalta esille konkreettisia ehdotuksia ja malleja, millä tavoin tietoturvaa voidaan tuotekehityksessä parantaa. Näistä esimerkkeinä haastatellut antoivat ketterän kehityksen menetelmien mukaisen niin kutsutun valmiin määritelmän tarkennuksen, koodiskannaukset, tietoturvaan kohdentuvat negatiiviset testaukset ja ominaisuuksien ohjeistusten tarkentamisen ja parantamisen.

Haastatteluista voidaan myös tietoturvan osalta todeta, että suurin osa haastatelluista ajatteli tietoturvan olevan osa organisaatiokulttuuria. Se onko se sitä tällä hetkellä, jakoi mielipiteitä ehkä hitusen enemmän sille kannalle, että kulttuuria ei ole tai ainakaan sitä ei tunnuta ehdittävän rikastamaan tarpeeksi. Hyvänä esimerkkinä oli toisen haastatellun sanoessa: *Espoon toimistolla kulunvalvonta on priima kunnossa*, toisen haastatellun taas kertoessa: *Oltiin kaksi päivää Espoon toimistolla ja kukaan ei kysynyt keitä ollaan*. Tämä konkreettinen esimerkki tietoturvan ja tietosuojan ensimmäiseltä tasolta kertoo, kuinka monimutkaisista ja monta organisaatiotasoa leikkaavista asioista on kyse. Yllämainitun lisäksi ehdottoman tärkeäksi osaksi tietoturvaa miellettiin rooli tai roolit, joilla on yksiselitteinen vastuu tietoturvaan liittyvistä asioista. Kuten tuoteomistajalla on vastuu, mutta ei tarve tehdä kaikkea tuotteeseen liittyvää, tulee tietoturvavastaavalla olla vastuu, mutta ei tehdä kaikkea tietoturvaan liittyvää.

Kaiken kaikkiaan tietoturvan isoimpana haasteena, mutta myös organisaation vahvuutena nähtiin sen kuuluminen kiinteästi osaksi organisaatiokulttuuria. Kun tietoturva-ajattelu on olemassa, kuten tutkimuksessa voidaan todeta, tärkeimmäksi haasteeksi muodostuu tuosta kulttuurista kiinni pitäminen nykyajan hektisessä yritysmaailmassa. Haastattelujen lisäksi tutkimuksessa pyrittiin ottamaan huomioon organisaatioon jo aikaisemmin toteutettujen tietoturva ja tietosuoja ohjeistusten ja toimintasuunnitelmien analysointi. Tietosuojan osalta organisaatiossa on otettu isoja askelia ja tietosuoja ajattelu on ulotettu läpi organisaation eri dokumenttien tukemana ja tätä työtä tukemassa on useita asiaan vihkiytyneitä rooleja. Tietoturvan osalta näin ei valitettavasti ole, suurelta

osin dokumentointi on puutteellista tai sitä ei ole, roolit ovat epäselviä ja prosessissa ei ole annettu selkeää tukea tämän työn tekemiselle.

8 POHDINTA

Tätä työtä suunnitellessani oli minulla kirkkaana mielessä rakentaa pyhä kolminaisuus näiden aihealueiden väliin. Tietoturvasta halusin punaisen langan, jonka kuljettaminen läpi organisaation pakottaisi pohtimaan, mitä tuotekehitykseltä oikeasti halutaan ja mihin tuotekehityksen pitää valmistautua. Mielestäni tässä onnistuin kohtalaisen hyvin.

Tässä työssä käsitelty organisaatio on osa isoa konsernia, jonka toiminta on, vaikkakin omassa solussaan, silti monelta osin erittäin läheisessä yhteydessä konsernin kanssa. Tämän seurauksena organisaatiossa noudatettavia tietoturva- ja tietosuojakäytäntöjä tulee myös niin sanotusti ylhäältä annettuna. Nämä asiat ovat yleensä konkreettisempia asioita, jotka liittyvät esimerkiksi käytettäviin ohjelmistoihin, toimistojen kulunvalvontaan, käyttäjätunnuspolitiikkaan ja vastaaviin. Toisin sanoen konserni hoitaa työkalut ja toimitilat omilla määräyksillään tietoturvallisiksi, mutta liiketoiminta itse määrittelee kaiken tämän ulkopuolella. Tietoturva ja tietosuoja työ on myös monelta osin parantunut viime vuosien aikana esimerkiksi GDPR asetuksen ohjaamana ja nyt on esimerkiksi jo erittäin epätodennäköistä saada kontrolloimatonta pääsyä organisaation toimitiloihin, vaikkakin tutkimus osoitti sen edelleen olevan mahdollista.

Vaikka tietoturvaa yleisesti pidetään tietynlaisena omana kokonaisuutenaan, niin ilman ymmärrystä moderneista tuotekehityksen prosesseista ja vaatimustenhallinnan merkityksestä ominaisuuksien määrittelyihin ja toteutuksiin, ei ole mahdollista toteuttaa tietoturvaa tavalla, joka oikeasti toimisi ja antaisi rutiininomaisesti tuotteelle tietoturvallisia ominaisuuksia, sama ajatus toistui myös haastatteluissa lähestulkoon kaikkien kanssa. Haastattelut paljastivat myös omasta mielestäni organisaation pahimman haavoittuvuuden, eli tietous ketteristä malleista ja vaatimusten hallinnasta oli huomattavan vajaata.

Tulevaisuuden toimia ajatellen, tämä on myös selkeä parannuskohta organisaatiolle. Tietoutta tarvitaan lisää ja tätä varten tulee organisaatiota

kouluttaa niin ketteriin malleihin, kuin vaatimusten hallintaan. Monen haastattelun kohdalla oli haastattelijalla alun perin tunne, että tällä ymmärryksellä ketteristä malleista, vaatimuksista tai tietoturvasta ei voida saada aikaan toimivaa prosessia ja mallia, jolla vältetään tietoturvatapaukset tulevaisuudessa. Lopulta kuitenkin aloin hahmottamaan asiaa toiselta kannalta ja ymmärsin, että jokaisella yrityksellä on omat tapansa rakentaa työ abstraktin asiantuntijatoiminnan ympärille ja punaisen langan tulee pystyä sitomaan toimintaan, vaikka se ei teoreettisesti välttämättä kaikkein puritaanisimmilla malleilla oikeaa olekaan.

Tämän opinnäytetyön tekemistä ohjasi koulutuksen alussa suunniteltu kehittämistehtävä tuotekehityksen kokonaisprosessista. Tälle prosessikehitykselle annetaan lähtölaukaus tässä työssä kerätyillä materiaaleilla, opeilla ja tutkimustuloksilla. Tehdyt esimerkkiprosessit, joissa käsitellään tämän työn aihealueita löytyvät liitteestä kaksi.

Tämän työn osalta on läpikäytynä tietoturvassa käytettävänä konsepti uhkamallinnuksesta ja siihen liittyvästä STRIDE-viitekehyksestä. Vaikka on huomioituna, että STRIDE-mallia ollaan korvaamassa uudemmilla, on malli itsessään vielä täysin käyttökelpoinen ja hyväksyttävä konkreettista mallia luotaessa. On enemmän organisaation päätös yhdessä, kuin kenenkään tahon yksin, mikä on malli ja suuntaus mihin tietoturvan parantamisessa mennään. Kaikki alkaa kuitenkin tarpeellisen prosessin luomisesta ja hyväksymisestä, johon on pyritty vastaamaan kehittämistehtävän osalta. Liitteen 2 kuviossa 32 on kuvattuna, kuinka näillä malleilla voidaan analysoida tekemistä jatkuvasti ja kuvioissa 33 ja 34 kuinka tietoturva voidaan ottaa huomioon useassakin kohdassa ydinprosessia.

