Ketterä laitteistokehitys

Turun yliopisto
Tietotekniikan laitos
TkK-tutkielma
Tietotekniikka
Toukokuu 2026
Peter Virta

Turun yliopiston laatujärjestelmän mukaisesti tämän julkaisun alkuperäisyys on tarkastettu Turnitin OriginalityCheck -järjestelmällä.

TURUN YLIOPISTO
Tietotekniikan laitos

Peter Virta: Ketterä laitteistokehitys

TkK-tutkielma, 21 s.
Tietotekniikka
Toukokuu 2026

Tutkielmassa tarkastellaan ketterää laitteistokehitystä sekä siihen liittyviä hyötyjä ja haasteita modernissa puolijohdeteollisuudessa. Työn tarkoituksena on selvittää, miten ohjelmistokehityksestä tuttuja iteratiivisia menetelmiä voidaan soveltaa fyysisen laitteiston suunnitteluun. Taustana ovat globaalit markkinapaineet ja muuttuvat arkkitehtuurivaatimukset. Tutkimus toteutettiin kirjallisuuskatsauksena. Aineisto koostuu 20 kansainvälisestä vertaisarvioidusta artikkelista vuosilta 2020–2025. Haku kohdistettiin tietokantoihin ACM Digital Library, IEEE Xplore ja ScienceDirect. Keskeisiä lähdeteoksia ovat muun muassa Rautakoura ja Hämäläinen (2023) sekä Berg ym. (2020).

Tulokset osoittavat, että ketterien prosessien keskeisimmät hyödyt liittyvät arkkitehtuurin joustavuuteen, korkeaan modulaarisuuteen, kehityksen nopeuteen sekä resurssien tehokkaaseen käyttöön. Puhdas ohjelmistokehityksen malli ei sellaisenaan sovellu laitteistoon. Käytännössä toimiviksi osoittautuvat hybridimallit, jotka yhdistävät ketterän ajattelutavan fyysisen tuotannon rajoitteisiin. Suurimmiksi haasteiksi tunnistettiin suunnittelun monimutkaisuus, kypsien validointityökalujen puute sekä alan käsitteellinen epäselvyys. Ketterälle laitteistokehitykselle ei ole toistaiseksi muodostunut vakiintunutta määritelmää.

Päätelmänä todetaan, että ketteryys laitteistopuolella vaatii ketterän ajattelutavan sovittamista fyysisen maailman rajoitteisiin, ei koodauskäytäntöjen suoraviivaista kopiointia. Jatkotoimenpiteenä suositellaan standardoitujen ketteryysmittareiden kehittämistä sekä suomalaisen kvanttilaskenta- ja tietoliikenneteollisuuden ketterien käytäntöjen empiiristä dokumentointia.

Asiasanat: ketterä laitteistokehitys, laitteistosuunnittelu, kirjallisuuskatsaus, laitteiston ja ohjelmiston yhteissuunnittelu

Sisällys

Kuvat

Taulukot

1 Johdanto

2000-luvun alkuvuosikymmeninä globaali laitteistokehitys on kokenut valtavan muutoksen. Perinteiset laitteistokehityksen ”vesiputousmallit” (joille on ominaista lineaariset, monivuotiset suunnittelusyklit, suuret etukäteisinvestoinnit ja jäykät toimitusketjut) sopivat huonosti yhteen modernin teknologisen kysynnän nopean tahdin kanssa [1], [2]. Elämme “Mooren lain” jälkeistä aikakautta.[1] Historiallinen suuntaus, jossa transistoritiheys yli kaksinkertaistuu kahden vuoden välein, on hidastunut. Tämä on siirtänyt alan painopisteen yleiskäyttöisistä suorittimista pitkälle erikoistuneisiin arkkitehtuureihin, kuten tekoälykiihdyttimiin (AI), kvanttisuorittimiin (QPU) sekä kehittyneeseen tietoliikenneinfrastruktuuriin [4]. Tämä muutos yhdessä ennennäkemättömien geopoliittisten ja taloudellisten paineiden kanssa on synnyttänyt globaalin liikkeen ketterien (agile) menetelmien integroimiseksi fyysiseen laitteistokehitykseen [5], [6].

Ymmärtääksemme tämän siirtymän kiireellisyyttä on tarkasteltava myös laajempaa geopoliittista yhteyttä. Viime vuosina maailmanlaajuisten puolijohdetoimitusketjujen hauraus on paljastunut karulla tavalla.[2] 2020-luvun alun vakava sirupula

[1]Mooren laki viittaa Gordon Mooren (Fairchild Semiconductorin ja Intelin perustajajäsen) vuonna 1965 *Electronics*-lehdessä tekemään empiiriseen havaintoon. Hän ennusti, että mikrosirun transistorien määrä kaksinkertaistuisi noin kahden vuoden välein, mikä toimi teknologia-alan räjähdysmäisen kasvun perustana yli puolen vuosisadan ajan [3].

[2]Vaikka 2020-luvun alun vakava maailmanlaajuinen sirupula johtui pääasiassa COVID-19-pandemian sulkutoimista sekä kysynnän nopeista muutoksista, toimitusketju on historiallisesti ollut erittäin haavoittuvainen luonnonkatastrofeille. Merkittäviä esimerkkejä ovat Japanin maanjäristys sekä Thaimaan tulvat vuonna 2011, Texasin poikkeukselliset talvipakkaset vuonna 2021 sekä Taiwanin ankarat kuivuudet [7], [8].

sai aikaan massiivisia valtioiden kehitysprojekteja ympäri maailmaa. Yhdysvaltain CHIPS and Science Act -laki ohjaa miljardeja dollareita kotimaiseen tuotantoon ja tutkimukseen tasapainottaakseen aasialaisten tehtaiden (foundry), kuten TSMC:n ja Samsungin, valta-asemaa [9]. Pääoman ohjaaminen tuotantoon on kuitenkin vain puoli totuutta, sillä nopeus, jolla uusia laitteistosuunnitelmia voidaan iteroida ja tuoda markkinoille, on yhtä kriittistä. Piilaakson teknologiajätit (Google, Amazon, Microsoft, Nvidia) ovat yhä enenevässä määrin siirtyneet kehittämään räätälöityä, omaa piiteknologiaa. Ne hyödyntävät ketterää laitteiston ja ohjelmiston yhteissuunnittelua optimoidakseen datakeskuksensa tekoälytyökuormia[3] varten tahdilla, johon perinteisemmät laitteistovalmistajat eivät pysty vastaamaan [11].

Euroopassa keskustelu keskittyy voimakkaasti teknologiseen riippumattomuuteen. Euroopan unionin sirusäädöksen (Chips Act) tavoitteena on kaksinkertaistaa Euroopan osuus globaaleista puolijohdemarkkinoista 20 prosenttiin vuoteen 2030 mennessä [12]. Euroopalta puuttuu kuitenkin Itä-Aasian kaltainen massiivinen ja yhtenäinen tehdasekosysteemi[4]. Kilpaillakseen Euroopan innovaatiot tukeutuvat voimakkaasti ketteryyteen, avoimen lähdekoodin laitteistoihin (kuten RISC-V-käskykanta-arkkitehtuuriin) ja tehtaattomiin (fabless) suunnitteluekosysteemeihin.

Pohjois-Eurooppa ja erityisesti Suomi toimivat erinomaisena koealustana tälle ketterälle tulevaisuudelle. Suomella on rikas laitteistosuunnittelun perintö, joka juontaa juurensa Nokian valta-aseman kulta-ajoista. Nykyään Suomi on globaali johtaja nousevissa laitteistoteknologioissa, erityisesti 5G/6G-infrastruktuurissa se-

[3]Suuret pilvipalvelutarjoajat (ns. hyperscalerit) siirtyvät nopeasti kohti vertikaalista integraatiota optimoidakseen suorituskykyä ja vähentääkseen riippuvuuttaan kaupallisista piireistä. Tunnettuja esimerkkejä näistä omista tekoälykiihdyttimistä ovat Googlen Tensor Processing Unit (TPU), NVIDIA:n Blackwell, Amazon (AWS) Trainium- ja Inferentia-sirut sekä Microsoftin Azure Maia [10].