Toinen haastatteluissa vahvasti mukana ollut ajatus, jonka korjaaminen on ehdottoman tärkeää yrityksellä, on tämän aihealueen roolien selkeyttäminen. Teknisten ihmisten ollessa kyseessä on ensiarvoisen tärkeää ymmärtää tarkasti määriteltyjen prosessien ja roolien merkitys. Tällöin ihmisten ei tarvitse olettaa ratkaisun olevan jonkun toisen työn takana, vaan myös itse ollaan valmiita ratkaisemaan ongelmia. Aina kuitenkin vaaditaan tietynlainen struktuuri, jotta autonominen ja moderni tuotekehitysyksikkö on voimissaan ja toimii

itseohjautuvasti läpi erilaisten ongelmien. Asiantuntijaorganisaatioissa roolien monitahoisuus ja niiden vastuualueiden hahmotelmat ovat kuitenkin aina suuri haaste ja vaativat paljon huomiota. Liitteen 2 kuvioissa 29, 30 ja 31 on tuotu esille minkälaisia rooleja ja missä yksiköissä tarvitaan prosessin toimintaa tukemaan.

Ihmisten ja prosessien lisäksi, erittäin tärkeänä asiana kehitettävien asioiden osalta on modernit tuotantoputkimallit. Kyseisen aihealueen osalta organisaatio on ottanut askeleita oikeaan suuntaan tämän opinnäytetyön tekemisen aikana ja aloittanut kehitysprojekteja, sekä perustanut yksiköitä asiaa edistämään. Tällä tulee olemaan erittäin vahva tietoturvaa parantava vaikutus jo lähitulevaisuudessa. Se myös pakottaa tietyllä tavalla tuotekehitystä ympäröivää organisaatiota sellaiseen muottiin, jossa tietoturvallinen ohjelmistokehitys tai käyttöönottoprojekti on toimiva ja luotettava.

Vaatimusten hallinnan osalta voidaan todeta tietyllä tavalla samaa kuin ketteristä menetelmistä. Organisaation tämän hetkinen tietous sen toteuttamisesta on sen verran ohutta, että toimivaa mallia ei pystytä ilman laajamittaista koulutusta ja läpikäyntiä rakentamaan. Tämä on myös kokonaisuuden kannalta aivan avainroolissa oleva osa prosessia, jonka tehokas toiminta on äärimmäisen tärkeää myös tietoturvan ja tietosuojan jatkuvan parantamisen ja kehittämisen kannalta. Organisaatio tarvitsee vuosittain satojen sidosryhmiltä tulevien vaatimusten hallintaan, niin toimivan hierarkisen tiketöintijärjestelmän, kuin prosessin tuen sen hallintaan. Liitteen 2 kuviossa 26 on kuvattuna esimerkki hierarkisesta järjestelmästä ja kuvioissa 35, 36, 37 ja 38 minkälaisia rooleja tarvitaan hierarkian eri vaiheissa.

Tietosuoja oli puolestaan tutkimuksenkin perusteella hyvin ymmärretty yrityksessä, tietoisuuden laajuus oli odotettavaa, koska koko yrityksen liiketoiminta perustuu työntekijöistä kerättyyn dataan ja sen muokkaamiseen. Asiakokonaisuuden myötä oli selkeästi rakentunut tietynlainen tietosuojan kulttuuri, joka läpi leikkasi hyvin pitkälti koko organisaatiota. Poikkeuksiakin oli, mutta ne olivat yksittäistapauksia ja eivät lopulta vaikuttaneet kokonaisuuteen. Sama asia voidaan todeta myös havainnoimalla yrityksen päivittäistä toimintaa, jota vahvasti hallitsee muistutukset henkilötietojen tärkeydestä ja jatkuvista

tietosuoja-asetusten läpikäyntiin ja ymmärryksen syventämiseen pyrkivistä palaverikutsuista.

Kokonaisuudessaan tehdyn tutkimuksen ja hankitun teorian pohjalta voidaan todeta, että on olemassa mahdollisuus ja työkalut pitää yllä tuotekehityksen tietoturvaa, mutta vain jos organisaationa hyväksytään sen aiheuttamat kustannukset ja hidastukset tuotantoon, sekä tätä kautta laskutukseen. Helppoa, yksinkertaista ja halpaa ratkaisua ei ole ja niiden toteuttamisella aikaansaadaan lähinnä valheellinen turvallisuuden tunne.

Tutkimus olisi ollut mahdollista toteuttaa myös kvantitatiivisilla menetelmillä, mutta näin tehtäessä jotain tärkeää saaduista tuloksista olisi menetetty. Kvalitatiivisissa menetelmissä ongelmia saattaa kuitenkin aiheuttaa tehtävät tulkinnat ja sen myötä tutkimuksen luotettavuuden arviointi. Siltikin, kun tarkoitus on saada oikea tilannekuva syvällä organisaation osaamisessa olevasta aihealueesta, niin kvalitatiiviset menetelmät olivat ainoa realistinen vaihtoehto.

Tutkimuksen suurimmilta osin nojatessa haastatteluihin, ollaan vielä edellä mainittujen tulkintaluottamusten lisäksi tilanteessa, jossa vastaajan rehellisyys vastauksissaan saattaa aiheuttaa suuriakin vaihteluita tuloksissa- Toisaalta jokainen haastateltava luottaa, että tuloksia läpikäytäessä hänen vastauksensa ovat samalla tavalla käsiteltyjä kuin kaikki muutkin aineiston osat ja tämän työn lukija luottaa, että kiteytetyt asiat ja ehdotetut korjaavat toimenpiteet ovat aitoja ja haastatteluihin nojaavia. Haastateltavia oli sopiva määrä verraten työn laajuuteen ja koska haastateltavat edustivat koko organisaatiota, voidaan todeta, että tutkimuksen näkökulmasta tarpeellinen aineisto saatiin kerättyä.

Tutkimuksen voidaan todeta olevan onnistunut, kun se on kuvattu niin, että se pystytään toistamaan ja siihen pystytään luottamaan ja kun siitä tehdyt päätelmät ja analyysit ovat johdonmukaisia ja vakuuttavia. Tämän työn tutkimusta tukee kerätty teoria aihealueiden ympärillä, sekä haastattelujen lisäksi tutkimuksessa käytetty organisaation muu toteutettu materiaali aihealueita koskien. Tutkimus toteutettiin kuukauden ajanjaksona alkuvuodesta 2020, mutta työn kokonaiskestoksi muodostui yhdeksän kuukautta. Heti

aikaisessa vaiheessa päätin lisätä opinnäytetyöhön näkökulman tietoturvasta, joka eroaa hieman kehittämistehtävänä tehdystä puhtaasta prosessityöstä tuotekehitykselle. Tutkimus kuitenkin tutki kokonaisuutta, koska asioiden eriyttäminen omiksi lohkoiksi olisi ollut kokonaisprosessin näkökulmasta mahdotonta.

LÄHTEET

Alasuutari, P. 2011. Laadullinen tutkimus 2.0. 2. painos. Tampere: Osuuskunta Vastapaino.

Asetus 2016/679/EU. luonnollisten henkilöiden suojelusta henkilötietojen käsittelyssä sekä näiden tietojen vapaasta liikkuvuudesta ja direktiivin 95/46/EY kumoamisesta (yleinen tietosuoja-asetus). Euroopan unionin virallinen lehti. 4.5.2016. Luettu 27.7.2020. https://eur-lex.europa.eu/legal-content/FI/TXT/HTML/?uri=CELEX:32016R0679&from=FI

Beyer, H. 2010. User-Centered Agile Methods. Yhdysvallat: Morgan & Claypool.
Center for Internet Security. 2020. Confidence in the Connected World About us. Tiedote. Luettu 20.7.2020. https://www.cisecurity.org/about-us/

Digi- ja väestötietovirasto. 2020. Digitaalisen turvallisuuden palvelut julkiselle sektorille. Tiedote. Päivitetty 9.4.2020. Luettu 21.7.2020. https://www.suomidigi.fi/ohjeet-ja-tuki/digitaalisen-turvallisuuden-palvelut-julkiselle-sektorille

Digi- ja väestötietovirasto. 2020. Sovelluskehityksen tietoturvaohje. Ohje. Päivitetty 9.6.2020. Luettu 22.7.2020. https://www.suomidigi.fi/ohjeet-ja-tuki/vahti-ohjeet/vahti-12013-sovelluskehityksen-tietoturvaohje

Digi- ja väestötietovirasto. 2020. Vahti ohjeet. Ohje. Päivitetty 6.7.2020. Luettu 21.7.2020. https://www.suomidigi.fi/ohjeet-ja-tuki/vahti-ohjeet

Eskola, J. & Suoranta, J. 1998. Johdatus laadulliseen tutkimukseen. 1. painos. Tampere: Osuuskunta Vastapaino.