[4]Huippuluokan siruvalmistuksen (esim. 7 nm:n ja 5 nm:n solmut) keskittyminen Aasiaan, pääasiassa Taiwanin TSMC:lle ja Etelä-Korean Samsungille, johtuu tehtaiden valtavista pääomakustannuksista (usein yli 5 miljardia dollaria tehdasta kohden) sekä massiivisten mittakaavaetujen tarpeesta. Eurooppa on historiallisesti keskittynyt erikoistuneisiin ja autoteollisuuden siruihin. Tämän vuoksi alueelta puuttuvat edistyneitä logiikkasiruja suunnittelevat tehtaattomat (fabless) yritykset, mikä taas on estänyt uudenlaisten huipputehtaiden vaatiman paikallisen kysynnän syntymisen [13].

kä kvanttilaskennassa. Esimerkiksi suomalaiset kvanttilaskennan startup-yritykset, kuten IQM, ovat saavuttaneet kansainvälistä huomiota rakentamalla suprajohtavia kvanttitietokoneita[5].[14], [15] Kvanttitietokoneen kehittäminen on äärimmäisen vaativaa laitteistosuunnittelua, joka toimii lähellä absoluuttista nollapistettä ja vaatii räätälöityä ohjauselektroniikkaa. Suomen hallituksen ja teollisuuden raportit korostavat, kuinka nämä startupit eivät voi tukeutua hitaisiin, perinteisiin vesiputousmalleihin [16]. Sen sijaan ne hyödyntävät tehtävävetoista, ketterää laitteistokehitystä. Ne tuottavat nopeasti prototyyppejä ohjauslaitteistoista ja käyttävät iteratiivista yhteissuunnittelua kvanttialgoritmien kanssa. Lisäksi ne hyödyntävät eurooppalaisia toimitusverkostoja, jotta suuntaa voidaan muuttaa nopeasti uusimman kokeellisen fysiikan datan perusteella. Vastaavasti Suomen televiestintäsektori hyödyntää uudelleenkonfiguroitavia laitteistoja (FPGA) ottaakseen käyttöön ketteriä reunalaskentasolmuja, joita voidaan päivittää langattomasti tukemaan kehittyviä 6G-standardeja [17].

Ympäri maailmaa vallitsee selkeä yhteisymmärrys siitä, että laitteistoteollisuuden on muututtava enemmän ohjelmistoteollisuuden kaltaiseksi[6]. Sirukehityksen valtavat kustannukset tarkoittavat, että epäonnistuminen ei ole vaihtoehto, mutta nopeasti liikkuvat tekoäly- ja kvanttimarkkinat vaativat jatkuvaa iterointia. Ketterä laitteistokehitys lupaa kuroa tämän umpeen hyödyntämällä generatiivisia laitteistokuvauskieliä (HDL), korkean tason synteesiä, avoimen lähdekoodin chiplet-komponentteja ja uudelleenkonfiguroitavaa piitä. Sen tavoitteena on muuttaa laitteistosuunnittelua ja madaltaa kynnystä niin, että pieni helsinkiläinen startup tai

[5]IQM on kasvanut nopeasti yhdeksi Euroopan merkittävimmistä kvanttiteknologia-alan toimijoista. Yritys on muun muassa vastannut Suomen ensimmäisten kvanttitietokoneiden (mukaan lukien 20- ja 54-kubitin järjestelmät) toimittamisesta VTT:lle ja avannut Espoon Otaniemeen oman teollisen tason kvanttisirutehtaan, mikä vähentää merkittävästi riippuvuutta globaaleista toimitusketjuista [14], [15].

[6]Tämä näkemys on ollut keskiössä muun muassa Yhdysvaltain puolustusministeriön tutkimusorganisaatio DARPA:n yli 1,5 miljardin dollarin *Electronics Resurgence Initiative* (ERI) -ohjelmassa. ERI-ohjelman keskeisenä tavoitteena on ollut tuoda avoimen lähdekoodin ohjelmistokehityksen ketteryys, automaatio ja yhteistyöedut laitteistojen ekosysteemiin, jotta monimutkaisten sirujen suunnitteluaika ja kustannukset saataisiin laskettua ohjelmistokehitystä vastaavalle tasolle [18].

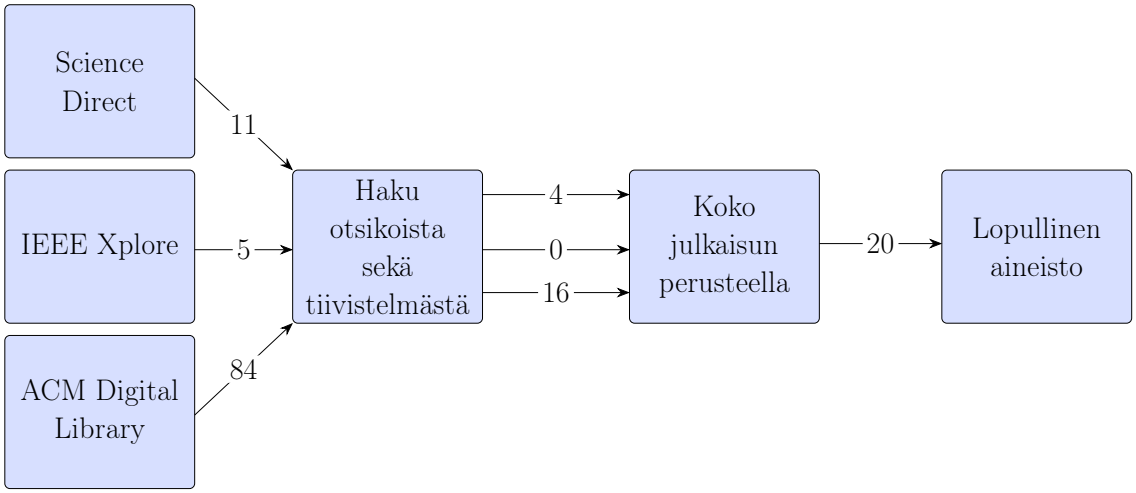
münchenläinen yliopistolaboratorio voi valmistaa prototyyppejä ja tuottaa monimutkaista piitä sellaisella nopeudella ja joustavuudella, joka on historiallisesti ollut varattu vain ohjelmistosovelluksille.[18]

Tässä tutkielmassa ketterällä laitteistokehityksellä tarkoitetaan lähestymistapaa, jossa fyysisen laitteiston suunnittelu omaksuu ohjelmistokehityksestä tuttuja työtapoja, kuten lyhyitä iteraatiosyklejä, nopeaa prototypointia, jatkuvaa palautetta sekä laitteiston ja ohjelmiston yhteissuunnittelua. Tästä taustasta nousevat tutkielman tutkimuskysymykset:

TK1: Mitä hyötyjä ketterällä laitteistokehityksellä on

TK2: Mitä haasteita ketterään laitteistokehitykseen liittyy

Tutkielma toteutettiin kirjallisuuskatsauksena. Haku tehtiin tietokannoissa ACM Digital Library, IEEE Xplore ja ScienceDirect, jotka valikoituivat ja artikkelit avautuivat yliopistolisenssin perusteella. Hakulausekkeen otsikon tuli sisältää sana `''hardware''` ja tiivistelmän sana `''agile''`. Haku tehtiin englanniksi ja suodatettiin kuvan 1.1 mukaisesti. Ensimmäinen haku tuotti yhteensä 100 artikkelia, joiden otsikot ja tiivistelmät käytiin läpi. Tämän jälkeen aineisto rajattiin 20 uusimpaan artikkeliin (2020–2025), jotta katsaus keskittyy alan tuoreimpaan kehitykseen. Näistä koostettiin tutkimuskysymyksiin 16 aliteemaa, joista kahdeksan on hyötyihin liittyvää (*flexibility*, *efficiency*, *modularity*, *cost-reduction*, *speed*, *reusability*, *feedback integration* ja *patchability*) sekä kahdeksan haasteisiin liittyvää (*complexity*, *quality-risk*, *adaptation friction*, *ambiguity*, *tooling deficits*, *bug cost*, *scalability* ja *vendor constraints*). Aliteemat ovat englanniksi, koska se on myös alkuperäismateriaalin kieli. Aliteemojen perusteella koostettiin taulukko 3.1.