GeeksforGeeks. 2020. What is information security. Artikkeli. Päivitetty 2.6.2020. Luettu 15.7.2020. https://www.geeksforgeeks.org/what-is-information-security/

Haikala, I. & Mikkonen, T. 2011. Ohjelmistotuotannon käytännöt. 12. painos. Hämeenlinna: Talentum Media Oy.

Hood, C., Fichtinger, S., Pautz, U. & Wiedemann, S. 2008. Requirements Management. The Interface Between Requirements Development and All Other Systems Engineering Processes. Saksa: Springer-Verlag AG.

Hundermark, P. 2014. Do better scrum. Version 3.3. 42agile – The agile coaching company.

Hyrynsalmi, S., Leppänen, V. & Rindell, K. 2015. A Comparison of Security Assurance Support of Agile Software Development Methods. International Conference on Computer Systems and Technologies – CompSysTech´15, 61-68.

Hyrynsalmi, S., Leppänen, V. & Rindell, K. 2016. Case Study of Security Development in an Agile Environment. Building Identity Management for a Government Agency. International Conference on Availability, Reliability and Security, 556-563.

Kananen, J. 2008. Kvalitatiivisen tutkimuksen teoria ja käytänteet. 1. painos. Jyväskylä: Jyväskylän kirjapaino Oy.

Lasse Koskela. n.d. Ketterän ohjelmistokehityksen julistus. Luettu 28.1.2020. https://agilemanifesto.org/iso/fi/manifesto.html

Lasse Koskela. n.d. Julistuksen takana olevat periaatteet. Luettu 28.1.2020. https://agilemanifesto.org/iso/fi/principles.html

Layton, M. & Ostermiller, S. 2017. Agile project management for dummies. 2. painos. Hoboken: John Wiley & Sons, Incorporated.

Lopez, O. 2011. Journal of Validation Technology. Yhdysvallat: Advanstar communications Inc.

Martin, J., Talabis, M. 2013. Information Security Risk Assesment Toolkit. Practical Assessments trough Data Collection and Data Analysis. Yhdysvallat: Elsevier Incorporated.

Measey, P., Wolf, L., Berridge, C., Gray, A., Levy., R., Oliver, L., Roberts, B., Short, M. & Wilmhurst, D. 2015. Agile Foundations. Principles, practices and frameworks. 1. painos. Yhdysvallat: BCS Learning & Development.

OWASP Foundation. 2020. About the OWASP Foundation. Tiedote. Luettu 31.7.2020. https://owasp.org/about/

OWASP Foundation. 2020. OWASP top ten. Listaus. Luettu 31.7.2020. https://owasp.org/www-project-top-ten/

Pressman, R. 2010. Software engineering. A practitioner´s approach. 7. painos. Yhdysvallat: McGraw-Hill, incorporated.

Rashina, H. 2019. Agile Processes in Software Engineering and Extreme Programming. Workshops. Sveitsi: Springer Nature Switzerland AG.

Scaled Agile Inc. 2020. Scaled agile framework. Prosessiohje. Luettu 25.3.2020. https://www.scaledagileframework.com/

Scaled Agile Inc. 2020. Scaled agile framework Portfolio Backlog. Prosessiohje. Päivitetty 31.1.2020. Luettu 26.3.2020. https://www.scaledagileframework.com/portfolio-backlog/

Shriram, A., Venkataramanan, N. 2017. Data privacy. principles and practice. Yhdysvallat: Taylor & Francis Group.

Shostack, A. 2014. Threat Modeling. Designing for Security. Yhdysvallat: John Wiley & Sons Inc.

Sommerville, I. 2006. Software Engineering. Eight Edition. Englanti: Pearson education limited.

Scrumguides.org. 2020. Official Scrum Guide. Prosessiohje. Julkaistu 1.11.2017. Ladattu 20.2.2020. https://www.scrumguides.org/docs/scrumguide/v2017/2017-Scrum-Guide-Finnish.pdf

Tietosuojavaltuutetun toimisto. n.d. Tietosuoja. Ohje. Luettu 16.7.2020. https://tietosuoja.fi/tietosuoja

Traficom. 2020. Tietoturva. Ohje. Päivitetty 9.7.2020. Luettu 15.7.2020. https://www.kyberturvallisuuskeskus.fi/fi/toimintamme/saantely-ja-valvonta/tietoturva

Valtiovarainministeriö. n.d. Voimassa olevat tietoturvaohjeet. Listaus. Luettu 20.7.2020. https://vm.fi/julkaisut/vahti

Visma Oy. 2019. Tietosuoja vai tietoturva. Blogi. Julkaistu 21.7.2019. Luettu 10.7.2020 https://www.visma.fi/blog/tietosuoja-tietoturva/

LIITTEET

Liite 1. Haastattelujen kysymykset

Yleiset kysymykset

1. Mikä on asemasi organisaatiossa?
2. Onko sinulla roolia tuotekehityksessä, jos niin mikä?
3. Onko sinulle selkeää, miten roolissasi tehdyt päätökset vaikuttavat tuotteeseen / tuotekehitysprosessiin?
4. Onko ketterän kehityksen menetelmät tuttuja? Jos ovat niin mitkä mallit erityisesti? Scrum, Kanban, SAFe, leSS
5. Onko ketterällä mallilla toimivan tuotekehityksen erilaiset roolit ja niiden vastuut tuttuja? Jos ovat niin mitä rooleja löytyy ja lyhyesti mitä vastuita eri rooleilla on?
6. Onko konsepti kokonaisketterästä prosessista tuttu?
7. Mitä eroa on tietoturvalla ja tietosuojalla?
8. Onko DPIA arviointi sinulle tuttu?

Vaatimusten hallinta kysymykset

9. Voitko luetella erilaisia tapoja ja mahdollisesti työkaluja vaatimusten hallintaan?
10. Miten mielestäsi vaatimusten hallinta olisi ideaalia toteuttaa?
11. Miten sidosryhmiltä tulleet muutokset olemassa olevissa vaatimuksissa tulisi ottaa huomioon kehityksessä olevan ominaisuuden kehittämisessä?
12. Miten sidosryhmiltä tulleet uudet vaatimukset tulisi ottaa huomioon kehityksessä olevan ominaisuuden kehittämisessä?
13. Missä vaiheessa uusia tai muuttuneita vaatimuksia ei enää tulisi ottaa huomioon?