Kuva 1.1: Aineistojen hakuprosessi

Tutkielma rakentuu seuraavasti: luvussa kaksi on kuvataan teoreettista taustaa ketteryydestä, laitteistokehityksen elinkaarta ja menetelmien vertailua. Luku kolme on työ tulokset, joka kertoo ketterän laitteistokehityksen hyödyt ja haasteet. Ja luku neljä on pohdintaa. Luku viisi esittää työn yhteenvedon.

2 Teoreettinen tausta

Ketterän laitteistokehityksen analyysi edellyttää, että lukijalla on selkeä käsitys kahdesta lähtökohtaisesti erilaisesta maailmasta, jotka ovat ohjelmistokehityksestä peräisin olevat ketterät menetelmät sekä fyysisen laitteiston suunnittelun elinkaari. Tässä luvussa avataan molemmat lyhyesti, jotta niiden välistä kuilua voidaan tarkastella myöhemmissä luvuissa. Juuri tämä kuilu on koko tutkimusaiheen ydin, sillä ketterä ajattelu syntyi maailmassa, jossa muutoksen kustannus on lähellä nollaa, kun taas laitteistossa jokainen muutos maksaa aikaa, materiaalia ja rahaa.

2.1 Mitä ketteryys on?

Ketterä ohjelmistokehitys sai vakiintuneen muotonsa vuoden 2001 Agile-manifestissa[1], jossa joukko ohjelmistokehittäjiä kiteytti vaihtoehdon raskaille, suunnitelmavetoisille prosesseille. Manifestin ydin on neljä arvoa, jotka painottavat yksilöitä ja vuorovaikutusta prosessien sijaan, toimivaa tuotetta kattavan dokumentaation sijaan, asiakasyhteistyötä sopimusneuvottelujen sijaan sekä muutokseen vastaamista suunnitelman noudattamisen sijaan. Käytännössä nämä arvot tarkoittavat lyhyitä, toistuvia kehityssyklejä, joissa tuotetta kehitetään vähän kerrallaan ja suuntaa korjataan jatkuvan palautteen perusteella.[6]

Tunnetuin ketterä viitekehys on Scrum, jonka peruskäsitteistö toistuu myös laitteistokehitystä käsittelevässä aineistossa. Työ jaetaan sprintteihin eli muutaman vii-

[1] https://agilemanifesto.org/

kon mittaisiin jaksoihin, joiden päätteeksi syntyy inkrementti, eli toimiva ja testattava lisäys tuotteeseen. Kehitystä tekee cross-functional-tiimi, jonka jäsenillä on yhdessä kaikki tuotteen valmistamiseen tarvittava osaaminen, jolloin työ ei pysähdy odottamaan toisen osaston panosta. Olennaista ei ole mikään yksittäinen seremonia vaan ajattelutapa, johon kuuluvat oppiminen tekemällä, jatkuva testaaminen ja siilojen purkaminen tiimien väliltä.[1]

2.2 Laitteistokehityksen elinkaari

Laitteistokehitys eroaa ohjelmistokehityksestä jo perusrakenteeltaan. Sen elinkaari etenee perinteisesti vaiheittain, ja jokainen vaihe sitoo edellisen tuloksia tavalla, jota on vaikea peruuttaa. Kehitys alkaa arkkitehtuurisuunnittelusta, jossa määritellään järjestelmän rakenne ja keskeiset valinnat. Tämän jälkeen rakennetaan prototyyppejä, joiden avulla suunnitelmaa kokeillaan käytännössä. Elektroniikan, mekaniikan ja sulautettujen järjestelmien suunnittelu etenee usein rinnakkain, sillä fyysinen tuote koostuu monesta toisistaan riippuvasta osasta.[6]

Prototyyppejä seuraa testaus, joka laitteistossa on huomattavasti kalliimpaa ja hitaampaa kuin ohjelmistossa, sillä vika voi edellyttää piirin uudelleenvalmistusta. Tämä työvaihe maksaa viikkoja ja usein satoja tuhansia euroja. Ennen massatuotantoa tehdään vielä valmistettavuussuunnittelu (DFM, Design for Manufacturing), jossa varmistetaan, että tuote on ylipäätään valmistettavissa kustannustehokkaasti ja luotettavasti. Vasta tämän jälkeen siirrytään massatuotantoon, jonka aloittamisen jälkeen suunnitelman muuttaminen on käytännössä erittäin kallista. Juuri tämä peruuttamattomuus tekee ketterien menetelmien soveltamisesta laitteistoon vaikean mutta kiinnostavan ongelman. Ohjelmistossa virheen voi korjata päivityksellä, kun taas laitteistossa se voi tarkoittaa koko valmistuserän hylkäämistä.[19]

2.3 Menetelmien vertailu

Taulukko 2.1 asettaa perinteisen vesiputousmallin, puhtaan Scrumin ja laitteistolle optimoidun hybridimallin rinnakkain neljän kriteerin kautta [5], [6]. Vertailu osoittaa, ettei kumpikaan puhdas malli sovi laitteistolle sellaisenaan. Vesiputousmalli on liian jäykkä innovatiivisille tuotteille, kun taas puhdas Scrum törmää fyysisen prototypoinnin kustannuksiin.

Taulukko 2.1: Kehitysmenetelmien vertailu laitteistokontekstissa.[5], [6]

	Perinteinen	**Agile & Hybridimalli**	
Arviointikriteeri	**Perinteinen vesiputousmalli**	**Puhdas Scrum / Agile**	**Laitteistolle optimoitu hybridimalli**
Muutosten kustannus	Erittäin korkea loppuvaiheessa.	Teoriassa matala, fyysisessä tuotteessa kallis.	Hallittu, koska muutokset rajataan moduuleihin.
Prototyyppien tiheys	Harvoin, vasta projektin loppupuolella.	Jokaisen sprintin jälkeen, mikä on fyysisesti vaikeaa.	Virstanpylväspohjainen tai virtuaalinen simulaatioiden avulla.
Riskienhallinta	Riskit realisoituvat vasta integraatiovaiheessa.	Riskit havaitaan heti iteraation aikana.	Iteratiivinen testaus simulaatioilla ja pikamalleilla.
Soveltuvuus laitteistolle	Erinomainen vakiintuneille ja tutuille tuotteille.	Heikko ilman merkittäviä prosessimuutoksia.	Paras valinta innovatiivisille ja kompleksisille tuotteille.

Kirjallisuus päätyy käytännössä hybridimalliin, jossa muutokset rajataan moduuleihin ja tiheät fyysiset prototyypit korvataan simulaatioilla sekä virstanpylväisiin perustuvilla sykleillä [20]–[23]. Riskit hallitaan testaamalla digitaalisesti ennen kallista piivalmistusta, mikä pitää fyysiset kustannukset kurissa [24]–[26]. Hybridimalli ei kuitenkaan ole vielä vakiintunut, yritystason muutosvastarinta ja perinteisen insinöörikulttuurin perintö hidastavat sen omaksumista [27]–[29]. Seuraavat kaksi lukua tarkastelevat, miten aineiston artikkelit ovat yrittäneet kuroa tätä kuilua umpeen ja mihin ne ovat törmänneet.