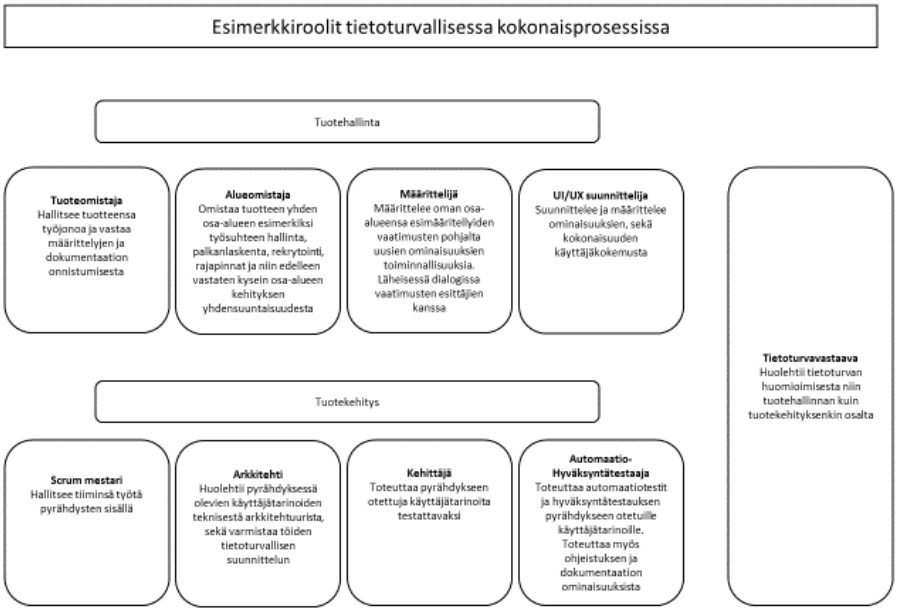
Tietoturvakysymykset

14. Onko tietoturvallisen kehityksen viitteelliset vaatimukset tai ohjeistukset tuttuja?
    a. OWASP ASVS
    b. VAHTI 1/2013
    c. CIS Benchmarks
    d. A29WP WP248
15. Mitä mielestäsi tarkoittaa tietoturvallisen ohjelmistokehityksen vaatimusten huomioiminen tuotekehitysprosessissa?
16. Onko sinulle selkeää mitä vastuita tietoturvallisessa ohjelmistokehityksessä sinulla on?
17. Miten organisaatio kokonaisuudessaan mielestäsi huomioi tietoturvan?
18. Miten organisaatio mielestäsi huomioi tietoturvan ohjelmistokehityksessä?
19. Mitä rooleja mielestäsi tarvitaan tietoturvallisessa ohjelmistokehityksessä? Voitko luetella mitä vastuita näillä rooleilla olisi?
20. Onko mielestäsi tekniset ympäristömme tietoturvallista ohjelmistokehitystä tukevia?
21. Onko mielestäsi käytettävät ohjelmistomme tietoturvallista ohjelmistokehitystä tukevia?
22. Mitä toimia mielestäsi pitäisi toteuttaa, jos tavoite on pystyä parantamaan tietoturvan tasoa niin ohjelmistokehityksessä kuin muutenkin organisaatiossa?
23. Haluaisitko olla mukana tietoturvallisen ohjelmistokehityksen prosessin luomisessa?
24. Haluaisitko olla mukana tietoturvallisen ohjelmistokehityksen prosessin kehittämisessä?

Liite 2. Kehittämistehtävä: Tietoturvan huomioiva kokonaisprosessi

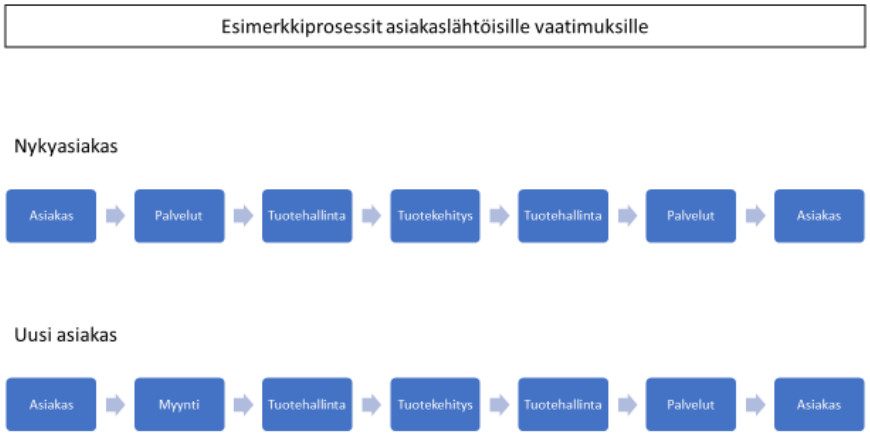
Opinnäytetyön ohessa tehtiin organisaatiolle kehittämistehtävä, jonka tarkoituksena oli luoda vaatimuksesta toimivaksi ominaisuudeksi luotu kokonaisprosessi ja siihen esimerkkejä. Tämän työn tietoturvanäkökulman myötä kehittämistehtävän prosesseissa on kuvattu niin rooleja, tietovirtoja kuin tietoturvan ja tietosuojan linkittymistä kokonaisprosessiin.

Tietoturvan kannalta tärkeimmät roolit tuotehallinnan ja tuotekehityksen läpikäyvässä kokonaisprosessissa sisältävät ketterän kehityksen rooleja tuotekehityksen puolella ja erilaisia asiantuntijarooleja tuotehallinnan puolella. Tietoturvavastaava voi toimia osana molempien prosesseja. Kuviossa 22 on tärkeimmät roolit kuvattuna.



KUVIO 22. Tietoturvallisen kokonaisprosessin roolit

Suurin osa ohjelmistoon muodostuvista ominaisuuksista on lähtöisin asiakkaiden ja sitä mukaan ohjelman käyttäjien vaatimuksista. Kuvio 23 kuvaa esimerkinomaisesti ylätasolta prosessia.

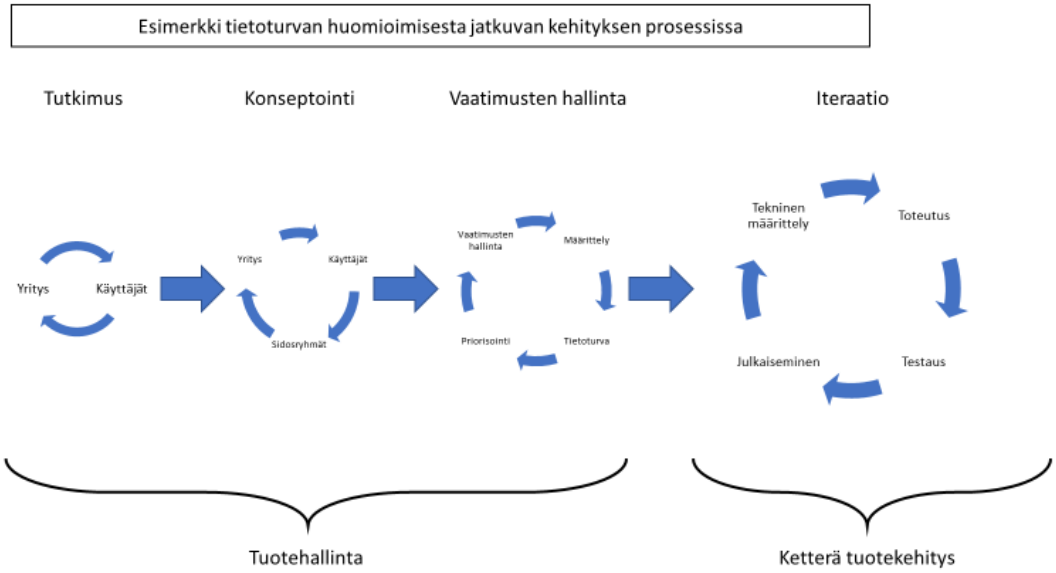


KUVIO 23. Asiakaslähtöiset vaatimukset

Iso osa ohjelmistoon kohdistuvista vaatimuksista tulee viranomaisvaatimuksista ja myös oman suunnittelun ja tutkimustyön puolesta. Kuviossa 24 on kuvattu tähän prosessiin osallistuvat tahot.

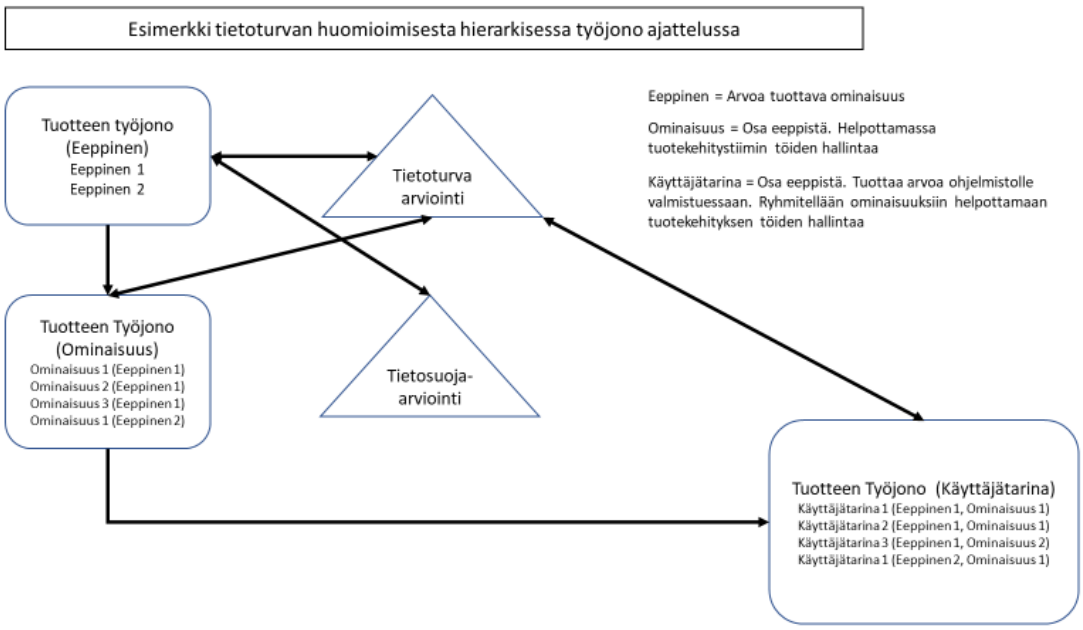
KUVIO 24. Oman tutkimustyön vaatimusten prosessit

Tuotteen kehitys on aina jatkuva prosessi, joka ei lopulta pääty ennen ohjelmiston poistumista. Prosessille tärkeää on jatkuva liike ja parannus, johon tarvitaan vähintään kahta eri yksikköä organisaatiosta. Kuvatun kaltaista prosessia on avattu kuviossa 25.



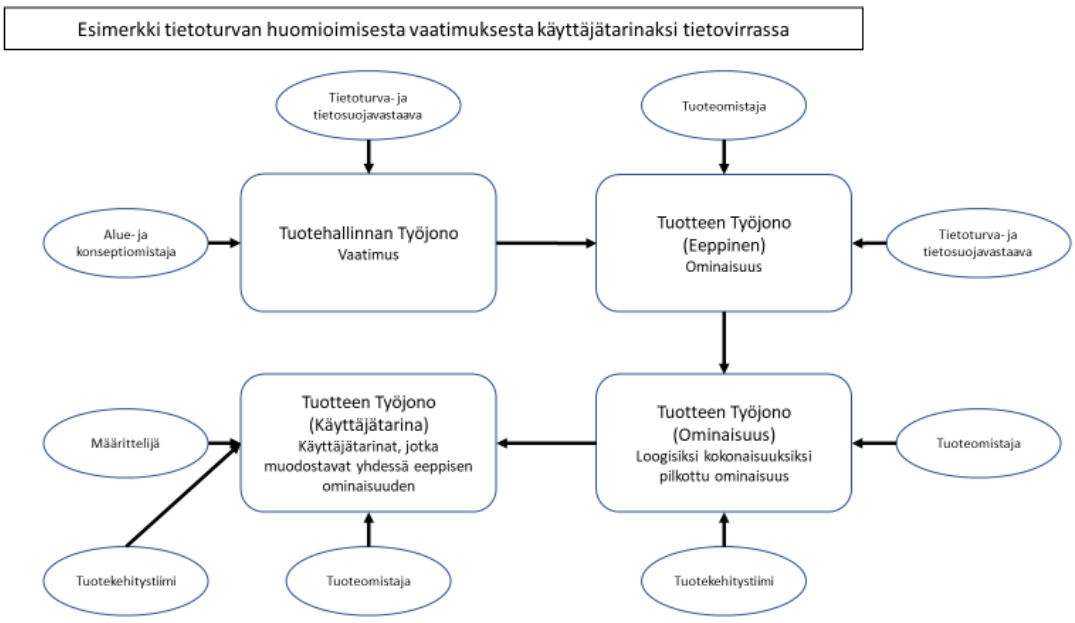
KUVIO 25. Tietoturvan huomioiminen jatkuvan kehityksen prosessissa

Toimiva tietoturva-ajattelu vaatii tuotteen vaatimuksista jalostettujen ominaisuuksien hallintaa hierarkkisessa tiketöintijärjestelmässä. Tätä hierarkiaa ja sitä miten tietoturvavaatimukset tulisi ottaa huomioon järjestelmässä on kuvattu kuviossa 26.



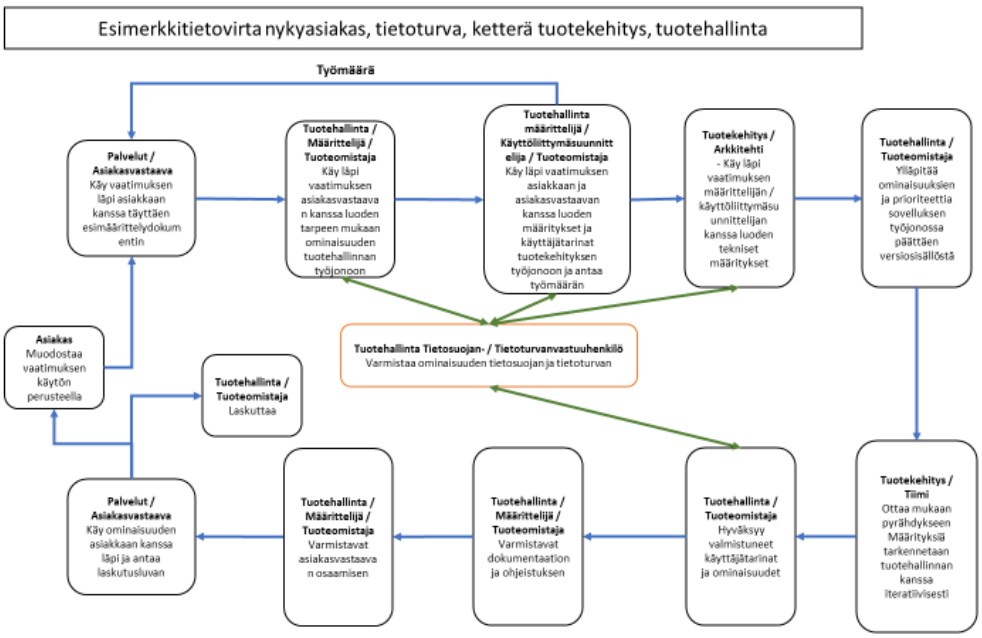
KUVIO 26. Hierarkinen vaatimusten hallinta

Tietoturva- ja tietosuoja vaatii toimiakseen kokonaisvaltaisen prosessin tarkastelun varsinkin niiden roolien näkökulmasta, jotka ovat ydinrooleja prosessissa. Kuviossa 27 on näytetty eri roolien merkitys tietovirralle.