3 Ketterän laitteistokehityksen hyödyt ja haasteet

Ketterän laitteistokehityksen lähtökohtana on ajatus siitä, että jos laitteistosuunnittelussa omaksutaan ohjelmistokehitykselle ominainen iteratiivisuus ja modulaarisuus, ala pystyy vastaamaan nopeammin modernien tekoäly- ja kvanttimarkkinoiden dynaamiseen kehitykseen. Vuosina 2020–2025 julkaistu 20 artikkelin tutkimusaineisto taulukkossa 3.1 osoittaa tämän tavoitteen olevan saavutettavissa usean toisistaan riippuvan tekijän kautta. Taulukossa 3.1 hyödyt on merkitty B1-B8 ja haasteet C1-C8. Lähdekirjallisuudesta nousevat hyödyt voidaan jakaa kahdeksaan keskeiseen teemaan. Niitä tarkastellaan tässä luvussa konkreettisten tutkimusesimerkkien valossa.

3.1 Ketterän laitteistokehityksen hyödyt (TK1)

Aineiston (taulukko 3.1) selvästi yleisin ja korostetuin teema on kehityksen joustavuus (*flexibility, B1*). Kun järjestelmäarkkitehtuuri pilkotaan pieniin, toisistaan riippumattomiin ja helposti vaihdettaviin osiin, projektin suuntaa ja teknisiä valintoja voidaan muuttaa joustavasti myös kesken kehitysprosessin. Esimerkiksi Huin ym. [34] kehittämä DIAG-suunnitteluvirta osoittaa, että yksittäisen komponentin muuttaminen tai poistaminen ei lamauta koko kehitysketjua, sillä arkkitehtuurin osat kommunikoivat keskenään joustavien palvelurajapintojen kautta kiinteiden integraatioiden sijaan. Tämä sama lähestymistapa toistuu useissa muissa aineiston

Taulukko 3.1: Tutkimuskysymysten aliteemat

	Hyödyt (Benefits)								**Haasteet (Challenges)**							
Artikkeli (Vuosi)	B1:Flexibility	B2:Efficiency	B3:Modularity	B4:Cost-Reduction	B5:Speed	B6:Reusability	B7:Feedback-Integration	B8:Patchability	C1:Complexity	C2:Quality-Risk	C3:Adaptation-Friction	C4:Ambiguity	C5:Tooling-Deficits	C6:Bug-Costs	C7:Scalability	C8:Vendor-Constraints
Bahr et al. (2020) [30]	x										x					
Berg et al. (2020) [27]	x								x	x	x				x	x
Bondar et al. (2025) [31]			x									x				
Carloni (2025) [32]		x	x		x				x							
He et al. (2023) [33]	x	x														
Hui et al. (2025) [34]				x	x				x							
Ma et al. (2022) [24]								x					x	x		
Minutoli et al. (2020) [35]			x	x									x			
Omidvarkarjan et al. (2020) [28]									x		x					
Pan et al. (2025) [22]	x	x	x		x											
Pearson et al. (2020) [26]	x						x									
Rautakoura & Hämäläinen (2023) [29]					x				x			x				
Romeral et al. (2023) [20]		x									x					
Santos et al. (2022) [36]			x		x	x										
Sorensen et al. (2020) [21]		x														
Thoma et al. (2024) [37]	x					x										
Wang et al. (2023) [23]	x						x									
Xu et al. (2024) [38]									x		x					
You et al. (2023) [39]	x							x								
Zuo et al. (2023) [25]								x					x			

tutkimuksissa, kuten salaussovellusten optimoinnissa [22] sekä Stanfordin yliopiston karkeajakoisissa rekonfiguroitavissa arkkitehtuureissa [30].

Tehokkuus (*efficiency, B2*) muodostaa joustavuudelle mitattavan vastineen. Kun kehityssyklien nopeus mahdollistaa useampien rinnakkaisten ratkaisuvaihtoehtojen testaamisen, optimaalisia arkkitehtuurivalintoja löydetään tehokkaammin. Pan ym. [22] raportoivat tutkimuksessaan saavuttaneensa jopa 34-kertaisen

parannuksen järjestelmän läpimenokyvyssä sekä 6,2-kertaisen parannuksen pinta-alatehokkuudessa verrattuna aiempiin joustaviin suoritusympäristöihin. Vakiintuneisiin ASIC-toteutuk-siin nähden tehokkuuden kasvu oli kolminkertainen. Vastaavia hyötyjä on dokumentoitu myös laitteiston ja ohjelmiston automatisoidussa yhteissuunnittelussa [21].

Koko ketterän ajattelutavan rakenteellisena perustana toimii modulaarisuus (*modularity, B3*). Cassel dos Santosin ym. [36] esittelemä ESP-alusta kuvaa tätä käytännössä, sillä siinä monimutkainen järjestelmäsiru (SoC) kootaan avoimen lähdekoodin laattamaisista komponenteista. Carlonin [32] mukaan kyseistä modulaarista alustaa hyödyntänyt pieni opiskelijaryhmä onnistui rakentamaan Linux-yhteensopivan 12 nm:n sirun vain muutamassa kuukaudessa. Tämä esimerkki osoittaa konkreettisesti, miten pitkälle viety modulaarisuus madaltaa monimutkaisen laitteistokehityksen aloituskynnystä.

Suunnittelukynnyksen madaltuminen heijastuu suoraan kustannusten laskuun (*cost-reduction, B4*), mikä on kriittinen tekijä toimialalla, jossa perinteiset kertakaikkiset suunnittelukustannukset (NRE) mitataan miljoonissa euroissa. Hui ym. [34] viittaavat tähän haasteeseen käsitteellä *4d-Problem*, jossa kehittäjän on kyettävä ratkaisemaan samanaikaisesti järjestelmän suorituskyvyn, monimutkaisuuden, kehitysaikataulun ja kustannusten välinen tasapaino. Romeralin ym. [20] systemaattinen katsaus vahvistaa teollisuuden kokemusten pohjalta, että ketterät menetelmät lyhentävät merkittävästi tuotteiden toimitusaikoja ja vähentävät siten kehityksen kokonaiskustannuksia.

Kehityksen nopeus (*speed, B5*) ilmenee lyhyempinä iteraatiosykleinä ja sitä kautta nopeampana markkinoille pääsynä. Minutolin ym. [35] kehittämä SODA-infrastruk-tuuri hyödyntää MLIR-pohjaista synteesivirtaa, jonka ansiosta koneoppimiskiihdyttimiä voidaan suunnitella kuukausien sijaan muutamassa päivässä. Bergin ym. [27] tekemä laajempi startup-selvitys täydentää tätä kuvaa osoittamalla,

että pohjoismaiset teknologiyritykset pitävät markkinoilletulon nopeutta kaikkein tärkeimpänä kilpailutekijänään.

Ohjelmistoteollisuudesta omaksutuista opeista suoraviivaisin on suunnitteluresurssien uudelleenkäytettävyys (*reusability, B6*). Thoman ym. [37] toteuttama XMSS- ja LMS-allekirjoituskiihdytin havainnollistaa tämän käytännössä, sillä kahden eri algoritmin yhteinen logiikka jaettiin siten, että ketterästi toteutettu monikäyttöinen versio oli vain noin 20 % suurempi kuin pelkkä yksittäinen kiinteä toteutus. Ekosysteemin laajuisesta uudelleenkäytettävyydestä hyötyy myös laajempi avoimen lähdekoodin laitteistoliike [32].

Palautteen dynaaminen integrointi (*feedback integration, B7*) tarkoittaa laitteistosuunnitelmien nopeaa mukauttamista uuden empiirisen tiedon valossa. Pearson ym. [26] kuvaavat tätä ilmiötä Tokamak Energy -fuusiotutkimusyhtiön tiekarttaprosessissa, jossa kokeellisen plasmafysiikan tuottama data ohjaa fyysisen laitteiston muutoksia kuukausien eikä vuosien sykleissä. Automatisoiduissa yhteissuunnittelukehitysympäristöissä tämä palautesilmukka on viety vielä pidemmälle [21], [22], [33], jolloin simulaattorin ja kääntäjän antama kvantitatiivinen palaute ohjaa arkkitehtuurivalintoja suoraan jokaisessa iteraatiossa.

Paikattavuus (*patchability, B8*) on aineistossa esiintyvistä teemoista spesifein, ja se kytkeytyy suoraan uudelleenkonfiguroitavan piiteknologian hyödyntämiseen. Ma ym. [24] sekä Zuon ym. [25] kehittämät edistykselliset virheenkorjaus- ja tallennustyökalut mahdollistavat sen, että viallinen toiminnallisuus voidaan korjata ja paikata suoraan valmiilla FPGA-alustalla ohjelmistopäivityksiä muistuttavalla tavalla.

3.2 Ketterän laitteistokehityksen haasteet (TK2)

Vaikka ketterän laitteistokehityksen tarjoamat mahdollisuudet ovat merkittäviä, fyysisen maailman ja materiaalien asettamat rajoitteet erottavat sen pysyvästi ohjelmistokehityksestä. Bittijonojen sijaan laitteistokehityksen lopputuotteena on ai-

na konkreettinen kappale, minkä vuoksi ohjelmistopuolelta siirretyt menetelmät törmäävät uudenlaisiin esteisiin. Tutkimusaineistosta nousee esiin kahdeksan toistuvaa haastetta, jotka liittyvät tuotteiden fyysiseen luonteeseen, suunnittelutyökalujen kehittymättömyyteen sekä itse kehitysliikkeen nuoreen ikään.

Aineiston (taulukko 3.1) selvästi universaalein haaste on järjestelmien suuri monimutkaisuus (*complexity, C1*). Hennessy ja Patterson [4] kuvaavat nykyistä tietokonearkkitehtuurin aikakautta uudeksi kultakaudeksi juuri siksi, että nykyiset tekoälykiihdyttimet, kvanttisuorittimet ja tietoliikennepiirit vaativat huomattavasti monimutkaisempia rakenteita kuin perinteiset yleiskäyttöiset prosessorit. Tämä monimutkaisuus heijastuu suoraan kehitysprosesseihin, sillä suunnitteluavaruus kasvaa valtavaksi [22], eri abstraktiotasojen integrointi on vaativaa [33] ja yhtenäisten työkaluketjujen puute lisää suunnittelijoiden kognitiivista kuormaa [30].

Nopeaan iterointiin liittyy aina kasvanut laaturiski (*quality-risk, C2*). Siinä missä ohjelmistovirhe voidaan yleensä korjata nopealla päivityksellä, fyysisessä laitteistossa virheen korjaaminen saattaa vaatia kokonaan uuden, miljoonia euroja maksavan valmistuserän tilaamista. Tämän vuoksi Wang ym. [23] ovat kehittäneet automaattisia ekvivalenssitarkistusmalleja, joiden tavoitteena on hyödyntää peräkkäisten suunnitteluiteraatioiden rakenteellisia yhtäläisyyksiä ja siten rajata varmistustyön vaatimaa aikaa. Myös Ma ym. [24] korostavat FPGA-tutkimuksessaan, että nopea iteroitavuus ilman riittävän kypsiä virheenkorjaustyökaluja johtaa herkästi virheiden kasautumiseen.

Uusien menetelmien käyttöönottoa hidastaa myös kulttuurinen ja organisatorinen omaksumiskitka (*adaptation friction, C3*). Omidvarkarjan ym. [28] kehittivät aihetta käsittelevän Bender-opetuspelin huomattuaan, että perinteinen insinöörikoulutus valmentaa opiskelijoita edelleen toimimaan lineaarisissa vesiputousmalleissa, mutta ei tarjoa valmiuksia epävarmassa ja jatkuvasti muuttuvassa toimintaympäristössä navigoimiseen. Bergin ym. [27] startup-tutkimus vahvistaa saman ilmiön

yritystasolla, sillä vakiintuneiden prosessien muuttaminen kesken käynnissä olevan hankkeen on organisaatiolle raskasta ja epäonnistuu ilman vahvaa ja sitoutunutta johtoa.

Käsitteellinen epäselvyys (*ambiguity, C4*) nousee aineistosta esiin yllättävän vahvana haasteena. Rautakoura ja Hämäläinen [29] toteavat systemaattisessa kartoituksessaan suoraan, ettei ketterälle laitteistokehitykselle ole olemassa yhtä vakiintunutta tai yleisesti hyväksyttyä määritelmää. Eri tutkimusryhmät ja yritykset tarkoittavat termillä hyvinkin erilaisia asioita, kuten generatiivisia laitteistokuvauskieliä, organisaatiotason projektinhallintaa tai uudelleenkonfiguroitavan piirin hyödyntämistä. Bondarin ym. [31] koulutusperspektiiviin sijoittuva tutkimus vahvistaa tämän saman ongelman, sillä käsitteen vakiintumattomuus vaikeuttaa osaamisen systemaattista opettamista ja arviointia.

Seuraavana keskeisenä esteenä ovat konkreettiset työkalupuutteet (*tooling deficits, C5*). Ohjelmistokehityksessä käytettävät ketterät ekosysteemit, kuten versionhallinta, jatkuva integraatio (CI/CD) ja kehitysympäristöt, ovat erittäin kypsiä, mutta laitteistopuolella vastaavia työkaluja ei ole saatavilla suoraan. Tämän vuoksi Sorensen ym. [21] päätyivät rakentamaan kokonaan oman simulaattorikehyksensä laitteiston ja ohjelmiston yhteissuunnittelua varten, kun taas Bahr ym. [30] ottivat käyttöön uuden DSL-pohjaisen suunnittelupinon. Vastaavasti Huin ym. [34] valinta käyttää SpinalHDL-kieltä johtui siitä, että perinteinen SystemVerilog on liian verbiili ja korkean tason synteesityökalut tuottavat usein tarpeetonta redundanssia.

Laitteistovirheiden tuottama korkea vikakustannus (*bug cost, C6*) kytkeytyy tiiviisti laaturiskiin, mutta muodostaa oman itsenäisen haasteensa fyysisten vaikutustensa vuoksi. Tämä taloudellinen ja ajallinen riski toimii ensisijaisena motivaationa useille aineistossa esitellyille validointityökaluille [23], [25]. Vaikka FPGA-pohjainen prototypointi pyrkii löytämään ja korjaamaan virheet ennen lopullista piivalmistusta [24], menetelmä ei ole aukoton, sillä ASIC-toteutuksen kriittiset fyysiset ominai-

suudet, kuten kellosignaalin jakautuminen, tehonkulutus ja lämpövaikutukset, eivät välity FPGA-mallinnuksen kautta sellaisenaan.

Skaalautuvuus (*scalability, C7*) haastaa ketterät prosessit kahdesta eri suunnasta. Ensinnäkin itse arkkitehtuurin on kyettävä skaalautumaan suuriin kokoluokkiin, mistä Cassel dos Santos ym. [36] antavat esimerkin moniprosessoristen sirujärjestelmien rakentamisessa. Toiseksi kehitysmenetelmien on kestettävä sovellusvaatimusten kasvu, minkä Pan ym. [22] sekä Thoma ym. [37] osoittavat post-quantum-salauksen yhteydessä, missä bittileveydet ja laskennan upotusasteet kasvavat jatkuvasti uusien hyökkäysmenetelmien kehittyessä.

Viimeisenä strategisena haasteena ovat toimittajariippuvuudet (*vendor constraints, C8*). Bergin ym. [27] havaintojen mukaan pienet laitteistostartupit ovat poikkeuksetta sidottuja ulkopuolisten valmistajien tuotantoaikatauluihin ja minimitilausmääriin (MOQ). Tämä riippuvuus näkyy myös laajemmalla makrotasolla CHIPS Act -keskusteluissa [31], [32], sillä eurooppalainen sirusäädös (Chips Act) pyrkii vastaamaan juuri siihen haasteeseen, että koko maanosa on tällä hetkellä riippuvainen muutamasta Itä-Aasiassa sijaitsevasta suuresta piitehtaasta.