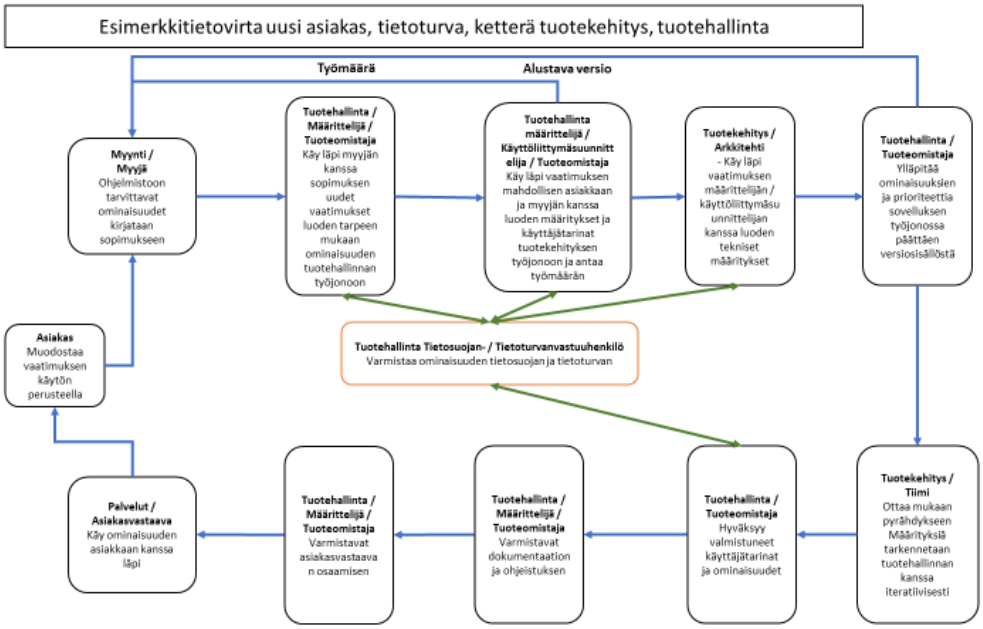
KUVIO 27. Tietoturva kokonaistietovirrassa

Tietovirta nykyasiakkaiden vaatimuksesta heillä käytössä olevaksi toimivaksi ominaisuudeksi vaatii monen osaston ja niihin kuuluvien roolin yhteistyötä. Tätä tietovirtaa on kuvattu kuviossa 28.



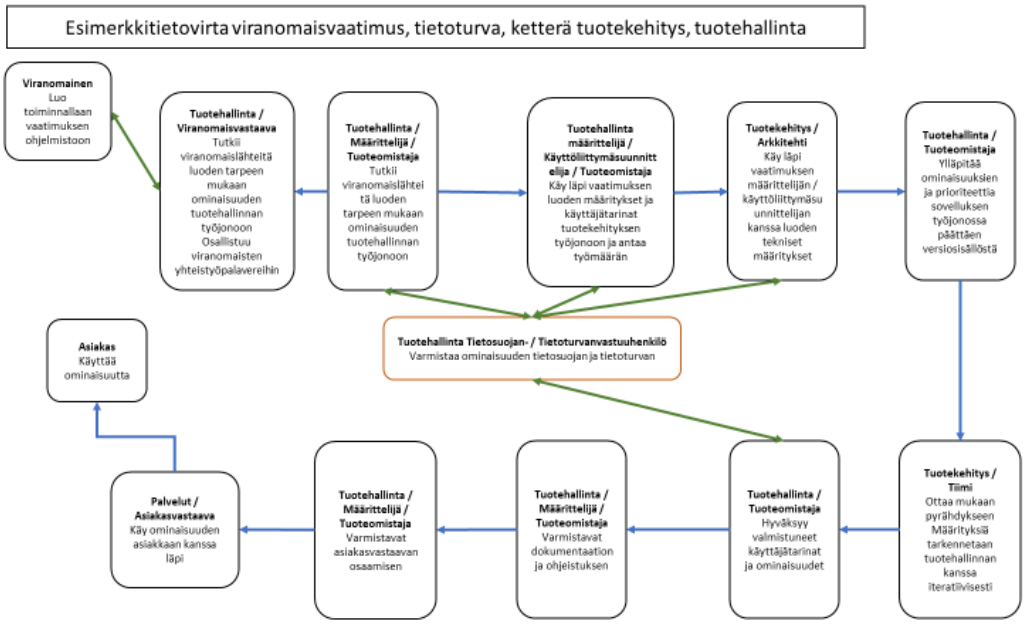
KUVIO 28. Nykyasiakkaan vaatimus toimivaksi ominaisuudeksi

Uuden asiakkaan vaatimuksesta toimivaksi ominaisuudeksi vaatimukset eroavat hieman nykyasiakkaista, johtuen mahdollisuudesta kirjauttaa asioita sopimuksiin. Tämän prosessin tietovirtaa on kuvattu kuviossa 29.



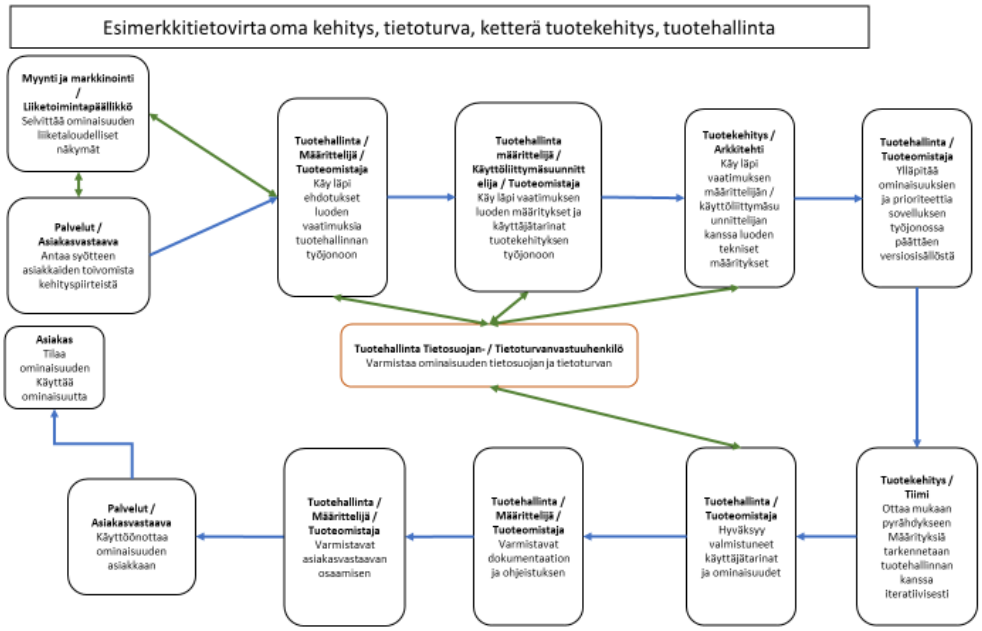
KUVIO 29. Uuden asiakkaan vaatimus toimivaksi ominaisuudeksi

Kaikessa ohjelmistokehityksessä viranomaiset luovat toiminnallaan erilaisia vaatimuksia ohjelmistoon. Tämä korostuu toimittaessa palkka- ja henkilöstöhallinnon alalla. Kuviossa 30 on pyritty kuvaamaan tämän prosessin tietovirtaa.