4 Analyysi ja pohdinta

Hyötyteemoista *flexibility*, *modularity*, *speed* ja *efficiency* esiintyivät selvästi useimmissa artikkeleissa. Käytännössä jokainen tekninen paperi nostaa nämä esille, ja ne muodostavat ketterän laitteistokehityksen ydinsanaston. *Reusability* ja *cost-reduction* olivat seuraavaksi yleisimpiä, ja niitä korostivat erityisesti avoimen lähdekoodin laitteistoa käsittelevät työt [32], [36] sekä teollisuusartikkelit [20]. *Feedback integration* esiintyi vahvimmin yhteissuunnittelukehyksissä [21], [22], [33] ja kasvuvaiheen startupeissa [26], [27]. *Patchability* jäi kapeimmaksi hyötyteemaksi ja keskittyi FPGA-tutkimukseen [24], [25].

Haasteteemoista *complexity* oli yleisin, sillä lähes kaikki artikkelit mainitsivat sen jossakin muodossa. *Tooling deficits* oli toiseksi yleisin, ja se motivoi suoraan suurta osaa aineiston teknisistä papereista. Sorensen, Bahr, Wang, Zuo, Hui ja Pan tutkimusryhmineen rakentavat kaikki uusia työkaluja vastatakseen havaitsemaansa puutteeseen. *Quality-risk* ja *bug cost* esiintyivät erityisesti FPGA-papereissa, joiden ydinongelma on nopean iteroinnin ja luotettavuuden tasapainottaminen. *Adaptation friction* näkyi kolmella tasolla, joita ovat koulutusvaje [28], [31], organisaatiomuutos [27] sekä uudelleensuunnittelukustannukset [22].

Erityisen merkittävä havainto oli *ambiguity*-teeman vahvuus. Rautakoura ja Hämäläinen [29] osoittavat suoraan, ettei ketterälle laitteistokehitykselle ole yhteisesti hyväksyttyä määritelmää. Tämä on huomattava löytö, koska se tarkoittaa, että vertaileva tutkimus on hankalaa, sillä kaksi paperia voi käyttää samaa termiä tar-

koittaen eri asioita. Aineisto tukee havaintoa empiirisesti, sillä artikkelien välillä on selvää variaatiota siinä, mihin asioihin ”ketterä” ulottuu [22], [28], [34]. *Scalability* oli vahva haasteteema sekä arkkitehtuuritasolla että algoritmitasolla, ja *vendor constraints* oli kapein mutta strategisesti tärkeä, etenkin Euroopan teknologisen omavaraisuuden [12] näkökulmasta.

Aineistossa erottui myös selvä jännite eri artikkelityyppien välillä. Tekniset artikkelit eli kompilaattorit, työkalut ja kiihdyttimet painottivat hyötyjä ja työkalupuutteita, kun taas katsausartikkelit [20], [29] ja koulutuspaperit [28], [31] keskittyivät enemmän organisaatio- ja menetelmähaasteisiin. Tämä jakautuminen tukee tulkintaa, jonka mukaan ketterä laitteistokehitys on teknisellä tasolla pidemmälle kehittynyt kuin organisaatiotasolla. Aineistoon mahtui myös kaksi artikkelia, joiden osuus aliteemoihin jäi heikoksi. Ne pidettiin mukana, koska katsauksen rajauskriteeri oli mekaaninen, eli 20 uusinta hakulausekkeen ehdot täyttävää artikkelia. Jälkikäteinen subjektiivinen karsinta sen mukaan, mikä artikkeli sopii teemoihin ”tarpeeksi hyvin”, olisi heikentänyt katsauksen toistettavuutta. You ym. [39] käsittelee konttiverkkojen tietoturvaa eikä keskity ketterään laitteistokehitykseen menetelmänä, vaikka sana ”agile” esiintyy tekstissä. Xu ym. [38] puolestaan käyttää sanaa ”agile” pääasiassa miehittämättömän lentolaitteen fyysisestä ketteryydestä eikä kehitysmenetelmästä. Näiden kahden artikkelin mukanaolo havainnollistaa konkreettisesti saman ongelman, joka nousee esiin myös *ambiguity*-teeman yhteydessä, sillä yksinkertainen avainsanapohjainen haku ei pysty erottelemaan sanan ”agile” eri käyttötarkoituksia. Näitä havaintoja syvennetään pohdintaluvussa.

Kirjallisuuskatsauksen tulokset nostavat esiin useita kriittisiä näkökulmia, jotka vaativat syvempää akateemista tarkastelua. Ketterän, kokeilevan hengen ja fyysisen tuotannon reunaehtojen yhteensovittaminen synnyttää monimutkaisen kokonaisuuden, jossa saavutettavat hyödyt ja kohdattavat haasteet kietoutuvat tiiviisti toisiinsa. Tutkielman keskeisimpänä ajatulsena voidaankin pitää sitä, että ketterä laitteis-

tokehitys ei ole ohjelmistomaailman työtapojen suoraa tai mekaanista kopioimista fyysiseen ympäristöön. Lyhyet sprintit tai jatkuvat tuotantojulkaisut eivät sellaisenaan sovellu perinteiseen sirusuunnitteluun, jossa yksittäiset valmistuserät vaativat suuria taloudellisia panostuksia ja pitkiä toimitusaikoja. Aineiston valossa ketteryys merkitsee ennen kaikkea joustavan ajattelutavan soveltamista, johon kuuluvat empiirinen oppiminen, jatkuva validointi, esteiden purkaminen tiimien väliltä sekä palautesilmukoiden rakentaminen. Esimerkiksi edistyksellinen FPGA-prototypointi [24], [25] ja simulaattoripohjaiset mallinnukset [21], [22] ilmentävät juuri tällaista sovitusta tuomalla ketteryyden tasolle, jossa muutoksen kustannukset on saatu painettua alas.

Prosessien joustavuus kulkee kuitenkin lähes poikkeuksetta käsi kädessä kasvavan monimutkaisuuden kanssa, sillä kehitysnopeuden maksimointi voi heikentää lopputuotteen laatua, ja pitkälle viety modulaarisuus vaatii tuekseen kehittyneitä työkaluja, joita ei markkinoilla ole vielä yleisesti saatavilla. Käsitteellisen epäselvyyden vahvuus haastaa samalla oletuksen siitä, että ketterä laitteistokehitys olisi jo vakiintunut tai yhtenäinen tieteenala. Kuten Rautakoura ja Hämäläinen [29] kartoitustutkimuksessaan toteavat, alalle ei ole muodostunut yhtä yhteisesti hyväksyttyä määritelmää. Eri tutkimusryhmät saattavat ratkaista hyvinkin samankaltaisia arkkitehtuuriongelmia, mutta heidän käyttämänsä termistö ja viitekehykset eroavat toisistaan merkittävästi. Tämä hajanaisuus näkyy selvästi esimerkiksi Panin ym. [22] ja Huin ym. [34] töiden välillä, mikä vaikeuttaa tutkimustulosten systemaattista vertailua ja alan kumulatiivista kehitystä.

Kirjallisuuden ajallinen jakauma osoittaa aiheen akateemisen kiinnostuksen olevan selvässä kasvussa, mikä heijastuu suoraan vuoden 2025 julkaisujen suuressa määrässä [22], [31], [32], [34]. Tämä kehitys tukee johdannossa esitettyä argumenttia siitä, että globaalit geopoliittiset paineet ja pyrkimykset teknologiseen omavaraisuuteen ovat kiihdyttäneet ketterien menetelmien tutkimusta konkreettisella tasolla [9],

[12]. Tästä huolimatta katsauksen metodologiset rajoitukset on syytä tuoda esiin. Käytetty hakulauseke tuotti huomattavan osan tuloksista ACM Digital Library -tietokannasta, kun taas laitteistoalalla keskeisen IEEE Xplore -tietokannan osuus jäi olemattomaksi. Tämä vinouma saattaa johtua siitä, että IEEE käyttää indeksoinnissaan erilaista asiasanastoa tai käytetty hakustrategia ei ollut täysin optimaalinen. Lisäksi kahden poikkeavan artikkelin [38], [39] valikoituminen mukaan aineistoon osoittaa käytännössä, miten mekaaniset hakukriteerit voivat poimia mukaan tutkimuksia, joissa termin merkitys viittaa johonkin muuhun kuin itse kehitysmenetelmään. Tulevassa tutkimuksessa hakulauseketta olisikin syytä tarkentaa kytkemällä ketteryyden käsite suoraan kehitysvirtoihin ja metodologioihin.