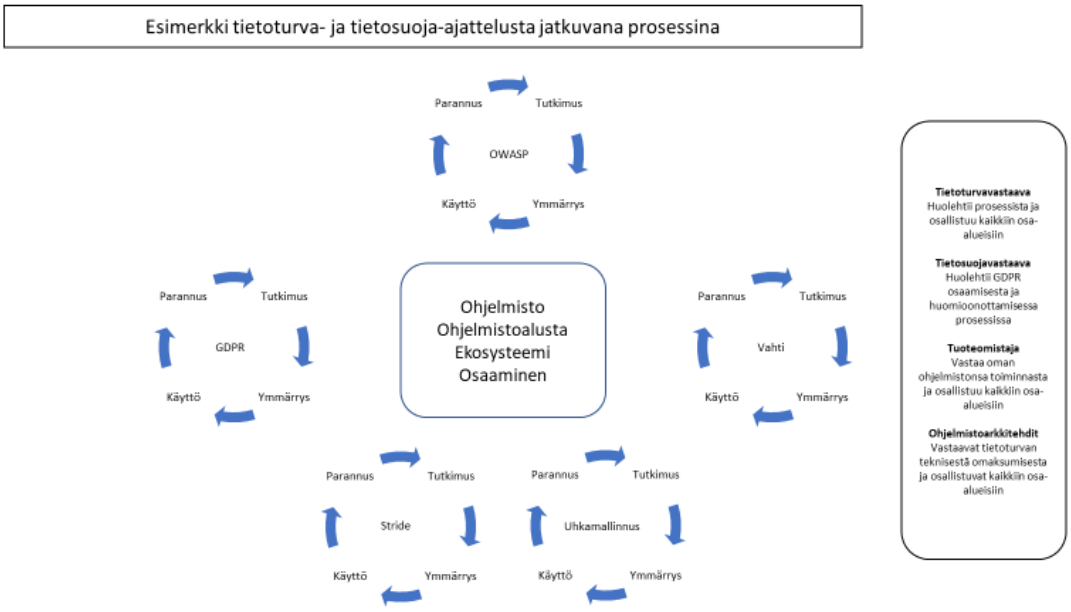
KUVIO 30. Viranomaisvaatimus toimivaksi ominaisuudeksi

Iso osa nykyaikaisen ohjelmiston kehityksestä on omaan tutkimukseen ja suunnitteluun pohjautuvaa. Kuviossa 31 on kuvattuna tietovirta itse luotujen vaatimusten jalostusta ominaisuudeksi.



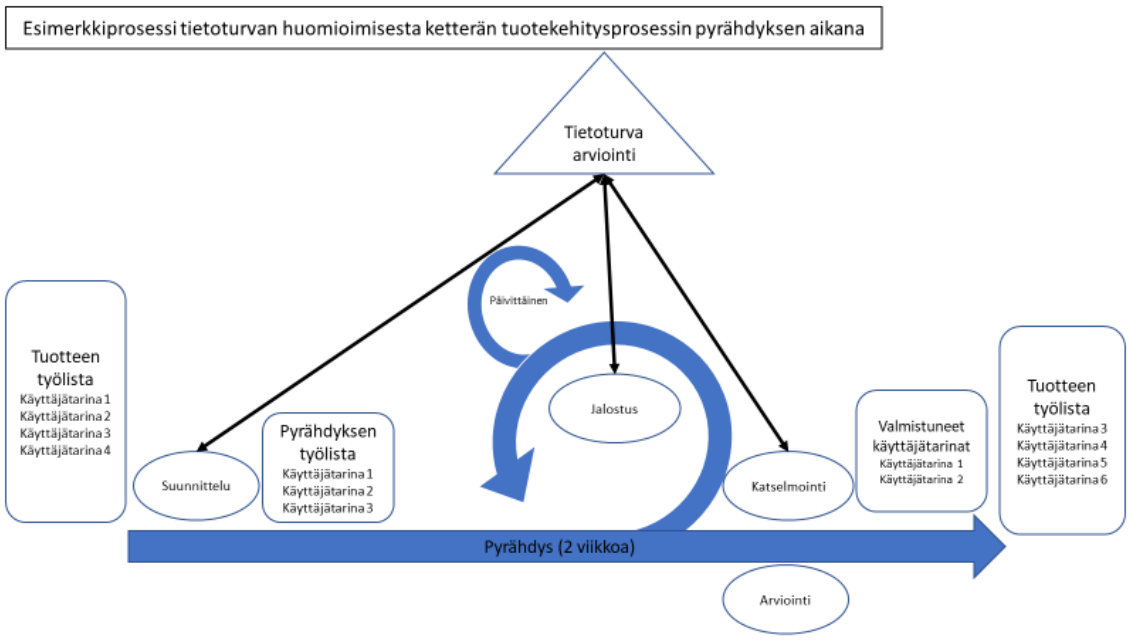
KUVIO 31. Omasta vaatimuksesta toimivaksi ominaisuudeksi

Tietoturva ja tietosuoja eivät kumpikaan ole kertaluonteisia prosesseja vaan jatkuvaa opiskelua ja parannusta. Kuviossa 32 on pyritty tuomaan esiin tätä ajattelua ja siinä mukana olevia rooleja.



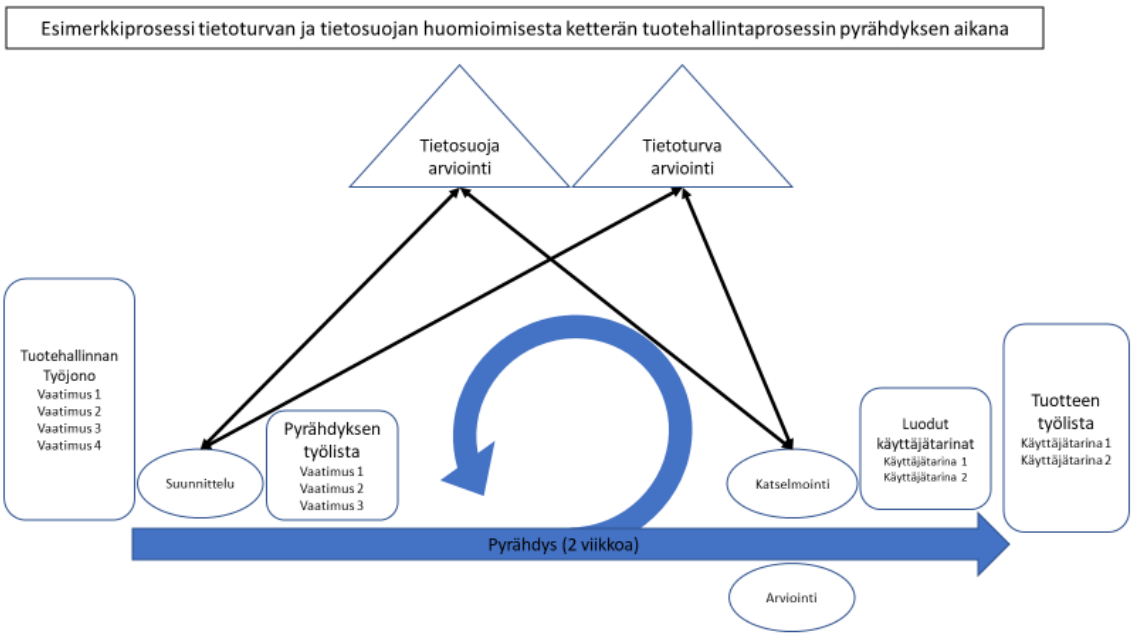
KUVIO 32. Tietoturva- ja tietosuoja-ajattelu

Ketterän kehityksen mukaisessa mallissa tietoturva-arviointi tarvitsee tuoda moneen eri kohtaan prosessia. Kuviossa 33 on kuvattuna, kuinka Scrum pyrähdyksessä voidaan ottaa huomioon tietoturva.



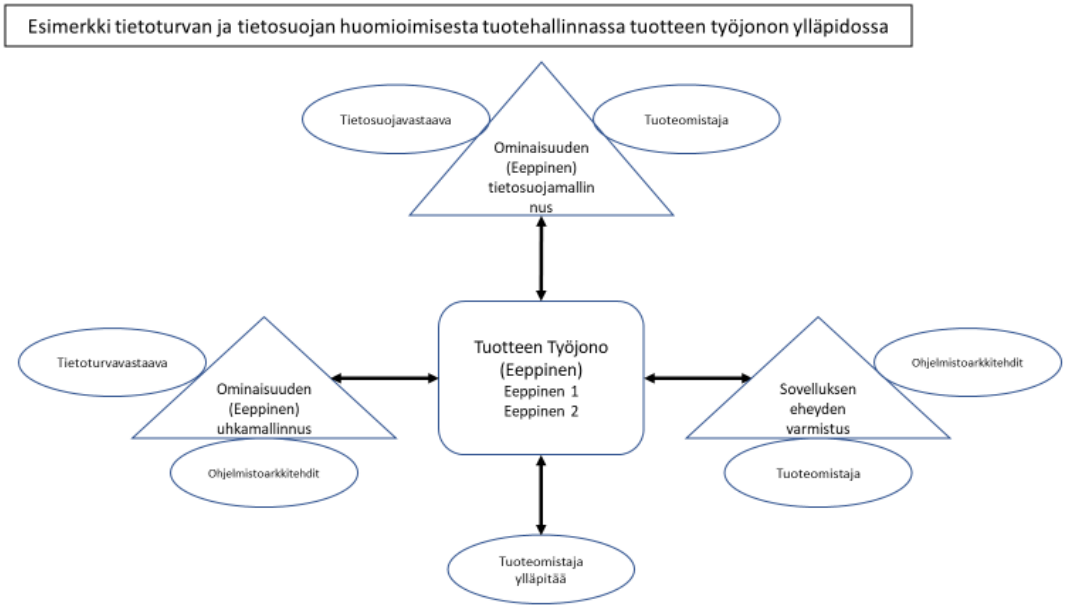
KUVIO 33. Esimerkki tietoturvan huomioimisesta tuotekehityksessä

Tuotekehityksen toimiessa ketterässä menetelmässä on myös sen ympärillä olevien tahojen järkevä toimia kyseisen mallin lainalaisuuksia noudattaen. Kuviossa 34 on mallinnettu kuinka tietoturva ja tietosuoja voidaan ottaa huomioon Scrum pyrähdyksen kaltaisessa prosessissa tuotehallinnan yksikössä.



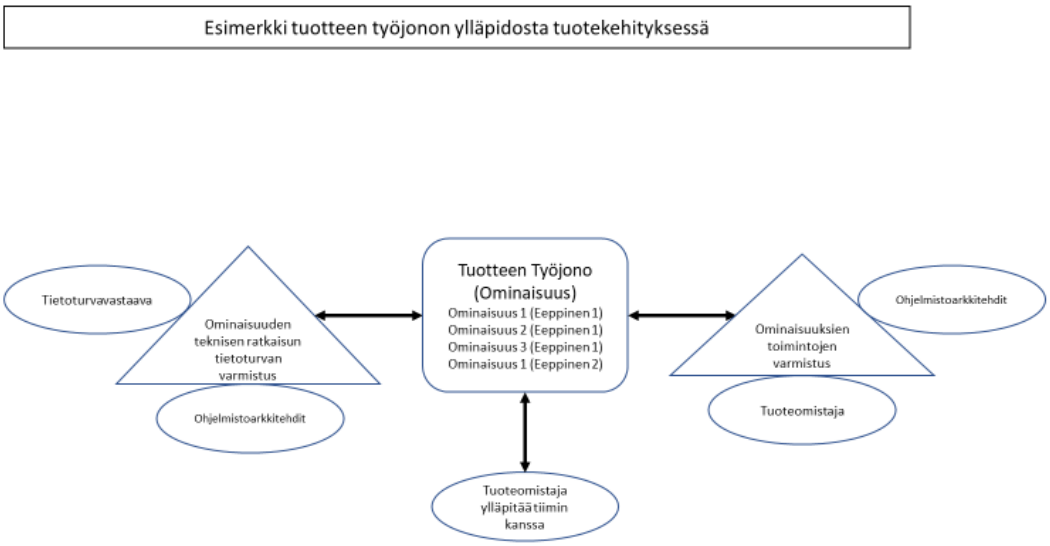
KUVIO 34. Esimerkki tietoturvan ja tietosuojan huomioimisesta tuotehallinnassa

Erittäin tärkeä osa niin tuotteen yleistä kehitystä, kuin tietoturvan ylläpitoa on tietoturvan ja tietosuojan huomioonottaminen tuotteen työjonon ylläpidossa. Kuviossa 35 on selkeytetty miten ja millä rooleilla tietoturvaa ja tietosuojaa voidaan huomioida tuotehallinnassa.



KUVIO 35. Esimerkki työjonon ylläpidosta tuotehallinnassa

Tuotteen työjono tulee ylläpitää myös tuotekehityksen henkilöiden kanssa, koska heillä on eniten tietoa tietoturvan teknisestä toteuttamisesta. Kuviossa 36 on esimerkinomaisesti kuvattu, kuinka eeppinen jaetaan eri ominaisuuksiksi ja kuinka kaikkien osalta tulee arvioida tietoturvaa.

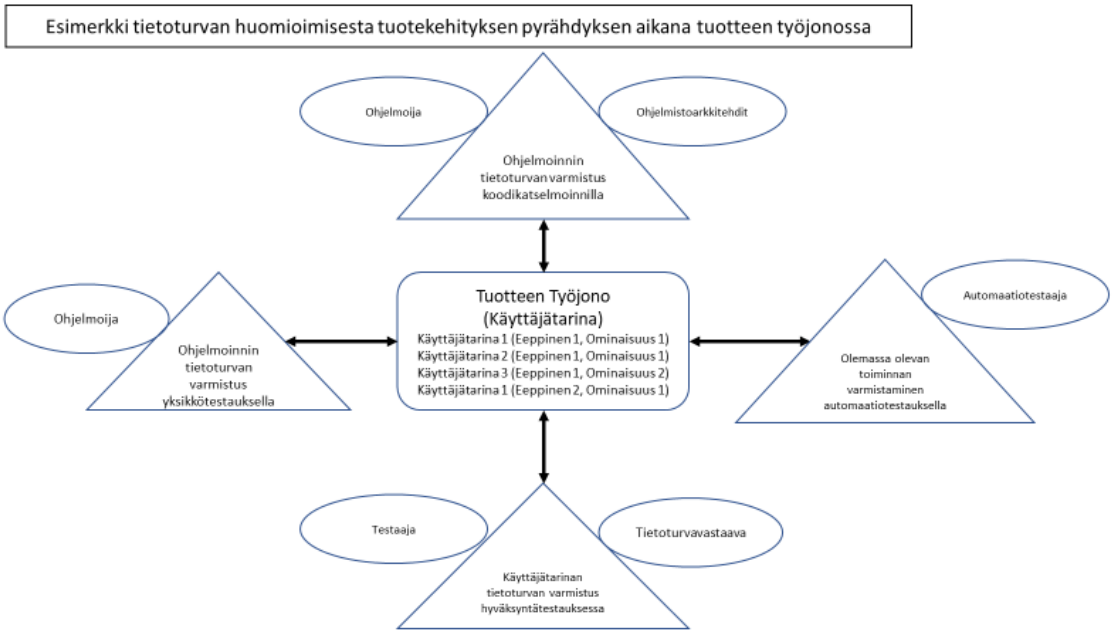


KUVIO 36. Esimerkki työjonon ylläpidosta tuotekehityksessä

Lopulta kaikki tuotteen kehityksessä nivoutuu kiinni toisiinsa käyttäjätarinoiden myötä. Kuviossa 37 on kuvattuna ne roolit ja toiminnot, joita tietoturvallinen ohjelmistokehitys vaatii tuotehallinnan osalta.

KUVIO 37. Esimerkki käyttäjätarinoiden ylläpidosta tuotehallinnassa

Tuotekehitykselle käyttäjätarinat kertovat sen, mihin ohjelmistoa kehitetään. Tuotekehityksessä myös tehdään lopulta se valinta, tehdäänkö ominaisuudesta teknisesti tietoturvallinen vain ei. Kuviossa 38 on kuvattuna ne roolit ja toiminnot, joita tietoturvan huomioonottaminen vaatii.



KUVIO 38. Esimerkki käyttäjätarinoiden ylläpidosta tuotekehityksessä