Akateemisesta näkökulmasta merkittävä puute on myös yhtenäisten, kvantitatiivisten mittareiden vähäisyys ketteryyden arvioinnissa. Vaikka Pan ym. [22] esittävät työssään tarkkoja lukuja kompilaatioajoista ja suunnittelusyklien pituuksista, valtaosa aineistosta luottaa puhtaasti kvalitatiiviseen kuvaukseen. Ilman standardoituja mittareita on erittäin vaikea arvioida tai vertailla eri ketterien menetelmien todellista tehokkuutta käytännössä. Suomalaisen tutkimuksen osalta aineisto jää odotetusti kapeaksi, sillä ainoa vertaisarvioitu kotimainen kirjoittaja oli Rautakoura [29]. Teollisuuden edelläkävijöiden käytännöt nojaavat toistaiseksi vertaisarvioimattomiin raportteihin ja lehdistötiedotteisiin [14], [17]. Tämä tarjoaa selvän paikan jatkotutkimukselle, sillä kotimaisen kvantti- ja tietoliikenneteollisuuden ketterien prosessien dokumentointi olisi alueellisesti erittäin arvokasta. Lopuksi on todettava, että aineisto painottuu voimakkaasti puhtaaseen sirusuunnitteluun sulautettujen järjestelmien jäädessä vähemmälle huomiolle. Tämä saattaa merkitä sitä, että sulautettujen järjestelmien kokonaisuus muodostaa oman, erillisen tutkimuskenttänsä, joka ei suoraan hyödynnä sirusuunnittelusta tuttua ketterää käsitteistöä, vaikka ne ovat Suomessa alan keskiössä [6].

5 Yhteenveto

Laitteistokehitys on 2020-luvulla siirtymässä pois perinteisistä vesiputousmalleista. Mooren lain hidastuminen on siirtänyt painopisteen erikoistuneisiin arkkitehtuureihin, kuten tekoälykiihdyttimiin ja kvanttisuorittimiin. Geopoliittiset paineet ja toimitusketjujen haavoittuvuus ovat synnyttäneet valtiollisia investointiohjelmia. Kilpailuetu riippuu yhä enemmän iterointinopeudesta, minkä vuoksi suuret teknologiayritykset ovat siirtyneet ketterään laitteiston ja ohjelmiston yhteissuunnitteluun. Euroopassa ja Suomessa kilpailukyky nojaa ketteryyteen ja avoimen lähdekoodin laitteistoihin, sillä Itä-Aasian laajuinen tehdasekosysteemi puuttuu. Vallitseva näkemys on, että laitteistoteollisuuden tulee omaksua ohjelmistokehityksen kaltaista iteratiivisuutta. Tähän tähtäävät generatiiviset laitteistokuvauskielet, korkean tason synteesi ja uudelleenkonfiguroituva pii.

(TK1) Tulosten valossa ketterän toimintatavan keskeisimmiksi hyödyiksi hahmottuvat kehitysprosessien parantunut joustavuus, arkkitehtuurien korkea modulaarisuus, nopeampi markkinoille pääsy sekä resurssien tehokas kokonaiskäyttö. Nämä tekijät muodostavat ketterän laitteistokehityksen toiminnallisen ytimen, kun taas piitason paikattavuus jää selvästi rajatummaksi, pääasiassa FPGA-ympäristöihin kytkeytyväksi eduksi.

(TK2) Menetelmien soveltamista varjostavat järjestelmien poikkeuksellisen suuri monimutkaisuus sekä standardoitujen kehitys- ja validointityökalujen merkittävät puutteet. Akateemisen ja teollisen kehityksen kannalta vakavin este on kuitenkin

ilmiötä vaivaava käsitteellinen epäselvyys, sillä alalta puuttuu yhtenäinen ja vakiintunut määritelmä, mikä vaikeuttaa menetelmien vertailua ja hidastaa alan yhteisen tietopohjan kehittymistä.

Tutkielman yleisenä päätelmänä voidaan todeta, että ketterä laitteistokehitys edustaa aitoa ja tarpeellista edistysaskelta, mutta on tutkimuskenttänä vielä vakiintumaton. Kyseessä ei ole ohjelmistokehityksen käytäntöjen suoraviivainen siirto, vaan ketterän ajattelutavan, kuten jatkuvan oppimisen ja tiiviin palautesilmukan, sovittaminen fyysisen maailman ja materiaalikustannusten asettamiin rajoitteisiin. Globaalit geopoliittiset jännitteet sekä Yhdysvaltojen ja Euroopan unionin mittavat sirusäädökset kiihdyttävät parhaillaan tätä siirtymää korostamalla teknologisen omavaraisuuden ja nopean innovoinnin merkitystä. Tässä murroksessa Pohjois-Euroopalla ja erityisesti Suomella on vahvan suunnitteluhistoriansa ansiosta luonteva paikka toimia koealustana, etenkin kvanttilaskennan ja tulevaisuuden tietoliikenneinfrastruktuurien alueilla.

Jatkotutkimuksen kannalta kiireellisin tarve on yhtenäisen käsitteistön ja kvantitatiivisten mittareiden kehittäminen. Tämä mahdollistaisi ketteryyden asteen luotettavan arvioinnin ja vertailun eri organisaatioiden välillä. Toiseksi työkaluekosysteemin kypsymistä olisi perusteltua seurata empiirisesti, erityisesti laitteistolähtöisten CI/CD-putkien ja nopeampien EDA-työkalujen osalta. Kolmanneksi prototyypistä tuotantoon siirtymisen kuilu kaipaa lisätutkimusta siitä, millaiset hybridiprosessit skaalautuvat parhaiten massatuotantoon. Neljänneksi aiheen tarkastelu toimialakohtaisesti tuottaisi konkreettisempaa tietoa kuin nykyinen, teknologialtaan hajanainen kirjallisuus. Luontevia kohteita olisivat esimerkiksi kvanttilaskennan ohjauselektroniikka ja 6G-reunalaskenta.

Lähdeluettelo

[1] I. Sommerville, *Software Engineering*, 10th (Global Edition). Pearson, 2015, ISBN: 1-292-09613-6.

[2] S. L. Pfleeger ja J. M. Atlee, *Software Engineering: Theory and Practice*, 3rd. Prentice Hall, 2005, ISBN: 0-13-146913-4.

[3] G. E. Moore, "Cramming more components onto integrated circuits", *Electronics*, vol. 38, nro 8, s. 114–117, 1965.

[4] J. L. Hennessy ja D. A. Patterson, "A New Golden Age for Computer Architecture", *Communications of the ACM*, vol. 62, nro 2, s. 48–60, 2019.

[5] C. Larman, *Agile and Iterative Development: A Manager's Guide*. Addison-Wesley Professional, 2003, ISBN: 0-13-111155-8.

[6] T. Lehtonen et al., *Sulautettujen järjestelmien ketterä käsikirja*, Finnish. Turku, Finland: University of Turku, Technology Research Center, 2014, Also available online at `http://embedded.utu.fi/kasikirja`, ISBN: 978-951-29-5838-2. url: `https://www.utupub.fi/handle/11111/584`.

[7] W. C. Shih, "Global supply chains in a post-pandemic world", *Harvard Business Review*, vol. 98, nro 5, s. 82–89, 2020.

[8] Y. N. Lee, *The global chip shortage is starting to have major real-world consequences*, CNBC, Verkkojulkaisu, 2021.

[9] D. Clark ja A. Swanson, *U.S. CHIPS and Science Act Passes, Aiming to Counter Asian Semiconductor Dominance*, 2022.

[10] N. P. Jouppi et al., "In-datacenter performance analysis of a tensor processing unit", teoksessa *Proceedings of the 44th Annual International Symposium on Computer Architecture (ISCA)*, IEEE/ACM, 2017, s. 1–12.

[11] C. Metz, *Hyperscalers and the Silicon Race: How Google, Amazon, and Microsoft are Designing Custom AI Chips*, 2024.

[12] European Commission, "The European Chips Act: Securing Europe's Technological Sovereignty", Official EU Policy Documentation, tekninen raportti, 2023.

[13] J.-P. Kleinhans ja N. Baisakova, "Understanding the global semiconductor value chain", Stiftung Neue Verantwortung (SNV), tekninen raportti, 2021.

[14] IQM Quantum Computers, *IQM Quantum Computers opens a new quantum processor fabrication facility in Finland*, Lehdistötiedote, Verkkojulkaisu, 2023.

[15] VTT Technical Research Centre of Finland, *Finland's next quantum computers: 20-qubit and 54-qubit systems under development*, Lehdistötiedote, Verkkojulkaisu, 2024.

[16] Yle News, *Finland's Quantum Leap: IQM and VTT Scale Up Superconducting Quantum Hardware*, 2025.

[17] Nokia Corporation Reports, "Agile Methodologies in 6G Edge Node Development", Telecom Infrastructure Tech Journal / Northern Europe, tekninen raportti, 2025.

[18] Defense Advanced Research Projects Agency (DARPA), "Electronics Resurgence Initiative (ERI)", U.S. Department of Defense, tekninen raportti, 2018, Ohjelmakuvaus ja tavoitteet (erityisesti IDEA- ja POSH-hankkeet).

[19] P. Weichbroth, "A Case Study on Implementing Agile Techniques and Practices: Rationale, Benefits, Barriers and Business Implications for Hardware Development", *Applied Sciences*, vol. 12, nro 17, 2022, ISSN: 2076-3417. DOI: 10.3390/app12178457. url: https://www.mdpi.com/2076-3417/12/17/8457.

[20] P. A. d. A. Romeral, E. Zancul ja D. Nakano, "Product Development Process for complex hardware-based solutions: current trends", *Procedia CIRP*, vol. 119, s. 885–890, 2023.

[21] T. Sorensen, A. Manocha, E. Tureci, M. Orenes-Vera, J. L. Aragón ja M. Martonosi, "A simulator and compiler framework for agile hardware-software co-design evaluation and exploration", teoksessa *Proceedings of the 39th International Conference on Computer-Aided Design (ICCAD '20)*, 2020, Article 97.

[22] T. Pan et al., "Finesse: An Agile Design Framework for Pairing-based Cryptography via Software/Hardware Co-Design", teoksessa *Proceedings of the 52nd Annual International Symposium on Computer Architecture (ISCA '25)*, 2025, s. 65–77.

[23] Y. Wang, F. Xie, Z. Yang, P. Cocchini ja J. Yang, "An Equivalence Checking Framework for Agile Hardware Design", teoksessa *Proceedings of the 28th Asia and South Pacific Design Automation Conference (ASPDAC '23)*, 2023, s. 26–32.

[24] J. Ma, G. Zuo, K. Loughlin, H. Zhang, A. Quinn ja B. Kasikci, "Debugging in the brave new world of reconfigurable hardware", teoksessa *Proceedings of the 27th ACM International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS '22)*, 2022, s. 946–962.

[25] G. Zuo, J. Ma, A. Quinn ja B. Kasikci, ”Vidi: Record Replay for Reconfigurable Hardware”, teoksessa *Proceedings of the 28th ACM International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS 2023)*, vol. 3, 2023, s. 806–820.

[26] R. Pearson, A. Costley, R. Phaal ja W. Nuttall, ”Technology Roadmapping for mission-led agile hardware development: a case study of a commercial fusion energy start-up”, *Technological Forecasting and Social Change*, vol. 158, s. 120 064, 2020.

[27] V. Berg, J. Birkeland, A. Nguyen-Duc, I. O. Pappas ja L. Jaccheri, ”Achieving agility and quality in product development - an empirical study of hardware startups”, *Journal of Systems and Software*, vol. 167, s. 110 599, 2020.

[28] D. Omidvarkarjan, J. Conrad, C. Herbst, C. Klahn ja M. Meboldt, ”Bender – An Educational Game for Teaching Agile Hardware Development”, *Procedia Manufacturing*, vol. 45, s. 313–318, 2020.

[29] A. Rautakoura ja T. Hämäläinen, ”Does SoC Hardware Development Become Agile by Saying So: A Literature Review and Mapping Study”, *ACM Transactions on Economics and Computation*, vol. 11, nro 1–2, Article 44, 2023.

[30] R. Bahr et al., ”Creating an agile hardware design flow”, teoksessa *Proceedings of the 57th ACM/EDAC/IEEE Design Automation Conference (DAC ’20)*, 2020, Article 142.

[31] K. Bondar et al., ”Microcredentials for Open Hardware and HPC Workforce Development: The Openchip Approach with RISC-V Ecosystem”, teoksessa *Proceedings of the SC ’25 Workshops of the International Conference for High Performance Computing, Networking, Storage and Analysis*, 2025, s. 416–423.

[32] L. Carloni, "Open-Source Hardware and Agile System-on-Chip Design", teoksessa *Proceedings of the 22nd ACM International Conference on Computing Frontiers: Workshops and Special Sessions (CF '25 Companion)*, 2025, s. 17.

[33] Z. He, A. Shen, Q. Li, Q. Cheng ja H. Yu, "Agile Hardware and Software Co-Design for RISC-V-Based Multi-Precision Deep Learning Microprocessor", teoksessa *Proceedings of the 28th Asia and South Pacific Design Automation Conference (ASPDAC '23)*, 2023, s. 490–495.

[34] H. Hui, J. Gu, X. Hu, S. Wei ja S. Yin, "DIAG: A Refined Four-layer Agile Hardware Developing Flow for Generating Flexible Reconfigurable Architectures", teoksessa *Proceedings of the 30th Asia and South Pacific Design Automation Conference (ASPDAC '25)*, 2025, s. 548–553.

[35] M. Minutoli et al., "SODA: a new synthesis infrastructure for agile hardware design of machine learning accelerators", teoksessa *Proceedings of the 39th International Conference on Computer-Aided Design (ICCAD '20)*, 2020, Article 98.

[36] M. C. d. Santos et al., "A Scalable Methodology for Agile Chip Development with Open-Source Hardware Components", teoksessa *Proceedings of the 41st IEEE/ACM International Conference on Computer-Aided Design (ICCAD '22)*, 2022, Article 20.

[37] J. P. Thoma, D. Hartlief ja T. Güneysu, "Agile Acceleration of Stateful Hash-based Signatures in Hardware", *ACM Transactions on Embedded Computing Systems*, vol. 23, nro 2, Article 29, 2024.

[38] Y. Xu, J. Yu, S. Zhang, Y. Xiang, H. Jia ja Y. Wang, "Invited: Automatic Hardware/Software Design for High-Speed Autonomous Unmanned Aerial Vehicles Guided by a Flight Model", teoksessa *Proceedings of the 61st ACM/IEEE Design Automation Conference (DAC '24)*, 2024, Article 368.

[39] M. You, J. Nam, M. Seo ja S. Shin, ”HELIOS: Hardware-assisted High-performance Security Extension for Cloud Networking”, teoksessa *Proceedings of the 2023 ACM Symposium on Cloud Computing (SoCC ’23)*, 2023, s. 486–501.